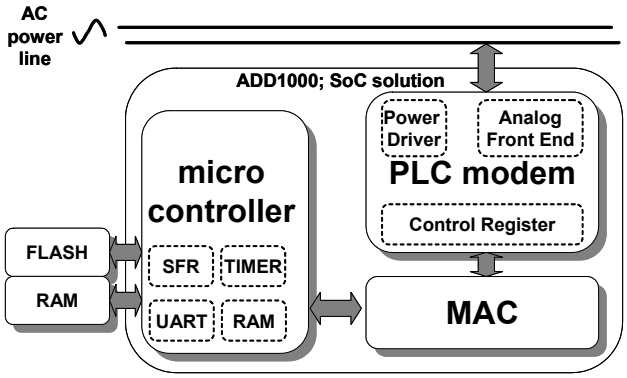
# ADD1000A

## POWER-LINE COMMUNICATIONS SYSTEM ON CHIP




The ADD1000A is a Power Line Communications System on Chip. It includes an enhanced 8051 microcontroller, a hardwired Medium Access Controller (MAC) and a Modem circuit for the EHS/KNX Power Line medium specifications. The system is powered using a single DC power supply.

The modem included in ADD1000A is implemented using digital circuitry, depending on the required characteristics an external passive or active interface can be used. The internal digital filtering and A/D conversion gives high sensibility and noise inmunity.

The MAC circuit is designed to reduce the computational load of the microcontroller. The FEC and FCS codes of the emitted messages can be calculated using the specific hardware features of the MAC. The same hardware can correct incoming data bytes using the incoming FECs, and it can compute the FCS of the incoming message to check frame correctness. The MAC also checks the received bit stream for correct (user defined) message headers, then the microcontroller is not interrupted until a correct header is received. To keep compatibility with full software communication libraries the MAC can be configured as a byte mode, or bit mode, modem interface.

The microcontroller includes some specific peripherals: a 4 input / 4 output dimmer for phase angle power regulation and a flash program loader that allows to store the microcontroller program in a standard SPI serial flash memory and to execute it from a SRAM. The flash loader includes features as in-system reprogrammability and multiple program storage. The microcontroller can be programmed to switch from one program to any other in the flash and to reboot to start execution of the new program.

ADD provides technical information, software and circuits, for the new designs that include the ADD1000A.

ADD can provide communication software libraries to the customers. The communication libraries, and a support line, are included in the ADD PLC design kits. The purchase of an ADD design kit allows to the client to the free use of the communication libraries and technical information in any product that includes the ADD1000A.

**CONTENTS:**

# CHAPTER 1. ADD1000AQF128

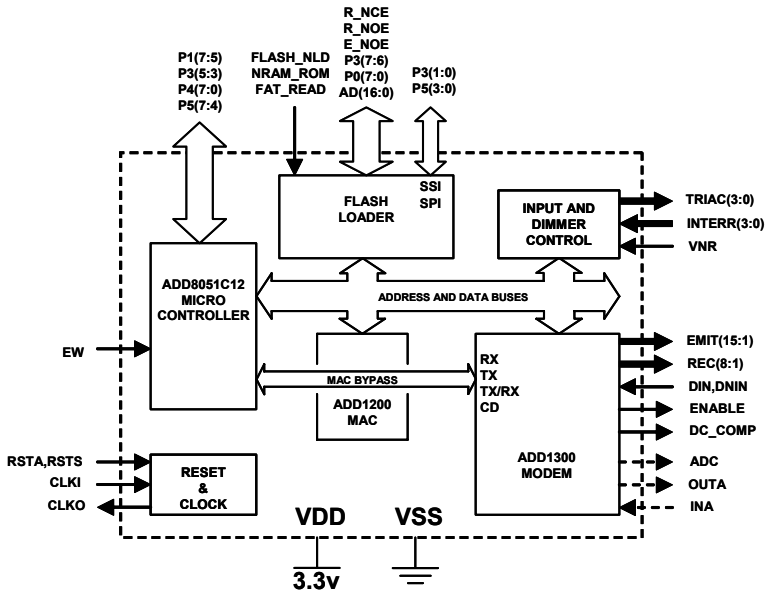



*Figure 1.1: ADD1000AQF128 block diagram*

## 1.1. OVERVIEW

The ADD1000AQF128 is a Power Line Communications System on Chip. It includes an enhanced 8051 microcontroller (IP core ADD8051C12A), a Medium Access Controller (MAC) (IP core ADD1200) and a Modem circuit for the EHS/KNX Power Line medium specifications (IP core ADD1300). The microcontroller includes some specific peripherals as a dimmer peripheral with 4 inputs and 4 outputs for power regulation, or a flash program loader that allows user to store the microcontroller program in a standard SPI serial flash memory and to execute it from a SRAM.

The ADD1000 is designed to be used by OEM and provides a low cost and small size solution for power line communications. Only a single 3.3v supply is needed.

ADD1000AQF128 is packed in a thin quad flat pack 128 leads, 0.4 mm pitch, package; this allows a reduced PCB area (16x16 mm) while maintaining a good thermal characteristics. It can be used in continuous emission against power line short circuit in an 85ºC ambient. The PLC signal amplitude is software programmable and is limited to VDD/2. The PLC signal intensity is limited by an external resistor array, a reduction in the resistor array increases the signal intensity injected to the power line with an increase of I.C. power dissipation and supply power consumption, see Table 1.1.

| | min | typ | max |
|---|---|---|---|
| VDD | 3.0v | 3.3v | 3.6v |
| $I_{VDD}$ | 40mA* | | 300mA** |
| Power disipation º | 120mW | | 475mW |
| Power consumptionºº | 120mW | | 1000mW |

* supply intensity in reception mode

** supply intensity in emission mode with a short circuit power-line load

º power dissipated in the integrated circuit in reception mode (min) and in emission against short circuit (max) (3.3v supply)

ºº VDD supply power consumption in reception mode (min) and in emission against short circuit (max) (3.3v supply)

*Table 1.1: electrical characteristics*

## 1.2. SYSTEM ARCHITECTURE

The IP blocks inside the ADD1000A are addressed by the microcontroller as external data ram entries. They are connected to the microcontroller using the address and data busses (P0 and P2) and the read/write control lines (ALE, PSEN, P3.6 and P3.7). Only the flash loader has a hardwired connection, because it acts as an address and data bus switcher.

The peripheral blocks are mapped in the upper entries of the data address space, address from FE00 to FFFF are reserved to this purpose. These two pages must not be used as data space for general purposes, the software must access these entries only to read or write peripheral registers. The entries in pages FE and FF that are not mapped to a peripheral register must not be used at all. The address and data buses and the read/write control signals are connected to circuit pins, these pins swings even when the microcontroller software access to the mapped peripherals.

The following tables (Table 1.2, Table 1.3, Table 1.4) show name and addresses of the peripheral registers.

| REG. NAME | ADDRESS |
|---|---|
| CONTROL | FE00 |
| FLAGS1 | FE01 |
| FLAGS2 | FE02 |
| DATA | FE03 |
| BAUDRATE_LOW | FE04 |
| BAUDRATE_HIGH | FE05 |

*Table 1.2: MAC registers*

| REG. NAME | ADDRESS |
|---|---|
| CONTROL | FE10 |
| DATA_H | FE11 |
| DATA_M | FE12 |
| DATA_L | FE13 |
| GAIN | FE14 |
| CONTROL2 | FE15 |
| C_TABLE_0 | FE20 |
| C_TABLE_... | FE21:FE3A |
| C_TABLE_27 | FE3B |
| VAL_TABLE_0 | FE40 |
| VAL_TABLE_... | FE41:FE5C |
| VAL_TABLE_29 | FE5D |

*Table 1.3: MODEM registers*

| REG. NAME | ADDRESS |
|---|---|
| VNR_TYPE | FEA0 |
| INP_ST | FEA1 |
| OUT_ST | FEA2 |
| OUT_REF_3 | FEA3 |
| OUT_REF_2 | FEA4 |
| OUT_REF_1 | FEA5 |
| OUT_REF_0 | FEA6 |
| POLARITY | FEA7 |

*Table 1.4: DIMMER PERIPHERAL registers*

As it is shown in Figure 1.1 peripheral blocks are connected to the microcontroller using address and data buses and the corresponding control signals; moreover, some generic port bits of the microcontroller are connected to peripheral control lines. These connections are detailed in Table 1.5 and Table 1.6.

| MICRO8051 | FLASH LOADER |
|---|---|
| P3.0 | SSI RxD |
| P3.1 | SSI TxD |
| P5.0 | SPI nCS |
| P5.1 | SPI SCK |
| P5.2 | SPI SI |
| P5.3 | SPI SO |

*Table 1.5: FLASH LOADER connection*

| MICRO8051 | MAC | |
|---|---|---|
| | bypass | running |
| P3.2 | Rx | Rx interrupt |
| P1.0 | CD | |
| P1.1 | Tx | |
| P1.2 | Tx/Rx | |

*Table 1.6: MAC connection*

Flash loader is connected to P3.0 and P3.1 and uses them as Rx/Tx to serial program download, after downloading the microcontroller takes control of those pins and uses them as Rx/Tx of its Standard Serial Interface (SSI). Pins P5(3:0) are used for the same purpose, program downloading, to connect with an external serial flash with Serial Peripheral Interface (SPI), after program loading these pins become standard microcontroller pin ports.

MAC peripheral can be used to implement the medium access control tasks or can be set in bypass mode, this allows user to implement medium access tasks in software. When the MAC is running P3.2 port is used to interrupt the microcontroller when a data byte is received. When the MAC is in bypass mode P3.2 and P1(2:0) are connected to the PLC modem control signals (Rx, CD, Tx, Rx/Tx).

Ports P1.4 and P1.3 are not available to the user and must not be written.

## 1.3. PROGRAM AND DATA MEMORIES

Program and Data for the microcontroller can be arranged in two different ways.

The program can be stored in an external ROM, as in a standard 8051 romless microcontroller, and an external SRAM is used to data storage, as it is shown in Figure 1.2.

Or, using the flash loader, the program is stored in a SPI serial flash and it is executed from a SRAM, see Figure 1.3. A serial link can be used to download the program from a computer to the serial flash, this allows in system reprogramming. The flash loader allows users to store several programs in the serial flash and to select (by software) the program to be executed. After power-up the flash loader forces a start-up cycle translating the selected program from the serial flash to the SRAM, the program is executed from the upper entries of the SRAM and the lower entries are used to store program data.

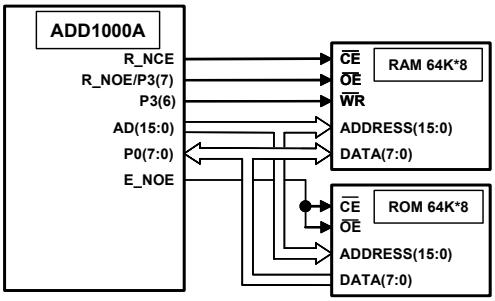



*Figure 1.2: ROM+SRAM schema*

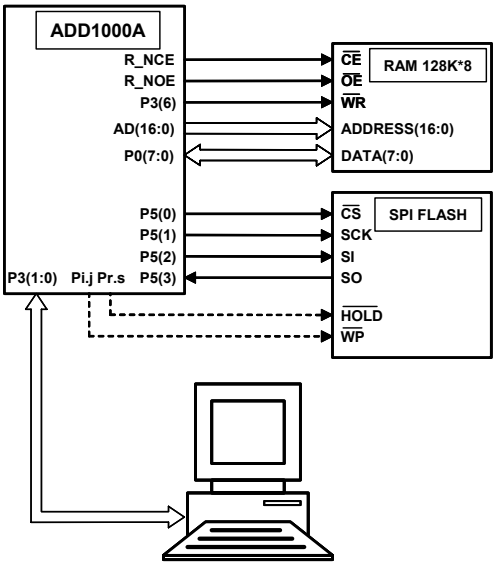



*Figure 1.3: serial flash storage schema*

Program storage configuration is selected using three control pins:

NRAM_ROM : when set to '0' the program is executed from the upper entries of the SRAM, when set to '0' the program is executed from the ROM

FLASH_NLD : when set to '0' the system is configured for program downloading using a serial connection, it must be set to '1' to run the program.

FAT_READ : when set to '0' the program must be stored in the serial flash starting at address "010000"(hex), when set to '1' the first entries (boot sector) of the serial flash contains the starting address of the program, this allows to store several programs in the serial flash and to change the program to be executed by writing the starting entry of the program in the boot sector of the serial flash.

## 1.4. CLOCK AND RESET

ADD1000AQF128 has two reset inputs (RSTA, RSTS). Any of these inputs must be held high for at least 32 clock cycles to ensure a correct start-up.

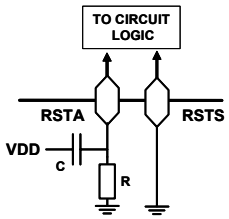



*Figure 1.4: power-up reset*

When a power-up reset is needed a simple RC network can be used, R and C values must be adjusted to ensure the minimum width for the reset pulse (i.e. 10uF and 10K, 1uF and 100K, …).

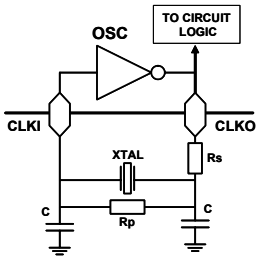



*Figure 1.5: oscillator circuit*

Clock signal must be applied to CLKI input when an external clock generator is used. When the internal oscillator circuit is used the external components must be connected to CLKI and CLKO as it is shown in Figure 1.5. Table 1.7 shows typical recommended values for the external components. Crystal frequency must always be 11.0592MHz.

| | |
|---|---|
| **XTAL** | 11.0592 MHz 50ppm |
| **C** | 18pF, 24pF |
| **Rp** | ≈1M |
| **Rs** | ≈300R |

*Table 1.7: oscillator components values*

## 1.5. PINOUT

| PIN | NAME | FUNCTION | DIR | I(mA) | RES | HY |
|---|---|---|---|---|---|---|
| 1 | RSTA | high active reset input | I | * | PD | Y |
| 2 | RSTS | high active reset input | I | * | PD | Y |
| 3 | VSS | DC supply ground | P | * | | |
| 4 | VDD | DC supply 3.3v | P | * | | |
| 5 | VSS | DC supply ground | P | * | | |
| 6 | EMIT1 | modem emission signal output | O | ±X | | |
| 7 | EMIT2 | modem emission signal output | O | ±X | | |
| 8 | EMIT3 | modem emission signal output | O | ±X | | |
| 9 | EMIT4 | modem emission signal output | O | ±X | | |
| 10 | VDD | DC supply 3.3v | P | * | | |
| 11 | VSS | DC supply ground | P | * | | |
| 12 | EMIT5 | Modem emission signal output | O | ±X | | |
| 13 | EMIT6 | modem emission signal output | O | ±X | | |
| 14 | EMIT7 | modem emission signal output | O | ±X | | |
| 15 | VSS | DC supply ground | P | * | | |
| 16 | VDD | DC supply 3.3v | P | * | | |
| 17 | EMIT8 | modem signal output | O | ±X | | |
| 18 | VDD | DC supply 3.3v | P | * | | |
| 19 | VSS | DC supply ground | P | * | | |
| 20 | EMIT9 | modem emission signal output | O | ±X | | |
| 21 | EMIT10 | modem emission signal output | O | ±X | | |
| 22 | EMIT11 | modem emission signal output | O | ±X | | |
| 23 | EMIT12 | modem emission signal output | O | ±X | | |
| 24 | VSS | DC supply ground | P | * | | |
| 25 | VDD | DC supply 3.3v | P | * | | |
| 26 | EMIT13 | modem emission signal output | O | ±X | | |
| 27 | EMIT14 | modem emission signal output | O | ±X | | |
| 28 | EMIT15 | modem emission signal output | O | ±X | | |
| 29 | VSS | DC supply ground | P | * | | |
| 30 | VDD | DC supply 3.3v | P | * | | |
| 31 | P15 | microcontroller port | B | ±6 | | |
| 32 | P16 | microcontroller port | B | ±6 | | |
| 33 | P17 | microcontroller port | B | ±6 | | |
| 34 | TRIAC0 | dimmer phase control signal output | O | ±12 | | |
| 35 | TRIAC1 | dimmer phase control signal output | O | ±12 | | |
| 36 | TRIAC2 | dimmer phase control signal output | O | ±12 | | |
| 37 | TRIAC3 | dimmer phase control signal output | O | ±12 | | |
| 38 | VNR | dimmer mains zero cross detection input | I | * | | Y |
| 39 | VDD | DC supply 3.3v | P | * | | |
| 40 | VSS | DC supply ground | P | * | | |
| 41 | INTERR0 | dimmer high voltage switch input signal | I | * | | Y |
| 42 | INTERR1 | dimmer high voltage switch input signal | I | * | | Y |
| 43 | INTERR2 | dimmer high voltage switch input signal | I | * | | Y |
| 44 | INTERR3 | dimmer high voltage switch input signal | I | * | | Y |

| PIN | NAME | FUNCTION | DIR | I(mA) | RES | HY |
|---|---|---|---|---|---|---|
| 45 | REC1 | reception loop output signal | O | ±6 | | |
| 46 | REC2 | reception loop output signal | O | ±6 | | |
| 47 | REC3 | reception loop output signal | O | ±6 | | |
| 48 | REC4 | reception loop output signal | O | ±6 | | |
| 49 | REC5 | reception loop output signal | O | ±6 | | |
| 50 | REC6 | reception loop output signal | O | ±6 | | |
| 51 | REC7 | reception loop output signal | O | ±6 | | |
| 52 | REC8 | reception loop output signal | O | ±6 | | |
| 53 | DNIN | input for comparator inverted output | I | * | | Y |
| 54 | DIN | input for comparator non inverted output | I | * | | Y |
| 55 | ENABLE | comparator latch control | O | ±12 | | |
| 56 | ADC | DONT CONNECT | O | ±2 | | |
| 57 | DC_COMP | reception loop DC adjust | O | ±12 | | |
| 58 | VSS | DC supply ground | P | * | | |
| 59 | VDD | DC supply 3.3v | P | * | | |
| 60 | INA | DONT CONNECT | I | * | | |
| 61 | OUTA | DONT CONNECT | O | ±12 | | |
| 62 | P57 | microcontroller port | B | ±2 | PU | |
| 63 | P56 | microcontroller port | B | ±2 | PU | |
| 64 | P55 | microcontroller port | B | ±2 | PU | |
| 65 | P54 | microcontroller port | B | ±2 | PU | |
| 66 | P53 | microcontroller port / flash loader SPI SO | B | ±2 | PU | |
| 67 | P52 | microcontroller port / flash loader SPI SI | B | ±2 | PU | |
| 68 | P51 | microcontroller port / flash loader SPI SCK | B | ±2 | PU | |
| 69 | P50 | microcontroller port / flash loader SPI nCS | B | ±2 | PU | |
| 70 | VDD | DC supply 3.3v | P | * | | |
| 71 | VSS | DC supply ground | P | * | | |
| 72 | P47 | microcontroller port | B | ±6 | PU | |
| 73 | P46 | microcontroller port | B | ±6 | PU | |
| 74 | P45 | microcontroller port | B | ±6 | PU | |
| 75 | P44 | microcontroller port | B | ±6 | PU | |
| 76 | P43 | microcontroller port | B | ±6 | PU | |
| 77 | P42 | microcontroller port | B | ±6 | PU | |
| 78 | P41 | microcontroller port | B | ±6 | PU | |
| 79 | P40 | microcontroller port | B | ±6 | PU | |
| 80 | P00 | microcontroller port | B | ±2 | PU | |
| 81 | P01 | microcontroller port | B | ±2 | PU | |
| 82 | P02 | microcontroller port | B | ±2 | PU | |
| 83 | P03 | microcontroller port | B | ±2 | PU | |
| 84 | P04 | microcontroller port | B | ±2 | PU | |
| 85 | P05 | microcontroller port | B | ±2 | PU | |
| 86 | P06 | microcontroller port | B | ±2 | PU | |
| 87 | P07 | microcontroller port | B | ±2 | PU | |
| 88 | R_NCE | microcontroller SRAM chip enable (active low) | O | ±2 | | |
| 89 | R_NOE | microcontroller SRAM output enable (active low) | O | ±2 | | |
| 90 | E_NOE | microcontroller ROM output enable (active low) | O | ±2 | | |
| 91 | VSS | DC supply ground | P | * | | |
| 92 | VDD | DC supply 3.3v | P | * | | |

| PIN | NAME | FUNCTION | DIR | I(mA) | RES | HY |
|---|---|---|---|---|---|---|
| 93 | AD0 | microcontroller data/prog. address bus | O | ±2 | | |
| 94 | AD1 | microcontroller data/prog. address bus | O | ±2 | | |
| 95 | AD2 | microcontroller data/prog. address bus | O | ±2 | | |
| 96 | AD3 | microcontroller data/prog. address bus | O | ±2 | | |
| 97 | AD10 | microcontroller data/prog. address bus | O | ±2 | | |
| 98 | AD11 | microcontroller data/prog. address bus | O | ±2 | | |
| 99 | AD9 | microcontroller data/prog. address bus | O | ±2 | | |
| 100 | AD8 | microcontroller data/prog. address bus | O | ±2 | | |
| 101 | AD13 | microcontroller data/prog. address bus | O | ±2 | | |
| 102 | VDD | DC supply 3.3v | P | * | | |
| 103 | VSS | DC supply ground | P | * | | |
| 104 | AD15 | microcontroller data/prog. address bus | O | ±2 | | |
| 105 | AD16 | microcontroller data/prog. address bus | O | ±2 | | |
| 106 | AD14 | microcontroller data/prog. address bus | O | ±2 | | |
| 107 | AD12 | microcontroller data/prog. address bus | O | ±2 | | |
| 108 | AD4 | microcontroller data/prog. address bus | O | ±2 | | |
| 109 | AD5 | microcontroller data/prog. address bus | O | ±2 | | |
| 110 | AD6 | microcontroller data/prog. address bus | O | ±2 | | |
| 111 | AD7 | microcontroller data/prog. address bus | O | ±2 | | |
| 112 | P37 | microcontroller port | B | ±2 | | |
| 113 | P36 | microcontroller port | B | ±2 | | |
| 114 | P35 | microcontroller port | B | ±2 | | |
| 115 | P34 | microcontroller port | B | ±2 | | |
| 116 | P33 | microcontroller port | B | ±2 | | |
| 117 | P31 | microcontroller port / flash loader SSI TxD | B | ±2 | | |
| 118 | P30 | microcontroller port / flash loader SSI RxD | B | ±2 | | |
| 119 | VDD | DC supply 3.3v | P | * | | |
| 120 | VSS | DC supply ground | P | * | | |
| 121 | CLKO | oscillator output | O | * | | |
| 122 | CLKI | oscillator/clock input | I | * | | |
| 123 | VSS | DC supply ground | P | * | | |
| 124 | VDD | DC supply 3.3v | P | * | | |
| 125 | NRAM_ROM | microcontroller program storage selection | I | * | PD | Y |
| 126 | FAT_READ | serial flash program storage mode selection | I | * | PU | Y |
| 127 | FLASH_NLD | serial flash programming enable | I | * | PU | Y |
| 128 | EW | microcontroller watchdog enable (active low) | I | * | PD | Y |

DIR : pin direction
- O : output
- I : input
- P : power

I : pin intensity
- + : source
- - : sink
- X : fixed by external resistor, see modem specifications

RES : pin pullup/pulldown resistor
- PU : 50K pullup
- PD : 50K pulldown

HY : input hysteresys

*Table 1.8: pinout specification*

## 1.6. PACKAGE

ADD1000AQF128 is packed in a thin quad flat 0.4mm pitch RoHS compliant package.

Table 1.9 shows the thermal resistance based on JEDEC standard (the test board size is 114.3mm x 76.2mm x 1.6mm, 4 copper layers).

| | θj-a | | | θj-c |
|---|---|---|---|---|
| **airflow** | 0m/s | 1m/s | 2m/s | |
| **ºC/W** | 41±15 | 39±15 | 37±15 | 6±15 |

*Table 1.9: QFP128 thermal resistance*

Figure 1.6 and Table 1.10 show package drawing and dimensions.

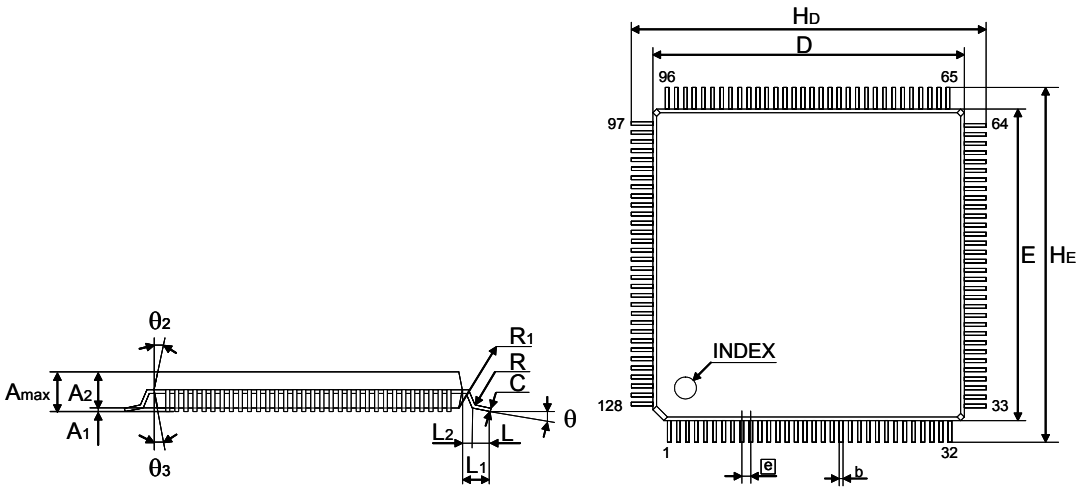



*Figure 1.6: package drawings*

| Symbol | Dimension in Milimeters | | | Dimension in Inches (for reference only) | | |
|---|---|---|---|---|---|---|
| | Min. | Nom. | Max. | Min. | Nom. | Max. |
| **E** | 13.9 | 14 | 14.1 | 0.548 | 0.551 | 0.555 |
| **D** | 13.9 | 14 | 14.1 | 0.548 | 0.551 | 0.555 |
| **A** | | | 1.7 | | | 0.066 |
| **A1** | | 0.1 | | | 0.004 | |
| **A2** | 1.3 | 1.4 | 1.5 | 0.052 | 0.055 | 0.059 |
| **e** | | 0.4 | | | 0.016 | |
| **b** | 0.11 | 0.16 | 0.26 | 0.005 | 0.006 | 0.010 |
| **C** | 0.1 | 0.125 | 0.175 | 0.004 | 0.005 | 0.006 |
| **θ** | 0° | | 10° | 0° | | 10° |
| **L** | 0.3 | 0.5 | 0.7 | 0.012 | 0.020 | 0.027 |
| **L1** | | 1 | | | 0.039 | |
| **L2** | | 0.5 | | | 0.020 | |
| **HE** | 15.6 | 16 | 16.4 | 0.615 | 0.630 | 0.645 |
| **HD** | 15.6 | 16 | 16.4 | 0.615 | 0.630 | 0.645 |
| **θ2** | | 12° | | | 12° | |
| **θ3** | | 12° | | | 12° | |
| **R** | | 0.2 | | | 0.008 | |
| **R1** | | 0.2 | | | 0.008 | |

*Table 1.10: package dimensions*

# CHAPTER 2. ADD8051C12A CORE DESCRIPTION

The following documentation provides a detailed description of the ADD8051C12A, architecture and hardware. ADD8051C12A is fully compatible with any 8051 legacy microcontroller, but there are some minor hardware differences.

Included in this documentation is a detailed description of registers and peripherals, some of those items are upgraded versions of the original 8051 microcontroller circuits, other are a simplified version.

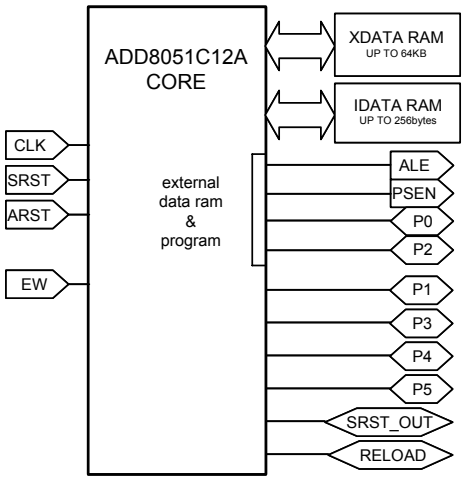



*Figure 2.1: core pinout*

## 2.1. PIN DESCRIPTION

CLK: clock input.

ARST, SRST: asynchronous and synchronous reset. SRST must be held active one clock cycle to ensure microcontroller restart.

EW: enable watchdog, low level active.

ALE/PROG: Address Latch Enable output pulses for latching the low byte of the address during accesses to external program/data memories. ALE pulses are emitted at a constant rate of 2 pulses each machine cycle. Except that one ALE pulse is skipped during each access to external data memory.

PSEN: Program Store Enable is the read strobe to external program memory. PSEN is activated twice each machine cycle, except that two PSEN pulses are skipped during accesses to external data memory.

PORT 0*: Port 0 is an 8-bit bidirectional port. Port 0 pins are not related to the SFR P0. Port 0 is the multiplexed low-order address and data bus to access external memories.

PORT 2: Port 2 is an 8-bit output port. Port 2 emits the high-order address byte during accesses to external program memory and to external data memory that use 16-bit addresses.

PORT 3*: Port 3 is an 8-bit pseudo bidirectional I/O port, external pull-ups must be added. It also serves the functions of various special features of the microcontroller:

Pin Alternate Function

P3.0 RxD (serial input port)

P3.1 TxD (serial output port)

P3.2 INT0 (external interrupt 0)

P3.3 INT1 (external interrupt 1)

P3.4 T0 (timer 0 external input)

P3.5 T1 (timer 1 external input)

P3.6 WR (external data memory write strobe)

P3.7 RD (external data memory read strobe)

PORTS 1, 4 and 5*: These ports are 8-bit pseudo bidirectional I/O port that needs pull-ups. Port pins that have 1s written to them are pulled high by the internal pull-ups, and in that state can be used as inputs. When a bit in a Port register has a 0 to 1 transition the related pin is driven high using transistor during 2 clock cycles, then the transistor is switched off and the pull-up resistor keeps the logic level. Port 4 pins can be configured either in pseudo-bidirectional mode or in push-pull, in push-pull the pin is always in output mode and logic 1 is always high driven.

(*) each bit of ports 0,1,3,4 and 5 is composed of three lines: data output, data input and output enable (low level active).

SPECIAL SFR OUTPUTS: some SFR bits have external connection to implement system functions

SRST_OUT: watchdog timer reset signal.

RELOAD: program reload control signal, software or watchdog timer activated, to force program reload from a serial storage device to a parallel SRAM.

INTERNAL MEMORIES CONNECTION: data, address and control signals to connect internal data memories. IDATA memory must always be present. XDATA memory is optional and can be distributed inside and/or outside the core, external XDATA is accessed through P0, P2 and P3 pins.

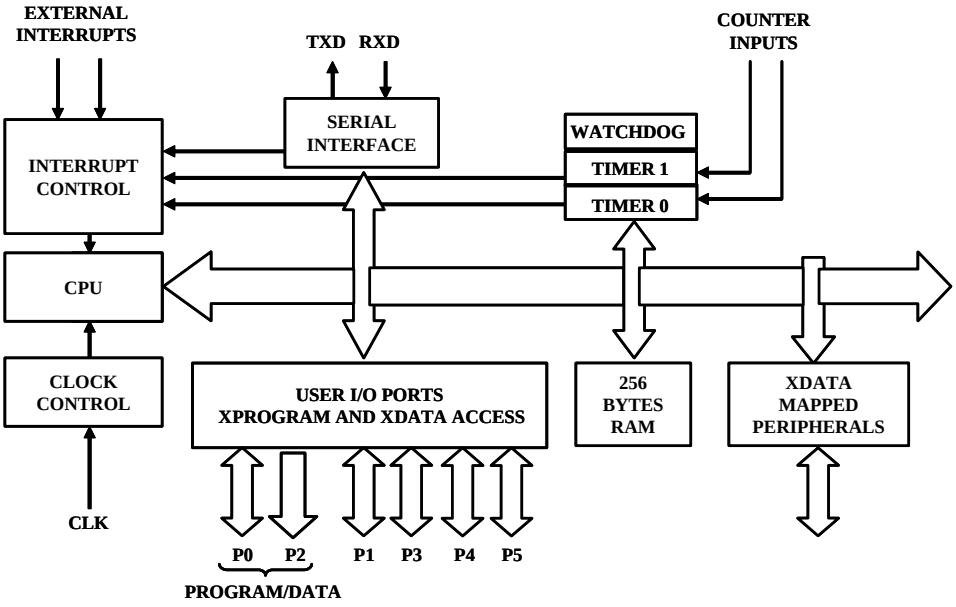



*Figure 2.2: block diagram*

## 2.2. MEMORY ORGANIZATION

ADD8051C12A has separate address spaces for program and data memory. The logical separation of program and data memory allows the data memory to be accessed by 8-bit addresses, which can be quickly stored and manipulated by an 8-bit CPU. Nevertheless, 16-bit data memory addresses can also be generated through the DPTR registers.

Program memory must only be read, not written to. There can be up to 64k bytes of program memory, and it can be internal and/or external. The read strobe for external program memory is the PSEN (program store enable).

Data Memory (RAM) occupies a separate address space from Program Memory. In the ADD8051C12A, the lowest 256 bytes of data memory (IDATA) are on chip. Up to 64k bytes of external RAM (XDATA) can be addressed in the external Data Memory space. The CPU generates read and write signals, RD and WR, as needed during external Data Memory accesses.

External Program Memory and external Data Memory may be combined and accessed using the same bus with the RD and PSEN signals functions to strobe to the external Program or Data memory.

The ADD8051C12A allows using any amount of Program Memory and XDATA Memory inside or outside the chip. This is a hardware feature and can not be modified by software. If a given program or xdata entry exists inside and outside the chip then the internal entry always override the external entry.

### 2.2.1. Program Memory

After reset, the CPU begins execution from location 0000H and stack pointer (SP) set to 07H.

Each interrupt is assigned a fixed location in Program Memory. The interrupt causes the CPU to jump to that location, where it commences execution of the service routine. External Interrupt 0, for example, is assigned to location 0003H. If External Interrupt 0 is going to be used, its service routine must begin at location 0003H. If the interrupt is not going to be used, its service location is available as general purpose Program Memory. The interrupt service locations are spaced at 8-byte intervals: 0003H for External Interrupt 0, 000BH for Timer 0, 0013H for External Interrupt 1 and 001BH for Timer 1. If an interrupt service routine is short enough, it can reside entirely within that 8-byte interval. Longer service routines can use a jump instruction to skip over subsequent interrupt locations, if other interrupts are in use.

For program execution16 I/O lines (Ports 0 and 2) are dedicated to bus functions during external Program Memory fetches. Port 0 (P0) serves as a multiplexed address/data bus. It emits the low byte of the Program Counter (PCL) as an address, and then goes into a float state awaiting the arrival of the code byte from the Program Memory. During the time that the low byte of the Program. Counter is valid on P0, the signal ALE (Address Latch Enable) clocks this byte into an address latch. Meanwhile, Port 2 (P2) emits the high byte of the Program Counter (PCH). Then PSEN strobes the Program

Memory and the code byte is read into the microcontroller.

Program Memory addresses are always 16 bits wide, even though the actual amount of Program Memory used may be less than 64k bytes. External program execution sacrifices two of the 8-bit ports, P0 and P2, to the function of addressing the Program Memory. To avoid unpredictable behavior user software must not access to the Special Function Registers (SFR) that control P0 and P2.

## 2.2.2. DATA MEMORY

Figure 2.3 shows the internal and external Data Memory spaces available to the microcontroller user.

Figure 2.4 shows a hardware configuration for accessing external RAM and program. Port 0 serves as a multiplexed address/data bus to the external memories, and lines of Port 2 are being used to page the RAM and program. The CPU generates RD and WR signals as needed during external RAM accesses. There can be up to 64k bytes of external Data Memory. External Data Memory addresses can be either 1 or 2 bytes wide. One-byte addresses are often used in conjunction with one or more other I/O lines to page the RAM, as shown in Figure 2.4.

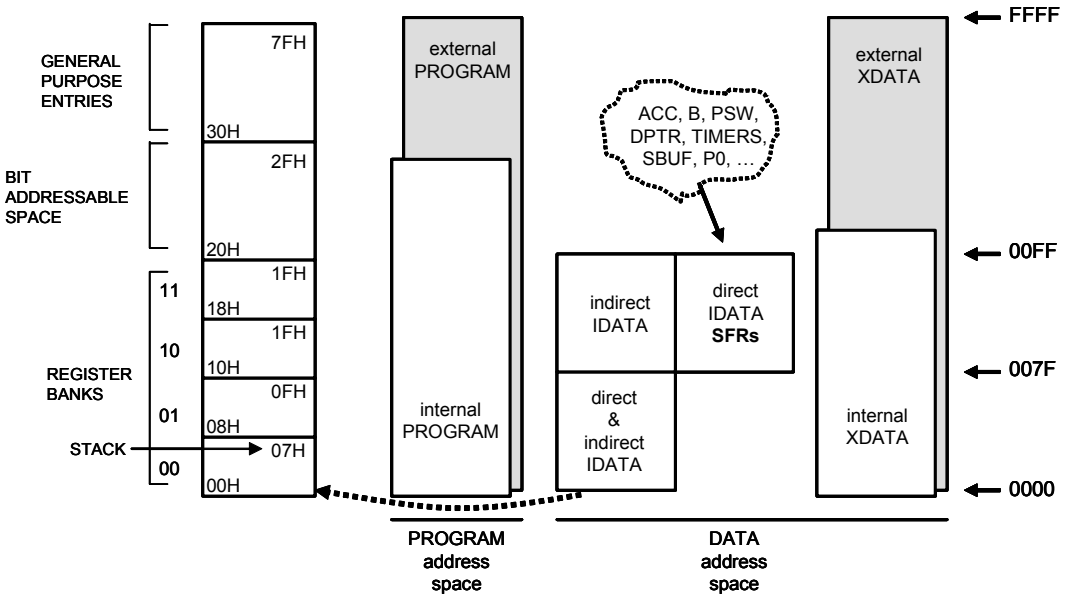



*Figure 2.3: program and data space*

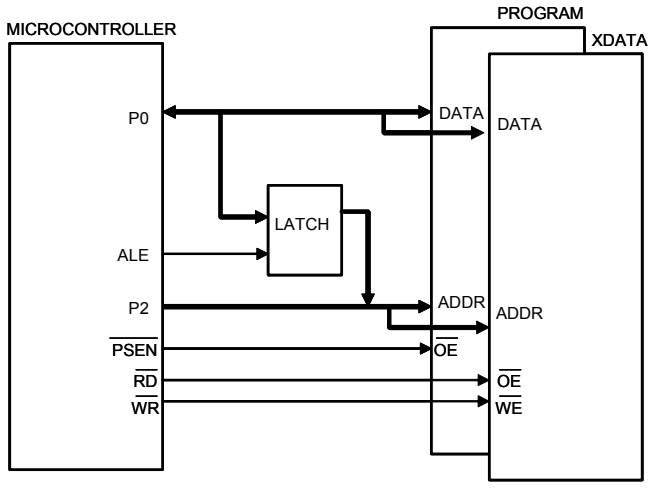



*Figure 2.4: connexion of external data and program memories*

Two-byte addresses can also be used, in which case the high address byte is emitted at Port 2.

Internal Data Memory is mapped in Figure 3. The internal IDATA memory space is shown divided into three blocks, which are generally referred to as the Lower 128, the Upper 128, and SFR space.

Internal Data Memory addresses are always one byte wide, which implies an address space of only 256 bytes. However, the addressing modes for internal RAM can in fact accommodate 384 bytes, using a simple trick. Direct addresses higher than 7FH access one memory space and indirect addresses higher than 7FH access a different memory space. Thus Figure 2.3 shows the Upper 128 and SFR space occupying the same block of addresses, 80H through FFH, although they are physically separate entities.

The lowest 32 bytes are grouped into 4 banks of 8 registers. Program instructions call out these registers as R0 through R7. Two bits in the Program Status Word (PSW) select which register bank is in use. This allows more efficient use of code space, since register instructions are shorter than instructions that use direct addressing.

The next 16 bytes above the register banks form a block of bit-addressable memory space. The 80C51 instruction set includes a wide selection of single-bit instructions, and the 128 bits in this area can be directly addressed by these instructions. The bit addresses in this area are 00H through 7FH.

All of the bytes in the Lower 128 can be accessed by either direct or indirect addressing.

The Upper 128 can only be accessed by indirect addressing.

Table 2.1 gives a look at the Special Function Register (SFR) space. SFRs include the Port registers, timers, peripheral controls, etc. These registers can only be accessed by direct addressing. Sixteen addresses in SFR space are both byte- and bit-addressable. The bit-addressable SFRs are those whose address ends in 0H or 8H. ADD8051C12A includes the same 21 sfrs included in the legacy 80C51 plus 7 additional sfrs to implement the extended features.

## 2.2.3. SFR Bank

Table 2.1 shows a map of the on-chip memory area called the Special Function Register (SFR) space. Shadowed entries are the registers not included in the standard architecture of the 8051, these entries are specific to ADD8051C12A. Blank entries are not implemented on the chip. User software should not write to these unimplemented locations. Read accesses to these addresses will in general return random data.

The functions of the SFRs are described in the text that follows.

**Accumulator**

ACC is the Accumulator register. The mnemonics for accumulator specific instructions, however, refer to the Accumulator simply as A.

**B Register**

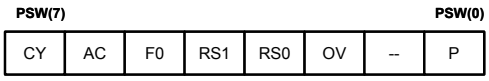
The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| F8 | | | | | | | | **T3** | FF |
| F0 | B | | | | | | | | F7 |
| E8 | | | | | | | | | EF |
| E0 | ACC | **CONF** | | | | | | | E7 |
| D8 | | | | | | | | | DF |
| D0 | PSW | | | | | | | | D7 |
| C8 | | | | | | | | | CF |
| C0 | **P4** | | | | **P5** | | | | C7 |
| B8 | IP | | | | | | | | BF |
| B0 | P3 | | | | | | | **IPH** | B7 |
| A8 | IE | | | | | | | | AF |
| A0 | P2 | | **AUX1** | | | | | | A7 |
| 98 | SCON | SBUF | | | | **P4M** | | | 9F |
| 90 | P1 | | | | | | | | 97 |
| 88 | TCON | TMOD | TL0 | TL1 | TH0 | TH1 | | | 8F |
| 80 | P0 | SP | DPL | DPH | | | | PCON | 87 |
| | BIT addr | | | | | | | | |

*Table 2.1: special function registers. (*shadowed sfrs are ADD8051C12A specific)*

### PSW register

The PSW register contains program status information as detailed in Figure 2.5.




PSW(0) : P : parity flag
PSW(1) : -- : user definable flag
PSW(2) : OV : overflow flag
PSW(3) : RS0 : register bank select control bit 0
PSW(4) : RS1 : register bank select control bit 1

| [RS1,RS0] | BANK | address |
|---|---|---|
| [ 0 , 0 ] | 0 | (07h:00h) |
| [ 0 , 1 ] | 1 | (0Fh:08h) |
| [ 1 , 0 ] | 2 | (17h:10h) |
| [ 1 , 1 ] | 3 | (1Fh:18h) |

PSW(5) : F0 : general purpose flag available to the user
PSW(6) : AC : auxiliary carry flag (3th to 4th bit carry)
PSW(7) : CY : carry flag

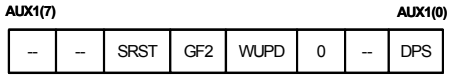
*Figure 2.5: Process Status Word*

### Stack Pointer

The Stack Pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at locations 08H.

### Auxiliary 1

The AUX1 register includes the data pointer selection bit, the software reset control and the switch to enable wake-up from power-down using external interrupts.




AUX1(0) : DPS : data pointer selector, selects between DPTR0 and DPTR1
AUX1(1) : -- : reserved bit
AUX1(2) : '0' : fixed '0'
AUX1(3) : WUPD : when set enables external interrupts driven wake-up from power-down
AUX1(4) : GF2 : general purpose flag, user definable flag
AUX1(5) : SRST: software reset
AUX1(7:6) : -- : reserved bit

*Figure 2.6: Auxiliary 1 register*

### Ports 0 to 5

P0, P1, P2, P3, P4 and P5 are the SFR registers of Ports 0, 1, 2, 3, 4 and 5 respectively. Writing a one/zero to a bit of a port SFR (P1, P3, P4, P5) causes the corresponding port output pin to switch low/high. When a port bit is used as an input the corresponding port SFR must be set to '1'. P0 and P2 ports are different, these ports are used as address and data busses for program and external data. P0 SFR does not control P0 port and P2 SFR is linked to P2 port only when MOVX @Ri instructions are executed, when these instructions are executed P2 SFR is used as most significant address byte.

Ports 1, 3 and 5 are pseudo bidirectional. Port 4 can be configured to push-pull or pseudo bidirectional using SFR P4M, when P4M(i) is set P4(i) is configured as push-pull.

### Data Pointer

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers. The same entry accesses two different data pointers, user software can switch between them using the data pointer selection bit in AUX1.

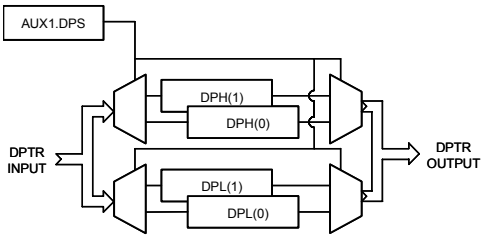



*Figure 2.7: DPTR selection*

### Serial Data Buffer

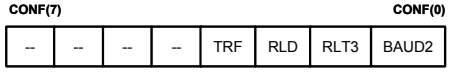
The Serial Buffer is actually two separate registers, a transmit buffer and a receive buffer. When data is moved to SBUF, it goes to the transmit buffer and is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

### Timer Registers

Register pairs (TH0, TL0), and (TH1, TL1) are the 16-bit Counting registers for Timer/Counters 0 and 1, respectively. T3 is the 8-bit counting register of the watchdog timer, the watchdog timer includes a 15-bit preescaler, the preescaler is set to 0 when T3 is reloaded. T3 reload is enabled only during 13 machine cycles after writing a '1' in PCON(4).

### CONF Register

CONF register includes specific configuration features for IC ADD1000.




CONF(0) : BAUD2 : doubles USART baudrates when set
CONF(1)* : RLT3 : when set it forces program reload from SPI flash after watchdog timeout
CONF(2)* : RLD: when set it forces program reload from SPI flash
CONF(3)* : TRF: reserved bit, don't use
CONF(7:4) : -- : reserved bits, don't use

*Figure 2.8: CONF register*

**Control Registers**

Special Function Registers IP, IE, TMOD, TCON, SCON and PCON contain control and status bits for the interrupt system, the Timer/Counters, and the serial port.

# 2.3. INSTRUCTION SET

The instruction set provides a variety of fast addressing modes for accessing the internal RAM to facilitate byte operations on small data structures. The instruction set provides extensive support for one-bit variables as a separate data type, allowing direct bit manipulation in control and logic systems that require Boolean processing.

## 2.3.1. Program Status Word

The Program Status Word (PSW) contains several status bits that reflect the current state of the CPU. The PSW, shown in Table 2.1, resides in the SFR space. It contains the Carry bit, the Auxiliary Carry (for BCD operations), the two register bank select bits, the Overflow flag, a Parity bit, and two user-definable status flags.

The Carry bit, other than serving the function of a Carry bit in arithmetic operations, also serves as the “Accumulator” for a number of Boolean operations.

The bits RS0 and RS1 are used to select one of the four register banks shown in Figure 2.7. A number of instructions refer to these RAM locations as R0 through R7. The selection of which of the four is being referred to is made on the basis of the RS0 and RS1 at execution time.

The Parity bit reflects the number of 1s in the Accumulator: P = 1 if the Accumulator contains an odd number of 1s, and P = 0 if the Accumulator contains an even number of 1s. Thus the number of 1s in the Accumulator plus P is always even. Two bits in the PSW are uncommitted and may be used as general purpose status flags.

## 2.3.2. Addressing Modes

The addressing modes are as follows:

**Direct Addressing**

In direct addressing the operand is specified by an 8-bit address field in the instruction. Only internal Data RAM and SFRs can be directly addressed.

**Indirect Addressing**

In indirect addressing the instruction specifies a register which contains the address of the operand. Both internal and external RAM can be indirectly addressed.

The address register for 8-bit addresses can be R0 or R1 of the selected bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit “data pointer” register, DPTR.

**Register Instructions**

The register banks, containing registers R0 through R7, can be accessed by certain instructions which carry a 3-bit register specification within the opcode of the instruction. Instructions that access the registers this way are code efficient, since this mode eliminates an address byte. When the instruction is executed, one of the eight registers in the selected bank is accessed. One of four banks is selected at execution time by the two bank select bits in the PSW.

**Register-Specific Instructions**

Some instructions are specific to a certain register. For example, some instructions always operate on the Accumulator, or Data Pointer, etc., so no address byte is needed to point to it. The opcode itself does that. Instructions that refer to the Accumulator as A assemble as accumulator specific opcodes.

**Immediate Constants**

The value of a constant can follow the opcode in Program Memory.

For example :

MOV A, 64H

loads the Accumulator with the number 64H.

**Indexed Addressing**

Only program Memory can be accessed with indexed addressing, and it can only be read. This addressing mode is intended for reading look-up tables in Program Memory A 16-bit base register (either DPTR or the Program Counter) points to the base of the table, and the Accumulator is set up with the table entry number.

The address of the table entry in Program Memory is formed by adding the Accumulator data to the base pointer.

Another type of indexed addressing is used in the “case jump” instruction. In this case the destination address of a jump instruction is computed as the sum of the base pointer and the Accumulator data.

## 2.3.3. Arithmetic Instructions

The menu of arithmetic instructions is listed in Table 1. The table indicates the addressing modes that can be used with each instruction to access the <byte> operand. For example, the ADD A,<byte> instruction can be written as:

ADD A, 6EH (direct addressing)

ADD A, @R1 (indirect addressing)

ADD A, R6 (register addressing)

ADD A, #206 (immediate constant)

The execution times listed in Table 2.2 assume a 12 clock cycles per machine cycle (mc), using a 12MHz clock 1 mc is executed in 1 us. All of the arithmetic instructions execute in 1mc except the INC DPTR instruction, which takes 2mc, and the multiply and divide instructions, which take 4mc.

Note that any byte in the internal Data Memory space can be incremented without going through the Accumulator. One of the INC instructions operates on the 16-bit Data Pointer. The Data Pointer is used to generate 16-bit addresses for external memory, so being able to increment it in one 16-bit operation is a useful feature.

The MUL AB instruction multiplies the Accumulator by the data in the B register and puts the 16-bit product into the concatenated B and Accumulator registers.

The DIV AB instruction divides the Accumulator by the data in the B register and leaves the 8-bit quotient in the Accumulator, and the 8-bit remainder in the B register Oddly enough, DIV AB finds less use in arithmetic “divide” routines than in radix conversions and programmable shift operations. In shift operations, dividing a number by 2n shifts its n bits to the right. Using DIV AB to perform the division completes the shift in 4mc and leaves the B register holding the bits that were shifted out.

The DA A instruction is for BCD arithmetic operations. In BCD arithmetic, ADD and ADDC instructions should always be followed by a DA A operation, to ensure that the result is also in BCD. Note that DA A will not convert a binary number to BCD. The DA A operation produces a meaningful result only as the second step in the addition of two BCD bytes.

## 2.3.4. Logical Instructions

Table 2.3 shows the list of logical instructions. The instructions that perform Boolean operations (AND, OR, Exclusive OR, NOT) on bytes perform the operation on a bit-by-bit basis. That is, if the Accumulator contains 00110101B and byte contains 01010011B, then:

ANL A, <byte>

will leave the Accumulator holding 00010001B.

The addressing modes that can be used to access the <byte> operand are listed in Table 2.2.

The ANL A, <byte> instruction may take any of the forms:

ANL A,6FH (direct addressing)

ANL A,@R0 (indirect addressing)

ANL A,R5 (register addressing)

ANL A,#42H (immediate constant)

All of the logical instructions that are Accumulator-specific execute in 1mc (using a 12MHz clock). The others take 2mc.

| MNEMONIC | OPERATION | ADDRESSING MODES | | | | EXECUTION TIMES (mc) |
|---|---|---|---|---|---|---|
| | | DIR | IND | REG | IMM | |
| ADD A,<byte> | A = A + <byte> | X | X | X | X | 1 |
| ADD A,<byte> | A = A + <byte> + C | X | X | X | X | 1 |
| SUBB A,<byte> | A = A – <byte> – C | X | X | X | X | 1 |
| INC A | A = A + 1 | Accumulator only | | | | 1 |
| INC<byte> | <byte> = <byte> + 1 | X | X | X | | 1 |
| INC DPTR | DPTR = DPTR + 1 | Data Pointer only | | | | 2 |
| DEC A | A = A – 1 | Accumulator only | | | | 1 |
| DEC<byte> | <byte> = <byte> – 1 | X | X | X | | 1 |
| MUL AB | B*A = B x A | ACC and B only | | | | 4 |
| DIV AB | A = Int[A/B]<br>B = Mod[A/B] | ACC and B only | | | | 4 |
| DA A | Decimal Adjust | Accumulator only | | | | 1 |

*Table 2.2: Arithmetic Instructions*

| MNEMONIC | OPERATION | ADDRESSING MODES | | | | EXECUTION TIME (mc) |
|---|---|---|---|---|---|---|
| | | DIR | IND | REG | IMM | |
| ANL A,<byte> | A = A.AND. <byte> | X | X | X | X | 1 |
| ANL <byte>,A | <byte> = <byte> .AND.A | X | | | | 1 |
| ANL <byte>,#data | <byte> = <byte> .AND.#data | X | | | | 2 |
| ORL A,<byte> | A = A.OR.<byte> | X | X | X | X | 1 |
| ORL <byte>,A | <byte> = <byte> .OR.A | X | | | | 1 |
| ORL <byte>,#data | <byte> = <byte> .OR.#data | X | | | | 2 |
| XRL A,<byte> | A = A.XOR. <byte> | X | X | X | X | 1 |
| XRL <byte>,A | <byte> = <byte> .XOR.A | X | | | | 1 |
| XRL <byte>,#data | <byte> = <byte> .XOR.#data | X | | | | 2 |
| CRL A | A = 00H | Accumulator only | | | | 1 |
| CPL A | A = .NOT.A | Accumulator only | | | | 1 |
| RL A | Rotate ACC Left 1 bit | Accumulator only | | | | 1 |
| RLC A | Rotate Left through Carry | Accumulator only | | | | 1 |
| RR A | Rotate ACC Right 1 bit | Accumulator only | | | | 1 |
| RRC A | Rotate Right through Carry | Accumulator only | | | | 1 |
| SWAP A | Swap Nibbles in A | Accumulator only | | | | 1 |

*Table 2.3: Logical Instructions*

| MNEMONIC | OPERATION | ADDRESSING MODES | | | | EXECUTION TIME (mc) |
|---|---|---|---|---|---|---|
| | | DIR | IND | REG | IMM | |
| MOV A,<src> | A = <src> | X | X | X | X | 1 |
| MOV <dest>,A | <dest> = A | X | X | X | | 1 |
| MOV <dest>,<src> | <dest> = <src> | X | X | X | X | 2 |
| MOV DPTR,#data16 | DPTR = 16-bit immediate constant | | | | X | 2 |
| PUSH <src> | INC SP:MOV“@SP”,<src> | X | | | | 2 |
| POP <dest> | MOV <dest>,“@SP”:DEC SP | X | | | | 2 |
| XCH A,<byte> | ACC and <byte> exchange data | X | X | X | | 1 |
| XCHD A,@Ri | ACC and @Ri exchange low nibbles | | X | | | 1 |

*Table 2.4: Data Tranfer Instructions that Access Internal Data Memory Space*

| ADDRESS WIDTH | MNEMONIC | OPERATION | EXECUTION TIME (mc) |
|---|---|---|---|
| 8 bits | MOVX A,@Ri | Read external RAM @Ri | 2 |
| 8 bits | MOVX @Ri,A | Write external RAM @ Ri | 2 |
| 16 bits | MOVX A,@DPTR | Read external RAM @ DPTR | 2 |
| 16 bits | MOVX @DPTR,A | Write external RAM @ DPTR | 2 |

*Table 2.5: Data Tranfer Instructions that Access External Data Memory Space*

| MNEMONIC | OPERATION | EXECUTION TIME (mc) |
|---|---|---|
| MOVC A,@A+DPTR | Read program memory at (A + DPTR) | 2 |
| MOVC A,@A+PC | Read program memory at (A + PC) | 2 |

*Table 2.6: Code Constants Read Instructions*

Note that Boolean operations can be performed on any byte in the internal Data Memory space without going through the Accumulator. The XRL <byte>, #data instruction, for example, offers a quick and easy way to invert port bits, as in XRL P1, #OFFH.

If the operation is in response to an interrupt, not using the Accumulator saves the time and effort to push it onto the stack in the service routine.

The Rotate instructions (RL, A, RLC A, etc.) shift the Accumulator 1 bit to the left or right. For a left rotation, the MSB rolls into the LSB position. For a right rotation, the LSB rolls into the MSB position.

The SWAP A instruction interchanges the high and low nibbles within the Accumulator. This is a useful operation in BCD manipulations.

## 2.3.5. Data Transfers

**Internal RAM**

Table 2.4 shows the menu of instructions that are available for moving data around within the internal memory spaces, and the addressing modes that can be used with each one. All of these instructions execute in either 1 or 2 mc.

The MOV <dest>, <src> instruction allows data to be transferred between any two internal RAM or SFR locations without going through the Accumulator. Remember, the upper 128 bytes of data RAM can be accessed only by indirect addressing, and SFR space only by direct addressing.

Note that in any 80C51 device, the stack resides in on-chip RAM, and grows upwards. The PUSH instruction first increments the Stack Pointer (SP), then copies the byte into the stack. PUSH and POP use only direct addressing to identify the byte being saved or restored, but the stack itself is accessed by indirect addressing using the SP register. This means the stack can go into the Upper 128 bytes of RAM, if they are implemented, but not into SFR space.

The Data Transfer instructions include a 16-bit MOV that can be used to initialize the Data Pointer (DPTR) for look-up tables in Program Memory, or for 16-bit external Data Memory accesses.

The XCH A, <byte> instruction causes the Accumulator and addressed byte to exchange data. The XCHD A, @Ri instruction is similar, but only the low nibbles are involved in the exchange.

**External RAM**

Table 2.5 shows a list of the Data Transfer instructions that access external Data Memory. Only indirect addressing can be used. The choice is whether to use a one-byte address, @Ri, where Ri can be either R0 or R1 of the selected register bank, or a two-byte address, @DPTR. The disadvantage to using 16-bit addresses if only a few k bytes of external RAM are involved is that 16-bit addresses use all 8 bits of Port 2 as address bus. On the other hand, 8-bit addresses allow one to address a few bytes of RAM without having to sacrifice all of Port 2. All of these instructions execute in 2 mc.

Note that in all external Data RAM accesses, the Accumulator is always either the destination or source of the data.

The read and write strobes to external RAM are activated only during the execution of a MOVX instruction. Normally these signals are inactive, and in fact if they're not going to be used at all, their pins are available as extra I/O lines.

**Code Constants**

Table 2.6 shows the two instructions that are available for reading code constants in Program Memory. Since these instructions access only Program Memory, the code constants can only be read, not updated.

The mnemonic is MOVC for "move constant." The first MOVC instruction in Table 2.6 can accommodate a table of up to 256 entries numbered 0 through 255. The number of the desired entry is loaded into the Accumulator, and the Data Pointer is set up to point to the beginning of the table. Then:

```
MOVC A,@A+DPTR
```

copies the desired table entry into the Accumulator.

The other MOVC instruction works the same way, except the Program Counter (PC) is used as the base address

```
MOVC A,@A+PC
```

## 2.3.6. Boolean Instructions

80C51 devices contain a complete Boolean (single-bit) processor. The internal RAM contains 128 addressable bits, and the SFR space can support up to 128 addressable bits as well. All of the port lines are bit-addressable, and each one can be treated as a separate single-bit port. The instructions that access these bits are not just conditional branches, but a complete menu of move, set, clear, complement, OR, and AND instructions. These kinds of bit operations are not easily obtained in other architectures with any amount of byte-oriented software. The instruction set for the Boolean processor is shown in Table 2.7. All bit accesses are by direct addressing.

Bit addresses 00H through 7FH are in the Lower 128, and bit addresses 80H through FFH are in SFR space.

The Carry bit in the PSW is used as the single-bit Accumulator of the Boolean processor. Bit instructions that refer to the Carry bit as C assemble as Carry-specific instructions (CLR C,

etc.). The Carry bit also has a direct address, since it resides in the PSW register, which is bit-addressable.

Note that the Boolean instruction set includes ANL and ORL operations, but not the XRL (Exclusive OR) operation.

There is a series of bit-test instructions which execute a jump if the addressed bit is set (JC, JB, JBC) or if the addressed bit is not set (JNC, JNB). In the above case, bit2 is being tested, and if bit2 = 0, the CPL C instruction is jumped over.

JBC executes the jump if the addressed bit is set, and also clears the bit. Thus a flag can be tested and cleared in one operation. All the PSW bits are directly addressable, so the Parity bit or the general purpose flags, for example, are also available to the bit-test instructions.

**Relative Offset**

The destination address for these jumps is specified to the assembler by a label or by an actual address in Program memory. However, the destination address assembles to a relative offset byte. This is a signed (two's complement) offset byte which is added to the PC in two's complement arithmetic if the jump is executed. The range of the jump is therefore –128 to +127 Program Memory bytes relative to the first byte following the instruction.

| MNEMONIC | OPERATION | EXECUTION TIME (mc) |
|---|---|---|
| ANL C,bit | C = C.AND.bit | 2 |
| ANL C,/bit | C = C.AND..NOT.bit | 2 |
| ORL C,bit | C = C.OR.bit | 2 |
| ORL C,/bit | C = C.OR..NOT.bit | 2 |
| MOV C,bit | C = bit | 1 |
| MOV bit,C | bit = C | 2 |
| CLR C | C = 0 | 1 |
| CLR bit | bit = 0 | 1 |
| SETB C | C = 1 | 1 |
| SETB bit | bit = 1 | 1 |
| CPL C | C = .NOT.C | 1 |
| CPL bit | bit = .NOT.bit | 1 |
| JC rel | Jump if C = 1 | 2 |
| JNC rel | Jump if C = 0 | 2 |
| JB bit,rel | Jump if bit = 1 | 2 |
| JNB bit,rel | Jump if bit = 0 | 2 |
| JBC bit,rel | Jump if bit = 1; CLR bit | 2 |

*Table 2.7: Boolean Instructions*

| MNEMONIC | OPERATION | EXECUTION TIME (mc) |
|---|---|---|
| (S)JMP addr | Jump to addr | 2 |
| (L)JMP addr | Jump to addr | 2 |
| (A)JMP addr | Jump to addr | 2 |
| JMP @A+DPTR | Jump to A + DPTR | 2 |
| (A,L)CALL addr | Call subroutine at addr | 2 |
| RET | Return from subroutine | 2 |
| RETI | Return from interrupt | 2 |
| NOP | No operation | 1 |

*Table 2.8: Unconditional Jump Instructions*

| MNEMONIC | OPERATION | ADDRESSING MODES | | | | EXECUTION TIME(mc) |
|---|---|---|---|---|---|---|
| | | DIR | IND | REG | IMM | |
| JZ rel | Jump if A = 0 | | | | | 2 |
| JNZ rel | Jump if A /= 0 | | | | | 2 |
| DJNZ <byte>,rel | Decrement and jump if not zero | X | | X | | 2 |
| CJNE A,<byte>,rel | Jump if A /= <byte> | X | | | X | 2 |
| CJNE <byte>,#data,rel | Jump if <byte> /= #data | | X | X | | 2 |

*Table 2.9: Conditional Jumps*

### 2.3.7. Jump Instructions

Table 2.8 shows the list of unconditional jumps with execution time.

The table lists SJMP, LJMP, and AJMP, which differ in the format of the destination address. JMP is a generic mnemonic which can be used if the programmer does not care which way the jump is encoded.

The SJMP instruction encodes the destination address as a relative offset, as described above. The instruction is 2 bytes long, consisting of the opcode and the relative offset byte. The jump distance is limited to a range of –128 to +127 bytes relative to the instruction following the SJMP.

The LJMP instruction encodes the destination address as a 16-bit constant. The instruction is 3 bytes long, consisting of the opcode and two address bytes. The destination address can be anywhere in the 64k Program Memory space.

The AJMP instruction encodes the destination address as an 11-bit constant. The instruction is 2 bytes long, consisting of the opcode, which itself contains 3 of the 11 address bits, followed by another byte containing the low 8 bits of the destination address. When the instruction is executed, these 11 bits are simply substituted for the low 11 bits in the PC. The high 5 bits stay the same. Hence the destination has to be within the same 2k block as the instruction following the AJMP.

The JMP @A+DPTR instruction supports case jumps. The destination address is computed at execution time as the sum of the 16-bit DPTR register and the Accumulator. Typically, DPTR is set up with the address of a jump table.

Table 2.8 shows two “CALL addr” instructions LCALL and ACALL, which differ in the format in which the subroutine address is given to the CPU. CALL is a generic mnemonic which can be used if the programmer does not care which way the address is encoded.

The LCALL instruction uses the 16-bit address format, and the subroutine can be anywhere in the 64k Program Memory space.

The ACALL instruction uses the 11-bit format, and the subroutine must be in the same 2k block as the instruction following the ACALL.

Subroutines should end with a RET instruction, which returns execution to the instruction following the CALL.

RETI is used to return from an interrupt service routine. The only difference between RET and RETI is that RETI tells the interrupt control system that the interrupt in progress is done. If there is no interrupt in progress at the time RETI is executed, then the RETI is functionally identical to RET.

Table 2.9 shows the list of conditional jumps available to the microcontroller user. All of these jumps specify the destination address by the relative offset method, and so are limited to a jump distance of –128 to +127 bytes from the instruction following the conditional jump instruction. Important to note, however, the user specifies to the assembler the actual destination address the same way as the other jumps: as a label or a 16-bit constant.

There is no Zero bit in the PSW. The JZ and JNZ instructions test the Accumulator data for that condition.

The DJNZ instruction (Decrement and Jump if Not Zero) is for loop control.

The CJNE instruction (Compare and Jump if Not Equal) can also be used for loop control. Two bytes are specified in the operand field of the instruction. The jump is executed only if the two bytes are not equal. Another application of this instruction is in “greater than, less than” comparisons. The two bytes in the operand field are taken as unsigned integers. If the first is less than the second, then the Carry bit is set (1). If the first is greater than or equal to the second, then the Carry bit is cleared.

## 2.4. CPU TIMING

Execution time is divided into machine cycles. A machine cycle consists of a sequence of 6 states, numbered S1 through S6. Each state time lasts for two clock periods. Thus a machine cycle takes 12 clock periods or 1μs if the clock frequency is 12MHz.

Each state is divided into a Phase 1 half and a Phase 2 half. Figure 2.9 shows that fetch/execute sequences in states and phases for various kinds of instructions. Normally two program fetches are generated during each machine cycle, even if the instruction being executed doesn’t require it. If the instruction being executed doesn’t need more code bytes, the CPU simply ignores the extra fetch, and the Program Counter is not incremented.

Execution of a one-cycle instruction (Figure 2.9a and Figure 2.9b) begins during State 1 of the machine cycle, when the opcode is latched into the Instruction Register. A second fetch occurs during S4 of the same machine cycle. Execution is complete at the end of State 6 of this machine cycle.

The MOVX instructions take two machine cycles to execute. No program fetch is generated during the second cycle of a MOVX instruction. This is the only time program fetches are skipped. The fetch/execute sequence for MOVX instructions is shown in Figure 2.9d.

The fetch/execute sequences are the same whether the Program Memory is internal or external to the chip. Execution times do not

depend on whether the Program Memory is internal or external.

Figure 2.10 shows the signals and timing involved in program fetches when the Program Memory is external. If Program Memory is external, then the Program Memory read strobe PSEN is normally activated twice per machine cycle, as shown in Figure 10a. If an access to external Data Memory occurs, as shown in Figure 10b, two PSENs are skipped, because the address and data bus are being used for the Data Memory access.

Note that a Data Memory bus cycle takes twice as much time as a Program Memory bus cycle. Figure 2.10 shows the relative timing of the addresses being emitted at Ports 0 and 2, and of ALE and PSEN. ALE is used to latch the low address byte from P0 into the address latch. Note that an ALE pulse is skipped during the execution of the MOVX instruction.

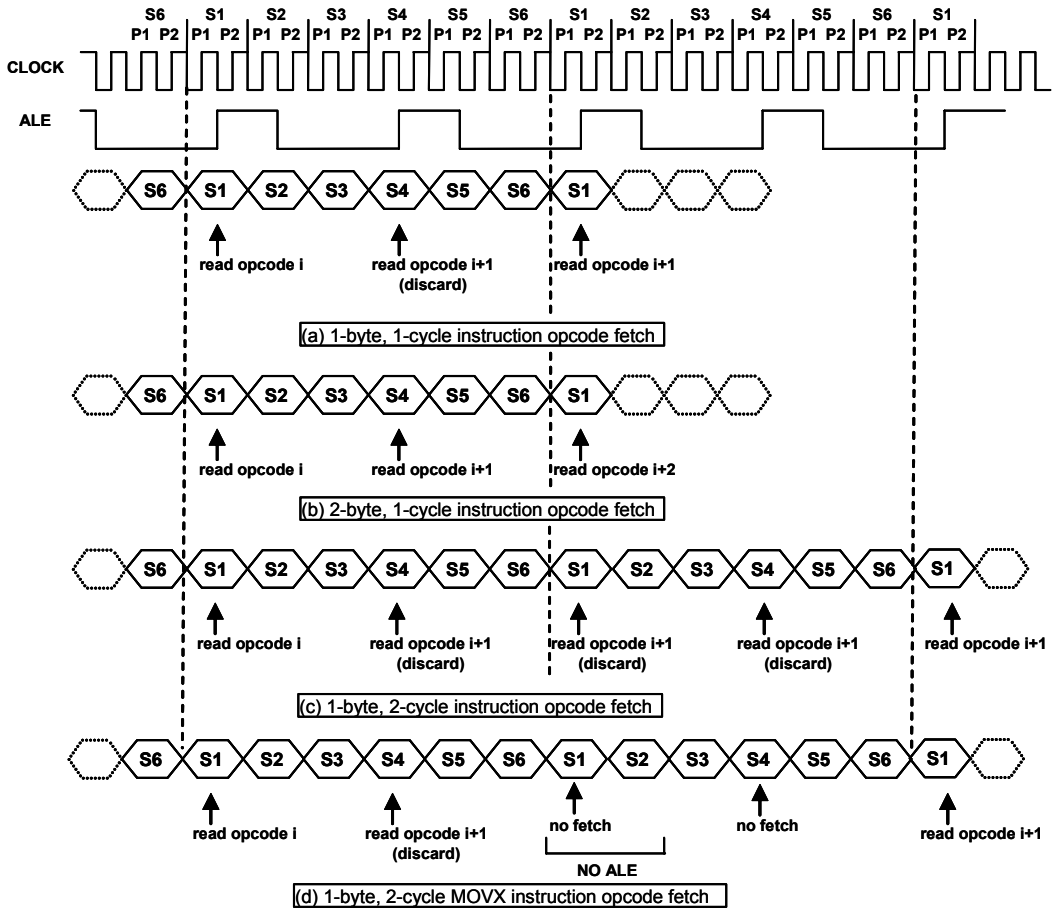



*Figure 2.9: States and Fetches Sequence*

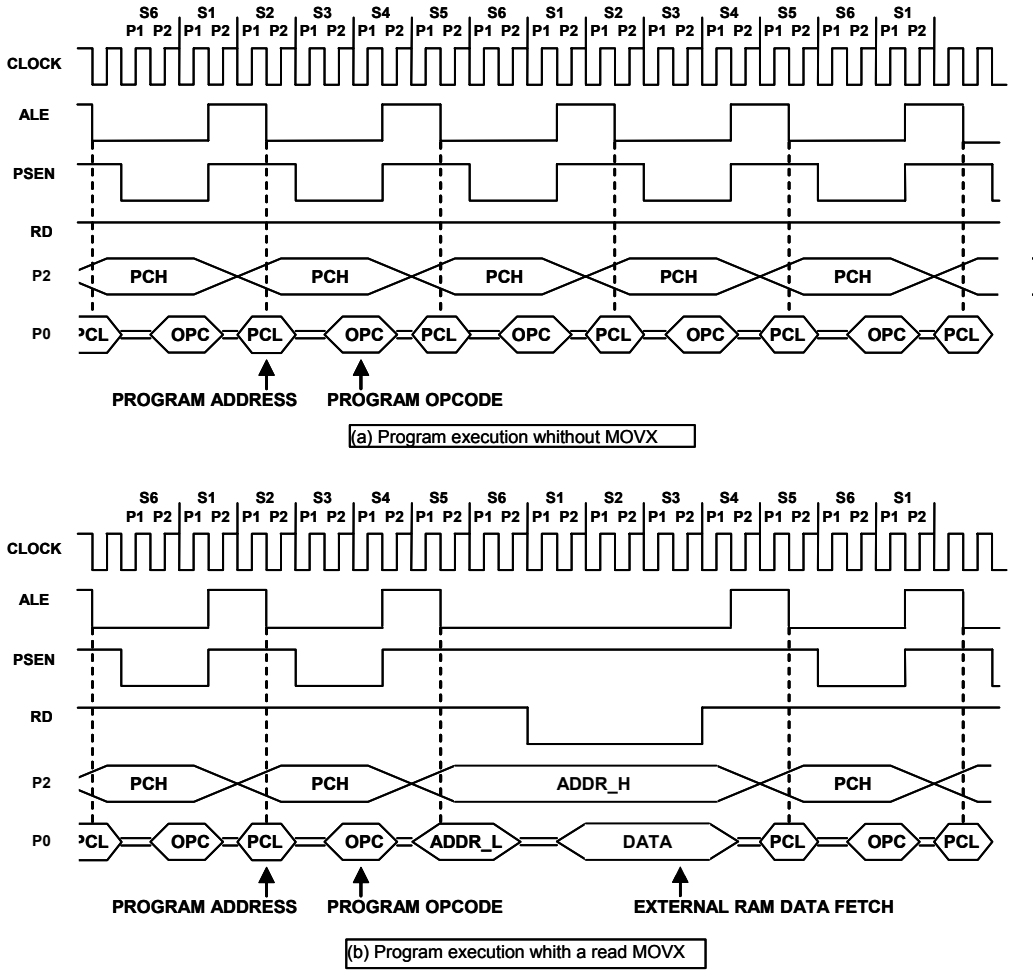



*Figure 2.10: Bus Cycles in 80C51 Family Devices Executing from External Program Memory*

## 2.4.1. Reset

The reset inputs are the ARST pin and SRST pin. ARST is an asynchronous reset and SRST is a synchronous reset. One of them must be used to ensure a correct initial state.

The reset writes 0s to the greater part of the SFRs. The port registers are initialized to FFH to keep port pins in pull-up state. The Stack Pointer is initialized to 07H. Table 2.10 lists the SFR reset values. The internal RAM is not affected.

## 2.4.2. Power-Saving Modes

ADD8051C12A has two power reducing modes, Idle and Power Down.

In the Idle mode (IDL = 1), the CPU is stopped but the peripheral circuits continues to run.

In Power Down (PD = 1), the CPU and the peripheral circuits are stopped, except that external interrupts hardware can be configured to run when in power down mode.

The Idle and Power Down Modes are activated by setting bits in Special Function Register PCON. If 1s are written to PD and IDL at the same time, PD takes precedence.

When watchdog is enabled power down modes are automatically disabled. Figure 2.11: PCON register details its contents.

PCON(7) PCON(0)

| SMOD | -- | -- | WLE | GF1 | GF0 | PD | IDL |
|---|---|---|---|---|---|---|---|

PCON(0) : IDL : sets idle mode
PCON(1) : PD : sets power-down mode
PCON(2) : GF0: general purpose flag bit, user programmable
PCON(3) : GF1: general purpose flag bit, user programmable
PCON(4) : WLE : watchdog load enable, it must be set by software to enable T3 reload, it is reset by hardware after 13 machine cycles
PCON(6:5) : -- : reserved bits, don't use
PCON(7) : SMOD : doble baud rate bit when timer 1 is used as time generator and serial port is in modes 1,2 or 3

*Figure 2.11: PCON register*

### 2.4.2.a. Idle Mode

An instruction that sets PCON.0 causes that to be the last instruction executed before going into the Idle mode, CPU is stopped but not to the Interrupt, Timer, and Serial Port functions. The CPU status is preserved in its entirety; the Stack Pointer, Program Counter, Program Status Word, Accumulator, and all other registers maintain their data during Idle. The port pins hold the logical states they had at the time Idle

was activated. ALE and PSEN hold at logic high levels.

There are two ways to terminate the Idle. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating the Idle mode. The interrupt will be serviced, and following RETI, the next instruction to be executed will be the one following the instruction that put the device into Idle.

The signal at the ARST or SRST pin clears the IDL bit directly. At this time, and with no delay or with one clock cycle delay, the CPU resumes program execution from where it left off; that is, at the instruction following the one that invoked the Idle Mode.

| S.F.R. | address | rst value |
|---|---|---|
| P0 | H'80 | 11111111 |
| SP | H'81 | 00000111 |
| DPL | H'82 | 00000000 |
| DPH | H'83 | 00000000 |
| PCON | H'87 | 00000000 |
| TCON | H'88 | 00000000 |
| TMOD | H'89 | 00000000 |
| TL0 | H'8A | 00000000 |
| TL1 | H'8B | 00000000 |
| TH0 | H'8C | 00000000 |
| TH1 | H'8D | 00000000 |
| P1 | H'90 | 11111111 |
| SCON | H'98 | 00000000 |
| SBUF | H'99 | 00000000 |
| P4M2 | H'9D | 00000000 |
| P2 | H'A0 | 11111111 |
| AUXR1 | H'A2 | 00000000 |
| IE | H'A8 | 00000000 |
| P3 | H'B0 | 11111111 |
| IP0H | H'B7 | 00000000 |
| IP0 | H'B8 | 00000000 |
| P4 | H'C0 | 11111111 |
| P5 | H'C4 | 11111111 |
| PSW | H'D0 | 00000000 |
| ACC | H'E0 | 00000000 |
| B | H'F0 | 00000000 |
| CONF | H'F1 | 00000000 |
| T3 | H'FF | 00000000 |

*Table 2.10: reset values*

#### 2.4.2.b. Power-Down Mode

An instruction that sets PCON.1 causes that to be the last instruction executed before going into the Power Down mode. In the Power Down mode all functions are stopped, except that external interrupts hardware. The contents of the on-chip RAM and Special Function Registers are maintained. The port pins output the values held by their respective SFRs. The ALE and PSEN output are held low.

There are two ways to terminate the Power Down mode. Hardware reset, reset redefines all the SFRs, but does not change the on-chip RAM. External interrupt, when they are enabled a low level or high to low transition at their inputs will cause a microcontroller restart and a jump to the corresponding interruption routine. As in idle mode the interrupt will be serviced, and following RETI, the next instruction to be executed will be the one following the instruction that put the device into power down.

## 2.5. INTERRUPTS

The microcontroller has 5 interrupt sources: 2 external interrupts, 2 timer interrupts, and the serial port interrupt.

### 2.5.1. INTERRUPT ENABLING

Each interrupt source can be individually enabled or disabled by setting or clearing a bit in the SFR named IE (Interrupt Enable). This register also contains a global disable bit, which can be cleared to disable all interrupts at once. Figure 2.12: Interrupt Control Registers shows the IE register.

### 2.5.2. INTERRUPT PRIORITIES

Each interrupt source can also be individually programmed to four priority levels by setting or clearing two bits in the SFRs named IP and IPH (Interrupt Priority). Figure 2.12 and Figure 2.13 show how the IE, IP and IPH registers and the polling sequence work to determine the interrupt to be serviced. The IP and IPH registers contain a number of unimplemented bits: 7, 6, and 5, these bits are reserved and user software should not write to these positions.

A low-priority interrupt can be interrupted by a high-priority interrupt, but not by another low-priority interrupt. A high-priority interrupt can't be interrupted by any other interrupt source.

If two interrupt requests of different priority levels are received simultaneously, the request of higher priority is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced. Thus within each priority level there is a second priority structure determined by the polling sequence as follows:

1.IE0 (highest priority)

2.TF0

3.IE1

4.TF1

5.RI+TI (lowest priority)

In operation, all the interrupt flags are latched into the interrupt control system during State 5 of every machine cycle. The samples are polled during the following machine cycle. If the flag for an enabled interrupt is found to be set (1), the interrupt system generates an LCALL to the appropriate location in Program Memory, unless some other condition blocks the interrupt. Several conditions can block an interrupt, among

them that an interrupt of equal or higher priority level is already in progress.

The hardware-generated LCALL causes the contents of the Program Counter to be pushed into the stack, and reloads the PC with the beginning address of the service routine. As previously noted the service routine for each interrupt begins at a fixed location.

Only the Program Counter is automatically pushed onto the stack, not the PSW or any other register.

Having only the PC automatically saved it is a programmer task to save other registers if necessary.

IE(7) IE(0)

| EA | X | X | ES | ET1 | EX1 | ET0 | EX0 |
|---|---|---|---|---|---|---|---|

**IE(0) : enables '1' or disables '0' external interrupt 0**
**IE(1) : enables '1' or disables '0' timer 0 interrupt**
**IE(2) : enables '1' or disables '0' external interrupt 1**
**IE(3) : enables '1' or disables '0' timer 1 interrupt**
**IE(4) : enables '1' or disables '0' serial port interrupt**
**IE(5) : reserved**
**IE(6) : reserved**
**IE(7) : enables '1' or disables '0' all interrupts**

IP(7) IP(0)

| X | X | X | PS | PT1 | PX1 | PT0 | PX0 |
|---|---|---|---|---|---|---|---|

IPH(7) IPH(0)

| X | X | X | PS | PT1 | PX1 | PT0 | PX0 |
|---|---|---|---|---|---|---|---|

**[IPH(0),IP(0)] : defines external interrupt 0 priority level**
**[IPH(1),IP(1)] : defines timer 0 interrupt priority level**
**[IPH(2),IP(2)] : defines external interrupt 1 priority level**
**[IPH(3),IP(3)] : defines timer 1 interrupt prioriry level**
**[IPH(4),IP(4)] : defines serial port interrupt priority level**
**IP(5) : reserved**
**IP(6) : reserved**
**IP(7) : reserved**
**IPH(5) : reserved**
**IPH(6) : reserved**
**IPH(7) : reserved**

*Figure 2.12: Interrupt Control Registers*

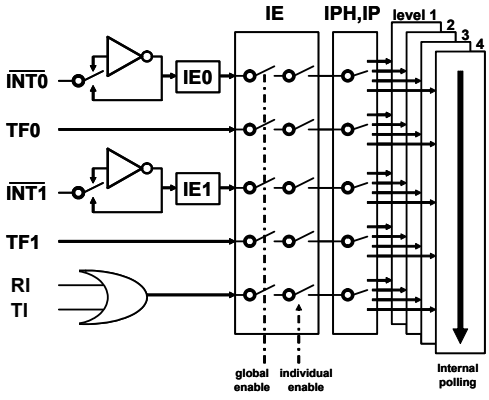



*Figure 2.13: Interrupt priorities*

## 2.5.3. INTERRUPT HANDLING

The External Interrupts INT0 and INT1 can each be either level-activated or transition-activated, depending on bits IT0 and IT1 in Register TCON. When external interrupts are transition-activated the INTx pin must show a high in one cycle and a low in the next cycle to generate the interrupt, so that an input high or low should hold for at least 1 machine cycle to ensure sampling. The flags that actually generate these interrupts are bits IE0 and IE1 in TCON. The flags are cleared by the hardware when the service routine is vectored to only if the interrupts are transition-activated. When the interrupts are level-activated the interrupt flags are controlled by the external source, so that the external source has to deactivate the request before the interrupt service routine is completed, or else another interrupt will be generated. The response time is always more than 3 cycles and less than 9 cycles depending on program execution and interrupt status (see figure 14).

The Timer 0 and Timer 1 Interrupts are generated by TF0 and TF1, which are set by a rollover. The flags are cleared by the on-chip hardware when the service routines are vectored to.

The Serial Port Interrupt is generated by the logical OR of RI and TI. They must be cleared in software.

All of the bits that generate interrupts can be set or cleared by software. That is, interrupts can be generated or pending interrupts can be canceled in software. Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE (Figure 2.12). IE also contains a global disable bit, EA, which disables all interrupts at once.

The interrupt flags are sampled at every machine cycle. The samples are polled during the following machine cycle. If one of the flags was in a set condition at the preceding cycle, the polling cycle will find it and the interrupt system will generate an LCALL to the appropriate service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority level is already in progress.
2. The current (polling) cycle is not the final cycle in the execution of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE or IP registers.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress will be completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any access to IE or IP, then at least one more instruction will be executed before any interrupt is vectored to.

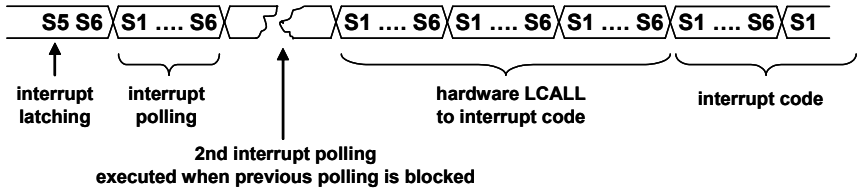



*Figure 2.14: Interrupt handling*

The polling cycle is repeated with each machine cycle, and the values polled are the values that were present at the previous machine cycle. The polling cycle/LCALL sequence is illustrated in Figure 2.14. The hardware-generated LCALL executes a program jump to fixed program entries as shown below:

| Source | Vector Address |
|---|---|
| IE0 | 0003H |
| TF0 | 000BH |
| IE1 | 0013H |
| TF1 | 001BH |
| RI+TI | 0023H |

Execution proceeds from that location until the RETI instruction is encountered then the interrupted program continues from where it left off.

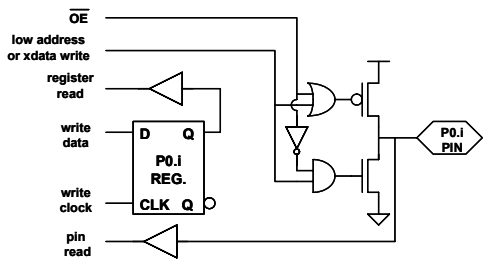
## 2.6. PORT OPERATION

The ports P0, P1, P3, P4 and P5 in the ADD8051C12A are bidirectional, P2 is an output port. Each port consists of a register (Special Function Registers P0 through P5), an output driver, and an input buffer.

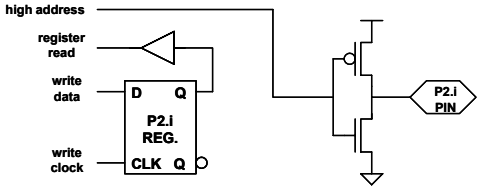
Ports 0 and 2 are used to access the external memories, ADD8051C12A included in ADD1000AQF128 uses external program storage, with this configuration ports P0 and P2 must be used only to access program code and external data. To access memories, Port 0 outputs the low byte of the external memory address, time-multiplexed with the byte being written or read. Port 2 outputs the high byte of the external memory address when the address is 16 bits wide. When a MOVX @Ri instruction is executed the Port 2 pins emit the P2 SFR content to be used as the most significant address byte.

Port 3 pins are multifunctional. They are port pins and also serve as input or output for the microcontroller peripherals:

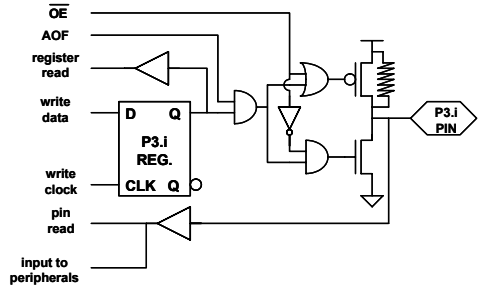
| Port Pin | Alternate Function |
|---|---|
| P3.0 | RxD (serial input port) |
| P3.1 | TxD (serial output port) |
| P3.2 | INT0 (external interrupt) |
| P3.3 | INT1 (external interrupt) |
| P3.4 | T0 (Timer/Counter 0 external input) |
| P3.5 | T1 (Timer/Counter 1 external input) |
| P3.6 | WR (ext. Data Memory write strobe) |
| P3.7 | RD (ext. Data Memory read strobe) |



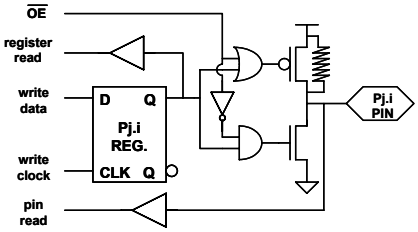

(a) P0 structure




(b) P2 structure




(c) P3 structure




(d) P1, P4 and P5 structure

*Figure 2.15: Port structure*

To use alternate functions the corresponding bit register in the port SFR must contain a 1.

## 2.6.1. I/O CONFIGURATION

Figure 2.15 shows a functional diagram of bit register and I/O buffer in each of the six ports. The level of the port pin is placed on the internal bus and instructions can read the port pin value or the SFR register value.

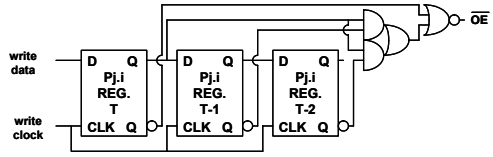
As shown in Figure 2.15, the output drivers of Port 0 and 2 are connected to an internal ADDR and ADDR/DATA bus by an internal CONTROL signal for use in external memory accesses.

Also shown in Figure 2.15 is that if a P3 bit register contains a 1, and then the output level is controlled by the signal labeled “alternate output function”.

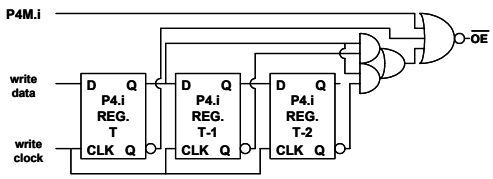
Ports 1, 3, 4 and 5 have internal pull-ups. Each I/O line can be independently used as an input or an output. (Port 0 and 2 may not be used as general purpose I/O, they are used for external memory access). To be used as an input, the port bit register must contain a 1, when an SFR port bit is set to ‘1’ the PMOS port driver is turned on two clock cycles and then is turned off (Figure 2.16a). Then, the pin is pulled high by a weak internal pull-up, and can be pulled low by an external source. In the ADD8051C12A included in ADD1000AQF128, the pull-up consists of 50K resistors.

Because Ports 1, 3, 4 and 5 have fixed internal pull-ups, they are called “pseudo-bidirectional” ports. P4 can be configured as a push-pull output port using P4M SFR, a ‘0’ configures port pin as pseudo-bidirectional and a ‘1’ as push-pull.

P4M has 0s as reset value and after reset P4 is configured as pseudo-bidirectional (Figure 2.16b). All the port registers in the ADD8051C12A have 1s written to them by the reset function, they are configured as input.




(a) P0, P1, P3 and P5 structure




(b) P4 structure

*Figure 2.16: Port OE structure*

## 2.6.2. READ-MODIFY-WRITE FEATURE

Some instructions that read a port read the register and others read the pin. Only the read-modify-write instruction read the register. The instructions that read the register rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the register. When the destination operand is a port, or a port bit, these instructions read the register rather than the pin: ANL, ORL, XRL, JBC, CPL, INC, DEC, DJNZ, MOV PX.Y,C, CLR PX.Y, SET PX.Y. The last three instructions are bit modify instructions. They read the port byte, all 8 bits, modify the target bit, then the modified byte is written to the register.

The-modify-write instructions are directed to the register to avoid a possible misinterpretation of the voltage level at the pin. When a port register is set to ‘1’ to using the pin as input an external source can be pulling the pin to ‘0’, if we need to know the configuration of the port we need to read the register and to know the state of the external source we need to read the pin.

## 2.6.3. ACCESSING EXTERNAL MEMORIES

Accesses to external memory are of two types: accesses to external program code and accesses to external program data. Both share ports P0 and P2 as address and data busses. Accesses to external program code use signal PSEN (program store enable) as the read strobe. Accesses to external program data use RD(P3.7) or WR(P3.6) to strobe the memory.

ADD8051C12A included in ADD1000AQF128 executes program from external memory and P0 and P2 ports can not be used as general purpose I/O.

Fetches from external program memory always use a 16-bit address, fetches to external data can use 16-bit or 8-bit address. Port P0 pins are not connected to P0 register, port P0 is exclusively used to output address (low byte) when program code or external data fetching, and as code byte input or external data byte input/output. Port P2 pins are used to output address (high byte) when program code or external data fetching, P2 register is connected to P2 pin only when a MOVX @Ri instruction is executed.

In any case, the low byte of the address is time-multiplexed with the data byte on Port 0. The ADDR/DATA signals drive both FETs in the Port 0 output buffers. ALE (Address Latch Enable) should be used to capture the address byte into an external register. The address byte is valid at the negative transition of ALE. Then, in a write cycle, the data byte to be written appears on Port 0 just before WR is activated, and remains there until after WR is deactivated. In a read cycle, the incoming byte is accepted at Port 0 just before the read strobe is deactivated.

## 2.7. TIMERS AND WATCHDOG

The ADD8051C12A has two 16-bit Timers/Counters: Timer 0 and Timer 1 that can be configured as timers or event counters (see Figure 2.17 and Figure 2.18). And one 8-bit, with 15-bit prescaler, watchdog timer: Timer 3 (Figure 2.19).

T3 is active only when EW pin is tied to '0'. T3 must be reloaded by software to avoid T3 rollover and program restart. Reload of T3 is allowed only if WLE watchdog load enable (PCON(4)) is set. WLE is reset by hardware 13 machine cycles after been set by user software. Reload of T3 register automatically resets the prescaler.

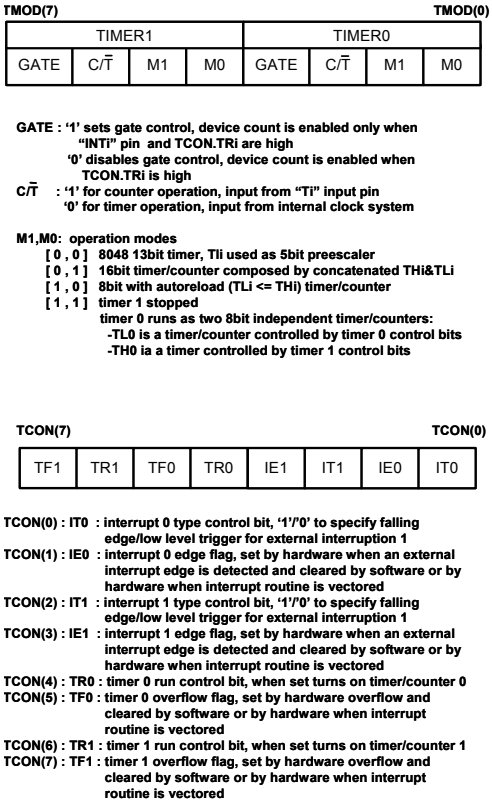



*Figure 2.17: TMOD and TCON registers*

When T0 or T1 are in the timer function, the registers are incremented every machine cycle. They count machine cycles. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T0(P3.4) or T1(P3.5).In counter function, the external input is sampled once every machine cycle. When the samples show a high in one cycle and a low in the next cycle (1-to-0 transition), the count is incremented. The register value is updated during the cycle following the one in which the transition was detected, the maximum count rate is 1/2 of the machine cycle frequency.

The timer or counter function is selected by control bits C/T in the Special Function Register TMOD. T0 and T1 have four operating modes, which are selected by bit-pairs (M1, M0) in TMOD. The operating modes are as follows (see figure 18).

**Mode 0**

A Timer (0 or 1) into Mode 0 looks like a 13 bit timer: an 8-bit counter (all 8 bits of THx) with a divide-by-32 prescaler (the lower 5 bits of TLx), see Figure 2.18. As the count rolls over from all 1s to all 0s, it sets the Timer interrupt flag TFi. The counted input is enabled to the Timer when TRi=1 and either GATE=0 or INT1=1. When GATE=1 the Timer can be controlled by external input INTx, to facilitate pulse width measurements. TRx is a control bit in the Special Function Register TCON (Figure 2.17). GATE is in TMOD.

**Mode 1**

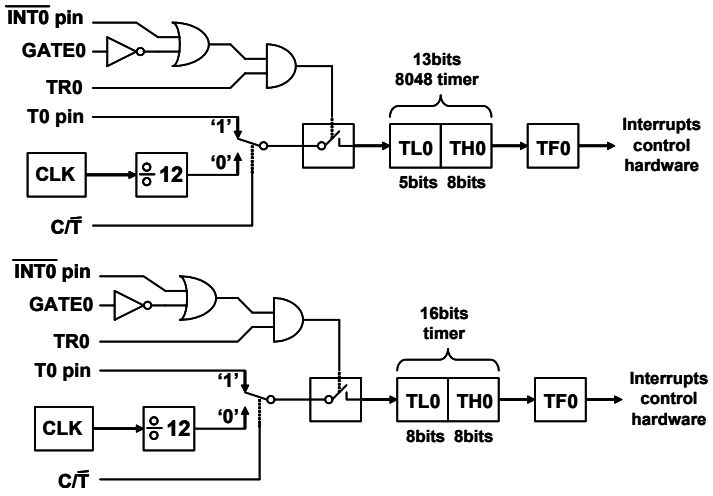
Into Mode 1 the Timers (0 or 1) run in the same way than Mode 0 but with all 16 bits.

**Mode 2**

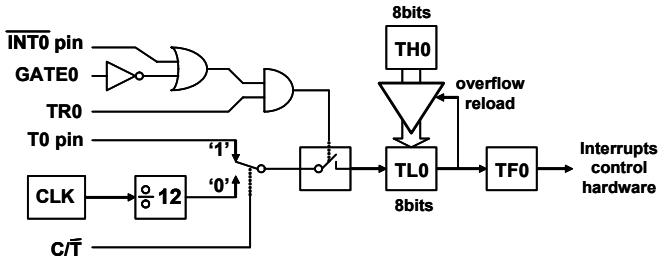
Mode 2 configures the Timer (0 or 1) register as an 8-bit Counter (TLx) with automatic reload of THx value. THx value is preset by software.

**Mode 3**

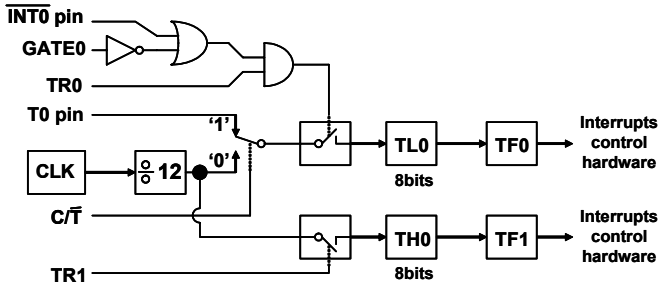
Into Mode 3 Timer 0 registers TL0 and TH0 are configured as two separate 8-bit counters (Figure 10). TL0 uses the Timer 0 control bits: C/T, GATE, TR0, INT0, and TF0. TH0 is locked into a timer function (counting machine cycles) and is controlled with TR1 and TF1 from Timer1.




(a) TIMERS01 in mode 0 and 1




(b) TIMERS01 in mode 2




(c) TIMER0 in mode 3

*Figure 2.18: T0 and T1 modes*

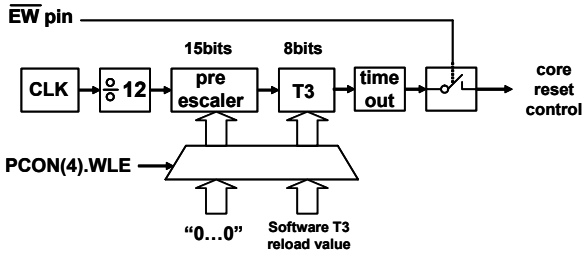



*Figure 2.19: TIMER3 watchdog*

TH0 controls the "Timer 1" interrupt. Timer 1 in Mode 3 holds its count. With Timer 0 in Mode 3, an 80C51 can look like it has three Timer/Counters. When Timer 0 is in Mode 3, Timer 1 can be used by the serial port as a baud rate generator, or in any application not requiring an interrupt.

## 2.8. STANDARD SERIAL INTERFACE

The serial port is full duplex, it can transmit and receive simultaneously. The serial port receive and transmit registers are both accessed at Special Function Register SBUF. Writing to SBUF loads the transmit register and starts transmission, and reading SBUF accesses a physically separate receive register. The reception register is buffered and it can commence reception of a second byte before a previously received byte has been read from the register. When a second byte is fully received the first one is lost.

The serial port control and status register is the Special Function Register SCON, shown in Figure 2.20. This register contains the mode selection bits, the 9th data bit for transmit and receive (TB8 and RB8), and the serial port interrupt bits (TI and RI).

SCON(7) SCON(0)

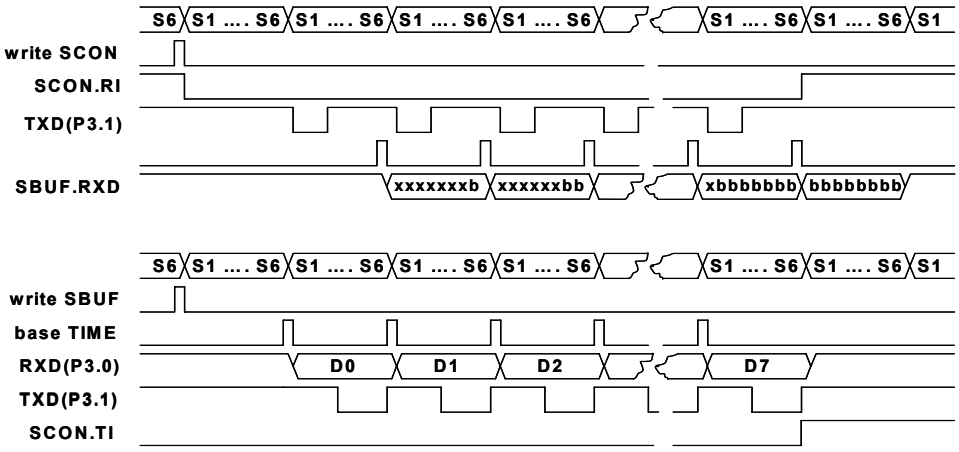
| SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |
|---|---|---|---|---|---|---|---|

SCON(0) : RI : reception interrupt flag, set by hardware at the end of reception, must be cleared by software
SCON(1) : TI : transmission interrupt flag, set by hardware at the end of transmision, must be cleared by software
SCON(2) : RB8 : 9th received bit in modes 2 and 3, received stop bit in mode 1 when SM2='0'
SCON(3) : TB8 : 9th transmitted bit in modes 2 and 3
SCON(4) : REN : serial port reception enable
SCON(6:5) : SM1,SM0 : operation modes

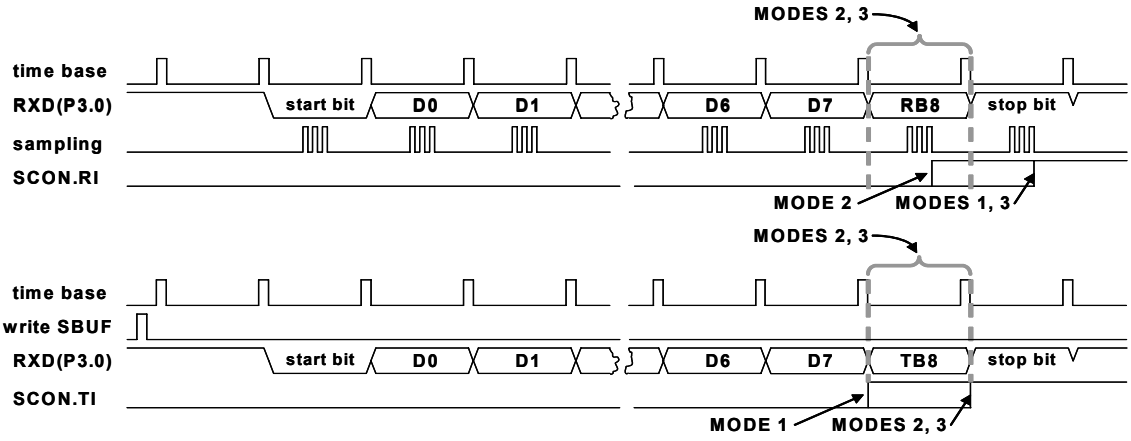
| [SM1,SM0] | MODE | baudrate |
|---|---|---|
| [ 0 , 0 ] | 0 | shift register | 1/mc |
| [ 0 , 1 ] | 1 | 8-bit UART | T1 overflow |
| [ 1 , 0 ] | 2 | 9-bit UART | 1/2mc or 1/4mc |
| [ 1 , 1 ] | 3 | 9-bit UART | T1 overflow |

SCON(7) : SM2 : reception control; in modes 2 and 3 with SM2='1' RI will be '0' if the 9th received bit is '0'; in mode 1 with SM2='1' RI will be '0' if the received stop bit is not correct; in mode 0 SM2 keeps a '0'.

*Figure 2.20: SCON register*




(a) Serial Port reception/emission mode 0




(b) Serial Port reception/emission modes 1,2,3

*Figure 2.21: Serial Port Waveforms*

The serial port can operate in 4 modes. Transmission is always initiated by any instruction that writes SBUF, and TI flag is set after transmission. Reception is initiated in Mode 0 by the condition RI = 0 and REN = 1, in the other modes by the incoming start bit if REN = 1. In Modes 2 and 3 9 data bits are received. The 9th one goes into RB8 and then comes a stop bit.

The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8 = 1. This feature is enabled by setting bit SM2 in SCON.

**Mode 0**: shift register mode, serial data enters and exits through RxD. TxD outputs the shift clock. 8 bits are transmitted/received (LSB first). The baud rate is fixed at 1/12 the oscillator frequency. The internal timing is such that one full machine cycle will elapse between "write to SBUF" and activation of transmission, transmission ends in the 10th machine cycle after "write to SBUF". Reception is initiated by the condition REN = 1 and RI = 0, after 10 machine cycles reception finish and RI is set.

**Mode 1**: 10 bits are transmitted (through TxD) or received (through RxD): a start bit (0), 8 data bits (LSB first), and a stop bit (1). On receive, the stop bit goes into RB8 in SFR SCON. The baud rate is controlled by T1 rollover. Transmission is initiated by any instruction that uses SBUF as a destination register and the bit times are synchronized to T1 rollover. Reception is initiated by a detected 1-to-0 transition at RxD, RxD is sampled at a rate of 16 times each bit time and the value accepted is the value that was seen in at least 2 of the 3 samples (7th, 8th, and 9th). A correct reception loads SBUF and RB8 and sets RI, this is achieved when at the end of the receive process the condition (RI='0' and (SM2='0' or received_stop_bit='1') is met, if the condition is not met the received frame is irretrievably lost.

**Modes 2 and 3**: 11 bits are transmitted (through TxD) or received (through RxD): start bit (0), 8 data bits (LSB first), a programmable 9th data bit (TB8), and a stop bit (1). The 9th transmitted data bit is loaded with TB8 value from SFR SCON. On receive, the 9th data bit goes into RB8 in SFR SCON. The stop bit is ignored. The baud rate is programmable to 1/32 or 1/64 the oscillator frequency. Mode 3 is the same as Mode 2 with programmable baud-rate controlled by T1 rollover.

The transmission begins with a "write to SUBF" instruction and finish at the 11th rollover after "write to SUBF", TI is set when transmission ends.

Reception is initiated by a detected 1-to-0 transition at RxD. For this purpose RxD is sampled at a rate of 16 times each bit. The value accepted is the value that was seen in at least 2 of the 3 samples. When reception ends SBUF and RB8 are loaded end RI is set. A frame is correctly received if, and only if, the following condition is met at the time the last bit is received (RI=0 and (SM2=0 or 9th_data_bit=1).

*If the condition is not met the received frame is irretrievably lost and RI is not set.*

**Baud Rates**

The baud rate in Mode 0 is fixed to Oscillator Frequency / 12.

The baud rate in Mode 2 depends on the value of bit SMOD in Special Function Register PCON. If SMOD = 0 (which is the value on reset), the baud rate is 1/64 the oscillator frequency If SMOD = 1, the baud rate is 1/32 the oscillator frequency.

$$\text{Baud Rate} = \frac{2^{SMOD}}{64} \times (OscillatorFrequency)$$

The baud rates in Modes 1 and 3 are determined by the Timer 1 overflow rate and the value of SMOD as follows:

$$\text{Baud Rate} = \frac{2^{SMOD}}{32} \times (Timer1OverflowRate)$$

In the most typical applications, it is configured for "timer" operation in the auto-reload mode. In that case the baud rate is given by the formula:

$$\text{Baud Rate} = \frac{2^{SMOD}}{32} \times \frac{OscillatorFrequency}{12 \times [256 - (TH1)]}$$

Figure 2.22 lists various commonly used baud rates and how they can be obtained using Timer 1.

| BAUD RATE | FOSC (MHz) | BAUD2 | SMOD | TIMER 1 | | |
|---|---|---|---|---|---|---|
| | | | | C/T | MODE | reload value |
| 38.4K | 11.0592 | 1 | 1 | 0 | 2 | H'FD |
| 19.2K | 11.0592 | 0 | 1 | 0 | 2 | H'FD |
| 9.6K | 11.0592 | 0 | 0 | 0 | 2 | H'FD |
| 4.8K | 11.0592 | 0 | 0 | 0 | 2 | H'FA |
| 2.4K | 11.0592 | 0 | 0 | 0 | 2 | H'F4 |
| 1.2K | 11.0592 | 0 | 0 | 0 | 2 | H'E8 |

*Figure 2.22: Baud Rates configuration examples*

# CHAPTER 3. MEDIUM ACCESS CONTROLLER

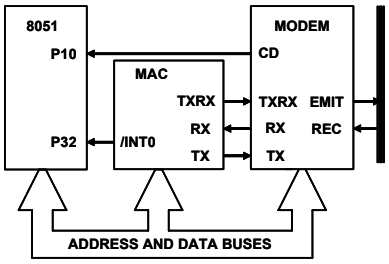



*Figure 3.1: System connection*

## 3.1. GENERAL DESCRIPTION

Common power line communication (PLC) systems are provided with an eight bits microcontroller and a modem. Medium Access Controller (MAC) software tasks spend a high percent of CPU time, so the ADD1000A has some of this tasks developed via hardware to reduce the CPU computational load of the 8051 integrated microcontroller.

MAC functional capabilities involve the construction of message packets, adding FEC (Forward Error Correcting Code) values to bytes and FCS (Frame Check Sequence) to packets.

The ADD1000A MAC is compatible with EHS and KONNEX. However, the design is very versatile and allows users to create a wide range of datagram structures with the only constrain of FEC and FCS hardware codes.

A general packet encapsulation has the following structure:

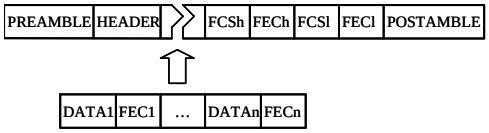



*Figure 3.2: Datagram Packet*

**PREAMBLE:** a 2 bytes field to allow bit synchronization.

**HEADER***:* a 2 bytes field to identify the datagram type.

**FEC:** a 6 bits field to protect one byte of data against burst noise.

**FCS***:* a 2 bytes field to protect all the datagram.

**POSTAMBLE:** a 2 bits field to allow full reception of significant data.

Hardware implemented FEC is capable of correcting a 3 bits burst in a block of 14. Polynomial used is:

$$G(x) = x^6 + x^5 + x^4 + x^3 + 1$$

Some errors, wider than the 3 bit burst, are wrongly corrected. These errors are detected by the Frame Check Sequence. The polynomial used to compute FCS is:

$$G(x) = x^{16} + x^{15} + x^2 + 1$$

## 3.2. MAC ARCHITECTURE

The microcontroller interfaces with the MAC addressing it as external data RAM. There are six configuration registers in the MAC to interact with the microcontroller, mapped from FE00'h to FE05'h addresses.

The control logic reads the configuration from the configuration registers and controls the MAC operation to perform the desired tasks. The emitter-receiver MAC logic sends or receives serial data through the modem (MSb first) with the selected baud rate.

The MAC is capable to store up to three bytes in a stack when is configured in transmission mode.

In reception mode the MAC circuitry uses the external interrupt 0 located at P32, and activates it each time that a new data byte is received.

## 3.3. CONFIGURATION REGISTERS

The following table shows the configuration registers address map.

| ADDRESS | REGISTER NAME |
|---|---|
| FE00 | CONTROL |
| FE01 | FLAGS1 |
| FE02 | FLAGS2 |
| FE03 | DATA |
| FE04 | BAUD_RATE high |
| FE05 | BAUD_RATE low |

*Table 3.1: Configuration Registers*

### 3.3.1. CONTROL register

This register holds mode operation bits that allow to the microcontroller to control the MAC behavior and some special purpose bits. The microcontroller has full access to this register.

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| A1 | A0 | BYTE | WHD | FCS | FEC | TXRX | STOP |

*Figure 3.3: CONTROL register*

**A1** and **A0** bits force the number of four bits packets of the preamble field that must precede the header field to consider an incoming data sequence as a valid start of packet. See also ZERO_A bit description.

| PREAMBLE | A1 | A0 |
|---|---|---|
| AAAA | 1 | 1 |
| AAA | 1 | 0 |
| AA | 0 | 1 |
| A | 0 | 0 |

*Table 3.2: Preamble configuration*

An increase of this constrain reduce the number of spurious start of packets received by the MAC circuitry, however the preamble field is not protected against burst noise and a bit error in this field causes the packet loss.

**BYTE:** When this bit is set to 1, byte mode is on. In byte mode, bytes are sent or received without FEC protection field.

**WHD:** This bit enables the WRITE HEADERS mode. In this mode the microcontroller configures the desired values of the four programmable headers. The TXRX bit must be cleared because the data flow goes from the microcontroller to the MAC.

**FCS:** When this bit is set to 1, FCS mode is on. In this mode the FCS field computation is assumed by the MAC and added at the end of the packet. If the microcontroller computes the FCS field then FCS bit value must be equal to zero.

**FEC:** In FEC mode (FEC bit set to 1) the FEC field computation is assumed by the MAC and added at the end of each byte. If the microcontroller computes the FEC field then FEC bit value must be equal to zero.

**TXRX:** When TXRX bit is on, the system receives data from the power line, otherwise sends data to the power line.

**STOP:** A MAC stop cycle resets the MAC logic circuitry and aborts the current reception or transmission process. A MAC stop cycle consist in generate a high active pulse in the STOP bit value. The microcontroller must generate a MAC stop cycle each time a packet reception or transmission is finished or aborting. A MAC stop cycle doesn't reset the configuration registers.

## 3.3.2. FLAGS1 register

This register holds a mode operation bit, some special purpose bits, and the RX and TX flags to control the transmission and reception processes. The microcontroller has full access to this register. Figure 3.4 shows the format of the FLAGS1 register.

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| BYP | NPST | CN_OK | ZERO_A | NFCS | RxFEC | RX | TX |

*Figure 3.4: FLAGS1 register*

**BYP:** This bit enables bypass mode operation. In this mode the MAC logic is disabled and the microcontroller takes control of the power line communication modem. See Bypass mode section for more details.

**NPST:** This bit must be set to 1 when POSTAMBLE is the next field to send in a transmission operation and FCS mode is off (In FCS mode the POSTAMBLE is appended without request). The MAC logic appends the POSTAMBLE field to the packet automatically. The POSTAMBLE is a 2 bits long field, where each bit is the complement of the last bit of the message.

**CN_OK:** The microcontroller must set to 1 this bit when the baud rate was configured at the BAUD_RATE high and BAUD_RATE low registers. If CN_OK bit is equal to zero the MAC is disabled.

**ZERO_A:** When this bit is set to 1 only the header field of the packet is verified by the MAC to accept an incoming data packet. This bit overrides the configuration selected with A1 and A0 bits of the CONTROL register.

**NFCS:** This bit must be set to 1 when FCS is the next field to send in a transmission operation or to receive in a reception operation. The MAC logic appends the FCS field to the packet automatically in a transmission operation and check the packet for any error in a reception operation.

**RxFEC:** If this option is on (RxFEC bit set to 1) the MAC logic also pass FEC fields to the microcontroller via DATA register, otherwise only data bytes are passed.

**RX:** Receive flag. Set by MAC at the end of a byte reception in byte mode operation or at the end of a byte+FEC reception in the other reception modes. The RX flag must be cleared by the microcontroller when the byte value has been read from the DATA register. When the MAC sets the RX value, an external interrupt (INT0) alerts to the microcontroller.

**TX:** Transmission flag. Set by microcontroller when a new byte is ready to be transmitted. TX flag is cleared by the MAC when a new byte is needed. The microcontroller must write a new byte in the DATA register when the TX flag is cleared.

## 3.3.3. FLAGS2 register

In this register are two bits for configuration purposes HD_SH and EXT_M, and five information bits controlled by the MAC circuitry. Information bits are protected against writing.

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| X | HD_SH | EXT_M | END_TX | HD1 | HD0 | EFCS | EFEC |

Note: x ≡ don't care

*Figure 3.5: FLAGS2 register*

Due to internal purposes, when the microcontroller reads anyone of the three more significative bits the obtained value is always '1' although the bit has been cleared previously. Figure 3.5 shows FLAGS2 register.

**HD_SH:** This bit is active high and allows receiving shielded header packets. A shielded header is a header field protected with FEC fields. Shielded headers are not compatible with EHS-KONNEX.

**EXT_M:** If this bit is set to 1 the modem must generate the sampler signal, otherwise is MAC generated.

**END_TXD:** This flag is set to 1 by the MAC when the current transmission finishes. A stop MAC cycle resets this value.

**HD1,HD0 :** When the system is in reception mode, the MAC logic circuitry looks for a programmed header value in the incoming data sequence, when a valid header value is detected the MAC pass to the microcontroller the next incoming byte and writes **HD1** and **HD0** bit values to identify the received header. **HD1** and **HD0** bit values are function of the received header value.

| HEADER | HD1 | HD0 |
|---|---|---|
| HEADER1 | 0 | 0 |
| HEADER2 | 0 | 1 |
| HEADER3 | 1 | 0 |
| HEADER4 | 1 | 1 |

*Table 3.3 Header configuration*

**EFCS:** This flag is set to 1 by the MAC when a FCS error is detected in the incoming packet. A stop MAC cycle resets its value.

**EFEC:** This flag is set to 1 by the MAC when a FEC error is detected in the incoming byte. A stop MAC cycle resets his value.

## 3.3.4. DATA register:

The DATA register holds the current byte value to be sent in a transmission operation or the last received byte in a reception operation, this includes data bytes and FEC fields. The DATA register is also used to program the header values (See WRITE HEADERS mode section for more details). The microcontroller has full access to this register.

Figure 3.6 shows format of DATA register for data bytes and FEC fields.

If the MAC is configured in FEC mode, automatically computes FEC fields and conducts the necessary operations. In this mode, if the microcontroller wants to know the received FEC value (RxFEC asserted) the MAC circuitry uses the non-inverted format (Figure 3.6b)).

If the FEC mode is deselected the MAC circuitry doesn't compute FEC fields. In order to send FEC fields in this mode, the microcontroller must pass FEC values to the MAC in the inverted format (Figure 3.6c)), this is due to the fact that FEC fields are sent to the power line inverted. In reception mode the MAC also uses the inverted format (Figure 3.6c)).

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

(a) data byte

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| 0 | 0 | FEC5 | FEC4 | FEC3 | FEC2 | FEC1 | FEC0 |

(b) FEC in FEC mode

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| 0 | 0 | /FEC5 | /FEC4 | /FEC3 | /FEC2 | /FEC1 | /FEC0 |

(c) FEC in not FEC mode

*Figure 3.6: DATA register*

## 3.3.5. BAUD_RATE registers

With CN_OK bit reset, the microcontroller can configure the baud rate of the system writing in the BAUD_RATE high and BAUD_RATE low registers the value of the baud rate parameter. The microcontroller has full access to these registers.

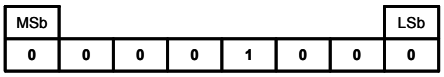
Expression for baud rate parameter:

$$brp = \frac{fclk}{2 * br} - 1$$

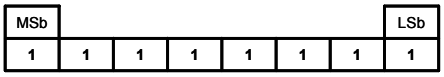
*brp ≡ Baud rate parameter.*

*fclk ≡ Frequency clock [Hz].*

*br ≡ Desired baud rate [bits/s].*




(a) BAUD RATE high




(b) BAUD RATE low

*Figure 3.7: BAUD RATE registers*

For example, to configure a 2400 bauds speed with an 11.0592 MHz clock the value of the baud rate parameter is:

$$brp = \frac{11059200}{2*2400} - 1 = 2303$$

So the microcontroller must write in the BAUD_RATE registers 2303 (08FF'h), as it shows Figure 3.7. After that, the microcontroller must set CN_OK bit at FLAGS1 register.

### 3.3.6. Reset values

The default values after reset of the MAC configuration registers are shown in Table 3.4.

| REGISTER NAME | RESET VALUE |
|---|---|
| CONTROL | 4E |
| FLAGS1 | 10 |
| FLAGS2 | 40 |
| DATA | 00 |
| BAUD_RATE high | 08 |
| BAUD_RATE low | FF |

*Table 3.4: Reset values*

With this default values the MAC configuration after reset is:

- Reception mode.
- 2400 bauds with an 11.0592 MHz frequency clock.
- FEC and FCS modes on.
- The MAC only checks the header field to validate a received packet.
- Modem sampler signal selected.
- The header isn't protected with FEC fields.
- CN_OK bit is cleared so the MAC circuitry is disabled.

The default values of the programmable headers are shown in Table 3.5.

| HD(1:0) | HEADER |
|---|---|
| 00 | 9B58 |
| 01 | E958 |
| 10 | 24E7 |
| 11 | A5D8 |

*Table 3.5: Reset header values*

## 3.4. MODES OF OPERATION

The ADD1000A MAC supports six modes of operation: BYPASS, BYTE, WRITE HEADERS, FEC, FCS and NULL mode.

The following table shows the selected operation mode in function of which configuration bits are asserted.

| (FEC,FCS) \ (BYP,WHD,BYTE) | 000 | 001 | 01X | 1XX |
|---|---|---|---|---|
| 00 | NULL | BYTE | WHD | BYPASS |
| 01 | FCS | BYTE | WHD | BYPASS |
| 11 | FEC + FCS | BYTE | WHD | BYPASS |
| 10 | FEC | BYTE | WHD | BYPASS |

*Table 3.6: Operation modes*

### 3.4.1. BYPASS mode

In BYPASS mode, the microcontroller takes control of the ADD1000A modem. The MAC circuitry is disabled and the MAC layer must be full implemented by software.

The following Table 3.7 shows the connections between the microcontroller and the modem in bypass mode.

| MICRO8051 | MAC | |
|---|---|---|
| | bypass | running |
| P3.2 | Rx | Rx interrupt |
| P1.0 | CD | |
| P1.1 | Tx | |
| P1.2 | Tx/Rx | |

*Table 3.7: Bypass connection*

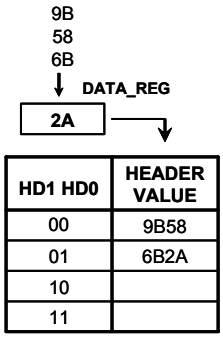
### 3.4.2. WRITE HEADERS mode

This mode is selected with the purpose to set the values of the four programmable headers. Up to four headers can be programmed, the MAC logic detects the number of headers programmed and use them to search valid incoming packets in receive mode.

In order to program the headers, the TXRX bit of the CONTROL register must be cleared, however when WRITE HEADERS mode is on the MAC circuitry is in reception mode.

The MAC has an eight bytes long stack to store the four header values.

The less significative byte of the last header is sent from the microcontroller to the MAC first. The MAC circuitry pushes the header values stored in the stack with the new incoming byte stored at DATA register as the following figure shows.




| HD1 HD0 | HEADER VALUE |
|---|---|
| 00 | 9B58 |
| 01 | 6B2A |
| 10 | |
| 11 | |

*Figure 3.8: Header stack*

The position of the headers in the stack determines the HD1 and HD0 bit values which are used by the MAC to indicate the header value of the received message.

### 3.4.3. BYTE mode

When BYTE mode is on, the MAC logic sends and receives data from the modem in byte format. After a data byte is sent to the microcontroller the next incoming data from the modem is assumed by the MAC as a new data byte. This mode should be used by the microcontroller to send PREAMBLE and HEADER fields of an EHS-KONNEX compatible packet.

### 3.4.4. FEC and FCS modes

In FEC mode (FEC bit asserted), the MAC after a data byte appends (TX) or checks (RX) his FEC. If FEC bit is cleared and the data bytes must be protected by FEC fields, the microcontroller must compute the FEC fields and send them to the MAC through the DATA register (See DATA register section for more details).

In FCS mode (FCS bit asserted) the MAC appends (TX) or checks (RX) the FCS of the packet. If FCS bit is cleared and the packet format includes a FCS field, the microcontroller must compute the FCS and send it to the MAC as a normal data field.

With FEC and FCS bits set, both modes are selected.

### 3.4.5. NULL mode

NULL mode is selected when no other mode is selected. This mode must be used only when FEC and FCS fields are computed by the microcontroller.

The difference between NULL mode and BYTE mode is the use of FEC fields. Figure 9 explains this fact with more detail.

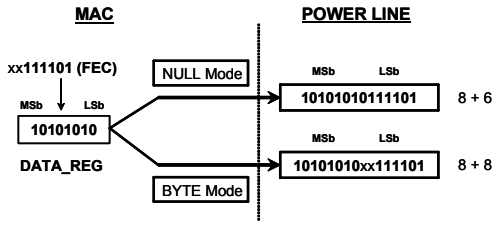



*Figure 3.9: NULL and BYTE modes, differences*

## 3.5. OPERATION PROCEDURES

This section describes the operation procedures for the microcontroller in order to interact with the MAC hardware.

The MAC initialization and MAC stop cycle procedures are common steps that must be used in all modes. The rest of procedures are dependent of the selected mode.

### 3.5.1. MAC initialization

After a system reset, a MAC initialization procedure must be executed by the microcontroller. The first steps are required to change default values, only the last step is obligatory.

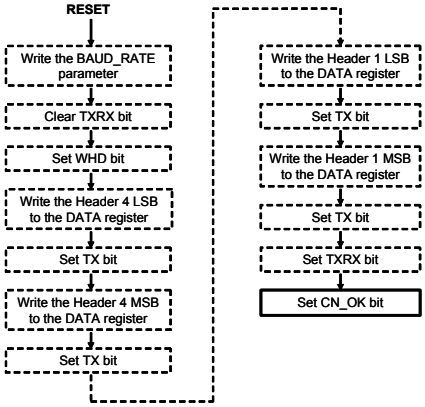



*Figure 3.10: MAC initialization*

### 3.5.2. MAC STOP cycle

A MAC stop cycle resets the MAC logic circuitry and aborts the current reception or transmission process. The microcontroller must generate a MAC stop cycle each time a packet reception or transmission is finished.

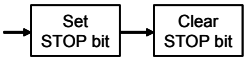



*Figure 3.11: MAC stop*

## 3.5.3. BYTE mode operation procedures

### 3.5.3.a. Transmission procedure

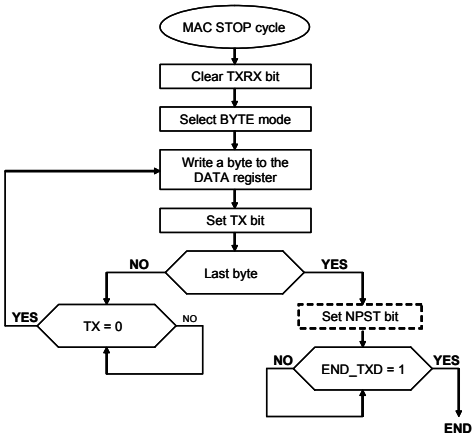



*Figure 3.12: Transmission procedure*

After a MAC stop cycle, the microcontroller must configure the MAC circuitry in transmission mode (clear TXRX bit) and byte mode (assert BYTE mode).

In order to send a byte the microcontroller must check that the TX flag is cleared, write the byte value to the DATA register and set the TX flag.

The END_TXD bit is set by the MAC logic when the packet transmission ends.

The dotted line marked state is required only if the packet has POSTAMBLE field.

### 3.5.3.b. Reception procedure

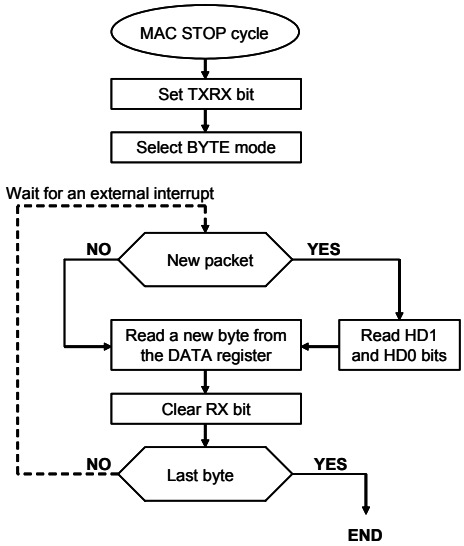



*Figure 3.13: BYTE mode reception procedure*

## 3.6. FEC and FCS mode operation procedures

### 3.6.1.a. Transmission procedure

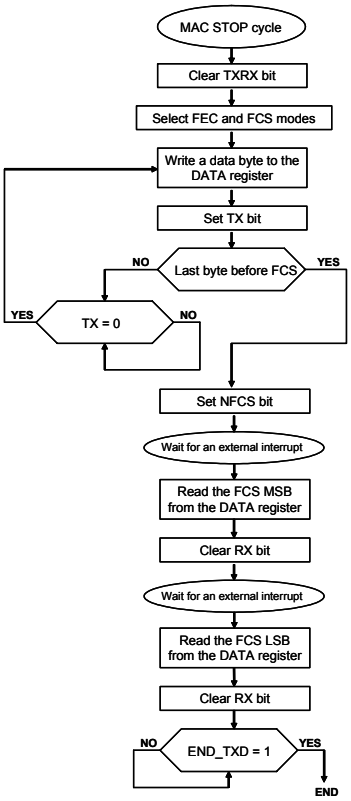



*Figure 3.14: FEC/FCS mode transmission proc.*

The microcontroller needs to know the transmitted FCS value to validate ACK packets, so in this mode the MAC generates external interrupts to pass the FCS field.

#### 3.6.1.b. Reception procedure

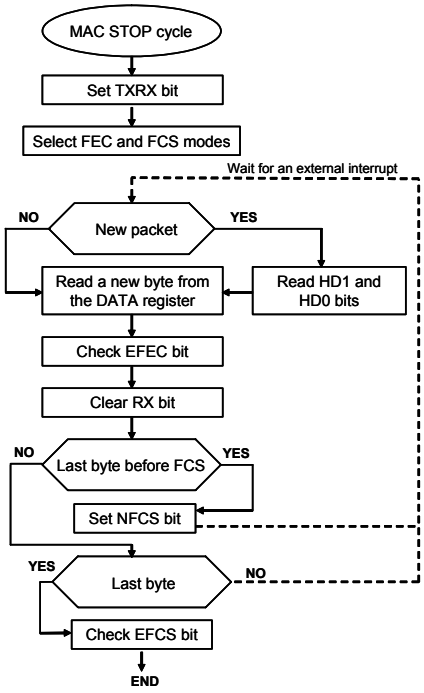



*Figure 3.15: FEC/FCS mode reception procedure*

If RxFEC bit is set, the MAC also generates an external interrupt when a FEC field is received.

## 3.6.2. NULL mode operation procedures

#### 3.6.2.a. Transmission procedure

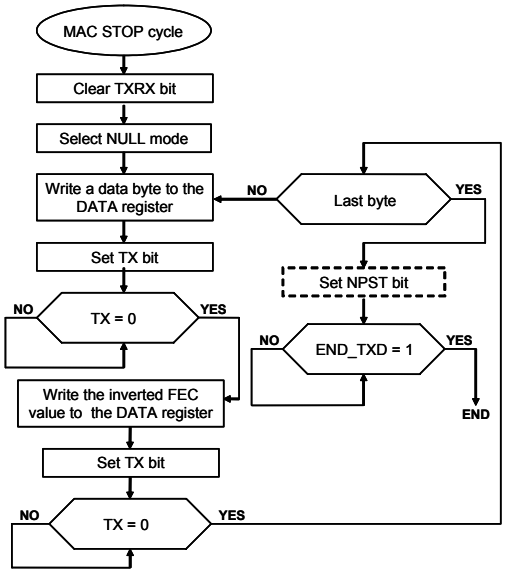



*Figure 3.16: NULL mode transmission procedure*

The dotted line marked state is required only if the packet has POSTAMBLE field.

#### 3.6.2.b. Reception procedure

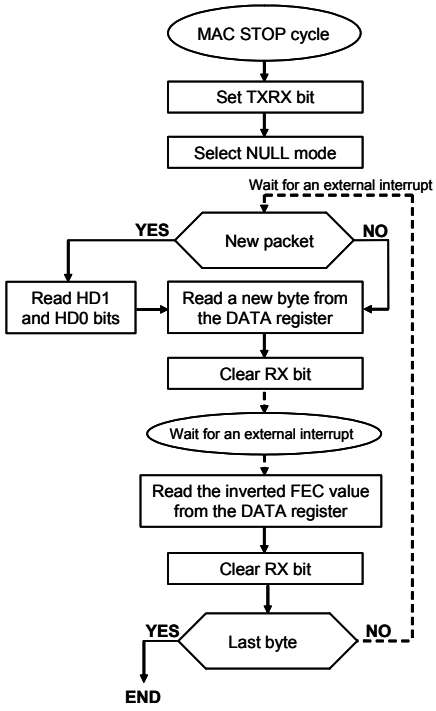



*Figure 3.17: NULL mode reception procedure*

# CHAPTER 4. PLC MODEM

## 4.1. INTRODUCTION

For the power line medium EHS/KNX defines a Minimum Shift Keying modulation (MSK or FSK with low frequency deviation), see Table 4.1.

| | |
|---|---|
| **CENTRAL FREQUENCY** | 132.5 kHz ± 0.2% (0.25 kHz) |
| **DEVIATION** | ± 0.6 kHz |
| **UPPER FREQUENCY** | 133.1 kHz ± 0.2% |
| **LOWER FREQUENCY** | 131.9 kHz ± 0.2% |

*Table 4.1: FSK modulation characteristics*

The bit period is 416.67 μs (2400 baud, 1200 baud is also supported). The lowest frequency codes a logic '1' and the highest frequency codes a logic '0'.

The PLC modem can be controlled using the hardwired MAC or it can be controlled by the microcontroller. Even when the microcontroller takes control of the modem, the MAC is still used (in bypass mode).

The modem has a set of configuration registers, these registers (see Table 4.2) are mapped in the external data space of the microcontroller. They are connected to ports P0 and P2, and to the write and read control signals P3.6 and P3.7. Many bits in these registers are not user definable, and the user must not write to these bits.

| REG. NAME | ADDRESS | RESET |
|---|---|---|
| CONTROL | FE10 | 00 |
| TRESHOLD_2 | FE11 | 00 |
| TRESHOLD_1 | FE12 | 00 |
| TRESHOLD_0 | FE13 | 00 |
| GAIN | FE14 | 0F |
| CONTROL2 | FE15 | 00 |
| C_TABLE_0 | FE20 | XX |
| C_TABLE_... | FE21:FE3A | XX |
| C_TABLE_27 | FE3B | XX |
| VAL_TABLE_0 | FE40 | XX |
| VAL_TABLE_... | FE41:FE5C | XX |
| VAL_TABLE_29 | FE5D | XX |

*Table 4.2: MODEM configuration registers*

In order to have direct access from the microcontroller to the modem, its control signals (TX,RX,TX/RX,CD) are connected to microcontroller ports (see Table 4.3). The port P3.2 has a double function. When the MAC is in full operation mode P3.2 is used to interrupt the microcontroller when a new byte, or a new byte plus fec, is received.

When the MAC is in bypass mode the reception output of the modem, RX, is connected to P3.2.

| MICRO8051 | MAC | |
|---|---|---|
| | **bypass** | **running** |
| P3.2 | Rx | Rx interrupt |
| P1.0 | CD | |
| P1.1 | Tx | |
| P1.2 | Tx/Rx | |

*Table 4.3: MAC connection*

When the MAC is used in full operation mode all the modem control signals (TX,RX,TX/RX,CD) are connected only to the MAC.

## 4.2. CONFIGURATION REGISTERS

### 4.2.1. GAIN register

The GAIN register controls the amplitude of the emitted PLC signal.

Aplc = (Vcc/2)*(GAIN(3:0)/15)

After reset GAIN register holds "00001111", then the system emits with maximum amplitude.

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| X | X | X | X | G3 | G2 | G1 | G0 |

*Figure 4.1: GAIN registers*

### 4.2.2. CONTROL2 register

The bit SQA selects the type of the emitted signal:

SQA = '1' => square waveform

SQA = '0' => sine waveform

The sine waveform is specified with the C_TABLE and the VAL_TABLE values.

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| X | X | X | X | X | X | X | SQA |

*Figure 4.2: CONTROL2 registers*

### 4.2.3. CONTROL and TRESHOLD registers

The CONTROL and TRESHOLD registers are to fix the carrier detect level. To fix the CD level the TRESHOLD register must to be written with the level value (see Table 4), and then WR_TRES bit in CONTROL register must be set to '1'. The RD_TRES bit, when set to '1', loads the TRESHOLD registers with the carrier detect level.

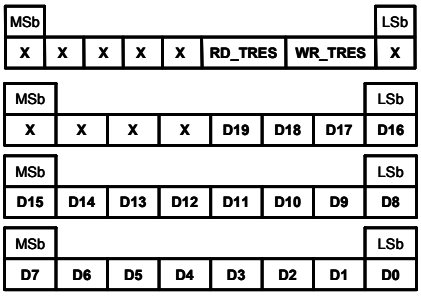



*Figure 4.3: CONTROL and TRESHOLD 2, 1 and 0 registers*

| CD (dBuV) | TRESH (Hex) |
|---|---|
| 90 | 01D000 |
| 85 | 00F000 |
| 80 | 00B000 |
| 75 | 004F00 |
| 70 | 002700 |
| 65 | 001D00 |
| 60 | 000C00 |
| 55 | 000700 |
| 50 | 000350 |

*Table 4.4: TRESHOLD(2,1,0) level values*

### 4.2.4. C_TABLE and VAL_TABLE registers

C_TABLE and VAL_TABLE contain values that define the waveform shape.

These values must be calculated for a specific coupling circuit. When a set of values designed for a coupling circuit are used with other different coupling circuit this could result in a CENELEC non compliant signal.

The default reset values of C_TABLE and VAL_TABLE are calculated for a passive coupling circuit.

## 4.3. EMISSION AND RECEPTION CHARACTERISTICS

The emitted signal is generated with a resistor array (see Figure 4.5 and Figure 4.6). The value of the resistors determines the intensity that will be injected in the power line, and then the signal level (see Figure 4.6).

The receiving characteristics, input sensitivity and input impedance, are greatly influenced by the receiving circuitry. Figure 4.7 shows an input interface designed for high input impedance; for lower input impedances a simplified input stage, without operational amplifiers, could be used.

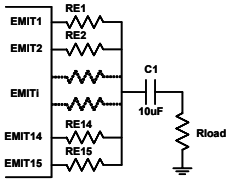



*Figure 4.4: simplified emission circuit*

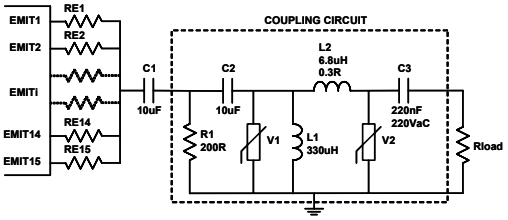



*Figure 4.5: emission circuit*

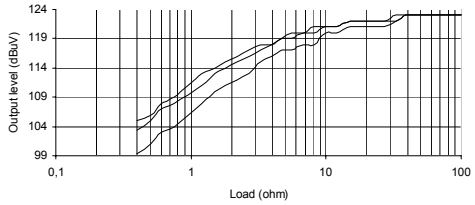



*Figure 4.6: maximum emission level vs load (RE=68R,34R,22R)*

| Symbol | Parameter | Test Condition | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| Iout | Output transmitting current | RE=68 Ω ,Load = 1Ω (Transmitting circuit as in fig. 5/6) | | 225 | | |
| Iout | Output transmitting current | RE=34 Ω ,Load = 1Ω (Transmitting circuit as in fig. 5/6) | | 350 | | |
| Iout | Output transmitting current | RE=22 Ω ,Load = 1Ω (Transmitting circuit as in fig. 5/6) | | 450 | | mArms |
| Vin | Input Sensitivity | Coupling circuit as in fig. 8 | | 100 | | µVrms |
| Rin | Input Impedance | (f=132.5kHz) Coupling circuit as in fig. 8 | 150 | 180 | 200 | Ω |
| Vcd | Carrier detect level | Software programmable | 50 | | 90 | µVrms |

*Table 4.5: TX/RX characteristics*

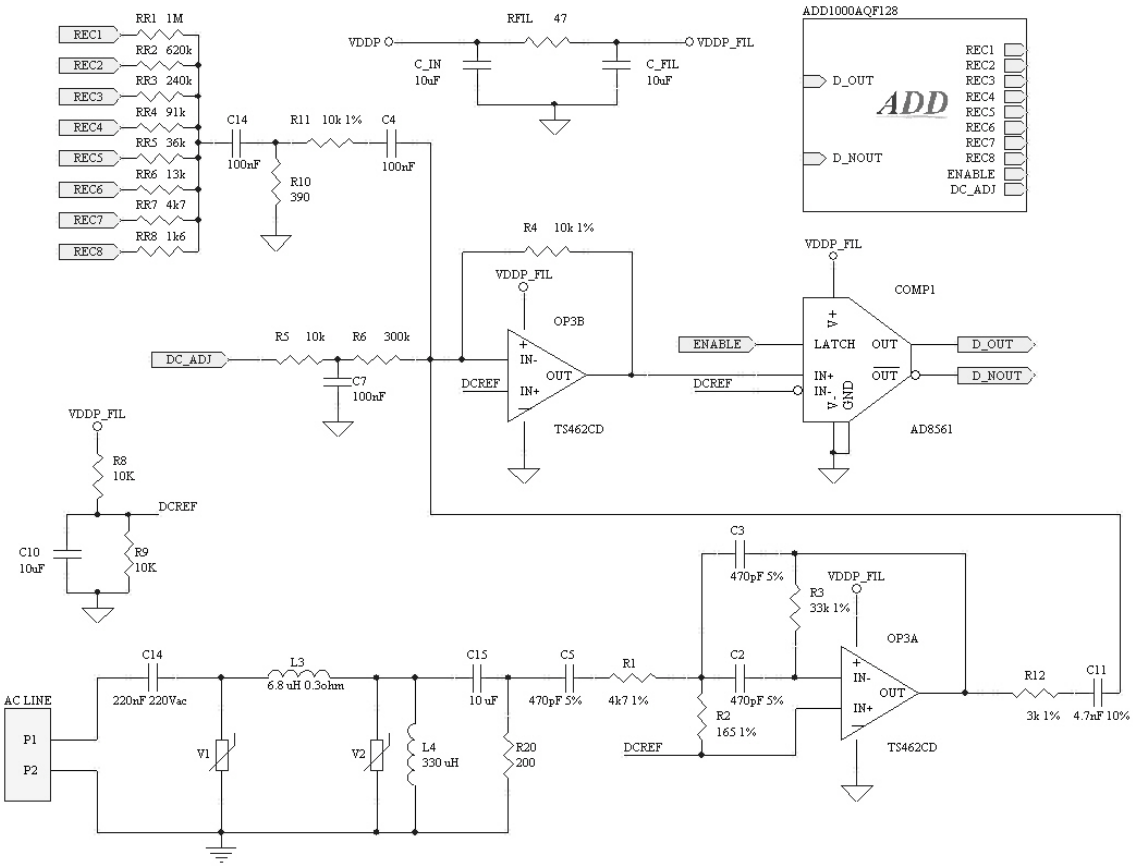



*Figure 4.7: reception circuit*

# CHAPTER 5. FLASH LOADER

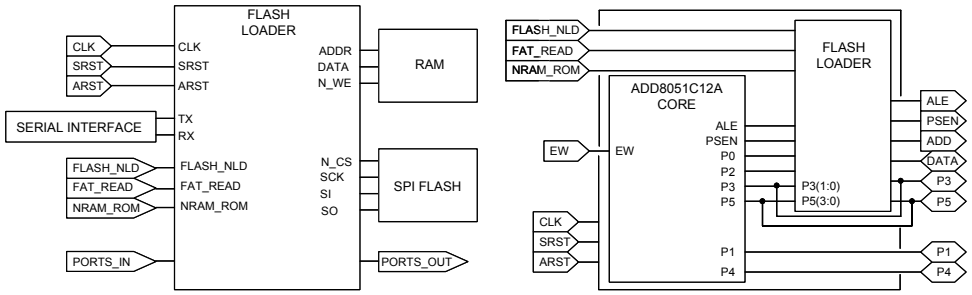



*Figure 5.1: flash loader IP pinout and connection with the microcontroller*

## 5.1. INTRODUCTION

During runtime, a traditional 8051 romless based system uses an external ROM memory to store the program and a RAM memory for volatile storage of data (Figure 5.2).

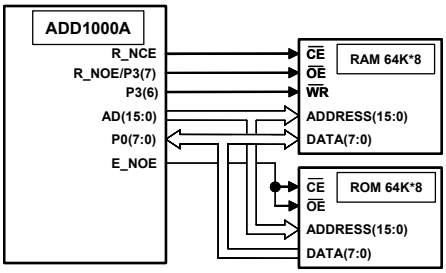



*Figure 5.2: ROM+SRAM schema*

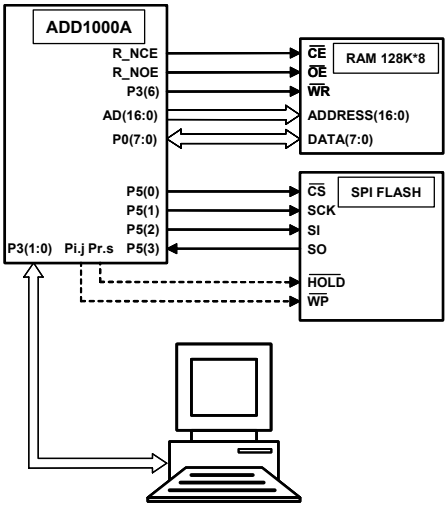



*Figure 5.3: serial flash storage schema*

When a user wants to upgrade the firmware, he must replace or remove the ROM memory from the system in order to reprogram it using an external programmer.

The ADD1000A supports this memory configuration and a more versatile one, in which, the program is permanently stored in a serial flash memory and during runtime a SRAM memory holds the program and the temporary data (Figure 5.3). The flash loader circuitry allows in-system programming via RS-232 serial interface without additional logic, that is, a user can simply update the firmware reprogramming the flash memory from a PC without disassembling the system.

The embedded ADD8051C12A microcontroller executes the program from the RAM memory, so at startup the flash loader transfers the program from the flash memory to the RAM memory.

## 5.2. PIN DESCRIPTION

**CLK, ARST, SRST:** clock, asynchronous and synchronous reset.

**FLASH_NLD:** program mode selection signal, active low. After power up, when set to '0' the program download mode is enabled, when set to '1' the system is in program running mode.

**FAT_READ:** FAT_READ mode selection signal, it selects how the program is stored in the serial flash and then how it must be read to be transferred to the SRAM. It is possible to store several programs in the same serial flash, and it is possible to select the program to be executed. See in-system programming section for more details.

**NRAM_ROM:** flash or traditional memory configuration selection signal. In order to use the ADD1000A with a flash memory the NRAM_ROM pin must be tied low. To use it with a ROM memory the NRAM_ROM pin must be set to '1'.

**RELOAD:** reload system signal, active high. When this signal is asserted the flash loader reloads the program from the flash memory to the RAM. In the ADD1000A the microcontroller can force program reloading by writing in a specific function register.

**TX, RX:** serial interface signal.

**N_CS, SCK, SI, SO:** SPI interface signal.

**ALE:** address latch enable signal.

**DRDY:** data ready indication signal. When the data bus has a valid data this signal is asserted.

**ADDR, DATA, N_WE:** RAM control signals.

**PORTS_IN, PORTS_OUT:** connection in and out of the microcontroller ports

As it is shown in Figure 5.1 the flash loader acts as a bus switcher. It takes control over the program and data buses (P0, P2, P3(7:6)), the serial port (P3(1:0)) and pins of port P5 (P5(3:0) that are used as serial peripheral bus to connect the serial flash.

| ADD8051C12 | FLASH LOADER |
|---|---|
| P3.0 | SSI RxD |
| P3.1 | SSI TxD |
| P5.0 | SPI nCS |
| P5.1 | SPI SCK |
| P5.2 | SPI SI |
| P5.3 | SPI SO |

*Table 5.1: ADD8051C12A/FLASH LOADER shared pins*

After program booting the shared pins are fully software controlled. This allows user to use the unoccupied entries of the serial flash to store non volatile program data. The microcontroller has full access to all the serial flash content, care must be taken to avoid unexpected writing over program code.

## 5.3. IN-SYSTEM PROGRAMMING

To enable in-system programming FLASH_NLD pin must be tied low. In this mode of operation the flash loader receives through the RS-232 serial interface (57600 bauds, 1 stop bit, without parity bit) any standard SPI flash command (mode 0) and it executes the requested instruction.

Figure 5.4 and Table 5.2 show RS-232 datagram formats.

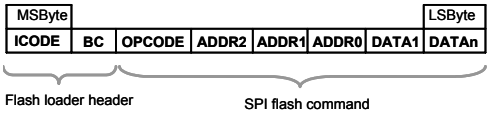



*Figure 5.4: RS-232 datagram format*

**ICODE**: This byte indicates to the flash loader the type of instruction to be executed.

**BC**: BYTECOUNT byte indicates the number of data bytes (BC +1: from 1 to 256) to read or write in the desired SPI instruction. If the requested instruction is not of reading or writing the BC field is not used and can take any value.

**OPCODE**: This byte contains the serial flash OPCODE that defines the operation to be executed.

**ADDR2 to ADDR0**: a 24 bits long field to store the flash address. ADDR2 is the most significative byte of the address field.

**DATA1 to DATAn:** data bytes of the SPI instruction (n = BC). The DATA1 byte is the first data byte of the write or read operation.

If the FAT_READ pin is tied low the start address of the program must be 010000 (Hex) otherwise, the start address must be stored at the three first bytes of the flash memory, the first byte is the most significative.

| ICODE (Hex) | DESCRIPTION | TYPICAL USE | RS-232 FORMAT (Hex) |
|---|---|---|---|
| 01 | Write data instruction. | Page Programming. | 01 N-1 OPC AD2 AD1 AD0 D1 ... DN |
| 02 | Read data instruction. | Read or Read_id. | 02 N-1 OPC AD2 AD1 AD0 |
| 04 | Instruction with only opcode field. | Write enable/disable. | 04 XX OPC |
| 08 | Instruction without data. | Page/sector erase. | 08 XX OPC AD2 AD1 AD0 |
| 10 | Read instruction without address field. | Read status register. | 10 N-1 OPC |
| 20 | Write instruction without address field. | Write status register. | 20 N-1 OPC D1 ... DN |

*Table 5.2: instruction datagram*

## 5.4. SYSTEM START-UP

If the ADD1000A FLASH_NLD pin is asserted and the system is configured to use a serial flash memory to store the program (NRAM_ROM pin tied low), the flash loader transfers the program (64K x 8)from the flash memory to the top half of the RAM memory (128K x 8).

When this step finishes, 200ms approximately, the microcontroller begins to execute the program and the system is ready for its use. The start-up cycle is performed each time the system is powered up. Microcontroller software can force start-up writing in a specific microcontroller register.

When FAT_READ is set to '0' the flash-loader reads the serial flash starting at "010000" address. When FAT_READ is set to '1' the flash loader reads the three first bytes and it uses the concatenated number as starting address for the program code:

ADDRESS(23:0) = B0(7:0) & B1(7:0) & B2(7:0)

A larger serial flash allows user to store several programs. The microcontroller can access the starting address in the serial flash to change it. Then when the system reboots and the start-up cycle is performed again a new program will be executed.

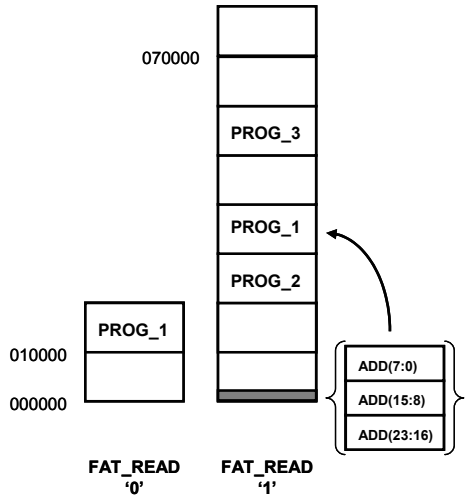



*Figure 5.5: serial flash program storage*

## 5.5. SUPPORTED SPI FLASH DEVICES

The requirements that must fulfill the flash memory to be compatible with the flash loader are:

- It must support SPI mode 0 (CLK signal is normally low).
- The read instruction operation code must be 03 (Hex).
- The page width must be 256 bytes.
- Its size must at least be 512Kb.

The Table 5.3 shows some of the compatible devices.

| DEVICE | VENDOR |
|---|---|
| M25PXX | STMicroelectronics (ST) |
| M25PEXX | STMicroelectronics (ST) |
| M45PEXX | STMicroelectronics (ST) |
| NX25PXX | NexFlash |
| SST25LFXXXX | Silicon Storage Technology (SST) |
| SST25VFXXXX | Silicon Storage Technology (SST) |
| PM25LVXXX | Programmable Microelectronics Corp. (PMC) |

*Table 5.3: serial flash devices*

# CHAPTER 6. DIMMER PERIPHERAL

## 6.1. GENERAL DESCRIPTION

The ADD1000A contains a dimmer peripheral to control up to four triacs and four switches. This peripheral is capable of doing a Phase Angle control in order to change the power of the lamps. Phase Angle control uses a low switch frequency to chop the power line sine wave. The firing angle ($\alpha$) of the switch can be varied. The average voltage will be proportional to the area under the sine wave. Thus, the average voltage is the integral from the firing angle to the zero crossing, the cosine of the firing angle:

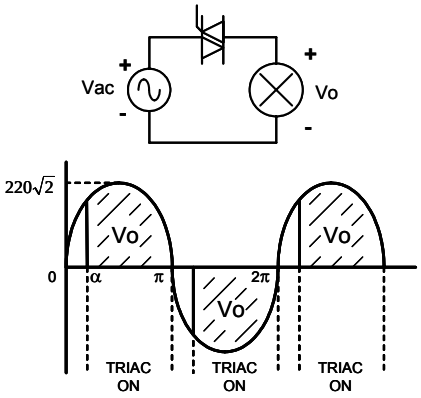



*Figure 6.1: phase control*

The configuration registers are used to change the firing angle, turn on/off the lamps, and read the state of the switches (see CONFIGURATION REGISTERS)

This control software tasks spend a high percent of CPU time, so the ADD1000A has some of these tasks developed via hardware to reduce the CPU computational load of the 8051 integrated microcontroller.

## 6.2. CONFIGURATION REGISTERS

The configuration registers of this controller are accessed by the microcontroller as external data RAM entries. They are placed in the upper side of the address space. Table 6.1 shows the configuration registers address map.

### 6.2.1. CONTROL REGISTER

This register selects the extern circuit type connected to VNR pin (see EXTERN CIRCUITS). If a zener diode is used, VT must be equal to zero. Otherwise, VT must be equal to one.

| ADDRESS | REGISTER NAME |
|---|---|
| FEA0 | VNR_TYPE |
| FEA1 | INP_ST |
| FEA2 | OUT_ST |
| FEA3 | OUT_REF3 |
| FEA4 | OUT_REF2 |
| FEA5 | OUT_REF1 |
| FEA6 | OUT_REF0 |
| FEA7 | POLARITY |

*Table 6.1: register address map*

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | **VT** |

*Figure 6.2: CONTROL register*

### 6.2.2. INP_ST REGISTER

This register contains the state of the switches and is a read-only register.

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | **S3** | **S2** | **S1** | **S0** |

*Figure 6.3: INP_ST register*

**$S_i$:** This bit contains the state of the switch connected to $INTERR_i$ pin. If the switch is ON, $S_i$ is equal to one, else to zero. (With i from 0 to 3).

### 6.2.3. OUT_ST REGISTER

OUT_ST register selects which lamps are switched on.

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| x | x | x | x | **L3** | **L2** | **L1** | **L0** |

Note: x ≡ don't care.

*Figure 6.4: OUT_ST register*

**$L_i$:** If this bit is equal to one, the lamp connected to $TRIAC_i$ is on. (i from 0 to 3).

### 6.2.4. OUT_REF REGISTERS

These four registers are used to adjust the power of the lamps. Each one must contain a value between 0 and 99 (50 Hz frequency). If values greater than 99 are used, the triac may be fired at the next semiperiod. If power frequency is 60Hz, values don't must be greater than 75.

The smaller value, the greater power. OUT_REF0, OUT_REF1, OUT_REF2 and OUT_REF3 show the power of the lamps connected to TRIAC0, TRIAC1, TRIAC2 and TRIAC3 pins respectively.

If OUT_REF value is lower than 15, the firing is continuous. Otherwise, the firing width is 200 us. With values greater than 90, the triac is not fired.

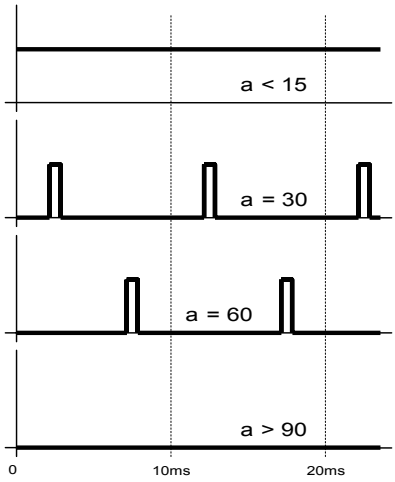



*Figure 6.5: firing vs. out_ref register*

### 6.2.5. POLARITY REGISTER

| MSb | | | | | | | LSb |
|---|---|---|---|---|---|---|---|
| **PE3** | **PE2** | **PE1** | **PE0** | **PT3** | **PT2** | **PT1** | **PT0** |

*Figure 6.6: POLARITY register*

**$PE_i$:** If this bit is set to one, the bit $S_i$ in INP_ST register is equal to one when the switch connected to $INTERR_i$ pin is ON, and it's equal to zero when the switch is OFF. If this bit is equal to zero, the bit $S_i$ is equal to one when the switch is OFF, and it's equal to zero when the switch is ON. (i from 0 to 3).

**$PT_i$:** If this bit is set to one, the triac connected to $TRIAC_i$ pin is firing with chip ground. If this bit is equal to zero, the triac is firing with Vcc. (i from 0 to 3).

### 6.2.6. RESET VALUES

The default values after reset of the dimmer peripheral configuration registers are shown in the following table:

| | Value (hex) |
|---|---|
| **VNR_TYPE** | 00 |
| **INP_ST** | 00 |
| **OUT_ST** | 00 |
| **OUT_REFn** | 00 |
| **POLARITY** | FF |

*Table 6.2: Reset values*

With this default values the dimmer peripheral configuration after reset is:

- All lamps are switched off.
- A zener diode is used to detect the zero crossing.
- All triacs are fired with chip ground.
- All switches are detected non-inverted.

## 6.3. EXTERN CIRCUITS

Some external components are necessary to control a lamp with the ADD1000A. The external circuitry must be different depending on the connection, or not, of the integrated circuit ground to power line Neutral. This is shown in Figure 6.7 and Figure 6.8.

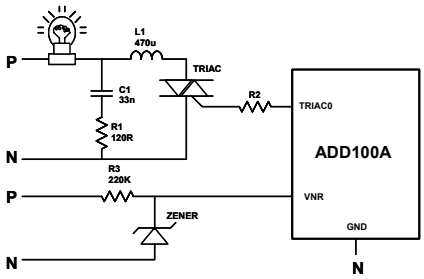



*Figure 6.7: phase control without ground connection*

VNR pin is used to detect the power line wave zero crossing. If the chip ground is not connected to the power line (N) an optocoupler must be used in the detection circuit.

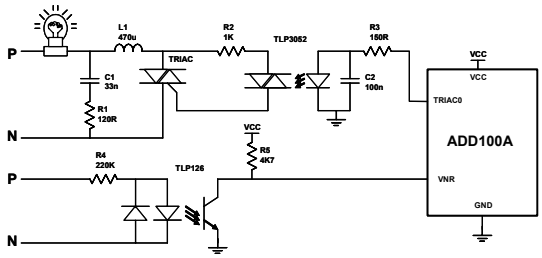



*Figure 6.8: phase control with ground connection*

To connect a mains switch to the integrated circuit, an optocoupled connection is the preferred option (see Figure 6.9).

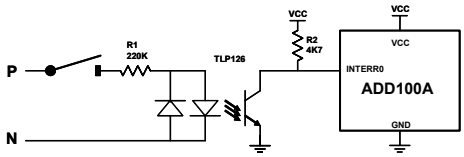



*Figure 6.9: mains switch connection*

Figure 6.10 represents the wave form when the switch is on. Pulse width depends on resistor R value. This peripheral can detect down to 2ms pulse width. So, resistor value not should be greater than 150k.

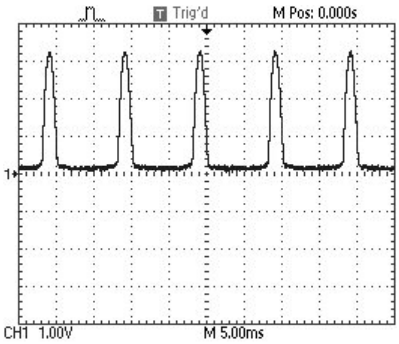



*Figure 6.10: Wave form with optocoupled connection*

If the switch uses logic levels, it must be connected to a 8051 port. It's possible to use a zener to detect the switch too, but it would be necessary to program specific software for it (see Figure 6.11).

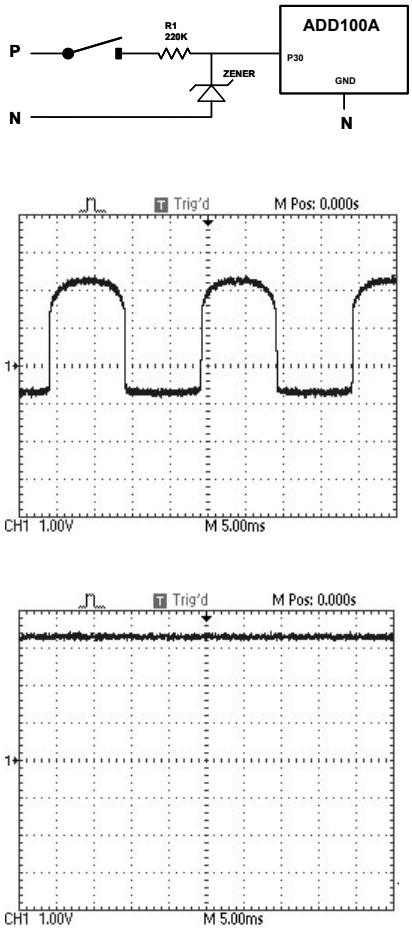



*Figure 6.11: Circuit and wave form for logic level switch detection*