

Intel 8051 Microcontroller



8051 Facts

- ▶ Developed by Intel in 1980, CISC, Harvard architecture, single chip microcontroller, become the industry standard till now!
- ▶ Most popular in the 1980s and early 1990s, today superseded by enhanced devices with 8051-compatible processor cores
- ▶ Manufactured by more than 20 independent manufacturers: Atmel, Infineon Technologies, Maxim , NXP , Winbond, ST Microelectronics, Silicon Laboratories , Texas Instruments and Cypress Semiconductor



Main Features

- ▶ 8-bit data bus and 8-bit ALU
- ▶ 16-bit address bus - 64 kB of RAM and ROM
- ▶ On-chip RAM - 128 (256) bytes ("Data Memory")
- ▶ On-chip ROM - 4 kB ("Program Memory")
- ▶ Four 8-bit bi-directional input/output ports
- ▶ One (two) UART (serial port)
- ▶ Two 16-bit timers
- ▶ Two-level interrupt priority
- ▶ Power saving mode
- ▶ All modern versions are CMOS



cont.

- ▶ The original 8051 core ran at 12 clock cycles per machine cycle, with most instructions executing in one or two machine cycles
- ▶ With a 12 MHz clock frequency, the 8051 achieved 1MIPS (for one-cycle instructions)
- ▶ Enhanced 8051 cores run at six, four, two, or even one clock per machine cycle, and have clock frequencies of up to 100 MHz
- ▶ Higher speed single cycle 8051 cores (<150MHz) are available for use in FPGAs and (>150MHz) in ASICs



Distinct Features

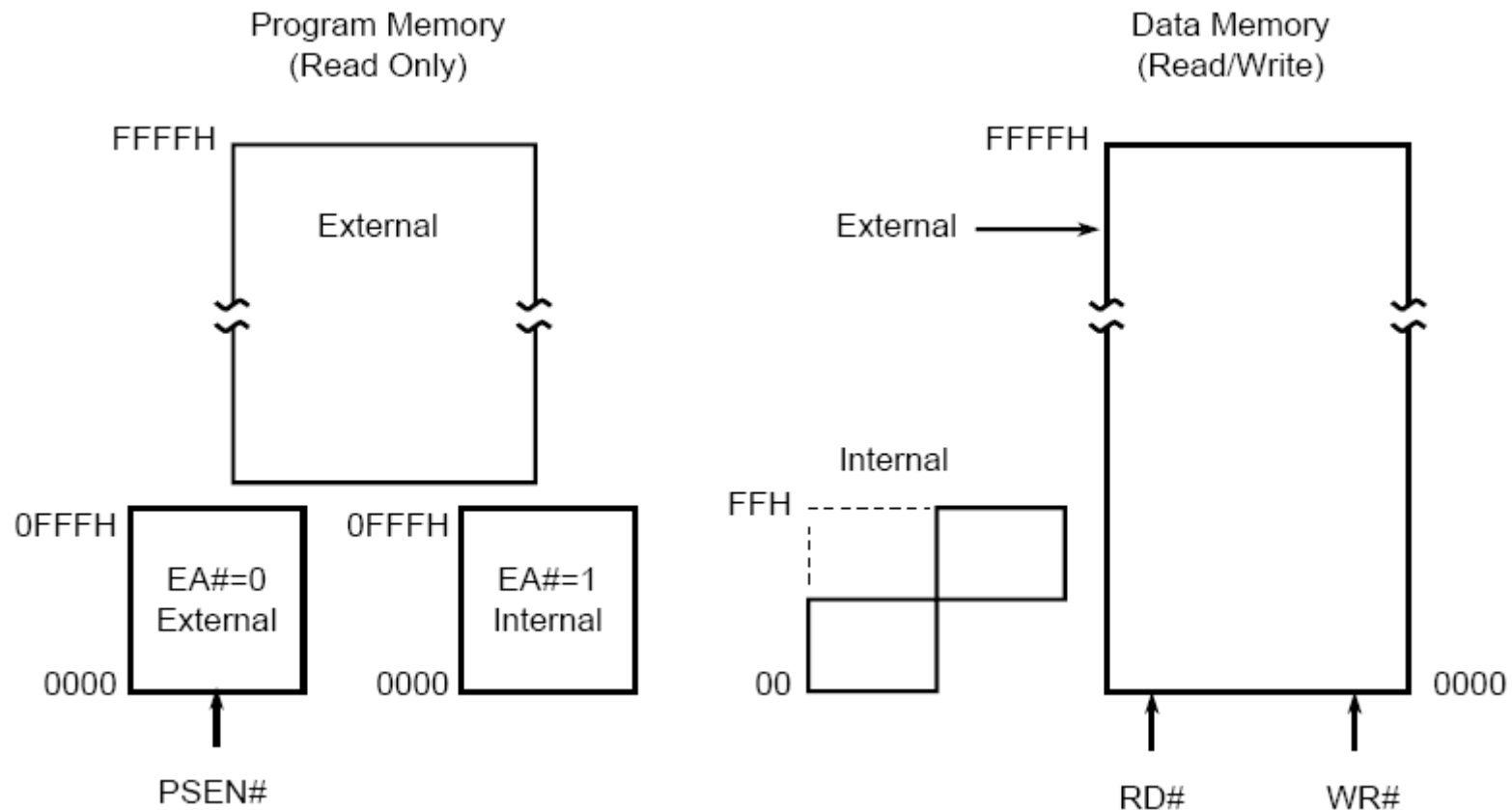
- ▶ Bit-level boolean logic operations can be carried out directly and efficiently on internal registers and RAM – especially valued in industrial control applications
- ▶ Four separate register sets, which can be used to greatly reduce interrupt latency - faster than storing interrupt context on a stack



Enhanced Versions

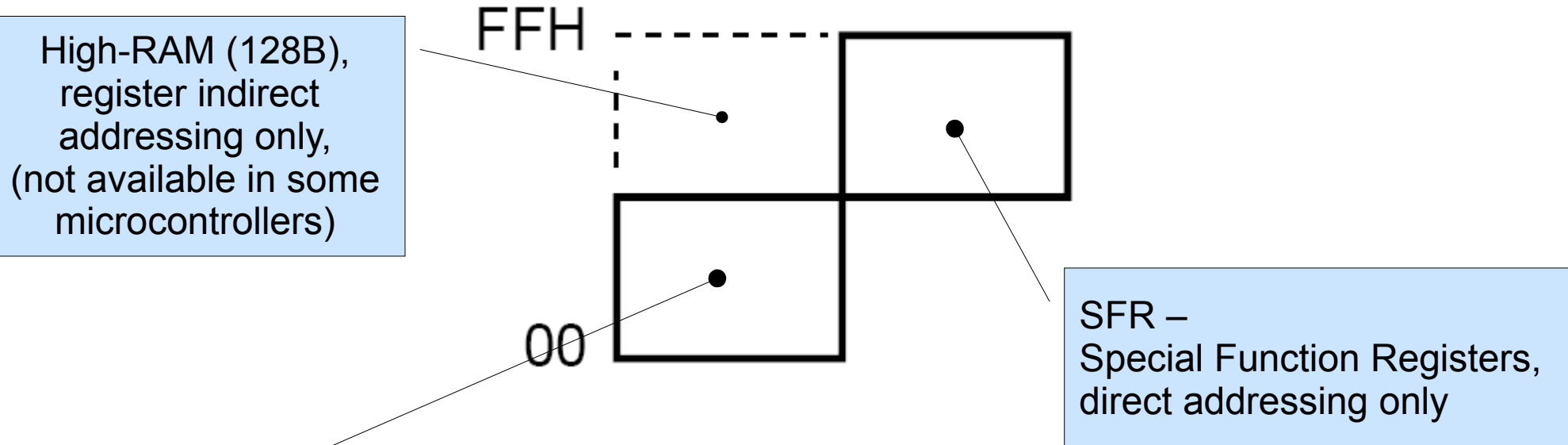
- built-in reset timers with brown-out detection
- on-chip oscillators
- self-programmable Flash ROM program memory
- bootloader code in ROM
- EEPROM non-volatile data storage
- I²C, SPI, and USB host interfaces
- PWM generators
- analog comparators
- A/D and D/A converters
- RTCs
- extra counters and timers
- in-circuit debugging facilities
- more interrupt sources
- extra power saving modes.

Memory Map



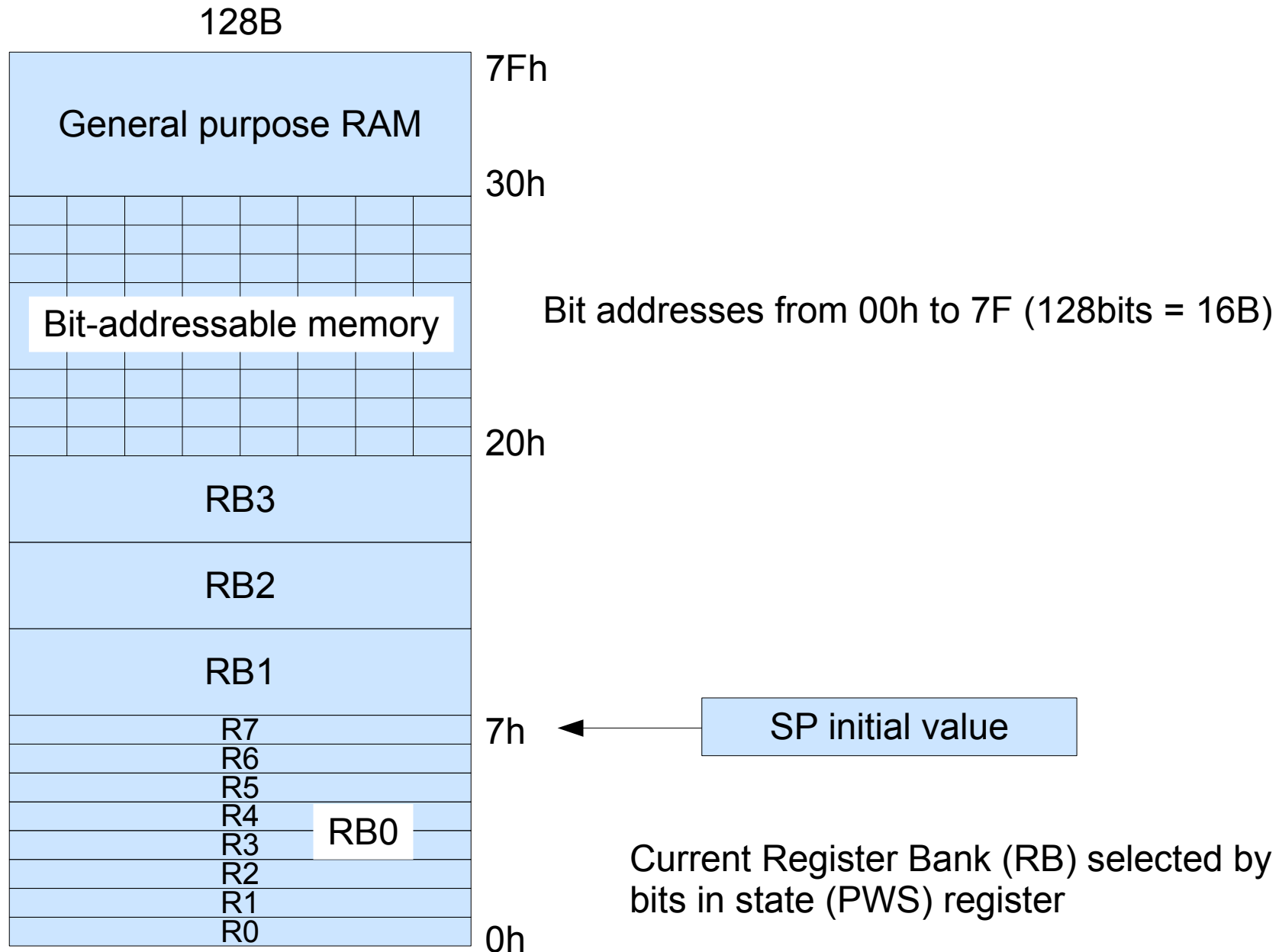
EA – controls low program-memory source
PSEN, RD and WR signals are generated for external memories during instruction fetch (PSEN) or external data access with MOVX instructions (RD or WR)

Internal Data Memory



- Low-RAM (128B)
- 4-banks of internal registers: RB0 – RB3 (00h-1Fh), 7-registers (R0-R7) in each bank
- Stack area – initial address 07h (grows towards higher addresses, SP on last element)
- Bit-addressable space (20h – 2Fh)
- General purpose space (30h – 7Fh)
- direct or indirect addressing modes possible

Low Data RAM





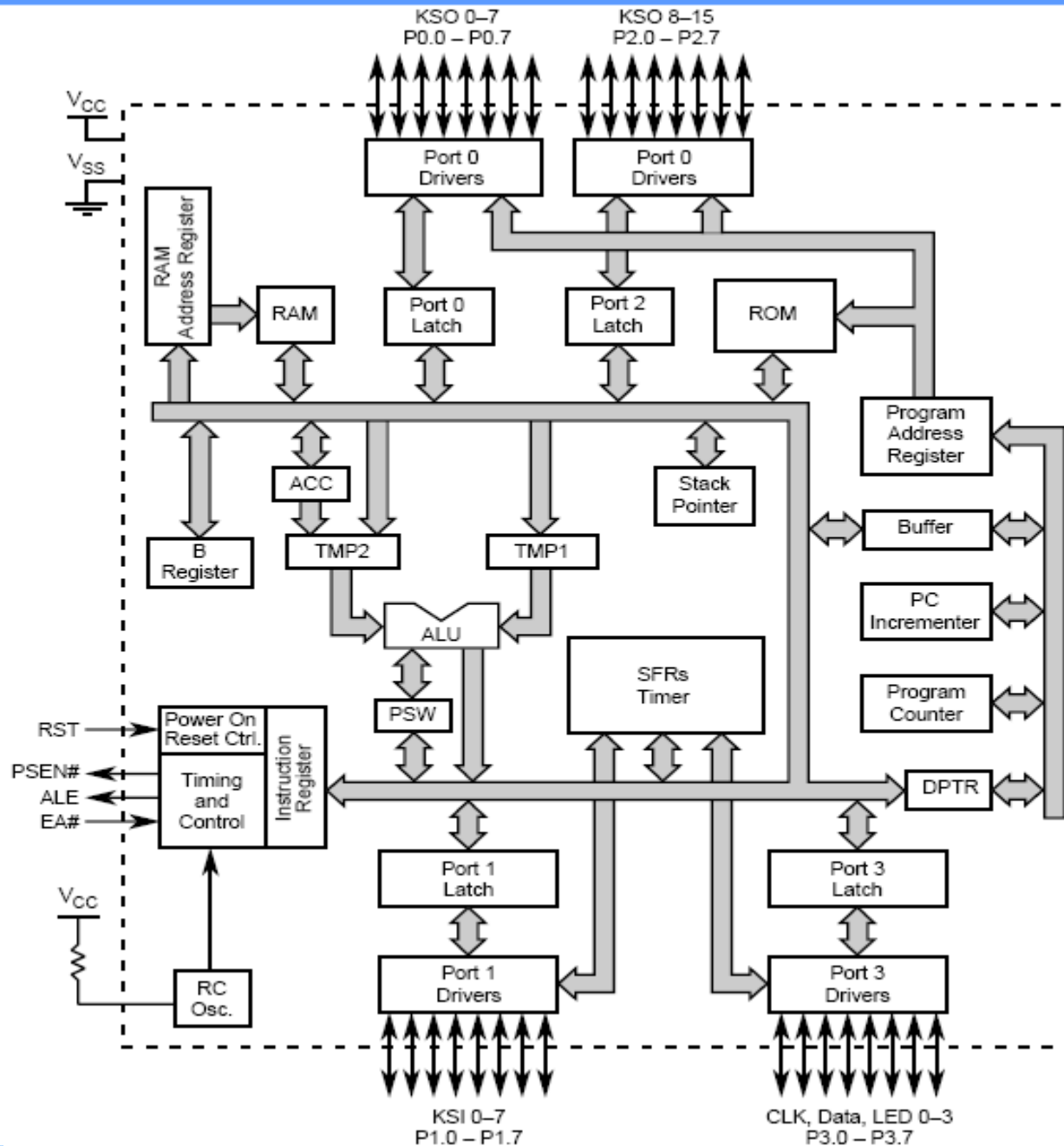
SFR

- ACC – Accumulator
- B – Aux. accumulator
- PWS – uC state register
- DPH, DPL – 16bit data pointer
- SP – Instruction & Stack pointers
- P0,P1,P2,P4 – I/O ports data registers
- IE – Interrupt mask
- IP,IPH – Interrupt priority registers
- PCON – Power-save control register
- TCON,TMOD – Timer control registers
- TL0,TH0 – 16-bit timer register
- AUXR – Aux. control bits

Enhanced versions of microcontroller introduce other specific registers

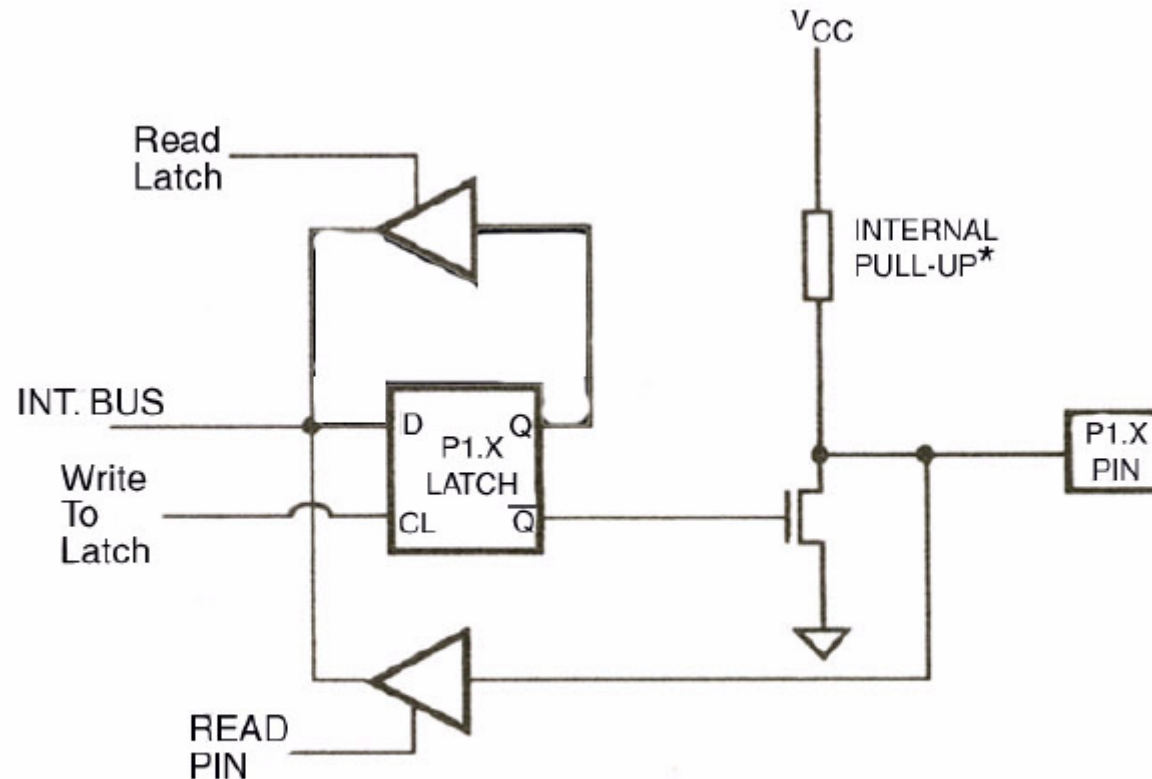
| | 0/8 | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F | |
|----|------------------|------------------|-----------------|-----------------|-----------------|-----|------------------|------------------|----|
| F8 | | | | | | | | | FF |
| F0 | B 00000000 | | | | | | | | F7 |
| E8 | | | | | | | | | EF |
| E0 | ACC 00000000 | | | | | | | | E7 |
| D8 | | | | | | | | | DF |
| D0 | PSW 00000000 | | | | | | | | D7 |
| C8 | | | | | | | | | CF |
| C0 | | | | | | | | | C7 |
| B8 | IP xxxx0000 | | | | | | | | BF |
| B0 | P3 11111111 | | | | | | | IPH xxxx0000 | B7 |
| A8 | IE 0xxx0000 | | | | | | | | AF |
| A0 | P2 11111111 | | | | | | | | A7 |
| 98 | | | | | | | | | 9F |
| 90 | P1 11111111 | | | | | | | | 97 |
| 88 | TCON 00000000 | TMOD xxxx0000 | TL0 00000000 | | TH0 00000000 | | AUXR xxxxxxx0 | | 8F |
| 80 | P0 11111111 | SP 00000111 | DPL 00000000 | DPH 00000000 | | | | PCON 00x00000 | 87 |
| | 0/8 | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F | |

Architecture Overview



Port 1 – Typical I/O

- ▶ I/O with weak pull-up (50k-100k)
- ▶ simplified direction setting (1→D: Input Pin)

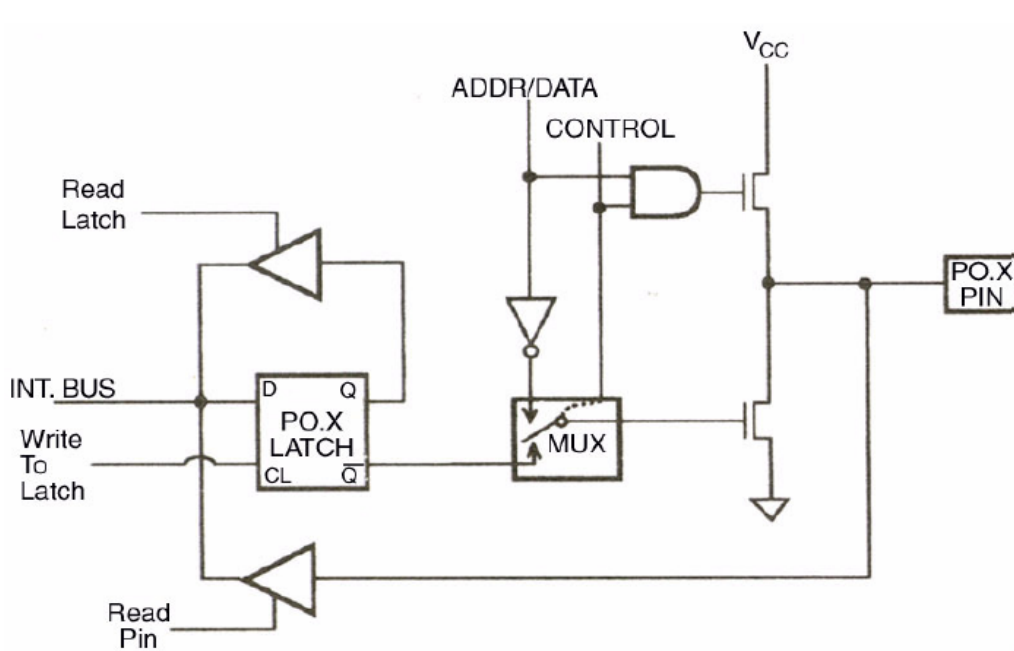


(B) PORT 1 BIT

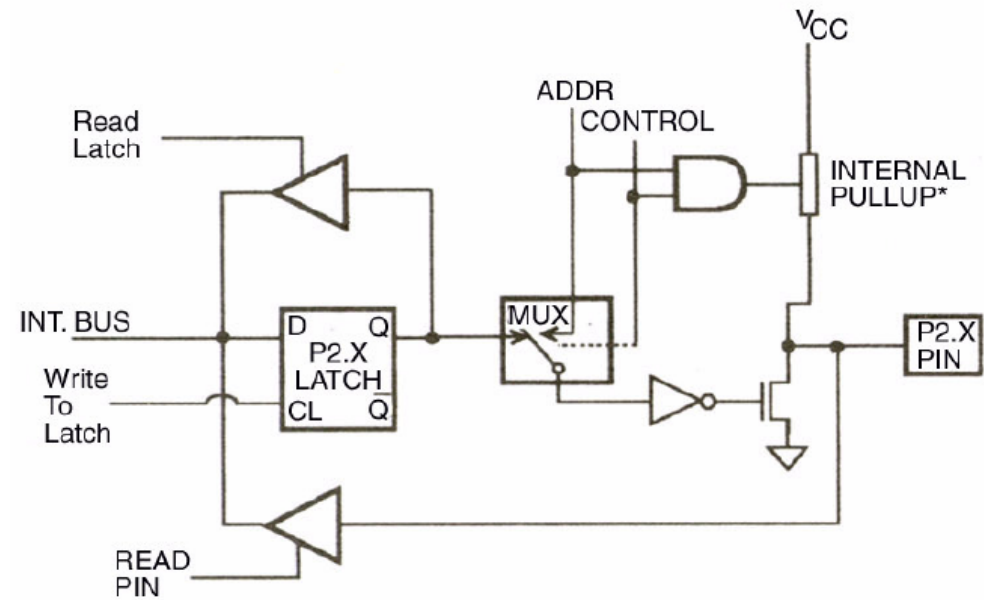


Port 0&2 – Memory Interface

- ▶ Memory interface independent from I/O settings
- ▶ Weak pull-up or open drain



(A) PORT 0 BIT

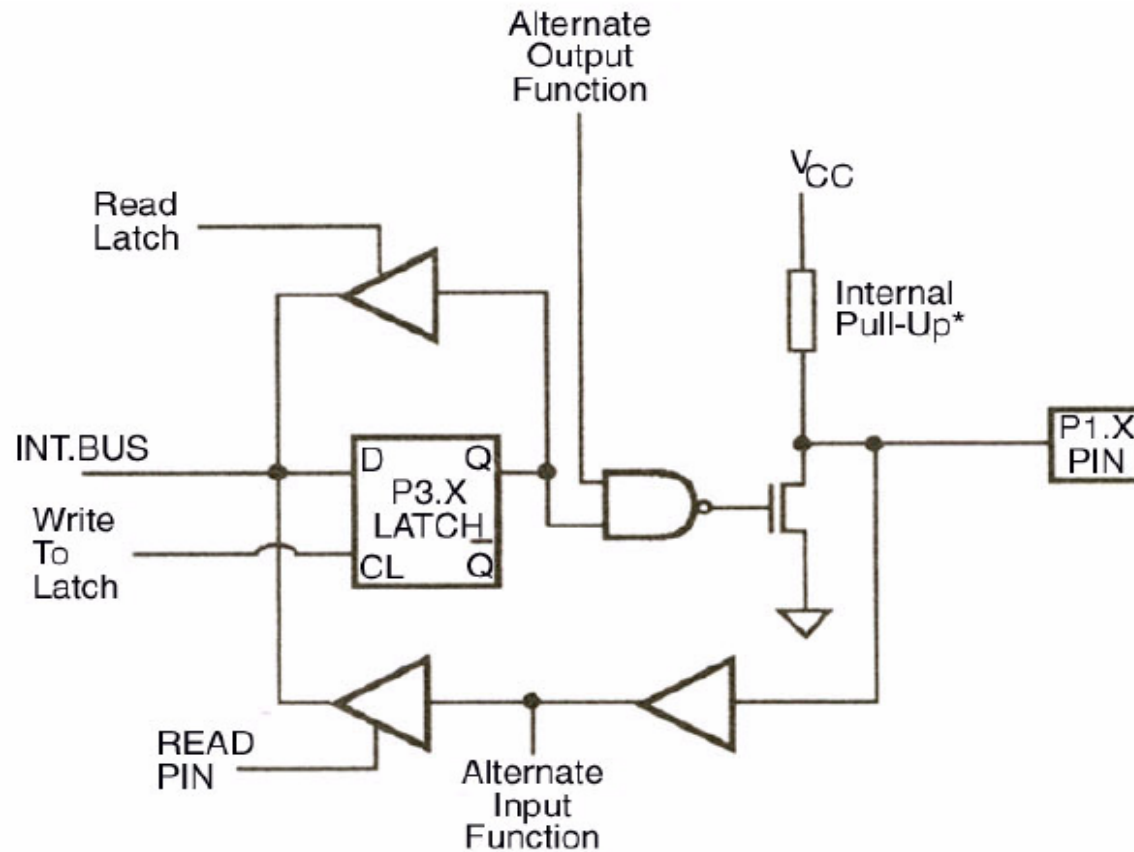


(C) PORT 2 BIT



Port 3 – Alternative Functions

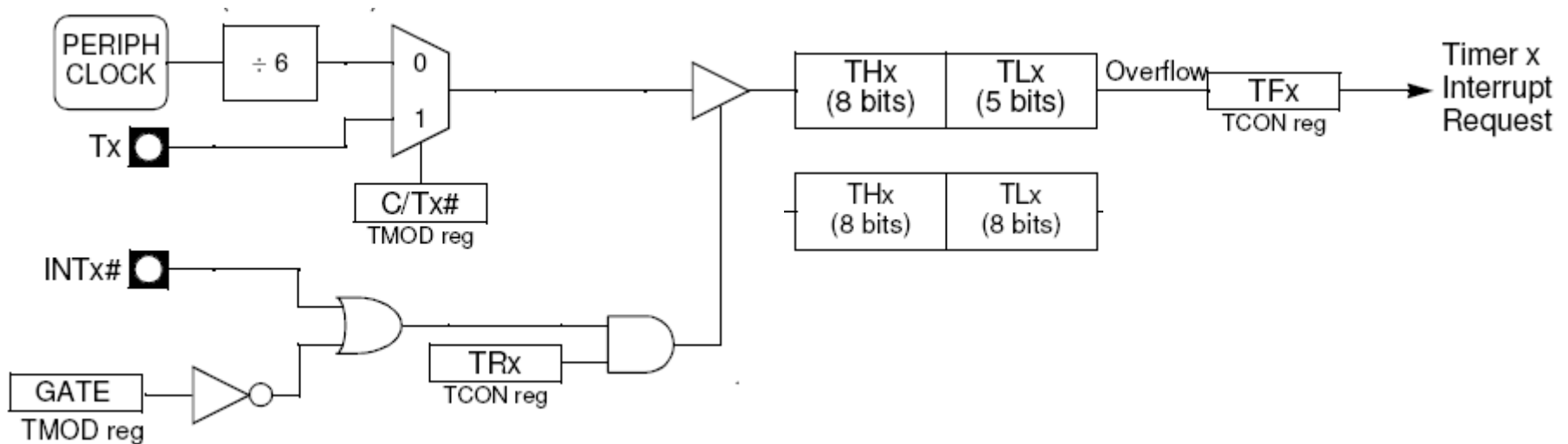
- ▶ Alternative input and output port functions



(D) PORT 3 BIT

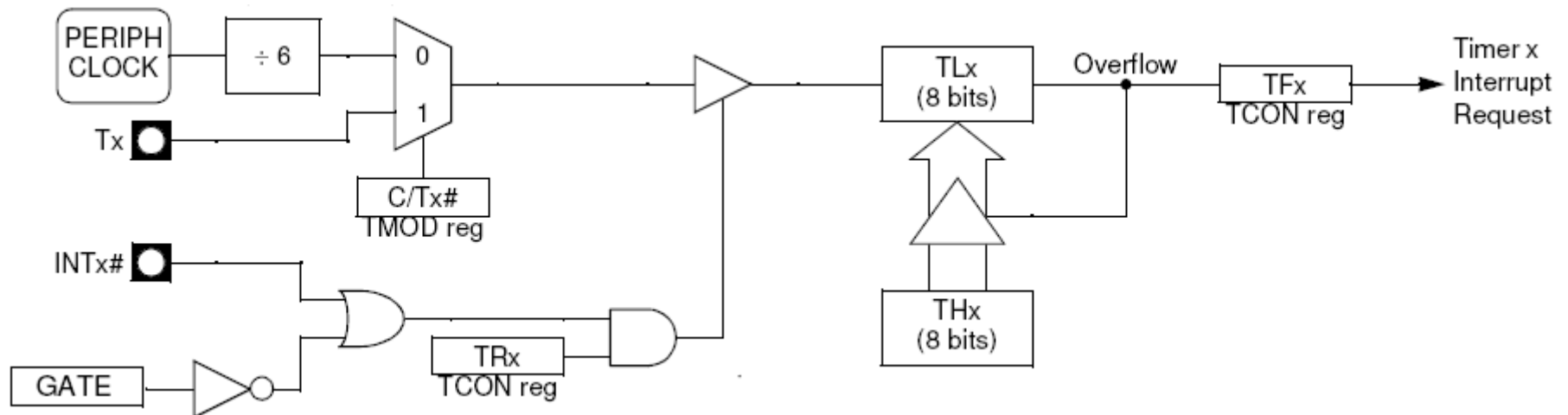
Timer 0 (1) – Mode 0&1

- ▶ Timer counts up and rolls over all 1s to all 0s
- ▶ Mode 0&1: 13 or 16-bit timer counter
- ▶ Clocked by uC oscillator or externally (Tx)
- ▶ Counting can be gated internally or externally (INTx)



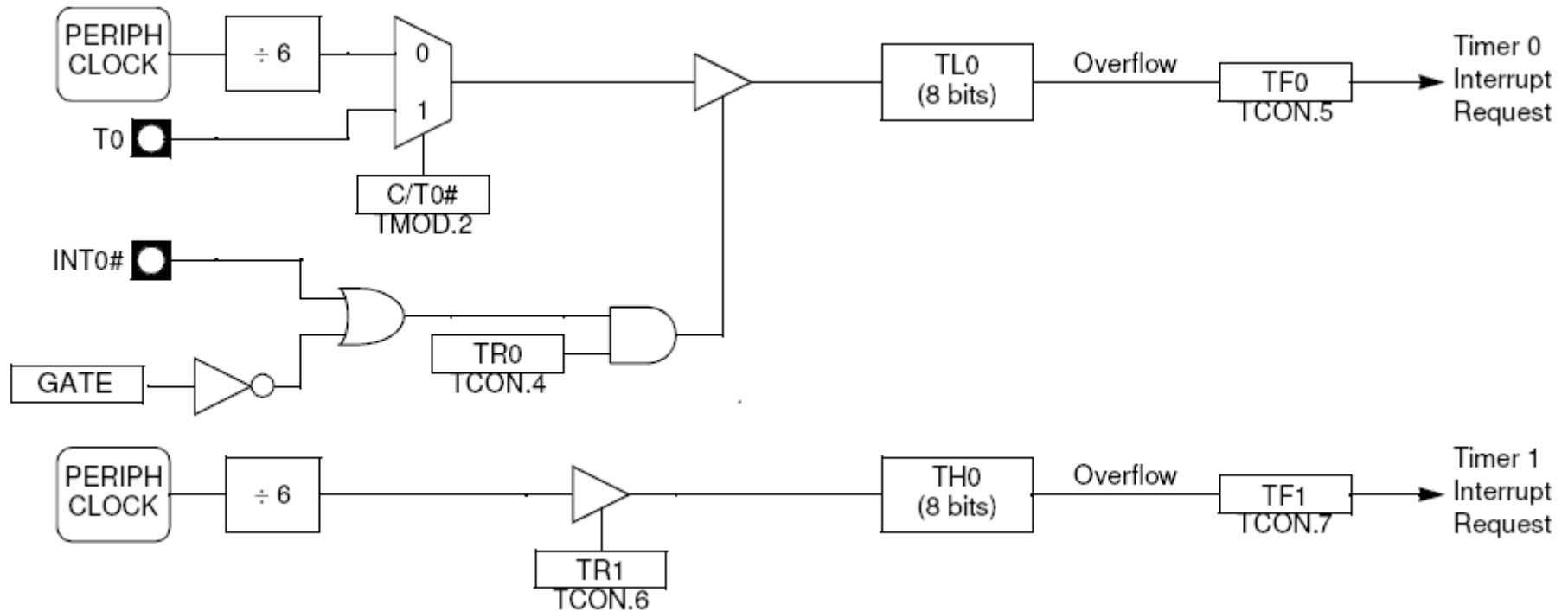
Timer 0 (1) – Mode 2

- ▶ 8-bit timer counter
- ▶ Timer counts up and is reloaded from register



Timer 0 (1) – Mode 3

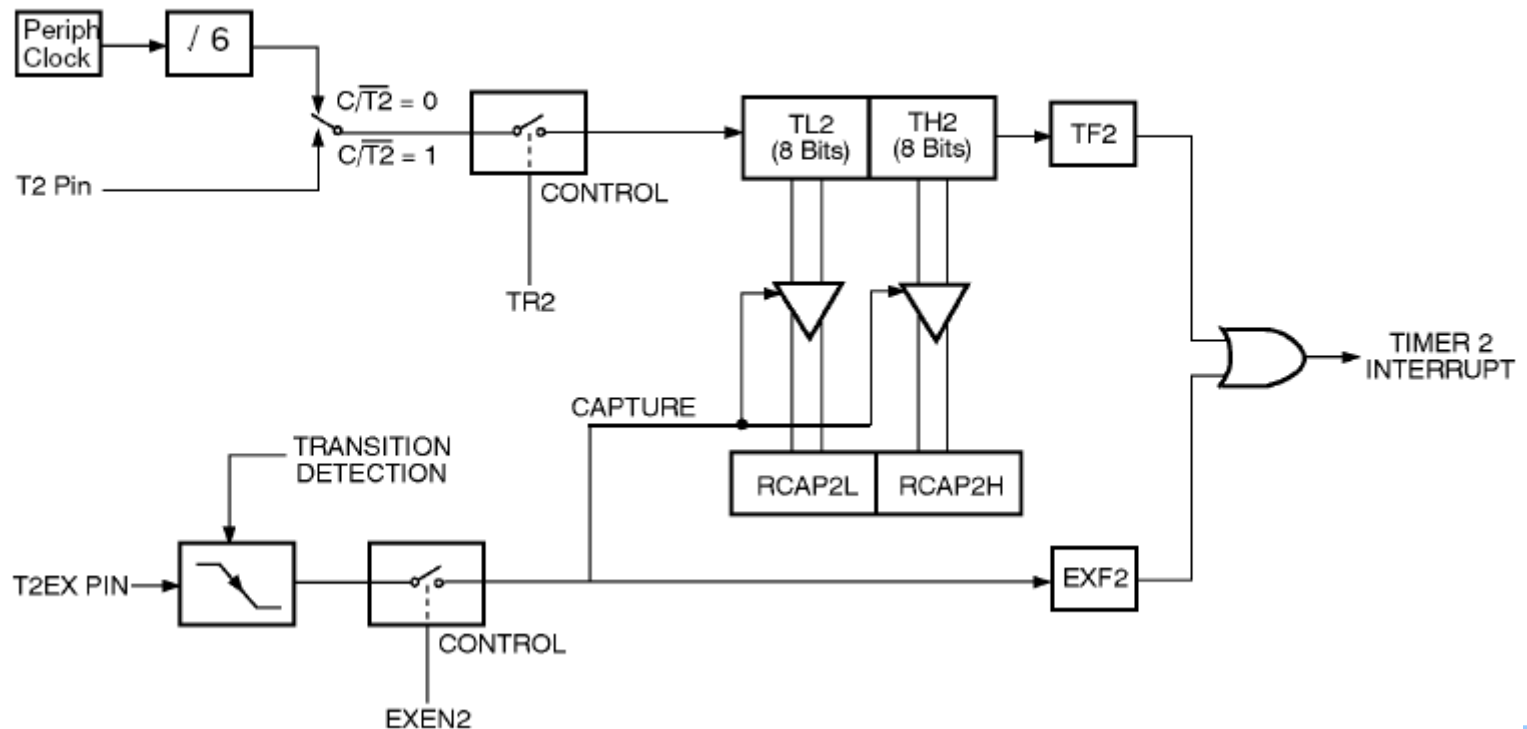
- ▶ Two 8-bit independent timers
- ▶ Timer 1 in mode 3 is halted



Timer 2

- ▶ Available in enhanced versions
- ▶ 16-bit timer, up or down counter
- ▶ 3 modes of operation:
 - auto-reload, capture, baud rate generator

Timer 2 in Capture Mode



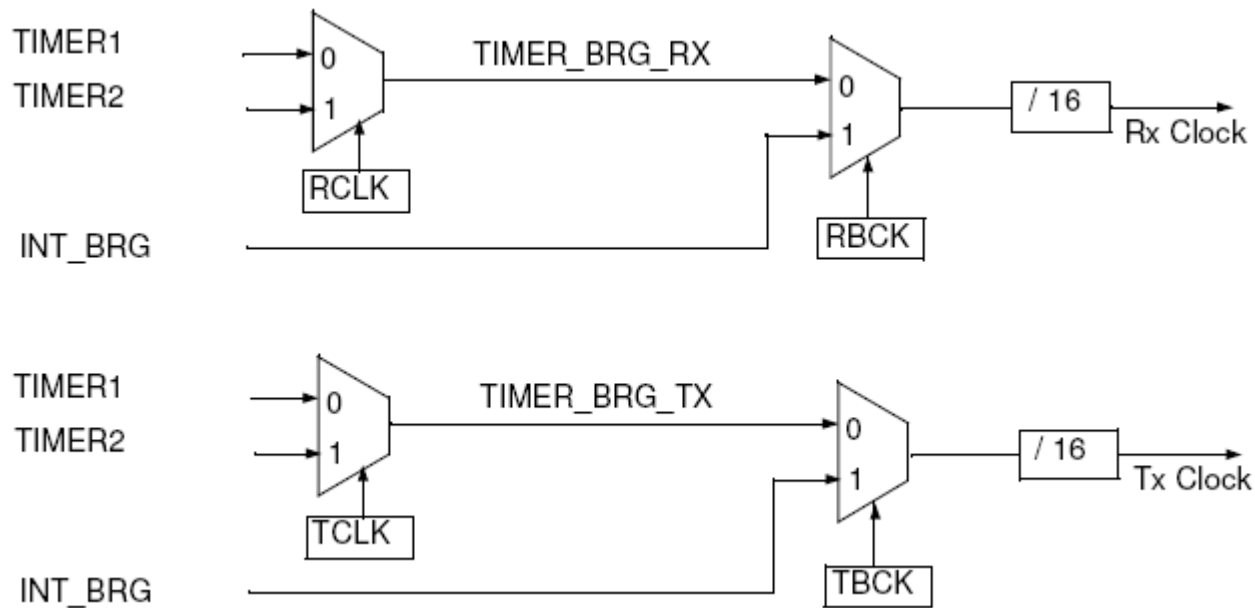


Serial Interface

- ▶ Many 8051s are equipped with UART operating in three full-duplex modes
- ▶ Modes of operation are highly configurable: frame format, parity, baud rate
- ▶ Asynchronous transmission and reception can occur simultaneously and at different baud rates

Baud Rate – Mode 1&3

- ▶ The baud rates in Modes 1 and 3 are determined by the Timer 1 or Timer 2 overflow rate
- ▶ The Baud Rate Generator for transmit and receive clocks can be selected separately





Commonly Used Baud Rates

► Timer 1 vs oscillator frequency

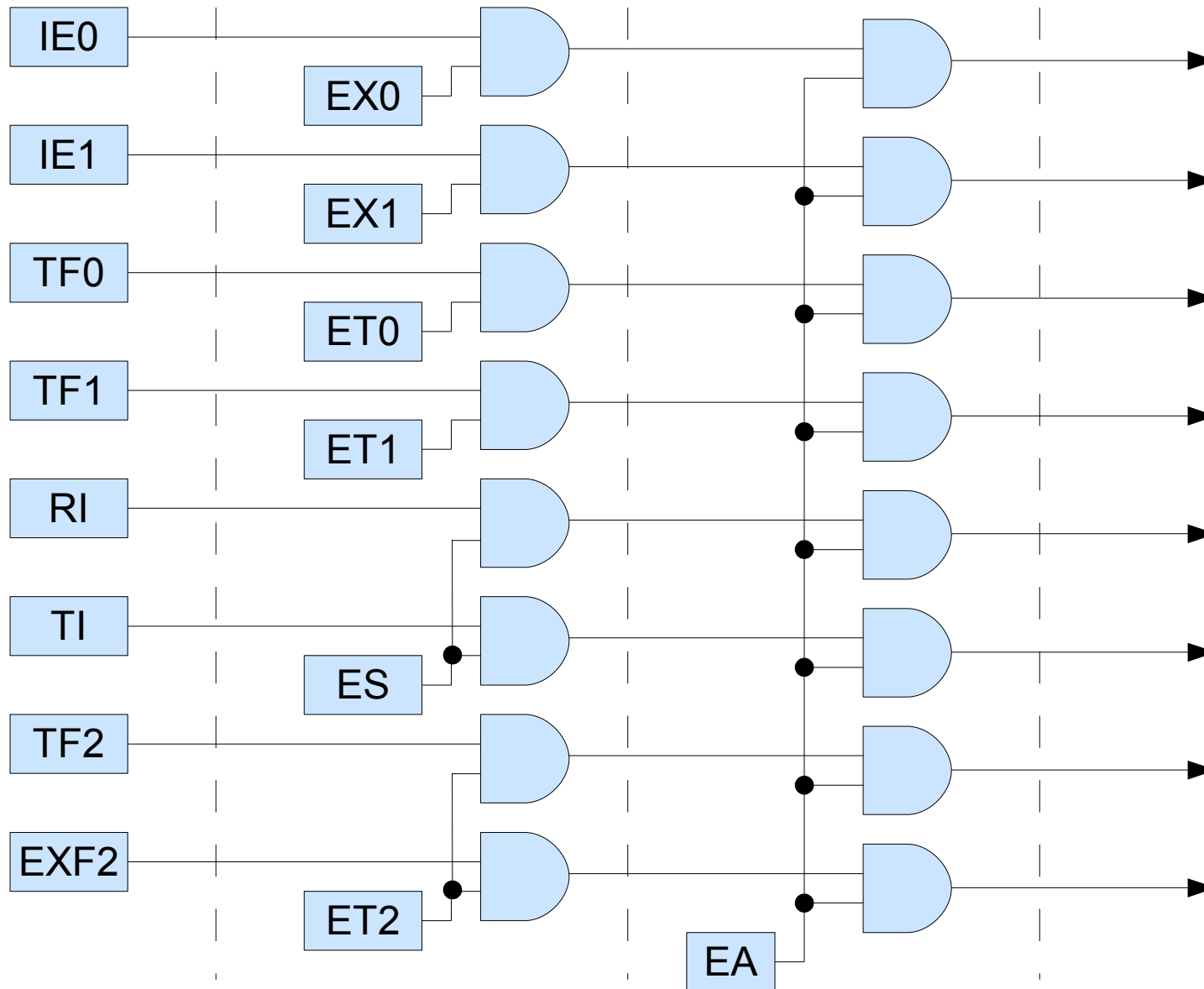
| Fosc (MHz) | 11.0592 | 12 | 14.7456 | 16 | 20 | SMOD |
|-----------------|---------|-----|---------|-----|-----|------|
| Baudrate | | | | | | |
| 150 | 40h | 30h | 00h | | | 0 |
| 300 | A0h | 98h | 80h | 75h | 52h | 0 |
| 600 | D0h | CCh | C0h | BBh | A9h | 0 |
| 1200 | E8h | E6h | E0h | DEh | D5h | 0 |
| 2400 | F4h | F3h | F0h | EFh | EAh | 0 |
| 4800 | | F3h | EFh | EFh | | 1 |
| 4800 | FAh | | F8h | | F5h | 0 |
| 9600 | FDh | | FCh | | | 0 |
| 9600 | | | | | F5h | 1 |
| 19200 | FDh | | FCh | | | 1 |
| 38400 | | | FEh | | | |
| 76800 | | | FFh | | | |



Interrupt System

- ▶ External Interrupts
- ▶ Timer Interrupts
- ▶ UART Interrupts
- ▶
- ▶ Interrupt requests bits:
 - IE0, IE1, TF0, TF1, TF2, EXF2, RI, TI
- ▶ Individual interrupt masks:
 - EX0, EX1, ET0, ET1, ET2, ES
- ▶ Global interrupt mask – 1bit EA

Interrupt System





Interrupt Priorities

- ▶ Interrupt of higher priority level interrupts the execution of lower priority level interrupt
- ▶ The final cycle in the execution of the instruction cannot be interrupted
- ▶ RETI or any access to the IE or IP registers cannot be interrupted

| | Source | Priority Within Level |
|---|------------|-----------------------|
| 1 | IE0 | (highest) |
| 2 | TF0 | |
| 3 | IE1 | |
| 4 | TF1 | |
| 5 | RI + TI | |
| 6 | TF2 + EXF2 | (lowest) |



Interrupt Vectors

- ▶ Interrupt vectors are fixed to the beginning of instruction memory
- ▶ If interrupts are used, the first instruction must be LCALL (3B) to the main program

| Source | Vector Address |
|------------|----------------|
| IE0 | 0003H |
| TF0 | 000BH |
| IE1 | 0013H |
| TF1 | 001BH |
| RI + TI | 0023H |
| TF2 + EXF2 | 002BH |



Addressing Modes

- ▶ **Direct Addressing (e.g. ADD A, 7FH)**
 - operand is specified by an 8-bit address field in the instruction, only lowest 128B of internal Data RAM and SFRs can be directly addressed
- ▶ **Indirect Addressing (e.g. ADD A, @R0)**
 - 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 register bank, or the Stack Pointer. The address register for 16-bit addresses can only be the 16-bit “data pointer” register, DPTR.



Addressing Modes

- ▶ Immediate Addressing (e.g. `MOV A, #100`)
 - The value of a constant is included in the instruction opcode in Program Memory
- ▶ Register Direct (e.g. `ADD A, R7`)
 - registers R0 through R7 (from current bank), can be accessed by 3-bit register specification within the opcode of the instruction



Addressing Modes

- ▶ Indexed Addressing (e.g. `MOV A, @A+PC`)
 - 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.


Arithmetic Instructions

| Mnemonic | Operation | Addressing Modes | | | | Execution Time in X1 Mode @12 MHz (µs) |
|----------------|--|-------------------|-----|-----|---------|---|
| | | Dir | Ind | Reg | Im m | |
| ADD A, <byte> | $A = A + \text{<byte>}$ | X | X | X | X | |
| ADDC A, <byte> | $A = A + \text{<byte>} + C$ | X | X | X | X | 1 |
| SUBB A, <byte> | $A = A - \text{<byte>} - C$ | X | X | X | X | 1 |
| INC A | $A = A + 1$ | Accumulator only | | | | 1 |
| INC <byte> | $\text{<byte>} = \text{<byte>} + 1$ | X | X | X | | 1 |
| INC DPTR | $DPTR = DPTR + 1$ | Data Pointer only | | | | 2 |
| DEC A | $A = A - 1$ | Accumulator only | | | | 1 |
| DEC <byte> | $\text{<byte>} = \text{<byte>} - 1$ | X | X | X | | 1 |
| MUL AB | $B:A = B \times A$ | ACC and B only | | | | 4 |
| DIV AB | $A = \text{Int}[A/B]$ $B = \text{Mod}[A/B]$ | ACC and B only | | | | 4 |
| DA A | Decimal Adjust | Accumulator only | | | | 1 |



Logical Instructions

| Mnemonic | Operation | Addressing Modes | | | | Execution Time @ 12MHz (µs) |
|-----------------------|----------------------------|------------------|-----|-----|-----|--------------------------------|
| | | 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 |
| CLR A | A = 00H | Accumulator only | | | | 1 |
| CLP 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 |



Boolean (Bit) Instructions

| Mnemonic | Operation | Execution Time @ 12MHz (µs) |
|-------------|---------------------------|--------------------------------|
| 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 |



Data Transfers

► Internal data RAM

| Mnemonic | Operation | Addressing Modes | | | | Execution Time @ 12MHz (µs) |
|------------------------|--------------------------------------|------------------|-----|-----|-----|--------------------------------|
| | | 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, # data 16 | 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 |



Data Transfers

▶ External data RAM

| Address Width | Mnemonic | Operation | Execution Time @ 12MHz (μs) |
|---------------|----------------|---------------------------|-----------------------------|
| 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 |

▶ Program ROM

| Mnemonic | Operation | Execution Time @ 12MHz (μs) |
|-------------------|-------------------------------|-----------------------------|
| MOVC A, @A + DPTR | Read Pgm Memory at (A + DPTR) | 2 |
| MOVC A, @A + PC | Read Pgm Memory at (A + PC) | 2 |



Example: Lookup Table

- ▶ Example: NBC→ASCII (0-9 digits)

```
    . . .  
    MOV     A, NBC_NUMBER  
    LCALL  TABLE  
    . . .  
TABLE:  INC     A  
        MOVC   A, @A+PC  
        RET  
        DB     48  
        DB     49  
        DB     50  
        DB     51  
    . . .
```



(Conditional) Jump Instructions

| Mnemonic | Operation | Addressing Modes | | | | Execution Time @ 12MHz (µs) |
|-----------------------|--------------------------------|------------------|-----|-----|-----|-----------------------------|
| | | 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 |
| CJNZ A,<byte>,rel | Jump if A = <byte> | X | | | X | 2 |
| CJNE <byte>,#data,rel | Jump if <byte> = #data | | X | X | | 2 |

| Mnemonic | Operation | Execution Time @ 12MHz (µs) |
|---------------|-------------------------|-----------------------------|
| JMP addr | Jump to addr | 2 |
| JMP @A + DPTR | Jump to A + DPTR | 2 |
| CALL addr | Call subroutine at addr | 2 |
| RET | Return from subroutine | 2 |
| RETI | Return from interrupt | 2 |
| NOP | No operation | 1 |



Example: Jump Table

```
MOV    DPTR,#JUMP_TABLE
MOV    A,INDEX_NUMBER
RL     A
JMP    @A+DPTR
...
JUMP_TABLE:
AJMP   CASE_0
AJMP   CASE_1
AJMP   CASE_2
AJMP   CASE_3
AJMP   CASE_4
```



Example: Loop

- ▶ Loop control with DJNZ instruction (Decrement and Jump if Not Zero)

```
        MOV        COUNTER,#10
LOOP:
    (begin loop)
    *
    *
    *
    (end loop)
    DJNZ         COUNTER, LOOP
    . . .
```



Program Status Word Register

| (MSB) | | | | (LSB) | | | |
|--|----------|---|-----|-------|----|---|---|
| CY | AC | F0 | RS1 | RS0 | OV | - | P |
| Symbol | Position | Name and Significance | | | | | |
| CY | PSW.7 | Carry flag | | | | | |
| AC | PSW.6 | Auxiliary Carry flag. (For BCD operations.) | | | | | |
| F0 | PSW.5 | Flag 0 (Available to the user for general purposes.) | | | | | |
| RS1 | PSW.4 | Register bank Select control bits 1 & 0. Set/cleared by software to determine working register bank (see Note). | | | | | |
| RS0 | PSW.3 | | | | | | |
| OV | PSW.2 | Overflow flag. | | | | | |
| - | PSW.1 | (reserved) | | | | | |
| P | PSW.0 | Parity flag. Set/cleared by hardware each instruction cycle to indicate and odd/even number of "one" bits in the accumulator, i.e., even parity. | | | | | |
| <p>Note:</p> <p>The contents of (RS1, RS0) enable the working register banks as follows:</p> <p>(0.0)-Bank 0(00H-07H) (0.1)-Bank 1(08H-0FH) (1.0)-Bank 2(10H-17H) (1.1)-Bank 3(18H-1FH)</p> | | | | | | | |



Programming Resources

▶ SDCC - Small Device C Compiler (GPL)

- optimizing ANSI - C compiler for Intel 8051, Maxim 80DS390, Zilog Z80, Motorola 68HC08, Microchip PIC16 and PIC18
- platforms: Linux, Windows, Mac OS X
- <http://sdcc.sourceforge.net>

▶ KEIL Embedded Development Tools (commercial)

- C compilers & assemblers, integrated environments for ARM7/ARM9/Cortex-M3, XC16x/C16x/ST10, 251, and 8051 MCU families
- Evaluation software with few limitations available!
- <http://www.keil.com>

Programming Resources

- ▶ Raisonance Embedded Development Tools
 - <http://www.raisonance.com>

8051 SOFTWARE TOOLS

| | | PRODUCTS | | | |
|----------------------------|-------------|---------------------------|----------------------|------------------------|--------------------------|
| Tool | Part number | Evaluation Version Eval51 | LITE Package RKitL51 | Compiler Package RCA51 | Enterprise Suite RKitE51 |
| IDE | RIDE | * | 32kb | * | * |
| Macro assembler | MA-51 | * | 32kb | * | * |
| Code banking linker | LX-51 | * | 32kb | * | * |
| Utilities | UTIL51 | * | 32kb | * | * |
| ANSI C compiler | RC-51 | * | <4kb | * | * |
| Simulator / debugger | SIMICE-51 | * | 32kb | 4kb | * |
| ROM- monitor | MON-51 | * | 32kb | 4kb | * |
| Tiny RTOS | KR-51 Tiny | 3 tasks | 8 tasks | 3 tasks | * |
| Scripts | RIDEScript | * | * | * | * |
| CodeCompressor | CComp-51 | (1) | (1) | (1) | * |
| Peripheral Development Kit | PDK | | | | * |
| Multiproc. simulation | MulSim51 | | | | * |
| RTOS | KR-51 | | | | * |



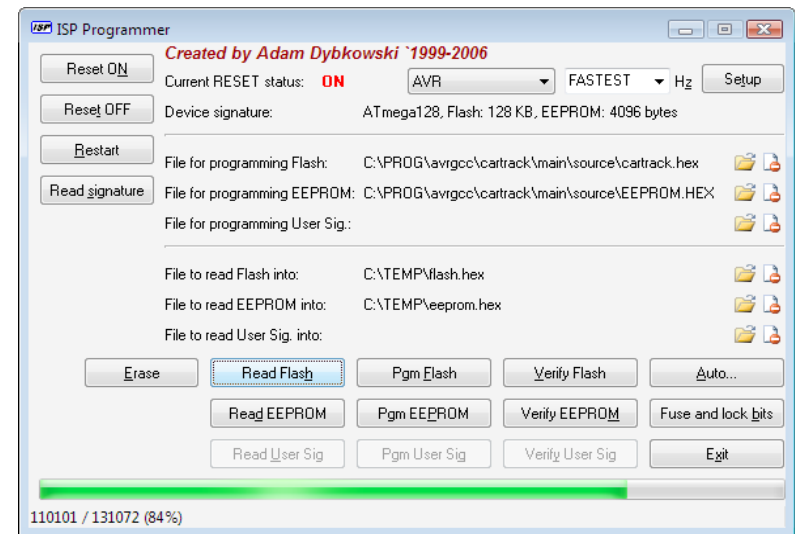
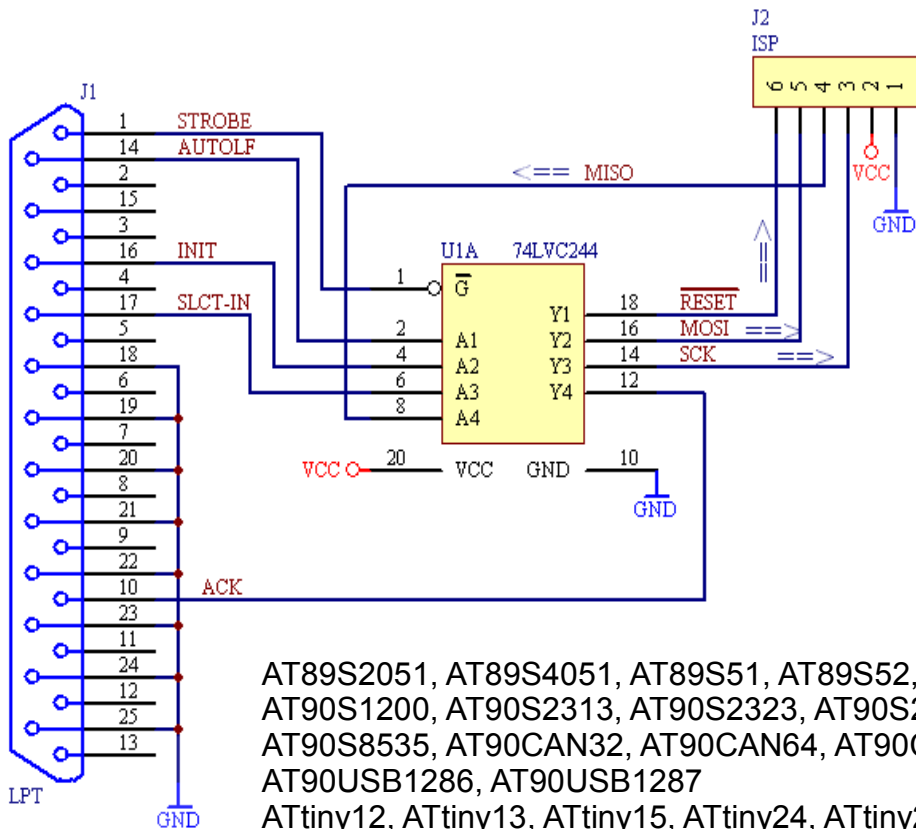
Device Programming

- ▶ Flash Programmers
- ▶ ISP Programmers
- ▶ Debugging Interfaces

Simple ISP Programmer

- ▶ Lots of free designs on the Internet:

<http://www.amwaw.edu.pl/~adybkows/elka/ispprog.html>



AT89S2051, AT89S4051, AT89S51, AT89S52, AT89S53, AT89S8252, AT89S8253
AT90S1200, AT90S2313, AT90S2323, AT90S2333, AT90S2343, AT90S4414, AT90S4433, AT90S4434, AT90S8515,
AT90S8535, AT90CAN32, AT90CAN64, AT90CAN128, AT90PWM2, AT90PWM3, AT90USB646, AT90USB647,
AT90USB1286, AT90USB1287
ATtiny12, ATtiny13, ATtiny15, ATtiny24, ATtiny25, ATtiny26, ATtiny44, ATtiny45, ATtiny84, ATtiny85, ATtiny261, ATtiny461,
ATtiny861, ATtiny2313
ATmega48, ATmega8, ATmega88, ATmega8515, ATmega8535, ATmega16, ATmega161, ATmega162, ATmega163,
ATmega164P, ATmega165P, ATmega168, ATmega169, ATmega32, ATmega323, ATmega324P, ATmega325,
ATmega329, ATmega64, ATmega128, ATmega640, ATmega644, ATmega644P, ATmega645, ATmega649, ATmega1280,
ATmega1281, ATmega2560, ATmega2561, ATmega3250, ATmega3290, ATmega6450, ATmega6490