MIT-WPU

School of Electrical Engineering

T.Y. B. Tech (Trimester V)

Microcontroller and Applications

ECE2003B

Unit-II

CIP-51 Architecture

CIP-51 Architecture

- Reset sources
- Oscillator options
- Port structure
- Timers
- Timer programming
- Interrupt handler
- Power management modes

Reset sources

Reset circuitry allows the controller to be easily placed in a predefined default condition. On entry to this reset state, the following occur:

- **CIP-51 halts program execution**
- **Special Function Registers (SFRs) are initialized to their defined reset values**
- **External Port pins are forced to a known state**
- **Interrupts and timers are disabled.**

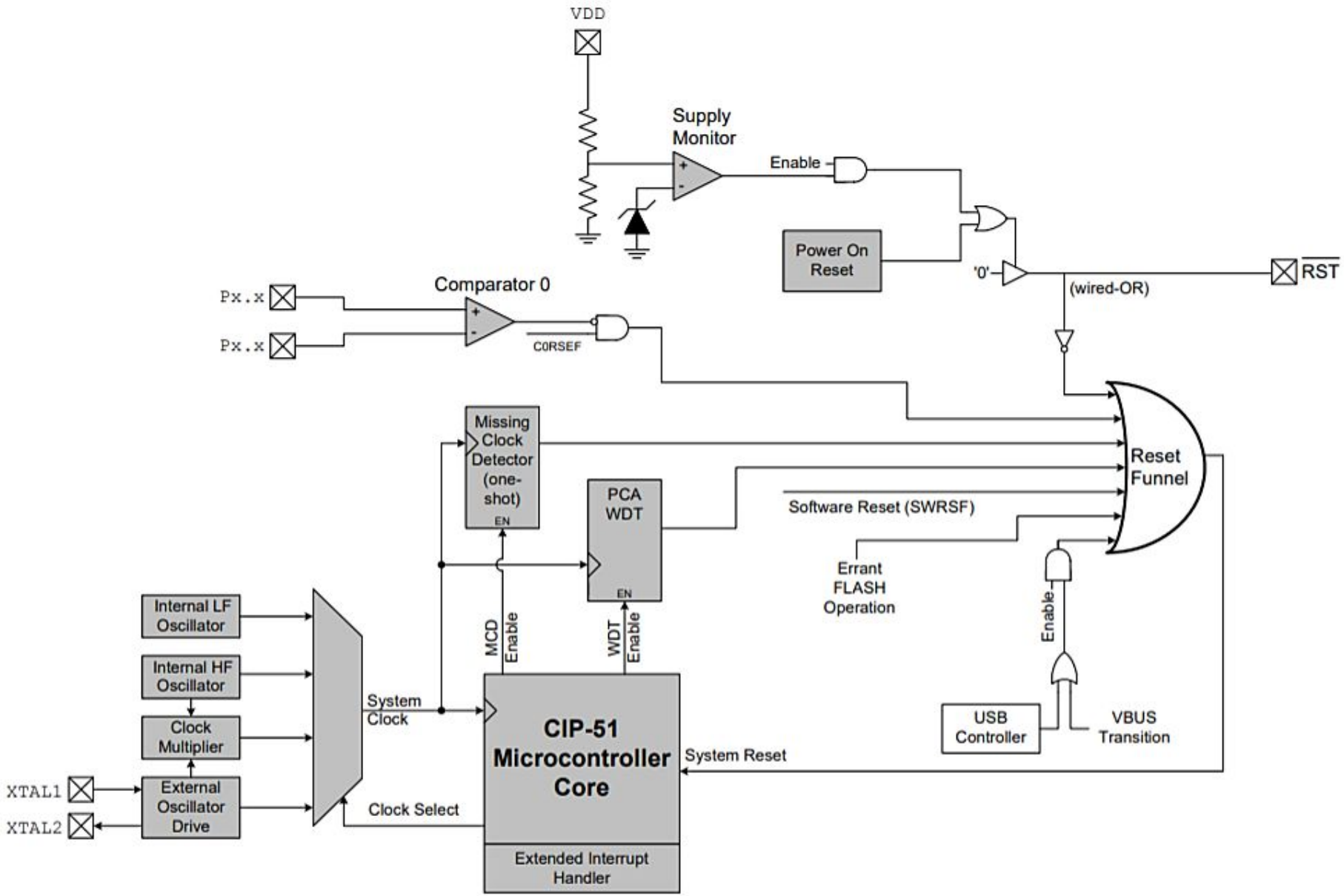
Reset sources

- Power-On Reset (POR)
- Power-Fail Reset / VDD Monitor
- External Reset
- Missing Clock Detector Reset
- Comparator0 Reset
- PCA Watchdog Timer Reset
- Flash Error Reset
- Software Reset
- USB Reset

Each reset source except for the POR, Reset Input Pin, or Flash error may be disabled by the user in software.

The WDT may be permanently enabled in software after a power-on reset during MCU initialization.

Reset sources



SFR: RSTSRC- Reset Source

R/W	R	R/W	R/W	R	R/W	R/W	R	Reset Value
USBRSF	FERROR	C0RSEF	SWRSF	WDTRSF	MCDRSF	PORSF	PINRSF	Variable
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xEF

Bit7: USBRSF: USB Reset Flag
0: **Read:** Last reset was not a USB reset; **Write:** USB resets disabled.
1: **Read:** Last reset was a USB reset; **Write:** USB resets enabled.

Bit6: FERROR: Flash Error Indicator.
0: Source of last reset was not a Flash read/write/erase error.
1: Source of last reset was a Flash read/write/erase error.

Bit5: C0RSEF: Comparator0 Reset Enable and Flag.
0: **Read:** Source of last reset was not Comparator0; **Write:** Comparator0 is not a reset source.
1: **Read:** Source of last reset was Comparator0; **Write:** Comparator0 is a reset source (active-low).

Bit4: SWRSF: Software Reset Force and Flag.
0: **Read:** Source of last reset was not a write to the SWRSF bit; **Write:** No Effect.
1: **Read:** Source of last was a write to the SWRSF bit; **Write:** Forces a system reset.

Bit3: WDTRSF: Watchdog Timer Reset Flag.
0: Source of last reset was not a WDT timeout.
1: Source of last reset was a WDT timeout.

Bit2: MCDRSF: Missing Clock Detector Flag.
0: **Read:** Source of last reset was not a Missing Clock Detector timeout; **Write:** Missing Clock Detector disabled.
1: **Read:** Source of last reset was a Missing Clock Detector timeout; **Write:** Missing Clock Detector enabled; triggers a reset if a missing clock condition is detected.

Bit1: PORSF: Power-On / V_{DD} Monitor Reset Flag.
This bit is set anytime a power-on reset occurs. Writing this bit selects/deselects the V_{DD} monitor as a reset source. **Note: writing '1' to this bit before the V_{DD} monitor is enabled and stabilized can cause a system reset.** See register VDM0CN (SFR Definition 11.1).
0: **Read:** Last reset was not a power-on or V_{DD} monitor reset; **Write:** V_{DD} monitor is not a reset source.
1: **Read:** Last reset was a power-on or V_{DD} monitor reset; all other reset flags indeterminate; **Write:** V_{DD} monitor is a reset source.

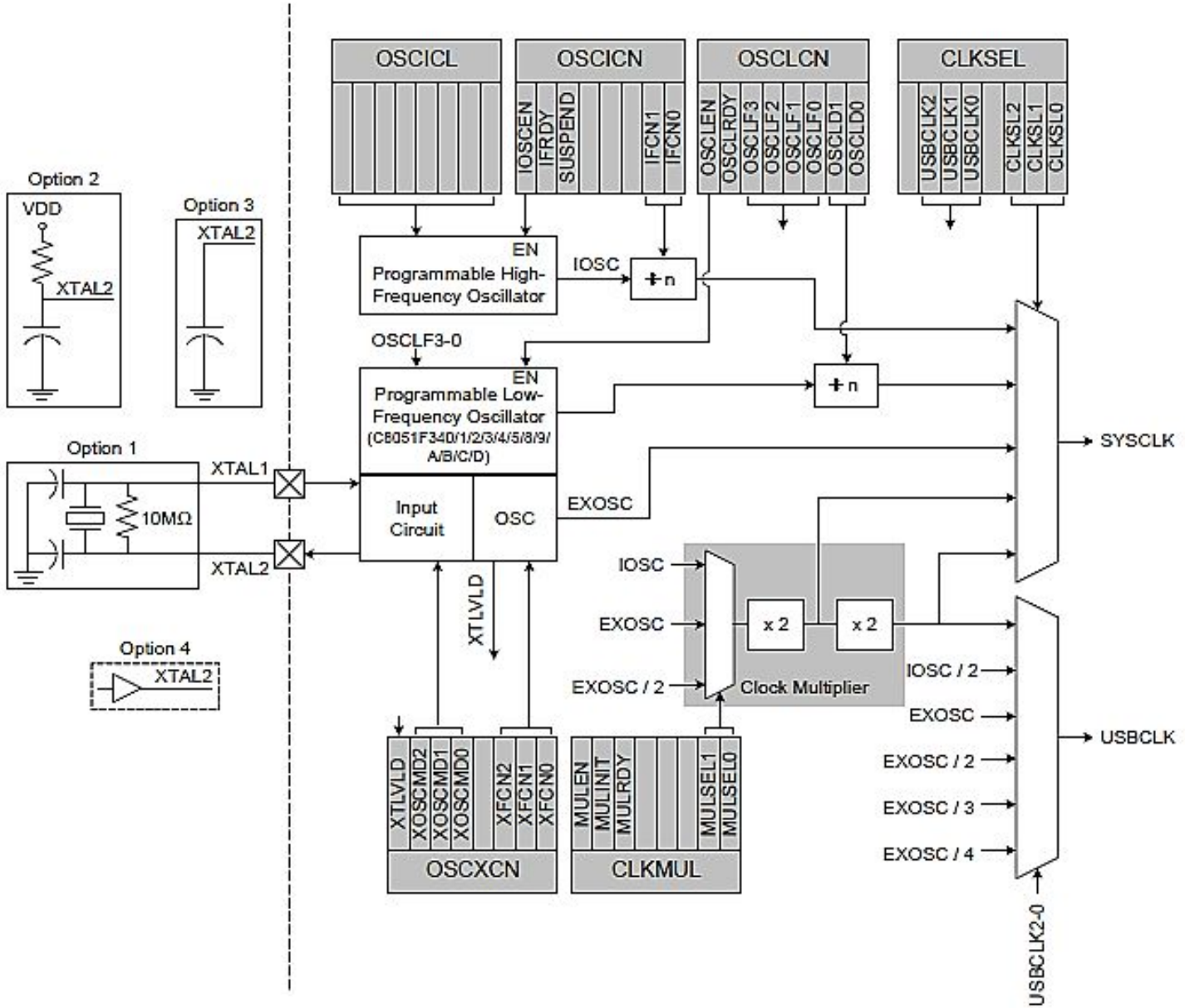
Bit0: PINRSF: HW Pin Reset Flag.
0: Source of last reset was not $\overline{RST}$ pin.
1: Source of last reset was $\overline{RST}$ pin.

Oscillator options

Four Oscillator options

- **Programmable internal high-frequency oscillator**
- **Programmable internal low-frequency oscillator**
- **External oscillator drive circuit**
- **4x Clock Multiplier**.

Oscillator Diagram



Oscillator options

- **The internal high-frequency and low-frequency oscillators can be enabled/disabled and adjusted using the special function registers OSCICN & OSCLCN**
- **The high-speed internal oscillator is factory calibrated to 12 MHz ±1.5%.**
- **The system clock (SYSCLK) can be derived from either of the internal oscillators, the external oscillator circuit, or the 4x Clock Multiplier divided by 2.**
- **The USB clock (USBCLK) can be derived from the internal oscillator, external oscillator, or 4x Clock Multiplier.**

SFR: OSCICN - Internal H-F Oscillator Control

SFR Definition 14.1. OSCICN: Internal H-F Oscillator Control

R/W	R	R/W	R	R/W	R/W	R/W	R/W	Reset Value
IOSCEN	IFRDY	SUSPEND	-	-	-	IFCN1	IFCN0	10000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xB2

Bit7: IOSCEN: Internal H-F Oscillator Enable Bit.
0: Internal H-F Oscillator Disabled.
1: Internal H-F Oscillator Enabled.

Bit6: IFRDY: Internal H-F Oscillator Frequency Ready Flag.
0: Internal H-F Oscillator is not running at programmed frequency.
1: Internal H-F Oscillator is running at programmed frequency.

Bit5: SUSPEND: Force Suspend
Writing a '1' to this bit will force the internal H-F oscillator to be stopped. The oscillator will be re-started on the next non-idle USB event (i.e., RESUME signaling) or VBUS interrupt event (see SFR Definition 8.1).

Bits4–2: UNUSED. Read = 000b, Write = don't care.

Bits1–0: IFCN1–0: Internal H-F Oscillator Frequency Control.
00: SYSCLK derived from Internal H-F Oscillator divided by 8.
01: SYSCLK derived from Internal H-F Oscillator divided by 4.
10: SYSCLK derived from Internal H-F Oscillator divided by 2.
11: SYSCLK derived from Internal H-F Oscillator divided by 1.

SFR: OSCLCN- Internal L-F Oscillator Control

SFR Definition 14.3. OSCLCN: Internal L-F Oscillator Control

R/W	R	R/W	R	R/W	R/W	R/W	R/W	Reset Value
OSCLEN	OSCLRDY	OSCLF3	OSCLF2	OSCLF1	OSCLF0	OSCLD1	OSCLD0	00vvvv00
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0x86

Bit7: OSCLEN: Internal L-F Oscillator Enable.
0: Internal L-F Oscillator Disabled.
1: Internal L-F Oscillator Enabled.

Bit6: OSCLRDY: Internal L-F Oscillator Ready Flag.
0: Internal L-F Oscillator frequency not stabilized.
1: Internal L-F Oscillator frequency stabilized.

Bits5–2: OSCLF[3:0]: Internal L-F Oscillator Frequency Control bits.
Fine-tune control bits for the internal L-F Oscillator frequency. When set to 0000b, the L-F oscillator operates at its fastest setting. When set to 1111b, the L-F oscillator operates at its slowest setting.

Bits1–0: OSCLD[1:0]: Internal L-F Oscillator Divider Select.
00: Divide by 8 selected.
01: Divide by 4selected.
10: Divide by 2 selected.
11: Divide by 1 selected.

SFR: CLKSEL - Clock Select

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	USBCLK			-	CLKSL			00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address
								0xA9

Bit 7: Unused. Read = 0b; Write = don't care.

Bits6–4: USBCLK2–0: USB Clock Select

These bits select the clock supplied to USB0. When operating USB0 in full-speed mode, the selected clock should be 48 MHz. When operating USB0 in low-speed mode, the selected clock should be 6 MHz.

USBCLK	Selected Clock
000	4x Clock Multiplier
001	Internal Oscillator / 2
010	External Oscillator
011	External Oscillator / 2
100	External Oscillator / 3
101	External Oscillator / 4
110	RESERVED
111	RESERVED

SFR: CLKSEL - Clock Select

Bit3: Unused. Read = 0b; Write = don't care.
Bits2–0: CLKSL2–0: System Clock Select
These bits select the system clock source. When operating from a system clock of 25 MHz or less, the FLRT bit should be set to '0'. When operating with a system clock of greater than 25 MHz (up to 48 MHz), the FLRT bit (FLSCL.4) should be set to '1'. See **Section "10. Prefetch Engine" on page 99** for more details.

CLKSL	Selected Clock
000	Internal Oscillator (as determined by the IFCN bits in register OSCICN)
001	External Oscillator
010	4x Clock Multiplier / 2
011*	4x Clock Multiplier*
100	Low-Frequency Oscillator
101-111	RESERVED
***Note:** This option is only available on 48 MHz devices.	

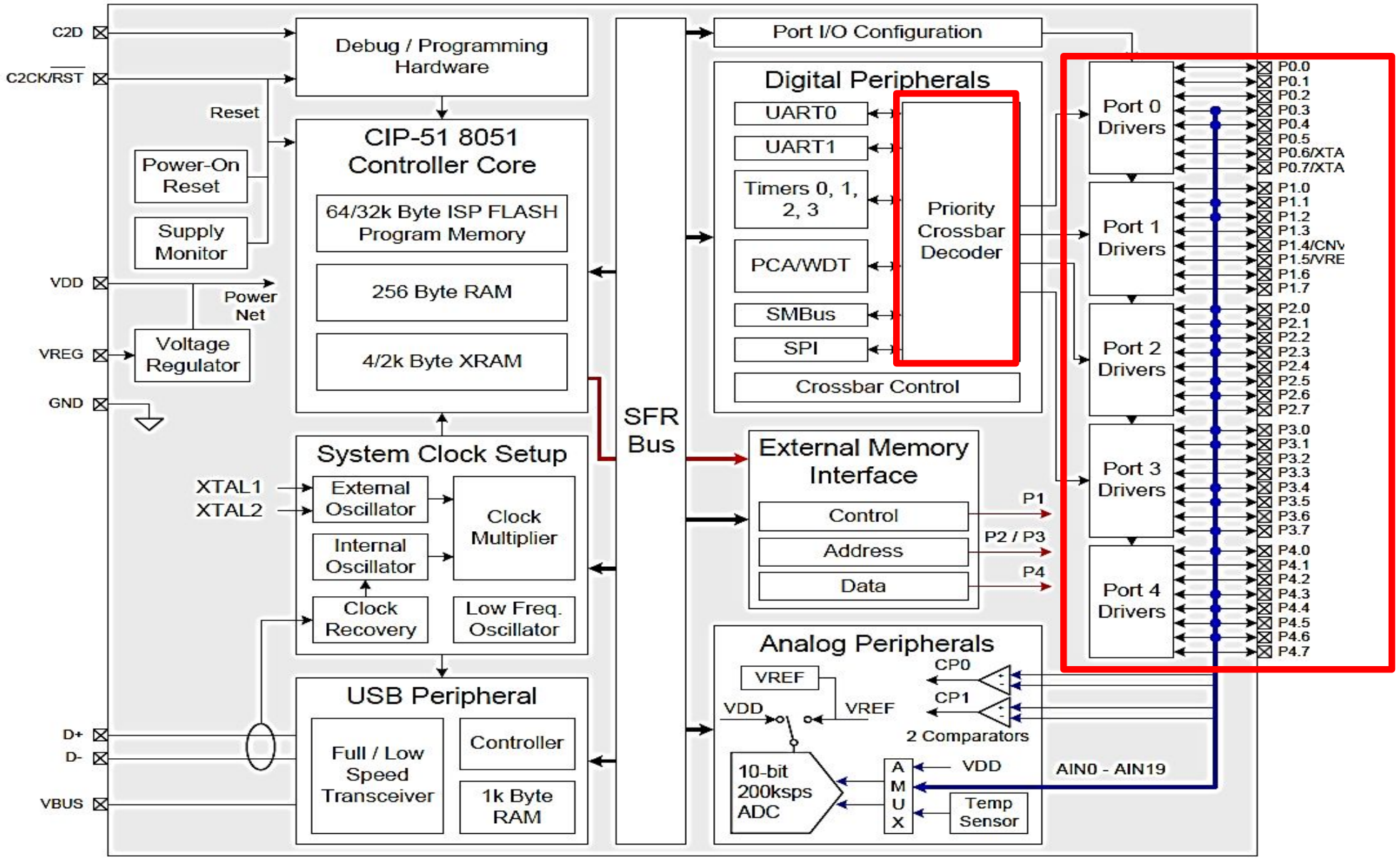
Port structure

- **Five ports - P0 to P4 (8 bit each) – 40 Pins GPIO**

- **Each of the Port pins can be defined as**

  - **Digital I/O- general-purpose I/O (GPIO)**
  - **Analog input**
  - **Port pins P0.0-P3.7 can be assigned to one of the internal digital resources**

Block Diagram C8051F340 – with highlighted ports



Purpose of the crossbar

- The crossbar is a multiplexer that routes digital peripherals to port pins. It must be properly enabled and configured when using peripherals.
- Using Crossbar, internal digital signals are mapped to port pins.
- Basically, it gives the designer the flexibility to route desired peripheral to port pin.

Ex. If we want a UART devices that needs to connect to pins P0.4 and P0.5. The crossbar allows you to bring that signal out to that pin or a different supported pin in another design.

Crossbar

- Problem: Many peripherals/functions are available inside the MCU
  - There are limited number of pins available to connect the peripherals to the outside world

- Solution: Pick and choose the peripherals that are necessary for an application, and assign only those to external pins
  - This is the function of the crossbar
  - Based on the application, the system designer makes the decision as to which peripherals are enabled, and which pins are used

- The C8051F340 has a rich set of digital resources like UARTs, system management bus (SMBus), timer control inputs and interrupts
  - These peripherals do not have dedicated pins through which they may be accessed
  - They are available through the four lower I/O ports (P0, P1, P2 and P3)
  - Each of the pins on P0, P1, P2 and P3 can be defined as a general purpose input/output (GPIO) pin or can be assigned to a digital peripheral
  - Lower ports have dual functionalities
  - This flexibility makes the MCU very versatile

Crossbar Pin Assignment and Allocation Priority

- The crossbar has a priority order in which peripherals are assigned to pins
  - UART0 has the highest priority and UART1 has the lowest priority
- There are three configuration registers, XBR0, XBR1 and XBR2, which are programmed to accomplish the pin allocations
- If the corresponding enable bits of the peripheral are set to a logic 1 in the crossbar registers, then the port pins are assigned to that peripheral
- Pin assignments to associated functions are done in groups
  - For example, TX0 and RX0 for UART0 are assigned together

- Example: If the UART0EN bit (XBR0.0) is set to logic 1, the TX0 and RX0 pins will be mapped to the port pins P0.4 and P0.5, respectively.
  - Since UART0 has the highest priority, its pins will always be mapped to P0.4 and P0.5 when UART0EN is set to logic 1 and will have precedence over any other peripheral allocation

XBR0, XBR1, XBR2

SFR Definition 15.1. XBR0: Port I/O Crossbar Register 0

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
CP1AE	CP1E	CP0AE	CP0E	SYSCKE	SMB0E	SPI0E	URT0E	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE1

SFR Definition 15.2. XBR1: Port I/O Crossbar Register 1

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
WEAKPUD	XBARE	T1E	T0E	ECIE	PCA0ME			00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE2

SFR Definition 15.3. XBR2: Port I/O Crossbar Register 2

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
							URT1E	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE3

XBR0 (Crossbar Register 0)

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
CP1AE	CP1E	CP0AE	CP0E	SYSCKE	SMB0E	SPI0E	URT0E	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0xE1

Bit7: CP1AE: Comparator1 Asynchronous Output Enable
0: Asynchronous CP1 unavailable at Port pin.
1: Asynchronous CP1 routed to Port pin.

Bit6: CP1E: Comparator1 Output Enable
0: CP1 unavailable at Port pin.
1: CP1 routed to Port pin.

Bit5: CP0AE: Comparator0 Asynchronous Output Enable
0: Asynchronous CP0 unavailable at Port pin.
1: Asynchronous CP0 routed to Port pin.

Bit4: CP0E: Comparator0 Output Enable
0: CP0 unavailable at Port pin.
1: CP0 routed to Port pin.

Bit3: SYSCKE: /SYSCLK Output Enable
0: /SYSCLK unavailable at Port pin.
1: /SYSCLK output routed to Port pin.

Bit2: SMB0E: SMBus I/O Enable
0: SMBus I/O unavailable at Port pins.
1: SMBus I/O routed to Port pins.

Bit1: SPI0E: SPI I/O Enable
0: SPI I/O unavailable at Port pins.
1: SPI I/O routed to Port pins.

Bit0: URT0E: UART0 I/O Output Enable
0: UART0 I/O unavailable at Port pins.
1: UART0 TX0, RX0 routed to Port pins P0.4 and P0.5.

XBR1 (Crossbar Register 1)

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
WEAKPUD	XBARE	T1E	T0E	ECIE	PCA0ME			00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE2

Bit7: WEAKPUD: Port I/O Weak Pull-up Disable.
0: Weak Pull-ups enabled (except for Ports whose I/O are configured as analog input or push-pull output).
1: Weak Pull-ups disabled.

Bit6: XBARE: Crossbar Enable.
0: Crossbar disabled; all Port drivers disabled.
1: Crossbar enabled.

Bit5: T1E: T1 Enable
0: T1 unavailable at Port pin.
1: T1 routed to Port pin.

Bit4: T0E: T0 Enable
0: T0 unavailable at Port pin.
1: T0 routed to Port pin.

Bit3: ECIE: PCA0 External Counter Input Enable
0: ECI unavailable at Port pin.
1: ECI routed to Port pin.

Bits2–0: PCA0ME: PCA Module I/O Enable Bits.
000: All PCA I/O unavailable at Port pins.
001: CEX0 routed to Port pin.
010: CEX0, CEX1 routed to Port pins.
011: CEX0, CEX1, CEX2 routed to Port pins.
100: CEX0, CEX1, CEX2, CEX3 routed to Port pins.
101: CEX0, CEX1, CEX2, CEX3, CEX4 routed to Port pins.
110: Reserved.
111: Reserved.

XBR2 (Crossbar Register 2)

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
							URT1E	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE3

Bits7–1: RESERVED: Always write to 0000000b
Bit0: URT1E: UART1 I/O Output Enable (C8051F340/1/4/5/8/A/B Only)
0: UART1 I/O unavailable at Port pins.
1: UART1 TX1, RX1 routed to Port pins.

Port structure

- Designer has complete control over which functions are assigned, limited only by the number of physical I/O pins.
- The Digital Crossbar allows mapping of internal digital system resources to Port I/O pins. On-chip counter/timers, serial buses, HW interrupts, comparator outputs, and other digital signals in the controller can be configured to appear on the Port I/O pins specified in the Crossbar Control registers. This allows the user to select the exact mix of general purpose Port I/O and digital resources needed for the end application.
- This resource assignment flexibility is achieved through the use of a Priority Crossbar Decoder.
- State of a Port I/O pin can always be read in the corresponding Port latch, regardless of the Crossbar settings.
- The Crossbar assigns the selected internal digital resources to the I/O pins based on the Priority Decoder
- The registers XBR0, XBR1, and XBR2 are defined in SFR definition
- These SFRs are used to select internal digital functions

Port I/O -Functional block diagram

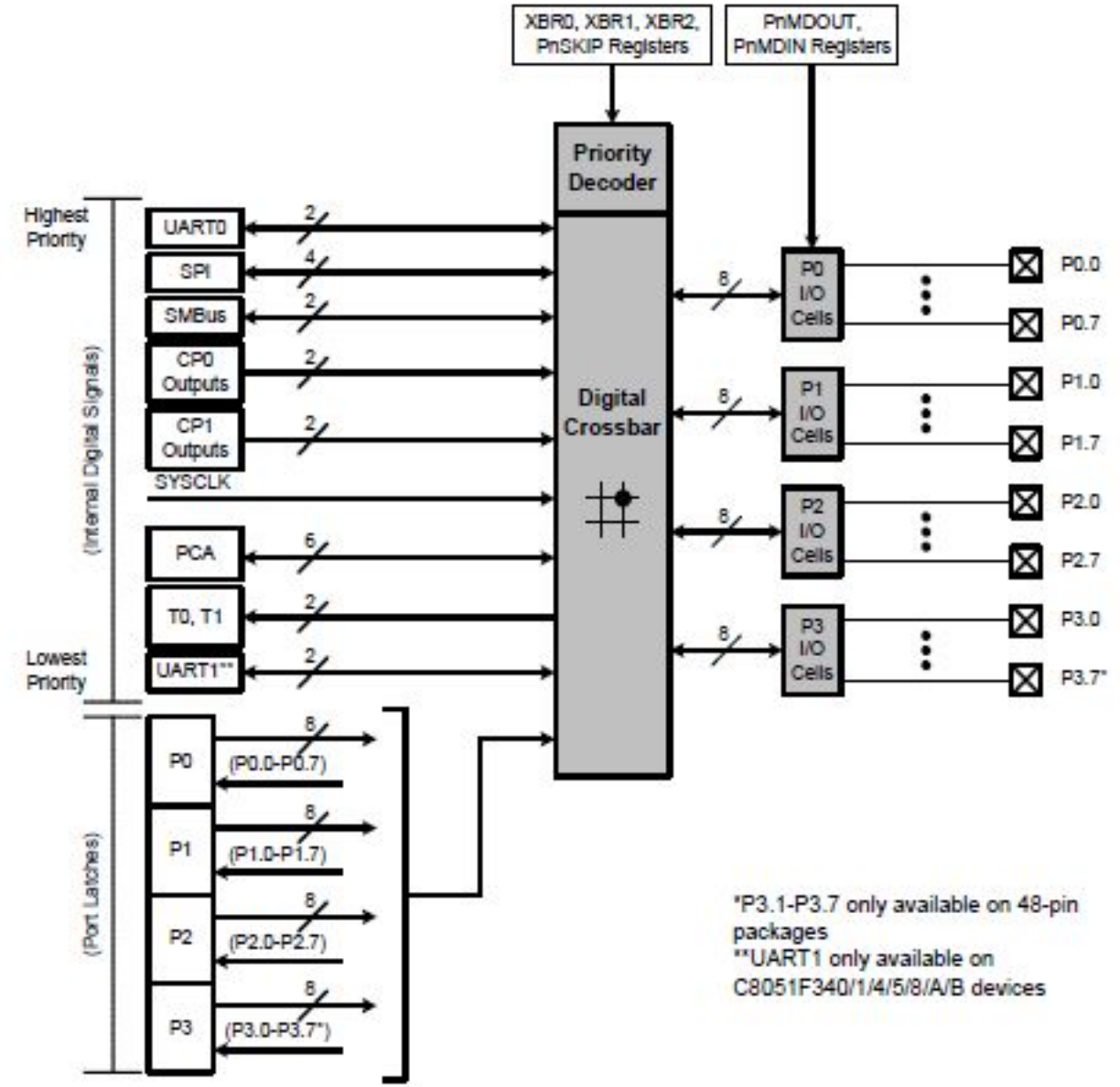


Figure 15.1. Port I/O Functional Block Diagram (Port 0 through Port 3)

Priority Crossbar Decoder

- The Priority Crossbar Decoder assigns a priority to each I/O function, starting at the top with UART0. When a digital resource is selected, the least-significant unassigned Port pin is assigned to that resource (excluding UART0, which is always at pins 4 and 5).

- If a Port pin is assigned, the Crossbar skips that pin when assigning the next selected resource. Additionally, the Crossbar will skip Port pins whose associated bits in the PnSKIP registers are set.

- The PnSKIP registers allow software to skip Port pins that are to be used for analog input, dedicated functions, or GPIO.

Important Note on Crossbar Configuration: If a Port pin is claimed by a peripheral without use of the Crossbar, its corresponding PnSKIP bit should be set.

The Crossbar must be enabled to use Ports P0, P1, P2, and P3 as standard Port I/O in output mode. These Port output drivers are disabled while the Crossbar is disabled.

Port 4 always functions as standard GPIO.

Port I/O Initialization

- Port I/O initialization consists of the following steps:

Step 1. Select the input mode (analog or digital) for all Port pins, using the Port Input Mode register (PnMDIN).

Step 2. Select the output mode (open-drain or push-pull) for all Port pins, using the Port Output Mode register (PnMDOUT).

Step 3. Select any pins to be skipped by the I/O Crossbar using the Port Skip registers (PnSKIP).

Step 4. Assign Port pins to desired peripherals (XBR0, XBR1).

Step 5. Enable the Crossbar (XBARE = '1').

Important points to configure the crossbar

- All pins have been properly set in the PxMDOUT registers to be configured to digital input or output

- Pins set to be ADC inputs are skipped by the crossbar using the PxSKIP register and configured to be analog inputs in the PxMDIN register. Also, a '1' should be written into the pin's corresponding bit in the Px register

- If using SPI, on certain devices there is a 3-wire mode option. Check that the peripheral is set to 3- or 4- wire mode before configuring the crossbar, since 3-wire mode will yeild a different pin assignment than 4-wire mode for peripherals of lower priority than SPI

Port pins must be configured as either analog or digital inputs

- **Analog Input-**

Any pins to be used as Comparator or ADC inputs should be configured as an analog inputs. When a pin is configured as an analog input, its weak pull-up, digital driver, and digital receiver are disabled. This process saves power and reduces noise on the analog input.

Additionally, all analog input pins should be configured to be skipped by the Crossbar (accomplished by setting the associated bits in PnSKIP).

- **Digital input-**

Port input mode is set in the PnMDIN register, where a '1' indicates a digital input, and a '0' indicates an analog input.

All pins default to digital inputs on reset.

General Purpose Port I/O

- Port pins that remain unassigned by the Crossbar and are not used by analog peripherals can be used for general purpose I/O.
- Ports 3-0 are accessed through corresponding special function registers (SFRs) that are both byte addressable and bit addressable.
- Port 4 uses an SFR which is byte-addressable.
- When writing to a Port, the value written to the SFR is latched to maintain the output data value at each pin.
- When reading, the logic levels of the Port's input pins are returned regardless of the XBRn settings

Port0 Latch

SFR Definition 15.4. P0: Port0 Latch

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	11111111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0x80

Bits7–0: P0.[7:0]
Write - Output appears on I/O pins per Crossbar Registers (when XBARE = '1').
0: Logic Low Output.
1: Logic High Output (high impedance if corresponding P0MDOUT.n bit = 0).
Read - Always reads '0' if selected as analog input in register P0MDIN. Directly reads Port pin when configured as digital input.
0: P0.n pin is logic low.
1: P0.n pin is logic high.

Port0 Input Mode

SFR Definition 15.5. P0MDIN: Port0 Input Mode

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								11111111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xF1

Bits7–0: Analog Input Configuration Bits for P0.7–P0.0 (respectively).
Port pins configured as analog inputs have their weak pull-up, digital driver, and digital receiver disabled.
0: Corresponding P0.n pin is configured as an analog input.
1: Corresponding P0.n pin is not configured as an analog input.

Port0 Output Mode

SFR Definition 15.6. P0MDOUT: Port0 Output Mode

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xA4

Bits7–0: Output Configuration Bits for P0.7–P0.0 (respectively): ignored if corresponding bit in register P0MDIN is logic 0.
0: Corresponding P0.n Output is open-drain.
1: Corresponding P0.n Output is push-pull.

(Note: When SDA and SCL appear on any of the Port I/O, each are open-drain regardless of the value of P0MDOUT).

Port0 Skip

SFR Definition 15.7. P0SKIP: Port0 Skip

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xD4

Bits7–0: P0SKIP[7:0]: Port0 Crossbar Skip Enable Bits.
These bits select Port pins to be skipped by the Crossbar Decoder. Port pins used as analog inputs (for ADC or Comparator) or used as special functions (VREF input, external oscillator circuit, CNVSTR input) should be skipped by the Crossbar.
0: Corresponding P0.n pin is not skipped by the Crossbar.
1: Corresponding P0.n pin is skipped by the Crossbar.

Port SFRs

- **Port 0 to Port 3**
  - **Portx Latch**
  - **Portx Input Mode**
  - **Portx Output Mode**
  - **Portx Skip**

- **Port 4 (Not assigned to crossbar)**
  - **Port4 Latch**
  - **Port4 Input Mode**
  - **Port4 Output Mode**

Timers in C8051F340 Microcontroller

Timers

- **Timers are used for: interval timing, event counting or baud rate generation**
- **In interval timing applications, a timer is programmed to overflow at a regular interval and the following:**
  - **Set the timer overflow flag or**
  - **Generate an interrupt**
  - **This can also be used to generate waveforms at set frequencies**
- **Event counting is used to determine the number of occurrences of an event, rather than to measure the elapsed time between events. In this case, the timer functions as a counter.**
  - **An "event" is any external stimulus that provides a high-to-low transition at the selected input pin**
- **The timers can also function as the baud rate generators for the C8051F340's internal serial ports (UART0 and UART1)**
  - **"Baud rate" is the bit rate of the serial port (the time period of a bit)**

Timers

- **CIP51 has four counter/timers: Timer 0, Timer 1, Timer 2 and Timer 3**
- **Timer 0 and Timer 1 are nearly identical and have four primary modes of operation.**
- **Timer 2 and Timer 3 offer 16-bit and split 8-bit timer functionality with auto-reload.**

Timer 0 and Timer 1 Modes:	Timer 2 Modes:	Timer 3 Modes:
13-bit counter/timer	16-bit timer with auto-reload	16-bit timer with auto-reload
16-bit counter/timer		
8-bit counter/timer with auto-reload	Two 8-bit timers with auto-reload	Two 8-bit timers with auto-reload
Two 8-bit counter/timers (Timer 0 only)		

Timer 0 & 1 - Mode 1: 16bit Counter/Timer

- Mode 1 configures Timer 0 and Timer 1 to operate as 16-bit counter/timers.

- TH0 and TL0 holds the count. When the count in TH0 and TL0 overflows from all ones to 0x00, the timer overflow flag is set. The count in TH0 and TL0 should be loaded for next iteration.

- If Timer 0 interrupts are enabled, an interrupt will occur when the TF0 flag is set. The count in TH0 and TL0 should be loaded for next iteration.

Timer 0 Mode 1 Block Diagram

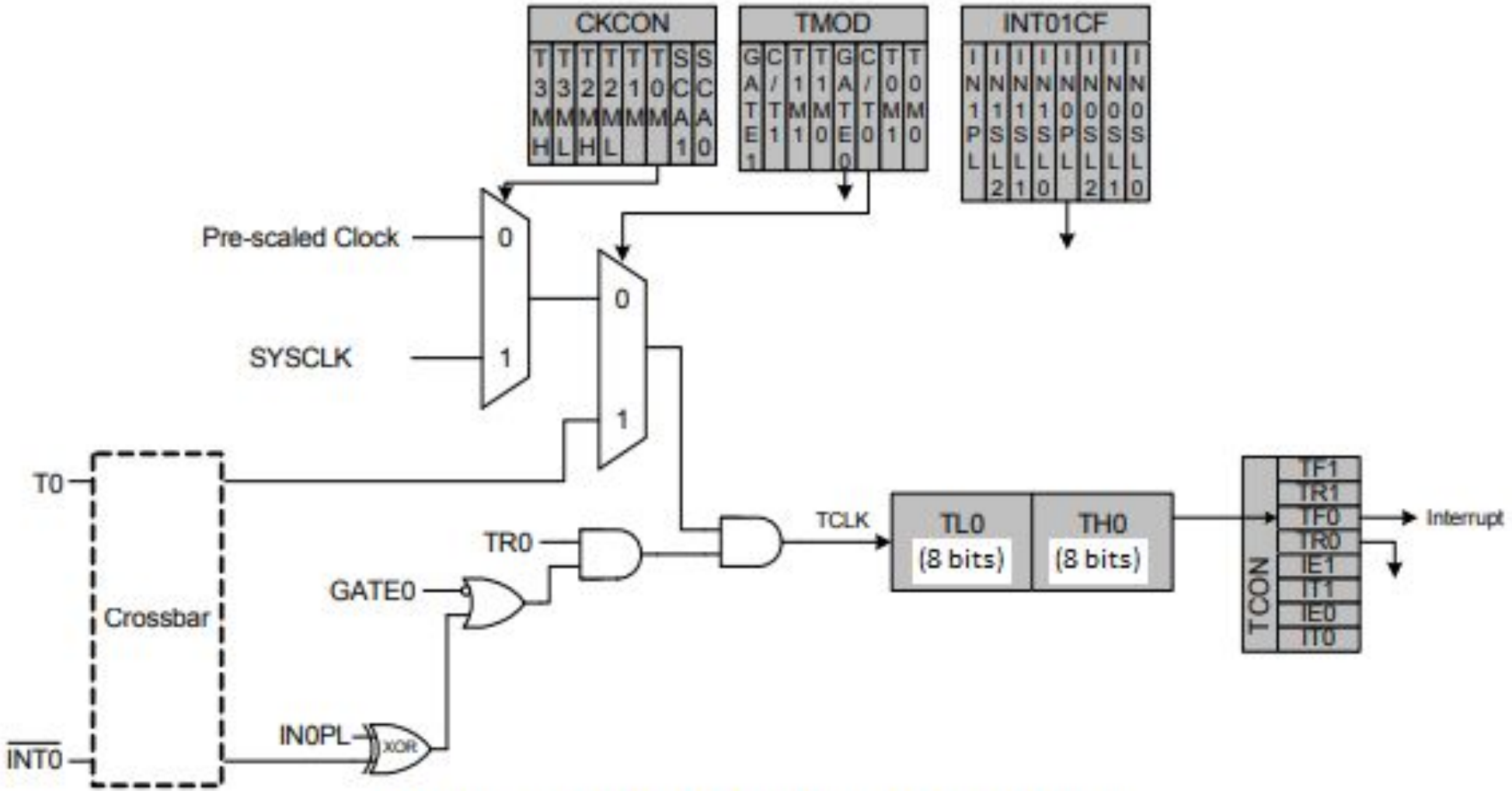


Figure 21.1. T0 Mode 0 Block Diagram

Timer 0 & 1 Mode 2: 8bit Counter/Timer with Auto-Reload

- Mode 2 configures Timer 0 and Timer 1 to operate as 8-bit counter/timers with automatic reload of the start value.

- TL0 holds the count and TH0 holds the reload value. When the counter in TL0 overflows from all ones to 0x00, the timer overflow flag TF0 (TCON.5) is set and the counter in TL0 is reloaded from TH0.

- If Timer 0 interrupts are enabled, an interrupt will occur when the TF0 flag is set. The reload value in TH0 is not changed. TL0 must be initialized to the desired value before enabling the timer for the first count to be correct. When in Mode 2, Timer 1 operates identically to Timer 0.

Timer 0 Mode 2 Block Diagram

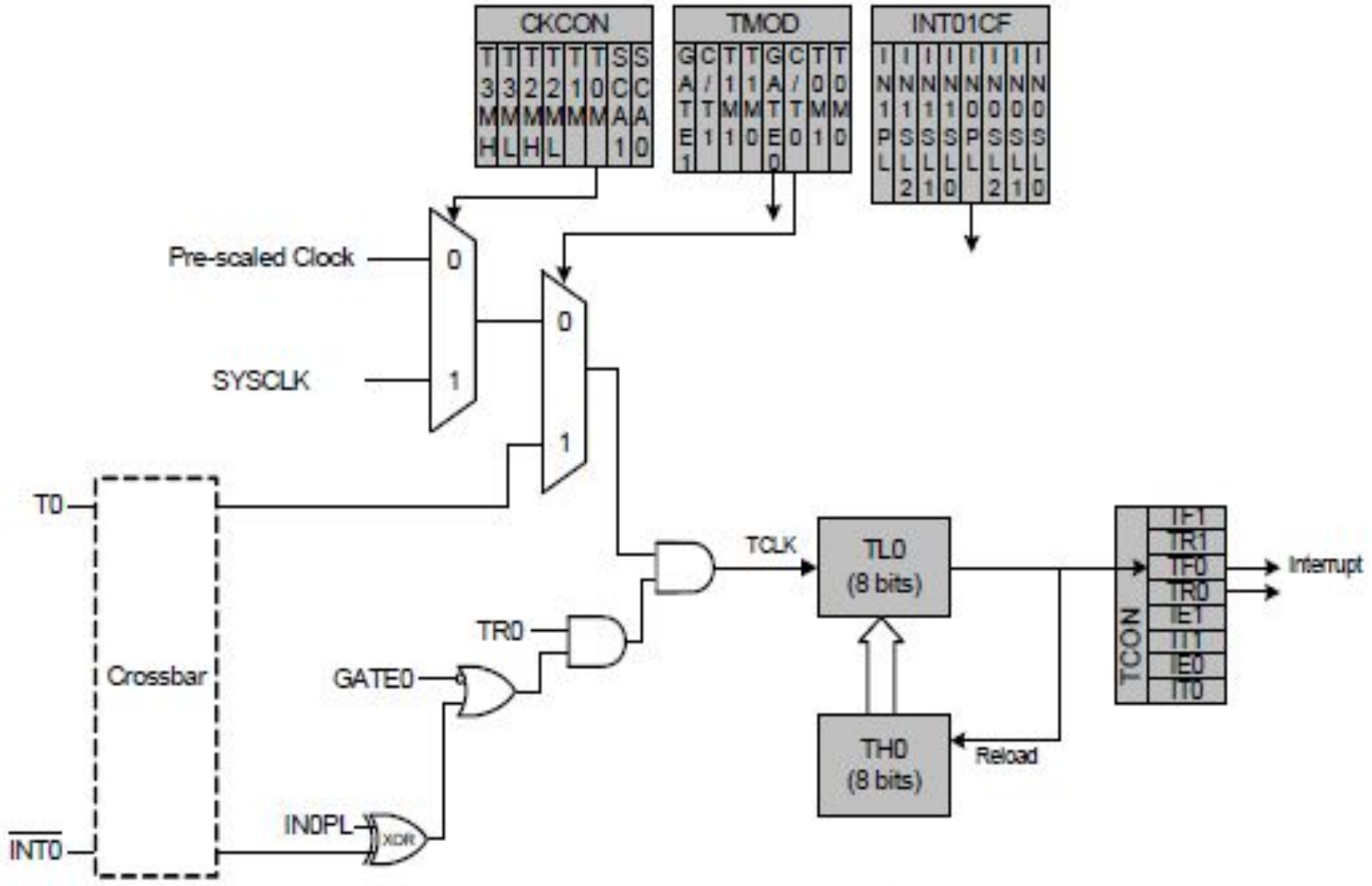


Figure 21.2. T0 Mode 2 Block Diagram

Steps to program 8-Bit Auto-Reload Mode (Mode 2)

- **This mode configures Timers 0 (and 1) to operate as 8-bit counter/timers with automatic reload of the start value**
- **The timer low byte (TLx) operates as an 8-bit timer while the timer high byte (THx) holds a reload value**
- **When the count in TLx overflows from FFH to 00H, the timer flag is set and the value in THx is automatically loaded into TLx**
- **Counting continues from the reload value up to the next FFH overflow, and so on**
- **This mode is convenient for creating regular periodic intervals, as the timer overflows at the same rate once TMOD and THx are initialized**
- **TLx must be initialized to the desired value before enabling the timer for the first count to be correct**
- **Timer 1 can be used as an 8-bit baud rate generator for UART0 and/or UART1 in mode 2**

SFR- CKCON: Clock Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
T3MH	T3ML	T2MH	T2ML	T1M	T0M	SCA1	SCA0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x8E

Bit7: T3MH: Timer 3 High Byte Clock Select.
This bit selects the clock supplied to the Timer 3 high byte if Timer 3 is configured in split 8-bit timer mode. T3MH is ignored if Timer 3 is in any other mode.
0: Timer 3 high byte uses the clock defined by the T3XCLK bit in TMR3CN.
1: Timer 3 high byte uses the system clock.

Bit6: T3ML: Timer 3 Low Byte Clock Select.
This bit selects the clock supplied to Timer 3. If Timer 3 is configured in split 8-bit timer mode, this bit selects the clock supplied to the lower 8-bit timer.
0: Timer 3 low byte uses the clock defined by the T3XCLK bit in TMR3CN.
1: Timer 3 low byte uses the system clock.

Bit5: T2MH: Timer 2 High Byte Clock Select.
This bit selects the clock supplied to the Timer 2 high byte if Timer 2 is configured in split 8-bit timer mode. T2MH is ignored if Timer 2 is in any other mode.
0: Timer 2 high byte uses the clock defined by the T2XCLK bit in TMR2CN.
1: Timer 2 high byte uses the system clock.

Bit4: T2ML: Timer 2 Low Byte Clock Select.
This bit selects the clock supplied to Timer 2. If Timer 2 is configured in split 8-bit timer mode, this bit selects the clock supplied to the lower 8-bit timer.
0: Timer 2 low byte uses the clock defined by the T2XCLK bit in TMR2CN.
1: Timer 2 low byte uses the system clock.

SFR- CKCON: Clock Control

Bit3: T1M: Timer 1 Clock Select.
This select the clock source supplied to Timer 1. T1M is ignored when C/T1 is set to logic 1.
0: Timer 1 uses the clock defined by the prescale bits, SCA1-SCA0.
1: Timer 1 uses the system clock.

Bit2: T0M: Timer 0 Clock Select.
This bit selects the clock source supplied to Timer 0. T0M is ignored when C/T0 is set to logic 1.
0: Counter/Timer 0 uses the clock defined by the prescale bits, SCA1-SCA0.
1: Counter/Timer 0 uses the system clock.

Bits1–0: SCA1-SCA0: Timer 0/1 Prescale Bits.
These bits control the division of the clock supplied to Timer 0 and/or Timer 1 if configured to use prescaled clock inputs.

SCA1	SCA0	Prescaled Clock
0	0	System clock divided by 12
0	1	System clock divided by 4
1	0	System clock divided by 48
1	1	External clock divided by 8

Note: External clock divided by 8 is synchronized with the system clock.

SFR TMOD: Timer Mode

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
GATE1	C/T1	T1M1	T1M0	GATE0	C/T0	T0M1	T0M0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0x89

Bit7: GATE1: Timer 1 Gate Control.
0: Timer 1 enabled when TR1 = 1 irrespective of $\overline{\text{INT1}}$ logic level.
1: Timer 1 enabled only when TR1 = 1 AND $\overline{\text{INT1}}$ is active as defined by bit IN1PL in register INT01CF (see SFR Definition 9.13).

Bit6: C/T1: Counter/Timer 1 Select.
0: Timer Function: Timer 1 incremented by clock defined by T1M bit (CKCON.3).
1: Counter Function: Timer 1 incremented by high-to-low transitions on external input pin (T1).

Bits5–4: T1M1–T1M0: Timer 1 Mode Select.
These bits select the Timer 1 operation mode.

T1M1	T1M0	Mode
0	0	Mode 0: 13-bit counter/timer
0	1	Mode 1: 16-bit counter/timer
1	0	Mode 2: 8-bit counter/timer with auto-reload
1	1	Mode 3: Timer 1 inactive

SFR TMOD: Timer Mode

Bit3: GATE0: Timer 0 Gate Control.
0: Timer 0 enabled when TR0 = 1 irrespective of $\overline{\text{INT0}}$ logic level.
1: Timer 0 enabled only when TR0 = 1 AND $\overline{\text{INT0}}$ is active as defined by bit IN0PL in register INT01CF (see SFR Definition 9.13).

Bit2: C/T0: Counter/Timer Select.
0: Timer Function: Timer 0 incremented by clock defined by T0M bit (CKCON.2).
1: Counter Function: Timer 0 incremented by high-to-low transitions on external input pin (T0).

Bits1–0: T0M1–T0M0: Timer 0 Mode Select.
These bits select the Timer 0 operation mode.

T0M1	T0M0	Mode
0	0	Mode 0: 13-bit counter/timer
0	1	Mode 1: 16-bit counter/timer
1	0	Mode 2: 8-bit counter/timer with auto-reload
1	1	Mode 3: Two 8-bit counter/timers

SFR TCON: Timer Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
						(bit addressable)		0x88

Bit7: TF1: Timer 1 Overflow Flag.
Set by hardware when Timer 1 overflows. This flag can be cleared by software but is automatically cleared when the CPU vectors to the Timer 1 interrupt service routine.
0: No Timer 1 overflow detected.
1: Timer 1 has overflowed.

Bit6: TR1: Timer 1 Run Control.
0: Timer 1 disabled.
1: Timer 1 enabled.

Bit5: TF0: Timer 0 Overflow Flag.
Set by hardware when Timer 0 overflows. This flag can be cleared by software but is automatically cleared when the CPU vectors to the Timer 0 interrupt service routine.
0: No Timer 0 overflow detected.
1: Timer 0 has overflowed.

Bit4: TR0: Timer 0 Run Control.
0: Timer 0 disabled.
1: Timer 0 enabled.

SFR TCON: Timer Control

Bit3: IE1: External Interrupt 1.
This flag is set by hardware when an edge/level of type defined by IT1 is detected. It can be cleared by software but is automatically cleared when the CPU vectors to the External Interrupt 1 service routine if IT1 = 1. When IT1 = 0, this flag is set to '1' when $\overline{INT1}$ is active as defined by bit IN1PL in register INT01CF (see SFR Definition 9.13).

Bit2: IT1: Interrupt 1 Type Select.
This bit selects whether the configured $\overline{INT1}$ interrupt will be edge or level sensitive. $\overline{INT1}$ is configured active low or high by the IN1PL bit in the IT01CF register (see SFR Definition 9.13).
0: $\overline{INT1}$ is level triggered.
1: $\overline{INT1}$ is edge triggered.

Bit1: IE0: External Interrupt 0.
This flag is set by hardware when an edge/level of type defined by IT0 is detected. It can be cleared by software but is automatically cleared when the CPU vectors to the External Interrupt 0 service routine if IT0 = 1. When IT0 = 0, this flag is set to '1' when $\overline{INT0}$ is active as defined by bit IN0PL in register INT01CF (see SFR Definition 9.13).

Bit0: IT0: Interrupt 0 Type Select.
This bit selects whether the configured $\overline{INT0}$ interrupt will be edge or level sensitive. $\overline{INT0}$ is configured active low or high by the IN0PL bit in register IT01CF (see SFR Definition 9.13).
0: $\overline{INT0}$ is level triggered.
1: $\overline{INT0}$ is edge triggered.

Configure TMOD register for the following:

- Timer 1 in mode 1 -

Gate1	C/T1	T1M1	T1M0	Gate0	C/T0	T0M1	T0M0

- Timer 1 in mode 2 –

Gate1	C/T1	T1M1	T1M0	Gate0	C/T0	T0M1	T0M0

- Timer 0 in mode 1 –
- Timer 0 in mode 2 –
- Timer 1 in mode 2 and Timer 0 in mode 1 –

Gate1	C/T1	T1M1	T1M0	Gate0	C/T0	T0M1	T0M0

Configure TMOD register for the following:

- Timer 1 in mode 1 - 10H

Gate1	C/T1	T1M1	T1M0	Gate0	C/T0	T0M1	T0M0
0	0	0	1	0	0	0	0

- Timer 1 in mode 2 – 20H

Gate1	C/T1	T1M1	T1M0	Gate0	C/T0	T0M1	T0M0
0	0	1	0	0	0	0	0

- Timer 0 in mode 1 – 01H
- Timer 0 in mode 2 – 02H
- Timer 1 in mode 2 and Timer 0 in mode 1 – 21H

Gate1	C/T1	T1M1	T1M0	Gate0	C/T0	T0M1	T0M0
0	0	1	0	0	0	0	1

Procedure for **Time to Count** Calculation – For Timer in Mode 1

Time period of 1clock cycle = (1/SystemClock)

Count = (Required delay/Time period of 1 clock cycle)

Value to be loaded in timer register,
Value (in decimal)= 65536-Count

Convert Value from decimal to hex (YYXX)

Load YY in THx, and XX in TLx register

Example for Time to Count Calculation for Timer in Mode 1

Delay Required= 3mSec

System Clock = 12 MHz

Required Delay = 3 msec

Time period of 1 clock cycle = $\frac{1}{12\,MHz}$ = 83.3 n sec

Count = $\frac{3\,msec}{83.3\,n\,sec}$ = 36014.4

Value = 65536 − 36014

= $(29522)_d$

= $(7352)_{hex}$

YY → THx

XX → TLx

Example for Time to Count Calculation for Timer in Mode 1

Delay Required= 2mSec

Example for Time to Count Calculation for Timer in Mode 1

Delay Required= 2mSec

System clock = 10 MHz.

Required Delay = 2 mSec

Time Period of 1 clock cycle = 0.1 MSec

$$\text{Count} = \frac{2\,\text{msec}}{0.1\,\text{MSec}} = 20{,}000$$

Value = 65536 - 20,000

$= (45536)_d$

$= (B1E0)_H$.

THx TLx.

Procedure for **Time to Count** Calculation– For Timer in Mode 2

Time period of 1clock cycle = (1/SystemClock)

Count = (Required delay/Time period of 1 clock cycle)

Value to be loaded in timer register,
Value (in decimal)= 256-Count

Convert Value from decimal to hex (XX)

Load XX in THx register

Example for Time to Count Calculation for Timer in Mode2

Delay Required= 20uSec

System clock = 12 MHz

Required Delay = 2 mSec

Time period of 1 clock cycle = $\frac{1}{12\ MHz}$ = 83.3 nSec

$$Count = \frac{20uSec}{83.3\ nSec} = 240.096$$

Value = 256 − 240

= (16)d

= $(10)_{Hex}$

↓

XX

↓

THx

Procedure for **Count to Time** Calculation – For Timer in Mode 1

Calculation of Count to time.

Count in TH_x and TL_x = $(YYXX)_{HEX}$

$= (N)_{DEC}$

Value = 65536 − N

Time period = Value × Time period of 1 clock cycle.

Example for **Count to time** calculation – For Mode 1

System Clock = 12 MHz

Time Period of 1 clock cycle = 83.3 nSec

Count in TH0 and TL0 = $(7352)_{Hex}$

$= (29522)_{Dec}$.

Value = 65536 - 29522

= 36014

Time Period = 36014 × 83.3 nSec

$= 2.99 \times 10^{-3}$

= 3 mSec.

Procedure for **Count to Time** Calculation – For Timer in Mode 2

Calculation of Count to time - Mode 2.

Count in THx = $(YY)_{Hex}$

= $(N)_{Dec}$.

Value = 256 - N

Time Period = Value × Time Period of 1 clock cycle

Example for **Count to time** calculation – For Timer in Mode 2

System Clock = 12 MHz

Time Period of 1 clock cycle = 83.3 nsec

Count ~~int~~ in TH0 = $(73)_{Hex}$

$= (115)_{Dec}$

Value = 256 - 115

= 141

Time Period = 141 × 83.3 n Sec

= 11.74 μ Sec

Exercise to be done at home

- Repeat similar such calculations for System Clock = 8MHz, 10MHz, 11.0592MHz and 12MHz considering different values of delay required.

Timer Programming

Steps for Programming Timer in Mode 1

1. Load the TMOD value to timer mode
2. Load the CKCON value to select the clock
3. Load the count in registers TLx and THx
4. Start the timer (SETB TRx)
5. Keep monitoring the timer flag (TFx)
6. Stop the timer (CLR TR0 or CLR TR1)
7. Clear the TF flag
8. Go back to step 3

Write an embedded C program to flash the LEDs connected to Port P4 continuously at an interval of 3mSec. Use Timer 0 in Mode 1 for generating the delay.

System Clock = 12 MHz

Required Delay = 3 msec

Time period of 1 clock cycle = $\frac{1}{12\,MHz}$ = 83.3 n sec

Count = $\frac{3\,msec}{83.3\,n\,sec}$ = 36014.4

Value = 65536 − 36014

= $(29522)_d$

= $(7352)_{hex}$

YY → TH_x

XX → TL_x

Program

```
#include “c8051F340.h”
void Delay();
int main()
{
   P4MDOUT = 0xFF;      // Configure Port4 as output
   TMOD = 0x01;         // Select Timer 0 in Mode 1
   CKCON = 0x04;        // Select System Clock for Timer 0
   while (1)
   {
      P4 = ~P4;         // Compliment Port4 for flashing
      Delay();          // Delay of 3ms between ON and OFF
                           status of LED
   }
   return 0;
}
```

```
void Delay()
{
   TH0 = 0x73; //Load High Byte of Count in TH0
   TL0 = 0x52;       //Load Low Byte of Count in TL0
   TR0 = 1;      //Start the Timer
    while (TF0 ==0 ); //Wait for overflow
    TF0 = 0;      //Clear overflow flag
    TR0 = 0;      //Stop the Timer
}
```

Write an Embedded C program to generate a square wave of 1KHz on P1.0 using Timer 1 in Mode 1. Assume system clock = 12MHz.

Calculations:

System clock = 12 MHz

Time Period for 1 clock cycle = $\frac{1}{12\,MHz}$ = 83.3 nsec

Square wave frequency = 1KHz

Time Period for One clock cycle (Square wave) = $\frac{1}{1\,KHz}$

$T = T_{ON} + T_{OFF} = 1\,msec$

For Square wave : $T_{ON} = T_{OFF} = \frac{T}{2}$

$\therefore T_{ON} = T_{OFF} = \frac{1\,msec}{2} = 0.5\,msec$

Contd.

Count to be loaded in TH1 & TL1 :

$$\text{Count} = \frac{0.5\ \text{msec}}{83.3\ \text{nsec}} = 6002.4$$

Value = 65536 - 6002

= (59534)d

= (E88E) Hex.

TH1 ← E8

TL1 ← 8E

Program for generating square wave

```
#include “c8051F340.h”
Sbit Pin = P1^0
void Delay();
int main()
{
   P1MDOUT = 0xFF;        // Configure Port1 as output
   XBR1 = 0x40;           // Enable crossbar for Port 1
   TMOD = 0x10;           // Select Timer 1 in Mode 1
   CKCON = 0x08;          // Select System Clock for Timer 1
   while (1)
   {
      Pin = ~Pin;      // Compliment P1.0 for square wave
      Delay();            // Delay of 0.5ms between high and low
                             level of square wave
   }
   return 0;
}
```

```
void Delay()
{
   TH1 = 0xE8; //Load High Byte of Count in TH1
   TL1 = 0x8E;       //Load Low Byte of Count in TL1
   TR1 = 1;      //Start the Timer
    while (TF1 ==0 ); //Wait for overflow
    TF1 = 0;      //Clear overflow flag
    TR1 = 0;      //Stop the Timer
}
```

Steps to Programming Timer in Mode 2

1. Load the TMOD value to timer mode
2. Load the CKCON value to select the clock
3. Load the count in register THx
4. Start the timer (SETB TRx)
5. Keep monitoring the timer flag (TFx)
6. Stop the timer (CLR TR0 or CLR TR1)
7. Clear the TF flag
8. Go back to step 4

Write an embedded C program to flash the LEDs connected to Port P4 continuously at an interval of 20uSec. Use Timer 1 in Mode 2 for generating the delay.

System clock = 12 MHz

Required Delay = 2 mSec

Time period of 1 clock cycle = $\frac{1}{12\ MHz}$ = 83.3 nSec

Count = $\frac{20uSec}{83.3\ nSec}$ = 240.096

Value = 256 − 240

= (16)d

= $(10)_{Hex}$

↓

XX

↓

THx

Program

```
#include “c8051F340.h”
void Delay();
int main()
{
   P4MDOUT = 0xFF;      // Configure Port4 as output
   TMOD = 0x20;         // Select Timer 1 in Mode 2
   CKCON = 0x08;        // Select System Clock for Timer 1
   TH1 = 0x10;          //Load High Byte of Count in TH1
   while (1)
   {
      P4 = ~P4;         // Compliment Port4 for flashing
      Delay();          // Delay of 3ms between ON and OFF
                           status of LED
   }
   return 0;
}
```

```
void Delay()
{
    TR1 = 1;      //Start the Timer
    while (TF1 ==0 ); //Wait for overflow
    TF1 = 0;      //Clear overflow flag
    TR1 = 0;      //Stop the Timer
}
```

Write an Embedded C program to generate a square wave of 25KHz on P2.3 using Timer 1 in Mode 2. Assume system clock = 12MHz.

Calculations:

System clock = 12MHz

Time Period for 1 clock cycle = $\frac{1}{12MHz}$ = 83.3 nSec

Frequency of Square wave = 25 KHz

$\therefore$ Time Period of Square Wave (T) = $\frac{1}{25KHz}$ = 40 MSec

$T = T_{ON} + T_{OFF} = 40MSec$

For Square wave : $T_{ON} = T_{OFF} = \frac{T}{2}$

$\therefore T_{ON} = T_{OFF} = \frac{40MSec}{2} = 20MSec$

Contd.

Count to be loaded in TH_1 :

$$\text{Count} = \frac{20\ \mu sec}{83.3\ n sec} \cong 241$$

$$\text{Value} = 256 - 241 = (15)_d$$

$$= (0F)_{Hex}$$

TH_1

Program

```
#include “c8051F340.h”
Sbit Pin = P2^3
void Delay();
int main()
{
   P2MDOUT = 0xFF;        //Configure Port2 as output
   XBR1 = 0x40;           //Enable crossbar for Port 2
   TMOD = 0x20;           //Select Timer 1 in Mode 2
   CKCON = 0x08;          // Select System Clock for Timer 1
   TH1 = 0x0F;          //Load High Byte of Count in TH1
   while (1)
   {
      Pin = ~Pin;       //Compliment P2.3 for square wave
      Delay();              //Delay of 20uSec between high and
                              low level of square wave
   }
   return 0;
}
```

```
void Delay()
{
    TR1 = 1;      //Start the Timer
    while (TF1 ==0 ); //Wait for overflow
    TF1 = 0;      //Clear overflow flag
    TR1 = 0;      //Stop the Timer
}
```

Introduction to Interrupts

- An interrupt is the occurrence of a condition that causes a temporary suspension of a program while the condition is serviced by another (sub) program

- Interrupts are important because they allow a system to respond asynchronously to an event and deal with the event while in the middle of performing another task

- An **interrupt driven system** gives the *illusion* of doing many things simultaneously

- The (sub) program that deals with an interrupt is called an **interrupt service routine (ISR)** or **interrupt handler**

Interrupts

- An *interrupt* is an external or internal event that interrupts the microcontroller to inform it that a device needs its service.

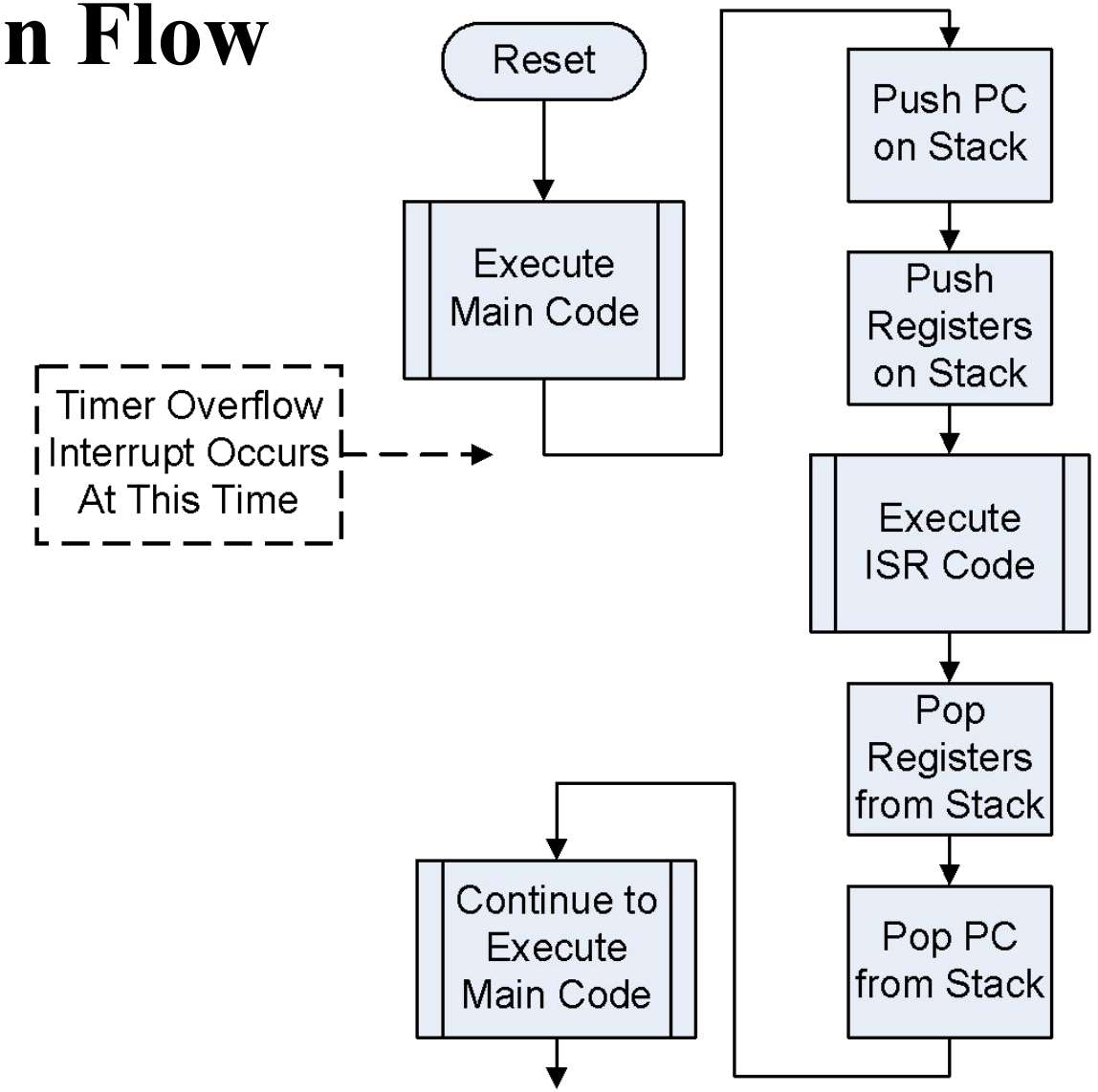
Interrupts vs. Polling

- A single microcontroller can serve several devices.
- There are two ways to do that:
  - interrupts
  - polling.
- The program which is associated with the interrupt is called the *interrupt service routine* (ISR) or *interrupt handler*.

Interrupts

- When an interrupt occurs, the main program temporarily suspends execution and branches to the ISR
- The ISR executes, performs the desired operation, and terminates with a “return from interrupt” (RETI) instruction
  - The RETI instruction is different from the normal “RET” instruction
- Finishes the current instruction and saves the contents of PC on stack.
- Jumps to a fixed location in memory depending on the type of an interrupt.
- Starts to execute the interrupt service routine until RETI (return from interrupt).
- Upon executing the RETI the microcontroller returns to the place where it was interrupted. Get pop PC from stack

Execution Flow



Interrupt handler

- **Interrupt handler provides 16 interrupt sources into the CIP-51 (as opposed to 7 for the standard 8051), allowing numerous analog and digital peripherals to interrupt the controller.**

- **An interrupt driven system requires less intervention by the MCU, giving it more effective throughput.**

- **Each interrupt source can be individually enabled or disabled through the use of an associated interrupt enable bit in an SFR (IE-EIE2).**

- **However, interrupts must first be globally enabled by setting the EA bit (IE.7) to logic 1 before the individual interrupt enables are recognized. Setting the EA bit to logic 0 disables all interrupt sources regardless of the individual interrupt-enable settings.**

Interrupt Summary

Table 9.4. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	Enable Flag	Priority Control
Reset	0x0000	Top	None	N/A	N/A	Always Enabled	Always Highest
External Interrupt 0 ($\overline{\text{INT0}}$)	0x0003	0	IE0 (TCON.1)	Y	Y	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0x000B	1	TF0 (TCON.5)	Y	Y	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1 ($\overline{\text{INT1}}$)	0x0013	2	IE1 (TCON.3)	Y	Y	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	0x001B	3	TF1 (TCON.7)	Y	Y	ET1 (IE.3)	PT1 (IP.3)
UART0	0x0023	4	RI0 (SCON0.0) TI0 (SCON0.1)	Y	N	ES0 (IE.4)	PS0 (IP.4)

IE: Interrupt Enable

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
EA	ESPI0	ET2	ES0	ET1	EX1	ET0	EX0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xA8

Bit3: ET1: Enable Timer 1 Interrupt.
This bit sets the masking of the Timer 1 interrupt.
0: Disable all Timer 1 interrupt.
1: Enable interrupt requests generated by the TF1 flag.

Bit2: EX1: Enable External Interrupt 1.
This bit sets the masking of External Interrupt 1.
0: Disable external interrupt 1.
1: Enable interrupt requests generated by the $\overline{\text{INT1}}$ input.

Bit1: ET0: Enable Timer 0 Interrupt.
This bit sets the masking of the Timer 0 interrupt.
0: Disable all Timer 0 interrupt.
1: Enable interrupt requests generated by the TF0 flag.

Bit0: EX0: Enable External Interrupt 0.
This bit sets the masking of External Interrupt 0.
0: Disable external interrupt 0.
1: Enable interrupt requests generated by the $\overline{\text{INT0}}$ input.

IP: Interrupt Priority

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	PSPI0	PT2	PS0	PT1	PX1	PT0	PX0	10000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
							(bit addressable)	0xB8

Bit3: PT1: Timer 1 Interrupt Priority Control.
This bit sets the priority of the Timer 1 interrupt.
0: Timer 1 interrupt set to low priority level.
1: Timer 1 interrupts set to high priority level.

Bit2: PX1: External Interrupt 1 Priority Control.
This bit sets the priority of the External Interrupt 1 interrupt.
0: External Interrupt 1 set to low priority level.
1: External Interrupt 1 set to high priority level.

Bit1: PT0: Timer 0 Interrupt Priority Control.
This bit sets the priority of the Timer 0 interrupt.
0: Timer 0 interrupt set to low priority level.
1: Timer 0 interrupt set to high priority level.

Bit0: PX0: External Interrupt 0 Priority Control.
This bit sets the priority of the External Interrupt 0 interrupt.
0: External Interrupt 0 set to low priority level.
1: External Interrupt 0 set to high priority level.

Power Management Modes

Power Management Modes

- **The CIP-51 core has two software programmable power management modes:**

  1. Idle

  2. Stop.

Power Control Register (PCON)

Power Control Register (PCON) is used to control the CIP-51's power management modes.

- **SFR Definition PCON: Power Control**

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
GF5	GF4	GF3	GF2	GF1	GF0	STOP	IDLE	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x87

Bits7–2: GF5–GF0: General Purpose Flags 5–0.
These are general purpose flags for use under software control.

Bit1: STOP: Stop Mode Select.
Setting this bit will place the CIP-51 in Stop mode. This bit will always be read as 0.
1: CPU goes into Stop mode (internal oscillator stopped).

Bit0: IDLE: Idle Mode Select.
Setting this bit will place the CIP-51 in Idle mode. This bit will always be read as 0.
1: CPU goes into Idle mode. (Shuts off clock to CPU, but clock to Timers, Interrupts, Serial Ports, and Analog Peripherals are still active.)

Idle mode

- **Idle mode-**
  - **Halts the CPU**
  - **internal registers and memory maintain their original data**
  - **Peripherals and clocks active.**
- **Setting the Idle Mode Select bit (PCON.0) causes the CIP-51 to halt the CPU and enter Idle.**
- **Idle mode is terminated when an enabled interrupt is asserted or a reset occurs. The assertion of an enabled interrupt will cause the Idle Mode Selection bit (PCON.0) to be cleared and the CPU to resume operation.**
- **If enabled, the Watchdog Timer (WDT) will eventually cause an internal watchdog reset and thereby terminate the Idle mode.**

Stop mode

- **Stop mode-**
  - **CPU is halted**
  - **All interrupts inactive**
  - **Internal oscillator is stopped**
  - **All digital & analog peripherals stopped**
  - **External oscillator circuit is not affected**
- **Setting the Stop Mode Select bit (PCON.1) causes the CIP-51 to enter Stop mode as soon as the instruction that sets the bit completes execution.**
- **Stop mode can only be terminated by an internal or external reset.**
- **If enabled, the Missing Clock Detector will cause an internal reset and there by terminate the Stop mode.**
- **The Missing Clock Detector should be disabled if the CPU is to be put to in STOP mode for longer than the MCD timeout of 100 µsec**

Reference

Datasheet:

https://www.silabs.com/documents/public/datasheets/C8051F34x.pdf