Module 4: Interfacing with Essential Peripherals

Welcome to Module 4! In this module, we will explore the fascinating world of peripheral devices and how microprocessors communicate and control them. Modern computing relies heavily on these specialized chips to extend the capabilities of the core processor, enabling tasks ranging from precise timing and communication to real-world data acquisition. We will dive into the principles and practical aspects of interfacing with Programmable Interval Timers, Serial and Parallel Communication interfaces, and Analog-to-Digital/Digital-to-Analog Converters.

4.1 Programmable Interval Timers (PIT): Principles of Operation, Modes, and Applications (e.g., 8253/8254)

Programmable Interval Timers (PITs) are essential components in microprocessor-based systems, providing precise timing, event counting, and waveform generation capabilities. Instead of relying on the CPU to manage time-critical operations, a PIT offloads these tasks, freeing up the CPU for other computations. The Intel 8253 and its enhanced version, the 8254, are popular examples of such devices.

4.1.1 Principles of Operation and Internal Structure

A PIT fundamentally consists of one or more independent, identical counters. Each counter is a down-counter that decrements its value with every clock pulse it receives. When the counter reaches zero, it typically generates an output signal or sets a flag, indicating that a programmed interval has elapsed or a certain number of events have occurred.

The 8253/8254, for instance, has three independent 16-bit counters (Counter 0, Counter 1, Counter 2). Each counter can operate in various modes, as programmed by the CPU.

Key Components within a PIT:

- Data Bus Buffer: This tri-state buffer connects the PIT to the microprocessor's data bus, enabling data transfer for control words and count values.
- Read/Write Logic: Decodes CPU signals (`RD`, `WR`, `CS`, address lines) to select internal registers (control word or counters).
- Control Word Register (CWR): An 8-bit register where the CPU writes a control word to configure the selected counter's mode, read/write format, and counting type (Binary/BCD).
- Counters (Counter 0, 1, 2): Each is a 16-bit programmable down-counter with specific pins:
  - CLK (Clock Input): External clock source for counting.
  - GATE (Gate Input): Control input to enable/disable counting based on the mode.

    - OUT (Output): Generates signals (pulses, square waves) as per the programmed mode.

Operation Cycle Summary:

1. Initialization: CPU writes a Control Word to the PIT's CWR.
2. Loading Count: CPU writes the initial 16-bit count value to the selected counter.
3. Counting: The counter decrements its value with each `CLK` pulse, considering the `GATE` input.
4. Output Generation: `OUT` pin generates signals based on the programmed mode and count reaching zero.
5. Re-loading (for repeating modes): Some modes automatically reload the initial count for continuous operation.

4.1.2 Modes of Operation (8253/8254)

The 8253/8254 supports six distinct modes of operation (Mode 0 to Mode 5).

- Mode 0: Interrupt on Terminal Count (Event Counter)
    - Function: Generates a single output pulse after a programmed count. `OUT` goes LOW immediately after loading count, then HIGH when count reaches zero.
    - Requirement: `GATE` must be HIGH for counting.
    - Application: Counting external events, single-shot delays.
- Mode 1: Hardware Retriggerable One-Shot
    - Function: Produces a single negative-going pulse of programmed duration in response to a rising edge trigger on `GATE`. Retriggering restarts the pulse.
    - Application: Pulse generation, watchdog timers.
- Mode 2: Rate Generator (Divide by N Counter)
    - Function: Generates a series of equally spaced negative pulses by dividing the input clock frequency. `OUT` goes LOW for one clock period when count reaches 1, then reloads `N`.
    - Requirement: `GATE` must be HIGH. `N` must be at least 2.
    - Formula: Output Frequency = Input Clock Frequency / N
    - Application: Baud rate generation, real-time clocks.
- Mode 3: Square Wave Generator
    - Function: Generates a continuous square wave with a 50% duty cycle, dividing the input clock frequency by `N`. `OUT` alternates HIGH and LOW for half the count duration.
    - Requirement: `GATE` must be HIGH.
    - Formula: Output Frequency = Input Clock Frequency / N
    - Application: Clock generation, audio tones, motor control (PWM).
- Mode 4: Software Triggered Strobe

    - Function: Generates a single negative-going pulse (one clock period) after a delay of `N` clock cycles, initiated by software (loading the counter).
    - Requirement: `GATE` must be HIGH.
    - Application: Generating delayed pulses for peripheral control.
- Mode 5: Hardware Triggered Strobe (Retriggerable)
    - Function: Similar to Mode 4, but the `N`-cycle delay is initiated by a rising edge hardware trigger on `GATE`. It is retriggerable.
    - Application: Delayed pulses triggered by external events.

4.1.3 Interfacing and Programming Example (8086)

Interfacing an 8253/8254 involves connecting its data bus, address lines, and control signals to the microprocessor.

Address Decoding Example: Assume the 8254 is at base address `0040H`. | A1 | A0 | Register Selection | Address (if base is 0040H) | | :-- | :-- | :-------------------------- | :------------------------ | | 0 | 0 | Counter 0 Data Register | 0040H | | 0 | 1 | Counter 1 Data Register | 0041H | | 1 | 0 | Counter 2 Data Register | 0042H | | 1 | 1 | Control Word Register (CWR) | 0043H |

Control Word Format (8-bit): | D7 D6 | D5 D4 | D3 D2 D1 | D0 | | :-------- | :---------- | :---------- | :---------- | | SC1 SC0 | RW1 RW0 | M2 M1 M0 | BCD | | (Counter) | (R/W Mode) | (Mode) | (Binary/BCD) |

Numerical Programming Example (8086): Configure Counter 0 of an 8254 (at base `0040H`) for Mode 3 (Square Wave), 16-bit binary count of `10000` (decimal), with a 1MHz input clock.

- Desired Count: 10000_decimal=2710_hex. (LSB: 10_hex, MSB: 27_hex)
- Output Frequency: 1,000,000textHz/10000=100textHz.
- Control Word:
    - Counter 0 (00)
    - Read/Write LSB then MSB (11)
    - Mode 3 (011)
    - Binary (0)
    - Control Word = `00110110` binary = `36H`

Code snippet
```
; 8086 Assembly Code Snippet
ASSUME CS:CODE

CODE SEGMENT
START:
   MOV AL, 36H     ; Control word: Counter 0, R/W LSB then MSB, Mode 3, Binary
   OUT 43H, AL     ; Send control word to 8254 CWR (0043H)

   MOV AX, 2710H   ; 16-bit count (10000 decimal)
```

```
    OUT 40H, AL     ; Send LSB (10H) to Counter 0 Data Register (0040H)
    MOV AL, AH      ; Get MSB (27H)
    OUT 40H, AL     ; Send MSB to Counter 0 Data Register (0040H)

    ; Counter 0 is now generating a 100 Hz square wave on its OUT0 pin.
    HLT
CODE ENDS
END START
```

Key Applications of PITs:

- Real-Time Clock (RTC) generation
- Baud rate generation for serial communication
- Event counting
- Pulse and square wave generation
- Watchdog timers

4.2 Serial Communication (UART/USART): Asynchronous and Synchronous Serial Communication, Baud Rate, and Protocols (RS-232, SPI, I2C - Conceptual)

Serial communication transmits data bits sequentially over a single line, offering simplicity and cost-effectiveness compared to parallel communication.

4.2.1 Asynchronous Serial Communication (UART)

Asynchronous communication does not use a shared clock line. Timing is embedded in the data stream using start and stop bits. UART (Universal Asynchronous Receiver/Transmitter) is the common hardware component.

Principles:

- Data Framing: Each character is framed with control bits:
    - Start Bit (1 bit): Always LOW (0), signals start of frame.
    - Data Bits (5-9 bits): The actual data.
    - Parity Bit (Optional): Used for basic error detection (Even/Odd parity).
    - Stop Bit(s) (1, 1.5, or 2 bits): Always HIGH (1), signals end of frame, returns line to idle.
- Idle State: Communication line is HIGH when no data is sent.
- Baud Rate: The rate of bit transmission per second (bits per second, bps). Both transmitter and receiver must operate at the same baud rate.
    - Formula: Baud Rate = (UART Clock Frequency) / (Divisor Value)
    - Numerical Example: For a 1.8432 MHz UART clock and a desired 9600 bps, the divisor would be 12 (since 1,843,200 / (16 * 12) = 9600).
- Synchronization: Receiver synchronizes by detecting the start bit's falling edge and samples data bits at their center.

Summary: Simple, low cost, fewer wires; suitable for low-to-medium speed, short-distance. Overhead bits reduce efficiency.

4.2.2 Synchronous Serial Communication (USART)

Synchronous communication uses a shared clock line to synchronize data transfer, eliminating the need for start/stop bits and improving efficiency. USART (Universal Synchronous/Asynchronous Receiver/Transmitter) is the capable hardware.

Principles:

- **Shared Clock: A dedicated clock line (e.g., SCLK) is provided by the transmitting device to synchronize data at the receiver.**
- **Data Transfer: Bits are transmitted continuously, synchronized by the clock. No framing overhead.**
- **Synchronization Word (Optional): May be used for initial synchronization in blocks of data.**

Summary: Higher data rates, more efficient (no overhead); requires an extra clock line, slightly more complex.

4.2.3 Common Serial Communication Protocols (Conceptual Overview)

Protocols define the rules for meaningful data exchange over serial connections.

- **RS-232:**
  - **Type: Asynchronous.**
  - **Physical Layer: Defines voltage levels (+3V to +15V for 0, -3V to -15V for 1), cabling (DB-9, DB-25 connectors).**
  - **Characteristics: Single-ended, susceptible to noise over distance.**
  - **Use: Legacy standard for modems, printers, industrial control, debugging.**
- **SPI (Serial Peripheral Interface):**
  - **Type: Synchronous.**
  - **Architecture: Master-slave. One master controls communication, multiple slaves possible (selected via Chip Select).**
  - **Dedicated Lines (typically 4):**
    - **SCLK: Serial Clock (from master).**
    - **MOSI: Master Out, Slave In (data from master to slave).**
    - **MISO: Master In, Slave Out (data from slave to master).**
    - **SS/CS: Slave Select / Chip Select (active LOW, for individual slave selection).**
  - **Duplex: Full duplex (simultaneous transmit/receive).**
  - **Characteristics: High speed, simple hardware, no addressing (uses CS).**
  - **Use: Microcontroller to sensor, EEPROM, SD card, display interfacing.**
- **I2C (Inter-Integrated Circuit):**
  - **Type: Synchronous.**
  - **Architecture: Multi-master, multi-slave.**
  - **Wires (2):**

        ■ SDA: Serial Data Line (bidirectional).
        ■ SCL: Serial Clock Line (from master).
    ○ Addressing: Each slave has a unique 7-bit or 10-bit address.
    ○ Duplex: Half duplex (one direction at a time).
    ○ Characteristics: Reduced wiring, supports many devices, moderate speed.
    ○ Use: Microcontroller to RTC, temperature sensors, small displays, memory.

4.3 Parallel Input/Output (PIO): Programmable Peripheral Interface (e.g., 8255), Modes of Operation, and Port Addressing

Parallel Input/Output (PIO) transfers multiple bits of data simultaneously. It offers high data rates over short distances. The Programmable Peripheral Interface (PPI) is a common IC for this, with the Intel 8255 being a widely used example.

4.3.1 Programmable Peripheral Interface (8255 PPI)

The 8255 is an 8-bit programmable parallel I/O chip designed to interface with microprocessors. It provides three 8-bit ports (Port A, Port B, Port C) that can be individually configured for various input/output modes.

Internal Structure of 8255:

- Data Bus Buffer: Connects the 8255 to the CPU's 8-bit data bus (D0-D7) for read/write operations.
- Read/Write Control Logic: Decodes `CS`, `RD`, `WR`, `A1`, `A0` signals to select the correct internal register.
    - Address Decoding Example: | A1 | A0 | Register Selection | Address (if base is 0070H) | | :-- | :-- | :------------------------- | :------------------------ | | 0 | 0 | Port A | 0070H | | 0 | 1 | Port B | 0071H | | 1 | 0 | Port C | 0072H | | 1 | 1 | Control Word Register (CWR) | 0073H |
- Group A Control: Manages Port A and the upper nibble of Port C (PC7-PC4).
- Group B Control: Manages Port B and the lower nibble of Port C (PC3-PC0).
- Ports (PA, PB, PC): Three 8-bit bidirectional ports. Port C can be split into two 4-bit nibbles for control/status signals.

4.3.2 Modes of Operation (8255)

The 8255 has two programming methods: the Mode Set Register and the Bit Set/Reset (BSR) Mode.

1. Mode Set Register (Control Word Register - CWR): This 8-bit register (`D7=1`) programs the overall operational mode of the 8255 ports.

- D7: Mode Set Flag (must be 1).
- D6, D5: Group A Mode Select (00=Mode 0, 01=Mode 1, 1X=Mode 2).
- D4: Port A Direction (1=Input, 0=Output).
- D3: Port C Upper (PC7-PC4) Direction (1=Input, 0=Output).

- D2: Group B Mode Select (0=Mode 0, 1=Mode 1).
- D1: Port B Direction (1=Input, 0=Output).
- D0: Port C Lower (PC3-PC0) Direction (1=Input, 0=Output).

Operational Modes for Ports:

- Mode 0: Basic Input/Output (Simple I/O)
    - Function: Simple unbuffered data read/write. No handshaking.
    - Characteristics: All ports (A, B, C) can be input or output. Outputs are latched.
    - Application: Driving LEDs, reading switches, basic data transfer.
- Mode 1: Strobed Input/Output (Handshake I/O)
    - Function: Provides handshaking for reliable data transfer with slower peripherals.
    - Characteristics: Port A and/or B use Mode 1. Port C bits provide handshaking signals (e.g., `STB`, `IBF`, `OBF`, `ACK`, `INT`).
    - Application: Interfacing with printers, keyboards, ADCs requiring control signals.
- Mode 2: Bidirectional Input/Output (Strobed Bidirectional)
    - Function: Allows Port A to be a bidirectional data bus with handshaking.
    - Characteristics: Only Port A can be Mode 2. Port B can be Mode 0/1. Port C provides handshaking signals for Port A.
    - Application: Two-way data communication with external memory or another CPU.

2. Bit Set/Reset (BSR) Mode: This mode (`D7=0`) allows individual bits of Port C (PC0-PC7) to be set (1) or reset (0).

- D7: Mode Set Flag (must be 0).
- D6, D5, D4: Must be 0 (don't care).
- D3 D2 D1: Bit Select (000=PC0, ..., 111=PC7).
- D0: Set/Reset (1=Set, 0=Reset).
- Application: Controlling individual control lines, generating pulses, setting/clearing flags.

4.3.3 Programming Example (8086)

Assume the 8255 CWR is at `0073H`.

Numerical Programming Example (8086): Configure 8255: Port A = Output (Mode 0), Port B = Input (Mode 0), Port C Upper = Output (Mode 0), Port C Lower = Input (Mode 0). Then, write `AAH` to Port A, and conceptually `55H` to Port C.

- Control Word for Mode Set:
    - D7=1, G_A_Mode=00, PA_Dir=0, PCU_Dir=0, G_B_Mode=0, PB_Dir=1, PCL_Dir=1
    - Control Word = `10000011` binary = `83H`

Code snippet

```
; 8086 Assembly Code Snippet
ASSUME CS:CODE

CODE SEGMENT
START:
   ; Configure 8255
   MOV AL, 83H     ; Control word: PA=OUT, PB=IN, PC(upper)=OUT, PC(lower)=IN, all
Mode 0
   OUT 73H, AL     ; Send control word to 8255 CWR (0073H)

   ; Write AAH to Port A
   MOV AL, 0AAH
   OUT 70H, AL     ; Write to Port A (0070H)

   ; Write 55H to Port C (conceptual: only affects output-configured bits)
   MOV AL, 55H
   OUT 72H, AL     ; Write to Port C (0072H)

   ; Read from Port B
   IN AL, 71H      ; Read data from Port B (0071H) into AL

   ; Read from Port C
   IN AL, 72H      ; Read data from Port C (0072H) into AL
                ; AL will contain current input from PC0-PC3, and output from PC4-PC7.
   HLT
CODE ENDS
END START
```

Key Applications of PPIs:

- **Keyboard and display interfacing**
- **Printer interfacing**
- **Motor control**
- **Reading digital sensors**
- **General purpose I/O expansion**

4.4 Analog-to-Digital Converters (ADCs): Principles, Types (SAR, Flash), and Interfacing Techniques

An Analog-to-Digital Converter (ADC) transforms continuous analog signals into discrete digital representations, enabling microprocessors to process real-world data like temperature or pressure.

4.4.1 Principles of Analog-to-Digital Conversion

The conversion process involves three main steps:

1. Sampling: Measuring the analog signal at discrete time intervals.
   - Requirement: Sampling rate must be at least twice the highest signal frequency (Nyquist-Shannon theorem).
2. Quantization: Approximating each sampled value to the nearest discrete digital level.
   - Resolution (N bits): Determines the number of levels (2N). Higher resolution means more levels and less error.
   - Step Size (LSB Value): The voltage change per digital increment.
     - Formula: Step Size = Reference Voltage (V_REF) / 2N
     - Numerical Example: For an 8-bit ADC with V_REF = 5V:
       - Number of levels = 28=256.
       - Step Size = 5V/256approx0.01953V per LSB.
       - Digital Output `10000000b` (128 decimal) corresponds to approx128times0.01953V=2.5V.
3. Encoding: Converting the quantized value into its binary code.

4.4.2 Types of ADCs

ADC architectures vary in speed, accuracy, and cost.

- 1. Successive Approximation Register (SAR) ADC:
  - Principle: Performs a binary search (bit by bit, MSB to LSB) to match the input analog voltage. It uses an internal DAC and a comparator.
  - Process: Sets MSB to 1, compares DAC output to input. If input is higher, MSB stays 1; else, MSB is 0. Repeats for each bit.
  - Advantages: Good balance of speed and accuracy, moderate power, cost-effective.
  - Disadvantages: Conversion time is proportional to resolution (N clock cycles for N bits).
  - Application: General-purpose data acquisition, sensor interfaces.
- 2. Flash ADC (Parallel ADC):
  - Principle: Fastest ADC type. Uses 2N−1 parallel comparators, each with a unique reference voltage from a resistor ladder.
  - Process: All comparisons occur simultaneously. A priority encoder converts the comparator outputs ("thermometer code") to binary.
  - Advantages: Extremely fast conversion (single clock cycle).
  - Disadvantages: High power, large chip area, high cost for higher resolutions (2N−1 comparators required).
  - Application: High-speed oscilloscopes, video processing, radar.

4.4.3 Interfacing Techniques (Conceptual)

Interfacing an ADC depends on its output format and control signals.

- Parallel Interface: ADC output lines connect directly to a parallel input port (e.g., 8255).

    - Process: CPU sends `START CONVERSION` signal. ADC converts. ADC asserts `EOC` (End of Conversion) or `DRDY` (Data Ready) signal. CPU reads data from input port.
    - Numerical Example (8086 with 8255):
        - Assume 8-bit ADC (ADC0804) to 8255 Port A (input). ADC `WR` to 8255 PC0 (output), ADC `INTR` to 8255 PC1 (input).
        - Steps:
            1. Configure 8255 (Port A input, PC0 output, PC1 input).
            2. Pulse PC0 (LOW then HIGH) to start ADC conversion.
            3. Poll PC1 (read Port C, check PC1 bit) until `INTR` goes LOW (conversion complete).
            4. Read digital data from 8255 Port A.
- Serial Interface: Many modern ADCs use serial protocols like SPI or I2C.
    - Process: CPU sends commands and receives data serially over the bus.

Key Applications of ADCs:

- Sensor interfacing (temperature, pressure, light)
- Audio/video digitization
- Medical instruments
- Industrial control and monitoring
- Digital Signal Processing (DSP) input

4.5 Digital-to-Analog Converters (DACs): Principles, Types (R-2R Ladder), and Interfacing Techniques

Digital-to-Analog Converters (DACs) convert discrete digital data into a continuous analog voltage or current. They are crucial for controlling analog devices with digital systems.

4.5.1 Principles of Digital-to-Analog Conversion

The core principle involves summing weighted electrical contributions corresponding to each bit of the digital input.

- Weighted Summation: Each digital input bit controls a switch, directing a weighted current or voltage (based on bit significance) to a summing point.
- Resolution (N bits): Determines the number of distinct analog output levels (2N). Higher resolution means finer output steps.
- Step Size (LSB Value): The smallest change in analog output for a 1 LSB digital input change.
    - Formula: Step Size = Reference Voltage (V_REF) / 2N
    - Numerical Example: For an 8-bit DAC with V_REF = 5V:
        - Number of levels = 28=256.
        - Step Size = 5V/256approx0.01953V per LSB.
        - Digital input `01000000b` (64 decimal) corresponds to approx64times0.01953V=1.25V.

4.5.2 Types of DACs

Different DAC architectures offer trade-offs in performance.

- 1. R-2R Ladder DAC:
  - Principle: Uses a network of resistors with only two values (R and 2R) to create binary-weighted currents. Switches (controlled by digital input) direct these currents to an op-amp for summation.
  - Advantages: Good accuracy, simple manufacturing due to only two resistor values, less sensitive to resistor tolerance than weighted-resistor DACs.
  - Application: Audio playback, waveform generation, motor speed control.
- Other DAC Types (Briefly):
  - Weighted Resistor DAC: Uses distinct precision resistors for each bit. Conceptually simple, but challenging for high accuracy due to wide resistor value range.
  - Delta-Sigma (DeltaSigma) DAC: High-resolution (16-24 bits) DACs used in audio and precision instrumentation, achieving performance through oversampling and noise shaping.
  - Pulse Width Modulation (PWM) DAC: A low-cost method where an analog voltage is generated by varying the duty cycle of a square wave, followed by a low-pass filter.

4.5.3 Interfacing Techniques (Conceptual)

Interfacing a DAC involves providing digital data to its input pins.

- Parallel Interface: DAC digital inputs connect directly to a parallel output port (e.g., 8255).
  - Process: CPU writes desired digital value to the output port. DAC converts this to analog. A `LOAD` or `WR` pulse might be needed to latch data.
  - Numerical Example (8086 with 8255):
    - Assume 8-bit DAC (DAC0808) to 8255 Port A (output). DAC `WR` to 8255 PC0 (output).
    - Steps:
      - Configure 8255 (Port A output, PC0 output).
      - Write desired digital value (e.g., `80H`) to 8255 Port A.
      - Pulse 8255 PC0 (LOW then HIGH) to latch the data into the DAC.
      - The DAC's analog output will now correspond to `80H`.
- Serial Interface: Many modern DACs use serial protocols like SPI or I2C for input.
  - Process: CPU sends digital data serially to the DAC.

Key Applications of DACs:

- **Audio playback**
- **Motor speed control**
- **Programmable power supplies**
- **Waveform generation**
- **Display brightness control**
- **Robotics (actuator control)**