 **INTRODUCTION:**

The STL034V1.0 is a versatile development board designed by Silicon Techno Labs for programming and testing projects based on the PIC16F877A microcontroller. This board features a Zero Insertion Force (ZIF) socket, which simplifies the insertion and removal of the microcontroller. It supports both AC and DC power supply adapters, facilitated by an onboard bridge rectifier and a 5V regulator (LM7805), with a recommended input voltage range of 9-12V. The board includes a DB9 RS232 port for serial communication with computers and other serial devices like GPS modules, GSM modems, and RFID readers, aided by an onboard MAX232 interface circuit. Additionally, it provides a standard ICSP (In-Circuit Serial Programming) header for convenient programming using Pickit-2 or Pickit-3 programmers. The board is equipped with power status indicators, a user switch for input, a user LED for output, and detailed pin labeling for all port extensions, making it an ideal platform for developing and testing various microcontroller-based applications.

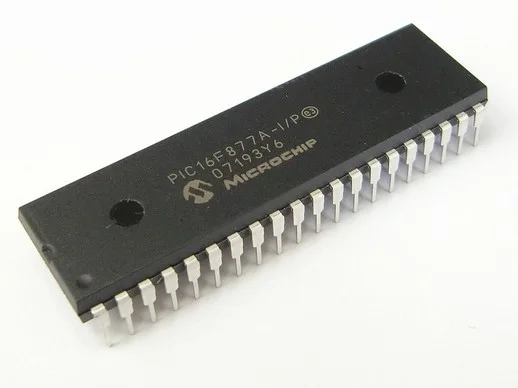
The STL034V1.0 development board is equipped with several essential components that make it highly functional and user-friendly for microcontroller projects there are three main components they are:

* PIC16F877A Microcontroller
* DB9 RS232 Port
* MAX232

Features of STL034V1.0:

* ON-BOARD BRIDGE RECTIFIER ENABLES THE BOARD TO ACCEPT BOTH AC AND DC INPUT VOLTAGES
* ZIF(ZERO INSERTION FORCE) SOCKET FOR EASY INSERTING AND REMOVING MICROCONTROLLER FROM BOARD.
* RECOMMENDED INPUT VOLTAGE: 9-12V
* MIN-MAX INPUT VOLTAGE: 9-18V
* 5 MM STANDARD DC PLUG-IN JACK FOR POWER SUPPLY
* ON-BOARD 5V REGULATOR (LM7805) CIRCUIT.
* ON-BOARD POWER SUPPLY ON-OFF SWITCH
* POWER STATUS LED(RED)
* ON-BOARD 20MHZ CRYSTAL
* PORT EXTENSIONS FOR ALL PORTS WITH DETAILED PIN LABELING FOR EASY IDENTIFICATION OF PINS.
* ON-BOARD MAX232 INTERFACE CIRCUIT FOR EASY COMMUNICATION WITH A COMPUTER AND OTHER SERIAL DEVICES (GPS MODULES, GSM MODEMS, ETC)
* ON-BOARD STANDARD ICSP (IN-CIRCUIT SERIAL PROGRAMMING) HEADER FOR LOADING HEX.
* 12V, 5V AND GND BUS PROVIDED FOR EXTERNAL PERIPHERAL.
* HIGH QUALITY FR-4(1.6 MM) PCB
* FOUR 3MM MOUNTING HOLE FOR EASY MOUNTING

**PIC16F877A Microcontroller:**

 The PIC16F877A is an 8-bit microcontroller from Microchip Technology, popular in embedded systems due to its versatile features. It includes 14 KB of Flash memory for program storage, 368 bytes of RAM, and 256 bytes of EEPROM for non-volatile data retention. The microcontroller has 33 I/O pins distributed across five ports, supporting digital I/O operations. It features three timers, a 10-bit ADC with 8 channels, two analog comparators, and supports USART, SPI, and I2C communication protocols. Additionally, it includes PWM modules and various power-saving modes, making it a robust and flexible choice for various applications.

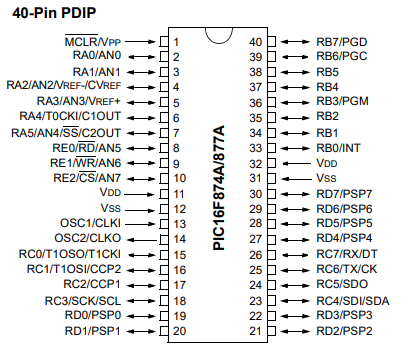
There are four main pic controllers:

• PIC16F873A • PIC16F874A

• PIC16F876A • PIC16F877A

In terms of programming and development, the PIC16F877A is supported by Microchip's MPLAB IDE, which provides a comprehensive development environment with debugging tools and libraries. This facilitates rapid prototyping and firmware development, crucial for speeding up time-to-market in embedded system projects. Overall, the PIC16F877A microcontroller stands out for its balance of performance, power efficiency, and versatility, making it a cornerstone in the field of embedded electronics and microcontroller-based systems. The PIC16F877A operates at speeds up to 20 MHz and features 33 I/O pins, which can be configured for various functions such as analog-to-digital conversion (ADC), pulse width modulation (PWM), and timers/counters. It supports up to 8 channels of 10-bit ADC for analog signal processing and features multiple timers/counters, including a 16-bit timer and two 8-bit timers, facilitating precise timing and event management in applications requiring time-sensitive operations.

**Peripheral Features:**

• **Timer0**: 8-bit timer/counter with 8-bit prescaler

• **Timer1**: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock

• **Timer2**: 8-bit timer/counter with 8-bit period register, prescaler and postscaler

• Two Capture, Compare, PWM modules - Capture is 16-bit, max. resolution is 12.5 ns - Compare is 16-bit, max. resolution is 200 ns - PWM max. resolution is 10-bit

• Synchronous Serial Port (SSP) with SPI™ (Master mode) and I2C™ (Master/Slave)

• Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection

• Parallel Slave Port (PSP) – 8 bits wide with external RD, WR and CS controls (40/44-pin only)

• Brown-out detection circuitry for Brown-out Reset (BOR)

**Special protocals:**

 **SPI (Serial Peripheral Interface)**: Used for high-speed synchronous serial communication between microcontrollers and peripheral devices.

 **I2C (Inter-Integrated Circuit)**: A multi-master, multi-slave, packet-switched protocol used for communication between integrated circuits on a board.

 **USART (Universal Synchronous/Asynchronous Receiver/Transmitter)**: Provides asynchronous and synchronous serial communication capabilities typically used for communication with peripherals and external devices.

**Internal architecture:**

The internal architecture of the PIC16F877A microcontroller consists of a CPU core responsible for executing instructions fetched from its program memory. It includes data memory (RAM) for temporary storage and EEPROM for non-volatile data storage. Peripherals like timers, ADC, and USART are integrated to facilitate precise timing, analog signal processing, and serial communication, making it well-suited for a wide range of embedded applications requiring efficient control and interfacing capabilities.

The internal architecture consists of:

 CPU (Central Processing Unit)

 Program Memory

 RAM (Random Access Memory)

 EEPROM (Electrically Erasable Programmable Read-Only Memory)

 I/O Ports

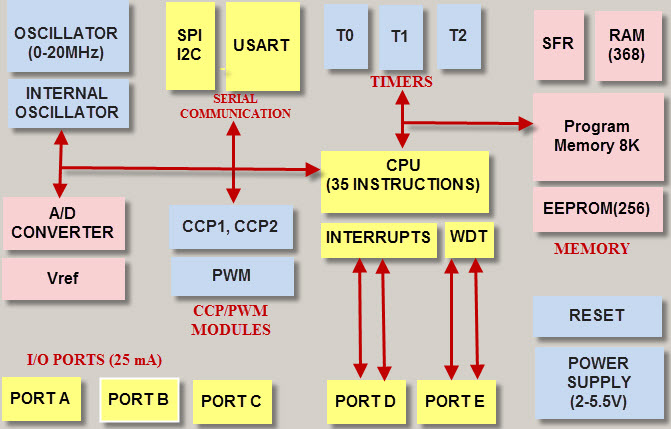
 Timers

 ADC (Analog-to-Digital Converter)

 Comparator

 USART (Universal Synchronous/Asynchronous Receiver/Transmitter)

 SPI (Serial Peripheral Interface)

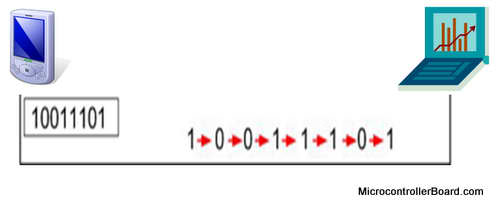
 I2C (Inter-Integrated Circuit)

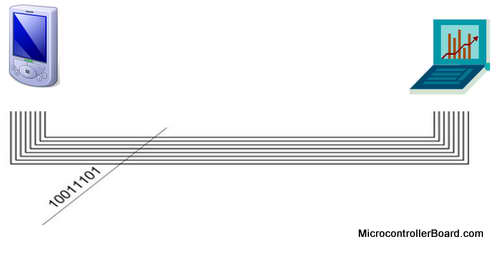
The internal architecture of the PIC16F877A microcontroller revolves around its central processing unit (CPU), which executes instructions fetched from program memory. This 8-bit RISC-based CPU is complemented by various peripheral modules integrated on-chip. These include:

1. **Memory Units**: The microcontroller includes 14 KB of flash program memory for storing instructions and data that are non-volatile and can be electrically erased and reprogrammed (EEPROM). It also has 368 bytes of RAM for temporary data storage during program execution.
2. **Peripheral Modules**: It features timers/counters for precise timing and event management, essential for tasks such as generating PWM signals or measuring time intervals. An Analog-to-Digital Converter (ADC) allows the microcontroller to convert analog signals into digital data, crucial for interfacing with sensors and other analog devices. Additionally, USART, SPI, and I2C modules provide serial communication interfaces, enabling communication with external devices and systems.
3. **I/O Ports**: The microcontroller includes numerous I/O pins that can be configured as digital inputs or outputs. These ports facilitate communication with external components and devices, allowing the microcontroller to interact with its environment.
4. **Interrupt System**: An interrupt system enhances the responsiveness of the microcontroller by allowing it to respond promptly to external events or triggers without continuously polling peripheral devices.

**TYPES OF COMMUNICATIONS:**

Pic microcontroller uses serial communication technique. Generally there are two types they are serial communication and parallel communication. When using the parallel communication, however, the number of bits will be transmitted at once from one computer to the second computer. When using the serial communication we transmit the multi-bit word bit after bit (when at any given moment only one bit will pass).





**Serial communication parallel communication**

In addition to the serial and parallel communications, there are 2 types of communication. They are Synchronous communication and Asynchronous communication.

**Synchronous communication**

When using the synchronous communication – the information is transmitted from the transmitter to the receiver:

* in sequence
* bit after bit
* with fixed baud rate
* and the clock frequency is transmitted along with the bits

That means that the transmitter and the receiver are synchronized between them by the same clock frequency. The clock frequency can be transmitted along with the information, while it is encoded in the information itself, or in many cases there is an additional wire for the clock.This type of communication is faster compare to the asynchronous communication since it is "constantly transmitting” the information, with no stops.

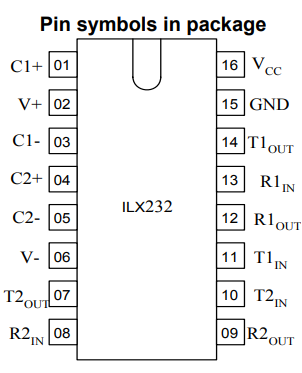
**Asynchronous communication**

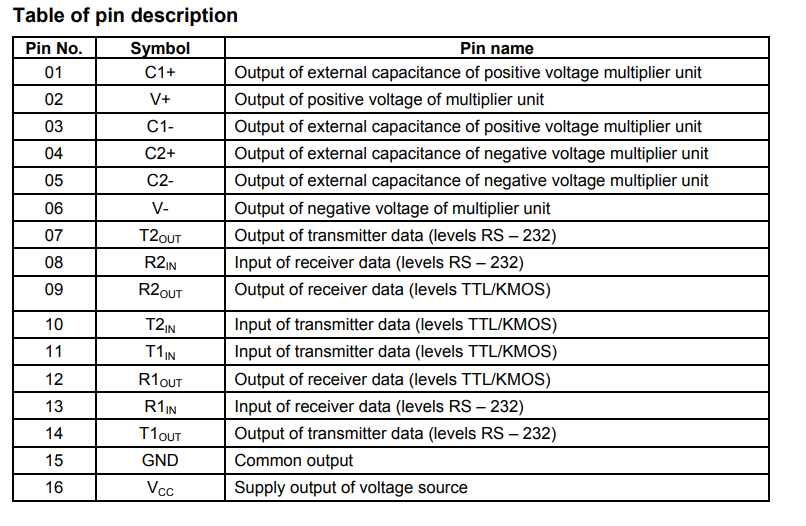
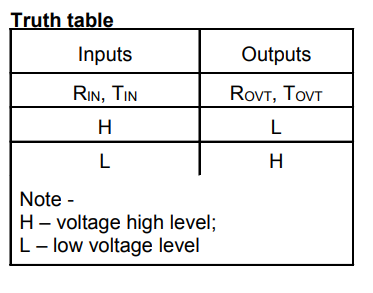
When using the asynchronous communication - the transmitter and the receiver refraining to transmit long sequences of bits because there isn't a full synchronization between the transmitter, that sends the data, and the receiver, that receives the data.

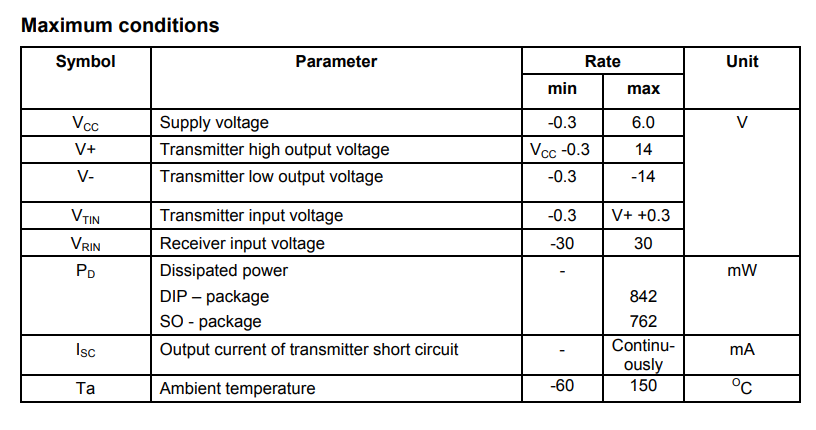
**Applications**

1. **Embedded Systems**: PIC microcontrollers are extensively used in embedded systems for controlling and monitoring tasks. They are employed in industrial automation, home appliances, medical devices, and automotive systems for tasks such as motor control, temperature sensing, and user interface management.
2. **Consumer Electronics**: They are found in consumer electronics such as smart devices, remote controls, and electronic toys due to their small size, low power consumption, and ability to interface with different types of sensors and actuators.
3. **Communication Systems**: PIC microcontrollers are used in communication systems for implementing protocols like UART, SPI, and I2C, enabling devices to communicate with each other over wired or wireless networks.
4. **Instrumentation**: They are used in instrumentation applications such as data acquisition systems, where they interface with sensors to measure and process analog signals into digital data for analysis and control.
5. **Power Electronics**: PIC microcontrollers are employed in power electronic systems for tasks like power monitoring, voltage regulation, and power factor correction, enhancing efficiency and reliability in energy management applications.
6. **Security Systems**: They play a role in security systems such as access control, alarm systems, and surveillance cameras, providing control and monitoring capabilities to ensure safety and security.
7. **Educational Tools**: PIC microcontrollers are widely used in educational settings to teach students about embedded systems design, programming, and electronics due to their accessibility and rich development environment.
8. **Robotics**: They are integral to robotics applications for controlling movement, sensor integration, and decision-making processes, enabling autonomous or semi-autonomous operation in various robotic platforms.

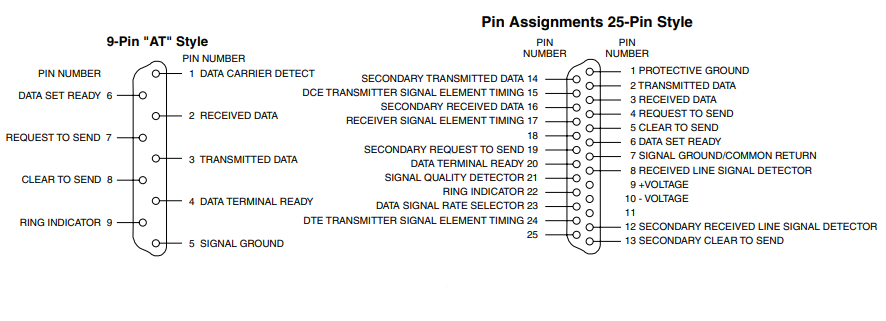
**ILX232(MAX232):**

IC ILX232 is purposed for application in high-performance information processing systems and control devices of wide application. Input voltage levels are compatible with standard СMOS levels. Output voltage levels are compatible with input levels of K-MOS, N-MOS and TTL integrated circuits. Supply voltage range from 2.0 to 6.0 V. Low input current: 1.0 mkA; 0.1 mkA at Т = 25 С. Output current 24 mA. Latching current not less than 450 mA at Т = 25 С Tolerable value of static potential not less than 2000V





**RS232:**

 RS232 (Recommended Standard 232) is a widely used serial communication protocol that facilitates data exchange between computers and peripheral devices. It was established by the Electronic Industries Association (EIA) and specifies the electrical characteristics and timing of signals, along with the meaning of signals and the physical size and pinout of connectors. RS232 typically uses a 9-pin (DB9) or 25-pin (DB25) connector and supports full-duplex communication, allowing simultaneous transmission and reception of data. It operates at various baud rates, commonly 9600 bps, and utilizes voltage levels to represent binary data, with a range of ±3 to ±15 volts for data transmission. Despite being largely superseded by USB and other modern interfaces, RS232 remains prevalent in industrial automation, networking equipment, and legacy systems due to its simplicity and reliability.

 **Pin 1 (DCD - Data Carrier Detect)**: This pin is used by the modem to indicate to the computer that it has detected a data carrier signal from a remote device.

 **Pin 2 (RXD - Receive Data)**: This pin carries the data received from the transmitting device.

 **Pin 3 (TXD - Transmit Data)**: This pin carries the data to be transmitted to the receiving device.

 **Pin 4 (DTR - Data Terminal Ready)**: This pin is used by the computer to signal that it is ready to communicate.

 **Pin 5 (GND - Ground)**: This pin provides the common ground reference for the communication.

 **Pin 6 (DSR - Data Set Ready)**: This pin is used by the modem to signal that it is ready to establish a communication link.

 **Pin 7 (RTS - Request to Send)**: This pin is used by the computer to prepare the modem for data transmission.

 **Pin 8 (CTS - Clear to Send)**: This pin is used by the modem to indicate that it is ready to receive data from the computer.

 **Pin 9 (RI - Ring Indicator)**: This pin is used by the modem to indicate an incoming call signal.

**INTERFACING PIC16F877A AND RS232 USING MAX232:**

Interfacing the PIC16F877A microcontroller with an RS232 serial communication standard requires voltage level conversion due to the different voltage requirements of TTL (Transistor-Transistor Logic) and RS232. The MAX232 IC facilitates this by converting TTL voltage levels to RS232 levels and vice versa.

**Voltage Levels:**

* **PIC16F877A**: Operates at TTL logic levels (0V to 5V).
  + Logic '0' : 0V
  + Logic '1' : 5V
* **RS232**: Uses ±12V for logic levels.
  + Logic '0' (Space): +3V to +15V
  + Logic '1' (Mark): -3V to -15V

Due to these differences, direct communication is not feasible, necessitating the use of the MAX232 IC for voltage level translation.

The MAX232 is a level shifter IC that converts TTL logic levels to RS232 levels and vice versa using internal charge pumps with the help of external capacitors.

* **Pin Functions**:
  + **T1IN/T2IN**: TTL input pins for data transmission.
  + **T1OUT/T2OUT**: RS232 output pins for transmitted data.
  + **R1IN/R2IN**: RS232 input pins for received data.
  + **R1OUT/R2OUT**: TTL output pins for received data.

**Circuit Connections:**

**Connecting the UART:**

* Microcontroller TX to MAX232 T1IN.
* MAX232 T1OUT to RS232 Connector RX.
* RS232 Connector TX to MAX232 R1IN.
* MAX232 R1OUT to Microcontroller RX.

**Capacitor Configuration:**

Connect four capacitors (1µF to 10µF) to the MAX232 at the specified pins (C1+, C1-, C2+, C2-) for charge pump operation.

**Power Supply:**

Provide a 5V supply to the MAX232 (Vcc) and connect the ground (GND).

**Data Transmission and Reception:**

**Transmission:**

The microcontroller places the data to be transmitted in the TXREG register. The data is sent to the MAX232 through T1IN.The MAX232 converts the TTL level to RS232 level and sends it out through T1OUT to the RS232 connector. The RS232 device receives the data.

**Reception:**

Data from the RS232 device enters the MAX232 through R1IN.The MAX232 converts the RS232 level to TTL level and outputs it through R1OUT.The microcontroller reads the data from the RCREG register.

**Capacitors in the MAX232 Circuit:**

The typical connections for the capacitors in the MAX232 circuit are as follows:

**C1+ and C1-:** Connected to one capacitor (typically 1µF to 10µF) which is used for the charge pump.

**C2+ and C2-:** Connected to another capacitor for the second part of the charge pump.

**V+ and V-:** The generated positive and negative voltages, where additional capacitors are connected to stabilize these voltages.

**C4 (between C1+ and C1-):**

Used for the first stage of the charge pump.

Helps in the initial step of boosting the voltage.

**C3 (between C2+ and C2-):**

Used for the second stage of the charge pump.

Further boosts the voltage to the required levels.

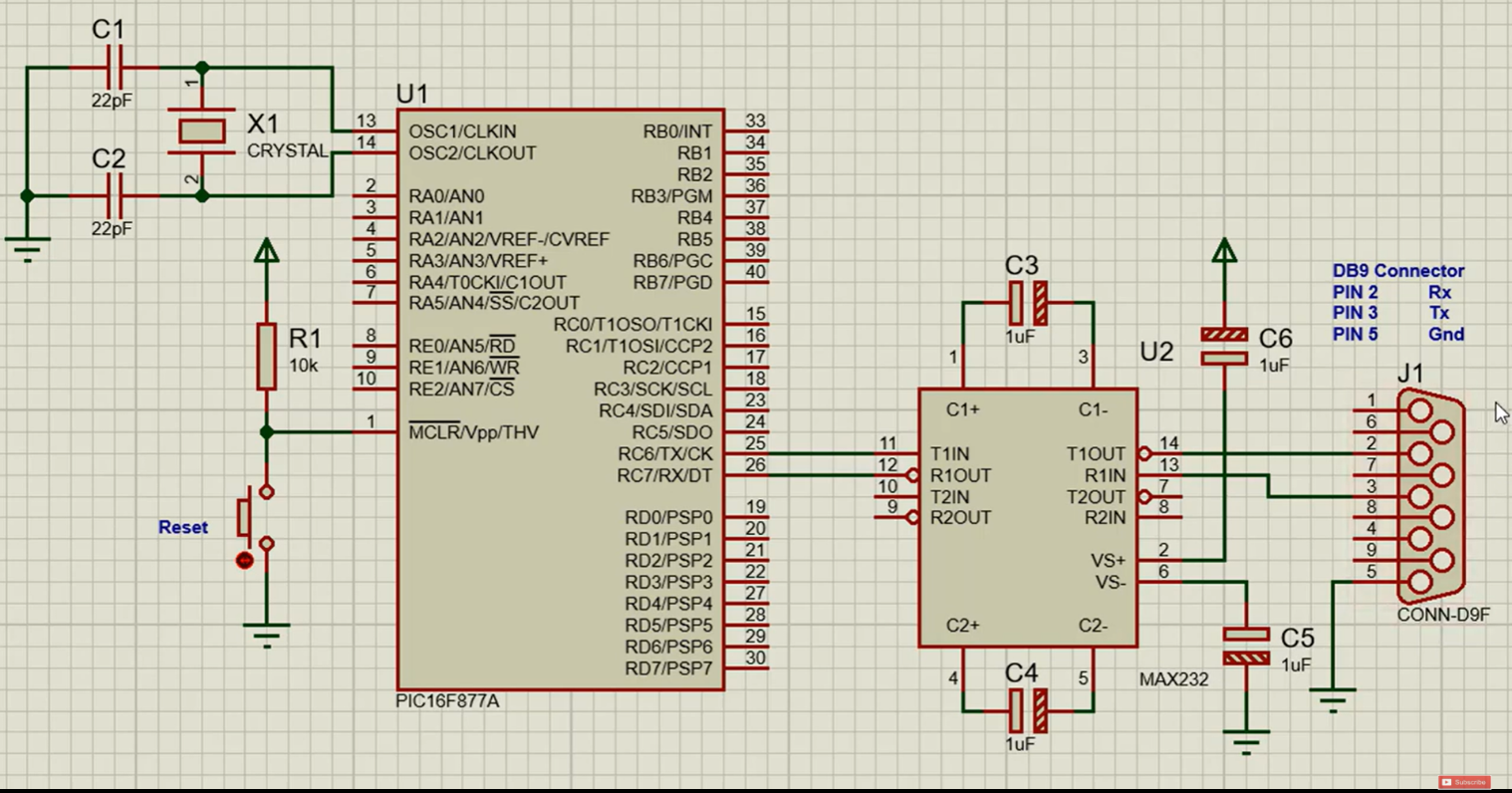
**C5 (between V+ and GND):**

Stabilizes the positive voltage generated by the charge pump.

Ensures a steady positive voltage level for RS232 communication.

**C6 (between V- and GND):**

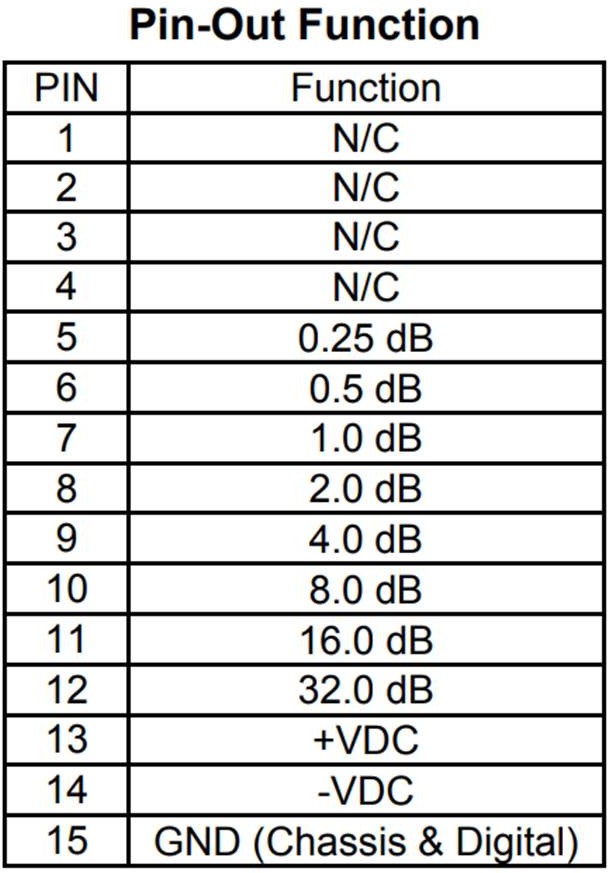
Stabilizes the negative voltage generated by the charge pump.

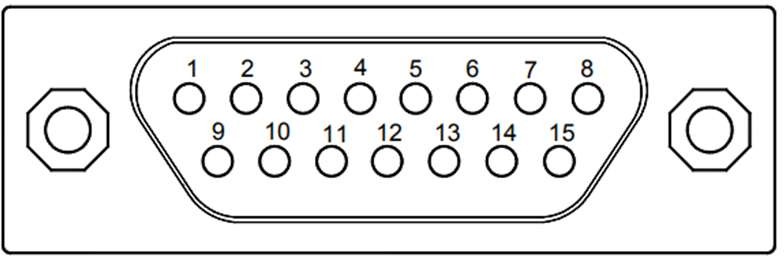
Ensures a steady negative voltage level for RS232 communication.

#### Programmable attenuator- AT-D005-HD:

The AT-D005-HD is a sophisticated programmable attenuator designed to provide precise control over signal attenuation in high-frequency applications. This device is engineered to meet the rigorous demands of modern communication systems, including RF and microwave testing, signal modulation, and automated testing environments. Its programmable nature allows for easy integration into various automated systems, oﬀering seamless control via digital

interfaces. With its high dynamic range and fine resolution, the AT-D005-HD ensures accurate signal management, making it an indispensable tool for engineers and technicians striving for precision in signal control and modulation.





#### PROCEDURE:

###### Circuit Design

* 1. **Microcontroller Setup**:
     + Connect the crystal oscillator and capacitors to the PIC16F877A for clock operation.
     + Connect VDD and VSS to power and ground, respectively.
     + Configure MCLR with a pull-up resistor (10 kΩ).

###### RS232 Interface:

* + - Connect the TX (RC6) and RX (RC7) pins of the PIC16F877A to the corresponding pins on the MAX232 IC.
    - Connect the MAX232 IC to the DB9 connector for serial communication with the PC.
    - Use capacitors (typically 1 µF) with the MAX232 for voltage level conversion.

###### Attenuator Control:

* + - Connect PORTB pins (RB0-RB7) of the PIC16F877A to the control inputs of the programmable attenuator.

###### Software Development

* 1. **Install mikroC PRO for PIC**:
     + Download and install mikroC PRO for PIC from the MikroElektronika website.

###### Write the Code in mikroC PRO:

* + - Open mikroC PRO and create a new project for the PIC16F877A.
    - Use the following code to implement UART communication and control the programmable attenuator
* 2.3 **Compile the Code**:
  + Compile the code using the mikroC PRO for PIC compiler to generate the HEX file.

###### Simulation with Proteus Design Suite

* 1. **Design the Circuit in Proteus**:
     + Open Proteus Design Suite and create a new project.
     + Place the PIC16F877A, MAX232, DB9 connector, crystal oscillator, capacitors, and the programmable attenuator in the workspace.
     + Connect the components according to the circuit design described earlier.

###### Load the HEX File:

* + - Double-click the PIC16F877A in the Proteus workspace to open its properties.
    - Load the compiled HEX file generated by mikroC PRO into the PIC16F877A.

###### Configure the Virtual Terminal:

* + - Place a Virtual Terminal in the workspace to simulate RS232 communication.
    - Connect the Virtual Terminal to the TX and RX pins of the MAX232.

###### Run the Simulation:

* + - Start the simulation in Proteus.
    - Use the Virtual Terminal to send commands to the PIC16F877A and observe the output.
    - Verify that valid commands (0-9, A-F) correctly set the attenuator value and are echoed back.
    - Ensure that invalid commands generate an "Invalid command" message.

###### Testing and Debugging

* 1. **Hardware Setup**:
     + Build the physical circuit on a breadboard, following the same design as in the Proteus simulation.
     + Connect the PC to the RS232 interface using a USB to serial adapter, if necessary.

###### Upload the Code:

* + - Use a PIC programmer (e.g., PICkit 3) to upload the HEX file to the PIC16F877A.

###### Terminal Software Configuration:

* + - Open terminal software (e.g., Tera Term) on the PC and configure it for 9600 bps, 8 data bits, no parity, and 1 stop bit.

###### Send Commands:

* + - Send valid commands (0-9, A-F) from the terminal software.
    - Observe the response from the microcontroller, ensuring the correct attenuator value is set and echoed back.
    - Send invalid commands to test error handling.

#### SOFTWARE CODE:

sbit TX\_PIN at RC6\_bit; sbit RX\_PIN at RC7\_bit;

sbit TX\_PIN\_Direction at TRISC6\_bit; sbit RX\_PIN\_Direction at TRISC7\_bit; char received\_char;

void main() {

char attenuator\_data; TRISC6\_bit = 0;

TRISC7\_bit = 1; UART1\_Init(9600); TRISB = 0x00; PORTB = 0x00;

while (1) {

if (UART1\_Data\_Ready()) { received\_char = UART1\_Read();

if (received\_char >= '0' && received\_char <= '9') { attenuator\_data = received\_char - '0';

PORTB = attenuator\_data; UART1\_Write(received\_char);

} else if (received\_char >= 'A' && received\_char <= 'F') { attenuator\_data = received\_char - 'A' + 10;

PORTB = attenuator\_data; UART1\_Write(received\_char);

} else {

UART1\_Write\_Text("Invalid command\n");

}

}

Delay\_ms(100); // Delay to prevent ﬂooding UART

}

}

#### Code Explanation:

###### Pin Configuration:

* + The TX\_PIN (RC6) and RX\_PIN (RC7) are configured for UART communication.
  + TX\_PIN\_Direction and RX\_PIN\_Direction are used to set the direction of these pins: TRISC6\_bit = 0; sets RC6 as an output, and TRISC7\_bit = 1; sets RC7 as an input.

###### UART Initialization:

* + UART1\_Init(9600); initializes the UART module with a baud rate of 9600 bps to facilitate RS232 communication.

###### Setting Up PORTB:

* + TRISB = 0x00; configures all PORTB pins as outputs, allowing them to control the attenuator.
  + PORTB = 0x00; initializes PORTB to zero.

###### Main Loop:

* + The program enters an infinite loop where it continuously checks for data received from the PC.
  + UART1\_Data\_Ready() checks if new data is available from the PC.
  + If data is received, UART1\_Read() reads the received character.

###### Command Processing:

* + The received character is checked to determine if it represents a valid command to set the attenuator.
  + Characters '0'-'9' are converted from ASCII to integer values 0-9 and used to set PORTB.
  + Characters 'A'-'F' are converted from ASCII to integer values 10-15 and used to set PORTB.
  + The command is echoed back to the PC using UART1\_Write().
  + If an invalid command is received, an error message "Invalid command\n" is sent back to the PC.

###### Delay:

* + Delay\_ms(100); adds a 100 ms delay between iterations to prevent flooding the UART interface with data.