INSERT PICTURES HERE

Description

The VIVA board is a hardware platform designed specifically for students and beginners to learn and experiment with microcontroller-based systems. It is equipped with two (2) microcontrollers and comes with various built-in peripherals and features that make it easier for students to start their journey in embedded systems and programming. If the Arduino bootloader is programmed onto the ATmega328, the ATMEL microcontroller is compatible with many Arduino sketches.

Features

- Microcontrollers: 8-bit, RISC
    - PIC: PIC16F1829
    - ATMEL: ATmega328P-AU
- Hello World! LEDs
- Programming interfaces: JTAG, and ICSP
- Capacitance keypad: UTSA letters
- Buttons: reset and interrupt signals
- Pin breakouts: 0.1” (2.54 mm) pin headers
- Communication bus: I2C
- USB-to-UART modules: CP2102 modules
- OLED module: I2C
- Prototyping area: 0.1” pitch, plated through holes
- VDD selector: 3.3V or 5.0V

Applications

- Learning Embedded Systems
- Programming Practice
- Experimentation and Prototyping
- Understanding Peripherals
- Project Building
- Debugging and Troubleshooting
- Educational Resources
- Cross-Platform Compatibility
- Preparing for Industry

Schematic Images

REV	DESCRIPTION	ECO	DATE
A1	INITIAL RELEASE	1023	

NOTES:
THIS PROJECT WAS CREATED IN SUPPORT OF UTSA ENGINEERING CURRICULUM.

PROJECT REVISION HISTORY:
v1.0, AUG.2014
INITIAL RELEASE

v2.0, JAN.2015
ADDED PROTOTYPING AREA

v3.0, AUG.2015
ADDED HIGH-SIDE TRANSISTOR SWITCH
ADDED ABC CAPACITANCE TOUCH BUTTONS

v4.0, AUG.2018
CONVERTED TO SURFACE MOUNT
REPLACED ABC WITH UTSA FOR CAPACITANCE TOUCH BUTTONS

v5.1, JAN.2022
EXPANDED TO 100 x 100 mm
ADDED ATmega328P
REMOVED HIGH-SIDE TRANSISTOR SWITCH

v5.2, MAY.2022
ADDED ATMEGA BUTTON
CHANGED OLED DISPLAY
ADDED DTR CAPACITOR

v5.3, JUNE.2023
ADDED TO miniPCB PROJECT: PN 11A-001, REV A1
UPDATED ATmega328P PIN BREAKOUTS
UPDATED FOOTPRINTS AND SILKSCREEN FOR HIGHER DENSITY LAYOUT

DATETIME STAMP
7/31/2023 8:48 PM

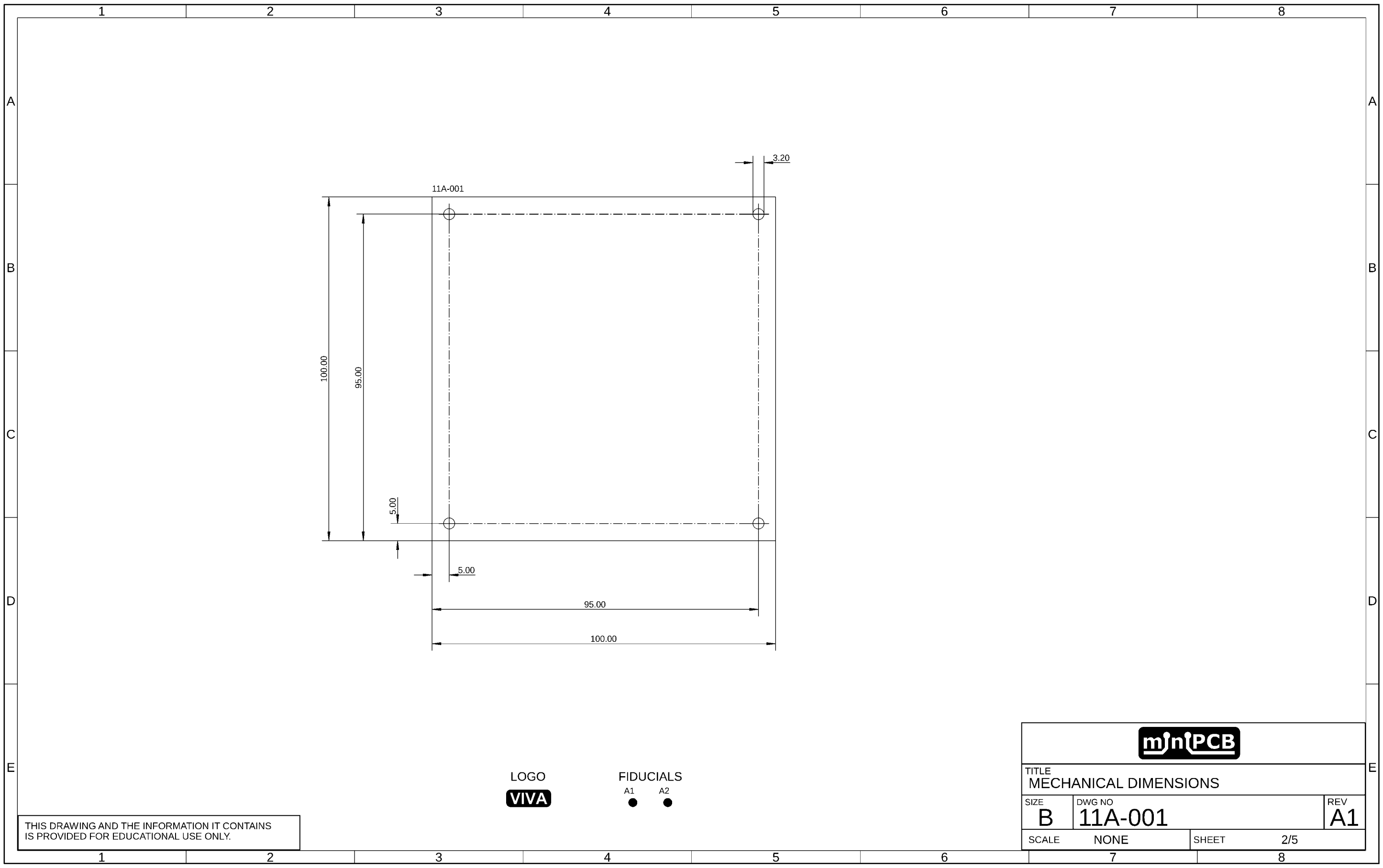
DR	N. MANTEUFEL	DATE 31JUL2023
ENG	P. MORTON	DATE 31JUL2023
QA		DATE

TITLE
VIVA UTSA BOARD

PRINTED CIRCUIT BOARD
COMPUTING
MICROCONTROLLER BOARD
PIC16F1829, ATmega328P-AU

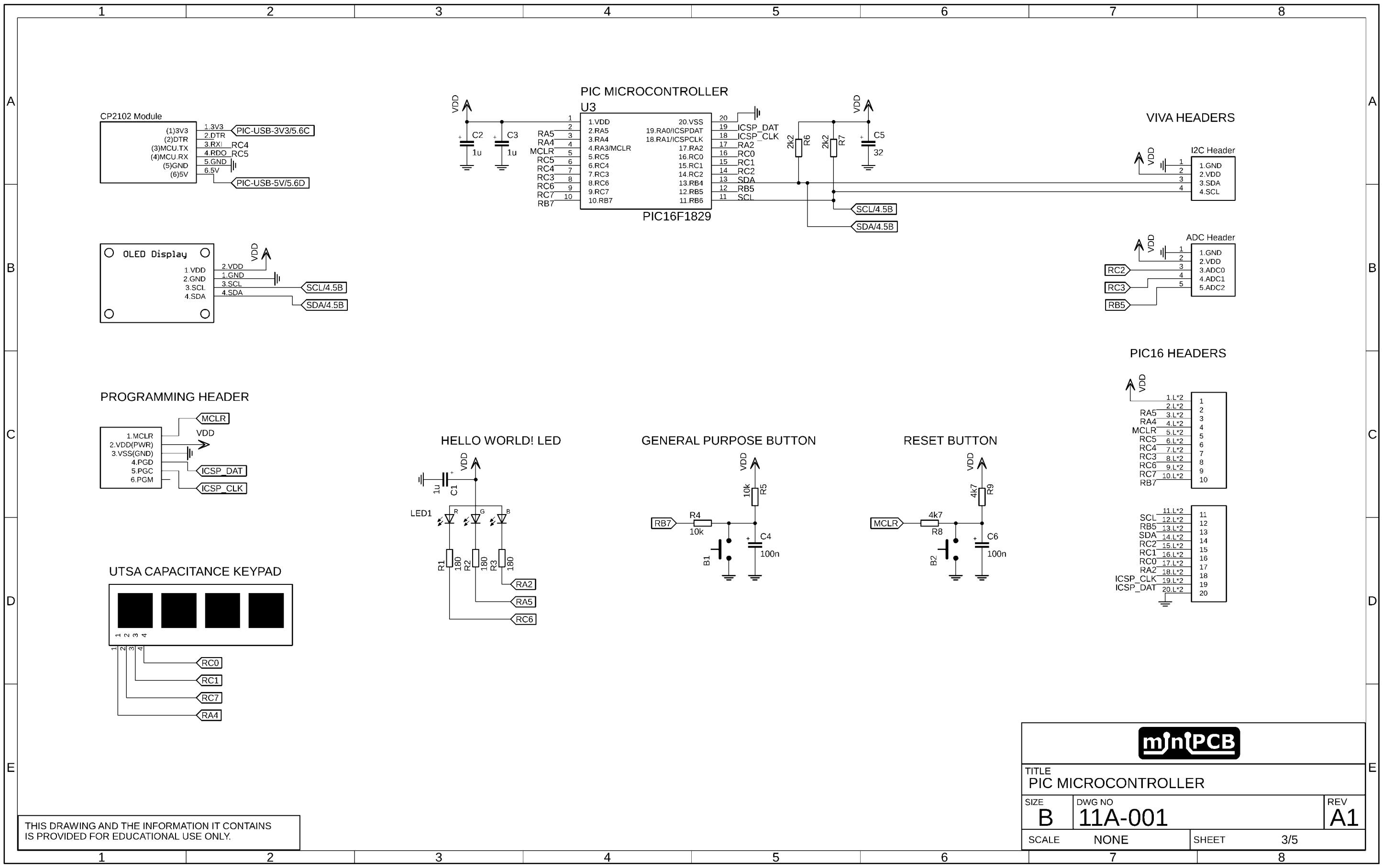
SIZE	DWG NO	REV
B	11A-001	A1

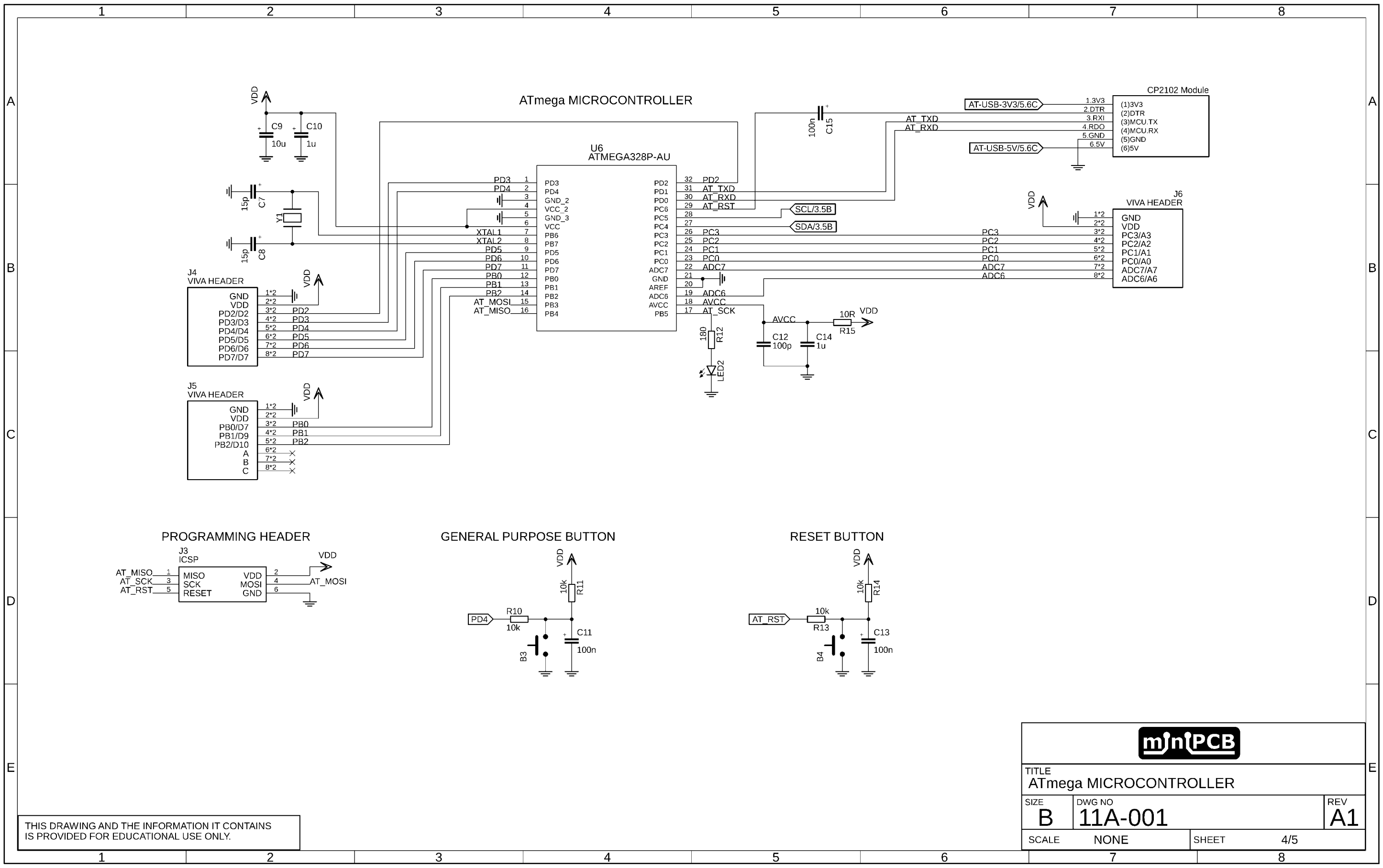
SCALE NONE SHEET 1/5

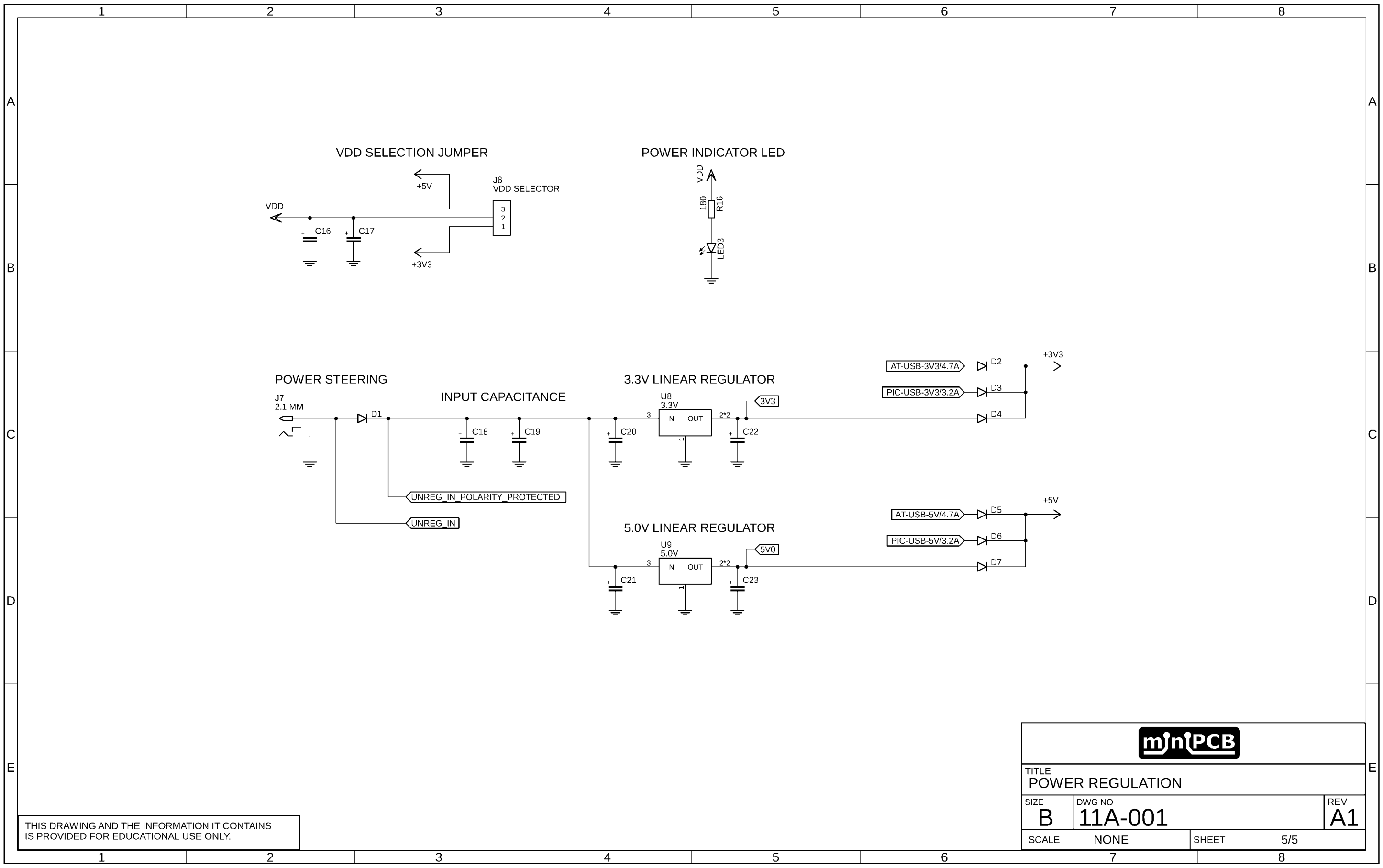

11A-001
3.20
100.00
95.00
5.00
5.00
95.00
100.00
LOGO
VIVA
FIDUCIALS
A1
A2
THIS DRAWING AND THE INFORMATION IT CONTAINS
IS PROVIDED FOR EDUCATIONAL USE ONLY.
TITLE
MECHANICAL DIMENSIONS
SIZE
B
DWG NO
11A-001
REV
A1
SCALE
NONE
SHEET
2/5

ECO: 1023

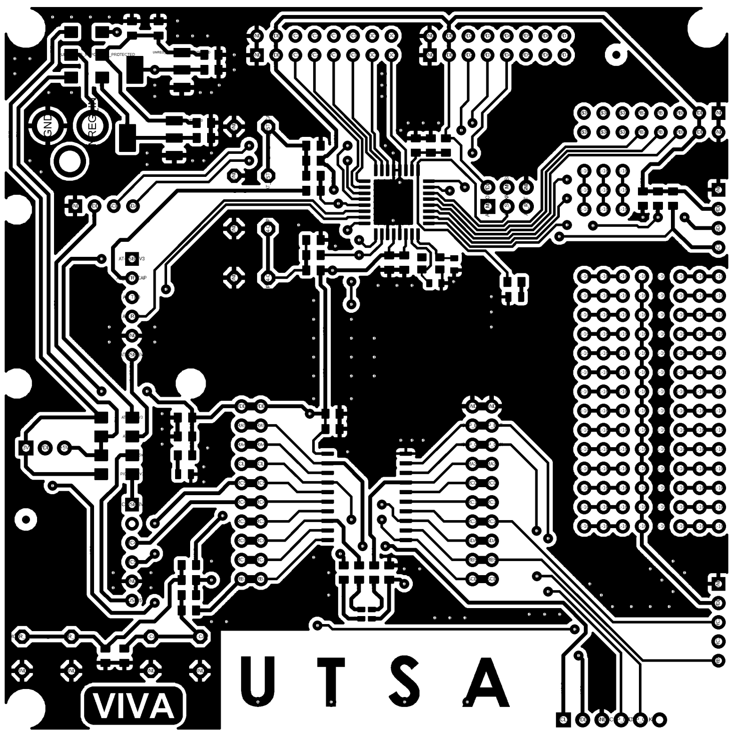
Date Printed: 28 June 2025




ATmega MICROCONTROLLER
U6
ATMEGA328P-AU
CP2102 Module
J6
VIVA HEADER
J4
VIVA HEADER
J5
VIVA HEADER
PROGRAMMING HEADER
J3
ICSP
GENERAL PURPOSE BUTTON
RESET BUTTON
THIS DRAWING AND THE INFORMATION IT CONTAINS
IS PROVIDED FOR EDUCATIONAL USE ONLY.
TITLE
ATmega MICROCONTROLLER
SIZE
B
DWG NO
11A-001
REV
A1
SCALE
NONE
SHEET
4/5

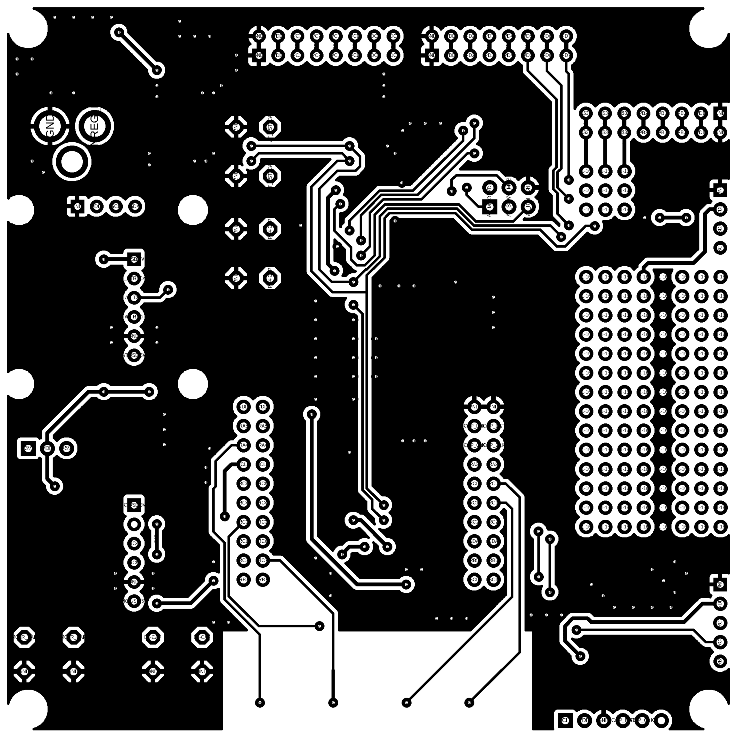

VDD SELECTION JUMPER
+5V
J8
VDD SELECTOR
VDD
C16
C17
+3V3
POWER INDICATOR LED
VDD
180
R16
LED3
POWER STEERING
J7
2.1 MM
D1
INPUT CAPACITANCE
C18
C19
UNREG_IN_POLARITY_PROTECTED
UNREG_IN
3.3V LINEAR REGULATOR
U8
3.3V
IN
OUT
C20
C22
3V3
AT-USB-3V3/4.7A
PIC-USB-3V3/3.2A
D2
D3
D4
+3V3
5.0V LINEAR REGULATOR
U9
5.0V
C21
C23
5V0
AT-USB-5V/4.7A
PIC-USB-5V/3.2A
D5
D6
D7
+5V
THIS DRAWING AND THE INFORMATION IT CONTAINS
IS PROVIDED FOR EDUCATIONAL USE ONLY.
TITLE
POWER REGULATION
SIZE
B
DWG NO
11A-001
REV
A1
SCALE
NONE
SHEET
5/5

Gerber Images



THIS DRAWING AND
THE INFORMATION IT CONTAINS
IS PROVIDED
FOR EDUCATIONAL USE ONLY.

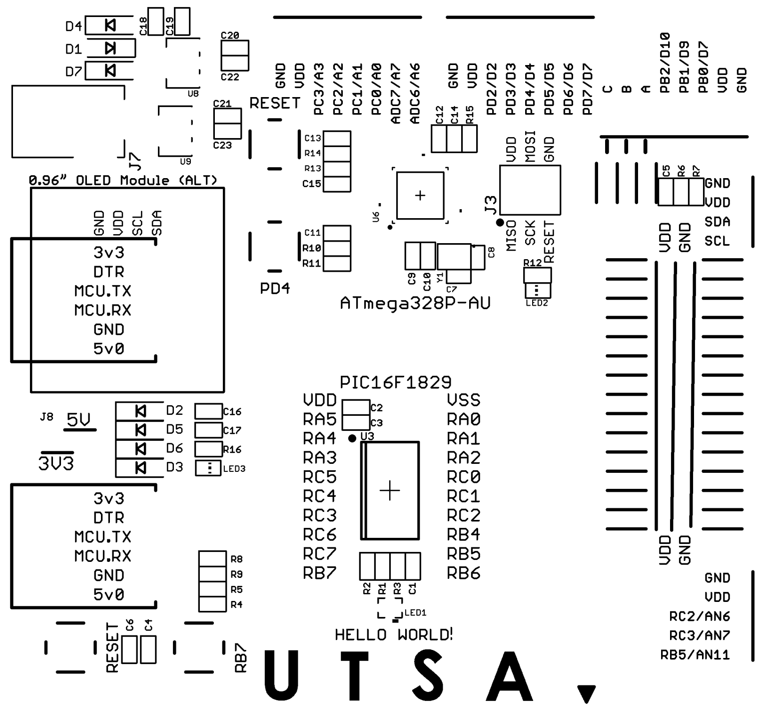
FILE DESCRIPTION	
TOP COPPER	
DWG NO	FILE EXT
11A-001	GLTX



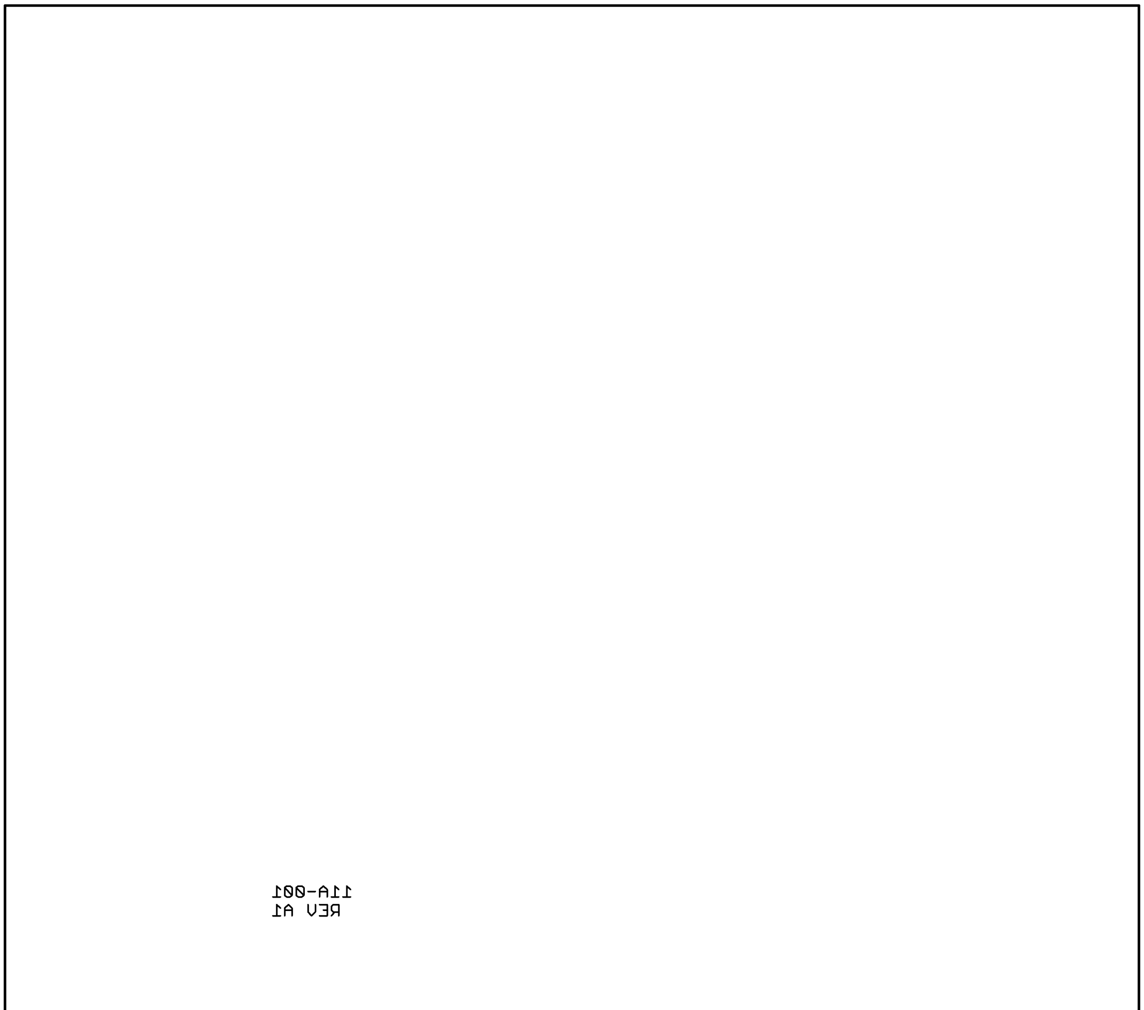
	THIS DRAWING AND THE INFORMATION IT CONTAINS IS PROVIDED FOR EDUCATIONAL USE ONLY.	FILE DESCRIPTION BOTTOM COPPER	
		DWG NO 11A-001	FILE EXT GLBX

ECO: 1023

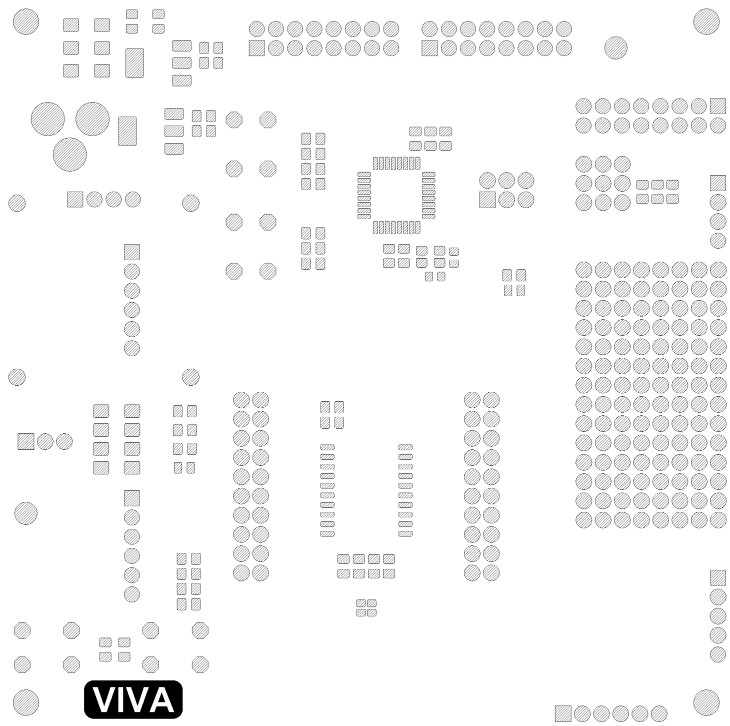
Date Printed: 28 June 2025



miniPCB	THIS DRAWING AND THE INFORMATION IT CONTAINS IS PROVIDED FOR EDUCATIONAL USE ONLY.	FILE DESCRIPTION TOP SILKSCREEN	
		DWG NO 11A-001	FILE EXT GOTX



miniPCB	THIS DRAWING AND THE INFORMATION IT CONTAINS IS PROVIDED FOR EDUCATIONAL USE ONLY.	FILE DESCRIPTION BOTTOM SILKSCREEN	
		DWG NO 11A-001	FILE EXT GOBX

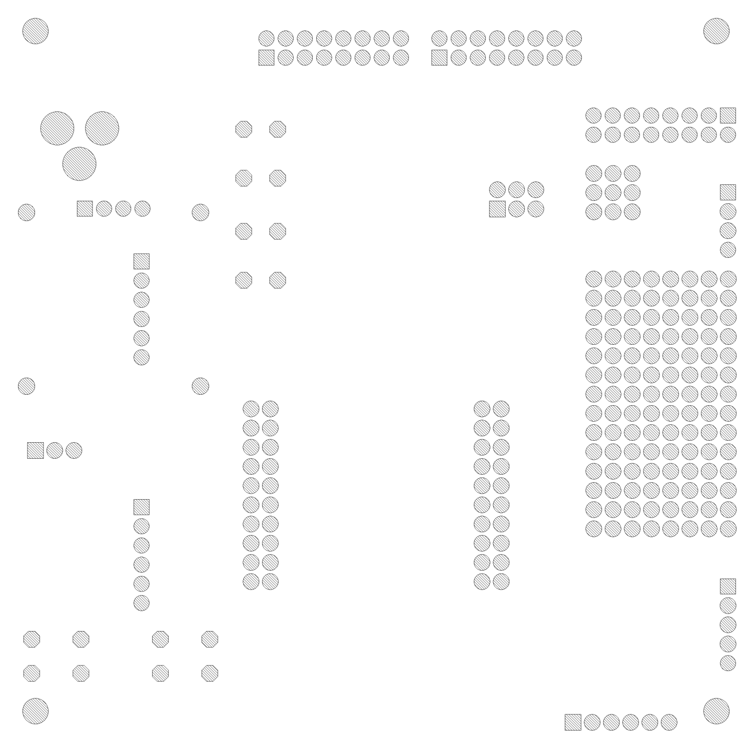


THIS DRAWING AND
THE INFORMATION IT CONTAINS
IS PROVIDED
FOR EDUCATIONAL USE ONLY.

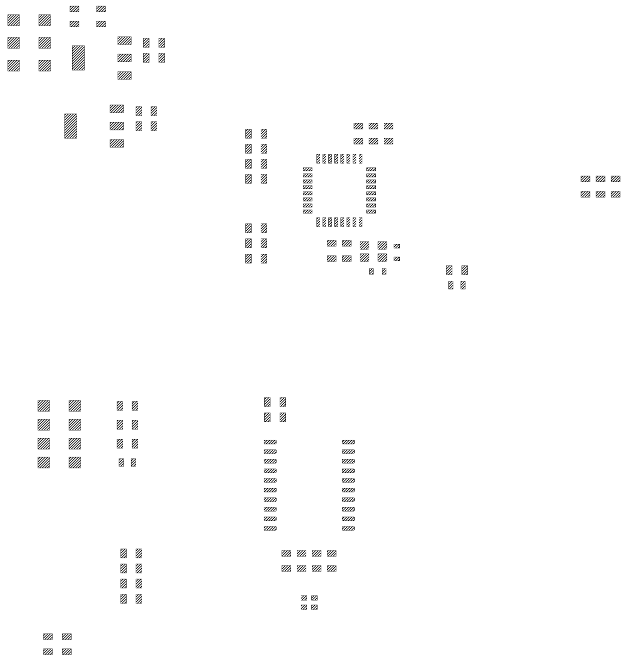
FILE DESCRIPTION
TOP SOLDERMASK

DWG NO
11A-001

FILE EXT
GSTX



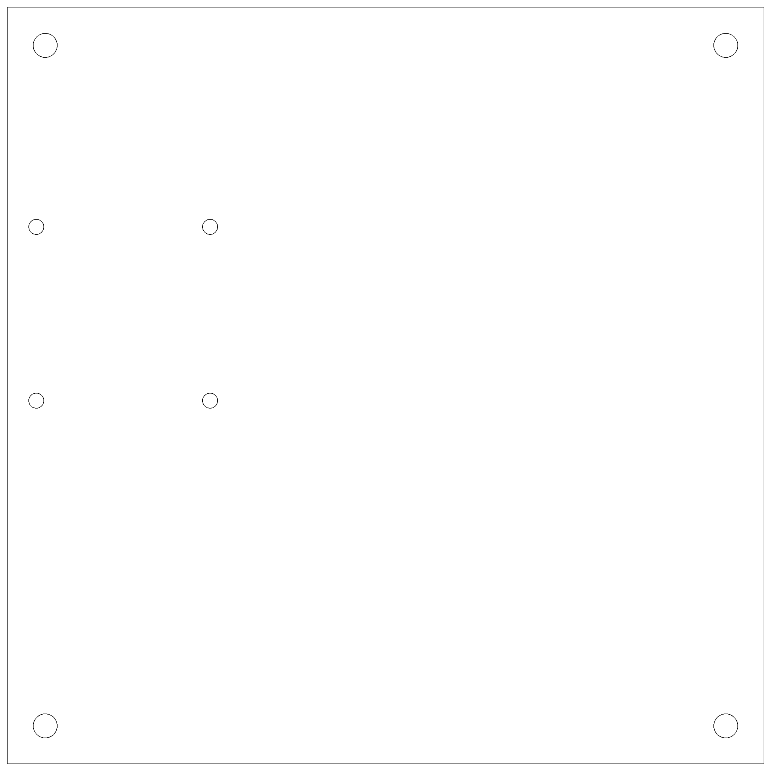
	THIS DRAWING AND THE INFORMATION IT CONTAINS IS PROVIDED FOR EDUCATIONAL USE ONLY.	FILE DESCRIPTION BOTTOM SOLDERMASK	
		DWG NO 11A-001	FILE EXT GSBX



THIS DRAWING AND
THE INFORMATION IT CONTAINS
IS PROVIDED
FOR EDUCATIONAL USE ONLY.

FILE DESCRIPTION TOP CREAM	
DWG NO 11A-001	FILE EXT GCTX

	THIS DRAWING AND THE INFORMATION IT CONTAINS IS PROVIDED FOR EDUCATIONAL USE ONLY.	FILE DESCRIPTION BOTTOM CREAM	
		DWG NO 11A-001	FILE EXT GCBX



	THIS DRAWING AND THE INFORMATION IT CONTAINS IS PROVIDED FOR EDUCATIONAL USE ONLY.	FILE DESCRIPTION EDGE	
		DWG NO 11A-001	FILE EXT GM1

miniPCB	THIS DRAWING AND THE INFORMATION IT CONTAINS IS PROVIDED FOR EDUCATIONAL USE ONLY.	FILE DESCRIPTION V-SCORE	
		DWG NO 11A-001	FILE EXT GM2

THIS DRAWING AND
THE INFORMATION IT CONTAINS
IS PROVIDED
FOR EDUCATIONAL USE ONLY.

FILE DESCRIPTION MILLING	
DWG NO 11A-001	FILE EXT GM3

What does ChatGPT say?

By Nolan Manteufel, 26JUL2023

The PIC16F1829 microcontroller is commonly used in a wide range of applications, including embedded systems, home automation, motor control, industrial automation, sensor interfacing, and more. PIC microcontroller popularity stems from their robustness, affordability, and the extensive community support and resources available for programming and development.

The ATmega328 microcontroller is famously known for being used on the Arduino Uno board, one of the most popular microcontroller development boards for hobbyists, students, and professionals. The Arduino ecosystem, along with the ATmega328, has fostered a large community of enthusiasts who have created countless projects and libraries, making it an excellent choice for anyone interested in learning about embedded systems, electronics, and programming. Due to its versatility and wide-ranging support, the ATmega328 is also found in various other custom-designed boards and commercial products.

The PIC16F1829 and ATmega328 are both manufactured by Microchip Technologies. Microchip's PIC and ATmega microcontroller families are popular for various applications due to their ease of use, low cost, and versatility.

Microchip is a prominent player in the semiconductor industry and is well-known for its PIC microcontrollers and AVR microcontrollers, which are widely used in embedded systems, IoT (Internet of Things) devices, consumer electronics, automotive applications, industrial automation, and more. The company has a strong focus on providing cost-effective and energy-efficient solutions to its customers.

Here are some common uses and benefits of microcontroller student development boards:

Learning Embedded Systems: These boards provide an excellent starting point for students to learn about embedded systems and the fundamentals of microcontroller programming. They offer a hands-on experience, allowing students to interact with the hardware directly.

Programming Practice: Students can practice programming in the language supported by the microcontroller (e.g., C, C++, or assembly language). They can write code to control the board's peripherals, such as LEDs, sensors, motors, and displays.

Experimentation and Prototyping: Development boards are ideal for quickly prototyping and testing ideas. Students can interface the board with various components and sensors to create interactive projects.

Understanding Peripherals: The boards typically come with a range of onboard peripherals such as GPIO (General Purpose Input/Output) pins, timers, UART, ADC, PWM, and more. Students can learn how these peripherals work and how to utilize them effectively.

Project Building: Students can use the development board as the heart of their own projects, building upon the provided features and adding external components as needed.

Debugging and Troubleshooting: The boards often include debugging features like debuggers, built-in LEDs, and UART interfaces, which help students understand and troubleshoot their code and hardware.

Educational Resources: Many development boards are supported by extensive educational resources, tutorials, and example projects. These resources can help students grasp complex concepts and provide inspiration for their projects.

Cross-Platform Compatibility: Some development boards are designed to be compatible with different programming environments and Integrated Development Environments (IDEs), allowing students to work with the tools they are comfortable with.

Preparing for Industry: Familiarity with microcontrollers and development boards is valuable in many engineering fields, such as electronics, robotics, automation, IoT (Internet of Things), and more. Experience with development boards can give students an edge when transitioning to real-world projects and job opportunities.

Are microcontrollers sensitive to ESD?

Yes, microcontrollers are sensitive to Electrostatic Discharge (ESD) and can be damaged if not handled and protected properly. ESD is a sudden and momentary electric current that flows between two objects with different electric potentials when they come into contact and then separate. This discharge can occur due to human contact, friction, or other factors.

Microcontrollers, like other electronic components, consist of sensitive integrated circuits with tiny transistors and other semiconductor elements. These elements are susceptible to damage from high-voltage ESD events. When exposed to an ESD event, the high energy discharge can create a potential difference that exceeds the microcontroller's specified voltage tolerance, leading to permanent damage or degradation of the device.

How to protect against ESD damage?

To protect microcontrollers from ESD damage during manufacturing, handling, and operation, several measures are typically taken:

ESD-Safe Handling: During manufacturing and assembly processes, operators use ESD-safe equipment, including wrist straps, grounded workstations, and anti-static mats to prevent ESD events that could damage the microcontrollers.

Packaging: Microcontrollers are often shipped and stored in anti-static packaging to shield them from ESD during transportation and handling.

Design and Layout: Circuit board designers take ESD protection into account while designing the layout of the microcontroller and its associated components. Proper grounding and placement of ESD protection components, such as TVS diodes or ESD suppressors, can help divert excess voltage away from the sensitive parts.

External Protection: External ESD protection components may be used on input/output pins and other vulnerable areas of the microcontroller to limit the voltage that reaches the internal circuitry during an ESD event.

ESD Testing and Certification: Microcontroller manufacturers subject their products to ESD testing to ensure they meet industry-standard ESD protection requirements.

When working with microcontrollers, it's essential to observe ESD-safe practices to prevent accidental damage. If you need to handle microcontrollers directly, use anti-static wrist straps and avoid working in environments with dry conditions that promote static buildup.

Remember that ESD protection is not only important during manufacturing but also in the field when integrating microcontrollers into end-user products. Proper precautions will help ensure the reliable and long-term operation of microcontroller-based systems.

How to program a bootloader on an ATmega328P-AU

Programming a bootloader on an ATmega328P-AU microcontroller involves several steps. Bootloaders allow you to program the microcontroller over a serial interface (e.g., UART) without using an external programmer. Here's a general guide to help you program a bootloader on an ATmega328P-AU:

Step 1: Choose a Bootloader:

Select a bootloader that is compatible with the ATmega328P-AU microcontroller. There are several popular bootloaders available, such as Optiboot and Arduino's bootloader.

Step 2: Set Up the Development Environment:

To program the bootloader, you'll need an Integrated Development Environment (IDE) and a compatible hardware programmer (e.g., USBasp, Arduino as ISP) to communicate with the ATmega328P-AU.

For Arduino-based development:

Install the Arduino IDE from the official website.

Select "Arduino Uno" or "Arduino Duemilanove w/ ATmega328" board from the "Tools" menu.

Step 3: Connect the Hardware:

Connect your ATmega328P-AU microcontroller to the hardware programmer following the correct pinout and connections for the specific programmer you are using.

Step 4: Burn the Bootloader:

Depending on the bootloader you selected, follow the appropriate steps to burn it onto the ATmega328P-AU. Here's how to do it with Optiboot (assumes you are using Arduino as ISP):

In the Arduino IDE, select "Arduino as ISP" as the programmer from the "Tools" menu.

Ensure that the correct "Board" and "Processor" are selected for the ATmega328P-AU.

Select "Burn Bootloader" from the "Tools" menu.

Step 5: Verify the Bootloader:

After burning the bootloader, verify that it was successfully programmed onto the ATmega328P-AU.

Step 6: Upload Programs via Bootloader:

With the bootloader in place, you can now upload programs to the ATmega328P-AU using the serial interface (e.g., USB-to-Serial adapter or FTDI).

To upload a program:

Connect your USB-to-Serial adapter to the appropriate pins (TX, RX, GND, VCC) on the ATmega328P-AU.

Select the correct board (e.g., "Arduino Uno" or "Arduino Duemilanove w/ ATmega328") and the connected serial port in the Arduino IDE.

Write your Arduino sketch and click "Upload" to program the ATmega328P-AU over the bootloader.

Remember to remove the hardware programmer from the ATmega328P-AU when using the bootloader to upload programs over the serial interface.

Keep in mind that the specific steps may vary slightly based on the bootloader, hardware programmer, and IDE you are using. Always refer to the documentation and resources provided with your selected bootloader and programmer for detailed instructions.

How to program the PIC16F1829

Programming the PIC16F1829 microcontroller involves several steps, including setting up the development environment, writing the code, compiling, and programming the microcontroller. Here's a general guide to help you get started with programming the PIC16F1829:

Step 1: Set Up the Development Environment:

To program the PIC16F1829, you'll need a suitable development environment and a programmer. Microchip's MPLAB X IDE is commonly used for PIC microcontroller development.

Download and install MPLAB X IDE from the Microchip website.

Install the required compiler, either XC8 (C compiler) or MPASM (assembly language compiler).

Step 2: Create a New Project:

Launch MPLAB X IDE, and create a new project.

Select the PIC16F1829 as the target microcontroller for the project.

Step 3: Write the Code:

In the project, create a new source file (e.g., main.c) and write your C or assembly code for the microcontroller. Include the necessary header files for the PIC16F1829.

Step 4: Compile the Code:

Click on the "Build Project" option in MPLAB X IDE to compile your code. Ensure there are no errors or warnings in the build output.

Step 5: Configure the Hardware:

Set up your hardware with the PIC16F1829 microcontroller. Connect the necessary components, such as power supply, crystal oscillator, and other peripherals, based on your application requirements.

Step 6: Program the PIC16F1829:

Connect the PIC16F1829 to a suitable programmer, such as PICkit or ICD, using the appropriate programming interface (ICSP or ICSP2).

In MPLAB X IDE, select the programmer and the target device (PIC16F1829).

Click on the "Program" option to flash the compiled code onto the PIC16F1829.

Step 7: Verify the Code Execution:

After programming the microcontroller, disconnect the programmer and connect your hardware circuit as required.

Power up the hardware and verify that your PIC16F1829 is running the programmed code as expected.

Step 8: Debugging (Optional):

MPLAB X IDE provides debugging capabilities to help you identify and fix issues in your code. You can use breakpoints, watch variables, and other debugging features for this purpose.

Step 9: Test and Iterate:

Test your microcontroller application thoroughly to ensure it meets the desired functionality and requirements.

Iterate and make any necessary changes to your code based on testing results.

Remember to refer to the PIC16F1829 datasheet and the documentation provided by Microchip for detailed information on its features, pin configuration, and programming requirements. The steps outlined above are a general guideline, and the specific process may vary depending on your chosen development environment and programming tools.

Can you generate a hello world program for a PIC16F1829?

Sure! Here's a simple "Hello, World!" program in C for the PIC16F1829 microcontroller. This program will make the microcontroller blink an LED to say "Hello" repeatedly.

```
#include <xc.h>

// Configuration bits for the PIC16F1829
#pragma config FOSC = INTOSCIO  // Internal oscillator, port function on RA6 and RA7
#pragma config WDTE = OFF       // Watchdog Timer disabled
#pragma config PWRTE = OFF      // Power-up Timer disabled
#pragma config MCLRE = OFF      // MCLR/VPP pin function is digital input
#pragma config BOREN = OFF      // Brown-out Reset disabled
#pragma config CP = OFF         // Code Protection disabled
#pragma config CPD = OFF        // Data Code Protection disabled
#pragma config LVP = OFF        // Low Voltage Programming disabled

#define _XTAL_FREQ 8000000      // Internal oscillator frequency in Hz

void main() {
   // Set the internal oscillator frequency to 8MHz
   OSCCONbits.IRCF = 0b111;

   // Set RA0 as an output (LED)
   TRISA0 = 0;

   while (1) {
      // Turn on the LED
      RA0 = 1;

      // Delay for a short period
      __delay_ms(500);

      // Turn off the LED
      RA0 = 0;

      // Delay for a short period
      __delay_ms(500);
   }
}
```

In this code, we set up the PIC16F1829 to use its internal oscillator at 8MHz (_XTAL_FREQ), and we configure RA0 as an output to drive an LED. The __delay_ms() function is used to create a delay between turning on and off the LED, creating a blinking effect.

To compile and program this code into the PIC16F1829, you'll need to use MPLAB X IDE along with the XC8 compiler, as mentioned in the previous responses. Additionally, ensure that you have connected an LED and a current-limiting resistor (e.g., 330Ω) to RA0 to see the blinking output.

Revision History

REV	DESCRIPTION	ECO	DATE
A	Initial Release	1023	