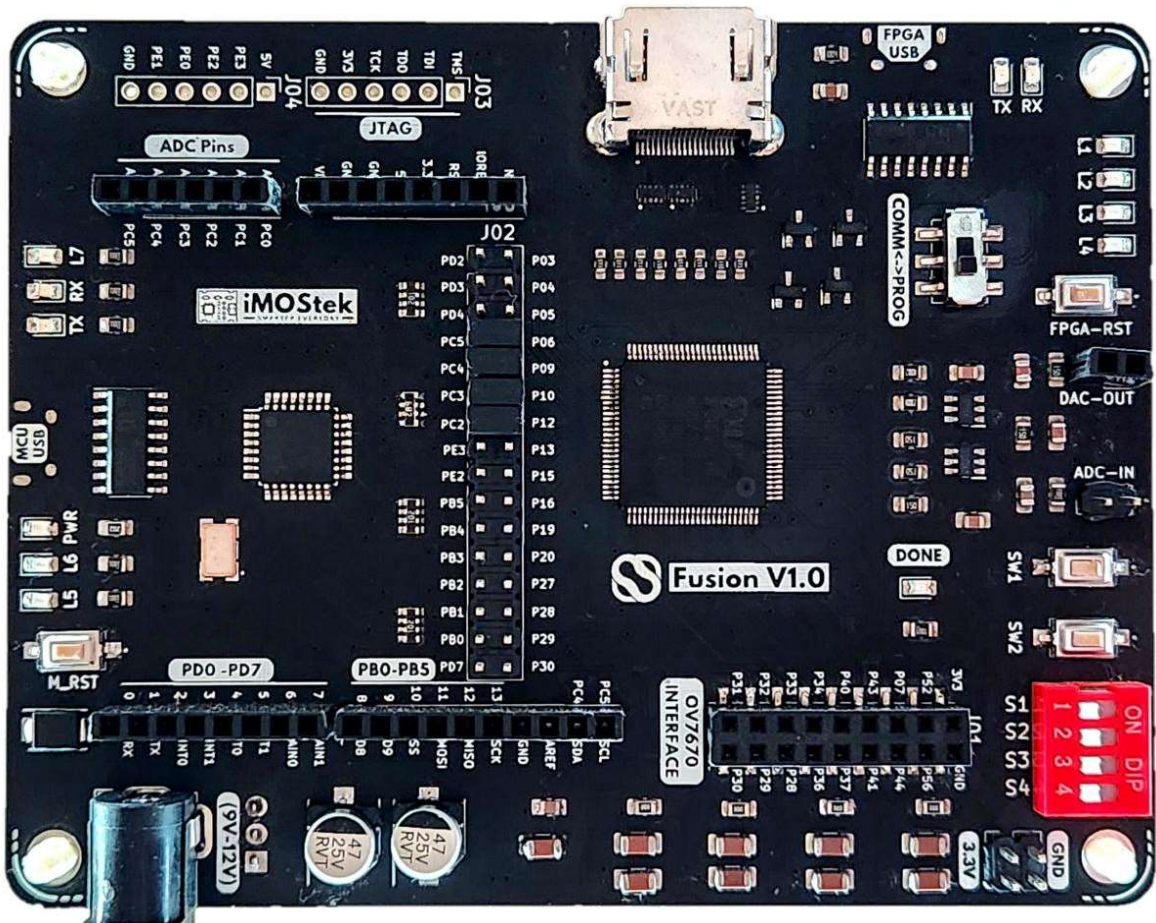


# Fusion 1.0 EVM

## Hardware and Software User's Guide



Document Revision 1.0

1/1/2024

## Table of Contents

<b>CHAPTER 1 AN INTRODUCTION TO FUSION V1.0 BOARD</b> .....	<b>1-4</b>
<i>PURPOSE</i> .....	1-4
<i>FUSION V1.0 HARDWARE SPECIFICATION</i> .....	1-6
➤ Hardware on FPGA Side: .....	1-6
➤ Hardware on Microcontroller Side:.....	1-7
<b>CHAPTER 2 CIRCUIT DESCRIPTION</b> .....	<b>2-8</b>
<i>BOARD LAYOUT</i> .....	2-8
Figure2-1 Highlight of the MCU and FPGA Sections on the PCB.....	2-8
Figure2-2. Overview of all Input Output Access Points on the Board.....	2-9
Table2-1. Microcontroller All Connection Details.....	2-10
Figure 2-3. ATMEGA328PB All Pin-Out Schematic .....	2-12
Table2-2. FPGA All Connection Details .....	2-13
<i>POWER SUPPLY</i> .....	2-15
➤ Supply Options: .....	2-15
Power Distribution Scheme: .....	2-15
Figure 2-4. Board segment showing all linear voltage regulators.....	2-15
Figure 2-5. Comprehensive power management of the board .....	2-16
➤ Clock Input.....	2-16
➤ Peripherals .....	2-16
Figure 2-6. ADC low pass filter line diagram.....	2-17
Figure 2-7. DAC low pass filter line diagram.....	2-18
Figure 2-8. Flash Memory Connection Typical Circuit.....	2-18
Figure 2-9. HDMI peripheral connection .....	2-19
Figure 2-10. Microcontroller-FPGA Communication Header Typical Circuit.....	2-21
<b>CHAPTER 3 GETTING STARTED WITH FUSION V1.0</b> .....	<b>3-22</b>
<i>INSTALLATION AND GETTING STATED WITH THE FUSION SERIAL PROGRAMMER</i> .....	3-22

Figure3-1. Found New Hardware.....	3-23
Figure3-2. Serial Programmer interface before connecting to FPGA Controller .....	3-23
Figure3-3. Programmer after connection establishment with FPGA Controller .....	3-24
Figure3-4. Code upload to on-board FPGA .....	3-25
Figure3-5. Code upload to on-board FPGA and Code Memory.....	3-26
Figure3-6. Pop-ups after successful code upload .....	3-27
Figure3-7. Updated Memory Segment 1 after successful code upload .....	3-27
Figure3-8. Uploading Code to FPGA from Memory Segment 1 .....	3-28
Figure3-9. Post Code Upload interface showing updated Power-on upload settings .....	3-29
Figure3-10. Serial Plotter Interface after Connection establishment .....	3-30
<i>CHECKING FUSION BOARD FPGA SECTION USING FACTORY DEFAULT FPGA CODE.....</i>	<i>3-30</i>
Figure3-11. Serial Programmer showing Board status after connection setup .....	3-31
<i>SET-UP ARDUINO IDE FOR FUSION BOARD MCU .....</i>	<i>3-32</i>
<b>CHAPTER 4 ADDITIONAL DETAILS .....</b>	<b>3-36</b>
<i>TABLE 4-1: TEXT FORM FOR FPGA “.UCF” FILE FOR ALL PORTS AVAILABLE ON FUSION V1.0 .....</i>	<i>3-36</i>
<i>REFERENCES.....</i>	<i>3-37</i>

## Chapter 1 An Introduction to Fusion V1.0 Board

This user's manual gives detailed overview of the development board and provides a general description of the features and functions to be considered while using this module. This user's manual applies to "Fusion V1.0" development board.

### Purpose

The Fusion Board series as the name suggests helps get the best of both word as it combines the prowess of Field-Programmable Gate Array (FPGA) with a Microcontroller (MCU). FPGA and MCU are very different kind of data processing ICs The purpose of using a FPGA alongside a microcontroller is to leverage the strengths of both technologies in a single system.

An FPGA, or Field-Programmable Gate Array, is a type of integrated circuit (IC) that is designed to be programmable by the user or designer after manufacturing. Unlike application-specific integrated circuits (ASICs) that are custom-designed for specific functions, FPGAs are flexible and can be reprogrammed or reconfigured to perform various digital logic functions. At its core, an FPGA consists of a large number of configurable logic blocks (CLBs) and programmable interconnects. These CLBs contain look-up tables (LUTs), flip-flops, multiplexers, and other elements that can be configured to implement various digital functions. The interconnects allow for the connection and communication between these logic blocks. The following describes few roles where FPGA is better suited at:

- ◇ Hardware customization: FPGAs offer the ability to implement custom digital logic circuits at the hardware level. This allows for the creation of highly specialized and optimized hardware accelerators or interfaces that can greatly enhance the performance and efficiency of the overall system.
- ◇ Real-time processing: FPGAs are known for their parallel processing capabilities and low-latency operation. By offloading time-critical or computationally intensive tasks to the FPGA, the microcontroller can focus on higher-level control tasks, while the FPGA handles real-time data processing or complex algorithm implementation.
- ◇ Interface flexibility: FPGAs can be used to implement a wide range of communication interfaces, such as high-speed serial protocols (e.g., PCIe, Ethernet) or custom protocols. This enables seamless integration with other devices or systems, providing enhanced connectivity options.
- ◇ System integration: FPGA-based systems can incorporate multiple peripherals, such as ADCs, DACs, sensor interfaces, and memory controllers, within a single chip. This integration can reduce component count, board space, and power consumption, leading to more compact and efficient designs.

- ◇ Prototyping and development: FPGAs offer the flexibility to rapidly prototype and test hardware designs. They allow for iterative development and debugging of custom circuits or algorithms, which can be advantageous in research, development, and proof-of-concept stages.
- ◇ Future-proofing: Since FPGAs can be reprogrammed, they provide the ability to adapt and update the system's functionality even after deployment. This flexibility can be valuable in scenarios where requirements change or new features need to be added without redesigning the entire hardware.

Now to talk about a Microcontroller or MCU, A microcontroller is a small integrated circuit (IC) that combines a microprocessor core with several peripherals and memory on a single chip. It is designed to control and manage a specific task or set of tasks in embedded systems. So, while your FPGA is busy working on complex real time operation the MCU can do the following with ease.

- ◇ System management: The microcontroller acts as the central processing unit responsible for overall system management and control. It handles tasks such as initializing the system, managing power, handling interrupts, and coordinating communication between different components.
- ◇ Communication with peripherals: Microcontrollers typically have built-in interfaces for various peripherals like UART, SPI, I2C, USB, Ethernet, etc. These interfaces enable communication with external devices such as sensors, actuators, memory modules, or other systems. The microcontroller manages the communication protocols and data exchange between the FPGA and these peripherals.
- ◇ Higher-level processing: While FPGAs excel at low-level parallel processing and hardware acceleration, microcontrollers are better suited for sequential processing and complex high-level control algorithm execution. They can handle tasks like data parsing, decision-making, state management, and executing control algorithms that require sequential or event-driven processing.
- ◇ System integration: Microcontrollers often handle the integration of the FPGA with other system components. They facilitate communication and data exchange between the FPGA and other on-board peripherals or external systems. They can also handle tasks like data buffering, protocol conversion, or data preprocessing before passing it to the FPGA or receiving data from the FPGA.
- ◇ Firmware management: Microcontrollers typically execute firmware or software code that provides the system's functionality. They handle tasks such as firmware loading, updating, and version control, ensuring the system operates with the desired software and can be easily maintained or upgraded.
- ◇ Power management: Microcontrollers can implement power-saving features and algorithms to optimize energy consumption in the system. They can control the

power supply to different components, enter low-power modes, and manage sleep/wake cycles to extend battery life or reduce power consumption.

In summary, in the combination of an FPGA and a while the FPGA handles specialized hardware tasks and accelerates specific operations, the MCU complements the FPGA by providing higher-level processing, system management, communication interfaces, user interaction, and integration with other components. It helps orchestrate the overall operation of the system. The combination of both technologies enables efficient and flexible system design making it a powerful solution for various applications, including embedded systems, robotics, signal processing, and high-performance computing.

## Fusion V1.0 Hardware Specification

The “Fusion V1.0” is a lightweight integrated FPGA-MCU development board; it is based on the Xilinx Spartan-3A FPGA and Microchip Atmega328 MCU. The board contains several peripherals connected directly with the FPGA, while the MCU circuit is identical to the “Arduino UNO” having the same form factor. Hence, you can plug-in all Arduino shield directly on MCU side. The FPGA connects to MCU using 16 I/O lines through Jumper/Header. So the board can be used as a stand-alone FPGA development board or Microcontroller Development board as well. There are many benefits of integrating the FPGA and MCU circuit, we shall learn them at later part of the document. The following summarizes all the in-built hardware features of “Fusion V1.0”:

### ➤ Hardware on FPGA Side:

- Xilinx XC3S50A FPGA (Upgradable to XC3S200A)
- ADC80S051C 8bit 500ksps SPI ADC (Upgradable to 10/12 bit ADC of same series)
- MCP4725 12bit 400ksps I2C DAC
- W25Q32JVSS 32Mbits QSPI Flash
- HDMI Connector for direct streaming to TV/Monitors with HDMI Capability.
- 18Pin Female Header directly compatible with OV7670 Camera Module.
- MicroUSB Connection for Serial-to-USB Communication with PC/Mobile
- 1 FPGA Reset and 2 User TACT Switches
- 1 4-Channel Slide Switch
- 4 Green User LEDs, 2 Yellow Serial Communication LEDs and 1 Blue DONE LED.
- 1 DPDT Switch to toggle between Programming and Communication
- 16Pin Male Header for connection with MCU
- 6 Pin Header for JTAG Programming

### ➤ **Hardware on Microcontroller Side:**

---

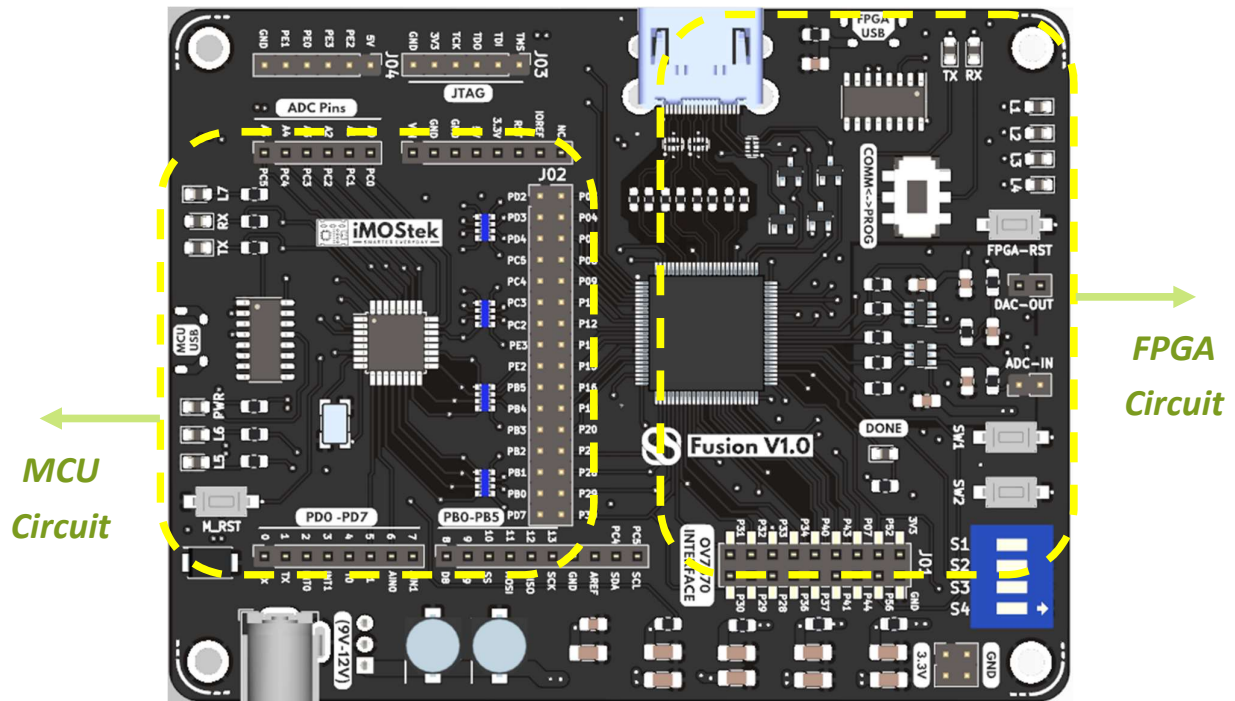
- Microchip Atmega328 Microcontroller (Bootloader Uploaded).
- Headers for Power, ADC and I/O lines Identical to Arduino Uno.
- 6Pin Extra I/O Header.
- MicroUSB for power and MCU Communication with PC/Mobile.
- 3 User LEDs
- 1 Tact Switch for MCU Reset



## Chapter 2 Circuit Description

### Board Layout

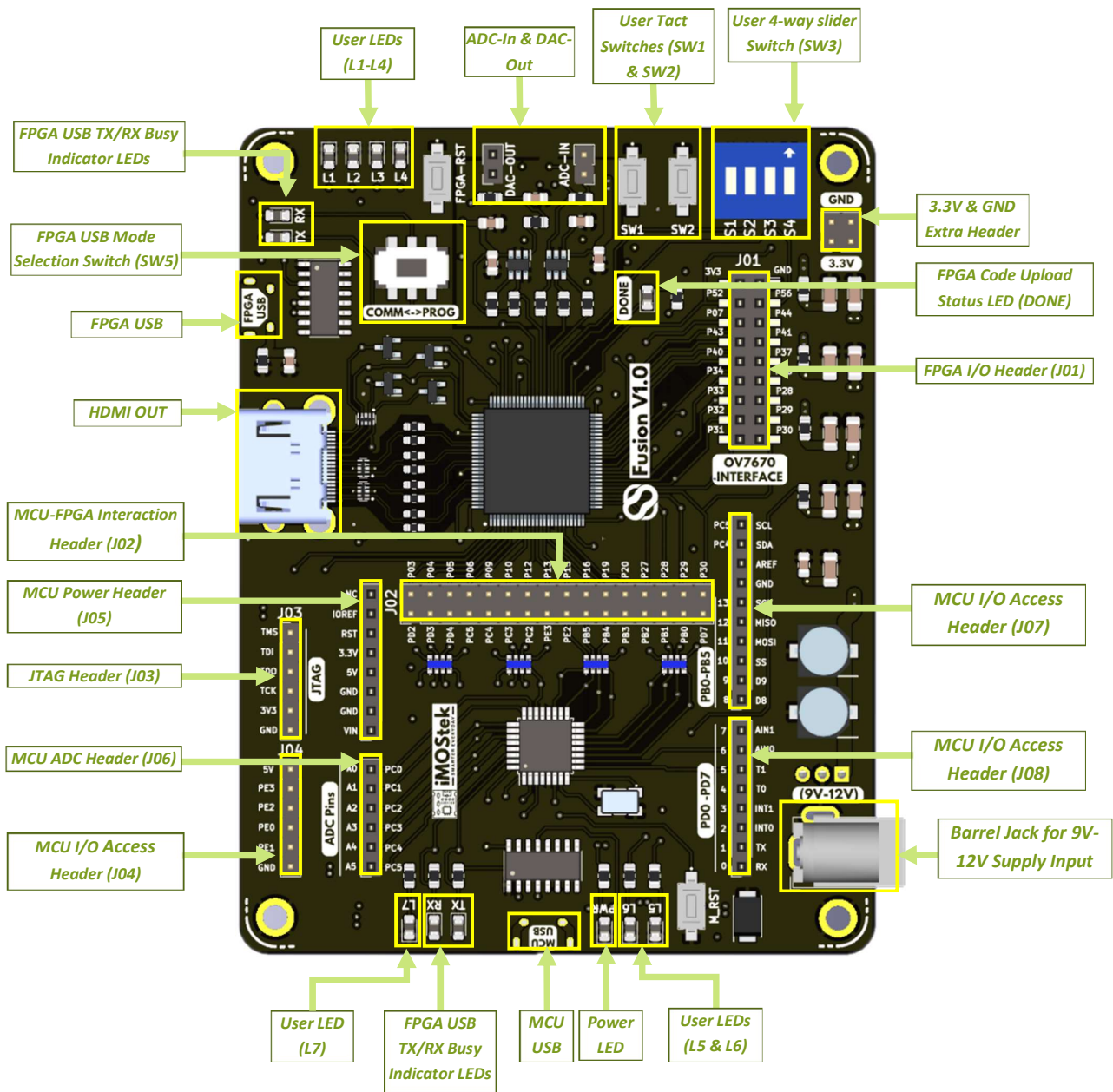
Figure2-1 Highlight of the MCU and FPGA Sections on the PCB



In fig.2-1 we have differentiated the MCU circuit and the FPGA circuit zones. Either of them has its own USB for serial communication. But the MCU side USB-in line can power up the whole board. The powering up precautions require proper elaboration so we shall discuss about power options in separate power section.



**Figure2-2. Overview of all Input Output Access Points on the Board**



The development board provides a versatile and robust platform for electronics development, offering a wide range of essential components such as ADC, DAC, Flash Memory, HDMI, and various headers. These components allow developers to explore and learn electronics development effectively. The board facilitates interfacing with the physical world, capturing and processing analog signals, generating analog outputs, storing data, and connecting to different input and output devices. This enables rapid prototyping and system development.

- ◇ In the following tables, a mapping of all I/O pins for both the microcontroller (MCU) and the FPGA is provided. This mapping helps users understand the available input/output options and facilitates the connection of external devices or components to the board.

**Table2-1. Microcontroller All Connection Details**

<b>a. J06 Header ADC Pin Details</b>						
Header Pin No.	MCU Pin No.	I/O Port ID	Digital Pin No.	IDE Pin No.	Sp. Func. 1	Sp. Func. 2
1	23	PC0	08	14	ADC0	MISO1
2	24	PC1	09	15	ADC1	SCK1
3	25	PC2	10	16	ADC2	-
4	26	PC3	11	17	ADC3	-
5	27	PC4	12	18	ADC4	SDA0
6	28	PC5	13	19	ADC5	SCL0
<b>b. J07 Header I/O Access Pin Details</b>						
Header Pin No.	MCU Pin No.	I/O Port ID	Digital Pin No.	IDE Pin No.	Sp. Func. 1	Sp. Func. 2
1	28	PC5	13	19	SCL0	LED6
2	27	PC4	12	18	SDA0	-
3	20	-	-	-	AREF	-
4	05	-	-	-	GND	-
5	17	PB5	15	13	SCK0	-
6	16	PB4	14	12	MISO0	-
7	15	PB3	03	11	MOSI0	-
8	14	PB2	02	10	SS0	-
9	13	PB1	01	9	-	-
10	12	PB0	00	8	-	-
<b>c. J08 Header I/O Access Pin Details</b>						
Header Pin No.	MCU Pin No.	I/O Port ID	Digital Pin No.	IDE Pin No.	Pin Function1	Pin Function2
1	11	PD7	23	7	AIN1	-
2	10	PD6	22	6	AIN0	-
3	09	PD5	21	5	T1	-
4	02	PD4	20	4	T0	-
5	01	PD3	19	3	INT1	-
6	32	PD2	18	2	INT0	-
7	31	PD1	17	1	RX	-

8	30	PD0	16	0	TX	-
---	----	-----	----	---	----	---

**d. J04 Header Extra I/O Access Pin Details (For Microchip ATMEGA328PB Only)**

Header Pin No.	MCU Pin No.	I/O Port ID	Digital Pin No.	IDE Pin No.	Pin Function1	Pin Function2
1	-	-	-		5V	-
2	19	PE3	26	20	SS1	ADC6
3	22	PE2	27	21	MOSI1	ADC7
4	03	PE0	-	22	SDA1	-
5	06	PE1	-	23	SCL1	-
6	-	-	-		GND	-

**e. J02 Header MCU-FPGA Communication lines**

Header Pin No.	MCU Pin No.	I/O Port ID	IDE Pin No.	FPGA Pin No.	UCF Ref. No.
1	32	PD2	2	P03	FP_UC_01
2	01	PD3	3	P04	FP_UC_02
3	02	PD4	4	P05	FP_UC_03
4	09	PC5	19	P06	FP_UC_04
5	27	PC4	18	P09	FP_UC_05
6	26	PC3	17	P10	FP_UC_06
7	25	PC2	16	P12	FP_UC_07
8	22	PE3	21	P13	FP_UC_08
9	19	PE2	20	P15	FP_UC_09
10	17	PB5	13	P16	FP_UC_10
11	16	PB4	12	P19	FP_UC_11
12	15	PB3	11	P20	FP_UC_12
13	14	PB2	10	P27	FP_UC_13
14	13	PB1	9	P28	FP_UC_14
15	12	PB0	8	P29	FP_UC_15
16	11	PD7	30	P30	FP_UC_16

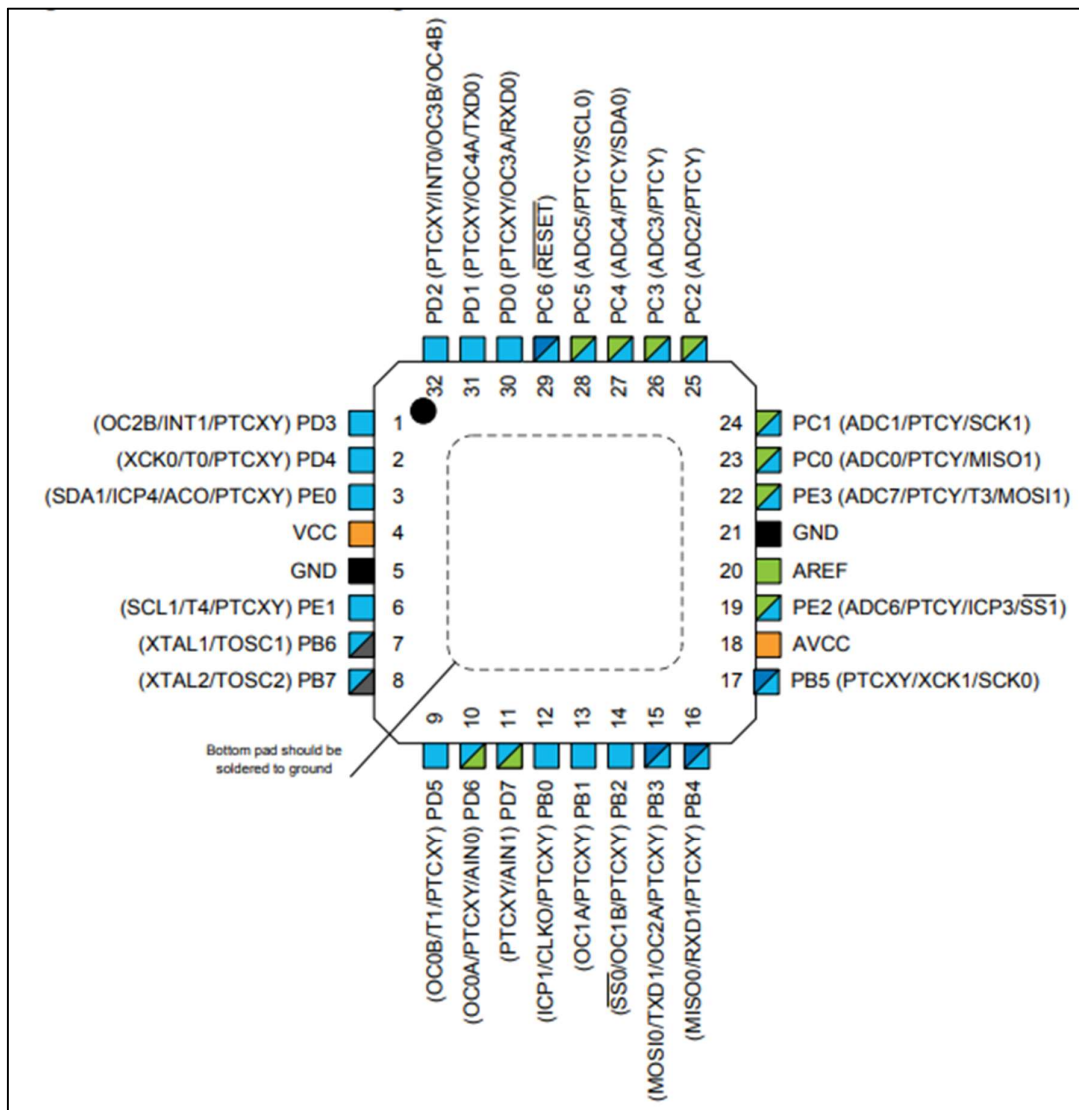
**f. Other peripheral connections with Microcontroller**

Peripheral Name	Peripheral Pin	IDE Pin No.	MCU Pin No.	I/O Port ID
LED (L5)	Anode	4	02	PD4
LED (L6)	Anode	13	17	PB5
LED (L7)	Anode	23	06	PE1
CH340	TX_Data	0	30	PD0
CH340	RX_Data	1	31	PD1
M_RST	Reset	-	29	PC6

**g. J05 Header Power Pin Details**

Header Pin No.	MCU Pin No.	I/O Port ID	Digital Pin No.	Pin Function	Voltage
1	-	-	-	No Connection	NC
2	-	-	-	IOREF	5V
3	29	-	-	RESET	-
4	-	-	-	Supply	3.3V
5	-	-	-	Supply	5V
6	-	-	-	Ground	GND
7	-	-	-	Ground	GND
8	-	-	-	Supply (Vin)	9V-12V

**Figure 2-3. ATMEGA328PB All Pin-Out Schematic**



**Table2-2. FPGA All Connection Details**

<b>a. J02 Header MCU-FPGA Communication lines (Details in Table 2-1)</b>				
<b>J01 Header I/O Lines Details (OV7670 Camera Module Connection)</b>				
Header Pin No.	Pin Type	FPGA Pin No.	FPGA Bank	UCP Pin Name
1	3.3V Supply	-	-	-
2	Ground	-	-	-
3	I/O/I2C Line*	52	1	DIP_SW4
4	I/O/I2C Line*	56	2	DIP_SW3
5	Input (0-3.3V)	07	3	GPI1
6	I/O (0-3.3V)	44	2	GPIO2
7		43	2	GPIO3
8		41	2	GPIO4
9		40	2	GPIO5
10		37	2	GPIO6
11		34	2	M_DO
12		36	2	GPIO7
13		33	2	M_IO2
14		28	2	FP_UC_14
15		32	2	M_DI
16		29	2	FP_UC_15
17		31	2	M_IO3
18		30	2	FP_UC_16
<b>b. J03 Header JTAG Line Details</b>				
Header Pin No.	Pin Type	FPGA Pin No.		
1	TMS	1		
2	TDI	2		
3	TDO	75		
4	TCK	76		
5	3.3V Supply	-		
6	Ground	-		

c. Peripherals Connection Details				
Peripheral Name	Peripheral Pin	FPGA Pin No.	FPGA Bank	UCP Pin Name
HDMI	CLK+	83	0	CLK_P
	CLK-	84	0	CLK_N
	D0+	85	0	D0_P
	D0-	86	0	D0_N
	D1+	88	0	D1_P
	D1-	89	0	D1_N
	D2+	93	0	D2_P
	D2-	94	0	D2_N
	CEC	98	0	FPGA_CEC
	SCL	78	0	FPGA_SCL
	SDA	77	0	FPGA_SDA
	HPD	73	1	FPGA_HPDP
W25Q32JVSS (Flash Memory)	CS	46	2	M_CS
	Clock	35	2	M_CLK
	Data In/IO0	32	2	M_DI
	Data Out/IO1	34	2	M_DO
	IO2	33	2	M_IO2
	IO3	31	2	M_IO3
ADC101S051CIMF (SPI ADC)	CS	59	1	A_CS
	Data	60	1	A_DT
	Clock	61	1	A_CLK
MCP4725 (I2C DAC)	Data	64	1	D_DT
	Clock	62	1	D_CLK
CH340	TX_Data	49	2	UART_TX
	RX_Data	50	2	UART_RX
Tact Switch	SW1	68	1	TACT_SW1
	SW2	82	0	TACT_SW2
Slide Switch	S1	39	2	SLDE_SW1
	S2	57	1	SLDE_SW2
	S3	56	1	SLDE_SW3
	S4	52	2	SLDE_SW4
LEDs	L1	72	1	LED1
	L2	71	1	LED2
	L3	70	1	LED3
	L4	65	1	LED4



## Power Supply

### ➤ Supply Options:

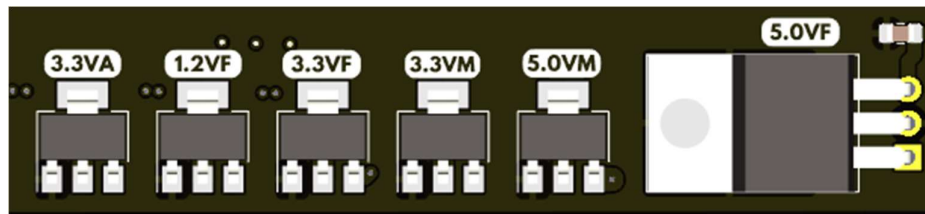
The microcontroller's USB input can deliver a 5V input voltage with a maximum current of 500mA. This power supply is typically sufficient for operating the module, including HDMI streaming. However, if an additional shield board is connected that requires more power, a dedicated power supply line is available.

To power the board, a 9V-12V DC input can be used through the barrel jack. This input can provide a maximum current of 1A. It is important to strictly adhere to the 9V-12V DC input range. When an input within this specified range is applied, a selection circuit automatically disables the USB power input while keeping the serial communication feature unaffected.

The current limits are enforced by using Resettable Fuses. Attempting to bypass these fuses will compromise the safety of the board and may lead to overcurrent damage.

### Power Distribution Scheme:

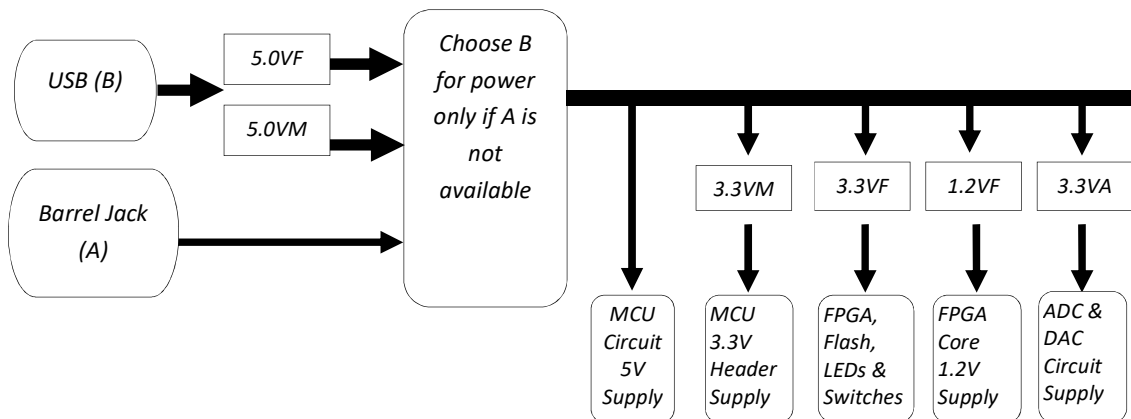
**Figure 2-4. Board segment showing all linear voltage regulators.**



The board is equipped with six regulators that cater to the specific power requirements of different modules. Each regulator can handle a maximum current of 1Amp individually. Since the board has an overall current limit of 1Amp, it ensures that none of the regulators will encounter overcurrent conditions.

To provide clarity, the output of each regulator is labelled with a unique tag name. The following block diagram illustrates the comprehensive power management of the board, using the respective tag names for the regulators.

**Figure 2-5. Comprehensive power management of the board**



### ➤ Clock Input

The board employs fixed clock sources for its processors. The **FPGA** is driven by a **50MHz oscillator**, which generates a **50MHz clock signal** independently. It is important to note that the oscillator used is not a crystal, but an oscillator itself. The generated frequency may exhibit a deviation of  $\pm 50\text{ppm}$  (parts per million).

In contrast, the **MCU** (Microcontroller Unit) on the board operates using a **16MHz crystal**. This crystal provides a stable **16MHz clock signal** for the MCU's operations.

### ➤ Peripherals

The list of all peripherals already has been given in Hardware Specification section of this document. Here we shall briefly discuss about them.

- **FPGA Side ADC**

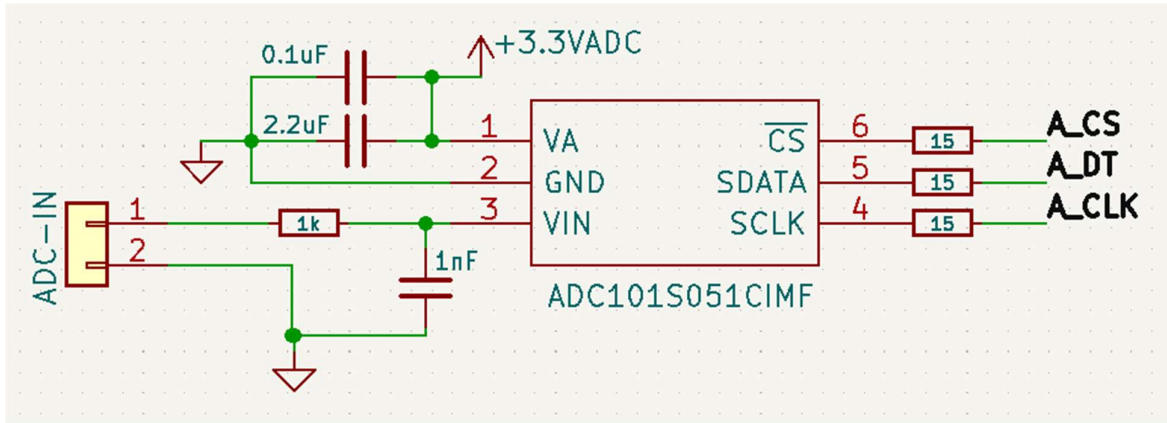
On the board, a dedicated 8-bit/10-bit ADC module (Datasheet) is present, which is connected to the FPGA. This ADC module allows for communication with any processor via the SPI (Serial Peripheral Interface) protocol, supporting a sampling rate of up to 200ksps (kilo-samples per second).

To ensure accurate and reliable conversions, the ADC is supplied with a dedicated 3.3V power supply and single point grounding scheme. This separate power supply helps minimize noise during the conversion process, thereby improving the overall quality of the acquired data.

Additionally, a minor RC (resistor-capacitor) low-pass filter is implemented to filter out high-frequency spikes and further enhance the accuracy of the ADC measurements. This

filter consists of a 1k $\Omega$  resistor and a 1nF capacitor, working together to attenuate high-frequency noise and maintain a clean analog signal input to the ADC.

**Figure 2-6. ADC low pass filter line diagram**



#### • MCU ADCs

There are 8 10bit ADC channels in ATMEGA328PB. They can operate at 10ksps. In the Fusion V1.0 6 ADC channels are shared for I/O interconnections discussed in "FPGA and Microcontroller Interconnection" section. The circuitry causes different termination resistance at different ADC channels. ADC AD0 & AD1 are kept dedicated with no termination (High Input Impedance). AD5 & AD6 have 750 $\Omega$  and AD4, AD3, AD2 & AD7 have 4k $\Omega$  termination resistance respectively.

#### • DAC

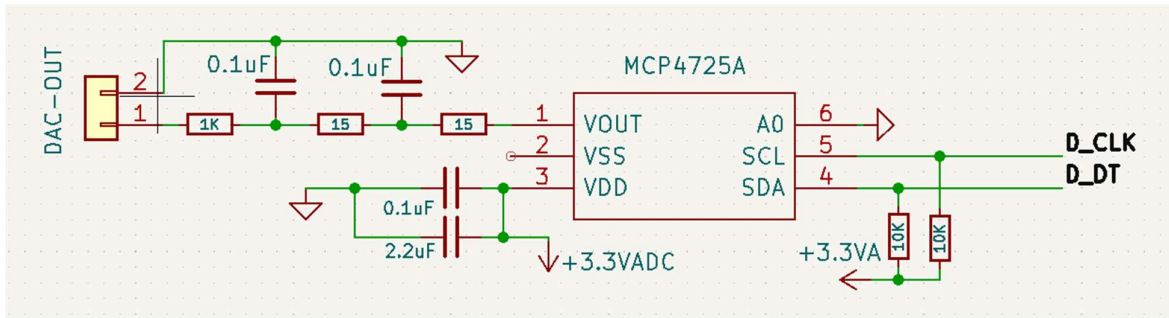
The board includes a 12-bit DAC (Datasheet) that is connected to the FPGA through a dedicated I2C line. The communication between the FPGA and the DAC is facilitated using the I2C protocol.

The current pull-up configuration on the I2C line allows the DAC to be operated at a maximum sampling rate of 400ksps (kilo-samples per second). This configuration ensures reliable and efficient communication between the FPGA and the DAC.

To ensure smooth transitions between different DAC output levels, multiple RC (resistor-capacitor) low-pass stages are incorporated in the output line. These stages act as filters, reducing high-frequency noise and providing a smoother analog output signal.

It is important to note that the output line is loaded with a 1k $\Omega$  impedance because the DAC does not have high current drive capability. Therefore, to interface the DAC output with a circuit that requires higher current, a unity gain buffer must be used.

**Figure 2-7. DAC low pass filter line diagram**

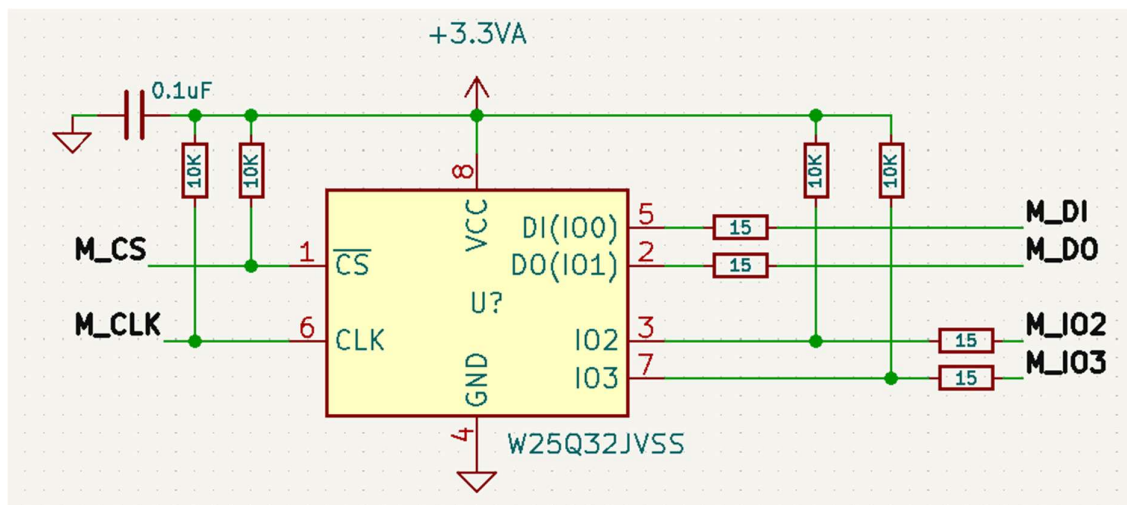


- Flash Memory**

In the system, there are two flash memories serving different purposes. The first one is a **512kB flash memory** that is dedicated and hardcoded to store the main **program files of the FPGA**. This allows the FPGA system to resume operation seamlessly after a shutdown or power cycle.

The second flash memory is a **32Mbit (equivalent to 4MB) flash memory, Winbond W25Q32JV**. This flash memory is controlled by the user and serves as a runtime **data storage space**. The flash can communicate with the FPGA in SPI and QSPI (High-Speed) mode facilitating efficient read and write operations. It enables the user to save and retrieve data required for specific operations during FPGA runtime.

**Figure 2-8. Flash Memory Connection Typical Circuit**



By leveraging the features of the Winbond W25Q32JV flash memory, the system can store runtime data, such as intermediate results, configurations, or other temporary information, providing flexibility and expandability to the FPGA-based application.

- **HDMI**

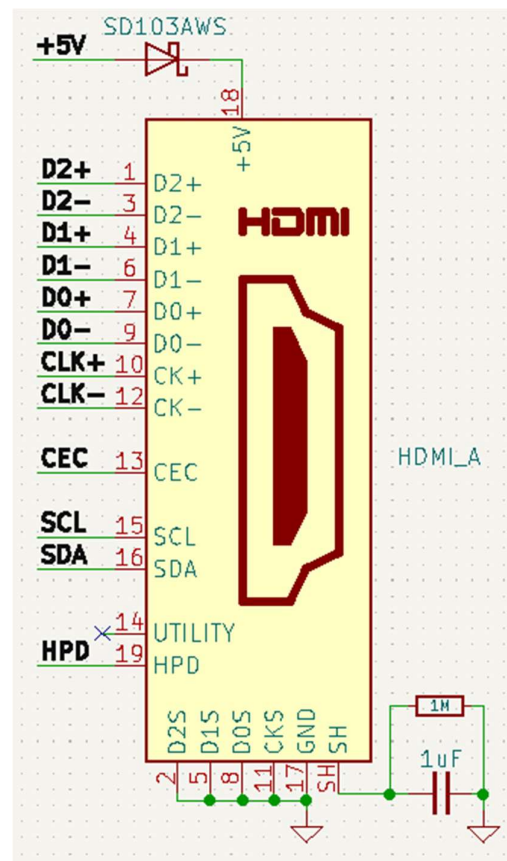
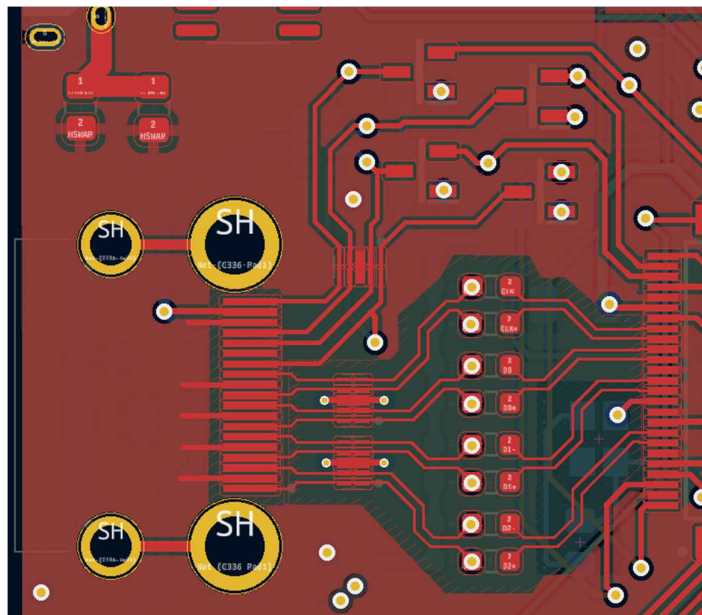
In Fusion V1.0, the FPGA is connected to an HDMI connector to facilitate streaming of audio and video signals.

All the specific communication lines involved in the HDMI connection include D0, D1, D2, CLK, HPD, CEC, SCL, and SDA have been provided. These lines carry various signals such as video data (D0, D1, D2), clock signal (CLK), hot-plug detection (HPD), consumer electronics control (CEC), and I2C bus lines (SCL and SDA) for control and configuration.

To ensure reliable and high-quality signal transmission, the clock signal and the three data lines (D0, D1, D2) are designed as TMDs pairs of 100Ω to match the impedance requirements of the HDMI standard. This helps to minimize signal distortion and maintain signal integrity throughout the transmission.

Furthermore, proper transient protection is implemented in the connections to safeguard against transient voltage spikes or disturbances that could potentially damage the circuitry. These protection measures help to ensure the longevity and reliability of the HDMI communication in the Fusion V1.0 FPGA system.

**Figure 2-9. HDMI peripheral connection**



As we saw all the peripherals available on the board are directly connected to the FPGA. While direct MCU communication with peripherals can be a valuable approach in certain scenarios where simplicity, understanding and rapid prototyping are the primary focus but fails to fully leverage the potential in terms of parallel processing, high-speed data handling and complex logic implementation. The primary purpose of connecting all peripherals directly to the FPGA is to implement complex digital logic circuits and provide hardware acceleration.

But at the same time connecting 16 I/O lines of the MCU with FPGA enables the MCU to directly access all the on-board peripherals if the user wants. By configuring the FPGA as a combinational circuit, the MCU can interact with and utilize the functionalities of the peripherals without the need for complex FPGA programming.

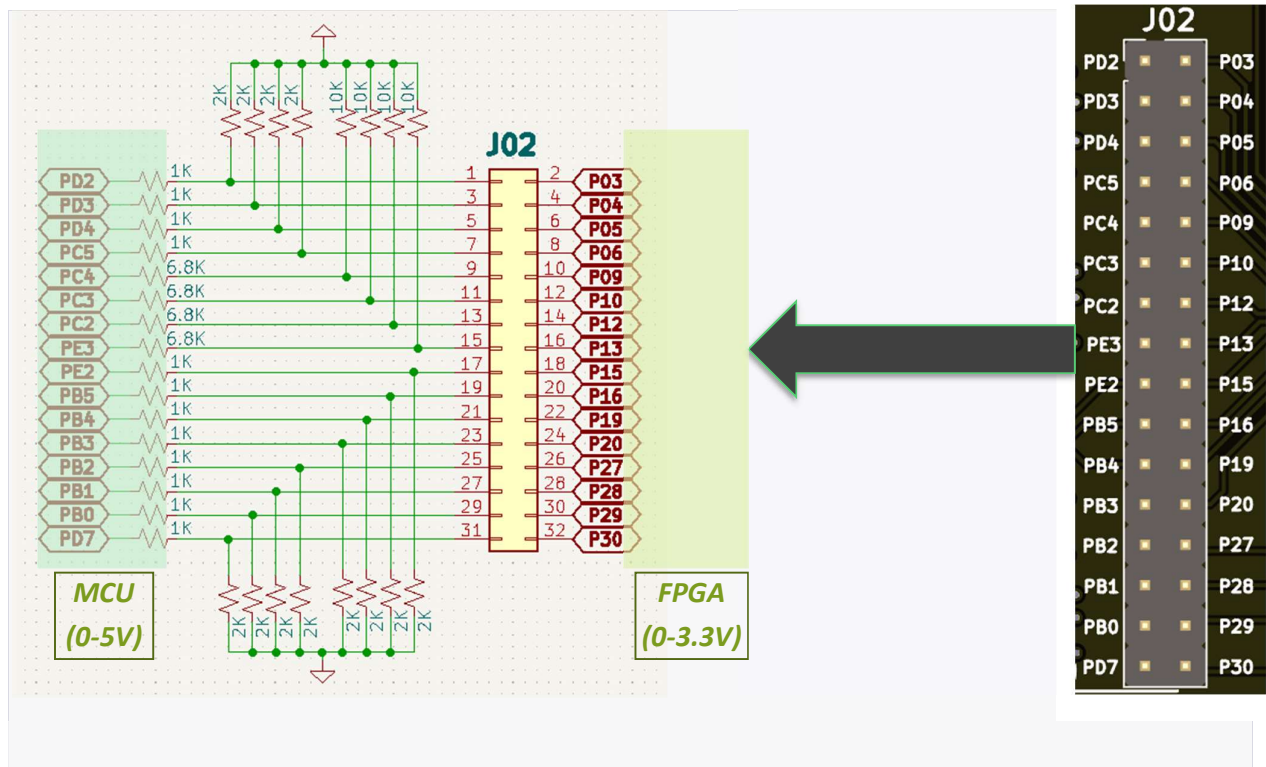
It's important to note that the choice between direct MCU communication and utilizing the full capabilities of the FPGA depends on the specific requirements, complexity of the system, performance needs and development constraints. Both approaches have their advantages and trade-offs, and the decision should be based on the specific project objectives and constraints.



- **FPGA and Microcontroller Interconnection**

The microcontroller I/O voltage level is 5V and the FPGA I/O voltage level is 3.3V. So direct connection between them is not a viable option. Considering the frequency response of the MCU the interconnects are level-shifted using resistor dividers as shown in fig 2.10.

**Figure 2-10. Microcontroller-FPGA Communication Header Typical Circuit**



Out of 16 interconnects there are 6 ADC lines. ADC lines ideally should have infinite termination impedance. But due to the resistor division the impedance seen by any input signal is low. ADC line AD5 & AD6 (Port ID PC5 & PE2) possess 750  $\Omega$  which is low for any ADC channel. PC4, PC3, PC2 & PE3 corresponds to ADC channels AD4, AD3, AD2 & AD7 respectively. These ADC lines are provided with 4k $\Omega$  termination by using 6.8k $\Omega$  & 10k $\Omega$  resistor division. This increase in ADC termination impedance comes at a cost of small reduction in I/O speed of those particular ports.

## Chapter 3 Getting Started with Fusion V1.0

To code and run the Fusion board you need three software. One for FPGA coding and binary (.bin) file generation, one for uploading the binary file to FPGA and one for MCU coding and programming.

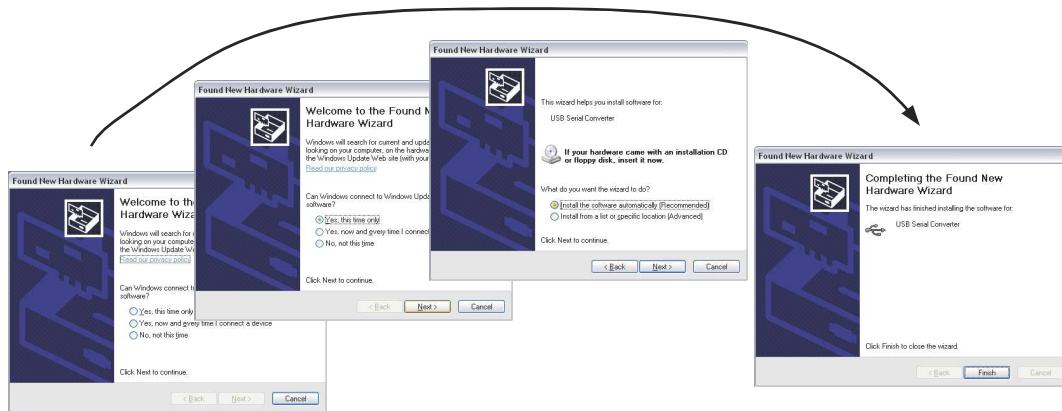
- The onboard FPGA is **Xilinx Spartan XC3S50A/ XC3S200A** requires **XILINX ISE Design Suite 14.7** for code writing, synthesis and binary file generation. This particular Xilinx Design Suite requires [VirtualBox 5.2.34](#) for Windows 10 and above platform. Go to our website for guidelines to install both the application.
- To upload any existing or newly generated “.bin” file to the FPGA, the provided **JTAG header J03** has to be used. But keeping in mind the fact that JTAG is expensive, a robust custom **Fusion Serial Programmer** application software which can seamlessly upload the generated binary file onto Fusion V1.0 board using the USB port.
- The on-board ATMEGA328PB MCU can be programmed by **Arduino IDE**. In this chapter we will elaborate the process to use **Board Manager and Library** required to properly program the MCU

**Fusion V1.0** being a beginner level board we have provided with many ready example codes. Go through the **e-Learning Section** of our website to understand the example codes provided to quickly start using the **EVM Module**.

### Installation and Getting Stated with the Fusion Serial Programmer

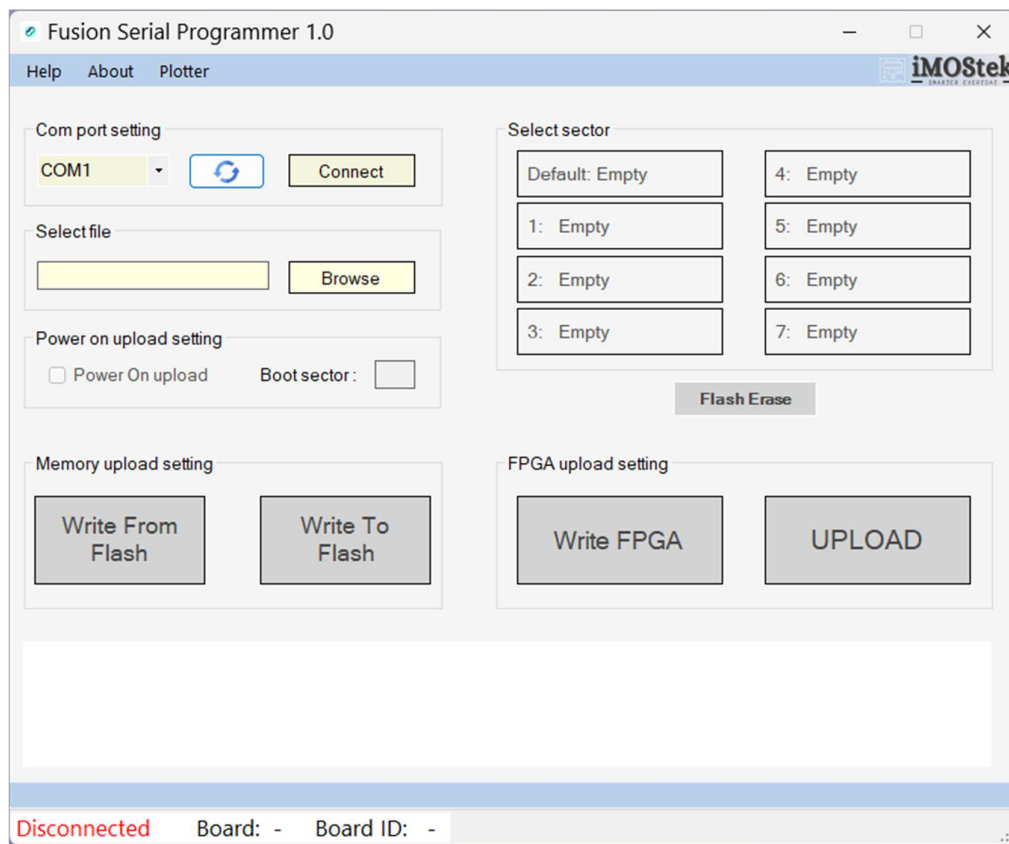
**Fusion Serial Programmer** can be installed in Windows PC by running the **FisionSerialProgrammer.exe** file located on the iMOSitek Web site **Documentation** Section. This file installs the graphical user interface (GUI) along with the USB drivers needed to communicate with the USB controller that resides on the EVM.

**!! Important: Before plugging in the USB cable for the first time, install the Serial Programmer onto your system.**




**Figure3-1. Found New Hardware**

Once installed and opened the GUI looks like fig. no 3-2. There are different sections out of which only 2 are enabled before FPGA is connected to the application.



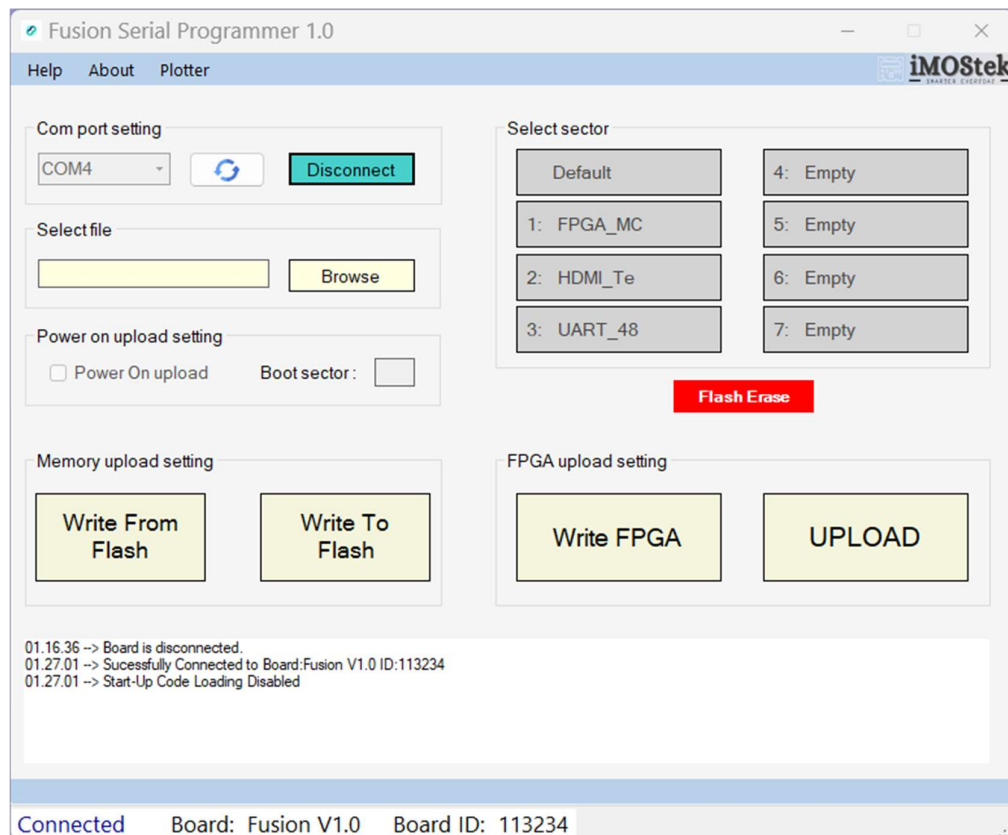
**Figure3-2. Serial Programmer interface before connecting to FPGA Controller**

- **“Com setting”** section is enabled to allow user select and connect to the correct **COM port** where the **FPGA controller** is physically connected. Use the  button in case the expected COM port is not present in the list. **Choose correct COM port** from the dropdown list, **application** communicates to the **FPGA controller** and after few bytes exchange a valid connection is established and updated on display area of the application.

**!! Important: Ensure the UART Slider switch on Fusion Board is in PROG mode otherwise FPGA controller will fail to communicate with the application.**



- The **"Select file"** section is always enabled. User can choose the bin file location using the **"Browse"** button.
- After successful connection to the FPGA Controller **Memory upload setting and FPGA upload setting** sections will be enabled as shown in fig 3-3. The FPGA does not have any non-volatile memory to hold the code, so at every start-up a program must be uploaded to the FPGA. Using the **"Write FPGA"** button user can directly write any code to the FPGA.
- Furthermore, Fusion V1.0 board has a **512KB flash memory and a FPGA program controller**. They allow user to save upto 8 binary files on the board itself. The **"Power on upload setting"** section provides user the control to choose any of the 8 codes to be uploaded onto the FPGA at system power-on.
- At successful connection establishment the application reads and display **codes available at different sectors of the memory** and **Power-on upload setting** and at status bar **Connection Status, Board Type and Board ID**.

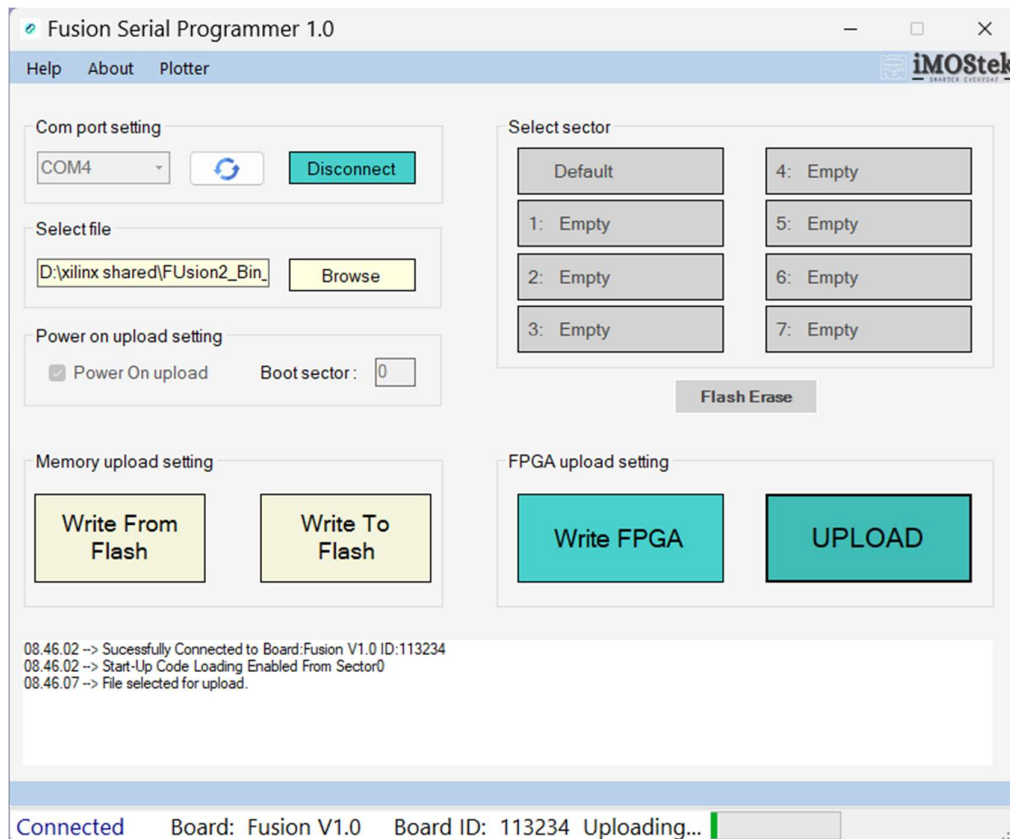


**Figure3-3. Programmer after connection establishment with FPGA Controller**

Now the User can upload bin file to FPGA/Memory using “**Memory upload setting**” and “**FPGA upload setting**” at different modes as discussed below:

➤ **Upload a program directly to the FPGA:** During the early stage of code development process multiple iterations are required. So it is better to just upload the code to FPGA and check the functionality. The binary file is written directly to the FPGA by the controller. Remember, this uploaded code works until the board is powered-down. At next power-up user has to write to FPGA again.

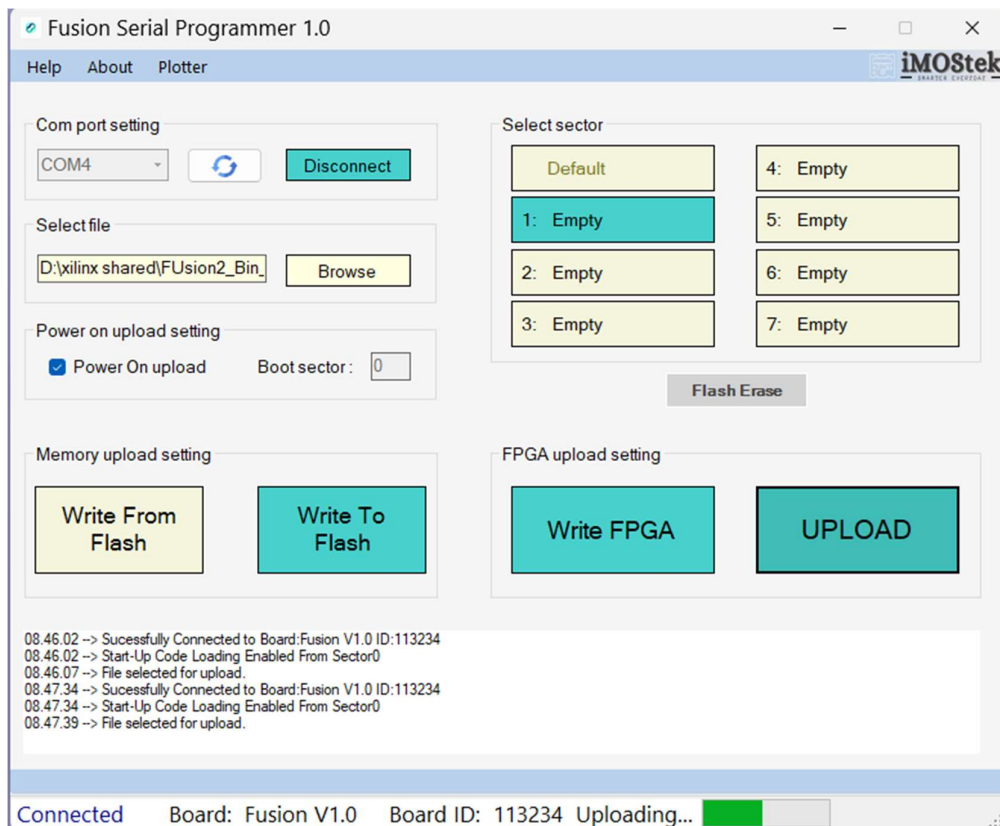
**Steps:** Select “**Write FPGA**” button > press “**UPLOAD**” button.



**Figure3-4. Code upload to on-board FPGA**

➤ **Upload to FPGA and Flash:** A program is perfected the User may keep the program in the FPGA memory. There are 8 sectors in the memory out of which 7 are rewritable. The “**Select sector**” section takes care of the location mapping. When the User chooses an option from “**Memory upload setting**” section, the “**Select sector**” section gets enabled. Now the User can write to or write from any of the sector depending on choice of the user. The “**Default**” sector contains a generic code to check functionality of all the modules associated with the FPGA, hence not provided with any provision to re-write.

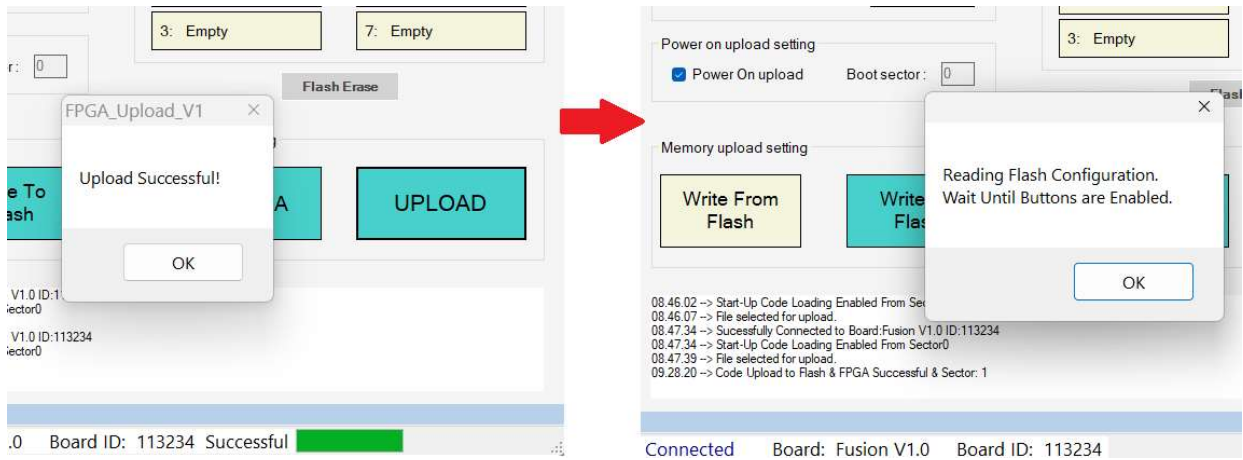
**Steps:** Select “**Write to Flash**” button > Select **Sector 1-7** (one sector) > Select “**Write FPGA**” button (If like to write to FPGA at the same time) > press “**UPLOAD**” button.



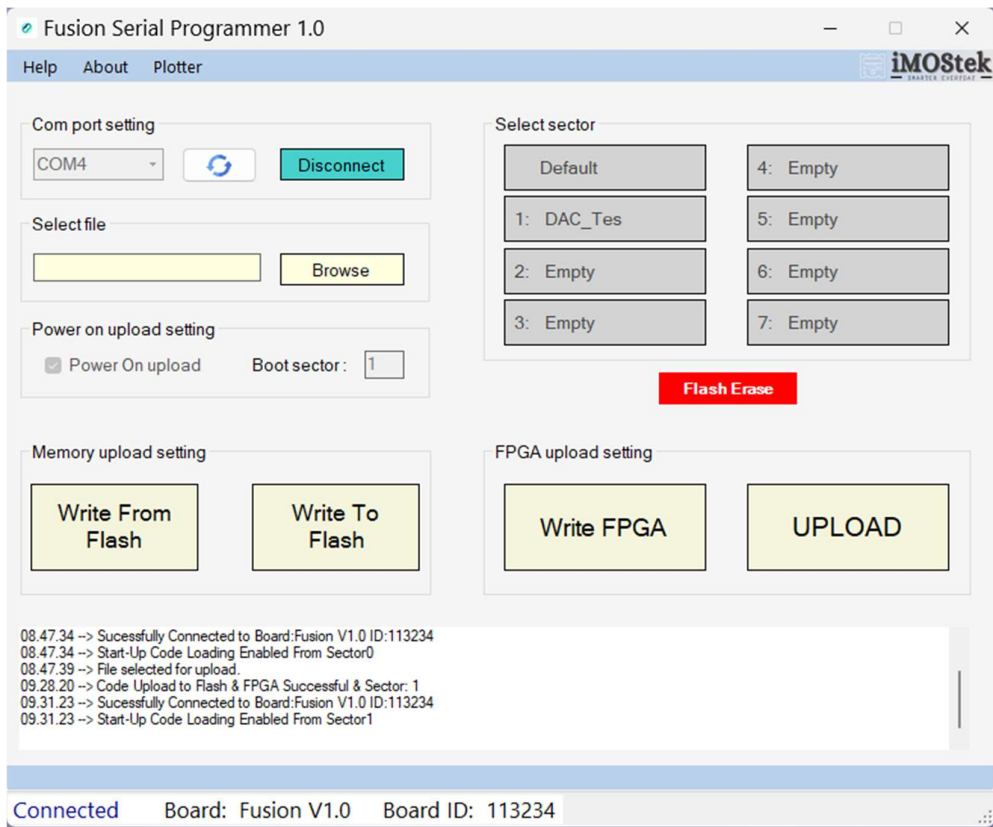
**Figure3-5. Code upload to on-board FPGA and Code Memory**

**Note:** If the upload process is successful a pop-up comes to notify the same user should press Ok to proceed. Then application shows another pop-up to notify user as it reads code memory status. Press Ok to proceed here as well. After successful memory read new **Power-on upload setting** and **Memory Segments** get updated to display uploaded code.





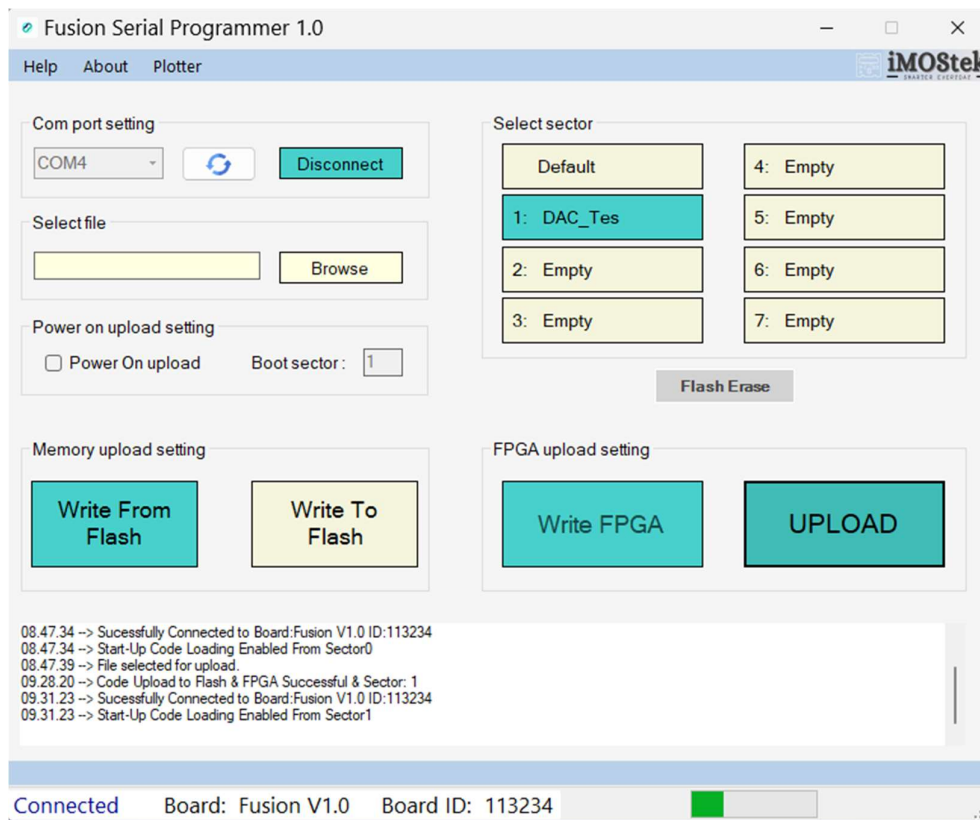
**Figure3-6. Pop-ups after successful code upload**



**Figure3-7. Updated Memory Segment 1 after successful code upload**

➤ **Upload from Flash:** The memory sector locations are mapped onto the buttons inside the **“Select sector”** section. The bin file names are clipped and stored in memory during upload to memory. For mapping the names are used as tags so that user can identify which sector has which code. If there is no identifiable code, the tag says **“Empty”**. Once user has uploaded codes to the memory sectors, he/she does not need to carry the bin files separately. From any computer using our serial programmer users can choose the code to be uploaded into the FPGA.

**Steps:** Select **“Write from Flash”** button > Select **Sector 1-7** (one sector) > press **“UPLOAD”** button. (The **“Write FPGA”** button gets selected by itself in this operation.)

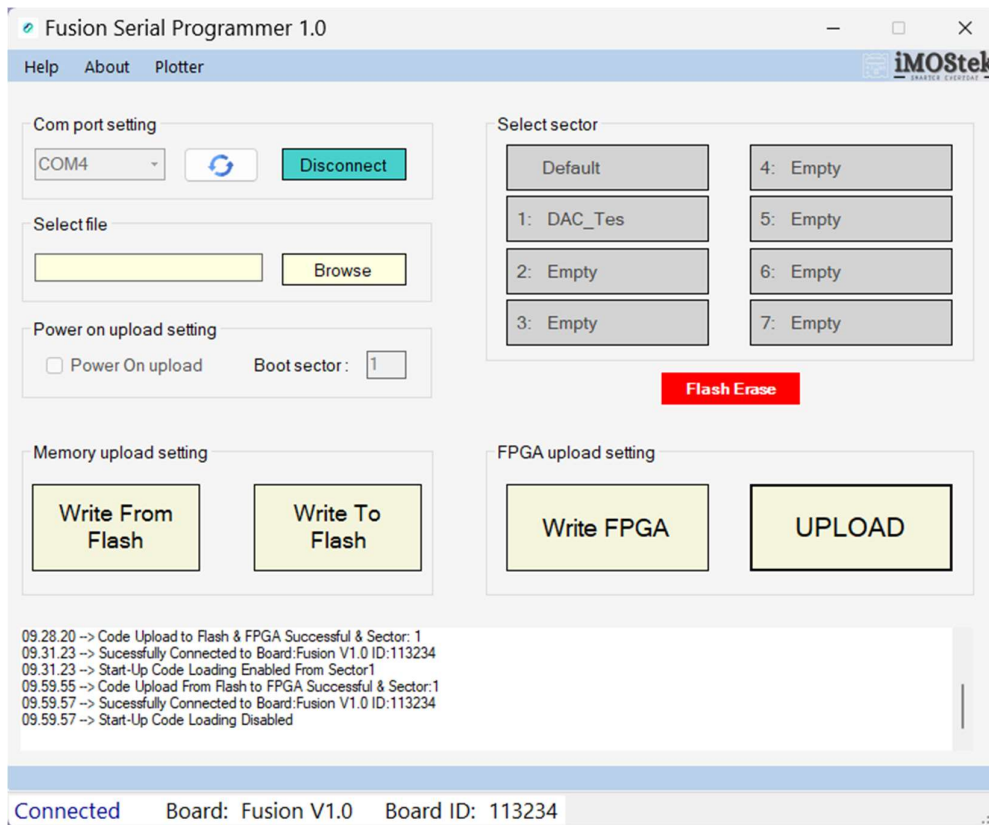


**Figure3-8. Uploading Code to FPGA from Memory Segment 1**

➤ **Power-on upload:** This is needed when User has to deploy the module in field where Serial Programmer access is not possible anymore.

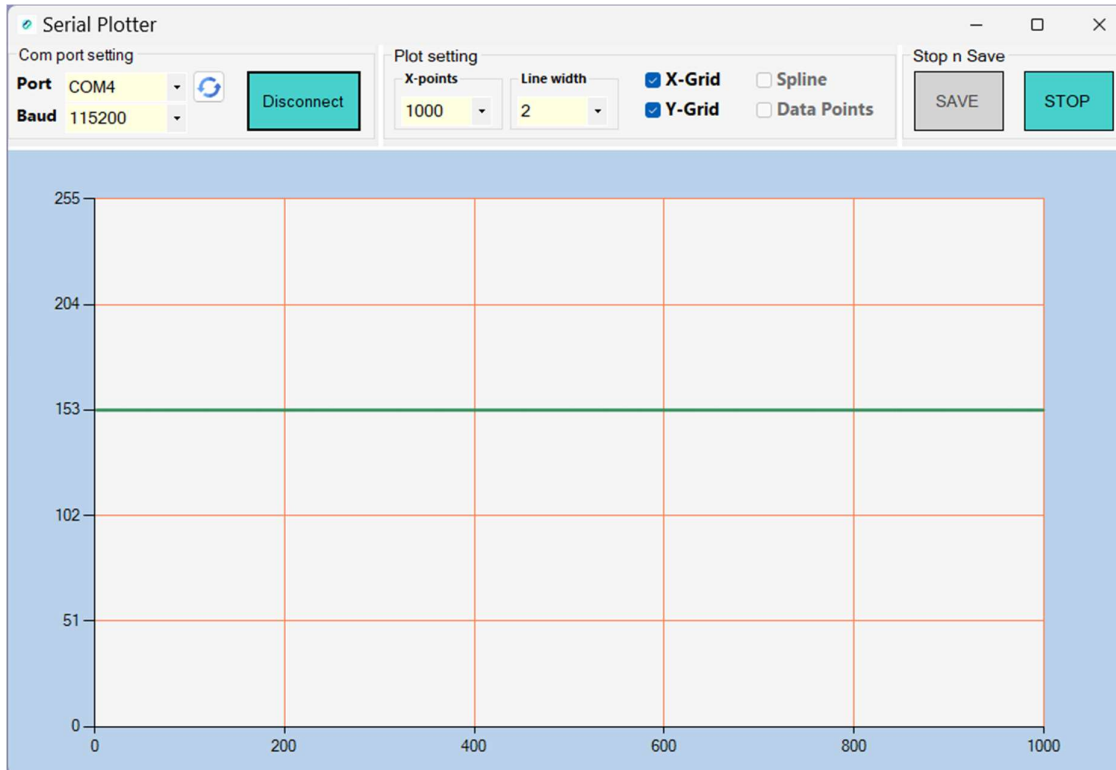
**Steps:** enable “**Power On upload**” > Select “**Write FPGA**” > Select “**Write From Flash**” / Select “**Write To Flash**” > Press “**Upload**”. Once the upload is successful, next time onwards at every power-on FPGA will boot-up the same code.

**Flash Erase:** **Flash Erase** button should be used carefully. It erases all the user stored codes of all the memory sectors. The **Sector-0** is not erased. Also after every Flash Erase operation **Power-on upload** is setting is **set and sector-0** is chosen for **power-on boot-up**.



**Figure3-9. Post Code Upload interface showing updated Power-on upload settings**

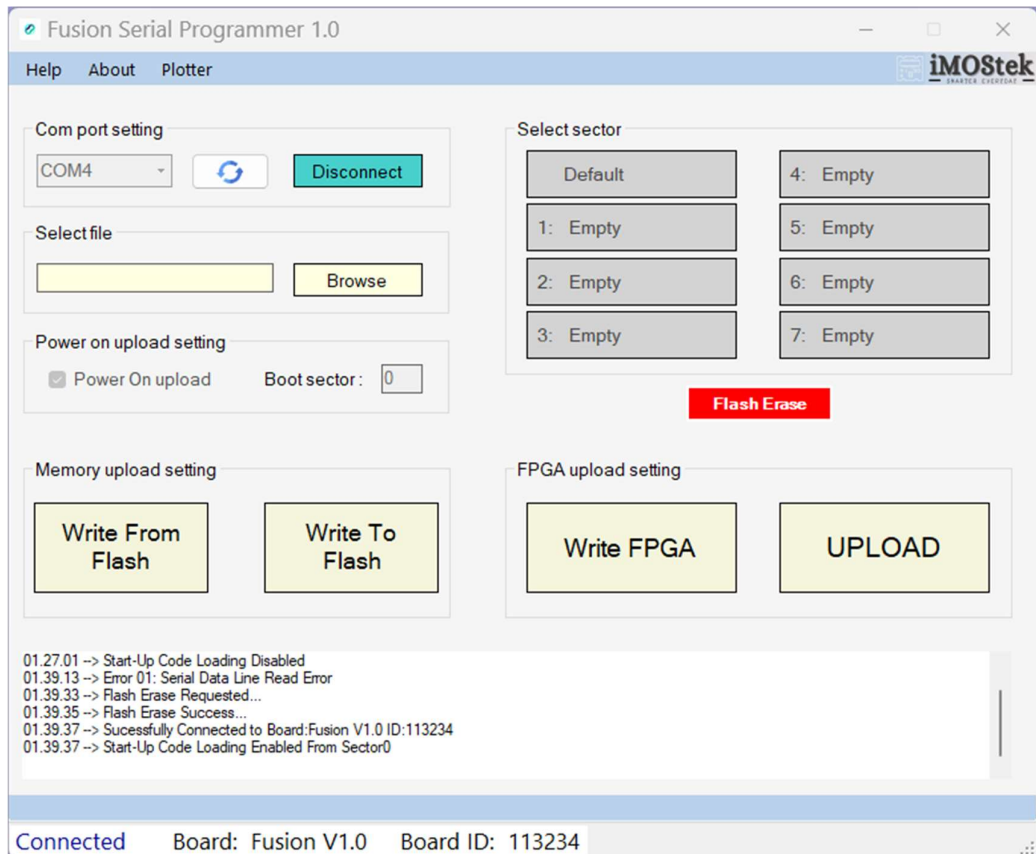
**Serial Plotter:** On menu bar **Plotter** option is there. This is basically a **UART data** plotter when **8-bit** data frame is sent by FPGA. To enable the data receive **SW5** toggle switch must be **set at COMM** mode. When Plotter window opens the **USB port is disconnected** by application. Set the required **BAUD Rate** and connect to the **same USB Port** to start receiving the **FPGA UART Data**.



**Figure3-10. Serial Plotter Interface after Connection establishment**

### Checking Fusion Board FPGA Section using Factory Default FPGA Code

The shipped board is loaded with a code at **Sector-0** segment of the code memory tagged as **"Default"** and the **Power-on upload setting** is set at **Sector-0**. So at first boot-up the **Default** code will automatically be loaded into the FPGA. Make sure the **PROG/COMM** toggle switch **SW5** is at **PROG** mode and connect both the USB cables (MCU USB connection for supply). Open the Serial Programmer, choose the correct port for FPGA USB and press connect. The Serial Programmer shows the overall status (



**Figure3-11. Serial Programmer showing Board status after connection setup**

At this state the Default code is already running in the FPGA. The code is a test code to check and confirm the on-board ADC, DAC and HDMI are working properly. Apart from that the code connects the switches and the LEDs as well as per the below mentioned logic:

- I. SW1 & SW2 tact switch press turns L1 & L2 LEDs on respectively.
- II. S3 slider at ON state turns L3 LED and ADC on.
- III. ADC data is sent to UART line at 115200 Baud at ON State.
- IV. S4 slider at ON state turns L4 LED and DAC on.
- V. DAC value is incremented by 1 digital bit at SW1 tact switch press.
- VI. HDMI streams a Ping-Pong Animation when S1 slider is ON state.

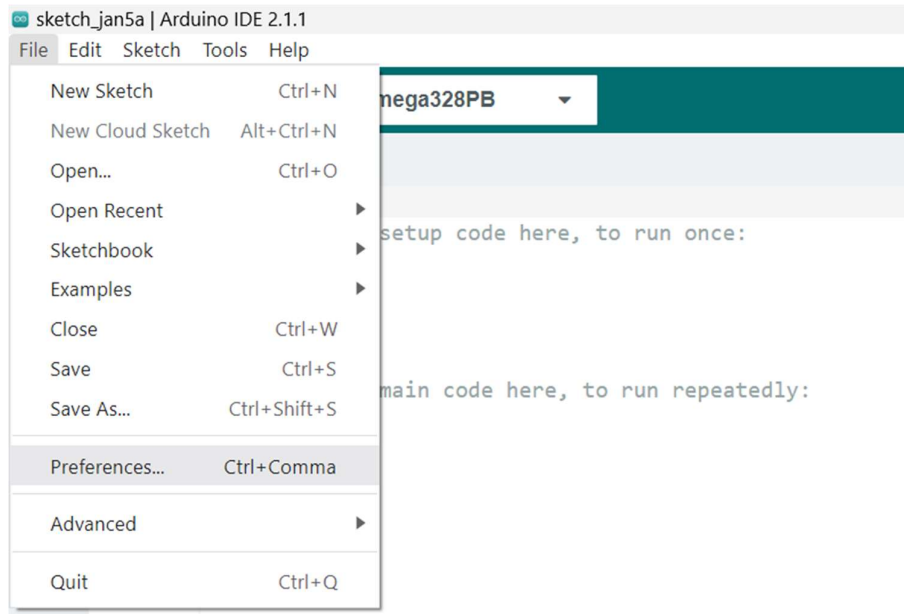
Verify all the events with the mentioned physical actions. To **verify ADC & DAC**, short their headers by jumper wire. Set **SW5** switch to **COMM mode**. Open Plotter window and connect. ADC data will be plotted. Now **press SW1** tact switch to **increment DAC output** which will be reflected in **Plotter**.

Note: To get the UART stream one must toggle the UART slider switch shown in fig 3-5 to **COMM** mode and remember to toggle back to **PROG** mode before uploading another code to the Fusion Board.

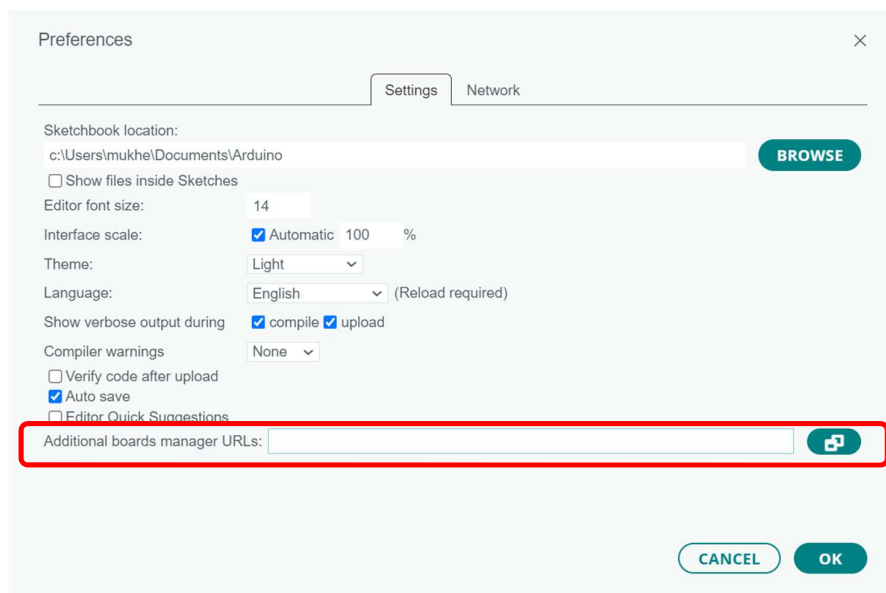
## Set-up Arduino IDE for Fusion Board MCU

This section describes how to quickly turn on and test the board. We are assuming user has already installed the Serial Programmer Software & Arduino IDE.

- First user has to install our Board Library into his Arduino IDE. For that go to **File** -> **Preferences**



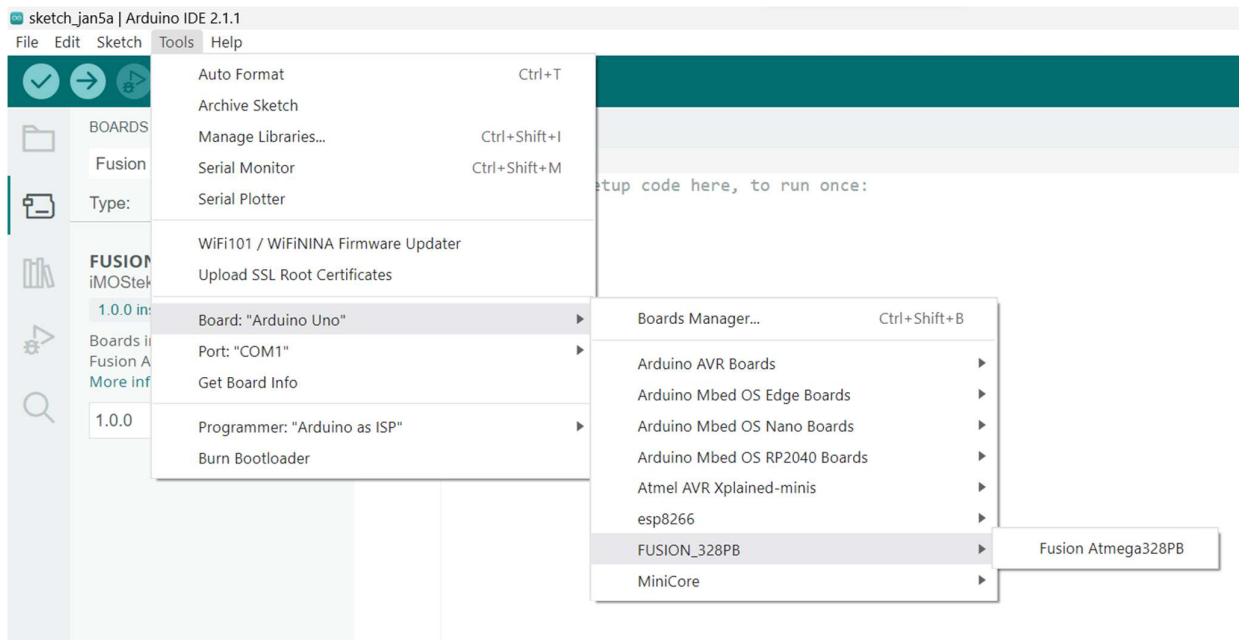
- In **Preferences** there is option to provide a specific JSON file path. Paste [https://github.com/Imostek/Fusion Boards/raw/main/package\\_fusion328pb\\_index.json](https://github.com/Imostek/Fusion_Boards/raw/main/package_fusion328pb_index.json) in the URL box and press **OK**



- User must ensure during this process valid internet connection is available. Now as the JSON file gets downloaded go to **Tools -> Board Manager**. There search for "Fusion\_328PB". If the JSON file is properly downloaded, this search will show Fusion board library. **Install the board library** latest version available.

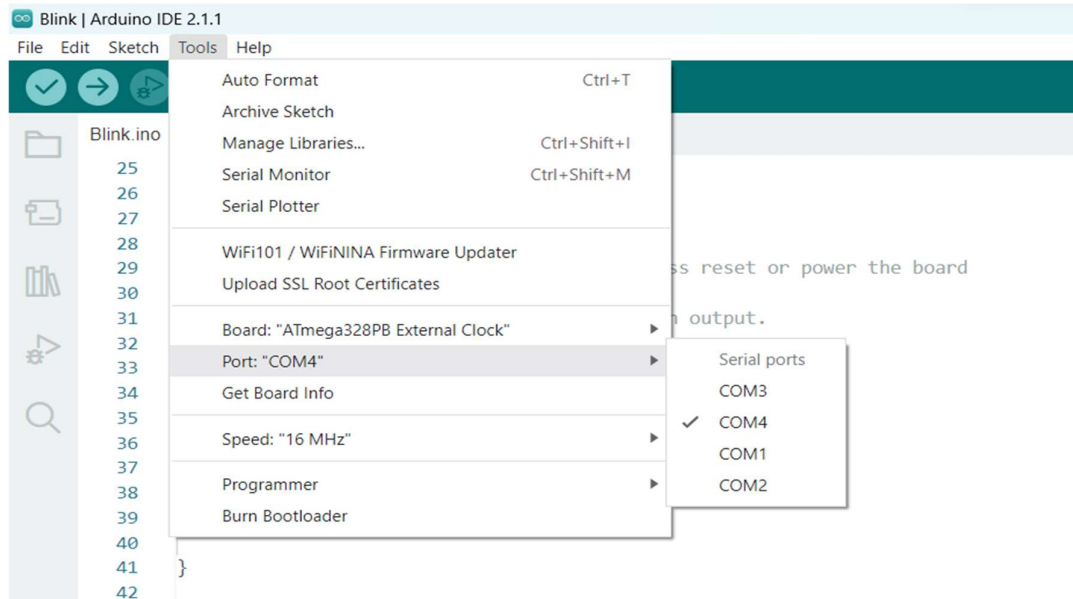


- Once installed go to **Tools -> Boards**. Search for "FUSION\_328PB" and then **Fusion\_Atmega328PB**

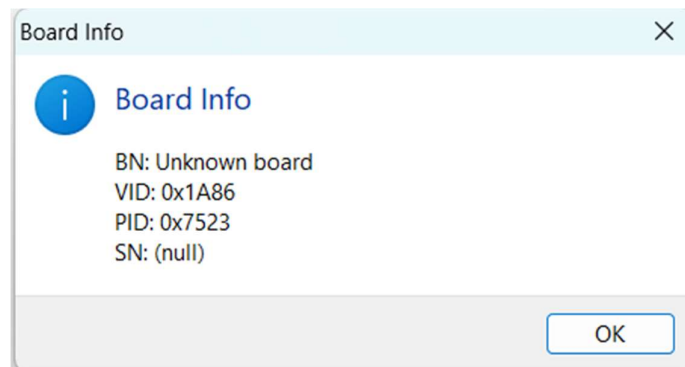




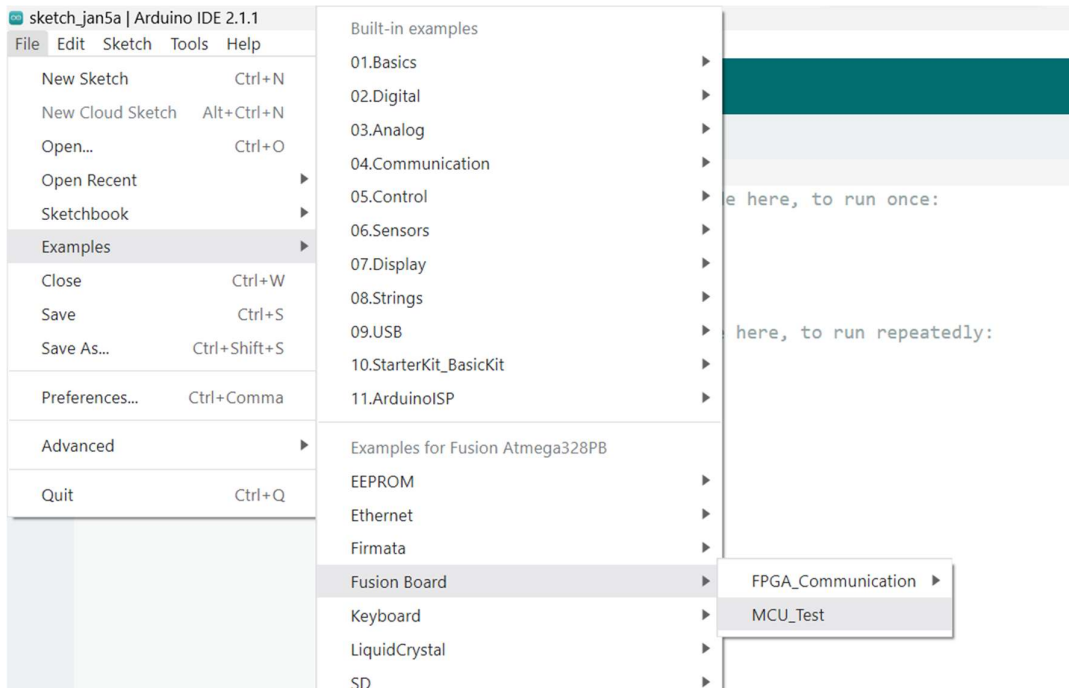
- Now connect the **board MCU USB** to PC using **MicroUSB cable**. The Power LED (RED) should be at **On** state as the board gets power. Now go to **Tools** -> **Port**. Select the correct port where MCU USB is available.



- Then go to **Tools** -> **Get Board Info**. If the board is connected then a valid VID and PID should be visible.



➤ Now go to **File -> Examples -> Fusion Board -> MCU\_Test**



**MCU\_Test** is a generic code does the following operations:

- I. Sets **ADC1** (PC1 Port) and **PC0** Port as **Input**.
- II. Sets **all other ports** as **output**.
- III. Port **PB5** (L6 LED Control line) **reflects PC0** port input status.
- IV. All other ports are set to **toggle at 500 mSec**. So if **PC0 is connected** to any of the output ports then **L6 LED blinks** at **same frequency as the other LEDs**.

Load the code to the MCU and validate all the functionalities mentioned.

## Chapter 4 Additional Details

**Table 4-1: Text form for FPGA “.ucf” file for all ports available on Fusion V1.0**

**//System Connections**

NET "SYS\_CLK" LOC = "P90";  
 NET "RST\_BAR" LOC = "P97";

**//All FPGA and MCU Interconnect//**

NET "FP\_UC\_01" LOC = "P3";  
 NET "FP\_UC\_02" LOC = "P4";  
 NET "FP\_UC\_03" LOC = "P5";  
 NET "FP\_UC\_04" LOC = "P6";  
 NET "FP\_UC\_05" LOC = "P9";  
 NET "FP\_UC\_06" LOC = "P10";  
 NET "FP\_UC\_07" LOC = "P12";  
 NET "FP\_UC\_08" LOC = "P13";  
 NET "FP\_UC\_09" LOC = "P15";  
 NET "FP\_UC\_10" LOC = "P16";  
 NET "FP\_UC\_11" LOC = "P19";  
 NET "FP\_UC\_12" LOC = "P20";  
 NET "FP\_UC\_13" LOC = "P27";  
 NET "FP\_UC\_14" LOC = "P28";  
 NET "FP\_UC\_15" LOC = "P29";  
 NET "FP\_UC\_16" LOC = "P30";

**//TACT Switch Connections//**

NET "TACT\_SW1" LOC = "P68";  
 NET "TACT\_SW2" LOC = "P82";

**//Dip Switch Connections//**

NET "SLIDE\_SW1" LOC = "P39";  
 NET "SLIDE\_SW2" LOC = "P57";  
 NET "SLIDE\_SW3" LOC = "P56";  
 NET "SLIDE\_SW4" LOC = "P52";

**//QSPI Flash Memory Connections//**

NET "M\_CS" LOC = "P46";  
 NET "M\_CLK" LOC = "P35";  
 NET "M\_DI" LOC = "P32";  
 NET "M\_DO" LOC = "P34";  
 NET "M\_IO2" LOC = "P33";  
 NET "M\_IO3" LOC = "P31";

**//GPIO Connections//**

NET "GPI1" LOC = "P7";  
 NET "GPIO2" LOC = "P44";  
 NET "GPIO3" LOC = "P43";  
 NET "GPIO4" LOC = "P41";  
 NET "GPIO5" LOC = "P40";  
 NET "GPIO6" LOC = "P37";  
 NET "GPIO7" LOC = "P36";

**//LED Connections//**

NET "LED1" LOC = "P72";  
 NET "LED2" LOC = "P71";  
 NET "LED3" LOC = "P70";  
 NET "LED4" LOC = "P65";

**//HDMI connection//**

NET "D0\_N" LOC = "P86";  
 NET "D0\_P" LOC = "P85";  
 NET "D1\_N" LOC = "P89";  
 NET "D1\_P" LOC = "P88";  
 NET "D2\_N" LOC = "P94";  
 NET "D2\_P" LOC = "P93";  
 NET "CLK\_N" LOC = "P84";  
 NET "CLK\_P" LOC = "P83";  
 NET "FPGA\_HPD" LOC = "P74";  
 NET "FPGA\_SDA" LOC = "P77";  
 NET "FPGA\_SCL" LOC = "P78";  
 NET "FPGA\_CEC" LOC = "P98";

**//ADC Connections//**

NET "A\_CS" LOC = "P59";  
 NET "A\_DT" LOC = "P60";  
 NET "A\_CLK" LOC = "P61";

**//DAC Connections//**

NET "D\_DT" LOC = "P64";  
 NET "D\_CLK" LOC = "P62";

**//UART Connections//**

NET "UART\_TX" LOC = "P49";  
 NET "UART\_RX" LOC = "P50";

## References

Go to our website [www.imostek.in](http://www.imostek.in) for additional documentation about Fusion V1.0 EVM board and online purchase. In Documentation section we have provided

- iMOStek Serial Programmer 1.0.4 Executable file
- ATMEGA328PB Datasheet
- Document on difference between ATMEGA328P and ATMEGA328PB
- Xilinx Spartan XC3S series FPGA Datasheet

Get VirtualBox 5.2.34 for Windows 10:

<https://download.virtualbox.org/virtualbox/5.2.34/VirtualBox-5.2.34-133893-Win.exe>

Xilinx ISE 14.7:

[https://www.xilinx.com/member/forms/download/xef.html?filename=Xilinx ISE 14.7 Win10 14.7 VM 0213 1.zip](https://www.xilinx.com/member/forms/download/xef.html?filename=Xilinx%20ISE%2014.7%20Win10%2014.7%20VM%200213%201.zip)