General Control Board V1.5 Documentation

Mechatronics systems for innovative solutions

Version: 1.5

Date: JULY 7, 2026

Table of Contents

Product overview 1

1.1 Introduction

This document describes the General Control board V1.5, a versatile dual-role Modbus communication platform based on the ESP32 microcontroller.

1.2 Introducing the General Control Board V1.5

The General Control board V1.5 is a versatile industrial-grade device designed for dual-role Modbus communication. It operates as a **Modbus Master** and **Modbus Slave** using two MAX485 modules. As a master, it reads sensor data from Modbus slave devices, timestamps it using an RTC module, and logs it to an SD card. As a slave, it can respond to incoming Modbus queries. It also has a DHT11 sensor for device temperature monitoring, and it is expandable for cloud or local network connectivity.

Component	Description
1- WT32 ETH01 ESP32	Main MCU
2- SD Card Module	For data storage
3AB- MAX485 x2	RS485 transceivers (1 for master, 1 slave)
4- DHT11 Sensor	For temperature and humidity readings
5 RTC DS1307	For real-time timestamping
6-Power Supply	Stable 12V with sufficient current

Key Features

- Dual-role: Simultaneous **Modbus Master** and **Slave** operations.
- SD card logging of timestamped sensor data.
- Real-Time Clock (RTC DS1307) for time-accurate logging.
- DHT11 for device temperature and humidity monitoring.
- SDI-12 interface.
- Watchdog Timer for system reliability.
- Multitasking with **FreeRTOS** (ESP32) & Modular code structure for scalability.

1.3 Connectivity & Networking Capabilities

The General Control Board V1.5, built on the ESP32S platform, features advanced wireless communication options for seamless integration into modern industrial and IoT systems. These connectivity features support local and remote data access, device configuration, and cloud integration.

Wi-Fi Networking

The ESP32's built-in Wi-Fi allows the board to operate as either a Station (STA), Access Point (AP), or both (AP+STA mode). This capability enables:

- **Remote monitoring and control** via web servers or mobile apps (e.g., Blynk,Azure, MQTT dashboards)
- **OTA (Over-the-Air) firmware updates** to ensure up-to-date functionality

Ethernet connectivity

Using dual-core Xtensa 832-bit LX6 microcontroller. Integrated SPI flash memory 32 Mbit\SRAM 520 KB, Supports MQTT, TCP server, TCP client, UDP server, UDP client working mode

Bluetooth Low Energy (BLE)

The board supports BLE , allowing:

- **Low-power communication** with smartphones or BLE-enabled gateways
- **Initial configuration and diagnostics** through mobile applications
- **Short-range peer-to-peer communication** where Wi-Fi is unavailable

ESP-NOW Protocol

ESP-NOW is an efficient, connectionless protocol ideal for short bursts of data between ESP32 devices. It allows:

- **Low-latency communication** between multiple boards
- **Reduced power consumption** due to minimal overhead
- **Secure communication** with peer-to-peer encryption support

12.4 Application Scenarios

Feature	Use Case Example
Wi-Fi	Uploading logs to cloud dashboards
BLE	Local configuration via smartphone

Feature	Use Case Example
ESP-NOW	Node-to-node communication in mesh setups
Ethernet	Connect to MQTT broker

Software 2

2.1 Software Requirements

The software stack for the General Control Board V1.5 is developed using the **ESP-IDF** or **Arduino framework for ESP32**, with additional libraries to support Modbus communication, SD card logging, timekeeping, and multitasking.

Development Environment

Tool/Platform	Description
Arduino IDE	Preferred for rapid prototyping and debugging
ESP32 Board Package	Latest ESP32 core (≥ v2.0.5 recommended)
PlatformIO (optional)	Alternative build system and project manager

Required Libraries (Arduino Framework)

cpp Copy Edit

```cpp
#include <ModbusRTUSlave.h>
#include <ModbusMaster.h>
#include <esp_task_wdt.h>
#include <Preferences.h>
#include <FS.h>
#include <SD.h>
#include <SPI.h>
#include <DHT.h>
#include <RTClib.h>
#include <Simple_Timer.h>
```

2.2 System Architecture and Software Design

Modbus Setup:

- **ModbusRTUSlave** used on UART2 (26,27) for slave operations.
- **ModbusMaster** array used on UART1(16,17) for querying up to 32 slave devices.
- Control flow for RS485 direction via **preTransmission()** and **postTransmission()** functions.

Task Scheduling:

- Uses **Simple_Timer** to schedule recurring tasks:
- **masterTask** every 8 seconds
- **RTC** runs normally on the loop
- **DHTtask** every 5 seconds
- **slaveTask** every 1 second (in FreeRTOS task)

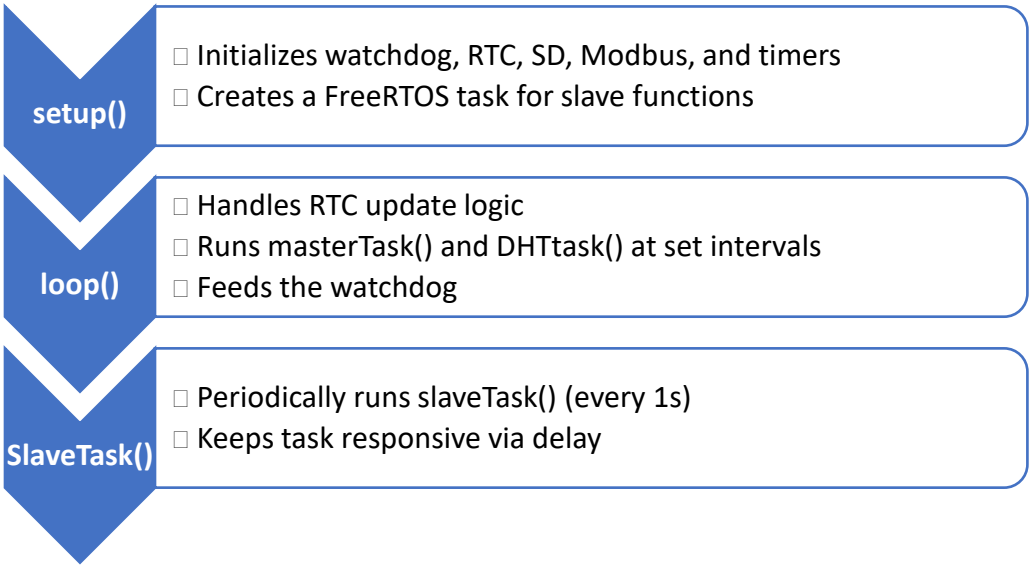
FreeRTOS Multitasking:

- SlaveTask() runs on **Core 0**, handling slave responses and Blynk connectivity.
- loop() runs on **Core 1**, handles timers and watchdog reset.

Sensor & RTC Integration:

- DHT11 sensor for periodic temperature/humidity readings.
- RTC_DS1307 used for timestamping SD card logs.

Execution Flow:



Data Logging Format

Stored in CSV format on SD card (example):

Timestamp, Temperature (°C), Humidity (%) , first reading , sec reading, ……etc 2025-04-30 10:00:08 , 24.3 , 58, X1 , X2 , ….. etc

2.3 Reliability Features

- **Watchdog Timer**: Prevents system lock-ups
- **Task pinning**: Keeps Modbus Slave isolated from main loop
- **Interval-based tasks**: Reduces CPU overuse

Installation 3

This section details the mechanical and electrical installation procedure for the General Control Board V1.5, ensuring proper integration into industrial panels and enclosures.

3.1 Enclosure and Mounting

The General Board comes pre-installed in a DIN rail-compatible plastic enclosure for secure mounting inside industrial control cabinets.

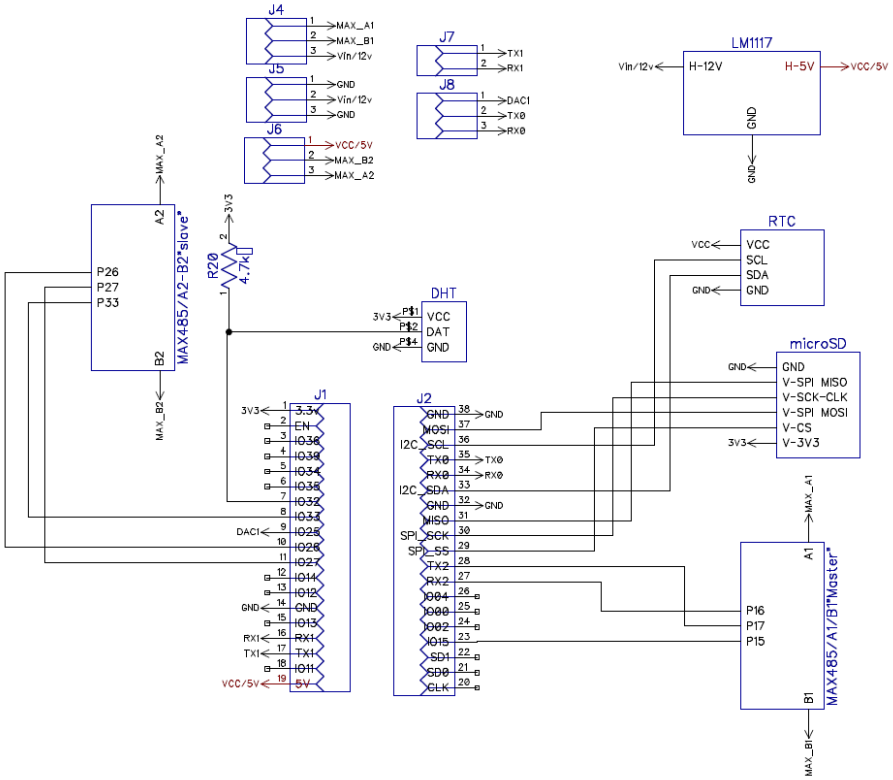
- **Mounting Method:**
  The enclosure features a **red DIN rail clip** mechanism (as shown in the photo). Simply hook the bottom edge onto the DIN rail and push until it snaps into place.
- **Mounting Holes:**
  Four Ø3 mm screw holes are provided at the corners for optional **panel mounting** or securing the enclosure if DIN rail installation is not desired.

3.3 Screw Terminal Layout and Functionality

Terminal Label	Function
VIN1 12V	12V input
GND	Ground reference (shared)

Terminal Label	Function
VIN2 12V	12V output to power any external 12V device or can be used as input
GND	Additional ground
VCC 5V	5V output
MAX_A1	RS485 MAX/A1 line (Modbus) (master)
MAX_B1	RS485 MAX/B1 line (Modbus) (master)
MAX_A2	RS485 MAX/A2 line (Modbus) (slave)
MAX_B2	RS485 MAX/B2 line (Modbus) (slave)
RX1/TX1	UART1 (Modbus Slave) – RS485 level
RX0/TX0	UART2 (Modbus Master) – RS485 level
DAC1	Reserved for analog or future use

Schematic



GPIO Mapping

- **DHT11 Data**: GPIO 32

- **MAX485_DE/RE**: Connect to GPIOs (define in variables.h)
- **UART1 (Master)**: GPIO (e.g., RX=16, TX=17)
- **GPIO** (can be used as SDI-12 interface)
- **SD Card**: SPI interface
- **RTC**: I2C (usually GPIO 21 and 22)