# 数据采集模块 - 代码修改与参数配置指南

本文档旨在为开发人员提供一份详细的技术指南，说明如何修改本数据采集模块源代码中的关键参数，以使其适应不同的硬件环境和应用需求。

## 1. 核心参数修改总览

下表汇总了用户最常需要修改的参数、它们的作用以及所在的文件位置。

|  |  |  |  |
| --- | --- | --- | --- |
| 参数项目 | 描述 | 所在文件 | 如何修改 |
| 数据上传周期 | Modbus 模式下，每轮数据采集和上报之间的间隔时间。 | main.c | 修改 Delay\_s() 函数的参数值。 |
| Modbus 读取地址 | 需要从 Modbus 从站设备读取的寄存器/线圈地址。 | Plc\_config.c | 修改配置数组中的地址字段。 |
| Modbus 从站地址 | 目标 Modbus 从站设备的 ID。 | Plc\_config.c | 修改配置数组中的从站地址字段。 |
| JSON 数据键名 | 在最终输出的 JSON 字符串中，每个数据点对应的 id。 | Plc\_config.c | 修改配置数组中的 json\_key 字段。 |
| 通信波特率 | Modbus (RS485) 和数据输出串口的通信速率。 | Modbus.c, Serial.c, RS232.c | 修改 USART\_InitStructure.USART\_BaudRate 的值。 |
| Modbus 超时时间 | 等待 Modbus 从站响应的最长时间。 | Modbus.c | 修改 readModbusData 等函数中的 timeout\_ms 变量。 |
| JSON 分包延时 | 发送每组 JSON 数据包之间的短暂延时。 | Json\_sender.c | 修改 Delay\_ms() 函数的参数值。 |

## 2. 分文件修改详解

### 2.1. 修改数据上传周期 (main.c)

数据上传的频率由主循环 while(1) 末尾的延时函数决定。

1.打开 main.c 文件。

2.定位到 while(1) 循环的末尾，找到以下代码块：

|  |
| --- |
| // 在 main.c 的 while(1) 循环内  if (choose == 1 || choose == 2)  {  // ... (数据采集和发送逻辑) ...  if (choose == 1) LED1\_Turn();  else LED2\_Turn();  // 修改这里的数值来改变上传周期，单位为秒  Delay\_s(5);  } |

3.修改方法: 将 Delay\_s(5); 中的数字 5 修改为您期望的秒数。例如，要实现每 30 秒上传一次，就改为 Delay\_s(30);。

### 2.2. 修改 Modbus 读取地址与从站 ID (Plc\_config.c)

所有需要采集的数据点都在此文件中定义，这是最常需要修改的文件。

1.打开 Plc\_config.c 文件。

2.修改保持寄存器 (FC03) 的地址:

★ 定位到 PlcDataPoint plc\_data\_points[] 数组。

★ 每一行代表一个采集任务。

★ 格式: {从机地址, 寄存器起始地址, 寄存器数量, JSON键名, ...}

|  |
| --- |
| // Plc\_config.c  PlcDataPoint plc\_data\_points[] = {  // {0x01, 0x138A, 1, "LW5002", ...},  // ^--从站ID ^--寄存器地址  {0x02, 0x2000, 2, "MotorSpeed", ...}, // 示例: 读取从站2的 0x2000 和 0x2001 寄存器  }; |

★ 修改方法: 直接修改第一个参数（从站地址）和第二个参数（寄存器地址）的十六进制数值。

3.修改线圈 (FC01) 或离散输入 (FC02) 的地址:

★ 定位到 BitDataPoint fc01\_coils[] 或 fc02\_discrete\_inputs[] 数组。

★ 格式: {从机地址, 线圈/离散输入地址, JSON键名, ...}

|  |
| --- |
| // Plc\_config.c  BitDataPoint fc01\_coils[] = {  // {0x01, 0x05, "LW105", ...},  // ^-- 从站ID ^-- 线圈地址  {0x03, 0x0010, "ValveStatus", ...}, // 示例: 读取从站3的地址为 0x0010 的线圈  }; |

★ 修改方法: 同上，直接修改对应的十六进制数值。

### 2.3. 修改通信波特率

默认波特率均为 115200。如果您的设备使用其他波特率，需要修改所有相关的串口初始化函数。

1.修改 Modbus (RS485) 波特率:

★ 打开 Modbus.c 文件。

★ 找到 USART3\_Init 函数。

★ 修改 USART\_InitStructure.USART\_BaudRate 的值。

|  |
| --- |
| // Modbus.c -> USART3\_Init()  USART\_InitStructure.USART\_BaudRate = 9600; // 示例: 修改为 9600 |

2.修改串口1 (TTL) 输出波特率:

★ 打开 Serial.c 文件。

★ 找到 Serial\_Init 函数。

★ 修改 USART\_InitStructure.USART\_BaudRate 的值。

|  |
| --- |
| // Serial.c -> Serial\_Init()  USART\_InitStructure.USART\_BaudRate = 9600; // 示例: 修改为 9600 |

3.修改串口2 (RS232) 输出波特率:

★ 打开 RS232.c 文件。

★ 找到 RS232\_Init 函数。

★ 修改 USART\_InitStructure.USART\_BaudRate 的值。

|  |
| --- |
| // RS232.c -> RS232\_Init()  USART\_InitStructure.USART\_BaudRate = 9600; // 示例: 修改为 9600 |

重要提示: 所有参与通信的设备（采集模块、PLC、接收上位机）的波特率必须完全一致，否则将导致通信失败或乱码。

### 2.4. 修改 Modbus 响应超时时间 (Modbus.c)

如果您的 Modbus 从站设备响应速度较慢，可能会导致采集超时失败。您可以适当延长超时时间。

1.打开 Modbus.c 文件。

2.找到 readModbusData 和 readCoilData 这两个函数。

3.在函数内部找到 timeout\_ms 变量的定义。

|  |
| --- |
| // Modbus.c -> readModbusData()  void readModbusData(...)  {  // ...  // 等待响应，带有200ms超时  uint32\_t timeout\_ms = 200; // 单位：毫秒  while (!modbus\_frame\_received && timeout\_ms > 0)  {  Delay\_ms(1);  timeout\_ms--;  }  // ...  } |

4.修改方法: 增加 timeout\_ms 的初始值。例如，修改为 uint32\_t timeout\_ms = 500;，将超时时间延长到 500 毫秒。

### 2.5. 修改 JSON 分包发送延时 (Json\_sender.c)

为了防止数据发送过快导致上位机处理不过来，程序在发送每组 JSON 数据后会有一个短暂延时。

1.打开 Json\_sender.c 文件。

2.在 SendAllDataAsJsonArray 和 SendBitDataAsJsonArray 函数的末尾，找到 Delay\_ms() 调用。

|  |
| --- |
| // Json\_sender.c -> SendAllDataAsJsonArray()  void SendAllDataAsJsonArray(...)  {  for (int i = 0; i < count; i += 2)  {  // ... (JSON 格式化与发送) ...  send\_func("]\r\n");  Delay\_ms(500); // 修改此处的延时，单位为毫秒  }  } |

3.修改方法: 修改 Delay\_ms(500); 中的数值。如果您的上位机性能较好，可以适当减小此值（如 Delay\_ms(100);）以加快整体上报速度。如果出现数据丢失，可以尝试增大此值。

通过以上步骤，您可以自由地定制此数据采集模块的各项核心参数，使其完美匹配您的项目需求。在修改前，建议备份原始代码。