本文主要介绍流水灯实验和串口通信的实验过程，对串口协议和RS-232标准，RS232电平与TTL电平的区别，以及"USB/TTL转232"模块（以CH340芯片模块为例）的工作原理这些知识也有了一定的涉及。

目录

##

一、了解串口协议以及"USB/TTL转232"模块的工作原理

1、串口协议

串口通信指串口按位（bit）发送和接收字节。尽管比特字节（byte）的串行通信慢，但是串口可以在使用一根线发送数据的同时用另一根线接收数据。串口通信协议是指规定了数据包的内容，内容包含了起始位、主体数据、校验位及停止位，双方需要约定一致的数据包格式才能正常收发数据的有关规范。在串口通信中，常用的协议包括RS-232、RS-422和RS-485。

按照数据的传输方向，串口通信分为：

1. 单工：数据传输只支持数据在一个方向上传输；
2. 半双工：允许数据在两个方向上传输。但是，在某一时刻，只允许数据在一个方向上传输，它实际上是一种切换方向的单工通信；它不需要独立的接收端和发送端，两者可以合并一起使用一个端口；

3. 全双工：允许数据同时在两个方向上传输。因此，全双工通信是两个单工通信方式的结合，需要独立的接收端和发送端。

按照通信方式，分为

1. 同步通信：带时钟同步信号传输。比如：SPI，IIC通信接口。
2. 异步通信：不带时钟同步信号。比如：UART(通用异步收发器)，单总线。

在同步通信中，收发设备上方会使用一根信号线传输信号，在时钟信号的驱动下双方进行协调，同步数据。例如，通信中通常双方会统一规定在时钟信号的上升沿或者下降沿对数据线进行采样。

在异步通信中不使用时钟信号进行数据同步，它们直接在数据信号中穿插一些用于同步的信号位，或者将主题数据进行打包，以数据帧的格式传输数据。通信中还需要双方规约好数据的传输速率（也就是波特率）等，以便更好地同步。常用的波特率有4800bps、9600bps、115200bps等。

总结：在同步通信中，数据信号所传输的内容绝大部分是有效数据，而异步通信中会则会包含数据帧的各种标识符，所以同步通讯效率高，但是同步通讯双方的时钟允许误差小，稍稍时钟出错就可能导致数据错乱，异步通讯双方的时钟允许误差较大。

2、RS-232标准

RS-232标准接口（又称EIA RS-232）是常用的串行通信接口标准之一，它是由美国电子工业协会(EIA)联合贝尔系统公司、调制解调器厂家及计算机终端生产厂家于1970年共同制定，其全名是“数据终端设备(DTE)和数据通信设备(DCE)之间串行二进制数据交换接口技术标准”。RS-232（ANSI/EIA-232标准）是IBM-PC及其兼容机上的串行连接标准。可用于许多用途，比如连接鼠标、打印机或者Modem，同时也可以接工业仪器仪表。用于驱动和连线的改进，实际应用中RS-232的传输长度或者速度常常超过标准的值。RS-232只限于PC串口和设备间点对点的通信。RS-232串口通信最远距离是50英尺。

3、RS232电平与 TTL电平的区别

一、主体不同
1、TTL232：晶体管-晶体管逻辑集成电路。
2、RS232：数据终端设备(DTE)和数据通信设备(DCE)之间串行二进制数据交换接口技术标准。
二、数字含义不同
1、TTL232：TTL232的0是用0v表示。1是用5V表示。
2、RS232：0是用+3V–+15V表示，1是用-3V—15V表示。
三、传输不同
1、TTL232：是以某个固定的速率去传输的，但是可以传输多个bit
2、RS232：以固定的某个速率（1200bps,9600bps,115200bps等），一次只能只传输一个bit

4、“USB/TTL转232”模块的工作原理

1、模块特点
CH340C USB转TTL模块以CH340C芯片为核心，内部自带晶振，最高波特率可达2Mbps，软件兼容CH341驱动，过流保护，引出相应的通讯接口与电源接口，通讯接口带有指示灯指示工作状态，通讯稳定，体积小。

1. 全速USB驱动，兼容USB2.0
2. 硬件全双工串口，内置收发缓冲区
3. 支持波特率50bps~2Mbps
4. 输出TTL电平3.3V，兼容5V的IO电平

2、模块接口引脚

Symbol（符号）	Type （类型）	Deion（描述）
TXD	输出	串行数据输出口
RXD	输入	串行数据输出口
GND	电源	接地引脚
3V3	电源	3.3V电源输出引脚（最高250mA）
5V	电源	3.3V电源输出引脚（最高250mA）
DTS	输出	MODEM联络输出信号，请求发送
DTR	输出	MODEM联络输出信号，数据终端就绪

CSDN @Xicun1984

3、模块用途

电脑USB端是USB电平，单片机的信号是TTL电平，两者的电平不同是无法进行通讯的，需要通过转换才能实现相互通讯。CH340C USB 转TTL模块就是实现USB电平与TTL电平相互转换的模块。USB：采用VCC、GND、D+、D-传输，电脑上的插口就是USB接口。
TTL：一般指单片机的逻辑电平，不同单片的供电的系统TTL的电平不一样，3.3V单片的TTL电平就是高电平3.3V（逻辑1），低电平0V（逻辑0）。

在调试单片机程序的时候，想了解程序的执行情况或相关信息，一般简单的做法就是用串口把信息发送给电脑，电脑接收到再通过上位机（串口调试助手）显示出来。但是单片机串口发送的信号是TTL电平，电脑能接收到的信号是USB电平，两者无法直接通讯。USB转TTL模块可以把串口发送的TTL信号转换成USB信号再发送给电脑，电脑就可以接收到单片机发送过来的信号并在上位机（串口调试助手）上显示出来。
4、硬件介绍

引脚序号	引脚名称	类型	引脚说明
1	GND	电源	公共接地端，直接连到USB总线的地线
3	RXD	输入	串行数据输入
4	V3	电源	在3.3V电源电源电压时连接VCC输入外部电源；在5V电源电压时外接容量为0.1uF退藕电容
5	D+	USB信号	直接连接到USB总线的D+数据线
6	D-	USB信号	直接连接到USB总线的D-数据线
7	XI	输入	CH340C内部自带晶振，必须悬空
8	XO	输出	CH340C内部自带晶振，必须悬空
9	CTS#	输入	MODEM联络输入信号，清除发送，低（高）有效

CSDN @Xicun1984

在CH340C的引脚功能表中红色部分是电源相关的引脚。对于不同电压供电系统的TTL电平是不一样的，大部分的系统是5V或3.3V供电。一般5V的系统是兼容3.3V的TTL电平的，但是3.3V系统是不兼容5V的。为了是能兼容3.3V与5V的系统，模块的电源使用3.3V电源供电。

电脑的USB接口电源输出时5V，最大电流是500mA，在电路中为了防止意外的误操作，在5V的电源端加了一个0.5A,6V的保险丝F1，当电压超过6V或电流超过0.5A保险丝就会断开对电路进行保护。

USB输出的电压是5V，而CH340C的芯片采用3.3V供电，为了使模块的供电为3.3V，在电路中加入了一个LDO（低压差线性稳压器）U2，它可以把5V稳压成3.3V，然后对CH340C进行供电（根据手册要求V3引脚也要接3.3V的电源）。每一个电源的输入端都会加上一个0.1uF的滤波电容。

模块的USB转TTL电路设计

在CH340C的引脚功能表中蓝色部分是信号相关的引脚，黑色部分的与设计无关的引脚，全部悬空。CH340C芯片的D-，D+与USB的D-，D+连接到一起作为USB电平的信号连接，同时引出TTL电平信号的接口TXD与RXD，还有两个MODEM输出信号接口RTS与DTR。

二、基于stm32CubeMX，配合Keil，重做流水灯程序并且实现串口通信

1、重做流水灯程序

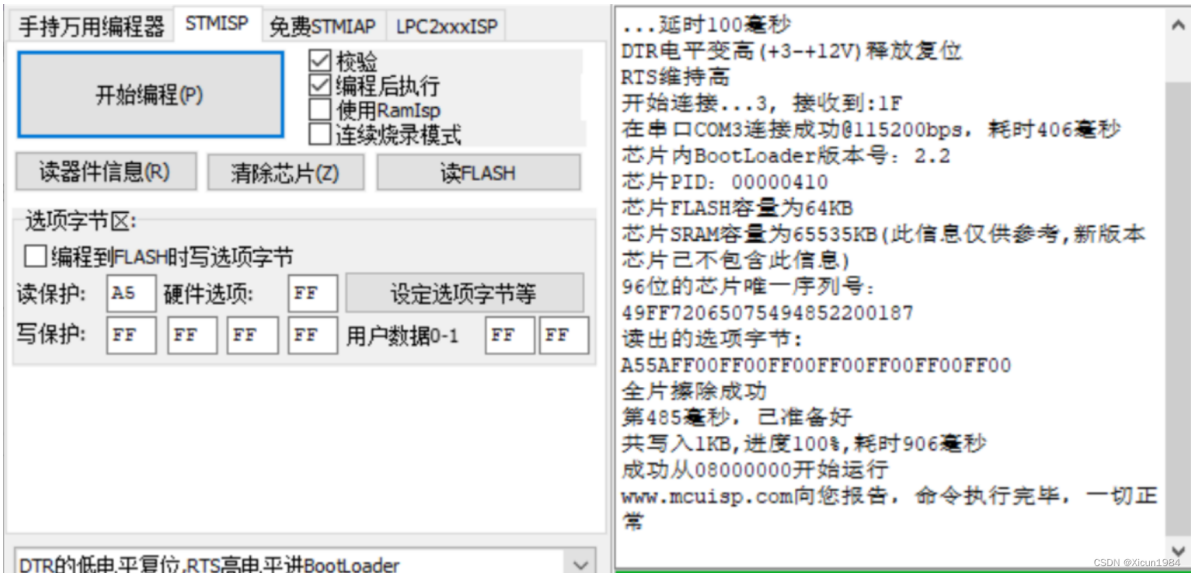
可以参考之前的博客内容，改变一下引脚位置，重做LED流水灯:https://blog.csdn.net/Xicun1984/article/details/127308398?spm=1001.2014.3001.5502

这里简单描述一下基于HAL库的流水灯实验：

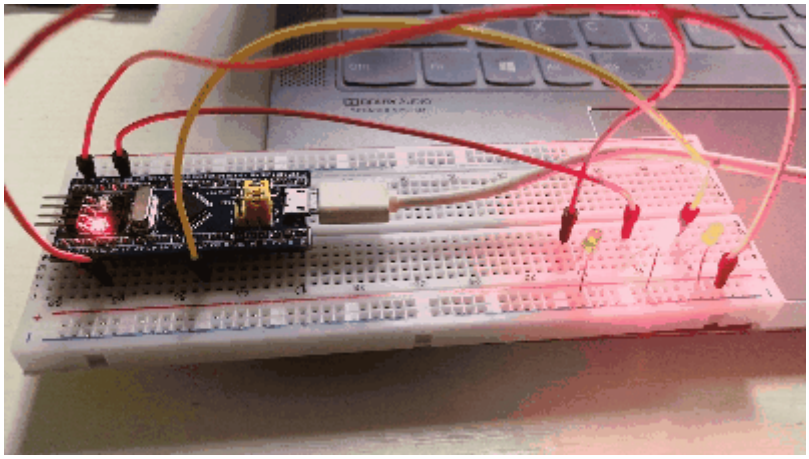
当我们在HAL库中将所有的基础配置设置好之后，点击生成工程，打开Keil文件，找到主函数的位置，将主函数的while循环部分改为这次的设置好的引脚代码：

```
HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,GPIO_PIN_RESET);HAL_GPIO_WritePin(GPIOA,GPIO_PIN_8,GPIO_PIN_SET);HAL_GPIO_WritePin(GPIOB,GPIO_PIN_4,GPIO_PIN_SET);HAL_Delay(1000);HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,GPIO_PIN_SET);HAL_GPIO_WritePin(GPIOA,GPIO_PIN_8,GPIO_PIN_RESET);HAL_GPIO_WritePin(GPIOB,GPIO_PIN_4,GPIO_PIN_SET);HAL_Delay(1000);HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,GPIO_PIN_SET);HAL_GPIO_WritePin(GPIOA,GPIO_PIN_8,GPIO_PIN_SET);HAL_GPIO_WritePin(GPIOB,GPIO_PIN_4,GPIO_PIN_RESET);HAL_Delay(1000);
```

接下来编译，烧录到STM32芯片之中



流水灯点亮情况：



2、完成一个STM32的USART串口通讯程序（查询方式即可，暂不要求采用中断方式）

（1）题目要求

完成一个STM32的USART串口通讯程序（查询方式即可，暂不要求采用中断方式），要求：

1）设置波特率为115200，1位停止位，无校验位；

2）STM32系统给上位机（win10）连续发送“hello windows！”。win10采用“串口助手”工具接收。

（2）USART概述

通用同步/异步串行接收/发送器
USART是一个全双工通用同步/异步串行收发模块，该接口是一个高度灵活的串行通信设备。

主要特点如下:

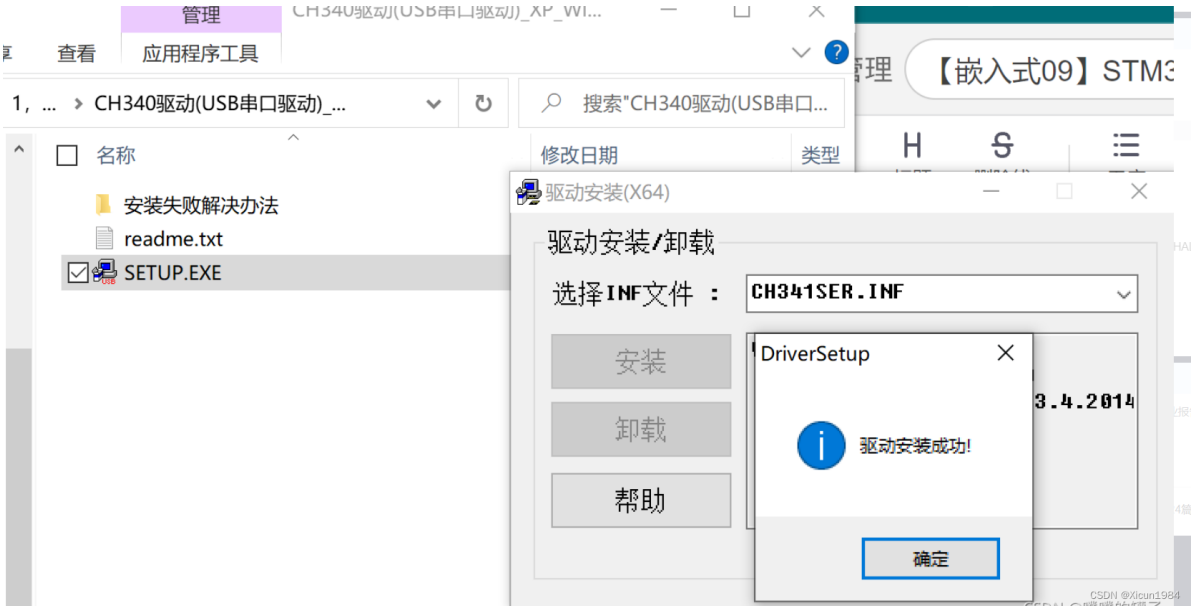
1. 全双工操作（相互独立的接收数据和发送数据）；
2. 同步操作时，可主机时钟同步，也可从机时钟同步；
3. 独立的高精度波特率发生器，不占用定时/计数器；
4. 支持5、6、7、8和9位数据位，1或2位停止位的串行数据帧结构；
5. 由硬件支持的奇偶校验位发生和检验；
6. 数据溢出检测；
7. 帧错误检测；
8. 包括错误起始位的检测噪声滤波器和数字低通滤波器；
9. 三个完全独立的中断，TX发送完成、TX发送数据寄存器空、RX接收完成；
10. 支持多机通信模式；
11. 支持倍速异步通信模式。

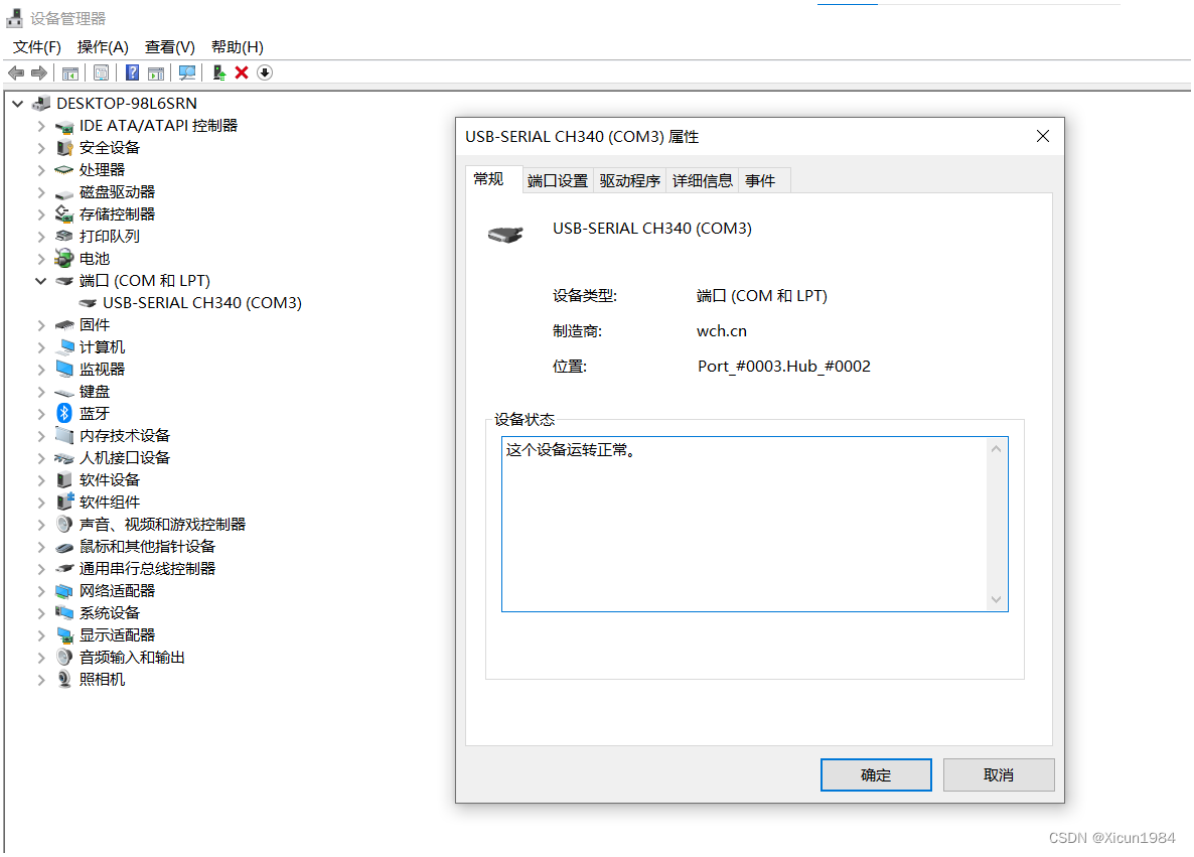
USART配置

STM32在只有一个中断的情况下，仍然需要配置优先级，其作用是使能某条中断的触发通道。STM32的中断有至多两个层次，分别是抢占优先级（主优先级）和子优先级（从优先级），而整个优先级设置参数的长度为4位，因此需要首先划分抢占优先级位数和子优先级位数，通过NVIC_PriorityGroupConfig()实现；
特定设备的中断优先级NVIC的属性包含在结构体NVIC_InitTypeDef中，其中字段NVIC_IRQChannel包含了设备的中断向量，保存在启动代码中；字段NVIC_IRQChannelPreemptionPriority为主优先级,NVIC_IRQChannelSubPriority为从优先级，取值的范围应根据位数划分的情况而定；最后NVIC_IRQChannelCmd字段是是否使能，一般置为ENABLE。最后通过NVIC_Init()来使能这一中断向量。

（3）实验环境准备

STM32F103C8T6最小核心板
USB转TTL
CH340驱动(USB串口驱动)_XP_WIN7共用
串口调试助手flymcu

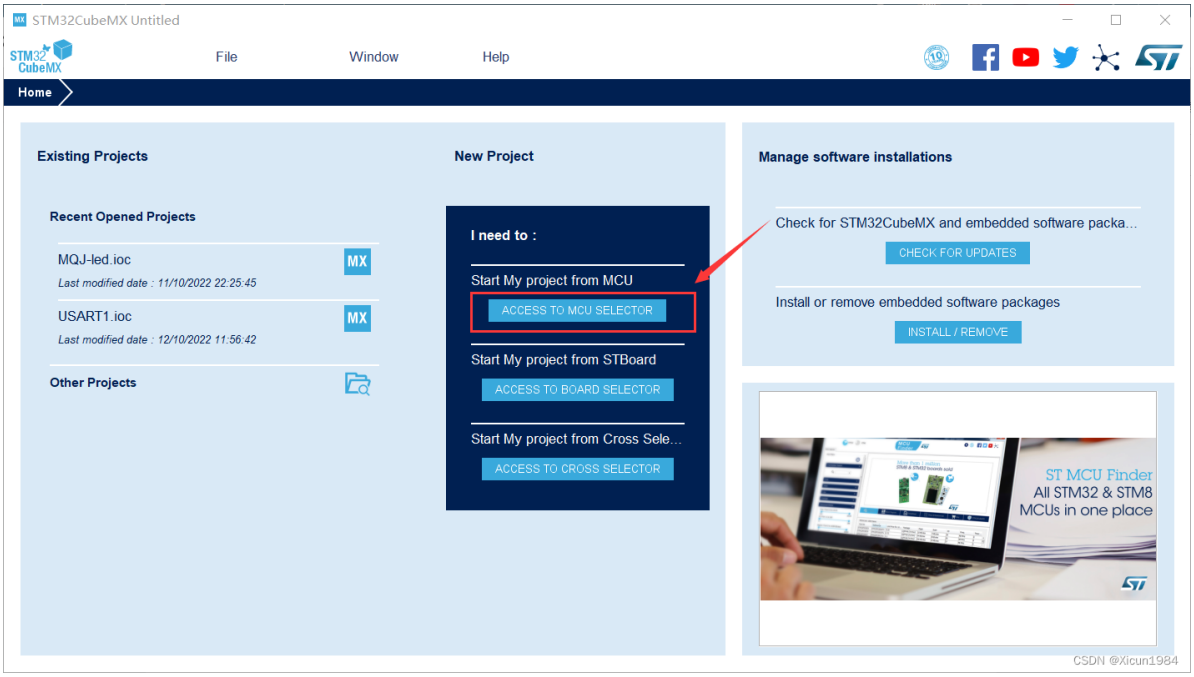




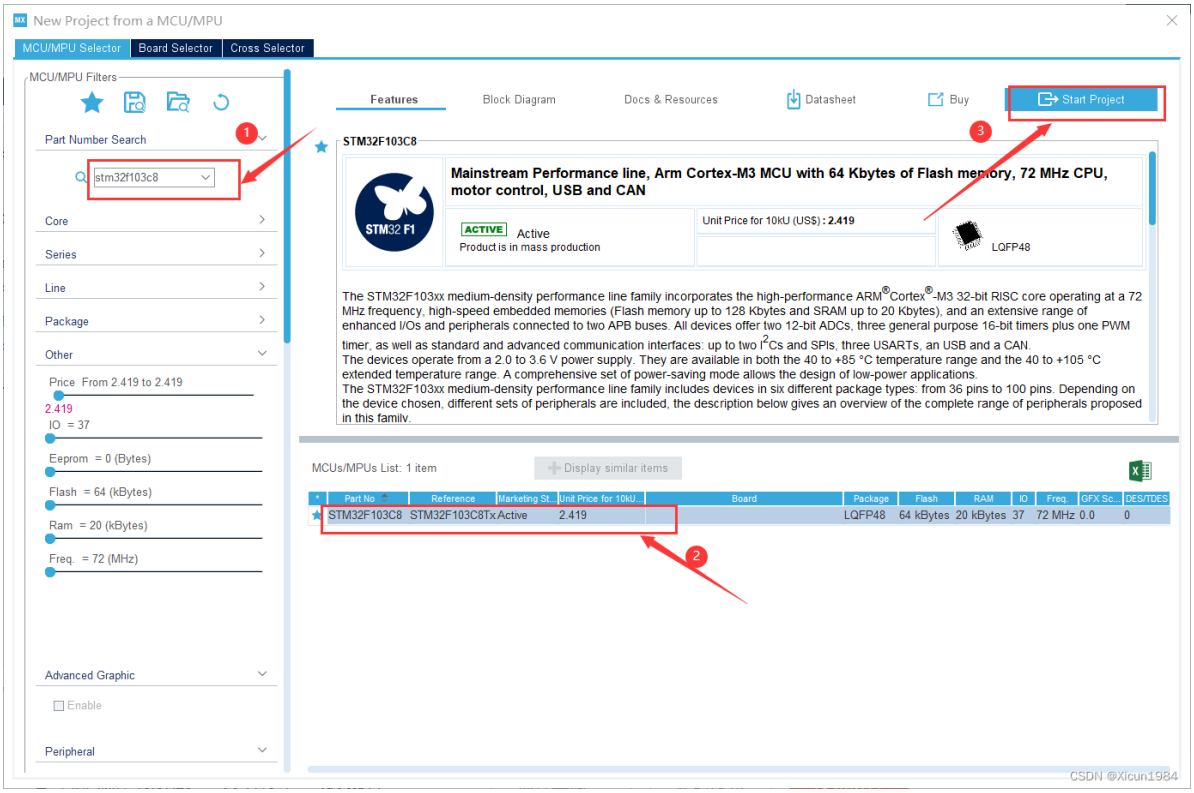
CSDN @Xicun1984

（4）HAL库实现

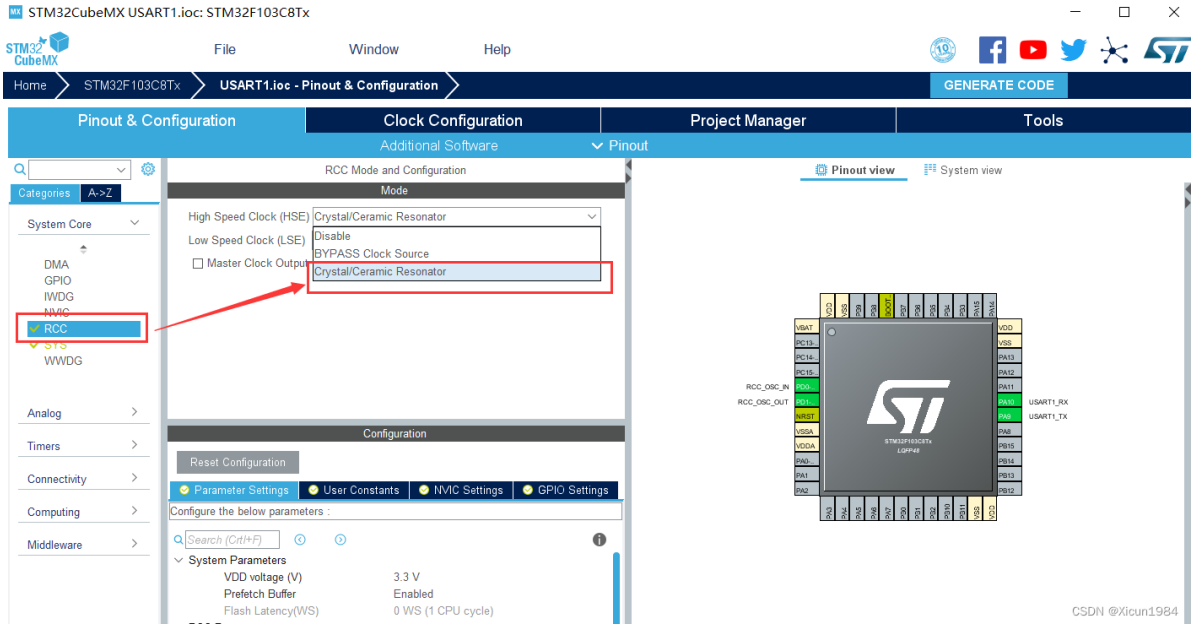
1、打开STM32CubeMX，选择新建工程：



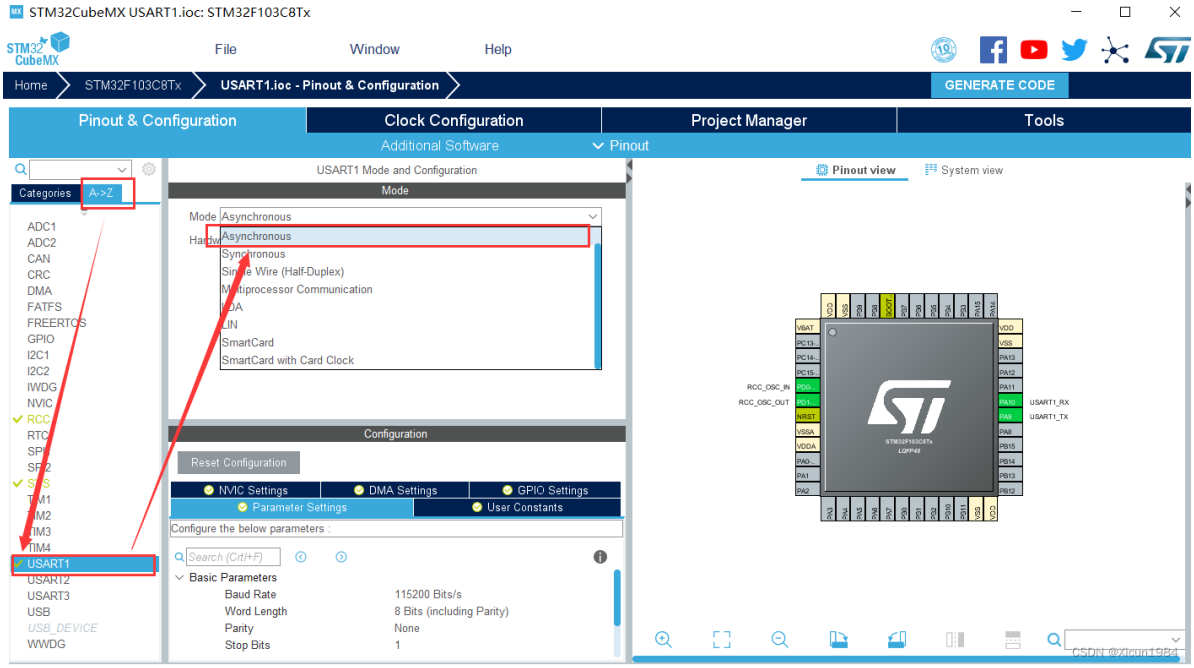
选择芯片STM32F103C8，启动工程



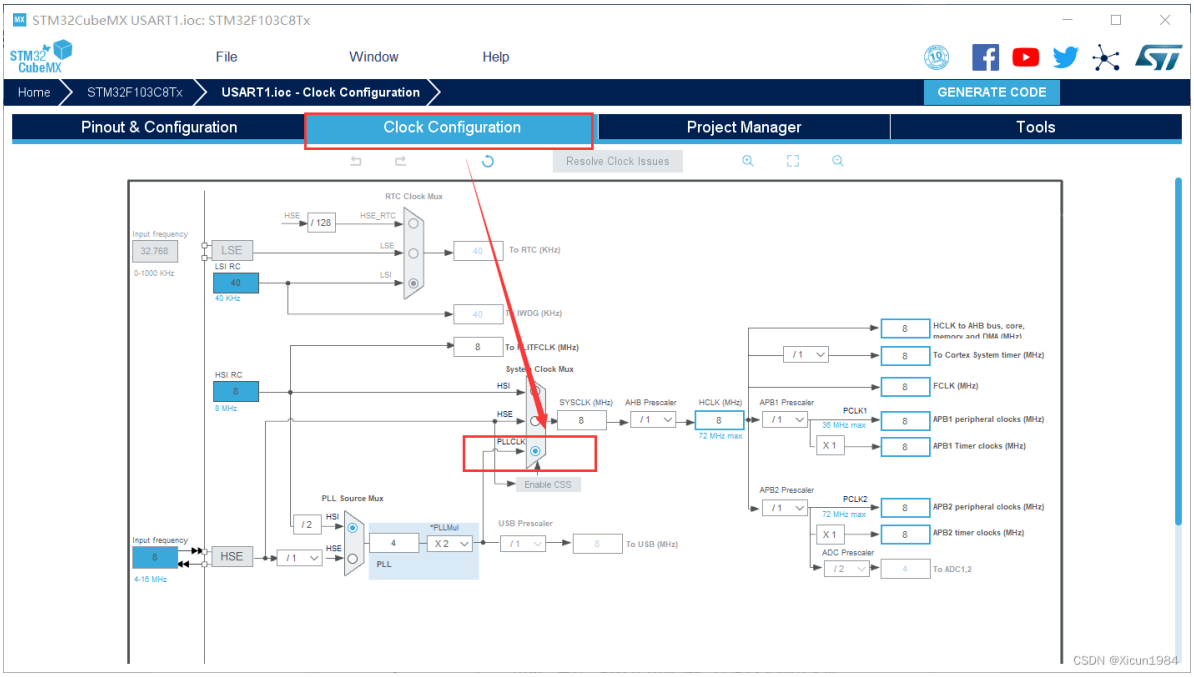
System Core-RCC-High Speed Clock-Crystal/Ceramic Resonator



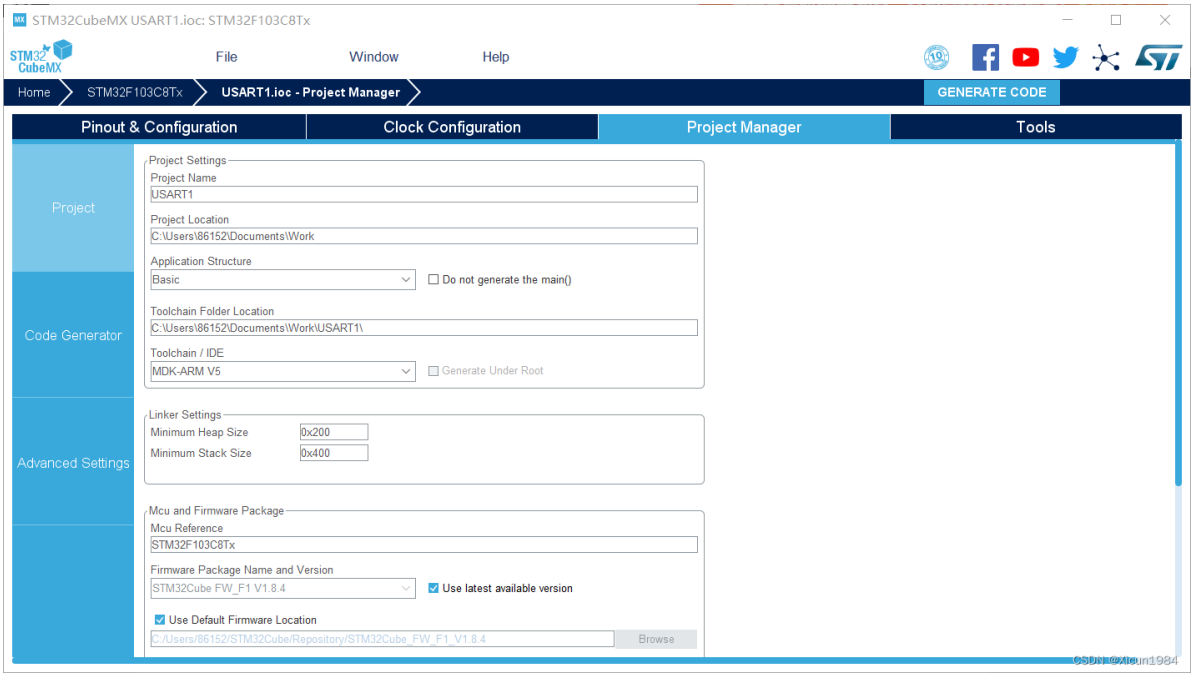
A->Z-USART1-Asynchronous

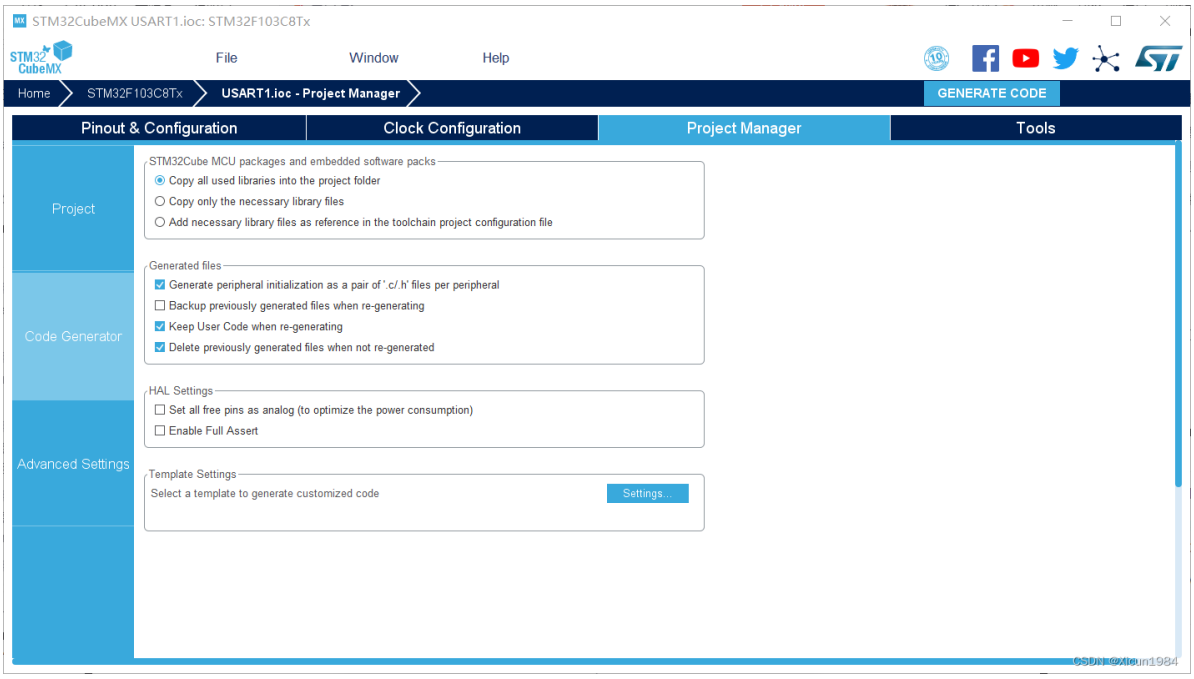


接下来在Clock Condigiguration中勾选PLLCLK：

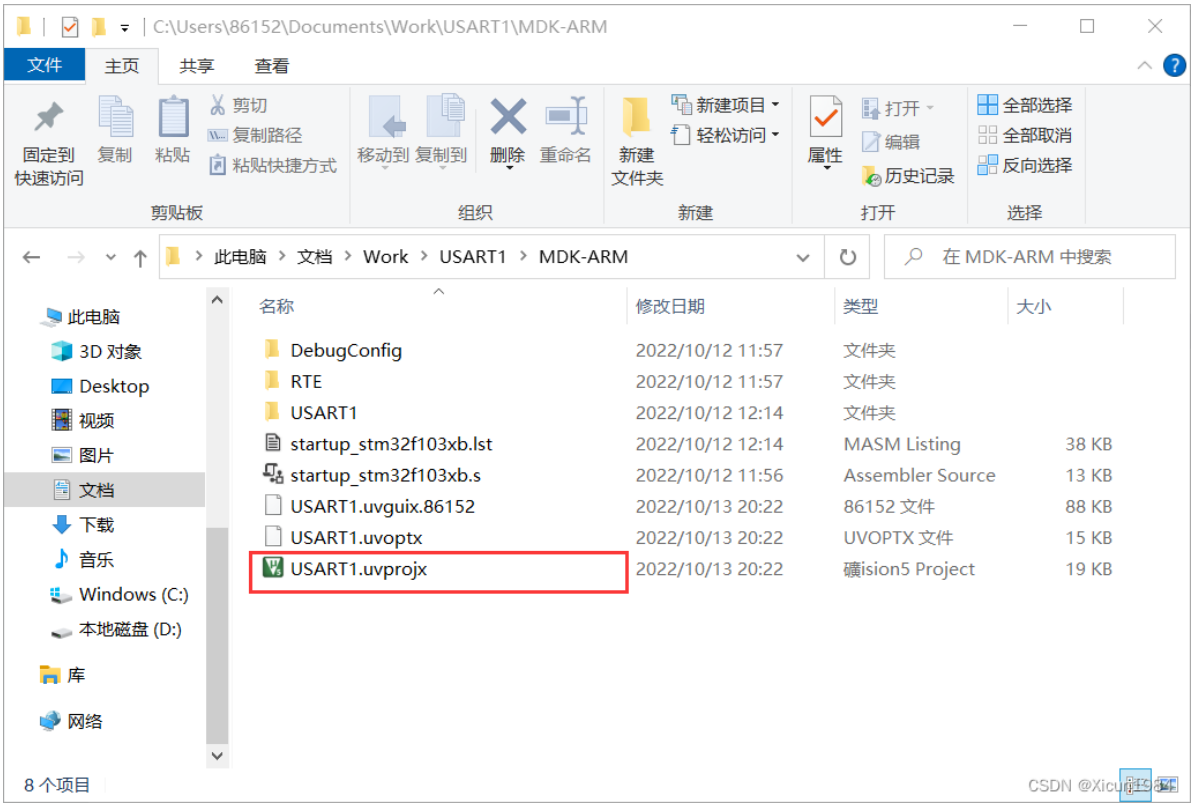

STM32CubeMX USART1.ioc: STM32F103C8Tx
File
Window
Help
Home
STM32F103C8Tx
USART1.ioc - Clock Configuration
GENERATE CODE
Pinout & Configuration
Clock Configuration
Project Manager
Tools
Resolve Clock Issues
RTC Clock Mux
HSE
/ 128
HSE_RTC
LSE
LSI
To RTC (KHz)
To IWDG (KHz)
To FLITFCLK (MHz)
System Clock Mux
HSI
HSE
PLLCLK
SYSCLK (MHz)
AHB Prescaler
HCLK (MHz)
72 MHz max
APB1 Prescaler
APB2 Prescaler
PCLK1
PCLK2
HCLK to AHB bus, core, memory and DMA (MHz)
To Cortex System timer (MHz)
FCLK (MHz)
APB1 peripheral clocks (MHz)
APB1 Timer clocks (MHz)
APB2 peripheral clocks (MHz)
APB2 timer clocks (MHz)
ADC Prescaler
To ADC1,2
PLL Source Mux
USB Prescaler
To USB (MHz)
Enable CSS
Input frequency
32.768
0-1000 KHz
LSE
LSI RC
40 KHz
HSI RC
8 MHz
4-16 MHz
PLL
*PLLMul
X 2
X 1
CSDN @Xicun1984

新建文件名


STM32CubeMX USART1.ioc: STM32F103C8Tx
File
Window
Help
Home
STM32F103C8Tx
USART1.ioc - Project Manager
GENERATE CODE
Pinout & Configuration
Clock Configuration
Project Manager
Tools
Project
Code Generator
Advanced Settings
Project Settings
Project Name
USART1
Project Location
C:\Users\86152\Documents\Work
Application Structure
Basic
Do not generate the main()
Toolchain Folder Location
C:\Users\86152\Documents\Work\USART1\
Toolchain / IDE
MDK-ARM V5
Generate Under Root
Linker Settings
Minimum Heap Size
0x200
Minimum Stack Size
0x400
Mcu and Firmware Package
Mcu Reference
STM32F103C8Tx
Firmware Package Name and Version
STM32Cube FW_F1 V1.8.4
Use latest available version
Use Default Firmware Location
C:/Users/86152/STM32Cube/Repository/STM32Cube_FW_F1_V1.8.4
Browse
CSDN @Xicun1984

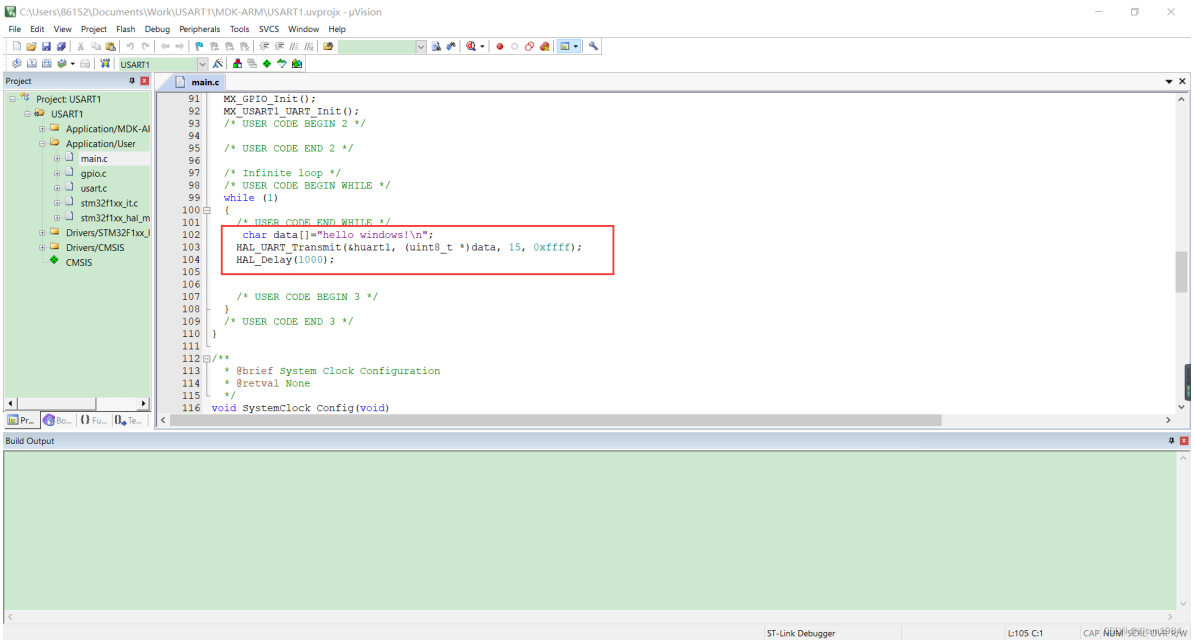


接下来就是生成代码，并且打开文件夹和Keil文件

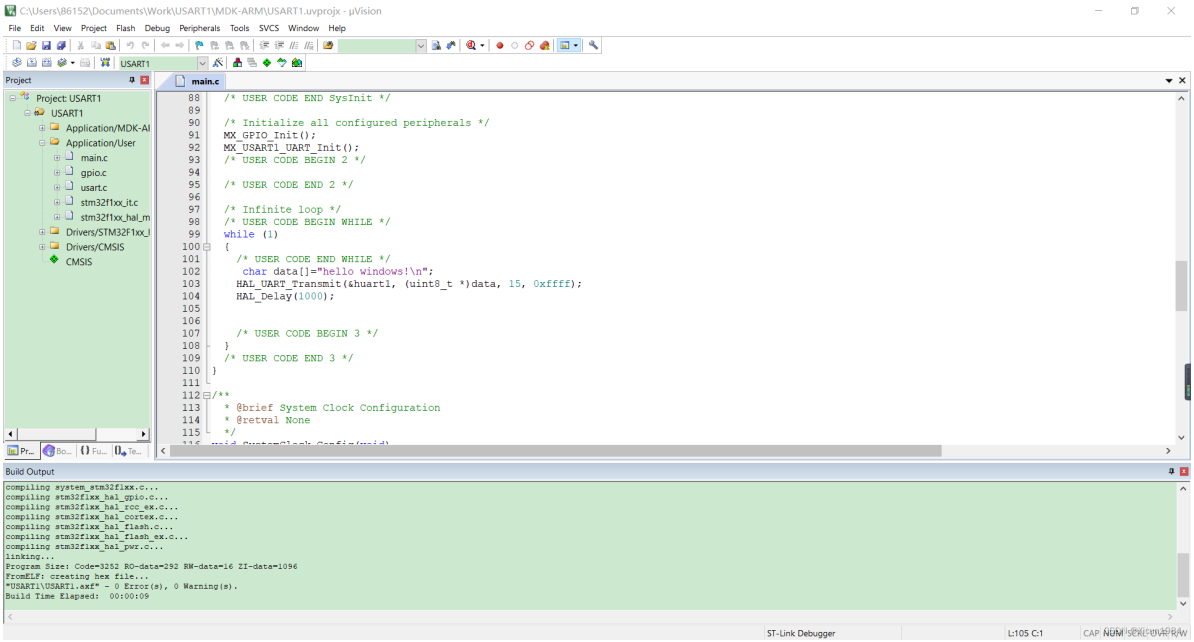


找到并打开main.c函数，打开，将以下的代码添加到while循环之中：

```
char data[]="hello windows!\n";HAL_UART_Transmit(&huart1, (uint8_t *)data, 15, 0xffff);HAL_Delay(1000);
```

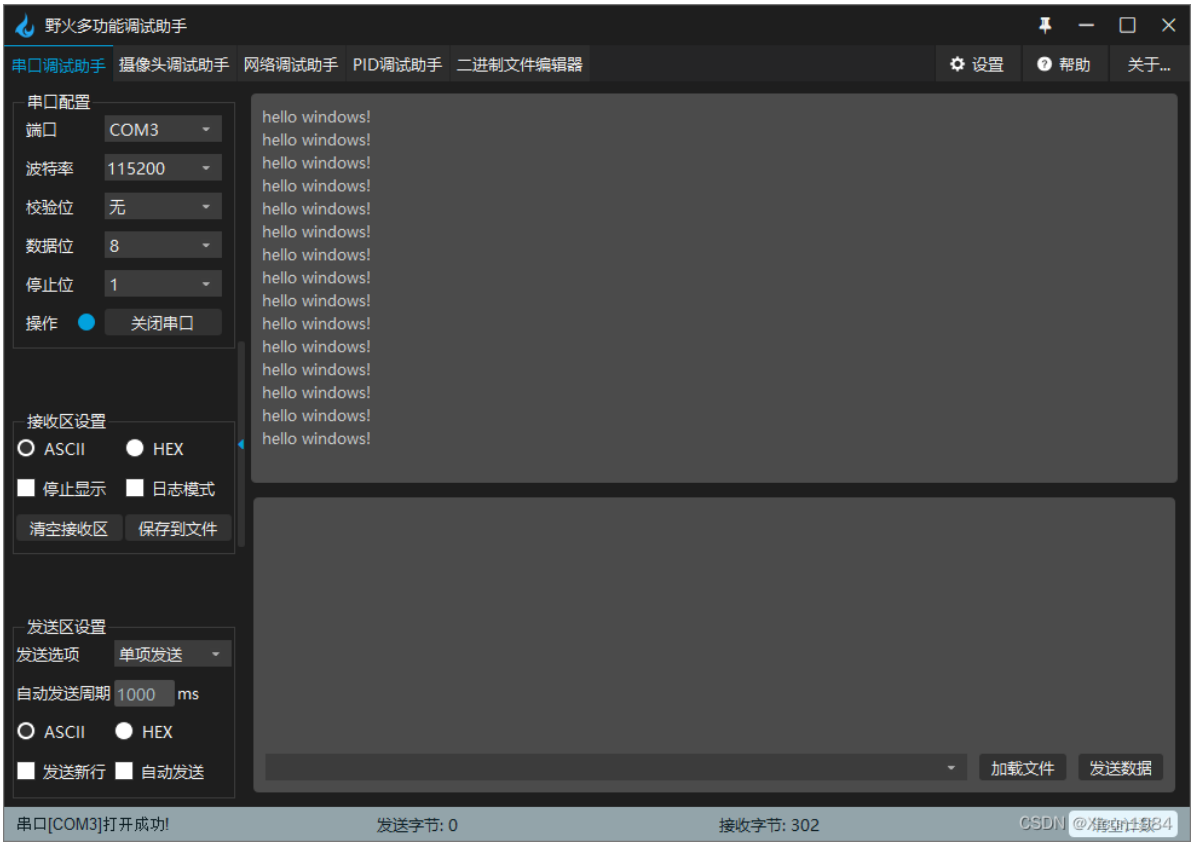


编译运行生成hex文件：



借用flymcu烧录运行：

调试结果：



说明代码运行成功，实现了串口通信。

3、逻辑分析仪观察时序波形和输出波形

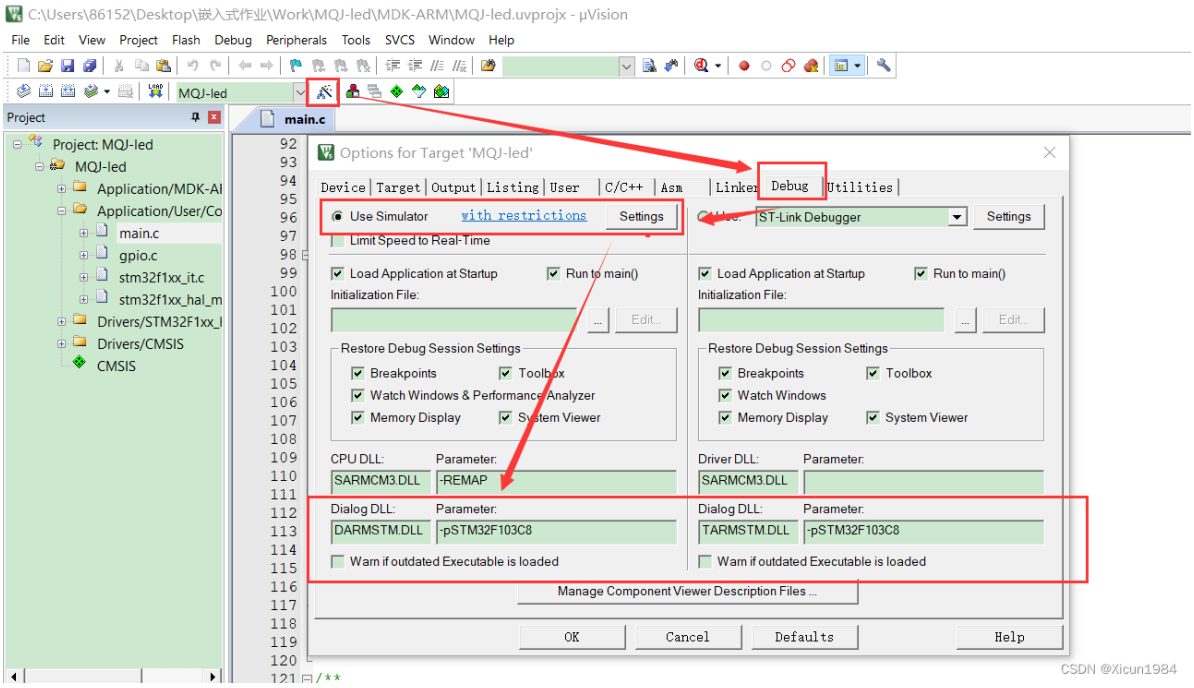
（1）要求

在没有示波器条件下，可以使用Keil的软件仿真逻辑分析仪功能观察管脚的时序波形，更方便动态跟踪调试和定位代码故障点。 请用此功能观察第1题中3个GPIO端口的输出波形，和第2题中串口输出波形，并分析时序状态正确与否，高低电平转换周期（LED闪烁周期）实际为多少。

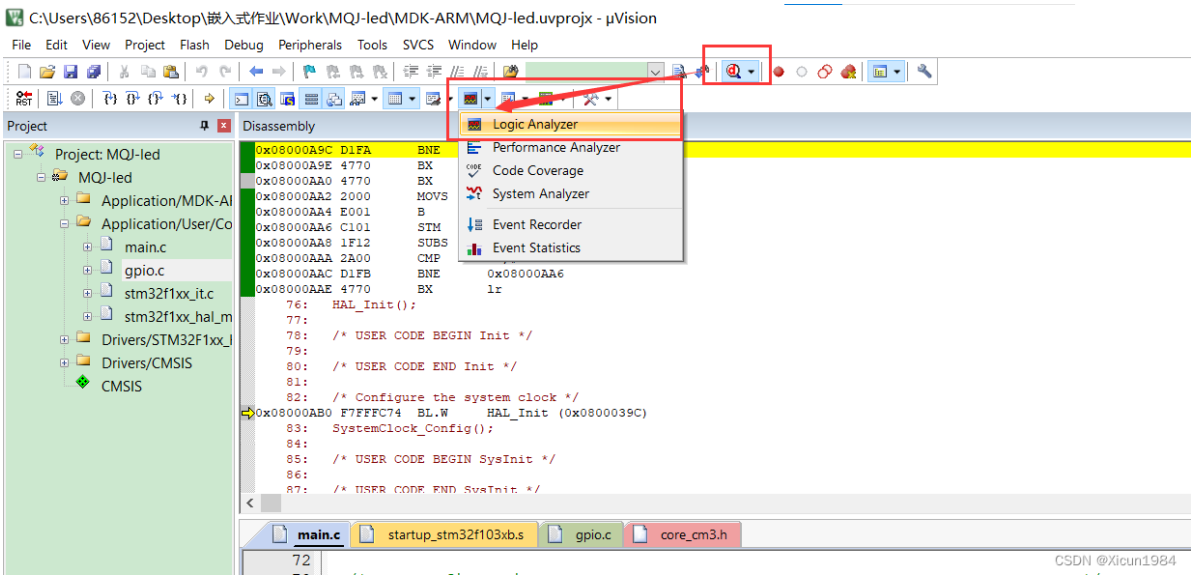
（2）观察GPIO端口输出波形

打开LED流水灯的keil文件，进行仿真设置

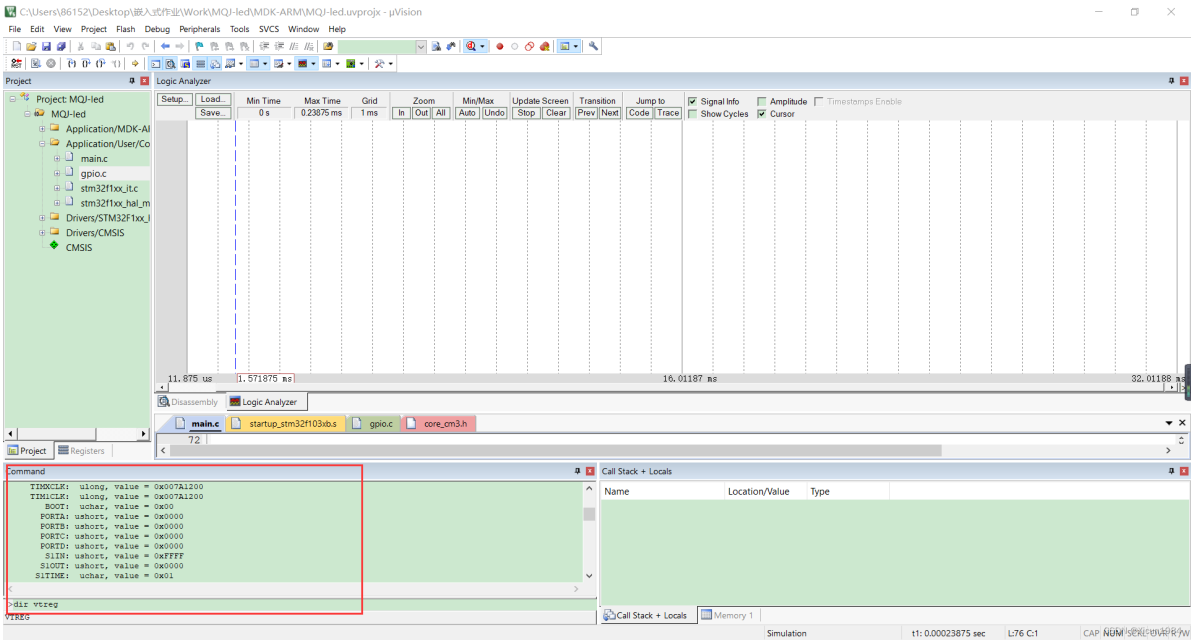
点击“魔方棒”，打开工程设置，在debug中修改以下设置。



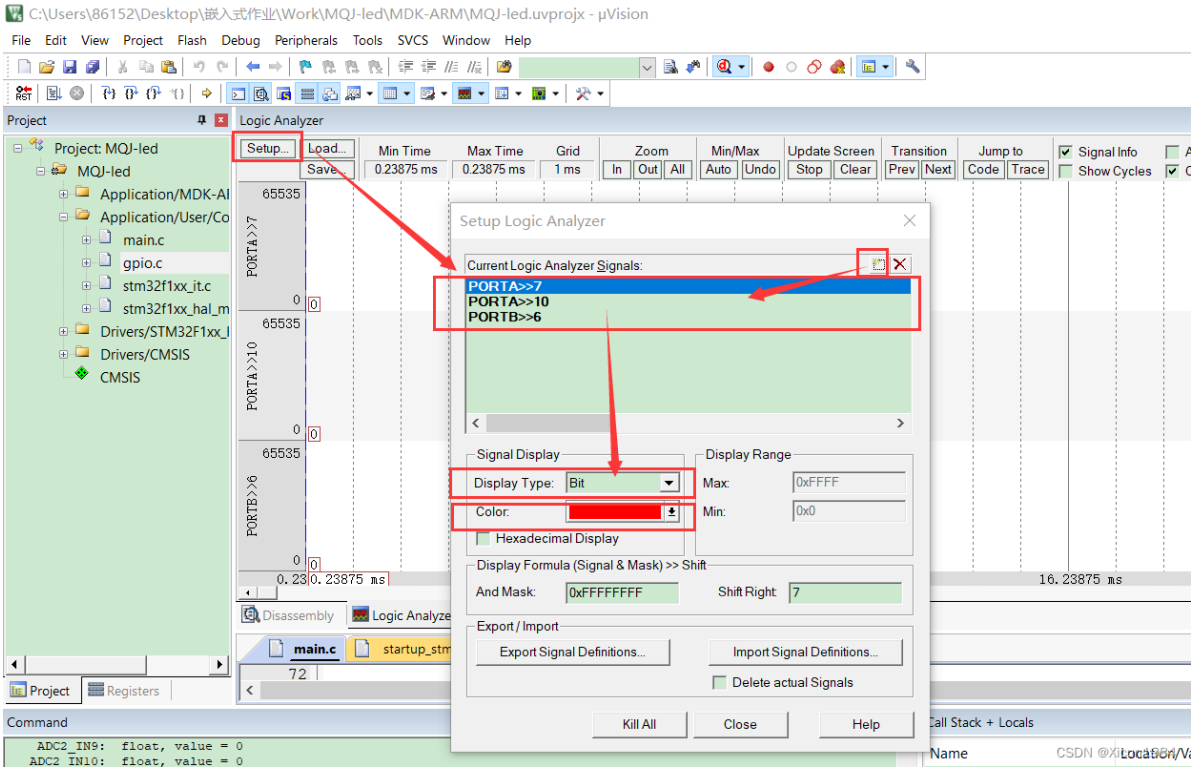
设置完成后，开启调试模式，打开逻辑分析仪



打开逻辑分析仪之后在左下角命令行输入dir vtreg命令，查看有哪些引脚可以被检测。

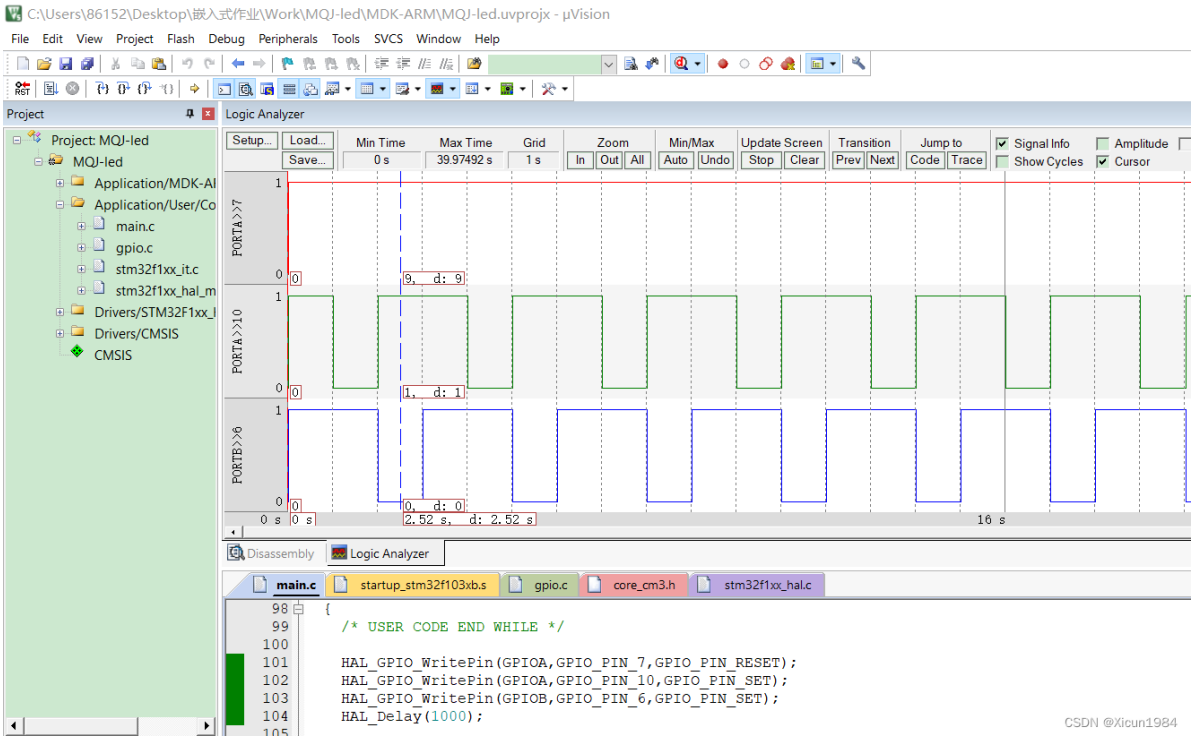


接下来在逻辑分析仪中添加要观察的引脚端口，可以使用上述方法找到并添加，本文推荐下面的方法，**直接以 PORTX >> X 的形式输入，内容取决于代码中定义的管脚：**



注意设置Display Type类型为Bit，颜色设置按照自己喜欢。

运行程序一段时间，当逻辑分析仪上呈现出明显的波形时，停止运行：

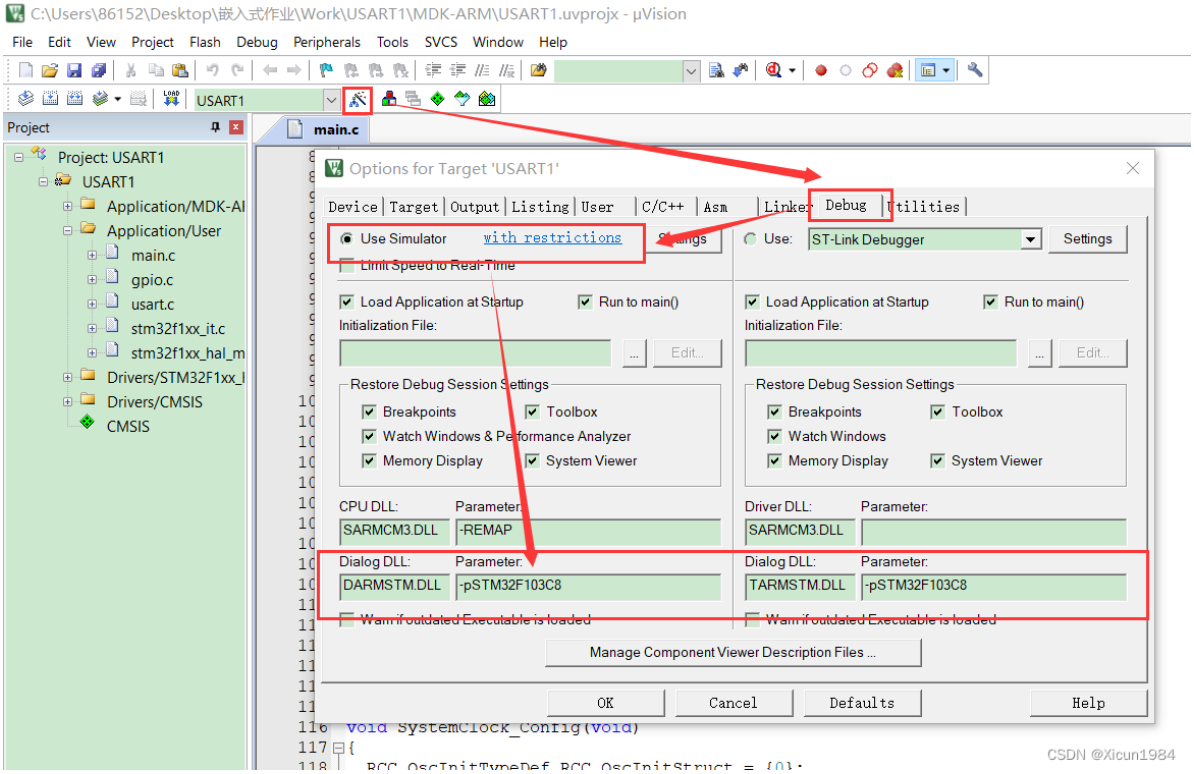


根据实验结果可得：每次变化时间基本接近1s，与预设时间差不多。

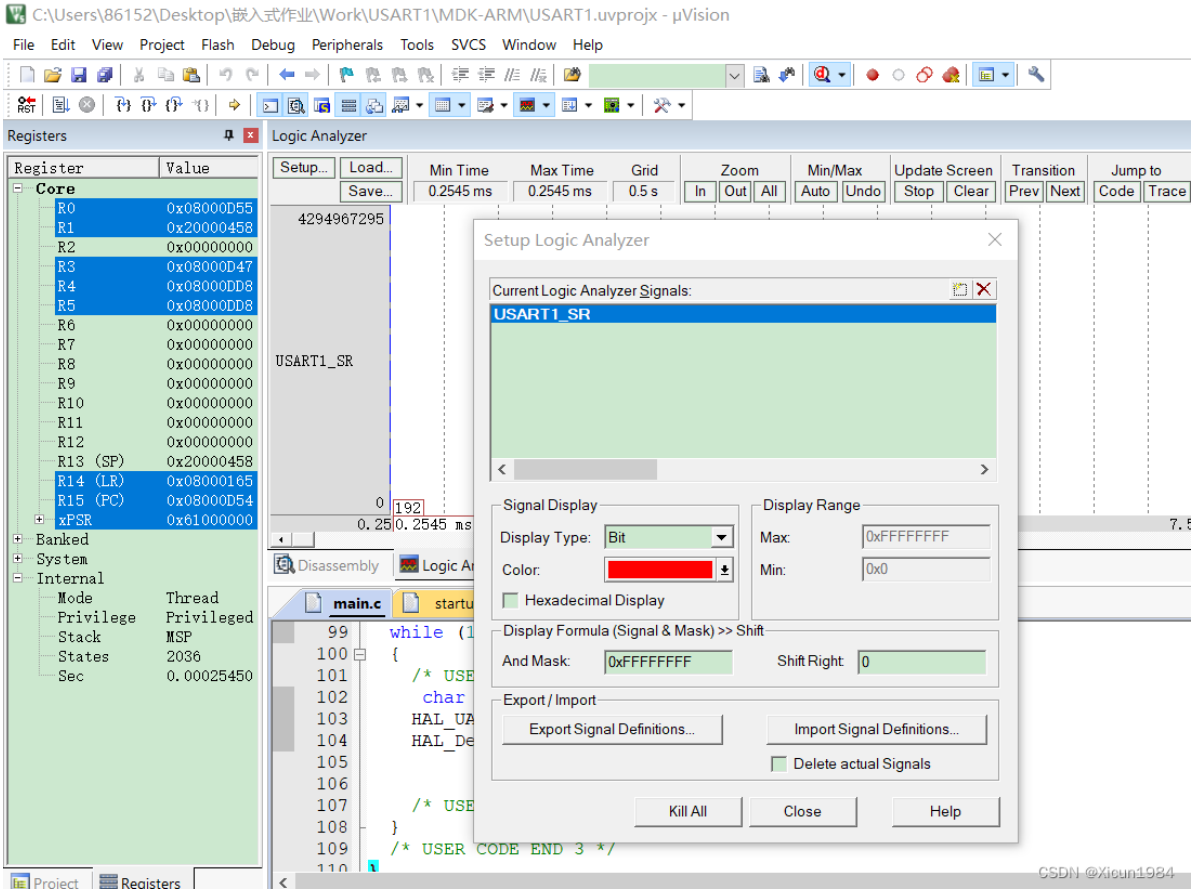
（3）观察串口输出波形

打开串口通信keil文件

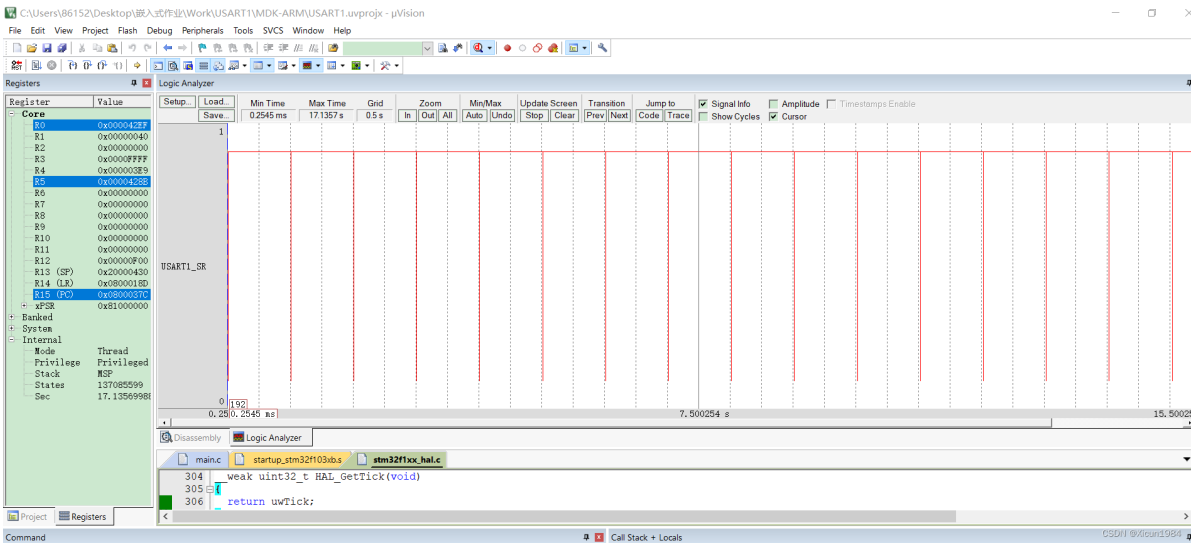
设置仿真环境：



添加端口



全速运行一段时间后停止，得到以下的波形图。



观察每次变化的时间，其实只有0.92s，并没有实际定义的1000ms。

三、总结

这次实验综合学习了流水灯实验和串口通信，对串口协议和RS-232标准，RS232电平与TTL电平的区别，以及"USB/TTL转232"模块（以CH340芯片模块为例）的工作原理这些知识也有了一定的了解，而且我发现HAL库提供的可视化界面大大提高了编程效率，无需再利用代码去配置管脚了，效果很好，受益匪浅。

四、参考文献

https://blog.csdn.net/zjszd/article/details/120933702
https://blog.csdn.net/junseven164/article/details/120807138
https://blog.csdn.net/as480133937/article/details/98885316
https://blog.csdn.net/as480133937/article/details/98947162
https://blog.csdn.net/vic_to_ry/article/details/110451036