血氧监测仪项目

尚硅谷研究院

版本: V1.0

1. 项目需求和技术选型

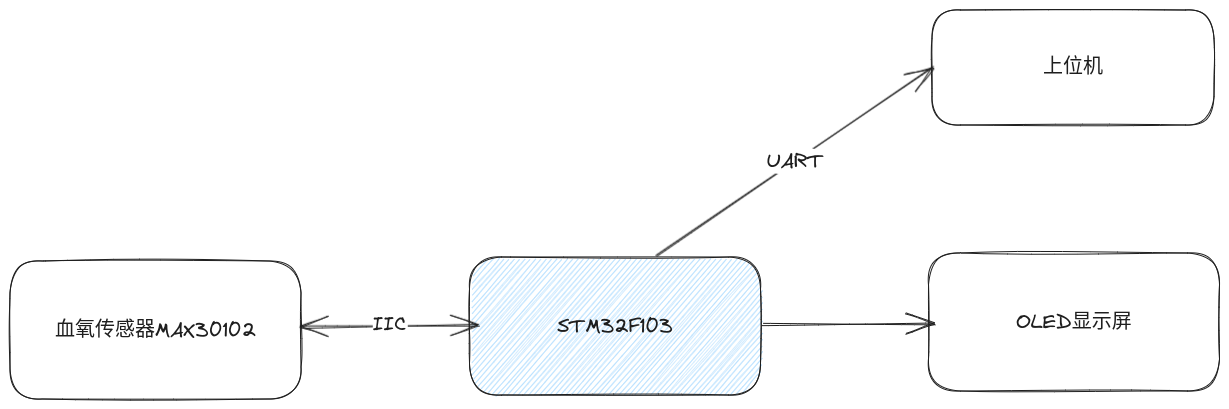
* 1. 硬件选型

我们的需求是制作一个血氧监测仪。血氧监测仪需要以下一些硬件设备：

* 指压血氧传感器：用来采集血氧信号，我们选择最常用的MAX30102传感器。
* 单片机：我们选择最常用的 STM32F103 系列的单片机。用来读写MAX30102传感器。
* OLED: 用来显示被测人员的血氧数据。我们选择最常用的 12864 OLED 显示屏。

* 1. 软件选型
* STM32CubeMX：用来产生样板代码。
* Clion/VSCode/Keil：用来开发单片机程序。

1. 项目架构图



1. 工具链

传统的 STM32 开发工具一般使用 Keil MDK-ARM IDE 来进行。但 Keil IDE 的编码体验非常糟糕（只支持 Windows 平台，代码高亮和补全都不好，经常误报错误和警告）。所以我们自己搭建一套开发体验（代码补全，高亮，提示，代码格式化等等）丝滑的工具链。选用的工具如下：

* STM32CubeMX：用来产生包含了 ST 官方 HAL 库的样板代码。非常的方便，基本上是开发 STM32 单片机程序的必备工具。[下载链接](https://www.st.com/zh/development-tools/stm32cubemx.html#overview)
* Clion/Visual Studio Code：用于开发单片机程序。
* openocd：开源的单片机程序烧写工具。[下载链接](https://gnutoolchains.com/arm-eabi/openocd/)。下载完解压之后，注意将 bin 文件夹的绝对路径添加到环境变量中。打开 PowerShell ，然后输入 openocd -v 确认安装成功。
* gcc-arm-none-eabi：开源的 ARM 指令集编译器。[下载链接](https://developer.arm.com/-/media/Files/downloads/gnu-rm/10.3-2021.10/gcc-arm-none-eabi-10.3-2021.10-win32.exe?rev=29bb46cfa0434fbda93abb33c1d480e6&hash=B2C5AAE07841929A0D0BF460896D6E52)。安装完之后，找到安装路径。然后将 bin 文件夹的绝对路径添加到环境变量中。打开 PowerShell ，然后输入 arm-none-eabi-gcc.exe -v 确认安装成功。

1. 项目代码编写

* 1. 使用 STM32CubeMX 生成的配置中断和外设初始化的代码

中断配置代码

Core/Src/stm32f1xx\_it.c

#include "main.h"  
#include "stm32f1xx\_it.h"  
  
#include "max30102\_for\_stm32\_hal.h"  
extern max30102\_t max30102;  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\* Cortex-M3 Processor Interruption and Exception Handlers \*/  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\*\*  
 \* @brief This function handles Non maskable interrupt.  
 \*/  
void NMI\_Handler(void)  
{  
}  
  
/\*\*  
 \* @brief This function handles Hard fault interrupt.  
 \*/  
void HardFault\_Handler(void)  
{  
 while (1)  
 {  
 }  
}  
  
/\*\*  
 \* @brief This function handles Memory management fault.  
 \*/  
void MemManage\_Handler(void)  
{  
 while (1)  
 {  
 }  
}  
  
/\*\*  
 \* @brief This function handles Prefetch fault, memory access fault.  
 \*/  
void BusFault\_Handler(void)  
{  
 while (1)  
 {  
 }  
}  
  
/\*\*  
 \* @brief This function handles Undefined instruction or illegal state.  
 \*/  
void UsageFault\_Handler(void)  
{  
 while (1)  
 {  
 }  
}  
  
/\*\*  
 \* @brief This function handles System service call via SWI instruction.  
 \*/  
void SVC\_Handler(void)  
{  
}  
  
/\*\*  
 \* @brief This function handles Debug monitor.  
 \*/  
void DebugMon\_Handler(void)  
{  
}  
  
/\*\*  
 \* @brief This function handles Pendable request for system service.  
 \*/  
void PendSV\_Handler(void)  
{  
}  
  
/\*\*  
 \* @brief This function handles System tick timer.  
 \*/  
void SysTick\_Handler(void)  
{  
 HAL\_IncTick();  
}  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\* STM32F1xx Peripheral Interrupt Handlers \*/  
/\* Add here the Interrupt Handlers for the used peripherals. \*/  
/\* For the available peripheral interrupt handler names, \*/  
/\* please refer to the startup file (startup\_stm32f1xx.s). \*/  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
/\*\*  
 \* @brief This function handles EXTI line0 interrupt.  
 \*/  
void EXTI9\_5\_IRQHandler(void)  
{  
 max30102\_on\_interrupt(&max30102);  
 HAL\_GPIO\_EXTI\_IRQHandler(GPIO\_PIN\_9);  
}

外设代码

Core/Src/stm32f1xx\_hal\_msp.c

#include "main.h"  
  
/\*\*  
 \* Initializes the Global MSP.  
 \*/  
void HAL\_MspInit(void)  
{  
 \_\_HAL\_RCC\_AFIO\_CLK\_ENABLE();  
 \_\_HAL\_RCC\_PWR\_CLK\_ENABLE();  
}  
  
/\*\*  
 \* @brief I2C MSP Initialization  
 \* This function configures the hardware resources used in this example  
 \* @param hi2c: I2C handle pointer  
 \* @retval None  
 \*/  
void HAL\_I2C\_MspInit(I2C\_HandleTypeDef \*hi2c)  
{  
 GPIO\_InitTypeDef GPIO\_InitStruct = {0};  
 if (hi2c->Instance == I2C1)  
 {  
 \_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();  
 /\*\*I2C1 GPIO Configuration  
 PB6 ------> I2C1\_SCL  
 PB7 ------> I2C1\_SDA  
 \*/  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_6 | GPIO\_PIN\_7;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_OD;  
 GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_HIGH;  
 HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);  
  
 /\* Peripheral clock enable \*/  
 \_\_HAL\_RCC\_I2C1\_CLK\_ENABLE();  
 }  
}  
  
/\*\*  
 \* @brief I2C MSP De-Initialization  
 \* This function freeze the hardware resources used in this example  
 \* @param hi2c: I2C handle pointer  
 \* @retval None  
 \*/  
void HAL\_I2C\_MspDeInit(I2C\_HandleTypeDef \*hi2c)  
{  
 if (hi2c->Instance == I2C1)  
 {  
 /\* Peripheral clock disable \*/  
 \_\_HAL\_RCC\_I2C1\_CLK\_DISABLE();  
  
 /\*\*I2C1 GPIO Configuration  
 PB6 ------> I2C1\_SCL  
 PB7 ------> I2C1\_SDA  
 \*/  
 HAL\_GPIO\_DeInit(GPIOB, GPIO\_PIN\_6);  
  
 HAL\_GPIO\_DeInit(GPIOB, GPIO\_PIN\_7);  
 }  
}  
  
/\*\*  
 \* @brief UART MSP Initialization  
 \* This function configures the hardware resources used in this example  
 \* @param huart: UART handle pointer  
 \* @retval None  
 \*/  
void HAL\_UART\_MspInit(UART\_HandleTypeDef \*huart)  
{  
 GPIO\_InitTypeDef GPIO\_InitStruct = {0};  
 if (huart->Instance == USART2)  
 {  
 /\* Peripheral clock enable \*/  
 \_\_HAL\_RCC\_USART2\_CLK\_ENABLE();  
  
 \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();  
 /\*\*USART2 GPIO Configuration  
 PA2 ------> USART2\_TX  
 PA3 ------> USART2\_RX  
 \*/  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_2;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;  
 GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_HIGH;  
 HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);  
  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_3;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;  
 GPIO\_InitStruct.Pull = GPIO\_NOPULL;  
 HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);  
 }  
}  
  
/\*\*  
 \* @brief UART MSP De-Initialization  
 \* This function freeze the hardware resources used in this example  
 \* @param huart: UART handle pointer  
 \* @retval None  
 \*/  
void HAL\_UART\_MspDeInit(UART\_HandleTypeDef \*huart)  
{  
 if (huart->Instance == USART2)  
 {  
 /\* Peripheral clock disable \*/  
 \_\_HAL\_RCC\_USART2\_CLK\_DISABLE();  
  
 /\*\*USART2 GPIO Configuration  
 PA2 ------> USART2\_TX  
 PA3 ------> USART2\_RX  
 \*/  
 HAL\_GPIO\_DeInit(GPIOA, GPIO\_PIN\_2 | GPIO\_PIN\_3);  
 }  
}

* 1. MAX30102驱动的编写

先来编写驱动程序的头文件，定义了一些宏定义和函数原型。

Core/Inc/max30102\_for\_stm32\_hal.h

#ifndef MAX30102\_FOR\_STM32\_HAL\_H  
#define MAX30102\_FOR\_STM32\_HAL\_H  
  
#include "main.h"  
#include <stdint.h>  
#include <string.h>  
#include <stdlib.h>  
#include <string.h>  
  
#define MAX30102\_I2C\_ADDR 0x57  
#define MAX30102\_I2C\_TIMEOUT 1000  
  
#define MAX30102\_BYTES\_PER\_SAMPLE 6  
#define MAX30102\_SAMPLE\_LEN\_MAX 32  
  
#define MAX30102\_INTERRUPT\_STATUS\_1 0x00  
#define MAX30102\_INTERRUPT\_STATUS\_2 0x01  
#define MAX30102\_INTERRUPT\_ENABLE\_1 0x02  
#define MAX30102\_INTERRUPT\_ENABLE\_2 0x03  
#define MAX30102\_INTERRUPT\_A\_FULL 7  
#define MAX30102\_INTERRUPT\_PPG\_RDY 6  
#define MAX30102\_INTERRUPT\_ALC\_OVF 5  
#define MAX30102\_INTERRUPT\_DIE\_TEMP\_RDY 1  
  
#define MAX30102\_FIFO\_WR\_PTR 0x04  
#define MAX30102\_OVF\_COUNTER 0x05  
#define MAX30102\_FIFO\_RD\_PTR 0x06  
  
#define MAX30102\_FIFO\_DATA 0x07  
  
#define MAX30102\_FIFO\_CONFIG 0x08  
#define MAX30102\_FIFO\_CONFIG\_SMP\_AVE 5  
#define MAX30102\_FIFO\_CONFIG\_ROLL\_OVER\_EN 4  
#define MAX30102\_FIFO\_CONFIG\_FIFO\_A\_FULL 0  
  
#define MAX30102\_MODE\_CONFIG 0x09  
#define MAX30102\_MODE\_SHDN 7  
#define MAX30102\_MODE\_RESET 6  
#define MAX30102\_MODE\_MODE 0  
  
#define MAX30102\_SPO2\_CONFIG 0x0a  
#define MAX30102\_SPO2\_ADC\_RGE 5  
#define MAX30102\_SPO2\_SR 2  
#define MAX30102\_SPO2\_LEW\_PW 0  
  
#define MAX30102\_LED\_IR\_PA1 0x0c  
#define MAX30102\_LED\_RED\_PA2 0x0d  
  
#define MAX30102\_MULTI\_LED\_CTRL\_1 0x11  
#define MAX30102\_MULTI\_LED\_CTRL\_SLOT2 4  
#define MAX30102\_MULTI\_LED\_CTRL\_SLOT1 0  
#define MAX30102\_MULTI\_LED\_CTRL\_2 0x12  
#define MAX30102\_MULTI\_LED\_CTRL\_SLOT4 4  
#define MAX30102\_MULTI\_LED\_CTRL\_SLOT3 0  
  
#define MAX30102\_DIE\_TINT 0x1f  
#define MAX30102\_DIE\_TFRAC 0x20  
#define MAX30102\_DIE\_TFRAC\_INCREMENT 0.0625f  
#define MAX30102\_DIE\_TEMP\_CONFIG 0x21  
#define MAX30102\_DIE\_TEMP\_EN 1  
  
typedef enum max30102\_mode\_t  
{  
 max30102\_heart\_rate = 0x02,  
 max30102\_spo2 = 0x03,  
 max30102\_multi\_led = 0x07  
} max30102\_mode\_t;  
  
typedef enum max30102\_smp\_ave\_t  
{  
 max30102\_smp\_ave\_1,  
 max30102\_smp\_ave\_2,  
 max30102\_smp\_ave\_4,  
 max30102\_smp\_ave\_8,  
 max30102\_smp\_ave\_16,  
 max30102\_smp\_ave\_32,  
} max30102\_smp\_ave\_t;  
  
typedef enum max30102\_sr\_t  
{  
 max30102\_sr\_50,  
 max30102\_sr\_100,  
 max30102\_sr\_200,  
 max30102\_sr\_400,  
 max30102\_sr\_800,  
 max30102\_sr\_1000,  
 max30102\_sr\_1600,  
 max30102\_sr\_3200  
} max30102\_sr\_t;  
  
typedef enum max30102\_led\_pw\_t  
{  
 max30102\_pw\_15\_bit,  
 max30102\_pw\_16\_bit,  
 max30102\_pw\_17\_bit,  
 max30102\_pw\_18\_bit  
} max30102\_led\_pw\_t;  
  
typedef enum max30102\_adc\_t  
{  
 max30102\_adc\_2048,  
 max30102\_adc\_4096,  
 max30102\_adc\_8192,  
 max30102\_adc\_16384  
} max30102\_adc\_t;  
  
typedef enum max30102\_multi\_led\_ctrl\_t  
{  
 max30102\_led\_off,  
 max30102\_led\_red,  
 max30102\_led\_ir  
} max30102\_multi\_led\_ctrl\_t;  
  
typedef struct max30102\_t  
{  
 I2C\_HandleTypeDef \*\_ui2c;  
 uint32\_t \_ir\_samples[32];  
 uint32\_t \_red\_samples[32];  
 uint8\_t \_interrupt\_flag;  
} max30102\_t;  
  
void max30102\_plot(uint32\_t ir\_sample, uint32\_t red\_sample);  
  
void max30102\_init(max30102\_t \*obj, I2C\_HandleTypeDef \*hi2c);  
void max30102\_write(max30102\_t \*obj, uint8\_t reg, uint8\_t \*buf, uint16\_t buflen);  
void max30102\_read(max30102\_t \*obj, uint8\_t reg, uint8\_t \*buf, uint16\_t buflen);  
  
void max30102\_reset(max30102\_t \*obj);  
  
void max30102\_set\_a\_full(max30102\_t \*obj, uint8\_t enable);  
void max30102\_set\_ppg\_rdy(max30102\_t \*obj, uint8\_t enable);  
void max30102\_set\_alc\_ovf(max30102\_t \*obj, uint8\_t enable);  
void max30102\_set\_die\_temp\_rdy(max30102\_t \*obj, uint8\_t enable);  
void max30102\_set\_die\_temp\_en(max30102\_t \*obj, uint8\_t enable);  
  
void max30102\_on\_interrupt(max30102\_t \*obj);  
uint8\_t max30102\_has\_interrupt(max30102\_t \*obj);  
void max30102\_interrupt\_handler(max30102\_t \*obj);  
  
void max30102\_shutdown(max30102\_t \*obj, uint8\_t shdn);  
  
void max30102\_set\_mode(max30102\_t \*obj, max30102\_mode\_t mode);  
void max30102\_set\_sampling\_rate(max30102\_t \*obj, max30102\_sr\_t sr);  
  
void max30102\_set\_led\_pulse\_width(max30102\_t \*obj, max30102\_led\_pw\_t pw);  
void max30102\_set\_adc\_resolution(max30102\_t \*obj, max30102\_adc\_t adc);  
  
void max30102\_set\_led\_current\_1(max30102\_t \*obj, float ma);  
void max30102\_set\_led\_current\_2(max30102\_t \*obj, float ma);  
void max30102\_set\_multi\_led\_slot\_1\_2(max30102\_t \*obj, max30102\_multi\_led\_ctrl\_t slot1, max30102\_multi\_led\_ctrl\_t slot2);  
void max30102\_set\_multi\_led\_slot\_3\_4(max30102\_t \*obj, max30102\_multi\_led\_ctrl\_t slot3, max30102\_multi\_led\_ctrl\_t slot4);  
  
void max30102\_set\_fifo\_config(max30102\_t \*obj, max30102\_smp\_ave\_t smp\_ave, uint8\_t roll\_over\_en, uint8\_t fifo\_a\_full);  
void max30102\_clear\_fifo(max30102\_t \*obj);  
void max30102\_read\_fifo(max30102\_t \*obj);  
  
void max30102\_read\_temp(max30102\_t \*obj, int8\_t \*temp\_int, uint8\_t \*temp\_frac);  
  
#endif

然后编写头文件对应的 C 语言代码。

Core/Src/max30102\_for\_stm32\_hal.c

#include "max30102\_for\_stm32\_hal.h"  
#include <stdio.h>  
  
#ifdef \_\_cplusplus  
extern "C"  
{  
#endif  
  
/\*\*  
 \* @brief Built-in plotting function. Called during an interrupt to print/plot the current sample.  
 \* @note Override this in your main.c if you do not use printf() for printing.  
 \* @param ir\_sample  
 \* @param red\_sample  
 \*/  
\_\_weak void max30102\_plot(uint32\_t ir\_sample, uint32\_t red\_sample)  
{  
 UNUSED(ir\_sample);  
 UNUSED(red\_sample);  
}  
  
/\*\*  
 \* @brief MAX30102 initiation function.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param hi2c Pointer to I2C object handle  
 \*/  
void max30102\_init(max30102\_t \*obj, I2C\_HandleTypeDef \*hi2c)  
{  
 obj->\_ui2c = hi2c;  
 obj->\_interrupt\_flag = 0;  
 memset(obj->\_ir\_samples, 0, MAX30102\_SAMPLE\_LEN\_MAX \* sizeof(uint32\_t));  
 memset(obj->\_red\_samples, 0, MAX30102\_SAMPLE\_LEN\_MAX \* sizeof(uint32\_t));  
}  
  
/\*\*  
 \* @brief Write buffer of buflen bytes to a register of the MAX30102.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param reg Register address to write to.  
 \* @param buf Pointer containing the bytes to write.  
 \* @param buflen Number of bytes to write.  
 \*/  
void max30102\_write(max30102\_t \*obj, uint8\_t reg, uint8\_t \*buf, uint16\_t buflen)  
{  
 uint8\_t \*payload = (uint8\_t \*)malloc((buflen + 1) \* sizeof(uint8\_t));  
 \*payload = reg;  
 if (buf != NULL && buflen != 0)  
 memcpy(payload + 1, buf, buflen);  
 HAL\_I2C\_Master\_Transmit(obj->\_ui2c, MAX30102\_I2C\_ADDR << 1, payload, buflen + 1, MAX30102\_I2C\_TIMEOUT);  
 free(payload);  
}  
  
/\*\*  
 \* @brief Read buflen bytes from a register of the MAX30102 and store to buffer.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param reg Register address to read from.  
 \* @param buf Pointer to the array to write to.  
 \* @param buflen Number of bytes to read.  
 \*/  
void max30102\_read(max30102\_t \*obj, uint8\_t reg, uint8\_t \*buf, uint16\_t buflen)  
{  
 uint8\_t reg\_addr = reg;  
 HAL\_I2C\_Master\_Transmit(obj->\_ui2c, MAX30102\_I2C\_ADDR << 1, &reg\_addr, 1, MAX30102\_I2C\_TIMEOUT);  
 HAL\_I2C\_Master\_Receive(obj->\_ui2c, MAX30102\_I2C\_ADDR << 1, buf, buflen, MAX30102\_I2C\_TIMEOUT);  
}  
  
/\*\*  
 \* @brief Reset the sensor.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \*/  
void max30102\_reset(max30102\_t \*obj)  
{  
 uint8\_t val = 0x40;  
 max30102\_write(obj, MAX30102\_MODE\_CONFIG, &val, 1);  
}  
  
/\*\*  
 \* @brief Enable A\_FULL interrupt.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param enable Enable (1) or disable (0).  
 \*/  
void max30102\_set\_a\_full(max30102\_t \*obj, uint8\_t enable)  
{  
 uint8\_t reg = 0;  
 max30102\_read(obj, MAX30102\_INTERRUPT\_ENABLE\_1, &reg, 1);  
 reg &= ~(0x01 << MAX30102\_INTERRUPT\_A\_FULL);  
 reg |= ((enable & 0x01) << MAX30102\_INTERRUPT\_A\_FULL);  
 max30102\_write(obj, MAX30102\_INTERRUPT\_ENABLE\_1, &reg, 1);  
}  
  
/\*\*  
 \* @brief Enable PPG\_RDY interrupt.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param enable Enable (1) or disable (0).  
 \*/  
void max30102\_set\_ppg\_rdy(max30102\_t \*obj, uint8\_t enable)  
{  
 uint8\_t reg = 0;  
 max30102\_read(obj, MAX30102\_INTERRUPT\_ENABLE\_1, &reg, 1);  
 reg &= ~(0x01 << MAX30102\_INTERRUPT\_PPG\_RDY);  
 reg |= ((enable & 0x01) << MAX30102\_INTERRUPT\_PPG\_RDY);  
 max30102\_write(obj, MAX30102\_INTERRUPT\_ENABLE\_1, &reg, 1);  
}  
  
/\*\*  
 \* @brief Enable ALC\_OVF interrupt.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param enable Enable (1) or disable (0).  
 \*/  
void max30102\_set\_alc\_ovf(max30102\_t \*obj, uint8\_t enable)  
{  
 uint8\_t reg = 0;  
 max30102\_read(obj, MAX30102\_INTERRUPT\_ENABLE\_1, &reg, 1);  
 reg &= ~(0x01 << MAX30102\_INTERRUPT\_ALC\_OVF);  
 reg |= ((enable & 0x01) << MAX30102\_INTERRUPT\_ALC\_OVF);  
 max30102\_write(obj, MAX30102\_INTERRUPT\_ENABLE\_1, &reg, 1);  
}  
  
/\*\*  
 \* @brief Enable DIE\_TEMP\_RDY interrupt.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param enable Enable (1) or disable (0).  
 \*/  
void max30102\_set\_die\_temp\_rdy(max30102\_t \*obj, uint8\_t enable)  
{  
 uint8\_t reg = (enable & 0x01) << MAX30102\_INTERRUPT\_DIE\_TEMP\_RDY;  
 max30102\_write(obj, MAX30102\_INTERRUPT\_ENABLE\_2, &reg, 1);  
}  
  
/\*\*  
 \* @brief Enable temperature measurement.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param enable Enable (1) or disable (0).  
 \*/  
void max30102\_set\_die\_temp\_en(max30102\_t \*obj, uint8\_t enable)  
{  
 uint8\_t reg = (enable & 0x01) << MAX30102\_DIE\_TEMP\_EN;  
 max30102\_write(obj, MAX30102\_DIE\_TEMP\_CONFIG, &reg, 1);  
}  
  
/\*\*  
 \* @brief Set interrupt flag on interrupt. To be called in the corresponding external interrupt handler.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \*/  
void max30102\_on\_interrupt(max30102\_t \*obj)  
{  
 obj->\_interrupt\_flag = 1;  
}  
  
/\*\*  
 \* @brief Check whether the interrupt flag is active.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @return uint8\_t Active (1) or inactive (0).  
 \*/  
uint8\_t max30102\_has\_interrupt(max30102\_t \*obj)  
{  
 return obj->\_interrupt\_flag;  
}  
  
/\*\*  
 \* @brief Read interrupt status registers (0x00 and 0x01) and perform corresponding tasks.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \*/  
void max30102\_interrupt\_handler(max30102\_t \*obj)  
{  
 uint8\_t reg[2] = {0x00};  
 // Interrupt flag in registers 0x00 and 0x01 are cleared on read  
 max30102\_read(obj, MAX30102\_INTERRUPT\_STATUS\_1, reg, 2);  
  
 if ((reg[0] >> MAX30102\_INTERRUPT\_A\_FULL) & 0x01)  
 {  
 // FIFO almost full  
 max30102\_read\_fifo(obj);  
 }  
  
 if ((reg[0] >> MAX30102\_INTERRUPT\_PPG\_RDY) & 0x01)  
 {  
 // New FIFO data ready  
 }  
  
 if ((reg[0] >> MAX30102\_INTERRUPT\_ALC\_OVF) & 0x01)  
 {  
 // Ambient light overflow  
 }  
  
 if ((reg[1] >> MAX30102\_INTERRUPT\_DIE\_TEMP\_RDY) & 0x01)  
 {  
 // Temperature data ready  
 int8\_t temp\_int;  
 uint8\_t temp\_frac;  
 max30102\_read\_temp(obj, &temp\_int, &temp\_frac);  
 // float temp = temp\_int + 0.0625f \* temp\_frac;  
 }  
  
 // Reset interrupt flag  
 obj->\_interrupt\_flag = 0;  
}  
  
/\*\*  
 \* @brief Shutdown the sensor.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param shdn Shutdown bit.  
 \*/  
void max30102\_shutdown(max30102\_t \*obj, uint8\_t shdn)  
{  
 uint8\_t config;  
 max30102\_read(obj, MAX30102\_MODE\_CONFIG, &config, 1);  
 config = (config & 0x7f) | (shdn << MAX30102\_MODE\_SHDN);  
 max30102\_write(obj, MAX30102\_MODE\_CONFIG, &config, 1);  
}  
  
/\*\*  
 \* @brief Set measurement mode.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param mode Measurement mode enum (max30102\_mode\_t).  
 \*/  
void max30102\_set\_mode(max30102\_t \*obj, max30102\_mode\_t mode)  
{  
 uint8\_t config;  
 max30102\_read(obj, MAX30102\_MODE\_CONFIG, &config, 1);  
 config = (config & 0xf8) | mode;  
 max30102\_write(obj, MAX30102\_MODE\_CONFIG, &config, 1);  
 max30102\_clear\_fifo(obj);  
}  
  
/\*\*  
 \* @brief Set sampling rate.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param sr Sampling rate enum (max30102\_spo2\_st\_t).  
 \*/  
void max30102\_set\_sampling\_rate(max30102\_t \*obj, max30102\_sr\_t sr)  
{  
 uint8\_t config;  
 max30102\_read(obj, MAX30102\_SPO2\_CONFIG, &config, 1);  
 config = (config & 0x63) << MAX30102\_SPO2\_SR;  
 max30102\_write(obj, MAX30102\_SPO2\_CONFIG, &config, 1);  
}  
  
/\*\*  
 \* @brief Set led pulse width.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param pw Pulse width enum (max30102\_led\_pw\_t).  
 \*/  
void max30102\_set\_led\_pulse\_width(max30102\_t \*obj, max30102\_led\_pw\_t pw)  
{  
 uint8\_t config;  
 max30102\_read(obj, MAX30102\_SPO2\_CONFIG, &config, 1);  
 config = (config & 0x7c) | (pw << MAX30102\_SPO2\_LEW\_PW);  
 max30102\_write(obj, MAX30102\_SPO2\_CONFIG, &config, 1);  
}  
  
/\*\*  
 \* @brief Set ADC resolution.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param adc ADC resolution enum (max30102\_adc\_t).  
 \*/  
void max30102\_set\_adc\_resolution(max30102\_t \*obj, max30102\_adc\_t adc)  
{  
 uint8\_t config;  
 max30102\_read(obj, MAX30102\_SPO2\_CONFIG, &config, 1);  
 config = (config & 0x1f) | (adc << MAX30102\_SPO2\_ADC\_RGE);  
 max30102\_write(obj, MAX30102\_SPO2\_CONFIG, &config, 1);  
}  
  
/\*\*  
 \* @brief Set LED current.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param ma LED current float (0 < ma < 51.0).  
 \*/  
void max30102\_set\_led\_current\_1(max30102\_t \*obj, float ma)  
{  
 uint8\_t pa = ma / 0.2;  
 max30102\_write(obj, MAX30102\_LED\_IR\_PA1, &pa, 1);  
}  
  
/\*\*  
 \* @brief Set LED current.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param ma LED current float (0 < ma < 51.0).  
 \*/  
void max30102\_set\_led\_current\_2(max30102\_t \*obj, float ma)  
{  
 uint8\_t pa = ma / 0.2;  
 max30102\_write(obj, MAX30102\_LED\_RED\_PA2, &pa, 1);  
}  
  
/\*\*  
 \* @brief Set slot mode when in multi-LED mode.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param slot1 Slot 1 mode enum (max30102\_multi\_led\_ctrl\_t).  
 \* @param slot2 Slot 2 mode enum (max30102\_multi\_led\_ctrl\_t).  
 \*/  
void max30102\_set\_multi\_led\_slot\_1\_2(max30102\_t \*obj, max30102\_multi\_led\_ctrl\_t slot1, max30102\_multi\_led\_ctrl\_t slot2)  
{  
 uint8\_t val = 0;  
 val |= ((slot1 << MAX30102\_MULTI\_LED\_CTRL\_SLOT1) | (slot2 << MAX30102\_MULTI\_LED\_CTRL\_SLOT2));  
 max30102\_write(obj, MAX30102\_MULTI\_LED\_CTRL\_1, &val, 1);  
}  
  
/\*\*  
 \* @brief Set slot mode when in multi-LED mode.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param slot1 Slot 1 mode enum (max30102\_multi\_led\_ctrl\_t).  
 \* @param slot2 Slot 2 mode enum (max30102\_multi\_led\_ctrl\_t).  
 \*/  
void max30102\_set\_multi\_led\_slot\_3\_4(max30102\_t \*obj, max30102\_multi\_led\_ctrl\_t slot3, max30102\_multi\_led\_ctrl\_t slot4)  
{  
 uint8\_t val = 0;  
 val |= ((slot3 << MAX30102\_MULTI\_LED\_CTRL\_SLOT3) | (slot4 << MAX30102\_MULTI\_LED\_CTRL\_SLOT4));  
 max30102\_write(obj, MAX30102\_MULTI\_LED\_CTRL\_2, &val, 1);  
}  
  
/\*\*  
 \* @brief  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param smp\_ave  
 \* @param roll\_over\_en Roll over enabled(1) or disabled(0).  
 \* @param fifo\_a\_full Number of empty samples when A\_FULL interrupt issued (0 < fifo\_a\_full < 15).  
 \*/  
void max30102\_set\_fifo\_config(max30102\_t \*obj, max30102\_smp\_ave\_t smp\_ave, uint8\_t roll\_over\_en, uint8\_t fifo\_a\_full)  
{  
 uint8\_t config = 0x00;  
 config |= smp\_ave << MAX30102\_FIFO\_CONFIG\_SMP\_AVE;  
 config |= ((roll\_over\_en & 0x01) << MAX30102\_FIFO\_CONFIG\_ROLL\_OVER\_EN);  
 config |= ((fifo\_a\_full & 0x0f) << MAX30102\_FIFO\_CONFIG\_FIFO\_A\_FULL);  
 max30102\_write(obj, MAX30102\_FIFO\_CONFIG, &config, 1);  
}  
  
/\*\*  
 \* @brief Clear all FIFO pointers in the sensor.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \*/  
void max30102\_clear\_fifo(max30102\_t \*obj)  
{  
 uint8\_t val = 0x00;  
 max30102\_write(obj, MAX30102\_FIFO\_WR\_PTR, &val, 3);  
 max30102\_write(obj, MAX30102\_FIFO\_RD\_PTR, &val, 3);  
 max30102\_write(obj, MAX30102\_OVF\_COUNTER, &val, 3);  
}  
  
/\*\*  
 \* @brief Read FIFO content and store to buffer in max30102\_t object instance.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \*/  
void max30102\_read\_fifo(max30102\_t \*obj)  
{  
 // First transaction: Get the FIFO\_WR\_PTR  
 uint8\_t wr\_ptr = 0, rd\_ptr = 0;  
 max30102\_read(obj, MAX30102\_FIFO\_WR\_PTR, &wr\_ptr, 1);  
 max30102\_read(obj, MAX30102\_FIFO\_RD\_PTR, &rd\_ptr, 1);  
  
 int8\_t num\_samples;  
  
 num\_samples = (int8\_t)wr\_ptr - (int8\_t)rd\_ptr;  
 if (num\_samples < 1)  
 {  
 num\_samples += 32;  
 }  
  
 // Second transaction: Read NUM\_SAMPLES\_TO\_READ samples from the FIFO  
 for (int8\_t i = 0; i < num\_samples; i++)  
 {  
 uint8\_t sample[6];  
 max30102\_read(obj, MAX30102\_FIFO\_DATA, sample, 6);  
 uint32\_t ir\_sample = ((uint32\_t)(sample[0] << 16) | (uint32\_t)(sample[1] << 8) | (uint32\_t)(sample[2])) & 0x3ffff;  
 uint32\_t red\_sample = ((uint32\_t)(sample[3] << 16) | (uint32\_t)(sample[4] << 8) | (uint32\_t)(sample[5])) & 0x3ffff;  
 obj->\_ir\_samples[i] = ir\_sample;  
 obj->\_red\_samples[i] = red\_sample;  
 max30102\_plot(ir\_sample, red\_sample);  
 }  
}  
  
/\*\*  
 \* @brief Read die temperature.  
 \*  
 \* @param obj Pointer to max30102\_t object instance.  
 \* @param temp\_int Pointer to store the integer part of temperature. Stored in 2's complement format.  
 \* @param temp\_frac Pointer to store the fractional part of temperature. Increments of 0.0625 deg C.  
 \*/  
  
void max30102\_read\_temp(max30102\_t \*obj, int8\_t \*temp\_int, uint8\_t \*temp\_frac)  
{  
 max30102\_read(obj, MAX30102\_DIE\_TINT, (uint8\_t \*)temp\_int, 1);  
 max30102\_read(obj, MAX30102\_DIE\_TFRAC, temp\_frac, 1);  
}  
  
#ifdef \_\_cplusplus  
}  
#endif

这样，驱动MAX30102的代码就已经写完了。

* 1. OLED驱动的编写

首先编写所需的字体文件，我们这里提供一份 ascii 编码也就是英文字母的字体文件。

Core/Inc/oledfont.h

#ifndef \_\_OLEDFONT\_H  
#define \_\_OLEDFONT\_H  
  
/\* 常用ASCII表  
 \* 偏移量32  
 \* ASCII字符集: !"#$%&'()\*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^\_`abcdefghijklmnopqrstuvwxyz{|}~  
 \* PC2LCD2002取模方式设置：阴码+逐列式+顺向+C51格式  
 \* 总共：3个字符集（12\*12、16\*16和24\*24），用户可以自行新增其他分辨率的字符集。  
 \* 每个字符所占用的字节数为:(size/8+((size%8)?1:0))\*(size/2),其中size:是字库生成时的点阵大小(12/16/24...)  
 \*/  
  
/\* 12\*12 ASCII字符集点阵 \*/  
const unsigned char oled\_asc2\_1206[95][12] = {  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*" ",0\*/  
 {0x00, 0x00, 0x00, 0x00, 0x3F, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"!",1\*/  
 {0x00, 0x00, 0x30, 0x00, 0x40, 0x00, 0x30, 0x00, 0x40, 0x00, 0x00, 0x00}, /\*""",2\*/  
 {0x09, 0x00, 0x0B, 0xC0, 0x3D, 0x00, 0x0B, 0xC0, 0x3D, 0x00, 0x09, 0x00}, /\*"#",3\*/  
 {0x18, 0xC0, 0x24, 0x40, 0x7F, 0xE0, 0x22, 0x40, 0x31, 0x80, 0x00, 0x00}, /\*"$",4\*/  
 {0x18, 0x00, 0x24, 0xC0, 0x1B, 0x00, 0x0D, 0x80, 0x32, 0x40, 0x01, 0x80}, /\*"%",5\*/  
 {0x03, 0x80, 0x1C, 0x40, 0x27, 0x40, 0x1C, 0x80, 0x07, 0x40, 0x00, 0x40}, /\*"&",6\*/  
 {0x10, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"'",7\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x80, 0x20, 0x40, 0x40, 0x20}, /\*"(",8\*/  
 {0x00, 0x00, 0x40, 0x20, 0x20, 0x40, 0x1F, 0x80, 0x00, 0x00, 0x00, 0x00}, /\*")",9\*/  
 {0x09, 0x00, 0x06, 0x00, 0x1F, 0x80, 0x06, 0x00, 0x09, 0x00, 0x00, 0x00}, /\*"\*",10\*/  
 {0x04, 0x00, 0x04, 0x00, 0x3F, 0x80, 0x04, 0x00, 0x04, 0x00, 0x00, 0x00}, /\*"+",11\*/  
 {0x00, 0x10, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*",",12\*/  
 {0x04, 0x00, 0x04, 0x00, 0x04, 0x00, 0x04, 0x00, 0x04, 0x00, 0x00, 0x00}, /\*"-",13\*/  
 {0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*".",14\*/  
 {0x00, 0x20, 0x01, 0xC0, 0x06, 0x00, 0x38, 0x00, 0x40, 0x00, 0x00, 0x00}, /\*"/",15\*/  
 {0x1F, 0x80, 0x20, 0x40, 0x20, 0x40, 0x20, 0x40, 0x1F, 0x80, 0x00, 0x00}, /\*"0",16\*/  
 {0x00, 0x00, 0x10, 0x40, 0x3F, 0xC0, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00}, /\*"1",17\*/  
 {0x18, 0xC0, 0x21, 0x40, 0x22, 0x40, 0x24, 0x40, 0x18, 0x40, 0x00, 0x00}, /\*"2",18\*/  
 {0x10, 0x80, 0x20, 0x40, 0x24, 0x40, 0x24, 0x40, 0x1B, 0x80, 0x00, 0x00}, /\*"3",19\*/  
 {0x02, 0x00, 0x0D, 0x00, 0x11, 0x00, 0x3F, 0xC0, 0x01, 0x40, 0x00, 0x00}, /\*"4",20\*/  
 {0x3C, 0x80, 0x24, 0x40, 0x24, 0x40, 0x24, 0x40, 0x23, 0x80, 0x00, 0x00}, /\*"5",21\*/  
 {0x1F, 0x80, 0x24, 0x40, 0x24, 0x40, 0x34, 0x40, 0x03, 0x80, 0x00, 0x00}, /\*"6",22\*/  
 {0x30, 0x00, 0x20, 0x00, 0x27, 0xC0, 0x38, 0x00, 0x20, 0x00, 0x00, 0x00}, /\*"7",23\*/  
 {0x1B, 0x80, 0x24, 0x40, 0x24, 0x40, 0x24, 0x40, 0x1B, 0x80, 0x00, 0x00}, /\*"8",24\*/  
 {0x1C, 0x00, 0x22, 0xC0, 0x22, 0x40, 0x22, 0x40, 0x1F, 0x80, 0x00, 0x00}, /\*"9",25\*/  
 {0x00, 0x00, 0x00, 0x00, 0x08, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*":",26\*/  
 {0x00, 0x00, 0x00, 0x00, 0x04, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*";",27\*/  
 {0x00, 0x00, 0x04, 0x00, 0x0A, 0x00, 0x11, 0x00, 0x20, 0x80, 0x40, 0x40}, /\*"<",28\*/  
 {0x09, 0x00, 0x09, 0x00, 0x09, 0x00, 0x09, 0x00, 0x09, 0x00, 0x00, 0x00}, /\*"=",29\*/  
 {0x00, 0x00, 0x40, 0x40, 0x20, 0x80, 0x11, 0x00, 0x0A, 0x00, 0x04, 0x00}, /\*">",30\*/  
 {0x18, 0x00, 0x20, 0x00, 0x23, 0x40, 0x24, 0x00, 0x18, 0x00, 0x00, 0x00}, /\*"?",31\*/  
 {0x1F, 0x80, 0x20, 0x40, 0x27, 0x40, 0x29, 0x40, 0x1F, 0x40, 0x00, 0x00}, /\*"@",32\*/  
 {0x00, 0x40, 0x07, 0xC0, 0x39, 0x00, 0x0F, 0x00, 0x01, 0xC0, 0x00, 0x40}, /\*"A",33\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x24, 0x40, 0x24, 0x40, 0x1B, 0x80, 0x00, 0x00}, /\*"B",34\*/  
 {0x1F, 0x80, 0x20, 0x40, 0x20, 0x40, 0x20, 0x40, 0x30, 0x80, 0x00, 0x00}, /\*"C",35\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x20, 0x40, 0x20, 0x40, 0x1F, 0x80, 0x00, 0x00}, /\*"D",36\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x24, 0x40, 0x2E, 0x40, 0x30, 0xC0, 0x00, 0x00}, /\*"E",37\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x24, 0x40, 0x2E, 0x00, 0x30, 0x00, 0x00, 0x00}, /\*"F",38\*/  
 {0x0F, 0x00, 0x10, 0x80, 0x20, 0x40, 0x22, 0x40, 0x33, 0x80, 0x02, 0x00}, /\*"G",39\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x04, 0x00, 0x04, 0x00, 0x3F, 0xC0, 0x20, 0x40}, /\*"H",40\*/  
 {0x20, 0x40, 0x20, 0x40, 0x3F, 0xC0, 0x20, 0x40, 0x20, 0x40, 0x00, 0x00}, /\*"I",41\*/  
 {0x00, 0x60, 0x20, 0x20, 0x20, 0x20, 0x3F, 0xC0, 0x20, 0x00, 0x20, 0x00}, /\*"J",42\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x24, 0x40, 0x0B, 0x00, 0x30, 0xC0, 0x20, 0x40}, /\*"K",43\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x20, 0x40, 0x00, 0x40, 0x00, 0x40, 0x00, 0xC0}, /\*"L",44\*/  
 {0x3F, 0xC0, 0x3C, 0x00, 0x03, 0xC0, 0x3C, 0x00, 0x3F, 0xC0, 0x00, 0x00}, /\*"M",45\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x0C, 0x40, 0x23, 0x00, 0x3F, 0xC0, 0x20, 0x00}, /\*"N",46\*/  
 {0x1F, 0x80, 0x20, 0x40, 0x20, 0x40, 0x20, 0x40, 0x1F, 0x80, 0x00, 0x00}, /\*"O",47\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x24, 0x40, 0x24, 0x00, 0x18, 0x00, 0x00, 0x00}, /\*"P",48\*/  
 {0x1F, 0x80, 0x21, 0x40, 0x21, 0x40, 0x20, 0xE0, 0x1F, 0xA0, 0x00, 0x00}, /\*"Q",49\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x24, 0x40, 0x26, 0x00, 0x19, 0xC0, 0x00, 0x40}, /\*"R",50\*/  
 {0x18, 0xC0, 0x24, 0x40, 0x24, 0x40, 0x22, 0x40, 0x31, 0x80, 0x00, 0x00}, /\*"S",51\*/  
 {0x30, 0x00, 0x20, 0x40, 0x3F, 0xC0, 0x20, 0x40, 0x30, 0x00, 0x00, 0x00}, /\*"T",52\*/  
 {0x20, 0x00, 0x3F, 0x80, 0x00, 0x40, 0x00, 0x40, 0x3F, 0x80, 0x20, 0x00}, /\*"U",53\*/  
 {0x20, 0x00, 0x3E, 0x00, 0x01, 0xC0, 0x07, 0x00, 0x38, 0x00, 0x20, 0x00}, /\*"V",54\*/  
 {0x38, 0x00, 0x07, 0xC0, 0x3C, 0x00, 0x07, 0xC0, 0x38, 0x00, 0x00, 0x00}, /\*"W",55\*/  
 {0x20, 0x40, 0x39, 0xC0, 0x06, 0x00, 0x39, 0xC0, 0x20, 0x40, 0x00, 0x00}, /\*"X",56\*/  
 {0x20, 0x00, 0x38, 0x40, 0x07, 0xC0, 0x38, 0x40, 0x20, 0x00, 0x00, 0x00}, /\*"Y",57\*/  
 {0x30, 0x40, 0x21, 0xC0, 0x26, 0x40, 0x38, 0x40, 0x20, 0xC0, 0x00, 0x00}, /\*"Z",58\*/  
 {0x00, 0x00, 0x00, 0x00, 0x7F, 0xE0, 0x40, 0x20, 0x40, 0x20, 0x00, 0x00}, /\*"[",59\*/  
 {0x00, 0x00, 0x70, 0x00, 0x0C, 0x00, 0x03, 0x80, 0x00, 0x40, 0x00, 0x00}, /\*"\",60\*/  
 {0x00, 0x00, 0x40, 0x20, 0x40, 0x20, 0x7F, 0xE0, 0x00, 0x00, 0x00, 0x00}, /\*"]",61\*/  
 {0x00, 0x00, 0x20, 0x00, 0x40, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"^",62\*/  
 {0x00, 0x10, 0x00, 0x10, 0x00, 0x10, 0x00, 0x10, 0x00, 0x10, 0x00, 0x10}, /\*"\_",63\*/  
 {0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"`",64\*/  
 {0x00, 0x00, 0x02, 0x80, 0x05, 0x40, 0x05, 0x40, 0x03, 0xC0, 0x00, 0x40}, /\*"a",65\*/  
 {0x20, 0x00, 0x3F, 0xC0, 0x04, 0x40, 0x04, 0x40, 0x03, 0x80, 0x00, 0x00}, /\*"b",66\*/  
 {0x00, 0x00, 0x03, 0x80, 0x04, 0x40, 0x04, 0x40, 0x06, 0x40, 0x00, 0x00}, /\*"c",67\*/  
 {0x00, 0x00, 0x03, 0x80, 0x04, 0x40, 0x24, 0x40, 0x3F, 0xC0, 0x00, 0x40}, /\*"d",68\*/  
 {0x00, 0x00, 0x03, 0x80, 0x05, 0x40, 0x05, 0x40, 0x03, 0x40, 0x00, 0x00}, /\*"e",69\*/  
 {0x00, 0x00, 0x04, 0x40, 0x1F, 0xC0, 0x24, 0x40, 0x24, 0x40, 0x20, 0x00}, /\*"f",70\*/  
 {0x00, 0x00, 0x02, 0xE0, 0x05, 0x50, 0x05, 0x50, 0x06, 0x50, 0x04, 0x20}, /\*"g",71\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x04, 0x40, 0x04, 0x00, 0x03, 0xC0, 0x00, 0x40}, /\*"h",72\*/  
 {0x00, 0x00, 0x04, 0x40, 0x27, 0xC0, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00}, /\*"i",73\*/  
 {0x00, 0x10, 0x00, 0x10, 0x04, 0x10, 0x27, 0xE0, 0x00, 0x00, 0x00, 0x00}, /\*"j",74\*/  
 {0x20, 0x40, 0x3F, 0xC0, 0x01, 0x40, 0x07, 0x00, 0x04, 0xC0, 0x04, 0x40}, /\*"k",75\*/  
 {0x20, 0x40, 0x20, 0x40, 0x3F, 0xC0, 0x00, 0x40, 0x00, 0x40, 0x00, 0x00}, /\*"l",76\*/  
 {0x07, 0xC0, 0x04, 0x00, 0x07, 0xC0, 0x04, 0x00, 0x03, 0xC0, 0x00, 0x00}, /\*"m",77\*/  
 {0x04, 0x40, 0x07, 0xC0, 0x04, 0x40, 0x04, 0x00, 0x03, 0xC0, 0x00, 0x40}, /\*"n",78\*/  
 {0x00, 0x00, 0x03, 0x80, 0x04, 0x40, 0x04, 0x40, 0x03, 0x80, 0x00, 0x00}, /\*"o",79\*/  
 {0x04, 0x10, 0x07, 0xF0, 0x04, 0x50, 0x04, 0x40, 0x03, 0x80, 0x00, 0x00}, /\*"p",80\*/  
 {0x00, 0x00, 0x03, 0x80, 0x04, 0x40, 0x04, 0x50, 0x07, 0xF0, 0x00, 0x10}, /\*"q",81\*/  
 {0x04, 0x40, 0x07, 0xC0, 0x02, 0x40, 0x04, 0x00, 0x04, 0x00, 0x00, 0x00}, /\*"r",82\*/  
 {0x00, 0x00, 0x06, 0x40, 0x05, 0x40, 0x05, 0x40, 0x04, 0xC0, 0x00, 0x00}, /\*"s",83\*/  
 {0x00, 0x00, 0x04, 0x00, 0x1F, 0x80, 0x04, 0x40, 0x00, 0x40, 0x00, 0x00}, /\*"t",84\*/  
 {0x04, 0x00, 0x07, 0x80, 0x00, 0x40, 0x04, 0x40, 0x07, 0xC0, 0x00, 0x40}, /\*"u",85\*/  
 {0x04, 0x00, 0x07, 0x00, 0x04, 0xC0, 0x01, 0x80, 0x06, 0x00, 0x04, 0x00}, /\*"v",86\*/  
 {0x06, 0x00, 0x01, 0xC0, 0x07, 0x00, 0x01, 0xC0, 0x06, 0x00, 0x00, 0x00}, /\*"w",87\*/  
 {0x04, 0x40, 0x06, 0xC0, 0x01, 0x00, 0x06, 0xC0, 0x04, 0x40, 0x00, 0x00}, /\*"x",88\*/  
 {0x04, 0x10, 0x07, 0x10, 0x04, 0xE0, 0x01, 0x80, 0x06, 0x00, 0x04, 0x00}, /\*"y",89\*/  
 {0x00, 0x00, 0x04, 0x40, 0x05, 0xC0, 0x06, 0x40, 0x04, 0x40, 0x00, 0x00}, /\*"z",90\*/  
 {0x00, 0x00, 0x00, 0x00, 0x04, 0x00, 0x7B, 0xE0, 0x40, 0x20, 0x00, 0x00}, /\*"{",91\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x00}, /\*"|",92\*/  
 {0x00, 0x00, 0x40, 0x20, 0x7B, 0xE0, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"}",93\*/  
 {0x40, 0x00, 0x80, 0x00, 0x40, 0x00, 0x20, 0x00, 0x20, 0x00, 0x40, 0x00}, /\*"~",94\*/  
};  
  
/\* 16\*16 ASCII字符集点阵 \*/  
const unsigned char oled\_asc2\_1608[95][16] = {  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*" ",0\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0xCC, 0x00, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"!",1\*/  
 {0x00, 0x00, 0x08, 0x00, 0x30, 0x00, 0x60, 0x00, 0x08, 0x00, 0x30, 0x00, 0x60, 0x00, 0x00, 0x00}, /\*""",2\*/  
 {0x02, 0x20, 0x03, 0xFC, 0x1E, 0x20, 0x02, 0x20, 0x03, 0xFC, 0x1E, 0x20, 0x02, 0x20, 0x00, 0x00}, /\*"#",3\*/  
 {0x00, 0x00, 0x0E, 0x18, 0x11, 0x04, 0x3F, 0xFF, 0x10, 0x84, 0x0C, 0x78, 0x00, 0x00, 0x00, 0x00}, /\*"$",4\*/  
 {0x0F, 0x00, 0x10, 0x84, 0x0F, 0x38, 0x00, 0xC0, 0x07, 0x78, 0x18, 0x84, 0x00, 0x78, 0x00, 0x00}, /\*"%",5\*/  
 {0x00, 0x78, 0x0F, 0x84, 0x10, 0xC4, 0x11, 0x24, 0x0E, 0x98, 0x00, 0xE4, 0x00, 0x84, 0x00, 0x08}, /\*"&",6\*/  
 {0x08, 0x00, 0x68, 0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"'",7\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xE0, 0x18, 0x18, 0x20, 0x04, 0x40, 0x02, 0x00, 0x00}, /\*"(",8\*/  
 {0x00, 0x00, 0x40, 0x02, 0x20, 0x04, 0x18, 0x18, 0x07, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*")",9\*/  
 {0x02, 0x40, 0x02, 0x40, 0x01, 0x80, 0x0F, 0xF0, 0x01, 0x80, 0x02, 0x40, 0x02, 0x40, 0x00, 0x00}, /\*"\*",10\*/  
 {0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x0F, 0xF8, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x00, 0x00}, /\*"+",11\*/  
 {0x00, 0x01, 0x00, 0x0D, 0x00, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*",",12\*/  
 {0x00, 0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80, 0x00, 0x80}, /\*"-",13\*/  
 {0x00, 0x00, 0x00, 0x0C, 0x00, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*".",14\*/  
 {0x00, 0x00, 0x00, 0x06, 0x00, 0x18, 0x00, 0x60, 0x01, 0x80, 0x06, 0x00, 0x18, 0x00, 0x20, 0x00}, /\*"/",15\*/  
 {0x00, 0x00, 0x07, 0xF0, 0x08, 0x08, 0x10, 0x04, 0x10, 0x04, 0x08, 0x08, 0x07, 0xF0, 0x00, 0x00}, /\*"0",16\*/  
 {0x00, 0x00, 0x08, 0x04, 0x08, 0x04, 0x1F, 0xFC, 0x00, 0x04, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00}, /\*"1",17\*/  
 {0x00, 0x00, 0x0E, 0x0C, 0x10, 0x14, 0x10, 0x24, 0x10, 0x44, 0x11, 0x84, 0x0E, 0x0C, 0x00, 0x00}, /\*"2",18\*/  
 {0x00, 0x00, 0x0C, 0x18, 0x10, 0x04, 0x11, 0x04, 0x11, 0x04, 0x12, 0x88, 0x0C, 0x70, 0x00, 0x00}, /\*"3",19\*/  
 {0x00, 0x00, 0x00, 0xE0, 0x03, 0x20, 0x04, 0x24, 0x08, 0x24, 0x1F, 0xFC, 0x00, 0x24, 0x00, 0x00}, /\*"4",20\*/  
 {0x00, 0x00, 0x1F, 0x98, 0x10, 0x84, 0x11, 0x04, 0x11, 0x04, 0x10, 0x88, 0x10, 0x70, 0x00, 0x00}, /\*"5",21\*/  
 {0x00, 0x00, 0x07, 0xF0, 0x08, 0x88, 0x11, 0x04, 0x11, 0x04, 0x18, 0x88, 0x00, 0x70, 0x00, 0x00}, /\*"6",22\*/  
 {0x00, 0x00, 0x1C, 0x00, 0x10, 0x00, 0x10, 0xFC, 0x13, 0x00, 0x1C, 0x00, 0x10, 0x00, 0x00, 0x00}, /\*"7",23\*/  
 {0x00, 0x00, 0x0E, 0x38, 0x11, 0x44, 0x10, 0x84, 0x10, 0x84, 0x11, 0x44, 0x0E, 0x38, 0x00, 0x00}, /\*"8",24\*/  
 {0x00, 0x00, 0x07, 0x00, 0x08, 0x8C, 0x10, 0x44, 0x10, 0x44, 0x08, 0x88, 0x07, 0xF0, 0x00, 0x00}, /\*"9",25\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x0C, 0x03, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*":",26\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*";",27\*/  
 {0x00, 0x00, 0x00, 0x80, 0x01, 0x40, 0x02, 0x20, 0x04, 0x10, 0x08, 0x08, 0x10, 0x04, 0x00, 0x00}, /\*"<",28\*/  
 {0x02, 0x20, 0x02, 0x20, 0x02, 0x20, 0x02, 0x20, 0x02, 0x20, 0x02, 0x20, 0x02, 0x20, 0x00, 0x00}, /\*"=",29\*/  
 {0x00, 0x00, 0x10, 0x04, 0x08, 0x08, 0x04, 0x10, 0x02, 0x20, 0x01, 0x40, 0x00, 0x80, 0x00, 0x00}, /\*">",30\*/  
 {0x00, 0x00, 0x0E, 0x00, 0x12, 0x00, 0x10, 0x0C, 0x10, 0x6C, 0x10, 0x80, 0x0F, 0x00, 0x00, 0x00}, /\*"?",31\*/  
 {0x03, 0xE0, 0x0C, 0x18, 0x13, 0xE4, 0x14, 0x24, 0x17, 0xC4, 0x08, 0x28, 0x07, 0xD0, 0x00, 0x00}, /\*"@",32\*/  
 {0x00, 0x04, 0x00, 0x3C, 0x03, 0xC4, 0x1C, 0x40, 0x07, 0x40, 0x00, 0xE4, 0x00, 0x1C, 0x00, 0x04}, /\*"A",33\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x11, 0x04, 0x11, 0x04, 0x11, 0x04, 0x0E, 0x88, 0x00, 0x70, 0x00, 0x00}, /\*"B",34\*/  
 {0x03, 0xE0, 0x0C, 0x18, 0x10, 0x04, 0x10, 0x04, 0x10, 0x04, 0x10, 0x08, 0x1C, 0x10, 0x00, 0x00}, /\*"C",35\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x10, 0x04, 0x10, 0x04, 0x10, 0x04, 0x08, 0x08, 0x07, 0xF0, 0x00, 0x00}, /\*"D",36\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x11, 0x04, 0x11, 0x04, 0x17, 0xC4, 0x10, 0x04, 0x08, 0x18, 0x00, 0x00}, /\*"E",37\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x11, 0x04, 0x11, 0x00, 0x17, 0xC0, 0x10, 0x00, 0x08, 0x00, 0x00, 0x00}, /\*"F",38\*/  
 {0x03, 0xE0, 0x0C, 0x18, 0x10, 0x04, 0x10, 0x04, 0x10, 0x44, 0x1C, 0x78, 0x00, 0x40, 0x00, 0x00}, /\*"G",39\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x10, 0x84, 0x00, 0x80, 0x00, 0x80, 0x10, 0x84, 0x1F, 0xFC, 0x10, 0x04}, /\*"H",40\*/  
 {0x00, 0x00, 0x10, 0x04, 0x10, 0x04, 0x1F, 0xFC, 0x10, 0x04, 0x10, 0x04, 0x00, 0x00, 0x00, 0x00}, /\*"I",41\*/  
 {0x00, 0x03, 0x00, 0x01, 0x10, 0x01, 0x10, 0x01, 0x1F, 0xFE, 0x10, 0x00, 0x10, 0x00, 0x00, 0x00}, /\*"J",42\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x11, 0x04, 0x03, 0x80, 0x14, 0x64, 0x18, 0x1C, 0x10, 0x04, 0x00, 0x00}, /\*"K",43\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x10, 0x04, 0x00, 0x04, 0x00, 0x04, 0x00, 0x04, 0x00, 0x0C, 0x00, 0x00}, /\*"L",44\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x1F, 0x00, 0x00, 0xFC, 0x1F, 0x00, 0x1F, 0xFC, 0x10, 0x04, 0x00, 0x00}, /\*"M",45\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x0C, 0x04, 0x03, 0x00, 0x00, 0xE0, 0x10, 0x18, 0x1F, 0xFC, 0x10, 0x00}, /\*"N",46\*/  
 {0x07, 0xF0, 0x08, 0x08, 0x10, 0x04, 0x10, 0x04, 0x10, 0x04, 0x08, 0x08, 0x07, 0xF0, 0x00, 0x00}, /\*"O",47\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x10, 0x84, 0x10, 0x80, 0x10, 0x80, 0x10, 0x80, 0x0F, 0x00, 0x00, 0x00}, /\*"P",48\*/  
 {0x07, 0xF0, 0x08, 0x18, 0x10, 0x24, 0x10, 0x24, 0x10, 0x1C, 0x08, 0x0A, 0x07, 0xF2, 0x00, 0x00}, /\*"Q",49\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x11, 0x04, 0x11, 0x00, 0x11, 0xC0, 0x11, 0x30, 0x0E, 0x0C, 0x00, 0x04}, /\*"R",50\*/  
 {0x00, 0x00, 0x0E, 0x1C, 0x11, 0x04, 0x10, 0x84, 0x10, 0x84, 0x10, 0x44, 0x1C, 0x38, 0x00, 0x00}, /\*"S",51\*/  
 {0x18, 0x00, 0x10, 0x00, 0x10, 0x04, 0x1F, 0xFC, 0x10, 0x04, 0x10, 0x00, 0x18, 0x00, 0x00, 0x00}, /\*"T",52\*/  
 {0x10, 0x00, 0x1F, 0xF8, 0x10, 0x04, 0x00, 0x04, 0x00, 0x04, 0x10, 0x04, 0x1F, 0xF8, 0x10, 0x00}, /\*"U",53\*/  
 {0x10, 0x00, 0x1E, 0x00, 0x11, 0xE0, 0x00, 0x1C, 0x00, 0x70, 0x13, 0x80, 0x1C, 0x00, 0x10, 0x00}, /\*"V",54\*/  
 {0x1F, 0xC0, 0x10, 0x3C, 0x00, 0xE0, 0x1F, 0x00, 0x00, 0xE0, 0x10, 0x3C, 0x1F, 0xC0, 0x00, 0x00}, /\*"W",55\*/  
 {0x10, 0x04, 0x18, 0x0C, 0x16, 0x34, 0x01, 0xC0, 0x01, 0xC0, 0x16, 0x34, 0x18, 0x0C, 0x10, 0x04}, /\*"X",56\*/  
 {0x10, 0x00, 0x1C, 0x00, 0x13, 0x04, 0x00, 0xFC, 0x13, 0x04, 0x1C, 0x00, 0x10, 0x00, 0x00, 0x00}, /\*"Y",57\*/  
 {0x08, 0x04, 0x10, 0x1C, 0x10, 0x64, 0x10, 0x84, 0x13, 0x04, 0x1C, 0x04, 0x10, 0x18, 0x00, 0x00}, /\*"Z",58\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7F, 0xFE, 0x40, 0x02, 0x40, 0x02, 0x40, 0x02, 0x00, 0x00}, /\*"[",59\*/  
 {0x00, 0x00, 0x30, 0x00, 0x0C, 0x00, 0x03, 0x80, 0x00, 0x60, 0x00, 0x1C, 0x00, 0x03, 0x00, 0x00}, /\*"\",60\*/  
 {0x00, 0x00, 0x40, 0x02, 0x40, 0x02, 0x40, 0x02, 0x7F, 0xFE, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"]",61\*/  
 {0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x40, 0x00, 0x40, 0x00, 0x40, 0x00, 0x20, 0x00, 0x00, 0x00}, /\*"^",62\*/  
 {0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01}, /\*"\_",63\*/  
 {0x00, 0x00, 0x40, 0x00, 0x40, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"`",64\*/  
 {0x00, 0x00, 0x00, 0x98, 0x01, 0x24, 0x01, 0x44, 0x01, 0x44, 0x01, 0x44, 0x00, 0xFC, 0x00, 0x04}, /\*"a",65\*/  
 {0x10, 0x00, 0x1F, 0xFC, 0x00, 0x88, 0x01, 0x04, 0x01, 0x04, 0x00, 0x88, 0x00, 0x70, 0x00, 0x00}, /\*"b",66\*/  
 {0x00, 0x00, 0x00, 0x70, 0x00, 0x88, 0x01, 0x04, 0x01, 0x04, 0x01, 0x04, 0x00, 0x88, 0x00, 0x00}, /\*"c",67\*/  
 {0x00, 0x00, 0x00, 0x70, 0x00, 0x88, 0x01, 0x04, 0x01, 0x04, 0x11, 0x08, 0x1F, 0xFC, 0x00, 0x04}, /\*"d",68\*/  
 {0x00, 0x00, 0x00, 0xF8, 0x01, 0x44, 0x01, 0x44, 0x01, 0x44, 0x01, 0x44, 0x00, 0xC8, 0x00, 0x00}, /\*"e",69\*/  
 {0x00, 0x00, 0x01, 0x04, 0x01, 0x04, 0x0F, 0xFC, 0x11, 0x04, 0x11, 0x04, 0x11, 0x00, 0x18, 0x00}, /\*"f",70\*/  
 {0x00, 0x00, 0x00, 0xD6, 0x01, 0x29, 0x01, 0x29, 0x01, 0x29, 0x01, 0xC9, 0x01, 0x06, 0x00, 0x00}, /\*"g",71\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x00, 0x84, 0x01, 0x00, 0x01, 0x00, 0x01, 0x04, 0x00, 0xFC, 0x00, 0x04}, /\*"h",72\*/  
 {0x00, 0x00, 0x01, 0x04, 0x19, 0x04, 0x19, 0xFC, 0x00, 0x04, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00}, /\*"i",73\*/  
 {0x00, 0x00, 0x00, 0x03, 0x00, 0x01, 0x01, 0x01, 0x19, 0x01, 0x19, 0xFE, 0x00, 0x00, 0x00, 0x00}, /\*"j",74\*/  
 {0x10, 0x04, 0x1F, 0xFC, 0x00, 0x24, 0x00, 0x40, 0x01, 0xB4, 0x01, 0x0C, 0x01, 0x04, 0x00, 0x00}, /\*"k",75\*/  
 {0x00, 0x00, 0x10, 0x04, 0x10, 0x04, 0x1F, 0xFC, 0x00, 0x04, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00}, /\*"l",76\*/  
 {0x01, 0x04, 0x01, 0xFC, 0x01, 0x04, 0x01, 0x00, 0x01, 0xFC, 0x01, 0x04, 0x01, 0x00, 0x00, 0xFC}, /\*"m",77\*/  
 {0x01, 0x04, 0x01, 0xFC, 0x00, 0x84, 0x01, 0x00, 0x01, 0x00, 0x01, 0x04, 0x00, 0xFC, 0x00, 0x04}, /\*"n",78\*/  
 {0x00, 0x00, 0x00, 0xF8, 0x01, 0x04, 0x01, 0x04, 0x01, 0x04, 0x01, 0x04, 0x00, 0xF8, 0x00, 0x00}, /\*"o",79\*/  
 {0x01, 0x01, 0x01, 0xFF, 0x00, 0x85, 0x01, 0x04, 0x01, 0x04, 0x00, 0x88, 0x00, 0x70, 0x00, 0x00}, /\*"p",80\*/  
 {0x00, 0x00, 0x00, 0x70, 0x00, 0x88, 0x01, 0x04, 0x01, 0x04, 0x01, 0x05, 0x01, 0xFF, 0x00, 0x01}, /\*"q",81\*/  
 {0x01, 0x04, 0x01, 0x04, 0x01, 0xFC, 0x00, 0x84, 0x01, 0x04, 0x01, 0x00, 0x01, 0x80, 0x00, 0x00}, /\*"r",82\*/  
 {0x00, 0x00, 0x00, 0xCC, 0x01, 0x24, 0x01, 0x24, 0x01, 0x24, 0x01, 0x24, 0x01, 0x98, 0x00, 0x00}, /\*"s",83\*/  
 {0x00, 0x00, 0x01, 0x00, 0x01, 0x00, 0x07, 0xF8, 0x01, 0x04, 0x01, 0x04, 0x00, 0x00, 0x00, 0x00}, /\*"t",84\*/  
 {0x01, 0x00, 0x01, 0xF8, 0x00, 0x04, 0x00, 0x04, 0x00, 0x04, 0x01, 0x08, 0x01, 0xFC, 0x00, 0x04}, /\*"u",85\*/  
 {0x01, 0x00, 0x01, 0x80, 0x01, 0x70, 0x00, 0x0C, 0x00, 0x10, 0x01, 0x60, 0x01, 0x80, 0x01, 0x00}, /\*"v",86\*/  
 {0x01, 0xF0, 0x01, 0x0C, 0x00, 0x30, 0x01, 0xC0, 0x00, 0x30, 0x01, 0x0C, 0x01, 0xF0, 0x01, 0x00}, /\*"w",87\*/  
 {0x00, 0x00, 0x01, 0x04, 0x01, 0x8C, 0x00, 0x74, 0x01, 0x70, 0x01, 0x8C, 0x01, 0x04, 0x00, 0x00}, /\*"x",88\*/  
 {0x01, 0x01, 0x01, 0x81, 0x01, 0x71, 0x00, 0x0E, 0x00, 0x18, 0x01, 0x60, 0x01, 0x80, 0x01, 0x00}, /\*"y",89\*/  
 {0x00, 0x00, 0x01, 0x84, 0x01, 0x0C, 0x01, 0x34, 0x01, 0x44, 0x01, 0x84, 0x01, 0x0C, 0x00, 0x00}, /\*"z",90\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x3E, 0xFC, 0x40, 0x02, 0x40, 0x02}, /\*"{",91\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"|",92\*/  
 {0x00, 0x00, 0x40, 0x02, 0x40, 0x02, 0x3E, 0xFC, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"}",93\*/  
 {0x00, 0x00, 0x60, 0x00, 0x80, 0x00, 0x80, 0x00, 0x40, 0x00, 0x40, 0x00, 0x20, 0x00, 0x20, 0x00}, /\*"~",94\*/  
};  
  
/\* 24\*24 ASICII字符集点阵 \*/  
const unsigned char oled\_asc2\_2412[95][36] = {  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*" ",0\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x80, 0x38, 0x0F, 0xFE, 0x38, 0x0F, 0x80, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"!",1\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x06, 0x00, 0x00, 0x0C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x31, 0x00, 0x00, 0x06, 0x00, 0x00, 0x0C, 0x00, 0x00, 0x38, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*""",2\*/  
 {0x00, 0x00, 0x00, 0x00, 0x61, 0x80, 0x00, 0x67, 0xF8, 0x07, 0xF9, 0x80, 0x00, 0x61, 0x80, 0x00, 0x61, 0x80, 0x00, 0x61, 0x80, 0x00, 0x61, 0x80, 0x00, 0x67, 0xF8, 0x07, 0xF9, 0x80, 0x00, 0x61, 0x80, 0x00, 0x00, 0x00}, /\*"#",3\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xC0, 0xE0, 0x03, 0xE0, 0xF0, 0x06, 0x30, 0x08, 0x04, 0x18, 0x08, 0x1F, 0xFF, 0xFE, 0x04, 0x0E, 0x08, 0x07, 0x87, 0xF0, 0x03, 0x81, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"$",4\*/  
 {0x01, 0xF0, 0x00, 0x06, 0x0C, 0x00, 0x04, 0x04, 0x08, 0x06, 0x0C, 0x70, 0x01, 0xF9, 0xC0, 0x00, 0x0E, 0x00, 0x00, 0x3B, 0xE0, 0x00, 0xEC, 0x18, 0x07, 0x08, 0x08, 0x04, 0x0C, 0x18, 0x00, 0x03, 0xE0, 0x00, 0x00, 0x00}, /\*"%",5\*/  
 {0x00, 0x01, 0xE0, 0x00, 0x07, 0xF0, 0x03, 0xF8, 0x18, 0x04, 0x1C, 0x08, 0x04, 0x17, 0x08, 0x07, 0xE1, 0xD0, 0x03, 0xC0, 0xE0, 0x00, 0x23, 0xB0, 0x00, 0x3C, 0x08, 0x00, 0x20, 0x08, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00}, /\*"&",6\*/  
 {0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x31, 0x00, 0x00, 0x32, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"'",7\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7F, 0x00, 0x01, 0xFF, 0xC0, 0x07, 0x80, 0xF0, 0x0C, 0x00, 0x18, 0x10, 0x00, 0x04, 0x20, 0x00, 0x02, 0x00, 0x00, 0x00}, /\*"(",8\*/  
 {0x00, 0x00, 0x00, 0x20, 0x00, 0x02, 0x10, 0x00, 0x04, 0x0C, 0x00, 0x18, 0x07, 0x80, 0xF0, 0x01, 0xFF, 0xC0, 0x00, 0x7F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*")",9\*/  
 {0x00, 0x00, 0x00, 0x00, 0x42, 0x00, 0x00, 0x66, 0x00, 0x00, 0x66, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x18, 0x00, 0x03, 0xFF, 0xC0, 0x00, 0x18, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x66, 0x00, 0x00, 0x66, 0x00, 0x00, 0x42, 0x00}, /\*"\*",10\*/  
 {0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x01, 0xFF, 0xC0, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00}, /\*"+",11\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x31, 0x00, 0x00, 0x32, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*",",12\*/  
 {0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00}, /\*"-",13\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x38, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*".",14\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x70, 0x00, 0x01, 0x80, 0x00, 0x0E, 0x00, 0x00, 0x38, 0x00, 0x00, 0xC0, 0x00, 0x07, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"/",15\*/  
 {0x00, 0x00, 0x00, 0x00, 0x7F, 0x80, 0x01, 0xFF, 0xE0, 0x03, 0x80, 0x70, 0x06, 0x00, 0x18, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x06, 0x00, 0x18, 0x03, 0x80, 0x70, 0x01, 0xFF, 0xE0, 0x00, 0x7F, 0x80, 0x00, 0x00, 0x00}, /\*"0",16\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x08, 0x01, 0x00, 0x08, 0x01, 0x00, 0x08, 0x03, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"1",17\*/  
 {0x00, 0x00, 0x00, 0x01, 0xC0, 0x38, 0x02, 0xC0, 0x58, 0x04, 0x00, 0x98, 0x04, 0x01, 0x18, 0x04, 0x02, 0x18, 0x04, 0x04, 0x18, 0x06, 0x1C, 0x18, 0x03, 0xF8, 0x18, 0x01, 0xE0, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"2",18\*/  
 {0x00, 0x00, 0x00, 0x01, 0xC0, 0xE0, 0x03, 0xC0, 0xF0, 0x04, 0x00, 0x08, 0x04, 0x08, 0x08, 0x04, 0x08, 0x08, 0x06, 0x18, 0x08, 0x03, 0xF4, 0x18, 0x01, 0xE7, 0xF0, 0x00, 0x01, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"3",19\*/  
 {0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x0D, 0x00, 0x00, 0x11, 0x00, 0x00, 0x61, 0x00, 0x00, 0x81, 0x08, 0x03, 0x01, 0x08, 0x07, 0xFF, 0xF8, 0x0F, 0xFF, 0xF8, 0x00, 0x01, 0x08, 0x00, 0x01, 0x08, 0x00, 0x00, 0x00}, /\*"4",20\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0x07, 0xFC, 0xD0, 0x06, 0x08, 0x08, 0x06, 0x10, 0x08, 0x06, 0x10, 0x08, 0x06, 0x10, 0x08, 0x06, 0x18, 0x38, 0x06, 0x0F, 0xF0, 0x06, 0x07, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"5",21\*/  
 {0x00, 0x00, 0x00, 0x00, 0x3F, 0x80, 0x01, 0xFF, 0xE0, 0x03, 0x84, 0x30, 0x02, 0x08, 0x18, 0x04, 0x10, 0x08, 0x04, 0x10, 0x08, 0x04, 0x10, 0x08, 0x07, 0x18, 0x10, 0x03, 0x0F, 0xF0, 0x00, 0x07, 0xC0, 0x00, 0x00, 0x00}, /\*"6",22\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xC0, 0x00, 0x07, 0x00, 0x00, 0x06, 0x00, 0x00, 0x06, 0x00, 0xF8, 0x06, 0x07, 0xF8, 0x06, 0x18, 0x00, 0x06, 0xE0, 0x00, 0x07, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"7",23\*/  
 {0x00, 0x00, 0x00, 0x01, 0xE1, 0xE0, 0x03, 0xF7, 0xF0, 0x06, 0x34, 0x10, 0x04, 0x18, 0x08, 0x04, 0x18, 0x08, 0x04, 0x0C, 0x08, 0x04, 0x0C, 0x08, 0x06, 0x16, 0x18, 0x03, 0xF3, 0xF0, 0x01, 0xC1, 0xE0, 0x00, 0x00, 0x00}, /\*"8",24\*/  
 {0x00, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x03, 0xFC, 0x30, 0x03, 0x06, 0x38, 0x04, 0x02, 0x08, 0x04, 0x02, 0x08, 0x04, 0x02, 0x08, 0x04, 0x04, 0x10, 0x03, 0x08, 0xF0, 0x01, 0xFF, 0xC0, 0x00, 0x7F, 0x00, 0x00, 0x00, 0x00}, /\*"9",25\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x38, 0x00, 0x70, 0x38, 0x00, 0x70, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*":",26\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x1A, 0x00, 0x30, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*";",27\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x14, 0x00, 0x00, 0x22, 0x00, 0x00, 0x41, 0x00, 0x00, 0x80, 0x80, 0x01, 0x00, 0x40, 0x02, 0x00, 0x20, 0x04, 0x00, 0x10, 0x08, 0x00, 0x08, 0x00, 0x00, 0x00}, /\*"<",28\*/  
 {0x00, 0x00, 0x00, 0x00, 0x21, 0x00, 0x00, 0x21, 0x00, 0x00, 0x21, 0x00, 0x00, 0x21, 0x00, 0x00, 0x21, 0x00, 0x00, 0x21, 0x00, 0x00, 0x21, 0x00, 0x00, 0x21, 0x00, 0x00, 0x21, 0x00, 0x00, 0x21, 0x00, 0x00, 0x00, 0x00}, /\*"=",29\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x08, 0x04, 0x00, 0x10, 0x02, 0x00, 0x20, 0x01, 0x00, 0x40, 0x00, 0x80, 0x80, 0x00, 0x41, 0x00, 0x00, 0x22, 0x00, 0x00, 0x14, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00}, /\*">",30\*/  
 {0x00, 0x00, 0x00, 0x03, 0xC0, 0x00, 0x04, 0xC0, 0x00, 0x04, 0x00, 0x00, 0x08, 0x00, 0x38, 0x08, 0x0F, 0x38, 0x08, 0x08, 0x38, 0x08, 0x10, 0x00, 0x0C, 0x30, 0x00, 0x07, 0xE0, 0x00, 0x03, 0xC0, 0x00, 0x00, 0x00, 0x00}, /\*"?",31\*/  
 {0x00, 0x00, 0x00, 0x00, 0x3F, 0x80, 0x00, 0xFF, 0xE0, 0x03, 0x80, 0x70, 0x02, 0x0F, 0x10, 0x06, 0x70, 0x88, 0x04, 0xC0, 0x88, 0x04, 0x83, 0x08, 0x04, 0x7F, 0x88, 0x02, 0xC0, 0x90, 0x03, 0x01, 0x20, 0x00, 0xFE, 0x40}, /\*"@",32\*/  
 {0x00, 0x00, 0x08, 0x00, 0x00, 0x18, 0x00, 0x01, 0xF8, 0x00, 0x3E, 0x08, 0x01, 0xC2, 0x00, 0x07, 0x02, 0x00, 0x07, 0xE2, 0x00, 0x00, 0xFE, 0x00, 0x00, 0x1F, 0xC8, 0x00, 0x01, 0xF8, 0x00, 0x00, 0x38, 0x00, 0x00, 0x08}, /\*"A",33\*/  
 {0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x08, 0x08, 0x04, 0x08, 0x08, 0x04, 0x08, 0x08, 0x04, 0x08, 0x08, 0x06, 0x18, 0x08, 0x03, 0xF4, 0x18, 0x01, 0xE7, 0xF0, 0x00, 0x01, 0xE0, 0x00, 0x00, 0x00}, /\*"B",34\*/  
 {0x00, 0x00, 0x00, 0x00, 0x3F, 0x80, 0x01, 0xFF, 0xE0, 0x03, 0x80, 0x70, 0x02, 0x00, 0x18, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x04, 0x00, 0x10, 0x06, 0x00, 0x20, 0x07, 0x80, 0xC0, 0x00, 0x00, 0x00}, /\*"C",35\*/  
 {0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x04, 0x00, 0x18, 0x02, 0x00, 0x10, 0x03, 0x80, 0x70, 0x01, 0xFF, 0xE0, 0x00, 0x7F, 0x80, 0x00, 0x00, 0x00}, /\*"D",36\*/  
 {0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x08, 0x08, 0x04, 0x08, 0x08, 0x04, 0x08, 0x08, 0x04, 0x08, 0x08, 0x04, 0x3E, 0x08, 0x04, 0x00, 0x08, 0x06, 0x00, 0x18, 0x01, 0x00, 0x60, 0x00, 0x00, 0x00}, /\*"E",37\*/  
 {0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x08, 0x08, 0x04, 0x08, 0x00, 0x04, 0x08, 0x00, 0x04, 0x08, 0x00, 0x04, 0x3E, 0x00, 0x06, 0x00, 0x00, 0x06, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x00, 0x00}, /\*"F",38\*/  
 {0x00, 0x00, 0x00, 0x00, 0x3F, 0x80, 0x01, 0xFF, 0xE0, 0x03, 0x80, 0x70, 0x06, 0x00, 0x18, 0x04, 0x00, 0x08, 0x04, 0x02, 0x08, 0x04, 0x02, 0x08, 0x02, 0x03, 0xF0, 0x07, 0x83, 0xF0, 0x00, 0x02, 0x00, 0x00, 0x02, 0x00}, /\*"G",39\*/  
 {0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x08, 0x08, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x04, 0x08, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x00, 0x08}, /\*"H",40\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"I",41\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x07, 0x00, 0x00, 0x01, 0x04, 0x00, 0x01, 0x04, 0x00, 0x01, 0x04, 0x00, 0x03, 0x07, 0xFF, 0xFE, 0x07, 0xFF, 0xFC, 0x04, 0x00, 0x00, 0x04, 0x00, 0x00, 0x04, 0x00, 0x00}, /\*"J",42\*/  
 {0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x0C, 0x08, 0x00, 0x18, 0x00, 0x00, 0x3E, 0x00, 0x04, 0xC7, 0x80, 0x05, 0x03, 0xC8, 0x06, 0x00, 0xF8, 0x04, 0x00, 0x38, 0x04, 0x00, 0x18, 0x00, 0x00, 0x08}, /\*"K",43\*/  
 {0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00}, /\*"L",44\*/  
 {0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0x80, 0x08, 0x07, 0xFC, 0x00, 0x00, 0x7F, 0xC0, 0x00, 0x03, 0xF8, 0x00, 0x07, 0xC0, 0x00, 0x78, 0x00, 0x07, 0x80, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x00, 0x08}, /\*"M",45\*/  
 {0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0x00, 0x08, 0x03, 0xC0, 0x00, 0x00, 0xE0, 0x00, 0x00, 0x38, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x07, 0x00, 0x00, 0x01, 0xC0, 0x04, 0x00, 0xF0, 0x07, 0xFF, 0xF8, 0x04, 0x00, 0x00}, /\*"N",46\*/  
 {0x00, 0x00, 0x00, 0x00, 0x7F, 0x80, 0x01, 0xFF, 0xE0, 0x03, 0x80, 0x70, 0x06, 0x00, 0x18, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x06, 0x00, 0x18, 0x03, 0x00, 0x30, 0x01, 0xFF, 0xE0, 0x00, 0x7F, 0x80, 0x00, 0x00, 0x00}, /\*"O",47\*/  
 {0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x04, 0x08, 0x04, 0x04, 0x00, 0x04, 0x04, 0x00, 0x04, 0x04, 0x00, 0x04, 0x04, 0x00, 0x06, 0x0C, 0x00, 0x03, 0xF8, 0x00, 0x01, 0xF0, 0x00, 0x00, 0x00, 0x00}, /\*"P",48\*/  
 {0x00, 0x00, 0x00, 0x00, 0x7F, 0x80, 0x01, 0xFF, 0xE0, 0x03, 0x80, 0x70, 0x06, 0x00, 0x88, 0x04, 0x00, 0x88, 0x04, 0x00, 0xC8, 0x06, 0x00, 0x3C, 0x03, 0x00, 0x3E, 0x01, 0xFF, 0xE6, 0x00, 0x7F, 0x84, 0x00, 0x00, 0x00}, /\*"Q",49\*/  
 {0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x08, 0x08, 0x04, 0x08, 0x00, 0x04, 0x0C, 0x00, 0x04, 0x0F, 0x00, 0x04, 0x0B, 0xC0, 0x06, 0x10, 0xF0, 0x03, 0xF0, 0x38, 0x01, 0xE0, 0x08, 0x00, 0x00, 0x08}, /\*"R",50\*/  
 {0x00, 0x00, 0x00, 0x01, 0xE0, 0xF8, 0x03, 0xF0, 0x30, 0x06, 0x30, 0x10, 0x04, 0x18, 0x08, 0x04, 0x18, 0x08, 0x04, 0x0C, 0x08, 0x04, 0x0C, 0x08, 0x02, 0x06, 0x18, 0x02, 0x07, 0xF0, 0x07, 0x81, 0xE0, 0x00, 0x00, 0x00}, /\*"S",51\*/  
 {0x01, 0x80, 0x00, 0x06, 0x00, 0x00, 0x04, 0x00, 0x00, 0x04, 0x00, 0x00, 0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0xF8, 0x04, 0x00, 0x08, 0x04, 0x00, 0x00, 0x04, 0x00, 0x00, 0x06, 0x00, 0x00, 0x01, 0x80, 0x00}, /\*"T",52\*/  
 {0x04, 0x00, 0x00, 0x07, 0xFF, 0xE0, 0x07, 0xFF, 0xF0, 0x04, 0x00, 0x18, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x04, 0x00, 0x10, 0x07, 0xFF, 0xE0, 0x04, 0x00, 0x00}, /\*"U",53\*/  
 {0x04, 0x00, 0x00, 0x06, 0x00, 0x00, 0x07, 0xE0, 0x00, 0x07, 0xFE, 0x00, 0x04, 0x1F, 0xE0, 0x00, 0x01, 0xF8, 0x00, 0x00, 0x38, 0x00, 0x01, 0xE0, 0x04, 0x3E, 0x00, 0x07, 0xC0, 0x00, 0x06, 0x00, 0x00, 0x04, 0x00, 0x00}, /\*"V",54\*/  
 {0x04, 0x00, 0x00, 0x07, 0xE0, 0x00, 0x07, 0xFF, 0xC0, 0x04, 0x1F, 0xF8, 0x00, 0x07, 0xC0, 0x07, 0xF8, 0x00, 0x07, 0xFF, 0x80, 0x04, 0x3F, 0xF8, 0x00, 0x07, 0xC0, 0x04, 0xF8, 0x00, 0x07, 0x00, 0x00, 0x04, 0x00, 0x00}, /\*"W",55\*/  
 {0x00, 0x00, 0x00, 0x04, 0x00, 0x08, 0x06, 0x00, 0x18, 0x07, 0xC0, 0x78, 0x05, 0xF1, 0xC8, 0x00, 0x3E, 0x00, 0x00, 0x1F, 0x80, 0x04, 0x63, 0xE8, 0x07, 0x80, 0xF8, 0x06, 0x00, 0x18, 0x04, 0x00, 0x08, 0x00, 0x00, 0x00}, /\*"X",56\*/  
 {0x04, 0x00, 0x00, 0x06, 0x00, 0x00, 0x07, 0x80, 0x00, 0x07, 0xE0, 0x08, 0x04, 0x7C, 0x08, 0x00, 0x1F, 0xF8, 0x00, 0x07, 0xF8, 0x00, 0x18, 0x08, 0x04, 0xE0, 0x08, 0x07, 0x00, 0x00, 0x06, 0x00, 0x00, 0x04, 0x00, 0x00}, /\*"Y",57\*/  
 {0x00, 0x00, 0x00, 0x01, 0x00, 0x08, 0x06, 0x00, 0x38, 0x04, 0x00, 0xF8, 0x04, 0x03, 0xE8, 0x04, 0x0F, 0x08, 0x04, 0x7C, 0x08, 0x05, 0xF0, 0x08, 0x07, 0xC0, 0x08, 0x07, 0x00, 0x18, 0x04, 0x00, 0x60, 0x00, 0x00, 0x00}, /\*"Z",58\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0xFF, 0xFE, 0x20, 0x00, 0x02, 0x20, 0x00, 0x02, 0x20, 0x00, 0x02, 0x20, 0x00, 0x02, 0x20, 0x00, 0x02, 0x00, 0x00, 0x00}, /\*"[",59\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x38, 0x00, 0x00, 0x06, 0x00, 0x00, 0x01, 0xC0, 0x00, 0x00, 0x30, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00}, /\*"\",60\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x02, 0x20, 0x00, 0x02, 0x20, 0x00, 0x02, 0x20, 0x00, 0x02, 0x20, 0x00, 0x02, 0x3F, 0xFF, 0xFE, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"]",61\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x10, 0x00, 0x00, 0x30, 0x00, 0x00, 0x20, 0x00, 0x00, 0x30, 0x00, 0x00, 0x10, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"^",62\*/  
 {0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x00, 0x00, 0x01}, /\*"\_",63\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x20, 0x00, 0x00, 0x10, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"`",64\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x00, 0x19, 0xF8, 0x00, 0x1B, 0x18, 0x00, 0x22, 0x08, 0x00, 0x26, 0x08, 0x00, 0x24, 0x08, 0x00, 0x24, 0x10, 0x00, 0x3F, 0xF8, 0x00, 0x1F, 0xF8, 0x00, 0x00, 0x08, 0x00, 0x00, 0x18}, /\*"a",65\*/  
 {0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x07, 0xFF, 0xF8, 0x0F, 0xFF, 0xF0, 0x00, 0x18, 0x18, 0x00, 0x10, 0x08, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x30, 0x18, 0x00, 0x1F, 0xF0, 0x00, 0x0F, 0xC0, 0x00, 0x00, 0x00}, /\*"b",66\*/  
 {0x00, 0x00, 0x00, 0x00, 0x07, 0xC0, 0x00, 0x1F, 0xF0, 0x00, 0x18, 0x30, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x3C, 0x08, 0x00, 0x1C, 0x10, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"c",67\*/  
 {0x00, 0x00, 0x00, 0x00, 0x07, 0xC0, 0x00, 0x1F, 0xF0, 0x00, 0x38, 0x18, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x04, 0x10, 0x10, 0x07, 0xFF, 0xF8, 0x0F, 0xFF, 0xF0, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00}, /\*"d",68\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xC0, 0x00, 0x1F, 0xF0, 0x00, 0x12, 0x30, 0x00, 0x22, 0x18, 0x00, 0x22, 0x08, 0x00, 0x22, 0x08, 0x00, 0x32, 0x08, 0x00, 0x1E, 0x10, 0x00, 0x0E, 0x20, 0x00, 0x00, 0x00}, /\*"e",69\*/  
 {0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x01, 0xFF, 0xF8, 0x03, 0xFF, 0xF8, 0x06, 0x20, 0x08, 0x04, 0x20, 0x08, 0x04, 0x20, 0x08, 0x07, 0x20, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"f",70\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x0E, 0x00, 0x0E, 0x6E, 0x00, 0x1F, 0xF3, 0x00, 0x31, 0xB1, 0x00, 0x20, 0xB1, 0x00, 0x20, 0xB1, 0x00, 0x31, 0x91, 0x00, 0x1F, 0x13, 0x00, 0x2E, 0x1E, 0x00, 0x20, 0x0E, 0x00, 0x30, 0x00}, /\*"g",71\*/  
 {0x00, 0x00, 0x00, 0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x0F, 0xFF, 0xF8, 0x00, 0x10, 0x08, 0x00, 0x20, 0x00, 0x00, 0x20, 0x00, 0x00, 0x20, 0x08, 0x00, 0x3F, 0xF8, 0x00, 0x1F, 0xF8, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00}, /\*"h",72\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x06, 0x3F, 0xF8, 0x06, 0x3F, 0xF8, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"i",73\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x03, 0x00, 0x20, 0x01, 0x00, 0x20, 0x01, 0x00, 0x20, 0x03, 0x06, 0x3F, 0xFE, 0x06, 0x3F, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"j",74\*/  
 {0x00, 0x00, 0x00, 0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x0F, 0xFF, 0xF8, 0x00, 0x01, 0x88, 0x00, 0x03, 0x00, 0x00, 0x2F, 0xC0, 0x00, 0x38, 0xF8, 0x00, 0x20, 0x38, 0x00, 0x20, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00}, /\*"k",75\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x04, 0x00, 0x08, 0x07, 0xFF, 0xF8, 0x0F, 0xFF, 0xF8, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"l",76\*/  
 {0x00, 0x20, 0x08, 0x00, 0x3F, 0xF8, 0x00, 0x3F, 0xF8, 0x00, 0x10, 0x08, 0x00, 0x20, 0x00, 0x00, 0x3F, 0xF8, 0x00, 0x3F, 0xF8, 0x00, 0x10, 0x08, 0x00, 0x20, 0x00, 0x00, 0x3F, 0xF8, 0x00, 0x3F, 0xF8, 0x00, 0x00, 0x08}, /\*"m",77\*/  
 {0x00, 0x00, 0x00, 0x00, 0x20, 0x08, 0x00, 0x3F, 0xF8, 0x00, 0x3F, 0xF8, 0x00, 0x10, 0x08, 0x00, 0x10, 0x00, 0x00, 0x20, 0x00, 0x00, 0x20, 0x08, 0x00, 0x3F, 0xF8, 0x00, 0x1F, 0xF8, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00}, /\*"n",78\*/  
 {0x00, 0x00, 0x00, 0x00, 0x07, 0xC0, 0x00, 0x0F, 0xF0, 0x00, 0x18, 0x30, 0x00, 0x30, 0x08, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x30, 0x08, 0x00, 0x18, 0x30, 0x00, 0x0F, 0xF0, 0x00, 0x07, 0xC0, 0x00, 0x00, 0x00}, /\*"o",79\*/  
 {0x00, 0x00, 0x00, 0x00, 0x20, 0x01, 0x00, 0x3F, 0xFF, 0x00, 0x3F, 0xFF, 0x00, 0x10, 0x11, 0x00, 0x20, 0x09, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x30, 0x38, 0x00, 0x1F, 0xF0, 0x00, 0x0F, 0xC0, 0x00, 0x00, 0x00}, /\*"p",80\*/  
 {0x00, 0x00, 0x00, 0x00, 0x07, 0xC0, 0x00, 0x1F, 0xF0, 0x00, 0x38, 0x18, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x20, 0x09, 0x00, 0x10, 0x11, 0x00, 0x1F, 0xFF, 0x00, 0x3F, 0xFF, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00}, /\*"q",81\*/  
 {0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x3F, 0xF8, 0x00, 0x3F, 0xF8, 0x00, 0x08, 0x08, 0x00, 0x10, 0x08, 0x00, 0x20, 0x08, 0x00, 0x20, 0x00, 0x00, 0x30, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00}, /\*"r",82\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x78, 0x00, 0x1E, 0x18, 0x00, 0x33, 0x08, 0x00, 0x23, 0x08, 0x00, 0x21, 0x08, 0x00, 0x21, 0x88, 0x00, 0x21, 0x98, 0x00, 0x30, 0xF0, 0x00, 0x38, 0x60, 0x00, 0x00, 0x00}, /\*"s",83\*/  
 {0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x20, 0x00, 0x00, 0x20, 0x00, 0x00, 0xFF, 0xF0, 0x03, 0xFF, 0xF8, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"t",84\*/  
 {0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x3F, 0xF0, 0x00, 0x7F, 0xF8, 0x00, 0x00, 0x18, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x20, 0x10, 0x00, 0x3F, 0xF8, 0x00, 0x7F, 0xF0, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00}, /\*"u",85\*/  
 {0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x30, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x3F, 0x80, 0x00, 0x23, 0xF0, 0x00, 0x00, 0x78, 0x00, 0x00, 0x70, 0x00, 0x23, 0x80, 0x00, 0x3C, 0x00, 0x00, 0x30, 0x00, 0x00, 0x20, 0x00}, /\*"v",86\*/  
 {0x00, 0x20, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x3F, 0xE0, 0x00, 0x23, 0xF8, 0x00, 0x00, 0xE0, 0x00, 0x27, 0x00, 0x00, 0x3E, 0x00, 0x00, 0x3F, 0xE0, 0x00, 0x21, 0xF8, 0x00, 0x01, 0xE0, 0x00, 0x3E, 0x00, 0x00, 0x20, 0x00}, /\*"w",87\*/  
 {0x00, 0x00, 0x00, 0x00, 0x20, 0x08, 0x00, 0x20, 0x08, 0x00, 0x38, 0x38, 0x00, 0x3E, 0x68, 0x00, 0x27, 0x80, 0x00, 0x03, 0xC8, 0x00, 0x2C, 0xF8, 0x00, 0x38, 0x38, 0x00, 0x20, 0x18, 0x00, 0x20, 0x08, 0x00, 0x00, 0x00}, /\*"x",88\*/  
 {0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x30, 0x03, 0x00, 0x3C, 0x01, 0x00, 0x3F, 0x83, 0x00, 0x23, 0xEC, 0x00, 0x00, 0x70, 0x00, 0x23, 0x80, 0x00, 0x3C, 0x00, 0x00, 0x20, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00}, /\*"y",89\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x08, 0x00, 0x20, 0x38, 0x00, 0x20, 0xF8, 0x00, 0x23, 0xE8, 0x00, 0x2F, 0x88, 0x00, 0x3E, 0x08, 0x00, 0x38, 0x08, 0x00, 0x20, 0x18, 0x00, 0x00, 0x70, 0x00, 0x00, 0x00}, /\*"z",90\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x14, 0x00, 0x1F, 0xF7, 0xFC, 0x30, 0x00, 0x06, 0x20, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"{",91\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"|",92\*/  
 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x02, 0x30, 0x00, 0x06, 0x1F, 0xF7, 0xFC, 0x00, 0x14, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}, /\*"}",93\*/  
 {0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x60, 0x00, 0x00, 0x40, 0x00, 0x00, 0x40, 0x00, 0x00, 0x20, 0x00, 0x00, 0x10, 0x00, 0x00, 0x08, 0x00, 0x00, 0x04, 0x00, 0x00, 0x04, 0x00, 0x00, 0x0C, 0x00, 0x00, 0x10, 0x00, 0x00}, /\*"~",94\*/  
};  
  
#endif

然后编写 oled 驱动的头文件。

Core/Inc/oled.h

#ifndef \_\_OLED\_H  
#define \_\_OLED\_H  
  
#include "stdint.h"  
#include "stdlib.h"  
  
/\* OLED模式设置  
 \* 0: 4线串行模式 （模块的BS1，BS2均接GND）  
 \* 1: 并行8080模式 （模块的BS1，BS2均接VCC）  
 \*/  
#define OLED\_MODE 1 /\* 默认使用8080并口模式 \*/  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
/\* OLED SPI模式引脚 定义 \*/  
/\* 注意:这里仅定义了 OLED 4线SPI模式驱动时的 引脚定义. 8位并口访问, 由于引脚太多,就不单独定义了. \*/  
  
#define OLED\_SPI\_RST\_PORT GPIOG  
#define OLED\_SPI\_RST\_PIN GPIO\_PIN\_15  
#define OLED\_SPI\_RST\_CLK\_ENABLE() \  
 do \  
 { \  
 \_\_HAL\_RCC\_GPIOG\_CLK\_ENABLE(); \  
 } while (0) /\* PG口时钟使能 \*/  
  
#define OLED\_SPI\_CS\_PORT GPIOD  
#define OLED\_SPI\_CS\_PIN GPIO\_PIN\_6  
#define OLED\_SPI\_CS\_CLK\_ENABLE() \  
 do \  
 { \  
 \_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE(); \  
 } while (0) /\* PD口时钟使能 \*/  
  
#define OLED\_SPI\_RS\_PORT GPIOD  
#define OLED\_SPI\_RS\_PIN GPIO\_PIN\_3  
#define OLED\_SPI\_RS\_CLK\_ENABLE() \  
 do \  
 { \  
 \_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE(); \  
 } while (0) /\* PD口时钟使能 \*/  
  
#define OLED\_SPI\_SCLK\_PORT GPIOC  
#define OLED\_SPI\_SCLK\_PIN GPIO\_PIN\_0  
#define OLED\_SPI\_SCLK\_CLK\_ENABLE() \  
 do \  
 { \  
 \_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE(); \  
 } while (0) /\* PC口时钟使能 \*/  
  
#define OLED\_SPI\_SDIN\_PORT GPIOC  
#define OLED\_SPI\_SDIN\_PIN GPIO\_PIN\_1  
#define OLED\_SPI\_SDIN\_CLK\_ENABLE() \  
 do \  
 { \  
 \_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE(); \  
 } while (0) /\* PC口时钟使能 \*/  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
/\* OLED SPI模式相关端口控制函数 定义  
 \* 注意:OLED\_RST/OLED\_CS/OLED\_RS,这三个是和80并口模式共用的,即80模式也必须实现这3个函数!  
 \*/  
#define OLED\_RST(x) \  
 do \  
 { \  
 x ? HAL\_GPIO\_WritePin(OLED\_SPI\_RST\_PORT, OLED\_SPI\_RST\_PIN, GPIO\_PIN\_SET) : HAL\_GPIO\_WritePin(OLED\_SPI\_RST\_PORT, OLED\_SPI\_RST\_PIN, GPIO\_PIN\_RESET); \  
 } while (0) /\* 设置RST引脚 \*/  
  
#define OLED\_CS(x) \  
 do \  
 { \  
 x ? HAL\_GPIO\_WritePin(OLED\_SPI\_CS\_PORT, OLED\_SPI\_CS\_PIN, GPIO\_PIN\_SET) : HAL\_GPIO\_WritePin(OLED\_SPI\_CS\_PORT, OLED\_SPI\_CS\_PIN, GPIO\_PIN\_RESET); \  
 } while (0) /\* 设置CS引脚 \*/  
#define OLED\_RS(x) \  
 do \  
 { \  
 x ? HAL\_GPIO\_WritePin(OLED\_SPI\_RS\_PORT, OLED\_SPI\_RS\_PIN, GPIO\_PIN\_SET) : HAL\_GPIO\_WritePin(OLED\_SPI\_RS\_PORT, OLED\_SPI\_RS\_PIN, GPIO\_PIN\_RESET); \  
 } while (0) /\* 设置RS引脚 \*/  
  
#define OLED\_SCLK(x) \  
 do \  
 { \  
 x ? HAL\_GPIO\_WritePin(OLED\_SPI\_SCLK\_PORT, OLED\_SPI\_SCLK\_PIN, GPIO\_PIN\_SET) : HAL\_GPIO\_WritePin(OLED\_SPI\_SCLK\_PORT, OLED\_SPI\_SCLK\_PIN, GPIO\_PIN\_RESET); \  
 } while (0) /\* 设置SCLK引脚 \*/  
#define OLED\_SDIN(x) \  
 do \  
 { \  
 x ? HAL\_GPIO\_WritePin(OLED\_SPI\_SDIN\_PORT, OLED\_SPI\_SDIN\_PIN, GPIO\_PIN\_SET) : HAL\_GPIO\_WritePin(OLED\_SPI\_SDIN\_PORT, OLED\_SPI\_SDIN\_PIN, GPIO\_PIN\_RESET); \  
 } while (0) /\* 设置SDIN引脚 \*/  
  
/\* OLED 80并口模式WR,RD端口控制函数 定义 \*/  
#define OLED\_WR(x) \  
 do \  
 { \  
 x ? HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_14, GPIO\_PIN\_SET) : HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_14, GPIO\_PIN\_RESET); \  
 } while (0) /\* 设置WR引脚 \*/  
  
#define OLED\_RD(x) \  
 do \  
 { \  
 x ? HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_13, GPIO\_PIN\_SET) : HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_13, GPIO\_PIN\_RESET); \  
 } while (0) /\* 设置RD引脚 \*/  
  
/\* 命令/数据 定义 \*/  
#define OLED\_CMD 0 /\* 写命令 \*/  
#define OLED\_DATA 1 /\* 写数据 \*/  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
static void oled\_wr\_byte(uint8\_t data, uint8\_t cmd); /\* 写一个字节到OLED \*/  
static uint32\_t oled\_pow(uint8\_t m, uint8\_t n); /\* OLED求平方函数 \*/  
  
void oled\_init(void); /\* OLED初始化 \*/  
void oled\_clear(void); /\* OLED清屏 \*/  
void oled\_display\_on(void); /\* 开启OLED显示 \*/  
void oled\_display\_off(void); /\* 关闭OLED显示 \*/  
void oled\_refresh\_gram(void); /\* 更新显存到OLED \*/  
void oled\_draw\_point(uint8\_t x, uint8\_t y, uint8\_t dot); /\* OLED画点 \*/  
void oled\_fill(uint8\_t x1, uint8\_t y1, uint8\_t x2, uint8\_t y2, uint8\_t dot); /\* OLED区域填充 \*/  
void oled\_show\_char(uint8\_t x, uint8\_t y, uint8\_t chr, uint8\_t size, uint8\_t mode); /\* OLED显示字符 \*/  
void oled\_show\_num(uint8\_t x, uint8\_t y, uint32\_t num, uint8\_t len, uint8\_t size); /\* OLED显示数字 \*/  
void oled\_show\_string(uint8\_t x, uint8\_t y, const char \*p, uint8\_t size); /\* OLED显示字符串 \*/  
  
#endif

然后编写头文件对应的 C 语言程序。

Core/Src/oled.c

#include "main.h"  
#include "stdlib.h"  
#include "oled.h"  
#include "oledfont.h"  
  
/\*  
 \* OLED的显存  
 \* 每个字节表示8个像素, 128,表示有128列, 8表示有64行, 高位表示第行数.  
 \* 比如:g\_oled\_gram[0][0],包含了第一列,第1~8行的数据. g\_oled\_gram[0][0].0,即表示坐标(0,0)  
 \* 类似的: g\_oled\_gram[1][0].1,表示坐标(1,1), g\_oled\_gram[10][1].2,表示坐标(10,10),  
 \*  
 \* 存放格式如下(高位表示高行数).  
 \* [0]0 1 2 3 ... 127  
 \* [1]0 1 2 3 ... 127  
 \* [2]0 1 2 3 ... 127  
 \* [3]0 1 2 3 ... 127  
 \* [4]0 1 2 3 ... 127  
 \* [5]0 1 2 3 ... 127  
 \* [6]0 1 2 3 ... 127  
 \* [7]0 1 2 3 ... 127  
 \*/  
static uint8\_t g\_oled\_gram[128][8];  
  
/\*\*  
 \* @brief 更新显存到OLED  
 \* @param 无  
 \* @retval 无  
 \*/  
void oled\_refresh\_gram(void)  
{  
 uint8\_t i, n;  
  
 for (i = 0; i < 8; i++)  
 {  
 oled\_wr\_byte(0xb0 + i, OLED\_CMD); /\* 设置页地址（0~7） \*/  
 oled\_wr\_byte(0x00, OLED\_CMD); /\* 设置显示位置—列低地址 \*/  
 oled\_wr\_byte(0x10, OLED\_CMD); /\* 设置显示位置—列高地址 \*/  
  
 for (n = 0; n < 128; n++)  
 {  
 oled\_wr\_byte(g\_oled\_gram[n][i], OLED\_DATA);  
 }  
 }  
}  
  
#if OLED\_MODE == 1 /\* 使用8080并口驱动OLED \*/  
  
/\*\*  
 \* @brief 通过拼凑的方法向OLED输出一个8位数据  
 \* @param data: 要输出的数据  
 \* @retval 无  
 \*/  
static void oled\_data\_out(uint8\_t data)  
{  
 GPIOC->ODR = (GPIOC->ODR & 0XFF00) | (data & 0X00FF);  
}  
  
/\*\*  
 \* @brief 向OLED写入一个字节  
 \* @param data: 要输出的数据  
 \* @param cmd: 数据/命令标志 0,表示命令;1,表示数据;  
 \* @retval 无  
 \*/  
static void oled\_wr\_byte(uint8\_t data, uint8\_t cmd)  
{  
 oled\_data\_out(data);  
 OLED\_RS(cmd);  
 OLED\_CS(0);  
 OLED\_WR(0);  
 OLED\_WR(1);  
 OLED\_CS(1);  
 OLED\_RS(1);  
}  
  
#else /\* 使用SPI驱动OLED \*/  
  
/\*\*  
 \* @brief 向OLED写入一个字节  
 \* @param data: 要输出的数据  
 \* @param cmd: 数据/命令标志 0,表示命令;1,表示数据;  
 \* @retval 无  
 \*/  
static void oled\_wr\_byte(uint8\_t data, uint8\_t cmd)  
{  
 uint8\_t i;  
 OLED\_RS(cmd); /\* 写命令 \*/  
 OLED\_CS(0);  
  
 for (i = 0; i < 8; i++)  
 {  
 OLED\_SCLK(0);  
  
 if (data & 0x80)  
 {  
 OLED\_SDIN(1);  
 }  
 else  
 {  
 OLED\_SDIN(0);  
 }  
  
 OLED\_SCLK(1);  
 data <<= 1;  
 }  
  
 OLED\_CS(1);  
 OLED\_RS(1);  
}  
  
#endif  
  
/\*\*  
 \* @brief 开启OLED显示  
 \* @param 无  
 \* @retval 无  
 \*/  
void oled\_display\_on(void)  
{  
 oled\_wr\_byte(0X8D, OLED\_CMD); /\* SET DCDC命令 \*/  
 oled\_wr\_byte(0X14, OLED\_CMD); /\* DCDC ON \*/  
 oled\_wr\_byte(0XAF, OLED\_CMD); /\* DISPLAY ON \*/  
}  
  
/\*\*  
 \* @brief 关闭OLED显示  
 \* @param 无  
 \* @retval 无  
 \*/  
void oled\_display\_off(void)  
{  
 oled\_wr\_byte(0X8D, OLED\_CMD); /\* SET DCDC命令 \*/  
 oled\_wr\_byte(0X10, OLED\_CMD); /\* DCDC OFF \*/  
 oled\_wr\_byte(0XAE, OLED\_CMD); /\* DISPLAY OFF \*/  
}  
  
/\*\*  
 \* @brief 清屏函数,清完屏,整个屏幕是黑色的!和没点亮一样!!!  
 \* @param 无  
 \* @retval 无  
 \*/  
void oled\_clear(void)  
{  
 uint8\_t i, n;  
  
 for (i = 0; i < 8; i++)  
 for (n = 0; n < 128; n++)  
 g\_oled\_gram[n][i] = 0X00;  
  
 oled\_refresh\_gram(); /\* 更新显示 \*/  
}  
  
/\*\*  
 \* @brief OLED画点  
 \* @param x : 0~127  
 \* @param y : 0~63  
 \* @param dot: 1 填充 0,清空  
 \* @retval 无  
 \*/  
void oled\_draw\_point(uint8\_t x, uint8\_t y, uint8\_t dot)  
{  
 uint8\_t pos, bx, temp = 0;  
  
 if (x > 127 || y > 63)  
 return; /\* 超出范围了. \*/  
  
 pos = y / 8; /\* 计算GRAM里面的y坐标所在的字节, 每个字节可以存储8个行坐标 \*/  
  
 bx = y % 8; /\* 取余数,方便计算y在对应字节里面的位置,及行(y)位置 \*/  
 temp = 1 << bx; /\* 高位表示低行号, 得到y对应的bit位置,将该bit先置1 \*/  
  
 if (dot) /\* 画实心点 \*/  
 {  
 g\_oled\_gram[x][pos] |= temp;  
 }  
 else /\* 画空点,即不显示 \*/  
 {  
 g\_oled\_gram[x][pos] &= ~temp;  
 }  
}  
  
/\*\*  
 \* @brief OLED填充区域填充  
 \* @note: 注意:需要确保: x1<=x2; y1<=y2 0<=x1<=127 0<=y1<=63  
 \* @param x1,y1: 起点坐标  
 \* @param x2,y2: 终点坐标  
 \* @param dot: 1 填充 0,清空  
 \* @retval 无  
 \*/  
void oled\_fill(uint8\_t x1, uint8\_t y1, uint8\_t x2, uint8\_t y2, uint8\_t dot)  
{  
 uint8\_t x, y;  
  
 for (x = x1; x <= x2; x++)  
 {  
 for (y = y1; y <= y2; y++)  
 oled\_draw\_point(x, y, dot);  
 }  
  
 oled\_refresh\_gram(); /\* 更新显示 \*/  
}  
  
/\*\*  
 \* @brief 在指定位置显示一个字符,包括部分字符  
 \* @param x : 0~127  
 \* @param y : 0~63  
 \* @param size: 选择字体 12/16/24  
 \* @param mode: 0,反白显示;1,正常显示  
 \* @retval 无  
 \*/  
void oled\_show\_char(uint8\_t x, uint8\_t y, uint8\_t chr, uint8\_t size, uint8\_t mode)  
{  
 uint8\_t temp, t, t1;  
 uint8\_t y0 = y;  
 uint8\_t \*pfont = 0;  
 uint8\_t csize = (size / 8 + ((size % 8) ? 1 : 0)) \* (size / 2); /\* 得到字体一个字符对应点阵集所占的字节数 \*/  
 chr = chr - ' '; /\* 得到偏移后的值,因为字库是从空格开始存储的,第一个字符是空格 \*/  
  
 if (size == 12) /\* 调用1206字体 \*/  
 {  
 pfont = (uint8\_t \*)oled\_asc2\_1206[chr];  
 }  
 else if (size == 16) /\* 调用1608字体 \*/  
 {  
 pfont = (uint8\_t \*)oled\_asc2\_1608[chr];  
 }  
 else if (size == 24) /\* 调用2412字体 \*/  
 {  
 pfont = (uint8\_t \*)oled\_asc2\_2412[chr];  
 }  
 else /\* 没有的字库 \*/  
 {  
 return;  
 }  
  
 for (t = 0; t < csize; t++)  
 {  
 temp = pfont[t];  
  
 for (t1 = 0; t1 < 8; t1++)  
 {  
 if (temp & 0x80)  
 oled\_draw\_point(x, y, mode);  
 else  
 oled\_draw\_point(x, y, !mode);  
  
 temp <<= 1;  
 y++;  
  
 if ((y - y0) == size)  
 {  
 y = y0;  
 x++;  
 break;  
 }  
 }  
 }  
}  
  
/\*\*  
 \* @brief 平方函数, m^n  
 \* @param m: 底数  
 \* @param n: 指数  
 \* @retval 无  
 \*/  
static uint32\_t oled\_pow(uint8\_t m, uint8\_t n)  
{  
 uint32\_t result = 1;  
  
 while (n--)  
 {  
 result \*= m;  
 }  
  
 return result;  
}  
  
/\*\*  
 \* @brief 显示len个数字  
 \* @param x,y : 起始坐标  
 \* @param num : 数值(0 ~ 2^32)  
 \* @param len : 显示数字的位数  
 \* @param size: 选择字体 12/16/24  
 \* @retval 无  
 \*/  
void oled\_show\_num(uint8\_t x, uint8\_t y, uint32\_t num, uint8\_t len, uint8\_t size)  
{  
 uint8\_t t, temp;  
 uint8\_t enshow = 0;  
  
 for (t = 0; t < len; t++) /\* 按总显示位数循环 \*/  
 {  
 temp = (num / oled\_pow(10, len - t - 1)) % 10; /\* 获取对应位的数字 \*/  
  
 if (enshow == 0 && t < (len - 1)) /\* 没有使能显示,且还有位要显示 \*/  
 {  
 if (temp == 0)  
 {  
 oled\_show\_char(x + (size / 2) \* t, y, ' ', size, 1); /\* 显示空格,站位 \*/  
 continue; /\* 继续下个一位 \*/  
 }  
 else  
 {  
 enshow = 1; /\* 使能显示 \*/  
 }  
 }  
  
 oled\_show\_char(x + (size / 2) \* t, y, temp + '0', size, 1); /\* 显示字符 \*/  
 }  
}  
  
/\*\*  
 \* @brief 显示字符串  
 \* @param x,y : 起始坐标  
 \* @param size: 选择字体 12/16/24  
 \* @param \*p : 字符串指针,指向字符串首地址  
 \* @retval 无  
 \*/  
void oled\_show\_string(uint8\_t x, uint8\_t y, const char \*p, uint8\_t size)  
{  
 while ((\*p <= '~') && (\*p >= ' ')) /\* 判断是不是非法字符! \*/  
 {  
 if (x > (128 - (size / 2))) /\* 宽度越界 \*/  
 {  
 x = 0;  
 y += size; /\* 换行 \*/  
 }  
  
 if (y > (64 - size)) /\* 高度越界 \*/  
 {  
 y = x = 0;  
 oled\_clear();  
 }  
  
 oled\_show\_char(x, y, \*p, size, 1); /\* 显示一个字符 \*/  
 x += size / 2; /\* ASCII字符宽度为汉字宽度的一半 \*/  
 p++;  
 }  
}  
  
/\*\*  
 \* @brief 初始化OLED(SSD1306)  
 \* @param 无  
 \* @retval 无  
 \*/  
void oled\_init(void)  
{  
 GPIO\_InitTypeDef gpio\_init\_struct;  
 \_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE(); /\* 使能PORTC时钟 \*/  
 \_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE(); /\* 使能PORTD时钟 \*/  
 \_\_HAL\_RCC\_GPIOG\_CLK\_ENABLE(); /\* 使能PORTG时钟 \*/  
  
#if OLED\_MODE == 1 /\* 使用8080并口模式 \*/  
  
 /\* PC0 ~ 7 设置 \*/  
 gpio\_init\_struct.Pin = GPIO\_PIN\_0 | GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3 | GPIO\_PIN\_4 | GPIO\_PIN\_5 | GPIO\_PIN\_6 | GPIO\_PIN\_7;  
 gpio\_init\_struct.Mode = GPIO\_MODE\_OUTPUT\_PP; /\* 推挽输出 \*/  
 gpio\_init\_struct.Pull = GPIO\_PULLUP; /\* 上拉 \*/  
 gpio\_init\_struct.Speed = GPIO\_SPEED\_FREQ\_MEDIUM; /\* 中速 \*/  
 HAL\_GPIO\_Init(GPIOC, &gpio\_init\_struct); /\* PC0 ~ 7 设置 \*/  
  
 gpio\_init\_struct.Pin = GPIO\_PIN\_3 | GPIO\_PIN\_6; /\* PD3, PD6 设置 \*/  
 gpio\_init\_struct.Mode = GPIO\_MODE\_OUTPUT\_PP; /\* 推挽输出 \*/  
 gpio\_init\_struct.Pull = GPIO\_PULLUP; /\* 上拉 \*/  
 gpio\_init\_struct.Speed = GPIO\_SPEED\_FREQ\_MEDIUM; /\* 中速 \*/  
 HAL\_GPIO\_Init(GPIOD, &gpio\_init\_struct); /\* PD3, PD6 设置 \*/  
  
 gpio\_init\_struct.Pin = GPIO\_PIN\_13 | GPIO\_PIN\_14 | GPIO\_PIN\_15;  
 gpio\_init\_struct.Mode = GPIO\_MODE\_OUTPUT\_PP; /\* 推挽输出 \*/  
 gpio\_init\_struct.Pull = GPIO\_PULLUP; /\* 上拉 \*/  
 gpio\_init\_struct.Speed = GPIO\_SPEED\_FREQ\_MEDIUM; /\* 中速 \*/  
 HAL\_GPIO\_Init(GPIOG, &gpio\_init\_struct); /\* WR/RD/RST引脚模式设置 \*/  
  
 OLED\_WR(1);  
 OLED\_RD(1);  
  
#else /\* 使用4线SPI 串口模式 \*/  
  
 gpio\_init\_struct.Pin = OLED\_SPI\_RST\_PIN; /\* RST引脚 \*/  
 gpio\_init\_struct.Mode = GPIO\_MODE\_OUTPUT\_PP; /\* 推挽输出 \*/  
 gpio\_init\_struct.Pull = GPIO\_PULLUP; /\* 上拉 \*/  
 gpio\_init\_struct.Speed = GPIO\_SPEED\_FREQ\_MEDIUM; /\* 中速 \*/  
 HAL\_GPIO\_Init(OLED\_SPI\_RST\_PORT, &gpio\_init\_struct); /\* RST引脚模式设置 \*/  
  
 gpio\_init\_struct.Pin = OLED\_SPI\_CS\_PIN; /\* CS引脚 \*/  
 gpio\_init\_struct.Mode = GPIO\_MODE\_OUTPUT\_PP; /\* 推挽输出 \*/  
 gpio\_init\_struct.Pull = GPIO\_PULLUP; /\* 上拉 \*/  
 gpio\_init\_struct.Speed = GPIO\_SPEED\_FREQ\_MEDIUM; /\* 中速 \*/  
 HAL\_GPIO\_Init(OLED\_SPI\_CS\_PORT, &gpio\_init\_struct); /\* CS引脚模式设置 \*/  
  
 gpio\_init\_struct.Pin = OLED\_SPI\_RS\_PIN; /\* RS引脚 \*/  
 gpio\_init\_struct.Mode = GPIO\_MODE\_OUTPUT\_PP; /\* 推挽输出 \*/  
 gpio\_init\_struct.Pull = GPIO\_PULLUP; /\* 上拉 \*/  
 gpio\_init\_struct.Speed = GPIO\_SPEED\_FREQ\_MEDIUM; /\* 中速 \*/  
 HAL\_GPIO\_Init(OLED\_SPI\_RS\_PORT, &gpio\_init\_struct); /\* RS引脚模式设置 \*/  
  
 gpio\_init\_struct.Pin = OLED\_SPI\_SCLK\_PIN; /\* SCLK引脚 \*/  
 gpio\_init\_struct.Mode = GPIO\_MODE\_OUTPUT\_PP; /\* 推挽输出 \*/  
 gpio\_init\_struct.Pull = GPIO\_PULLUP; /\* 上拉 \*/  
 gpio\_init\_struct.Speed = GPIO\_SPEED\_FREQ\_MEDIUM; /\* 中速 \*/  
 HAL\_GPIO\_Init(OLED\_SPI\_SCLK\_PORT, &gpio\_init\_struct); /\* SCLK引脚模式设置 \*/  
  
 gpio\_init\_struct.Pin = OLED\_SPI\_SDIN\_PIN; /\* SDIN引脚模式设置 \*/  
 gpio\_init\_struct.Mode = GPIO\_MODE\_OUTPUT\_PP; /\* 推挽输出 \*/  
 gpio\_init\_struct.Pull = GPIO\_PULLUP; /\* 上拉 \*/  
 gpio\_init\_struct.Speed = GPIO\_SPEED\_FREQ\_MEDIUM; /\* 中速 \*/  
 HAL\_GPIO\_Init(OLED\_SPI\_SDIN\_PORT, &gpio\_init\_struct); /\* SDIN引脚模式设置 \*/  
  
 OLED\_SDIN(1);  
 OLED\_SCLK(1);  
#endif  
 OLED\_CS(1);  
 OLED\_RS(1);  
  
 OLED\_RST(0);  
 HAL\_Delay(100);  
 // delay\_ms(100);  
 OLED\_RST(1);  
  
 oled\_wr\_byte(0xAE, OLED\_CMD); /\* 关闭显示 \*/  
 oled\_wr\_byte(0xD5, OLED\_CMD); /\* 设置时钟分频因子,震荡频率 \*/  
 oled\_wr\_byte(80, OLED\_CMD); /\* [3:0],分频因子;[7:4],震荡频率 \*/  
 oled\_wr\_byte(0xA8, OLED\_CMD); /\* 设置驱动路数 \*/  
 oled\_wr\_byte(0X3F, OLED\_CMD); /\* 默认0X3F(1/64) \*/  
 oled\_wr\_byte(0xD3, OLED\_CMD); /\* 设置显示偏移 \*/  
 oled\_wr\_byte(0X00, OLED\_CMD); /\* 默认为0 \*/  
  
 oled\_wr\_byte(0x40, OLED\_CMD); /\* 设置显示开始行 [5:0],行数. \*/  
  
 oled\_wr\_byte(0x8D, OLED\_CMD); /\* 电荷泵设置 \*/  
 oled\_wr\_byte(0x14, OLED\_CMD); /\* bit2，开启/关闭 \*/  
 oled\_wr\_byte(0x20, OLED\_CMD); /\* 设置内存地址模式 \*/  
 oled\_wr\_byte(0x02, OLED\_CMD); /\* [1:0],00，列地址模式;01，行地址模式;10,页地址模式;默认10; \*/  
 oled\_wr\_byte(0xA1, OLED\_CMD); /\* 段重定义设置,bit0:0,0->0;1,0->127; \*/  
 oled\_wr\_byte(0xC8, OLED\_CMD); /\* 设置COM扫描方向;bit3:0,普通模式;1,重定义模式 COM[N-1]->COM0;N:驱动路数 \*/  
 oled\_wr\_byte(0xDA, OLED\_CMD); /\* 设置COM硬件引脚配置 \*/  
 oled\_wr\_byte(0x12, OLED\_CMD); /\* [5:4]配置 \*/  
  
 oled\_wr\_byte(0x81, OLED\_CMD); /\* 对比度设置 \*/  
 oled\_wr\_byte(0xEF, OLED\_CMD); /\* 1~255;默认0X7F (亮度设置,越大越亮) \*/  
 oled\_wr\_byte(0xD9, OLED\_CMD); /\* 设置预充电周期 \*/  
 oled\_wr\_byte(0xf1, OLED\_CMD); /\* [3:0],PHASE 1;[7:4],PHASE 2; \*/  
 oled\_wr\_byte(0xDB, OLED\_CMD); /\* 设置VCOMH 电压倍率 \*/  
 oled\_wr\_byte(0x30, OLED\_CMD); /\* [6:4] 000,0.65\*vcc;001,0.77\*vcc;011,0.83\*vcc; \*/  
  
 oled\_wr\_byte(0xA4, OLED\_CMD); /\* 全局显示开启;bit0:1,开启;0,关闭;(白屏/黑屏) \*/  
 oled\_wr\_byte(0xA6, OLED\_CMD); /\* 设置显示方式;bit0:1,反相显示;0,正常显示 \*/  
 oled\_wr\_byte(0xAF, OLED\_CMD); /\* 开启显示 \*/  
 oled\_clear();  
}

* 1. 主程序的编写

主程序主要做的事情是读取传感器的数据，做一些简单的处理，然后将数据显示在 oled 屏幕上。

主程序如下

Core/Src/main.c

#include "main.h"  
  
#include <stdio.h>  
#include "max30102\_for\_stm32\_hal.h"  
  
I2C\_HandleTypeDef hi2c1;  
  
UART\_HandleTypeDef huart2;  
  
char tx\_buffer[100];  
  
// printf() function  
int \_\_io\_putchar(int ch)  
{  
 HAL\_UART\_Transmit(&huart2, (uint8\_t \*)&ch, 1, 0xFFFF);  
 return ch;  
}  
  
// Override plot function  
void max30102\_plot(uint32\_t ir\_sample, uint32\_t red\_sample)  
{  
 sprintf(tx\_buffer, "ir:%lu,r:%lu\n", ir\_sample, red\_sample); // Print IR and Red  
 HAL\_UART\_Transmit(&huart2, tx\_buffer, strlen(tx\_buffer), HAL\_MAX\_DELAY);  
}  
  
// MAX30102 object  
max30102\_t max30102;  
  
/\* Private function prototypes -----------------------------------------------\*/  
void SystemClock\_Config(void);  
static void MX\_GPIO\_Init(void);  
static void MX\_I2C1\_Init(void);  
static void MX\_USART2\_UART\_Init(void);  
  
/\*\*  
 \* @brief The application entry point.  
 \* @retval int  
 \*/  
int main(void)  
{  
 /\* MCU Configuration--------------------------------------------------------\*/  
  
 /\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/  
 HAL\_Init();  
  
 /\* Configure the system clock \*/  
 SystemClock\_Config();  
  
 /\* Initialize all configured peripherals \*/  
 MX\_GPIO\_Init();  
 MX\_I2C1\_Init();  
 MX\_USART2\_UART\_Init();  
  
 oled\_init(); /\* 初始化OLED \*/  
 oled\_show\_string(0, 0, "ATGUIGU", 24);  
 oled\_show\_string(0, 24, "Avg BPM: 72", 16);  
 oled\_show\_string(0, 52, "SPO2: 98", 12);  
 oled\_refresh\_gram(); /\* 更新显示到OLED \*/  
  
 sprintf(tx\_buffer, "hello world");  
 HAL\_UART\_Transmit(&huart2, tx\_buffer, strlen(tx\_buffer), HAL\_MAX\_DELAY);  
  
 // Initiation  
 max30102\_init(&max30102, &hi2c1);  
 max30102\_reset(&max30102);  
 max30102\_clear\_fifo(&max30102);  
 max30102\_set\_fifo\_config(&max30102, max30102\_smp\_ave\_8, 1, 7);  
  
 // Sensor settings  
 max30102\_set\_led\_pulse\_width(&max30102, max30102\_pw\_16\_bit);  
 max30102\_set\_adc\_resolution(&max30102, max30102\_adc\_2048);  
 max30102\_set\_sampling\_rate(&max30102, max30102\_sr\_800);  
 max30102\_set\_led\_current\_1(&max30102, 6.2);  
 max30102\_set\_led\_current\_2(&max30102, 6.2);  
  
 // Enter SpO2 mode  
 max30102\_set\_mode(&max30102, max30102\_spo2);  
 max30102\_set\_a\_full(&max30102, 1);  
  
 // Initiate 1 temperature measurement  
 max30102\_set\_die\_temp\_en(&max30102, 1);  
 max30102\_set\_die\_temp\_rdy(&max30102, 1);  
  
 uint8\_t en\_reg[2] = {0};  
 max30102\_read(&max30102, 0x00, en\_reg, 1);  
  
 /\* Infinite loop \*/  
 /\* USER CODE BEGIN WHILE \*/  
 while (1)  
 {  
 if (max30102\_has\_interrupt(&max30102))  
 {  
 max30102\_interrupt\_handler(&max30102);  
 }  
 }  
 /\* USER CODE END 3 \*/  
}  
  
/\*\*  
 \* @brief System Clock Configuration  
 \* @retval None  
 \*/  
void SystemClock\_Config(void)  
{  
 RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};  
 RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};  
  
 /\*\* Initializes the RCC Oscillators according to the specified parameters  
 \* in the RCC\_OscInitTypeDef structure.  
 \*/  
 RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;  
 RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;  
 RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;  
 RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;  
 if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)  
 {  
 Error\_Handler();  
 }  
  
 /\*\* Initializes the CPU, AHB and APB buses clocks  
 \*/  
 RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK | RCC\_CLOCKTYPE\_SYSCLK | RCC\_CLOCKTYPE\_PCLK1 | RCC\_CLOCKTYPE\_PCLK2;  
 RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;  
 RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;  
 RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;  
 RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;  
  
 if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != HAL\_OK)  
 {  
 Error\_Handler();  
 }  
}  
  
/\*\*  
 \* @brief I2C1 Initialization Function  
 \* @param None  
 \* @retval None  
 \*/  
static void MX\_I2C1\_Init(void)  
{  
 hi2c1.Instance = I2C1;  
 hi2c1.Init.ClockSpeed = 400000;  
 hi2c1.Init.DutyCycle = I2C\_DUTYCYCLE\_2;  
 hi2c1.Init.OwnAddress1 = 0;  
 hi2c1.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;  
 hi2c1.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;  
 hi2c1.Init.OwnAddress2 = 0;  
 hi2c1.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;  
 hi2c1.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;  
 if (HAL\_I2C\_Init(&hi2c1) != HAL\_OK)  
 {  
 Error\_Handler();  
 }  
}  
  
/\*\*  
 \* @brief USART2 Initialization Function  
 \* @param None  
 \* @retval None  
 \*/  
static void MX\_USART2\_UART\_Init(void)  
{  
 huart2.Instance = USART2;  
 huart2.Init.BaudRate = 115200;  
 huart2.Init.WordLength = UART\_WORDLENGTH\_8B;  
 huart2.Init.StopBits = UART\_STOPBITS\_1;  
 huart2.Init.Parity = UART\_PARITY\_NONE;  
 huart2.Init.Mode = UART\_MODE\_TX\_RX;  
 huart2.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;  
 huart2.Init.OverSampling = UART\_OVERSAMPLING\_16;  
 if (HAL\_UART\_Init(&huart2) != HAL\_OK)  
 {  
 Error\_Handler();  
 }  
}  
  
/\*\*  
 \* @brief GPIO Initialization Function  
 \* @param None  
 \* @retval None  
 \*/  
static void MX\_GPIO\_Init(void)  
{  
 GPIO\_InitTypeDef GPIO\_InitStruct = {0};  
 /\* GPIO Ports Clock Enable \*/  
 \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();  
 \_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();  
  
 /\*Configure GPIO pin : PB0 \*/  
 GPIO\_InitStruct.Pin = GPIO\_PIN\_9;  
 GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_FALLING;  
 GPIO\_InitStruct.Pull = GPIO\_PULLUP;  
 HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);  
  
 /\* EXTI interrupt init\*/  
 HAL\_NVIC\_SetPriority(EXTI9\_5\_IRQn, 0, 0);  
 HAL\_NVIC\_EnableIRQ(EXTI9\_5\_IRQn);  
  
}  
  
/\*\*  
 \* @brief This function is executed in case of error occurrence.  
 \* @retval None  
 \*/  
void Error\_Handler(void)  
{  
 /\* USER CODE BEGIN Error\_Handler\_Debug \*/  
 /\* User can add his own implementation to report the HAL error return state \*/  
 \_\_disable\_irq();  
 while (1)  
 {  
 }  
 /\* USER CODE END Error\_Handler\_Debug \*/  
}  
  
#ifdef USE\_FULL\_ASSERT  
/\*\*  
 \* @brief Reports the name of the source file and the source line number  
 \* where the assert\_param error has occurred.  
 \* @param file: pointer to the source file name  
 \* @param line: assert\_param error line source number  
 \* @retval None  
 \*/  
void assert\_failed(uint8\_t \*file, uint32\_t line)  
{  
 /\* USER CODE BEGIN 6 \*/  
 /\* User can add his own implementation to report the file name and line number,  
 ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/  
 /\* USER CODE END 6 \*/  
}  
#endif /\* USE\_FULL\_ASSERT \*/