

从零开始使用 CubeMX 创建以太网工程

前言

在前面一篇文章中,介绍了如何使用 CubeMX 来建立一个简单的 TCPEchoserver 工程。但是在新建 CubeMX 项目时,是通 过直接选择 ST 的开发板的方式实现的。对于大多数实际的开发场景,可能并不是在 ST 的开发板上进行的,所以在这篇文章 中,我将介绍如何从零开始建立一个以太网工程。

今年 ST 推出的 Nucleo-144 板子上集成了以太网接口,所以在本文中,将以 STM32F746-Nucelo 板为例,通过 CubeMXv4.18 来新建一个 TCPEchoserver 的程序。

用 CubeMX 建立基于 STM32F746-Nucleo 的工程

用 CubeMX 进行初始化配置

这回我们直接选择 STM32F746-Nucleo 板上对应的芯片 STM32F746ZGT6U,而不是选择 STM32F746-Nucleo 板。

1. 新建一个 Project,在向导中选择 STM32F746ZGT6U。

New Project						N SOM	X			
MCU Selector Boar	MCU Selector Board Selector									
MCU Filters										
Series :				Lines :	Package :					
STM32F7				 STM32F7x6 	▼ LQFP144	 More Filters 	•			
Peripheral Selection	n			MCUs List: 3 Items						
Peripherals	Nb	Max		MCU	Lines	Package	E.			
ADC 12-bit	0	24		STM32F746ZETx	STM32F7x6	LQFP 144	^			
ADC 16-bit	0	0		STM32F746ZGTx	STM32F7x6	LQFP 144				
CAN	0	2		STM32F756ZGTx	STM32F7x6	LQFP 144				
COMP	0	0								
DAC 12-bit	0	2								
DCMI										
DFSDM	0	0								
DSIHOST										
Ethernet										

这个时候我们看到的还是一个空的工程。如下图:

STM32CubeMX Untitled: STM3.	2F746ZGTx				2
File Project Pinout Window He	lp				
💽 🦀 🖬 🖬 🖌 🦀 🖉	Keep Current Signals Placemen	t 🤊 (° <u>5</u> 🗕 🎯 🕂 Find	 ■ ■	Show user Label 🛛 🤨 🌵	
Pinout Clock Configuration Configu	uration Power Consumption Calculate	х			_
Configuration MiddleWares 0 6 ATTS 0 7 EERTOS 0 0 UNTP 0 0 USB_DEVICE 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0					
Series	lines	Mau	Parkage	Required Perinherals	
STM32F7	STM32F7x6	STM32E746ZETx	LOFP 144	None	
STM32F7	STM32F7x6	STM32F746ZGTx	LOFP144	None	-

- 2.外设使能,引脚配置
- 2.1 以太网外设引脚配置



Nucleo-144 板上用的 PHY LAN8742A, RMII 接口。在 Cubemx 中使能 ETH 外设,选择 RMII 接口。Cubemx 会自动配置对

应的以太网接口。如下图:



STM32 的很多引脚都有复用功能,同一个功能也可以 remap 到不同的引脚。所以这里要记得将 CubeMX 自动配置的引脚和 实际电路中使用的引脚进行对比,保证是一致的。

从 UM1974 中可以找到 Nucleo-144 板上以太网引脚分配表。对比这张表格和 CubeMX 的默认配置,会发现 PB11, PB12 引 脚在 STM32F746-Nucleo 板中没有用做以太网的接口,而是用作其他用途了。

STM32F746-Nucleo 上的引脚分配:

Pin name	Function	Conflict with ST Zio connector signal	Configuration when using Ethernet	Configuration when using ST Zio or ST morpho connector
PA1	RMII Reference Clock	-	SB13 ON	SB13 OFF
PA2	RMII MDIO	-	SB160 ON	SB160 OFF
PC1	RMII MDC	-	SB164 ON	SB164 OFF
PA7	RMII RX Data Valid	D11	JP6 ON	JP6 OFF
PC4	RMII RXD0	-	SB178 ON	SB178 OFF
PC5	RMII RXD1	-	SB181 ON	SB181 OFF
PG11	RMII TX Enable	-	SB183 ON	SB183 OFF

Pin name	Function	Conflict with ST Zio connector signal	Configuration when using Ethernet	Configuration when using ST Zio or ST morpho connector	
PG13	RXII TXD0	-	SB182 ON	SB182 OFF	
PB13	RMII TXD1	I2S_A_CK	JP7 ON	JP7 OFF	

CubeMX 的默认分配:

	Pin Name	Signal on Pin	GPIO o	GPIO	GPIO P	Maxim	
	PA1	ETH_REF_CLK	n/a	Alternat	No pull	Very High	Ē
	PA2	ETH_MDIO	n/a	Alternat	No pull	Very High	Γ
	PA7	ETH_CRS_DV	n/a	Alternat	No pull	Very High	Γ
ſ	PB11	ETH_TX_EN	n/a	Alternat	No pull	Very High	
	PB12	ETH_TXD0	n/a	Alternat	No pull	Very High	Γ
	PB13	ETH_TXD1	n/a	Alternat	No pull	Very High	Г
	PC1	ETH_MDC	n/a	Alternat	No pull	Very High	
	PC4	ETH_RXD0	n/a	Alternat	No pull	Very High	Γ
	PC5	ETH_RXD1	n/a	Alternat	No pull	Very High	Γ

在 Cubemx 中修改引脚配置:



修改方法见下图,用同样的方法配置 PG11 和 PG13。配置 PG11 和 PG13 后,对应 PB11 和 PB12 会自动清除之前的配置, 以免冲突。



到现在位置,已经将所有的 GPIO 口都配置好了。

在 Configuration 页面中,还可以看到所有配置的 GPIO。并可以做进一步的配置,这里就先用默认的设置。

STM32CubeMX Untitled*: STM32	F746ZGTx	-				Pin Configu	uration		·	_		— X
File Project Window Help						ETH						
🗈 🕒 🖬 🖬 🖓 🍊 🗐	+ - 🤨 🜩 🤹					Count Counts						
Pinout 😢 Clock Configuration Configuration Power Consumption Calculator							-F)				Show only	/ Modified Pins
Configuration			Middlewares			Pin Name	Signal on Pin	GPIO output I	GPIO mode	GPIO Pull-up/ Maximum out	User Label	Modified
User-defined					Þ	AZ	ETH_MDIO	n/a	Alternate Func	No pull-up and Very High		
					P	A7	ETH_CRS_DV	n/a	Alternate Func	No pull-up and Very High		E
Enabled					P	C1	ETH_ND1	n/a n/a	Alternate Func	No pull-up and Very High		
					P	C4	ETH_RXD0	n/a	Alternate Func	No pull-up and Very High		
- Enabled	Multimedia	Connectivity	Analog	System	P	C5	ETH_RXD1 ETH_TX_EN	n/a n/a	Alternate Func	No pull-up and Very High		
Peripherals					P	G13	ETH_TXD0	n/a	Alternate Func.	. No pull-up and Very High		
		ETH 🔪		CORTEX_M7 ARM.		1						
Activated												
📄 💿 DMA2D												
Activated												
ETH				GPIO								
Mode:RMII						PA1 Configura	tion :					
🖕 💿 IWDG				NVIC		GPIO mode			4	Viternate Function Push Pull		•
Civated				000								
e 💿 RNG				RUC		GPIO Pull-up/F	Pull-down		P	io pull-up and no pull-down		•
Activated	-		J						-			
				,		Maximum outp	out speed		N	lery High		
MCUs Selection Output						Liser Label						
Series	Lines	Mau	Package	Required Peripherals		000 0000						
SIM32-7 5	iM32F7x6	SIM32F7462G1x	LQHP144	None								
STM32F7	5TM32F7x6	STM32F756ZGTx	LQFP144	None		Group By F	Peripherals			Apply	Ok	Cancel

2.2 使能 LwIP 协议栈

在这个工程内,我们会用到 LwIP 协议栈,所以还需要在这一页的 Middlewares 部分将 LWIP 勾选上。之后就可以在 Configuration 页面对 LWIP 协议栈进行配置了。





3.时钟配置

接下来进行时钟配置。CubeMX 默认系统时钟 16MHz,但以太网外设需要至少 25MHz 的系统时钟,所以这里会看到 Clock Configuration 页面显示 X

打开 Clock Configuration 页面会自动跳出一个提示框,可以选择让 CubeMX 来帮你自动调整时钟配置,也可以自己手动进行 调整。这里,我选择让 CubeMX 自动配置, CubeMX 会自动将时钟配成 216MHz。



4. 配置以太网参数

STM32CubeMX Untitled*: STI	M32F1	746ZGTx						
File [Project] Window Help								
🖪 🖻 🖩 🖯 🗛 📥	¢ į .	🛏 🕘 🧈 🦆						
Pinout Clock Configuration Conf	figurati	on Power Consumption Calculator						
Configuration - MiddleWares - O FATES	Î	(1)		Middlewares				
User-defined								
📰 Enabled								
Enabled	=	Multimedia	Connectivity	Analog	System			
Peripherals CC Activated DHA2D Activated EIH Mode:RMII NOde:RMII Activated Activated			2					
RNG Activated	+							
	P	· · · · · · · · · · · · · · · · · · ·			•			

在 Parameter Settings 页面,可以配置 MAC 地址,PHY 的地址,是否进行自动协商等。

这里,我们设置了 MAC 地址为本地地址 02:00:0:00:00:00。LAN8742a 的 PHY 地址由上电时 PHYAD0 的状态决定。根据 STM32F746-Nucleo 板的原理图,设置 PHY 地址为 0。

ETH Configuration		×
User Constants	s NVIC Settings	GPIO Settings anced Parameters
Configure the below parameters :		
Search : Search (Crtl+F)	• •	
Advanced : Ethernet Media Cor	figuration	
Auto Negotiation	Enabled	
Ethernet MAC Address	02:00:00:00:00:00	
PHY Address	0	
Ethernet Basic Configuration		
TX IP Header Checksum C	Polling Mode	
	of the other of	



接收数据的模式有轮询和中断两种方式,中断方式需要和操作系统一起使用,这里我们没有使用任何操作系统,所以在 RX Mode 这一项只能选择 Polling Mode。

最后一项是"TX IP Header Checksum Computation",STM32 的 MAC 控制器可以在发送数据时自动添加 IP 数据报的 checksum,如果需要这项功能,就将这一项设置为"By hardware"

在 Advanced Parameters 页,可以根据所用的 PHY 修改寄存器的地址和一些 MASK 的设置。因为 STM32F746 的两款开发 板上用的都是 PHY LAN8742A,所以 CubeMX 中默认的配置是以 LAN8742A 为例进行设置的。所以这里,我们不需要做任 何修改就可以直接用。但如果是其他的 PHY,可以在 PHY 这一项选择"user PHY",然后根据所用 PHY 的数据手册,配置下 面的参数,对于部分无法通过 CubeMX 进行配置的参数,需要手动的修改代码。将有冲突的地方删除,或者添加某个功能。 Advanced Parameters 页分为三个部分:

- External PHY Configuration 。复位延时,读/写超时的参数设置
- Common :External PHY Configuration。PHY 的基础寄存器配置,这部分寄存器对于大部分 PHY 都是相同或类似的。
- Extended :External PHY Configuration。 PHY 的扩展寄存器配置,这部分对于每个 PHY 都是不一样的。如果是使用非 CubeMX 默认的 PHY,这部分内容需要特别注意。

ETH Configuration	
Parameter Settings 🗹 Advanced Parameters 🗹 User Const	ants 🖋 NVIC Settings 🖋 GPIO Settings
Configure the below parameters :	
Search : Search (Crtl+F)	
External PHY Configuration	
PHY	LAN8742A_PHY_ADDRESS
PHY Address Value	0
PHY Reset delay these values are based on a 1 ms Systick .	0x000000FF
PHY Configuration delay	0x00000FFF
PHY Read TimeOut	0x0000FFFF
PHY Write TimeOut	0x0000FFFF
Common : External PHY Configuration	
Transceiver Basic Control Register	0x00
Transceiver Basic Status Register	0x01
PHY Reset	0x8000
Select loop-back mode	0x4000
Set the full-duplex mode at 100 Mb/s	0x2100
Set the half-duplex mode at 100 Mb/s	0x2000
Set the full-duplex mode at 10 Mb/s	0x0100
Set the half-duplex mode at 10 Mb/s	0x0000
Enable auto-negotiation function	0x1000
Restart auto-negotiation function	0x0200
Select the power down mode	0x0800
Isolate PHY from MII	0x0400
Auto-Negotiation process completed	0x0020
Valid link established	0x0004
Jabber condition detected	0x0002
Extended : External PHY Configuration	
PHY special control/status register Offset	0x10
PHY Speed mask	0x0002
PHY Duplex mask	0x0004
 * PHY Interrupt Source Flag register Offset 	0x000B
* PHY Link down inturrupt	0x000B

4. 修改 LWIP 的参数

配置好以太网的参数后,点击 OK,回到 CubeMX 的配置界面。选择 LWIP 继续进行参数配置。





首先是 GeneralSettings 页面,在这里我们可以看到 LWIP 的版本号。配置 IP 地址信息,可以选择通过 DHCP 的方式动态分 配 IP,也可以分配一个静态的 IP 地址。这里,我们选择配置静态的 IP 地址 192.168.0.10,子网掩码 255.255.255.0,网关 192.168.0.1。ICMP 协议打开,因为我们用的是 TCP 协议,所以把 UDP 协议关掉。

不用担心不知道每项参数是做什么用的,选择每一项参数后都会在窗口的底部显示该项参数的解释。



在 Key Options 这一页里,有更多的参数可以配置。关于接收/发送内存的配置也是在这里。选择右上方的"Show Advanced Parameters"后,还有更多的参数配置项。这里,我们也可以不做修改,使用默认值。CubeMX 中每个参数项的名称和代码中的名称相同,这样也方便了在代码中进行查找。



LWIP Configuration	X
Perf/Checks Statistics Check General Settings Key Options	Isum 🖌 Debug 🖋 User Constants IPv6 🖋 HTTPD 🖋 SNMP 🖋 SNTP
Configure the below parameters : Search : Search (Crtl+F)	Show Advanced Parameters
Platform Specific Locking	A
SYS_LIGHTWEIGHT_PROT (Memo Disa	bled
NO_SYS (LwIP Facilities) LwI	P Facilities Disabled
NO_SYS_NO_TIMERS (Drop Supp Disa	bled
Memory Options	
MEM_SIZE (Heap Memory Size) 160	D Byte(s)
 Internal Memory Pool Sizes 	
MEMP_NUM_PBUF (Number of Me 16	E
MEMP_NUM_RAW_PCB (Number o 4	
MEMP_NUM_TCP_PCB_LISTEN (N 8	
MEMP_NUM_TCP_SEG (Number of 16	
MEMP_NUM_LOCALHOSTLIST (Nu 1	
Pbuf Options	
PBUF_POOL_SIZE (Number of Buf 16	
PBUF_POOL_BUFSIZE (Size of eac 592	Byte(s)
ARP Options	
LWIP_ARP (ARP Functionality) Ena	bled
TCP Options	
TCP_TTL (Number of Time-To-Live 255	Node(s)
TCP_WND (TCP Receive Window 214	4 Byte(s)
TCP_QUEUE_OOSEQ (Allow Out Ena	bled 👻

到此为止,我们在 CubeMX 中需要做的配置就全部完成了。选择 Project——>Generate Code,生成初始的工程。

))ect Code Generator Adv	vanced Settings
Project Settings	
Project Name	
TEST_F746	
Project Location	
C:\Users\amanda shen\Des	ktop\ Browse
Toolchain Folder Location	
C:\Users\amanda shen\Des	ktop\TEST_F746\
Toolchain / IDE	
Ewapa	- Correcto Under Dant
Linker Settings	
Linker Settings Minimum Heap Size	0×200
Linker Settings Minimum Heap Size Minimum Stack Size	0x200
Linker Settings Minimum Heap Size Minimum Stack Size	0x200 0x400
Linker Settings Minimum Heap Size Minimum Stack Size	0x200 0x400
Linker Settings Minimum Heap Size Minimum Stack Size Mcu and Firmware Package	0×200 0×400
Linker Settings Minimum Heap Size Minimum Stack Size Mcu and Firmware Package Mcu Reference	0x200 0x400
Linker Settings Minimum Heap Size Minimum Stack Size Mou and Firmware Package Mou Reference STM32F7462/GTx	0x200 0x400
Linker Settings Minimum Heap Size Minimum Stack Size Mcu and Firmware Package Mcu Reference STM32F746ZGTx	0x200 0x400
Linker Settings Minimum Heap Size Minimum Stack Size Mcu and Firmware Package Mcu Reference STM32F7462GTx Firmware Package Name and	0x200
Linker Settings Minimum Heap Size Minimum Stack Size Mcu and Firmware Package Mcu Reference STM32F7462GTx Firmware Package Name and STM32Cube FW_F771.5.1	0x200 0x400 d Version
Linker Settings Minimum Heap Size Minimum Stack Size Mcu and Firmware Package Mcu Reference STM32F7462GTx Firmware Package Name and STM32Cube FW_F7 V1.5.1	0x200 0x400 d Version

添加用户代码

用 IAR 打开前面已经生成好的工程。我们还需要两步就可以完成一个简单的 TCP EchoServer 程序了。

1.新建 tcp_echoserver.c 文件,在 tcp_echoserver.c 里要做下面这几件事情:

- 1) 新建一个 tcp_echoserver_pcb(调用 tcp_new 函数);
- 2) 将新建的 tcp_echoserver_pcb 与要监听的端口绑定(调用 tcp_bind 函数)
- 3)转成监听状态(调用 tcp_listen 函数)
- 4) 注册回调函数 tcp_echoserver_accept,当有新连接建立后会调用该函数(调用 tcp_accept 函数)
- 5) 注册回调函数 tcp_echoserver_recv,当该连接接收到数据后会调用该函数(调用 tcp_recv 函数)
- 6) 完成 tcp_echoserver_recv 函数,在该函数内,将收到的数据再发出去。



需要注意,本文的目的是示例如何用 CubeMX 建立一个简单的 TCP EchoServer 程序,所以考虑的都是最基本简单的情况。 比如,在回发数据部分,我们假设 Client 发来的数据都在一个 Pbuf 的大小以内。 完成 tcp_echoserver.c 后,将其加入到工程项目中。

```
#include "stats.h"
#include "tcp.h"
void tcp_echoserver_init(void);
static err_t tcp_echoserver_accept(void *arg, struct tcp_pcb *newpcb,err_t err);
static err t tcp echoserver recv(void *arg, struct tcp pcb *tpcb, struct pbuf *p,err t
err);
static struct tcp_pcb *tcp_echoserver_pcb;
void tcp_echoserver_init(void)
  err_t err;
 tcp_echoserver_pcb = tcp_new();
  if(tcp_echoserver_pcb !=NULL)
    err = tcp_bind(tcp_echoserver_pcb, IP_ADDR_ANY, 7);
    if(err == ERR OK)
     tcp_echoserver_pcb = tcp_listen(tcp_echoserver_pcb);
      tcp_accept(tcp_echoserver_pcb,tcp_echoserver_accept);
    else
      memp_free(MEMP_TCP_PCB, tcp_echoserver_pcb);
    }
static err_t tcp_echoserver_accept(void *arg, struct tcp_pcb *newpcb,err_t err)
    /* initialize lwip tcp_recv callback function for newpcb */
    tcp recv(newpcb, tcp echoserver recv);
    return ERR_OK;
static err_t tcp_echoserver_recv(void *arg, struct tcp_pcb *tpcb, struct pbuf *p,err_t
err)
  tcp_write(tpcb,p->payload,p->len,1);
```

pbuf free(p);



return ERR_OK;

2. 在 main 函数中添加 tcp_echoserver_init(),在 while(1)中添加 MX_LWIP_Process()查询接收数据。记得要将代码加 在/*USER CODE BEGIN*/和/*USER CODE END*/之间,这样才不会在下次用 CubeMX 生成代码时被覆盖掉。

```
extern void tcp_echoserver_init(void);
int main(void)
{
    MX_LWIP_Init();
    /* USER CODE BEGIN 2 */
    tcp_echoserver_init();
    /* USER CODE END 2 */
    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
    /* USER CODE BEGIN 3 */
    /* USER CODE BEGIN 3 */
    MX_LWIP_Process();
    }
    /* USER CODE END 3 */
}
```

一个简单的 TCP Echoserver 程序就完成了。

测试结果

我们来看一看 ping 测试和 TCP 测试工具的结果

1. 通过电脑(192.168.0.11) ping STM32F746-Nucleo 板(192.168.0.10)





2.通过 TCP 测试工具模拟客户端,向 STM32F746-Nucleo 板发一串数据。

🎪 TCP&UDP测试工具 - [192.168.0.	10:7]		
: Operate (0) View (V) Windows (W) Hel	.p (H) Language		×
🔚 CreateConn 🗞 CreateServer 🛛 🔉 Ste	ertServer 淃 🐼 😤 Co	nnect 🐲 🛬 DisconnAll 🔀 DeleteConn 💸 🔟 😹 🖕	
Properties 🛛 🕂 🗙	192. 168. 0. 10:	7	4 Þ 🗙
⊖ Client Mode ► 192 158, 0, 10:7 Server Mode	Internet Internet	Send AtuoSend Eve DO as Send Stop Send Hex Send File Send Received Clear Option 12345676910123457676910123457676910123457676910123457676910123457676910123457769101234577691012345776910123457769101234577691012345776910123457769101234577691012345776910123457769101234577691012345776910123457769101234577691012345776910123457769101234577691012345776910123457769100000000000000000000000000000000000	BroadOpt: 9910123456
	Send 99	Save (In Time)	
	Recv 99 Clear	12345670910123456789101234567891012345678910123456789101234567891012345678910123456789101234567891012345678910	910123456

测试结果说明我们刚刚建立的 TCP EchoServer 程序已经能正常工作了。



重要通知 - 请仔细阅读

意法半导体公司及其子公司("ST")保留随时对ST产品和/或本文档进行变更、更正、增强、修改和改进的权利,恕不另行通知。买方在订货之前应获取关于ST产品的最新信息。ST产品的销售依照订单确认时的相关ST销售条款。

买方自行负责对ST 产品的选择和使用, ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的ST 产品如有不同于此处提供的信息的规定,将导致ST 针对该产品授予的任何保证失效。

ST 和ST 徽标是ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。

© 2015 STMicroelectronics - 保留所有权利