## 教你一步步使用 KEIL-MDK 开发 XMC1300

目前,手里有一块 XMC1302-Boot Kit 开发板,该开发板采用 XMC1302-038-200MCU, 是 XMC1300 系列中最高级的型号,外设资源丰富,采用 M0 内核,并有辅助浮点处理。 官网使用手册和例程中,大部分都是采用 DAVE 平台来讲解的,采用 KEI-MDK 环境的资料 非常少,下面就用此板子作为讲解平台,以一个翻转板上 6 路 LED 灯为 DEMO 一步步讲解 如何建立 MDK 工程,及编写外设底层驱动的过程!硬件平台的外观如下,板载一个 J-LINK 仿真器,采用 XMC4200MCU 实现,如果你有这个板子,可以对照着实践一下:



首先打开 KEIL-MDK4,这里用的是 V4.72A 版本,这个版本是带有 XMC1300 系列型号的, 采用 MDK5 版本也可以,需要安装 XMC1300 的补丁包。



建立新工程,XMC1302-LED,然后选项 MCU 型号

Select Device for Target 'Target 1'
CPU Vendor: Infineon Device: XMC1302-200 Toolset: ARM
Description: De
OK Cancel Help

在这个版本里面,选择 XMC1302-200 这个型号,点击确定后,KEIL 会自动增加 startup\_XMC1300.s 的启动文件。



关闭 KEIL-MDK 工程,返回到建立工程的文件夹,在文件夹下面建立如下文件夹,并增加文件。



其中 driver 放置底层外设驱动,lib 放置 XMC1300 的库文件,有些需要自己实现。 MDK 放置工程文件,startup 放置启动文件.s,user 文件夹放置主函数和预编译文件, XMC1302 放置必要的系统文件,比如 XMC1300.h 文件,时钟配置文件,中断文件,如图:



Clean.bat 是一个批处理命令,用于清理工程编译产生的额外文件,要不会占用很多空间。 建立好工程文件夹,在 MDK 文件夹里面,找到工程文件名,再打开,之前,把刚才建 立的工程文件已经放置到里面了。在工程里面,修改 MCU 配置,组名称,并建立一个组为 user,吧 main.c、xmc1300\_it.c和 system\_XMC1300.c 加入,其中 system\_XMC1300.c 文件在 MDK 里面自带,可以从安装文件里面找到,xmc1300\_it.c 中断文件需要自己实现,这个根据 启动文件名来实现一下,不难

Eile Edit View Project Flash Debug Periphera	als Iools SVCS Window Help	
	🕨 🥐 隐 隐 諱 淳 /// /// 🞯 PO_O_toggle(); 🛛 🛛 📓 🥔 🔍 🖕 🔿 🔗 👧 🗐 🗸 🔦	
😻 🔛 🕮 🧼 🤐 🙀 XMC1302-LED	🖻 🐔 🗟	
Project 🥑 🗵 🚺 startu	p_XMC1300.s 📔 🔝 startup_XMC1300.s 🗡 🖹 Main.c	<b>▼</b> ×
→         16           →         14 (true)           →	IS SOFTWARE IS PROVIDED BY THE COPYDIGHT HOLDERS AND CONTRIBUTORS "AS 13" ***********************************	
37 **	**	~
■ Pr 《B   U F   U→T		2
Build Output		<u>g</u> ×
4		<u>~</u> (>

建立好文件组和添加文件后,就需要对 MDK 的工程进行配置,配置后才能顺利实现程序下载。



在函数中,因为要实现翻转 LED 灯,需要用到 GPIO 函数,这里使用自带的宏定义.h 文件。不设置工程路径,KEIL 会提示错误。

	log(bg(   Fise	Ing   Oser		72111   J	STICKET	pendg   ocir:	i cres	
nfineon XMC130	2-200	_		Code (	ieneration	1		
		⊻tal (MHz): 8	3.0					
)perating system	None		•	ΓU	se Cross-M	Module Optimiza	ition	
ystem-Viewer Fi	e (.Sfr):			ΓU	se MicroL	IB L	Bjg Endian	
SFD\Infineon\X	MC1300\xmc130	0.SFR						
Read/Only Me	mory Areas			-Read/	Write Men	nory Areas		
default off-chip	o Start	Size	Startup	default	off-chip	Start	Size	Nolnit
□ ROM1	:		C		RAM1:			
□ ROM2	:		- C		RAM2:			
П ВОМЗ	:	í –	- c	Г	RAM3:		<u> </u>	
on-chi	p	1			on-chip	1	1	
IROM1	0x10001000	0x32000	•		IBAM1:	0x20000000	0x4000	
IROM2		-	- C	Г	IRAM2:			- F
	1					,	1	

在 KEIL 工程选项中,首先修改晶体频率,开发板采用的是内部晶振,8MHZ,原来默认 12MHZ。

🖏 Options for Target 'XMC1302-LED'	×
Device Target Output Listing User   C/C++   Asm   Linker   Debug   Utilities	
Select Folder for Objects <u>Name of Executable</u> : XMC1302-LED	
Image: Create Executable:       .\XMC1302-LED         Image: Debug Information       Image: Create Batch File         Image: Create HEX File       Image: Create HEX File	
OK Cancel Defaults Help	

在输出设置里面,选择创建 hex 文件选项,方便通过其它方式下载

🖫 Options for Target 'XMC1302-LED'
Device   Target   Output   Listing   User   C/C++   Asm   Linker   Debug   Utilities
Preprocessor Symbols           Define:
Undefine:
Strict ANSI C       Warnings:         Optimization:       Level 0 (-00)         Doptimize for Time       Enum Container always int         Plain Char is Signed       Thumb Mode         Split Load and Store Multiple       Read-Only Position Independent         One ELF Section per Function       Read-Write Position Independent
Include Paths Misc Controls Compiler control string -ccpu Cortex-M0 -g -00apcs=interwork -1 d:\Keil\ARM\RV31\INC
OK Cancel Defaults Help

在【C++】选项卡里面,增加包含的工作路径,设置好的路径如下

Folder Setup 🤶 🏹
Setup Compiler Include Paths:
\xmc1302 \xsratup \user \lib \lib\inc \lib\src \driver
OK Cancel

最重要的是,设置调试环境,选择 J-LINK/J-Trace Cortex 仿真器,默认是 ULINK2,点击 【Settings】

🕱 Options for Target 'XMC1302-LED'	
Device   Target   Output   Listing   User   C/C++   J	Asm Linker Debug Vtilities
Use Simulator     Settings     Settings	C ∐se: J-LINK / J-Trace Cortex ▼ Settings
✓ Load Application at Startup ✓ Run to main() Initialization File:	Load Application at Startup E Run to main() Initialization File:
Edit	Edit
Restore Debug Session Settings	Restore Debug Session Settings
I Breakpoints I Toolbox	Breakpoints     Toolbox
✓ Watch Windows & Performance Analyzer	✓ Watch Windows
Memory Display	I✓ Memory Display
CPU DLL: Parameter:	Driver DLL: Parameter:
SARMCM3.DLL	SARMCM3.DLL
Dialog DLL: Parameter: DARMCM1.DLL -pCM0	Dialog DLL: Parameter: TARMCM1.DLL -pCM0
OK Car	ucel Defaults Help

这里系统已经识别到了仿真器型号,可以看到 J-LINK 的详细信息,默认是 JTAG 方式,但是没有发现 MCU 内核。这个是很多初学者容易迷惑的地方,其实 XMC1302 开发板的 J-LINK 仅支持 SW 方式,不支持 JTAG 方式,所以,需要修改端口为【SW】

Cortex JLink/JTrace Target Driver	Setup 🔀
Debug Trace Flash Download	
J-Link / J-Trace Adapter SN: 591003473 Device: J-Link Lite-XMC4200 Rev.1 HW : V1.00 dll: V4.64 FW : J-Link Lite-XMC4200 Rev.1 c Port: Max Clock: JTAG  Auto Clk Auto Clk JTAG De	vice Chain
Connect & Reset Options Connect: Normal  Reset: Normal Reset after Connect	Cache Options  Cache Code  Cache Code  Cache Memory  Download Options  Download Options  Download to Flash
Interface USB C TCP/IP Scan State: ready	Port (Auto: 0) Autodetect JLink Info JLink Cmd
0K	Cancel Help

修改后, J-LINK 仿真器就识别到了

Cortex JLink/JTrace Target D	priver Setup 🛛 🔀
Debug Trace Flash Download	
J-Link / J-Trace Adapter	SW Device
SN: 591003479	IDCODE Device Name Move
Device: J-Link Lite-XMC4200 Rev.1	SWD Ox0BB11477 ARM CoreSight SW-DP
HW: V1.00 dll: V4.64	Down
FW : J-Link Lite-XMC4200 Rev.1 c Port: Max Clock: 2MHz Auto Clk	Automatic Detection ID CODE:     Manual Configuration Device Name:     Add Delete Update IR Ien:
Connect & Reset Options Connect: Normal Reset: Normal Rese	Trail     ✓     Cache Options     Download Options       Imal     ✓     Cache Code     ✓     Verify Code Download       Imal     ✓     Cache Memory     ✓     Download to Flash
Interface TCP/IP USB C TCP/IP Scan State: ready	ettings Port (Auto: 0) O . O . 1 : O Ping Misc JLink Info JLink Cmd
	OK Cancel Help

如果仿真器界面出来这个信息,恭喜,你的板子硬件连接是成功的! 下面需要增加下载算法了,不增加下载算法,KEIL 会给出 DLL 错误的提示

Cortex JLink/JIrace Tar	get Driver S	etup		
Debug   Trace Flash Downloa	d			
Download Function C Erase Full Chip C Erase Sectors C Do not Erase	<ul> <li>Program</li> <li>Verify</li> <li>Reset and Run</li> </ul>	BAM for A	Algorithm 0x20000000 Size: 0x0800	
Programming Algorithm				
		Start:	Size:	
	Add	Remove		
	ОК	Cane	el	Help

Add Flash Programmin	g Algorithm		×
Description	Device Type	Device Size	>
TMS570LS_F021 256KB Flash TMS570LS_F021 384KB Flash TSX1001 Code FEPBOM	On-chip Flash On-chip Flash On-chip Flash	256К 384k 32k	
XMC1100 64kB Flash XMC1200 200kB Flash	On-chip Flash On-chip Flash On-chip Flash	64k 200k	
XMC1300 200kB Flash XMC4200/4100 128kB cach	On-chip Flash On-chip Flash	200k 128k	
XMC4200/4100 256kB cach XMC4200/4100 64kB cache XMC4200/4100 120kB Electric	On-chip Flash On-chip Flash On-chip Flash	256k 64k 120k	
XMC4200/4100 1266kB Flash XMC4200/4100 256kB Flash XMC4200/4100 64kB Flash	On-chip Flash On-chip Flash On-chip Flash	256k 64k	
XMC4400 256kB cached Fla XMC4400 512kB cached Fla	On-chip Flash On-chip Flash	256k 512k	
XMC4400 256kB Flash XMC4400 512kB Flash	On-chip Flash On-chip Flash	256k 512k	-
Add	Cancel		

增加后,在下载功能中,勾选【复位并运行】

Cortex JLink/JIrace Target Driver Setup
Debug Trace Flash Download
Download Function       C Erase Full Chip       ✓ Program         C Erase Sectors       ✓ Verify       Start:       0x20000000       Size:       0x0800         C Do not Erase       ✓ Reset and Run       ✓       Finite Start:       0x20000000       Size:       0x0800
Programming Algorithm
Description Device Type Device Size Address Range
Start: 0x10001000 Size: 0x00032000
Add Remove
OK Cancel Help

除此之外,还需要在【Utilities】中,设置一下仿真器,和【Debug】的设置对应

## 找到该选项卡,点击【ADD】增加下载算法

🕱 Options for Target 'XMC1302-LED'
Device   Target   Output   Listing   User   C/C++   Asm   Linker   Debug   Utilities
Configure Flash Menu Command
Use Target Driver for Flash Programming
J-LINK / J-Trace Cortex Settings V Update Target before Debugging
Init File: Edit
C Use External Tool for Flash Programming
Command:
Arguments:
🔲 Run Independent
OK Cancel Defaults Help

这样设置后, 仿真和下载程序环境就设置好了。

在主函数 main.c 里面实现如下

前面是时钟配置,设置为8MHZ,内部时钟,中间是将GPIO 配置为推挽输出,再针对 systick 定时器进行设置,定时 10ms 的基准

```
//----CLOCK-SETUP------
SCU GENERAL->PASSWD = 0x000000COUL;
SCU_CLK->CLKCR = 0x3FF00400UL; // 8 MHz MCLK, 8 MHz PCLK
while((SCU_CLK->CLKCR) &0x40000000UL); // wait for VDDC to stabilize
SCU GENERAL->PASSWD = 0x000000C3UL;
//-----
                  _____
//----PIN-SETUP------
                       _____
PO_O_set_mode(OUTPUT_PP_GP);
P0_1_set_mode(OUTPUT_PP_GP);
P0_6_set_mode(OUTPUT_PP_GP);
P0_7_set_mode(OUTPUT_PP_GP);
P0_8_set_mode(OUTPUT_PP_GP);
P0_9_set_mode(OUTPUT_PP_GP);
1/---
SystemCoreClockUpdate();
SysTick_Config(SystemCoreClock / 10); // 0.1s interrupt base
//---
                                                   _____
                           _____
```

在中断里面, 计数器, 每计数到 5, 也就是 50ms 定时, 翻转 6 个 GPIO

```
=/**
   * @brief This function handles SysTick Handle
   * @param None
   * @retval None
   */
 void SysTick Handler(void)
- {
       systick_num ++;
       if(systick_num==5)
       { PO_0_toggle();
         PO_1_toggle();
         PO_6_toggle();
         PO 7 toggle();
         PO 8 toggle();
         PO 9 toggle();
         systick_num=0;
       }
 }
```

将程序编译一下,在 MDK 里面经常会出现包含文件的错误提示,这个对程序不影响,可以忽略



经过上面的设置,一个可以运行的工程环境就建立起来了。 把程序下载一下



FLASH 被擦除,程序运行了,运行现象:



在熟悉 KEIL-MDK 建立工程环境和设置仿真器后,可以在上面增加需要的底层驱动。和相关的库文件,这里可以结合 DAVE 产生的驱动文件源码来实现!

Fengye5340