

USR-GM3 SDK 使用说明

文件版本: V1.1



目录

USR-GM3 SDK 使用说明	1
1.简介.....	4
2.安装开发环境.....	4
2.1.安装 SDK.....	4
2.2.配置代码路径.....	15
3.编译工程.....	16
4.烧录调试.....	18
4.1.烧录程序.....	18
4.2.调试程序.....	20
5. API 函数	23
5.1.程序开发说明.....	23
5.2.函数说明.....	23
5.2.1.1. Usr_OpenUart	23
5.2.1.2. Usr_CloseUart.....	24
5.2.1.3. Usr_SendUartData	24
5.2.1.4. Usr_RegisterUartRxCallback	24
5.2.2.1. Usr_OpenSock	25
5.2.2.2. Usr_CloseSock.....	25
5.2.2.3. Usr_SetApn	26
5.2.2.4. Usr_SetSockParam.....	26
5.2.2.5. Usr_SendSockData.....	26
5.2.2.6. Usr_RegisterSockRxCallback	27
5.2.2.7. Usr_GetSockStatus.....	27
5.2.3.1. Usr_SendSmsData.....	28
5.2.3.2. Usr_SetSmsDest.....	28
5.2.3.3. Usr_RegisterSmsRxCallback	28
5.2.4.1. Usr_StartTimer	29
5.2.4.2. Usr_StopTimer	29
5.2.4.3. Usr_SetTimer	29
5.2.5.1. Usr_SendAtCmd	30
5.2.5.2. Usr_RegisterCmdRxCallback	30
5.2.7.1. Usr_SetEcho.....	31
5.2.7.2. Usr_Restart.....	31
5.2.7.3. Usr_ReloadUserDefault	31
5.2.7.4. Usr_ReloadUsrFactory	32
5.2.7.5. Usr_SaveCurrentSetting.....	32
5.2.7.6. Usr_SaveAsUserDefault	32
5.2.7.7. Usr_EnableRFC	32
5.2.7.8. Usr_EnableUartCMD.....	33
5.2.7.9. Usr_EnableNetCMD	33
5.2.7.10. Usr_GetIMEI.....	33

5.2.8.1. GPIO 定义	34
5.2.8.2. Usr_GpioInit	34
5.2.8.3. Usr_SetGpio	34
5.2.8.4. Usr_GetGpio	35
5.2.9.1. Usr_GetSysTime	35
5.2.9.2. Usr_SetSysTime	35
6.联系方式.....	37
7.免责声明.....	37
8.更新历史.....	37

1. 简介

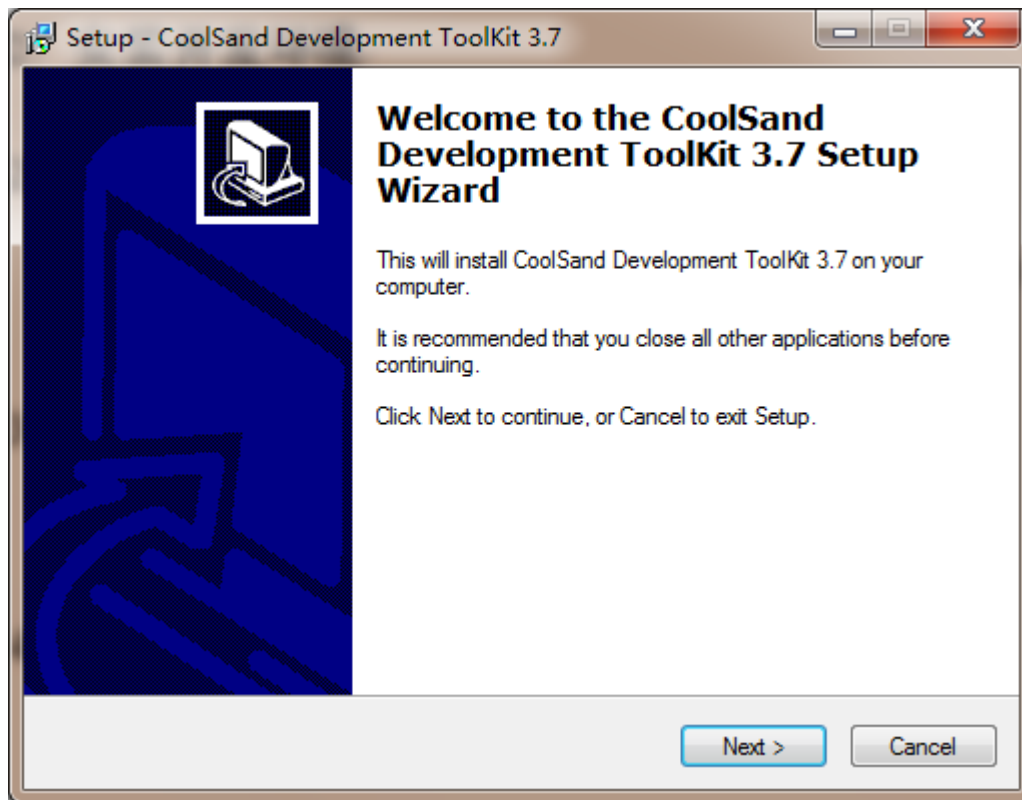
为方便用户深度使用我公司 GM3 系列产品，特意开放一些常用功能作为接口，让用户在可允许的范围内自定义编程。开放的接口包括：1 路 UART 接口，5 路定时器接口，2 路 Socket 接口，短信收发接口，系统 AT 指令接口，自定义 AT 指令接口，RTC 时钟接口，10 路复用 GPIO 接口和部分系统相关接口。用户可以根据以上接口完成自定义功能的开发，更加灵活的运用到各种场景当中。

2. 安装开发环境

2.1. 安装 SDK

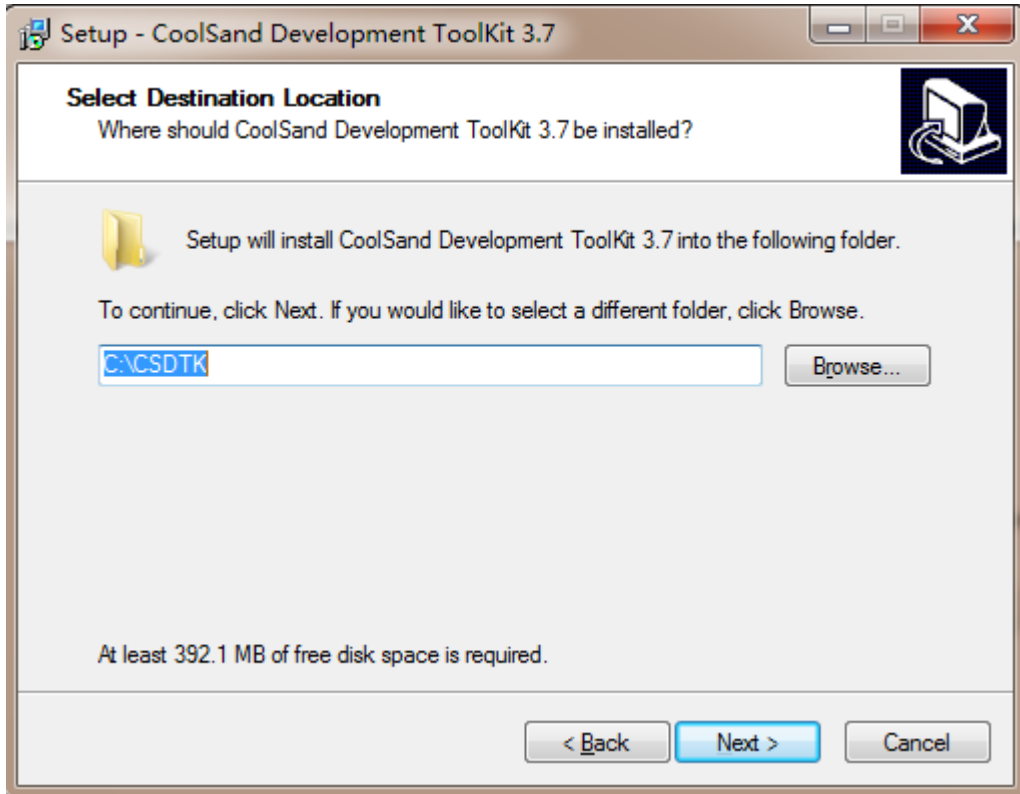
- (1) 双击 CSDTK3.7_Cygwin1.5.25_Svn_1.5.4_Full_Setup.exe 进行安装，如下图：

注：安装前请关闭防火墙，防止误报。

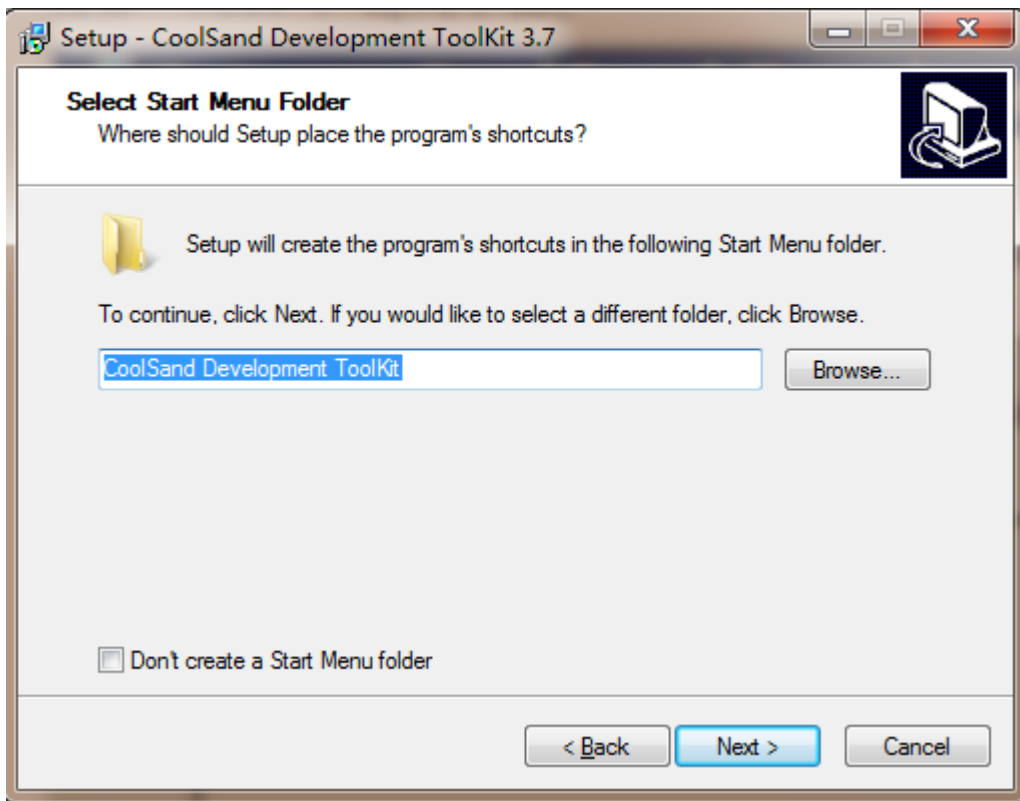


第一次建立编译环境，请严格按照文档来配置，包括路径名称，目录结构，大小写等。对环境熟悉了之后再按照自己的需求定制。

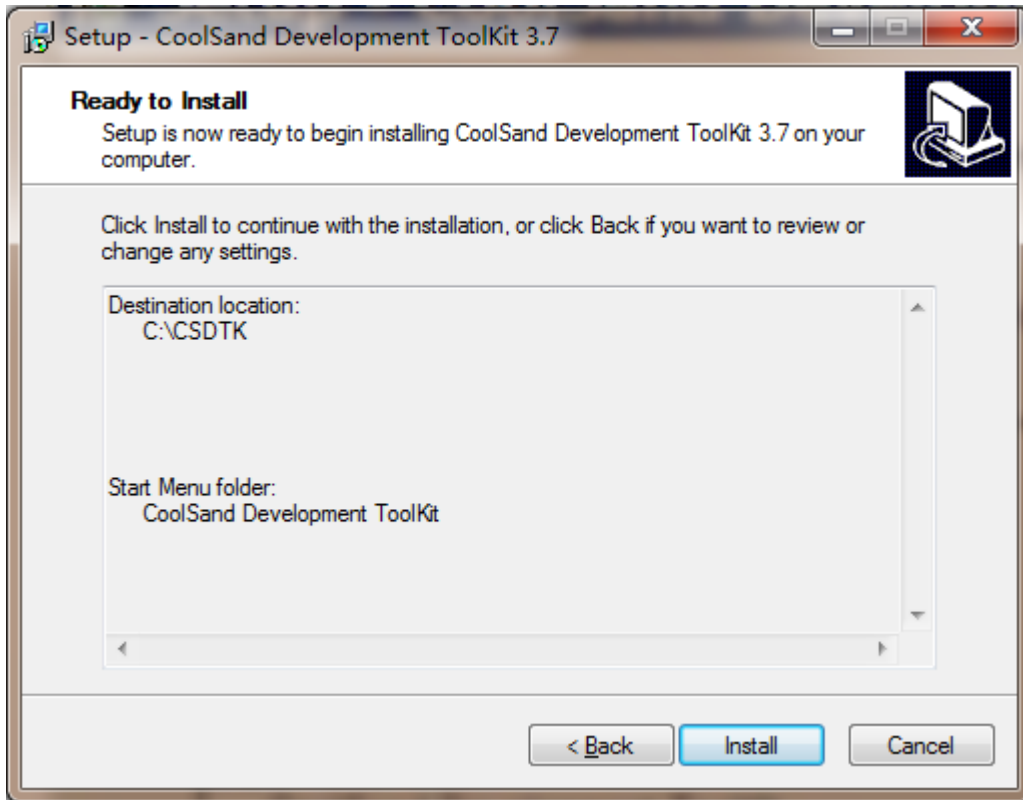
- (2) 选择“Next”，进入下图：



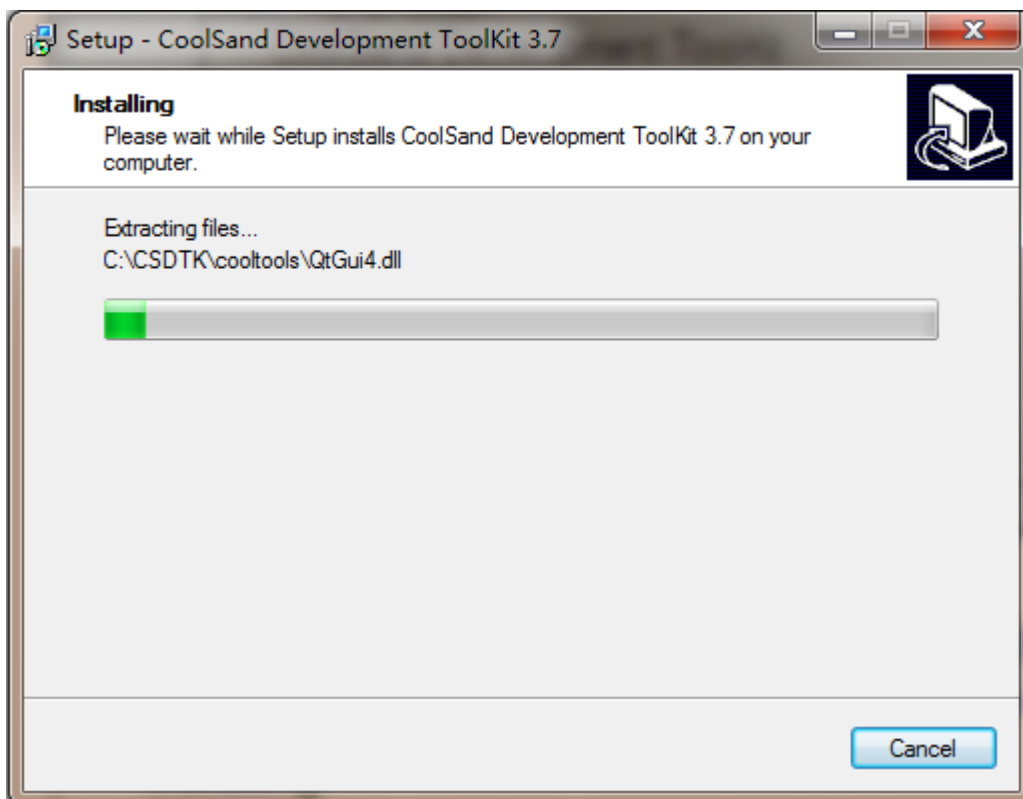
这里选择的是 CSDTK 的安装路径，建议用默认的 C:\CSDTK，选择“Next”，进入下图：



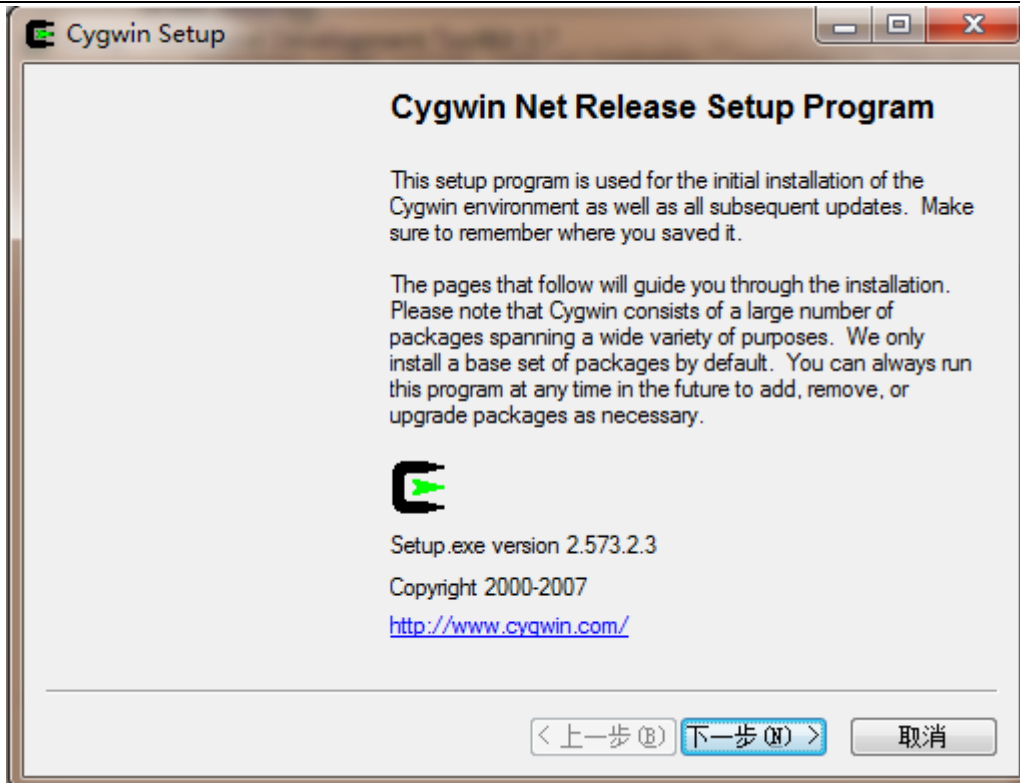
选择“Next”进入下图：



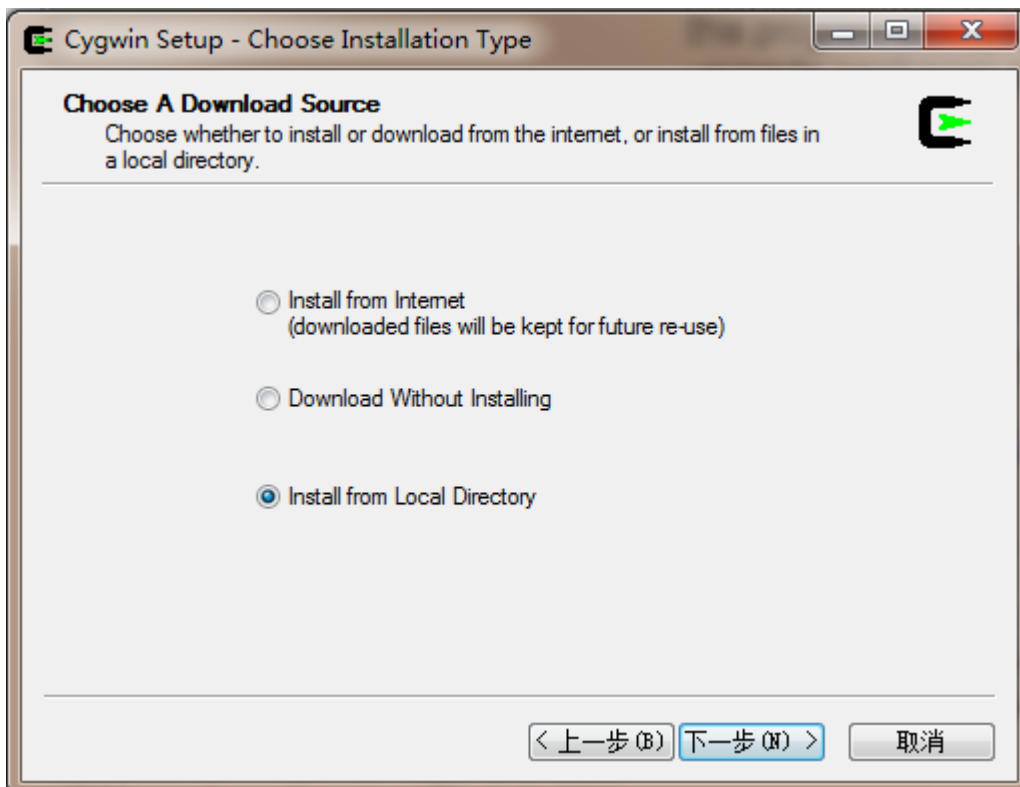
选择“Install”，进行安装。



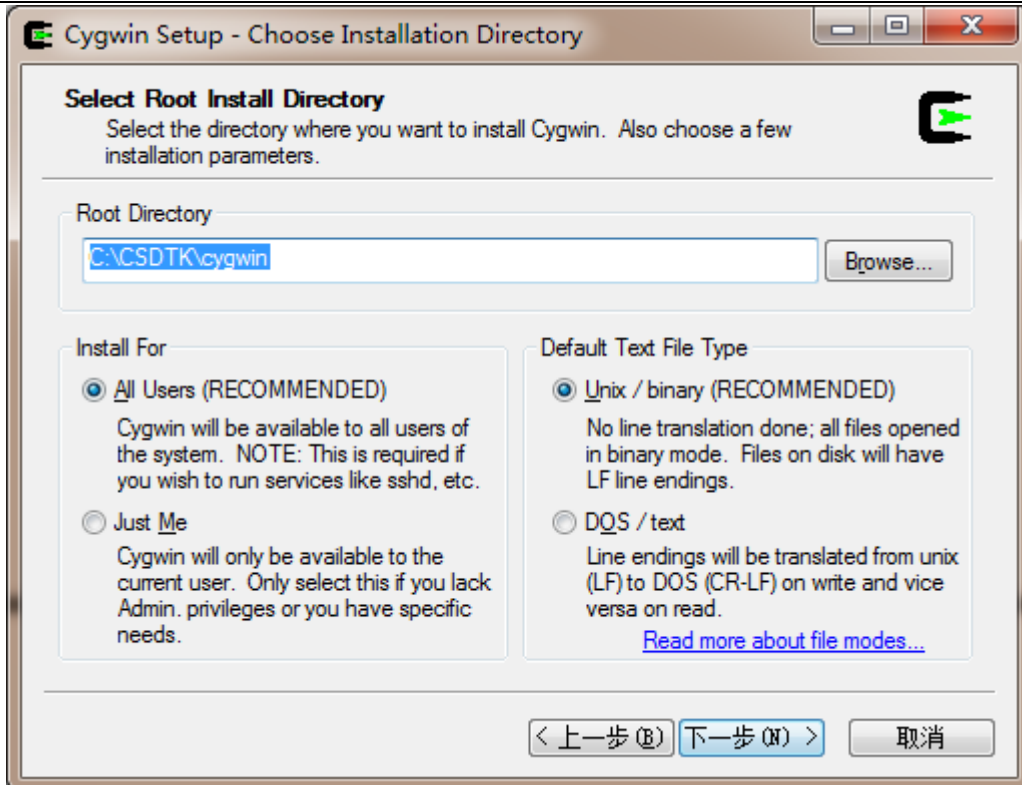
(3) 等待进度条完成之后会自动安装 cooltools 和交叉编译器，如下图：



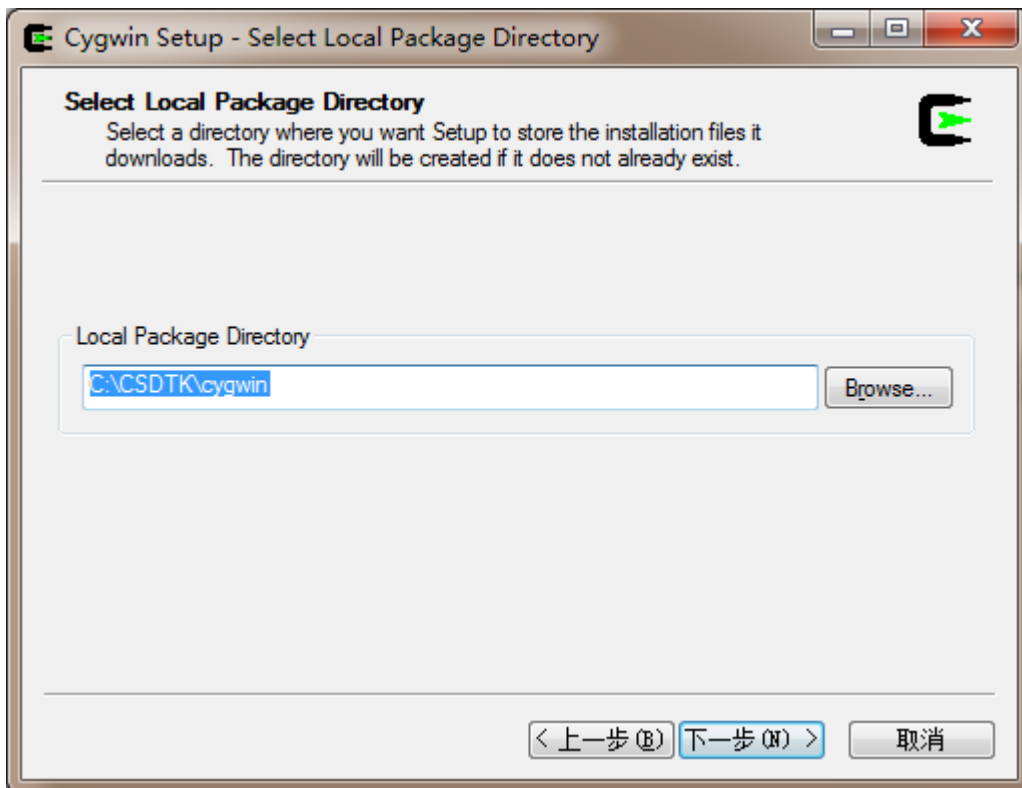
选择“下一步”，进入下图



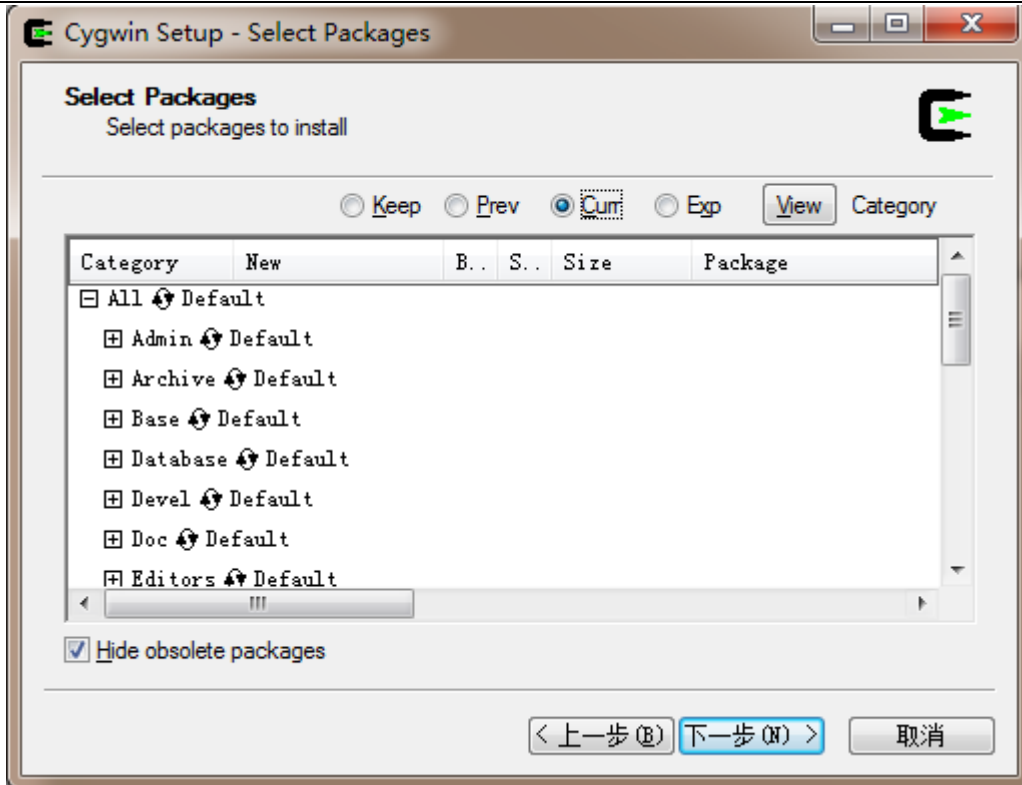
选择第三项默认选项，点击“下一步”，进入下图：



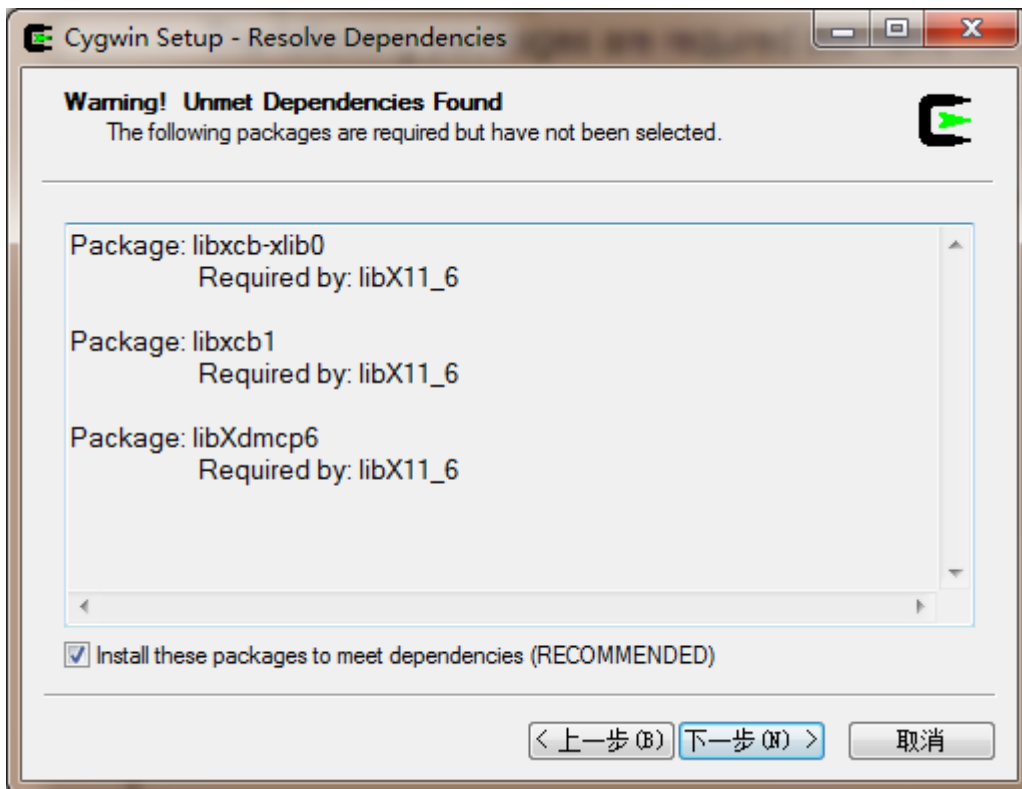
这里选择的是 cygwin 的安装路径，建议使用默认的 C:\CSDTK\cygwin。在“Install For”标签里建议选择“**All User**”，在“Default Text File Type”标签里必须选择“**Unix/binary**”。选择“**下一步**”，进入下图：

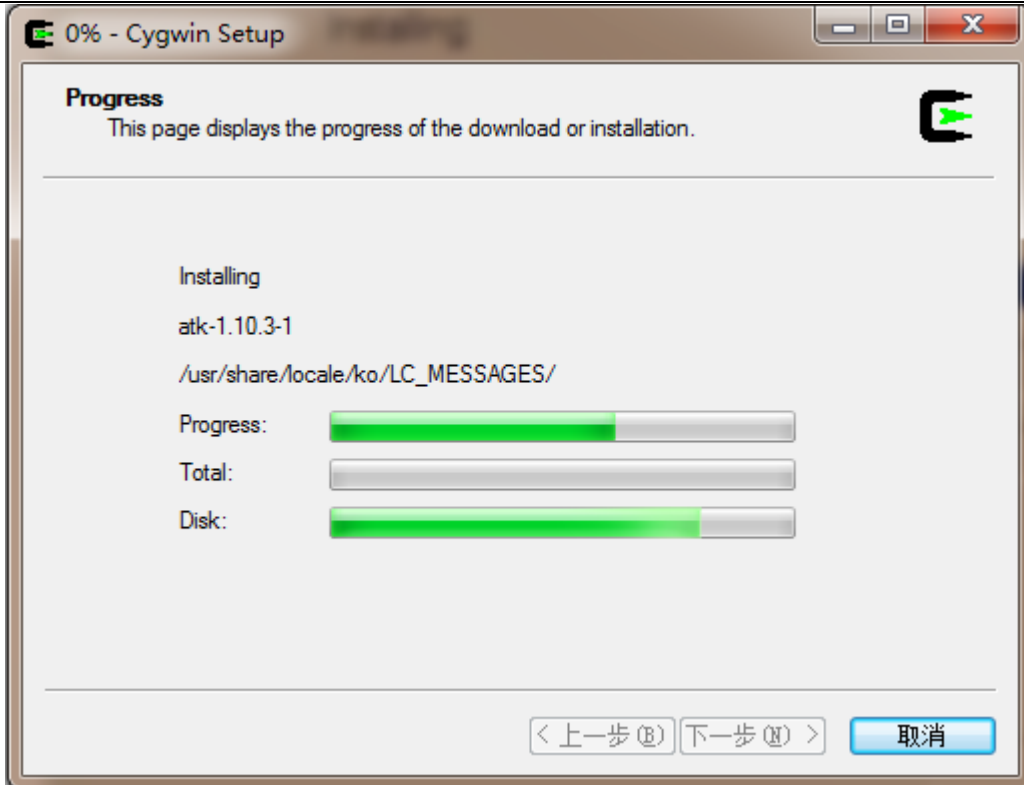


这里要选择的是 cygwin 的本地安装包的路径，我们之前的安装过程已经将其保存在 C:\CSDTK\cygwin，所以直接选择“**下一步**”即可，进入下图：

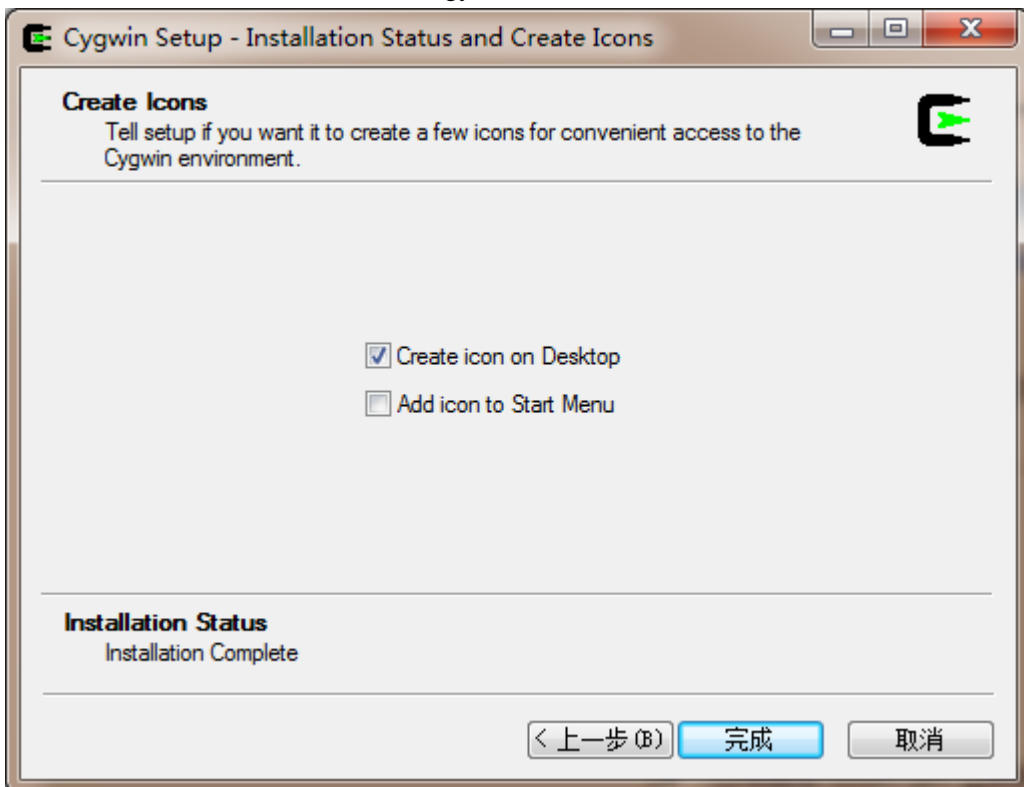


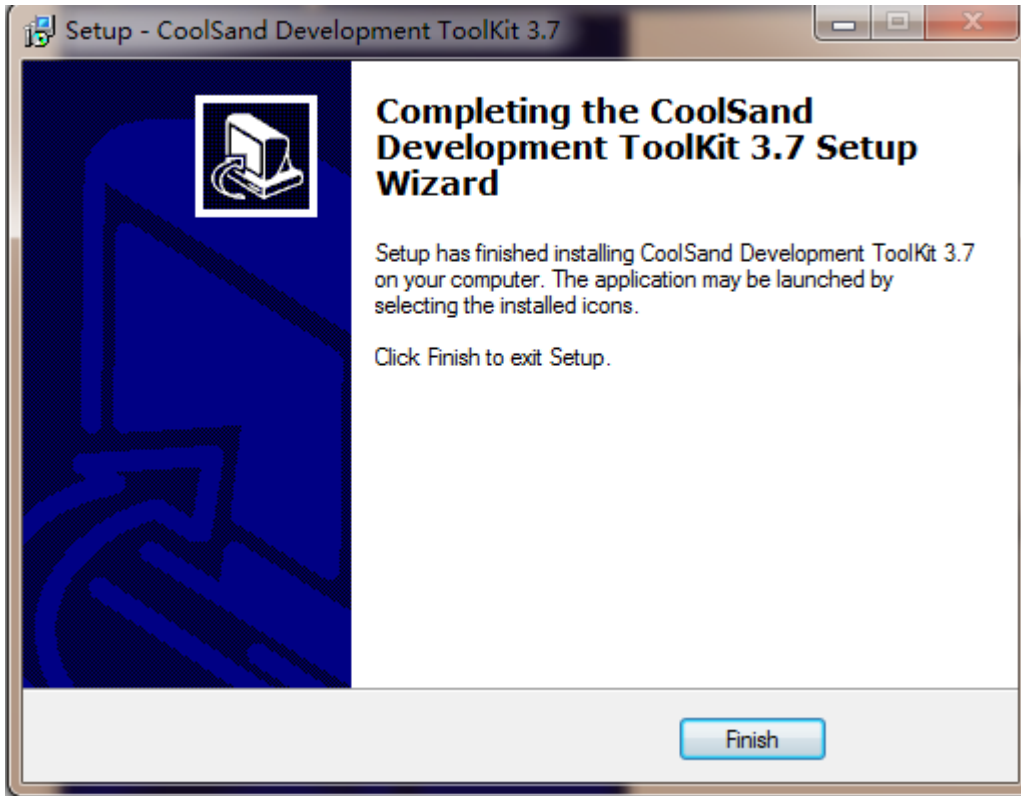
这一步是要选择安装的包，直接选择“下一步”，然后选择“安装”。





等待安装完成后点击“完成”结束 cygwin 安装。



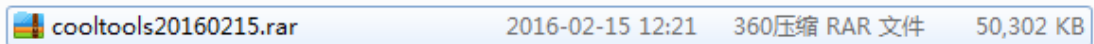


点击“Finish”结束安装。

(4) 更新 cooltools 工具

删除 C:\CSDTK\cooltools 目录

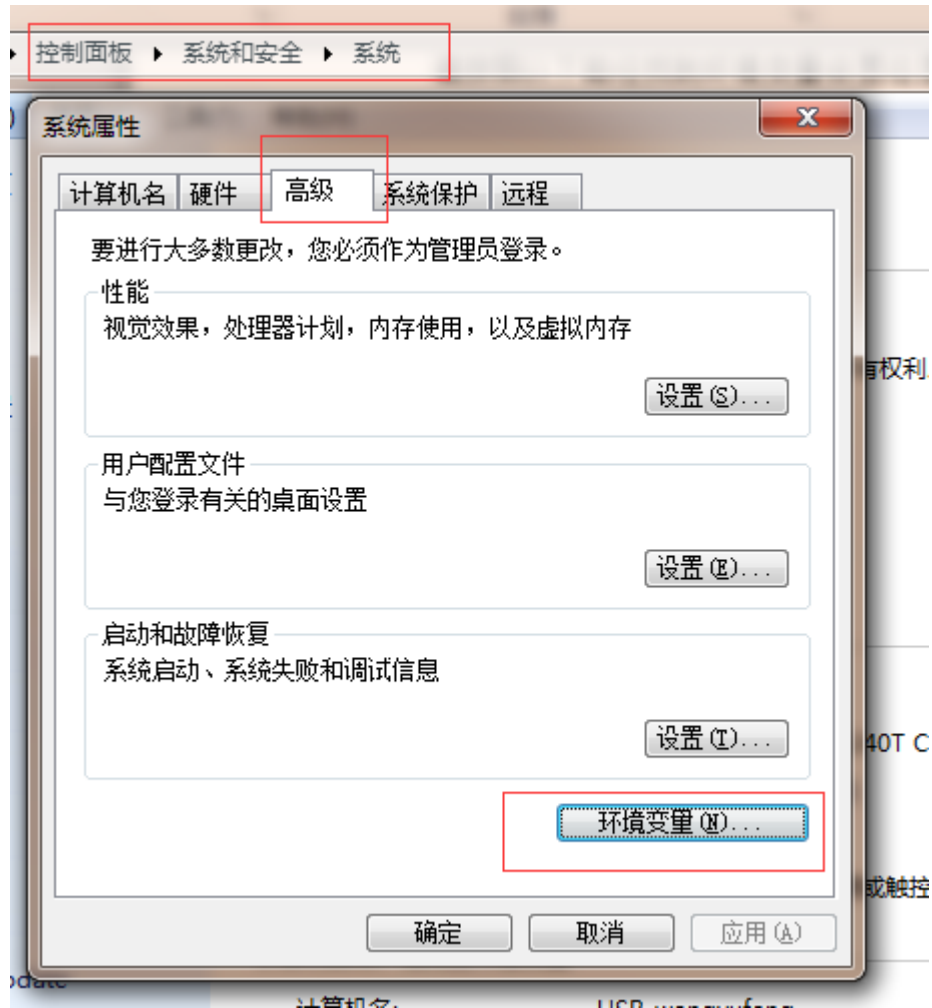
将我们提供的最新的文件解压到目录当中，如下图文件：



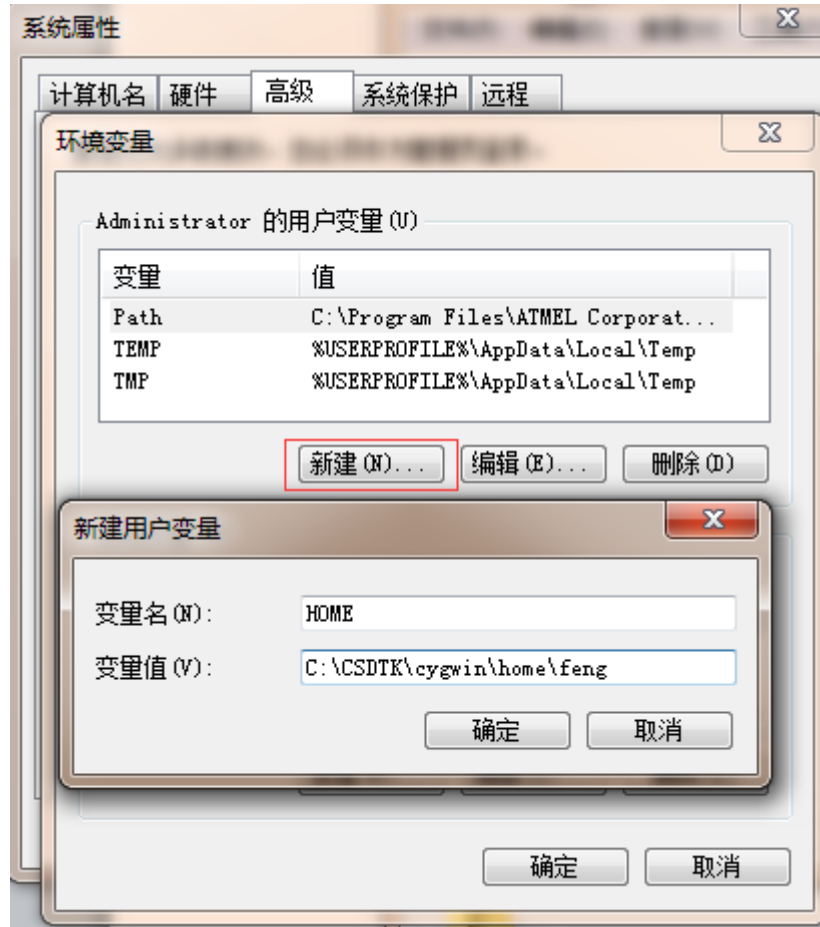
请确保更新操作过程中不能修改原始的文件结构。

(5) 设置一个 HOME 的环境变量

请按照以下路径找到环境变量设置位置，如下图：

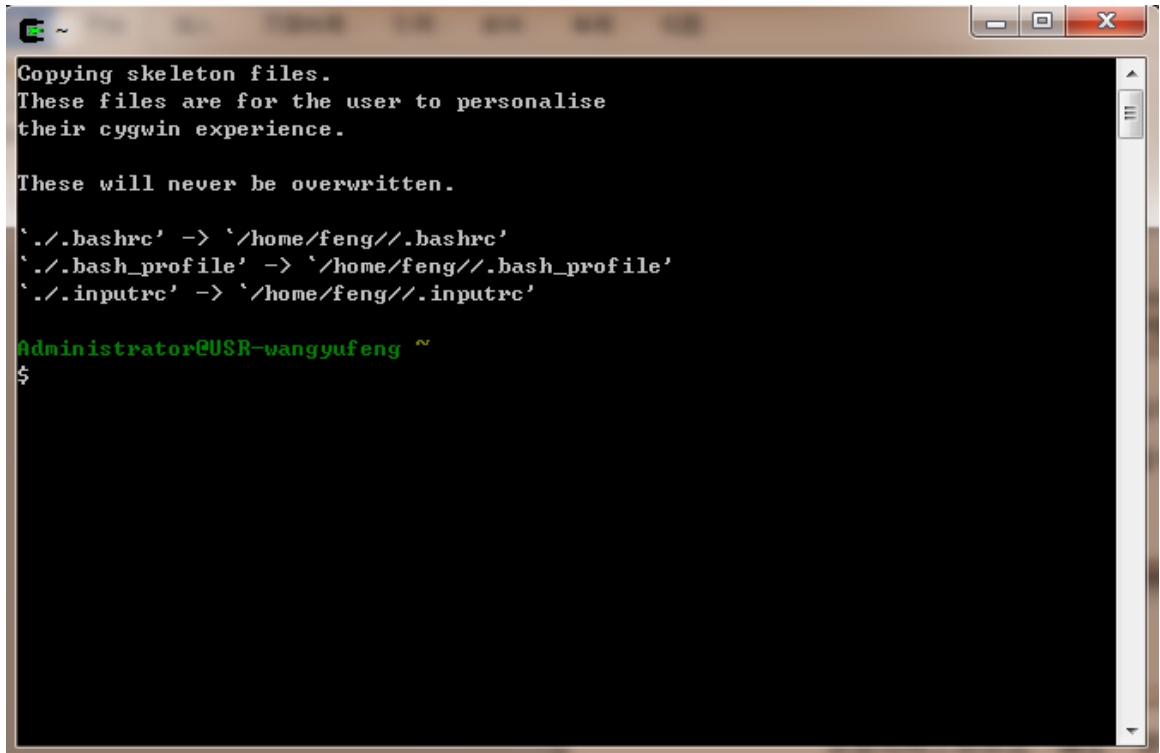


点击环境变量，新建用户变量，其中变量值中的“feng”可以自己定义名称，如下图：



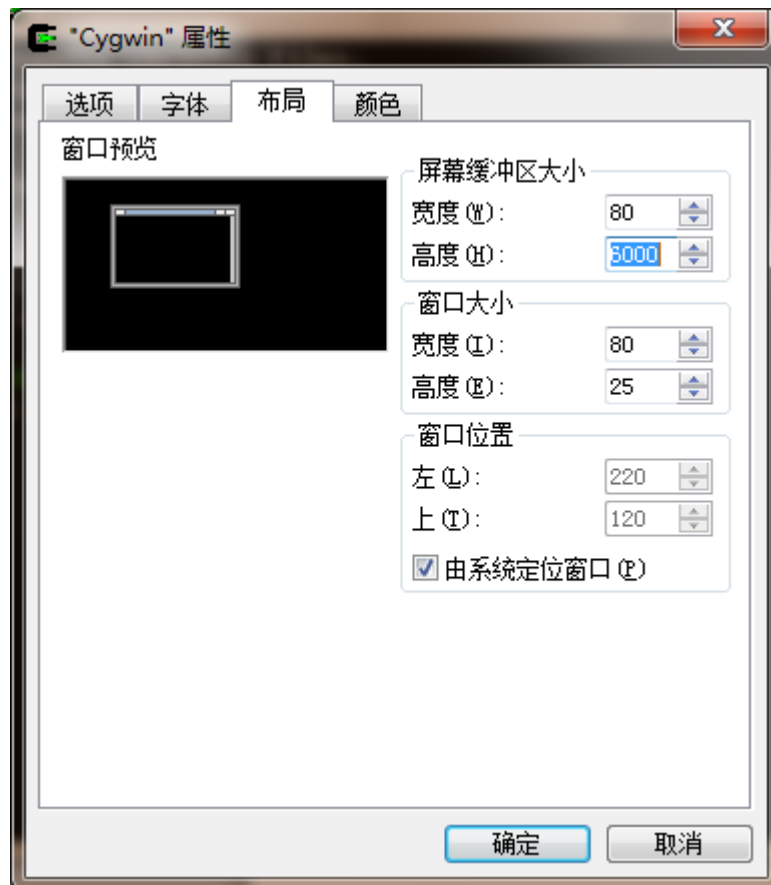
设置完成之后点击“确定”进行保存。

- (6) 启动桌面上的 cygwin 程序，会在 C:\CSDTK\cygwin\home\下创建一个以当前用户名命名的目录：



```
Copying skeleton files.  
These files are for the user to personalise  
their cygwin experience.  
  
These will never be overwritten.  
  
'./.bashrc' -> '/home/feng/./.bashrc'  
'./.bash_profile' -> '/home/feng/./.bash_profile'  
'./.inputrc' -> '/home/feng/./.inputrc'  
  
Administrator@USR-wangyufeng ~  
$
```

建议将 cygwin 窗口的缓冲区高度增大一些，右键窗口标题栏，选择属性，如下图：



2.2. 配置代码路径

- (1) 使用 UltraEdit 编辑 C:\CSDTK\cygwin\.bashrc 文件，若打开时提示要转换成 DOS 格式，请选否。
- (2) 将 118 行：

export PROJ_ROOT=`cygpath "C:\projects"` 如下图：

```
116
117 # Root directorty for all the projects
118 export PROJ_ROOT=`cygpath "C:\projects"`
119
```

替换成：

export PROJ_ROOT=`cygpath "D:\projects"` 后面路径为源码位置

```
116
117 # Root directorty for all the projects
118 export PROJ_ROOT=`cygpath "D:\projects"`
119
120 # type work <project_name> to switch the environment to the required project
121 alias work='source ./cygenv.sh'
```

- (3) 将 124 行：

export PATH=/usr/bin:/crosscompiler/bin:/cooltools/: 如下图：

```
122
123 # set path to include the crosscompiler and the cooltools
124 export PATH=/bin:/usr/bin:/crosscompiler/bin:/cygdrive/C/CSDTK/cooltools:
125
126 # Create the link /cygdrive/n -> /n if it does not exist
127 # Cooltools will need that in order to
```

替换成：

export

PATH=/bin:/usr/bin:/crosscompiler/bin:/cygdrive/C/CSDTK/cooltools:/cygdrive/C/P

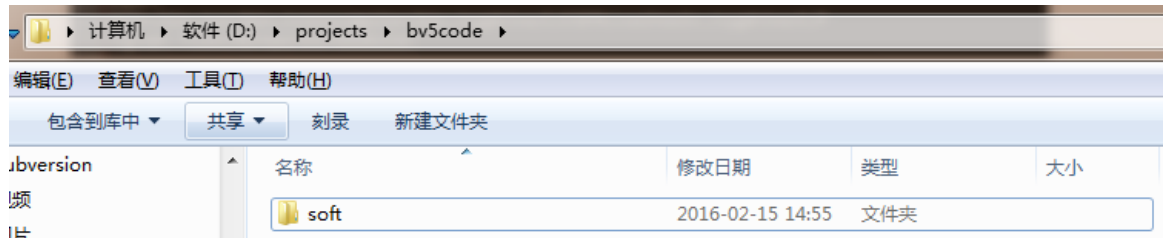
rogram\ Files\Xoreax\IncrediBuild:

```
123 # set path to include the crosscompiler and the cooltools
124 export
125 PATH=/bin:/usr/bin:/crosscompiler/bin:/cygdrive/C/CSDTK/cooltools:/cygdrive/C/P
126 rogram\ Files\Xoreax\IncrediBuild:
```

- (4) 保存并关闭此文件。
- (5) 重新打开 cygwin，系统会将以上的修改应用到 C:\CSDTK\cygwin\home\feng\.bashrc 文件中

3. 编译工程

- (1) 将我司提供的最新的代码包解压到配置代码路径时填写的路径，我们写的路径是 D:\projects，首先我们要到 D 盘下建立这个目录，然后我们在这个目录下建立 bv5code 文件夹，然后将 soft 解压到 bv5code 中。如下图：



- (2) 切换到代码工作区域，输入以下命令：
work bv5code 如下图：



如果显示如下图所示，则进入工作成功：

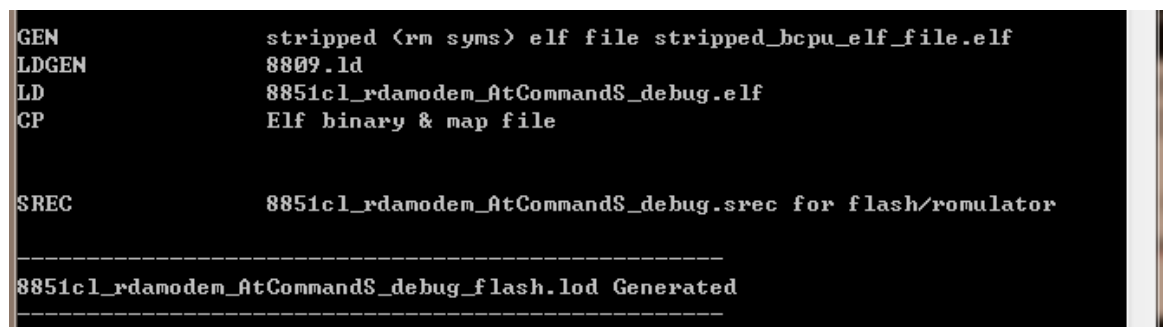


每次打开 cygwin 时都需要将工作区域切换到该处，当然用户可以自定义工作区域，但是要在 projects 这个目录下面，然后 work 后面跟自定义的工作区域的名字。

- (3) 切换到工作区域之后可以输入我们的编译命令进行编译
ctmake -j 16 CT_TARGET=8851cl_rdamodem CT_USER=FAE CT_RELEASE=debug
WITH_SVN=0 CT_MODEM=1 CT_PRODUCT=AtCommandS at/usr/
如下图所示：

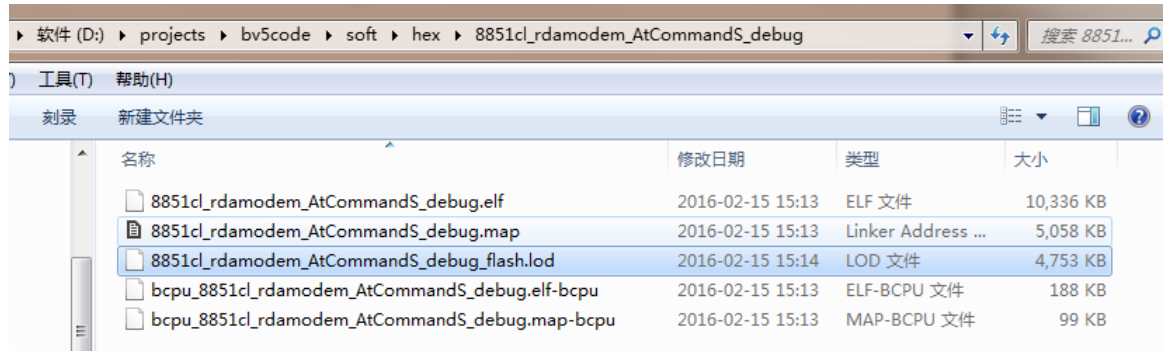


如果编译过程中没有错误的话，编译成功则会生成可下载的 lod 文件，如下图所示：



如果编译出错，则编译过程会停止，可以在编译信息找到出错的位置和出错的原因，可以方便的进行调试。

- (4) 我们可以到工作区的 Hex 文件夹下面找到编译生成的文件，如下图所示：

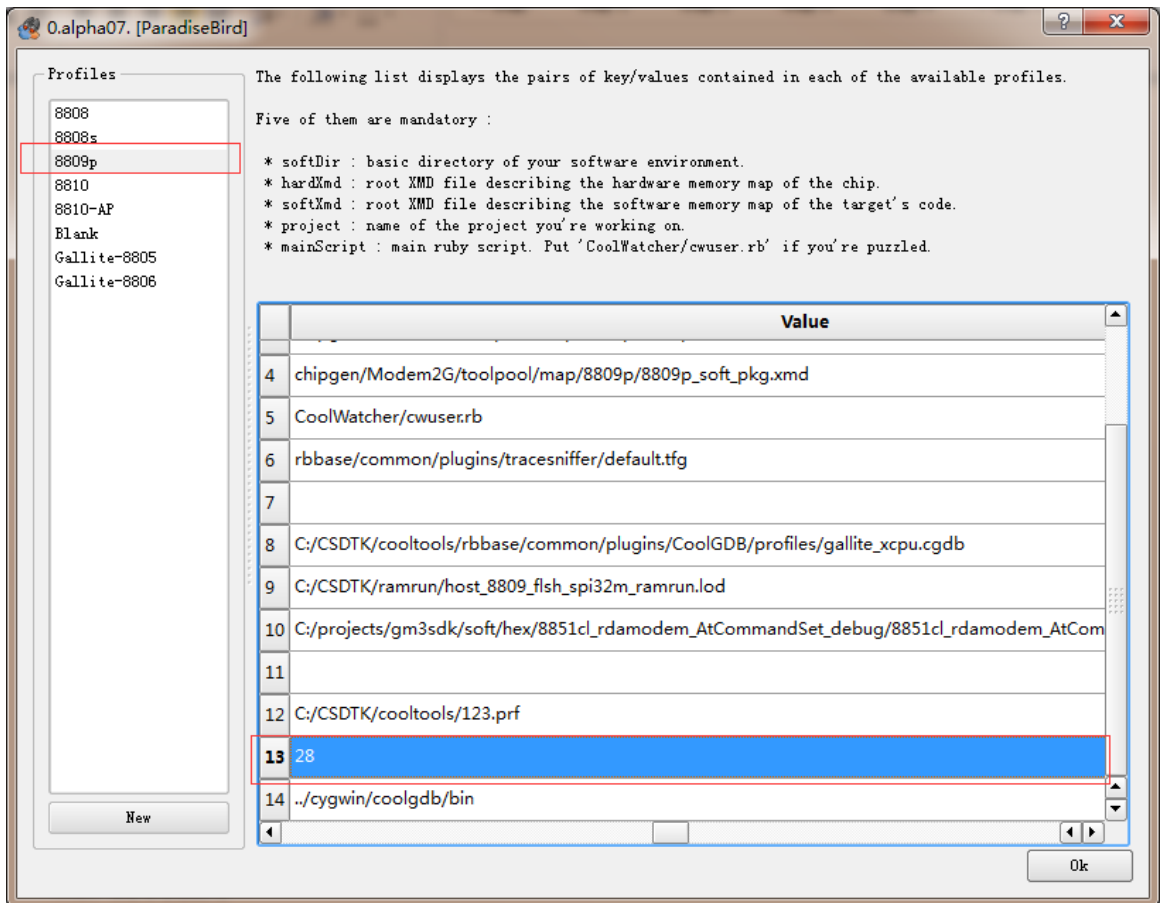


- (5) 在编译指令里面有一项参数是 CT_RELEASE=debug, 如果这个参数是 debug, 那么将会生成 debug 版的固件, 该固件的程序运行时打印 trace 信息并且当出现运行异常时会停止运行, 可以使用调试软件抓取异常位置, 如果该参数是 release 时, 软件运行过程中不会打印 trace 信息并且程序异常时直接自动重启。一般来说, 程序开发过程中生成 debug 版, 程序发布时生成 release 版。

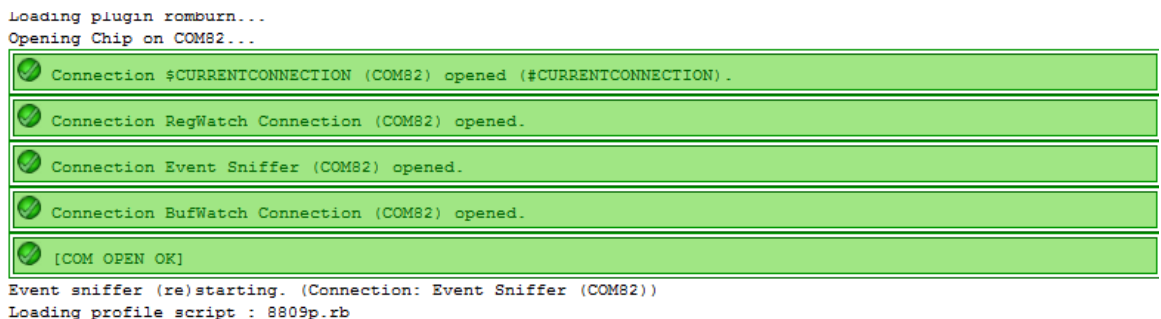
4. 烧录调试

4.1. 烧录程序

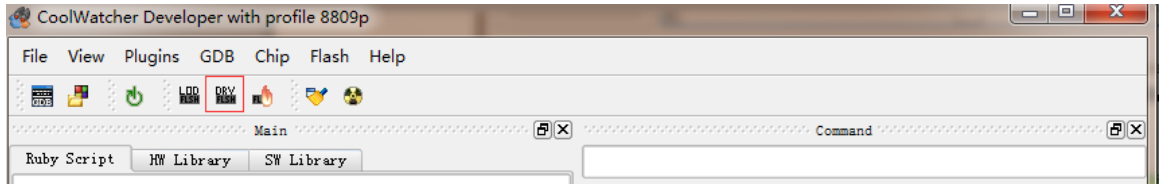
- (1) 找到 C:\CSDTK\cooltools 目录下的 coolwatcher.exe 文件, 为快速访问, 可以在桌面创建其快捷方式。
- (2) 双击打开此文件, 如下图所示:



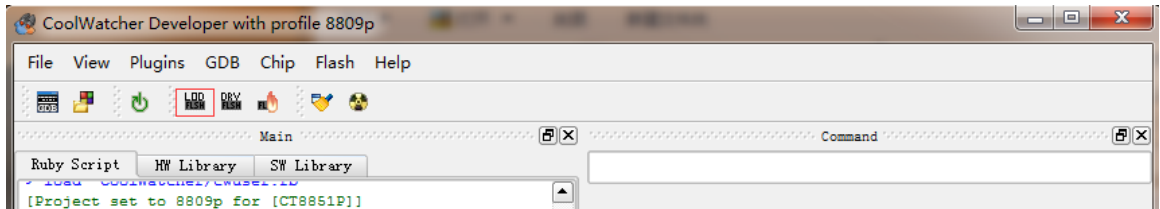
在 Profiles 中选择 8809p, 在 value 中的 13 行输入待下载模块的下载串口的串口号, 点击 OK。显示如下信息代表串口打开成功:



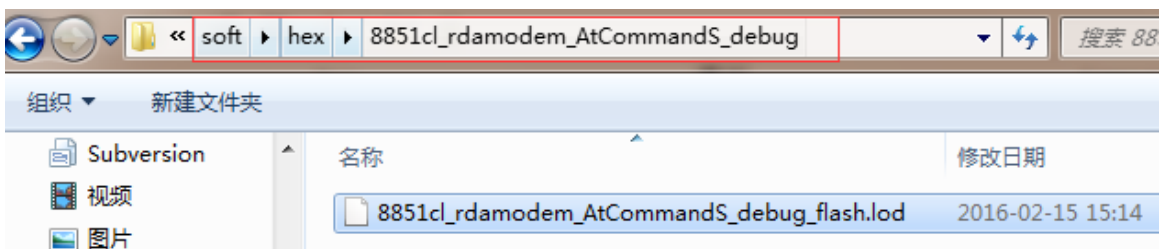
- (3) 将我们提供的 ramrun 压缩包解压到 C:\CSDTK 目录下, 点击如图所示的按钮, 加载 C:\CSDTK\ramrun 目录下的: host_8809_flash_spi32m_ramrun.lod 文件, 该文件只有在程序第一次运行时候需要加载, 以后不需要加载。



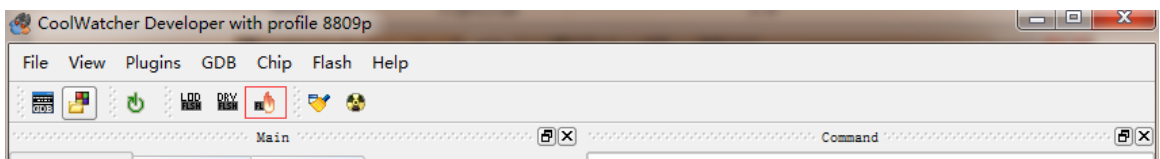
- (4) 加载编译后的 lod 文件。点击如下按钮，找到工作区的 hex 文件夹，并找到编译后的 lod 文件。



加载 lod 文件，如下图所示：



- (5) 开始烧写程序，点击如下图所示的按钮，开始执行烧写任务。



会有如下信息显示：

```
> rastpiGo()
Fastpf V2 over Host
Loading the lod files:
8851cl_rdamodem_AtCommandS_debug_flash.lod
Using flash programmer:
host_8809_flsh_spi32m_ramrun.lod
Ramrunning the flash programmer...

Resetting SPI flash...
Done.

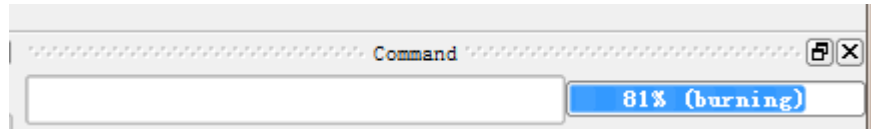
Entered Host Monitor mode.

Configuring EBC RAM ...
Done.

0.031000 0.016000 0.047000 ( 0.176000)

Verify enabled: true
Fastpf Protocol Version: 1.4
Got FPC buffer size: 32768
Fastpfing...
```

并且显示下载的进度条：



下载完成之后，程序自动运行。

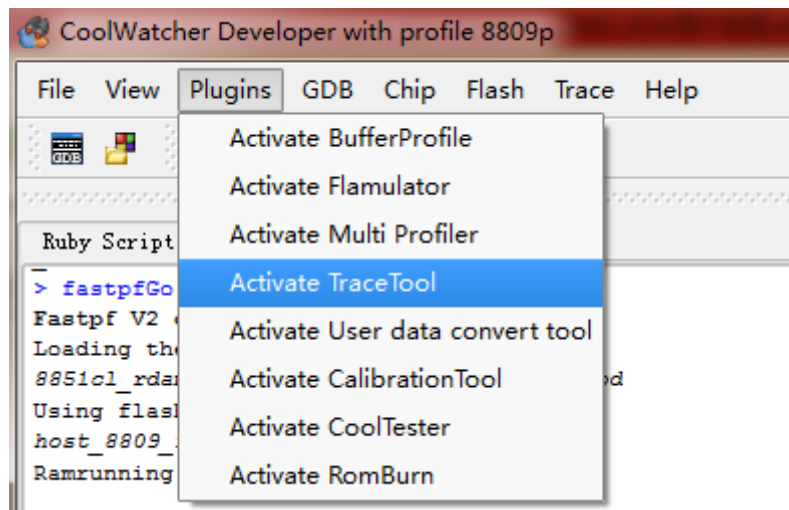
4.2. 调试程序

程序开发过程中为方便观察程序的执行情况，用户可在程序中增加 trace 打印语句，例如在程序某处打印一个变量的值可以用：

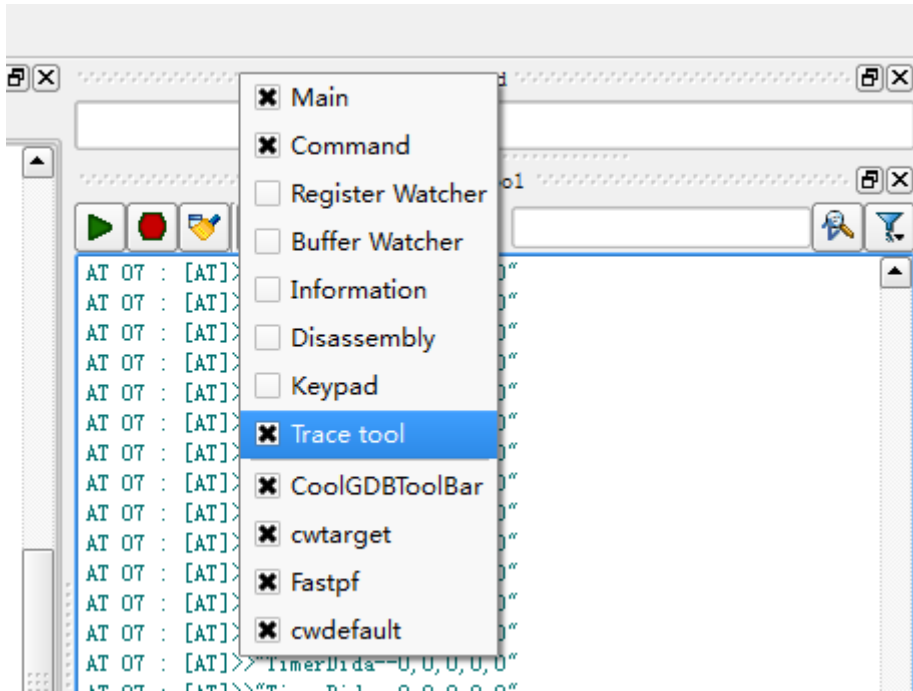
```
AT_TC(USR_TRACE,"data=%d",num);
```

程序编译成 debug 版下载到模块之后，我们就可以使用工具进行抓取：

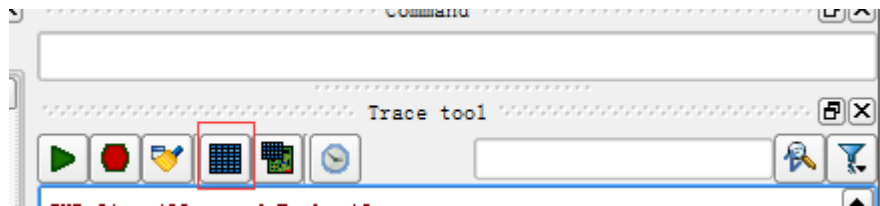
(1) 首先我们运行 Trace 模块：



(2) 在空白区域右击鼠标，然后选 Trace Tool



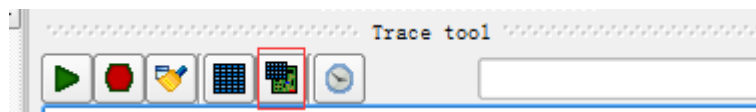
(3) 点击选择设置 Trace 等级



(4) 选择我们 trace 打印时的等级是 7，选择 7，然后点击 apply。

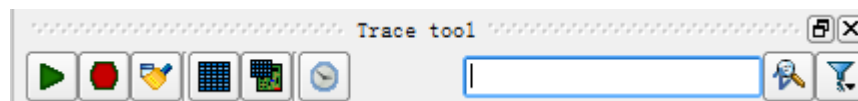
SND	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	All	None
API	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	All	None
MMI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	All	None
SIM	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	All	None
AT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	All	None
M2A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	All	None

(6) 点击如下按钮执行操作：

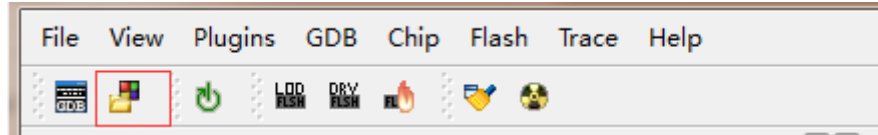


(7) 如果没有 trace 信息出现，点击第一个三角号开始运行。

(8) 在如下的位置输入要观察的 Trace 信息可以进行搜索

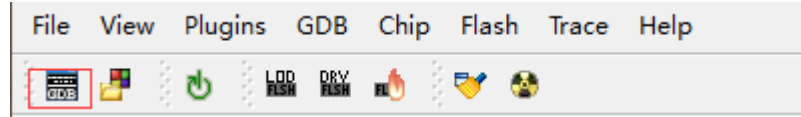


(9) 如果程序出现跑死的情况，可以进行 GDB 调试，第一次运行需要加载文件，点击如下图所示的图标：



找到 C:\CSDTK\cooltools\rbbase\common\plugins\CoolGDB\profiles\gallite_xcpu.cgd 文件，此文件也只需要加载一次即可。

(10) 然后点击如下图标可以打开 gdb 调试窗口，可以定位到死机的位置。



5. API 函数

5.1. 程序开发说明

用户找到 Soft 文件夹下面的 At 文件夹，在里面有一个 usr 文件夹，此目录是开放给用户进行编程的目录，其中 Include 为头文件的位置，Src 是用户源代码的位置，其中我们已经定义好了一个.c 文件，里面有我们编写的测试程序，可以方便用户快速熟悉接口的使用方法。

其中 Usr_AppMain 函数用户可作为主函数，系统运行之后将会调用此函数。其中 USR_WAIT_TASK_INIT 函数是等待注册网络，只有网络注册成功之后才会往下进行。因为结尾处有死循环，所以此函数中的内容只会被调用一次。

所有可用的接口文件都在 Include 目录下中的 Usr_gm3_app.h 中列出，用户可以切换到此位置进行查看。

5.2. 函数说明

5.2.1. UART 函数

5.2.1.1. Usr_OpenUart

功能：打开串口

```
BOOL Usr_OpenUart(UART_ID id,UART_PARAM *uartCfg);
```

参数：UART_ID 串口号 当前开放 UART_1

UART_PARAM 串口参数

```
rate      波特率
    HAL_UART_BAUD_RATE_2400
    HAL_UART_BAUD_RATE_4800
    HAL_UART_BAUD_RATE_9600
    HAL_UART_BAUD_RATE_14400
    HAL_UART_BAUD_RATE_19200
    HAL_UART_BAUD_RATE_28800
    HAL_UART_BAUD_RATE_33600
    HAL_UART_BAUD_RATE_38400
    HAL_UART_BAUD_RATE_57600
    HAL_UART_BAUD_RATE_115200
    HAL_UART_BAUD_RATE_230400
    HAL_UART_BAUD_RATE_460800
    HAL_UART_BAUD_RATE_921600
parity    校验位
    HAL_UART_NO_PARITY
    HAL_UART_ODD_PARITY
```

HAL_UART_EVEN_PARITY
data 数据位
HAL_UART_7_DATA_BITS
HAL_UART_8_DATA_BITS
stop 停止位
HAL_UART_1_STOP_BIT
HAL_UART_2_STOP_BITS
flow 流控
HAL_FLOW_NONE
HAL_FLOW_RTS_CTS
HAL_FLOW_RS485

返回值:

成功: TRUE 失败: FALSE

版本要求: V1.0

5.2.1.2. Usr_CloseUart

功能: 关闭串口

BOOL Usr_CloseUart(UART_ID id);

参数: UART_ID 串口号 当前开放 UART_1

返回值:

成功: TRUE 失败: FALSE

版本要求: V1.0

5.2.1.3. Usr_SendUartData

功能: 通过串口发送数据

U16 Usr_SendUartData (UART_ID id,U16 *data,U16 len)

参数: UART_ID 串口号

*data 要发送数据的指针

len 要发送数据的长度

返回值:

成功: 成功发送数据长度 失败: 0

版本要求: V1.0

5.2.1.4. Usr_RegisterUartRxCallback

功能: 注册串口接收数据回调函数

VOID Usr_RegisterUartRxCallback(UART_ID id,UART_CALLBACK_FUNC func)

参数: UART_ID 串口号
func 回调函数的名称

示例:

要处理串口数据的函数为:

```
U16 Usr_Uart_Get(U8 *data,U16 len)
```

则注册方法为:

```
Usr_RegisterUartRxCallback(UART_1, Usr_Uart_Get);
```

这样串口接收的数据直接转到 Usr_Uart_Get 方法中进行处理。

返回值:

空

版本要求: V1.0

5.2.2.Socket 函数

5.2.2.1. Usr_OpenSock

功能: 使能 Socket 连接

```
VOID Usr_OpenSock(SOCK_ID id)
```

参数: SOCK_ID Scket ID 号

SOCK1

SOCK2

返回值:

空

版本要求: V1.0

5.2.2.2. Usr_CloseSock

功能: 关闭 Socket 连接

```
VOID Usr_CloseSock(SOCK_ID id)
```

参数: SOCK_ID Scket ID 号

SOCK1

SOCK2

返回值:

空

版本要求: V1.0

5.2.2.3. Usr_SetApn

功能：设置联网 APN 参数

BOOL Usr_SetApn(SYS_PARAM_APN *apn)

参数：SYS_PARAM_APN APN 信息

apn[50]

user[50]

psw[50]

返回值：

成功：TRUE 失败：FALSE

版本要求：V1.0

5.2.2.4. Usr_SetSockParam

功能：设置联网参数

BOOL Usr_SetSockParam(SOCK_ID id,GM3_SOCK_PARAM *param)

参数：SOCK_ID Socket ID 号

GM3_SOCK_PARAM

sock_type

TCP_TYPE

UDP_TYPE

link_type

SHORT_LINK

LONG_LINK

addr[100]

server_port

示例：请参考源码当中的连接程序

返回值：

成功：TRUE 失败：FALSE

版本要求：V1.0

5.2.2.5. Usr_SendSockData

功能：通过 Socket 发送数据

INT32 Usr_SendSockData(SOCK_ID id,U8 *data,U16 len)

参数：SOCK_ID Socket ID 号

*data 发送数据指针

len 发送数据长度

返回值:

成功: 发送数据长度 失败: -1

版本要求: V1.0

5.2.2.6. Usr_RegisterSockRxCallback

功能: 注册 Scket 数据接收回调函数

VOID Usr_RegisterSockRxCallback(SOCK_ID id,SOCK_CALLBACK_FUNC func)

参数: SOCK_ID Socket ID 号

func 处理接收数据函数名

示例: 请参考串口注册回调或查看代码示例

返回值:

空

版本要求: V1.0

5.2.2.7. Usr_GetSockStatus

功能: 获取当前 Socket 的状态

BOOL Usr_GetSockStatus(SOCK_ID id,U8 *state)

参数: SOCK_ID Socket ID 号

*state 保存状态的指针

说明: 状态一共有如下所示状态:

AT_TCPIP_STATE_IPINITIAL	0
AT_TCPIP_STATE_IPSTART	1
AT_TCPIP_STATE_IPCONFIG	2
AT_TCPIP_STATE_IND	3
AT_TCPIP_STATE_GPRSACT	4
AT_TCPIP_STATE_IPSTATUS	5
AT_TCPIP_STATE_CONNECTING	6
AT_TCPIP_STATE_CLOSE	7
AT_TCPIP_STATE_CONNECTOK	8
AT_TCPIP_STATE_SOCKETOK	9
AT_TCPIP_STATE_UDPOK	10

返回值:

成功: TRUE 失败: FALSE

版本要求: V1.0

5.2.3.SMS 接口

5.2.3.1. Usr_SendSmsData

功能：发送短信函数

U16 Usr_SendSmsData(UINT8 *phone, SMS_SEND_TYPE type, UINT8 *data, U16 len)

参数：*phone 接收短信的手机号

type 发送短信的类型

ASCLL

GSM

UCS2

*data 发送数据指针

len 发送数据长度

返回值：

成功：发送短信长度 失败：0

版本要求：V1.0

5.2.3.2. Usr_SetSmsDest

功能：设置短信透传目的号码

BOOL Usr_SetSmsDest(char *phone, U8 len)

参数：*phone 手机号码指针

len 手机号码长度

返回值：

成功：发送短信长度 失败：0

版本要求：V1.0

5.2.3.3. Usr_RegisterSmsRxCallback

功能：注册短信接收回调函数

VOID Usr_RegisterSmsRxCallback(SMS_CALLBACK_FUNC func)

参数：func 短信接收处理函数名

示例：请参考串口注册回调或查看代码示例

返回值：

空

版本要求：V1.0

5.2.4.Timer 函数

5.2.4.1. Usr_StartTimer

功能：启动定时器

VOID Usr_StartTimer(TIMER_ID id)

参数：TIMER_ID 定时器 ID,一共五路定时器可用

TIMER1

TIMER2

TIMER3

TIMER4

TIMER5

返回值：

空

版本要求：V1.0

5.2.4.2. Usr_StopTimer

功能：关闭定时器

VOID Usr_StopTimer(TIMER_ID id)

参数：TIMER_ID 定时器 ID,一共五路定时器可用

TIMER1

TIMER2

TIMER3

TIMER4

TIMER5

返回值：

空

版本要求：V1.0

5.2.4.3. Usr_SetTimer

功能：设置定时器相关信息

VOID Usr_SetTimer(TIMER_ID id,U32 period,TIMER_CALLBACK_FUNC func)

参数：TIMER_ID 定时器 ID

TIMER1

TIMER2

TIMER3

TIMER4

TIMER5

period 定时器触发时间，单位秒

func 定时器触发时调用的函数，函数中建议不要增加延时操作

返回值:

空

版本要求: V1.0

5.2.5.系统 AT 指令函数

5.2.5.1. Usr_SendAtCmd

功能: 执行系统 AT 指令

BOOL Usr_SendAtCmd(U8 *data,U16 len)

参数: *data AT 指令，注意要符合 At 指令格式后面跟”\r\n”

len AT 指令长度

示例: Usr_SendAtCmd("AT+VER?\r\n",strlen("AT+VER?\r\n"));

返回值:

成功: TRUE 失败: FALSE

版本要求: V1.0

5.2.5.2. Usr_RegisterCmdRxCallback

功能: 注册 AT 指令处理回调函数

VOID Usr_RegisterCmdRxCallback(CMD_CALLBACK_FUNC func)

参数: func AT 指令处理函数名

返回值:

空

版本要求: V1.0

5.2.6.自定义 AT 指令

为方便用户添加自定义 AT 指令，我们已经将 AT 指令处理的框架做好，用户按照要求添加即可。

(1) 进入 usr_gm3_cmd_table.h 文件当中，里面已经有一个示例指令

{(UINT8*)" +TEST", AT_GC_USR_CmdFunc_TEST, SA_CMDCLS_TCPIP, 1, AT_IPR_PERM},

该代码表示增加一个 AT 指令为 TEST，当接收到该 AT 指令时调用 AT_GC_USR_CmdFunc_TEST 函数进行处理，该函数在 usr_gm3_sdk.c 当中可以看到。

- (2) 在 AT_GC_USR_CmdFunc_TEST 当中我们已经把查询，设置的框架做好，用户根据示例代码中的注释进行完善即可。

5.2.7. 系统接口函数

5.2.7.1. Usr_SetEcho

功能：设置是否打开回显

VOID Usr_SetEcho(BOOL b)

参数：TRUE 打开回显

FALSE 关闭回显

返回值：

空

版本要求：V1.0

5.2.7.2. Usr_Restart

功能：重启模块

VOID Usr_Restart()

参数：

空

返回值：

空

版本要求：V1.0

5.2.7.3. Usr_ReloadUserDefault

功能：恢复用户出厂设置

VOID Usr_ReloadUserDefault()

参数：

空

返回值：

空

版本要求：V1.0

5.2.7.4. **Usr_ReloadUsrFactory**

功能：恢复默认出厂设置

VOID Usr_ReloadUsrFactory()

参数：

空

返回值：

空

版本要求：V1.0

5.2.7.5. **Usr_SaveCurrentSetting**

功能：保存当前参数

VOID Usr_SaveCurrentSetting ()

参数：

空

返回值：

空

版本要求：V1.0

5.2.7.6. **Usr_SaveAsUserDefault**

功能：保存当前设置为用户默认设置

VOID Usr_SaveAsUserDefault ()

参数：

空

返回值：

空

版本要求：V1.0

5.2.7.7. **Usr_EnableRFC**

功能：设置是否打开 RFC2217 功能

VOID Usr_EnableRFC (BOOL b)

参数：TRUE 打开

FALSE 关闭

返回值:

空

版本要求: V1.0

5.2.7.8. Usr_EnableUartCMD

功能: 设置是否打开串口 AT 指令功能

VOID Usr_EnableUartCMD (BOOL b)

参数: TRUE 打开

FALSE 关闭

返回值:

空

版本要求: V1.0

5.2.7.9. Usr_EnableNetCMD

功能: 设置是否打开网络 AT 指令功能

VOID Usr_EnableNetCMD (BOOL b)

参数: TRUE 打开

FALSE 关闭

返回值:

空

版本要求: V1.0

5.2.7.10. Usr_GetIMEI

功能: 获取模块的 IMEI

VOID Usr_GetIMEI(U8 *imei)

参数: *imei imei 保存位置的指针

返回值:

空

版本要求: V1.0

5.2.8.GPIO 控制

GM3 SDK 允许用户控制外部的 GPIO,我们一共为客户开放 6 路的 GPIO 控制和 4 路 LED 指示灯控制,用户在使用过程中根据自己的需要选择进行控制。

5.2.8.1. GPIO 定义

GPIO 定义	HAL_APO_ID_T	PIN	原引脚功能
PIN_GPIO_1	g_gm3.gpio_1	30	UART2_RTS
PIN_GPIO_2	g_gm3.gpio_2	8	I2C_SCL
PIN_GPIO_3	g_gm3.gpio_3	31	UART2_CTS
PIN_GPIO_4	g_gm3.gpio_4	37	UART1_CTS
PIN_GPIO_5	g_gm3.gpio_5	7	485_EN
PIN_GPIO_6	g_gm3.gpio_6	26	SIM_INTN
PIN_GPIO_7	g_gm3.gpio_7	10	GPRS_LED
PIN_GPIO_8	g_gm3.gpio_8	11	LINKA_LED
PIN_GPIO_9	g_gm3.gpio_9	12	LINKB_LED
PIN_GPIO_10	g_gm3.gpio_10	13	DATA_LED

5.2.8.2. Usr_GpioInit

功能: 初始化 GPIO

VOID Usr_GpioInit(HAL_APO_ID_T id,U8 mode)

参数: id 见 GPIO 中 HAL_APO_ID_T 定义
 mode 设置 GPIO 模式 1 为输出, 0 为输入

返回值:

空

版本要求: V1.0

5.2.8.3. Usr_SetGpio

功能: 设置 GPIO 状态

VOID Usr_SetGpio(HAL_APO_ID_T id,U8 mode)

参数: id 见 GPIO 中 HAL_APO_ID_T 定义
 mode 设置 GPIO 状态 1 为高电平, 0 为低电平

返回值:

空

版本要求: V1.0

5.2.8.4. Usr_GetGpio

功能：获取 GPIO 状态

U32 Usr_GetGpio(HAL_APO_ID_T id)

参数：id 见 GPIO 中 HAL_APO_ID_T 定义

返回值：

1 为高电平状态

0 为低电平状态

版本要求：V1.0

5.2.9.RTC 时钟接口

GM3 模块自带 RTC 时钟，通过外部 RTC 供电引脚供电，可以实现模块掉电计时，外部引脚为 PIN16，名为 VBAT_RTC，该引脚可以使用 2.8V 进行供电。

5.2.9.1. Usr_GetSysTime

功能：获取系统时间

BOOL Usr_GetSysTime(HAL_TIM_RTC_TIME_T* rtcTime)

参数：rtcTime HAL_TIM_RTC_TIME_T 时钟结构体

UINT8 sec; Second

UINT8 min; Minute

UINT8 hour; Hour

UINT8 day; Day

UINT8 month; Month

UINT8 year; Year

UINT8 wDay; Week Day

返回值：

成功：TRUE 失败：FALSE

版本要求：V1.0

5.2.9.2. Usr_SetSysTime

功能：设置系统时间

BOOL Usr_SetSysTime(HAL_TIM_RTC_TIME_T* rtcTime)

参数: rtcTime HAL_TIM_RTC_TIME_T 时钟结构体

UINT8 sec;	Second
UINT8 min;	Minute
UINT8 hour;	Hour
UINT8 day;	Day
UINT8 month;	Month
UINT8 year;	Year
UINT8 wDay;	Week Day

返回值:

成功: TRUE 失败: FALSE

版本要求: V1.0

6. 联系方式

公 司：济南有人物联网技术有限公司

地 址：山东省济南市高新区新泺大街 1166 号奥盛大厦 1 号楼 11 层

网 址：<http://www.usr.cn>

客户支持中心：<http://h.usr.cn>

邮 箱：sales@usr.cn

企 业 QQ：8000 25565

电 话：4000-255-652 或者 0531-88826739

有人愿景：国内联网通讯第一品牌

公司文化：有人在认真做事!

产品理念：简单 可靠 价格合理

有人信条：天道酬勤 厚德载物 共同成长

7. 免责声明

本文档提供有关 USR-GM3 SDK 产品的信息，本文档未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除在其产品的销售条款和条件声明的责任之外，我公司概不承担任何其它责任。并且，我公司对本产品的销售和/或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性，适销性或对任何专利权，版权或其它知识产权的侵权责任等均不作担保。本公司可能随时对产品规格及产品描述做出修改，恕不另行通知。

8. 更新历史

2016-02-15 版本 V1.0.0 创立

2016-03-01 版本 V1.1.0 增加多路 GPIO 控制，增加 RTC 时钟