SC0041
示例代码

AT32F407 SNMP使用指南

前言

简单网路管理协定(SNMP, Simple Network Management Protocol)不仅为网络设备管理时重要的通讯协定之一，亦可以运用于无线网络用以侦测恶意无线基地台，网络管理人员依据侦测结果即时加以隔离处置，避免内部重要资料泄漏或遭受外部蓄意攻击，以维护信息安全。

本应用笔记为SNMP的架构及使用，并进行一个简要的说明，使用者可以基于这个架构，发展符合自己需求的网络管理系统。

注：本示例代码是基于雅特力提供的V2.x.x板级支持包（BSP）而开发，对于其他版本的BSP，需要注意使用上的区别。

支持型号列表：

支持型号	AT32F407xx

目录

图目录

1 概述

1.1 SNMP 基本组件

一个SNMP管理的网络由下列三个关键组件组成：

1. 网络管理系统(NMSs, Network Management Systems)
2. 被管理的装置(Managed Device)
3. 代理者(Agent)

一个网络管理系统执行应用程序，以该应用程序监视并控制被管理的装置。可以称为管理实体(managing entity)，网络管理员在这里与网络装置进行互动。网络管理系统提供网络管理需要的大量运算和存储资源。一个被管理的网络可能存在一个以上的网络管理系统。

一个被管理的装置是一个网络节点，它包含一个存在于被管理的网络中的SNMP代理者。被管理的装置透过管理资讯库(MIB)收集并储存管理信息，并且让网络管理系统能够透过SNMP代理者取得这项信息。

代理者是一种存在于被管理的装置中的网络管理软体模块。代理者控制本地机器的管理信息，以和SNMP兼容的格式传送这项信息。

1.2 SNMP 架构

从体系结构上来讲，SNMP框架由主代理、子代理和管理站组成。

1.2.1 主代理

主代理是一个在可执行SNMP的网络组件上运作的软体，可回应从管理站发出的SNMP要求。它的角色类似客户端/服务器结构(Client/Server)术语中的服务器。主代理依赖子代理提供有关特定功能的管理信息。

如果系统当前拥有多个可管理的子系统，主代理就会传递它从一个或多个子代理处收到的请求。这些子代理在一个子系统以及对那个子系统进行监测和管理操作的接口内为关心的对象建模。主代理和子代理的角色可以合并，在这种情况下我们可以简单的称之为代理(agent)。

1.2.2 子代理

子代理是一个在可执行SNMP的网络组件上运作的软件，执行在特定子系统的特定管理资料库(MIB, Management Information Base)中定义的信息和管理功能。子代理的一些能力有：

1. 搜集主代理的资料
2. 组态主代理的参数
3. 回应管理者的要求
4. 产生警告或陷阱

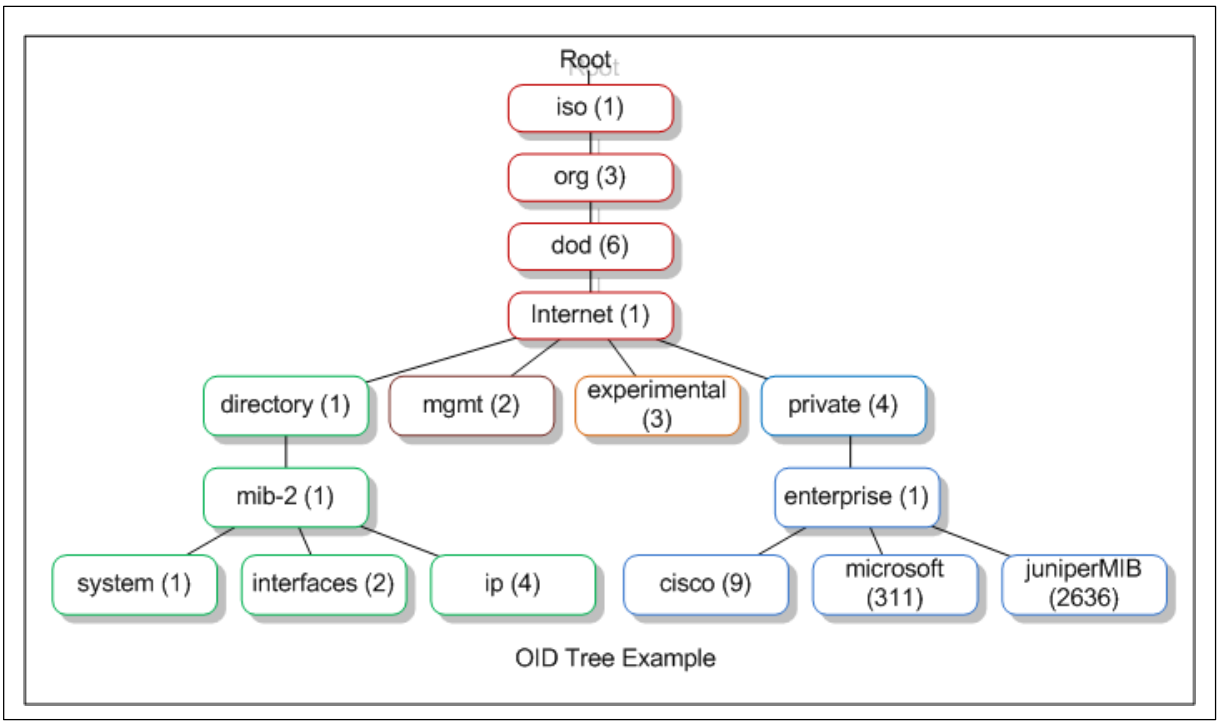
对协定和管理资料结构的良好分离使得使用SNMP来监测和管理同一网络内上百的不同子系统非常简单。MIB模型运行管理OSI参考模型的所有层，并且可以扩展至诸如资料库，电子邮件以及J2EE参考模型之类的应用。

1.2.3 管理站

管理者或者管理站提供第三个组件。它和一个客户端/服务器结构下的用户端一样工作。它根据一个管理员或应用程序的行为发出管理操作的请求，也接收从代理处获得的TRAP。

SNMP Agent是以变量方式呈现被管理装置的相关信息，每个变量皆有其唯一的物件识别码(Object Identifier; OID)，而OID是以阶层方式被描述于资料MIB，例如OID为1.3.6.1.4.1.9代表Cisco公司。

图 1. MIB 阶层架构

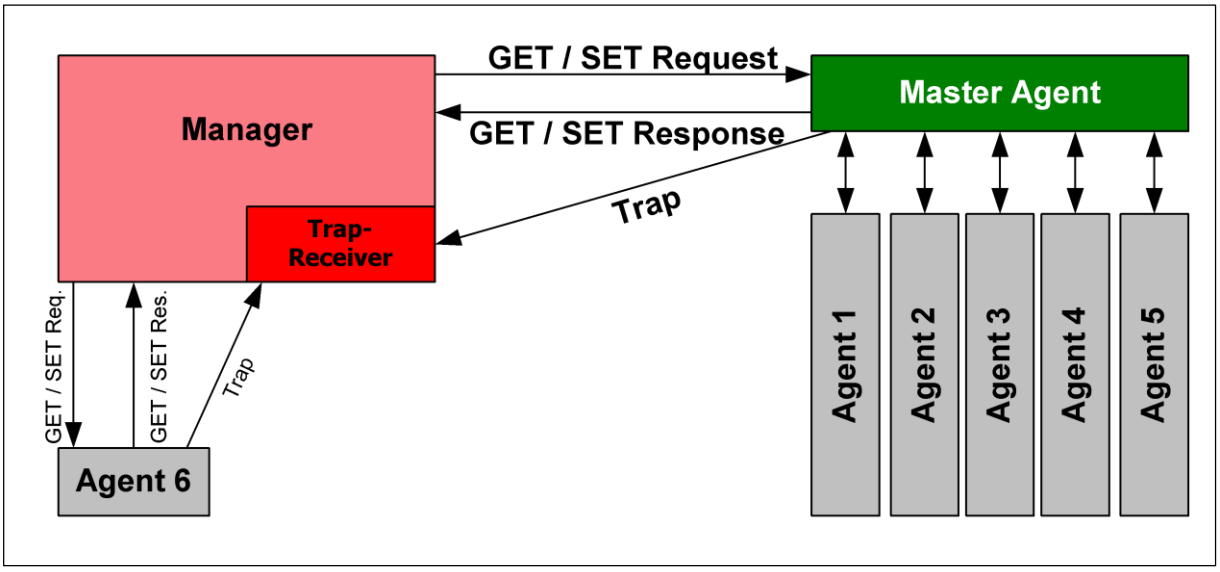


1.3 SNMP 协定的原理及演进

SNMP是运作于OSI模型的应用层，管理端经由UDP传送request至代理(port 161)，代理透过来源端口传送response至管理端。此外，当被监控设备发生异常事件时，例如cold start或link down，代理可经由UDP主动传送notification至管理端(port 162)。

管理端一方面可经由Get、GetNext或GetBulk指令向代理撷取被监控设备的相关信息，另一方面亦可透过代理经由Trap或Inform指令主动传送资料。此外，管理端使用Set指令以达到系统管理的目的。

图 2. SNMP 通讯架构



2 Agent 端软件配置

此应用指南的范例中，以AT32F407做为Agent，使用版载MAC作为以太网接口，KEIL提供的RL-TCPnet为协议栈，SNMP协议位于OSI模型中的应用层，以下将说明如何使用基于RL-TCPNET的SNMP Agent。

2.1 RL-TCPnet

RL-TCPnet是RL-ARM库的组件，提供TCP/IP协议栈的实现。堆栈旨在减少内存使用量和代码大小，这使其适用于资源有限的设备，例如嵌入式系统。RL-TCPnet库是ARM7, ARM9, Cortex-M1和Cortex-M3体系结构的软件例程基础实现。

本使用指南使用RL-TCPnet作为TCP/IP协议栈，在实现以太网通信上相较于LwIP来得容易许多，只要将Keil提供的库给包到工程文件中即可，这里针对将RL-TCPnet给包到工程后，如何配置参数确保以太网通讯正常。

以太网相关的宏定义都放置在Net_Config.c，配置流程如下：

1. 首先将以太网始能的宏设置为1: #define ETH_ENABLE 1
2. 设置MAC Address的宏，在代码中的宏定义为_MACx, x = 1-6
3. 设置IP Address的宏，在代码中的宏定义为_IPx, x = 1-4
4. 设置子网路掩码的宏，在代码中的宏定义为_MSKx, x = 1-4
5. 设置网关地址的宏，在代码中的宏定义为_GWx, x = 1-4
6. 在主函数中，调用init_TcpNet()初始化RL-TCPnet协议栈
7. 在主函数中循环调用main_TcpNet()，轮询以太网相关功能（eth_run_link(), ip_run_local(), icmp_run_engune()等等）

图 3. 以太网相关参数的宏定义

```
41  // <i> Enable or disable Ethernet Network Interface
42  #define ETH_ENABLE      1
43
44  //   <h>MAC Address
45  //   ==============
46  //   <i> Local Ethernet MAC Address
47  //   <i> Value FF:FF:FF:FF:FF:FF is not allowed.
48  //   <i> It is an ethernet Broadcast MAC address.
49  //     <o>Address byte 1 <0x00-0xff:2>
50  //     <i> LSB is an ethernet Multicast bit.
51  //     <i> Must be 0 for local MAC address.
52  //     <i> Default: 0x00
53  #define _MAC1           0x1E
54
55  //     <o>Address byte 2 <0x00-0xff>
56  //     <i> Default: 0x30
57  #define _MAC2           0x30
58
59  //     <o>Address byte 3 <0x00-0xff>
60  //     <i> Default: 0x6C
61  #define _MAC3           0x6C
62
63  //     <o>Address byte 4 <0x00-0xff>
64  //     <i> Default: 0x00
65  #define _MAC4           0xA2
66
67  //     <o>Address byte 5 <0x00-0xff>
68  //     <i> Default: 0x00
69  #define _MAC5           0x45
70
71  //     <o>Address byte 6 <0x00-0xff>
72  //     <i> Default: 0x01
73  #define _MAC6           0x5E
```

图 4. 主函式及主回圈中呼叫 RL-TCPnet 初始化及轮询 API

```
/* Gloable functions ------------------------------------------------------*/
/**
  * @brief  Main Function.
  * @param  None
  * @retval None
  */
int main(void)
{
  /* ---------BSP Init ---------------------------------------------------*/
  AT32_Board_Init();
  UART_Print_Init(115200);
  /* Setup AT32 system (clocks, Ethernet, GPIO, NVIC) */
  System_Setup();
  Delay_init();

  init_TcpNet();
  while(1)
  {
    timer_tick();
    main_TcpNet();
  }
}
```

完成以上步骤，TCP/IP协议栈打通，可以尝试跟PC对接后以ping确认

图 5. 透过 PC 去 ping 开发版

```
系統管理員: C:\Windows\system32\cmd.exe
    最小值 = 0ms, 最大值 = 803ms, 平均 = 390ms

C:\Users\joe_chen>ping 192.168.1.100

Ping 192.168.1.100 (使用 32 位元組的資料):
回覆自 192.168.1.100: 位元組=32 時間=880ms TTL=128
回覆自 192.168.1.100: 位元組=32 時間<1ms TTL=128
回覆自 192.168.1.100: 位元組=32 時間<1ms TTL=128
回覆自 192.168.1.100: 位元組=32 時間=1ms TTL=128

192.168.1.100 的 Ping 統計資料:
    封包: 已傳送 = 4, 已收到 = 4, 已遺失 = 0 (0% 遺失),
大約的來回時間 (毫秒):
    最小值 = 0ms, 最大值 = 880ms, 平均 = 220ms

C:\Users\joe_chen>ping 192.168.1.100

Ping 192.168.1.100 (使用 32 位元組的資料):
回覆自 192.168.1.100: 位元組=32 時間<1ms TTL=128
回覆自 192.168.1.100: 位元組=32 時間=1ms TTL=128
回覆自 192.168.1.100: 位元組=32 時間<1ms TTL=128
回覆自 192.168.1.100: 位元組=32 時間<1ms TTL=128

192.168.1.100 的 Ping 統計資料:
    封包: 已傳送 = 4, 已收到 = 4, 已遺失 = 0 (0% 遺失),
大約的來回時間 (毫秒):
    最小值 = 0ms, 最大值 = 1ms, 平均 = 0ms

C:\Users\joe_chen>
新酷音輸入法 (PIME) 全 :
```

2.2 SNMP Agent

要使用SNMP Agent的功能，首先必须要要将Net_Config.c中的SNMP_ENABLE宏定义打开，才会对SNMP相关模块进行初始化。

打开SNMP模块后，最重要的部份就是构筑管理资料库(Management Information Base; MIB)，MIB相关的代码都放在snmp_mib.c，管理站想要查询的数据都由MIB提供，代码中将MIB建构成一个异质阵列，每个元素都是一个结构体，里面包含了Object Type, Object ID length, Object ID value, Size of a variable, Pointer to a variable和Write/Read event callback function，我们只要按照结构中的元素填写需要的资料即可被管理站访问。

代码中的MIB可以分成两个部份，第一个是System MIB的部份，这个部份属于协议标准的区块，可以更改部份属性，比如说是回应的字符串或是数值，但大部分的属性是不能修改的；另一个部份则是Experimental MIB，这个部份可以新增使用者自己定义的MIB元素，比如透过管理站来开关版载的LED。

图 6. SNMP MIB

```
40  const MIB_ENTRY snmp_mib[] = {
41
42    /* ---------- System MIB ----------- */
43
44    /* SysDescr Entry */
45    { MIB_OCTET_STR | MIB_ATR_RO,
46      8, {OID0(1,3), 6, 1, 2, 1, 1, 1, 0},
47      MIB_STR("Embedded System SNMP V1.0"),
48      NULL },
49    /* SysObjectID Entry */
50    { MIB_OBJECT_ID | MIB_ATR_RO,
51      8, {OID0(1,3), 6, 1, 2, 1, 1, 2, 0},
52      MIB_STR("\x2b\x06\x01\x02\x01\x01\x02\x00"),
53      NULL },
54    /* SysUpTime Entry */
55    { MIB_TIME_TICKS | MIB_ATR_RO,
56      8, {OID0(1,3), 6, 1, 2, 1, 1, 3, 0},
57      4, &snmp_SysUpTime,
58      NULL },
59    /* SysContact Entry */
60    { MIB_OCTET_STR | MIB_ATR_RO,
61      8, {OID0(1,3), 6, 1, 2, 1, 1, 4, 0},
62      MIB_STR("test@arterytek.com"),
63      NULL },
64    /* SysName Entry */
65    { MIB_OCTET_STR | MIB_ATR_RO,
66      8, {OID0(1,3), 6, 1, 2, 1, 1, 5, 0},
67      MIB_STR("Evaluation board"),
68      NULL },
69    /* SysLocation Entry */
70    { MIB_OCTET_STR | MIB_ATR_RO,
71      8, {OID0(1,3), 6, 1, 2, 1, 1, 6, 0},
72      MIB_STR("Local"),
73      NULL },
74    /* SysServices Entry */
75    { MIB_INTEGER | MIB_ATR_RO,
76      8, {OID0(1,3), 6, 1, 2, 1, 1, 7, 0},
77      MIB_INT(sysServices),
78      NULL },
```

2.3 SNMP 管理站

2.3.1 软件环境

2.3.1.1 Python

建立了SNMP Agent后，需要一个SNMP管理站进行沟通，这时候就需要一个SNMP管理站。

网络上有很多SNMP管理站的套装软件，使用者可以使用任何一套自己熟悉的软件来跟SNMP Agent沟通，本使用指南提供的是开源软件SNMP tool，该Tool需要在Python的环境下方可执行，请务必安装Python，这里提供Python的官方连结，请用户自行下载安装。

Python官方链接：https://www.python.org

2.3.1.2 SNMP Tool

这个工具是用纯Python编写的命令行SNMP工具的集合，并与Net-SNMP随附的标准SNMP工具紧密结合，例如：snmpget, snmpwalk等。

在大多数情况下，用户应该能够在Net-SNMP工具和该项目提供的工具之间进行切换，而无须在命令行上进行太多更改。

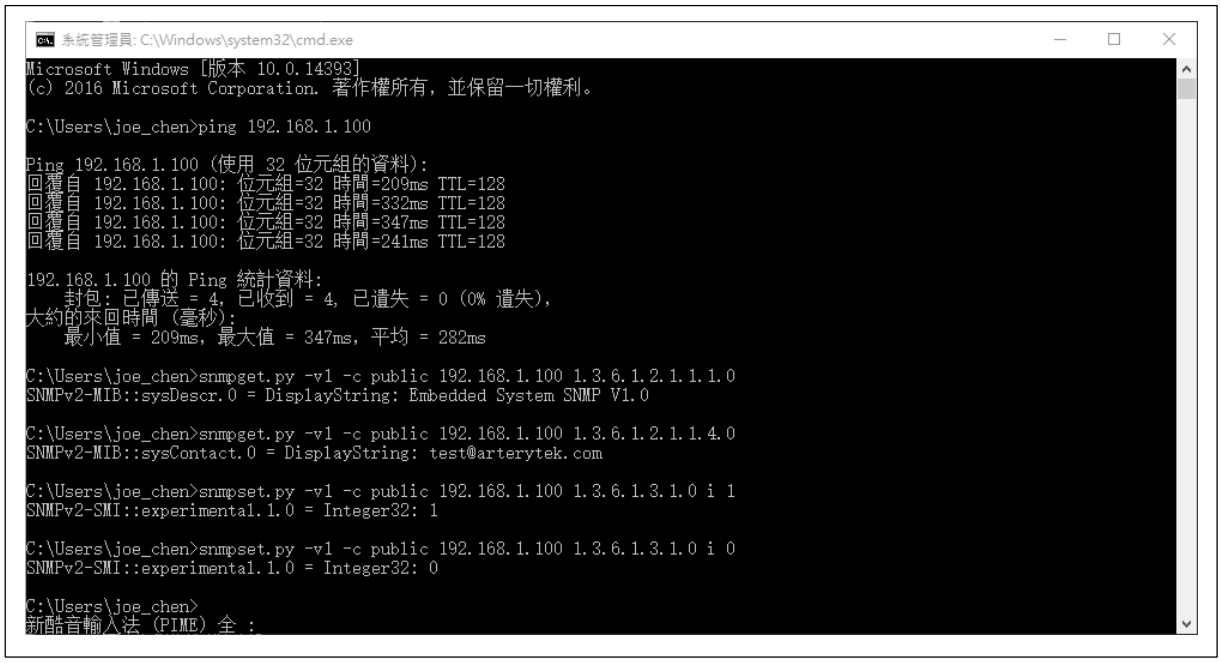
该项目的目标是将SNMP工具引入更广泛的计算平台，并利用高级编程语言轻送地引入SNMP新功能。

这些SNMP工具是免费和开源的。源代码托管在GitHub存储库中，并且按BSD样式第二条款授权进行分发。

此工具可以在官方网站进行下载，详细的安装说明不在此赘述，请使用者自行参阅官方的安装说明。若有缺少某些相依性套件，请自行上Python的开发者网站补齐。

安装完成后，即可使用SNMP command与SNMP Agent进行通讯。

图 7. 使用 SNMP command 存取 MIB



3 版本历史

表 1. 文档版本历史

日期	版本	变更
2021.12.07	2.0.0	将代码及内容更新到V2
2022.03.04	2.0.1	修正目录

重要通知 – 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途(及其依据任何司法管辖区的法律的对应情况)，或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：(A) 对安全性有特别要求的应用，如：生命支持、主动植入设备或对产品功能安全有要求的系统；(B) 航空应用；(C) 汽车应用或汽车环境；(D) 航天应用或航天环境，且/或(E) 武器。因雅特力产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。