**软件详细设计文档**

**IPC**

|  |  |  |  |
| --- | --- | --- | --- |
| 作者:  欧阳汉 | 评审: | 批准: |  |
| 编辑日期:  2017/04/20 | 评审日期: | 批准日期： | 实施日期： |

**目录**

1 本文档版本 1

2 评审 1

3 介绍 1

3.1 定义，简写和缩写 1

3.2 参考文档 1

4 模块设计 2

4.1 模块A设计 2

4.1.1 分解 2

4.1.2 描述 2

4.1.3 外部接口 2

4.1.3.1 输入接口 3

4.1.3.2 输出接口 16

4.1.3.3 验证准则 17

4.1.4 内部接口 17

4.1.4.1 com\_VTPProcess\_5ms 18

4.1.4.2 com\_NWMModeCheck\_10ms 18

4.1.4.3 com\_NetworkAllSleepCheck\_10ms 18

4.1.4.4 com\_NWMProcess\_10ms 18

4.1.4.5 com\_NetworkProcess\_10ms 18

4.1.4.6 com\_HwComSubCheck 18

4.1.4.7 com\_HwComCheck\_10ms 18

4.1.4.8 com\_ComDiagFlagCheck\_10ms 19

4.1.4.9 com\_VehSpdComCheck\_10ms 19

4.1.4.10 v\_nmco\_ioctl 19

4.1.4.11 com\_CheckKeyInOutState 19

4.1.4.12 com\_CheckiWakeFlag 19

4.1.4.13 com\_CheckiKeepFlag 20

4.1.4.14 com\_VTPInterfaceinit 20

4.1.4.15 com\_TrcvInit 20

4.1.4.16 com\_GetNetworkKeepTime 20

4.1.4.17 网络中断 20

4.1.4.18 VTP备用 21

4.1.4.19 验证准则 21

5 数据结构设计 22

6 变量设计 23

6.1 com\_AllFlag\_tag 23

6.1.1 定义 23

6.1.2 描述 23

7 函数设计 24

8 未解决问题 25

9 附录 26

10 模板更改记录 27

# 本文档版本

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订原因** | **修订日期** | **作者** |
| 003 | 增加VTP7.1的两个备用函数 | 2015/4/13 | 于震 |
| 002 | 增加NVM\_WriteDataByIndex | 2015/3/25 | 于震 |
| 001 | 移植自spc560x | 2015/1/30 | 于震 |

# 评审

|  |  |  |
| --- | --- | --- |
| **评审项目** | **评审人** | **评审日期** |
|  |  |  |
|  |  |  |

# 介绍

本文档描述了基于IP31平台网关代码的网络部分代码设计。

## 定义，简写和缩写

参见abbreviation.xlsx

|  |  |
| --- | --- |
| **缩写** | **释义** |
| COM | communication |
| CAN | Controller Area Network |
| LIN | Local Interconnect Network |
| PMM | Power Moding Module |
| NWM | Network Management |

## 参考文档

|  |  |  |
| --- | --- | --- |
| **编号** | **文档名** | **位置** |
| 1 | 2.软件结构化设计指导规范.docx | D:\2.0 Project\8.0 IP31\1.0 kick off |
| 2 | 软件结构图.vsd | D:\2.0 Project\8.0 IP31\1.0 kick off |
| 3 | opt\_ip31\_gw\_SW optimizing Schedule.xlsx | D:\2.0 Project\8.0 IP31\1.0 kick off |

# 模块设计

## 模块A设计

网络通信模块主要负责

1、CAN/LIN网络的启动与关闭，包括在钥匙状态、PMM状态、车速、发动机运行状态、网络Wake/Keep状态、7DF帧状态等。

2、网络管理NWM模块

3、网络的任务处理函数

### 分解

图2 网络管理子模块图

NWM

Network Start/Stop

PT CAN

CH CAN

BD CAN

DIAG CAN

Network Task Function

INFO CAN

### 描述

网络模块用于启停网络、网络管理等功能

### 外部接口



#### 输入接口

##### NVM\_WriteDataByIndex

|  |  |
| --- | --- |
| **原型** | U8 NVM\_WriteDataByIndex (U8 nvmdataindex, const U8\* databuffer, U16 datasize) |
| **描述** | 根据数据名，向Mirror写入数据，并设置该数据的操作状态 |
| **参数** | 输入：  nvmdataindex: 参见NVM\_INDEX\_ENUM ，Mirror中的数据目录  \*databuffer:将要被写入的数据  datasize:数据长度 |
| **返回值** | E\_NVM\_OK = (0x00u),/\*operation is OK\*/  E\_NVM\_PA\_INVALID,/\*Parameter is invalid\*/  E\_NVM\_BUSY,/\*data is writting to EEPROM\*/  E\_NVM\_LAST\_FAILURE,/\*write job failured last time\*/  E\_NVM\_EER\_EEP/\*EEP Destroyed\*/ |

##### DIAG\_GetComCtlNetEnableStatus

|  |  |
| --- | --- |
| **原型** | U8 DIAG\_GetComCtlNetEnableStatus (void); |
| **描述** | 提供网络通信控制状态 |
| **参数** | 无 |
| **返回值** | 控制状态 |

##### DIAG\_GetComCtlNetStatus

|  |  |
| --- | --- |
| **原型** | U8 DIAG\_GetComCtlNetStatus(void); |
| **描述** | 提供网络通信控制标志 |
| **参数** | 无 |
| **返回值** | 控制标志 |

##### DIAG\_GetForceSysAsleepFlag

|  |  |
| --- | --- |
| **原型** | BOOL DIAG\_GetForceSysAsleepFlag(void) |
| **描述** | 提供强制休眠标识 |
| **参数** | 无 |
| **返回值** | 0-DISABLE:禁止强制休眠；  1-ENABLE:使能强制休眠 |

##### DIAG\_GetIOCTLInfomation

|  |  |
| --- | --- |
| **原型** | U8 DIAG\_GetIOCTLInfomation(U8 IOContrlIndex, IOControl\_ST \* iocontrolinfo) |
| **描述** | 获得IO控制的参数信息，包括控制使能标识，控制值 |
| **参数** | IOControlIndex：  参见变量IOControl\_ENUM的定义  Iocontrolinfo:IO控制参数信息 |
| **返回值** | OP\_OK:操作完成  OP\_ERR:参数错误 |

##### DIAG\_GetNetworkCommDelayTime

|  |  |
| --- | --- |
| **原型** | U32 DIAG\_GetNetworkCommDelayTime (U8 index ) |
| **描述** | 提供网络休眠延迟时间 |
| **参数** | 输入参数：  1: INDEX\_HSCAN1\_WAKEN  2: INDEX\_HSCAN1\_OFF  3: INDEX\_HSCAN2\_WAKEN  4: INDEX\_HSCAN2\_OFF  5: INDEX\_HSCAN3\_WAKEN  6: INDEX\_HSCAN3\_OFF  7: INDEX\_HSCAN4\_WAKEN  8: INDEX\_HSCAN4\_OFF  9: INDEX\_HSCAN5\_WAKEN  10:INDEX\_HSCAN5\_OFF  11:INDEX\_Reserved\_2  12:INDEX\_Reserved\_3 |
| **返回值** | 提供网络休眠延迟时间，单位为毫秒（ms）  范围为：0-1275000 |

##### DIAG\_GetRelayControlFlag

|  |  |
| --- | --- |
| **原型** | BOOL DIAG\_GetRelayControlFlag(U8 index) |
| **描述** | 提供网关控制各继电器的使能标识 |
| **参数** | Index：  INDEX\_CTLFLAG\_RAP\_RELAY = (0x00u),  INDEX\_CTLFLAG\_KLR\_RELAY,  INDEX\_CTLFLAG\_KL15\_RELAY,  INDEX\_CTLFLAG\_KL30\_RELAY,  INDEX\_CTLFLAG\_IGN\_RELAY,  INDEX\_CTLFLAG\_Acc\_Wakeup\_RELAY,  INDEX\_CTLFLAG\_Wakeup\_Enable\_RELAY |
| **返回值** | 0- E\_NOT\_CONTROL  1-control |

##### DIAG\_GetSlipControlType

|  |  |
| --- | --- |
| **原型** | U8 DIAG\_GetSlipControlType(void) |
| **描述** | 提供SlipControl类型配置 |
| **参数** | 无 |
| **返回值** | 0-invalid  1-ESP  2-ABS |

##### IOM\_GetDebPinVal

|  |  |
| --- | --- |
| **原型** | U8 IOM\_GetDebPinVal(U8 Index, PU8 PinValue) |
| **描述** | 读取某一端口去抖后状态。  Get debounced pin value |
| **参数** | U8 Index：端口索引。  IomPin\_ENUM（0-INDEX\_IOM\_PIN\_INMAX）  PinValue 指向存储IO 端口状态的指针 ，返回IO端口状态  (0x00u)：STATUS\_IOM\_PIN\_LOW,  (0x01u)：STATUS\_IOM\_PIN\_HIGH,  (0x02u)：STATUS\_IOM\_PIN\_INVALID |
| **返回值** | (0x02u): STATUS\_IOM\_INVALID输入参数错误  (0x01u)：STATUS\_IOM\_OK 正确  (0x00u)：STATUS\_IOM\_ERR 过程处理错误 |

##### IOM\_SetPinVal

|  |  |
| --- | --- |
| **原型** | U8 IOM\_SetPinVal(U8 Index,U8 PinValue) |
| **描述** | PIN端口状态置高 |
| **参数** | U8 Index：端口索引。  IomPin\_ENUM：INDEX\_IOM\_PIN\_INMAX-INDEX\_IOM\_PIN\_ALL  U8 PinValue：设置端口的状态  (0x00u)：STATUS\_IOM\_PIN\_LOW,  (0x01u)：STATUS\_IOM\_PIN\_HIGH,  (0x02u)：STATUS\_IOM\_PIN\_INVALID |
| **返回值** | (0x02u): STATUS\_IOM\_INVALID输入参数错误  (0x01u)：STATUS\_IOM\_OK 正确  (0x00u)：STATUS\_IOM\_ERR 过程处理错误 |

##### l\_sch\_set

|  |  |
| --- | --- |
| **原型** | void l\_sch\_set(l\_ifc\_handle iii, l\_schedule\_handle schedule\_iii, l\_u8 entry); |
| **描述** | Sets up the next schedule to be followed by the l\_sch\_tick function for a certain interface iii. The new schedule will be activated as soon as the current schedule reaches its next schedule entry point. |
| **参数** | Iii : interface handle  Schedule\_iii : schedule handle  Entry : schedule entry point. |
| **返回值** | NA |

##### NWAP\_MonitorRPDRead\_5ms

|  |  |
| --- | --- |
| **原型** | void NWAP\_MonitorRPDRead\_5ms(void); |
| **描述** | Lin监控读取函数 |
| **参数** | NA |
| **返回值** | NA |

##### NWAP\_RoutingSignals\_5ms

|  |  |
| --- | --- |
| **原型** | void NWAP\_RoutingSignals\_5ms(void); |
| **描述** | 网关路由信号执行函数 |
| **参数** | NA |
| **返回值** | NA |

##### NWAP\_RPDErrorProcessing\_5ms

|  |  |
| --- | --- |
| **原型** | void NWAP\_RPDErrorProcessing\_5ms(void); |
| **描述** | Lin故障检测执行函数 |
| **参数** | NA |
| **返回值** | NA |

##### NWAP\_SetLinDiagFlag

|  |  |
| --- | --- |
| **原型** | void NWAP\_SetLinDiagFlag(BOOL tFlag); |
| **描述** | Lin诊断开启标志位获取函数 |
| **参数** | tFlag：LinDiagFlag 状态 |
| **返回值** | NA |

##### NWAP\_SysVolForNetDiagCheck\_5ms

|  |  |
| --- | --- |
| **原型** | void NWAP\_SysVolForNetDiagCheck\_5ms(void); |
| **描述** | 诊断电压检查函数 |
| **参数** | NA |
| **返回值** | NA |

##### PWR\_GetKL15Status

|  |  |
| --- | --- |
| **原型** | U8 PWR\_GetKL15Status(void) |
| **描述** | 获得PWR模块中KL15硬线状态 |
| **参数** | 无 |
| **返回值** | (0x00u)：STATUS\_PWR\_KL\_LOW  (0x01u)：STATUS\_PWR\_KL\_HIGH,  (0x02u)：STATUS\_PWR\_KL\_INVALID |

##### PWR\_GetKLRStatus

|  |  |
| --- | --- |
| **原型** | U8 PWR\_GetKLRStatus(void) |
| **描述** | 获得PWR模块中KLR硬线状态 |
| **参数** | 无 |
| **返回值** | (0x00u)：STATUS\_PWR\_KL\_LOW  (0x01u)：STATUS\_PWR\_KL\_HIGH,  (0x02u)：STATUS\_PWR\_KL\_INVALID |

##### PWR\_GetLocalPowerMode

|  |  |
| --- | --- |
| **原型** | U8 PWR\_GetLocalPowerMode(void) |
| **描述** | 获得PWR模块中PowerMode状态 |
| **参数** | 无 |
| **返回值** | U8:Power Mode 模式  (0x00u)：STATUS\_PWR\_MODE\_OFF,  (0x01u) ：STATUS\_PWR\_MODE\_ACC,  (0x02u) ：STATUS\_PWR\_MODE\_RUN,  (0x03u) ：STATUS\_PWR\_MODE\_CRANK,  (0x04u) ：STATUS\_PWR\_MODE\_UNKNOW, |

##### RST\_GetWakeUpSource

|  |  |
| --- | --- |
| **原型** | U32 RST\_GetWakeUpSource(void) |
| **描述** | 取得本次唤醒源。 |
| **参数** | none |
| **返回值** | #define WAKE\_KLR WKPU\_WAKEUP\_3  #define WAKE\_KL15 WKPU\_WAKEUP\_13  #define WAKE\_DLIS WKPU\_WAKEUP\_9  #define WAKE\_WakeUpEnINT WKPU\_WAKEUP\_8  #define WAKE\_CAN1RX WKPU\_WAKEUP\_7  #define WAKE\_CAN2RX WKPU\_WAKEUP\_6  #define WAKE\_CAN3RX WKPU\_WAKEUP\_4  #define WAKE\_CAN5RX WKPU\_WAKEUP\_5  #define WAKE\_BATTERY (0xFF000000u) |

##### SCM\_SetChannelMode

|  |  |
| --- | --- |
| **原型** | U8 SCM\_SetChannelMode(U8 Index, U8 Mode) |
| **描述** | 设置驱动通道模式 |
| **参数** | U8 Index：驱动通道索引  (0x00u)：INDEX\_SCM\_KLRSW  (0x01u)：INDEX\_SCM\_ACCWAKEUP,  (0x02u)：INDEX\_SCM\_KL15RELAY,  (0x03u)：INDEX\_SCM\_WAKEUPENABLE,  (0x04u)：INDEX\_SCM\_IGNRELAY  U8 Mode ：驱动通道输出状态  (0x00u)：MODE\_SCM\_CHANNEL\_OFF  (0x01u)：MODE\_SCM\_CHANNEL\_ON |
| **返回值** | (0x00u)：STATUS\_SCM\_ERR 执行过程错误  (0x01u)：STATUS\_SCM\_OK 正确  (0x02u): STATUS\_SCM\_INVALID输入参数错误 |

##### v\_clear\_flag

|  |  |
| --- | --- |
| **原型** | void v\_clear\_flag(v\_fl\_handle f); |
| **描述** | Sets the current value of the flag specified by the handle f to zero. |
| **参数** | F: Flag handle |
| **返回值** | None. |

##### v\_ctl\_busoffint

|  |  |
| --- | --- |
| **原型** | void v\_ctl\_busoffint(v\_c\_handle c); |
| **描述** | Performs controller-specific interrupt handling. |
| **参数** | c: controller handle |
| **返回值** | None. |

##### v\_ctl\_con

|  |  |
| --- | --- |
| **原型** | void v\_ctl\_con(v\_c\_handle c); |
| **描述** | Connects the controller to the network |
| **参数** | c: controller handle |
| **返回值** | None. |

##### v\_ctl\_discon

|  |  |
| --- | --- |
| **原型** | void v\_ctl\_discon(v\_c\_handle c); |
| **描述** | Disconnects the controller to the network |
| **参数** | c: controller handle |
| **返回值** | None. |

##### v\_ctl\_init

|  |  |
| --- | --- |
| **原型** | v\_uint8 v\_ctl\_init(v\_c\_handle c); |
| **描述** | Performs hardware-specific initialisation of the controller specified by c. |
| **参数** | c: controller handle |
| **返回值** | None. |

##### v\_ctl\_rxint

|  |  |
| --- | --- |
| **原型** | v\_imf\_handle v\_ctl\_rxint(v\_c\_handle c); |
| **描述** | Performs controller-specific interrupt handling. |
| **参数** | c: controller handle |
| **返回值** | None. |

##### v\_ctl\_txint

|  |  |
| --- | --- |
| **原型** | v\_imf\_handle v\_ctl\_txint(v\_c\_handle c); |
| **描述** | Performs controller-specific interrupt handling. |
| **参数** | c: controller handle |
| **返回值** | None. |

##### v\_fmd\_set

|  |  |
| --- | --- |
| **原型** | void v\_fmd\_set(v\_c\_handle c, v\_fmd\_handle fmd); |
| **描述** | switch to the specified frame mode for controller c. |
| **参数** | c: controller handle  fmd : the frame mode for controller c |
| **返回值** | None. |

##### v\_gateway

|  |  |
| --- | --- |
| **原型** | void v\_gateway(v\_cfg\_handle c); |
| **描述** | Copies incoming signals to outgoing signals |
| **参数** | c: network configuration handle |
| **返回值** | None. |

##### v\_imf\_rx

|  |  |
| --- | --- |
| **原型** | v\_bool v\_imf\_rx(v\_imf\_handle f); |
| **描述** | Reads the immediate frame |
| **参数** | F: immediate frame handle |
| **返回值** | The value of frame |

##### v\_imf\_tx

|  |  |
| --- | --- |
| **原型** | void v\_imf\_tx(v\_imf\_handle f); |
| **描述** | Sets the contents of the immediate frame |
| **参数** | F: immediate frame handle |
| **返回值** | None. |

##### v\_input

|  |  |
| --- | --- |
| **原型** | void v\_input(v\_cfg\_handle c); |
| **描述** | Performs input processing |
| **参数** | c: network configuration handle |
| **返回值** | None. |

##### v\_nwm\_fmd\_set

|  |  |
| --- | --- |
| **原型** | v\_bool v\_nwm\_fmd\_set ( v\_nwm\_instance\_handle nwm,  const v\_fmd\_handle \*fmd ); |
| **描述** | changes the frame mode changes the frame mode |
| **参数** | Nwm: network management instance  Fmd: frame mode |
| **返回值** | None. |

##### v\_nwm\_get\_slave\_nodes\_status

|  |  |
| --- | --- |
| **原型** | v\_uint32 v\_nwm\_get\_slave\_nodes\_status ( v\_nwm\_instance\_handlenwm ); |
| **描述** | returns the status, absent or present, of each slave node |
| **参数** | Nwm : The network management instance. |
| **返回值** | Bit = 0 : node is absent  Bit = 1 : node is present |

##### v\_nwm\_get\_state

|  |  |
| --- | --- |
| **原型** | v\_nwm\_state v\_nwm\_get\_state( v\_nwm\_instance\_handle nwm ); |
| **描述** | returns the current state of the state machine for the network management instance |
| **参数** | Nwm : The network management instance. |
| **返回值** | V\_NWM\_STARTUP (0x01)  V\_NWM\_NORMAL (0x02)  V\_NWM\_BUSOFF (0x03)  V\_NWM\_BUSOFF\_WAIT (0x04)  V\_NWM\_NETWORK\_SLEEP (0x05)  V\_NWM\_WAIT\_FOR\_NETWORK\_SLEEP (0x06)  V\_NWM\_WAKEUP\_NETWORK (0x07)  V\_NWM\_WAKEUP\_PENDING (0x08)  V\_NWM\_EXPULSION (0x09)  V\_NWM\_ISOLATED (0x0A)  V\_NWM\_STOPPED (0x0B) |

##### v\_nwm\_init

|  |  |
| --- | --- |
| **原型** | v\_uint8 v\_nwm\_init ( v\_cfg\_handle c, const v\_nwm\_instance\_handle nwm[ ], const v\_uint8 num\_of\_nwm); |
| **描述** | initialises the NWM module |
| **参数** | C : network configuration handle  Nwm : nwm instance handle  Num\_of\_nwm : the number of nwm CAN |
| **返回值** | 0 ： The initialisation succeeded.  0x01 – 0x7F ：Volcano initialisation failed. The return value is according to the return value from v\_sys\_init()  0x80 ： Controller initialisation failed. One or more controller has failed during initialisation in one or more NWM instances. |

##### v\_nwm\_master\_processing

|  |  |
| --- | --- |
| **原型** | v\_bool v\_nwm\_master\_processing ( v\_nwm\_instance\_handle nwm); |
| **描述** | NWM processing function |
| **参数** | F: Flag handle |
| **返回值** | None. |

##### v\_nwm\_resume

|  |  |
| --- | --- |
| **原型** | void v\_nwm\_resume ( v\_nwm\_instance\_handle nwm ); |
| **描述** | connects the controller(s) in the network management instance nwm to the bus andtransfers nwm to the state Network Sleep, |
| **参数** | Nwm : The network management instance. |
| **返回值** | None. |

##### v\_nwm\_set\_network\_mode

|  |  |
| --- | --- |
| **原型** | void v\_nwm\_set\_network\_mode(v\_nwm\_instance\_handle nwm,v\_uint8 v); |
| **描述** | sets the value of the signal NetworkMode |
| **参数** | Nwm : The network management instance. |
| **返回值** | None. |

##### v\_nwm\_stop

|  |  |
| --- | --- |
| **原型** | void v\_nwm\_stop ( v\_nwm\_instance\_handle nwm ); |
| **描述** | disconnects the controller |
| **参数** | Nwm : The network management instance. |
| **返回值** | None. |

##### v\_output

|  |  |
| --- | --- |
| **原型** | void v\_output(v\_cfg\_handle c); |
| **描述** | Performs output processing |
| **参数** | C : network configuration handle |
| **返回值** | None. |

##### v\_rd\_16

|  |  |
| --- | --- |
| **原型** | v\_uint16 v\_rd\_16(v\_s\_16\_handle s); |
| **描述** | Reads and returns the current value of the signal specified by the handle s |
| **参数** | S: signal handle |
| **返回值** | The value of signal s |

##### v\_rd\_8

|  |  |
| --- | --- |
| **原型** | v\_uint8 v\_rd\_8(v\_s\_8\_handle s); |
| **描述** | Reads and returns the current value of the signal specified by the handle s |
| **参数** | S: signal handle |
| **返回值** | The value of signal s |

##### v\_rd\_bytes

|  |  |
| --- | --- |
| **原型** | void v\_rd\_bytes(v\_s\_bytes\_handle s, v\_uint8 \*dest, v\_uint8 n, v\_uint8 o); |
| **描述** | Copies n bytes of the current value of the signal |
| **参数** | S: signal handle  Dest : the destenation pointer  N : the number of bytes  O : the offset |
| **返回值** | None. |

##### v\_sb\_tick

|  |  |
| --- | --- |
| **原型** | v\_bool v\_sb\_tick(v\_c\_handle c); |
| **描述** | follows a schedule for each sub-bus network interface. |
| **参数** | C : controller handle |
| **返回值** | None. |

##### v\_test\_flag

|  |  |
| --- | --- |
| **原型** | v\_bool v\_test\_flag(v\_fl\_handle f); |
| **描述** | Returns a C boolean indicating the current state of the flag specified by f |
| **参数** | F: Flag handle |
| **返回值** | 0 : the flag is clear,  Non-zero: otherwise. |

##### v\_wr\_8

|  |  |
| --- | --- |
| **原型** | void v\_wr\_8(v\_s\_8\_handle s, v\_uint8 v); |
| **描述** | Sets the current value of the signal specified by the handle s to the value of v. |
| **参数** | S : single handle  V: the value set to S |
| **返回值** | None. |

##### v\_wr\_bytes

|  |  |
| --- | --- |
| **原型** | void v\_wr\_bytes(v\_s\_bytes\_handle s, v\_uint8 \*src, v\_uint8 n, v\_uint8 o); |
| **描述** | Sets n bytes of the current value of the signal specified by the handle s to the sequence of bytes beginning in the location pointed to by src |
| **参数** | S : signal handle  Src : source pointer  V : the value  O : offset |
| **返回值** | None. |

#### 输出接口

##### COM\_Init

|  |  |
| --- | --- |
| **原型** | void COM\_Init(void) |
| **描述** | 初始化网络管理、VTP interface(CAN/LIN控制器)、上电后开启网络（硬线唤醒与网络唤醒）、初始化TPM和DSM |
| **参数** | NA |
| **返回值** | NA |

##### COM\_Process\_5ms

|  |  |
| --- | --- |
| **原型** | void COM\_Process\_5ms(void) |
| **描述** | VTP 执行函数 |
| **参数** | NA |
| **返回值** | NA |

##### COM\_GetNetWakeSource

|  |  |
| --- | --- |
| **原型** | U8 COM\_GetNetWakeSource(void); |
| **描述** | 网络唤醒源获取函数 |
| **参数** | NA |
| **返回值** | NWM\_WAKE\_UP  NOT\_NWM\_WAKE\_UP |

##### COM\_GetNetComDiagFlag

|  |  |
| --- | --- |
| **原型** | BOOL COM\_GetNetComDiagFlag(void); |
| **描述** | 网络通信1s后诊断开启标志位函数 |
| **参数** | NA |
| **返回值** | TRUE ： 开启  FALSE： 关闭 |

##### COM\_GetNetComFlag

|  |  |
| --- | --- |
| **原型** | BOOL COM\_GetNetComFlag(void); |
| **描述** | 网络通信开启标志位函数 |
| **参数** | NA |
| **返回值** | TRUE ： 开启  FALSE： 关闭 |

##### COM\_GetAllNetSleepFlag

|  |  |
| --- | --- |
| **原型** | BOOL COM\_GetAllNetSleepFlag(void); |
| **描述** | 网络全部睡眠标志位函数 |
| **参数** | NA |
| **返回值** | TRUE ： 开启  FALSE： 关闭 |

##### COM\_SetAllTrcv

|  |  |
| --- | --- |
| **原型** | void COM\_SetAllTrcv(U8 status) |
| **描述** | CAN收发器控制函数 |
| **参数** | NA |
| **返回值** | COM\_MODULE\_SHUT\_DOWN： 关闭Transceiver  COM\_MODULE\_NORMAL： 开启Transceiver |

#### 验证准则

NA

### 内部接口

#### com\_VTPProcess\_5ms

|  |  |
| --- | --- |
| **原型** | void com\_VTPProcess\_5ms(void) |
| **描述** | 网络VTP函数 |
| **参数** | NA |
| **返回值** | NA |

#### com\_NWMModeCheck\_10ms

|  |  |
| --- | --- |
| **原型** | void com\_NWMModeCheck\_10ms(void) |
| **描述** | 网络管理状态检查函数：网络休眠时间RouteControl |
| **参数** | NA |
| **返回值** | NA |

#### com\_NetworkAllSleepCheck\_10ms

|  |  |
| --- | --- |
| **原型** | void com\_NetworkAllSleepCheck\_10ms(void) |
| **描述** | 检查是否所有网络进入睡眠状态函数 |
| **参数** | NA |
| **返回值** | NA |

#### com\_NWMProcess\_10ms

|  |  |
| --- | --- |
| **原型** | void com\_NWMProcess\_10ms(void) |
| **描述** | 网络管理执行函数 |
| **参数** | NA |
| **返回值** | NA |

#### com\_NetworkProcess\_10ms

|  |  |
| --- | --- |
| **原型** | void com\_NetworkProcess\_10ms(void) |
| **描述** | 网络状态执行函数 |
| **参数** | NA |
| **返回值** | NA |

#### com\_HwComSubCheck

|  |  |
| --- | --- |
| **原型** | void com\_HwComSubCheck(void); |
| **描述** | 硬线唤醒开关检查子函数 |
| **参数** | NA |
| **返回值** | NA |

#### com\_HwComCheck\_10ms

|  |  |
| --- | --- |
| **原型** | void com\_HwComCheck\_10ms (void) |
| **描述** | 硬线唤醒开关检查函数 |
| **参数** | NA |
| **返回值** | NA |

#### com\_ComDiagFlagCheck\_10ms

|  |  |
| --- | --- |
| **原型** | void com\_ComDiagFlagCheck\_10ms(void); |
| **描述** | 网络通信故障诊断开关检查函数 |
| **参数** | NA |
| **返回值** | NA |

#### com\_VehSpdComCheck\_10ms

|  |  |
| --- | --- |
| **原型** | void com\_VehSpdComCheck\_10ms(void) |
| **描述** | 车速保持网络开关检查函数 |
| **参数** | NA |
| **返回值** | NA |

#### v\_nmco\_ioctl

|  |  |
| --- | --- |
| **原型** | v\_uint16 v\_nmco\_ioctl( v\_c\_handle c, v\_uint16 op ) |
| **描述** | NM callout function block must add because the new version COM software |
| **参数** | NA |
| **返回值** | NA |

#### com\_CheckKeyInOutState

|  |  |
| --- | --- |
| **原型** | U8 com\_CheckKeyInOutState(void) |
| **描述** | 检查低配钥匙IN/OUT动作 |
| **参数** | NA |
| **返回值** | 0：FALSE： key without In/Out change  1：TRUE： key with In/Out change |

#### com\_CheckiWakeFlag

|  |  |
| --- | --- |
| **原型** | BOOL com\_CheckiWakeFlag(void) |
| **描述** | 检查网络管理帧与7DF帧唤醒 |
| **参数** | NA |
| **返回值** | TURE ： 网络上存在唤醒帧或7DF  FALSE ： 网络上不存在唤醒帧或7DF |

#### com\_CheckiKeepFlag

|  |  |
| --- | --- |
| **原型** | U8 com\_CheckiKeepFlag(void) |
| **描述** | 检查网络管理帧与7DF帧保持 |
| **参数** | NA |
| **返回值** | TURE ： 网络上存在保持帧或7DF  FALSE ： 网络上不存在保持帧或7DF |

#### com\_VTPInterfaceinit

|  |  |
| --- | --- |
| **原型** | void com\_VTPInterfaceinit(void); |
| **描述** | 网络通信接口初始化函数 |
| **参数** | NA |
| **返回值** | NA |

#### com\_TrcvInit

|  |  |
| --- | --- |
| **原型** | void com\_TrcvInit(void); |
| **描述** | CAN收发器transceiver初始化函数 |
| **参数** | NA |
| **返回值** | NA |

#### com\_GetNetworkKeepTime

|  |  |
| --- | --- |
| **原型** | U16 com\_GetNetworkKeepTime(com\_NetworkComCategory\_ENU); |
| **描述** | 网络保持时间获取函数 |
| **参数** | Index： NWM\_COM ： 网络唤醒后的网络保持时间  HW\_COM ： 硬线唤醒后的网络保持时间 |
| **返回值** | 网络保持时间 |

#### 网络中断

|  |  |
| --- | --- |
| **原型** | CAN\_Mailbox\_Interrupt\_0  CAN\_Mailbox\_Interrupt\_1  CAN\_Mailbox\_Interrupt\_2  CAN\_Mailbox\_Interrupt\_3  CAN\_Mailbox\_Interrupt\_5  CAN\_Error\_Interrupt\_0  CAN\_Error\_Interrupt\_1  CAN\_Error\_Interrupt\_2  CAN\_Error\_Interrupt\_3  CAN\_Error\_Interrupt\_5  LIN\_Rx\_Interrupt\_0  LIN\_Tx\_Interrupt\_0  LIN\_Error\_Interrupt\_0 |
| **描述** | CAN、LIN中断函数 |
| **参数** | NA |
| **返回值** | NA |

#### VTP备用

|  |  |
| --- | --- |
| **原型** | v\_ctl\_ints\_disable  v\_ctl\_ints\_restore  v\_co\_tx\_frame\_checksum  v\_co\_rx\_frame\_checksum |
| **描述** | 供VTP使用 |
| **参数** | NA |
| **返回值** | NA |

#### 验证准则

NA

# 数据结构设计

注意

1. 所有数据结构除结构体外，都加AAA/ZZZ/ALL成员
2. 结构体长度要尽量保证为integer长度的整数倍

# 变量设计

## com\_AllFlag\_tag

### 定义

typedef struct com\_SingleFlag\_tag  
{  
 U16 NetComFlag : 1;  
 U16 HWComFlag : 1;  
 U16 VehSpdComFlag : 1;  
 U16 NwmComFlag : 1;  
 U16 AllNetSleepFlag : 1;  
 U16 DiagFrameFlag : 1;  
 U16 NetComDiagFlag :1 ;  
 U16 HSC3WakeFlag :1 ;  
 U16 HSC3KeepFlag :1 ;  
 U16 HSC5WakeFlag :1 ;  
 U16 HSC5KeepFlag :1 ;  
 U16 reserve : 5 ;  
}com\_SingleFlag\_ST;

typedef union com\_AllFlag\_tag  
{  
 U16 AllFlag;  
 com\_SingleFlag\_ST SingleFlag;  
}com\_AllFlag\_UN;

### 描述

Com模块所有flag，用位域表示

# 函数设计

# 未解决问题

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **事项** | **评价 /建议方案** | **人员职责** | **解决日期** |
|  | N/A | N/A | N/A | N/A |
|  | N/A | N/A | N/A | N/A |

# 附录

N/A

# 模板更改记录

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订原因** | **修订日期** | **作者** |
| 003 | 1.删除swad设计文档的内容；  2.增加验证准则的内容；  3.将外部接口分为输入与输出两部分； | 2014/12/15 | 曹娣 |
| 002 | 增加文档修订功能 | 2014/3/19 | 曹娣 |
| 001 | 初始版 | 2013/7/9 | 杨沉陈 |
|  |  |  |  |
|  |  |  |  |