

# NS9360 ITRON SDK SNMP&MIB Custom Guide



1.	はじめに	3
2.	ファイル構成	3
2.1	ディレクトリ構成	3
2.2	ファイル構成	3
2.3	SNMPファイル詳細	4
2.3.1	ヘッダファイル	4
2.3.2	ソースファイル	4
2.4	MIB情報ファイル詳細	5
2.4.1	ヘッダファイル	5
2.4.2	ソースファイル	6
3.	MIBの登録	7
3.1	MIB情報の登録	7
3.1.1	MIB情報 登録テーブルの説明	7
3.1.2	固定OIDの登録	13
3.2	登録済みMIB	14
3.2.1	Systemグループ	14
3.2.2	IPグループ	15
3.2.3	ICMPグループ	16
3.2.4	TCPグループ	16
3.2.5	UDPグループ	17
3.2.6	SNMPグループ	17
3.3	TRAP情報の登録	18
3.3.1	トラップ情報登録エリア	18
3.3.2	トラップ登録テーブル	20
4.	SNMPパラメータ設定	21
4.1	コミュニティ名の定義	21
4.2	フレーム受信ポートの設定	22
4.3	トラップ送信先UDPポートの設定	22
4.4	トラップ送信先IPアドレスの設定	23
4.5	トラップ送信Enableフラグ	23

## 1. はじめに

本書は、AZ9360SDK上で動作するSNMPのパラメータと、SNMPでアクセス可能なMIB情報のカスタマイズ方法について明記します。

## 2. ファイル構成

### 2.1 ディレクトリ構成

AZ9360SDKをインストール後、以下のフォルダにMIB用のファイルが展開されます。

¥itron¥tcp¥

MIB_TABLE.C	MIB 情報登録テーブルファイル
MIB_SYS.C	標準 MIB SYSTEM グループ用アクセス関数群ファイル
MIB_IF.C	標準 MIB I/F グループ用アクセス関数群ファイル
MIB_TABLE.H	MIB 情報登録テーブルヘッダファイル
MIB_USER.H	MIB 情報ユーザーカスタマイズヘッダファイル

### 2.2 ファイル構成

#### (1) SNMP関連ファイル

ファイル名	ファイル種別	説明
SNMP_USER.C	Source	SNMP パラメータ設定用 Source ファイルです。
PS_SNMP.H	Header	SNMP の Header ファイルです。
SNMP_IF.H	Header	SNMP-APL 間 I/F 宣言ファイルです。
SNMP_DATA.H	Header	プロトコルスタック内の MIB データエリアの宣言ファイルです。

#### (2) MIB関連ファイル

ファイル名	ファイル種別	説明
MIB_TABLE.C	Source	MIB 情報登録ファイルです。
MIB_SYS.C	Source	標準 MIB system グループ用サンプル Source ファイルです。
MIB_IF.C	Source	標準 MIB interface グループ用サンプル Source ファイルです。
MIB_TABLE.H	Header	MIB 登録用のヘッダファイルです。
MIB_USER.H	Header	MIB 登録のうち、ユーザーカスタマイズの必要な項目を集めたヘッダファイルです。

## 2.3 SNMPファイル詳細

### 2.3.1 ヘッダファイル

- (1) `PS_SNMP.H`  
SNMP用ヘッダファイルです。  
ユーザカスタマイズの必要な項目はありません。
- (2) `SNMP_IF.H`  
SNMPの開始/停止、およびTrap送信の制御用I/F関数のプロトタイプ宣言を行っているファイルです。  
ユーザカスタマイズの必要な項目はありません。
- (3) `SNMP_DATA.H`  
ySOCK6内部で使用するMIBデータエリアの宣言ファイルです。  
ユーザカスタマイズの必要な項目はありません。  
また、APLでインクルードを行う必要もありません。

### 2.3.2 ソースファイル

- (1) `SNMP_USER.C`  
SNMPパラメータの初期化関数群です。

## 2.4 MIB情報ファイル詳細

### 2.4.1 ヘッダファイル

#### (1) MIB\_TABLE.H

各種MIB情報用の構造体宣言等を行っているファイルです。  
ユーザカスタマイズを行う項目はありません。

#### (2) MIB\_USER.H

MIB情報の登録を行う際に、ユーザカスタマイズが必要なものを宣言しているファイルです。  
以下に各項目の説明を行います。

##### ①#define 宣言

・#define ENTRY\_MIB\_NUM

MIB情報テーブルに登録するMIBの数を示します。

サンプルでは、“109”となっています。

尚、ストップ分は、除いてください。

・#define ENTRY\_FIXOID\_NUM

MIB番号の固定番号登録テーブルに登録する固定番号情報の数を示します。

サンプルでは、標準MIB群に登録しているのので、“1”となっています。

最大3（3件分）まで指定可能です。

・#define ENTRY\_TRAP\_NUM

トラップ送信情報テーブルに登録するTrapの数を示します。

最大16まで指定（登録）可能です。

・#define ENTRY\_TRAPID\_COMMUNITY

コミュニティ名NGTrapの登録IDを示します。

・#define ENTRY\_TRAPID\_BOOT

ColdスタートTRAPの登録IDを示します。

・#define ENTERPRISE\_OID\_LEN

エンタープライズ番号のレンジを示します。

・#define ENTERPRISE\_OID

エンタープライズ番号を示します。

・#define PRIVATEMIB\_BASE\_OID\_LEN

プライベートMIBのベースMIB番号レンジを示します。

・#define PRIVATEMIB\_BASE\_OID

プライベートMIBのベースMIB番号を示します。

##### ②プロトタイプ宣言

MIBデータのGet/Set関数等、MIB情報テーブルに登録する関数のプロトタイプ宣言を行ってください。

## 2.4.2 ソースファイル

- (1) `MIB_TABLE.C`  
SNMPで読み出すMIB情報を登録するテーブル群ファイルです。ユーザーでカスタマイズすることができます。
- (2) `MIB_SYS.C`  
標準MIBのSystemグループにあるMIBデータのアクセス関数群です。ユーザーでカスタマイズすることができます。
- (3) `MIB_IF.C`  
標準MIBのInterfaceグループにあるMIBデータのアクセス関数群です。ユーザーでカスタマイズすることができます。

### 3. MIBの登録

本SNMPでは、使用するMIBの情報を、独自のテーブルに登録する必要があります。

#### 3.1 MIB情報の登録

`mib__table.c`に、サポートするMIBオブジェクトの情報を登録するテーブルが宣言されています。

このテーブルに必要な情報をセットすることにより、MIBオブジェクトの登録をすることが可能です。

##### 3.1.1 MIB情報 登録テーブルの説明

MIB情報を登録するテーブルの構造は、あらかじめMIB\_\_TABLE.Hに宣言されています。

##### (1) テーブル構造体

```
struct MIB_INFO_TBL {
    U16      ObjectLength;          /* オブジェクトレングス */
    U8       ObjectID[32];         /* オブジェクト ID */
    U16      AccessLevel;          /* アクセスレベル */
    U16      DataKind;             /* データ種別 */
    T_DATA_ACCESS Read;            /* Read データのアクセスポインタ */
    T_DATA_ACCESS Write;           /* Write データのアクセスポインタ */
    U32      (*GetObjectType) ();  /* MIBデータ数取得関数のポインタ */
    U32      (*GetInstanceIdentifier) (); /* インスタンス識別子取得関数のポインタ */
    struct MIB_INFO_TBL *NextMib; /* 次の登録 MIB テーブルポインタ */
};
typedef struct MIB_INFO_TBL MIB_TBL;
```

##### (2) テーブル宣言

```
const MIB_TBL SNM_MibTable[ENTRY_MIB_NUM+1] = {
    .
    .
    .
    /*== [] ストップ (テーブルの最後には必ずこのデータを入れる) ==*/
    {TABLE_STOPPER, /* オブジェクトレングス */
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, /* オブジェクト ID */
    TABLE_STOPPER, /* アクセスレベル */
    TABLE_STOPPER, /* データ種別 */
    (void *)TABLE_STOPPER, /* Read 関数のポインタ */
    (void *)TABLE_STOPPER, /* Write 関数のポインタ */
    (void *)TABLE_STOPPER, /* 予約エリア 1 */
    (void *)TABLE_STOPPER, /* 予約エリア 2 */
    (MIB_TBL *)TABLE_STOPPER /* 予約エリア 3 */
    }
};
```

(3) 各メンバ説明

① ObjectLength

登録MIBのMIB番号（以下OID）のレングスを設定するエリアです。  
このエリアのサイズは、16bitですが、設定可能な値は、1～32までとなります。

ex) iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1).sysDescr(1)を登録する場合  
ObjectLength=8となります。

② ObjectID[32]

登録MIBのOIDを設定するエリアです。  
32桁まで設定することが可能です。

ex) iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1).sysDescr(1)を登録する場合  
ObjectID[32]=[1,3,6,1,2,1,1,1,0,0……0]となります。

③ AccessLevel

MIBオブジェクトのアクセスレベルを設定するエリアです。  
アクセスレベルとは、Read Only/Read Write/Not Accessible  
のことを示します。  
このエリアに設定できる値は、以下の通りとなっております。  
設定するには、必ずシンボルを使用してください。

表3.1.1-1 アクセスレベル一覧

アクセスレベル	コード	シンボル	備考
Read Only	0x01	READ_ONLY	
Read/Write	0x03	READ_WRITE	
Not Access	0x8000	NOT_ACCESSIBLE	

ex) iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1).sysDescr(1)を登録する場合  
アクセスレベル=READ\_ONLYとなります。

④ DataKind

MIBデータのデータ種別及び、データのGet/Setを行う際のアクセス方法を設定するエリアです。

本エリアには、データ種別と、アクセス方法をORSETした情報を設定してください。

データ種別として設定できる値は、以下のとおりです。実際に設定を行う場合は、シンボル定義を使用してください。

表 3.1.1-2 データ種別一覧

データ種別	コード	シンボル定義	備考
INTEGER	0x02	INTEGER	MIBデータが整数値の場合に使用する
OCTET_STRING	0x04	OCTET_STRING	MIBデータが文字列の場合に使用する
OBJECT_IDENTIFIER	0x06	OBJECT_IDENTIFIER	MIBデータが別のOID(プライベートMIB等)の場合に使用する
PhysAddress	0x04	PHYS_ADDRESS	MIBデータがH/Wアドレスの場合に使用する。
IpAddress	0x40	IP_ADDRESS	MIBデータがIPアドレスの場合に使用する
Counter	0x41	COUNTER	MIBデータがカウンタの場合に使用する
Gauge	0x42	GAUGE	
TimeTicks	0x43	TIME_TICKS	MIBデータが10msec積算タイマ値の場合に使用する
Opaque	0x44	OPAQUE	

アクセス方法は、MIBデータのGet/Setを行う際に、データを設定しているエリアを直接アクセスするか、関数経由にてアクセスするかの識別フラグです。

本フラグの設定により、以降Read及びWriteエリアに設定する内容が変わります。

Read及び、Writeエリアにつきましては、次項を参照ください。

尚、アクセス方法には以下の制限次項があります。

i) エリア直接制御を行えるのは、データ種別が、32bitデータの時だけです。

(32bitデータ=Integer, Counter, Gauge, TimeTicks, Opaque)

その他のデータ種別の場合は、必ず関数経由アクセスとしてください。

ii) テーブル型MIBのデータアクセスにつきましては、データ種別にかかわらず、必ず関数経由アクセスを設定してください。

表 3.1.1-2 アクセス方法一覧

シンボル定義	コード	アクセス方法
ACFLG_AREA	0xC000	エリア直接アクセス
ACFLG_FUNC	0x8000	関数経由アクセス

ex) iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1).sysDescr(1)を登録する場合  
DataKind=ACFLG\_FUNC | OCTET\_STRINGとなります。

⑤ Read

MIBデータのGetを行う場合の、アクセスポインタを設定するエリアです。  
DataKind エリアにて設定したアクセス方法により、MIBデータが格納されているエリアのポインタか、MIBデータをReadする関数のポインタかどちらかを設定してください。

i) T\_DATA\_ACCESS の構成

T\_DATA\_ACCESS は、MIB\_TABLE.H にあらかじめ宣言されています。

```
typedef union {
    U32          *Dataptr;      /* エリア直接アクセス用ポインタ */
    S32          (*Func)();    /* 関数経由アクセス用ポインタ */
} T_DATA_ACCESS;
```

ii) Read 関数の定義

関数経由にて MIB データを Read する場合は、以下の条件に沿って関数を作成し、その関数のポインタを Read エリアに設定してください。

・関数定義

```
S32 ReadFunc (T_MIBDATA *ptr, U32 InstNum)
```

・引数

SNMPより Read 関数を CALL する際には、以下の引数を使用いたします。

```
T_MIBDATA *pt      = Read データを格納するエリアのポインタ
U32 InstNum       = 読み出すデータの Index 値 (0~)
```

T\_MIBDATA につきましては、MIB\_TABLE.H にあらかじめ宣言されています。

```
typedef struct {
    U32          length;      /* Read データの長さ数格納先エリア */
    union {
        U32      *d_32bit;   /* 32bit データ以外の場合に使用 */
        U8       *d_str;     /* 32bit データ以外のデータ格納先ポインタ */
    } DataPtr;
} T_MIBDATA;
```

・戻り値

```
1      : 正常終了
0      : 異常終了
```

※関数名は、ユーザーにて任意に決定してください。

## ⑥Write

MIBデータのSetを行う場合の、アクセスポインタを設定するエリアです。  
DataKind エリアにて設定したアクセス方法により、MIBデータを格納するエリアのポインタか、  
MIBデータをWriteする関数のポインタかどちらかを設定してください。

### i) T\_DATA\_ACCESS の構成

⑤Read と同一の共用体を使用します。

### ii) Write関数の定義

関数経由にてMIBデータをWriteする場合は、以下の条件に沿って関数を作成し、その関数の  
ポインタをWriteエリアに設定してください。

#### ・関数定義

```
S32 WriteFunc (T_MIBDATA *ptr, U32 InstNum)
```

#### ・引数

```
T_MIBDATA *pt      = Readデータを格納するエリアのポインタ  
U32 InstNum       = 読み出すデータのIndex値 (0~)
```

T\_MIBDATA につきましては、MIB\_TABLE.H にあらかじめ宣言されています。

```
typedef struct {  
    U32      length; /* Writeデータのレングス数を示す */  
    union { /* 32bitデータ以外の場合に使用します */  
        U32 *d_32bit; /* 32bitデータの場合のWriteデータ格納ポインタ */  
        U8  *d_str; /* 32bitデータ以外の場合のWriteデータ格納先ポインタ */  
    } DataPtr;  
} T_MIBDATA;
```

#### ・戻り値

```
1      : 正常終了  
0      : 異常終了
```

※関数名は、ユーザーにて任意に決定してください。

#### ⑦ GetObjectType

MIBが保持しているデータ数を取得する関数のポインタを登録するエリアです。  
MIBがスカラー型の場合は、保持しているデータ数は1となります。

U32 GetObjectType(void)

引数 : なし

戻り値 : 1~0xffffffff (データ数)

※関数名は、ユーザーにて任意に決定してください。

#### ⑧ GetInstIdentifier

OIDに付加するインスタンス識別子を取得する関数のポインタを登録するエリアです。  
MIBオブジェクトがスカラー型の場合は、インスタンス識別子は、必ず“0”となります。

U32 GetInstance(U32 InstNum, U8 \*Area)

引数 : InstNum= 要求しているデータのインデックス値  
\*Area = インスタンス識別子を格納するエリアのポインタ

戻り値 : インスタンス番号のレングス (≥1)

※関数名は、ユーザーにて任意に決定してください。

#### ⑨ NextMib

MIBOID上、次に位置するMIBオブジェクトの情報テーブルのポインタを登録するエリアです。  
最後に位置している場合は、NULLコードを設定してください。

#### (4) MIB登録の注意

MIB情報テーブルに、MIBを設定する場合は、OIDの昇順に並べて設定してください。  
昇順に並べていない場合、SNMPマネージャからのGetNextフレームに対する応答が正常に行えなくなる場合がありますので、ご注意ください。

### 3.1.2 固定OIDの登録

SNMPマネージャからの要求OIDを、MIB情報テーブルから検索する際に、検索処理を簡素化するため、OIDの先頭部分からの共通部分を指定していただく必要があります。

(以降、共通部分を固定OIDといたします。)

標準MIB（標準MIB）のみ登録する場合は、サンプルソースのままとしておくことをお勧めします。別途、プライベートMIBを登録する場合は、標準MIBのあとにプライベートMIBの固定OIDを登録してください。

#### (1) 固定OID 登録テーブルの構成

```
typedef struct {
    U16    Length;          /* 固定OIDのレングス          */
    U8     Oid[32];        /* OID                          */
}FIX_OID_TBL;
```

以下に各メンバの説明を記載いたします。

尚、設定サンプルにつきましては、MIB\_\_TABLE.Cを参照願います。

##### ① Length

固定OIDのレングス数を設定いたします。

16bitエリアですが、実際は、1～32までの範囲を設定可能としています。

##### ② Oid[32]

固定OIDを設定いたします。

MAX32桁まで設定可能としています。

#### (2) 固定OID登録テーブルの宣言

MIB\_\_TABLE.Cに固定OID登録テーブルの宣言を行っております。

必要に応じて、本テーブルに設定の追加&修正を行ってください。

```
const    FIX_OID_TBL    SNM_FixOidTbl [ENTRY_FIXOID_NUM+1] = {
    {5, {1, 3, 6, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}},
    .
    .
    .
    /*== [] ストップ (テーブルの最後には必ずこのデータを入れる) ==*/
    {TABLE_STOPPER, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}
};
```

#### (3) 固定OID設定時の注意事項

MIBの検索を行う際に、下3桁以上のOIDを使用します。そのため、固定OIDを設定する場合、固定OIDを除いたときの、残りのMIBのOIDが3桁以上になるようにしてください。

※サンプル設定での、iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1).sysDescr(1)の残りOIDは、1.1.1となります。

### 3.2 登録済みMIB

あらかじめ標準MIBのうち、必須項目となっているグループ及び、SNMPグループをサンプルとして、登録しております。

登録済みMIB以外に必要なMIBがある場合は、ソースファイルを参考に登録を行ってください。

また、あらかじめ登録してあるMIBが不要な場合は、コメントアウトにより、削除可能となっております。

登録済みの標準MIBは、以下のとおりです。

Systemグループ  
IPグループ  
ICMPグループ  
TCPグループ  
UDPグループ  
SNMPグループ

#### 3.2.1 Systemグループ

##### (1) MIB構成

```
iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1)
  sysDescr(1)
  sysObjectID(2)
  sysUpTime(3)
  sysContact(4)
  sysName(5)
  sysLocation(6)
  sysServices(7)
```

##### (2) 注意事項

sysObjectID および、sysUpTime は Trap フレーム送信時に必要な MIB です。  
標準 MIB を使用しない場合でも、この 2 項目は残しておくことをお勧めします。

### 3.2.2 IPグループ

ipForwardTable につきましては、ySOCK2 未サポート機能についての情報であるため、削除しております。

#### (1) MIB構成

```
iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).ip(4)
  ipForwarding(1)
  ipDefaultTTL(2)
  ipInReceives(3)
  ipInHdrErrors(4)
  ipInAddrErrors(5)
  ipForwDatagrams(6)
  ipInUnknownProtos(7)
  ipInDiscards(8)
  ipInDelivers(9)
  ipOutRequests(10)
  ipOutDiscards(11)
  ipOutNoRoutes(12)
  ipOutDiscards(13)
  ipOutNoRoutes(14)
  ipReasmTimeout(15)
  ipReasmReqds(16)
  ipReasmOKs(17)
  ipReasmFails(18)
  ipReasmFragOKs(19)
  ipFragFails(20)
  ipFragCreates(21)
  ipAddrTable(22).ipAddrEntry(1).ipAdEntAddr(1)
    ipAdEntIfIndex(2)
    ipAdEntNetMask(3)
    ipAdEntBcastAddr(4)
    ipAdEntReasmMaxSize(5)
  ipNetToMediaTable(24).ipNetToMediaEntry(1).ipNetToMediaIfIndex(1)
    ipNetToMediaPhyAddress(2)
    ipNetToMediaNetAddress(3)
    ipNetToMediaType(4)
  ipRoutingDiscards(25)
```

### 3.2.3 ICMPグループ

```
iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).icmp(5)
  icmpInMsgs(1)
  icmpInErrors(2)
  icmpInDestUnreachs(3)
  icmpInTimeExcds(4)
  icmpInParmProbs(5)
  icmpInSrcQuenchs(6)
  icmpInRedirects(7)
  icmpInEchoes(8)
  icmpInEchoReps(9)
  icmpInTimestamps(10)
  icmpInTimestampReps(11)
  icmpInAddrMasks(12)
  icmpInAddrMaskReps(13)
  icmpOutMsgs(14)
  icmpOutErrors(15)
  icmpOutDestUnreachs(16)
  icmpOutTimeExcds(17)
  icmpOutParmProbs(18)
  icmpOutSrcQuenchs(19)
  icmpOutRedirects(20)
  icmpOutEchoes(21)
  icmpOutEchoReps(22)
  icmpOutTimestamps(23)
  icmpOutTimestampReps(24)
  icmpOutAddrMasks(25)
  icmpOutAddrMaskReps(26)
```

### 3.2.4 TCPグループ

tcpConnTable は未サポートとしています。

```
iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).tcp(6)
  tcpRtoAlgorithm(1)
  tcpRtoMin(2)
  tcpRtoMax(3)
  tcpMaxConn(4)
  tcpActiveOpens(5)
  tcpPassiveOpens(6)
  tcpAttemptFails(7)
  tcpEstabResets(8)
  tcpCurrEstab(9)
  tcpInSegs(10)
  tcpOutSegs(11)
  tcpRetransSegs(12)
  tcpInErrs(14)
  tcpOutRsts(15)
```

### 3.2.5 UDPグループ

iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).udp(7)  
  udpInDatagrams(1)  
  udpNoPorts(2)  
  udpInErrors(3)  
  udpOutDatagrams(4)

### 3.2.6 SNMPグループ

iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).snmp(11)  
  snmpInPkts(1)  
  snmpOutPkts(2)  
  snmpInBadVersions(3)  
  snmpInBadCommunityNames(4)  
  snmpInBadCommunityUses(5)  
  snmpInASNParseErrs(6)  
  snmpInTooBigs(8)  
  snmpInNoSuchNames(9)  
  snmpInBadValues(10)  
  snmpInReadOnlys(11)  
  snmpInGenErrs(12)  
  snmpInTotalReqVars(13)  
  snmpInTotalSetVars(14)  
  snmpInGetRequests(15)  
  snmpInNexts(16)  
  snmpInSetRequests(17)  
  snmpInGetResponses(18)  
  snmpInTraps(19)  
  snmpOutTooBigs(20)  
  snmpOutNoSuchNames(21)  
  snmpOutBadValues(22)  
  snmpOutGenErrs(24)  
  snmpOutGetRequests(25)  
  snmpOutGetNexts(26)  
  snmpOutSetRequests(27)  
  snmpOutGetResponses(28)  
  snmpOutTraps(29)  
  snmpEnableAuthenTraps(30)

### 3.3 TRAP情報の登録

TRAPフレームの送信用として、トラップ登録テーブルと、トラップ情報設定エリアの2つがあります。TRAPフレームの送信を行う場合は、それぞれのテーブルにデータを設定する必要があります。

各テーブルの構成を以下に記述します。

#### 3.3.1 トラップ情報登録エリア

トラップ情報登録エリア用の構造体が用意されています。

各種送信するトラップ毎に登録エリアを作成してください。

尚、コミュニティ名NGトラップの情報と、Coldスタートトラップの情報については、サンプルとして、作成してあります。(ただし、送信許可フラグは0クリアとしています。)

新規に追加する場合は、これらの設定エリアを参照しながら、追加を行ってください。

##### (1) 構造体宣言

```
typedef struct {
    U16      *InhFlag;      /* 送信許可フラグへのポインタ */
    U8       TrapKind;     /* Trap 種別 */
    U8       SpTrap;       /* SpecificTrapCode */
    U16      MibNum;       /* 付加 MIB 数 */
    tMIB_INFO MibInfo[7];  /* フレーム付加の MIB 情報 */
} TRAP_TBL;
```

以下に各エリアの説明を記載いたします。

##### ① \*InhFlag

Trap別送信許可フラグのポインタを設定するエリアです。

ポインタのデータにより、そのTrapフレームを送信する／しないを設定することが可能です。フラグ情報は、以下のとおりです。

```
*InhFlag = 0: Trapフレーム送信可
*InhFlag = 1: Trapフレーム送信不可
```

② TrapKind

Trap 種別を設定するエリアです。  
 設定できる内容は、以下のとおりです。  
 本エリアに設定する場合は、シンボル名を使用してください。

表 3.3.2-1 Trap 種別一覧

シンボル	コード	説明
COLD_START	0	送信プロトコルエンティティが再初期化され、エージェントの設定または、エンティティの実装が変更された可能性がある。
WARM_START	1	送信プロトコルが再初期化されたが、エージェントの設定とプロトコルエンティティの実装は変更されていない。
LINK_DOWN	2	通信リンクに障害が発生した。
LINK_UP	3	通信リンクが回復した。
AUTHENTICATION_FAILURE	4	エージェントは、マネージャから不正なメッセージを受信した。
EGP_NEIGHBOR_LOSS	5	EGP ピア어의ノドがダウンした。
ENTERPRISE_SPECIFIC	6	特殊なトラップが発生した。

③ SpTrap

ユーザカスタマイズ用のトラップコードを設定するエリアです。  
 TrapKind=ENTERPRISE\_SPECIFIC の場合は、本エリアはユニークな値が設定可能です。  
 それ以外の場合は、0を設定することをお勧めします。

④ MibNum

Trap フレームに付加する MIB の数を設定するエリアです。  
 本エリアは、1~7 まで設定可能です。  
 尚、付加する MIB なし (0 設定) はサポートしておりませんので、必ず 1 件以上の MIB を設定するようにしてください。

⑤ MibInfo[7]

Trap フレームに付加する MIB の情報テーブルポインタ及び、付加する MIB データのデータ Index を設定するエリアです。  
 本エリアは、最大 7 件のテーブルポインタを登録することが可能です。  
 また、登録する MIB が、7 件未満の場合は、残りのポインタに、NULL を設定してください。

本エリアの構造体宣言は、以下の通りです。

```
typedef struct {
    U32      InstNum;          /* データ Index 番号          */
    MIB_TBL *MibPtr;          /* 登録 MIB テーブルポインタ  */
} tMIB_INFO;                /* Trap フレームに付加する MIB 情報構造体 */
```

i) InstNum

データ Index を設定するエリアです。  
 付加する MIB がスカラー型の場合は、設定値は 0 としてください。  
 テーブル型の場合は、付加するデータの Index 番号 - 1 を設定してください。

e x) テーブル型の 2 番目のデータを付加する場合。

InstNum = 1 とします。

ii) \*MibPtr

付加する MIB の情報が設定されている MIB 情報テーブルポインタを設定するエリアです。

### 3.3.2 トラップ登録テーブル

送信するトラップの種類別に、トラップ情報登録テーブルのポインタを設定するテーブルです。  
このテーブルは、送信するトラップの種類+ストップ分のインデックスが必要となります。  
また、このテーブルに、NULLポインタを設定した場合は、そのトラップは送信されません。

このテーブルの `Index` 値が、運用中に使用するトラップ種別コードとなります。  
尚、テーブルの先頭 `Index` は、コミュニティ名NGトラップ専用、2番目の `Index` は、Cold  
スタートトラップ先頭となっております。

#### (1) エリア宣言

```
const TRAP_TBL*      SNM_TrapPtrTbl [ENTRY_TRAP_NUM + 1] = {  
    NULL,                /* Community 名 NG Trap      */  
    NULL,                /* BOOT Trap                  */  
    (TRAP_TBL *) TABLE_STOPPER  
};
```

## 4. SNMP パラメータ設定

SNMPパラメータとは、以下の情報を示します。

- (1) コミュニティ名
- (2) フレーム受信用UDPポートNo.
- (3) トラップ送信用UDPポートNo. : (登録トラップ数分)
- (4) トラップ送信先SNMPマネージャ IPアドレス
- (5) トラップ送信する/しないフラグ

これらのパラメータは、ユーザシステムにより、保存場所を各種不揮発性媒体にすることが可能です。そのため、本SNMPでは、各パラメータ別の初期化関数の定義と、動作時に参照するパラメータエリアの定義についてのみ行います。

パラメータの保存方法および、初期化関数の処理につきましては、ユーザカスタマイズといたします。

### 4.1 コミュニティ名の定義

登録IPアドレス毎にコミュニティ名を設定する必要があります。

- (1) エリア宣言

```
typedef struct {
    U8      Len;                /* コミュニティ名 レングス数 */
    U8      Name[255];         /* コミュニティ名 文字列 */
}COMUNITY_INFO;
```

```
COMUNITY_INFO  SNMP_Comunity[MAX_IPADRS];
```

※ "MAX\_IPADRS"は、登録IPアドレス数です。

詳細につきましては、「TCP/IPプロトコルスタック ySOCK2 導入説明書」を参照願います。

- (2) 初期化関数

```
void  SNMP_InitCommunityName(void)
```

- (3) 注意事項

- ①コミュニティ名レングス数=0に設定した場合、SNMPでは、“public” (デフォルトコミュニティ名)を使用します。

#### 4.2 フレーム受信ポートの設定

登録 IP アドレス毎に SNMP フレームの受信ポート No. を設定する必要があります。

(1) 格納エリア

```
U16          SNMP_RecvPort[MAX_IPADRS];
```

(2) 初期化関数

```
void         SNMP_InitRecvPort(void)
```

(3) 注意事項

- ①ポート No.=0とした場合、SNMPでは、161（デフォルトポート No.）を使用します。
- ②本ポート No.を161以外に設定する場合は、SNMP及び、その他のプロトコルが正常に通信できなくなる可能性がありますので、他のプロトコルのポート No.と重複しないようにしてください。

#### 4.3 トラップ送信先UDPポートの設定

送信できるトラップの種類分、UDPポートの設定を行うことができます。

(1) 格納エリア

```
U16          SNMP_TrapPortTbl[MAX_IPADRS][ENTRY_TRAP_NUM];
```

(2) 初期化関数

```
void         SNMP_InitTrapPort(void)
```

(3) 注意事項

- ①ポート No.=0とした場合、SNMPでは、全てのトラップを送信する際に、162（デフォルトポート No.）を使用します。
- ②本ポート No.を162以外に設定する場合は、SNMP及び、その他のプロトコルが正常に通信できなくなる可能性がありますので、他のプロトコルのポート No.と重複しないようにしてください。

#### 4.4 トラップ送信先IPアドレスの設定

トラップフレームを送信する際の、送信先SNMPマネージャのIPアドレスを設定します。  
登録IPアドレス毎に、1件のIPアドレスを設定することが可能です。

##### (1) 格納エリア

```
typedef struct {  
    U16          flag;          /* SNMP マネージャ IP アドレス登録 0=未登録, 1=登録 */  
    U16          IpLength;      /* IP アドレス長      */  
    U8           IpAdrs[16];    /* IP アドレス値      */  
} MANE_INFO;                  /* マネージャ IP アドレス用 構造体 */  
  
MANE_INFO      SNMP_TrapSendManagerIPAdrs[MAX_IPADRS];
```

##### (2) 初期化関数

```
void          SNMP_InitTrapSendManagerIPAdrs(void)
```

##### (3) 注意事項

- ①本項目を設定しなかった場合は、トラップ送信は行われません。

#### 4.5 トラップ送信Enableフラグ

トラップフレームの送信する/しないを設定します。  
登録IPアドレス毎に、フラグ設定が可能です。

##### (1) エリア宣言

```
U32  SNMP_TrapEnableFlag[MAX_IPADRS];
```

##### (2) 初期化関数

```
void          SNMP_InitTrapEnableFlag(void)
```

##### (3) 注意事項

なし。