システムマクロトレース API ライブラリ
リファレンスマニュアル
Linux／Android 編

株式会社ＤＴＳインサイト

システムマクロトレース API ライブラリ リファレンスマニュアル

発行：株式会社ＤＴＳインサイト

改訂履歴

第 1 版	2012 年 10 月	新規発行
第 2 版	2014 年 11 月	誤記修正

目次

1 はじめに

システムマクロトレース API ライブラリ（以下、API ライブラリ）は、システムマクロトレース情報にデバッグ情報を出力するためのライブラリです。

1.1 API ライブラリリファレンスマニュアルとは

API ライブラリをユーザーシステム環境に組み込み、任意の関数内に本マニュアルに記載されたデバッグ関数を挿入することで、トレース情報を収集、解析することができます。

本マニュアルでは、

- API ライブラリ関数リファレンス(C/C++)
- API ライブラリ関数リファレンス(Java)
- 効果的なご利用方法
- 注意事項

以上について記載します。

本書内では、Java 言語の「メソッド」を「関数」として記述しています。

システムマクロトレース API ライブラリの組み込み方法は、
『システムマクロトレース ポーティングガイド』をご覧ください。

1.2 ユーザーランド(C/C++)API ライブラリ関数のヘッダファイル

デバッグ情報を出力したい C/C++言語ソースにライブラリ関数の呼び出しを追加してください。
また、使用するライブラリ関数に対応したヘッダファイルをインクルードしてください。

ライブラリ関数仕様については、「2 API ライブラリ関数リファレンス(C/C++)」をご覧ください。

下の API ライブラリヘッダファイルをソースファイル中にインクルードしてください。
（本ファイルは各種 API ライブラリ用ヘッダファイルを統合したものですので、このファイル以外のインクルード記述は不要です）

SMTAPI.h	システムマクロトレース API ライブラリ用ヘッダファイル

ソースファイル記述例

```
#include "SMTAPI.h"
```

1.3 カーネルランド API ライブラリ関数のヘッダファイル

デバッグ情報を出力したいソースにライブラリ関数の呼び出しを追加してください。特にコンテキストスイッチ情報を出力するライブラリ関数を追加しないとプロセスや割込み処理の遷移情報を記録、閲覧できません。ご注意ください。

ライブラリ関数仕様については、「2 API ライブラリ関数リファレンス(C/C++)」をご覧ください。

また、使用するライブラリ関数に対応したヘッダファイルをインクルードしてください。
コンパイルオプションでインクルードパスを特に指定しない場合、ヘッダファイルのインクルード宣言は下記のようにパス表記が必要です。

SMTAPI.h	システムマクロトレース API ライブラリ用ヘッダファイル

ソースファイル記述例

```
#include <smt/SMTAPI.h>
```

1.4 Android Java アプリケーションからの API ライブラリ関数呼び出し

デバッグ情報を出力したい Java 言語ソースにライブラリ関数の呼び出しを追加してください。

ライブラリ関数仕様については、「3 API ライブラリ関数リファレンス(Java)」をご覧ください。

2 API ライブラリ関数リファレンス(C/C++)

API ライブラリで提供する関数は、以下のグループに分類されます。

・デバッグプリント文制御関数
・デバッグプリント関数
・ポートアウト関数
・コンテキストスイッチ関数
・システムコール関数

関数一覧

グループ	関数名	ヘッダファイル	機能説明	ユーザーランド	カーネルランド
初期化	_SMT_Open	SMTAPI.h	API ライブラリのオープンおよびデバッグ情報出力の有効化	○	
	_SMT_Close	SMTAPI.h	API ライブラリのクローズ	○	
	_SMT_GetDebugLevel	SMTAPI.h	デバッグプリント出力レベルマスク値の参照	○	○
デバッグプリント文制御	_SMT_SetDebugLevel	SMTAPI.h	デバッグプリント出力レベルマスク値の設定	○	○
	_SMT_Puts	SMTAPI.h	puts 相当。 文字列の出力	○	○
デバッグプリント	_SMT_Printf	SMTAPI.h	printf 相当。 書式指定付き文字列の出力	○	○
	_SMT_UsrMsgTag0 _SMT_UsrMsgTag1 _SMT_UsrMsgTag2 _SMT_UsrMsgTag3 _SMT_UsrMsgTag4	SMTAPI.h	ユーザー定義メッセージタグ出力	○	○
ポートアウト	_SMT_PortOut	SMTAPI.h	ポート情報出力	○	○

関数一覧

グループ	関数名	ヘッダファイル	機能説明	ユーザーランド	カーネルランド
コンテキストスイッチ	_SMT_OsSwitch_Process	SMTAPI.h	プロセスのスイッチ情報出力		○
	_SMT_OsSwitch_ThreadProcess	SMTAPI.h	スレッドおよびプロセスのスイッチ情報出力		○
	_SMT_OsSwitch_Process_Name	SMTAPI.h	プロセスのスイッチ情報､名称の出力		○
	_SMT_OsSwitch_ThreadProcess_Name	SMTAPI.h	スレッドおよびプロセスのスイッチ情報、名称の出力		○
	_SMT_OsSwitch_Irq_in	SMTAPI.h	割込み処理開始の情報出力		○
	_SMT_OsSwitch_Irq_out	SMTAPI.h	割込み処理終了の情報出力		○
	_SMT_OsSwitch_Idle	SMTAPI.h	アイドルタスクへのスイッチ情報出力		○
システムコール	_SMT_OsCall0	SMTAPI.h	システムコール処理開始､終了の情報出力（引数なし）		○
	_SMT_OsCall1	SMTAPI.h	同上（引数 1 個）		○
	_SMT_OsCall2	SMTAPI.h	同上（引数 2 個）		○
	_SMT_OsCall3	SMTAPI.h	同上（引数 3 個）		○
	_SMT_OsCall4	SMTAPI.h	同上（引数 4 個）		○
	_SMT_OsCall5	SMTAPI.h	同上（引数 5 個）		○
	_SMT_OsCall6	SMTAPI.h	同上（引数 6 個）		○
	_SMT_OsCall7	SMTAPI.h	同上（引数 7 個）		○
	_SMT_OsCall8	SMTAPI.h	同上（引数 8 個）		○
	_SMT_OsCall9	SMTAPI.h	同上（引数 9 個）		○
	_SMT_OsCall10	SMTAPI.h	同上（引数 10 個）		○
	_SMT_OsCall11	SMTAPI.h	同上（引数 11 個）		○
	_SMT_OsCall12	SMTAPI.h	同上（引数 12 個）		○
	_SMT_OsCall13	SMTAPI.h	同上（引数 13 個）		○
	_SMT_OsCall14	SMTAPI.h	同上（引数 14 個）		○
	_SMT_OsCall15	SMTAPI.h	同上（引数 15 個）		○
	_SMT_OsCall16	SMTAPI.h	同上（引数 16 個）		○

2.1 初期化関数

初期化関数は、API ライブラリの初期化を行うための関数です。

2.1.1 _SMT_Open 関数

■機能

API ライブラリをオープンします。

■呼び出し手順

```
#include "SMTAPI.h"

_SMT_Open();
```

■引数

No.	名前	型	意味
-	-	-	-

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

_SMT_Open 関数は初期化済み API ライブラリをオープンし、デバッグ情報出力を有効にします。
ユーザーランドの関数については、オープン前に使用すると内部で自動的に本関数を呼び出すので、本関数の呼び出しは省略できます。
カーネルランドでは本関数の呼び出しは不要です。

2.1.2 _SMT_Close 関数

■機能

API ライブラリをクローズします。

■呼び出し手順

```
#include "SMTAPI.h"

_SMT_Close();
```

■引数

No.	名前	型	意味
-	-	-	-

■戻り値

型　：　int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

_SMT_Close 関数はオープン済み API ライブラリをクローズします。
通常は使用しないでください。
API ライブラリを既にオープンし、かつ exec システムコールを呼び出す場合のみ、exec 直前に必ずこの関数を呼び出してください

2.2 デバッグプリント文制御関数

デバッグプリント文制御関数は、デバッグプリント文のマスク設定値の設定・取得を行います。

2.2.1 _SMT_GetDebugLevel 関数

■機能

現在のデバッグプリント出力レベルマスク設定値を返します。

■呼び出し手順

```
#include "SMTAPI.h"

int level;
level=_SMT_GetDebugLevel();
```

■引数

No.	名前	型	意味
-	-	-	-

■戻り値

型　:　int
現在のデバッグプリント出力レベルマスク設定値

_SMT_GetDebugLevel 関数は、現在のデバッグプリント出力レベルマスク設定値を返します。

2.2.2 _SMT_SetDebugLevel 関数

■機能

デバッグプリント出力レベルマスク値を設定します。

■呼び出し手順

```
#include "SMTAPI.h"

int level = 0;

_SMT_SetDebugLevel(level);
```

■引数

No.	名前	型	意味
1	level	int	デバッグプリント出力レベルマスク設定値

■戻り値

型　：　void

_SMT_SetDebugLevel 関数は、デバッグプリント出力レベルマスク設定値を設定します。

2.3 デバッグプリント関数

デバッグプリント関数は、デバッグ用の文字列を出力するための関数です。

2.3.1 _SMT_Puts 関数

■機能

トレース上に文字列を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

const char *s = "Debug Message";
int level = 0;

_SMT_Puts(level, s);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグプリント出力レベル値
2	s	const char 型を指すポインタ	出力する文字列へのポインタ

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

_SMT_Puts 関数はシステムマクロトレース上に文字列を出力します。複雑な書式変換を行わない分、コードサイズ、実行速度の面で、_SMT_Printf 関数より有利です。

出力可能な文字列長は、終端文字を含めて最大で 124byte です。
これより長い文字列を指定した場合は切り捨てます。

API ライブラリがオープンされていない場合は、関数内で自動的にオープンします。

2.3.2 _SMT_Printf 関数

■機能

データを書式に従って変換し、システムマクロトレース上に文字列を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

const char *format;
int level = 0;

_SMT_Printf(level, format, [arg]…);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグ文出力レベル値
2	format	const char 型を指すポインタ	書式を示す文字列へのポインタ
3	arg	書式に従った型	書式に従って出力されるデータ

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

_SMT_Printf 関数は、format が示す書式を文字列に従って、引数 arg を変換し、トレース情報として出力します。
_SMT_Printf 関数では､書式変換に標準ライブラリ関数の vsnprintf 関数を使用しています。
書式の仕様は標準ライブラリ関数の仕様をご覧ください。

出力可能な文字列長は、書式変換後の長さで、終端文字を含めて最大で 124byte です。
これより長い文字列を指定した場合は切り捨てます。

API ライブラリがオープンされていない場合は、関数内で自動的にオープンします。

2.3.3 _SMT_UsrMsgTag0 関数

■機能

ユーザー定義メッセージタグ番号をシステムマクロトレース情報として出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int level = 0;
unsigned long TagNum = 0x123;

_SMT_UsrMsgTag0(level, TagNum);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグ文出力レベル値
2	TagNum	unsigned long 型	メッセージタグ番号 0~1023

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

_SMT_UserTagMsg0 関数は、ユーザー任意のタグ番号を出力します。
タグ番号に対応したユーザー定義メッセージを表示できます。

API ライブラリがオープンされていない場合は、関数内で自動的にオープンします。

2.3.4 _SMT_UsrMsgTag1 関数

■機能

ユーザー定義メッセージタグ番号と 1 つのデータをシステムマクロトレース情報として出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int level = 0;
unsigned long TagNum = 0x123;

_SMT_UsrMsgTag1(level,TagNum,arg1);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグ文出力レベル値
2	TagNum	unsigned long 型	メッセージタグ番号 0~1023
3	arg1	unsigned long 型	書式に従って出力されるデータ

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

_SMT_UserTagMsg1 関数は、ユーザー任意のタグ番号とデータを出力します。
タグ番号に対応したユーザー定義メッセージを表示できます。

API ライブラリがオープンされていない場合は、関数内で自動的にオープンします。

2.3.5 _SMT_UsrMsgTag2 関数

■機能

ユーザー定義メッセージタグ番号と 2 つのデータをシステムマクロトレース情報として出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int level = 0;
unsigned long TagNum = 0x123;

_SMT_UsrMsgTag2(level, TagNum,arg1,arg2);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグ文出力レベル値
2	TagNum	unsigned long 型	メッセージタグ番号 0~1023
3	arg1	unsigned long 型	書式に従って出力されるデータ
4	arg2	unsigned long 型	書式に従って出力されるデータ

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

_SMT_UserTagMsg2 関数は、ユーザー任意のタグ番号とデータを出力します。
タグ番号に対応したユーザー定義メッセージを表示できます。

API ライブラリがオープンされていない場合は、関数内で自動的にオープンします。

2.3.6 _SMT_UsrMsgTag3 関数

■機能

ユーザー定義メッセージタグ番号と 3 つのデータををシステムマクロトレース情報として出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int level = 0;
unsigned long TagNum = 0x123;

_SMT_UsrMsgTag3(level,TagNum,arg1,arg2,arg3);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグ文出力レベル値
2	TagNum	unsigned long 型	メッセージタグ番号 0~1023
3	arg1	unsigned long 型	書式に従って出力されるデータ
4	arg2	unsigned long 型	書式に従って出力されるデータ
5	arg3	unsigned long 型	書式に従って出力されるデータ

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

_SMT_UserTagMsg3 関数は、ユーザー任意のタグ番号とデータを出力します。
タグ番号に対応したユーザー定義メッセージを表示できます。

API ライブラリがオープンされていない場合は、関数内で自動的にオープンします。

2.3.7 _SMT_UsrMsgTag4 関数

■機能

ユーザー定義メッセージタグ番号と 4 つのデータをシステムマクロトレース情報として出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int level = 0;
unsigned long TagNum = 0x123;

_SMT_UsrMsgTag4(level,TagNum,arg1,arg2,arg3,arg4);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグ文出力レベル値
2	TagNum	unsigned long 型	メッセージタグ番号 0~1023
3	arg1	unsigned long 型	書式に従って出力されるデータ
4	arg2	unsigned long 型	書式に従って出力されるデータ
5	arg3	unsigned long 型	書式に従って出力されるデータ
6	arg4	unsigned long 型	書式に従って出力されるデータ

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

_SMT_UserTagMsg4 関数は、ユーザー任意のタグ番号とデータを出力します。
タグ番号に対応したユーザー定義メッセージを表示できます。

API ライブラリがオープンされていない場合は、関数内で自動的にオープンします。

2.3.8 デバッグプリント出力レベル

デバッグプリント関数群は、引数に出力レベル値を持ち、０～１５のレベルを設定することができます。
これは、_SMT_SetDebugLevel 関数で設定されるデバッグプリント出力レベルマスクパターンと比較されデバッグプリント出力のフィルタリングを実現します。

デバッグプリント出力レベルマスクパターン値は、ビットパターンとして定義され、各ビット位置がレベル値に対応します。
ビット値が０のレベルは出力され、１のレベルは出力されません。

例えば、以下のようにデバッグプリント出力レベルマスク設定値が 0xFFFC の場合、レベル０とレベル１として記述されたデバッグプリントのみ出力されます。

ソースコード記述

```
…
_SMT_SetDebugLevel( 0xFFFC);

_SMT_Puts( 0, "Level 0 Message");
_SMT_Puts( 1, "Level 1 Message");
_SMT_Puts( 2, "Level 2 Message");
_SMT_Puts( 3, "Level 3 Message");
…
```

出力結果

```
Level 0 Message
Level 1 Message
```

デバッグプリント出力レベルマスク設定値は、初期状態では 0x0000 となっており、すべてのレベルが出力有効となっています。この初期値は、SMTUserDef.h に定義されており変更が可能です(グローバル変数初期化のタイミングでこの定義値で初期化されます)。

```
#define     SMT_LV_MSK    0x0000
```

2.4 ポートアウト関数

ポートアウト関数は、ポートアクセス情報をシステムマクロトレース情報として出力します。

2.4.1 _SMT_PortOut 関数

■機能

ポートアクセス情報を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

unsigned long addr;
unsigned char data8;
unsigned short data16;
unsigned long data32;

_SMT_PortOut( addr, data8, _SMT_PSZ8, _SMT_PW);  // 8bit write
_SMT_PortOut( addr, data8, _SMT_PSZ8, _SMT_PR);  // 8bit read
_SMT_PortOut( addr, data16, _SMT_PSZ16, _SMT_PW);// 16bit write
_SMT_PortOut( addr, data16, _SMT_PSZ16, _SMT_PR);// 16bit read
_SMT_PortOut( addr, data32, _SMT_PSZ32, _SMT_PW);// 32bit write
_SMT_PortOut( addr, data32, _SMT_PSZ32, _SMT_PR);// 32bit read
```

■引数

No.	名前	型	意味
1	addr	unsigned long	アドレス値
2	data	unsigned long	データ値
3	size	_SMT_PSZ	ポートサイズ 定義値 _SMT_PSZ8 : 8bit _SMT_PSZ16 : 16bit _SMT_PSZ32 : 32bit
4	rw	_SMT_PRW	アクセスタイプ 定義値 _SMT_PW: ライト _SMT_PR: リード

■戻り値

型 : int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

_SMT_PortOut 関数は、ポートアクセス情報を出力します。

ポートサイズ、アクセスタイプの指定には、引数表に記載の定義値を使用してください。

API ライブラリがオープンされていない場合は、関数内で自動的にオープンします。

2.5 コンテキストスイッチ関数

プロセスやスレッドの ID または名称、割込みの情報を出力するための関数です。

2.5.1 _SMT_OsSwitch_Process 関数

■機能

プロセス ID を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

unsigned long processid;

_SMT_OsSwitch_Process( processid );
```

■引数

No.	名前	型	意味
1	processid	unsigned long	プロセス ID 値

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.5.2 _SMT_OsSwitch_ThreadProcess 関数

■機能

スレッド ID およびプロセス ID を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

unsigned long threadid;
unsigned long processid;

_SMT_OsSwitch_ThreadProcess(hreadid, processid);
```

■引数

No.	名前	型	意味
1	threadid	unsigned long	スレッド ID 値
2	processid	unsigned long	プロセス ID 値

■戻り値

型 ： int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.5.3 _SMT_OsSwitch_Process_Name 関数

■機能

プロセス ID とプロセス名称を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

unsigned long processid;
const char *pname;

_SMT_OsSwitch_Process_Name(processid, pname);
```

■引数

No.	名前	型	意味
1	processid	unsigned long	プロセス ID 値
2	pname	const char*	プロセス名称文字列（終端文字を含め最大 56 文字）

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.5.4 _SMT_OsSwitch_ThreadProcess_Name 関数

■機能

スレッド ID、プロセス ID、スレッド名称およびプロセス名称を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

unsigned long threadid;
unsigned long processid;
const char *tname;
const char *pname;

_SMT_OsSwitch_ThreadProcess_Name(threadid, processid, tname, pname);
```

■引数

No.	名前	型	意味
1	threadid	unsigned long	スレッド ID 値
2	processid	unsigned long	プロセス ID 値
3	tname	const char*	スレッド名称文字列（終端文字を含め最大 56 文字）
4	pname	const char*	プロセス名称文字列（終端文字を含め最大 56 文字）

■戻り値

型 ： int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.5.5 _SMT_OsSwitch_Irq_in 関数

■機能

割込み処理開始と割込み番号を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

unsigned long irqid;

_SMT_OsSwitch_Irq_in(irqid);
```

■引数

No.	名前	型	意味
1	irqid	unsigned long	割込み番号

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.5.6 _SMT_OsSwitch_Irq_out 関数

■機能

割込み処理終了と割込み番号を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

unsigned long irqid;

_SMT_ OsSwitch_Irq_out (irqid);
```

■引数

No.	名前	型	意味
1	irqid	unsigned long	割込み番号

■戻り値

型 ： int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.5.7 _SMT_OsSwitch_Idle 関数

■機能

アイドルプロセスへのスイッチを出力します。

■呼び出し手順

```
#include "SMTAPI.h"

_SMT_ OsSwitch_Idle ();
```

■引数

No.	名前	型	意味

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6 システムコール関数

システムコールの番号、開始終了属性、および引数を出力する関数です。引数は 32bit サイズであり、最大 16 個まで出力可能です。

2.6.1 _SMT_OsCall0 関数

■機能

システムコール番号、呼び出し属性を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;

_SMT_OsCall0( osc, _SMT_OS_ATTR_CALL );
_SMT_OsCall0( osc, _SMT_OS_ATTR_RET );
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了

■戻り値

型 ： int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.2 _SMT_OsCall1 関数

■機能

システムコール番号、呼び出し属性、引数 1 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1;

_SMT_OsCall1( osc, _SMT_OS_ATTR_CALL,arg1);
_SMT_OsCall1( osc, _SMT_OS_ATTR_RET,arg1);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1	unsigned long	引数 1

■戻り値

型 ： int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.3 _SMT_OsCall2 関数

■機能

システムコール番号、呼び出し属性、引数 2 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2;

_SMT_OsCall2( osc, _SMT_OS_ATTR_CALL,arg1,arg2);
_SMT_OsCall2( osc, _SMT_OS_ATTR_RET,arg1,arg2);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg2	unsigned long	引数 1～引数 2

■戻り値

型 ： int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.4 _SMT_OsCall3 関数

■機能

システムコール番号、呼び出し属性、引数 3 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3;

_SMT_OsCall3( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3);
_SMT_OsCall3( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg3	unsigned long	引数 1～引数 3

■戻り値

型 ： int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.5 _SMT_OsCall4 関数

■機能

システムコール番号、呼び出し属性、引数 4 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4;

_SMT_OsCall4( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4);
_SMT_OsCall4( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg4	unsigned long	引数 1～引数 4

■戻り値

型 : int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.6 _SMT_OsCall5 関数

■機能

システムコール番号、呼び出し属性、引数 5 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5;

_SMT_OsCall5( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5);
_SMT_OsCall5( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg5	unsigned long	引数 1～引数 5

■戻り値

型 ： int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.7 _SMT_OsCall6 関数

■機能

システムコール番号、呼び出し属性、引数 6 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5,arg6;

_SMT_OsCall6( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5,arg6);
_SMT_OsCall6( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5,arg6);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg6	unsigned long	引数 1～引数 6

■戻り値

型 : int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.8 _SMT_OsCall7 関数

■機能

システムコール番号、呼び出し属性、引数 7 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5,arg6,arg7;

_SMT_OsCall7( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5,arg6
,arg7);
_SMT_OsCall7( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5,arg6
,arg7);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg7	unsigned long	引数 1～引数 7

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.9 _SMT_OsCall8 関数

■機能

システムコール番号、呼び出し属性、引数 8 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8;

_SMT_OsCall8( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8);
_SMT_OsCall8( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg8	unsigned long	引数 1～引数 8

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.10 _SMT_OsCall9 関数

■機能

システムコール番号、呼び出し属性、引数 9 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9;

_SMT_OsCall9( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9);
_SMT_OsCall9( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg9	unsigned long	引数 1～引数 9

■戻り値

型 : int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.11 _SMT_OsCall10 関数

■機能

システムコール番号、呼び出し属性、引数 10 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10;

_SMT_OsCall10( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10);
_SMT_OsCall10( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg10	unsigned long	引数 1～引数 10

■戻り値

型 : int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.12 _SMT_OsCall11 関数

■機能

システムコール番号、呼び出し属性、引数 11 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10
,arg11;

_SMT_OsCall11( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11);
_SMT_OsCall11( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg11	unsigned long	引数 1～引数 11

■戻り値

型 : int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.13 _SMT_OsCall12 関数

■機能

システムコール番号、呼び出し属性、引数 12 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10
,arg11,arg12;

_SMT_OsCall12( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11,arg12);
_SMT_OsCall12( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11,arg12);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg12	unsigned long	引数 1～引数 12

■戻り値

型 : int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.14 _SMT_OsCall13 関数

■機能

システムコール番号、呼び出し属性、引数 13 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10
,arg11,arg12,arg13;

_SMT_OsCall12( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11,arg12,arg13);
_SMT_OsCall12( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11,arg12,arg13);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg13	unsigned long	引数 1～引数 13

■戻り値

型 : int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.15 _SMT_ OsCall14 関数

■機能

システムコール番号、呼び出し属性、引数 14 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10
,arg11,arg12,arg13,arg14;

_SMT_OsCall12( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11,arg12,arg13,arg14);
_SMT_OsCall12( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11,arg12,arg13,arg14);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg14	unsigned long	引数 1～引数 14

■戻り値

型 ： int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.16 _SMT_ OsCall15 関数

■機能

システムコール番号、呼び出し属性、引数 15 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10
,arg11,arg12,arg13,arg14,arg15;

_SMT_OsCall12( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11,arg12,arg13,arg14,arg15);
_SMT_OsCall12( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11,arg12,arg13,arg14,arg15);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg15	unsigned long	引数 1～引数 15

■戻り値

型 ： int

実行結果を返します。

_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

2.6.17 _SMT_ OsCall16 関数

■機能

システムコール番号、呼び出し属性、引数 16 個を出力します。

■呼び出し手順

```
#include "SMTAPI.h"

int osc;
unsigned long arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10
,arg11,arg12,arg13,arg14,arg15,arg15;

_SMT_OsCall12( osc, _SMT_OS_ATTR_CALL,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11,arg12,arg13,arg14,arg15,arg16);
_SMT_OsCall12( osc, _SMT_OS_ATTR_RET,arg1,arg2,arg3,arg4,arg5,arg6
,arg7,arg8,arg9,arg10,arg11,arg12,arg13,arg14,arg15,arg16);
```

■引数

No.	名前	型	意味
1	osc	int	システムコール番号
2	attr	_SMT_OS_CALL	属性 定義値 _SMT_OS_ATTR_CALL : システムコール開始 _SMT_OS_ATTR_RET : システムコール終了
3	arg1 ～ arg16	unsigned long	引数 1～引数 16

■戻り値

型 : int
実行結果を返します。
_SMT_OK（正常終了）又は_SMT_NG（異常終了）を返します。

3 API ライブラリ関数リファレンス(Java)

Java ソース用に提供するライブラリ関数は、以下のグループに分類されます。
これらの関数は、プラットフォーム上のネイティブコード層に組み込まれた C//C++用 SMT API ライブラリ関数の呼び出しインターフェイスを提供します。

Java 関数と C/C++関数の対応

グループ	Java 関数名	C/C++関数名
デバッグプリント文制御	GetDebugLevel	_SMT_GetDebugLevel
	SetDebugLevel	_SMT_SetDebugLevel
デバッグプリント	Puts	_SMT_Puts
	—[1]	_SMT_Printf
	UsrMsgTag0 UsrMsgTag1 UsrMsgTag2 UsrMsgTag3 UsrMsgTag4	_SMT_UsrMsgTag0 _SMT_UsrMsgTag1 _SMT_UsrMsgTag2 _SMT_UsrMsgTag3 _SMT_UsrMsgTag4
ポートアウト	PortOut	_SMT_PortOut

Java ソース用 API は、パッケージ名 jp.co.dts_insight.emb、クラス名 SMT に実装されています。

[1] Java 言語ソースからの Printf には対応していません。書式変換済みの文字列を生成し、Puts を使用して出力してください。

3.1 デバッグプリント文制御関数

デバッグプリント文制御関数は、デバッグプリント文のマスク設定値の設定・取得を行います。

3.1.1 GetDebugLevel 関数

■機能

現在のデバッグプリント出力レベルマスク設定値を返します。

■呼び出し手順

```
import jp.co.dts_insight.emb *;

int level;

level = SMT.GetDebugLevel();
```

■引数

No.	名前	型	意味
-	-	-	-

■戻り値

型 ： int
現在のデバッグプリント出力レベルマスク設定値

3.1.2 SetDebugLevel 関数

■機能

デバッグプリント出力レベルマスク値を設定します。

■呼び出し手順

```
import jp.co.dts_insight.emb.*;

int level = 0;

SMT.SetDebugLevel(level);
```

■引数

No.	名前	型	意味
1	level	int	デバッグプリント出力レベルマスク設定値

■戻り値

型　：　void

3.2 デバッグプリント関数

デバッグプリント関数は、デバッグ用の文字列を出力するための関数です。

3.2.1 Puts 関数

■機能

トレース上に文字列を出力します。

■呼び出し手順

```
import jp.co.dts_insight.emb.*;

String s = "Debug Message";
int level = 0;

SMT.Puts(level, s);  // エンコードを指定しない場合
SMT.Puts(level, s, “UTF-8”);       // エンコードを指定する場合
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグプリント出力レベル値
2	s	String 型	出力する文字列
3	enc	String 型	文字エンコーディングを指定します。 指定がない場合は“UTF-8”として 扱います。 文字エンコーディング処理には、 String#getBytes(String enc) を使用しています。

■戻り値

型　：　int
実行結果を返します。
0（正常終了）又は 1（異常終了）を返します。

出力可能な文字列長は、終端文字を含めて最大で 124byte です。
これより長い文字列を指定した場合は切り捨てます。

3.2.2 UsrMsgTag0 関数

■機能

ユーザー定義メッセージタグ番号をシステムマクロトレース情報として出力します。

■呼び出し手順

```
import jp.co.dts_insight.emb.*;

int level = 0;
int TagNum = 0x123;

SMT.UsrMsgTag0(level, TagNum);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグ文出力レベル値
2	TagNum	int 型	メッセージタグ番号 0~1023

■戻り値

型 ： int
実行結果を返します。
0（正常終了）又は 1（異常終了）を返します。

UserTagMsg0 関数は、ユーザー任意のタグ番号を出力します。
タグ番号に対応したユーザー定義メッセージを表示できます。

3.2.3 UsrMsgTag1 関数

■機能

ユーザー定義メッセージタグ番号と 1 つのデータをシステムマクロトレース情報として出力します。

■呼び出し手順

```
import jp.co.dts_insight.emb.*;

int level = 0;
int TagNum = 0x123;

SMT.UsrMsgTag1(level,TagNum,arg1);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグ文出力レベル値
2	TagNum	int 型	メッセージタグ番号 0~1023
3	arg1	int 型	書式に従って出力されるデータ

■戻り値

型 ： int
実行結果を返します。
0（正常終了）又は 1（異常終了）を返します。

UserTagMsg1 関数は、ユーザー任意のタグ番号とデータを出力します。
タグ番号に対応したユーザー定義メッセージを表示できます。

3.2.4 UsrMsgTag2 関数

■機能

ユーザー定義メッセージタグ番号と 2 つのデータをシステムマクロトレース情報として出力します。

■呼び出し手順

```
import jp.co.dts_insight.emb.*;

int level = 0;
int TagNum = 0x123;

SMT.UsrMsgTag2(level, TagNum,arg1,arg2);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグ文出力レベル値
2	TagNum	int 型	メッセージタグ番号 0~1023
3	arg1	int 型	書式に従って出力されるデータ
4	arg2	int 型	書式に従って出力されるデータ

■戻り値

型 ： int
実行結果を返します。
0（正常終了）又は 1（異常終了）を返します。

UserTagMsg2 関数は、ユーザー任意のタグ番号とデータを出力します。
タグ番号に対応したユーザー定義メッセージを表示できます。

3.2.5 UsrMsgTag3 関数

■機能

ユーザー定義メッセージタグ番号と 3 つのデータををシステムマクロトレース情報として出力します。

■呼び出し手順

```
import jp.co.dts_insight.emb.*;

int level = 0;
int TagNum = 0x123;

SMT.UsrMsgTag3(level,TagNum,arg1,arg2,arg3);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグ文出力レベル値
2	TagNum	int 型	メッセージタグ番号 0~1023
3	arg1	int 型	書式に従って出力されるデータ
4	arg2	int 型	書式に従って出力されるデータ
5	arg3	int 型	書式に従って出力されるデータ

■戻り値

型　：　int
実行結果を返します。
0（正常終了）又は 1（異常終了）を返します。

UserTagMsg3 関数は、ユーザー任意のタグ番号とデータを出力します。
タグ番号に対応したユーザー定義メッセージを表示できます。

3.2.6 UsrMsgTag4 関数

■機能

ユーザー定義メッセージタグ番号と 4 つのデータをシステムマクロトレース情報として出力します。

■呼び出し手順

```
import jp.co.dts_insight.emb.*;

int level = 0;
int TagNum = 0x123;

SMT.UsrMsgTag4(level,TagNum,arg1,arg2,arg3,arg4);
```

■引数

No.	名前	型	意味
1	level	int 型	デバッグ文出力レベル値
2	TagNum	int 型	メッセージタグ番号 0~1023
3	arg1	int 型	書式に従って出力されるデータ
4	arg2	int 型	書式に従って出力されるデータ
5	arg3	int 型	書式に従って出力されるデータ
6	arg4	int 型	書式に従って出力されるデータ

■戻り値

型 ： int
実行結果を返します。
0（正常終了）又は 1（異常終了）を返します。

UserTagMsg4 関数は、ユーザー任意のタグ番号とデータを出力します。
タグ番号に対応したユーザー定義メッセージを表示できます。

3.3 ポートアウト関数

ポートアウト関数は、ポートアクセス情報をシステムマクロトレース情報として出力します。

3.3.1 PortOut 関数

■機能

ポートアクセス情報を出力します。

■呼び出し手順

```
import jp.co.dts_insight.emb.*;

int addr;
int data;

SMT.PortOut( addr, data, SMT.PSZ.PSZ8, SMT.PRW.PW);     // 8bit write
SMT.PortOut( addr, data, SMT.PSZ.PSZ8, SMT.PRW.PR);     // 8bit read
SMT.PortOut( addr, data, SMT.PSZ.PSZ16, SMT.PRW.PW);    // 16bit write
SMT.PortOut( addr, data, SMT.PSZ.PSZ16, SMT.PRW.PR);    // 16bit read
SMT.PortOut( addr, data, SMT.PSZ.PSZ32, SMT.PRW.PW);    // 32bit write
SMT.PortOut( addr, data, SMT.PSZ.PSZ32, SMT.PRW.PR);    // 32bit read
```

■引数

No.	名前	型	意味
1	addr	int 型	アドレス値
2	data	int 型	データ値
3	size	PSZ	ポートサイズ 定義値 PSZ8 : 8bit PSZ16 : 16bit PSZ32 : 32bit
4	rw	PRW	アクセスタイプ 定義値 PW: ライト PR: リード

■戻り値

型 : int
実行結果を返します。
0（正常終了）又は 1（異常終了）を返します。

PortOut 関数は、ポートアクセス情報を出力します。
ポートサイズ、アクセスタイプの指定には、引数表に記載の定義値を使用してください。

4 効果的なご利用方法

API ライブラリを効果的に使用するためのヒントを記載しています。

4.1 オーバーヘッドを抑止する

出力するデバッグ情報量はライブラリ関数によって異なります。情報が多いほど関数呼び出しのオーバーヘッドが増加しますので、用途に合わせてオーバーヘッドの少ないライブラリ関数をご利用ください。

通常、表中の上の項目ほどオーバーヘッドを抑止できます。

関数名	機能説明
_SMT_UsrMsgTag0	ユーザー定義メッセージタグを出力します。
_SMT_UsrMsgTag1	ユーザー定義メッセージタグと引数 1 個を出力します。
_SMT_UsrMsgTag2	ユーザー定義メッセージタグと引数 2 個を出力します。
_SMT_PortOut	ポート情報を出力します。
_SMT_UsrMsgTag3	ユーザー定義メッセージタグと引数 3 個を出力します。
_SMT_UsrMsgTag4	ユーザー定義メッセージタグと引数 4 個を出力します。
_SMT_Puts	C 言語の puts 相当です。文字列を出力します。
_SMT_Printf	printf 相当です。書式指定付き文字列を出力します。

デバッグプリントメッセージを出力する場合、_SMT_Puts などの API は出力する文字数に比例してオーバーヘッドが増加します。なるべくユーザーメッセージタグ出力を利用されることをおすすめします。外部ファイルに文字列を登録しておき、メッセージに対応する番号のみをトレース情報として出力することで、オーバーヘッドを抑止できます。

4.2 ビルド時にライブラリ関数を無効化する

C/C++言語ソース中に記述するライブラリ関数名を全て大文字にすると、ビルド時にマクロ定義でまとめて無効化できます。Java 言語ソースでは本機能はご使用いただけません。
マクロ定義は 4 種類あり、関数グループ毎にも無効化できます。

マクロ定義	無効になる関数グループ
_SMT_ALL_DIS	デバッグプリント ポートアウト コンテキストスイッチ システムコール
_SMT_PRINT_DIS	デバッグプリント
_SMT_PORT_DIS	ポートアウト
_SMT_OS_DIS	コンテキストスイッチ システムコール

マクロ定義場所の例

- コンパイルオプション
- ヘッダファイル
- ソースファイルの先頭

例えば、ライブラリ関数を以下のようにソースコードに記述していると

```
_SMT_Puts(“Message1”);        // マクロ定義に依存せず常に有効します
_SMT_PUTS(“Message2”);        // マクロ定義によって有効/無効の切り替えをします
```

マクロ定義無しの場合の出力結果

```
Message1
Message2
```

_SMT_ALL_DIS または_SMT_PRINT_DIS のマクロ定義がある場合の出力結果

```
Message1
```

となります。

4.3 デバッグプリント出力をレベル指定してマスクする

デバッグプリント関数群は引数に出力レベル値を持ち、0～15 の値を設定することができます。このレベル値とデバッグプリント出力レベルマスクパターンを組み合わせ、レベルごとに出力を有効または無効化できます。
マスクパターンはビットパターンとして定義され、各ビット位置がレベル値に対応します。ビット値が 0 のレベルは出力され、1 のレベルは出力されません。

マスクパターンは、C/C++言語ソースでは_SMT_SetDebugLevel 関数で、Java 言語ソースでは SetDebugLevel 関数で設定します。
例えば、以下のようにデバッグプリント出力レベルマスク設定値が 0xFFFC の場合、レベル 0 とレベル 1 のデバッグプリントのみ出力されます。

ソースコード記述

```
…
_SMT_SetDebugLevel( 0xFFFC );
_SMT_Puts( 0, “Level 0 Message”);
_SMT_Puts( 1, “Level 1 Message”);
_SMT_Puts( 2, “Level 2 Message”);
_SMT_Puts( 3, “Level 3 Message”);
…
```

出力結果

```
Level 0 Message
Level 1 Message
```

出力レベルマスク設定の初期値は 0x0000 であり、全レベルで出力が有効となっています。この初期値は SMTUserDef.h で以下のように定義されています。

```
#define       SMT_LV_MSK    0x0000
```

5 注意事項

5.1 デバッグプリント関数について

デバッグプリント関数に出力可能な文字列長は、書式変換後の長さで、終端文字を含め必ず 124byte 以内としてください。それ以上の文字列を指定した場合、意図しない動作をする可能性があります。