■　関数引数が VOID ポインタの場合の設定方法

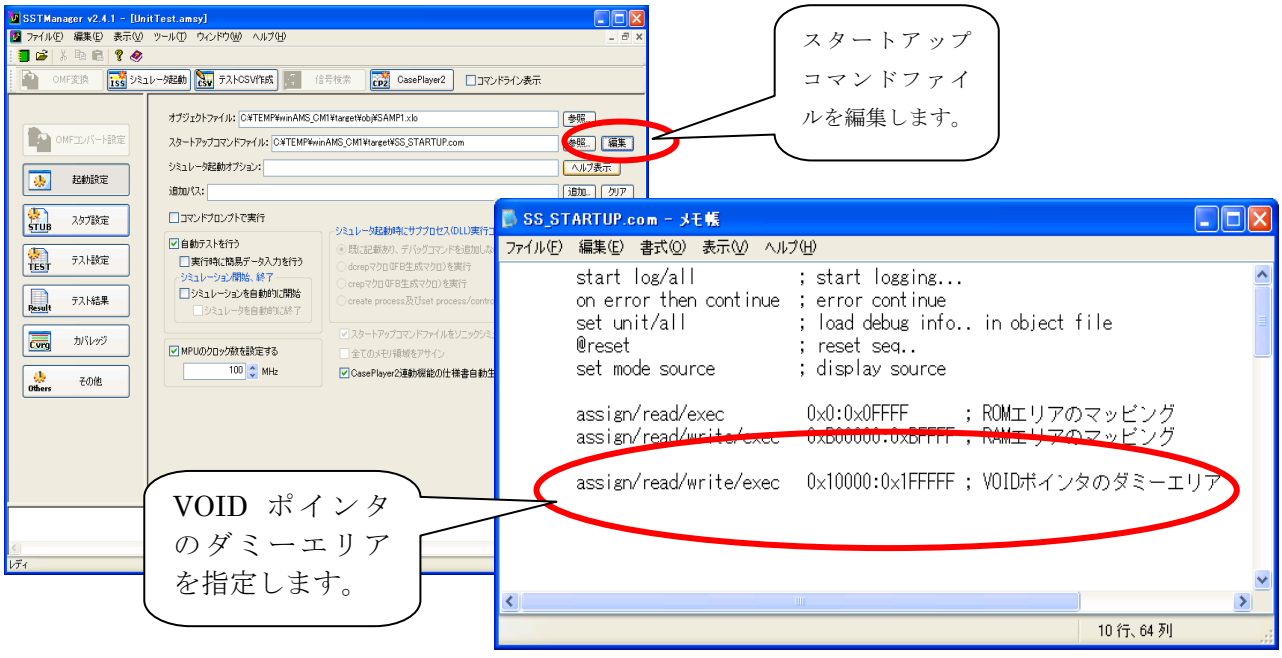
①構造体の型宣言がある場合
カバレッジマスタ winAMS に置ける設定方法は、ポインタ変数にメモリマップされたアドレスを指定を行うことによって、入力データとしての値の設定をシンボル指定が可能となります。

サンプルプログラム

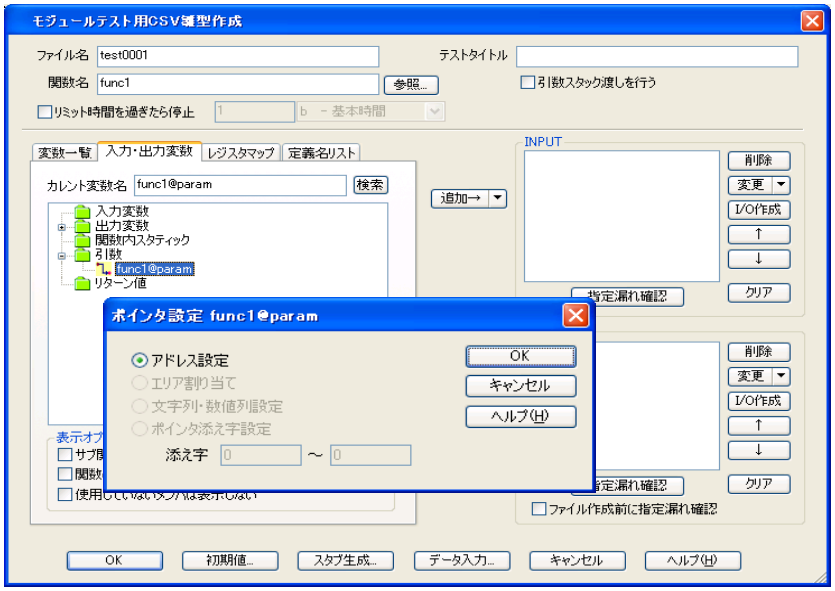
```
struct TEST
{
        int enable;
        int mode;
        int input;
} t_test1;

void func1(void * param)
{
int enable,mode,input;
                enable = ((struct TEST *)param)->enable;
                mode    = ((struct TEST *)param)->mode;
                input   = ((struct TEST *)param)->input;

                if( enable )
                {
                        switch( mode )
                        {
                        case 0:
                                gb_result.data = input;
                                break;
                        case 1:
                                gb_result.data = input * 10;
                                break;
                        case 2:
                                gb_result.data = input * 100;
                                break;
                        case 3:
                                if( input>100 )
                                        gb_result.data = 10000;
                                else
                                        gb_result.data = input*100;
                                break;
                        default:
                                gb_result.data = -1;
                        }
                        // return code
                        gb_result.ret_code = TRUE;
                }
                else
                {
                        gb_result.data = 0;
                        gb_result.ret_code = FALSE;
                }
}
```

VOID * 宣言

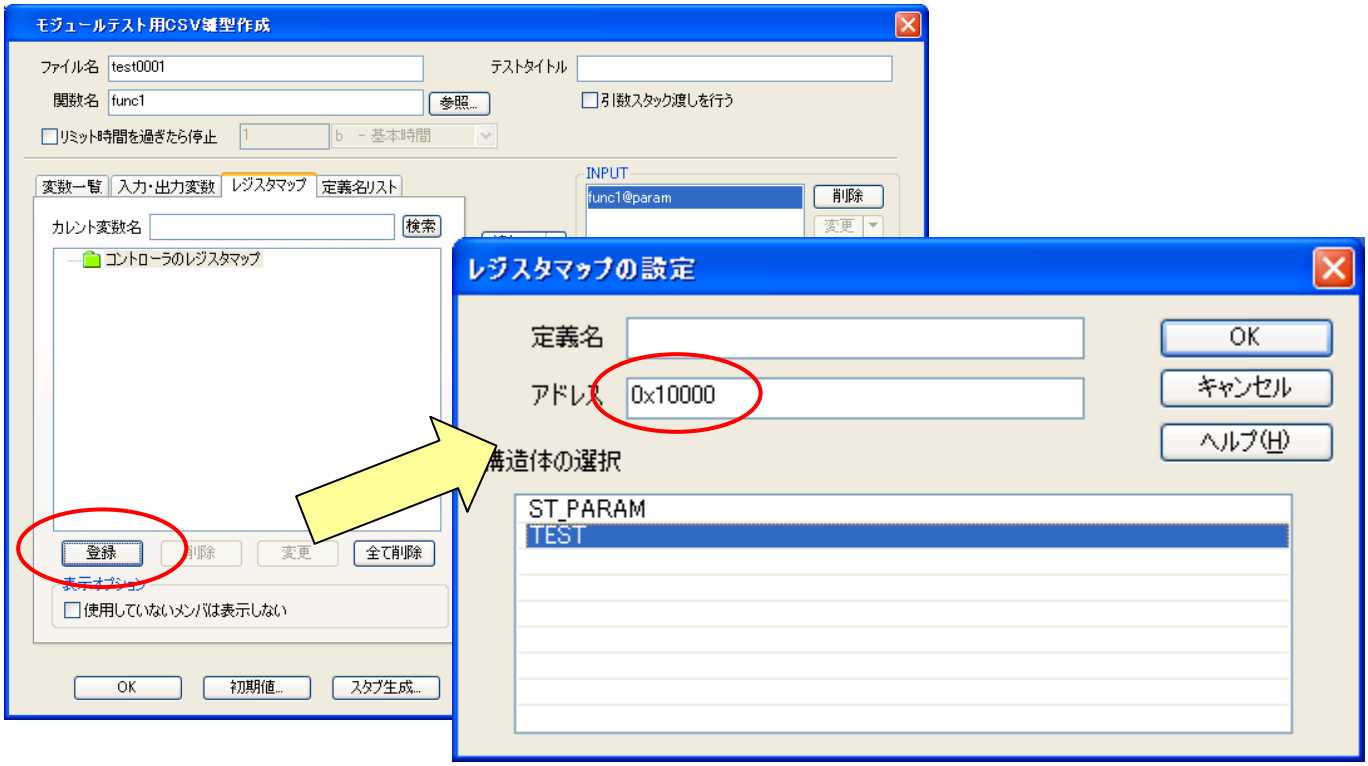
手順１
マイコンシミュレータの空きエリアを使用し、ダミーの VOID ポインターエリアを登録します。
既に、メモリマップ情報が用意されている場合（実オブジェクトの場所）は不要です。
●スタートアップコマンドに ASSIGN コマンドを使用して任意のアドレスマッピングします。



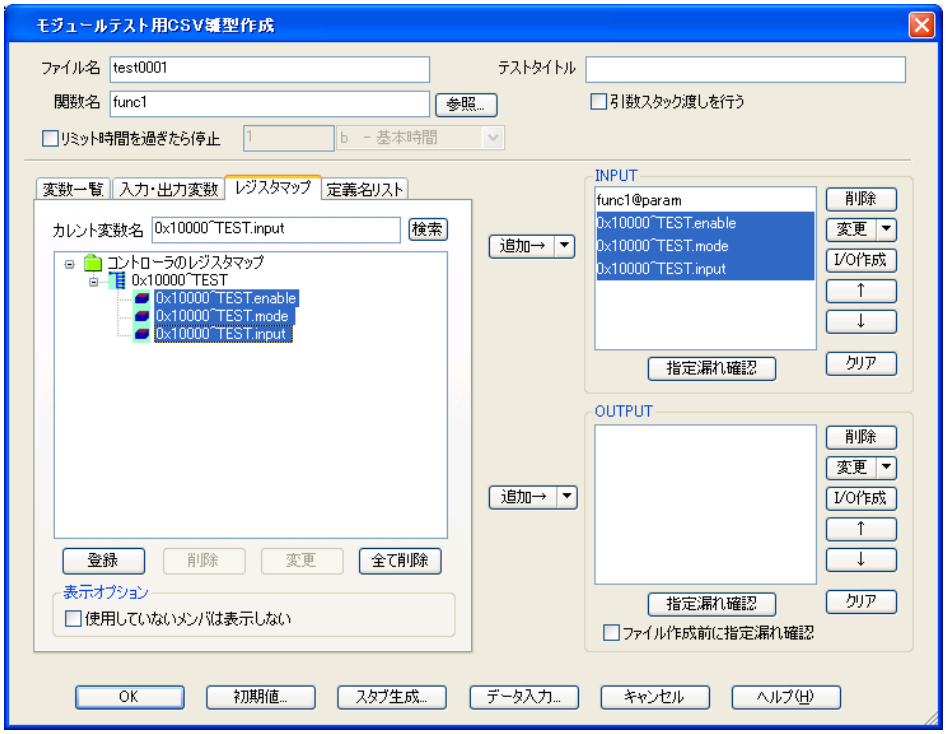
手順２
CSV 雛形作成から、関数引数である func1@parm をアドレス指定で INPUT に登録します。



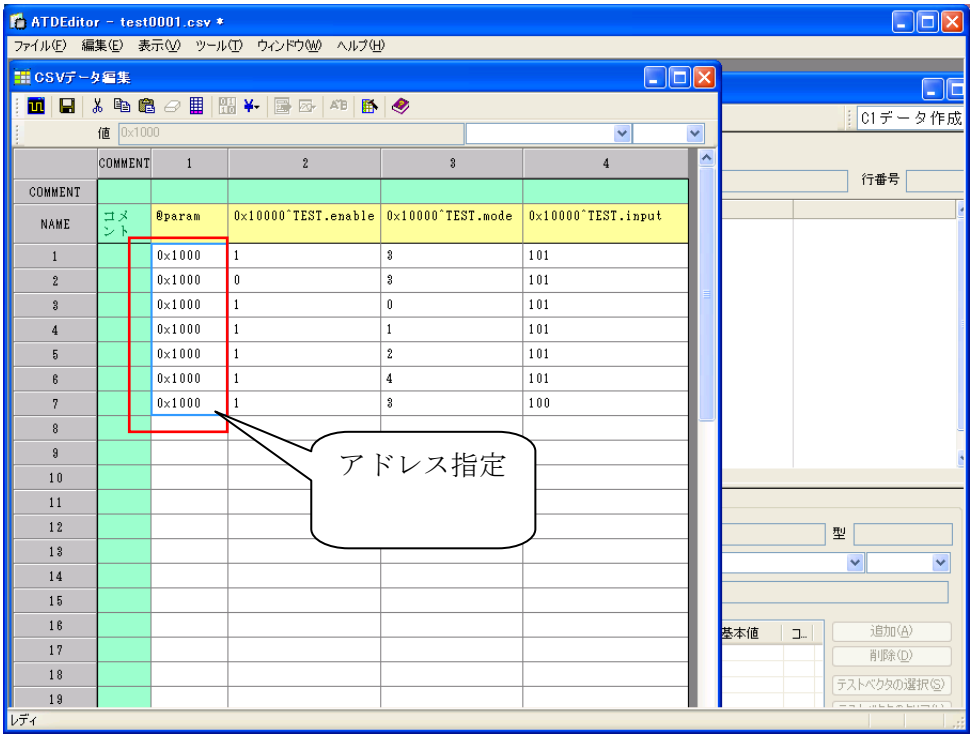
手順３
レジスタマップへタブを変更し“登録“を選択し、レジスタマップ設定を表示します。
その後、構造体を選択し、手順１で設定したダミーエリアのアドレスを指定します。
※定義名は任意で入力します。(例では TEST という定義名にします。)



手順４
レジスタマップから、入力データとして登録するシンボル（メンバー）を INPUT に登録します。



CSV 編集(または Excel)にて手順３で設定したアドレスを引数にポインタアドレスとして記述します。



②構造体の型宣言がない場合(typedef により別名定義されている場合)
下記のサンプルプログラムの様に構造体の型が宣言されていない場合は、先に記述した「レジスタマップを使用する方法」では、テストが実施出来ません。※構造体に型宣言がない為、レジスタマップの構造体が選択できない状態になる為。
今回の様な場合では、ダミーシンボルを作成して、そのシンボルにテストデータを設定してそのアドレスを関数のポインタに設定する事により、テストを実行します。

サンプルプログラム

```
typedef struct
{
	int enable;
	int mode;
	int input;
} TEST;

void func1(void * param)
{
int enable,mode,input;
		enable = ((struct TEST *)param)->enable;
		mode   = ((struct TEST *)param)->mode;
		input  = ((struct TEST *)param)->input;

		if( enable )
		{
			switch( mode )
			{
			case 0:
				gb_result.data = input;
				break;
			case 1:
				gb_result.data = input * 10;
				break;
			case 2:
				gb_result.data = input * 100;
				break;
			case 3:
				if( input>100 )
					gb_result.data = 10000;
				else
					gb_result.data = input*100;
				break;
			default:
				gb_result.data = -1;
			}
			// return code
			gb_result.ret_code = TRUE;
		}
		else
		{
			gb_result.data = 0;
			gb_result.ret_code = FALSE;
		}
}
```

構造体の型宣言がない

VOID *　宣言

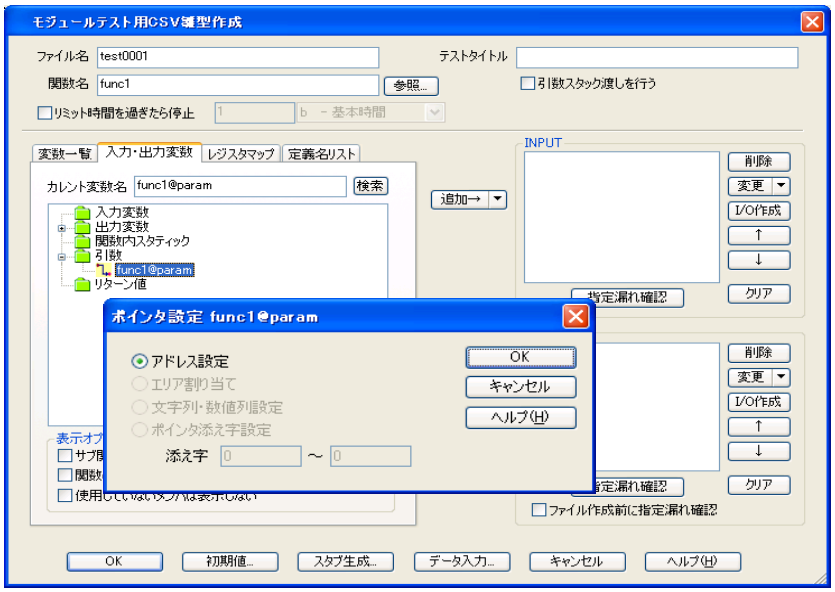
手順１
テストデータを設定する為のダミーシンボル若しくはダミー関数を作成します。
※管理上、テスト対象関数のスタブ関数にシンボルを作成する事を推奨致します。
※ダミーシンボルを作成するファイルは運用に合わせて、任意で決定して下さい。
　作成したファイルをコンパイルして、テスト対象オブジェクトにリンクする必要があります。

サンプルプログラム(ダミーシンボル)

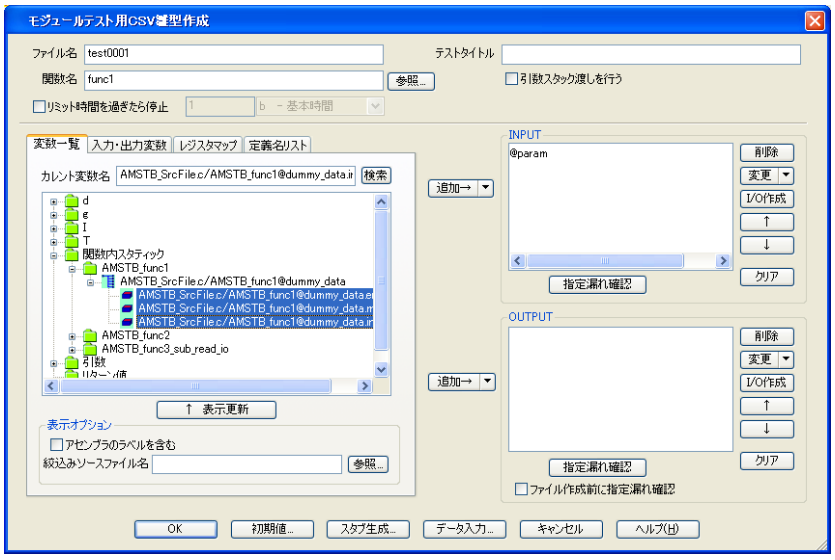
```
■ダミーシンボルの場合
static volatile TEST dummy_data;


■スタブ関数の場合
void AMSTB_func1(void * param)
{
	static volatile TEST dummy_data

	((TEST *)param)->enable = dummy_data.enable;
	((TEST *)param)->mode = dummy_data.mode;
	((TEST *)param)->input = dummy_data.input;
}
```

スタブ関数で対応する場合は、スタブ関数として使用する場合も考慮して、作成して下さい。

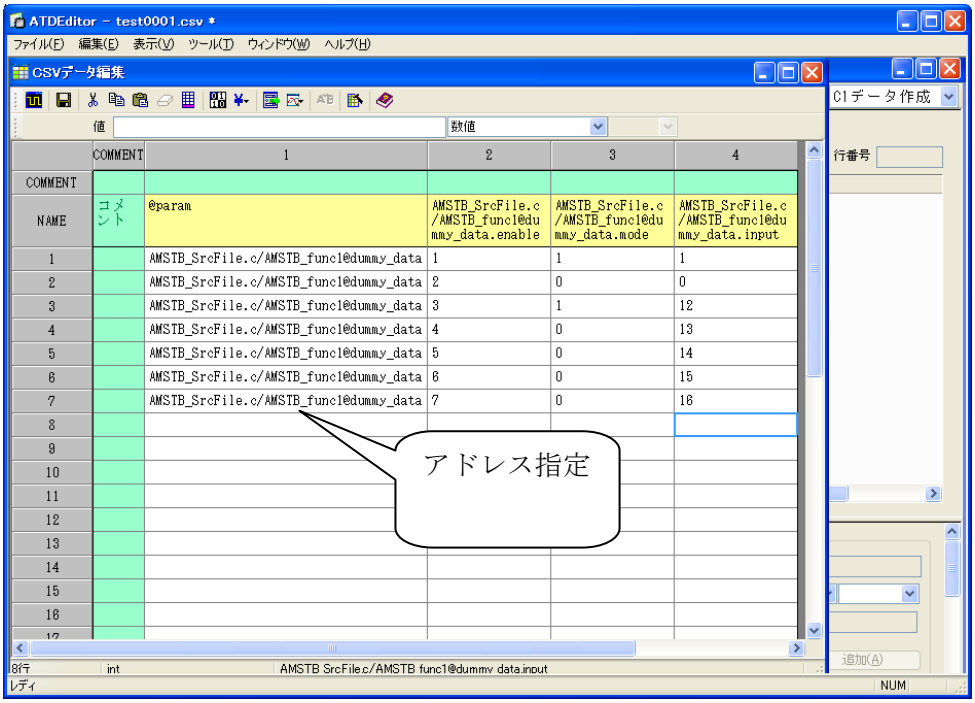
手順２
CSV 雛形作成から、関数引数である func1@parm をアドレス指定で INPUT に登録します。



手順３
ダミーシンボルである dummy_data.enable、dummy_data.mode、dummy_data.input を INPUT に登録します。



手順４
CSV 編集(または Excel)にて手順３で設定したシンボルのアドレスを引数にポインタアドレスとして記述します。



	COMMENT	1	2	3	4
COMMENT					
NAME	コメント	@param	AMSTB_SrcFile.c/AMSTB_func1@dummy_data.enable	AMSTB_SrcFile.c/AMSTB_func1@dummy_data.mode	AMSTB_SrcFile.c/AMSTB_func1@dummy_data.input
1		AMSTB_SrcFile.c/AMSTB_func1@dummy_data	1	1	1
2		AMSTB_SrcFile.c/AMSTB_func1@dummy_data	2	0	0
3		AMSTB_SrcFile.c/AMSTB_func1@dummy_data	3	1	12
4		AMSTB_SrcFile.c/AMSTB_func1@dummy_data	4	0	13
5		AMSTB_SrcFile.c/AMSTB_func1@dummy_data	5	0	14
6		AMSTB_SrcFile.c/AMSTB_func1@dummy_data	6	0	15
7		AMSTB_SrcFile.c/AMSTB_func1@dummy_data	7	0	16
8					
9					
10					
11					
12					
13					
14					
15					
16					