

```

/*****
/*          S B U T I L . C          */
/**-----**/
/* Task      : Allocates functions for initialization of      */
/*            SBBASE structure.                                */
/**-----**/
/* Author     : Michael Tischer / Bruno Jennrich              */
/* Developed on : 03/20/1994                                    */
/* Last update  : 04/06/1995                                   */
/**-----**/
/* COMPILER    : Borland C++ 3.1, Microsoft Visual C++ 1.5    */
/*****
#ifdef __SBUTIL_C
#define __SBUTIL_C

/*-- Add include files -----*/

#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include <io.h>
#include <fcntl.h>

#include "types.h"
#include "sbutil.h"
#include "args.h"

/*****
/* sb_GetEnviron : Initialize SBBASE structure on the basis of passed */
/*                environment string (BLASTER environment variable) */
/**-----**/
/* Input   : pSBBASE - Address of structure to be initialized      */
/*           pEnv    - Address of environment string                */
/* Output  : == 0 : Correct initialization                          */
/*           == -1 : Defective initialization                       */
/* Info   : - This structure must be passed to the individual modules */
/*           (???_SetBase()-functions), which then carry out      */
/*           initializations of the structure                      */
/*           (e.g., DSP version number).                           */
/*****
INT sb_GetEnviron( PSBBASE pSBBASE, PCHAR pEnv )
{
    if( pEnv )
    {
        pSBBASE->iDspPort = atoi( strchr( pEnv, "A" ), -1 );
        pSBBASE->iMixPort = atoi( strchr( pEnv, "M" ), -1 );
        if( pSBBASE->iMixPort == -1 )
            pSBBASE->iMixPort = pSBBASE->iDspPort;
        pSBBASE->iMpuPort = atoi( strchr( pEnv, "P" ), -1 );
        pSBBASE->uDspVersion = 0xFFFFU; /* will be initialized later */
        pSBBASE->iDspIrq = atoi( strchr( pEnv, "I" ), -1 );
        pSBBASE->iDspDmaB = atoi( strchr( pEnv, "D" ), -1 );
        pSBBASE->iDspDmaW = atoi( strchr( pEnv, "H" ), -1 );
        return NO_ERROR;
    }
    return ERROR;
}

/*****
/* sb_Print : Display SBBASE structure */
/**-----**/
/* Input   : pSBBASE - Address of structure to be displayed      */
/*****
VOID sb_Print( PSBBASE pSBBASE )
{
    printf("DSP-Port: 0x%X \n", pSBBASE->iDspPort );
    printf("MIX-Port: 0x%X \n", pSBBASE->iMixPort);
    printf("MPU-Port: 0x%X \n", pSBBASE->iMpuPort );
    if( pSBBASE->uDspVersion != 0xFFFFU )
    {
        printf("DSP-Version: %d.%02d \n", HIBYTE( pSBBASE->uDspVersion ),
               LOBYTE( pSBBASE->uDspVersion ) );
        printf("Card : %s\n", pSBBASE->pDspName );
    }
    else printf("DSP not initialized!\n");
}

```

```

printf("DSP-IRQ: %d \n", pSBBASE->iDspIrq );
printf("DSP- 8 Bit DMA: %d \n", pSBBASE->iDspDmaB );
printf("DSP-16 Bit DMA: %d \n", pSBBASE->iDspDmaW );
}

/*****
/* sb_LoadDriver : Load Sound blaster driver */
**-----**
/* Input : lpName - Name of driver to be loaded (filename) */
/* Output : Address of driver entry point */
/* == NULL - Driver could not be loaded */
*****/
LPVOID sb_LoadDriver( PCHAR pName )
{
    INT iHandle = -1;
    WORD uSegment = 0;

    if ( _dos_open( pName, O_RDWR, &iHandle ) == 0 )
    {
        WORD uDummy;
        LONG lFileSize;
        LONG lNumSeg;
        LPBYTE lpMem;

        /* Determine size of driver */
        lFileSize = filelength( iHandle );
        lNumSeg = lFileSize / 16L + 1; /* Size in paragraphs */
        if( lNumSeg < 0xFFFFU )
        {
            /* Memory must begin at paragraph limit */
            if( _dos_allocmem( ( WORD ) ( lNumSeg ), &uSegment ) == 0 )
            {
                lpMem = MK_FP( uSegment, 0 );
                while( lFileSize )
                {
                    if( _dos_read( iHandle, lpMem,
                        ( lFileSize > 65535L ) ? 65535U :
                        ( WORD ) lFileSize, &uDummy ) == 0 )
                    {
                        WORD iInc;
                        iInc = ( lFileSize > 65535L ) ? 65535U : ( WORD ) lFileSize;
                        lpMem += iInc;
                        lFileSize -= iInc;
                    }
                    else /* Release after read error */
                    {
                        _dos_freemem( uSegment );
                        uSegment = 0;
                        lFileSize = 0;
                    }
                }
                else uSegment = 0;
            }
            _dos_close( iHandle );
        }
        return ( LPVOID ) MK_FP( uSegment, 0 );
    }
}

/*****
/* sb_UnloadDriver : Remove Sound Blaster driver from memory */
**-----**
/* Input : lpEntry : Address of memory block that the driver */
/* allocates. */
*****/
VOID sb_UnloadDriver( LPVOID lpEntry )
{
    if( lpEntry ) _dos_freemem( FP_SEG( lpEntry ) );
}
#endif

```