

```

/*****
/*          D M A U T I L . H          */
**-----**
/* task          : Header file for DMA_UTIL.C          */
**-----**
/* author          : Michael Tischer / Bruno Jennrich          */
/* developed on    : 3/20/1994          */
/* last update     : 4/07/1995          */
**-----**
/* COMPILER        : Borland C++ 3.1, Microsoft Visual C++ 1.5          */
/*****
#ifndef __INC_DMA_UTIL_H
#define __INC_DMA_UTIL_H

#include "types.h"

/*- read bits from the status registers (0x08, 0xD0 ) -----*/
#define STATUS_REQ3 0x80          /* bit set: respective DMA channel */
#define STATUS_REQ2 0x40          /* contains DMA request.          */
#define STATUS_REQ1 0x20          /* Request                          */
#define STATUS_REQ0 0x10
#define STATUS_TC3 0x08          /* bit set: Since last reading of the */
#define STATUS_TC2 0x04          /* status register, a DMA            */
#define STATUS_TC1 0x02          /* transfer is terminated.            */
#define STATUS_TC0 0x01          /* Terminal Count                    */

/*- write bit from the command register (0x08, 0xD0) -----*/
/*- bit 7 set: DMA Acknowledge line HIGH active -----*/
#define COMMAND_DACKLEVEL 0x80

/*- bit 6 set: REQ Acknowledge line LOW active -----*/
#define COMMAND_DREQLEVEL 0x40

/*- after DMA-Request, controller sends DMA-Acknowledge! -----*/
/*- bit 5 set: EXTENDED Write, else LATE Write -----*/
#define COMMAND_EXTWRITE 0x20

/*- bit 4 set: established priority -----*/
#define COMMAND_FIXEDPRI 0x10

/*- bit 3 set: compromised clock speed -----*/
#define COMMAND_COMPRESS 0x08

/*- bit 2 set: controller disabled -----*/
#define COMMAND_INACTIVE 0x04

/*- bit 1 set: address hold for channel 0/4 disabled -----*/
#define COMMAND_ADH0 0x02

/*- bit 0 set: memory/memory, else memory/peripheral -----*/
#define COMMAND_MEM2MEM 0x01

/*- write bits for the request register ( 0x09, 0xD2 ) -----*/
#define REQUEST_RESERVED 0xF8          /* set reserved bits always =0 */
#define REQUEST_SET 0x04          /* set DMA request */
#define REQUEST_CLR 0x00          /* clear DMA request */
#define REQUEST_MSK 0x03 /* Indicate channel for both lower bits */

/*- write bits for the channel masking register ( 0x0A, 0xD4 ) -----*/
#define CHANNEL_RESERVED 0xF8          /* set reserved bits always =0 */
#define CHANNEL_SET 0x04          /* mask/lock DMA channel */
#define CHANNEL_CLR 0x00          /* clear DMA channel */
#define CHANNEL_MSK 0x03 /* Indicate channel for both lower bits */

/*- write bits from the mode register (0x0B, 0xD6) -----*/
#define MODE_DEMAND 0x00          /* transfer "on call" */
#define MODE_SINGLE 0x40          /* transfer single values */
#define MODE_BLOCK 0x80          /* block transfer */
#define MODE_CASCADE 0xC0          /* transfer cascade */

#define MODE_DECREMENT 0x20          /* decrement */
#define MODE_AUTOINIT 0x10          /* auto initialization after end */

#define MODE_VERIFY 0x00          /* verify */
#define MODE_WRITE 0x04          /* write to memory */
#define MODE_READ 0x08          /* read from memory */

```

```

#define MODE_INVALID      0x0C                                /* invalid */

#define MODE_CHANNELMSK 0x03 /* Indicate channel for both lower bits */

/*- Prototypes -----*/
VOID dma_SetRequest ( INT iChan );
VOID dma_ClrRequest ( INT iChan );
VOID dma_SetMask    ( INT iChan );
VOID dma_ClrMask    ( INT iChan );
VOID dma_ClrFlipFlop ( INT iChan );
BYTE dma_ReadStatus ( INT iChan );
WORD dma_ReadCount  ( INT iChan );
VOID dma_Mem2Mem( BYTE bPage, WORD uSource, WORD uDest, WORD uSize );
WORD dma_Until64kPage( LPVOID lpMem, WORD uSize );
VOID dma_SetChannel ( INT iChan, LPVOID lpMem,
                      WORD uSize, BYTE bMode );

#define DMA_8BIT ( BYTE) 0x00
#define DMA_16BIT ( BYTE) 0x01

LPVOID dma_AllocMem ( WORD uSize );
VOID dma_Free      ( LPVOID lpMem );
#endif

```