```c
/*
 * dma.c
 *
 *            Functions for using the DMA controller
 *
 */

#include <dma.h>

// initialise the DMA controller
void DMA_init(void) {

       /* Select MAT0.0 instead of UART0 Tx */
       LPC_SC->DMAREQSEL = 0x01;

       LPC_SC->PCONP |= 1<<29;          //Power GPDMA module

       /* Enable the GPDMA controller */
       LPC_GPDMA->DMACConfig = 1;

       /* Enable synchro logic request 8, MAT0.0 */
       LPC_GPDMA->DMACSync   = 1 << 8;
}

// Configure the DMA controller
void DMA_config(signed long* outputBuffer) {

       LPC_GPDMACH0->DMACCSrcAddr  = (uint32_t) outputBuffer;
       LPC_GPDMACH0->DMACCDestAddr = (uint32_t) &(LPC_DAC->DACR);
       LPC_GPDMACH0->DMACCLLI      = 0;           // linked lists for ch0
       LPC_GPDMACH0->DMACCControl  = bufferSize// transfer size (0 - 11) = 32
                    | (0 << 12)        // source burst size (12 - 14) = 1
                    | (0 << 15)        // destination burst size (15 - 17) = 1
                    | (2 << 18)        // source width (18 - 20) = 32 bit
                    | (2 << 21)        // destination width (21 - 23) = 32 bit
                    | (0 << 24)        // source AHB select (24) = AHB 0
                    | (0 << 25)        // destination AHB select (25) = AHB 0
                    | (1 << 26)        // source increment (26) = increment
                    | (0 << 27)        // destination increment (27) = no increment
                    | (0 << 28)        // mode select (28) = access in user mode
                    | (0 << 29)        // (29) = access not bufferable
                    | (0 << 30)        // (30) = access not cacheable
                    | (0 << 31);       // terminal count interrupt disabled

       LPC_GPDMACH0->DMACCConfig   =  1           // channel enabled (0)
                    | (0 << 1)         // source peripheral (1 - 5) = none
                    | (8 << 6)         // destination request peripheral (6 - 10) = MAT0.0
                    | (1 << 11)        // flow control (11 - 13) = MEM. to PER.
                    | (0 << 14)        // (14) = mask out error interrupt
                    | (0 << 15)        // (15) = mask out terminal count interrupt
                    | (0 << 16)        // (16) = no locked transfers
                    | (0 << 18);       // (27) = no HALT
}

void DMA_waitForTransfer(void) {
       // wait for the DMA to finish
       while (LPC_GPDMACH0->DMACCConfig & 1);
}
```