```c
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include <device.h>
#define BlockSize 16384
#define height 4095
uint8 sourceArray[BlockSize];
uint8 destinationArray[BlockSize]={0};
volatile uint8 flag_DMADone;
uint8 DMA_Chan;
uint8 DMA_TD[1];
uint64 time = 0;
int counter = 0;
CY_ISR (Interrupt){
    Timer_1_Stop();
    time = ((4294967296 - Timer_1_ReadCounter())/24);
    LCD_Position(1u,0u);
    LCD_PrintNumber(time);
    CyPins_SetPin(LED_1_0);
}

void main()
{
  /* Defines for DMA_1 */
#define DMA_1_BYTES_PER_BURST 4
#define DMA_1_REQUEST_PER_BURST 0
#define DMA_1_SRC_BASE (CYDEV_SRAM_BASE)
#define DMA_1_DST_BASE (CYDEV_SRAM_BASE)
    #define True 1
    #define False 0
char same = True;
/* Variable declarations for DMA_1 */
/* Move these variable declarations to the top of the function */
uint8 DMA_1_Chan;
uint8 DMA_1_TD[5];

/* DMA Configuration for DMA_1 */
DMA_1_Chan = DMA_1_DmaInitialize(DMA_1_BYTES_PER_BURST, ↩
DMA_1_REQUEST_PER_BURST,
    HI16(DMA_1_SRC_BASE), HI16(DMA_1_DST_BASE));
DMA_1_TD[0] = CyDmaTdAllocate();
DMA_1_TD[1] = CyDmaTdAllocate();
DMA_1_TD[2] = CyDmaTdAllocate();
DMA_1_TD[3] = CyDmaTdAllocate();
```

```
DMA_1_TD[4] = CyDmaTdAllocate();
CyDmaTdSetConfiguration(DMA_1_TD[0], 4092, DMA_1_TD[1], TD_SWAP_EN | 
TD_SWAP_SIZE4 | TD_INC_SRC_ADR | TD_INC_DST_ADR | TD_AUTO_EXEC_NEXT);
CyDmaTdSetConfiguration(DMA_1_TD[1], 4092, DMA_1_TD[2], TD_SWAP_EN | 
TD_SWAP_SIZE4 | TD_INC_SRC_ADR | TD_INC_DST_ADR | TD_AUTO_EXEC_NEXT);
CyDmaTdSetConfiguration(DMA_1_TD[2], 4092, DMA_1_TD[3], TD_SWAP_EN | 
TD_SWAP_SIZE4 | TD_INC_SRC_ADR | TD_INC_DST_ADR | TD_AUTO_EXEC_NEXT);
CyDmaTdSetConfiguration(DMA_1_TD[3], 4092, DMA_1_TD[4], TD_SWAP_EN | 
TD_SWAP_SIZE4 | TD_INC_SRC_ADR | TD_INC_DST_ADR | TD_AUTO_EXEC_NEXT);
CyDmaTdSetConfiguration(DMA_1_TD[4], 16, CY_DMA_DISABLE_TD, TD_SWAP_EN | 
TD_SWAP_SIZE4 | DMA_1__TD_TERMOUT_EN | TD_INC_SRC_ADR | TD_INC_DST_ADR);
CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)sourceArray), LO16((uint32)
destinationArray));
CyDmaTdSetAddress(DMA_1_TD[1], LO16((uint32)sourceArray+4092), LO16((uint32)
destinationArray+4092));
CyDmaTdSetAddress(DMA_1_TD[2], LO16((uint32)sourceArray+8184), LO16((uint32)
destinationArray+8184));
CyDmaTdSetAddress(DMA_1_TD[3], LO16((uint32)sourceArray+12276), LO16((uint32)
destinationArray+12276));
CyDmaTdSetAddress(DMA_1_TD[4], LO16((uint32)sourceArray+16368), LO16((uint32)
destinationArray+16368));
CyDmaChSetInitialTd(DMA_1_Chan, DMA_1_TD[0]);
CyDmaChEnable(DMA_1_Chan, 1);



    /* Start LCD and enable all interrupts */
    LCD_Start();
    /* Enable Interrupts */
    ISR_Start();
    ISR_StartEx(Interrupt);
    CYGlobalIntEnable;
    /* Display the destination array contents before the data transfer */
    LCD_Position(0, 0);
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    int i,j,k;
    for(i = 0;i < BlockSize; i++){
           sourceArray[i] = i % 256;
    }
    for(j = 0;j < BlockSize; j++){
           destinationArray[j] = 0;
    }
    Timer_1_Start();
    CyDmaChSetRequest(DMA_Chan, CPU_REQ);

    int row = 1;
    int count4 = 0;
    for(k = 0;k < BlockSize; k++){
            if(destinationArray[k] != sourceArray[(row*4)-count4-1]){
                same = False;
                counter++;
            }
            count4++;
            if(count4 == 4){
                count4 = 0;
                row++;
            }
    }


    for(;;)
    {
        LCD_Position(0u,0u);
        if(same == True)
        LCD_PrintString("True");
        if(same == False)
        LCD_PrintString("False");
        LCD_Position(0u,7u);
        LCD_PrintInt16(counter);
        /* Place your application code here. */
    }
}

/* [] END OF FILE */
```