MicroConverter® Technical Note – uC019
DMA To XRAM on the ADuC831/832

Introduction

The on-chip ADC has been designed to run at a maximum conversion speed of 5 uS (200 kHz sampling rate). When converting at this rate the ADuC831/ADuC832 micro has 5 uS to read the ADC result and store the result in memory for further post processing all within 5 uS otherwise the next ADC sample could be lost. In an interrupt driven routine the micro would also have to jump to the ADC Interrupt Service routine which will also increase the time required to store the ADC results. In applications where the ADuC831/ADuC832 cannot sustain the interrupt rate, an ADC DMA mode is provided.

Configuring RAM for DMA

The DMA Address pointer must be set to the start address of where the ADC Results are to be written. This is done by writing to the DMA mode Address Pointers, DMAL, DMAH, DMAP. DMAL must be written first, followed by DMAH and then by DMAP.

The external memory must be preconfigured. This consists of writing the required ADC channel Ids into the top four bits of every second memory location in the external SRAM starting at the first address specified by the DMA address pointer. As the ADC DMA mode operates independently form the ADuC831/ADuC832 core it is necessary to provide it with a stop command. This is done by duplicating the last channel ID to be converted followed by "1111" into the next channel selection field. A typical preconfiguration of external memory is as follows.

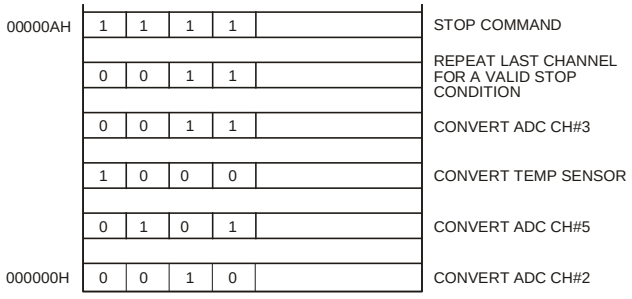

Figure 1

DMA Initiation

The DMA is initiated by writing to the ADC SFRs in the following sequence.

a. ADCCON2 is written to enable the DMA mode, i.e
   MOV ADCCON2, #40H ; DMA Mode enabled
b. ADCCON1 is written to configure the conversion time and power up of the ADC. It can also enable Timer 2 driven conversions or External Triggered conversions if required.
c. ADC conversions are initiated. This is done by starting single conversions, starting Timer 2 running for Timer 2 conversions or by receiving an external trigger.

When the DMA conversions are completed, the ADC interrupt bit ADCI is set by hardware and the external SRAM contains the new ADC conversion results as shown below. It should be noted that no result is written to the last two memory locations.

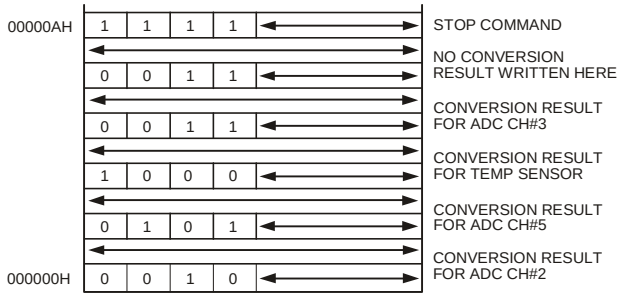

Figure 2

Using Internal and/or External XRAM

The CFG83x.0 bit determines whether or not the internal XRAM is enabled. This memory maps in as shown in figure 3.

If the internal XRAM is enabled the user can use both it and the external XRAM for DMA. The RAM is set up as above and the DMA results automatically switch from the internal XRAM to external XRAM.

If the internal XRAM is disabled, DMA can only be used to send results to the external XRAM.

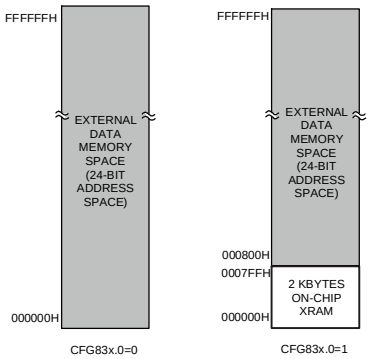

Figure 3

XRAM and PORT 0/2 Access

During ADC DMA mode the MicroConverter is free to continue code execution. However certain features are gated "off" during the ADC DMA since they are being used by the DMA mode.

Access to Ports 0 and 2 are also disabled when performing DMA conversions to the external XRAM since these ports are used to interface the external XRAM. Note that during DMA to the internally contained XRAM ports 0 and 2 are available for use.

Since the DMA mode uses the XRAM data-bus during DMA operation access to both the internal and external XRAM is affected by DMA operation.

If the user has enabled the internal XRAM and doing DMA into the internal XRAM the mircocontroller core **will not** be able to access the XRAM.

If the user has not enabled the internal XRAM and is performing DMA conversions to an external XRAM, the microcontroller **will not** be able to access the external XRAM.

If the user has enabled the internal XRAM but is doing DMA conversions to XRAM locations after 7FFH (i.e. to an external XRAM) then the microcontroller core **will** be able to access the internal XRAM.

The extended stack pointer can be enabled in the CFG83x register. It will use XRAM once the 256 bytes of internal RAM are used. The operation of the extended stack pointer will be affected in the same way as for the microcontroller core.

DMA 'C' Example

In the following example DMA conversions are performed on channel 0 of the ADC. Using the Menu driven routine results can be displayed back on a hyperterminal window.

Note: The following code is for the ADuC832. The same code can be used for the ADuC831 if the baud rate is adjusted for 9600. For an 11.0592MHz crystal this would be

```
SCON = 0x052;
T3CON = 0x085;
T3FD = 0x08;
```

in the `init832` function. The line `pllcon = 0x00;` should be deleted.

```
/*************************************************************************

 Author        : ADI - Apps            www.analog.com/MicroConverter

 Date          : APRIL 2002

 File          : DMAEXAPLE.C

 Hardware      : ADuC832

 Description    : DMA MENU DRIVEN EXAMPLE CODE

*************************************************************************/

#pragma DEBUG OBJECTEXTEND CODE // pragma lines can contain state C51
#include <stdio.h>              // declarations for I/O functions
#include <ctype.h>
#include <ADuC832.h>            // 8052 & ADuC832 predefined symbols
int i=0,flag=0;

void adc_int() interrupt 6{
    ADCCON1=0x00;
    flag  = 1;
        return;
            }

void init832(void)              // Initialize internal peripherals
 {
  PLLCON=0x00;
  flag = 0;

  SCON = 0x52 ;                 // 8bit, noparity, 1stopbit
  T3CON = 0x85;
  T3FD = 0x02D;
 }

void memDump(unsigned int sadr, unsigned int eadr)
 {
  unsigned char xdata *xramPtr;

  printf("       +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F\n");

  sadr&= 0xff00;
  for(xramPtr=(char xdata *)sadr; xramPtr <= (char xdata *)eadr; xramPtr+= 16)
   {
    printf( "%04P %02BX %02BX %02BX %02BX ",xramPtr,*(xramPtr+0),*(xramPtr+1),*(xramPtr+2),*(xramPtr+3));
    printf(      "%02BX %02BX %02BX %02BX "        ,*(xramPtr+4),*(xramPtr+5),*(xramPtr+6),*(xramPtr+7));
    printf(      "%02BX %02BX %02BX %02BX "        ,*(xramPtr+8),*(xramPtr+9),*(xramPtr+10),*(xramPtr+11));
    printf(      "%02BX %02BX %02BX %02BX\n"       ,*(xramPtr+12),*(xramPtr+13),*(xramPtr+14),*(xramPtr+15));
    if (RI)
     if (getchar()==0x03) break;
   }
  printf("\n");
 }
```

```
void prepareDma(void)
 {
  unsigned int xdata *xdatPtr;

  for(xdatPtr=0; xdatPtr<=0x7ffd; xdatPtr++)
   {*xdatPtr= 0x0000;}
  xdatPtr=0x7ffe; *xdatPtr=0xf000;
 }



void DmaSingle(void)            // Single DMA Loop
 {

    ADCCON1=0x00;               // Make sure ADC is in Power Down mode
    DMAL=0;
    DMAH=0;
    DMAP=0;
    prepareDma();
    ADCCON2=0x40;               // Enable the DMA
    ADCCON1=0xA4;               //
    EA = 1;
    EADC = 1;

                                // LAUNCH DMA conversion...  when finished, ADC interrupt will clear C
    CCONV = 1;
    while (flag == 0)
    {printf("loop\n");}

 }



void DmaTimer2(void)            // Timer 2 DMA Loop
 {
    DMAL=0;
    DMAH=0;
    DMAP=0;
    prepareDma();

    RCAP2L=0x0F6;               // sample period = 2 * T2 reload prd
    RCAP2H=0x0FF;               //    = 2*(10000h-FFF6h)*5.722us
    TL2=0x0F6;                  //    = 2*9*5.722us
    TH2=0x0FF;                  //    = 102.99us

    ADCCON2=0x40;               // Enable the DMA
    ADCCON1=0xA6;
    EA = 1;
    EADC = 1;

    TR2 = 1;                    //Start DMA
     while (flag == 0)
    {printf("loop\n");}
}
```

```
void DmaExternal(void)          // DMA on an external trigger
 {
   ADCCON1=0x00;                // Make sure ADC is in Power Down mode
    DMAL=0;
    DMAH=0;
    DMAP=0;
    prepareDma();
    ADCCON2=0x40;               // Enable the DMA
    ADCCON1=0xA4;               //
    EA = 1;
    EADC = 1;

    ADCCON1 ^= 0x01;            // enable hardware CONVST pin
    while (flag == 0)
    {printf("loop\n");}
 }


void dispHelp(void)
 {
  printf("Dump XRAM ssss - eeee D 0xssss 0xeeee:\n");
  printf("DMA operation Continuous Conversions DMA LOOP M\n");
  printf("TIMER2 DMA Operation  N\n");
  printf("External Trigger Operation O\n");
  printf("\n");
 }

 /*************************************************************************
MAIN function
*************************************************************************/
void main(void)
 {
  unsigned char command;
  unsigned int sadr;
  unsigned int eadr;

  init832();

  printf("   \n\n"); //Sync
  printf("Check Program - Type ? for help\n");

  while(1)
   {
   flag=0;
   printf("Cmd>"); scanf("%c",&command); command=toupper(command);
     if(command != 'D') printf("\n");

    switch (command)
     {
      case 'D': scanf("%x %x",&sadr,&eadr);
                memDump(sadr,eadr);
                break;
      case '?': dispHelp(); break;
      case 'M': DmaSingle(); break;
      case 'N': DmaTimer2(); break;
      case 'O': DmaExternal(); break;

      default:  if(command!=0x0d) printf("Unknown command 0x%02BX- Hit ? for help\n\n",command);
     }
   }
 }
```