TN06004

LPC2000 ADC code example

Paul Seerden – 2006 March 27

Technical note

Introduction

This technical note gives two software examples showing the use the 10-bit A/D converter of the Philips Semiconductors LPC2000 microcontroller family. The examples are written for the LPC2129 (and tested on an MCB2100 board), but are valid for all earlier Philips ARM devices (without individual result registers).

Software controlled start – Polling mode

In the first software example all four analog inputs are converted sequentially. UART0 is used to send the conversion results to a PC running a terminal emulator program (for example HyperTerminal). The UART driver routines are not shown in this note.

The function *ADC_Read()* initializes the AD converter, selects the right channel and starts the conversion. Next, it waits for the conversion to be completed and it returns the 10-bit ADC value.

```
#include <LPC21xx.H>                   // LPC21xx definitions

extern void UART0_Init(void);
extern void Print12B(unsigned short w);
extern void PrintString(const char *s);


static unsigned short ADC_Read(unsigned char ch)
{
  unsigned int i;

    ADCR  = 0x00200300 | ch;           // Init ADC (Pclk = 12MHz) and select channel
    ADCR |= 0x01000000;                // Start A/D Conversion
    do
    {
        i = ADDR;                      // Read A/D Data Register
    } while ((i & 0x80000000) == 0);   // Wait for end of A/D Conversion

    return (i >> 6) & 0x03FF;          // bit 6:15 is 10 bit AD value
}

int main(void)
{
    UART0_Init();                      // Initialize UART0

    PrintString("\nLPC2129 ADC test:\n\n"
                  "AIN0   AIN1   AIN2   AIN3\n\n");

    while (1)
    {
        Print12B(ADC_Read(1));        // convert and print channel AIN0
        PrintString("    ");
        Print12B(ADC_Read(2));        // convert and print channel AIN1
        PrintString("    ");
        Print12B(ADC_Read(4));        // convert and print channel AIN2
        PrintString("    ");
        Print12B(ADC_Read(8));        // convert and print channel AIN3
        PrintString("\r");
    }
}
```

Burst mode – Interrupt driven

The second example shows the conversion results using the same UART0 (send output) routines. The AD converter however, is now initialized in burst - and interrupt driven mode. In burst mode the AD converter does repeated conversions scanning the input channels selected by the SEL field of the A/D control register ADCR (in our case all four channels of an LPC2129). After every conversion an interrupt (VIC channel 0) is generated. Inside the interrupt handler the CHN (channel) bits of the data register are used to determine which input channel was converted.

```
#include <LPC21xx.H>                   // LPC21xx definitions

extern void UART0_Init(void);
extern void Print12B(unsigned short w);
extern void PrintString(const char *s);

static unsigned short ADCresult[4];


void ADC_Isr(void) __irq
{
  unsigned int r,ch;

    r = ADDR;                          // Read Data Register and clear DONE flag
    ch = (r >> 24) & 0x07;             // which channel was converted
    ADCresult[ch] = (r>>6) & 0x03FF;   // bit 6:15 is 10 bit AD value
    VICVectAddr = 0;                   // reset VIC
}

int main(void)
{
    VICVectAddr0  = (unsigned int) &ADC_Isr;
    VICVectCntl0  = 0x32;              // Channel0 on Source#18 ... enabled
    VICIntEnable |= 0x40000;           // 18th bit is the ADC

    ADCR  = 0x0020780F;                // Init ADC (Pclk = 12MHz)
    ADCR |= 0x00010000;                // start burst mode now, see errata ADC.2

    UART0_Init();                      // Initialize UART0

    PrintString("\nLPC2129 ADC test:\n\n"
                  "AIN0   AIN1   AIN2   AIN3\n\n");

    while (1)
    {
        Print12B(ADCresult[0]);        // print result channel AIN0
        PrintString("    ");
        Print12B(ADCresult[1]);        // print result channel AIN1
        PrintString("    ");
        Print12B(ADCresult[2]);        // print result channel AIN2
        PrintString("    ");
        Print12B(ADCresult[3]);        // print result channel AIN3
        PrintString("\r");
    }
}
```

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.