使用MAXQ2000评估板的应用实例

由于提供了与这些工具集成的标准ANSI C工具与开发环境，对新的或不熟悉的处理器，大大简化了应用程序开发。为MAXQ产品线处理器提供的工具包括IAR的ANSI C编译器以及IAR Embedded Workbench集成开发环境。只要具备这些程序并具有MAXQ特殊用途寄存器的基础知识，开发者可以快速简单地开始为MAXQ架构编写应用程序。为了说明MAXQ架构的开发过程有多简单，最方便的办法就是举一个应用程序实例。

由于提供了与这些工具集成的标准ANSI C工具与开发环境，对新的或不熟悉的处理器，大大简化了应用程序开发。

这里提到的应用程序使用了MAXQ2000处理器以及MAXQ2000评估板。MAXQ2000具有广泛的集成外设，包括：

- 132字段的LCD控制器
- 集成SPI端口，具备主机与从机模式
- 1-Wire总线主机
- 两个串行UART
- 硬件乘法器
- 三个16位定时器/计数器
- 看门狗定时器
- 32位实时时钟，具有亚秒与日历闹钟
- 支持在线调试的JTAG接口

应用程序概述

该实例展示了LCD控制器、SPI端口的主机模式、UART之一、硬件乘法器，以及定时器之一的使用。定时器用来产生周期性的中断。出现中断时，MAXQ2000读取一个温度读数，并用LCD与其中一个串口输出结果。SPI端口与包含ADC的MAX1407数据采集系统(DAS)接口。将热敏电阻与MAX1407的ADC相连，然后获得温度读数。

LCD控制器的使用

为了使用LCD，必须配置两个控制寄存器。一旦这些寄存器设置完毕，将LCD数据寄存器之一中的位置位，就可以点亮LCD上的字段。下列代码说明了在该应用实例中如何配置LCD控制器。

```
void initLCD()
{
  LCRA_bit.FRM = 7;    // Set up frame frequency.
  LCRA_bit.LCCS = 1;   // Set clock source to HFClk / 128.
  LCRA_bit.DUTY = 0;   // Set up static duty cycle.
  LCRA_bit.LRA = 0;    // Set R-adj to 0.
  LCRA_bit.LRIGC = 1; // Select external LCD drive power.

  LCFG_bit.PCF = 0x0F;// Set up all segments as outputs.
  LCFG_bit.OPM = 1;    // Set to normal operation mode.
  LCFG_bit.DPE = 1;    // Enable display.
}
```

通过 SPI通信

对SPIB寄存器的写操作将启动SPI主机与从机之间的双向通信。

三个寄存器用来控制MAXQ2000支持的不同SPI模式。为了与MAX1407通信，用以下代码初始化SPI部分，并将其置为正确的模式。

```
PD5 |= 0x070;             //  Set CS, SCLK, and DOUT pins as output.
PD5 &= ~0x080;            //  Set DIN pin as input.
SPICK = 0x10;             //  Configure SPI for rising edge, sample input
SPICF = 0x00;             //  on inactive edge, 8 bit, divide by 16.
SPICN_bit.MSTM = 1;       //  Set Q2000 as the master.
SPICN_bit.SPIEN = 1;      //  Enable SPI.
```

只要设置了SPI配置寄存器，就可以用SPIB寄存器发送与接收数据。对该寄存器进行写操作将启动SPI主机与从机之间的双向通信。传输完成后，SPICN寄存器中的STBY位被置位。以下给出了SPI发送与接收的代码。

```
unsigned int sendSPI(unsigned int spib)
{
  SPIB = spib;            //  Load the data to send
  while(SPICN_bit.STBY);  //  Loop until the data has been sent.
  SPICN_bit.SPIC = 0;     //  Clear the SPI transfer complete flag.
  return SPIB;
}
```

写串行端口

在该应用实例中，用MAXQ2000的一个串口输出当前的温度读数。数据写到端口之前，应用程序必须先设置波特率与串口模式。此外，需要初始化几个寄存器来开放串口通信。

```
void initSerial()
{
  SCON0_bit.SM1 = 1;      //  Set to Mode 1.
  SCON0_bit.REN = 1;      //  Enable receives.
  SMD0_bit.SMOD = 1;      //  Set baud rate to 16 times the baud clock.
  PR0 = 0x3AFB;           //  Set phase for 115200 with a 16MHz crystal.
  SCON0_bit.TI = 0;       //  Clear the transmit flag.
  SBUF0 = 0x0D;           //  Send carriage return to start communication.
}
```

对于MAXQ架构，中断必须分三级开放：全局、每个模块内部、局部。

正如SPI通信子程序，一个寄存器用来发送并接收串行数据。对SBUF0寄存器的写操作将启动一次传输。当数据在串口上有效时，读取SBUF0寄存器将取回输入数据。该实例程序中使用了以下函数向串口输出数据。

```
int putchar(int ch)
{
  while(SCON0_bit.TI == 0); // Wait until we can send.
  SCON0_bit.TI = 0;         // Clear the sent flag.
  SBUF0 = ch;               // Send the char.
  return ch;
}
```

使用定时器产生周期性中断

该应用实例中用到的最后一部分是16位定时器之一。该定时器用来产生中断，触发每秒二次的温度读数。在该实例中配置计数器时，编程者必须设定重装值，指定时钟信号源，并启动定时器。以下代码给出了初始化定时器0所需的步骤。

```
T2V0 = 0x00000;         //  Set current timer value.
T2R0 = 0x00BDC;         //  Set reload value.
T2CFG0_bit.T2DIV = 7;   //  Set div 128 mode.
T2CNA0_bit.TR2 = 1;     //  Start the timer.
```

在该实例中将该定时器用作中断源，还需要一些更多的步骤。对于MAXQ架构，中断必须分三级开放：全局、每个模块内部、局部。使用IAR的编译器，通过调用__enable_interrupt()函数开放全局中断。这就有效地将中断与控制(IC)寄存器中的中断全局使能(IGE)位置位。由于定时器0位于模块3中，将中断屏蔽寄存器(IMR)中的位3置位，开放该模块的中断。将定时器/计数器2控制寄存器A (T2CNA)中的定时器中断使能(ET2)位置位，开放局部中断。以下给出了该应用实例中执行的这些步骤。

```
__enable_interrupt()
T2CNA0_bit.ET2 = 1;     //  Enable interrupts.
IMR |= 0x08;            //  Enable the interrupts for module 3.
```

最后，使用中断还需要初始化中断向量。IAR的编译器允许每个模块使用不同的中断处理函数。为特定模块设置中断处理程序，需要使用#pragma向量指示器。中断处理函数还应当首先通过__interrupt关键字声明。该应用实例以如下方式为模块3声明中断处理程序。

在了解了一些外设寄存器的细节后，编程者可以简单地为MAXQ处理器开发应用程序。

```
#pragma vector = 3
__interrupt void timerInterrupt()
{
  // Add interrupt handler here.
}
```

结论

正如这些代码实例说明的，编程者在了解了一些外设寄存器的详细内容后，可以很容易地为MAXQ2000处理器及MAXQ产品线中的其它处理器开发应用程序。由于允许用ANSI兼容的C语言编写代码，IAR的Embedded Workbench的加入加速了开发过程。

该应用实例完整的源代码可以从www.maxim-ic.com.cn/MAXQ_code下载。必要的布线与准备的详细内容，请阅读代码开头的说明与注释。更多关于IAR Embedded Workbench使用的详细内容，请参考本出版物的第二篇文章，标题为“在MAXQ环境中编程”。