Basic Express BX-24
Application Note

Contador de pulsos con Interrupciones de Hardware

Introducción

Esta nota de aplicación ilustra cómo utilizar las interrupciones por hardware de BasicX para contar los pulsos. La llamada al sistema ***WaitForInterrupt*** se utiliza para obtener acceso al pin INT1 del procesador BX-24. En el momento en el que se ordena una tarea a *WaitForInterrupt,* se bloquea la tarea hasta que tenga lugar la interrupción, lo cual implica que se consumen muy pocos recursos del CPU, comparado con otros sistemas como bucles de polling.

Llamada del sistema WaitForInterrupt

WaitForInterrupt permite a los programas BasicX responder de manera inmediata a las interrupciones del hardware en un pin I/O. En el siguiente programa de ejemplo, la tarea PulseCountTask ejecuta un bucle continuo llamado WaitForInterrupt. La tarea bloquea la llamada hasta que se produzca un flanco de subida en el pin INT1, que equivale al pin 11 en un controlador BX-24.

Cuando se produce una interrupción, el número de pulsos PulseCount aumenta, y el indicador LED del BX-24 se ilumina para proporcionar un efecto visual.

Otra tarea dentro del programa principal convierte el número de pulsos en una cadena y la transmite a través del puerto serie Com1. El número de pulsos se transmite una vez por segundo. El código de ejemplo (un programa similar está incluido en el fichero ***CountingPulses.bas***, que acompaña a este documento):

```
Private Const StackSize As Integer = 25
Private PulseCountStack(1 To StackSize) As Byte
Private PulseCount As Integer
'-----------------------------------------------------
Public Sub Main()

    Call InitializeBX24

    PulseCount = 0

    CallTask "PulseCountTask", PulseCountStack

    Do
        Debug.Print CStr(PulseCount)
        Call Delay(1.0)
    Loop

End Sub
```

```
Private Sub PulseCountTask()

    Const LEDpin As Byte = 25   ' Indicador LED Rojo
    Const LEDon  As Byte = 0
    Const LEDoff As Byte = 1
    Do
        Call WaitForInterrupt(bxPinRisingEdge)
        PulseCount = PulseCount + 1
        Call PutPin(LEDpin, LEDon)
        ' Debounce.
        Call Sleep(0.1)
        Call PutPin(LEDpin, LEDoff)
    Loop
End Sub
'-------------------------------------------------------
Private Sub InitializeBX24()
' BX-24 sólo -- configure INT0 como input-pullup para
' evitar que provoque una interrupción indeseada.
' INT0 es el pin 11(interno)del chip 8535, y no está
' conectado a ningún pin externo.
    Register.DDRD  = Register.DDRD And bx1111_1011
    Register.PORTD = Register.PORTD Or bx0000_0100

End Sub
'-------------------------------------------------------
```

Aunque la tarea principal transmite el número de pulsos a un ritmo pausado, la tarea PulseCountTask, que se ejecuta en un segundo plano, es capaz de responder rápidamente a un flanco de subida en el pin de interrupción.

Dependiendo de la frecuencia con la que se produzcan los pulsos, la tarea PulseCountTask puede estar la mayor parte del tiempo en modo WaitForInterrupt sin consumir el tiempo que tardan tareas de la CPU alternativas como bucles de polling. Esto normalmente amplía el tiempo disponible de la CPU para otras tareas, a menos que la tasa de pulsos sea tan elevada que la tarea PulseCountTask monopolice la CPU.

Tenga en cuenta que el parámetro de WaitForInterrupt puede configurar 3 tipos diferentes de disparadores de interrupción – flanco de bajada, flaco de subida o nivel lógico bajo. La biblioteca del sistema de BasicX proporciona las variables predefinidas bxPinFallingEdge, bxPinRisingEdge y bxPinLow para estos parámetros.

2.00.A

Traducción Española: Alicia Bernal, revisión: Pablo Pompa www.superrobotica.com