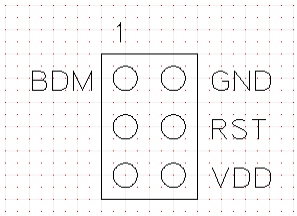
FreesBee

Índice de contenido

Características

El FreesBee es una interfaz de programación y debugging para XBee Programable, basada en la herramienta open source USBDM (http://sourceforge.net/projects/usbdm/).
La implementación particular contempla sólo lo relacionado para esta aplicación, otros usos no están soportados. El circuito esquemático del mismo figura en la última hoja de este documento, puede consultarlo para mayor información.
La alimentación se toma del conector USB y del conector de programación y debugging (BDM, Background Debug Module), en el cual, a su vez, estará conectado a la alimentación del XBee. El pinout es el correspondiente a una conexión directa a pines en la placa de desarrollo del usuario, o la XBoard, siguiendo la conexión standard para BDM:



Tensión de operación: USB
Tensión en el pin VDD: la alimentación provista al módulo XBee

Las conexiones no poseen mayores protecciones que las elementales

Jumpers

No existen jumpers configurables por el usuario

Instalación

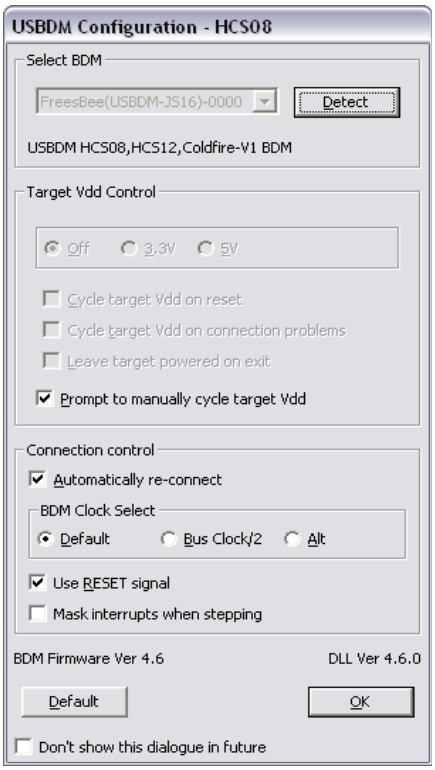
1. Descomprima los archivos contenidos en el archivo de software FreesBee_User.zip. Recuerde la localización.
2. Conecte el FreesBee a un puerto USB de su computadora donde correrá el entorno de desarrollo.
3. Al aparecer el diálogo de nuevo hardware, diríjalo a que encuentre los drivers correspondientes en el subdirectorio User\Win32\USB_Driver, donde descomprimió el archivo en el punto 1. Esto depende de su sistema operativo, consulte la documentación correspondiente.
4. Si no lo hizo anteriormente, instale el entorno de desarrollo. El único soportado es **CodeWarrior Development Studio for Microcontrollers V6.3**
5. Ejecute el archivo User\Win32\Install Codewarrior Files.cmd, esto copiará los archivos necesarios para el funcionamiento del FreesBee. Atención, si usted utilizaba otra herramienta basada en OSBDM o USBDM, esto

reescribirá los archivos.

6. Ejecute el archivo User\Win32\Install Codewarrior Wizards.cmd.

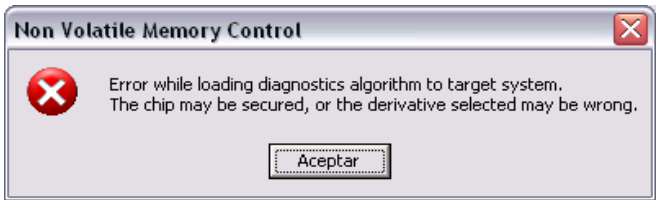
Utilización

Conéctelo a la XBoard (o a un conector BDM).
No conecte o desconecte el FreesBee del XBee mientras éste está alimentado.
Dentro de CodeWarrior IDE, seleccione la herramienta como HCS08 Open Source BDM. La configuración es la siguiente:

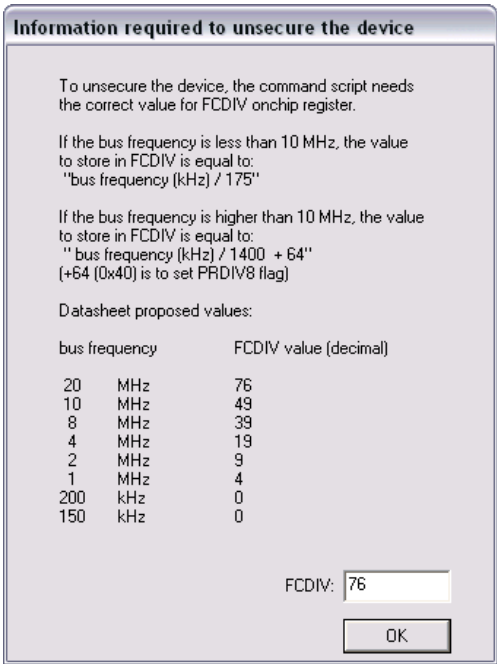


Unsecure

Los módulos vienen de fábrica con un bootloader cargado, el cual no permite debugging.



Para poder acceder a estas funciones es necesario grabar otro bootloader o un programa del usuario. Para lograr esto, debemos primero borar la memoria flash del micro. Esto requiere conocer un parámetro de funcionamiento del bootloader: la velocidad del bus configurada para el microcontrolador. Afortunadamente, observando el código fuente sabemos que ésta es cercana a los 20MHz, con lo cual configuraremos el valor 76 en FDIV al realizar el procedimiento de ***Unsecure***, dentro del menú de *HCS08 Open Source BDM*:



Clock Trim

Para poder depurar un programa en el microcontrolador, primero hay que borrar la memoria flash. Esto borra el valor de calibración de la referencia del oscilador interno del microcontrolador, conocido por su nombre en inglés: clock trim. Esto corresponde a un valor de 9-bits que se almacena en dos posiciones de memoria flash, siendo manualmente recuperado y escrito en los registros de control del micro por el código de arranque del usuario, y/o el bootloader.
El código de bootloader provisto por Digi espera un valor grabado en estas posiciones, con lo cual al encontrarse con 0xFFFF inicializa incorrectamente el oscilador interno y genera una referencia muy errada, que imposibilita la comunicación serie.

Alternativas

Disponemos de una serie de alternativas para resolver este inconveniente

Valor por defecto (mitad de rango)

Es posible modificar el bootloader para que observe el valor existente en la memoria y ante su ausencia utilice valores por defecto. La modificación siguiente en *main.c* realiza esta función, introduciendo un error que en la mayoría de los casos permite operar normalmente:

en las líneas 136 y 137, donde dice:

```
ICSTRM = NVICSTRM; //these are defaults on startup, and now for watchdogs and other errors
ICSSC = MULTIPLY_BY_1216 | NVFTRIM;
```

se modifica a :

```
ICSSC = MULTIPLY_BY_1216;
if(NVICSTRM != 0xFF){
       ICSTRM = NVICSTRM;
       ICSSC |= NVFTRIM_FTRIM;
}
```

Valor fijo en código

Otra alternativa, una vez que sabemos cuál es el valor correcto, es incluirlo en el código del bootloader. Por ejemplo, el código siguiente define el valor 0xA7:

```
const unsigned char NVICSTRM_INIT @0x0000FFAF = 0xA7;      // hard code ICSTRIM
const unsigned char NVFTRIM_INIT @0x0000FFAE = 0x00;       // hard code FTRIM
```

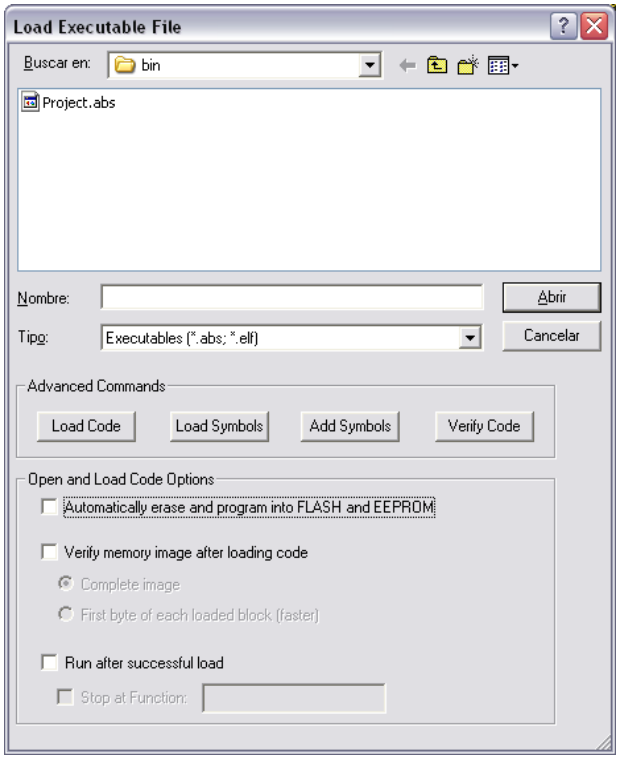
El valor lo podemos determinar siguiendo el procedimiento que se describe a continuación o mediante el programa HCS08_FlashProgrammer.

Regeneración de la referencia "en caliente"

Mediante un simple programa, modificado de una nota de aplicación, es posible tomar una señal de frecuencia conocida y en base a ella ajustar la referencia del oscilador interno del microcontrolador. El ejemplo que se provee mide el espacio entre caracteres '@' transmitidos por la PC a 9600 bps, y ajusta el clock del micro de manera acorde. La precisión empleada es de 8-bits.

Procedimiento

- En la XBoard, unir TD con DSR mediante una R de 33ohms
- Observar el pin CTS con osciloscopio
- Cargar el programa deseado en el micro con el debugger. En vez de ejecutar, seleccionar ***Load..***. en el menú de *HCS08 Open Source BDM*
- Ubicar y seleccionar el archivo *Program.abs* que se incluye con los archivos de usuario del FreesBee, en el directorio *XBee\ClockTrim\bin*. Este programa se carga en RAM y deberemos ejecutarlo allí. **No cargar todavía**.
- Configurar para que no borre la memoria y que no arranque luego de cargar, y cargar el programa:



- En la ventana de comandos ingresar `GO 0x80` y presionar <ENTER>, el programa de ajuste estará corriendo en RAM, esperando ver actividad en DSR
- Abrir un programa terminal, X-CTU es ideal si utilizamos su solapa Terminal. Configurar para 9600 8N1 sin control de flujo
- Presionar la tecla '@' (mantenerla presionada), en unos segundos debería verse actividad en el osciloscopio.
  - Si se ve una señal de 1,6MHz, hemos realizado la calibración, podemos leer el valor en la posición 0xFFAF (NVICSTRM)
  - Si se ve una señal de alrededor de 150KHz (aproximadamente) algo falló (memoria protegida, memoria no borrada, fallo de grabación)
  - Cualquier otra cosa es un problema y debe verse sobre el código a ver qué está ocurriendo

Modificación del bootloader

El bootloader por defecto protege su zona de memoria, esto hace que no se pueda grabar el valor de clock trim una vez que el bootloader se cargó en la flash. Para poder grabarlo, deberemos modificar la línea 71 de *main.c*, de modo de no escribir en el registro de protección:

```
// const unsigned char NVPROT_INIT @0x0000FFBD = BOOTLOADER_START-2;//this redirects the interupt
vectors and locks the bootloader
```

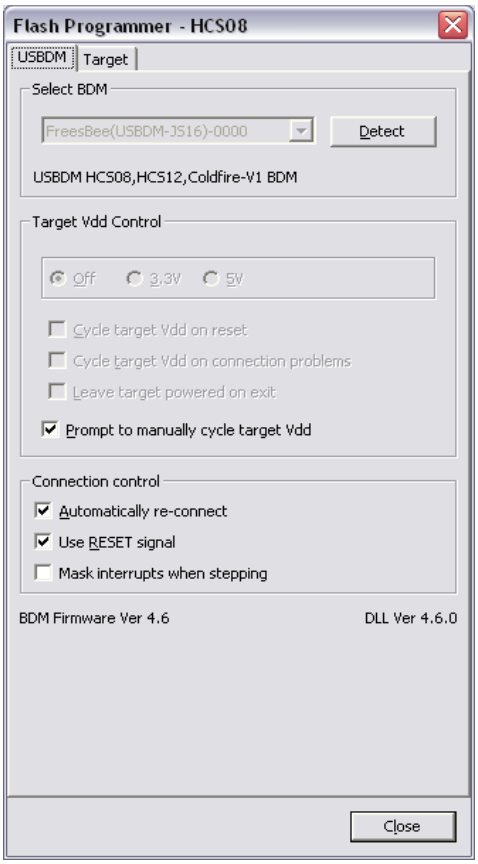
El código provisto incluye un bootloader ya modificado para esto; se encuentra en el directorio *XBee\Bootloader\bin*

Regeneración de la referencia "en frío"

Utilizaremos el programa HCS08_FlashProgrammer para grabar una imagen de flash y regenerar la calibración

HCS08_FlashProgrammer

Para grabar una imágen de memoria en el micro (archivo .s19), por ejemplo el mismo bootloader y dejar un módulo casi como nuevo, disponemos de un programa denominado HCS08_FlashProgrammer. Este programa se encuentra en el directorio Win32\Utilities y requiere de algunas libraries de dicho directorio para funcionar. Con él podemos además regenerar la calibración del oscilador interno (clock trim) al realizar la grabación y recuperar el acceso a un módulo en que el micro queda en modo STOP.
La siguiente es la configuración sugerida:

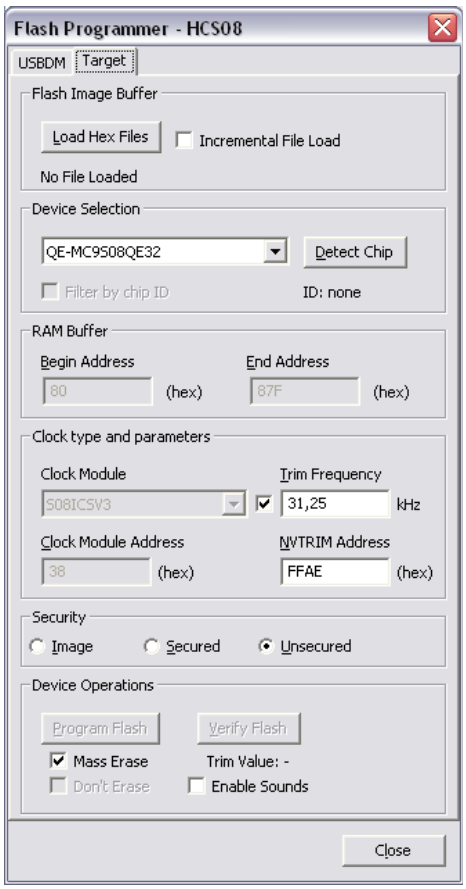


Grabación de la flash y clock trim

Ambos procedimientos se realizan de forma conjunta, dado que el borrado de memoria anterior a la grabación elimina el valor de configuración y debe regenerarse.

El código provisto incluye una imagen del bootloader modificado que puede grabarse con este programa; se encuentra en el directorio *XBee\Bootloader\bin*

La siguiente es la configuración sugerida:



Stop mode

Cuando el procesador ejecuta una instrucción STOP, el BDM no puede volver a ingresar. Si el código que grabamos en un módulo hace que el procesador quede en modo STOP y no salga de él (no hay fuentes de interrupción previstas), el procedimiento standard de conexión fallará y no tendremos acceso al micro desde el debugger. Para recuperar el acceso simplemente debemos presionar el botón “Detect Chip” que se observa en la imagen del párrafo anterior; el programa intentará conectarse y al no poder hacerlo nos instruirá a desconectar y reconectar el hardware, para reiniciarlo con el pin de RESET en estado bajo, de modo de poder ganar acceso al BDM. Una vez detectado, ya podemos salir del programa y acceder al chip con el debugger, o grabarle directamente una nueva imagen.

Circuito esquemático