Seminario de Electrónica: Sistemas Embebidos - Trabajo Práctico N° 1

LPC43xx Entradas y Salidas (Digitales) de Propósito General (GPIO)

Objetivo:

- **Uso del IDE** (edición, compilación y depuración de programas)
- **Uso de GPIO** (manejo de Salidas y de Entradas Digitales)
- **Documentar lo que se solicita en c/ítems**

Referencias (descargar a fin de usarlas durante la realización del TP)**:**

- **LPCXpresso-Intro: http://campus.fi.uba.ar/pluginfile.php/155949/mod_resource/content/4/Sistemas_Embebidos-2016_2doC-LPCXpresso-Intro-Cruz.pdf**
- **LPCXpresso-Salidas: http://campus.fi.uba.ar/pluginfile.php/156011/mod_resource/content/4/Sistemas_Embebidos-2016_2doC-LPCXpresso-Salidas-Cruz.pdf**
- **LPCXpresso-Entradas: http://campus.fi.uba.ar/pluginfile.php/156013/mod_resource/content/4/Sistemas_Embebidos-2016_2doC-LPCXpresso-Entradas-Cruz.pdf**
- **LPCXpresso-Systick: http://campus.fi.uba.ar/pluginfile.php/156031/mod_resource/content/5/Sistemas_Embebidos-2016_2doC-LPCXpresso-Systick-Cruz.pdf**
- **LPC435X_3X_2X_1X Product Data Sheet: http://campus.fi.uba.ar/mod/resource/view.php?id=28519**
- **LPC43XX User Manual** (Chapter 1, 18 & 19)**: http://campus.fi.uba.ar/mod/resource/view.php?id=77765**
- **EDU-CIAA-NXP (web site): http://proyecto-ciaa.com.ar/devwiki/doku.php?id=desarrollo:edu-ciaa:edu-ciaa-nxp**
- **EDU-CIAA-NXP (esquemático): http://www.proyecto-ciaa.com.ar/devwiki/lib/exe/fetch.php?media=desarrollo:edu-ciaa:edu-ciaa-nxp:edu-ciaa-nxp_color.pdf**
- **EDU-CIAA-NXP (pinout): http://proyecto-ciaa.com.ar/devwiki/lib/exe/fetch.php?media=desarrollo:edu-ciaa:edu-ciaa-nxp_pinout_a4_v4r2_es.pdf**

1. Uso del **IDE** (Integrated Development Environment) **MCUXpresso** (p/Linux) **o LPCXpresso** (p/Windows)
   a. Instale **OpenOCD 0.10.0** desde el siguiente sitio (seleccione el instalador según su Sistema Operativo): https://github.com/gnuarmeclipse/openocd/releases/tag/gae-0.10.0-20160110
   b. Instale **Git** desde el siguiente sitio (seleccione el instalador según su Sistema Operativo, usaremos Git por línea de comando): https://git-scm.com/ (https://git-scm.com/book/en/v2/Getting-Started-Installing-Git)
   c. Registrarse, Descargar, Instalar, Ejecutar y Licenciar **MCUXpresso IDE v10.1.1** (o posterior) o **LPCXpresso IDE v8.2.0** (o posterior) https://www.nxp.com/support/developer-resources/software-development-tools/mcuxpresso-software-and-tools/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE?tab=Design_Tools_Tab
      i. Dentro de MCUXpresso o LPCXpresso, agregue el plug-in **OpenOCD Debugging**
      Menú **Help → Install New Software …** Work with: http://gnuarmeclipse.sourceforge.net/updates
      Seleccione el plug-in y luego siga las instrucciones del asistente (**GNU ARM C/C++ OpenOCD Debugging**)
      Instalar VCP driver del chip **U6** FT2232-H (conversor USB-Serie) **DEBUG** y configurarlo como se explica en:
      http://proyecto-ciaa.com.ar/devwiki/lib/exe/fetch.php?media=docu:fw:bm:ide:installciaa_ide_windows_v1.0.pdf o
      http://proyecto-ciaa.com.ar/devwiki/lib/exe/fetch.php?media=docu:fw:bm:ide:installciaa_ide_linux_v1.0.pdf
      ii. Antes de ejecutar conectar la placa **EDU-CIAA-NXP** a su PC
         1. Usando la línea de comandos, clone el repositorio de trabajo y cree una copia del archivo **project.mk.template** llamada **project.mk**:
            git clone **https://github.com/ciaa/firmware_v2.git**
            cd **firmware_v2**
            git status -s
            git checkout **master**
            cp **project.mk.template project.mk**
            En el archivo **project.mk** podrá configurar el **proyecto**, el **procesador** y la **placa** a utilizar, por ejemplo:
            PROJECT = **sapi_examples/edu-ciaa-nxp/bare_metal/gpio/gpio_02_blinky**
            TARGET = **lpc4337_m4**
            BOARD = **edu_ciaa_nxp**
         2. Seleccionar como nombre de Workspace: **workspace-SE-2018- TPs**
         3. MCUXpresso ha abierto su vista **Develop**, en ella editaremos y compilaremos fuentes C (.h & .c), procedamos a abrir firmware_v2 del proyecto CIAA.
            Mediante **Fiel -> New -> Other -> C/C++ -> Makefile Project with Existing Code …**
            a. En **Existing Code Location Browse…** busque la carpeta **firmware_v2** (clonada en 1). Si no eligió una ubicación en particular, por defecto debería estar en directorio del usuario (/home/usuario).
            b. **Destilde** la opción **C++**
            c. **Seleccione** la opción **NXP MCU Tools**
         4. Haga clic en **firmware_v2** y a continuación en **Build firmware_v2 [Debug]**. Debería observar el proceso de compilación que finalizará con una salida **Console** similar a la siguiente (el directorio de salida varía según el microcontrolador elegido):

```
*** linking project gpio_02_blinky ***
   text    data          bss     dec     hex    filename
   8252          8       36     8296    2068
out/lpc4337_m4/gpio_02_blinky.axf
copy from `out/lpc4337_m4/gpio_02_blinky.axf' [elf32-littlearm] to
`out/lpc4337_m4/gpio_02_blinky.bin' [binary]
*** post-build ***
make[1]: Leaving directory '<USER_PATH>/firmware_v2'

Build complete
```

5. Configuración de **Debug**:
   a. Clic derecho en workspace → Debug As → **Debug Configurations…**
   b. Doble clic en **GDB OpenOCD Debugging**
   c. Clic en **Search Project…** seleccione el archivo **out/lpc4337_m4/gpio_02_blinky.axf** (si no aparece automáticamente)
      Otra opción es hacer **clic** en Browser y seleccionar **firmware_v2**, automaticamente debe seleccionar **out/lpc4337_m4/gpio_02_blinky.axf** como C/C++ Application
   d. Pestaña **Debugger**, sección OpenOCD Setup, haga clic en **Browse…** para navegar a la carpeta de instalación de OpenOCD, luego a la carpeta bin y finalmente al ejecutable **openocd.exe**
   e. **Config options** ingrese el siguiente texto: **-f etc/openocd/lpc4337.cfg**
   f. Sección **GDB Client Setup**, en el campo **Executable** escriba: **arm-none-eabi-gdb**
   g. A partir de este punto es necesario tener la **EDU-CIAA-NXP** conectada a la PC a través de la interfaz **Debug**
   h. Clic en **Apply** , luego en **Debug**, debería comenzar la sesión de Debug con un **breakpoint** en la primer línea de la función **main()**
   i. MCUXpresso ha abierto su vista **Debug**, allí ejecutaremos **gpio_02_blinky** (ejemplo de aplicación)
6. **Documentar** mediante tablas c/texto e imágenes la estructura de **archivos**, su tipo/contenido (especialmente **readme.txt**) de c/proyecto **sapi_examples/edu-ciaa-nxp/bare_metal/gpio/gpio_02_blinky**
7. **Documentar** mediante tablas c/texto e imágenes la **secuencia de comandos**: Clean **firmware_v2** -> Build **firmware_v2** -> Debug **firmware_v2** -> Ejecutar **gpio_02_blinky** (ejemplo de aplicación)
   a. Completo (**Resume**), detener (**Suspend**) y resetear (**Restart**)
   b. Por etapas colocando **breakpoints** (Resume)
   c. Por línea de código (**Step Into**, **Step Over**, **Step Return**)
   d. Recuerde siempre abandonar Debug (**Terminate**) antes de Editar o Compilar algún archivo, o Abandonar el IDE (**Exit**)
8. Migrar el proyecto **sapi_examples/edu-ciaa-nxp/bare_metal/gpio/gpio_02_blinky** (parpadeo del **LEDs** c/**sAPI**) a: **projects/TP1**
   a. **Documentar** los pasos a seguir para concretar una **migración exitosa**
   b. Identificar funciones de librería **sAPI** útiles para el **parpadeo de un led**
      i. **Documentar** mediante tablas c/texto e imágenes la secuencia de **funciones** invocadas durante la ejecución del ejemplo de aplicación, en qué archivo se encuentran, su descripción detallada, qué efecto tiene la aplicación sobre el hardware (identificar circuitos, puertos, pines, niveles, etc.) así como la interacción entre las mismas (tanto en **ResetISR()** como en **main()**)
      ii. **Idem c** pero con **datos** (definiciones, constantes, variables, estructuras, etc.) (tanto en **ResetISR()** como en **main()**)
9. Ingresar a https://github.com/
   a. **Crear** una **cuenta** si no dispone de una (informe su nombre de usuario al docente para ser agregado al *team* correspondiente a su grupo)
   b. **Crear** un **repositorio** denominado **TP1**. La URL del mismo será parecida a https://user...name/TP1
   c. **Realizar** un **commit/push** inicial del código actual. Usando la línea de comandos:
      cd path/a/**firmware_v2/projects/TP1**
      git init
      git remote add origin [url del repositorio]
      git status #muestra el estado actual del repositorio
      git add - - all
      git status
      git commit –m ”commit inicial”
      git push -u origin master

   De ahora en adelante, **actualizar** **su repositorio** mediante **commit**/**push**

2. Migrar el proyecto **sapi_examples/edu-ciaa-nxp/bare_metal/gpio/gpio_01_switches_leds** (sensado de **Push Buttons** c/**sAPI**) a: **projects/TP1**
    a. **Documentar** los pasos a seguir para mantener en el archivo **TP1.c** los fuentes del TP1-1 y **TP1-2** (**compilación condicional**)
    b. Identificar funciones de librería **sAPI** útiles para el **sensado de un pulsador**
        i. **Documentar** mediante tablas c/texto e imágenes la secuencia de **funciones** invocadas durante la ejecución del ejemplo de aplicación, en qué archivo se encuentran, su descripción detallada, qué efecto tiene la aplicación sobre el hardware (identificar circuitos, puertos, pines, niveles, etc.) así como la interacción entre las mismas (tanto en **ResetISR()** como en **main()**)
        ii. **Idem c** pero con **datos** (definiciones, constantes, variables, estructuras, etc.) (tanto en **ResetISR()** como en **main()**)

3. Migrar el proyecto **sapi_examples/edu-ciaa-nxp/bare_metal/gpio/tick_01_tickHook** (uso de **tickHooks** c/**sAPI**) a: **projects/TP1**
    a. Mediante **compilación condicional**, mantener en el archivo **TP1.c** los fuentes del TP1-1, TP1-2 y **TP1-3**
    b. Identificar funciones de librería **sAPI** útiles para el **colgarse de un tick**
        i. **Documentar** mediante tablas c/texto e imágenes la secuencia de **funciones** invocadas durante la ejecución del ejemplo de aplicación, en qué archivo se encuentran, su descripción detallada, qué efecto tiene la aplicación sobre el hardware (identificar circuitos, puertos, pines, niveles, etc.) así como la interacción entre las mismas (tanto en **ResetISR()** como en **main()**)
        ii. **Idem c** pero con **datos** (definiciones, constantes, variables, estructuras, etc.) (tanto en **ResetISR()** como en **main()**)

4. Hacer portable el uso de **tickHooks** & **LEDs** c/**sAPI**
    a. Mediante **compilación condicional**, mantener en el archivo **TP1.c** los fuentes del TP1-1, TP1-2, TP1-3 y **TP1-4**
    b. **Documentar** la modificación la configuración del tickHook en función de la constante **TICKRATE_MS** (1mS/10mS/100mS)
    c. **Documentar** la modificación el parpadeo del led en función de la constante **LED_TOGGLE_MS** (100mS/500mS/1000mS)
    d. **Modificar** ejemplo de aplicación para soportar todos los LEDs (encender/apagar -**500mS/500mS**- uno a la vez en secuencia)

5. Agregar al ejemplo anterior (**4.d**) envío de mensajes de depuración por puerto serie c/**sAPI**
    a. Mediante **compilación condicional**, mantener en el archivo **TP1.c** los fuentes del TP1-1, TP1-2, TP1-3, TP1-4 y **TP1-5**
    b. Para usar **UART2** como interfaz de **DEBUG**, instalar y ejecutar en su PC un programa de **terminal** comunicación serie, p/e Putty y el driver del chip **U6 FT2232-H** (conversor USB-Serie) **DEBUG**:
        i. Los pasos son: **Habilitar** - **Configurar** - **Invocar** la funcionalidad

```
/* The DEBUG* functions are sAPI debug print functions.
   Code that uses the DEBUG* functions will have their I/O routed to
   the sAPI DEBUG UART. */
DEBUG_PRINT_ENABLE;

/* UART for debug messages. */
debugPrintConfigUart( UART_USB, 115200 );

debugPrintString( "DEBUG c/sAPI\r\n" );
debugPrintString( "LED Toggle\n" );
```

    c. Identificar funciones de librería **sAPI** útiles para esta nueva funcionalidad
        i. **Documentar** mediante tablas c/texto e imágenes la secuencia de **funciones** invocadas durante la ejecución del ejemplo de aplicación, en qué archivo se encuentran, su descripción detallada, qué efecto tiene la aplicación sobre el hardware (identificar circuitos, puertos, pines, niveles, etc.) así como la interacción entre las mismas (tanto en **ResetISR()** como en **main()**)
        ii. **Idem c** pero con **datos** (definiciones, constantes, variables, estructuras, etc.) (tanto en **ResetISR()** como en **main()**)

6. Agregar al ejemplo anterior (**5.a**) sensado de **Push Buttons** c/**sAPI**
    a. Mediante **compilación condicional**, mantener en el archivo **TP1.c** los fuentes del TP1-1, TP1-2, TP1-3, TP1-4, TP1-5 y **TP1-6**
    b. Identificar funciones de librería **sAPI** útiles para el **sensado de un pulsador**
    c. **Modificar** ejemplo de aplicación para soportar todos los LEDs (apagar/encender uno a la vez en secuencia al oprimir un pulsador)
    d. En caso que no funcione correctamente el ejemplo de aplicación **documentar** la forma de la señal digital “pulsador” a sensar: no oprimido // transición a oprimido // mantener oprimido // transición a no oprimido) // …