Laboratorio 9 – Mass Storage

**Ajustando el ambiente de desarrollo**

**Pre-requisitos**

* Debe de tener instalado MDK ARM
* Debe tener instalado el software pack de STM32F4xx
* Keil::MDK Middleware
* ARM::CMSIS
* Keil::ARM\_Compiler
* Se debe usar la licencia profesional de MDK ARM

**Conecte el STM32F419I-Discovery Board a la PC:**

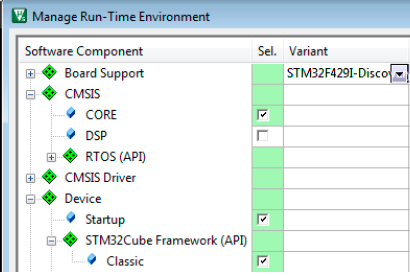
1. Use el cable USB-mini para conectar a su computador la STM32F4-Discovery usando la conexión a USB ST-Link.

**Paso 1: Creación del Proyecto**

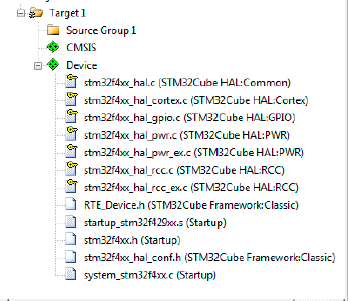
***Crear un Nuevo Proyecto con la tarjeta de Evaluación***

**Crear un Nuevo Proyecto con los archivos de inicialización del módulo main:**

1. En la terminal de Keil seleccione Project->New uVision Project. Se abrirá la ventana de nuevo proyecto.
2. Crear un folder y nombrarlo para su proyecto. Por ejemplo en ..\LabUSB y con un nombre de proyecto apropiado. Si por ejemplo lo guarda como USB, será USB.uvprojx.
3. Seleccionar el dispositivo. STM32F429ZITx
4. Hacer click en OK y la ventana de Manage Run-Time Environment se abrirá.
5. Exapnda y seleccione CMSIS:Core y Device:Startup



1. La mayoría de los dispositivos tienen abstraccione de hardware listadas aquí. También debe seleccionar el STM32Cube Framework (API):Clasic
2. Si la columna Sel se presenta en naranja, ir a la sección inferior izquierda y hacer click en Resolve.
3. Click en OK para cerrar la ventana y observe los archivos que creó uVision



Añadir el archivo main.c

1. Presione botón derecho sobre “Source Group 1” y seleccione el añadir Nuevo item al grupo “Source Group 1”.
2. En la ventana que se ha abirrto seleccione **User Code Template**. Seleccione **'main' module for STM32Cube**. Y este inicializará y configurará el sistema y reloj.
3. Seleccione **Add**.

**Ajustar la velocidad del CPU**

El cristal externo de oscilador está a una frecuencia de 8MHz.

1. Seleccione Target Options o ALT-F7 y seleccione la pestaña **Target**.
2. Ingrese **8 MHz** en la caja **Xtal (MHz)**

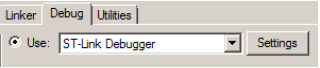


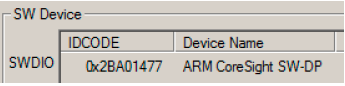
1. Seleccione la pestaña de C/C++.
2. Ingrese el **HSE\_VALUE=8000000** en la ventana de **Define**. Este valor representa el cristal externo y ajustará el valor del CPU a 168 MHz en system\_stmf4xx.c
3. Hacer click en OK y cierre esta ventana.
4. Seleccione **File** **Save All**
5. Compile el Proyecto y sus archivos. Deberá no haber errores o advertencies in la ventana de salida. Si los consigue por favor elimine los errores y configure el target para quemar el código.

**Configurar el adaptador de depuración**

**Seleccione el adaptador de depuración ST-Link V2:**

1. Seleccione las opciones de destino  o ALT-F7. Seleccione la pestaña de depuración .
2. En el cuadro de Uso seleccione “ST-Link depurador”.
3. Haga clic en **Settings**. En el cuadro **Port**, seleccione **SW** (para Serial-Wire Debug SWD).
4. En el cuadro SWDIO debe ver un IDCODE y **ARM CoreSight** **SW-DP**  válidos. Esto indica que μVision está conectado a la depuración de STM32.





.

**Si ve un error o nada en el cuadro SWDIO, debe solucionar esto antes de**

**Puede continuar. Asegúrese de que la placa esté conectada.**

**Configurar el Visor de cables de serie (SWV):**

1. Seleccione la pestaña **Trace**. En el cuadro **Core Clock**, ingrese 168 MHz y seleccione **Trace Enable**. Esto ajusta la velocidad de la señal SWV UART y el temporizador del depurador muestra. Deseleccionar **EXCTRC** (Excepción Tracing). Deje todas las demás configuraciones en sus valores predeterminados.



Nota: Soldadura Puente SB9 debe ser superada para SWV funcione. Para nuestro caso no es tan importante.

**Seleccione el algoritmo de programación Flash:**

1. Seleccione la pestaña **Flash Download**.
2. Confirmar la programación STM32F4xx Flash es de 2 MB

El algoritmo se selecciona como se muestra aquí:

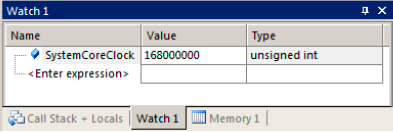


Si no, haga clic en Agregar para elegirlo.

1. Haga clic en Aceptar dos veces para volver al menú principal.
2. Compile los archivos fuente del proyecto: 
3. Programe la Flash y e inicie la depuración haciendo clic en el icono para entrar en el modo de depuración de μVision: 
4. Haga clic en el icono RUN
5. El programa se está ejecutando ahora. Nota: es posible detener el programa con el icono de detención.

**Insertar una variable global en la ventana Inspección:**

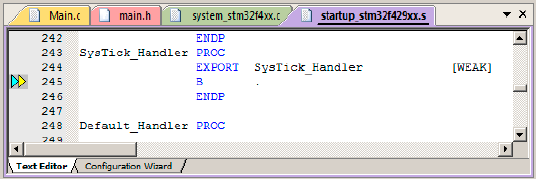
1. En la pestaña Project en **Device**, haga doble clic en **system\_stm32f4xx.c** para abrirlo.
2. Encuentra la variable **SystemCoreClock**. Se declara cerca de la línea 138.
3. Haga clic con el botón derecho en él y seleccione Agregar **SystemCoreClock** a… y seleccione **Watch 1**. **Watch 1** se abrirá automáticamente si no está abierto y mostrará esta variable.



1. En la ventana Watch 1, haga clic derecho en SystemCoreClock en la columna Name y deseleccione hexadecimal.
2. SystemCoreClock Ahora se mostrará con el correcto valor de frecuencia de 168 MHz.

Nota: Puede añadir variables a la ventana watch mientras su programa se está ejecutando.

1. Detener el programa . El contador de programa (R15) será detenido en **B** en el **SysTick\_Handler**. La instrucción B es un salto a ella misma y esto se puede evitar que se detenga en este punto cuando se agregue código adicional del usuario. Como nosotros vamos a utilizar CMSIS-RTX RTOS, esto no se requiere aquí
2. La flecha amarilla es el programa contador (PC).
3. Salga del modo de depuración .



Hasta este punto lo que tenemos:

Hemos seleccionado el adaptador de depuración, permitido el SWD y el SWV

El programar la flash.

También demostramos cómo mostrar el reloj de la CPU en una ventana de watch.

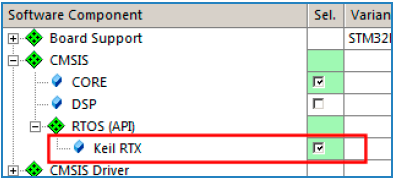
**Paso 2: Añadir CMSIS-RTOS**

***Añada y configure CMSIS-RTOS RTX para una simple aplicación de parpadear un LED.***

**Seleccione y Configure RTX RTOS:**

1. Abra la ventana Manage Run-Time Environment: 

2. Bajo **CMSIS: RTOS (API)** seleccione **Keil RTX**como se muestra aquí:

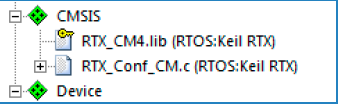


3. Haga clic en Aceptar para cerrar esta ventana.

4. En la ventana Proyecto, observe que se agregan dos nuevos archivos en

Encabezado de CMSIS: **RTX\_CM4.lib** y **RTX\_Conf\_CM.c**

5. Haga doble clic en **RTX\_Conf\_CM.c** para abrirlo.



6. Haga clic en la pestaña **Configuration Wizard** en la parte inferior.

7. Expanda RTX Kernel Timer Tick Timer y cambie RTOS Kernel Timer input clock frequency [Hz] a 168000000 (168 MHz).

**Agregue el archivo de origen Timer.c y agregue la llamada a la función de inicialización del temporizador:**

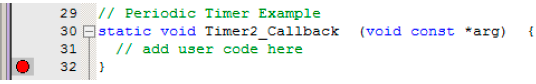
1. En la ventana Proyecto, en Target 1, haga clic con el botón derecho en **Source Group 1** y seleccione Add New Item Group ‘Source Group 1’...
2. En la ventana que se abre, seleccione **User Code Template**. Seleccione **CMSIS-RTOS Timer**.
3. Haga clic en **Add**. Nota Timer.c se agrega al Source Group 1 en la ventana Proyecto.
4. Haga clic en la pestaña main.c para enfocarla y editarla.
5. En main.c cerca de la línea 76, agregue esta línea: extern void Init\_Timers (void);
6. En main.c, cerca de la línea 103, justo después de SystemClock\_Config ();, agregue Init\_Timers (); Init\_Timers crea dos temporizadores: Timer1 (un disparo) y Timer2, que es un temporizador periódico de 1 segundo. Timer2 llama a una función de callback (puntero a función).
7. Seleccione **File → Save All** 
8. Compile los archivos de origen del proyecto haciendo clic en el icono Rebuild . No habrá errores ni advertencias en el Build Output Window. Si hay errores o advertencias, corríjalos antes de continuar.

**Demostrar que el temporizador está funcionando:**

1. Programe el Flash y entre en el modo de depuración : Haga clic en el icono RUN .

CONSEJO: para programar el Flash manualmente, seleccione el icono Load: 

1. El programa se está ejecutando.
2. En Timer.c, cerca de la línea 32, establezca un breakpoint haciendo clic en el cuadro gris. Aparecerá un círculo rojo. El recuadro gris indica ese lenguaje ensamblador. Las instrucciones están presentes y un punto de interrupción de hardware será legal.



1. El programa pronto se detendrá aquí.
2. Haga clic en RUN y en 1 segundo se detendrá aquí nuevamente cuando se active el Timer2.
3. Elimine el breakpoint para el siguiente paso.

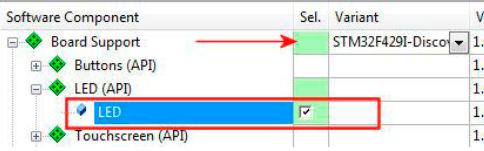
Lo que tenemos en este punto:

Agregamos el RTX RTOS a su proyecto.

Habilitamos un temporizador periódico y demostramos que el programa se está ejecutando.

**Parpadeo del LED:**

1. Salga del modo de depuración. 
2. Abra la ventana Manage Run-Time Environment: 
3. Expanda el Board Support Package (asegúrese de que STM32F429I-Discovery está seleccionado - ver la flecha)



1. Debajo de Board Support: LED (API) seleccionar LED
2. Haga clic en Aceptar para cerrar esta ventana.

En la ventana proyecto, se crea un encabezado llamado Board Support. Este contiene un archivo LED\_F429Discovery.c. Esto configura la E/S de pines utilizados por los LED con una rutina llamada LED\_Initialize. Las funciones LED\_On y LED\_Off se utilizan para controlar el LED.

**Añadir código C a Blink LED LD3:**

En main.c, cerca de la línea 45, agregue #include "Board\_LED.h"

Nota: Puede aparecer un error en esta línea. Por favor ignora esto por ahora. Asegúrese de que las líneas de origen estén escritas exactamente como se muestra para evitar errores. Utilice su mejor criterio en cuanto a dónde se debe agregar el código fuente. Los números de línea pueden cambiar con diferentes versiones de las plantillas de software.

CONSEJO: También puede seleccionar #incluye de una lista:

* Seleccione una línea en un archivo de código fuente y haga clic derecho sobre ella.
* Seleccione Insertar '#include file' . Se abrirá un menú con los #includes proporcionados que puede seleccionar.



1. En main.c, cerca de la línea 104, agregue LED\_Initialize (); justo después de Init\_Timers (); es un buen lugar
2. En Timer.c, cerca de la línea 3, agregue estas dos líneas:

#include "Board\_LED.h"

static int timer\_cnt = 0;

1. En Timer.c dentro de la función Timer2\_Callback cerca de la línea 32, agregue este código en la sección de código de usuario (reemplace la línea // add user code here):

timer\_cnt ++;

if (timer\_cnt & 1)

LED\_On (0);

else

LED\_Off (0);

1. Seleccione File / Save All 
2. Compilar el proyecto: 

No habrá errores ni advertencias en la ventana general de resultados.

1. Programe la Flash y entre en el modo de depuración: 
2. Haga clic en RUN. 
3. El LED PG13 (verde) ahora parpadeará de acuerdo con el temporizador que ha creado.
4. Deje el programa en ejecución para los siguientes pasos.

CONSEJO: En la llamada a la función LED\_On: (0) es el LED verde. Utilizando (1) parpadeará el LED rojo.

Lo que tenemos en este punto:

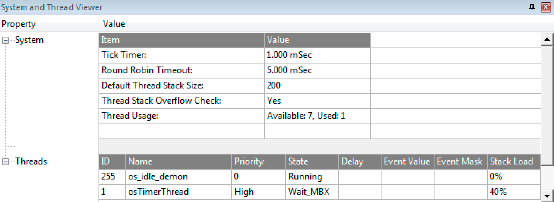
Hemos seleccionado un controlador de LED del CMSIS-Pack BSP para crear un LED que parpeadee.

Ha creado un programa simple que parpadea este LED cada 1 segundo usando un temporizador.

***RTX Kernel Awareness***

**System y Thread Viewer:**

1. Con el programa corriendo, abrir Debug → OS Support y seleccione System and Thread Viewer. Se abrirá esta ventana:

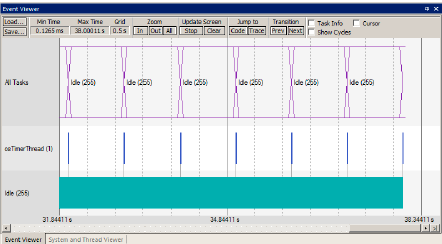


Nota: os\_idle\_demon y osTimerThread, estas tareas ya fueron creadas.

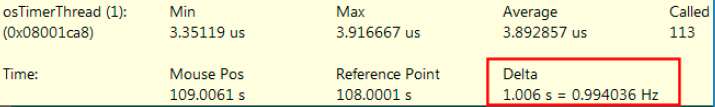
1. Establecer breakpoint de hardware en la función de callback de Timer.c en Timer2\_Callback como lo hizo previamente cerca de las líneas 31 a 35.
2. Al hacer clic en RUN, el estado de estos dos subprocesos se actualizará en tiempo real hasta que el programa se detenga.
3. Tenga en cuenta los otros campos que describen RTX.
4. Esta es una implementación RTX muy simple. Añadiremos más hilos/procesos. Estos hilos se añadirán automáticamente a esta ventana a medida que los creas. Esta ventana no necesita configuración o algo adicional en su código fuente.
5. Eliminar el punto de interrupción.

**Visor de eventos:**

1. Abra Debug → OS Support y seleccione Event Viewer. La siguiente ventana se abre. Cambie el tamaño a su convenciencia. Si aquí no muestra nada de información la causa más probable es que el SWV esté inhabilitado o la frecuencia de reloj de la CPU sea incorrecta. Ver Serial Wire Viewer en la última página para útil.



1. Haga clic en RUN.
2. Usando In , Out y All en el campo Zoom, configurar la cuadrícula de unos 0,5 segundos.
3. Es fácil ver cuando los hilos están corriendo. Tenga en cuenta la mayor parte del tiempo el inactivo el hilo se está ejecutando.
4. Puede ver de un vistazo el tiempo de su implementación RTX y si se está comportando como espera.
5. A medida que agregue nuevas tareas, se agregarán automáticamente. El Visor de eventos utiliza el Serial Wire Viewer (SWV).
6. Haga clic en Detener en la pantalla de actualización.
7. Habilitar la Task Info y el Cursor.
8. Haga clic en uno de los eventos osTimerThread (1) . Aparecerá una línea roja.
9. Coloque su mouse sobre el siguiente evento Timer Thread. Mantenga su cursor en la fila osTimerThread para corregir el muestreo.
10. Se abrirá la siguiente ventana. Tenga en cuenta que el tiempo (Delta) entre los subprocesos es de aproximadamente 1.006 segundos. Esto está cerca de la frecuencia del parpadeo del LED. Considere que hay un muestreo de error presente
11. Detenga el procesador.
12. Salir del modo de depuración.

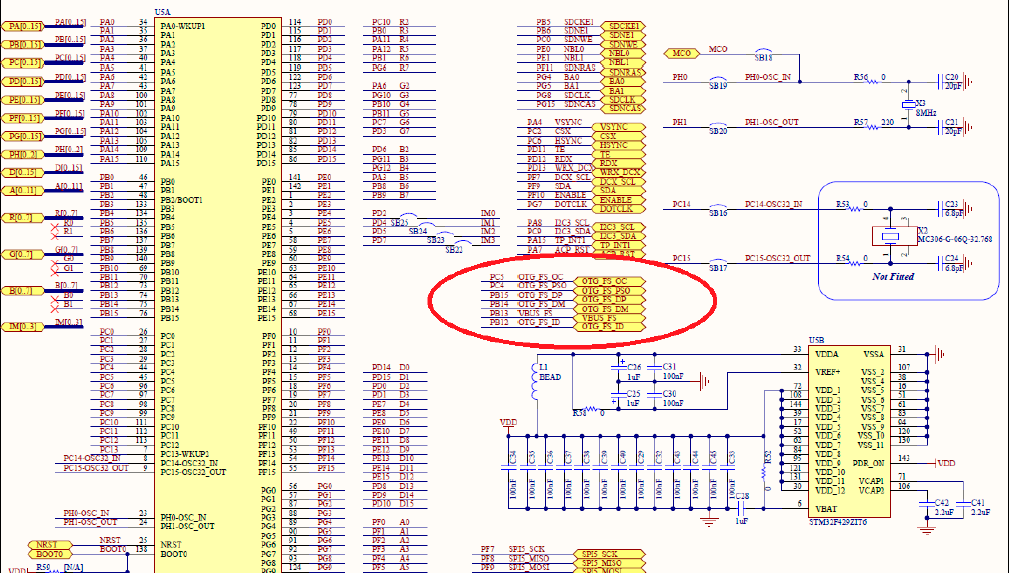


**Paso 3: Añadir USB Host con Mass Storage Support**

***Configure el CMSIS-Driver para el componente USB***

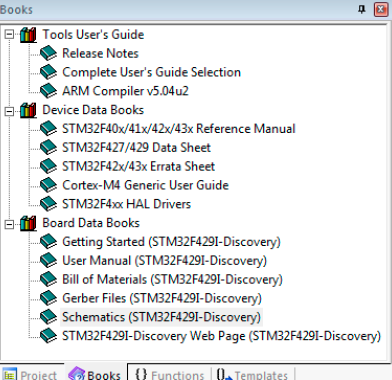
Para configurar correctamente el USB Host Middleware es necesario entender que tiene el conector USB disponible en el dispositivo

El kit de descubrimiento STM32F429I proporciona un conector USB que interactúa con el USB OTG STM32F429 que se encuentra a través del PHY en (GPIOB.14 y GPIOB.15). El pin de encendido / apagado VBUS está activo en la posición baja GPIOC.4. El pin de detección de sobrecorriente es activo bajo en GPIOC.5. Ya que solo estamos usando la interfaz USB Host, podemos ignorar el los otros pines del OTG.



Este esquema forma parte del paquete de software para el STM32F4 que se encuentra en la parte de libros en el IDE. Estos documentos utilizando la pestañas que son libros y otros documentos que se encuentran aquí son

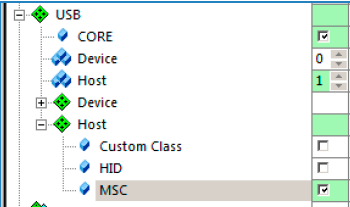
hojas de datos, guías de iniciación de STMicroelectronics, el compilador ARM y Manuales de µVision y más. La pestaña de libros se encuentra con el Proyecto y Pestañas de funciones.



**Agregue el componente de middleware USB Host al proyecto**

Como queremos conectar una memoria USB a la placa de desarrollo, debemos agregar soporte para el almacenamiento masivo USB de Clase (MSC) al proyecto:

1. Abra la ventana Gestionar entorno de tiempo de ejecución:
2. En USB: Host , seleccione MSC como se muestra aquí:

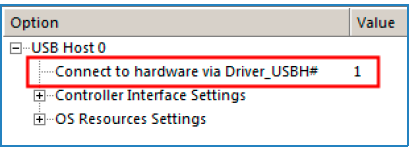


Asegúrese de no seleccionar accidentalmente MSC en el encabezado del dispositivo. Estamos configurando el STM32 como un Host y no como un Dispositivo.

1. En CMSIS Driver: USB Host (API), seleccione High-speed
2. Haga clic en Resolve para agregar otros componentes de middleware obligatorios.
3. Haga clic en Aceptar para cerrar esta ventana.

**Conecte el host USB 0 al hardware y aumente el tamaño del stack:**

1. En la ventana proyecto, debajo del encabezado USB, haga doble clic en USBH\_Config\_0.c (Host) para abrirlo.
2. Haga clic en la pestaña Configuration Wizard y luego en Expand All.

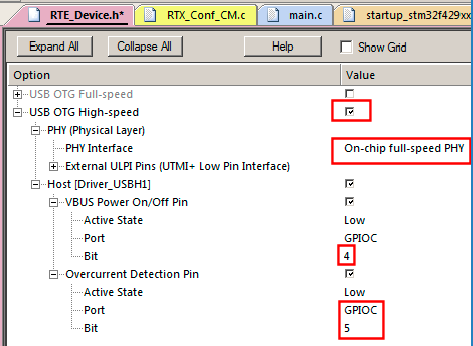


1. Establezca Connect to hardware via Driver\_USBH # en 1 . Nota: La interfaz de alta velocidad USB OTG está representada por Driver\_USBH1
2. Este es el controlador CMSIS que se configuró en el paso anterior.
3. Cambie el tamaño del Core Thread Stack Size en la parte inferior de la archivo de configuración a 540 . Usando el valor por defecto, el programa se detendrá con un desbordamiento de pila.
4. Seleccione Archivo / Guardar todo o

.

**Configure el controlador CMSIS para el host USB**

1. En la ventana proyecto, debajo del encabezado USB, haga clic doble en RTE\_Device.h para abrirlo y editarlo.
2. Haga clic en su pestaña Configuration Wizard.
3. Habilite USB OTG High-Speed como se muestra aquí:



1. Establezca los parámetros de hardware para el USB OTG High-Speed exactamente como se muestra aquí:
   * Ambos puertos deben ser GPIOC y el primer bit es 4 y el segundo es 5
   * Cambie la interfaz PHY a On-chip Full-Speed PHY.

A continuación configuraremos los recursos de stack, heap y thread para el componentes de middleware que acabamos de añadir.

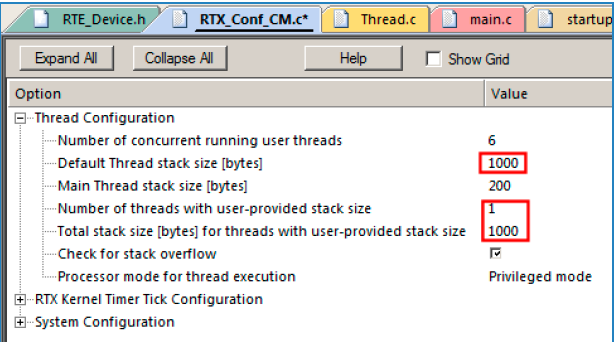
**Configurar el stack y los recursos de memoria de procesos**

Los requisitos de recursos del componente USB se pueden encontrar en la documentación de Middleware a la que se puede acceder utilizando el enlace al lado del componente USB en la ventana Manage Run-Time Environment:



**Configurar los tamaños de USB de Heap y Thread Stack:**

1. En la ventana Proyecto , bajo el encabezado Device , haga doble clic en startup\_stm32f429xx.s para abrirlo.
2. Seleccione Configuration Wizard. Confirme que el tamaño de stack esté establecido en 0x400 bytes y el tamaño de heap esté configurado en 0x200 .
3. Bajo el encabezado CMSIS , haga doble clic en RTX\_Conf\_CM.c para abrirlo.
4. Cambie el Default Thread stack size [bytes] a 1000.
5. Establezca el Number of threads with user-provided stack size en 1 .
6. Establecer Total stack size [bytes] for theads with user-provided stack size a 1000 como se muestra aquí:



**Establezca la letra de unidad predeterminada:**

1. En la ventana proyecto, debajo del encabezado Device, haga doble clic en FS\_Config.c para abrirlo.
2. Seleccione la pestaña Configuration Wizard.
3. Para USB Mass Storage Drive, el File System espera que la letra de la unidad sea U0. Asi que cambie Initial Current Drive a U0:
4. Seleccione File / Save All o.
5. Compilar el proyecto:

No se generarán errores ni advertencias como se muestra en la ventana general de resultados. Por favor, corrija cualquier error o advertencia antes continuar.

A continuación, agregaremos el código de usuario para acceder a un dispositivo USB (la memoria USB)

**Agregue el código de usuario que accede al dispositivo de almacenamiento USB**

Usaremos un subproceso CMSIS-RTOS para implementar el acceso a un archivo en la memoria USB.

**Añadir Thread.c:**

1. Haga clic derecho en Source Group 1 en la ventana Proyecto. Seleccione  Add New ítem to Group ‘Source Group 1’ …
2. Seleccione User Code Template
3. Bajo el encabezado CMSIS y en la columna Nombre, seleccione CMSIS-RTOS Thread.
4. Haga clic en Add. Esto agrega el archivo Thread.c a su proyecto.

**Agregue USBH\_MSC.c y USBH\_MSC.h:**

1. Haga clic derecho nuevamente en Source Group 1 en la ventana Proyecto. Seleccione Add New ítem to Group ‘Source Group 1’
2. Seleccione User Code Template.
3. Bajo el encabezado USB y en la columna Nombre, seleccione USB Mass Storage Access y haga clic en Add.
4. Los archivos USBH\_MSC.c y USBH\_MSC.h ahora se agregan a su proyecto bajo el encabezado Source Group 1.
5. Estos proporcionan las funciones de acceso relevantes para el dispositivo de almacenamiento USB.
6. Seleccione File / Save All

**Modificar Thread.c:**

Para permitir el acceso a los archivos, agregamos el siguiente código de aplicación en el módulo Thread.c:

1. Haga doble clic en Thread.c para abrirlo y editarlo.
2. Tenga en cuenta que cerca de las líneas 17 y 18 hay dos líneas C: return (0); y}
3. Borre todo después de estas dos líneas pero sin incluirlas. Comience a eliminar con el void Thread (línea 20).

Agregue este código a Thread.c:

**#include "USBH\_MSC.h"**

**char fbuf[200] = { 0 };**

**void Thread (void const \*argument) {**

**static unsigned int result;**

**static FILE \*f;**

**USBH\_Initialize (0);**

**while (1) {**

**result = USBH\_MSC\_DriveMount ("U0:");**

**if (result == USBH\_MSC\_OK) {**

**f = fopen ("Test.txt", "r");**

**if (f) {**

**fread (fbuf, sizeof (fbuf), 1, f);**

**fclose (f);**

**}**

**}**

**osDelay (1000);**

**}**

**}**

4. Asegúrese de tener al menos una nueva línea (CR) al final del texto. De lo contrario, esto generará warnings a la hora de compilar.

**Para iniciar este nuevo hilo RTX:**

1. En main.c cerca de la línea 77, agregue después de extern void Init\_Timers: extern void Init\_Thread (void);
2. En main.c cerca de la línea 112, agregue antes de osKernelStart ();: Init\_Thread ();
3. Seleccione File / Save All.

Lo que tenemos en este punto:

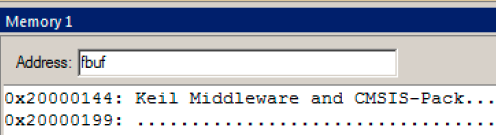
Agregamos el código para abrir, leer y cerrar los datos en el archivo Test.txt ubicado en una Memoria USB conectada al usuario USB

**Prepare una memoria USB:**

1. Tome una memoria USB y cree un archivo llamado Test.txt que contenga un mensaje corto con caracteres ASCII.
2. Escribiremos internamente el mensaje Keil Middleware y CMSIS-Pack .
3. Enchufe el cable adaptador al conector USB de la placa STM32F429I-Dicovery etiquetado como USB USER.

**Build y Run:**

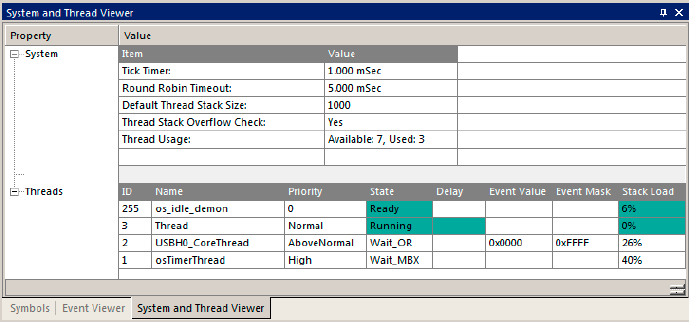
1. Compilar el proyecto. Puede recibir una advertencia de USBH\_MSC.c que puede ignorarse de forma segura.
2. Ingrese al modo de depuración.
3. Haga clic en la pestaña Memory 1 . Introduzca fbuf en esta ventana
4. Haga clic derecho en cualquier parte del área de campo de datos y seleccione Ascii
5. Establezca un punto de interrupción en Thread.c en fclose (f) cerca de la línea 35.
6. Haga clic en RUN. En unos segundos el texto aparecera en la ventana de memoria 1.
7. El programa se detendrá en el punto de interrupción del hardware.
8. Para repetir esta secuencia, haga clic en el icono RESET y luego RUN.



.

**System and Thread Viewer:**

1. Seleccione el tab System and Thread Viewer o seleccione Debug → OS Support → System and Thread Viewer si no está abierto. Tenga en cuenta el proceso Thread está corriendo y el os\_idle\_demon esta listo para correr la siguiente tarea.
2. Haga clic en el icono RESET y luego en RUN. Ud. observará el idle demon correr el programa se ejecuta y Thread entra en el estado de ejecución cuando se alcanza el punto de interrupción.
3. Elimine el punto de interrupción en Thread.c en la línea fclose (f).
4. Haga clic en RUN.
5. Deje el programa en ejecución para los siguientes pasos.

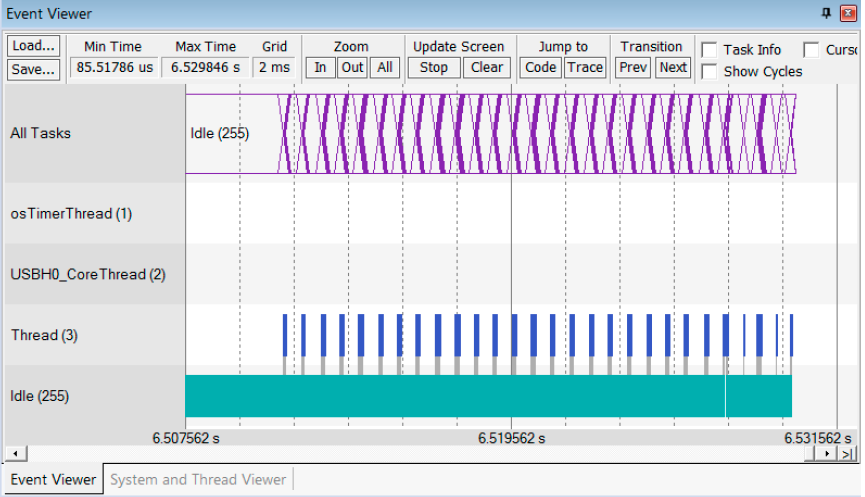


**Viendo la actividad RTX con el Visor de Eventos:**

Nota : Si esta ventana está en blanco, el Serial Wire Viewer debe estar configurado y SB9 puenteado.

1. Seleccione la pestaña Event Viewer, si aún no está abierto: seleccione Debug → OS Support → Event Viewer.
2. Ajuste el ancho de la columna para que todos los nombres de los procesos sean visibles como se muestra a continuación. Los datos serán visibles si el Serial Wire Viewer (SWV) está configurado correctamente.
3. Establezca la cuadrícula en 2 ms usando Zoom In y Zoom Out. Desplácese hasta el final del Event Viewer como se muestra a continuación.

Observe los subprocesos visibles: Los datos del subproceso (3) muestran la actividad de este subproceso antes del punto de interrupción. El observador, que la mayor parte del tiempo de procesor fue dedicado en el demonio inactivo. Puede ajustar estos tiempos para ajustar su aplicación.



**Más actividad de visualización de RTX con el Visor de eventos:**

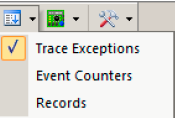
1. Seleccione Stop en el cuadro Update Screen.
2. Ajuste la rejilla a 10 ms .
3. Desplácese hacia atrás en el tiempo y podrá ver cuándo los otros subprocesos estaban activos.
4. Recuerde que puede habilitar los cuadros Cursor e Información de tareas para medir los tiempos de estos eventos.

**Modificando la ventana de memoria 1:**

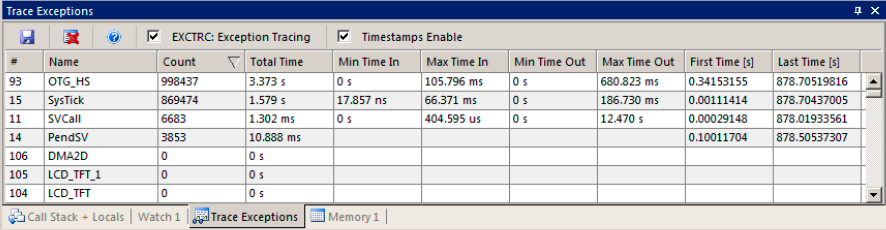
1. En la ventana Memory 1 que muestra el texto, haga clic con el botón derecho en uno de los caracteres y seleccione Modify Memory @address
2. Ingrese un 0 y presione Enter.
3. El proceso que seleccionó se cambiará a 0 y luego volverá al original a medida que Text.txt es leído nuevamente por el proceso Thread
4. La ventana de Memory 1 se actualiza en tiempo real y se puede cambiar mientras el programa se está ejecutando.

**Ventana de traza de excepción:**

1. Abra la ventana Trace Exception: haga clic en la flecha hacia abajo que se encuentra junto al icono de rastreo:

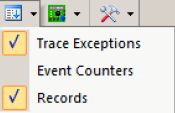


1. Seleccione Trace Exceptions. La ventana Rastrear excepciones se abre con su propia pestaña.
2. Habilite EXCTRC: Exception Tracing como se muestra en la ventana a continuación:
3. Haga clic en el encabezado de la columna Count hasta que aparezca el triángulo hacia abajo Las excepciones activas serán se mostrará con varias estadísticas como se muestra a continuación. Nota: esta ventana se actualiza mientras el programa se está ejecutando.



**Ventana Trace Records:**

1. Abra la ventana Trace Records: haga clic en la flecha hacia abajo junto al icono Trace:



1. Haga doble clic dentro de ella para borrar la ventana.
2. Las excepciones se enumerarán como ocurrieron como se muestra a continuación.
3. Haga clic derecho en esta ventana y puede filtrar diferentes tipos de eventos.
4. Una "x" en la columna Ovf significa que se perdió un cuadro. Esto se debe a que había demasiada salida de datos en el bit 1, pin de salida de cable serie (SWO). Puede aliviar esto deseleccionando las marcas de tiempo y el bit 31 de ITM. Los desbordamientos puede desaparecer, pero el Visor de eventos no funcionará sin estos dos atributos establecidos.
5. Una "x" en la columna Dly significa que la marca de tiempo podría no ser precisa en este momento. µVision se recupera de tales desbordamientos de datos de seguimiento SWV.
6. También puede aliviar los desbordamientos utilizando un adaptador de depuración Keil ULINK pro . ULINK pro puede utilizar la traza ETM de 4 bits lo que proporciona más ancho de banda. Una placa debe estar equipada con el conector ETM CoreSight de 20 pines (no disponible en el tablero STM32F429i-Discovery).
7. Cierre la ventana Trace Records.
8. Desactive EXCTRC: Exception Tracing en la ventana Trace Exception.
9. Detenga el procesador.
10. Cierre las dos ventanas Trace.
11. Salga del modo de depuración

