Keil MDK 高级调试——Event Recorder

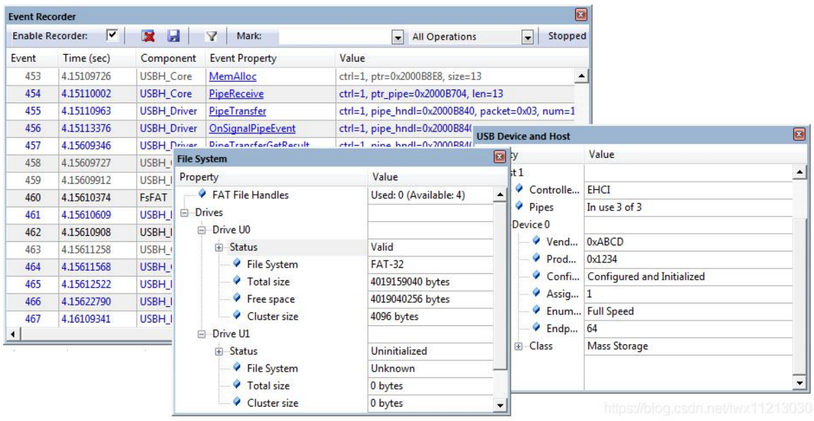
前言

如今，灵活且易于使用的中间件组件在现代微控制器应用中至关重要。这些软件组件通常是应用程序程序员的“黑匣子”。即使提供了全面的文档和源代码，对潜在问题的分析也具有挑战性。MDK 调试器提供了 Component Viewer 和 Event Recorder，它们显示软件组件的执行状态和事件信息，从而帮助您了解和分析内部操作。所需的基础结构已在 MDK 中间件和 Keil RTX5 中实现。但是，可以将这些视图轻松添加到其他软件组件或用户应用程序。

什么是 Event Recorder?

就是通过在代码中调用 Event API 来记录软件运行的一些标志信息，然后通过 MDK 相关的调试窗口以图形化的形式显示出来。

说明：MDK 调试器提供了 Component Viewer 和 Event Recorder，它们显示软件组件的执行状态和事件信息，从而帮助您了解和分析内部操作。 所需的基础结构已在 MDK 中间件和 Keil RTX5 中实现。 但是，可以将这些视图轻松添加到其他软件组件或用户应用程序。



Event	Time (sec)	Component	Event Property	Value
453	4.15109726	USBH_Core	MemAlloc	ctrl=1, ptr=0x2000B8E8, size=13
454	4.15110002	USBH_Core	PipeReceive	ctrl=1, ptr_pipe=0x2000B704, len=13
455	4.15110963	USBH_Driver	PipeTransfer	ctrl=1, pipe_hndl=0x2000B840, packet=0x03, num=1
456	4.15113376	USBH_Driver	OnSignalPipeEvent	ctrl=1, pipe_hndl=0x2000B84
457	4.15609346	USBH_Driver	PipeTransferGetResult	ctrl=1, pipe_hndl=0x2000B84
458	4.15609727	USBH_		
459	4.15609912	USBH_		
460	4.15610374	FsFAT		
461	4.15610609	USBH_		
462	4.15610908	USBH_		
463	4.15611258	USBH_		
464	4.15611568	USBH_		
465	4.15612522	USBH_		
466	4.15622790	USBH_		
467	4.16109341	USBH_		

File System

Property	Value
FAT File Handles	Used: 0 (Available: 4)
Drives	
Drive U0	
Status	Valid
File System	FAT-32
Total size	4019159040 bytes
Free space	4019040256 bytes
Cluster size	4096 bytes
Drive U1	
Status	Uninitialized
File System	Unknown
Total size	0 bytes
Cluster size	0 bytes

USB Device and Host

ty	Value
t 1	
Controlle...	EHCI
Pipes	In use 3 of 3
Device 0	
Vend...	0xABCD
Prod...	0x1234
Confi...	Configured and Initialized
Assig...	1
Enum...	Full Speed
Endp...	64
Class	Mass Storage

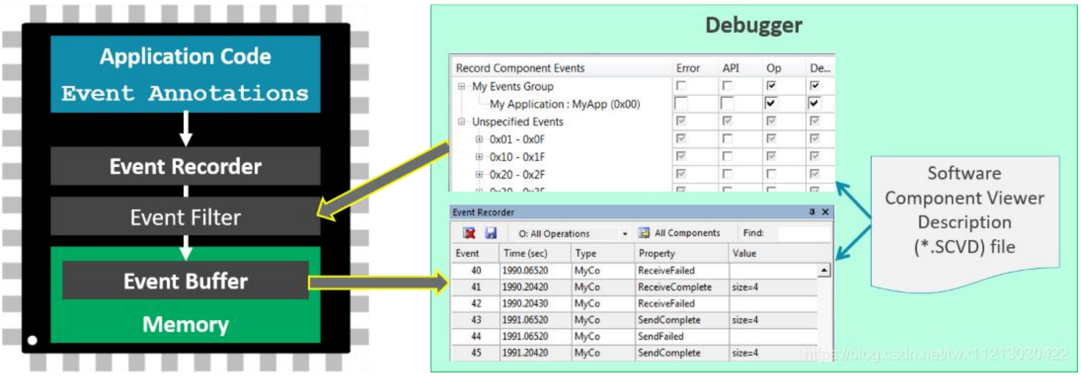
Event Recorder 与传统的串口 printf 调试区别及优势?

1. printf 调试信息只能以字符串的形式显示，Event Recorder 提供信息的分类可以在特定的窗口中以图形化的形式显示

2. 传统串口 printf 调试需要占用芯片的外设资源(一个串口)且由于串口速度限制。而 Event Recorder 是直接获取的芯片内存的数据速度与 CPU 主频一样。

3. 传统串口 printf 调试对于中断应用的调试会受限制，由于需要调用 printf 函数期间会占用不少 CPU 时间。而 Event Recorder 是直接获取内存的数据，速度快。

4. Event Recorder 不需要占用芯片的外设资源



Record Component Events	Error	API	Op	De...
⊟ My Events Group	□	□	☑	☑
My Application : MyApp (0x00)	□	□	☑	☑
⊟ Unspecified Events	☑	☑	☑	☑
⊞ 0x01 - 0x0F	☑	□	☑	☑
⊞ 0x10 - 0x1F	☑	□	☑	☑
⊞ 0x20 - 0x2F	☑	□	□	☑

Event Recorder

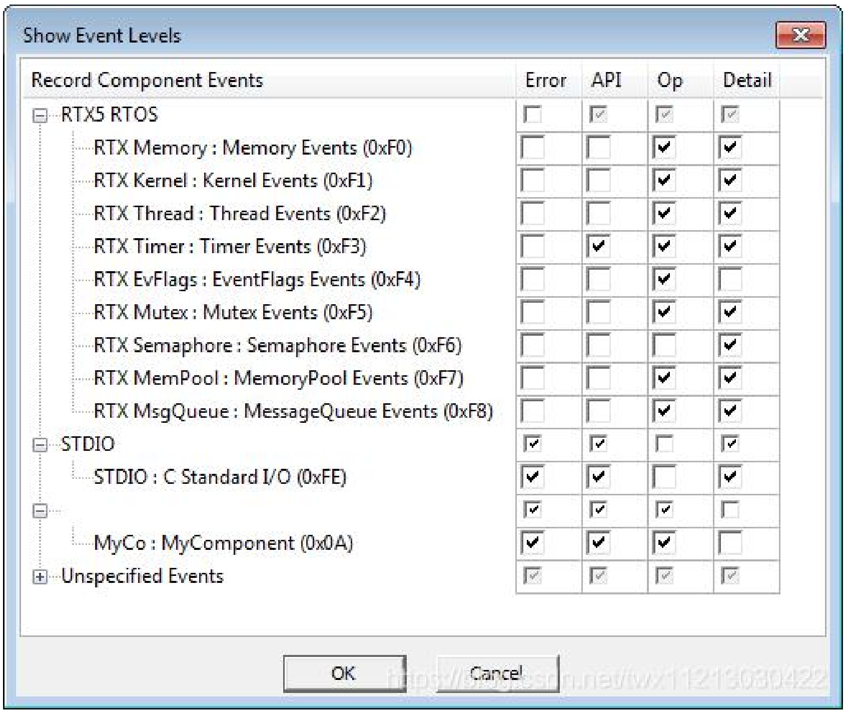
O: All Operations ▾ All Components Find:

Event	Time (sec)	Type	Property	Value
40	1990.06520	MyCo	ReceiveFailed	
41	1990.20420	MyCo	ReceiveComplete	size=4
42	1990.20430	MyCo	ReceiveFailed	
43	1991.06520	MyCo	SendComplete	size=4
44	1991.06520	MyCo	SendFailed	
45	1991.20420	MyCo	SendComplete	size=4

Event Recorder 有什么作用?

1. 获取软件组件或者用户代码中的运行情况并通过 MDK 的 Event Recorder 和 Component Viewer 窗口清晰明了的显示

2. 支持 Keil RTX 操作系统调试以及 MDK 自带的中间件(文件系统、USB 组件、网络组件、图形界面等)的调试。可以清楚的记录中间件以及 RTX 的执行情况



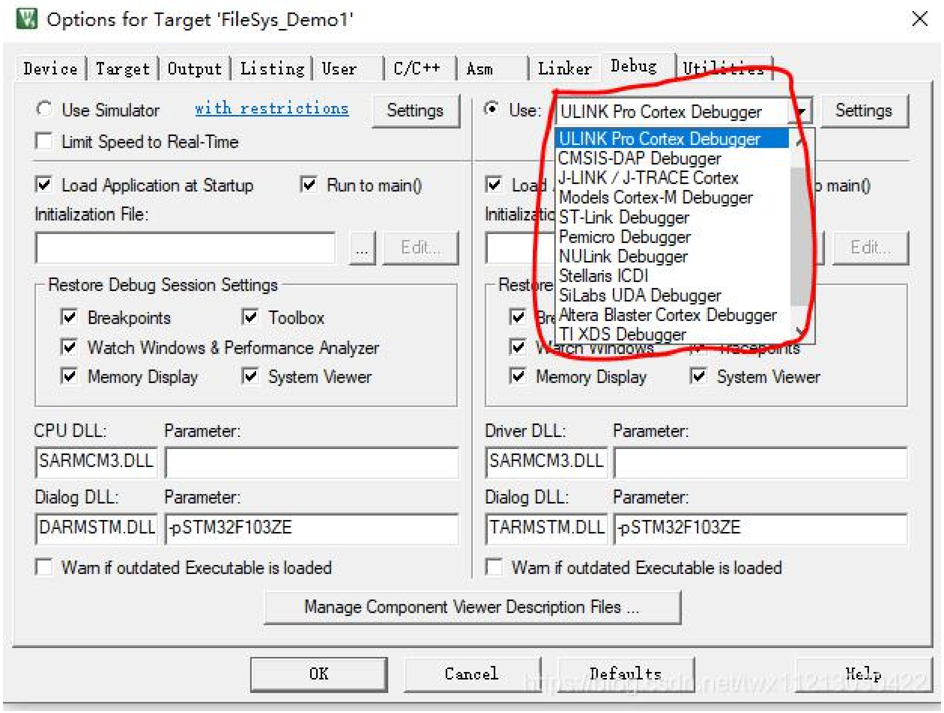
Record Component Events	Error	API	Op	Detail
⊟ RTX5 RTOS	□	☑	☑	☑
RTX Memory : Memory Events (0xF0)	□	□	☑	☑
RTX Kernel : Kernel Events (0xF1)	□	□	☑	☑
RTX Thread : Thread Events (0xF2)	□	□	☑	☑
RTX Timer : Timer Events (0xF3)	□	☑	☑	☑
RTX EvFlags : EventFlags Events (0xF4)	□	□	☑	□
RTX Mutex : Mutex Events (0xF5)	□	□	☑	☑
RTX Semaphore : Semaphore Events (0xF6)	□	□	□	☑
RTX MemPool : MemoryPool Events (0xF7)	□	□	☑	☑
RTX MsgQueue : MessageQueue Events (0xF8)	□	□	☑	☑
⊟ STDIO	☑	☑	□	☑
STDIO : C Standard I/O (0xFE)	☑	☑	□	☑
⊟	☑	☑	☑	□
MyCo : MyComponent (0x0A)	☑	☑	☑	□
⊞ Unspecified Events	☑	☑	☑	☑

3. 支持用户自定义即可以将用户自己写的代码也添加到 Event Recorder 窗口中并显示。具体请参考 arm 官方的帮助文档。

怎么使用 Event Recorder?

1. 准备工作

- 硬件调试器：ULink、Jlink、ST-Link、CMSIS-DAP 等只要是 MDK 支持的调试器就行



Options for Target 'FileSys_Demo1'

○ Use Simulator with restrictions

□ Limit Speed to Real-Time

☑ Load Application at Startup ☑ Run to main()

Initialization File:

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows & Performance Analyzer

☑ Memory Display ☑ System Viewer

CPU DLL: SARMCM3.DLL Parameter:

Dialog DLL: DARMSTM.DLL Parameter: -pSTM32F103ZE

□ Warn if outdated Executable is loaded

● Use: ULINK Pro Cortex Debugger

☑ Load ... main()

Initializatio

Restore

☑ Bre

☑ Watch Windows ☑ Tracepoints

☑ Memory Display ☑ System Viewer

Driver DLL: SARMCM3.DLL Parameter:

Dialog DLL: TARMSTM.DLL Parameter: -pSTM32F103ZE

□ Warn if outdated Executable is loaded

- 如果 MDK 的 RTE 中没有找到 Comiler 组件。就需要在 Keil 官网下载下载 Compiler 组件。https://www.keil.com/dd2/pack/

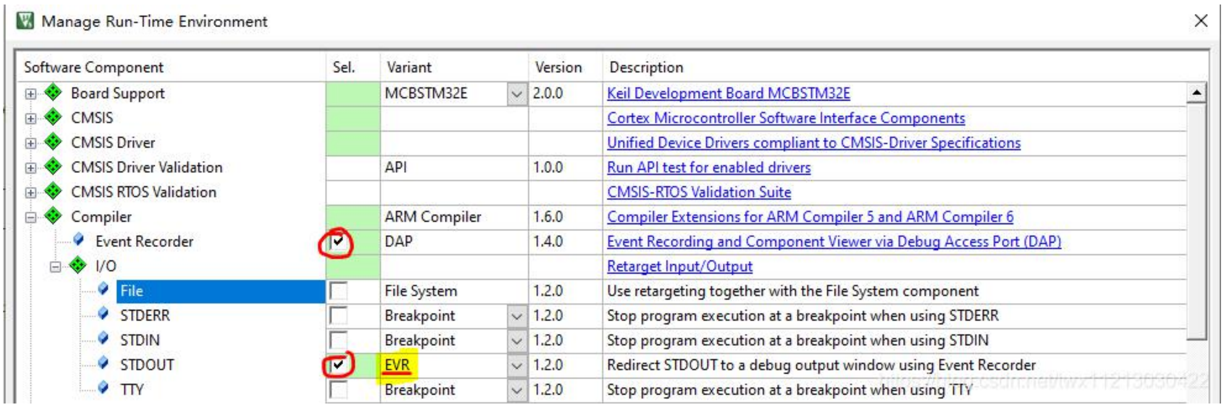
2. 开始使用

- 新建工程… 此处省略。如果不懂的如何创建工程的，请自行百度

3. 配置工程

- 打开 RTE(Manage Run-Time Environment 窗口)，在 Compiler 组件下勾选 Event Recorder

- 将 Compiler 组件下的 I/O 组件下中的 STDOUT 并将其修改为 EVR。这一步是为了将 printf 函数重定向到 Event Recorder 上



Manage Run-Time Environment

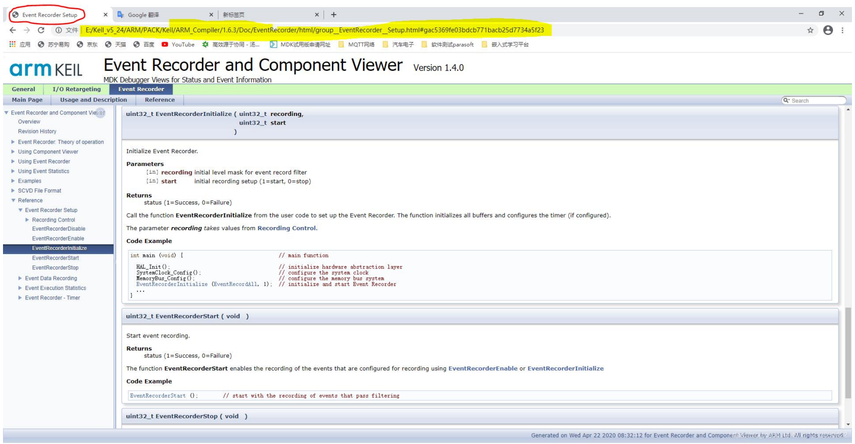
Software Component	Sel.	Variant	Version	Description
⊞ Board Support	🟩	MCBSTM32E	2.0.0	Keil Development Board MCBSTM32E
⊞ CMSIS	🟩			Cortex Microcontroller Software Interface Components
⊞ CMSIS Driver	🟩			Unified Device Drivers compliant to CMSIS-Driver Specifications
⊞ CMSIS Driver Validation		API	1.0.0	Run API test for enabled drivers
⊞ CMSIS RTOS Validation				CMSIS-RTOS Validation Suite
⊟ Compiler	🟩	ARM Compiler	1.6.0	Compiler Extensions for ARM Compiler 5 and ARM Compiler 6
Event Recorder	☑	DAP	1.4.0	Event Recording and Component Viewer via Debug Access Port (DAP)
⊟ I/O	🟩			Retarget Input/Output
File	☐	File System	1.2.0	Use retargeting together with the File System component
STDERR	☐	Breakpoint	1.2.0	Stop program execution at a breakpoint when using STDERR
STDIN	☐	Breakpoint	1.2.0	Stop program execution at a breakpoint when using STDIN
STDOUT	☑	EVR	1.2.0	Redirect STDOUT to a debug output window using Event Recorder
TTY	☐	Breakpoint	1.2.0	Stop program execution at a breakpoint when using TTY

4. 添加 Event Recorder 代码

在需要使用 Event Recorder 的代码处，调用 Event Recorder API 函数。同时如果需要进行 printf 输出的地方直接使用 printf 函数即可。

注：需要添加 EventRecorder.h 头文件。

具体的 Event Recorder API 函数，请参考帮助文档。在 MDK 的安装目录下，如：



E:/Keil_v5_24/ARM/PACK/Keil/ARM_Compiler/1.6.3/Doc/EventRecorder/html/group__EventRecorder__Setup.html#gac5369fe03bdcb771bacb25d7734a5f23

Event Recorder and Component Viewer Version 1.4.0

MDK Debugger Views for Status and Event Information

```
uint32_t EventRecorderInitialize ( uint32_t recording,
                                   uint32_t start
                                 )
```

Initialize Event Recorder.

Parameters

[in] **recording** initial level mask for event record filter

[in] **start** initial recording setup (1=start, 0=stop)

Returns

status (1=Success, 0=Failure)

Call the function **EventRecorderInitialize** from the user code to set up the Event Recorder. The function initializes all buffers and configures the timer (if configured).

The parameter ***recording*** *takes* values from **Recording Control**.

Code Example

```
int main (void) {                                   // main function

  HAL_Init ();                                      // initialize hardware abstraction layer
  SystemClock_Config ();                            // configure the system clock
  MemoryBus_Config ();                              // configure the memory bus system
  EventRecorderInitialize (EventRecordAll, 1);      // initialize and start Event Recorder
  ...
}
```

```
uint32_t EventRecorderStart ( void )
```

Start event recording.

Returns

status (1=Success, 0=Failure)

The function **EventRecorderStart** enables the recording of the events that are configured for recording using **EventRecorderEnable** or **EventRecorderInitialize**

Code Example

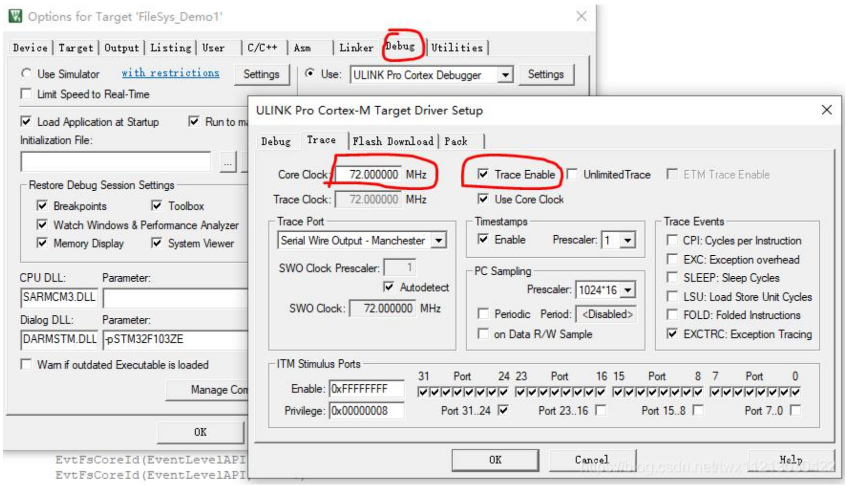
```
EventRecorderStart ();        // start with the recording of events that pass filtering
```

```
uint32_t EventRecorderStop ( void )
```

Generated on Wed Apr 22 2020 08:32:12 for Event Recorder and Component Viewer by ARM Ltd. All rights reserved.

5. 配置调试器

点击 Options for Target 图标 >> Debug >> Trace。主要设置 Core Clock 为具体的芯片内核时钟频率，然后勾选 Trace Enable



Options for Target 'FileSys_Demo1'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

○ Use Simulator with restrictions Settings ● Use: ULINK Pro Cortex Debugger Settings

□ Limit Speed to Real-Time

☑ Load Application at Startup ☑ Run to main()

Initialization File:

Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance Analyzer
☑ Memory Display ☑ System Viewer

CPU DLL: SARMCM3.DLL Parameter:

Dialog DLL: DARMSTM.DLL Parameter: -pSTM32F103ZE

□ Warn if outdated Executable is loaded

ULINK Pro Cortex-M Target Driver Setup

Debug | Trace | Flash Download | Pack

Core Clock: 72.000000 MHz ☑ Trace Enable □ Unlimited Trace □ ETM Trace Enable

Trace Clock: 72.000000 MHz ☑ Use Core Clock

Trace Port: Serial Wire Output - Manchester

SWO Clock Prescaler: 1 ☑ Autodetect

SWO Clock: 72.000000 MHz

Timestamps: ☑ Enable Prescaler: 1

PC Sampling Prescaler: 1024*16 □ Periodic Period: <Disabled> □ on Data R/W Sample

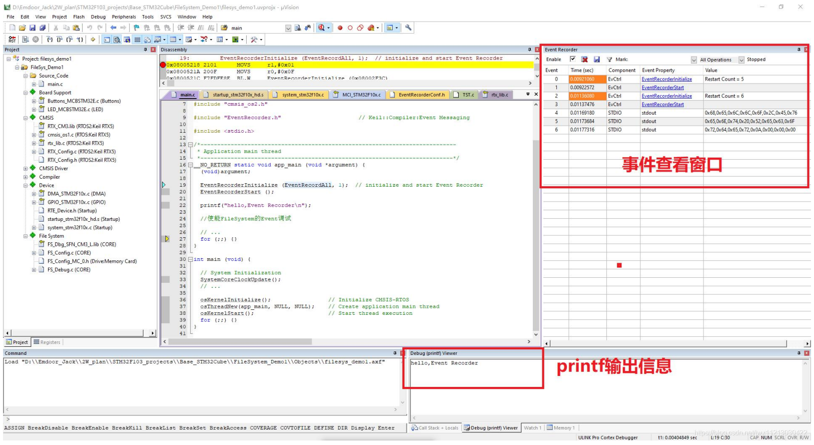
Trace Events:
□ CPI: Cycles per Instruction
□ EXC: Exception overhead
□ SLEEP: Sleep Cycles
□ LSU: Load Store Unit Cycles
□ FOLD: Folded Instructions
☑ EXCTRC: Exception Tracing

ITM Stimulus Ports
Enable: 0xFFFFFFFF
Privilege: 0x00000008 Port 31..24 ☑ Port 23..16 □ Port 15..8 □ Port 7..0 □

Manage Component Viewer Description Files
OK
OK
Cancel
Help
EvtFsCoreId(EventLevelAPI
EvtFsCoreId(EventLevelAPI

6. 编译并开始调试工程

点击调试按钮，点击 View - Analysis Windows - Event Recorder



Disassembly

```
     19:          EventRecorderInitialize (EventRecordAll, 1);  // initialize and start Event Recorder
0x08005218 2101      MOVS     r1,#0x01
0x0800521A 200F      MOVS     r0,#0x0F
0x0800521C F7FDFE8E  BL.W     EventRecorderInitialize (0x08002F3C)
```

main.c

```
#include "cmsis_os2.h"

#include "EventRecorder.h"                    // Keil::Compiler:Event Messaging

#include <stdio.h>

/*----------------------------------------------------------------------------
 * Application main thread
 *---------------------------------------------------------------------------*/
__NO_RETURN static void app_main (void *argument) {
  (void)argument;

  EventRecorderInitialize (EventRecordAll, 1);  // initialize and start Event Recorder
  EventRecorderStart ();

  printf("hello,Event Recorder\n");

  //使能FileSystem的Event调试

  // ...
  for (;;) {}
}

int main (void) {

  // System Initialization
  SystemCoreClockUpdate();
  // ...

  osKernelInitialize();                 // Initialize CMSIS-RTOS
  osThreadNew(app_main, NULL, NULL);    // Create application main thread
  osKernelStart();                      // Start thread execution
  for (;;) {}
}
```

Event Recorder

Event	Time (sec)	Component	Event Property	Value
0	0.00921060	EvCtrl	EventRecorderInitialize	Restart Count = 5
1	0.00922572	EvCtrl	EventRecorderStart	
2	0.01136080	EvCtrl	EventRecorderInitialize	Restart Count = 6
3	0.01137476	EvCtrl	EventRecorderStart	
4	0.01169180	STDIO	stdout	0x68,0x65,0x6C,0x6C,0x6F,0x2C,0x45,0x76
5	0.01173684	STDIO	stdout	0x65,0x6E,0x74,0x20,0x52,0x65,0x63,0x6F
6	0.01177316	STDIO	stdout	0x72,0x64,0x65,0x72,0x0A,0x00,0x00,0x00

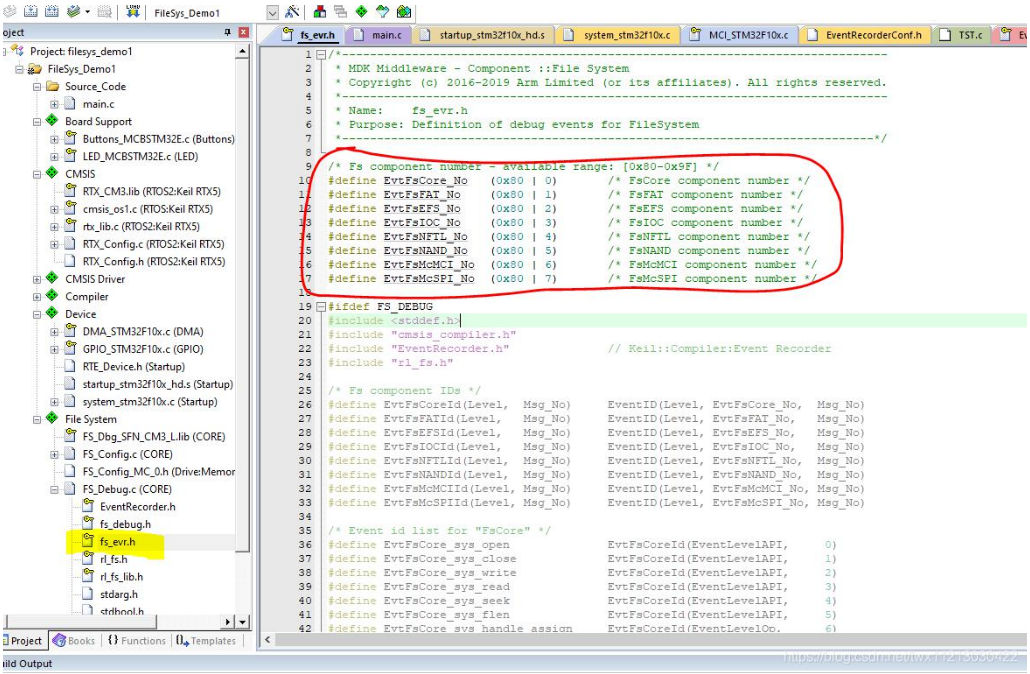
Debug (printf) Viewer

```
hello,Event Recorder
```

备注说明：

关于 MDK 组件调试中 Event id 中的 component number 的取值及对应的组件事件。

请参考 MDK 组件的头文件。如：FileSystem 中的 Debug 头文件：



```
/*------------------------------------------------------------------------------
 * MDK Middleware - Component ::File System
 * Copyright (c) 2016-2019 Arm Limited (or its affiliates). All rights reserved.
 *------------------------------------------------------------------------------
 * Name:    fs_evr.h
 * Purpose: Definition of debug events for FileSystem
 *----------------------------------------------------------------------------*/

/* Fs component number - available range: [0x80-0x9F] */
#define EvtFsCore_No    (0x80 | 0)        /* FsCore component number */
#define EvtFsFAT_No     (0x80 | 1)        /* FsFAT component number */
#define EvtFsEFS_No     (0x80 | 2)        /* FsEFS component number */
#define EvtFsIOC_No     (0x80 | 3)        /* FsIOC component number */
#define EvtFsNFTL_No    (0x80 | 4)        /* FsNFTL component number */
#define EvtFsNAND_No    (0x80 | 5)        /* FsNAND component number */
#define EvtFsMcMCI_No   (0x80 | 6)        /* FsMcMCI component number */
#define EvtFsMcSPI_No   (0x80 | 7)        /* FsMcSPI component number */

#ifdef FS_DEBUG
#include <stddef.h>
#include "cmsis_compiler.h"
#include "EventRecorder.h"                // Keil::Compiler:Event Recorder
#include "rl_fs.h"

/* Fs component IDs */
#define EvtFsCoreId(Level,  Msg_No)     EventID(Level, EvtFsCore_No,  Msg_No)
#define EvtFsFATId(Level,   Msg_No)     EventID(Level, EvtFsFAT_No,   Msg_No)
#define EvtFsEFSId(Level,   Msg_No)     EventID(Level, EvtFsEFS_No,   Msg_No)
#define EvtFsIOCId(Level,   Msg_No)     EventID(Level, EvtFsIOC_No,   Msg_No)
#define EvtFsNFTLId(Level,  Msg_No)     EventID(Level, EvtFsNFTL_No,  Msg_No)
#define EvtFsNANDId(Level,  Msg_No)     EventID(Level, EvtFsNAND_No,  Msg_No)
#define EvtFsMcMCIId(Level, Msg_No)     EventID(Level, EvtFsMcMCI_No, Msg_No)
#define EvtFsMcSPIId(Level, Msg_No)     EventID(Level, EvtFsMcSPI_No, Msg_No)

/* Event id list for "FsCore" */
#define EvtFsCore_sys_open              EvtFsCoreId(EventLevelAPI,     0)
#define EvtFsCore_sys_close             EvtFsCoreId(EventLevelAPI,     1)
#define EvtFsCore_sys_write             EvtFsCoreId(EventLevelAPI,     2)
#define EvtFsCore_sys_read              EvtFsCoreId(EventLevelAPI,     3)
#define EvtFsCore_sys_seek              EvtFsCoreId(EventLevelAPI,     4)
#define EvtFsCore_sys_flen              EvtFsCoreId(EventLevelAPI,     5)
#define EvtFsCore_sys_handle_assign     EvtFsCoreId(EventLevelOp,      6)
```