

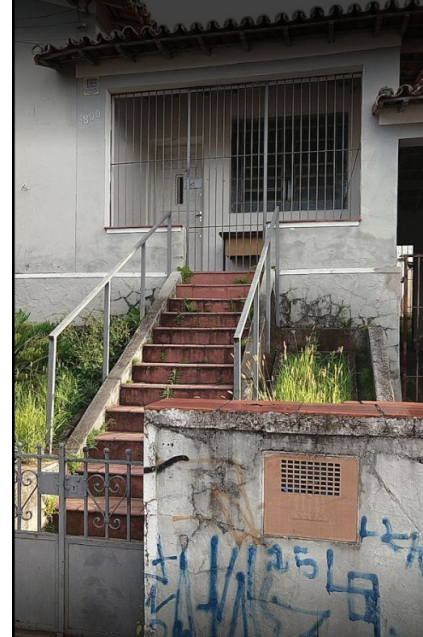


Aprendendo a programar microcontroladores ARM da ST com FreeRTOS

Jorge Guzman

0 Laboratório Hacker de Campinas

- Sala Central (Oficinas, Palestras)
- Sala Coringa (Biblioteca)
- Laboratorio de Eletronica
- Cozinha
- Marcenaria
- Area externa (Area de testes)
- Network





SEGGER

- Atua na indústria de sistemas embarcados.
 - Desenvolvimento de middleware para dispositivos embarcados
 - Ferramentas de desenvolvimento e programação
- **J-Link** ferramenta de gravação é debug
 - ARM7/ARM9/ARM11
 - Cortex M0/M0+/M1/M3/M4/M7/M23/M33
 - Cortex R4/R5/R8
 - Cortex A5/A7/A8/A9/A12/A15/A17
 - Renesas RX
 - Microchip PIC32J





LIB RTT

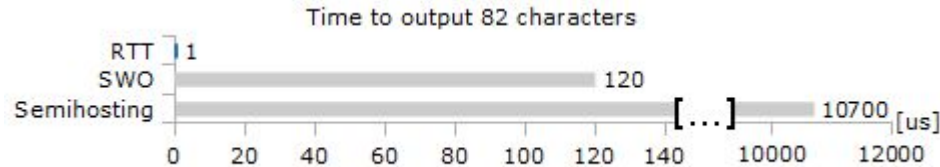
- A biblioteca RTT (Real Time Transfer) possibilita transferir dados em alta velocidade entre o Host (PC) e o target (microcontrolador).
- Não usa nenhum pino adicional do microcontrolador como uma UART, conexão SWO ou semihost; ela se dá apenas entre uma região de memória RAM do microcontrolador e o J-Link.
- A lib armazena as strings de saída nessa região de memória usando uma rotina de memcpy, posteriormente essas strings são enviadas ao J-Link via SWD ou JTAG (Protocolos de debug).
- Organização dos arquivos da lib RTT:

```
RTT
├── SEGGER_RTT.c
├── SEGGER_RTT_Conf.h
├── SEGGER_RTT.h
├── SEGGER_RTT_printf.c
Syscalls
├── SEGGER_RTT_Syscalls_GCC.c
├── SEGGER_RTT_Syscalls_IAR.c
├── SEGGER_RTT_Syscalls_KEIL.c
└── SEGGER_RTT_Syscalls_SES.c
```



VELOCIDADE DE TRANSMISSÃO

- O RTT gasta 1 microssegundo para enviar 82 caracteres com o microcontrolador rodando a 168 MHz; em contraponto, a conexão SWO gasta 120 microssegundos, e a conexão semihost gasta 10,7 milissegundos.
- A velocidade máxima de envio de dados depende do tamanho do buffer e da taxa de comunicação do J-Link com o microcontrolador.





RTT – PRINCIPAIS FUNÇÕES

- `SEGGER_RTT_ConfigUpBuffer`: Inicializa a estrutura do bloco de controle RTT ao usar somente destinos de RAM.
- `SEGGER_RTT_printf`: Com ela é possível transmitir mensagens formatadas
- `SEGGER_RTT_Write`: Transmite uma string.
- `SEGGER_RTT_Read`: Com ela é possível receber mensagens.



USANDO A LIB RTT

```
void rtt_ex(void)
{
    uint16_t counter = 0;
    uint16_t len = 0;
    uint8_t payload[20];

    SEGGER_RTT_ConfigUpBuffer(0, NULL, NULL, 0, SEGGER_RTT_MODE_BLOCK_IF_FIFO_FULL);

    while(1)
    {
        len = sprintf((char*)payload, "Counter: %d\n", rtt_ex++);

        SEGGER_RTT_Write(0, (const char*)payload, len);

        HAL_Delay(20);
    }
}
```



SOFTWARE – JLink RTT VIEWER

- Para análise os dados enviados pelo microcontrolador via RTT podemos usar o software RTT Viewer

A screenshot of the J-Link RTT Viewer V6.35g (beta) application window. The window has a menu bar with 'File', 'Terminals', 'Input', 'Logging', and 'Help'. Below the menu bar is a tabbed interface with 'Log', 'All Terminals', 'Terminal 0', and 'Terminal 1'. The 'Log' tab is active, displaying a black terminal window with white text. The text shows a series of printf statements and their outputs, including 'SEGGER Real-Time-Terminal Sample', '##### Testing SEGGER printf() #####', and various formatted strings and numbers. At the bottom of the window, there is a status bar that reads 'RTT Viewer connected.' and '0.002 MB'. There are also 'Enter' and 'Clear' buttons at the bottom of the terminal window.

```
J-Link RTT Viewer V6.35g (beta)
File Terminals Input Logging Help
Log All Terminals Terminal 0 Terminal 1
SEGGER Real-Time-Terminal Sample

##### Testing SEGGER printf() #####
printf Test: %c, 'S' : S.
printf Test: %5c, 'E' : E.
printf Test: %-5c, 'G' : G.
printf Test: %5.3c, 'G' : G.
printf Test: %.3c, 'E' : E.
printfTest: %c, 'R' : R.
printf Test: %s, "RTT" : RTT.
printf Test: %s, "RTT\r\nRocks." : RTT
Rocks..
printf Test: %u, 12345 : 12345.
printf Test: %+u, 12345 : 12345.
printf Test: %.3u, 12345 : 12345.
printf Test: %.6u, 12345 : 012345.
printf Test: %6.3u, 12345 : 12345.
printf Test: %8.6u, 12345 : 012345.

RTT Viewer connected. 0.002 MB
```




LIB SYSVIEW

- É um de ferramentas para análise visual.
- Fornece visão completa de um aplicativo, ajudando no desenvolvimento e análise de threads e eventos.
- Permite a gravação contínua de dados é análise do chaveamento de contexto entre interrupções e threads.
- Usa a lib RTT



SYSVIEW – RTOS SUPORTADOS

- A lib SystemView pode ser incluído em projetos com os RTOS:
 - embOS
 - uC/OS-II – uC/OS-III,
 - Micrium OS Kernel
 - FreeRTOS v8/v9/v10



ARQUIVOS DA LIB SYSVIEW

- Organização dos arquivos da lib SystemView

Target source package

File	Description
./Src/Config/Global.h	Global data types for SystemView.
./Src/Config/SEGGER_RTT_Conf.h	SEGGER Real Time Transfer (RTT) configuration file.
./Src/Config/SEGGER_SYSVIEW_Conf.h	SEGGER SYSTEMVIEW configuration file.
./Src/Sample/embOS	Initialization and configuration of SystemView with embOS.
./Src/Sample/FreeRTOSV8	Initialization and configuration of SystemView with FreeRTOS V8.
./Src/Sample/FreeRTOSV9	Initialization and configuration of SystemView with FreeRTOS V9.
./Src/Sample/MicriumOSKernel	Initialization and configuration of SystemView with the Micrium OS Kernel.
./Src/Sample/NoOS	Initialization and configuration of SystemView with no OS.
./Src/Sample/uCOS-III	Initialization and configuration of SystemView with uC/OS-III.
./Src/SEGGER/SEGGER.h	Global types & general purpose utility functions.
./Src/SEGGER/SEGGER_RTT.c	SEGGER RTT module source.
./Src/SEGGER/SEGGER_RTT.h	SEGGER RTT module header.
./Src/SEGGER/SEGGER_SYSVIEW.c	SEGGER SYSTEMVIEW module source.
./Src/SEGGER/SEGGER_SYSVIEW.h	SEGGER SYSTEMVIEW module header.
./Src/SEGGER/SEGGER_SYSVIEW_ConfDefault.h	SEGGER SYSTEMVIEW configuration fallback.
./Src/SEGGER/SEGGER_SYSVIEW_Int.h	SEGGER SYSTEMVIEW internal header.



ROTINAS DE TRACE

```
FreeRTOS.h
393 #endif
394
395 #ifndef traceQUEUE_CREATE
396 #define traceQUEUE_CREATE( pxNewQueue )
397 #endif
398
399 #ifndef traceQUEUE_CREATE_FAILED
400 #define traceQUEUE_CREATE_FAILED( ucQueueType )
401 #endif
402
403 #ifndef traceCREATE_MUTEX
404 #define traceCREATE_MUTEX( pxNewQueue )
405 #endif
406
```

```
SEGGER_SYSVIEW_FreeRTOS.h
226
227 #define traceQUEUE_CREATE( pxNewQueue ) SEGGER_SYSVIEW_RecordU32x3( apiID_OFFSET +
228 #define traceQUEUE_DELETE( pxQueue ) SEGGER_SYSVIEW_RecordU32 ( apiID_OFFSET +
229 #define traceQUEUE_PEEK( pxQueue ) SEGGER_SYSVIEW_RecordU32x4( apiID_OFFSET +
230 #define traceQUEUE_PEEK_FROM_ISR( pxQueue ) SEGGER_SYSVIEW_RecordU32x2( apiID_OFFSET +
231 #define traceQUEUE_PEEK_FROM_ISR_FAILED( pxQueue ) SEGGER_SYSVIEW_RecordU32x2( apiID_OFFSET +
232 #define traceQUEUE_RECEIVE( pxQueue ) SEGGER_SYSVIEW_RecordU32x4( apiID_OFFSET +
```

```
queue.c
400 /*
401 */
402 static void prvInitialiseNewQueue( const UBaseType_t uxQueueLength, const UBaseType_t
403 {
404 /* Remove compiler warnings about unused parameters should
405 configure TRACE FACILITY not be set to 1. */
406 ( void ) ucQueueType;
407
408 if( uxItemSize == ( UBaseType_t ) 0 )
409 {
410 /* No RAM was allocated for the queue storage area, but PC head cannot
411 be set to NULL because NULL is used as a key to say the queue is used as
412 a mutex. Therefore just set pcHead to point to the queue as a benign
413 value that is known to be within the memory map. */
414 pxNewQueue->pcHead = ( int8_t * ) pxNewQueue;
415 }
416 else
417 {
418 /* Set the head to the start of the queue storage area. */
419 pxNewQueue->pcHead = ( int8_t * ) pucQueueStorage;
420 }
421
422 /* Initialise the queue members as described where the queue type is
423 defined. */
424 pxNewQueue->uxLength = uxQueueLength;
425 pxNewQueue->uxItemSize = uxItemSize;
426 ( void ) xQueueGenericReset( pxNewQueue, pdTRUE );
427
428 #if ( configUSE_TRACE_FACILITY == 1 )
429 {
430 pxNewQueue->ucQueueType = ucQueueType;
431 }
432 #endif /* configUSE_TRACE_FACILITY */
433
434 #if( configUSE_QUEUE_SETS == 1 )
435 {
436 pxNewQueue->pxQueueSetContainer = NULL;
437 }
438 #endif /* configUSE_QUEUE_SETS */
439
440 traceQUEUE_CREATE( pxNewQueue );
441 }
```



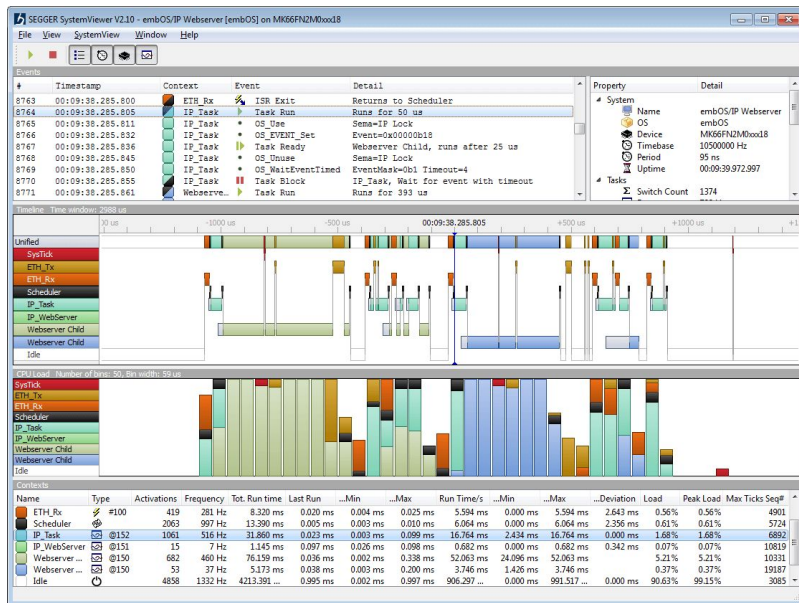
SYSVIEW – PRINCIPAIS FUNÇÕES

- `SEGGER_SYSVIEW_Conf`: Configura e inicializa Systemview pelo firmware. Usar função antes de inicializar o scheduler do FreeRTOS.
- `SEGGER_SYSVIEW_Start`: Inicializa gravação da aplicação, rotina chamada ao usar o software Systemview.
- `SEGGER_SYSVIEW_Print`: Imprime uma mensagem formatada no console do SystemView
- `SEGGER_SYSVIEW_Warn`: Imprime uma mensagem formatada no formato de **Warning** no console do SystemView
- `SEGGER_SYSVIEW_Error`: Imprime uma mensagem formatada no formato de **Error** no console do SystemView
- `SEGGER_SYSVIEW_RecordEnterISR`: Inicializa a gravação ao entrar em uma ISR
- `SEGGER_SYSVIEW_RecordExitISR`: Finaliza a gravação ao sair de uma ISR



SOFTWARE – SYSTEMVIEW

- Versão free gravação de até 1 milhão de eventos





MÃOS À OBRA



INSTALANDO SOFTWARES

Converter ST-Link em um J-Link: <https://www.embarcados.com.br/convertendo-st-link-em-um-j-link/>

ST-Link Reflash Utility: https://www.segger.com/downloads/jlink/#STLink_Reflash

J-Link Software: <https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack>

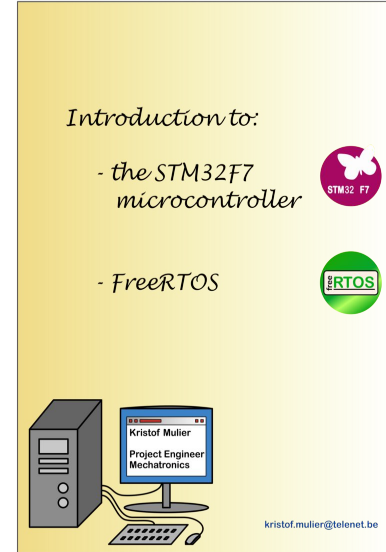
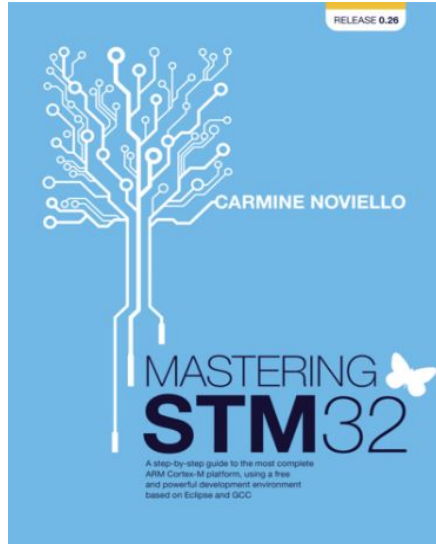
SystemView: <https://www.segger.com/downloads/free-utilities/#SystemView>

J-Link RTT Viewer: <https://www.segger.com/products/debug-probes/j-link/technology/about-real-time-transfer/>

Manual SystemView: <https://www.segger.com/downloads/free-utilities/UM08027>



Livros



Referências:

https://docs.aws.amazon.com/pt_br/freertos-kernel

<https://e-labworks.com/treinamentos/freertos/slides>



Mídias Sociais

Telegram

https://t.me/lhc_campinas

Facebook:

<https://www.facebook.com/LabHackerCampinas/>

Web:

<https://lhc.net.br/wiki/Categoria:Eventos>