Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

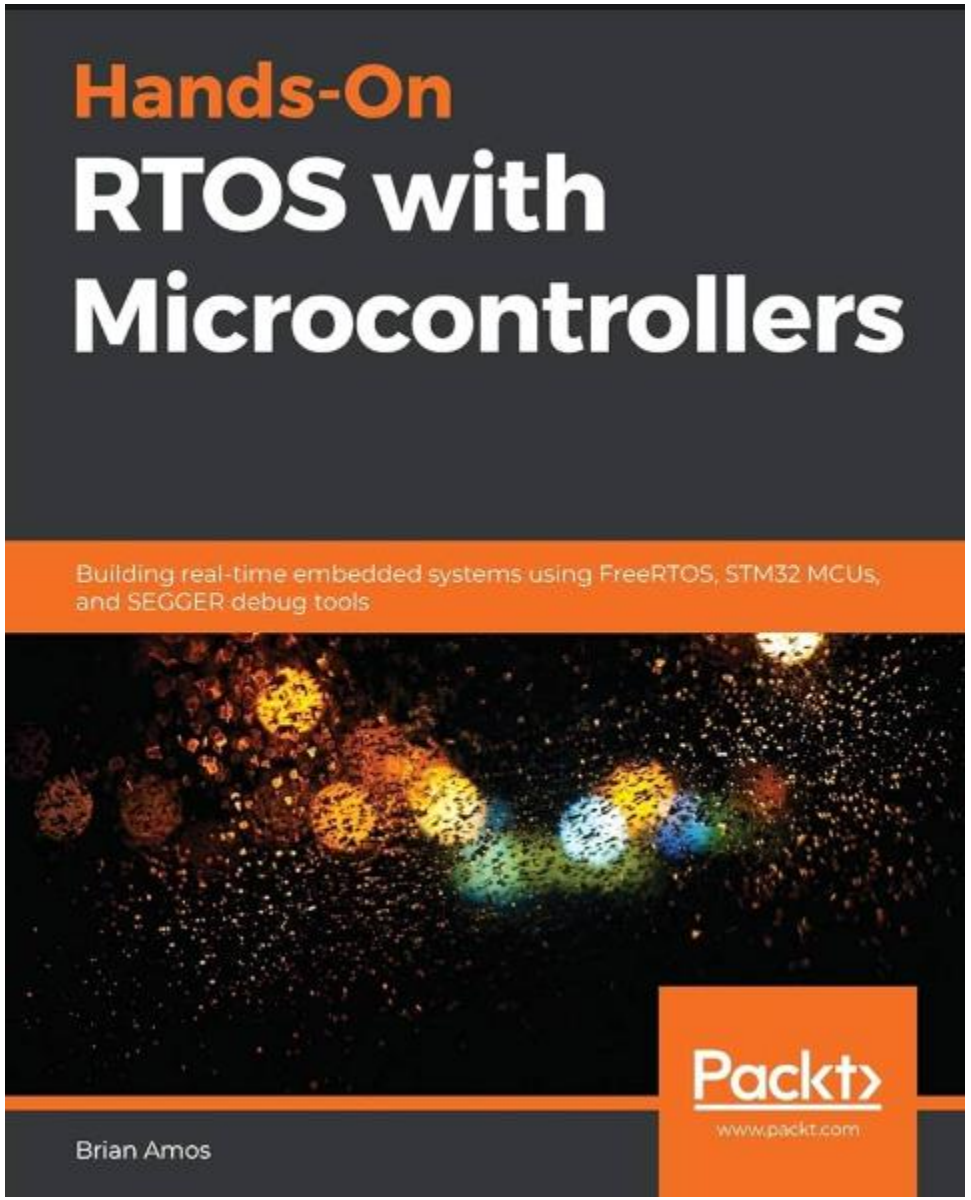
CEG4566/CSI4541/SEG4545

Conception de systèmes informatiques en temps réel

Hiver 2024

Professeur : Mohamed Ali Ibrahim, ing., Ph.D.

Source :


Hands-On
RTOS with
Microcontrollers
Building real-time embedded systems using FreeRTOS, STM32 MCUs, and SEGGER debug tools
Packt
www.packt.com
Brian Amos

Chapitre 6 :
Outils de débogage pour les systèmes en temps réel

Aperçu

- L'importance d'excellents outils de débogage
- Utilisation de SEGGER J-Link
- Utilisation de l'ozone SEGGER
- Utilisation de SEGGER SystemView
- Autres outils intéressants

L'importance d'excellents outils de débogage (1/3)

- Lors du développement d'un logiciel, il est trop facile de commencer à écrire du code sans penser à tous les détails.
- Grâce aux outils de génération de code et aux bibliothèques tierces, nous pouvons très rapidement développer une application riche en fonctionnalités et la faire tourner sur du matériel réel dans un délai assez court.
- Cependant, lorsqu'il s'agit de faire fonctionner chaque partie d'un système à 100 %, les choses sont un peu plus difficiles.
- Si un système est mis en place trop rapidement et que les composants n'ont pas été correctement testés avant d'être intégrés, certains éléments fonctionneront la plupart du temps, mais pas toujours.

L'importance d'excellents outils de débogage (2/3)

- Souvent, avec les systèmes embarqués, seules quelques parties de l'application sous-jacente sont visibles.
- Il peut être difficile d'évaluer la santé globale du système du point de vue de l'utilisateur.
- Historiquement, les bons outils de débogage étaient moins courants pour les travaux embarqués que pour les travaux non embarqués.
- Le fait de placer des instructions d'impression partout ne vous mène qu'à un certain point, provoque des problèmes de synchronisation, et ainsi de suite.
- Le clignotement des DEL est encombrant et ne donne pas beaucoup d'informations.
- L'analyse des signaux par le matériel peut aider à vérifier les symptômes, mais ne permet pas toujours d'isoler la cause première d'un problème.
- Essayer de comprendre quel code est réellement en cours d'exécution (et quand) dans un système piloté par les événements est un véritable défi sans les outils permettant de visualiser l'exécution.

L'importance d'excellents outils de débogage (3/3)

- C'est pourquoi il est extrêmement utile de disposer d'une variété d'outils familiers.
- Il vous permet de concentrer vos efforts sur le développement de petites parties de l'application en toute confiance.
- La confiance vient de la vérification rigoureuse de chaque élément de fonctionnalité au fur et à mesure de son développement et de son intégration au reste du système.
- Cependant, pour effectuer la vérification, nous devons avoir la transparence dans différentes parties du code (et pas seulement dans les parties observables de l'extérieur du système).
- Au cours de la vérification, il arrive souvent qu'il soit nécessaire d'observer l'exécution des tâches entre elles.
- Deux domaines importants nous aident à atteindre les objectifs de transparence du système et de relations observables entre les tâches :
  - Débogage en fonction du RTOS et
  - Visualisation du RTOS.

Débogage en fonction du RTOS (1/6)

- Avec les configurations de débogage traditionnelles utilisées pour le codage bare-metal (par exemple, sans système d'exploitation), il n'y avait qu'une seule pile à observer.
- Le modèle de programmation étant une super boucle unique avec quelques interruptions, cela n'a pas posé beaucoup de problèmes.
- A tout moment, l'état du système peut être discerné par les éléments suivants :
  - Savoir dans quelle fonction se trouve le compteur de programme (PC)
  - Connaître les interruptions actives
  - Examiner la valeur des principales variables globales
  - Observer/dérouler la pile

Débogage en fonction du RTOS (2/6)

- Avec un système basé sur un RTOS, l'approche de base est très similaire, mais le modèle de programmation est étendu pour inclure des tâches multiples fonctionnant en parallèle.
- N'oubliez pas que chaque tâche est en fait une boucle infinie isolée.
- Étant donné que chaque tâche possède sa propre pile et peut se trouver dans différents états de fonctionnement, des informations supplémentaires sont nécessaires pour discerner l'état global du système :
  - Connaître l'état opérationnel actuel de chaque tâche
  - Connaître la tâche et la fonction du PC
  - Savoir quelles interruptions sont actives
  - Examiner la valeur des principales variables globales
  - Observer/dérouler la pile de chaque tâche

Débogage en fonction du RTOS (3/6)

- En raison de la nature limitée des systèmes embarqués, l'utilisation de la pile est souvent un problème en raison de la mémoire vive limitée des MCU.
- Dans une application bare-metal, il n'y a qu'une seule pile. Dans une application RTOS, chaque tâche possède sa propre pile, ce qui signifie qu'il y a plus de choses à surveiller.
- L'utilisation d'un système de débogage qui fournit des informations sur la pile en fonction du RTOS permet d'évaluer rapidement l'utilisation de la pile de chaque tâche du système.
- Le contrôle des performances les plus défavorables de la réponse à l'événement est également un aspect essentiel du développement des systèmes en temps réel.
- Nous devons veiller à ce que le système réagisse en temps voulu aux événements critiques.

Débogage en fonction du RTOS (4/6)

- Il existe de nombreuses façons d'aborder ce problème.
- En supposant que l'événement trouve son origine dans un signal matériel extérieur au MCU (ce qui est vrai la plupart du temps), un analyseur logique ou un oscilloscope peut être utilisé pour surveiller le signal.
- Un code peut être inséré dans l'application pour faire basculer une broche sur le MCU après que cet événement a été traité et la différence de temps peut être surveillée.
- En fonction du système, de l'accès à l'équipement d'essai et des événements en question, cette méthode centrée sur le matériel peut s'avérer pratique.

Débogage en fonction du RTOS (5/6)

- Une autre méthode consiste à utiliser un logiciel en combinaison avec une instrumentation dans le RTOS.
- Avec cette méthode, de petits crochets sont ajoutés dans le RTOS qui notifient le système de surveillance lorsque des événements se produisent.
- Ces événements sont ensuite transmis par le MCU à un PC de développement qui exécute un programme de visualisation.
- C'est sur cette méthode que nous allons nous concentrer dans ce cours
  - en utilisant SEGGER SystemView.
- Cela permet de collecter une grande quantité d'informations et de statistiques avec très peu d'efforts de développement.

Débogage en fonction du RTOS (6/6)

- Le léger inconvénient de cette méthode est qu'elle ne comporte qu'une très faible part d'incertitude, puisqu'il s'agit d'une approche purement logicielle/firmware.
- Il s'appuie sur le MCU pour enregistrer le moment où les événements se produisent, ce qui signifie que si une interruption est traitée avec un retard important, elle ne sera pas enregistrée avec précision.
- Elle dépend aussi fortement de la disponibilité de la mémoire vive ou des cycles de l'unité centrale.
- Cette approche peut s'avérer peu concluante sur des systèmes lourdement chargés et dépourvus d'une mémoire vive suffisante.
- Toutefois, ces inconvénients peuvent être contournés et ne sont pas rencontrés sur la plupart des systèmes.

Visualisation du RTOS (1/2)

- Il est également important de pouvoir voir quelles tâches sont en cours et comment elles interagissent.
- Dans un environnement de planification préemptive, des relations complexes peuvent se développer entre les tâches.
- Par exemple, pour qu'un événement soit pris en charge, plusieurs tâches doivent interagir les unes avec les autres. En outre, il peut y avoir plusieurs autres tâches qui se disputent le temps du processeur.
- Dans ce cas, un système mal conçu qui ne respecte pas les délais peut être perçu comme lent par l'utilisateur.
- Grâce à la visualisation des tâches, un programmeur peut littéralement voir les relations entre toutes les tâches du système, ce qui facilite considérablement l'analyse.

Visualisation du RTOS (2/2)

- La capacité à discerner facilement l'état dans lequel se trouvent les tâches sur une période donnée est extrêmement utile pour démêler les relations complexes entre les tâches.
- SEGGER SystemView sera également utilisé pour visualiser les relations entre les tâches.

- Afin d'effectuer une analyse approfondie d'un système en fonctionnement, nous avons besoin d'un moyen de nous connecter à l'unité MCU et d'obtenir des informations.
- Sur les MCU Cortex-M, cette opération s'effectue le plus efficacement avec une sonde de débogage externe.

Utilisation de SEGGER J-Link

- Une sonde de débogage est un dispositif qui permet à un ordinateur de communiquer et de programmer la mémoire flash non volatile d'un MCU.
- Il communique avec un matériel spécial sur le MCU (appelé Coresight sur les processeurs ARM Cortex-M).
- Les sondes de débogage SEGGER J-Link et J-Trace sont parmi les plus populaires de l'industrie.
- SEGGER propose également des logiciels utiles qui s'intègrent gratuitement à ses outils.
- L'accessibilité de ces outils et la qualité du logiciel qui les accompagne en font un excellent outil pour cet ouvrage.

Utilisation de SEGGER J-Link

- Il existe de nombreuses options pour choisir les sondes de débogage de SEGGER
  - nous passerons brièvement en revue certaines des options actuellement disponibles et examinerons les exigences matérielles de chacune d'entre elles.

Options matérielles

- SEGGER propose de nombreuses options matérielles qui couvrent une large gamme de prix et de capacités.
- Pour obtenir une liste complète et actualisée, consultez leur site web à l'adresse suivante : https://www.segger.com/products/debug-probes/j-link/models/model-overview/.
- Les modèles se répartissent généralement en deux grandes catégories :
  - débogueurs avec prise en charge complète de Cortex-M Trace et
  - ceux qui n'en ont pas.

Segger J-Trace (1/3)

- Les débogueurs qui prennent en charge l'intégralité des traces sont appelés J-Trace.
- La macrocellule Cortex Embedded Trace (Cortex ETM) est un élément matériel supplémentaire à l'intérieur du MCU qui permet d'enregistrer chaque instruction exécutée.
- La transmission de toutes ces informations à partir du MCU nécessite quelques broches supplémentaires pour l'horloge des données (une ligne d'horloge et 1 à 4 lignes de données).
- La possibilité de retracer chaque instruction exécutée par le MCU permet d'utiliser des fonctionnalités telles que la couverture du code, qui donne un aperçu de la quantité de code exécutée (ligne par ligne).
- Le fait de savoir exactement quelles lignes de code ont été exécutées et à quel moment nous permet de voir où un programme passe le plus clair de son temps.
- Lorsque nous savons quelles lignes de code sont exécutées le plus souvent, il est possible d'optimiser cette petite partie du code lorsqu'une amélioration des performances est nécessaire.

Segger J-Trace (2/3)

- Pour tirer pleinement parti des fonctions de traçage avancées, il est nécessaire de disposer de tous les éléments suivants :
  - Le MCU doit être équipé du matériel ETM.
  - Le boîtier spécifique du MCU doit amener les signaux ETM sur les broches.
  - La configuration périphérique ne doit pas partager les signaux ETM avec d'autres fonctions.
  - Le circuit du système doit être conçu pour intégrer les signaux ETM et un connecteur.

Segger J-Trace (3/3)

- Toutes ces fonctionnalités ont bien sûr un prix.
- Les modèles J-Trace se situent dans le haut de la gamme de SEGGER, à la fois en termes de fonctionnalités et de prix (typiquement plus de 1000 dollars US).
- À moins que vous ne développiez un matériel entièrement personnalisé, attendez-vous également à payer une carte d'évaluation complète (plus de 200 USD) plutôt que le matériel de développement à bas prix utilisé dans ce livre.
- Si ces coûts sont généralement tout à fait raisonnables pour un budget d'ingénierie à part entière lors du développement d'un nouveau produit, ils sont trop onéreux pour être largement accessibles aux particuliers...

SEGGER J-Link (1/2)

- SEGGER J-Link a existé sous différentes formes et s'est développé pour englober plusieurs modèles.
- En général, les modèles haut de gamme offrent des vitesses d'horloge plus élevées et une expérience plus riche (téléchargements plus rapides, débogage réactif, etc.).
- Quelques modèles EDU sont vendus avec un rabais très important à des fins éducatives (d'où la désignation EDU).
- Ces modèles sont pleinement fonctionnels mais ne peuvent être utilisés à des fins commerciales.

SEGGER J-Link (2/2)

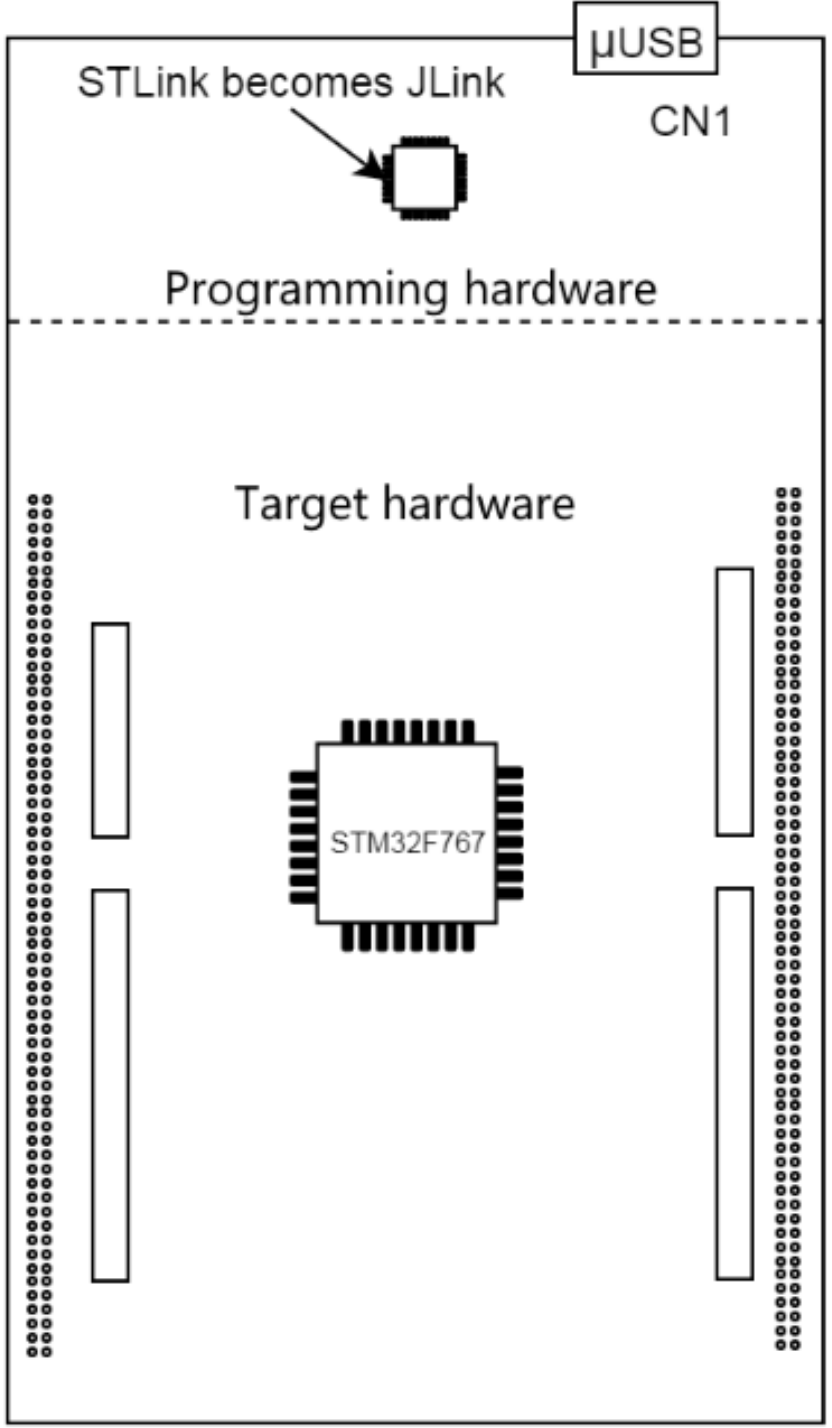
- SEGGER a fait un excellent travail en concevant des interfaces logicielles qui ne sont pas liées au matériel sous-jacent.
- C'est pourquoi leurs outils logiciels fonctionnent sans modification sur différents modèles de débogueurs de matériel.
- C'est ainsi qu'est née l'option matérielle que nous utiliserons dans le cadre de ce cours
  - le SEGGER J-Link à bord.

SEGGER J-Link à bord

- La variante matérielle spécifique de ST-Link que nous utiliserons dans nos exercices n'est pas fabriquée par SEGGER.
- Il s'agit du circuit ST-Link déjà inclus dans la carte de développement Nucleo.
- Les cartes Nucleo comportent deux sous-circuits distincts : le matériel de programmation et le matériel cible.
- Le sous-circuit matériel de programmation est généralement appelé ST-Link.
- Ce matériel de programmation est en fait un autre MCU STM qui est responsable de la communication avec le PC et de la programmation du matériel cible.
  - le Nucleo-f446RE/STM32F767.
- Le matériel Nucleo étant principalement destiné à l'écosystème ARM Mbed, le MCU ST-Link est programmé avec un micrologiciel qui implémente à la fois les fonctionnalités ST-Link et Mbed (diapositive suivante).

J-Link embarqué sur STM Nucleo

Afin d'utiliser le matériel de programmation de la carte Nucleo en tant que SEGGER JLink, nous allons remplacer son firmware par le firmware embarqué SEGGER J-Link.



Installation du J-Link (1/2)

- Des instructions d'installation détaillées sont disponibles auprès de SEGGER à l'adresse https://www.segger.com/products/debug-probes/j-link/models/other-j-links/st-link-on-board/.
- Quelques notes sont également incluses ici pour plus de commodité.
- Afin de convertir le ST-Link embarqué en J-Link, nous allons télécharger et installer deux logiciels :
  - les outils J-Link et l'utilitaire de re-flashage ST-Link.
- Vous devriez déjà avoir les pilotes ST-Link nécessaires installés à partir de l'installation de STM32CubeIDE effectuée dans le chapitre précédent (diapositive suivante).

Installation du J-Link (1/2)

1. Si STM32CubeIDE n'est pas déjà installé, téléchargez et installez les pilotes ST-Link à partir de http://www.st.com/en/development-tools/stsw-link009.html (cette étape est facultative).
2. Téléchargez les utilitaires J-Link appropriés pour votre système d'exploitation à partir de https://www.segger.com/downloads/jlink.
3. Installez les utilitaires J-Link - les options par défaut conviennent.
4. Téléchargez l'utilitaire SEGGER J-Link Reflash (pour Windows OS uniquement) à partir de https://www.segger.com/downloads/jlink#STLink_Reflash.
5. Décompressez le contenu de STLinkReflash_<version>.zip - il contiendra deux fichiers :
   - JLinkARM.dll
   - STLinkReflash.exe

Conversion de ST-Link en J-Link

- Suivez ces étapes pour télécharger le micrologiciel J-Link sur le ST-Link de la carte de développement Nucleo :

  1. Branchez un câble micro USB sur CN1 de la carte Nucleo et reliez-le à votre PC Windows.
  2. Ouvrez STLinkReflash_<version>.exe.
  3. Lisez et acceptez les deux accords de licence.
  4. Sélectionnez la première option : Passer à J-Link.

- Le matériel de débogage sur la carte Nucleo est maintenant un SEGGER J-Link !
- Maintenant qu'un J-Link est présent, nous pourrons utiliser d'autres outils logiciels SEGGER, tels qu'Ozone et SystemView, pour déboguer et visualiser nos applications.

Utilisation de SEGGER SystemView

- SEGGER SystemView est un autre outil logiciel qui peut être utilisé avec les sondes de débogage SEGGER. Il permet de visualiser le flux des tâches et des interruptions dans un système.
- SystemView fonctionne en ajoutant une petite quantité de code dans le projet.
- FreeRTOS dispose déjà de Trace Hook Macros, qui a été spécifiquement conçu pour ajouter ce type de fonctionnalité tierce.
- SystemView n'a aucune capacité de programmation ou de débogage, ce n'est qu'un visualiseur.

Installation de SystemView

- Deux étapes principales sont nécessaires pour rendre votre système visible avec SystemView.
- Le logiciel doit être installé et le code source doit être instrumenté afin qu'il communique son état via l'interface de débogage.

Installation de SystemView

- Pour installer SystemView, procédez comme suit :
  - Téléchargez SystemView pour votre système d'exploitation.
  - Il s'agit de l'installateur binaire principal (https://www.segger.com/downloads/free-utilities).
  - Installer en utilisant les options par défaut.

Configuration du code source

- Pour que SystemView puisse afficher une visualisation des tâches en cours d'exécution sur un système, il faut lui fournir des informations telles que les noms des tâches, les priorités et l'état actuel des tâches.

- Il y a des crochets présents dans FreeRTOS pour presque tout ce dont SystemView a besoin.

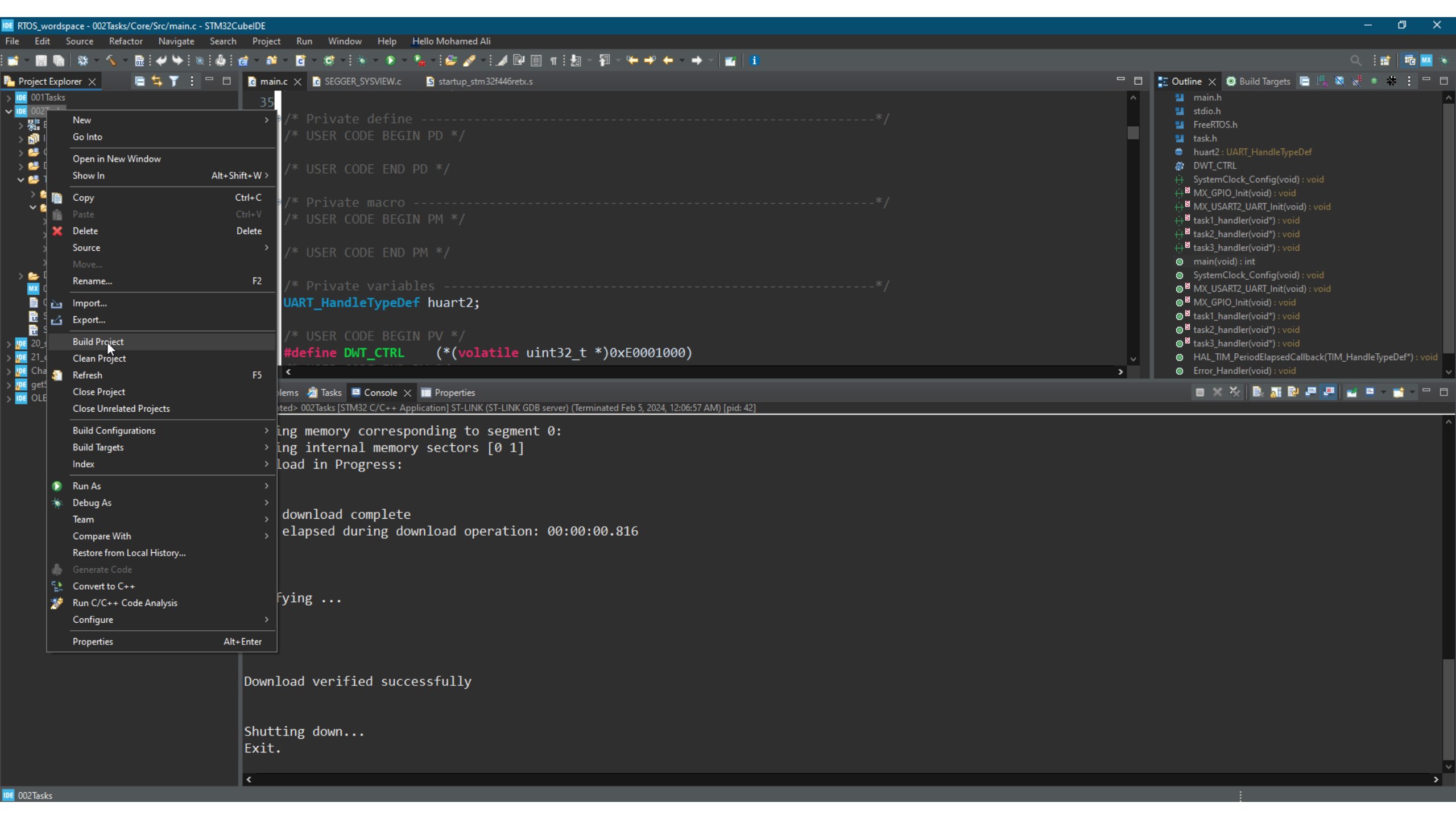
- Quelques fichiers de configuration sont utilisés pour établir une correspondance entre les crochets de trace déjà présents dans FreeRTOS et utilisés par SystemView.

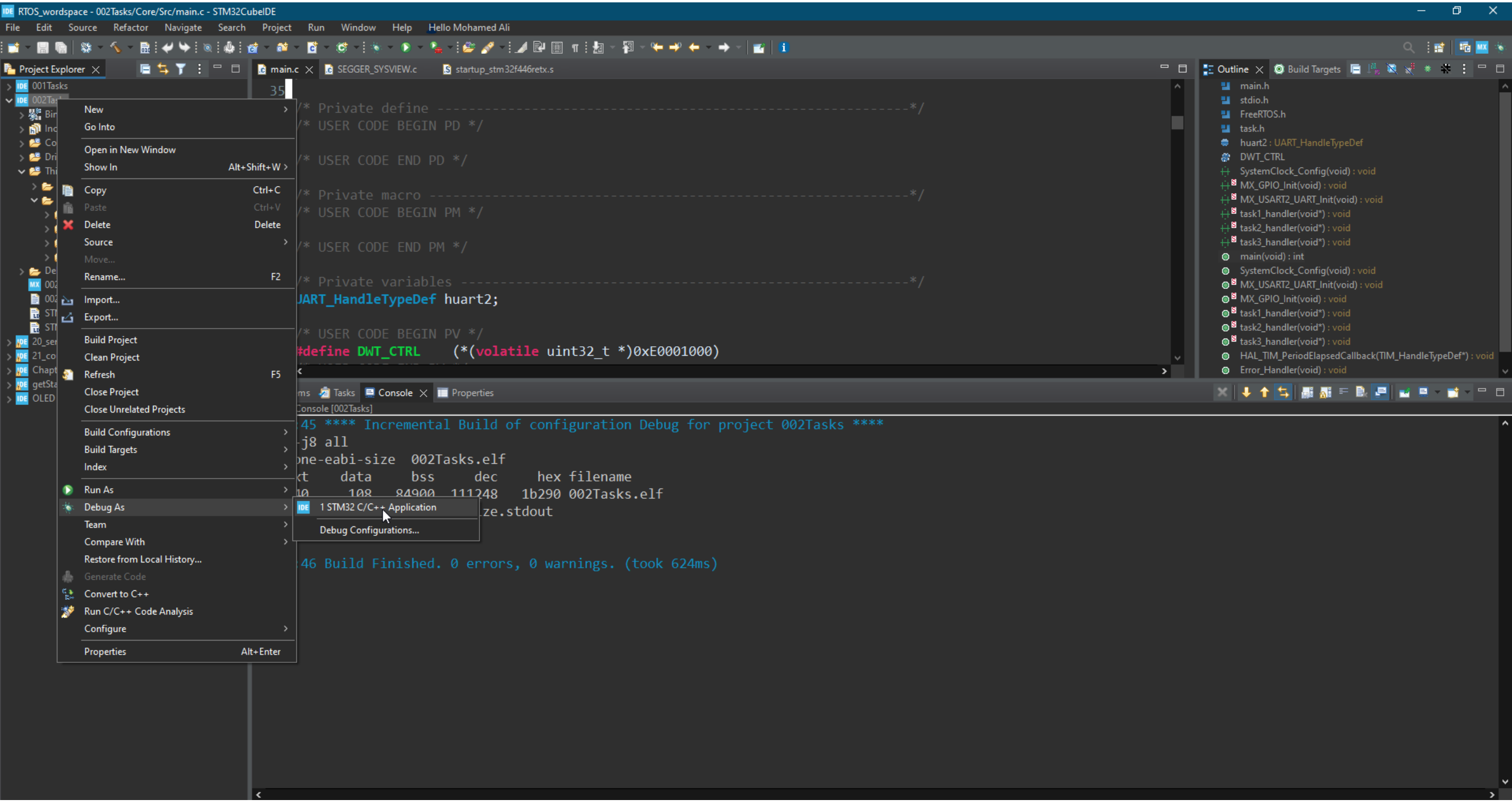
- Les informations doivent être collectées, et c'est là que la configuration spécifique du RTOS et les sources cibles de SystemView entrent en jeu (diapositive suivante).

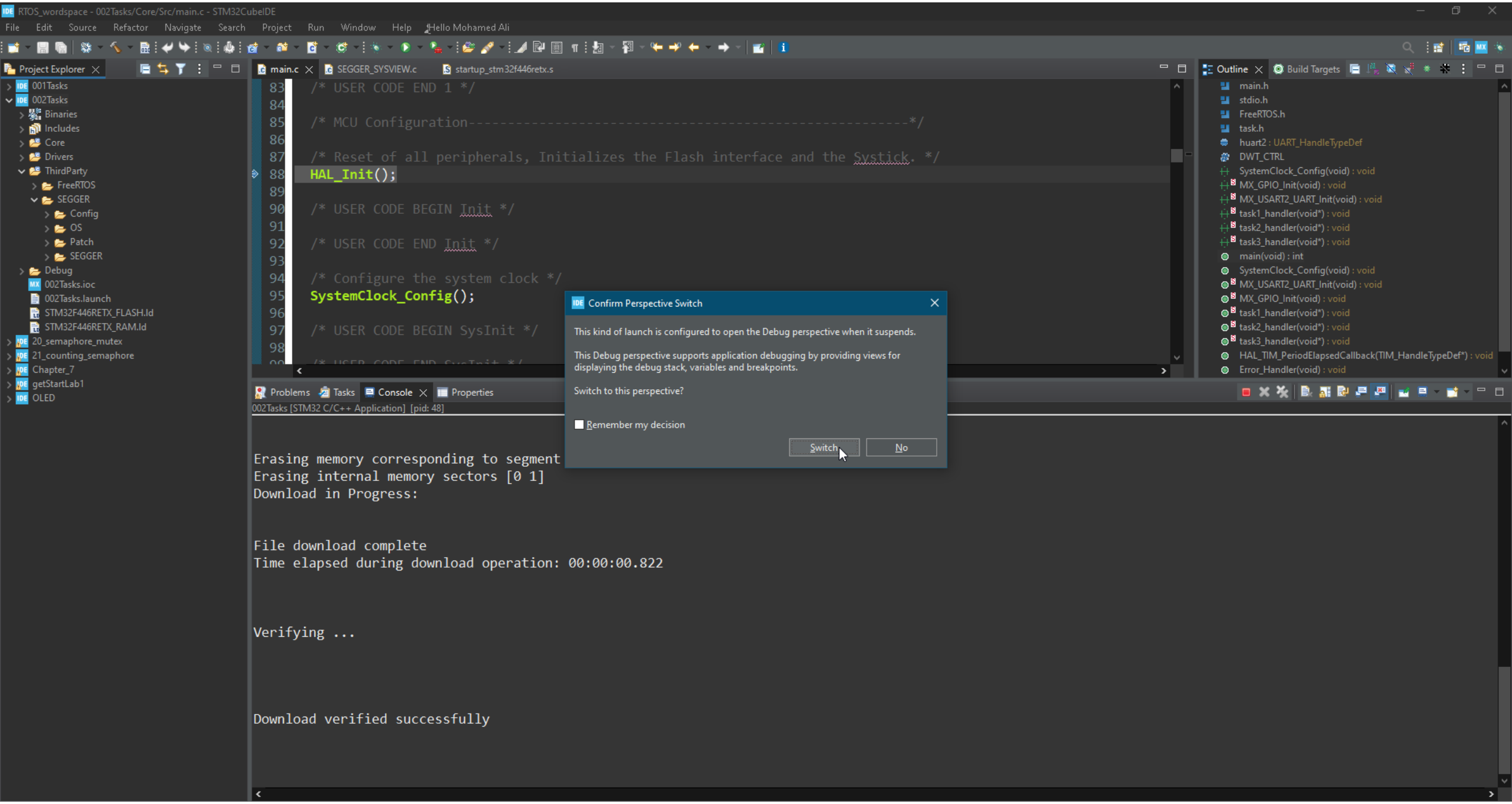
Utilisation de SystemView

Compilation et débogage :

1. S'assurer que les paramètres du chemin d'accès sont définis.
2. Compilez et flashez votre application FreeRTOS et SystemView.
3. Passez en mode de débogage en utilisant votre IDE.
4. Exécuter et faire une pause après quelques secondes.


RTOS_wordspace - 002Tasks/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help Hello Mohamed Ali
Project Explorer
001Tasks
002Tasks
New
Go Into
Open in New Window
Show In Alt+Shift+W
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Build Project
Clean Project
Refresh F5
Close Project
Close Unrelated Projects
Build Configurations
Build Targets
Index
Run As
Debug As
Team
Compare With
Restore from Local History...
Generate Code
Convert to C++
Run C/C++ Code Analysis
Configure
Properties Alt+Enter
main.c
SEGGER_SYSVIEW.c
startup_stm32f446retx.s
/* Private define ----------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro ----------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ----------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
#define DWT_CTRL (*(volatile uint32_t *)0xE0001000)
Outline
Build Targets
main.h
stdio.h
FreeRTOS.h
task.h
huart2 : UART_HandleTypeDef
DWT_CTRL
SystemClock_Config(void) : void
MX_GPIO_Init(void) : void
MX_USART2_UART_Init(void) : void
task1_handler(void*) : void
task2_handler(void*) : void
task3_handler(void*) : void
main(void) : int
HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef*) : void
Error_Handler(void) : void
Tasks
Console
Properties
002Tasks [STM32 C/C++ Application] ST-LINK (ST-LINK GDB server) (Terminated Feb 5, 2024, 12:06:57 AM) [pid: 42]
ing memory corresponding to segment 0:
ing internal memory sectors [0 1]
load in Progress:
download complete
elapsed during download operation: 00:00:00.816
fying ...
Download verified successfully
Shutting down...
Exit.
002Tasks


RTOS_wordspace - 002Tasks/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help Hello Mohamed Ali
Project Explorer
001Tasks
002Tasks
New
Go Into
Open in New Window
Show In Alt+Shift+W
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Build Project
Clean Project
Refresh F5
Close Project
Close Unrelated Projects
Build Configurations
Build Targets
Index
Run As
Debug As
Team
Compare With
Restore from Local History...
Generate Code
Convert to C++
Run C/C++ Code Analysis
Configure
Properties Alt+Enter
1 STM32 C/C++ Application
Debug Configurations...
main.c
SEGGER_SYSVIEW.c
startup_stm32f446retx.s
35
/* Private define ----------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro ----------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ----------*/
UART_HandleTypeDef huart2;
/* USER CODE BEGIN PV */
#define DWT_CTRL (*(volatile uint32_t *)0xE0001000)
Outline
Build Targets
main.h
stdio.h
FreeRTOS.h
task.h
huart2 : UART_HandleTypeDef
DWT_CTRL
SystemClock_Config(void) : void
MX_GPIO_Init(void) : void
MX_USART2_UART_Init(void) : void
task1_handler(void*) : void
task2_handler(void*) : void
task3_handler(void*) : void
main(void) : int
HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef*) : void
Error_Handler(void) : void
Tasks
Console
Properties
Console [002Tasks]
45 **** Incremental Build of configuration Debug for project 002Tasks ****
-j8 all
one-eabi-size 002Tasks.elf
text data bss dec hex filename
10 108 84900 111248 1b290 002Tasks.elf
ze.stdout
46 Build Finished. 0 errors, 0 warnings. (took 624ms)


RTOS_wordspace - 002Tasks/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help Hello Mohamed Ali
Project Explorer
001Tasks
002Tasks
Binaries
Includes
Core
Drivers
ThirdParty
FreeRTOS
SEGGER
Config
OS
Patch
SEGGER
Debug
002Tasks.ioc
002Tasks.launch
STM32F446RETX_FLASH.ld
STM32F446RETX_RAM.ld
20_semaphore_mutex
21_counting_semaphore
Chapter_7
getStartLab1
OLED
main.c
SEGGER_SYSVIEW.c
startup_stm32f446retx.s
83 /* USER CODE END 1 */
84
85 /* MCU Configuration--*/
86
87 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
88 HAL_Init();
89
90 /* USER CODE BEGIN Init */
91
92 /* USER CODE END Init */
93
94 /* Configure the system clock */
95 SystemClock_Config();
96
97 /* USER CODE BEGIN SysInit */
98
Outline
Build Targets
main.h
stdio.h
FreeRTOS.h
task.h
huart2 : UART_HandleTypeDef
DWT_CTRL
SystemClock_Config(void) : void
MX_GPIO_Init(void) : void
MX_USART2_UART_Init(void) : void
task1_handler(void*) : void
task2_handler(void*) : void
task3_handler(void*) : void
main(void) : int
SystemClock_Config(void) : void
MX_USART2_UART_Init(void) : void
MX_GPIO_Init(void) : void
task1_handler(void*) : void
task2_handler(void*) : void
task3_handler(void*) : void
HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef*) : void
Error_Handler(void) : void
Confirm Perspective Switch
This kind of launch is configured to open the Debug perspective when it suspends.
This Debug perspective supports application debugging by providing views for displaying the debug stack, variables and breakpoints.
Switch to this perspective?
Remember my decision
Switch
No
Problems Tasks Console Properties
002Tasks [STM32 C/C++ Application] [pid: 48]
Erasing memory corresponding to segment
Erasing internal memory sectors [0 1]
Download in Progress:
File download complete
Time elapsed during download operation: 00:00:00.822
Verifying ...
Download verified successfully

RTOS_wordspace - 002Tasks/Core/Src/main.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help Hello Mohamed Ali

Debug Proj...

002Tasks [STM32 C/C++ Applic
002Tasks.elf [cores: 0]
Thread #1 [main] 1 [core
main() at main.c:88 0x
arm-none-eabi-gdb (12.1.90
ST-LINK (ST-LINK GDB server

main.c SEGGER_SYSVIEW.c startup_stm32f446retx.s

```
83  /* USER CODE END 1 */
84
85  /* MCU Configuration--------------------------------------------------------*/
86
87  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
88  HAL_Init();
89
90  /* USER CODE BEGIN Init */
91
92  /* USER CODE END Init */
93
94  /* Configure the system clock */
95  SystemClock_Config();
96
97  /* USER CODE BEGIN SysInit */
98
99  /* USER CODE END SysInit */
100
101 /* Initialize all configured peripherals */
102 MX_GPIO_Init();
103 MX_USART2_UART_Init();
104 /* USER CODE BEGIN 2 */
105
```

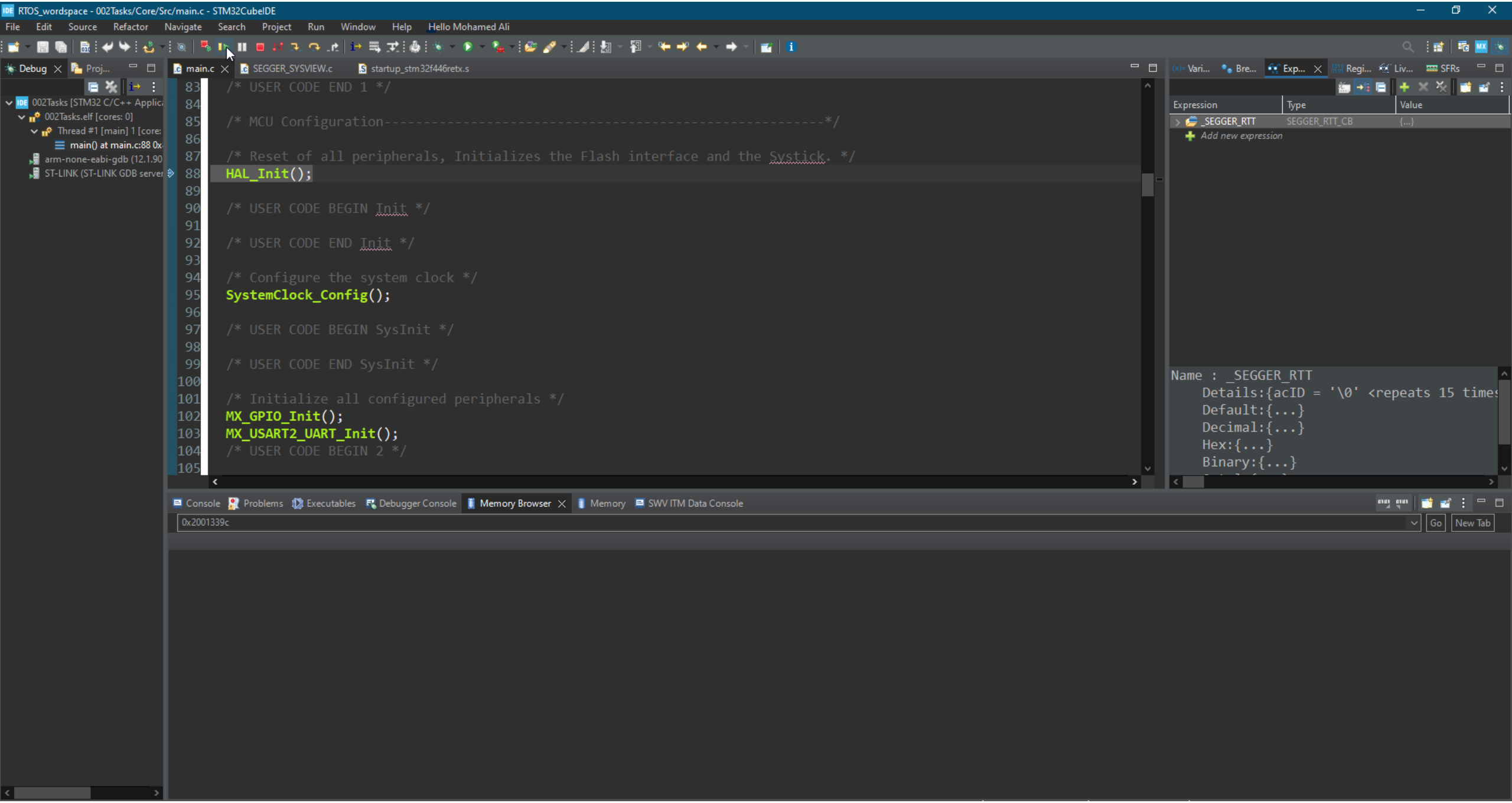
Vari... Bre... Exp... Regi... Liv... SFRs

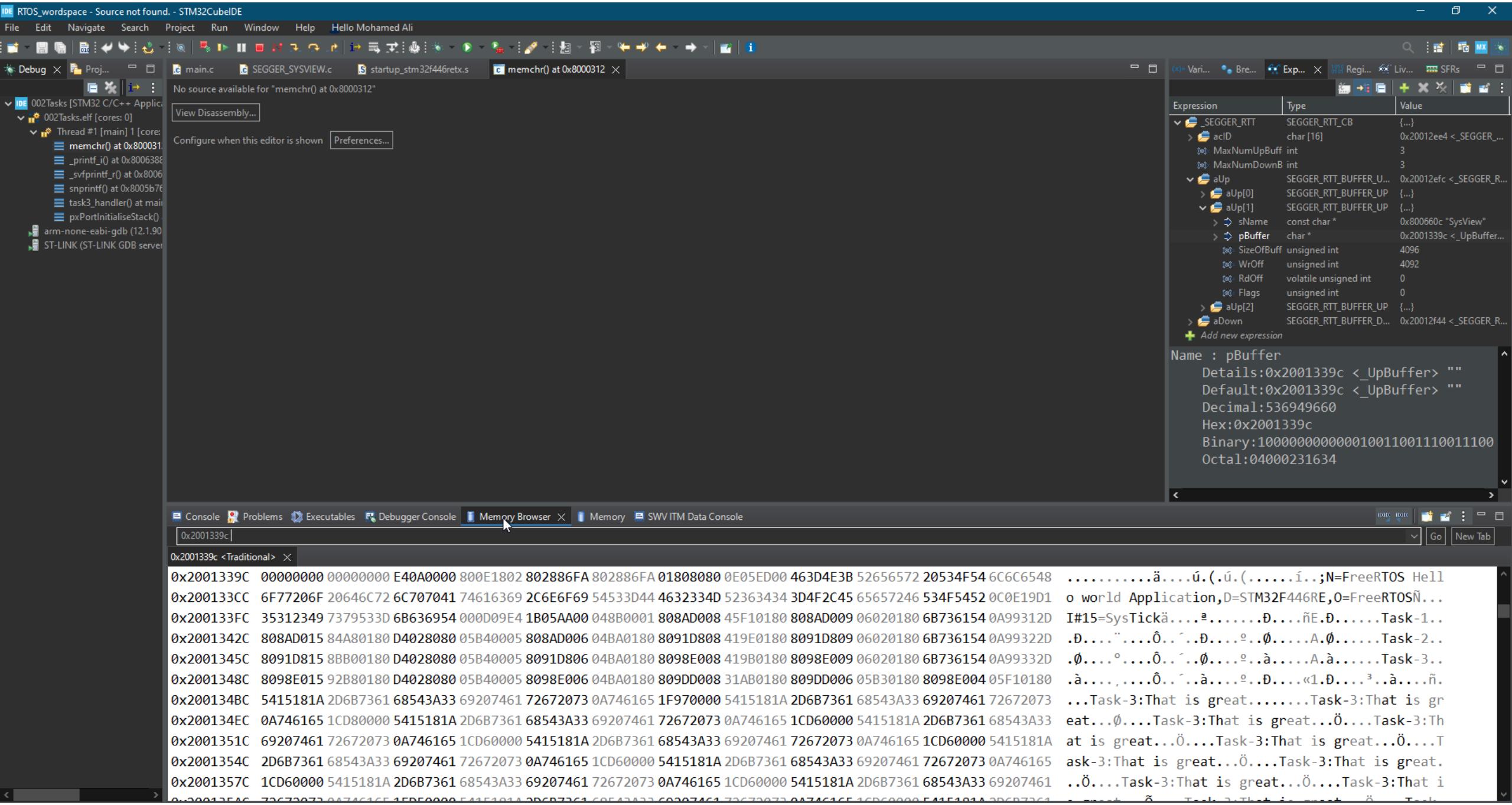
Expression	Type	Value
_SEGGER_RTT	SEGGER_RTT_CB	{...}
Add new expression		

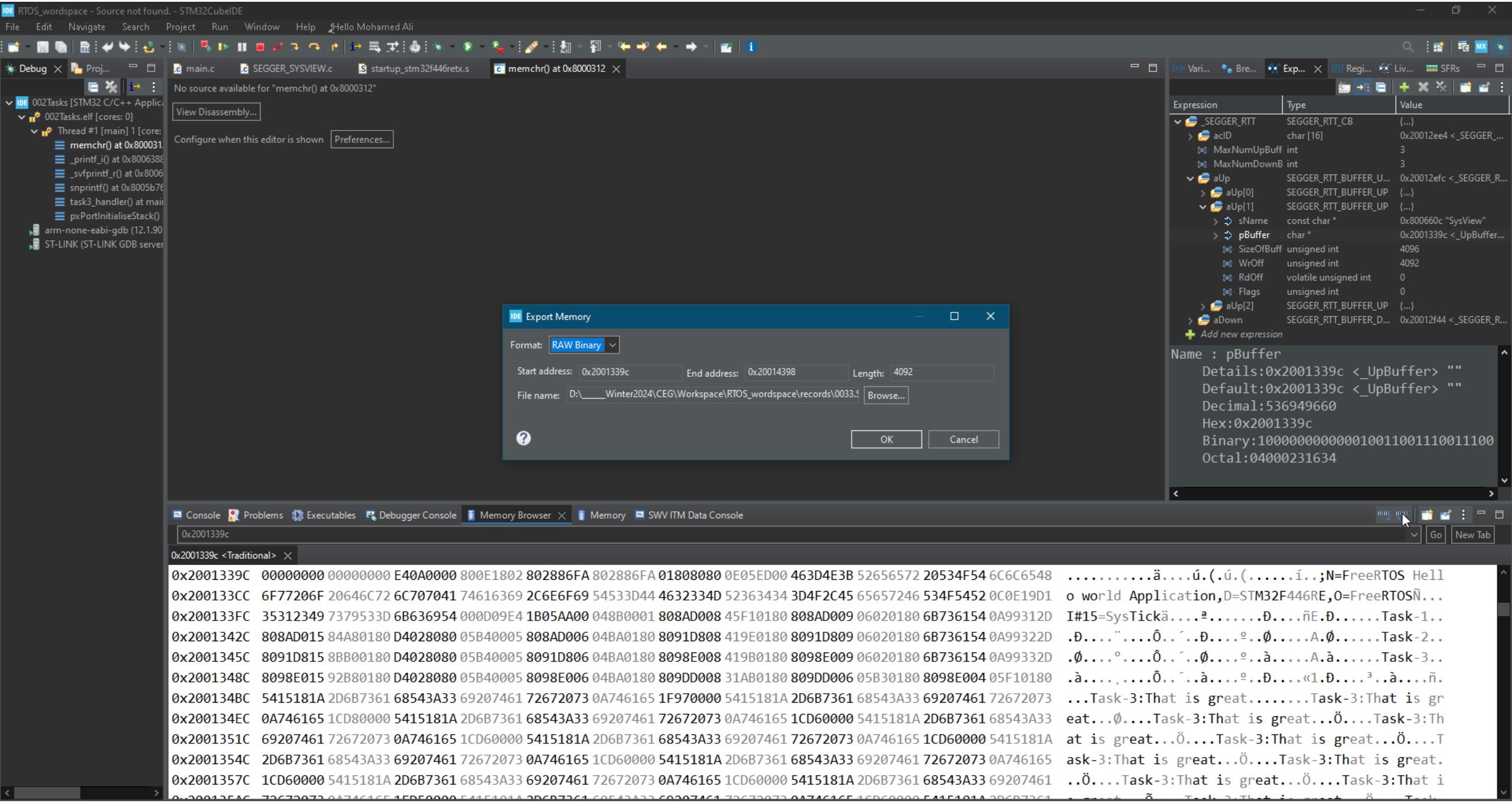
```
Name : _SEGGER_RTT
    Details:{acID = '\0' <repeats 15 times
    Default:{...}
    Decimal:{...}
    Hex:{...}
    Binary:{...}
```

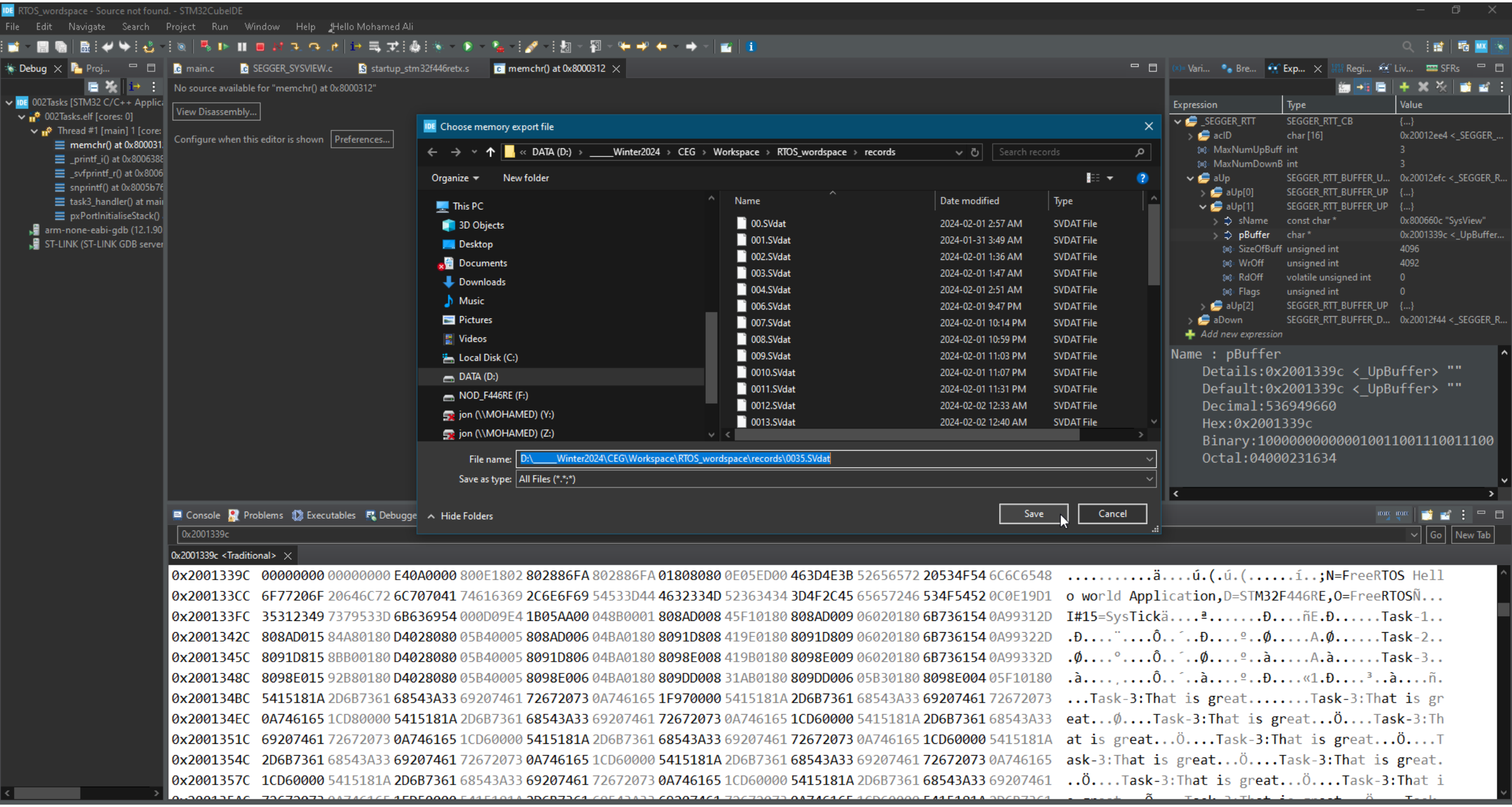
Console Problems Executables Debugger Console Memory Browser Memory SWV ITM Data Console

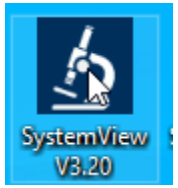
0x2001339c Go New Tab

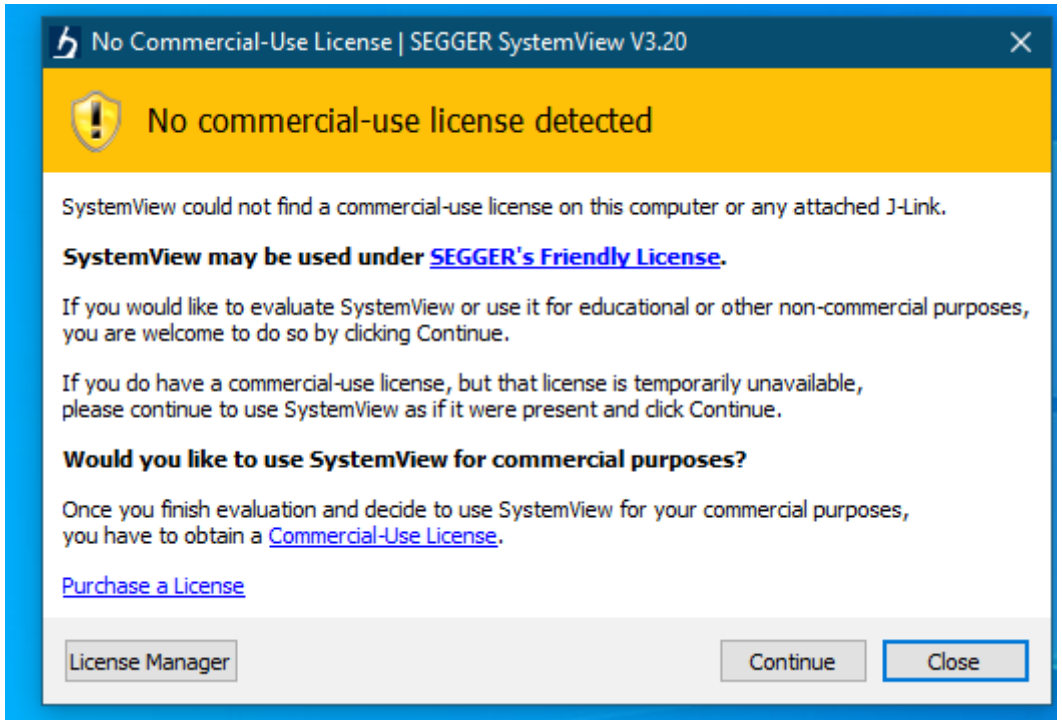

RTOS_wordspace - 002Tasks/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help Hello Mohamed Ali
Debug
Proj...
002Tasks [STM32 C/C++ Applic
002Tasks.elf [cores: 0]
Thread #1 [main] 1 [core:
main() at main.c:88 0x
arm-none-eabi-gdb (12.1.90
ST-LINK (ST-LINK GDB server
main.c
SEGGER_SYSVIEW.c
startup_stm32f446retx.s
83 /* USER CODE END 1 */
84
85 /* MCU Configuration--*/
86
87 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
88 HAL_Init();
89
90 /* USER CODE BEGIN Init */
91
92 /* USER CODE END Init */
93
94 /* Configure the system clock */
95 SystemClock_Config();
96
97 /* USER CODE BEGIN SysInit */
98
99 /* USER CODE END SysInit */
100
101 /* Initialize all configured peripherals */
102 MX_GPIO_Init();
103 MX_USART2_UART_Init();
104 /* USER CODE BEGIN 2 */
105
Vari...
Bre...
Exp...
Regi...
Liv...
SFRs
Expression
Type
Value
_SEGGER_RTT
SEGGER_RTT_CB
{...}
Add new expression
Name : _SEGGER_RTT
Details:{acID = '\0' <repeats 15 times
Default:{...}
Decimal:{...}
Hex:{...}
Binary:{...}
Console
Problems
Executables
Debugger Console
Memory Browser
Memory
SWV ITM Data Console
0x2001339c
Go
New Tab

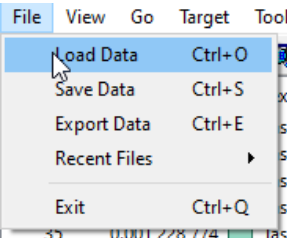

RTOS_wordspace - Source not found. - STM32CubeIDE
File Edit Navigate Search Project Run Window Help Hello Mohamed Ali
Debug
002Tasks [STM32 C/C++ Applic
002Tasks.elf [cores: 0]
Thread #1 [main] 1 [core:
memchr() at 0x800031
_printf_i() at 0x8006388
_svfprintf_r() at 0x8006
snprintf() at 0x8005b76
task3_handler() at mai
pxPortInitialiseStack()
arm-none-eabi-gdb (12.1.90
ST-LINK (ST-LINK GDB server
main.c
SEGGER_SYSVIEW.c
startup_stm32f446retx.s
memchr() at 0x8000312
No source available for "memchr() at 0x8000312"
View Disassembly...
Configure when this editor is shown
Preferences...
Vari...
Bre...
Exp...
Regi...
Liv...
SFRs
Expression
Type
Value
_SEGGER_RTT
SEGGER_RTT_CB
acID
char [16]
0x20012ee4 <_SEGGER_...
MaxNumUpBuff
int
3
MaxNumDownB
int
3
aUp
SEGGER_RTT_BUFFER_U...
0x20012efc <_SEGGER_R...
aUp[0]
SEGGER_RTT_BUFFER_UP
aUp[1]
SEGGER_RTT_BUFFER_UP
sName
const char *
0x800660c "SysView"
pBuffer
char *
0x2001339c <_UpBuffer...
SizeOfBuff
unsigned int
4096
WrOff
unsigned int
4092
RdOff
volatile unsigned int
0
Flags
unsigned int
0
aUp[2]
SEGGER_RTT_BUFFER_UP
aDown
SEGGER_RTT_BUFFER_D...
0x20012f44 <_SEGGER_R...
Add new expression
Name : pBuffer
Details:0x2001339c <_UpBuffer> ""
Default:0x2001339c <_UpBuffer> ""
Decimal:536949660
Hex:0x2001339c
Binary:10000000000001001100111001100
Octal:04000231634
Console
Problems
Executables
Debugger Console
Memory Browser
Memory
SWV ITM Data Console
0x2001339c
Go
New Tab
0x2001339c <Traditional>
0x2001339C 00000000 00000000 E40A0000 800E1802 802886FA 802886FA 01808080 0E05ED00 463D4E3B 52656572 20534F54 6C6C6548
0x200133CC 6F77206F 20646C72 6C707041 74616369 2C6E6F69 54533D44 4632334D 52363434 3D4F2C45 65657246 534F5452 0C0E19D1
0x200133FC 35312349 7379533D 6B636954 000D09E4 1B05AA00 048B0001 808AD008 45F10180 808AD009 06020180 6B736154 0A99312D
0x2001342C 808AD015 84A80180 D4028080 05B40005 808AD006 04BA0180 8091D808 419E0180 8091D809 06020180 6B736154 0A99322D
0x2001345C 8091D815 8BB00180 D4028080 05B40005 8091D806 04BA0180 8098E008 419B0180 8098E009 06020180 6B736154 0A99332D
0x2001348C 8098E015 92B80180 D4028080 05B40005 8098E006 04BA0180 809DD008 31AB0180 809DD006 05B30180 8098E004 05F10180
0x200134BC 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1F970000 5415181A 2D6B7361 68543A33 69207461 72672073
0x200134EC 0A746165 1CD80000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33
0x2001351C 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A
0x2001354C 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165
0x2001357C 1CD60000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33 69207461


RTOS_wordspace - Source not found. - STM32CubeIDE
File Edit Navigate Search Project Run Window Help Hello Mohamed Ali
Debug
002Tasks [STM32 C/C++ Applica
002Tasks.elf [cores: 0]
Thread #1 [main] 1 [core:
memchr() at 0x800031
_printf_i() at 0x800638
_svfprintf_r() at 0x8006
snprintf() at 0x8005b76
task3_handler() at mai
pxPortInitialiseStack()
arm-none-eabi-gdb (12.1.90
ST-LINK (ST-LINK GDB server
main.c
SEGGER_SYSVIEW.c
startup_stm32f446retx.s
memchr() at 0x8000312
No source available for "memchr() at 0x8000312"
View Disassembly...
Configure when this editor is shown
Preferences...
Export Memory
Format: RAW Binary
Start address: 0x2001339c
End address: 0x20014398
Length: 4092
File name: D:____Winter2024\CEG\Workspace\RTOS_wordspace\records\0033.
Browse...
OK
Cancel
Vari...
Bre...
Exp...
Regi...
Liv...
SFRs
Expression
Type
Value
_SEGGER_RTT
SEGGER_RTT_CB
acID
char [16]
0x20012ee4 <_SEGGER_...
MaxNumUpBuff
int
3
MaxNumDownB
int
3
aUp
SEGGER_RTT_BUFFER_U...
0x20012efc <_SEGGER_R...
aUp[0]
SEGGER_RTT_BUFFER_UP
aUp[1]
SEGGER_RTT_BUFFER_UP
sName
const char *
0x800660c "SysView"
pBuffer
char *
0x2001339c <_UpBuffer...
SizeOfBuff
unsigned int
4096
WrOff
unsigned int
4092
RdOff
volatile unsigned int
0
Flags
unsigned int
0
aUp[2]
SEGGER_RTT_BUFFER_UP
aDown
SEGGER_RTT_BUFFER_D...
0x20012f44 <_SEGGER_R...
Add new expression
Name : pBuffer
Details:0x2001339c <_UpBuffer> ""
Default:0x2001339c <_UpBuffer> ""
Decimal:536949660
Hex:0x2001339c
Binary:10000000000001001100111001100
Octal:04000231634
Console
Problems
Executables
Debugger Console
Memory Browser
Memory
SWV ITM Data Console
0x2001339c
Go
New Tab
0x2001339c <Traditional>
0x2001339C 00000000 00000000 E40A0000 800E1802 802886FA 802886FA 01808080 0E05ED00 463D4E3B 52656572 20534F54 6C6C6548
0x200133CC 6F77206F 20646C72 6C707041 74616369 2C6E6F69 54533D44 4632334D 52363434 3D4F2C45 65657246 534F5452 0C0E19D1
0x200133FC 35312349 7379533D 6B636954 000D09E4 1B05AA00 048B0001 808AD008 45F10180 808AD009 06020180 6B736154 0A99312D
0x2001342C 808AD015 84A80180 D4028080 05B40005 808AD006 04BA0180 8091D808 419E0180 8091D809 06020180 6B736154 0A99322D
0x2001345C 8091D815 8BB00180 D4028080 05B40005 8091D806 04BA0180 8098E008 419B0180 8098E009 06020180 6B736154 0A99332D
0x2001348C 8098E015 92B80180 D4028080 05B40005 8098E006 04BA0180 809DD008 31AB0180 809DD006 05B30180 8098E004 05F10180
0x200134BC 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1F970000 5415181A 2D6B7361 68543A33 69207461 72672073
0x200134EC 0A746165 1CD80000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33
0x2001351C 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A
0x2001354C 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165
0x2001357C 1CD60000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33 69207461


RTOS_wordspace - Source not found. - STM32CubeIDE
File Edit Navigate Search Project Run Window Help Hello Mohamed Ali
Debug
002Tasks [STM32 C/C++ Applic
002Tasks.elf [cores: 0]
Thread #1 [main] 1 [core
memchr() at 0x800031
_printf_i() at 0x800638
_svfprintf_r() at 0x8006
snprintf() at 0x8005b76
task3_handler() at mai
pxPortInitialiseStack()
arm-none-eabi-gdb (12.1.90
ST-LINK (ST-LINK GDB server
main.c
SEGGER_SYSVIEW.c
startup_stm32f446retx.s
memchr() at 0x8000312
No source available for "memchr() at 0x8000312"
View Disassembly...
Configure when this editor is shown
Preferences...
Choose memory export file
DATA (D:) › ____Winter2024 › CEG › Workspace › RTOS_wordspace › records
Search records
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
DATA (D:)
NOD_F446RE (F:)
jon (\\MOHAMED) (Y:)
jon (\\MOHAMED) (Z:)
Name
Date modified
Type
00.SVdat 2024-02-01 2:57 AM SVDAT File
001.SVdat 2024-01-31 3:49 AM SVDAT File
002.SVdat 2024-02-01 1:36 AM SVDAT File
003.SVdat 2024-02-01 1:47 AM SVDAT File
004.SVdat 2024-02-01 2:51 AM SVDAT File
006.SVdat 2024-02-01 9:47 PM SVDAT File
007.SVdat 2024-02-01 10:14 PM SVDAT File
008.SVdat 2024-02-01 10:59 PM SVDAT File
009.SVdat 2024-02-01 11:03 PM SVDAT File
0010.SVdat 2024-02-01 11:07 PM SVDAT File
0011.SVdat 2024-02-01 11:31 PM SVDAT File
0012.SVdat 2024-02-02 12:33 AM SVDAT File
0013.SVdat 2024-02-02 12:40 AM SVDAT File
File name: D:____Winter2024\CEG\Workspace\RTOS_wordspace\records\0035.SVdat
Save as type: All Files (*.*;*)
Hide Folders
Save
Cancel
Expression
Type
Value
_SEGGER_RTT SEGGER_RTT_CB {...}
acID char [16] 0x20012ee4 <_SEGGER_...
MaxNumUpBuff int 3
MaxNumDownB int 3
aUp SEGGER_RTT_BUFFER_U... 0x20012efc <_SEGGER_R...
aUp[0] SEGGER_RTT_BUFFER_UP {...}
aUp[1] SEGGER_RTT_BUFFER_UP {...}
sName const char * 0x800660c "SysView"
pBuffer char * 0x2001339c <_UpBuffer...
SizeOfBuff unsigned int 4096
WrOff unsigned int 4092
RdOff volatile unsigned int 0
Flags unsigned int 0
aUp[2] SEGGER_RTT_BUFFER_UP {...}
aDown SEGGER_RTT_BUFFER_D... 0x20012f44 <_SEGGER_R...
Add new expression
Name : pBuffer
Details:0x2001339c <_UpBuffer> ""
Default:0x2001339c <_UpBuffer> ""
Decimal:536949660
Hex:0x2001339c
Binary:10000000000001001100111001100
Octal:04000231634
Console Problems Executables Debugge
0x2001339c
Go
New Tab
0x2001339c <Traditional>
0x2001339C 00000000 00000000 E40A0000 800E1802 802886FA 802886FA 01808080 0E05ED00 463D4E3B 52656572 20534F54 6C6C6548
0x200133CC 6F77206F 20646C72 6C707041 74616369 2C6E6F69 54533D44 4632334D 52363434 3D4F2C45 65657246 534F5452 0C0E19D1
0x200133FC 35312349 7379533D 6B636954 000D09E4 1B05AA00 048B0001 808AD008 45F10180 808AD009 06020180 6B736154 0A99312D
0x2001342C 808AD015 84A80180 D4028080 05B40005 808AD006 04BA0180 8091D808 419E0180 8091D809 06020180 6B736154 0A99322D
0x2001345C 8091D815 8BB00180 D4028080 05B40005 8091D806 04BA0180 8098E008 419B0180 8098E009 06020180 6B736154 0A99332D
0x2001348C 8098E015 92B80180 D4028080 05B40005 8098E006 04BA0180 809DD008 31AB0180 809DD006 05B30180 8098E004 05F10180
0x200134BC 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1F970000 5415181A 2D6B7361 68543A33 69207461 72672073
0x200134EC 0A746165 1CD80000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33
0x2001351C 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A
0x2001354C 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165
0x2001357C 1CD60000 5415181A 2D6B7361 68543A33 69207461 72672073 0A746165 1CD60000 5415181A 2D6B7361 68543A33 69207461

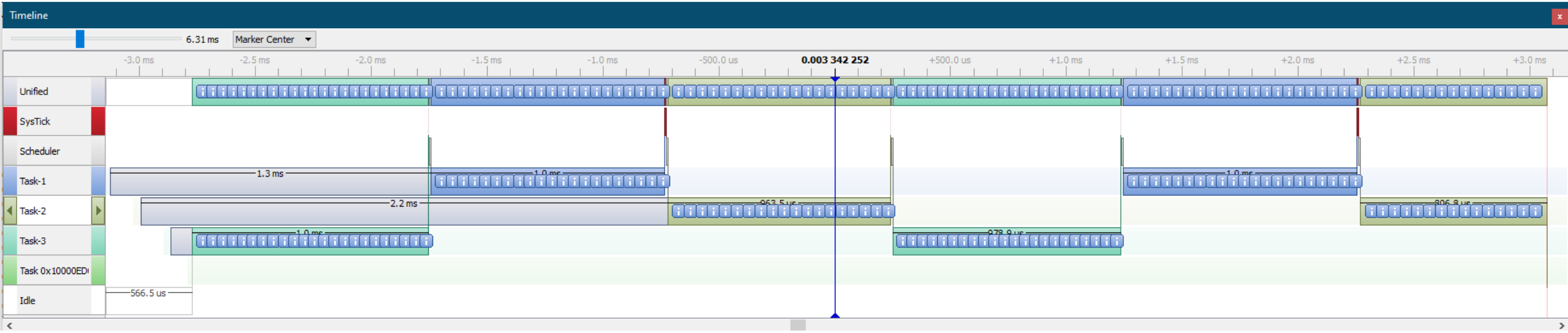

SystemView
V3.20


No Commercial-Use License | SEGGER SystemView V3.20
No commercial-use license detected
SystemView could not find a commercial-use license on this computer or any attached J-Link.
SystemView may be used under SEGGER's Friendly License.
If you would like to evaluate SystemView or use it for educational or other non-commercial purposes, you are welcome to do so by clicking Continue.
If you do have a commercial-use license, but that license is temporarily unavailable, please continue to use SystemView as if it were present and click Continue.
Would you like to use SystemView for commercial purposes?
Once you finish evaluation and decide to use SystemView for your commercial purposes, you have to obtain a Commercial-Use License.
Purchase a License
License Manager
Continue
Close

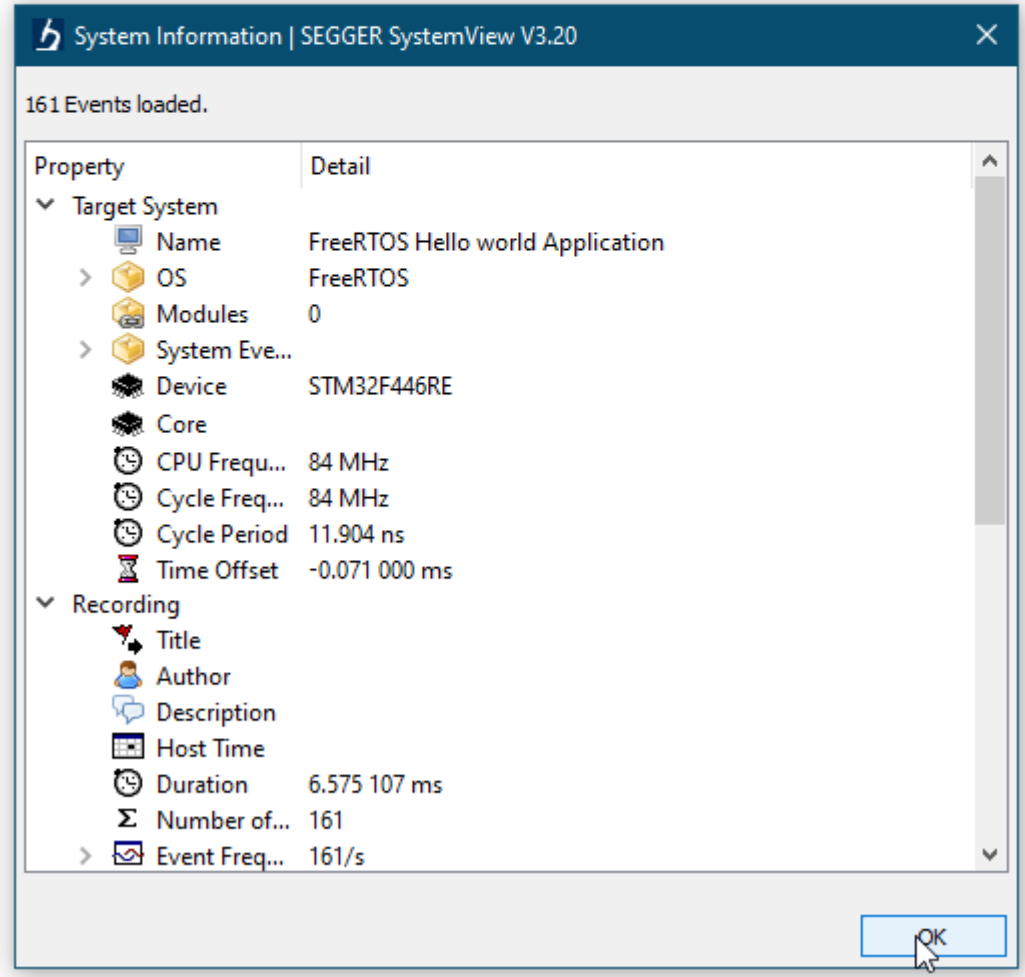


			Event	Detail
		sk-3	Log	Task-3:That is great
		sk-3	Log	Task-3:That is great
		sk-3	Log	Task-3:That is great
		sk-3	Log	Task-3:That is great
35	0.001 228 774	Task-3	Log	Task-3:That is great
36	0.001 272 464	Task-3	Log	Task-3:That is great
37	0.001 316 155	Task-3	Log	Task-3:That is great
38	0.001 359 845	Task-3	Log	Task-3:That is great
39	0.001 403 536	Task-3	Log	Task-3:That is great
40	0.001 447 226	Task-3	Log	Task-3:That is great
41	0.001 490 917	Task-3	Log	Task-3:That is great
42	0.001 534 607	Task-3	Log	Task-3:That is great
43	0.001 578 298	Task-3	Log	Task-3:That is great
44	0.001 584 143	SysTick	ISR Enter	Runs for 5.345 us
45	0.001 589 488	SysTick	ISR Exit	Returns to Scheduler
46	0.001 597 500	Task-1	Task Run	Runs for 1.013 ms
47	0.001 644 738	Task-1	Log	Task-1: hi how are you?
48	0.001 692 357	Task-1	Log	Task-1: hi how are you?
49	0.001 739 952	Task-1	Log	Task-1: hi how are you?
50	0.001 787 548	Task-1	Log	Task-1: hi how are you?
51	0.001 835 143	Task-1	Log	Task-1: hi how are you?
52	0.001 882 738	Task-1	Log	Task-1: hi how are you?
53	0.001 930 333	Task-1	Log	Task-1: hi how are you?
54	0.001 980 976	Task-1	Log	Task-1: hi how are you?
55	0.002 028 571	Task-1	Log	Task-1: hi how are you?
56	0.002 076 167	Task-1	Log	Task-1: hi how are you?
57	0.002 123 762	Task-1	Log	Task-1: hi how are you?
58	0.002 171 357	Task-1	Log	Task-1: hi how are you?
59	0.002 218 952	Task-1	Log	Task-1: hi how are you?
60	0.002 266 548	Task-1	Log	Task-1: hi how are you?
61	0.002 314 143	Task-1	Log	Task-1: hi how are you?
62	0.002 361 738	Task-1	Log	Task-1: hi how are you?
63	0.002 409 333	Task-1	Log	Task-1: hi how are you?
64	0.002 456 929	Task-1	Log	Task-1: hi how are you?
65	0.002 504 524	Task-1	Log	Task-1: hi how are you?
66	0.002 552 119	Task-1	Log	Task-1: hi how are you?
67	0.002 599 714	Task-1	Log	Task-1: hi how are you?
68	0.002 605 726	SysTick	ISR Enter	Runs for 5.357 us
69	0.002 611 083	SysTick	ISR Exit	Returns to Scheduler
70	0.002 619 095	Task-2	Task Run	Runs for 968.929 us

Chronologie



#	Time	Context	Event	Detail
0	0.000 000 000	Idle	Start	
1	0.000 008 917	Idle	Init	Cycle Freq.: 84000000, CPU Freq.: 84000000, ID Base: 0x10000000, ID Shift: 0
2	0.000 047 976	Idle	System Description	N=FreeRTOS Hello world Application,D=STM32F446RE,O=FreeRTOS
3	0.000 062 881	Idle	System Description	I#15=SysTick
4	0.000 071 000	Idle	System Time (us)	0 us
5	0.000 077 226	Idle	Num Modules	Registered Modules: 0
6	0.000 183 714	Idle	Task Create	Task-1 (0x10000550)
7	0.000 199 250	Idle	Task Info	Task-1 (0x10000550): Priority = 2
8	0.000 207 488	Idle	Stack Info	Task-1 (0x10000550): 724 @ 0x20000228
9	0.000 214 274	Idle	Task Ready	Task-1, runs after 1.383 ms
10	0.000 313 679	Idle	Task Create	Task-2 (0x100008D8)
11	0.000 329 214	Idle	Task Info	Task-2 (0x100008D8): Priority = 2
12	0.000 337 452	Idle	Stack Info	Task-2 (0x100008D8): 724 @ 0x200005B0
13	0.000 344 238	Idle	Task Ready	Task-2, runs after 2.274 ms
14	0.000 443 607	Idle	Task Create	Task-3 (0x10000C60)
15	0.000 459 143	Idle	Task Info	Task-3 (0x10000C60): Priority = 2
16	0.000 467 381	Idle	Stack Info	Task-3 (0x10000C60): 724 @ 0x20000938
17	0.000 474 167	Idle	Task Ready	Task-3, runs after 92.369 us
18	0.000 549 345	Idle	Task Create	Task 0x10000ED0 (0x10000ED0)
19	0.000 557 571	Idle	Task Ready	Task 0x10000ED0
20	0.000 566 536	Task-3	Task Run	Runs for 1.022 ms
21	0.000 614 048	Task-3	Log	Task-3:That is great
22	0.000 657 762	Task-3	Log	Task-3:That is great
23	0.000 701 452	Task-3	Log	Task-3:That is great
24	0.000 745 143	Task-3	Log	Task-3:That is great
25	0.000 788 833	Task-3	Log	Task-3:That is great
26	0.000 832 524	Task-3	Log	Task-3:That is great
27	0.000 876 214	Task-3	Log	Task-3:That is great
28	0.000 919 905	Task-3	Log	Task-3:That is great
29	0.000 966 631	Task-3	Log	Task-3:That is great
30	0.001 010 321	Task-3	Log	Task-3:That is great
31	0.001 054 012	Task-3	Log	Task-3:That is great
32	0.001 097 702	Task-3	Log	Task-3:That is great
33	0.001 141 393	Task-3	Log	Task-3:That is great
34	0.001 185 083	Task-3	Log	Task-3:That is great
35	0.001 228 774	Task-3	Log	Task-3:That is great
36	0.001 272 464	Task-3	Log	Task-3:That is great
37	0.001 316 155	Task-3	Log	Task-3:That is great
38	0.001 359 845	Task-3	Log	Task-3:That is great
39	0.001 403 536	Task-3	Log	Task-3:That is great



Temps d'exécution

Runtime

	Min	25%	50%	75%	Max	
SysTick	5.345 us	5.345 us	5.357 us	5.357 us	5.524 us	
Scheduler	8.012 us	8.012 us	8.012 us	8.012 us	8.143 us	
Task-1	1.008 ms	0.000 ms	1.007 ms	1.007 ms	1.008 ms	
Task-2	806.869 us	0.000 us	805.952 us	805.952 us	963.571 us	
Task-3	0.978 ms	0.000 ms	0.978 ms	0.978 ms	1.017 ms	
Task 0x10000ED0	0 ns	0 ns	0 ns	0 ns	0 ns	No runtime!
Idle	0 ns	0 ns	0 ns	0 ns	0 ns	No runtime!

Contextes

Contexts

Name	Type	Stack Information	Activations	Total Blocked Time	Total Run Time	Time Interrupted	CPU Load	Last Run Time	Min Run Time	Max Run Time	Min Blocked Time	Max Blocked Time	Run Time/s	Min Run Time/s	Max Run Time/s
SysTick	#15		5		0.000 026 940 s	0.000 000 ms	0.41 %	0.005 357 ms	0.005 345 ms	0.005 524 ms			0.026 940 ms	0.000 000 ms	0.000 000 ms
Scheduler			5		0.000 040 190 s	0.000 000 ms	0.61 %	0.008 012 ms	0.008 012 ms	0.008 143 ms			0.040 190 ms	0.000 000 ms	0.000 000 ms
Task-1	@2	724 @ 0x20000228	2	0.001 383 226 s	0.002 027 333 s	0.010 714 ms	30.67 %	1.008 393 ms	1.008 226 ms	1.008 393 ms	1.383 226 ms	1.383 226 ms	2.016 619 ms	0.000 000 ms	0.000 000 ms
Task-2	@2	724 @ 0x200005B0	2	0.002 274 857 s	0.001 775 798 s	0.005 357 ms	26.93 %	0.806 869 ms	0.806 869 ms	0.963 571 ms	2.274 857 ms	2.274 857 ms	1.770 440 ms	0.000 000 ms	0.000 000 ms
Task-3	@2	724 @ 0x20000938	2	0.000 092 369 s	0.002 007 417 s	0.010 869 ms	30.37 %	0.978 940 ms	0.978 940 ms	1.017 607 ms	0.092 369 ms	0.092 369 ms	1.996 548 ms	0.000 000 ms	0.000 000 ms
Task 0x10000ED0	@0	0 @ 0x00000000	1	0.000 000 000 s	0.000 000 000 s	0.000 000 ms	0.00 %	0.000 000 ms	0.000 000 ms	0.000 000 ms	0.000 000 ms	0.000 000 ms	0.000 000 ms	0.000 000 ms	0.000 000 ms
Idle			0		0.000 000 000 s	0.000 000 ms	0.00 %	0.000 000 ms	0.000 000 ms	0.000 000 ms			0.000 000 ms	0.000 000 ms	0.000 000 ms

161 Events | 6.575 107 ms | Loaded

Résumé

- Dans ce chapitre, nous avons expliqué pourquoi il est important d'avoir accès à d'excellents outils de débogage.
- Les outils exacts que nous utiliserons pour analyser le comportement du système (SystemView) ont été présentés.
- Vous avez également été guidé dans la mise en place de ces outils en vue de leur utilisation dans le cadre de projets futurs.
- Vers la fin, nous avons abordé quelques autres outils qui ne seront pas traités dans ce livre, simplement pour les faire connaître.
- Maintenant que nous avons abordé le choix du MCU et de l'IDE, et que nous disposons de tous les outils nécessaires, nous avons suffisamment de connaissances pour entrer dans le vif du sujet, à savoir le développement d'applications RTOS.
- L'utilisation de cet ensemble d'outils vous aidera à acquérir une compréhension approfondie du comportement et de la programmation du RTOS, alors que nous nous plongerons dans des exemples pratiques dans les chapitres suivants.
- Vous pourrez également utiliser ce même outil pour créer des applications en temps réel