

Les 5 bogues de développement RTOS les plus courants et comment les repérer

Auteur : Johan Kraft, fondateur et PDG, [Percepio AB](#)

Qu'est-ce qui rend le développement RTOS si difficile ?

Lors de l'utilisation d'un système d'exploitation temps réel (RTOS), les développeurs dans l'embarqué travaillent à un niveau d'abstraction plus élevé, semblable au passage de la programmation en assembleur au langage C, ce qui peut faciliter la conception d'applications complexes. Mais si l'utilisation d'un RTOS réduit la complexité du code source de l'application, il ne réduit pas la complexité inhérente à l'application elle-même. Cela peut rendre l'application difficile à valider et à déboguer.

Le principal objectif d'un RTOS est de permettre le multitâche, où plusieurs modules de programme - ou tâches - peuvent s'exécuter en parallèle pour atteindre un objectif commun. Le RTOS crée l'illusion d'une exécution parallèle en commutant rapidement d'une tâche à l'autre.

Les développeurs peuvent conserver un certain contrôle sur des éléments tels que les priorités des tâches, ce qui permet en retour un comportement déterministe et temps réel, mais ils perdent également un certain contrôle sur les détails les plus fins. Par exemple, le déroulement du programme n'est plus apparent à partir du code source, puisque le RTOS décide quelle tâche exécuter à un moment donné. Ce qui rend si difficile le développement basé sur RTOS, c'est que les tâches RTOS ne sont pas des entités isolées, mais possèdent des dépendances qui peuvent retarder l'exécution des tâches de manière inattendue. De subtils choix de codage peuvent provoquer des erreurs insaisissables ou des problèmes de performances dans le produit final. Un ensemble de tâches RTOS apparemment simple peut entraîner un comportement d'exécution étonnamment complexe lors de l'exécution en tant que système ; il peut y avoir d'innombrables scénarios d'exécution qu'il est impossible de couvrir entièrement par des tests ou des revues de code.

La difficulté pour les développeurs est que le développement lui-même a évolué au cours du temps. A l'origine en assembleur, puis passant au langage C en réalisant des super boucles et au RTOS avec planification des tâches. A côté de cela l'évolution des outils de débogage pour le développement d'applications temps réel n'a pas suivi. Alors que les outils de débogage standard sont toujours principalement axés sur les points d'arrêt et l'exécution du code source en pas à pas, la trace et la visualisation RTOS donnent aux développeurs un aperçu complet du comportement de l'application au niveau du système.

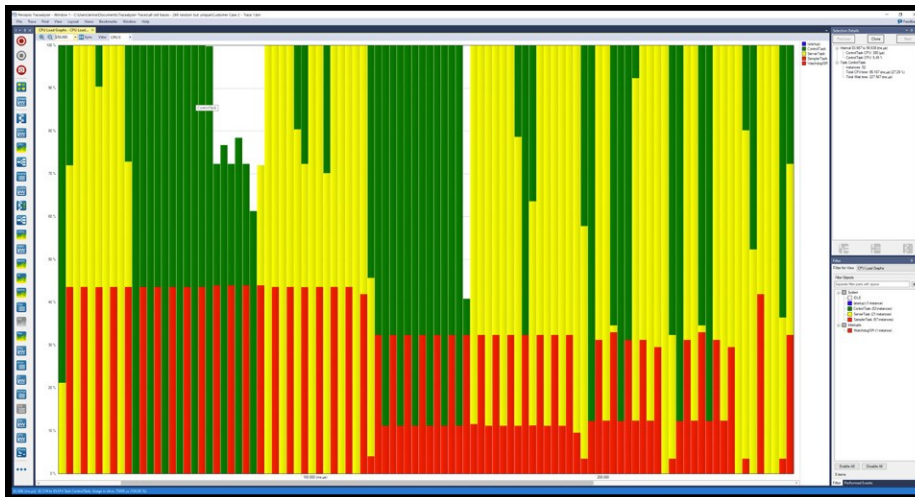
Nous avons identifié cinq des bogues d'application en temps réel les plus courants

À l'aide de la trace et de la visualisation du RTOS, nous montrons comment les développeurs peuvent travailler à un niveau d'abstraction plus élevé et avoir une vue chronologique de l'exécution de leur application que les débogueurs traditionnels ne permettent pas.

1. Surcharge CPU

Symptômes : Une ou plusieurs tâches s'exécutent lentement ou pas du tout

Dans les systèmes embarqués multitâches, certaines tâches peuvent manquer de temps CPU. Une raison classique en est que les priorités des tâches sont erronées. Les tâches de priorité plus élevée s'exécutent toujours avant les tâches de priorité inférieure, mais si les premières utilisent trop de temps CPU, les secondes ne disposent que de trop peu de temps CPU pour s'exécuter. On parle alors de tâches en manque de temps CPU



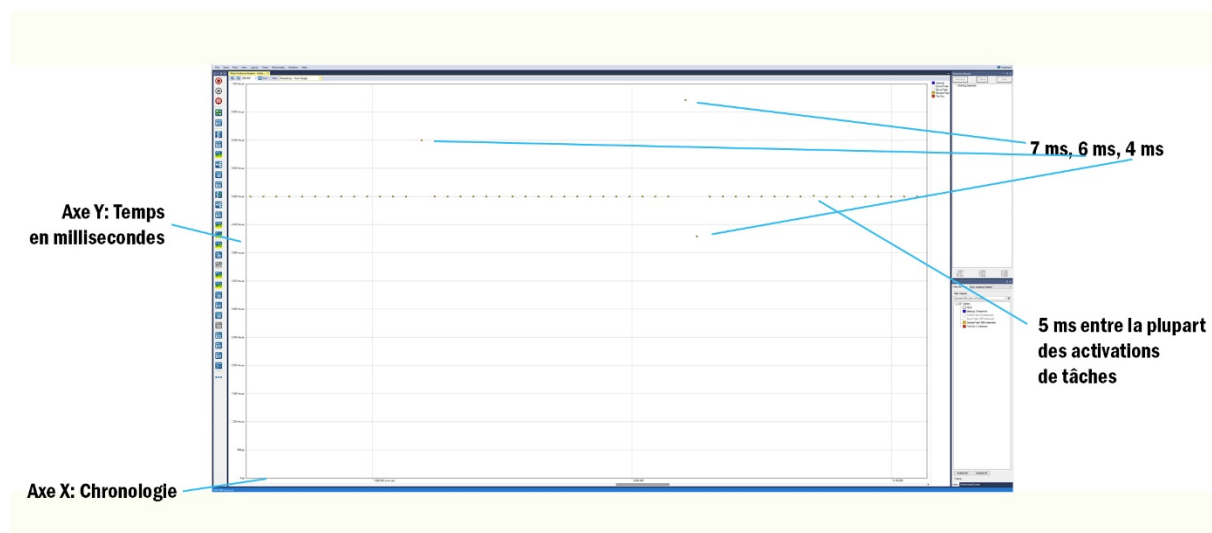
Cela se produit lorsque la charge du processeur approche 100%.

Une solution naïve serait de relever les priorités des tâches affectées, mais cela rend alors la notion de priorité inutile. Au lieu de cela, une solution consisterait à réserver une priorité élevée aux tâches prévisibles, périodiques et ayant un temps d'exécution court par rapport à la période d'exécution de la tâche. Les tâches hautement prioritaires (urgentes) et qui consomment beaucoup de temps CPU doivent être subdivisées en plusieurs tâches. Cela isole les éléments temps-critiques dans une tâche rapide de haut niveau de priorité qui permettra ensuite l'exécution des tâches de priorité moyenne ou faible pour effectuer la majeure partie de leur traitement.

2. Gigue

Symptômes : Mauvaises performances de contrôle ; Perte des données ou dysfonctionnement intermittent du système

Lorsqu'une tâche est censée s'exécuter à intervalles réguliers, par exemple pour ajuster la vitesse du moteur toutes les 5 millisecondes, le système est sensible aux retards aléatoires, également appelés gigue. Lorsqu'une tâche a un retard d'exécution, elle doit attendre plus longtemps que son temps de veille prévu avant la prochaine activation. Alors que la gigue inférieure est difficile à éviter et peut ne pas être un problème, il en va autrement d'une gigue excessive.



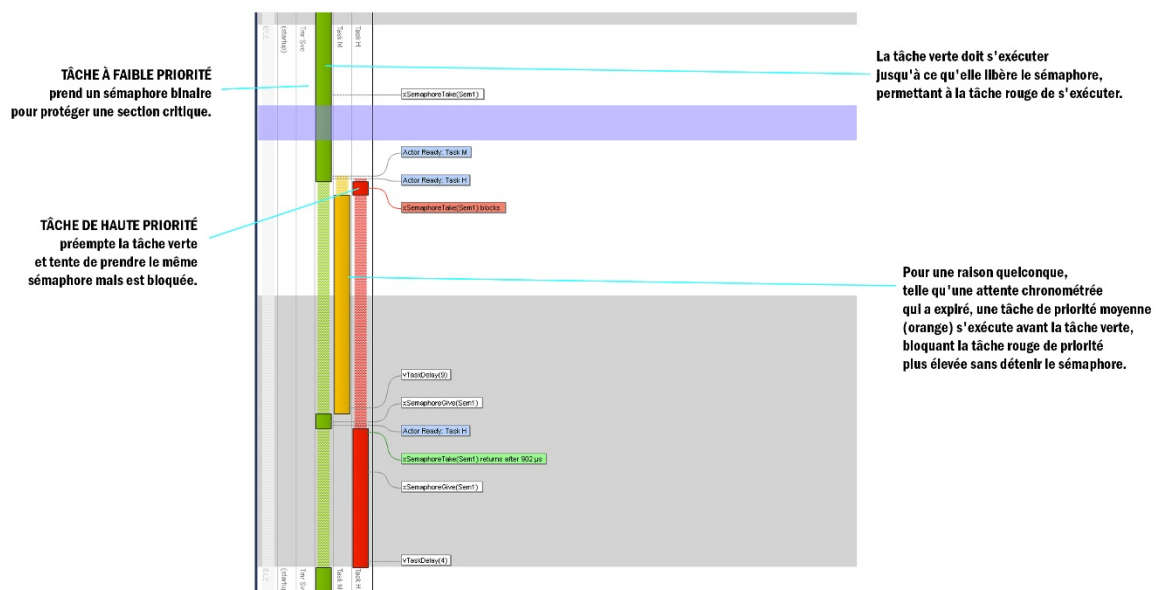
Les symptômes visibles d'une gigue excessive peuvent être des performances de contrôle médiocres et une perte de données intermittente. La cause peut en être une surcharge du processeur, qui est corrigée en définissant correctement les priorités des tâches et en évitant que les tâches de haute priorité s'exécutent longtemps, mais il existe plusieurs autres causes possibles, telles qu'une mauvaise configuration RTOS. Les développeurs doivent également regarder la valeur du tick

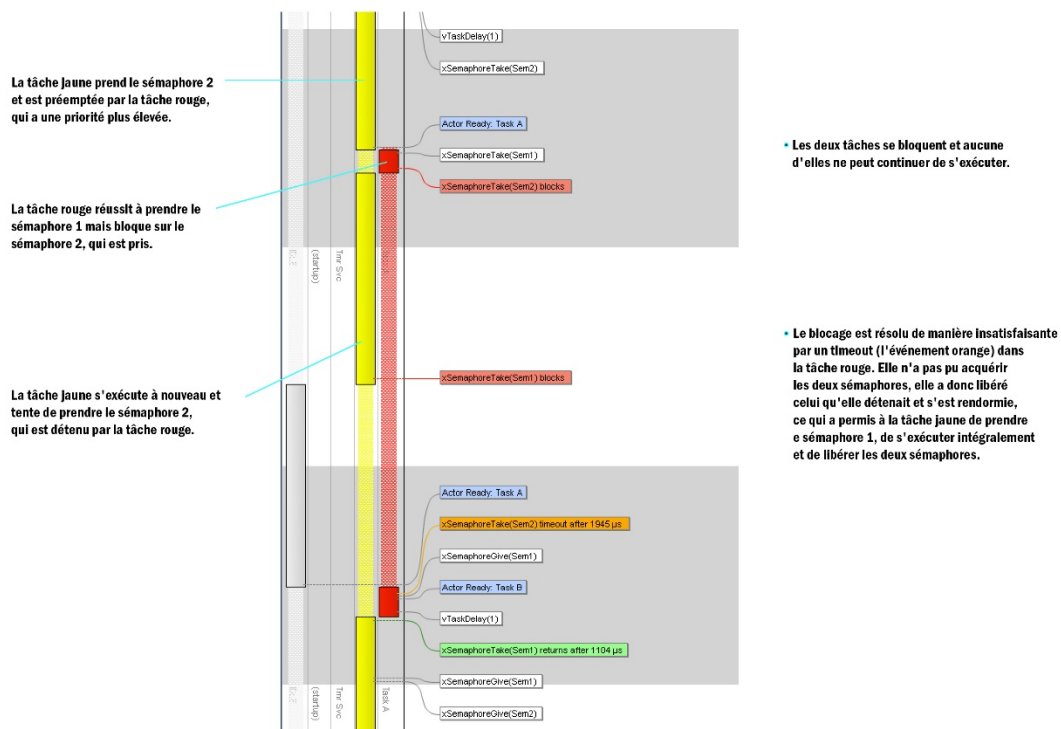
système (à quelle fréquence se produit l'interruption du « timer » du RTOS). Idéalement, le temps entre deux ticks doit être beaucoup plus court que la période des tâches les plus fréquentes du système.

3. Inversion de priorité

Symptômes : Le système cesse momentanément de répondre ; le système plante de manière aléatoire

Un RTOS doté d'un ordonnanceur à priorités fixes doit planifier les tâches de priorité élevée avant celles de priorité inférieure. Cependant, l'inverse peut parfois se produire et les tâches de priorité inférieure sont exécutées en premier. Ce phénomène est appelé inversion de priorité. Il peut se produire en conjonction avec un objet de synchronisation tel qu'une file d'attente de messages ou un sémaphore. L'inversion de priorité peut être atténuée, à condition que le RTOS sous-jacent prenne en charge ce que l'on appelle l'héritage de priorité, mais son élimination complète nécessite une conception minutieuse de l'application. L'astuce consiste à identifier que cela se produit.



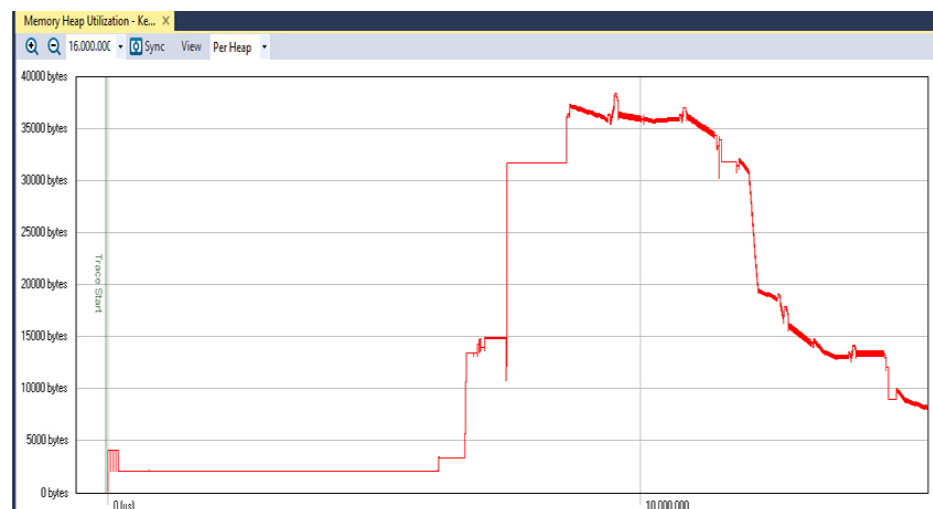


5. Fuites de mémoire

Symptômes : L'allocation de mémoire échoue pendant le fonctionnement

L'allocation dynamique de mémoire n'est généralement pas recommandée pour les logiciels embarqués, mais elle est parfois motivée par diverses raisons (bonnes ou mauvaises). De plus, des bibliothèques de logiciels tiers ou des équipes de développement externes peuvent l'utiliser à l'insu du développeur principal. Le problème avec l'allocation dynamique de mémoire est que chaque bloc de mémoire alloué doit être libéré une fois que le bloc de mémoire n'est plus utilisé. Tout manquement entraînera une fuite de mémoire et l'application finira par manquer de mémoire.

La vue Utilisation de la Mémoire affiche la quantité de mémoire actuellement allouée au fil du temps.



Une fuite de mémoire est particulièrement dangereuse si elle ne se produit qu'occasionnellement, car une fuite de mémoire lente est facilement non-détectée pendant les tests fonctionnels, mais peut provoquer des erreurs critiques plus tard dans une unité déployée. Compte tenu du caractère de long terme de nombreux systèmes embarqués, combiné à des pannes potentiellement fatales ou spectaculaires des systèmes à sécurité critique, il est important d'identifier et de corriger rapidement les bogues de fuite de mémoire. Avec Tracealyzer, les développeurs peuvent surveiller

les appels RTOS pour l'allocation dynamique de mémoire et mettre en évidence les fuites de mémoire suspectées.

Comment rechercher et corriger les bogues de développement RTOS

Ce ne sont là que les cinq bogues de développement RTOS les plus courants, mais d'innombrables autres peuvent se cacher dans votre code. Vous pouvez essayer de déboguer à tâtons, en essayant différentes solutions les unes après les autres pour que l'application s'exécute correctement. Mais pour résoudre le problème, vous devez comprendre le séquencement spécifique des événements logiciels qui l'ont causé, y compris les interactions entre l'application et le RTOS. Les outils de débogage traditionnels ne permettent pas cette compréhension. La visualisation de traces RTOS, qui peut être considérée comme une vidéo au ralenti des composants internes de l'application, donne l'assurance qu'une application RTOS fonctionne comme prévu et peut être le moyen le plus rapide de détecter et de corriger les bogues.