Utilisation du watchdog ESP32

watchdog ESP32

L'ESP-IDF prend en charge deux types de chiens de garde : le minuteur de surveillance d'interruption et le minuteur de surveillance de tâches (TWDT). Le minuteur de surveillance d'interruption et le TWDT peuvent tous deux être activés à l'aide du menu de configuration du projet , mais le TWDT peut également être activé pendant l'exécution. L'Interrupt Watchdog est chargé de détecter les cas où le changement de tâche FreeRTOS est bloqué pendant une période prolongée. Le TWDT est chargé de détecter les instances de tâches exécutées sans céder pendant une période prolongée. Chien de garde d'interruption

Le chien de garde des interruptions s'assure que l'interruption de commutation de tâche FreeRTOS n'est pas bloquée pendant une longue période. C'est mauvais car aucune autre tâche, y compris les tâches potentiellement importantes comme la tâche WiFi et la tâche inactive, ne peut obtenir de temps d'exécution du processeur. Une interruption de changement de tâche bloquée peut se produire parce qu'un programme s'exécute dans une boucle infinie avec des interruptions désactivées ou se bloque lors d'une interruption.

L'action par défaut du chien de garde d'interruption consiste à appeler le gestionnaire de panique. provoquant un vidage du registre et une opportunité pour le programmeur de découvrir, en utilisant OpenOCD ou gdbstub, quel morceau de code est bloqué avec les interruptions désactivées. Selon la configuration du gestionnaire de panique, il peut également réinitialiser aveuglément le processeur, ce qui peut être préféré dans un environnement de production.

Le chien de garde d'interruption est construit autour du chien de garde matériel du groupe de minuterie 1. Si ce chien de garde, pour une raison quelconque, ne peut pas exécuter le gestionnaire NMI qui appelle le gestionnaire de panique (par exemple parce que l'IRAM est écrasé par des déchets), il réinitialisera le SOC. Si le gestionnaire de panique s'exécute, il affichera la raison de la panique sous la forme « Délai d'expiration de l'interruption sur CPU0 » ou « Délai d'expiration de l'interruption sur CPU1 » (le cas échéant). Configuration

Le chien de garde d'interruption est activé par défaut via l' indicateur de configuration CONFIG_ESP_INT_WDT . Le délai d'attente est configuré en définissant CONFIG_ESP_INT_WDT_TIMEOUT_MS . Le délai d'expiration par défaut est plus élevé si la prise en charge de la PSRAM est activée, car une section critique ou une routine d'interruption qui accède à une grande quantité de PSRAM prendra plus de temps dans certaines circonstances. Le délai d'expiration INT WDT doit toujours être plus long que la période entre les ticks FreeRTOS (voir CONFIG_FREERTOS_HZ). Réglage

Si vous constatez que le délai d'attente du chien de garde d'interruption se déclenche parce qu'une interruption ou une section critique s'exécute plus longtemps que le délai d'attente, envisagez de réécrire le code : les sections critiques doivent être aussi courtes que possible, les calculs non critiques étant effectués en dehors de la section critique. Les gestionnaires d'interruptions doivent également effectuer le minimum de calculs possible, envisager de placer les données dans une file d'attente à partir de l'ISR et de les traiter dans une tâche. Ni les sections critiques ni les gestionnaires d'interruption ne devraient jamais bloquer l'attente qu'un autre événement se produise.

Si modifier le code pour réduire le temps de traitement n'est pas possible ou souhaitable, il est

possible d'augmenter le paramètre CONFIG_ESP_INT_WDT_TIMEOUT_MS à la place. Minuterie de surveillance des tâches

Le Task Watchdog Timer (TWDT) est chargé de détecter les instances de tâches exécutées pendant une période prolongée sans céder. Il s'agit d'un symptôme de manque de CPU et est généralement causé par une boucle de tâche de priorité plus élevée sans céder à une tâche de priorité inférieure, privant ainsi la tâche de priorité inférieure du temps CPU. Cela peut être le signe d'un code mal écrit qui tourne en boucle sur un périphérique ou d'une tâche bloquée dans une boucle infinie.

Par défaut, le TWDT surveillera les tâches inactives de chaque CPU, mais n'importe quelle tâche peut s'abonner pour être surveillée par le TWDT. Chaque tâche surveillée doit « réinitialiser » périodiquement le TWDT pour indiquer qu'elle s'est vu attribuer du temps CPU. Si une tâche n'est pas réinitialisée dans le délai d'expiration du TWDT, un avertissement sera imprimé avec des informations sur les tâches qui n'ont pas réussi à réinitialiser le TWDT à temps et sur les tâches en cours d'exécution.

Il est également possible de redéfinir la fonction esp_task_wdt_isr_user_handler dans le code utilisateur, afin de recevoir l'événement timeout et de le gérer différemment.

Le TWDT est construit autour du temporisateur de surveillance matérielle dans le groupe de temporisateurs 0. Le TWDT peut être initialisé par un appel esp_task_wdt_init()qui configurera le temporisateur matériel. Une tâche peut alors s'abonner au TWDT en utilisant esp_task_wdt_add()afin d'être regardée. Chaque tâche souscrite doit appeler périodiquement esp_task_wdt_reset()pour réinitialiser le TWDT. L'échec d'un appel périodique d'une tâche souscrite esp_task_wdt_reset()indique qu'une ou plusieurs tâches ont manqué de temps CPU ou sont bloquées dans une boucle quelque part.

Une tâche surveillée peut être désabonnée du TWDT à l'aide de esp_task_wdt_delete(). Une tâche désabonnée ne devrait plus appeler esp_task_wdt_reset(). Une fois que toutes les tâches se sont désabonnées du TWDT, le TWDT peut être désinitialisé en appelant esp_task_wdt_deinit().

Le délai d'expiration par défaut pour le TWDT est défini à l'aide de l'élément de configuration CONFIG_ESP_TASK_WDT_TIMEOUT_S . Cette valeur doit être définie au moins aussi longtemps que vous pensez qu'une tâche unique devra monopoliser le processeur (par exemple, si vous prévoyez que l'application effectuera un long calcul intensif et ne devrait pas céder le pas à d'autres tâches). Il est également possible de modifier ce délai d'attente au moment de l'exécution en appelant esp_task_wdt_init().

Les options de configuration suivantes contrôlent la configuration de TWDT au démarrage. Ils sont tous activés par défaut :

- CONFIG_ESP_TASK_WDT - le TWDT est initialisé automatiquement au démarrage. Si cette option est désactivée, il est toujours possible d'initialiser la tâche WDT au moment de l'exécution en appelantesp_task_wdt_init().
- CONFIG_ESP_TASK_WDT_CHECK_IDLE_TASK_CPU0 - La tâche inactive CPU0 est abonnée au TWDT lors du démarrage. Si cette option est désactivée, il est toujours possible de souscrire à la tâche inactive en appelantesp_task_wdt_add()à tout moment.
- CONFIG_ESP_TASK_WDT_CHECK_IDLE_TASK_CPU1 - La tâche inactive CPU1 est abonnée au TWDT lors du démarrage.

JTAG et chiens de garde

Lors du débogage à l'aide d'OpenOCD, les processeurs seront arrêtés chaque fois qu'un point d'arrêt est atteint. Cependant, si les temporisateurs de surveillance continuent de s'exécuter lorsqu'un point d'arrêt est rencontré, ils finiront par déclencher une réinitialisation, ce qui rendra très difficile le débogage du code. Par conséquent, OpenOCD désactivera les minuteries matérielles des chiens de garde d'interruption et de tâche à chaque point d'arrêt. De plus, OpenOCD ne les réactivera pas en quittant le point d'arrêt. Cela signifie que les fonctionnalités de surveillance des interruptions et des tâches seront essentiellement désactivées. Aucun avertissement ou panique de l'un ou l'autre des chiens de garde ne sera généré lorsque l'ESP32 est connecté à OpenOCD via JTAG. Référence de l'API de surveillance des interruptions En tête de fichier

- esp_common/include/esp_int_wdt.h

Les fonctions

vide esp_int_wdt_init(vide)

- Initialisez les parties non spécifiques au processeur du chien de garde d'interruption. Ceci est appelé dans le code d'initialisation si le chien de garde d'interruption est activé dans menuconfig.

Référence de l'API de surveillance des tâches

Un exemple complet utilisant Task Watchdog est disponible dans esp-idf : system/task_watchdog En tête de fichier

- esp_common/include/esp_task_wdt.h

Les fonctions

esp_err_t esp_task_wdt_init(délai d'attente uint32_t , bool panic)

- Initialiser le minuteur de surveillance des tâches (TWDT)
- Cette fonction configure et initialise le TWDT. Si le TWDT est déjà initialisé lorsque cette fonction est appelée, cette fonction mettra à jour le délai d'attente et les configurations de panique du TWDT. Après avoir initialisé le TWDT, n'importe quelle tâche peut choisir d'être surveillée par le TWDT en s'y abonnant à l'aide de esp_task_wdt_add().
- ESP_OK : l'initialisation a réussi
- ESP_ERR_NO_MEM : L'initialisation a échoué en raison d'un manque de mémoire
- esp_task_wdt_init() ne doit être appelé qu'après le démarrage du planificateur
- Paramètres
- [in] timeout: Délai d'expiration de TWDT en secondes

- [in] panic : indicateur qui contrôle si le gestionnaire de panique sera exécuté lorsque le délai TWDT expire

esp_err_t esp_task_wdt_deinit(vide)

- Désinitialiser le minuteur de surveillance des tâches (TWDT)
- Cette fonction désinitialisera le TWDT. L'appel de cette fonction alors que les tâches sont toujours inscrites au TWDT, ou lorsque le TWDT est déjà désinitialisé, entraînera le retour d'un code d'erreur.
- ESP_OK : TWDT désinitialisé avec succès
- ESP_ERR_INVALID_STATE : Erreur, les tâches sont toujours inscrites au TWDT
- ESP_ERR_NOT_FOUND : Erreur, TWDT a déjà été désinitialisé

esp_err_t esp_task_wdt_add(descripteur TaskHandle_t)

- Abonnez-vous à une tâche au minuteur de surveillance des tâches (TWDT)
- Cette fonction souscrit une tâche au TWDT. Chaque tâche abonnée doit appeler périodiquement esp_task_wdt_reset() pour empêcher le TWDT d'expirer son délai d'attente. Si vous ne le faites pas, cela entraînera un délai d'attente TWDT. Si la tâche souscrite est l'une des tâches inactives, cette fonction permettra automatiquement à esp_task_wdt_reset() d'être appelée depuis le hook Idle de la tâche inactive. L'appel de cette fonction alors que le TWDT n'est pas initialisé ou la tentative d'abonnement à une tâche déjà souscrite entraînera le retour d'un code d'erreur.
- ESP_OK : abonnement réussi à la tâche au TWDT
- ESP_ERR_INVALID_ARG : Erreur, la tâche est déjà souscrite
- ESP_ERR_NO_MEM : Erreur, impossible de souscrire à la tâche en raison d'un manque de mémoire
- ESP_ERR_INVALID_STATE : Erreur, le TWDT n'a pas encore été initialisé
- Paramètres
- [in] handle: Gestion de la tâche. Saisissez NULL pour abonner la tâche en cours d'exécution au TWDT

esp_err_t esp_task_wdt_reset(vide)

- Réinitialisez le minuteur de surveillance des tâches (TWDT) pour le compte de la tâche en cours d'exécution.
- Cette fonction réinitialisera le TWDT au nom de la tâche en cours d'exécution. Chaque tâche abonnée doit appeler périodiquement cette fonction pour éviter que le TWDT n'expire. Si une ou plusieurs tâches souscrites ne parviennent pas à réinitialiser le TWDT pour leur propre compte, un délai d'attente TWDT se produira. Si les tâches IDLE ont été abonnés au TWDT, elles appelleront automatiquement cette fonction depuis leurs hooks inactifs. L'appel de cette fonction à partir d'une tâche qui n'est pas abonnée au TWDT, ou lorsque le TWDT n'est pas initialisé, entraînera le retour d'un code d'erreur.

- ESP_OK : Réinitialisation réussie du TWDT pour le compte de la tâche en cours d'exécution
- ESP_ERR_NOT_FOUND : Erreur, la tâche en cours d'exécution ne s'est pas abonnée au TWDT
- ESP_ERR_INVALID_STATE : Erreur, le TWDT n'a pas encore été initialisé

esp_err_t esp_task_wdt_delete(descripteur TaskHandle_t)

- Désabonne une tâche du minuteur de surveillance des tâches (TWDT)
- Cette fonction désabonnera une tâche du TWDT. Après avoir été désabonnée, la tâche ne doit plus appeler esp_task_wdt_reset(). Si la tâche est une tâche IDLE, cette fonction désactivera automatiquement l'appel de esp_task_wdt_reset() depuis Idle Hook. L'appel de cette fonction alors que le TWDT n'est pas initialisé ou la tentative de désabonnement d'une tâche déjà désabonnée du TWDT entraînera le retour d'un code d'erreur.
- ESP_OK : désabonnement réussi de la tâche du TWDT
- ESP_ERR_INVALID_ARG : Erreur, la tâche est déjà désabonnée
- ESP_ERR_INVALID_STATE : Erreur, le TWDT n'a pas encore été initialisé
- Paramètres
- [in] handle: Gestion de la tâche. Saisissez NULL pour vous désabonner de la tâche en cours d’exécution.

esp_err_t esp_task_wdt_status(descripteur TaskHandle_t)

- Demander si une tâche est abonnée au minuteur de surveillance des tâches (TWDT)
- Cette fonction demandera si une tâche est actuellement abonnée au TWDT ou si le TWDT est initialisé.
- ESP_OK : La tâche est actuellement abonnée au TWDT
- ESP_ERR_NOT_FOUND : La tâche n'est actuellement pas abonnée au TWDT
- ESP_ERR_INVALID_STATE : Le TWDT n'est pas initialisé, donc aucune tâche ne peut être souscrite
- Paramètres
- [in] handle: Gestion de la tâche. Saisissez NULL pour interroger la tâche en cours d’exécution.

From:
http://chanterie37.fr/fablab37110/ - **Castel'Lab le Fablab MJC de Château-Renault**

Permanent link:
http://chanterie37.fr/fablab37110/doku.php?id=start:arduino:esp32:cours:watchdog

Last update: **2023/10/29 12:12**