highintegritysystems

WITTENSTEIN

SAFERTOS® データシート

SAFERTOS® は、安全性に重点を置いたリアルタイム OS です。最小限のリソースで高いパフォーマンスと事前認証済みの信頼性を提供します。

機能概要

SAFERTOS プリエンプティブスケジューラは以下の機能を提供します。

- 任意の数のタスクを作成することができます。
- 各タスクには優先度が割り当てられます。
- 任意の数のタスクが同じ優先度を持つことができます。
- 実行可能な最も高い優先度のタスクがスケジューラによって実行状態になります。
- 同じ優先度のタスクはラウンドロビン方式で実行されます。
- キューを使用して、タスク間や割り込みサービスルーチンとタスク間でデータを送信することができます。
- バイナリセマフォとカウントセマフォをサポートします。
- ミューテックスと再帰的ミューテックスは、優先度継承メカニズムをサポートしています。
- タスクは、固定期間の間ブロックすることも、特定の時刻に達するまでブロックすることもできます。
- タスク通知は、キュー、セマフォ、イベントグループの代わりに利用できる軽量な手段です。
- イベントグループやイベントフラグを使用すると、タスクは単一のイベントや、同じイベントグループ内の複数のイベントの組み合わせによって通知されることができます。
- ソフトウェアタイマーは、スケジューラの時間経過に基づいて定期的にタスクを実行する機能です。ハードウェアに依存せず、柔軟にタイミングを制御することができます。
- タスクの分離と隔離は、タスクごとに MPU/MMU のメモリ領域を設定することで実現されます。
- ストリームバッファは、タスク間や割り込みからタスクへの通信に使用されるプリミティブです。主に単一のリーダーと単一のライターを想定した設計で、連続したバイトストリームをコピーして転送します。
- 省電力モードをサポートすることで、プロセッサを長時間超低消費電力状態に維持できます。
- イベントマルチプレックスを使用すると、タスクが複数のイベントを待機し、発生したイベントに応じて再開できます。

主な特徴

IEC 61508-1,-3,-4 SIL3 事前認証済み
ISO 26262-2,-6,-8 ASILD 事前認証済み
IEC 62304、FDA 510(k) 準拠
充実な設計保証パック

タスクの自律性と独立性

SAFERTOS をアプリケーションに組み込むことで、アプリケーションが複数の自律的なタスクで構成されます。それぞれのタスクが提供する機能が合わさって、システム全体の機能を実現します。

各タスクは独自のコンテキスト内で実行され、システム内の他のタスクやスケジューラには影響しません。

タスク状態

一度に実行されるタスクは 1 つだけです。スケジューラは、各タスクの優先度と状態に基づいて、どのタスクを実行するかを選択する役割を担っています。

タスクは、「表 1. タスク状態」に示されたいずれかの状態を持ちます。タスクの状態遷移は、「図 1. タスク状態遷移」に示されています。

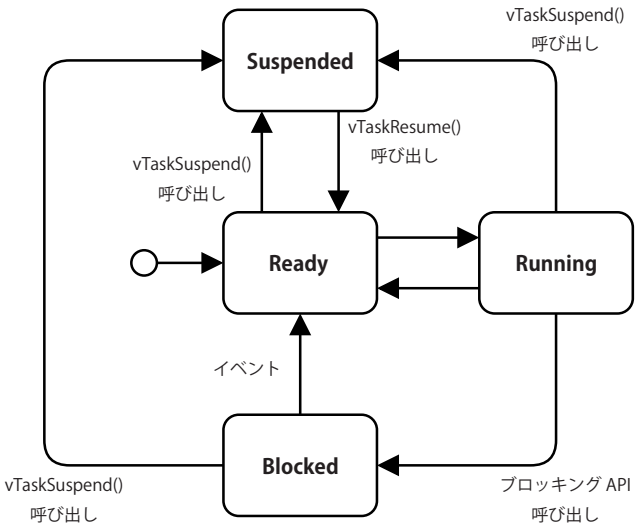

図 1. タスク状態遷移

DIOIZ
Enhance Your Embedded System

株式会社ディオイズジャパン
https://www.dioiz-japan.com
dioiz@dioiz.com

タスク状態	説明
実行状態 (Running)	スケジュラーにより選択された、現在プロセッサを使用しているタスクです。
ブロック状態 (Blocked)	イベントを待っているタスクは、イベントが発生するまで処理を続行できません。ブロック状態のタスクには常にタイムアウト時間が設定されており、その期間が経過するとブロック状態から解除されます。
サスペンド状態 (Suspended)	タスクが xTaskSuspend API 関数を呼び出すと、サスペンド状態に入ります。この状態は、xTaskResume API 関数を呼び出して再開されるまで続きます。
実行可能状態 (Ready)	タスクが実行可能な状態であるものの、スケジューラによって現在選択されていない状態です。

表 1. タスクの状態

タスクの優先度

タスクを作成する際に優先度を設定しますが、実行中にその優先度を変更することもできます。

数値が低いほど優先度が低く、最も低い優先度は 0 です。数値が高いほど優先度が高く、最も高い優先度はユーザが設定可能です。

スケジューラ

スケジューラの役割は以下の通りです。

- どのタスクを実行状態にするかを決定し、それに応じてコンテキストの切り替えを行います。
- 時間の経過を測定します。
- タイムアウト時間が経過すると、タスクをブロック状態から実行可能状態に移行させます。

時間の測定

周期的なタイマー割り込みを使用して時間を測定します。2 回の連続したタイマー割り込みの間の時間を 1 つの「ティック」と定義し、時間は「ティック」単位で測定および指定されます。

スケジューリングポリシー

スケジューラは、実行可能なタスクの中で最も高い優先度を持つタスクを実行状態に選択します。つまり、実行可能なタスクの中で最も高い優先度のタスクが実行されます。ブロック状態やサスペンド状態のタスクは実行されません。

コンパクトなフットプリント

典型的な ROM 要件	16-32K bytes
典型的な RAM 要件	1 k bytes
典型的なスタック要件	**タスクあたり** 400 bytes

異なるタスクに同じ優先度が割り当てられることがあります。その場合、同じ優先度のタスクは順番に実行されます。スケジューラは、各タスクを最大 1 ティックだけ実行し、その後、同じ優先度の別のタスクを実行します。

スケジューラは同じ優先度のタスクを順番に実行することを保証しますが、各タスクが均等に処理時間を得る保証はありません。

譲渡

譲渡とは、タスクが自発的に実行状態から実行可能状態に戻ることを指します。タスクが譲渡すると、スケジューラはどのタスクを実行状態にするか再評価します。もし、譲渡したタスクと同じかそれ以上の優先度の高いタスクが実行可能状態にない場合、譲渡したタスクが再び実行状態になります。

タスクは taskYIELD マクロを明示的に呼び出すか、他のタスクの状態や優先度を変更する API 関数を使って、CPU の制御を譲渡することができます。

スケジューラの状態

スケジューラは、「表 2. スケジューラの状態」に示されたいずれかの状態になります。スケジューラの状態遷移は、「図 2. スケジューラの状態遷移」に示されています。

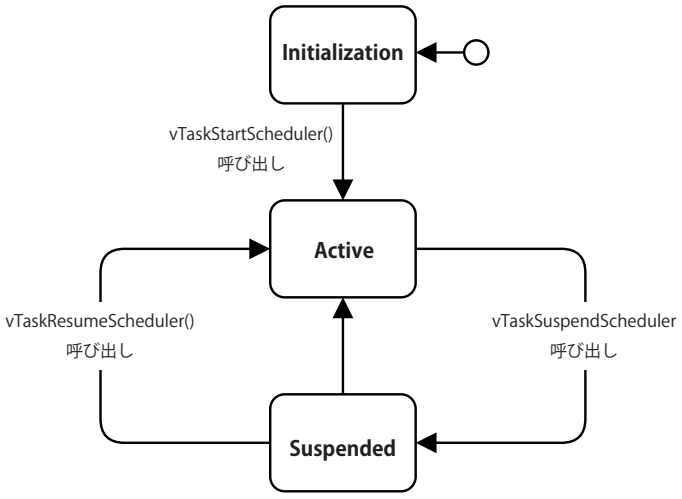

図 2. スケジューラの状態遷移

xTaskStartScheduler API 関数を呼び出してスケジューラを開始します。このとき、アイドルタスクが作成されます。アイドルタスクはブロック状態やサスペンド状態になることはありません。常に少なくとも 1 つのタスクが実行状態であるように設計されているためです。

スケジューラは、vTaskSuspendScheduler API を呼び出すとサスペンド状態に入り、xTaskResumeScheduler API を呼び出すとアクティブ状態に戻ります。

株式会社ディオイズジャパン
https://www.dioiz-japan.com
dioiz@dioiz.com

スケジューラの状態	説明
初期状態 (Initialization)	スケジューラが開始される前の状態です。この状態ではスケジューラがタスクの実行を制御することはありません。この状態でもタスクやキューを作成できます。
アクティブ状態 (Active)	スケジューラが実行可能なタスクの中から選択して実行状態にします。
サスペンド状態 (Suspended)	スケジューラがアクティブ状態に戻るまで、スケジューラがサスペンド状態に入った時に実行中だったタスクは、実行状態を維持します。

表 2. スケジューラの状態

データの整合性を保証するために、他のタスクや割り込みからの中断なしに原子性を持って実行しなければならないコードのセクションをクリティカルリージョンと呼びます。伝統的には、クリティカルリージョンに入る際に割り込みを無効にし、終了時に再度有効にします。そのために、taskENTER_CRITICAL と taskEXIT_CRITICAL マクロが用意されています。

taskENTER_CRITICAL と taskEXIT_CRITICAL マクロを使用したクリティカルリージョンでは、クリティカルリージョンの間にアプリケーションが割り込みに応答できなくなる欠点があります。スケジューラのサスペンションメカニズムを使用することで、クリティカルリージョン内でも割り込みを許可できます

vTaskSuspendScheduler API を呼び出してスケジューラがサスペンド状態にあると、別のタスクへの切り替えは発生しません。クリティカルリージョンを実行しているタスクは、xTaskResumeScheduler API を呼び出すまで実行状態のタスクとして維持されることが保証されます。

スケジューラがサスペンド状態にある間も、割り込みは有効のままです。スケジューラのサスペンションメカニズムを使用したクリティカルリージョンは他のタスクのデータアクセスからは保護されますが、割り込みからのアクセスには保護されません。そのため、スケジューラがサスペンド状態にある間は、タスクと割り込みの間でキューを使用する方が安全です。

スケジューラがサスペンド状態にある間は、実行可能状態に入った高優先度タスクへの切り替えが、xTaskResumeScheduler API が呼び出されるまで保留されます。そのため、スケジューラを長時間サスペンド状態にしておくことは望ましくありません。高優先度タスクの応答性が低下するためです。

タスク間通信

SAFERTOS は、タスク間でデータを安全に転送できる柔軟なキューを提供します。このキューは、データ転送、同期、セマフォ型の動作など、様々な目的に利用できます。

キューの特性

- キューの作成時に、各アイテムのサイズとキューが保持できる最大アイテム数が設定されます。
- アイテムは xQueueSendAPI 関数を使用してキューに送信されます。
- アイテムは xQueueReceive API 関数を使用してキューから読み取ります。
- キューは FIFO バッファです。キューに最初に送信されたアイテムが最初に取り出されます。
- キューを通じて転送されるデータはバイト単位でコピーされます。データが送信されるときにキューにバイト単位でコピーされ、受信されるときに再びバイト単位で取り出されます。
- キューは複数の送信者と受信者を持つことができます。

キューイベント

キューに送信するデータや受信するデータは、キューイベントと呼ばれます。

xQueueSend API を呼び出す際、タスクはキューが満杯の場合にスペースが空くまでのブロック期間を指定できます。タスクはキューイベントでブロックされ、他のタスクや割り込みがキューからアイテムを削除すると、自動的にブロック状態が解除されます。

xQueueReceive API を呼び出す際、タスクはキューが空の場合にデータが利用可能になるまでのブロック期間を指定できます。この場合、タスクはキューイベントでブロックされ、他のタスクや割り込みがキューにデータを書き込むと、自動的にブロック状態が解除されます。

同じイベントを待機しているタスクが複数ある場合、イベント発生時には最も高い優先度を持つタスクがブロック状態から解除されます。優先度が同じタスクが複数ある場合は、最も長くブロック状態にあったタスクが解除されます。

バイナリセマフォ

セマフォは、タスクがデータや他のリソースへの排他的なアクセスを要求するための手段です。タスクがセマフォを保持している間、他のタスクはそのリソースへのアクセスが制限されていることを認識します。

リソースへのアクセスを得るためには、タスクはまずセマフォを「取得」する必要があります。リソースの使用が終わったら、セマフォを「解放」します。セマフォを取得できない場合は、リソースが他のタスクによって使用中であるため、セマフォが利用可能になるまで待機します。ブロック状態にあるタスクは、セマフォが利用可能になると自動的に実行可能状態に戻ります。

SAFERTOS には、バイナリセマフォをサポートする API 関数が含まれています。コードサイズを小さく保つために、セマフォの実装にはキューの機能が利用されています。

バイナリセマフォは、最大で 1 つのアイテムを含むキューと考えることができます。効率性を高めるために、アイテムのサイズをゼロに設定することで、実際にはデータのコピーは行われません。重要なのは、キューが空か満杯かという情報であり、データの値そのものではありません。

リソースが利用可能なとき、バイナリセマフォは満杯の状態です。セマフォを「取得」するには、タスクがキューからアイテムを受信し、キューは空になります。セマフォを「解放」するには、タスクがキューにアイテムを送信し、キューは再び満杯になります。もしキューがすでに空の状態で受信を試みた場合、タスクはリソースにアクセスできないことを認識し、再びリソースが利用可能になるまでブロック状態に入るかどうかを指定できます。

カウントセマフォ

カウントセマフォもバイナリセマフォと似た方式で実装されています。SAFERTOS では、カウントセマフォの実装にキューの機能が利用されています。

カウントセマフォには最大カウント値が設定されており、カウントがゼロではない限りリソースは利用可能です。バイナリセマフォと同様に、カウントがゼロに達するとリソースは利用不可能となります。リソースが再び利用可能になるまで、タスクをブロック状態にするかどうかを指定できます。

タスクと割り込み間の通信

SAFERTOS は、割り込みサービスルーチンとタスク間の通信及び同期をサポートする API を提供しています。

ミューテックス

ミューテックスは、優先度継承機能を備えたバイナリセマフォです。バイナリセマフォはタスク間やタスクと割り込み間の同期を実装するのに適していますが、ミューテックスはシンプルな相互排他を実装するのに適しています。

ミューテックスが相互排他に使用される場合、リソースを保護するためのトークンのように機能します。タスクがリソースにアクセスするには、まずそのトークンを「取得」しなければなりません。リソースの使用が終了したら、トークンを「解放」して他のタスクが同じリソースにアクセスできるようにします。

ミューテックスは、ブロック時間を指定することもできます。これは、タスクがミューテックスを「取得」しようとする際に、ブロック状態に入る最大の「ティック」数を示します。バイナリセマフォとは異なり、ミューテックスは優先度継承の機構を採用しています。例えば、高優先度のタスクが低優先度のタスクによって保持されているミューテックスにアクセスしようとする場合、ミューテックスを保持している低優先度のタスクの優先度が、一時的にアクセスを求めた高優先度のタスクの優先度まで引き上げられます。このメカニズムにより、高優先度タスクがブロック状態に留まる時間が最短に保たれ、「優先度逆転」が最小限に抑えられます。

優先度の継承は優先度逆転を完全に解決するわけではありません。その影響を最小限に抑えるだけです。ハードリアルタイムアプリケーションは、優先度逆転が最初から発生しないように設計するべきです。

再帰ミューテックス

ミューテックスは、所有者が繰り返し「取得」する再帰的な使用も可能です。xSemaphoreTakeRecursive API でリクエストしたミューテックスは、xSemaphoreGiveRecursive API を呼び出すまで再び利用可能にはなりません。例えば、タスクが同じミューテックスを 5 回「取得」した場合、他のタスクがそのミューテックスにアクセスするには、所有者がそのミューテックスを正確に 5 回「解放」する必要があります。

このタイプのミューテックスは優先度継承メカニズムを使用しています。そのため、タスクがセマフォを「取得」した場合、セマフォが不要になった時点で必ず「解放」する必要があります。

ミューテックス型セマフォは、割り込みサービスルーチン内では使用できません。

株式会社ディオイズジャパン
https://www.dioiz-japan.com
dioiz@dioiz.com

イベントフラグ

イベントフラグは、タスクにイベントの発生を通知するために使用されます。単一のイベントやイベントグループ（複数のイベントフラグの組み合わせ）を使ってタスクを起こすことができます。

イベントグループ API 関数は、タスクがイベントグループ内の一つまたは複数のイベントフラグをセット・クリアする機能を提供します。また、イベントが設定されるまでタスクを待機させる機能も提供します。

イベントグループはタスクの同期にも使用できます。これを「タスクのランデブー」と呼ぶこともあります。タスクの同期ポイントとは、アプリケーションコード内でタスクがブロック状態で待機し、すべての同期対象となる他のタスクがその同期ポイントに到達するのを待つ場所です。

タスク通知

タスク通知機能は、キュー、セマフォ、イベントグループよりも軽量な代替手段を提供します。タスク通知は、1 つのタスクがデータを消費する場合に適しており、パフォーマンスの向上と RAM 使用量の削減が可能です。

各タスクは 32 ビットの通知値を持っています。タスク通知は、タスクに直接送信されるイベントで、通知待ちタスクをブロック状態から解除します。また、通知待ちタスクの通知値を更新することもできます。タスク通知は、次の方法で通知値を更新できます。

- 以前の値を上書きせずに設定します。
- 通知値を上書きします。
- 通知値の 1 つ以上のビットをセットします。
- 通知値をインクリメントします。

この柔軟性により、キュー、バイナリセマフォ、カウントセマフォ、またはイベントグループを作成する必要があった場面でも、タスク通知を代用できます。タスク通知を使用して直接タスクをブロック解除する方が、バイナリセマフォでタスクを解除するよりも最大 45% 速く、RAM の使用量も少なく済みます。

通知は xTaskNotifySend API を使用して送信されます。通知待ちタスクが xTaskNotifyWait または ulTaskNotifyWait API 関数のいずれかを呼び出すまで、通知の記録は残ります。通知待ちタスクが既に通知を待ってブロック状態にある場合、通知が到達するとブロック状態から解除され、通知はクリアされます。

イベントマルチプレックス

SAFERTOS API は、タスクが単一のイベントが発生するまでブロック状態で待機させます。タスクは、キュー、セマフォ、ミューテックス、イベントフラグ、タスク通知、またはタイマーの期限切れによってブロック状態から解除されます。

イベントマルチプレクサ機能により、タスクは複数のイベントを待機することができます。タスクがブロック状態から解除されると、発生したイベントが通知されます。SAFERTOS API は、以下の機能を提供します。

- イベントマルチプレクサオブジェクトの作成
- イベントマルチプレクサからオブジェクトイベントの追加、変更、及び削除
- イベントマルチプレクサオブジェクトでの待機

ストリームバッファー

ストリームバッファ機能は、タスク間や割り込みサービスルーチンからタスクへのバイトのストリーム転送を可能にします。また、マルチコアデバイスの異なるマイクロプロセッサコア間でのストリーム転送もサポートしています。

1 回の読み取りまたは書き込み操作で、任意のバイト数にアクセスできます。送信者はデータをストリームバッファにコピーし、受信者はバッファからデータを読み取ります。ストリームバッファは、単一のリーダーと単一のライター間で効率的なデータ転送を提供します。

タスクの空間的分離

SAFERTOS は、タスクごとに MPU 領域の定義と操作をサポートしています。この機能により、開発者はタスク間の空間的な分離を考慮して設計できます。効果的に使用することで、タスクが意図せずまたは不注意に不正なメモリ領域にアクセスするのを防ぐことができます。

MPU の実装は、選択したプロセッサコアに密接に関連しています。MPU はメモリ領域に対するアクセス権限を設定する手段を提供します。設定可能なメモリ領域の数、サイズ、およびアドレッシングはプロセッサに依存します。各領域にはアクセス権限を設定でき、コードの実行を許可または拒否することができます。また、領域は読み取り専用アクセス、読み取り / 書き込みアクセス、またはアクセス不可に設定できます。これらの設定は、特権モードとユーザーモードの両方に適用されます。

カーネルのコードおよびデータ領域には、特権アクセスが設定されます。残りの MPU 領域はタスクに割り当てられ、適切なアクセスレベル（ユーザーまたは特権）に設定できます。タスクに関連する MPU メモリ領域は、各コンテキストスイッチ時に再設定されるため、各タスクは個別のメモリアクセスプロファイルを持つことができます。

株式会社ディオイズジャパン
https://www.dioiz-japan.com
dioiz@dioiz.com

メモリ領域が割り当てられた後、MPU を有効にするとプロセッサの Memory Manage Fault ハンドラも有効化されます。メモリ領域へのアクセス違反が発生すると、Memory Manage Fault が発生し、ハンドラが呼び出されます。プロセッサが MMU のみをサポートしている場合、MMU は MPU のように動作するように設定されます。

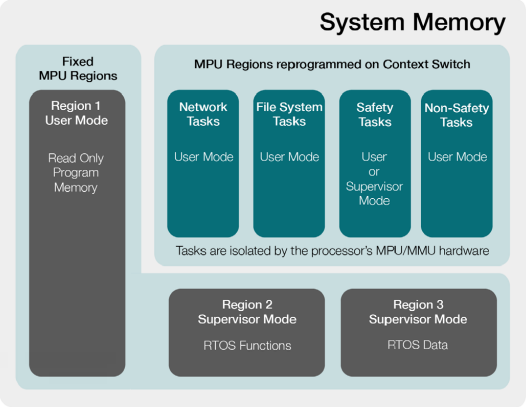

図 3. MPU タスク分離

タスクローカルストレージポインタ

スレッドローカルストレージは、アプリケーション作成者がタスクの制御ブロック内に値を格納し、その値をタスクごとに特有のものにします。この機能により、各タスクがそれぞれ独自の値を持つことができます。

スレッドローカルストレージは、通常、単一スレッドプログラムがグローバル変数に格納するような値を保存するために使用されます。たとえば、多くのライブラリは「errno」というグローバル変数を参照し、ライブラリ関数がエラーを呼び出し元に返すと、呼び出し元の関数は errno の値を調べてエラーの詳細を特定します。単一スレッドのアプリケーションでは、errno をグローバル変数として宣言するだけで十分ですが、マルチスレッドアプリケーションでは、各スレッド（タスク）が独自の errno 値を持つ必要があります。これは、別のタスクが errno の値を上書きして不正な値を読み取る可能性があるためです。

低電力モード

プロセッサの消費電力を削減するためには、アイドルタスクのフック関数を使用してプロセッサを低電力状態にすることが一般的です。この方法では、RTOS のティック割り込みを利用してプロセッサを定期的に低電力状態にすることができますが、CPU の使用状況によって達成できる省電力効果には限界があります。

代案として、SAFERTOS のティックレス超低電力モードを利用する方法があります。このモードでは、アイドル期間中に RTOS のティック割り込みを完全に停止し、システムを省電力モードに移行させることで電力を節約します。ティック割り込みが停止される一方で、プロセッサの割り込みは有効のままです。カーネルがタスクを再開するまで、より長く深い省電力モードを維持することができます。

エラーチェック

SAFERTOS は、ホストアプリケーションによる不正使用のリスクを軽減するために、API の入力値の有効性を徹底的にチェックします。API パラメータの値が無効と判断された場合、API 関数は操作を行わず、エラーコードを返します。このエラーコードは、発生したエラーの内容を示します。

SAFERTOS は、データの破損を検出するために、実行時に次の整合性チェックを行います。

- 実行中でないタスクの実行コンテキストは、そのタスクに割り当てられたスタックに保存されます。ただし、タスクのコンテキストは、スタックに十分な空き領域が残っている場合にのみ保存されます。
- タスクが実行状態に入る前に、そのタスクに関連付けられたタスク制御ブロック（TCB）が有効であることを確認するためのチェックが行われます。これは、主要なデータパラメータとミラーコピーを照合することで実現されます。

フック関数	説明
アプリケーション エラー フック関数	致命的なエラー、例えばスケジューラのデータ構造の破損やコンテキストスイッチ中のスタックオーバーフローの可能性が検出された場合に呼び出されます。アプリケーションエラーフックは、ホストアプリケーションがシステムを安全な状態に保つためのアプリケーション固有のエラーハンドリングを実行できるようにします。エラーフック関数は必須です。
アプリケーション タスク削除 フック関数	タスクが削除されたときに呼び出されます。このフック関数は、タスクに割り当てられていたメモリが解放され、他の目的で使用できるようになったことをホストアプリケーションに通知します。
アプリケーション アイドル フック関数	アイドルタスクのコンテキスト内でアプリケーション固有の機能を実行できます。アイドルタスクフックを使用して、低優先度のアプリケーション固有のバックグラウンドタスクを実行したり、プロセッサを低電力のスリープモードにすることが一般的です。
ティック フック関数	ティックハンドラの実行時にティックフック関数が呼び出されます。これにより、アプリケーション固有の機能を定期的に実行したり、アプリケーションタイマーを実装することが可能です。
SetupTimerInterrupt フック関数	SAFERTOS のティック割り込みを生成するために代替のタイマーを使用できるようにします。このフック関数は、デフォルトの実装の代わりにタイマ周辺機器を初期化し、開始するために使用されます。

表 3. フック機能

株式会社ディオイズジャパン
https://www.dioiz-japan.com
dioiz@dioiz.com

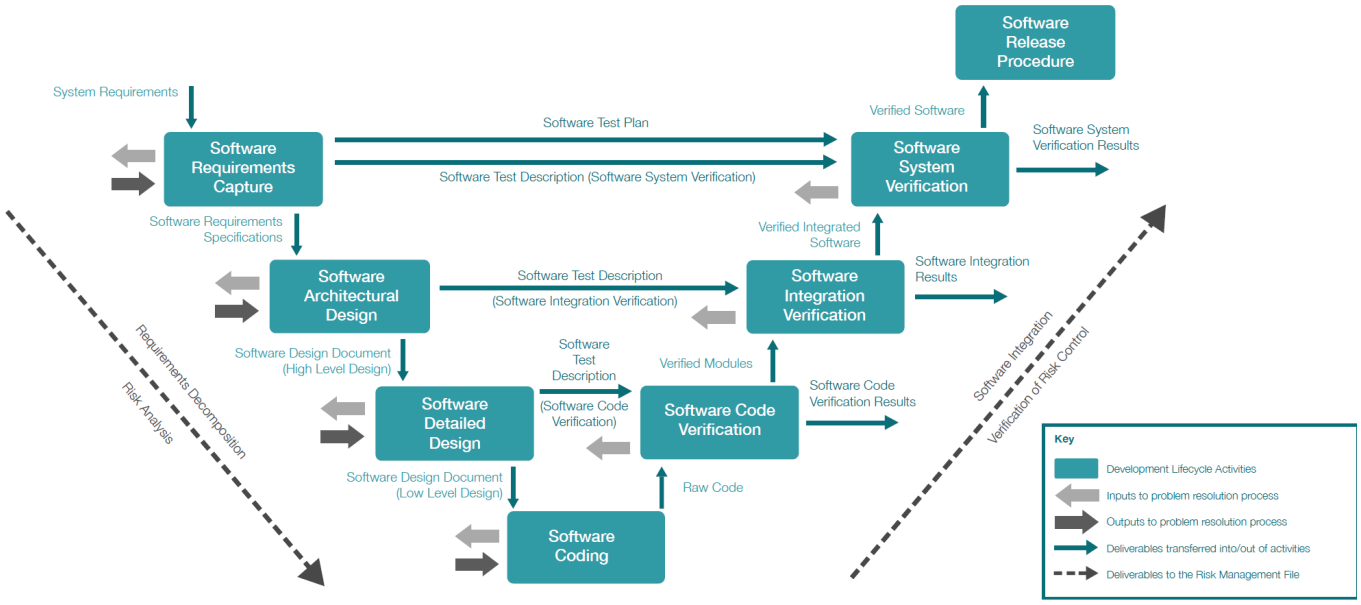

図 4. SAFERTOS の開発ライフ

- ティックカウント値を増加させる前に、現在のティックカウント値が最後に書き込まれた期待される値と一致しているかどうかを確認します。
- MPU 実装をサポートする場合は、正しいプロセッサモードと特権レベルが復元されていることを確認します。

これらの整合性チェックのいずれかが失敗した場合、アプリケーションエラーフック関数が呼び出されます。

フック機能

1 つの必須フック関数と 4 つのオプションのフック関数が用意されています。詳細は、「表 3. フック関数」をご参照ください。

開発ライフサイクル

SAFERTOS は、FreeRTOS カーネルを補完するように設計されており、共通の機能を持ちながらも、安全性が重視された独自の実装が特徴です。

FreeRTOS カーネルの機能モデルに対して HAZOP が徹底的に実施され、機能モデルと API 内のすべての弱点が特定され、解決されました。その結果、得られた安全性および機能要件は、ソフトウェア単体として最高レベルである IEC 61508 SIL3 の開発ライフサイクルに従って開発されました。

コードベースは、MISRA ガイドラインの大部分に準拠して開発されており、安全性を高めるための独自の追加ルールも適用されています。

SAFERTOS の開発ライフサイクルを支えているのは、要求管理プロセスです。SAFERTOS プロジェクト全体にわたり、最初の顧客要求から最後の検証テスト結果に至るまで、100% のトレーサビリティを達成しています。この 100% のトレーサビリティは、製品の認証において重要な要素であり、設計の完全性を示すものです。

設計保証パック (DAP)

SAFERTOS には、設計保証パック（DAP）が付属しています。DAP には、開発および安全ライフサイクル計画、要求仕様書、設計文書、HAZOP、ソースコード、すべての検証および確認文書、そして関連する証拠が含まれており、開発ライフサイクル全体で作成されたすべての設計成果物が含まれています。

ユーザーおよびセーフティマニュアルに加え、完全なテストハーネスも提供されます。これにより、お客様は認証プロセスに必要なすべての情報を揃えることができます。

SAFERTOS セーフティマニュアルには、SAFERTOS に含まれるすべてのコンポーネントとそのチェックサムが詳しく記載されています。また、SAFERTOS を開発環境に統合する際のインストール手順やプロセスも明確に示されています。サポートが必要な場合は、経験豊富なエンジニアが支援を提供します。

DAP は、各文書へのリンクが含まれたグラフィカルインターフェースを通じてアクセスできます。

株式会社ディオイズジャパン
https://www.dioiz-japan.com
dioiz@dioiz.com

DAP には以下の文書が含まれています。

- SAFERTOS ユーザーマニュアル : SAFERTOS の概要、タスク、キュー、スケジューリングメカニズムの説明、インストールおよび設定手順を提供し、API リファレンスも含まれています。
- FreeRTOS から SAFERTOS へのアップグレード：FreeRTOS から SAFERTOS への移行時に修正が必要な領域を強調しています。
- デモプログラム : SAFERTOS の機能を評価できるサンプルプログラムが含まれています。
- ソフトウェアバージョン説明書 : リリースされたソフトウェアの内容、関連するチェックサム、そしてその提供物のライフサイクル全体にわたる変更履歴を含みます。
- ソフトウェア開発計画 : IEC 61508 SIL 3 に準拠した開発ライフサイクルを定義し、これに基づいて SAFERTOS の開発を行います。
- 構成管理計画 : 構成管理ツールの概要を示し、構成管理対象のアイテム、構成管理ルールおよび手順を特定し、これに基づいて SAFERTOS の開発における運用方法を説明します。
- ソフトウェアテスト計画 : 各検証および妥当性確認（V&V）フェーズの目的と関連するテスト環境を定義し、V&V の全体スケジュールを示します。
- 顧客要求仕様書 : SAFERTOS 製品に対する顧客の要求事項を定義します。
- ソフトウェア要求仕様書 : SAFERTOS の機能要求および適格性確認方法に関するソフトウェア要求事項を定義します。
- アーキテクチャソフトウェア設計説明書 : SAFERTOS 製品の構造および設計制約に関するアーキテクチャソフトウェア設計を説明します。
- 詳細ソフトウェア設計説明書 : SAFERTOS 製品の個々のモジュールに関する詳細なソフトウェア設計を説明します。
- HAZOP レポート : この文書には、SAFERTOS の危険性と操作性に関する調査、危険評価、リスク削減方法、安全関連の要求事項、および残留リスクの詳細が含まれます。
- API 使用安全レビュー : API を構成する関数やマクロを分析し、SAFERTOS API の実際のまたは潜在的な動作がユーザーやその他の関係者による不安全または不適切な使用につながる可能性があるかどうかを評価します。
- ソフトウェアテスト説明書 : 各テストケース内のすべてのテストステップについて、入力、期待される出力、および操作手順を詳細に説明します。
- カーネルソースコード : C およびアセンブラ形式で提供される完全なソースコードとビルドファイル
- テストハーネス : SAFERTOS 製品の検証に使用されるテストハーネスの完全なソースコードとビルドファイルが提供されます。
- テストハーネスビルド手順 : 製品の正式なテストを行う際に、ソフトウェアテストハーネスを構築し使用するための手順を提供します。
- テストログ : 実際のテスト結果ログファイル
- ソフトウェアテスト報告書 : ソフトウェアテスト報告書は、各テストケースの詳細に入る前に、V&V プロセスから得られた結果の概要を示します。
- IEC 61508-3 SIL3 主張を支持する証拠 : この主張は、SAFERTOS の要求事項が仕様され、満たされていること、及びそれが IEC 61508-3 安全性インテグリティレベル（SIL）3 のソフトウェア開発要求を満たすように開発および認証されていることを支持するすべての証拠をまとめ、参照します。これは、主張、議論、および証拠のアプローチを使用しています。
- IEC 61508 準拠マトリックス : IEC 61508: 2010 の要求事項と SAFERTOS の開発ライフサイクルとのクロスリファレンスを提供します。

SAFERTOS コンフィグレーション

SAFERTOS は、使用するマイクロプロセッサやツールチェーンに応じて異なるバリアント（派生版）で提供されます。SAFERTOS は高い信頼性を誇る RTOS であり、API やコアの設計、コードはすべてのバリアントで共通です。ポートレイヤーは、選択したマイクロプロセッサに合わせてカスタマイズされます。各 SAFERTOS バリアントは、IEC 61508 準拠の完全な開発ライフサイクルに基づいて開発されています。

認証

SAFERTOS は 2007 年に TÜV SÜD によって初めて認証され、世界初の事前認証リアルタイム OS となりました。SAFERTOS は、61508-3 SIL3 や ISO 26262-6 ASILD の事前認証ソフトウェアコンポーネントとして提供されます。また、SAFERTOS は IEC 62304 Class C、IEC 61508 SIL3、および IEC 61508 のドメイン適応に対する認証が必要な開発に適しています。

株式会社ディオイズジャパン
https://www.dioiz-japan.com
dioiz@dioiz.com