AN3007

megaAVR® 0系でのFreeRTOSの開始に際して

要点

- FreeRTOS™の構成設定と基本的な機能
- ﾃﾞﾊﾞｯｸﾞと代表的な誤り
- Atmel | STARTｺｰﾄﾞ例

序説

著者: Eira Mørch-Thoresen, Microchip Technology Inc.

FreeRTOSは組み込み装置用実時間ｵﾍﾟﾚｰﾃｨﾝｸﾞ ｼｽﾃﾑです。これは小さくて簡単なように設計され、従って、殆どCで書かれた少しのﾌｧｲﾙだけで成ります。

組み込みｼｽﾃﾑが厳密に定義された時間の長さ内で或る事象に応答することができなければならないことを意味する実時間組み込み応用には度々ﾏｲｸﾛ ｺﾝﾄﾛｰﾗが使われます。ｼｽﾃﾑがそれらの期限に合致することを保証するため、RTOSは何れかの時間の瞬間でどの作業(ﾀｽｸ)を動かすかを決める計画部(ｽｹｼﾞｭｰﾗ)を持ちます。

FreeRTOSは作業(ﾀｽｸ)に対して、作業(ﾀｽｸ)通信と計画の機能を提供し、事実上のﾏｲｸﾛ ｺﾝﾄﾛｰﾗ用標準実時間ｵﾍﾟﾚｰﾃｨﾝｸﾞ ｼｽﾃﾑ(RTOS)になってきました。FreeRTOSの主な設計目標は頑強性、使い易さ、小さな占有量です。

この資料はFreeRTOSをどう構成設定することができるかの記述から始まり、その後に中断機能、作業(ﾀｽｸ)相互通信の仕組み、計画の説明へ行きます。実演ｺｰﾄﾞについての章の前にﾃﾞﾊﾞｯｸﾞについての章も含まれます。この応用記述は実演での作業(ﾀｽｸ)の各々に対する統一模式化言語(UML:Unified Modeling Language)図も提供します。

目次

1. 関連ﾃﾞﾊﾞｲｽ

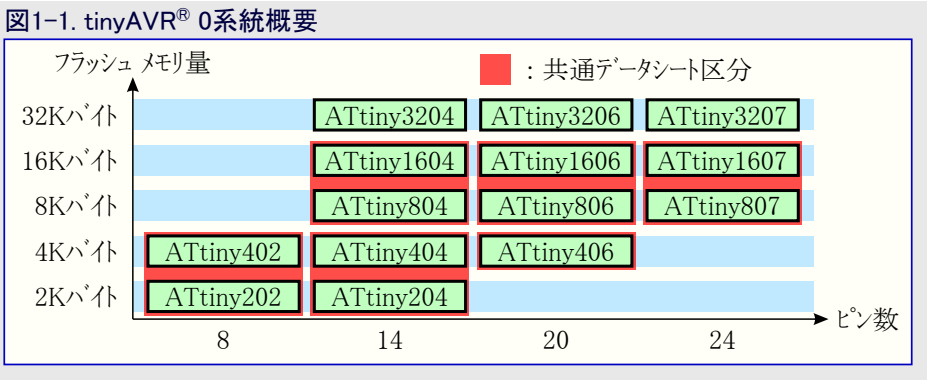
本章はこの資料に関連するﾃﾞﾊﾞｲｽを一覧にします。

1.1. tinyAVR® 0系統

下図はﾋﾟﾝ数の変種とﾒﾓﾘ量を展開してtinyAVR® 0系統ﾃﾞﾊﾞｲｽを示します。

- これらのﾃﾞﾊﾞｲｽが完全にﾋﾟﾝと機能が互換のため、垂直方向移植はｺｰﾄﾞ変更なしで可能です。
- 左への水平方向移植はﾋﾟﾝ数、従って利用可能な機能を減らします。

図1-1. tinyAVR® 0系統概要



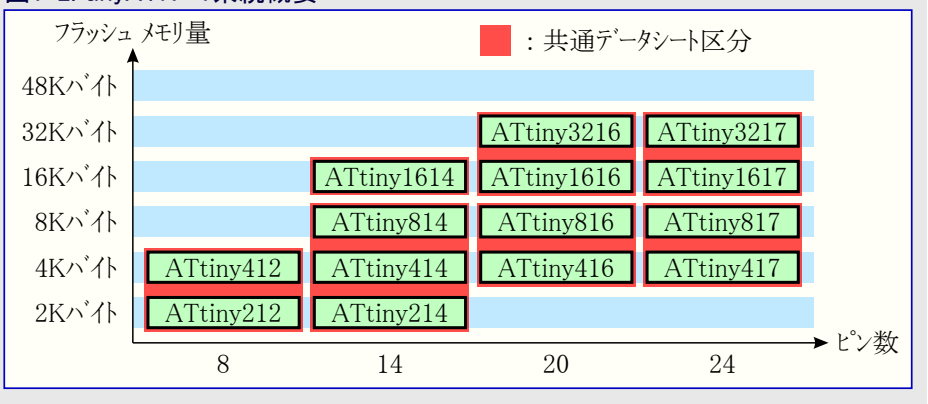
異なるﾌﾗｯｼｭ ﾒﾓﾘ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

1.2. tinyAVR® 1系統

下図はﾋﾟﾝ配置変種とﾒﾓﾘ量を展開してtinyAVR® 1系統ﾃﾞﾊﾞｲｽを示します。

- これらのﾃﾞﾊﾞｲｽがﾋﾟﾝ互換で同じまたはより多くの機能を提供するため、垂直上方向移植はｺｰﾄﾞ変更なしに可能です。下方向移植はより少ない利用可能ないくつかの周辺機能の実体のためにｺｰﾄﾞ変更が必要かもしれません。
- 左への水平方向移植はﾋﾟﾝ数、従って利用可能な機能を減らします。

図1-2. tinyAVR® 1系統概要



異なるﾌﾗｯｼｭ ﾒﾓﾘ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

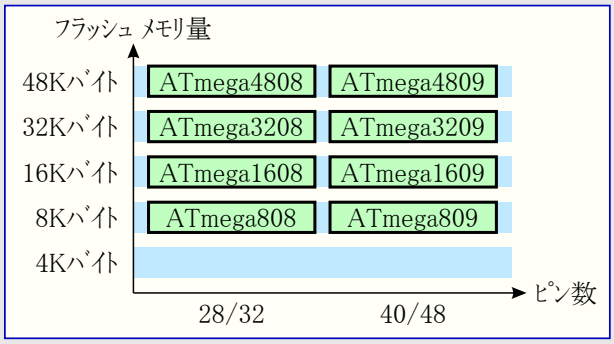
1.3. megaAVR® 0系統

右図はﾋﾟﾝ配置変種とﾒﾓﾘ量を展開してmegaAVR® 0系統ﾃﾞﾊﾞｲｽを示します。

- これらのﾃﾞﾊﾞｲｽが完全にﾋﾟﾝと機能が互換のため、垂直方向移植はｺｰﾄﾞ変更なしで可能です。
- 左への水平方向移植はﾋﾟﾝ数、従って利用可能な機能を減らします。

異なるﾌﾗｯｼｭ ﾒﾓﾘ量を持つﾃﾞﾊﾞｲｽは一般的に異なるSRAMとEEPROMの量を持ちます。

図1-3. megaAVR® 0系統概要



2. Atmel | STARTからの開始

FreeRTOS応用を作成する最も簡単な方法はAtmel | STARTから例をﾀﾞｳﾝﾛｰﾄﾞしてそれらで作業することです。これは全てのFreeRTOSﾌｧｲﾙが含まれて正しく動くことを保証します。実演ｺｰﾄﾞはその後に取り去ることができ、応用開発を始めることができます。

この応用記述では次のAtmel | START例ﾌﾟﾛｼﾞｪｸﾄが使われます。ATmega4809 FreeRTOSExample

3. FreeRTOS構成設定

FreeRTOSは"FreeRTOSConfig.h"と呼ばれる構成設定ﾌｧｲﾙを使います。これを変えることにより、望むやり方でFreeRTOS応用を動くように独自化することができます。構成設定ﾌｧｲﾙは前処理部(ﾌﾟﾘﾌﾟﾛｾｯｻ)ﾊﾟｽに含まれなければなりません。

変える代表的な構成設定はﾒﾓﾘの大きさ、ｽﾀｯｸ任意選択だけでなく、ｸﾛｯｸと刻時速度もです。更に、作業(ﾀｽｸ)の優先権と先取り権を構成設定することができます。構成設定ﾌｧｲﾙは多数の任意選択を含み、この応用記述は利用可能な任意選択の一部だけを網羅します。完全な概要についてはhttps://www.freertos.org/a00110.htmlを参照してください。

3.1. ｸﾛｯｸと刻時速度の構成設定

実時間ｵﾍﾟﾚｰﾃｨﾝｸﾞ ｼｽﾃﾑ(RTOS)は計時器に基づく時間単位であるｼｽﾃﾑ刻時を使います。FreeRTOSは自身の刻時割り込みを生成するためにﾏｲｸﾛ ｺﾝﾄﾛｰﾗのTCB0を使います。FreeRTOSｶｰﾈﾙは刻時に使う時間を測定し、刻時が起こる時毎に作業(ﾀｽｸ)が起き上がるまたは中断解除されるべきかを計画部(ｽｹｼﾞｭｰﾗ)が調べます。

configCPU_CLOCK_HZ定義は正しいFreeRTOSﾀｲﾐﾝｸﾞのために構成設定されなければなりません。TCB0ｸﾛｯｸの周波数はここで入力されるべきです。TCB0は既定によって「関連ﾃﾞﾊﾞｲｽ」章で一覧にされたﾃﾞﾊﾞｲｽに対してCPUと同じｸﾛｯｸ周波数で動作します。

configTICK_RATE_HZはRTOS刻時割り込みの周波数を決めるのに使われます。刻時発生時毎にいくつかのCPU付随負荷があるため、刻時速度はCPU周波数に比べて高すぎる設定にすべきではありません。

複数の作業(ﾀｽｸ)が同じ優先権を持つ場合、RTOS計画部はconfigUSE_TIME_SLICINGが1に設定されるなら、刻時毎にこれらの作業を切り替えます。従って、より高い刻時速度を使うと、CPUをより小さな(短い)各作業用時間にさせます。例えば、1000Hzの刻時速度が使われる場合、各作業用割り当て時間はT=1/f=1/1000Hzでこれは1msです。

3.2. ﾒﾓﾘ構成設定

FreeRTOSは2つのﾒﾓﾘ割り当ての仕組み、静的割り当てと動的割り当てを持ちます。静的ﾒﾓﾘ割り当てはﾒﾓﾘの割り当てと割り当て解除を応用それ自身に必要とします。これは実装がより難しいかもしれませんが、より多くの制御を提供します。動的割り当ての仕組みを使うと、ﾒﾓﾘ割り当てがAPI関数内で自動的に起きます。応用は作成と削除の関数だけを呼ばなければなりません。動的割り当てを使うには、構成設定ﾌｧｲﾙでconfigSUPPORT_DYNAMIC_ALLOCATION定義を1に設定してください。

動的割り当てでのFreeRTOSはﾋｰﾌﾟと呼ばれる領域に割り当てられたRAMの部分を持つことが必要です。"Create"で終わるFreeRTOS関数はﾋｰﾌﾟ上のﾒﾓﾘを割り当てます。xTaskCreate()呼び出しによって作業(ﾀｽｸ)を作成すると、ﾋｰﾌﾟでｽﾀｯｸと作業制御部(TCB:Task Control Block)から成るその特定作業(ﾀｽｸ)用ﾒﾓﾘが割り当てられます。作業(ﾀｽｸ)ｽﾀｯｸの大きさは作業(ﾀｽｸ)を作成する時に定義されます。待ち行列(ｷｭｰ)、ﾐｭｰﾃｯｸｽ、ｾﾏﾌｫも作られる時にﾋｰﾌﾟ ﾒﾓﾘを割り当てます。正しいｽﾀｯｸの大きさを定義する助言については「FreeRTOSでのﾃﾞﾊﾞｯｸﾞ」章をご覧ください。FreeRTOSﾒﾓﾘがどう働くかの図解については**図3-1.**を参照してください。

動的割り当てが使われる時にconfigTOTAL_HEAP_SIZEは充分大きいことが必要です。換言すると、全ての作業(ﾀｽｸ)ｽﾀｯｸの大きさと他の割り当てられたﾒﾓﾘに合うように充分大きくなければなりません。実演例ではﾋｰﾌﾟの大きさが0x800(2048ﾊﾞｲﾄ)に設定されます。ﾋｰﾌﾟの大きさを最適化するのにxPortGetFreeHeapSize()を使うことができます。この関数は未割り当てﾋｰﾌﾟの量を返します。正しいﾋｰﾌﾟの大きさを見つける方法の助言については「FreeRTOSでのﾃﾞﾊﾞｯｸﾞ」章をご覧ください。

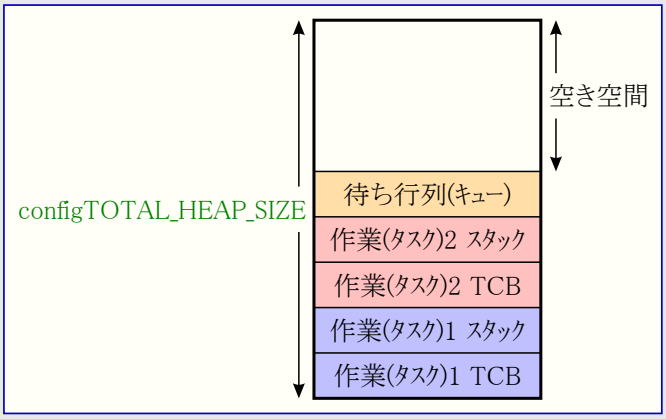
FreeRTOSではﾋｰﾌﾟ1～5と名付けられる利用可能な5つの異なるﾋｰﾌﾟ実装があります。使うﾋｰﾌﾟ割り当てを選ぶにはfreeRTOS/portable/MemMangﾌｫﾙﾀﾞで見つかるheap.cﾌｧｲﾙでUSE_HEAP定義を変更してください。

Atmel | START例はﾋｰﾌﾟ1実装を使います。ﾋｰﾌﾟ1は一旦ﾒﾓﾘが取られると開放や再割り当てができないという点で静的割り当てと同様です。ﾋｰﾌﾟ1は簡単ですが、それらのｵﾌﾞｼﾞｪｸﾄの作成と破棄が応用にﾒﾓﾘを使い果たさせるため、作業(ﾀｽｸ)と待ち行列(ｷｭｰ)、ﾐｭｰﾃｯｸｽ、ｾﾏﾌｫのような他のFreeRTOSｵﾌﾟｼﾞｪｸﾄは応用の生涯を通してﾋｰﾌﾟを維持することが必要とされます。

ﾋｰﾌﾟ2の仕組みはﾋｰﾌﾟ1とは異なり、解放されることをﾒﾓﾘに許しますが、割り当て失敗に終わり得る断片化になることを利用可能な空きﾒﾓﾘに起こさせるかもしれないので、割り当てられたﾒﾓﾘが不揃いな大きさの場合に注意して使われるべきです。

ﾋｰﾌﾟ3,4,5はもっと高度な仕組みです。より多くの情報についてはhttps://www.freertos.org/a00111.htmlを参照してください。

図3-1. FreeRTOSﾒﾓﾘ



3.2.1. スタック構成設定

FreeRTOSでは各作業(タスク)がそれ自身の独立したスタックを持ちます。作業(タスク)スタックは動的メモリ割り当て使用時にヒープ上に割り当てられます。アイドル作業(タスク)を除き、スタックは入力パラメータの1つが作業(タスク)用のスタックの大きさであるxTaskCreate()を呼ぶ時に割り当てられます。

アイドル作業(タスク)は最低走行できる1つの作業(タスク)が常にあることを保証するため、vTaskStartScheduler()を呼ぶ時に作成されます。アイドル作業(タスク)のスタックの大きさは構成設定ファイル内のconfigMINIMAL_STACK_SIZEによって与えられます。

割り当てられたスタック メモリだけでなく、待ち行列(キュー)、セマフォなどで割り当てられたメモリの合計がconfigTOTAL_HEAP_SIZEで定義されたヒープの大きさを超えないことが重要です。

4. RTOS開発者のように考える

実時間プログラミングとそれの厳密な時間制限は開発者の考え方を変えます。代表的な状態機構実装の使用に代えて、応用は複数の作業(タスク)に分割されます。1つの作業(タスク)は応用の一部分を満足し、同時に他の作業(タスク)はその他の責任を持ちます。例えば、Atmel | START例ではLEDを点滅する1つの作業(タスク)、OLED画面に時間を書く1つの作業(タスク)、それと各種の責任を持つ多数の他の作業(タスク)があります。実時間システムではこれらの作業(タスク)が同時に動き、計画部がそれを許す時間毎にそれらのコードの一部が実行されます。これは全てのものが順次且つ常に同じ順で評価される代表的な直線状応用コードと異なります。

他よりももっと重要な作業(タスク)があり、より速い応用時間を必要とする場合、RTOSはより低い優先権の作業(タスク)に割り込むことをより高い優先権の作業(タスク)に許すように構成設定することができます。これは先取り権と呼ばれます。

応用を正しく動かすため、作業(タスク)は害を起こし得る時に互いに割り込まないことを確実にすることが重要です。この例は別の作業(タスク)がそれの計算を行うのを終了されない前に作業(タスク)が共有された資源の値を読む場合かもしれません。その後、読んだ値は不正かもしれず、それが再び他の問題を引き起こすかもしれません。このような異常を防ぐため、相互排他を保証するための作業(タスク)通信の使い方を知ることが重要です。これは「作業(タスク)通信」項で検討されます。

右と下の図は自家醸造用の簡単なプログラムの流れと直線状プログラミングから並列または同時の作業(タスク)への移行がどのように見えるかを概説します。

図4-1. 直線状プログラミング

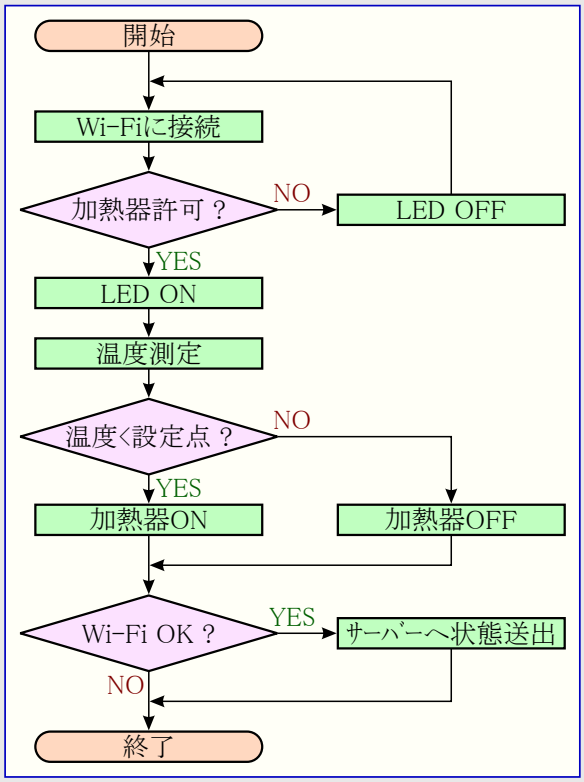
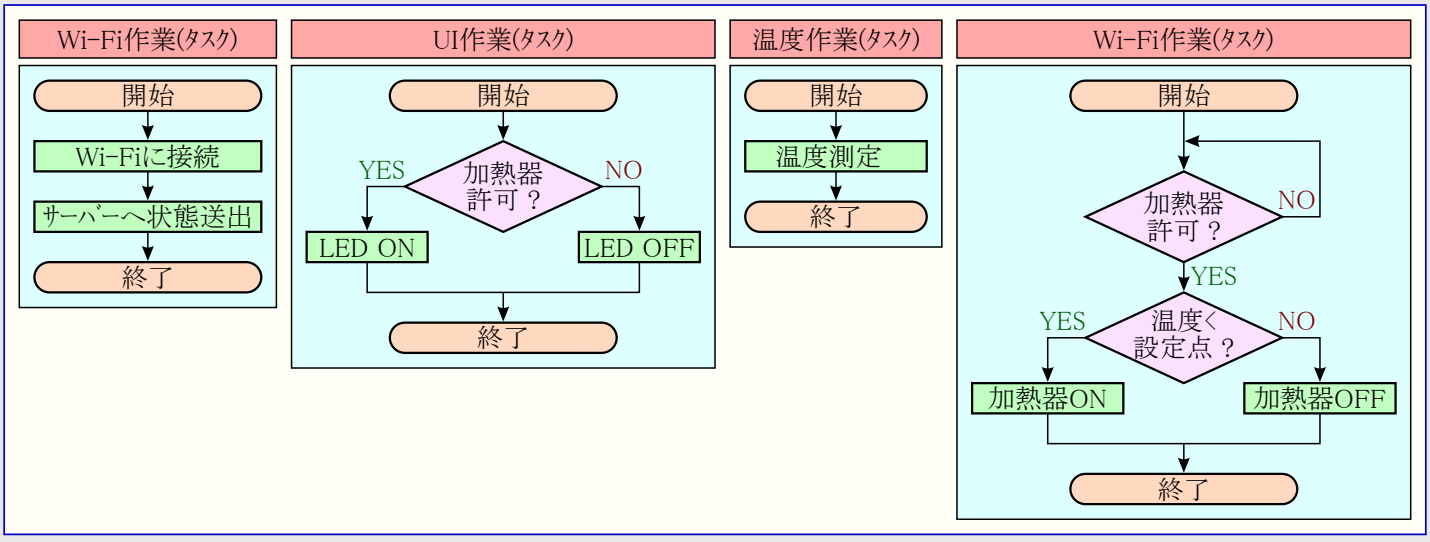


図4-2. 作業(タスク)駆動コード



4.1. 作業(タスク)

作業(タスク)はxTaskCreate(taskName, pcName, stackDepth, pvParameters, priority, pxCreatedTask)関数を呼ぶことによって作成されます。作業(タスク)は代表的に無限繰り返しとして実装されます。作業(タスク)を実装する方法は下をご覧ください。

```
xTaskCreate(exampleTask, "example", configMINIMAL_STACK_SIZE, NULL, 1, NULL);

void exampleTask(void *pvParams)
{
	while(1)
	{
		// 課題の作業を行ってください。
	}
}
```

4.2. 中断対非中断の機能

中断機能は継続実行から作業(タスク)を中断します。作業(タスク)が中断されると、RTOSは実行を違う作業(タスク)に切り替えます。これはCPUが何も行わない周期が決してないのでCPU利用を最適化します。

代表的な作業(タスク)はそれらが遅延関数を呼ぶ時、それらが通信や他の作業(タスク)によって使われる資源を待つ時、または割り込み(ISR)が呼ばれる時に中断します。vTaskDelay(ticksToDelay)を呼ぶと、作業(タスク)は指定された刻時数の間中断します。実際の時間、作業(タスク)は構成設定ファイルで指定された刻時速度に応じて中断します。

作業(タスク)が資源を共有すると、それらはそれら自身でそれを使い始めるのに先立って互いにその資源を使い終わるのを待たなければなりません。このように資源を待つことも中断と呼ばれます。このような動きを作成するためにセマフォとミューテックスを使うことができます。共有した資源と同期のもっと徹底的な説明は次章で見つかります。

4.3. 作業(タスク)通信

FreeRTOSは5つの主な作業(タスク)相互通信機構である、待ち行列(キュー)、セマフォ、ミューテックス、流れ緩衝部、メッセージ緩衝部を提供します。これら全てに共通するのは資源やデータが利用不能の場合に作業(タスク)を中断にすることができることです。例えば、作業(タスク)が待ち行列(キュー)全体に要素の挿入を望む場合、或る時間の間、待ち(中断)になり、その後に待ち行列内で何れか利用可能な空間があるかを調べます。

4.3.1. 待ち行列(キュー)

待ち行列は使用者定義可能な固定長の作業(タスク)相互通信を提供します。開発者は待ち行列を作成する時にメッセージ長を指定します。これはQueueHandle_t queueName = xQueueCreate(queueLength, elementSize)を呼ぶことによって行われます。queueLength入力パラメータは待ち行列が保持することができる要素数を指定します。elementSizeは各要素のバイトでの大きさを指定します。待ち行列内の全ての要素は等しい大きさでなければなりません。待ち行列が作成されると、作業(タスク)は待ち行列を通してデータを送受信することによって互いに通信することができます。待ち行列は受信側が最初に挿入された項目を常に受け取るようなFIFO構造(先入れ/先出し)です。

待ち行列に対する送りと受け取りの関数は次のとおりです。

```
xQueueSend(queueName, *itemToQueue, ticksToWait)
xQueueReceive(queueName, *buffer, ticksToWait)
```

最後の引数のticksToWaitは待ち行列全体で利用可能な空間を待つ時に、送る作業(タスク)がどの位の長さ中断すべきかを指定します。同様に、これは空の待ち行列から何かの要素の受け取りを待つ時に、受け取り作業(タスク)がどの位の長さ中断するかを指定します。portMAX_DELAY定義はここで使うことができます。構成設定ファイルでINCLUDE_vTaskSuspendが1に設定される場合、portMAX_DELAYは(制限時間を除いて)無期限に作業(タスク)を中断にさせます。0に設定する場合、中断時間は0xFFFFです。

送信と受信の関数がISRから呼ばれる場合、xQueueSendFromISR()とxQueueReceiveFromISR()が使われなければなりません。

4.3.2. セマフォ

セマフォは同期に使われ、作業(タスク)間で共有された資源へのアクセスを制御することです。セマフォは2値または計数のどちらかにすることができ、本質上、単に非負の整数計数です。

2値セマフォは1に初期化され、一度に1つの作業(タスク)によってだけ扱うことができる資源を保護するのに使うことができます。作業(タスク)が資源を取ると、セマフォは0に減少されます。別の作業(タスク)がその後にその資源の使用を望んでセマフォが0であることを知る場合、それは中断します。最初の作業(タスク)がその資源の使用を終了すると、セマフォは増加し、従って、他の作業(タスク)に対して利用可能になります。2値セマフォはSemaphoreHandle_t semaphoreName = xSemaphoreCreateBinary(void)で作成することができます。

計数セマフォは同じ規則で動きますが、一度に複数の作業(タスク)によって使うことができる資源用です。例えば、10台の車用の空き場所を持つ車庫を持つ場合、10セマフォのアクセスを許すことができます。0に達して誰かが出る前で誰も入庫を許されなくなるまで、車が入る毎にセマフォが1減少されます。計数セマフォは資源へのアクセスを同時に持ち得る作業(タスク)数に初期化されるべきで、SemaphoreHandle_tsemaphoreName = xSemaphoreCreateCounting(maxCount, initialCount)で作成されます。

作業(タスク)がセマフォによって保護された資源を欲する時はxSemaphoreTake(semaphoreName, ticksToWait)関数を呼びます。セマフォが0と等しい場合、作業(タスク)はticksToWaitで指定された時間の間中断します。作業(タスク)がセマフォの使用を終えた時にxSemaphoreGive(semaphoreName)関数が呼ばれます。

4.3.3. ミューテックス

ミューテックスは2値セマフォに良く似ていますが、加えて、優先権継承機構を提供します。高優先権作業(タスク)が既により低い優先権作業(タスク)によって取られた資源のアクセスから中断された場合、より低い優先権作業(タスク)はそれがミューテックスを開放してしまうまでその高優先権作業(タスク)の優先権を引き継ぎます。これは低優先権作業(タスク)が今や他の中優先権作業(タスク)によって先取りすることができないため、(元の)高優先権作業(タスク)の中断時間が最小化されることを保証します。

ミューテックスはsemaphoreHandle_t mutexName = xSemaphoreCreateMutex(void)関数を使うことによって作成されます。

優先権継承は割り込みに対してではなく作業(タスク)に対してだけ感知するため、ミューテックスは割り込みから使われるべきではありません。

4.3.4. 流れ緩衝部

流れ緩衝部は2つ作業(タスク)間またはISRから作業(タスク)へデータを転送します。待ち行列と異なり、流れ緩衝部は1つの読み手だけと1つの書き手だけがあると仮定します。流れ緩衝部作成時に緩衝部の最大の大きさが指定されます。名前が意味するように、流れ緩衝部は書き手と読み手間で転送されるバイトの流れを許します。バイトの流れは緩衝部の大きさ内である限り、任意の長さにすることができます。

流れ緩衝部はxStreamBufferCreate(BufferSizeBytes, TriggerLevelBytes)関数を使って作成されます。最初の入力パラメータは緩衝部が保持することができる総バイト数を指定します。2つ目の引数の起動水準は中断された受け手が中断状態外へ移されるのに先立って緩衝部になければならないバイト数を指定します。

送り手は緩衝部が満杯の場合に中断するかもしれません。データを待つ間に送り手がどの位の長さ中断されるべきかは送出関数で設定される制限時間によって与えられます。同様に、受け手は緩衝部が空の場合に中断します。

待ち行列と異なり、送り手は緩衝部にメッセージを置く前に準備が整った完全なメッセージを持つことを必要とされません。

送りと受けの関数が下で示されます。

```
xStreamBufferSend(streamBuffer, *pvTxData, dataLengthBytes, ticksToWait)
xStreamBufferReceive(streamBuffer, *pvRxData, bufferLengthBytes, ticksToWait)
```

4.3.5. メッセージ緩衝部

メッセージ緩衝部は流れ緩衝部に良く似たように動きますが、或る量のデータの要求を受け手に持たせる代わりに、メッセージの先頭部分がメッセージ長を指定します。加えて、メッセージは個別バイトとしてではなく、指定した長さとして読み出すことだけができます。

メッセージ緩衝部は流れ緩衝部を使って実装され、故にここで1つの読み手と1つの書き手だけの仮定も適用します。メッセージ緩衝部はまた、緩衝部が満杯の場合に送り手が中断し、緩衝部が空の場合に読み手が中断するような流れ緩衝部と同じ方法で動きます。

メッセージ緩衝部はxMessageBufferCreate(bufferSizeBytes)関数で作成されます。指定した大きさは含むことができるべき(メッセージではなく)緩衝部の総バイト数と等しくあるべきです。

データの送出と受け取りは下の関数を使うことによって行われます。

```
xMessageBufferSend(messageBuffer, *pvTxData, xDataLengthBytes, ticksToWait)
xMessageBufferReceive(messageBuffer, *pvRxData, bufferLengthBytes, ticksToWait)
```

4.4. 計画

計画はどの作業(タスク)を何時動かすかを決めるソフトウェアです。FreeRTOSは作業(タスク)優先権処理と言うことになると、先取り権の有りと無しの2つの動作形態を持ちます。どの動作を使うかは構成設定ファイルで設定することができます。協調動作とも呼ばれる先取り権なし動作使用時、中断機能とtaskYIELD()関数を通してCPUを明け渡す作業(タスク)にするのは開発者次第です。

先取り計画部使用時、作業(タスク)はより高い優先権の作業(タスク)が非中断になると、自動的にCPUを明け渡します。けれども、1つの例外があり、より高い作業(タスク)がISRから中断される時に、作業(タスク)切り替え発生のためにISRの最後でtaskYIELD_FROM_ISR()関数が呼ばれなければなりません。

configUSE_TIME_SLICINGが1に設定される場合、計画部は刻時発生毎に同じ優先権の作業(タスク)を先取りします。時分割は協調動作で利用不可です。

両動作で、計画部は常に中断されていない最高優先権の作業(タスク)に切り替えます。同じ優先権で中断されていない複数の作業(タスク)がある場合、計画部は各作業(タスク)を順に実行するように選びます。これは一般的にラウンドロビン計画として参照されます。

先取り動作ではより高い優先権作業(タスク)が非中断にされると直ぐに切り替えられます。協調動作ではセマフォの開放がより高い優先権作業(タスク)を非中断にするかもしれませんが、実際の作業(タスク)切り替えは現在実行中の作業(タスク)がtaskYIELD()関数を呼ぶ、または中断状態に入る時にだけ起きます。

計画のより多くの情報については「FreeRTOS実時間カーネルの習得 – 実践指導の手引き(Mastering the FreeRTOS Real Time Kernel – a Hands On Tutorial Guide)」で「計画算法(Scheduling Algorithms)」をご覧ください。

5. FreeRTOSでのﾃﾞﾊﾞｯｸﾞ

本章はいくつかの最も一般的な誤りを網羅してそれらをどう解決するかのいくつかの情報を提供をします。

5.1. ﾋｰﾌﾟ ﾃﾞﾊﾞｯｸﾞ

小さすぎる大きさのﾋｰﾌﾟ ﾒﾓﾘ割り当てや、応用に適合しないﾋｰﾌﾟ実装の選択のためにﾋｰﾌﾟに関する問題が発生するかもしれません。

xPortGetFreeHeapSize()関数は呼ばれた時に未割り当てにされたままのﾋｰﾌﾟ空間の総量を返します。これが非常に小さい場合、ﾋｰﾌﾟの大きさの増加が考慮されるかもしれません。

応用に適合していないﾋｰﾌﾟ実装選択も問題を起こし得ます。「ﾒﾓﾘ構成設定」項で言及したように、5つの異なるﾋｰﾌﾟ実装があります。それらのいくつかは割り当てたﾒﾓﾘを開放せず、一方で他は不揃いな大きさのﾒﾓﾘ割り当てでの問題を持ちます。特定応用に対するﾋｰﾌﾟ要件が考慮されなければなりません。各ﾋｰﾌﾟ実装のより多くの情報についてはhttps://www.freertos.org/a00111.htmlをご覧ください。

5.2. ｽﾀｯｸ溢れ検査

ｽﾀｯｸ溢れは非常に一般的な支援要請源ですが、FreeRTOSはこのような問題のﾃﾞﾊﾞｯｸﾞを手助けするいくつかの機能を提供します。

ｽﾀｯｸ溢れは作業(ﾀｽｸ)がｽﾀｯｸ上のそれ用にある空間よりも多くの要素を押し込もうとする時に起きます。作業(ﾀｽｸ)がそれのｽﾀｯｸで溢れる近くを探し出すのに、uxTaskGetStackHighWaterMark(TaskHandle_t xTask)関数を使うことができます。これは語で最小未使用ｽﾀｯｸを返します。8ﾋﾞｯﾄMCUが使われるなら、1の返り値は1ﾊﾞｲﾄを意味します。32ﾋﾞｯﾄMCUについて1語は4ﾊﾞｲﾄを意味します。この関数を使うには構成設定ﾌｧｲﾙでINCLUDE_uxTaskGetStackHighWaterMarkが1に設定されなければなりません。

走行時の間にｽﾀｯｸ溢れを調べることも可能です。configCHECK_FOR_STACK_OVERFLOWを1に設定することにより、ｶｰﾈﾙはｽﾀｯｸﾎﾟｲﾝﾀが有効なｽﾀｯｸ空間内に留まることを調べます。そうでなければ、ｽﾀｯｸ溢れ引っ掛け関数が呼ばれます。引っ掛け関数は何かが起きた時に反応して違う動きを提供することを応用に許す関数です。例えば、ｽﾀｯｸが溢れてしまった時にLEDをONにすることができます。別の任意選択は異常ﾒｯｾｰｼﾞを出力してｼｽﾃﾑを再起動することかもしれません。vApplicationStackOverflowHook()関数は応用によって提供されなければなりません。

FreeRTOSとｽﾀｯｸ溢れについてのより多くの情報に関してはhttps://www.freertos.org/Stacks-and-stack-overflow-checking.htmlをご覧ください。

5.3. 追跡

以前にFreeRTOS+Traceとして知られるTracealyzerは組み込み応用がどのように行動するかのﾃﾞｰﾀを収集する分析ﾂｰﾙです。このﾃﾞｰﾀは障害対策や応用の性能最適化の時に役立ちます。Tracealyzerはどの作業(ﾀｽｸ)が何時動くかを画像視覚化し、これは渇望のような問題を見つけるの役立ちます。Atmel StudioでTracealyzerをどう使うかの説明については「AN2783:ATmega4809でPercepio®追跡を使うFreeRTOS™」応用記述を参照してください。更に多くの情報についてはhttps://www.freertos.org/FreeRTOS-Plus/FreeRTOS_Plus_Trace/RTOS_Trace_Instructions.shtmlをご覧ください。

6. 実演

Atmel | STARTでのATmega4809用FreeRTOS例はここ(http://start.atmel.com/#examples/ATmega4809/FreeRTOS)で入手可能です。必要とされるﾊｰﾄﾞｳｪｱはATmega4809 Xplained ProとEXT3経由で接続されたOLED1 Xplained Proです。実演応用はOLED表示器で時間を表示している間、LEDを点滅します。釦押下は対応するLEDも点灯して同時に最後の釦事象を反映するようにOLED表示器を更新します。

6.1. 必要とされるﾊｰﾄﾞｳｪｱ

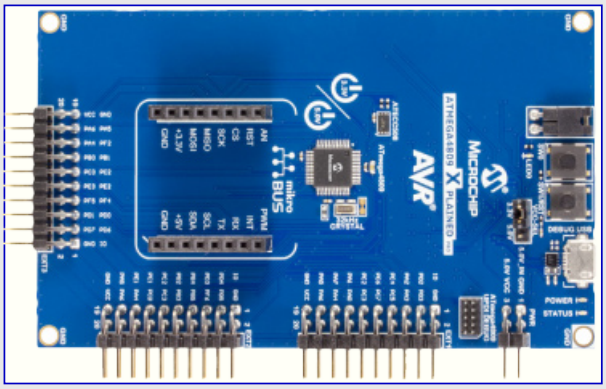
Atmel | START例ﾌﾟﾛｼﾞｪｸﾄに対して必要とされるﾊｰﾄﾞｳｪｱが下で記述されます。

6.1.1. ATmega4809 Xplained Pro評価ｷｯﾄ

ATmega4809 Xplained Pro評価ｷｯﾄはATmega4809 AVR®ﾏｲｸﾛ ｺﾝﾄﾛｰﾗ(MCU)を評価するためのﾊｰﾄﾞｳｪｱ基盤です。

ｳｪﾌﾞ頁: https://www.microchip.com/developmenttools/ProductDetails/atmega4809-xpro

図6-1. ATmega4809 Xplained Pro

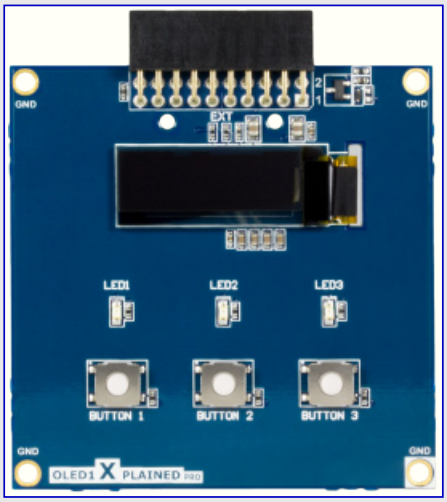


6.1.2. OLED1 Xplained Pro拡張キット

OLED1 Xplained Proは128×32 OLED表示器、3つのLED、3つの押し釦を持つ拡張キットです。これはどれかのXplained Pro評価キットの拡張ヘッダに接続します。

ウェブ頁: https://www.microchip.com/developmenttools/ProductDetails/ATOLED1-XPRO

図6-2. OLED1 Xplained Pro



6.2. 作業(タスク)への分割

例応用の全体的な目的はLED点滅、釦押下処理、OLED画面での時間表示です。「RTOS開発者のように考える」章で記述された考え方を再考してください。それに応じて各種の仕事が個別作業(タスク)に分割されます。

応用が何をすべきかを考えると、250ms毎にLEDを点滅する責任がある1つの作業(タスク)を持つことが有用に思えます。押された釦の処理も個別作業(タスク)であるべきです。同時書き込みのためのどんな衝突も引き起こすことなくOLEDに書くための何らかのクロック処理と方法の必要もあります。

この例は7つの作業(タスク)、2つの待ち行列(キュー)、1つのミューテックス、1つの流れ緩衝部、1つのメッセージ緩衝部を使います。「実装」項はこれらの作業(タスク)がどう実装されるかを詳細に検討します。下は各作業(タスク)とそれらが使う通信方法の短い説明です。

状態LED作業(タスク)

250ms毎にATmega4809 Xplained Pro上のLED0を点滅します。通信は使いません。

キーボード作業(タスク)

キー(釦)の1つの状態が変化したか、即ち、釦が押された、または開放されたかを検出します。key_queueと呼ばれる待ち行列(キュー)に更新したキー状態を送ります。

主作業(タスク)

key_queueを通して更新したキー状態を受け取り、OLED画面に出力される文字列を作成します。書く前に、作業(タスク)は画面を保護するoled_semaphoreと呼ばれるミューテックスに尋ねます。作業(タスク)はled_queueに押された釦についての情報も送ります。

LED作業(タスク)

led_queueを通して最後の釦事象についての情報を受け取り、それによってLEDの状態を設定します。

クロック作業(タスク)

秒毎に時間を増やし、oled_semaphoreを取った後にOLED画面に書きます。必要とするなら、新しい時間を設定することもできます。

端末送信作業(タスク)

terminal_tx_bufferと呼ばれるメッセージ緩衝部上のデータを受け取り、その後にそれをUSART送信緩衝部に置きます。

端末受信作業(タスク)

terminal_rx_bufferと呼ばれる流れ緩衝部を通して情報を受け取り、受け取ったメッセージに基づいてクロック作業(タスク)に新しい時間を設定することを要求します。

6.3. 共有される資源

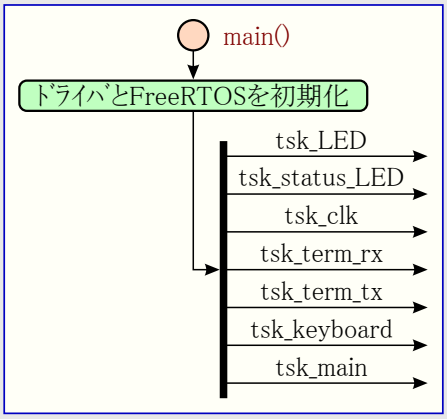
この応用での共有資源はOLED画面です。主作業とクロック作業の両作業(タスク)は表示器への書き込みを求めますが、同時に1つだけがそれを行うことができます。2つの作業(タスク)が同時に画面へ書こうとした場合、プログラムが破壊するかもしれず、またはいくつかの予期せぬ動きになるかもしれません。従って、OLEDへの書き込み処理はミューテックスによって保護されます。

6.4. 実装

本項は各作業(ﾀｽｸ)の実装の短い説明を提供します。各作業(ﾀｽｸ)に対するUML図も提供されます。UMLは統一模式化言語(Unifie d Modeling Language)を表し、ｼｽﾃﾑの設計を可視化する標準的な方法を提供します。この資料で使われる図について、FreeRTOSで行わなければならない部分は灰色で色付けされます。

図6-3.で示されるように、主(main)でﾄﾞﾗｲﾊﾞとFreeRTOSの初期化だけでなく、作業(ﾀｽｸ)の作成も行われます。

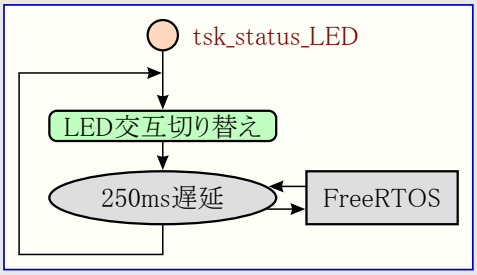
図6-3. 作業(ﾀｽｸ)とﾄﾞﾗｲﾊﾞの初期化



6.4.1. 状態LED作業(ﾀｽｸ)

これは多分、応用の最も簡単な作業(ﾀｽｸ)です。これはLED交互切り替え関数と250 msの遅延から成ります。遅延関数は入力ﾊﾟﾗﾒｰﾀとして刻時数を取ります。msを刻時に変換するのにpdMS_TO_TICKS(250)関数が使われます。UML図が**図6-4.**で示されます。

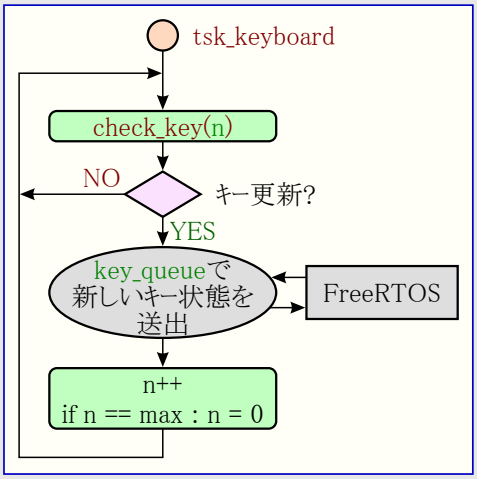
図6-4. 状態LED作業(ﾀｽｸ)



6.4.2. ｷｰﾎﾞｰﾄﾞ作業(ﾀｽｸ)

tsk_keyboard()ｷｰﾎﾞｰﾄﾞ作業(ﾀｽｸ)は釦押下を処理する作業(ﾀｽｸ)です。釦の1つの状態が変えられた場合、新しい状態がkey_queueに置かれます。key_queueの受け手は次項で記述される主作業(ﾀｽｸ)です。UML図については**図6-5.**を参照してください。

図6-5. ｷｰﾎﾞｰﾄﾞ作業(ﾀｽｸ)

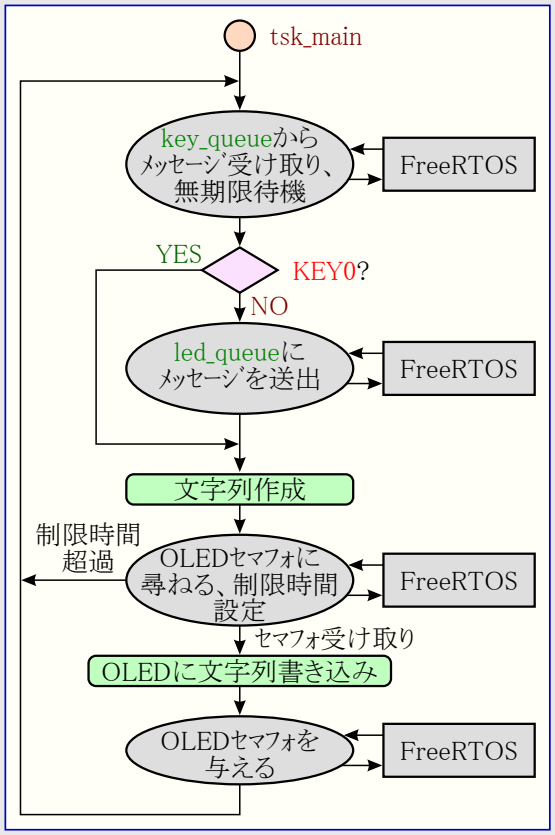


6.4.3. 主作業(ﾀｽｸ)

tsk_main()作業(ﾀｽｸ)はｷｰﾎﾞｰﾄﾞ作業(ﾀｽｸ)によってkey_queueに置かれたものを受け取ります。受け取った情報に基づいて最後の釦事象を反映する文字列を作成します。この文字列はその後、OLEDに書かれます。このような文字列の例は"Button 2 Released(釦2開放)"と"Button 1 Pushed(釦1押下)"です。

主作業(ﾀｽｸ)は対応するLEDが点灯するような押された釦についての情報をled_queueにも送ります。主作業(ﾀｽｸ)のUML図については**図6-6.**をご覧ください。

図6-6. 主作業(ﾀｽｸ)

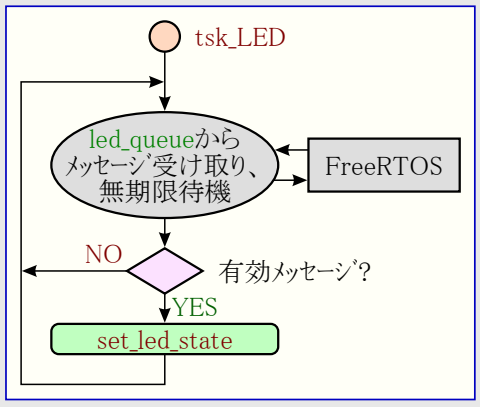


6.4.4. LED作業(ﾀｽｸ)

LED作業(ﾀｽｸ)の責任は釦押下と共にLEDを点灯し、押されていない釦のLEDをOFFにすることです。LEDは釦が押され続ける限り、点灯します。

最新の釦事象についての情報はled_queueを通して受け取られます。前項で言及されるように、この待ち行列(ｷｭｰ)へ送るのは主作業(ﾀｽｸ)です。UML図については**図6-7.**を参照してください。

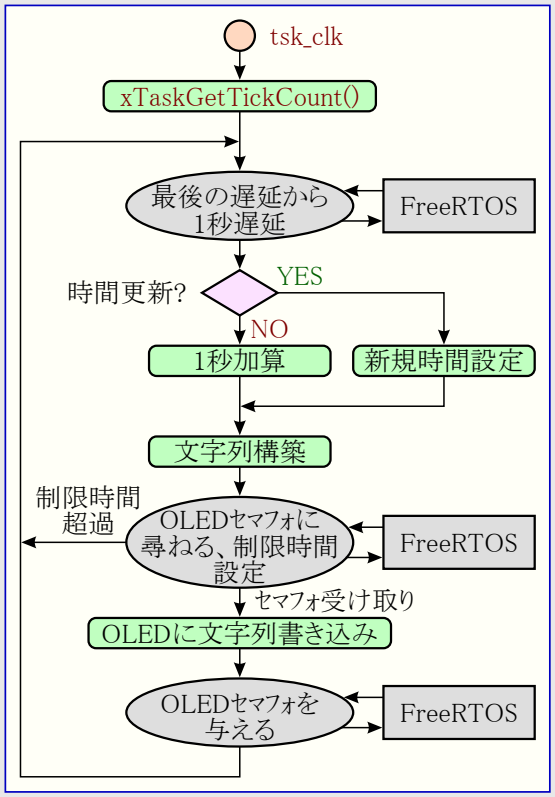
図6-7. LED作業(ﾀｽｸ)



6.4.5. クロック作業(タスク)

クロック作業(タスク)は秒毎にOLEDに時間を書く責任があります。これは書く時にoled_semaphoreを取ることによって行われます。クロック作業(タスク)は秒毎に時間を増す(進める)責任もあります。UML図については**図6-8.**をご覧ください。

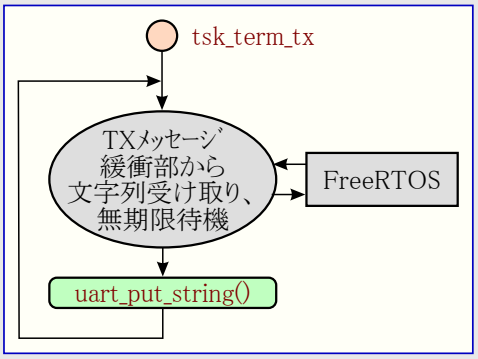
図6-8. クロック作業(タスク)



6.4.6. 端末送信作業(タスク)

図6-9.で示されるように、端末送信作業(タスク)はUSART経由でterminal_tx_bufferと呼ばれるメッセージ緩衝部で受け取ったものを送出する責任があります。

図6-9. 端末送信作業(タスク)



6.4.7. 端末受信作業(ﾀｽｸ)

ISRは受信したUSARTﾃﾞｰﾀをterminal_rx_bufferと呼ばれる流れ緩衝部に置きます。端末受信作業(ﾀｽｸ)はその後にterminal_rx_buffer上の受信ﾃﾞｰﾀを取り、受信したﾒｯｾｰｼﾞに基づいて制限時間超過が起きない限り、時間更新関数を呼びます。端末受信作業(ﾀｽｸ)は時間が設定されたか、または要求が制限時間超過にされたかの最終的な事象についてterminal_tx_buffer経由で端末送信作業(ﾀｽｸ)に報告を返します。UML図については**図6-10.**を参照してください。

図6-10. 端末受信作業(ﾀｽｸ)

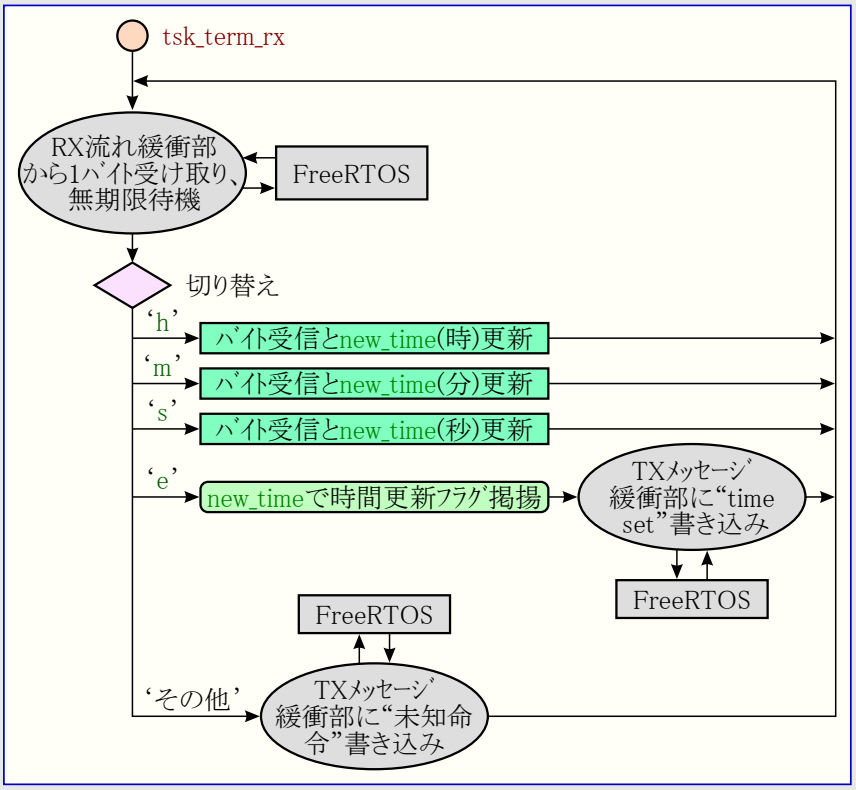


図6-11.は「ﾊﾞｲﾄ受信とnew_time更新」部のもっと詳細な説明を示します。

図6-11. ﾊﾞｲﾄ受信と時間更新

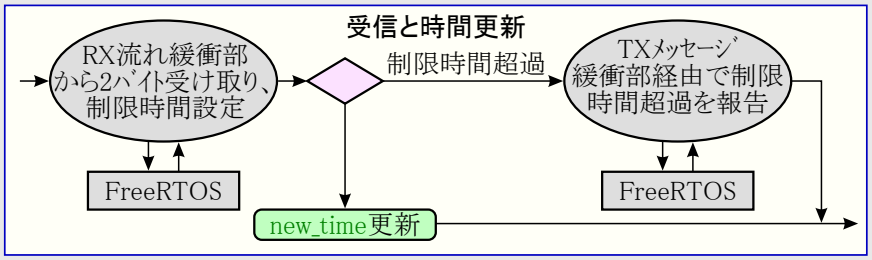
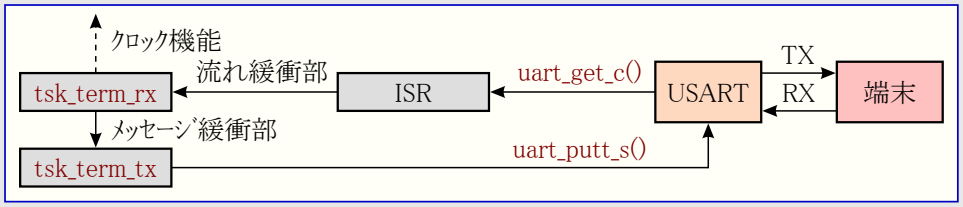


図6-12.は2つの端末作業(ﾀｽｸ)だけでなく、USART通信間の通信も記述します。

図6-12. 端末送信作業(ﾀｽｸ)と端末受信作業(ﾀｽｸ)間の通信



7. Atmel | STARTからのソース コード取得

コード例は画像使用者インターフェース(GUI)を通して応用コードの構成設定を許すウェブに基づくAtmel | STARTを通して利用可能です。コードは下の直接コード例リンクまたはAtmel | START先頭頁のBrowse examples(例検索)釦経由Atmel StudioとIAR Embedded Workbench®の両方に対してダウンロードすることができます。

Atmel | STARTウェブ頁：http://start.atmel.com/

コード例

ATmega4809 FreeRTOS例 (ATmega4809 FreeRTOS例)

例プロジェクトについての詳細と情報に関してはAtmel | STARTでUser guide(使用者の手引き)をクリックしてください。User guide釦はAtmel | STARTプロジェクト形態設定部内の一覧画面でプロジェクト名をクリックすることにより、例閲覧部で見つけることができます。

Atmel Studio

Download selected example(選んだ例をダウンロード)をクリックすることにより、Atmel | STARTで例閲覧部からAtmel Studio用.atzipファイルとしてコードをダウンロードしてください。Atmel | START内からファイルをダウンロードするには、Export project(プロジェクトをエクスポート)に続いてDownloaf pack(一括ダウンロード)をクリックしてください。

ダウンロードした.atzipファイルをダブル クリックしてください。プロジェクトがAtmel Studio 7.0に導入されます。

IAR Embedded Workbench

IAR Embedded Workbenchでプロジェクトをインポートする方法の情報についてはAtmel | START使用者の手引きを開き、Using Atmel Start Output in External Tools(外部ツールでAtmel START出力を使用)とIAR Embedded Workbenchを選んでください。Atmel | START使用者の手引きへのリンクは共に頁の右上隅に置かれたAtmel | START先頭頁からHelp(手助け)またはプロジェクト形態設定部内のHelp And Support(手助けと支援)をクリックすることによって見つけることができます。

8. 改訂履歴

資料改訂	日付	注釈
A	2019年4月	初版資料公開

Microchipｳｪﾌﾞ ｻｲﾄ

Microchipはhttp://www.microchip.com/で当社のｳｪﾌﾞ ｻｲﾄ経由でのｵﾝﾗｲﾝ支援を提供します。このｳｪﾌﾞ ｻｲﾄはお客様がﾌｧｲﾙや情報を容易に利用可能にする手段として使われます。お気に入りのｲﾝﾀｰﾈｯﾄ ﾌﾞﾗｳｻﾞを用いてｱｸｾｽすることができ、ｳｪﾌﾞ ｻｲﾄは以下の情報を含みます。

- **製品支援** – ﾃﾞｰﾀｼｰﾄと障害情報、応用記述と試供ﾌﾟﾛｸﾞﾗﾑ、設計資源、使用者の手引きとﾊｰﾄﾞｳｪｱ支援資料、最新ｿﾌﾄｳｪｱ配布と保管されたｿﾌﾄｳｪｱ
- **全般的な技術支援** – 良くある質問(FAQ)、技術支援要求、ｵﾝﾗｲﾝ検討ｸﾞﾙｰﾌﾟ、Microchip相談役ﾌﾟﾛｸﾞﾗﾑ員一覧
- **Microshipの事業** – 製品選択器と注文の手引き、最新Microchip報道発表、ｾﾐﾅｰとｲﾍﾞﾝﾄの一覧、Microchip営業所の一覧、代理店と代表する工場

お客様への変更通知ｻｰﾋﾞｽ

Microchipのお客様通知ｻｰﾋﾞｽはMicrochip製品を最新に保つのに役立ちます。加入者は指定した製品系統や興味のある開発ﾂｰﾙに関連する変更、更新、改訂、障害情報がある場合に必ず電子ﾒｰﾙ通知を受け取ります。

登録するにはhttp://www.microchip.com/でMicrochipのｳｪﾌﾞ ｻｲﾄをｱｸｾｽしてください。”Support”下で”Customer Change Notification”をｸﾘｯｸして登録指示に従ってください。

お客様支援

Microchip製品の使用者は以下のいくつかのﾁｬﾈﾙを通して支援を受け取ることができます。

- 代理店または販売会社
- 最寄りの営業所
- 現場応用技術者(FAE:Field Aplication Engineer)
- 技術支援

お客様は支援に関してこれらの代理店、販売会社、または現場応用技術者(FAE)に連絡を取るべきです。最寄りの営業所もお客様の手助けに利用できます。営業所と位置の一覧はこの資料の後ろに含まれます。

技術支援はhttp://www.microchip.com/supportでのｳｪﾌﾞ ｻｲﾄを通して利用できます。

Microchipﾃﾞﾊﾞｲｽ ｺｰﾄﾞ保護機能

Microchipﾃﾞﾊﾞｲｽでの以下のｺｰﾄﾞ保護機能の詳細に注意してください。

- Microchip製品はそれら特定のMicrochipﾃﾞｰﾀｼｰﾄに含まれる仕様に合致します。
- Microchipは意図した方法と通常条件下で使用される時に、その製品系統が今日の市場でその種類の最も安全な系統の1つであると考えます。
- ｺｰﾄﾞ保護機能を破るのに使用される不正でおそらく違法な方法があります。当社の知る限りこれらの方法の全てはMicrochipのﾃﾞｰﾀｼｰﾄに含まれた動作仕様外の方法でMicrochip製品を使用することが必要です。おそらく、それを行う人は知的財産の窃盗に関与しています。
- Microchipはそれらのｺｰﾄﾞの完全性について心配されているお客様と共に働きたいと思います。
- Microchipや他のどの半導体製造業者もそれらのｺｰﾄﾞの安全を保証することはできません。ｺｰﾄﾞ保護は当社が製品を”破ることができない”として保証すると言うことを意味しません。

ｺｰﾄﾞ保護は常に進化しています。Microchipは当社製品のｺｰﾄﾞ保護機能を継続的に改善することを約束します。Microchipのｺｰﾄﾞ保護機能を破る試みはﾃﾞｼﾞﾀﾙ ﾐﾚﾆｱﾑ著作権法に違反するかもしれません。そのような行為があなたのｿﾌﾄｳｪｱや他の著作物に不正なｱｸｾｽを許す場合、その法律下の救済のために訴権を持つかもしれません。

法的通知

ﾃﾞﾊﾞｲｽ応用などに関してこの刊行物に含まれる情報は皆さまの便宜のためにだけ提供され、更新によって取り換えられるかもしれません。皆さまの応用が皆さまの仕様に合致するのを保証するのは皆さまの責任です。**Microchipはその条件、品質、性能、商品性、目的適合性を含め、明示的にも黙示的にもその情報に関連して書面または表記された書面または黙示の如何なる表明や保証もしません。**Microchipはこの情報とそれの使用から生じる全責任を否認します。生命維持や安全応用でのMicrochipﾃﾞﾊﾞｲｽの使用は完全に購入者の危険性で、購入者はそのような使用に起因する全ての損害、請求、訴訟、費用からMicrochipを擁護し、補償し、免責にすることに同意します。他に言及されない限り、Microchipのどの知的財産権下でも暗黙的または違う方法で許認可は譲渡されません。

商標

DNVによって認証された品質管理ｼｽﾃﾑ

ISO/TS 16949

Microchipはその世界的な本社、ｱﾘｿﾞﾅ州のﾁｬﾝﾄﾞﾗｰとﾃﾝﾍﾟ、ｵﾚｺﾞﾝ州ｸﾞﾗｼｬﾑの設計とｳｪﾊｰ製造設備とｶﾘﾌｫﾙﾆｱとｲﾝﾄﾞの設計ｾﾝﾀｰに対してISO/TS-16949:2009認証を取得しました。当社の品質ｼｽﾃﾑの処理と手続きはPIC® MCUとdsPIC® DSC、KEELOQ符号飛び回りﾃﾞﾊﾞｲｽ、直列EEPROM、ﾏｲｸﾛ周辺機能、不揮発性ﾒﾓﾘ、ｱﾅﾛｸﾞ製品用です。加えて、開発ｼｽﾃﾑの設計と製造のためのMicrochipの品質ｼｽﾃﾑはISO 9001:2000認証取得です。

日本語© *HERO* 2019.

本応用記述はMicrochipのAN3007応用記述(DS00003007A-2019年4月)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。

世界的な販売とサービス

米国	亜細亜/太平洋	亜細亜/太平洋	欧州
本社 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 技術支援: http://www.microchip.com/support ｳｪﾌﾞｱﾄﾞﾚｽ: www.microchip.com **ｱﾄﾗﾝﾀ** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **ｵｰｽﾁﾝ** TX Tel: 512-257-3370 **ﾎﾞｽﾄﾝ** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **ｼｶｺﾞ** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **ﾀﾞﾗｽ** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **ﾃﾞﾄﾛｲﾄ** Novi, MI Tel: 248-848-4000 **ﾋｭｰｽﾄﾝ** TX Tel: 281-894-5983 **ｲﾝﾃﾞｱﾅﾎﾟﾘｽ** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **ﾛｻﾝｾﾞﾙｽ** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **ﾛｰﾘｰ** NC Tel: 919-844-7510 **ﾆｭｰﾖｰｸ** NY Tel: 631-435-6000 **ｻﾝﾎｾ** CA Tel: 408-735-9110 Tel: 408-436-4270 **ｶﾅﾀﾞ - ﾄﾛﾝﾄ** Tel: 905-695-1980 Fax: 905-695-2078	**ｵｰｽﾄﾗﾘｱ - ｼﾄﾞﾆｰ** Tel: 61-2-9868-6733 **中国 - 北京** Tel: 86-10-8569-7000 **中国 - 成都** Tel: 86-28-8665-5511 **中国 - 重慶** Tel: 86-23-8980-9588 **中国 - 東莞** Tel: 86-769-8702-9880 **中国 - 広州** Tel: 86-20-8755-8029 **中国 - 杭州** Tel: 86-571-8792-8115 **中国 - 香港特別行政区** Tel: 852-2943-5100 **中国 - 南京** Tel: 86-25-8473-2460 **中国 - 青島** Tel: 86-532-8502-7355 **中国 - 上海** Tel: 86-21-3326-8000 **中国 - 瀋陽** Tel: 86-24-2334-2829 **中国 - 深圳** Tel: 86-755-8864-2200 **中国 - 蘇州** Tel: 86-186-6233-1526 **中国 - 武漢** Tel: 86-27-5980-5300 **中国 - 西安** Tel: 86-29-8833-7252 **中国 - 廈門** Tel: 86-592-2388138 **中国 - 珠海** Tel: 86-756-3210040	**ｲﾝﾄﾞ - ﾊﾝｶﾞﾛｰﾙ** Tel: 91-80-3090-4444 **ｲﾝﾄﾞ - ﾆｭｰﾃﾞﾘｰ** Tel: 91-11-4160-8631 **ｲﾝﾄﾞ - ﾌﾟﾈｰ** Tel: 91-20-4121-0141 **日本 - 大阪** Tel: 81-6-6152-7160 **日本 - 東京** Tel: 81-3-6880-3770 **韓国 - 大邱** Tel: 82-53-744-4301 **韓国 - ｿｳﾙ** Tel: 82-2-554-7200 **ﾏﾚｰｼｱ - ｸｱﾗﾙﾝﾌﾟｰﾙ** Tel: 60-3-7651-7906 **ﾏﾚｰｼｱ - ﾍﾟﾅﾝ** Tel: 60-4-227-8870 **ﾌｨﾘﾋﾟﾝ - ﾏﾆﾗ** Tel: 63-2-634-9065 **ｼﾝｶﾞﾎﾟｰﾙ** Tel: 65-6334-8870 **台湾 - 新竹** Tel: 886-3-577-8366 **台湾 - 高雄** Tel: 886-7-213-7830 **台湾 - 台北** Tel: 886-2-2508-8600 **ﾀｲ - ﾊﾞﾝｺｸ** Tel: 66-2-694-1351 **ﾍﾞﾄﾅﾑ - ﾎｰﾁﾐﾝ** Tel: 84-28-5448-2100	**ｵｰｽﾄﾘｱ - ｳﾞｪﾙｽ** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **ﾃﾞﾝﾏｰｸ - ｺﾍﾟﾝﾊｰｹﾞﾝ** Tel: 45-4450-2828 Fax: 45-4485-2829 **ﾌｨﾝﾗﾝﾄﾞ - ｴｽﾎﾟｰ** Tel: 358-9-4520-820 **ﾌﾗﾝｽ - ﾊﾟﾘ** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **ﾄﾞｲﾂ - ｶﾞﾙﾋﾝｸﾞ** Tel: 49-8931-9700 **ﾄﾞｲﾂ - ﾊｰﾝ** Tel: 49-2129-3766400 **ﾄﾞｲﾂ - ﾊｲﾙﾌﾞﾛﾝ** Tel: 49-7131-67-3636 **ﾄﾞｲﾂ - ｶｰﾙｽﾙｰｴ** Tel: 49-721-625370 **ﾄﾞｲﾂ - ﾐｭﾝﾍﾝ** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **ﾄﾞｲﾂ - ﾛｰｾﾞﾝﾊｲﾑ** Tel: 49-8031-354-560 **ｲｽﾗｴﾙ - ﾗｰﾅﾅ** Tel: 972-9-744-7705 **ｲﾀﾘｱ - ﾐﾗﾉ** Tel: 39-0331-742611 Fax: 39-0331-466781 **ｲﾀﾘｱ - ﾊﾟﾄﾞﾊﾞ** Tel: 39-049-7625286 **ｵﾗﾝﾀﾞ - ﾃﾞﾙｰﾈﾝ** Tel: 31-416-690399 Fax: 31-416-690340 **ﾉﾙｳｪｰ - ﾄﾛﾝﾊｲﾑ** Tel: 47-72884388 **ﾎﾟｰﾗﾝﾄﾞ - ﾜﾙｼｬﾜ** Tel: 48-22-3325737 **ﾙｰﾏﾆｱ - ﾌﾞｶﾚｽﾄ** Tel: 40-21-407-87-50 **ｽﾍﾟｲﾝ - ﾏﾄﾞﾘｰﾄﾞ** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **ｽｳｪｰﾃﾞﾝ - ｲｪｰﾃﾎﾞﾘ** Tel: 46-31-704-60-40 **ｽｳｪｰﾃﾞﾝ - ｽﾄｯｸﾎﾙﾑ** Tel: 46-8-5090-4654 **ｲｷﾞﾘｽ - ｳｫｰｷﾝｶﾞﾑ** Tel: 44-118-921-5800 Fax: 44-118-921-5820