



Systèmes et réseaux temps réel  
Chap IV: Gestion des files d'attente (Queues)

Systèmes et réseaux temps réel  
(youssefrochdi@yahoo.fr)

74



### Objectifs

- Définir et Caractériser une file d'attente (Queue)
- Apprendre à
  - créer/supprimer une file d'attente (Queue).
  - Envoyer (Ecrire) dans une /Retirer (Lire) à partir d'une Queue.
- Comprendre le blocage sur une Queue.
- Comprendre l'intérêt des Queues de données comme moyen de communication entre les tâches.
- Influence des priorités/durée de blocage sur l'état de la Queue.

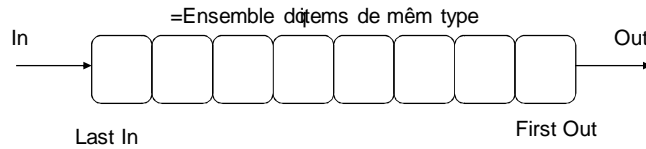
---

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr)

75

## 1- Définition d'une File d'attente (Queue)

Structure de données (FIFO First In First Out)



Caractéristiques:

- "Handle (Identificateur)
- "Nombre maxi d'items (capacité)
- "Taille mémoire de chaque item (Type)
- "Niveau de remplissage

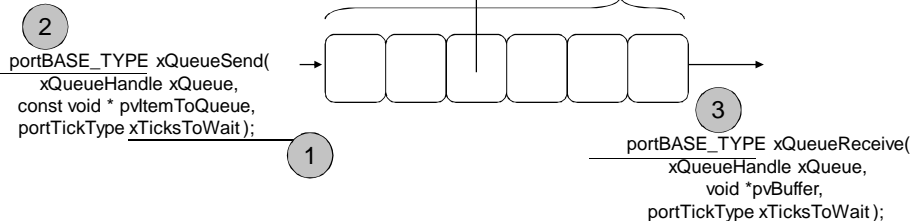
Opérations:

- "Créer/Supprimer une file
- "Insérer (à la fin)
- "Retirer(depuis le début)
- "Tester niveau de remplissage

## 2- Opérations sur les Queues (API freertos) (1)

Création / Ecriture /Lecture

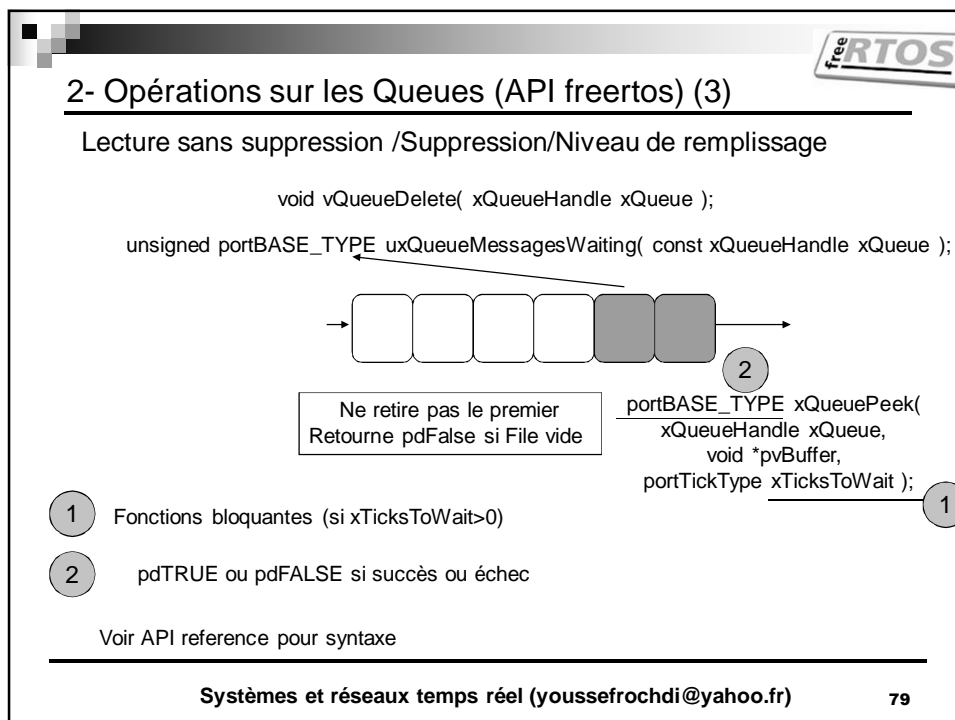
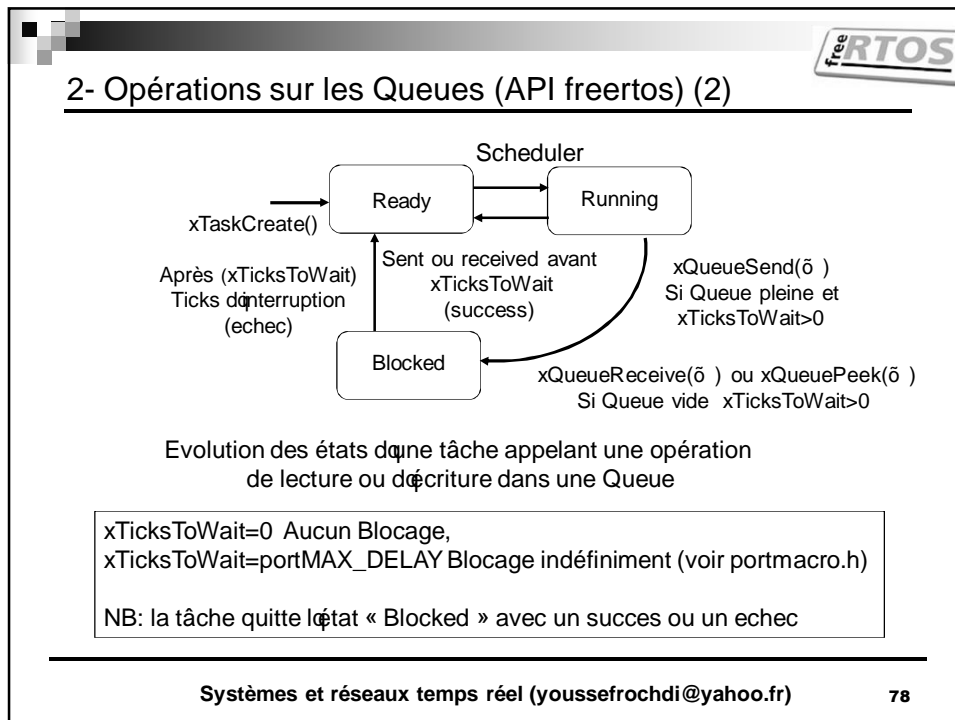
```
xQueueHandle xQueueCreate( unsigned portBASE_TYPE uxQueueLength,  
                           unsigned portBASE_TYPE uxItemSize );
```




1 Fonctions bloquantes (si xTicksToWait>0 et Queue vide (lecture)  
ou Queue pleine (Ecriture))

2 pdTRUE ou errQUEUE\_FULL si succès ou échec

3 pdTRUE ou pdFalse si succès ou échec Voir API reference pour syntaxe

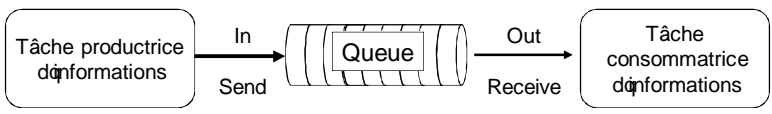




### 3- Intérêts (1)


- Les tâches peuvent être vues comme des petits programmes « autonomes » ; à tout moment au plus une tâche est en exécution
- Toutefois, ces tâches constituent une seule application; souvent les tâches doivent collaborer à travers un échange d'informations

→ Les files peuvent servir comme moyens de communications d'informations (messages) entre tâches.



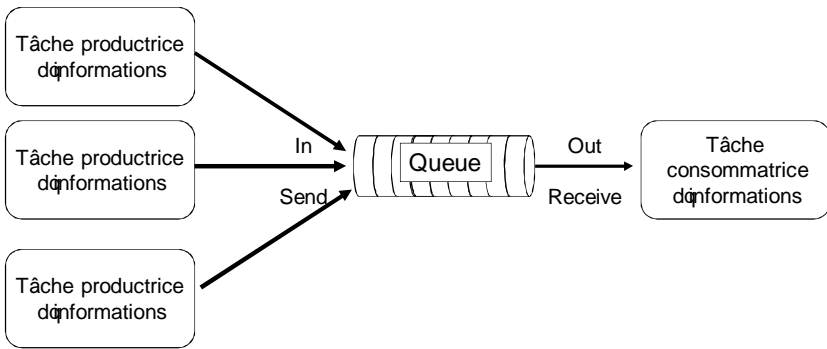
---

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 80



### 3- Intérêts (2)

- Une queue n'est pas forcément liée à une seule tâche et on peut avoir d'autres types d'échange: par exemple plusieurs-à-1.
- Exemple: Affichage de messages issues de plusieurs tâches sur un LCD d'un système embarqué



---

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 81

freeRTOS

### 3- Intérêts (3)

- Ou 1-à-plusieurs ou plusieurs à plusieurs.
- Un peu plus rare

Tâche productrice d'informations → In / Send → Queue → Out / Receive → Tâche consommatrice d'informations

---

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 82

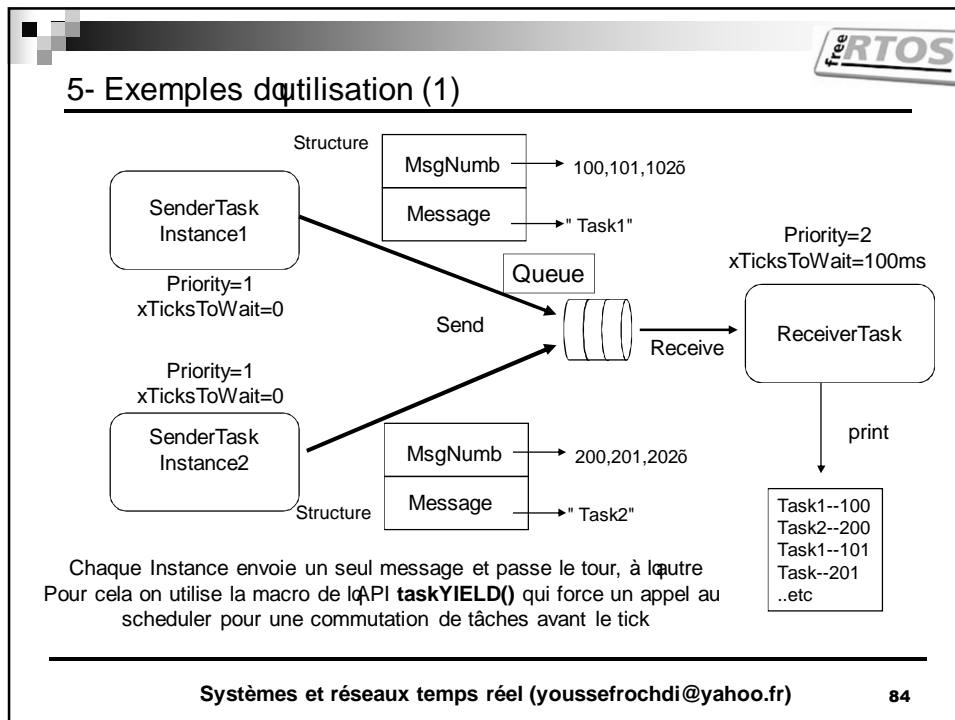
freeRTOS

### 4- Blocage multiple

- Si plusieurs tâches bloquées, en attente d'une lecture sur une même queue vide, la plus prioritaire parmi ces tâches se débloque au moment où un élément est écrit dans la Queue par une autre tâche (Déblocage asynchrone).
- Si plusieurs tâches bloquées, en attente d'une écriture dans une même queue pleine, la plus prioritaire parmi ces tâches se débloque au moment où un élément est lu de la Queue par une autre tâche (Déblocage asynchrone).
- Si ces tâches ont même priorités, elles se débloquent à tour de rôle (Tourniquet) → file d'attente FIFO des tâches bloquées de même priorité en attente de la disponibilité de la Queue.

---

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 83



**freeRTOS**


### 5- Exemples d'utilisation (2)

```

/*A task to send item to the queue.*/
static void vSenderTask( void *pvParameters )
{
    portBASE_TYPE xStatus;

    /* ... */
    for( ;; )
    {
        /* ... */
        xStatus = xQueueSendToBack( xQueue, pvParameters, 0 );
        if( xStatus != pdPASS )
        {
            /* The send operation could not complete because the queue was full */
            vPrintString( "Could not send to the queue.\r\n" );
        }
        else //Item successfully sent, increment Msg Number
        {
            ((ItemStruct*)pvParameters)->usMsgNumb++;
        }
        /* ... */
        taskYIELD(); // Force une commutation de tâche avant le tick
    }
}
  
```

**Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr)** **85**




### 5- Exemples d'utilisation (3)

```
static void vReceiverTask( void *pvParameters )
{
    /* Declare the variable that will hold the values received from the queue. */
    ItemStruct AnItem;
    portBASE_TYPE xStatus;
    /*Define de Number of Ticks to wait in blocked state : 100 ms*/
    const portTickType xTicksToWait = 100 / portTICK_RATE_MS;
    /* This task is also defined within an infinite loop. */
    for( ;; ) {
        /* ... */
        if( uxQueueMessagesWaiting( xQueue ) != 0 )
        {
            vPrintString( "Queue should have been empty!\r\n" );
        }
        /* ... */
        xStatus = xQueueReceive( xQueue, &AnItem, xTicksToWait );
        if( xStatus == pdPASS )
        {
            /* ... */
            vPrintStringAndNumber( AnItem.ucMessage, AnItem.usMsgNumb );
        }
        else { /* ... */
            vPrintString( "Could not receive from the queue.\r\n" );
        }
    }
}
```

---

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 86



### 5- Exemples d'utilisation (4)

```
xQueue = xQueueCreate( 3, sizeof( ItemStruct ) );
if( xQueue == 0 )
{
    /* Failed to create the queue. */
    printf( "%s", "Mémoire insuffisante..." );
    exit( 0 );
}
/* ... */
strcpy( AnItem1.ucMessage, "Task1" );
AnItem1.usMsgNumb = 100;
xTaskCreate( vSenderTask, "Sender1", 1000, ( void * ) &AnItem1, mainSEND_PRIOR, NULL );
strcpy( AnItem2.ucMessage, "Task2" );
AnItem2.usMsgNumb = 200;
xTaskCreate( vSenderTask, "Sender2", 1000, ( void * ) &AnItem2, mainSEND_PRIOR, NULL );
/* ... */
xTaskCreate( vReceiverTask, "Receiver", 1000, NULL, mainRECEIV_PRIOR, NULL );
/* Start the scheduler so our tasks start executing. */
vTaskStartScheduler();
/* ... */
return 0;
}
```

---

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 87

## 5- Exemples d'utilisation (5)

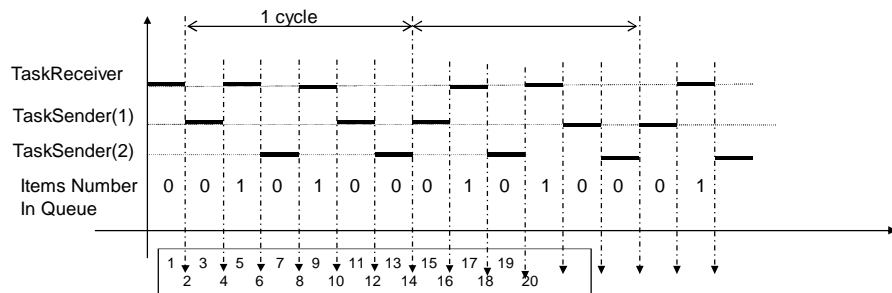
```
Trace started. Hit a key to dump trace file to disk.
Task1--100
Task2--200
Task1--101
Task2--201
Task1--102
Task2--202
Task1--103
Task2--203
Task1--104
Task2--204
Task1--105
Task2--205
Task1--106
Task2--206
Task1--107
Task2--207
Task1--108
Task2--208
Task1--109
Task2--209
Task1--110
```

Pour ralentir la fréquence des échanges TicksToWait=2s et  
#define configTICK\_RATE\_HZ( 1 )

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr)

88

## 5- Exemples d'utilisation (6)



- 1- Rec tente de lire dans la file vide.
- 2- Rec se bloque pour max 100ms.
- 3- Sen1 envoie l'item N°100 dans la file.
- 4- Rec se débloque et préempte Sen1 (asynchrone)
- 5- Rec lit et affiche item N°100 et tente de lire un autre.
- 6- Rec se bloque pour max 100ms.
- 7- Sen2 envoie l'item N°200 dans la file.
- 8- Rec se débloque et préempte Sen2. (asynchrone)
- 9- Rec lit et affiche l'item N°200 et tente de lire un autre
- 10- Rec se bloque pour max 100ms.
- 11- Sen1 incrémente compteur et appelle TaskYield (Async).

- 12- Scheduler trouve une autre tâche Sen2 prête de mm priorité que Sen1 : commutation forcée Sen1→Sen2
- 13- Sen2 incrémente compteur et appelle TaskYield (Async).
- 14- Scheduler trouve une autre tâche Sen1 prête de mm priorité que Sen2 : commutation forcée Sen2→Sen1
- 15- Sen1 envoie item N°101.
- 16- Rec se débloque et préempte Sen1. (asynchrone)
- 17- Rec lit et affiche item N°101 et tente de lire un autre.
- 18- Rec se bloque pour max 100ms.
- 19- Sen2 envoie l'item N°201 dans la file.
- 20- Rec se débloque et préempte Sen2. (asynchrone)

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr)

89

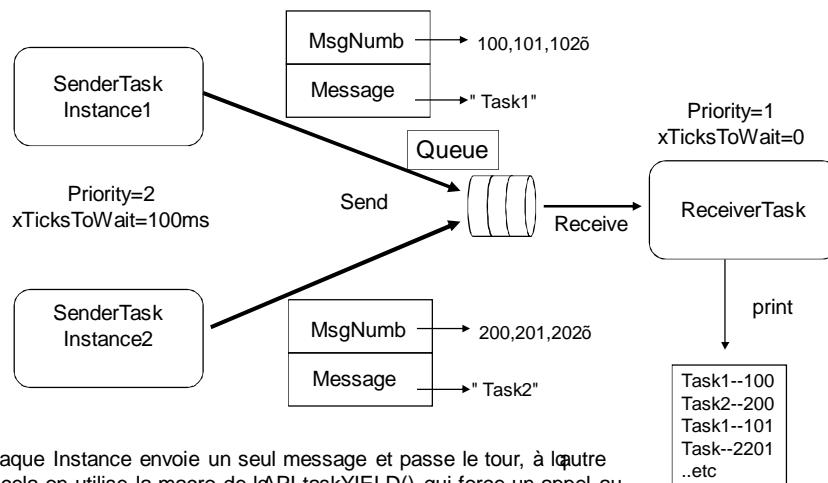


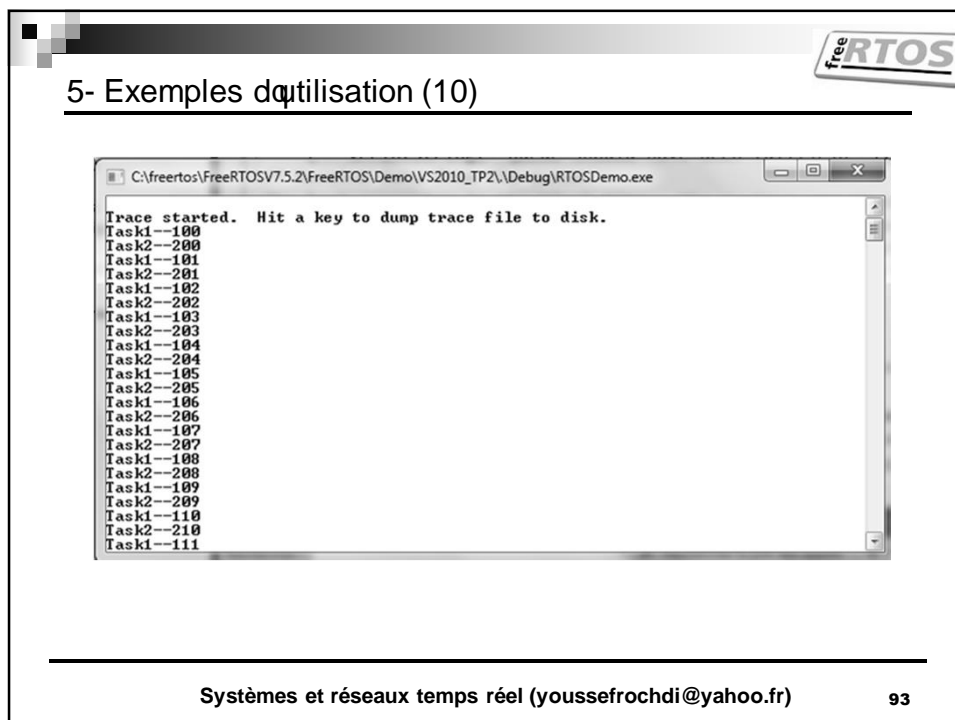
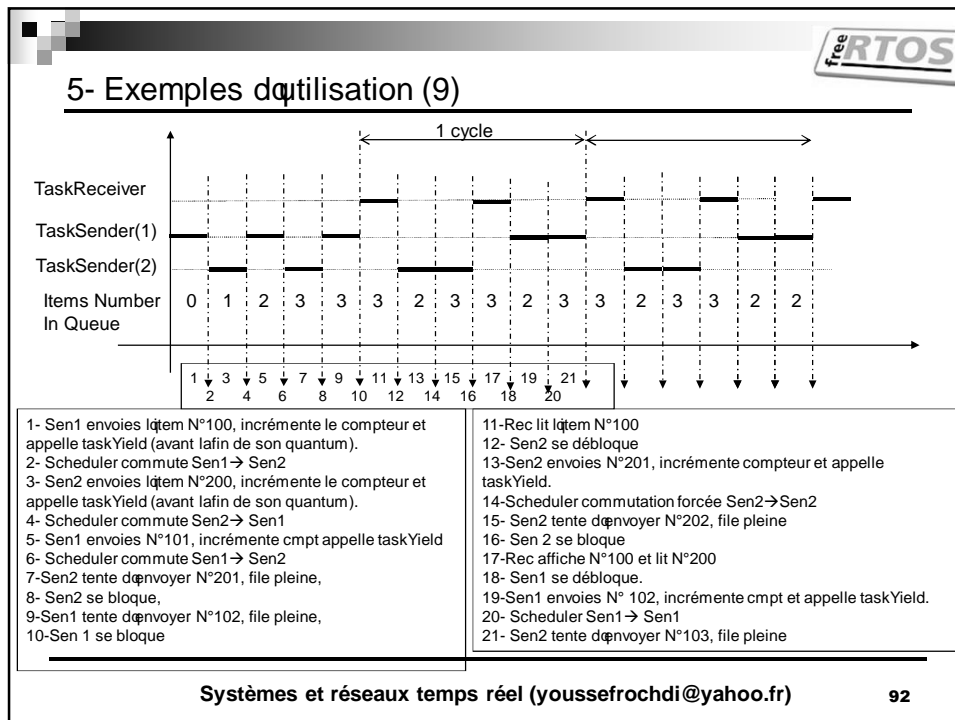
## 5- Exemples d'utilisation (7)

### ■ Conclusions:

- La file ne se remplit jamais (au plus elle contient 1 item): dès qu'un sender envoie le receiver, plus prioritaire, se débloque et lit et affiche.
- L'appel à taskYield() (commutation forcée avant le tick) ne sert à rien dans ce cas on peut l'enlever l'affichage obtenu reste le même.
- NB: la préemption des senders par le receiver se fait de manière asynchrone.

## 5- Exemples d'utilisation (8)





## 5- Exemples d'utilisation (11)


### ■ Conclusions:

- La file est presque toujours pleine (au min elle contient 2 item): dès que Receiver lit un item, un sender, plus prioritaire, se débloque et envoie un item pour remplir à nouveau la file.
- Le appel à taskYield() (commutation forcée avant le tick) sert au début pour permettre aux Senders d'envoyer chacun un item alternativement. Après elle ne sert plus à rien.
- Si taskYield est enlevée et si le quantum est suffisamment élevé l'affichage sera:

```
Task1 . 100  
Task1 . 101  
Task1 . 102  
Task2 . 200  
Task1 . 103  
Task2 . 201  
Task1 . 104
```



Systèmes et réseaux temps réel  
Chap V: Etude de cas  
Implémentation de freertos sur une plateforme  
basée sur un PIC 16 bits (Microchip  
PIC24FJ128GA010)




## Objectifs

- Comprendre le principe de portabilité de freertos en étudiant une implémentation particulière.
- Se familiariser avec les outils de développements et de debugage d'applications pour PIC.
- Développer et Tester des petites applications multitâches sur une plateforme basée sur un PIC 16bits.
- Vérifier les concepts de base étudiés jusqu'à maintenant.

---

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 96




## 1- Caractéristiques du PIC24FJ128GA010

<b>High-Performance CPU:</b> <ul style="list-style-type: none"><li>• Modified Harvard Architecture</li><li>• Up to 16 MIPS Operation @ 32 MHz</li><li>• 8 MHz Internal Oscillator with 4x PLL Option and Multiple Divide Options</li><li>• 17-Bit x 17-Bit Single-Cycle Hardware Multiplier</li><li>• 32-Bit by 16-Bit Hardware Divider</li><li>• 16 x 16-Bit Working Register Array</li><li>• C Compiler Optimized Instruction Set Architecture:<ul style="list-style-type: none"><li>- 76 base instructions</li><li>- Flexible addressing modes</li></ul></li><li>• Two Address Generation Units for Separate Read and Write Addressing of Data Memory</li></ul>	<ul style="list-style-type: none"><li>• Hardware Real-Time Clock/Calendar (RTCC):<ul style="list-style-type: none"><li>- Provides clock, calendar and alarm functions</li></ul></li><li>• Programmable Cyclic Redundancy Check (CRC)<ul style="list-style-type: none"><li>- User-programmable polynomial</li><li>- 8/16-level FIFO buffer</li></ul></li><li>• Five 16-Bit Timers/Counters with Programmable Prescaler</li><li>• Five 16-Bit Capture Inputs</li><li>• Five 16-Bit Compare/PWM Outputs</li><li>• High-Current Sink/Source (18 mA/18 mA) on All I/O Pins</li><li>• Configurable, Open-Drain Output on Digital I/O Pins</li><li>• Up to 5 External Interrupt Sources</li><li>• 5.5V Tolerant Input (digital pins only)</li></ul>
<b>Peripheral Features:</b> <ul style="list-style-type: none"><li>• Two 3-Wire/4-Wire SPI modules, Supporting 4 Frame modes with 8-Level FIFO Buffer</li><li>• Two I<sup>2</sup>C™ modules Support Multi-Master/Slave mode and 7-Bit/10-Bit Addressing</li><li>• Two UART modules:<ul style="list-style-type: none"><li>- Supports RS-232, RS-485 and LIN/J2602</li><li>- On-chip hardware encoder/decoder for IrDA®</li><li>- Auto-wake-up on Start bit</li><li>- Auto-Baud Detect</li><li>- 4-level FIFO buffer</li></ul></li><li>• Parallel Master Slave Port (PMP/PSP):<ul style="list-style-type: none"><li>- Supports 8-bit or 16-bit data</li><li>- Supports 16 address lines</li></ul></li></ul>	<b>Analog Features:</b> <ul style="list-style-type: none"><li>• 10-Bit, Up to 16-Channel Analog-to-Digital Converter<ul style="list-style-type: none"><li>- 500 ksp/s conversion rate</li><li>- Conversion available during Sleep and Idle</li></ul></li><li>• Dual Analog Comparators with Programmable Input/Output Configuration</li></ul> <p>Datasheet disponible sur le Web A télécharger</p>

---

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 97




## 2- Outils de travail et docs (1)

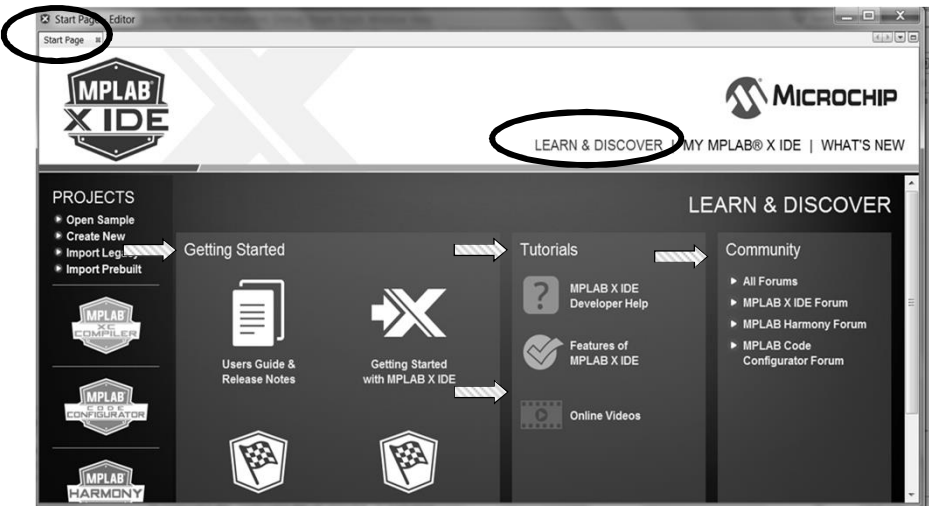
- Datasheet détaillé par section :  
<https://www.microchip.com/wwwproducts/en/PIC24FJ128GA010>
- Environnement de développement intégré IDE: MplabX.
- Compilateurs: C30 ou X16 (version Lite optimisation niveau 1).
- Simulateurs : Intégré à MPLABX, Proteus Spice , Matlab
- Carte de prototypage par exemple « **The Explorer 16 Development Board** » conçue et vendue par microchip:  
<http://ww1.microchip.com/downloads/en/DeviceDoc/50001589b.pdf>

---

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 98



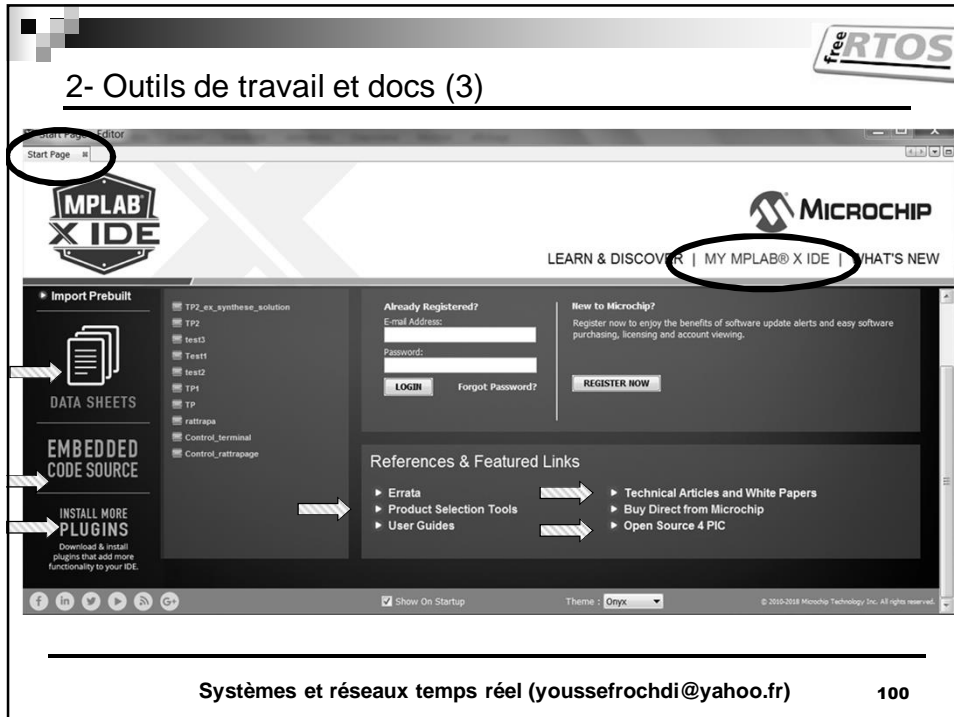
## 2- Outils de travail et docs (2)



Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 99

freeRTOS

## 2- Outils de travail et docs (3)



Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 100

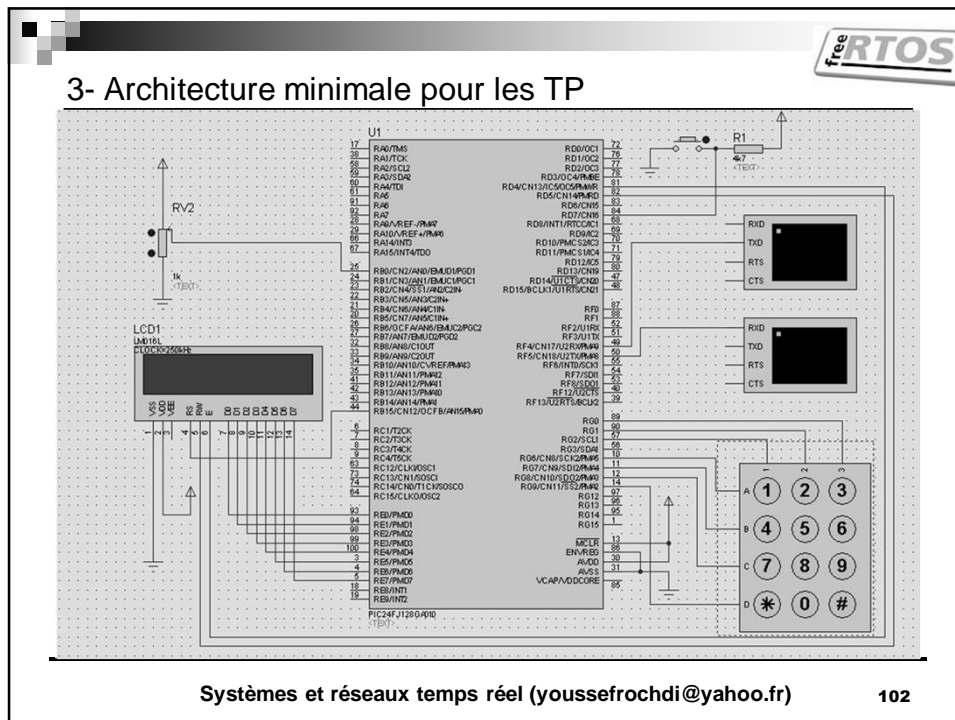
freeRTOS


## 2- Outils de travail et docs (4)

Documentation:

- Pour la librairie des périphériques: voir  
C:\Program Files (x86)\Microchip\xc16\v1.35\docs\periph\_libsPIC24F\Peripheral Library.chm.
- Pour les caractéristiques du compilateur, des types de données, modèle mémoire  
C:\Program Files (x86)\Microchip\xc16\v1.35\docs\ MPLAB\_XC16\_C\_Compiler\_Users\_Guide
- Pour les définitions des noms des registres, bits de config ..etc voir  
C:\Program Files (x86)\Microchip\xc16\v1.35\support\PIC24F\h\p24FJ128GA010.h
- Pour les macros de configuration des périphériques voir les fichiers .h correspondant par exemple adc.h pour le périphérique ADC, dans:  
C:\Program Files (x86)\Microchip\xc16\v1.35\support\peripheral\_24F
- Self-Paced Training : microchipdeveloper.com/

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 101





#### 4- Travaux pratiques

---

- Pour plus de détails consulter le manuel de TP.
- D'autres détails sont données en commentaire dans le code source.

---

Systèmes et réseaux temps réel (youssefrochdi@yahoo.fr) 104