トロンフォーラム主催

【実習】μT-Kernel入門

プログラミング演習テキスト

目次

演習１ タスク起動と優先度・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 2

演習２ 起床待ちと起床・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 8

演習３ セマフォ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・14

演習４ イベントフラグ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ 22

演習５ メールボックス・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・29

演習６ 割込みハンドラ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・35

演習７ 周期ハンドラ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・43

演習８ タイムシェアリング・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・49

演習９ 総合演習１スロットゲーム・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・55

演習１０ 総合演習２音楽プレーヤ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・57

回答例・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・60

演習1 タスク起動と優先度

■ プロジェクトフォルダ

C:\ut\_realos\utkernel\7-M\uT-Kernel\_Samples\sta\_pri

■ プロゴラム概要

・ タスクは，led1(優先度1)，led2(優先度2)，led3(優先度3) から構成される

・ 初期化タスク uinit\_task はタスクを動的に生成し，led3 を開始する

・ 各タスクがより優先順位の高いタスクの起動と字タスクの終了を繰り返す

led1タスク led2タスク led3タスク

優先度：１ 優先度：２ 優先度：３

スタート

スタート

自タスク終了

led1()

タスク起動

自タスク終了

スタート

Led2()

タスク起動

図1.1 sta\_priの各タスクのフローチャート

|  |  |  |
| --- | --- | --- |
| 1 | /\* |  |
| 2 | \*---------------------------------------------------------------------- | |
| 3 | \* micro T-Kernel | |
| 4 | \* |  |
| 5 | \* Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved. | |
| 6 | \* micro T-Kernel is distributed under the micro T-License. | |
| 7 | \*---------------------------------------------------------------------- | |
| 8 | \* micro T-REALOS/M3 | |
| 9 | \* |  |
| 10 | \* Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved. | |
| 11 | \* This product uses the Source Code of micro T-Kernel under the micro | |
| 12 | \* T-License granted by the T-Engine forum (http://www.t-engine.org). | |
| 13 | \* Last update : 2010/05/11 | |
| 14 | \*---------------------------------------------------------------------- | |
| 15 | \* |  |
| 16 | \* Version: 1.01.00 | |
| 17 | \* Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25. | |
| 18 | \* |  |
| 19 | \*---------------------------------------------------------------------- | |
| 20 | \*/ |  |
| 21 |  |  |
| 22 | #include <tk/tkernel.h> | |
| 23 | #include "init\_task.h" | |
| 24 | #include "mcu.h" | |
| 25 |  |  |
| 26 | #define USE\_WDG 0 |  |
| 27 |  |  |
| 28 | #if USE\_WDG == 0 | |
| 29 | #define WDG\_CTL (\*(volatile UW \*)0x40011008) | |
| 30 | #define WDG\_LCK (\*(volatile UW \*)0x40011C00) | |
| 31 | #endif |  |
| 32 |  |  |
| 33 | ID LED1\_ID; | |
| 34 | ID LED2\_ID; | |
| 35 | ID LED3\_ID; | |
| 36 |  |  |
| 37 | extern void \_\_reset( void ); | |
| 38 |  |  |
| 39 | EXPORT int main( void ) | |
| 40 | { |  |
| 41 | #if USE\_WDG == 0 | |
| 42 | WDG\_LCK = 0x1acce551; | |
| 43 | WDG\_LCK = 0xe5331aae; | |
| 44 | WDG\_CTL = 0x00000000; | |
| 45 | #endif |  |
| 46 |  |  |
| 47 | \_\_reset(); | |
| 48 | while(1); | |
| 49 | } |  |
| 50 |  |  |
| 51 | void delay (void) | |
| 52 | { |  |
| 53 | Int i; |  |
| 54 | For(i=0;i<200000;i++){} | |
| 55 | } |  |
| 56 |  |  |
| 57 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 58 | /\* LED1 task \*/ | |
| 59 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 60 |  |  |
| 61 | Static void led1( INT stacd, VP exinf ) | |
| 62 | { |  |
| 63 | printf("===Start led1()\n"); | |
| 64 | FM3\_GPIO->PDOR3 ^= 0x0004; | |
| 65 |  |  |
| 66 | printf("led1: goes into DORMANT state\n"); | |
| 67 |  |  |
| 68 | printf("===End led1()\n"); | |
| 69 | FM3\_GPIO->PDOR3 ^= 0x0004; | |
| 70 |  |  |
| 71 | /\* Exit Task \*/  ① | |
| 72 | } |  |
| 73 |  |  |
| 74 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 75 | /\* LED2 task \*/ | |
| 76 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 77 |  |  |
| 78 | Static void led2( INT stacd, VP exinf ) | |
| 79 | { |  |
| 80 | ER ercd; |  |
| 81 |  |  |
| 82 | printf("===Start led2()\n"); | |
| 83 |  |  |
| 84 | FM3\_GPIO->PDOR3 ^= 0x0008; | |
| 85 |  |  |
| 86 | printf("led2: starts led1\n"); | |
| 87 |  |  |
| 88 | Ercd = /\* Start Task LED1 \*/  ② | |
| 89 | if(ercd != E\_OK){while(1){}} | |
| 90 |  |  |
| 91 | printf("led2: goes into DORMANT state\n"); | |
| 92 |  |  |
| 93 | printf("===End led2()\n"); | |
| 94 |  |  |
| 95 | FM3\_GPIO->PDOR3 ^= 0x0008; | |
| 96 |  |  |
| 97 | /\* Exit Task \*/  ① | |
| 98 | } |  |
| 99 |  |  |
| 100 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 101 | /\* LED3 task \*/ | |
| 102 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 103 |  |  |
| 104 | Static void led3( INT stacd, VP exinf ) | |
| 105 | { |  |
| 106 | ER ercd; |  |
| 107 |  |  |
| 108 | printf("===Start led3()\n"); | |
| 109 |  |  |
| 110 | FM3\_GPIO->PDOR3 ^= 0x0010; | |
| 111 |  |  |
| 112 | While(1) |  |
| 113 | { |  |
| 114 | printf("led3: starts led2\n"); | |
| 115 |  |  |
| 116 | ercd = /\* Start Task LED2 \*/  ③ | |
| 117 | if(ercd != E\_OK){while(1){}} | |
| 118 | } |  |
| 119 | } |  |
| 120 |  |  |
| 121 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 122 | /\* Initialize task \*/ | |
| 123 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 124 |  |  |
| 125 | void uinit\_task(void) | |
| 126 | { |  |
| 127 | ER ercd; |  |
| 128 |  |  |
| 129 | T\_CTSK ctsk; | |
| 130 |  |  |
| 131 | static INT led1\_stack[0x400/4]; | |
| 132 | static INT led2\_stack[0x400/4]; | |
| 133 | static INT led3\_stack[0x400/4]; | |
| 134 |  |  |
| 135 | SysTick\_init(); | |
| 136 | sys\_init(); | |
| 137 |  |  |
| 138 | LCD\_sys\_init(); | |
| 139 | LCD\_init(); | |
| 140 | LCD\_reset(); | |
| 141 |  |  |
| 142 | LCD\_gotoxy(0,0); | |
| 143 | LCD\_sendstring("uT-Kernel"); | |
| 144 | LCD\_gotoxy(0,1); | |
| 145 | LCD\_sendstring("start&priority"); | |
| 146 |  |  |
| 147 | ctsk.exinf = (VP)1; | |
| 148 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 149 | ctsk.task = led1; | |
| 150 | ctsk.itskpri = 1; | |
| 151 | ctsk.stksz = (W)sizeof(led1\_stack); | |
| 152 | ctsk.bufptr = led1\_stack; | |
| 153 | LED1\_ID = tk\_cre\_tsk(&ctsk); | |
| 154 |  |  |
| 155 | ctsk.exinf = (VP)2; | |
| 156 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 157 | ctsk.task = led2; | |
| 158 | ctsk.itskpri = 2; | |
| 159 | ctsk.stksz = (W)sizeof(led2\_stack); | |
| 160 | ctsk.bufptr = led2\_stack; | |
| 161 | LED2\_ID = tk\_cre\_tsk(&ctsk); | |
| 162 |  |  |
| 163 | ctsk.exinf = (VP)3; | |
| 164 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 165 | ctsk.task = led3; | |
| 166 | ctsk.itskpri = 3; | |
| 167 | ctsk.stksz = (W)sizeof(led3\_stack); | |
| 168 | ctsk.bufptr = led3\_stack; | |
| 169 | LED3\_ID = tk\_cre\_tsk(&ctsk); | |
| 170 |  |  |
| 171 | ercd = /\* Start Task LED3 \*/  ④ | |
| 172 | if(ercd != E\_OK){while(1){}} | |
| 173 | } |  |

演習２ 起床待ちと起床

■ プロジェクトフォルダ

C:\ut\_realos\utkernel\7-M\uT-Kernel\_Samples\slp\_wu

■ プロゴラム概要

・ タスクは，led1(優先度6)，led2(優先度5)，led3(優先度4) から構成される

・ 初期化タスク uinit\_task はタスクを動的に生成し，

他のタスクに起床されるまで待つ．

他タスクを起床したら，志度タスク起床待ち状態に遷移する．

led1タスク led2タスク led3タスク

優先度：６ 優先度：5 優先度：４

スタート

スタート

led2()

タスク起床

スタート

自タスク

起床待ち

自タスク

起床待ち

led3()

タスク起床

図2 slp\_wupの各タスクのフローチャート

|  |  |  |
| --- | --- | --- |
| 1 | /\* |  |
| 2 | \*---------------------------------------------------------------------- | |
| 3 | \* micro T-Kernel | |
| 4 | \* |  |
| 5 | \* Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved. | |
| 6 | \* micro T-Kernel is distributed under the micro T-License. | |
| 7 | \*---------------------------------------------------------------------- | |
| 8 | \* micro T-REALOS/M3 | |
| 9 | \* |  |
| 10 | \* Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved. | |
| 11 | \* This product uses the Source Code of micro T-Kernel under the micro | |
| 12 | \* T-License granted by the T-Engine forum (http://www.t-engine.org). | |
| 13 | \* Last update : 2010/05/11 | |
| 14 | \*---------------------------------------------------------------------- | |
| 15 | \* |  |
| 16 | \* Version: 1.01.00 | |
| 17 | \* Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25. | |
| 18 | \* |  |
| 19 | \*---------------------------------------------------------------------- | |
| 20 | \*/ |  |
| 21 |  |  |
| 22 | #include <tk/tkernel.h> | |
| 23 | #include "init\_task.h" | |
| 24 | #include "mcu.h" | |
| 25 |  |  |
| 26 | #define USE\_WDG 0 |  |
| 27 |  |  |
| 28 | #if USE\_WDG == 0 | |
| 29 | #define WDG\_CTL (\*(volatile UW \*)0x40011008) | |
| 30 | #define WDG\_LCK (\*(volatile UW \*)0x40011C00) | |
| 31 | #endif |  |
| 32 |  |  |
| 33 | ID LED1\_ID; | |
| 34 | ID LED2\_ID; | |
| 35 | ID LED3\_ID; | |
| 36 |  |  |
| 37 | extern void \_\_reset( void ); | |
| 38 |  |  |
| 39 | EXPORT int main( void ) | |
| 40 | { |  |
| 41 | #if USE\_WDG == 0 | |
| 42 | WDG\_LCK = 0x1acce551; | |
| 43 | WDG\_LCK = 0xe5331aae; | |
| 44 | WDG\_CTL = 0x00000000; | |
| 45 | #endif |  |
| 46 |  |  |
| 47 | \_\_reset(); | |
| 48 | while(1); | |
| 49 | } |  |
| 50 |  |  |
| 51 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 52 | /\* LED1 task \*/ | |
| 53 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 54 |  |  |
| 55 | static void led1( INT stacd, VP exinf ) | |
| 56 | { |  |
| 57 | ER ercd; |  |
| 58 |  |  |
| 59 | while(1) |  |
| 60 | { |  |
| 61 | printf("led1: RUNNING\n"); | |
| 62 | FM3\_GPIO->PDOR3 ^= 0x0004; | |
| 63 |  |  |
| 64 | printf("led1: WAKEUP led2\n"); | |
| 65 | FM3\_GPIO->PDOR3 ^= 0x0004; | |
| 66 |  |  |
| 67 | ercd = /\* Wake Up Task LED2 \*/  ① | |
| 68 | if(ercd != E\_OK){while(1){}} | |
| 69 | } |  |
| 70 | } |  |
| 71 |  |  |
| 72 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 73 | /\* LED2 task \*/ | |
| 74 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 75 |  |  |
| 76 | static void led2( INT stacd, VP exinf ) | |
| 77 | { |  |
| 78 | ER ercd; |  |
| 79 |  |  |
| 80 | while(1) |  |
| 81 | { |  |
| 82 | printf("led2: RUNNING\n"); | |
| 83 | FM3\_GPIO->PDOR3 ^= 0x0008; | |
| 84 |  |  |
| 85 | printf("led2: SLEEP\n"); | |
| 86 | FM3\_GPIO->PDOR3 ^= 0x0008; | |
| 87 |  |  |
| 88 | ercd = /\* Sleep Task (No Timeout) \*/  ② | |
| 89 | if(ercd != E\_OK){while(1){}} | |
| 90 |  |  |
| 91 | printf("led2: RUNNING\n"); | |
| 92 | FM3\_GPIO->PDOR3 ^= 0x0008; | |
| 93 |  |  |
| 94 | printf("led2: WAKEUP led3\n"); | |
| 95 | FM3\_GPIO->PDOR3 ^= 0x0008; | |
| 96 |  |  |
| 97 | ercd = /\* Wake Up LED3 \*/  ③ | |
| 98 | if(ercd != E\_OK){while(1){}} | |
| 99 | } |  |
| 100 | } |  |
| 101 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 102 | /\* LED3 task \*/ | |
| 103 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 104 |  |  |
| 105 | static void led3( INT stacd, VP exinf ) | |
| 106 | { |  |
| 107 | ER ercd; |  |
| 108 |  |  |
| 109 | while(1) |  |
| 110 | { |  |
| 111 | printf("led3: RUNNING\n"); | |
| 112 | FM3\_GPIO->PDOR3 ^= 0x0010; | |
| 113 |  |  |
| 114 | printf("led3: SLEEP\n"); | |
| 115 | FM3\_GPIO->PDOR3 ^= 0x0010; | |
| 116 |  |  |
| 117 | ercd = /\* Sleep Task (No Timeout) \*/  ② | |
| 118 | if(ercd != E\_OK){while(1){}} | |
| 119 | } |  |
| 120 | } |  |
| 121 |  |  |
| 122 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 123 | /\* Initialize task \*/ | |
| 124 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 125 |  |  |
| 126 | void uinit\_task(void) | |
| 127 | { |  |
| 128 | ER ercd; | |
| 129 |  |  |
| 130 | T\_CTSK ctsk; | |
| 131 |  |  |
| 132 | static INT led1\_stack[0x400/4]; | |
| 133 | static INT led2\_stack[0x400/4]; | |
| 134 | static INT led3\_stack[0x400/4]; | |
| 135 |  |  |
| 136 | SysTick\_init(); | |
| 137 | sys\_init(); | |
| 138 |  |  |
| 139 | LCD\_sys\_init(); | |
| 140 | LCD\_init(); | |
| 141 | LCD\_reset(); | |
| 142 |  |  |
| 143 | LCD\_gotoxy(0,0); | |
| 144 | LCD\_sendstring("uT-Kernel"); | |
| 145 | LCD\_gotoxy(0,1); | |
| 146 | LCD\_sendstring("sleep&wake-up"); | |
| 147 |  |  |
| 148 | ctsk.exinf = (VP)1; | |
| 149 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 150 | ctsk.task = led1; | |
| 151 | ctsk.itskpri = 6; | |
| 152 | ctsk.stksz = (W)sizeof(led1\_stack); | |
| 153 | ctsk.bufptr = led1\_stack; | |
| 154 | LED1\_ID = tk\_cre\_tsk(&ctsk); | |
| 155 |  |  |
| 156 | ctsk.exinf = (VP)2; | |
| 157 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 158 | ctsk.task = led2; | |
| 159 | ctsk.itskpri = 5; | |
| 160 | ctsk.stksz = (W)sizeof(led2\_stack); | |
| 161 | ctsk.bufptr = led2\_stack; | |
| 162 | LED2\_ID = tk\_cre\_tsk(&ctsk); | |
| 163 |  |  |
| 164 | ctsk.exinf = (VP)3; | |
| 165 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 166 | ctsk.task = led3; | |
| 167 | ctsk.itskpri = 4; | |
| 168 | ctsk.stksz = (W)sizeof(led3\_stack); | |
| 169 | ctsk.bufptr = led3\_stack; | |
| 170 | LED3\_ID = tk\_cre\_tsk(&ctsk); | |
| 171 |  |  |
| 172 | ercd = tk\_sta\_tsk(LED3\_ID, NULL); | |
| 173 | if(ercd != E\_OK){while(1){}} | |
| 174 |  |  |
| 175 | ercd = tk\_sta\_tsk(LED2\_ID, NULL); | |
| 176 | if(ercd != E\_OK){while(1){}} | |
| 177 |  |  |
| 178 | ercd = tk\_sta\_tsk(LED1\_ID, NULL); | |
| 179 | if(ercd != E\_OK){while(1){}} | |
| 180 | } |  |

演習３ セマフォ

■ プロジェクトフォルダ

C:\ut\_realos\utkernel\7-M\uT-Kernel\_Samples\sem

■ プロゴラム概要

・ タスクは，led1(優先度1)，led2(優先度2)，led3(優先度3)，psw1 から構成される

・ 初期化タスク uinit\_task はタスクを動的に生成し，各タスクを順番に起動する

・ led1，led2，led3がそれぞれ1つ，2つ，3つのセマフォを要求し，待ち状態に遷移する．

その後，psw1が押されるたびにセマフォが返却され，必要なセマフォ数が確保できるように

なったら，led1，led2，led3の待ち状態が解除される．

led1タスク led2タスク

優先度：１ 優先度：２

スタート

セマフォx1返却

セマフォx1獲得

スタート

セマフォx2返却

セマフォx2獲得

図3 演習3のled1, led2タスクのフローチャート

led3タスク psw1タスク

優先度：３ 優先度：４

スタート

セマフォx3獲得

スタート

PSW1

押下待ち

セマフォx1返却

図4 演習3のled3, psw1タスクのフローチャート

|  |  |  |
| --- | --- | --- |
| 1 | /\* |  |
| 2 | \*---------------------------------------------------------------------- | |
| 3 | \* micro T-Kernel | |
| 4 | \* |  |
| 5 | \* Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved. | |
| 6 | \* micro T-Kernel is distributed under the micro T-License. | |
| 7 | \*---------------------------------------------------------------------- | |
| 8 | \* micro T-REALOS/M3 | |
| 9 | \* |  |
| 10 | \* Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved. | |
| 11 | \* This product uses the Source Code of micro T-Kernel under the micro | |
| 12 | \* T-License granted by the T-Engine forum (http://www.t-engine.org). | |
| 13 | \* Last update : 2010/05/11 | |
| 14 | \*---------------------------------------------------------------------- | |
| 15 | \* |  |
| 16 | \* Version: 1.01.00 | |
| 17 | \* Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25. | |
| 18 | \* |  |
| 19 | \*---------------------------------------------------------------------- | |
| 20 | \*/ |  |
| 21 |  |  |
| 22 | /\* |  |
| 23 | \* | Init\_task.c (usermain) |
| 24 | \* | User Main |
| 25 | \*/ |  |
| 26 |  |  |
| 27 | #include <tk/tkernel.h> | |
| 28 |  |  |
| 29 | #include "init\_task.h" | |
| 30 |  |  |
| 31 | #include "mcu.h" | |
| 32 |  |  |
| 33 | #define USE\_WDG 0 |  |
| 34 |  |  |
| 35 | #if USE\_WDG == 0 | |
| 36 | #define WDG\_CTL (\*(volatile UW \*)0x40011008) | |
| 37 | #define WDG\_LCK (\*(volatile UW \*)0x40011C00) | |
| 38 | #endif |  |
| 39 |  |  |
| 40 | ID SEM1\_ID; | |
| 41 | ID LED1\_ID; | |
| 42 | ID LED2\_ID; | |
| 43 | ID LED3\_ID; | |
| 44 | ID PSW1\_ID; | |
| 45 |  |  |
| 46 | extern void \_\_reset( void ); | |
| 47 |  |  |
| 48 | EXPORT int main( void ) | |
| 49 | { |  |
| 50 | #if USE\_WDG == 0 | |
| 51 | WDG\_LCK = 0x1acce551; | |
| 52 | WDG\_LCK = 0xe5331aae; | |
| 53 | WDG\_CTL = 0x00000000; | |
| 54 | #endif |  |
| 55 |  |  |
| 56 | \_\_reset(); | |
| 57 | while(1); | |
| 58 | } |  |
| 59 |  |  |
| 60 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 61 | /\* LED1 task \*/ | |
| 62 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 63 |  |  |
| 64 | static void led1( INT stacd, VP exinf ) | |
| 65 | { |  |
| 66 | ER ercd; |  |
| 67 |  |  |
| 68 | while(1){ |  |
| 69 | ercd = /\* Wait Semaphore x 1 \*/  ① | |
| 70 | if(ercd != E\_OK){while(1){}} | |
| 71 |  |  |
| 72 | FM3\_GPIO->PDOR3 = 0x001c; | |
| 73 | FM3\_GPIO->PDOR3 ^= 0x0004; | |
| 74 |  |  |
| 75 | ercd = /\* Signal Semaphore x 1 \*/  ② | |
| 76 | if(ercd != E\_OK){while(1){}} | |
| 77 | } |  |
| 78 | } |  |
| 79 |  |  |
| 80 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 81 | /\* LED2 task \*/ | |
| 82 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 83 |  |  |
| 84 | static void led2( INT stacd, VP exinf ) | |
| 85 | { |  |
| 86 | ER ercd; |  |
| 87 |  |  |
| 88 | while(1){ |  |
| 89 |  |  |
| 90 | ercd = /\* Wait Semaphore x 2 \*/  ③ | |
| 91 | if(ercd != E\_OK){while(1){}} | |
| 92 |  |  |
| 93 | FM3\_GPIO->PDOR3 ^= 0x000c; | |
| 94 |  |  |
| 95 | ercd = /\* Signal Semaphore x 2 \*/  ④ | |
| 96 | if(ercd != E\_OK){while(1){}} | |
| 97 | } |  |
| 98 | } |  |
| 99 |  |  |
| 100 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 101 | /\* LED3 task \*/ | |
| 102 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 103 |  |  |
| 104 | static void led3( INT stacd, VP exinf ) | |
| 105 | { |  |
| 106 | ER ercd; |  |
| 107 |  |  |
| 108 | while(1){ |  |
| 109 |  |  |
| 110 | ercd = /\* Wait Semaphore x 3 \*/  ⑤ | |
| 111 | if(ercd != E\_OK){while(1){}} | |
| 112 |  |  |
| 113 | FM3\_GPIO->PDOR3 ^= 0x0018; | |
| 114 | } |  |
| 115 | } |  |
| 116 |  |  |
| 117 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 118 | /\* Switch1 task \*/ | |
| 119 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 120 |  |  |
| 121 | static void psw1( INT stacd, VP exinf ) | |
| 122 | { |  |
| 123 | ER ercd; |  |
| 124 |  |  |
| 125 | int sw1=0; | |
| 126 | int count; | |
| 127 | while(1){ |  |
| 128 | if(sw1) |  |
| 129 | { |  |
| 130 | if(FM3\_GPIO->PDIR5\_f.P0) | |
| 131 | { |  |
| 132 | count++; | |
| 133 | if(count>3) | |
| 134 | { |  |
| 135 | Count=0; | |
| 136 | sw1=0; | |
| 137 | } |  |
| 138 | } |  |
| 139 | Else |  |
| 140 | { |  |
| 141 | count=0; | |
| 142 | } |  |
| 143 | } |  |
| 144 | Else |  |
| 145 | { |  |
| 146 | if(!FM3\_GPIO->PDIR5\_f.P0) | |
| 147 | { |  |
| 148 | count++; | |
| 149 | if(count>3) | |
| 150 | { | |
| 151 | ercd = /\* Signal Semaphore x 1 \*/  ② | |
| 152 | if(ercd != E\_OK){while(1){}} | |
| 153 | count=0; | |
| 154 | sw1=1; | |
| 155 | } |  |
| 156 | } |  |
| 157 | Else |  |
| 158 | { |  |
| 159 | count=0; | |
| 160 | } |  |
| 161 |  |  |
| 162 | } |  |
| 163 | } |  |
| 164 | } |  |
| 165 |  |  |
| 166 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 167 | /\* Initialize task \*/ | |
| 168 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 169 |  |  |
| 170 | void uinit\_task(void) | |
| 171 | { |  |
| 172 | T\_CTSK ctsk; | |
| 173 | T\_CSEM csem; | |
| 174 | static INT led1\_stack[0x400/4]; | |
| 175 | static INT led2\_stack[0x400/4]; | |
| 176 | static INT led3\_stack[0x400/4]; | |
| 177 | static INT psw1\_stack[0x400/4]; | |
| 178 |  |  |
| 179 | /\*system clock set\*/ | |
| 180 | SysTick\_init(); | |
| 181 | sys\_init(); | |
| 182 |  |  |
| 183 | LCD\_sys\_init(); | |
| 184 | LCD\_init(); | |
| 185 | LCD\_reset(); | |
| 186 |  |  |
| 187 | LCD\_gotoxy(0,0); | |
| 188 | LCD\_sendstring("ut-Kernel"); | |
| 189 | LCD\_gotoxy(0,1); | |
| 190 | LCD\_sendstring("semaphore"); | |
| 191 |  |  |
| 192 | csem.sematr = TA\_TFIFO | TA\_FIRST; | |
| 193 | csem.isemcnt = 0; | |
| 194 | csem.maxsem = 3; | |
| 195 | SEM1\_ID = /\* Create Semaphore \*/  ⑥ | |
| 196 |  |  |
| 197 | ctsk.exinf = (VP)1; | |
| 198 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 199 | ctsk.task = led1; | |
| 200 | ctsk.itskpri = 1; | |
| 201 | ctsk.stksz = (W)sizeof(led1\_stack); | |
| 202 | ctsk.bufptr = led1\_stack; | |
| 203 | LED1\_ID = tk\_cre\_tsk(&ctsk); | |
| 204 |  |  |
| 205 | ctsk.exinf = (VP)2; | |
| 206 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 207 | ctsk.task = led2; | |
| 208 | ctsk.itskpri = 2; | |
| 209 | ctsk.stksz = (W)sizeof(led2\_stack); | |
| 210 | ctsk.bufptr = led2\_stack; | |
| 211 | LED2\_ID = tk\_cre\_tsk(&ctsk); | |
| 212 |  |  |
| 213 | ctsk.exinf = (VP)3; | |
| 214 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 215 | ctsk.task = led3; | |
| 216 | ctsk.itskpri = 3; | |
| 217 | ctsk.stksz = (W)sizeof(led3\_stack); | |
| 218 | ctsk.bufptr = led3\_stack; | |
| 219 | LED3\_ID = tk\_cre\_tsk(&ctsk); | |
| 220 |  |  |
| 221 | ctsk.exinf = (VP)4; | |
| 222 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 223 | ctsk.task = psw1; | |
| 224 | ctsk.itskpri = 4; | |
| 225 | ctsk.stksz = (W)sizeof(psw1\_stack); | |
| 226 | ctsk.bufptr = psw1\_stack; | |
| 227 | PSW1\_ID = tk\_cre\_tsk(&ctsk); | |
| 228 |  |  |
| 229 | Tk\_sta\_tsk(LED1\_ID, NULL); | |
| 230 | Tk\_sta\_tsk(LED2\_ID, NULL); | |
| 231 | tk\_sta\_tsk(LED3\_ID, NULL); | |
| 232 | tk\_sta\_tsk(PSW1\_ID, NULL); | |
| 233 | } |  |

演習４ イベントフラグ

■ プロジェクトフォルダ

C:\ut\_realos\utkernel\7-M\uT-Kernel\_Samples\ev\_flag

■ プロゴラム概要

・ タスクは，led1(優先度1)，psw1(優先度2) から構成される

・ 初期化タスク uinit\_task はタスクを動的に生成し，各タスクを順番に起動する

・ psw1タスクはpsw1スイッチの状態をサンプリングし，psw1スイッチが押されたら

フラグパターンを0x00から増加させる（0x00，0x01，0x02，0x03，・・・）

・ led1は要求するフラグパターンを設定し，パターンが実行されるまで待状態に遷移する．

psw1によって発行されるパターンが要求パターンと一致すれば，待ち状態が解除される．

led1タスク psw1タスク

優先度：１ 優先度：2

スタート

スタート

イベントフラグ

待ち(0x3,OR）

クリアイベント

フラグ

クリアイベント

フラグ

イベントフラグ

待ち(0x3,OR）

クリアイベント

フラグ

イベントフラグ

待ち(0x3,AND）

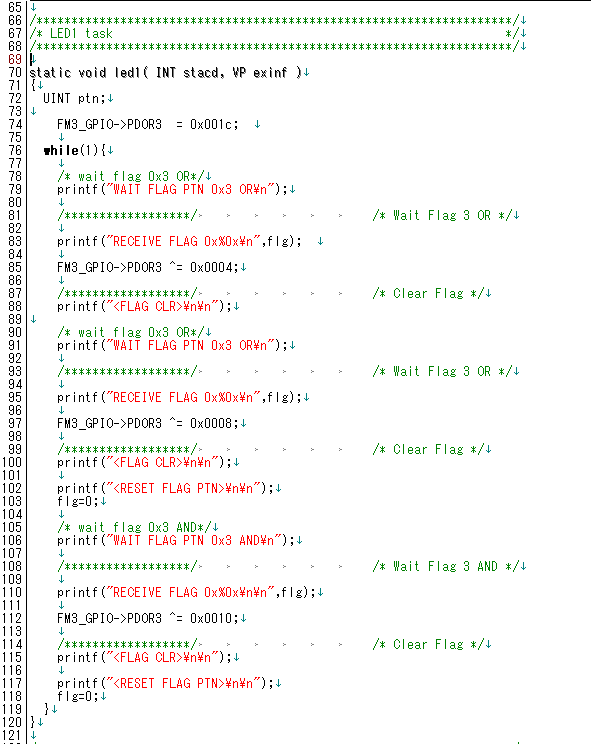
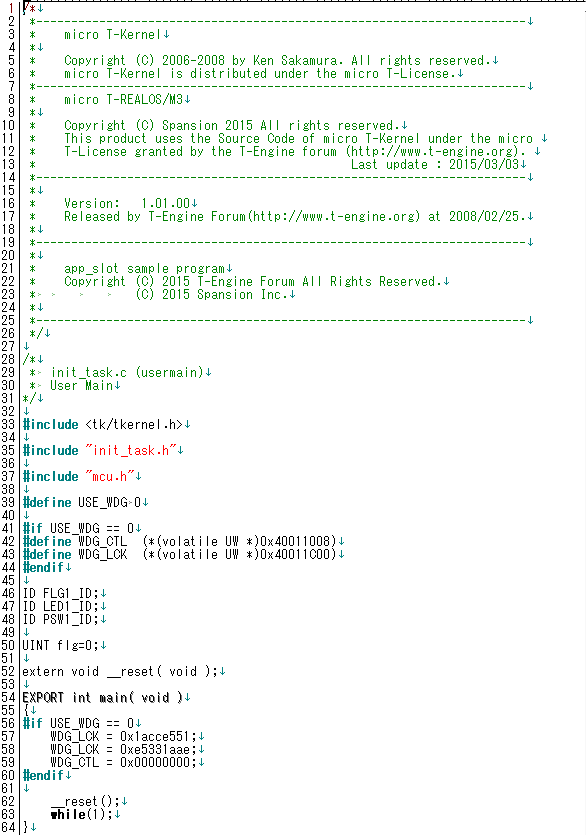
イベントフラグ

パターン++

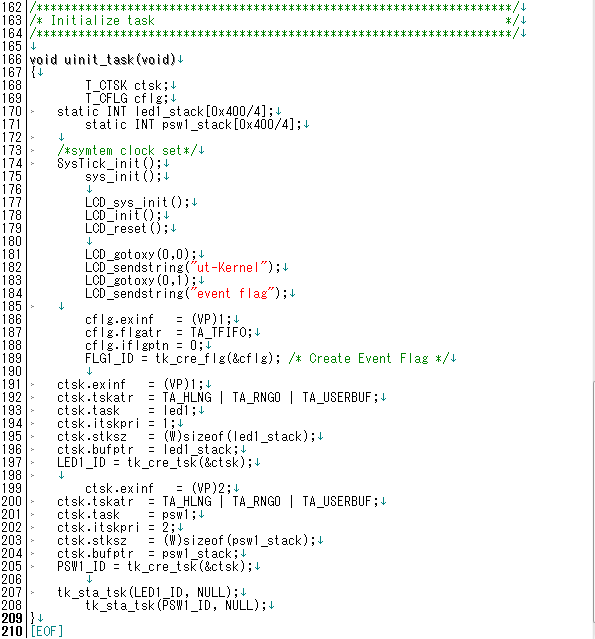
PSW1

押下待ち

図5 演習４の各タスクのフローチャート







演習５ メールボックス

■ プロジェクトフォルダ

C:\ut\_realos\utkernel\7-M\uT-Kernel\_Samples\mailbox

■ プロゴラム概要

・ タスクは，rcv1(優先度1)，snd1(優先度2) から構成される

・ 初期化タスク uinit\_task はタスクを動的に生成し，各タスクを順番に起動する

・ rcv1が実行状態になったら受信できるメッセージが発生するまで待ちます．

そこで，snd1からメッセージが送信されたら，rcv1の待ち状態が解除され，

rcv1が受信したメッセージがLCDとターミナルI/Oに表示される．

rcv1タスク snd1タスク

優先度：１ 優先度：2

スタート

メッセージ表示

メッセージ受信

スタート

メッセージ送信

図6 演習５の各タスクのフローチャート

|  |  |  |
| --- | --- | --- |
| 1 | /\* |  |
| 2 | \*---------------------------------------------------------------------- | |
| 3 | \* micro T-Kernel | |
| 4 | \* |  |
| 5 | \* Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved. | |
| 6 | \* micro T-Kernel is distributed under the micro T-License. | |
| 7 | \*---------------------------------------------------------------------- | |
| 8 | \* micro T-REALOS/M3 | |
| 9 | \* |  |
| 10 | \* Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved. | |
| 11 | \* This product uses the Source Code of micro T-Kernel under the micro | |
| 12 | \* T-License granted by the T-Engine forum (http://www.t-engine.org). | |
| 13 | \* Last update : 2010/05/11 | |
| 14 | \*---------------------------------------------------------------------- | |
| 15 | \* |  |
| 16 | \* Version: 1.01.00 | |
| 17 | \* Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25. | |
| 18 | \* |  |
| 19 | \*---------------------------------------------------------------------- | |
| 20 | \*/ |  |
| 21 |  |  |
| 22 | /\* |  |
| 23 | \* | init\_task.c (usermain) |
| 24 | \* | User Main |
| 25 | \*/ |  |
| 26 |  |  |
| 27 | #include <tk/tkernel.h> | |
| 28 |  |  |
| 29 | #include "init\_task.h" | |
| 30 |  |  |
| 31 | #include "mcu.h" | |
| 32 |  |  |
| 33 | #define USE\_WDG 0 |  |
| 34 |  |  |
| 35 | #if USE\_WDG == 0 | |
| 36 | #define WDG\_CTL (\*(volatile UW \*)0x40011008) | |
| 37 | #define WDG\_LCK (\*(volatile UW \*)0x40011C00) | |
| 38 | #endif |  |
| 39 |  |  |
| 40 | ID MBX1\_ID; | |
| 41 | ID RCV1\_ID; | |
| 42 | ID SND1\_ID; | |
| 43 |  |  |
| 44 | UINT flg=0; | |
| 45 |  |  |
| 46 | extern void \_\_reset( void ); | |
| 47 |  |  |
| 48 | EXPORT int main( void ) | |
| 49 | { |  |
| 50 | #if USE\_WDG == 0 | |
| 51 | WDG\_LCK = 0x1acce551; | |
| 52 | WDG\_LCK = 0xe5331aae; | |
| 53 | WDG\_CTL = 0x00000000; | |
| 54 | #endif |  |
| 55 |  |  |
| 56 | \_\_reset(); | |
| 57 | while(1); | |
| 58 | } |  |
| 59 |  |  |
| 60 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 61 | /\* RECEIVE MESSAGE TASK \*/ | |
| 62 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 63 |  |  |
| 64 | static void rcv1( INT stacd, VP exinf ) | |
| 65 | { |  |
| 66 | ER ercd; |  |
| 67 |  |  |
| 68 | typedef struct | |
| 69 | { |  |
| 70 | T\_MSG header; | |
| 71 | UB msg\_cont[64]; | |
| 72 | } mbx\_tag; | |
| 73 | mbx\_tag\* rx\_mail; | |
| 74 |  |  |
| 75 |  |  |
| 76 | while(1){ |  |
| 77 | ercd = /\* Recieve Mailbox \*/  ① | |
| 78 | if(ercd != E\_OK){while(1){}} | |
| 79 |  |  |
| 80 | printf("#Receive Message = %s\n", rx\_mail->msg\_cont); | |
| 81 |  |  |
| 82 | LCD\_gotoxy(0,0); | |
| 83 | LCD\_sendstring("Message: "); | |
| 84 | LCD\_gotoxy(0,1); | |
| 85 | LCD\_sendstring(rx\_mail->msg\_cont); | |
| 86 | } |  |
| 87 | } |  |
| 88 |  |  |
| 89 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 90 | /\* SEND MESSAGE TASK \*/ | |
| 91 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 92 |  |  |
| 93 | static void snd1( INT stacd, VP exinf ) | |
| 94 | { |  |
| 95 | ER ercd; |  |
| 96 | UB str[] = "T-Engine Forum"; | |
| 97 |  |  |
| 98 | Typedef struct { | |
| 99 | T\_MSG header; | |
| 100 | UB msg\_cont[64]; | |
| 101 | } mbx\_tag; | |
| 102 | mbx\_tag tx\_mail; | |
| 103 |  |  |
| 104 | strcpy(tx\_mail.msg\_cont, str); | |
| 105 |  |  |
| 106 | while(1){ |  |
| 107 | printf("#Send Message\n"); | |
| 108 | Ercd = /\* Send Mailbox \*/  ② | |
| 109 | if(ercd != E\_OK){while(1){}} | |
| 110 | } |  |
| 111 | } |  |
| 112 |  |  |
| 113 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 114 | /\* Initialize task \*/ | |
| 115 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 116 |  |  |
| 117 | void uinit\_task(void) | |
| 118 | { |  |
| 119 | ER ercd; | |
| 120 |  |  |
| 121 | T\_CTSK ctsk; | |
| 122 | T\_CMBX cmbx; | |
| 123 | static INT rcv1\_stack[0x400/4]; | |
| 124 | static INT snd1\_stack[0x400/4]; | |
| 125 |  |  |
| 126 | /\*system clock set\*/ | |
| 127 | SysTick\_init(); | |
| 128 | sys\_init(); | |
| 129 |  |  |
| 130 | LCD\_sys\_init(); | |
| 131 | LCD\_init(); | |
| 132 | LCD\_reset(); | |
| 133 |  |  |
| 134 | LCD\_gotoxy(0,0); | |
| 135 | LCD\_sendstring("ut-Kernel"); | |
| 136 | LCD\_gotoxy(0,1); | |
| 137 | LCD\_sendstring("mailbox"); | |
| 138 |  |  |
| 139 | cmbx.exinf = (VP)1; | |
| 140 | cmbx.mbxatr = TA\_TFIFO; | |
| 141 | MBX1\_ID = /\* Create Mailbox \*/  ③ | |
| 142 |  |  |
| 143 | ctsk.exinf = (VP)1; | |
| 144 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 145 | ctsk.task = rcv1; | |
| 146 | ctsk.itskpri = 1; | |
| 147 | ctsk.stksz = (W)sizeof(rcv1\_stack); | |
| 148 | ctsk.bufptr = rcv1\_stack; | |
| 149 | RCV1\_ID = tk\_cre\_tsk(&ctsk); | |
| 150 |  |  |
| 151 | ctsk.exinf = (VP)2; | |
| 152 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 153 | ctsk.task = snd1; | |
| 154 | ctsk.itskpri = 2; | |
| 155 | ctsk.stksz = (W)sizeof(snd1\_stack); | |
| 156 | ctsk.bufptr = snd1\_stack; | |
| 157 | SND1\_ID = tk\_cre\_tsk(&ctsk); | |
| 158 |  |  |
| 159 | ercd = tk\_sta\_tsk(RCV1\_ID, NULL); | |
| 160 | if(ercd != E\_OK){while(1){}} | |
| 161 | ercd = tk\_sta\_tsk(SND1\_ID, NULL); |  |
| 162 | if(ercd != E\_OK){while(1){}} | |
| 163 | } | |  |

演習６ 割込みハンドラ

■ プロジェクトフォルダ

C:\ut\_realos\utkernel\7-M\uT-Kernel\_Samples\int

■ プロゴラム概要

・ タスクは，led1(優先度1)，led2(優先度2) と割込みハンドラ int1 から構成される

・ 初期化タスク uinit\_task は割込みハンドラ int1 の登録を行った後，

タスクを動的に生成し，各タスクを順番に起動する

・ led1とled2が実行状態からすぐに起床待ち状態に遷移する．

psw1またはpsw2が押されたら，割込みハンドラが起動し，led1またはled2を起床する．

その後，起床されたタスクが該当するLEDを点灯/消灯し，再度起床待ち状態に遷移する．

led1タスク led2タスク

優先度：１ 優先度：２

スタート

LED1 ON/OFF

スタート

LED2 ON/OFF

自タスク

起床待ち

自タスク

起床待ち

図7 演習６のled1, led2タスクのフローチャート

int1 ()

PSW1/2割込みハンドラ

END

スタート

led1タスク

タスク起床

led2タスク

タスク起床

スイッチ判定

PSW1

PSW2

END

図8 演習６の割り込みハンドラのフローチャート

|  |  |  |
| --- | --- | --- |
| 1 | /\* |  |
| 2 | \*---------------------------------------------------------------------- | |
| 3 | \* micro T-Kernel | |
| 4 | \* |  |
| 5 | \* Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved. | |
| 6 | \* micro T-Kernel is distributed under the micro T-License. | |
| 7 | \*---------------------------------------------------------------------- | |
| 8 | \* micro T-REALOS/M3 | |
| 9 | \* |  |
| 10 | \* Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved. | |
| 11 | \* This product uses the Source Code of micro T-Kernel under the micro | |
| 12 | \* T-License granted by the T-Engine forum (http://www.t-engine.org). | |
| 13 | \* Last update : 2010/05/11 | |
| 14 | \*---------------------------------------------------------------------- | |
| 15 | \* |  |
| 16 | \* Version: 1.01.00 | |
| 17 | \* Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25. | |
| 18 | \* |  |
| 19 | \*---------------------------------------------------------------------- | |
| 20 | \*/ |  |
| 21 |  |  |
| 22 | /\* |  |
| 23 | \* | init\_task.c (usermain) |
| 24 | \* | User Main |
| 25 | \*/ |  |
| 26 |  |  |
| 27 | #include <tk/tkernel.h> | |
| 28 |  |  |
| 29 | #include "init\_task.h" | |
| 30 |  |  |
| 31 | #include "mcu.h" | |
| 32 |  |  |
| 33 | #define USE\_WDG | 0 |
| 34 |  |  |
| 35 | #if USE\_WDG == 0 | |
| 36 | #define WDG\_CTL (\*(volatile UW \*)0x40011008) | |
| 37 | #define WDG\_LCK (\*(volatile UW \*)0x40011C00) | |
| 38 | #endif |  |
| 39 |  |  |
| 40 | ID LED1\_ID; | |
| 41 | ID LED2\_ID; | |
| 42 |  |  |
| 43 | extern void \_\_reset( void ); | |
| 44 |  |  |
| 45 | EXPORT int main( void ) | |
| 46 | { |  |
| 47 | #if USE\_WDG == 0 | |
| 48 | WDG\_LCK = 0x1acce551; | |
| 49 | WDG\_LCK = 0xe5331aae; | |
| 50 | WDG\_CTL = 0x00000000; | |
| 51 | #endif |  |
| 52 |  |  |
| 53 | \_\_reset(); | |
| 54 | While(1); | |
| 55 | } |  |
| 56 |  |  |
| 57 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 58 | /\* LED1 task \*/ | |
| 59 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 60 |  |  |
| 61 | static void led1( INT stacd, VP exinf ) | |
| 62 | { |  |
| 63 | ER ercd; |  |
| 64 |  |  |
| 65 | while(1) |  |
| 66 | { |  |
| 67 | Ercd = tk\_slp\_tsk(TMO\_FEVR); | |
| 68 | if(ercd != E\_OK){while(1){}} | |
| 69 |  |  |
| 70 | FM3\_GPIO->PDOR3 ^= 0x0004; | |
| 71 | } |  |
| 72 | } |  |
| 73 |  |  |
| 74 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 75 | /\* LED2 task \*/ | |
| 76 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 77 |  |  |
| 78 | static void led2( INT stacd, VP exinf ) | |
| 79 | { |  |
| 80 | ER ercd; |  |
| 81 |  |  |
| 82 | while(1) |  |
| 83 | { |  |
| 84 | Ercd = tk\_slp\_tsk(TMO\_FEVR); | |
| 85 | if(ercd != E\_OK){while(1){}} | |
| 86 |  |  |
| 87 | FM3\_GPIO->PDOR3 ^= 0x0008; | |
| 88 | } |  |
| 89 | } |  |
| 90 |  |  |
| 91 |  |  |
| 92 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 93 | /\* SWITCH1 task \*/ | |
| 94 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 95 |  |  |
| 96 | static void psw1(void) | |
| 97 | { |  |
| 98 | ER ercd; |  |
| 99 |  |  |
| 100 | ercd = tk\_wup\_tsk(LED1\_ID); | |
| 101 | if(ercd != E\_OK){while(1){}} | |
| 102 | } |  |
| 103 |  |  |
| 104 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 105 | /\* SWITCH2 task \*/ | |
| 106 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 107 |  |  |
| 108 | static void psw2(void) | |
| 109 | { |  |
| 110 | ER ercd; |  |
| 111 |  |  |
| 112 | ercd = tk\_wup\_tsk(LED2\_ID); | |
| 113 | if(ercd != E\_OK){while(1){}} | |
| 114 | } |  |
| 115 |  |  |
| 116 |  |  |
| 117 |  |  |
| 118 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 119 | /\* INTERRUPT HANDLER \*/ | |
| 120 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 121 |  |  |
| 122 | void int1(void) | |
| 123 | { |  |
| 124 | if(FM3\_EXTI->EIRR\_f.ER0) | |
| 125 | { |  |
| 126 | FM3\_EXTI->EICL\_f.ECL0 = 0; | |
| 127 | Psw1(); | |
| 128 | } |  |
| 129 | else if(FM3\_EXTI->EIRR\_f.ER1) | |
| 130 | { |  |
| 131 | FM3\_EXTI->EICL\_f.ECL1 = 0; | |
| 132 | Psw2(); | |
| 133 | } |  |
| 134 | Else |  |
| 135 | { |  |
| 136 | FM3\_EXTI->EIRR = 0x0000; | |
| 137 | FM3\_GPIO->PDOR3 ^= 0x000c; | |
| 138 | } |  |
| 139 | } |  |
| 140 |  |  |
| 141 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 142 | /\* Initialize task \*/ | |
| 143 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 144 |  |  |
| 145 | void uinit\_task(void) | |
| 146 | { |  |
| 147 | ER ercd; | |
| 148 | UINT intst; | |
| 149 |  |  |
| 150 | T\_CTSK ctsk; | |
| 151 | T\_DINT dint; | |
| 152 |  |  |
| 153 | static INT led1\_stack[0x400/4]; | |
| 154 | static INT led2\_stack[0x400/4]; | |
| 155 |  |  |
| 156 | /\*system clock set\*/ | |
| 157 | DI(intst); | |
| 158 |  |  |
| 159 | SysTick\_init(); | |
| 160 | sys\_init(); | |
| 161 |  |  |
| 162 | dint.intatr = TA\_HLNG; | |
| 163 | dint.inthdr = int1; | |
| 164 | ercd = /\* Define Interrupt \*/  ① | |
| 165 | if(ercd != E\_OK){while(1){}} | |
| 166 |  |  |
| 167 | EI(intst); | |
| 168 |  |  |
| 169 | LCD\_sys\_init(); | |
| 170 | LCD\_init(); | |
| 171 | LCD\_reset(); | |
| 172 |  |  |
| 173 | LCD\_gotoxy(0,0); | |
| 174 | LCD\_sendstring("uT-Kernel"); | |
| 175 |  |  |
| 176 | LCD\_gotoxy(0,1); | |
| 177 | LCD\_sendstring("Int Handler"); | |
| 178 |  |  |
| 179 | ctsk.exinf = (VP)1; | |
| 180 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 181 | ctsk.task = led1; | |
| 182 | ctsk.itskpri = 1; | |
| 183 | ctsk.stksz = (W)sizeof(led1\_stack); | |
| 184 | ctsk.bufptr = led1\_stack; | |
| 185 | LED1\_ID = tk\_cre\_tsk(&ctsk); | |
| 186 |  |  |
| 187 | ctsk.exinf = (VP)2; | |
| 188 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 189 | ctsk.task = led2; | |
| 190 | ctsk.itskpri = 2; | |
| 191 | ctsk.stksz = (W)sizeof(led2\_stack); | |
| 192 | ctsk.bufptr = led2\_stack; | |
| 193 | LED2\_ID = tk\_cre\_tsk(&ctsk); | |
| 194 |  |  |
| 195 | tk\_sta\_tsk(LED1\_ID, NULL); | |
| 196 | tk\_sta\_tsk(LED2\_ID, NULL); | |
| 197 | } |  |

演習７ 周期ハンドラ

■ プロジェクトフォルダ

C:\ut\_realos\utkernel\7-M\uT-Kernel\_Samples\cyclic

■ プロゴラム概要

・タスクは，lcd1(優先度1)，周期ハンドラ cyc1 から構成される

・初期化タスク uinit\_task は周期ハンドラとタスクを動的に生成し，タスクを起動させる．

周期ハンドラはlcd1タスクにより動作開始される．

・lcd1タスクは時間を表示したら，起床待ち状態に遷移する．

また，周期ハンドラ cyc1 は1s毎に起動しており，周期ハンドラの中で時間カウントされている．

時間カウントが終了したらlcd1タスクを起床させる．

起床したlcd1タスクは時間を表示する．以降この処理を繰り返す．

lcd1タスク cyc1タスク

優先度：１ 周期：1000ms

スタート

時間表示

スタート

時間++

自タスク

起床待ち

lcd1()

タスク起床

図9 演習７の各タスクのフローチャート

|  |  |  |
| --- | --- | --- |
| 1 | /\* |  |
| 2 | \*---------------------------------------------------------------------- | |
| 3 | \* micro T-Kernel | |
| 4 | \* |  |
| 5 | \* Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved. | |
| 6 | \* micro T-Kernel is distributed under the micro T-License. | |
| 7 | \*---------------------------------------------------------------------- | |
| 8 | \* micro T-REALOS/M3 | |
| 9 | \* |  |
| 10 | \* Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved. | |
| 11 | \* This product uses the Source Code of micro T-Kernel under the micro | |
| 12 | \* T-License granted by the T-Engine forum (http://www.t-engine.org). | |
| 13 | \* Last update : 2010/05/11 | |
| 14 | \*---------------------------------------------------------------------- | |
| 15 | \* |  |
| 16 | \* Version: 1.01.00 | |
| 17 | \* Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25. | |
| 18 | \* |  |
| 19 | \*---------------------------------------------------------------------- | |
| 20 | \*/ |  |
| 21 |  |  |
| 22 | /\* |  |
| 23 | \* | init\_task.c (usermain) |
| 24 | \* | User Main |
| 25 | \*/ |  |
| 26 |  |  |
| 27 | #include <tk/tkernel.h> | |
| 28 |  |  |
| 29 | #include "init\_task.h" | |
| 30 |  |  |
| 31 | #include "mcu.h" | |
| 32 |  |  |
| 33 | #define USE\_WDG 0 | |
| 34 |  |  |
| 35 | #if USE\_WDG == 0 | |
| 36 | #define WDG\_CTL (\*(volatile UW \*)0x40011008) | |
| 37 | #define WDG\_LCK (\*(volatile UW \*)0x40011C00) | |
| 38 | #endif |  |
| 39 |  |  |
| 40 | ID LCD1\_ID; | |
| 41 | ID CYC1\_ID; | |
| 42 |  |  |
| 43 | unsigned int sec = 0; | |
| 44 | unsigned int dsec = 0; | |
| 45 | unsigned int min = 0; | |
| 46 | unsigned int dmin = 0; | |
| 47 |  |  |
| 48 | extern void \_\_reset( void ); | |
| 49 |  |  |
| 50 | EXPORT int main( void ) | |
| 51 | { |  |
| 52 | #if USE\_WDG == 0 | |
| 53 | WDG\_LCK = 0x1acce551; | |
| 54 | WDG\_LCK = 0xe5331aae; | |
| 55 | WDG\_CTL = 0x00000000; | |
| 56 | #endif |  |
| 57 |  |  |
| 58 | \_\_reset(); | |
| 59 | while(1); | |
| 60 | } |  |
| 61 |  |  |
| 62 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 63 | /\* LCD TASK \*/ | |
| 64 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 65 |  |  |
| 66 | static void lcd1( INT stacd, VP exinf ) | |
| 67 | { |  |
| 68 | ER ercd; | |
| 69 |  |  |
| 70 | ercd = /\* Start Cyclic Handler \*/  ① | |
| 71 | if(ercd != E\_OK){while(1){}} | |
| 72 |  |  |
| 73 | LCD\_gotoxy(2,1); | |
| 74 | LCD\_sendstring(":"); | |
| 75 |  |  |
| 76 | while(1) | |
| 77 | { |  |
| 78 | LCD\_gotoxy(0,1); | |
| 79 | LCD\_sendchar(dmin+'0'); | |
| 80 |  |  |
| 81 | LCD\_gotoxy(1,1); | |
| 82 | LCD\_sendchar(min+'0'); | |
| 83 |  |  |
| 84 | LCD\_gotoxy(3,1); | |
| 85 | LCD\_sendchar(dsec+'0'); | |
| 86 |  |  |
| 87 | LCD\_gotoxy(4,1); | |
| 88 | LCD\_sendchar(sec+'0'); | |
| 89 |  |  |
| 90 | ercd = tk\_slp\_tsk(TMO\_FEVR); | |
| 91 | if(ercd != E\_OK){while(1){}} | |
| 92 | } |  |
| 93 | } |  |
| 94 |  |  |
| 95 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 96 | /\* CYCLIC HANDLER \*/ | |
| 97 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 98 |  |  |
| 99 | static void cyc1( INT stacd, VP exinf ) | |
| 100 | { |  |
| 101 | ER ercd; |  |
| 102 |  |  |
| 103 | sec++; |  |
| 104 |  |  |
| 105 | if(sec==10) | |
| 106 | { |  |
| 107 | sec=0; |  |
| 108 | dsec++; | |
| 109 |  |  |
| 110 | if(dsec==6) | |
| 111 | { |  |
| 112 |  |  |
| 113 | dsec=0; | |
| 114 | min++; | |
| 115 |  |  |
| 116 | if(min==10) | |
| 117 | { |  |
| 118 | min=0; | |
| 119 | dmin++; | |
| 120 |  |  |
| 121 | if(dmin==6) | |
| 122 |  |  |
| 123 | dmin=0; | |
| 124 | } |  |
| 125 | } |  |
| 126 | } |  |
| 127 |  |  |
| 128 | ercd = tk\_wup\_tsk(LCD1\_ID); | |
| 129 | if(ercd != E\_OK){while(1){}} | |
| 130 | } |  |
| 131 |  |  |
| 132 |  |  |
| 133 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 134 | /\* Initialize task \*/ | |
| 135 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 136 |  |  |
| 137 | void uinit\_task(void) | |
| 138 | { |  |
| 139 | T\_CTSK ctsk; | |
| 140 | T\_CCYC ccyc; | |
| 141 | static INT lcd1\_stack[0x400/4]; | |
| 142 |  |  |
| 143 | /\*system clock set\*/ | |
| 144 | SysTick\_init(); | |
| 145 | sys\_init(); | |
| 146 |  |  |
| 147 | LCD\_sys\_init(); | |
| 148 | LCD\_init(); | |
| 149 | LCD\_reset(); | |
| 150 |  |  |
| 151 | LCD\_gotoxy(0,0); | |
| 152 | LCD\_sendstring("cyclic handler"); | |
| 153 |  |  |
| 154 | ccyc.exinf = (VP)1; | |
| 155 | ccyc.cycatr = TA\_HLNG; | |
| 156 | ccyc.cychdr = cyc1; | |
| 157 | ccyc.cyctim = /\* Cycle Time = 1s \*/  ② | |
| 158 | ccyc.cycphs = 0; | |
| 159 | CYC1\_ID = /\* Create Cyclic Handler \*/  ③ | |
| 160 |  |  |
| 161 | ctsk.exinf = (VP)1; | |
| 162 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 163 | ctsk.task = lcd1; | |
| 164 | ctsk.itskpri = 1; | |
| 165 | ctsk.stksz = (W)sizeof(lcd1\_stack); | |
| 166 | ctsk.bufptr = lcd1\_stack; | |
| 167 | LCD1\_ID = tk\_cre\_tsk(&ctsk); | |
| 168 |  |  |
| 169 | tk\_sta\_tsk(LCD1\_ID, NULL); | |
| 170 | } |  |

演習８ タイムシェアリング

■ プロジェクトフォルダ

C:\ut\_realos\utkernel\7-M\uT-Kernel\_Samples\time\_share

■ プロゴラム概要

・タスクは，led1(優先度2)，led2(優先度2) ，led3(優先度2) と

周期ハンドラ cyc1 から構成される

・初期化タスク uinit\_task は周期ハンドラとタスクを動的に生成し，各タスクを起動させる．

・各タスクは実行状態になったら該当するLEDを点灯．

また，周期ハンドラ cyc1 は1s毎に起動しており，周期ハンドラの中で各タスクの優先順位を

回転させている．これにより全タスクの優先度が同じにもかかわらず，実行中のタスクが順番に

切り替わる．

led1タスク led2タスク

優先度：２ 優先度：２

スタート

LED1 ON/OFF

スタート

LED2 ON/OFF

led3タスク cyc1ハンドラ

優先度：２ 周期：1000ms

スタート

LED3 ON/OFF

スタート

タスク優先順位

回転

図10 演習８の各タスクのフローチャート

|  |  |  |
| --- | --- | --- |
| 1 | /\* |  |
| 2 | \*---------------------------------------------------------------------- | |
| 3 | \* micro T-Kernel | |
| 4 | \* |  |
| 5 | \* Copyright (C) 2006-2008 by Ken Sakamura. All rights reserved. | |
| 6 | \* micro T-Kernel is distributed under the micro T-License. | |
| 7 | \*---------------------------------------------------------------------- | |
| 8 | \* micro T-REALOS/M3 | |
| 9 | \* |  |
| 10 | \* Copyright (C) FUJITSU SEMICONDUCTOR LIMITED 2010 All rights reserved. | |
| 11 | \* This product uses the Source Code of micro T-Kernel under the micro | |
| 12 | \* T-License granted by the T-Engine forum (http://www.t-engine.org). | |
| 13 | \* Last update : 2010/05/11 | |
| 14 | \*---------------------------------------------------------------------- | |
| 15 | \* |  |
| 16 | \* Version: 1.01.00 | |
| 17 | \* Released by T-Engine Forum(http://www.t-engine.org) at 2008/02/25. | |
| 18 | \* |  |
| 19 | \*---------------------------------------------------------------------- | |
| 20 | \*/ |  |
| 21 |  |  |
| 22 | /\* |  |
| 23 | \* | init\_task.c (usermain) |
| 24 | \* | User Main |
| 25 | \*/ |  |
| 26 |  |  |
| 27 | #include <tk/tkernel.h> | |
| 28 |  |  |
| 29 | #include "init\_task.h" | |
| 30 |  |  |
| 31 | #include "mcu.h" | |
| 32 |  |  |
| 33 | #define USE\_WDG | 0 |
| 34 |  |  |
| 35 | #if USE\_WDG == 0 | |
| 36 | #define WDG\_CTL (\*(volatile UW \*)0x40011008) | |
| 37 | #define WDG\_LCK (\*(volatile UW \*)0x40011C00) | |
| 38 | #endif |  |
| 39 |  |  |
| 40 | ID LED1\_ID; | |
| 41 | ID LED2\_ID; | |
| 42 | ID LED3\_ID; | |
| 43 | ID CYC1\_ID; | |
| 44 |  |  |
| 45 | extern void \_\_reset( void ); | |
| 46 |  |  |
| 47 | EXPORT int main( void ) | |
| 48 | { |  |
| 49 | #if USE\_WDG == 0 | |
| 50 | WDG\_LCK = 0x1acce551; | |
| 51 | WDG\_LCK = 0xe5331aae; | |
| 52 | WDG\_CTL = 0x00000000; | |
| 53 | #endif |  |
| 54 |  |  |
| 55 | \_\_reset(); | |
| 56 | while(1); | |
| 57 | } |  |
| 58 |  |  |
| 59 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 60 | /\* LED1 TASK \*/ | |
| 61 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 62 |  |  |
| 63 | static void led1( INT stacd, VP exinf ) | |
| 64 | { |  |
| 65 | while(1) | |
| 66 | { |  |
| 67 | FM3\_GPIO->PDOR3 = 0x001c; | |
| 68 | FM3\_GPIO->PDOR3 ^= 0x0004; | |
| 69 | } |  |
| 70 | } |  |
| 71 |  |  |
| 72 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 73 | /\* LED2 TASK \*/ | |
| 74 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 75 |  |  |
| 76 | static void led2( INT stacd, VP exinf ) | |
| 77 | { |  |
| 78 | while(1) | |
| 79 | { |  |
| 80 | FM3\_GPIO->PDOR3 = 0x001c; | |
| 81 | FM3\_GPIO->PDOR3 ^= 0x0008; | |
| 82 | } |  |
| 83 | } |  |
| 84 |  |  |
| 85 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 86 | /\* LED3 TASK \*/ | |
| 87 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 88 |  |  |
| 89 | static void led3( INT stacd, VP exinf ) | |
| 90 | { |  |
| 91 | while(1) | |
| 92 | { |  |
| 93 | FM3\_GPIO->PDOR3 = 0x001c; | |
| 94 | FM3\_GPIO->PDOR3 ^= 0x0010; | |
| 95 | } |  |
| 96 | } |  |
| 97 |  |  |
| 98 |  |  |
| 99 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 100 | /\* CYCLIC HANDLER \*/ | |
| 101 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 102 |  |  |
| 103 | static void cyc1( INT stacd, VP exinf ) | |
| 104 | { |  |
| 105 | ER ercd; |  |
| 106 |  |  |
| 107 | ercd = /\* Rotate Ready Queue \*/  ① | |
| 108 | if(ercd != E\_OK){while(1){}} | |
| 109 | } |  |
| 110 |  |  |
| 111 |  |  |
| 112 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 113 | /\* Initialize task \*/ | |
| 114 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ | |
| 115 |  |  |
| 116 | void uinit\_task(void) | |
| 117 | { |  |
| 118 | ER ercd; | |
| 119 |  |  |
| 120 | T\_CTSK ctsk; | |
| 121 | T\_CCYC ccyc; | |
| 122 | static INT led1\_stack[0x400/4]; | |
| 123 | static INT led2\_stack[0x400/4]; | |
| 124 | static INT led3\_stack[0x400/4]; | |
| 125 |  |  |
| 126 | /\*system clock set\*/ | |
| 127 | SysTick\_init(); | |
| 128 | sys\_init(); | |
| 129 |  |  |
| 130 | LCD\_sys\_init(); | |
| 131 | LCD\_init(); | |
| 132 | LCD\_reset(); | |
| 133 |  |  |
| 134 | LCD\_gotoxy(0,0); | |
| 135 | LCD\_sendstring("uT-Kernel"); | |
| 136 | LCD\_gotoxy(0,1); | |
| 137 | LCD\_sendstring("time sharing"); | |
| 138 |  |  |
| 139 | ccyc.exinf = (VP)1; | |
| 140 | ccyc.cycatr = TA\_HLNG | TA\_STA; | |
| 141 | ccyc.cychdr = cyc1; | |
| 142 | ccyc.cyctim = /\* Cycle Time = 1s \*/  ② | |
| 143 | ccyc.cycphs = 0; | |
| 144 | CYC1\_ID = /\* Create Cyclic Handler \*/  ③ | |
| 145 |  |  |
| 146 | ctsk.exinf = (VP)1; | |
| 147 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 148 | ctsk.task = led1; | |
| 149 | ctsk.itskpri = 2; | |
| 150 | ctsk.stksz = (W)sizeof(led1\_stack); | |
| 151 | ctsk.bufptr = led1\_stack; | |
| 152 | LED1\_ID = tk\_cre\_tsk(&ctsk); | |
| 153 |  |  |
| 154 | ctsk.exinf = (VP)1; | |
| 155 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 156 | ctsk.task = led2; | |
| 157 | ctsk.itskpri = 2; | |
| 158 | ctsk.stksz = (W)sizeof(led2\_stack); | |
| 159 | ctsk.bufptr = led2\_stack; | |
| 160 | LED2\_ID = tk\_cre\_tsk(&ctsk); | |
| 161 |  |  |
| 162 | ctsk.exinf = (VP)1; | |
| 163 | ctsk.tskatr = TA\_HLNG | TA\_RNG0 | TA\_USERBUF; | |
| 164 | ctsk.task = led3; | |
| 165 | ctsk.itskpri = 2; | |
| 166 | ctsk.stksz = (W)sizeof(led3\_stack); | |
| 167 | ctsk.bufptr = led3\_stack; | |
| 168 | LED3\_ID = tk\_cre\_tsk(&ctsk); | |
| 169 |  |  |
| 170 | ercd = tk\_sta\_tsk(LED1\_ID, NULL); | |
| 171 | if(ercd != E\_OK){while(1){}} | |
| 172 |  |  |
| 173 | ercd = tk\_sta\_tsk(LED2\_ID, NULL); | |
| 174 | if(ercd != E\_OK){while(1){}} | |
| 175 |  |  |
| 176 | ercd = tk\_sta\_tsk(LED3\_ID, NULL); | |
| 177 | if(ercd != E\_OK){while(1){}} | |
| 178 | } |  |

演習9 総合演習１ スロットゲーム

■ プロジェクトフォルダ

C:\ut\_realos\utkernel\7-M\uT-Kernel\_Samples\app\_slot

■ プロゴラム概要

1. LCDにスロットの画面を表示 (初期状態：A|A|A）

2. 10ms毎にADCの値を検出し、

・回転速度を算出

3. ADCの値に基づいてスロットの回転スピードを設定 (100ms ～ 1s)

4. PSW1が押されたら、スロットの回転を

開始/左停止/真中停止/右停止/停止（これを繰り返す）

5. スロットが完全に停止したら、勝敗を判定し、結果をLCDに表示

演習10 総合演習２ 音楽プレーヤ

■ プロジェクトフォルダ

C:\ut\_realos\utkernel\7-M\uT-Kernel\_Samples\app\_music\_player

■ app\_music\_playerプログラム概要

1. 先頭の曲情報をLCDに表示

2. ジョイスティック(SW6)で曲を選択

3. PSW1押下で曲を再生

・再生時間が00:00からカウントアップされる

・曲の長さ分再生したら再生停止

・再度PSWを押したら一時停止

・（曲データはLED1～LED8に表示される）

4. PSW2押下で再生を停止

トロンフォーラム

トロン講習会

【実習】μT-Kernel入門

プログラミング演習テキスト

-------------------------------------------------------------------------------

2016年7月21日発行

発行所

トロンフォーラム

(YRPユビキタス・ネットワーキング研究所内)

〒141-0031 東京都品川区西五反田2-12-3 第一誠実ビル9F

URL: http://www.tron.org/ja/

TEL:03-5437-0572(代表) FAX:03-5437-2399(代表)

-------------------------------------------------------------------------------

　　　本テキストは、クリエイティブ・コモンズ 表示 - 継承 4.0 国際 ライセンスの下に提供

　　　されています。

<https://creativecommons.org/licenses/by-sa/4.0/>　　http://creativecommons.jp/wp/wp-content/uploads/2009/10/by-sa.png

　　　Copyright ©2016 TRON Forum

　　　【ご注意およびお願い】

　　　1.本テキストの中で第三者が著作権等の権利を有している箇所については、利用者の方が

　　　当該第三者から利用許諾を得てください。

　　　2.本テキストの内容については、その正確性、網羅性、特定目的への適合性等、一切の保

　　　証をしないほか、本テキストを利用したことにより損害が生じても著者は責任を負いま

　　　せん。

　　　3.本テキストをご利用いただく際、可能であれば office@tron.org までご利用者のお名前、

　　　ご所属、ご連絡先メールアドレスをご連絡いただければ幸いです。

---------------------------------------------------------------------------------------------------------------------

トロンフォーラム©2016

Cypress©2016

富士通エレクトロニクス株式会社©2016

Printed in Japan

回答例

演習１ タスク起動と優先度

1. tk\_ext\_tsk(); ② tk\_sta\_tsk(LED1\_ID,NULL);

③ tk\_sta\_tsk(LED2\_ID,NULL); ④ tk\_sta\_tsk(LED3\_ID, NULL);

演習２　起床待ちと起床

① tk\_wup\_tsk(LED2\_ID); ② tk\_slp\_tsk(TMO\_FEVR);

1. tk\_wup\_tsk(LED3\_ID);

演習３ セマフォ

① tk\_wai\_sem(SEM1\_ID,1,TMO\_FEVR); ② tk\_sig\_sem(SEM1\_ID,1);

③ tk\_wai\_sem(SEM1\_ID,2,TMO\_FEVR); ④ tk\_sig\_sem(SEM1\_ID,2);

⑤ tk\_wai\_sem(SEM1\_ID,3,TMO\_FEVR); ⑥ tk\_cre\_sem(&csem);

演習４　イベントフラグ

1. tk\_wai\_flg(FLG1\_ID,3,TWF\_ORW,&ptn,TMO\_FEVR);
2. tk\_clr\_flg(FLG1\_ID,0);
3. tk\_wai\_flg(FLG1\_ID,3,TWF\_ANDW,&ptn,TMO\_FEVR);
4. tk\_set\_flg(FLG1\_ID,flg);
5. tk\_cre\_flg(&cflg);

演習５　メールボックス

1. tk\_rcv\_mbx(MBX1\_ID,(T\_MSG\*\*)&rx\_mail,TMO\_FEVR);
2. tk\_snd\_mbx(MBX1\_ID,(T\_MSG\*)&tx\_mail);
3. tk\_cre\_mbx(&cmbx);

演習６　割込みハンドラ

1. tk\_def\_int(20,&dint);

演習７　周期ハンドラ

① tk\_sta\_cyc(CYC1\_ID); ② 1000 ③ tk\_cre\_cyc(&ccyc);

演習８　タイムシェアリング

1. tk\_rot\_rdq(TPRI\_RUN);