µC3/Standard ユーザーズガイド
プロセッサ依存部　ARM926EJ-S 編

Rev.2.0

August 20, 2015

はじめに

本書はμC3/Standard のプロセッサコア ARM926EJ-S に依存する事項を説明します。プロセッサに非依存の共通マニュアルである「μC3/Standard ユーザーズガイド」とあわせてお読みください。

TRON は、"The Real-time Operation system Nucleus"の略称です。
μITRON は、"Micro Industrial TRON"の略称です。
μITRON4.0 仕様の仕様書は、トロン協会のホームページ（http://www.assoc.tron.org/）から入手することができます。
μC3 は、イー・フォース株式会社の登録商標です。
本書で記載されている内容は、予告無く変更される場合があります。

改訂記録

Rev.	発行日	改訂内容	
		ページ	内容
1 版	2008.09	—	—
2 版	—	7,10,11, 13	Code Composer Studio への対応に関して追加
3 版	—	6,17,21, 22,23-27	Sample の内容を変更 パラメータを変更 【プログラム例】の_ddr_ti_dm355uart_init();のパラメータを変更 TMPA910 ドライバについて説明を追加
4 版	—	28-31	OMAP-L137、L138 ドライバについて説明を追加
5 版	—	6,32-35	ファイルツリーにドキュメントフォルダを追加 LPC-3200 用ドライバの説明を追加
6 版	—	19-43	i.MX25 と DM365 用ドライバの説明を追加し、第 5 章の構成を変更
7 版	—	24,28,29, 44-47	DM355 ドライバの dis_int/ena_int の使用制限を追加 DM365 ドライバの dis_int/ena_int の使用制限を追加 DM365 周期タイマドライバの TACE_GPT マクロの説明を追加 DM6446 用ドライバの説明を追加
8 版	—	37,39,40, 43,45,46	OMAP-L137,L138 ドライバの割込み受け付ける文章に修正 OMAP-L137,L138 ドライバの割込みレベルを 2～31 に修正 LPC3200 ドライバに clr_int を追記 LPC3200 標準 COM ドライバのハードフェアフロー制御の説明追記 DM6446 ドライバの割込み受け付ける文章に修正 DM6446 ドライバの割込みレベルを 2～31 に修正

9 版	―	8,11,13, 14	RVDS への対応に関して説明を追加
10 版	―	―	章 5.7.3_ddr_ti_omapl1xgpt_init の CH_GPT の設定範囲を 1～2 に修正
2.0	2015.08.20	―	文書管理番号は Rev 形式に変更 改訂記録の書式を変更 章 1.1，章 1.2 のフォルダ構成，パスの注釈を追加 本ドキュメントのファイル名を変更

目次

第１章　ファイル構成

１．１　ファイルツリー

デフォルトのインストールフォルダ C:¥uC3Std[1]のままインストールした場合には、
以下のフォルダ構成になります。

C:¥uC3Std

- Document ..... ユーザーズマニュアル、カーネルの Readme 等のドキュメント
- Kernel
  - inc ..... 共通インクルードファイル
    - ARMv5T ..... ARMv5TE 固有のインクルードファイル
  - lib ..... カーネルライブラリ
    - ARMv5T ..... カーネルライブラ生成用プロジェクトファイル
  - src ..... 共通ソースファイル
    - ARMv5T ..... ARMv5TE 固有のソースファイル
- Driver
  - inc ..... ドライバのインクルードファイル
  - src ..... ドライバのソースファイル
- Sample
  - XXXXXX .....評価ボード用のサンプル

[1] パッケージバージョンが 2.0.0 以降の場合，フォルダは C:¥uC3¥Standard となります。

μC3/Standard を使用するユーザプログラムは、インクルードパスとして以下を指定してください。

C:¥uC3Std[2]¥Kernel¥inc

C:¥uC3Std¥Kernel¥inc¥ARMv5T

C:¥uC3Std¥Driver¥inc

１．２　カーネルライブラリ

カーネルライブラリとして、リトルエンディアンとビッグエンディアン、ARM ステートと Thumb ステートの組み合わせで、次の 4 種類を用意しています。

【IAR Embedded Workbench】

ARM/Thumb	エンディアン	ライブラリ名
ARM	Little	uC3armv5tl.a
Thumb	Little	uC3thmv5tl.a
ARM	Big	uC3armv5tb.a
Thumb	Big	uC3thmv5tb.a

【Code Composer Studio】

ARM/Thumb	エンディアン	ライブラリ名
ARM	Little	uC3armv5tl.lib
Thumb	Little	uC3thmv5tl.lib
ARM	Big	uC3armv5tb.lib
Thumb	Big	uC3thmv5tb.lib

【ARM　RVDS 】

ARM/Thumb	エンディアン	ライブラリ名
ARM	Little	uC3armv5tl.l
Thumb	Little	uC3thmv5tl.l
ARM	Big	uC3armv5tb.l
Thumb	Big	uC3thmv5tb.l

Thumb ステートのカーネルライブラリは、ARM ステートの使用を最小限にとどめたカーネルで、コンパクトにできています。通常は、Thumb ステートのカーネルライブラリの使用をお勧めします。

[2] パッケージバージョンが 2.0.0 以降の場合，フォルダは C:¥uC3¥Standard となります。

第２章　プロセッサに依存する状態

２．１　割込みの種類

ARM コアでは、IRQ と FIQ の２系統の割込みを持っています。μC3/Standard では、IRQ 割込みのみを管理し、FIQ 割込みは常に許可状態となり管理していません。

２．１．１　割込みマスク

割込みをマスクするために、ステータスレジスタの IRQ 割込みビットを操作しています。chg_ims はマスクする割込みレベルを指定し、get_ims はマスクしている割込みレベルを返します。割込みレベル 0 は IRQ 割込み許可で、割込みレベル 1 は IRQ 割込み禁止です。

２．２　CPU ロック状態

μC3/Standard ARM 版の実装では、ステータスレジスタの IRQ 割込みビットを操作し、loc_cpu による CPU ロック状態を IRQ 割込み禁止、unl_cpu による CPU ロックの解除状態を IRQ 割込み許可として定義しています。したがって、loc_cpu()と chg_ims(1)、unl_cpu()と chg_ims(0)は同じ働きをします。

また、割込みハンドラ（割込みサービスルーチンを含め）の開始直後の状態は、割込みコントローラのドライバにより異なっています。特に明記していない場合は、CPU ロック解除状態で開始されます。

２．３　ディスパッチの保留状態

非タスクコンテキストでは、常に、ディスパッチ保留状態です。一方、タスクコンテキストでは、以下のいずれかの条件に当てはまれば、ディスパッチ保留状態です。

・CPU ロック状態
・ディスパッチ禁止状態

２．４　データ型

ARM 依存のμITRON4.0 仕様で規定しているデータ型は次の通りです。

INT	符号付き 32 ビット整数
UINT	符号無し 32 ビット整数
VP_INT	データタイプが定まらないものへのポインタまたは符号付き 32 ビット整数
FLGPTN	符号無し 32 ビット整数

２．５　割込みの禁止／許可

各割込み要因に対する割込み禁止／許可を行うシステムコール（dis_int, ena_int）の実装の有無は、割込みコントローラのドライバにより異なっています。

２．６　各コンテキストの実行時のモード

非タスクコンテキストは、システムモードで実行されます。タスクコンテキストは、デフォルトではシステムモードで実行されますが、タスク生成時のタスク属性に TA_USR を指定することにより、ユーザモードで実行することもできます。

第３章　カーネルのコンフィグレーション

３．１　システムスタック

μC3/Standard は、非タスクコンテキストはすべてシステムスタックで実行されます。そのため、システムスタックのサイズは、一番多くスタック領域を消費するタイムイベントハンドラのスタックサイズと、割込み処理（割込みハンドラや割込みサービスルーチン）で消費するスタックサイズを加算した値が目安になります。また、多重割込みが想定される場合には、それらのスタックサイズを加算する必要があります。

３．１．１　IAR Embedded Workbench でのシステムスタックの定義方法

EWARM の場合では、EW を起動して"options"を開きます。Category "Linker"の"Config"タブの"Linker configuration file"の"Edit"ボタンをクリックし、"Linker configuration file editor"を開きます。"Stack/Heap Sizes"タブの"CSTACK"の値がシステムスタックのサイズとなり、これを修正します。

３．１．２　Code Composer Studio でのシステムスタックの定義方法

CCS の場合では、リンカコマンドファイル（.cmd）の中でセクション定義と共に、領域の確保を行います。領域の確保は、シンボル_stack_end の定義前に、ポインタを加算することにより行えます。以下の例では、16KB（0x4000）を確保しています。

```
. += 4000h;
_stack_end = .;
```

３．１．３　RVDS でのシステムスタックの定義方法

RVDS の場合では、スキャッタファイル（.scat）の中で STACK セクションの定義と共に、領域の確保を行います。

３．２　各モードのスタック

μC3/Standard ARM 版では、ユーザ(システム)モード以外に、割込み(IRQ)モードとスーパバイザ(SVC)モードを使用するため、スタック領域の確保が必要です。その他のモードもデバッグのためにも、スタック領域を確保することをお勧めします。

スーパバイザモードのスタックサイズは、最少で 24 バイトを、ユーザ定義の例外ハンドラを用いる場合は、それで使用するスタックサイズを加算したサイズにします。

割込み(IRQ)モードのスタックサイズは、単一の割込み発生で 64 バイトを使用します。割込みがネストする場合は、ネスト数×64 バイトを加算したサイズにします。

３．２．１　IAR Embedded Workbench での各モードのスタックの定義方法

システムスタックの定義方法と同様に、"Linker configuration file editor"を開き設定します。

"Stack/Heap Sizes"タブの"SVC_STACK"の値がスーパバイザモードのスタックサイズに、"IRQ_STACK"の値が割込み(IRQ)モードのスタックサイズに、"FIQ_STACK"の値が高速割込み(FIQ)モードのスタックサイズに、"UND_STACK"の値が未定義(UND)モードのスタックサイズに、"ABT_STACK"の値がアボート(ABT)モードのスタックサイズになり、これを修正します。

３．２．２　Code Composer Studio での各モードのスタックの定義方法

システムスタックの定義方法と同様に、リンカコマンドファイルの中で領域の確保を行います。以下の例では、IRQ 割込みモードに 512B（0x200）を、その他のモードに 256B（0x100）を確保しています。

```
. += 100h;                  未定義モードのスタックサイズ
_und_stack_end = .;
. += 100h;                  スーパバイザのスタックサイズ
_svc_stack_end = .;
. += 100h;                  アボートモードのスタックサイズ
_abt_stack_end = .;
. += 200h;                  IRQ 割込みモードのスタックサイズ
_irq_stack_end = .;
. += 100h;                  FIQ 割込みモードのスタックサイズ
_fiq_stack_end = .;
```

３．２．３　RVDS での各モードのスタックの定義方法

システムスタックの定義方法と同様に、スキャッタファイルの中で領域の確保を行います。セクション名に制限はありませんが、サンプルのスタートアップファイルでは、スーパバイザモードは"SVC_STACK"、割込み(IRQ)モードは"IRQ_STACK"、高速割込み(FIQ)モードは"FIQ_STACK"、未定義(UND)モードは"UND_STACK"、アボート(ABT)モードは"ABT_STACK"のセクション名を使っています。

セクション VECTTBL はベクタテーブルで 8K バイト、セクション VINFTBL はダミーデータで 4 バイトが確保されます。セクション SYS は、カーネルの管理領域として必須で、256 バイトが確保されます。また、スキャッタファイルは、次のように記述します。

```
RAM_LOAD 0x80000000
{
    RAM_EXEC 0x80000000
    {
        prstmx25.o (.intvec, +FIRST)     ; Initialization code
        * (+RO)                          ;
    }

    RAM_DATA 0x80800000
    {
        * (+RW)                          ;
    }
    RAM_BSS +0x0
    {
        * (+ZI)                          ;
    }
    VINFTBL +0x0
    {
        * (VINFTBL)                      ;
    }
    VECTTBL +0x0
    {
        * (VECTTBL)                      ;
    }

    SYS_DATA +0x0
    {
        * (STKMEM)                       ;
        * (MPLMEM)                       ;
        * (SYSMEM)                       ;
    }

    FIQ_STACK 0x80FFF000 EMPTY 0x00000100
    {

    }
    UND_STACK +0x0       EMPTY 0x00000100
    {

    }
    ABT_STACK +0x0       EMPTY 0x00000100
    {

    }
    SVC_STACK +0x0       EMPTY 0x00000100
    {

    }
    IRQ_STACK +0x0       EMPTY 0x00000400
    {

    }
    STACK +0x0           EMPTY 0x00000700
    {

    }
    SYS_TBL +0x0
    {
        * (SYS)                          ;
    }
}
```

第４章　システムコール

４．１　CPU 例外ハンドラ

SWI 例外の CPU 例外ハンドラのみ、CPU 例外ハンドラ番号 EXC_SVC として定義可能です。ただし、システムコールの呼び出しができないなど、μITRON4.0 仕様の CPU 例外ハンドラとしての機能を満たしていません。"SWI xx"命令のイミディエート値は、0 から 9 までをμC3/Standard が予約し、10 以上をユーザに開放しています。

SWI 例外の CPU 例外ハンドラは、次の形式で記述します。swino は、"SWI xx"命令のイミディエート値です。

```
void svc_exchdr(UW swino)
{
    SWI 例外の CPU 例外ハンドラ本体
}
```

４．２　割込みハンドラと割込みサービスルーチン

割込みコントローラのドライバに依存するため、対応する割込みコントローラのドライバを参照してください。

第５章　CPU 依存ドライバ

５．１　標準 MMU ドライバ DDR_ARM926_MMU.xxx

_ddr_arm926_mmu_init　　MMU の初期化

【書式】

void　_ddr_arm926_mmu_init(void)

【解説】

MMU を初期化し、物理アドレスから仮想アドレスへの変換と属性を割り付け、変換テーブルのアドレスを定義します。通常は、main 関数の実行前の、初期化時に行います。このファイルはアセンブラ言語で記述され、開発環境により異なった拡張子を持ちます。

尚、属性の定義は以下から選択します。

ATR_RO　書き込み禁止(特権モードでは読み書き可)
ATR_RW　読み書き可
ATR_NA　アクセス禁止
ATR_CE　キャッシュ使用
ATR_CD　キャッシュ未使用
ATR_BE　書き込みバッファ使用
ATR_BD　書き込みバッファ未使用

これらの初期化内容は、DDR_ARM926_MMU_cfg.xxx ファイル（開発環境により異なった拡張子を持つ）に記述します。領域は、次のように 4 つのパラメータを一組にして複数組を定義でき、0 のパラメータを終端とします。また、領域の定義が重複した場合には、後に定義した内容が優先されます。

以下は、EWARM での記述例となり、CCS では疑似命令などが異なります。

```
        DATA
mmu_cfgtbl
;               バイト数,       物理アドレス,    仮想アドレス,   属性
        DCD     0x00001000,     0xC0000000,     0x00000000,     ATR_RO+ATR_CE+ATR_BD
                :
        DCD     0x00000000,     0x00000000,     0x00000000,     0
mmu_ttb
        DCD     0xC3F00000
```

_kernel_synch_cache　　データ同期バッファ

【書式】

```
void　_kernel_synch_cache(void)
```

【解説】

書き込みバッファにデータが存在する場合には、そのデータを書き戻します。

_kernel_clean_data_cache　　データキャッシュのクリーニング

【書式】

```
void　_kernel_clean_data_cache(void *addr, SIZE size)
```

【パラメータ】

void *	addr	クリーニングする領域の先頭番地
SIZE	size	クリーニングする領域のバイト数

【解説】

addr で指定される番地から、size で指定されるバイト数のデータキャッシュを、外部メモリに書き戻し無効化します。当該領域のデータキャッシュが存在しない場合には、影響を与えません。また、同じキャッシュラインにクリーニングする領域と他の領域が存在する場合には、その領域もクリーニングされます。

_kernel_invalid_data_cache　　データキャッシュの無効化

【書式】

void　_kernel_invalid_data_cache(void *addr, SIZE size)

【パラメータ】

void*	addr	無効化する領域の先頭番地
SIZE	size	無効化する領域のバイト数

【解説】

addr で指定される番地から、size で指定されるバイト数のデータキャッシュを無効化します。当該領域のデータキャッシュが存在しない場合には、影響を与えません。また、同じキャッシュラインに無効化する領域と他の領域が存在する場合には、その領域も無効化されます。

_kernel_invalid_inst_cache　　命令キャッシュの無効化

【書式】

void　_kernel_invalid_inst_cache(void *addr, SIZE size)

【パラメータ】

void *	addr	無効化する領域の先頭番地
SIZE	size	無効化する領域のバイト数

【解説】

addr で指定される番地から、size で指定されるバイト数の命令キャッシュを無効化します。当該領域の命令キャッシュが存在しない場合には、影響を与えません。また、同じキャッシュラインに無効化する領域と他の領域が存在する場合には、その領域も無効化されます。

５．２　i.MX21 のドライバ

５．２．１　I/O アドレスと割込み番号の定義

i.MX21 の I/O アドレスと割込み番号の定義ファイルは、ドライバのインクルードフォルダに mc9328mx21.h を用意しています。

５．２．２　割込みドライバ DDR_FS_MX21AITC.c

割込み要因毎に込み番号を持ち、それぞれ独立した割込みハンドラの生成や割込みサービスルーチンの生成が可能になっています。また、設定された割込み優先度に従って多重割込みも受け付けます。

割込みハンドラや割込みサービスルーチンの開始直後は、CPU ロックは解除状態になっています。

_ddr_fs_mx21aitc_init	割込みコントローラの初期化

【書式】

```
void　_ddr_fs_mx21aitc_init(void)
```

【解説】

割込みコントローラを初期化します。また、割込みコントローラの初期化は、次のように、カーネルの起動前に行います。

【プログラム例】

```
_ddr_fs_mx21aitc_init();
        :
start_uC3(&csys, initpr);
```

５．２．３　周期タイマドライバ DDR_FS_MX21GPT.c

_ddr_fs_mx21gpt_init　　周期タイマの初期化

【書式】

void　_ddr_fs_mx21gpt_init(UINT tick)

【パラメータ】

UINT	tick	チック時間

【解説】

周期タイマに用いるため、GPT（General Purpose Timer）の 1 チャネルを初期化します。これの初期化内容を、次の要領で DDR_FS_MX21GPT_cfg.h ファイルに記述します。

周期タイマ割込みの割込みレベルを、0（最低）～15（最高）のいずれかを指定します。

```
#define　IPL_GPT　0
```

周期タイマとして用いる GPT のチャネルを、1～3 のいずれかを指定します。

```
#define　CH_GPT　1
```

５．２．４　標準 COM ドライバ DDR_FS_MX21UART.c

_ddr_mx21uart_init　　UART ポートの初期化

【書式】

ER　_ddr_mx21uart_init(ID devid, volatile struct t_uart *uart_port)

【パラメータ】

ID	devid	デバイスの ID 番号
volatile struct t_uart *	uart_port	UART のデバイスアドレス

【解説】

指定された UART（Universal Asynchronous Receiver/Transmitter）の 1 チャネルを、指定されたデバイス ID の標準 COM ポートとして初期化します。

【プログラム例】

UART のチャネル 4 を ID_UART4 のデバイス ID 番号で初期化する場合には、次のような記述になります。

```
#include "mc9328mx21.h"
extern ER   _ddr_mx21uart_init(ID, volatile struct t_uart *);
            :
      _ddr_mx21uart_init(ID_UART4, &UART4);
```

初期化する内容は、初期化する全チャネルをまとめ、DDR_FS_MX21UART_cfg.h ファイルに記述します。 1 つのチャネルは、次の 10 のパラメータにより構成されます。

```
#define   UART_n              /*チャネル n を使用する*/

#define   CTSRTS_n    1       /* CTS/RTS ピン有効(1)/無効(0) */
#define   TXBUF_SZn   1024    /*送信バッファサイズ(0 以上) */
#define   RXBUF_SZn   1024    /*受信バッファサイズ(1 以上) */
#define   XOFF_SZn    768     /* XOFF 送出受信バッファデータ数トリガ*/
#define   XON_SZn     128     /* XON 送出受信バッファデータ数トリガ*/
#define   RXTL_n      24      /*レシーブ FIFO データ数トリガ(0～32) */
#define   TXTL_n      8       /*トランスミット FIFO データ数トリガ(2～32) */
#define   RTSTL_n     24      /* RTS 出力アクティブトリガ(0～32) */
#define   IPL_UARTn   2       /*割込みレベル(0～15) */
```

５．３　i.MX25 のドライバ

５．３．１　I/O アドレスと割込み番号の定義

i.MX25 の I/O アドレスと割込み番号の定義ファイルは、ドライバのインクルードフォルダに imx25.h を用意しています。

５．３．２　割込みドライバ DDR_FS_MX25ASIC.c

割込み要因毎に込み番号を持ち、それぞれ独立した割込みハンドラの生成や割込みサービスルーチンの生成が可能になっています。また、設定された割込み優先度に従って多重割込みも受け付けます。

割込みハンドラや割込みサービスルーチンの開始直後は、CPU ロックは解除状態になっています。

_ddr_fs_mx25asic_init	割込みコントローラの初期化

【書式】

```
void　_ddr_fs_mx25asic_init(void)
```

【解説】

割込みコントローラを初期化します。また、割込みコントローラの初期化は、次のように、カーネルの起動前に行います。

【プログラム例】

```
_ddr_fs_mx25asic_init();
        :
start_uC3(&csys, initpr);
```

dis_int　　割込みの禁止

【書式】

ER ercd = dis_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを禁止します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。

ena_int　　割込みの許可

【書式】

ER ercd = ena_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを許可します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。

５．３．３　周期タイマドライバ DDR_FS_MX25EPIT.c

_ddr_fs_mx25epit_init　　周期タイマの初期化

【書式】

```
void　_ddr_fs_mx25epit_init(UINT tick)
```

【パラメータ】

UINT	tick	チック時間

【解説】

周期タイマに用いるため、EPIT（Enhanced Periodic Interrupt Timer）の 1 チャネルを初期化します。これの初期化内容を、次の要領で DDR_FS_MX25EPIT_cfg.h ファイルに記述します。

外付けの高周波発振器の周波数を Hz 単位で指定します。

```
#define CLKIH　　24000000
```

外付けのリアルタイムクロック用の発振器の周波数を Hz 単位で指定します。

```
#define CLKIL　　32768
```

周期タイマ割込みの割込みレベルを、0（最低）～15（最高）のいずれかを指定します。

```
#define　IPL_EPIT　0
```

周期タイマとして用いる EPIT のチャネルを、1～2 のいずれかを指定します。

```
#define　CH_EPIT　1
```

５．３．４　標準 COM ドライバ DDR_FS_MX25UART.c

_ddr_mx25uart_init　　UART ポートの初期化

【書式】

ER　_ddr_mx25uart_init(ID devid, volatile struct t_uart *uart_port)

【パラメータ】

ID	devid	デバイスの ID 番号
volatile struct t_uart *	uart_port	UART のデバイスアドレス

【解説】

指定された UART（Universal Asynchronous Receiver/Transmitter）の 1 チャネルを、指定されたデバイス ID の標準 COM ポートとして初期化します。

【プログラム例】

UART のチャネル 4 を ID_UART4 のデバイス ID 番号で初期化する場合には、次のような記述になります。

```
#include "imx25.h"
extern ER   _ddr_mx25uart_init(ID, volatile struct t_uart *);
            :
      _ddr_mx25uart_init(ID_UART4, &REG_UART4);
```

初期化する内容は、初期化する全チャネルをまとめ、DDR_FS_MX25UART_cfg.h ファイルに記述します。 1 つのチャネルは、次の 10 のパラメータにより構成されます。

```
#define   UART_n                /*チャネル n を使用する*/

#define   TXBUF_SZn   1024      /*  送信バッファサイズ(0 以上) */
#define   RXBUF_SZn   1024      /*  受信バッファサイズ(1 以上) */
#define   XOFF_SZn    768       /* XOFF 送出受信バッファデータ数トリガ*/
#define   XON_SZn     128       /* XON 送出受信バッファデータ数トリガ*/
#define   RXTL_n      24        /*レシーブ FIFO データ数トリガ(0～32) */
#define   TXTL_n      8         /*トランスミット FIFO データ数トリガ(2～32) */
#define   RTSTL_n     24        /* RTS 出力アクティブトリガ(0～32) */
#define   DCEDTE_n    1         /* DCE(0)/DTE(1)の選択  */
#define   IPL_UARTn   2         /*  割込みレベル(0～15) */
```

５．４　DM355 のドライバ

５．４．１　I/O アドレスと割込み番号の定義

DM355 の I/O アドレスと割込み番号の定義ファイルは、ドライバのインクルードフォルダに dm355.h を用意しています。

５．４．２　割込みドライバ DDR_TI_DM355AITC.c

割込み要因毎に割込み番号を持ち、それぞれ独立した割込みハンドラの生成や割込みサービスルーチンの生成が可能になっています。また、本割込みドライバでは多重割込みを受け付けません。

割込みハンドラや割込みサービスルーチンの開始直後は、CPU ロックの解除状態になっています。

_ddr_ti_dm355aitc_init　　割込みコントローラの初期化

【書式】

```
void　_ddr_ti_dm355aitc_init(void)
```

【解説】

割込みコントローラを初期化します。また、割込みコントローラの初期化は、次のように、カーネルの起動前に行います。

【プログラム例】

```
_ddr_ti_dm355aitc_init();
        :
start_uC3(&csys, initpr);
```

dis_int　　割込みの禁止

【書式】

ER ercd = dis_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを禁止します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。ただし、割込みハンドラからは呼び出すことはできません。

ena_int　　割込みの許可

【書式】

ER ercd = ena_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを許可します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。ただし、割込みハンドラからは呼び出すことはできません。

５．４．３　周期タイマドライバ DDR_TI_DM355GPT.c

_ddr_ti_dm355gpt_init　　周期タイマの初期化

【書式】

void　_ddr_ti_dm355gpt_init(UINT tick)

【パラメータ】

UINT	tick	チック時間

【解説】

周期タイマに用いるため、GPT（General Purpose Timer）の 1 チャネルを初期化します。これの初期化内容を、次の要領で DDR_TI_DM355GPT_cfg.h ファイルに記述します。

周期タイマ割込みの割込みレベルを、2（最高）～7（最低）のいずれかを指定します。

```
#define　IPL_GPT　4
```

周期タイマとして用いる GPT のチャネルを、1～3 のいずれかを指定します。

```
#define　CH_GPT　1
```

トレース機能のタイムスタンプに用いる場合には、周期タイマとして用いた GPT 以外のチャネルを指定します。トレース機能を用いない場合には、定義しません。

```
#define　TRACE_GPT　2
```

GPT のクロックソースの周波数を指定します。

```
#define　AUXCLK　(24*1000000uL)
```

５．４．４　標準 COM ドライバ DDR_TI_DM355UART.c

_ddr_ti_dm355uart_init　UART ポートの初期化

【書式】

ER　_ddr_ti_dm355uart_init(ID devid, volatile struct t_uart *uart_port)

【パラメータ】

ID	devid	デバイスの ID 番号
volatile struct t_uart *	uart_port	UART のデバイスアドレス

【解説】

指定された UART（Universal Asynchronous Receiver/Transmitter）の 1 チャネルを、指定されたデバイス ID の標準 COM ポートとして初期化します。

【プログラム例】

例えば、UART のチャネル 1 を ID_UART1 のデバイス ID 番号で初期化する場合には、次のような記述になります。

```
#include "dm355.h"
extern ER　_ddr_ti_dm355uart_init(ID, volatile struct t_uart *);
:
_ddr_ti_dm355uart_init(ID_UART1, &REG_UART1);
```

初期化する内容は、初期化する全チャネルをまとめ、DDR_TI_DM355UART_cfg.h ファイルに記述します。１つのチャネルは、次の 8 つのパラメータにより構成されます。

```
#define　UART_n                /*チャネル n を使用する*/

#define　CTSRTS_n　1　　/* CTS/RTS ピン有効(1)/無効(0) */
#define　TXBUF_SZn　1024　　/*送信バッファサイズ(0 以上) */
#define　RXBUF_SZn　1024　　/*受信バッファサイズ(1 以上) */
#define　XOFF_SZn　768　　/* XOFF 送出受信バッファデータ数トリガ*/
#define　XON_SZn　128　　/* XON 送出受信バッファデータ数トリガ*/
#define　RXTL_n　8　　/*レシーブ FIFO データ数トリガ(1,4,8,14) */
#define　IPL_UARTn　2　　/*割込みレベル(2～7) */
```

５．５　DM365 のドライバ

５．５．１　I/O アドレスと割込み番号の定義

DM365 の I/O アドレスと割込み番号の定義ファイルは、ドライバのインクルードフォルダに dm365.h を用意しています。

５．５．２　割込みドライバ DDR_TI_DM36xAITC.c

割込み要因毎に割込み番号を持ち、それぞれ独立した割込みハンドラの生成や割込みサービスルーチンの生成が可能になっています。また、本割込みドライバでは多重割込みを受け付けません。

割込みハンドラや割込みサービスルーチンの開始直後は、CPU ロックの解除状態になっています。

_ddr_ti_dm36xaitc_init　　割込みコントローラの初期化

【書式】

```
void   _ddr_ti_dm36xaitc_init(void)
```

【解説】

割込みコントローラを初期化します。また、割込みコントローラの初期化は、次のように、カーネルの起動前に行います。

【プログラム例】

```
_ddr_ti_dm36xaitc_init();
        :
start_uC3(&csys, initpr);
```

dis_int　　割込みの禁止

【書式】

ER ercd = dis_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを禁止します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。ただし、割込みハンドラからは呼び出すことはできません。

ena_int　　割込みの許可

【書式】

ER ercd = ena_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを許可します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。ただし、割込みハンドラからは呼び出すことはできません。

５．９．３　周期タイマドライバ DDR_TI_DM36xGPT.c

_ddr_ti_dm36xgpt_init　　周期タイマの初期化

【書式】

void　_ddr_ti_dm36xgpt_init(UINT tick)

【パラメータ】

UINT	tick	チック時間

【解説】

周期タイマに用いるため、GPT（General Purpose Timer）の 1 チャネルを初期化します。これの初期化内容を、次の要領で DDR_TI_DM36xGPT_cfg.h ファイルに記述します。

周期タイマ割込みの割込みレベルを、2（最高）～7（最低）のいずれかを指定します。

```
#define　IPL_GPT　4
```

周期タイマとして用いる GPT のチャネルを、1～4 のいずれかを指定します。

```
#define　CH_GPT　1
```

トレース機能のタイムスタンプに用いる場合には、周期タイマとして用いた GPT 以外のチャネルを指定します。トレース機能を用いない場合には、定義しません。

```
#define　TRACE_GPT　2
```

GPT のクロックソースの周波数を指定します。

```
#define　AUXCLK　(24*1000000uL)
```

４．９．４　標準 COM ドライバ DDR_TI_DM36xUART.c

_ddr_ti_dm36xuart_init　UART ポートの初期化

【書式】

ER　_ddr_ti_dm36xuart_init(ID devid, volatile struct t_uart *uart_port)

【パラメータ】

ID	devid	デバイスの ID 番号
volatile struct t_uart *	uart_port	UART のデバイスアドレス

【解説】

指定された UART（Universal Asynchronous Receiver/Transmitter）の 1 チャネルを、指定されたデバイス ID の標準 COM ポートとして初期化します。

【プログラム例】

例えば、UART のチャネル 1 を ID_UART1 のデバイス ID 番号で初期化する場合には、次のような記述になります。

```
#include "dm365.h"
extern ER　_ddr_ti_dm36xuart_init(ID, volatile struct t_uart *);
:
_ddr_ti_dm36xuart_init(ID_UART1, &REG_UART1);
```

初期化する内容は、初期化する全チャネルをまとめ、DDR_TI_DM36xUART_cfg.h ファイルに記述します。１つのチャネルは、次の 8 つのパラメータにより構成されます。

```
#define　UART_n                /*チャネル n を使用する*/

#define　CTSRTS_n　1　　/* CTS/RTS ピン有効(1)/無効(0) */
#define　TXBUF_SZn　1024　　/*送信バッファサイズ(0 以上) */
#define　RXBUF_SZn　1024　　/*受信バッファサイズ(1 以上) */
#define　XOFF_SZn　768　　/* XOFF 送出受信バッファデータ数トリガ*/
#define　XON_SZn　128　　/* XON 送出受信バッファデータ数トリガ*/
#define　RXTL_n　8　　/*レシーブ FIFO データ数トリガ(1,4,8,14) */
#define　IPL_UARTn　2　　/*割込みレベル(2～7) */
```

５．６　TMPA910 のドライバ

５．６．１　I/O アドレスと割込み番号の定義

TMPA910 の I/O アドレスと割込み番号の定義ファイルは、ドライバのインクルードフォルダに TMPA910.h を用意しています。

５．６．２　割込みドライバ DDR_TMPA910_VIC0.c

割込み要因毎に割込み番号を持ち、それぞれ独立した割込みハンドラの生成や割込みサービスルーチンの生成が可能になっています。また、本割込みドライバでは多重割込みを受け付けません。

割込みハンドラや割込みサービスルーチンの開始直後は、CPU ロックの解除状態になっています。

_ddr_tmpa910_vic0_init　　割込みコントローラの初期化

【書式】

```
void   _ddr_tmpa910_vic0_init(void)
```

【解説】

割込みコントローラを初期化します。また、割込みコントローラの初期化は、次のように、カーネルの起動前に行います。

【プログラム例】

```
_ddr_tmpa910_vic0_init();
        :
start_uC3(&csys, initpr);
```

dis_int　　割込みの禁止

【書式】

ER ercd = dis_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを禁止します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。

ena_int　　割込みの許可

【書式】

ER ercd = ena_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを許可します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。

５．６．３　周期タイマドライバ DDR_TMPA910_TIMER0.c

_ddr_tmpa910_timer0_init　周期タイマの初期化

【書式】

void　_ddr_tmpa910_timer0_init(UINT tick)

【パラメータ】

UINT	tick	チック時間

【解説】

周期タイマに用いるため、内蔵 Timer の 1 チャネルを初期化します。これの初期化内容を、次の要領で DDR_TMPA910_TIMER0_cfg.h ファイルに記述します。

周期タイマ割込みの割込みレベルを、0（最高）～15（最低）のいずれかを指定します。

#define　IPL_GPT 15

周期タイマとして用いる GPT のチャネルを、0～5 のいずれかを指定します。

#define　CH_GPT　1

TMPA910 への入力クロックソースの周波数を指定します。

#define　CLKIN　(24*1000000uL)

５．６．４　標準 COM ドライバ DDR_TMPA910_UART0.c

_ddr_tmpa910_uart0_init　UART ポートの初期化

【書式】

ER　_ddr_tmpa910_uart_init(ID devid, volatile struct t_uart *uart_port)

【パラメータ】

ID	devid	デバイスの ID 番号
volatile struct t_uart *	uart_port	UART のデバイスアドレス

【解説】

指定された UART（Universal Asynchronous Receiver/Transmitter）の 1 チャネルを、指定されたデバイス ID の標準 COM ポートとして初期化します。

【プログラム例】

例えば、UART のチャネル 0 を ID_UART0 のデバイス ID 番号で初期化する場合には、次のような記述になります。

```
#include "TMPA910.h"
extern ER　_ddr_tmpa910_uart0_init(ID, volatile struct t_uart *);
:
_ddr_tmpa910_uart0_init (ID_UART0, &REG_UART0);
```

初期化する内容は、初期化する全チャネルをまとめ、DDR_TMPA910_UART0_cfg.h ファイルに記述します。チャネル共通では、次のパラメータにより構成されます。

```
#define　UART_n                /* チャネル n を使用する*/

#define　TXBUF_SZn　1024　　/* 送信バッファサイズ(0 以上) */
#define　RXBUF_SZn　1024　　/* 受信バッファサイズ(1 以上) */
#define　XOFF_SZn　768　　/* XOFF 送出受信バッファデータ数トリガ*/
#define　XON_SZn　128　　/* XON 送出受信バッファデータ数トリガ*/
#define　RTRG_n　13　　/* レシーブ FIFO データ数トリガ(3,5,9,13,15) */
#define　TTRG_n　4　　/* トランスミッタ FIFO データ数トリガ(2,4,8,12,14) */
#define　IPL_UARTn　14　　/* 割込みレベル(0～15) */
#define　FIFO_n　1　　/* FIFO 機能有効(1)/無効(0) */
#define　CTS_n　1　　/* CTS ピン有効(1)/無効(0) */
```

チャネル 0 では、さらに 5 つのパラメータがあります。

```
#define   RTS_0         1       /* RTS ピン有効(1)/無効(0) */
#define   DSR_0         1       /* DSR ピン有効(1)/無効(0) */
#define   DTR_0         1       /* DTR ピン有効(1)/無効(0) */
#define   DCD_0         1       /* DCD ピン有効(1)/無効(0) */
#define   RI_0          1       /* RI ピン有効(1)/無効(0) */
```

５．７　OMAP-L137、L138 のドライバ

５．７．１　I/O アドレスと割込み番号の定義

OMAP-L137、L138 の I/O アドレスと割込み番号の定義ファイルは、ドライバのインクルードフォルダに omapl1x.h を用意しています。

５．７．２　割込みドライバ DDR_TI_OMAPL1xAITC.c

割込み要因毎に割込み番号を持ち、それぞれ独立した割込みハンドラの生成や割込みサービスルーチンの生成が可能になっています。また、設定された割込みレベルに従って多重割込みを受け付け、使用できる割り込みレベルは 2（最高）～31（最低）です。
割込みハンドラや割込みサービスルーチンの開始直後は、CPU ロックの解除状態になっています。

_ddr_ti_omapl1xaitc_init　　割込みコントローラの初期化

【書式】

```
void　_ddr_ti_omapl1xaitc_init(void)
```

【解説】

割込みコントローラを初期化します。また、割込みコントローラの初期化は、次のように、カーネルの起動前に行います。

【プログラム例】

```
_ddr_ti_omapl1xaitc_init();
                    :
start_uC3(&csys, initpr);
```

dis_int　　割込みの禁止

【書式】

ER ercd = dis_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを禁止します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。

ena_int　　割込みの許可

【書式】

ER ercd = ena_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを許可します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。

５．７．３　周期タイマドライバ DDR_TI_OMAPL1xGPT.c

_ddr_ti_omapl1xgpt_init　　周期タイマの初期化

【書式】

void　_ddr_ti_omapl1xgpt_init (UINT tick)

【パラメータ】

UINT	tick	チック時間

【解説】

OS で使用する周期タイマ（GPT（General Purpose Timer））を初期化します。本ドライバの設定は下記のマクロをファイルの DDR_TI_OMAPL1xGPT_cfg.h に記述してください。

周期タイマの割込みの割込みレベルを 2（最高）～31（最低）で指定します。

```
#define　IPL_GPT　4
```

周期タイマとして使用する GPT のチャネルを 1～2 で指定します。

```
#define　CH_GPT　1
```

トレース機能のタイムスタンプ用のチャネルを 1～2 で指定します。CH_GPT と同じチャネルを指定することはできません。トレース機能を使用しない場合は定義する必要はありません。

```
#define　TRACE_GPT　2
```

GPT のクロックソースの周波数を指定します。

```
#define　AUXCLK　(24*1000000uL)
```

５．７．４　標準 COM ドライバ DDR_TI_OMAPL1xUART.c

_ddr_ti_omapl1xuart_init　　UART ポートの初期化

【書式】

ER　_ddr_ti_omapl1xuart_init (ID devid, volatile struct t_uart *uart_port)

【パラメータ】

ID	devid	デバイスの ID 番号
volatile struct t_uart *	uart_port	UART のデバイスアドレス

【解説】

指定された UART（Universal Asynchronous Receiver/Transmitter）の 1 チャネルを、指定されたデバイス ID の標準 COM ポートとして初期化します。

【プログラム例】

例えば、UART のチャネル 1 を ID_UART1 のデバイス ID 番号で初期化する場合には、次のような記述になります。

```
#include "omapl1x.h"
extern ER　_ddr_ti_omapl1xuart_init(ID, volatile struct t_uart *);
:
_ddr_ti_omapl1xuart_init(ID_UART1, &REG_UART1);
```

初期化する内容は、初期化する全チャネルをまとめ、DDR_TI_OMAPL1xUART_cfg.h ファイルに記述します。１つのチャネルは、次の 8 つのパラメータにより構成されます。

```
#define　UART_n                  /*チャネル n を使用する*/

#define　CTSRTS_n　　0　　/* CTS/RTS ピン有効(1)/無効(0) 、チャネル 0 のみ有効
                              にできる */
#define　TXBUF_SZn　1024　　/*送信バッファサイズ(0 以上) */
#define　RXBUF_SZn　1024　　/*受信バッファサイズ(1 以上) */
#define　XOFF_SZn　　768　　/* XOFF 送出受信バッファデータ数トリガ*/
#define　XON_SZn　　128　　/* XON 送出受信バッファデータ数トリガ*/
#define　RXTL_n　　8　　/*レシーブ FIFO データ数トリガ(1,4,8,14) */
#define　IPL_UARTn　2　　/*割込みレベル(2～31) */
```

５．８　LPC3200 のドライバ

５．８．１　I/O アドレスと割込み番号の定義

LPC3200 の I/O アドレスと割込み番号の定義ファイルは、ドライバのインクルードフォルダに LPC3200.h を用意しています。

５．８．２　割込みドライバ DDR_LPC3200_INTC.c

割込み要因毎に割込み番号を持ち、それぞれ独立した割込みハンドラの生成や割込みサービスルーチンの生成が可能になっています。また、本割込みドライバでは、ソフトウェアの制御により多重割込みを受け付けます。

割込みハンドラや割込みサービスルーチンの開始直後は、CPU ロックの解除状態になっています。

_ddr_lpc3200_intc_init　　割込みコントローラの初期化

【書式】

```
void　_ddr_lpc3200_intc_init (void)
```

【解説】

割込みコントローラを初期化します。また、割込みコントローラの初期化は、次のように、カーネルの起動前に行います。

割込みレベル数を、1～8 のいずれかを指定します。割込みレベル数を 8 とした場合は、指定できる割込みレベルは 0（最高）～7（最低）です。尚、多重割り込みの制御はソフトウェアにより行うため、レベル数が大きいほどオーバヘッドも大きくなります。

【プログラム例】

```
_ddr_lpc3200_intc_init ();
                              :
start_uC3(&csys, initpr);
```

初期化する内容は、DDR_LPC3200_INTC_cfg_cfg.h ファイルに記述します。

```
#define MAX_LEVEL     8          /*  割込みレベル数  */
```

dis_int　　割込みの禁止

【書式】

ER ercd = dis_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを禁止します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。

ena_int　　割込みの許可

【書式】

ER ercd = ena_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを許可します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。

clr_int	割込み要求のクリア

【書式】

ER ercd = clr_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定されたエッヂトリガの割込み要求をクリアします。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。

５．８．３　周期タイマドライバ DDR_LPC3200_TIMER.c

_ddr_lpc3200_timer_init　　周期タイマの初期化

【書式】

void　_ddr_lpc3200_timer_init (UINT tick)

【パラメータ】

UINT	tick	チック時間

【解説】

周期タイマに用いるため、Standard Timer の 1 チャネルを初期化します。これの初期化内容を、次の要領で DDR_LPC3200_TIMER_cfg.h ファイルに記述します。

メイン発振器の周波数を指定します。

```
#define MAINOSC　　13000000
```

RTC 発振器の周波数を指定します。

```
#define RTCOSC　　32768
```

周期タイマ割込みの割込みレベルを、0（最高）～N（最低）のいずれかを指定します。

```
#define　IPL_TIMER　4
```

周期タイマとして用いる GPT のチャネルを、0～5 のいずれかを指定します。

```
#define　CH_TIMER　1
```

５．８．４　標準 COM ドライバ DDR_LPC3200_UART.c

_ddr_lpc3200_uart_init　　UART ポートの初期化

【書式】

ER　_ddr_lpc3200_uart_init (ID devid, volatile struct t_uart *uart_port)

【パラメータ】

ID	devid	デバイスの ID 番号
volatile struct t_uart *	uart_port	UART のデバイスアドレス

【解説】

指定された Standard UART（Universal Asynchronous Receiver/Transmitter）の 1 チャネルを、指定されたデバイス ID の標準 COM ポートとして初期化します。

RTS/CTS 信号線を用いたハードウェアフロー制御を使えますが、信号線をソフトウェアによって制御するため、RTS 信号の出力が遅れる場合があります。

【プログラム例】

例えば、UART のチャネル 1 を ID_UART1 のデバイス ID 番号で初期化する場合には、次のような記述になります。

```
#include "LPC3200.h"
extern ER　_ddr_lpc3200_uart_init (ID, volatile struct t_uart *);
:
_ddr_lpc3200_uart_init (ID_UART3, &REG_UART3);
```

初期化する内容は、初期化する全チャネルをまとめ、DDR_LPC3200_UART_cfg.h ファイルに記述します。１つのチャネルは、次の 11 つのパラメータにより構成されます。

```
#define　UART_n                  /*チャネル n を使用する*/

#define　TXBUF_SZn　1024　　/*送信バッファサイズ(0 以上) */
#define　RXBUF_SZn　1024　　/*受信バッファサイズ(1 以上) */
#define　XOFF_SZn　768　　/*XOFF 送出受信バッファデータ数トリガ*/
#define　XON_SZn　128　　/*XON 送出受信バッファデータ数トリガ*/
#define　RTSOFF_SZn　960　　/*RTS OFF 受信バッファデータ数トリガ*/
#define　RTSON_SZn　64　　/*RTS ON 受信バッファデータ数トリガ*/
#define　RTRG_n　8　　/*受信 FIFO データ数トリガ(16,32,48,60) */
#define　TTRG_n　0　　/*送信 FIFO データ数トリガ(0 固定) */
#define　IPL_UARTn　2　　/*割込みレベル(2～7) */
#define　FIFO_n　0　　/*レシーブ FIFO データ数トリガ(16,32,48,60) */
```

５．９　DM6446 のドライバ

５．９．１　I/O アドレスと割込み番号の定義

DM6446 の I/O アドレスと割込み番号の定義ファイルは、ドライバのインクルードフォルダに dm644x.h を用意しています。

５．９．２　割込みドライバ DDR_TI_DM644xAITC.c

割込み要因毎に割込み番号を持ち、それぞれ独立した割込みハンドラの生成や割込みサービスルーチンの生成が可能になっています。また、設定された割込みレベルに従って多重割込みを受け付け、使用できる割り込みレベルは 2（最高）～31（最低）です。

割込みハンドラや割込みサービスルーチンの開始直後は、CPU ロックの解除状態になっています。

_ddr_ti_dm644xaitc_init　　割込みコントローラの初期化

【書式】

```
void　_ddr_ti_dm644xaitc_init(void)
```

【解説】

割込みコントローラを初期化します。また、割込みコントローラの初期化は、次のように、カーネルの起動前に行います。

【プログラム例】

```
_ddr_ti_dm644xaitc_init();
        :
start_uC3(&csys, initpr);
```

dis_int　　割込みの禁止

【書式】

ER ercd = dis_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを禁止します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。ただし、割込みハンドラからは呼び出すことはできません。

ena_int　　割込みの許可

【書式】

ER ercd = ena_int(INTNO intno)

【パラメータ】

INTNO	intno	割込み番号

【戻り値】

ER	ercd	正常終了（E_OK）またはエラーコード

【エラーコード】

E_PAR	パラメータエラー

【解説】

intno で指定される割込みを許可します。このシステムコールは、タスクコンテキストだけでなく、非タスクコンテキストからも呼び出すことができます。ただし、割込みハンドラからは呼び出すことはできません。

５．９．３　周期タイマドライバ DDR_TI_DM644xGPT.c

_ddr_ti_dm644xgpt_init　周期タイマの初期化

【書式】

void　_ddr_ti_dm644xgpt_init(UINT tick)

【パラメータ】

UINT	tick	チック時間

【解説】

周期タイマに用いるため、GPT（General Purpose Timer）の 1 チャネルを初期化します。これの初期化内容を、次の要領で DDR_TI_DM644xGPT_cfg.h ファイルに記述します。

周期タイマ割込みの割込みレベルを、2（最高）～31（最低）のいずれかを指定します。

```
#define　IPL_GPT　4
```

周期タイマとして用いる GPT のチャネルを、1～2 のいずれかを指定します。

```
#define　CH_GPT　1
```

トレース機能のタイムスタンプに用いる場合には、周期タイマとして用いた GPT 以外ののチャネルを指定します。トレース機能を用いない場合には、定義しません。

```
#define　TRACE_GPT　2
```

５．９．４　標準 COM ドライバ DDR_TI_DM644xUART.c

_ddr_ti_dm644xuart_init　UART ポートの初期化

【書式】

ER　_ddr_ti_dm644xuart_init(ID devid, volatile struct t_uart *uart_port)

【パラメータ】

ID	devid	デバイスの ID 番号
volatile struct t_uart *	uart_port	UART のデバイスアドレス

【解説】

指定された UART（Universal Asynchronous Receiver/Transmitter）の 1 チャネルを、指定されたデバイス ID の標準 COM ポートとして初期化します。

【プログラム例】

例えば、UART のチャネル 1 を ID_UART1 のデバイス ID 番号で初期化する場合には、次のような記述になります。

```
#include "dm644x.h"
extern ER　_ddr_ti_dm644xuart_init(ID, volatile struct t_uart *);
:
_ddr_ti_dm644xuart_init(ID_UART1, &REG_UART1);
```

初期化する内容は、初期化する全チャネルをまとめ、DDR_TI_DM644xUART_cfg.h ファイルに記述します。１つのチャネルは、次の 8 つのパラメータにより構成されます。

```
#define　UART_n                /*チャネル n を使用する*/

#define　CTSRTS_n　1　　/* CTS/RTS ピン有効(1)/無効(0) */
#define　TXBUF_SZn　1024　　/*送信バッファサイズ(0 以上) */
#define　RXBUF_SZn　1024　　/*受信バッファサイズ(1 以上) */
#define　XOFF_SZn　768　　/* XOFF 送出受信バッファデータ数トリガ*/
#define　XON_SZn　128　　/* XON 送出受信バッファデータ数トリガ*/
#define　RXTL_n　8　　/*レシーブ FIFO データ数トリガ(1,4,8,14) */
#define　IPL_UARTn　2　　/*割込みレベル(2～31) */
```

第６章　プロセッサに依存した注意事項

特にありません。

µC3/Standard ユーザーズガイド プロセッサ依存部 ARM926EJ-S 編

2008年　9月　　初版
2008年 11月　　第２版
2009年　5月　　第３版
2009年 12月　　第４版
2010年 10月　　第５版
2010年 11月　　第６版
2010年 11月　　第７版
2011年　4月　　第８版
2012年　1月　　第９版
2013年　1月9日 第１０版
2015年08月20日 Rev. 2.0

イー・フォース株式会社　　http://www.eforce.co.jp/
TEL 03-5614-6918　FAX 03-5614-6919
お問い合わせ info@eforce.co.jp