

スレッドについて

Ver. 0.1.5 2008-10-16

目次

更新履歴.....	2
1 スレッド情報が格納されている場所.....	3
2 OSThreadInfo スレッドシステム情報	4
3 OSThread スレッド構造体.....	5

更新履歴

2008/10/16 TWL-SDK への収録にあたり、表記の変更

2005/09/27 OSThread 構造体の説明に `alarmForSleep` を追加した。

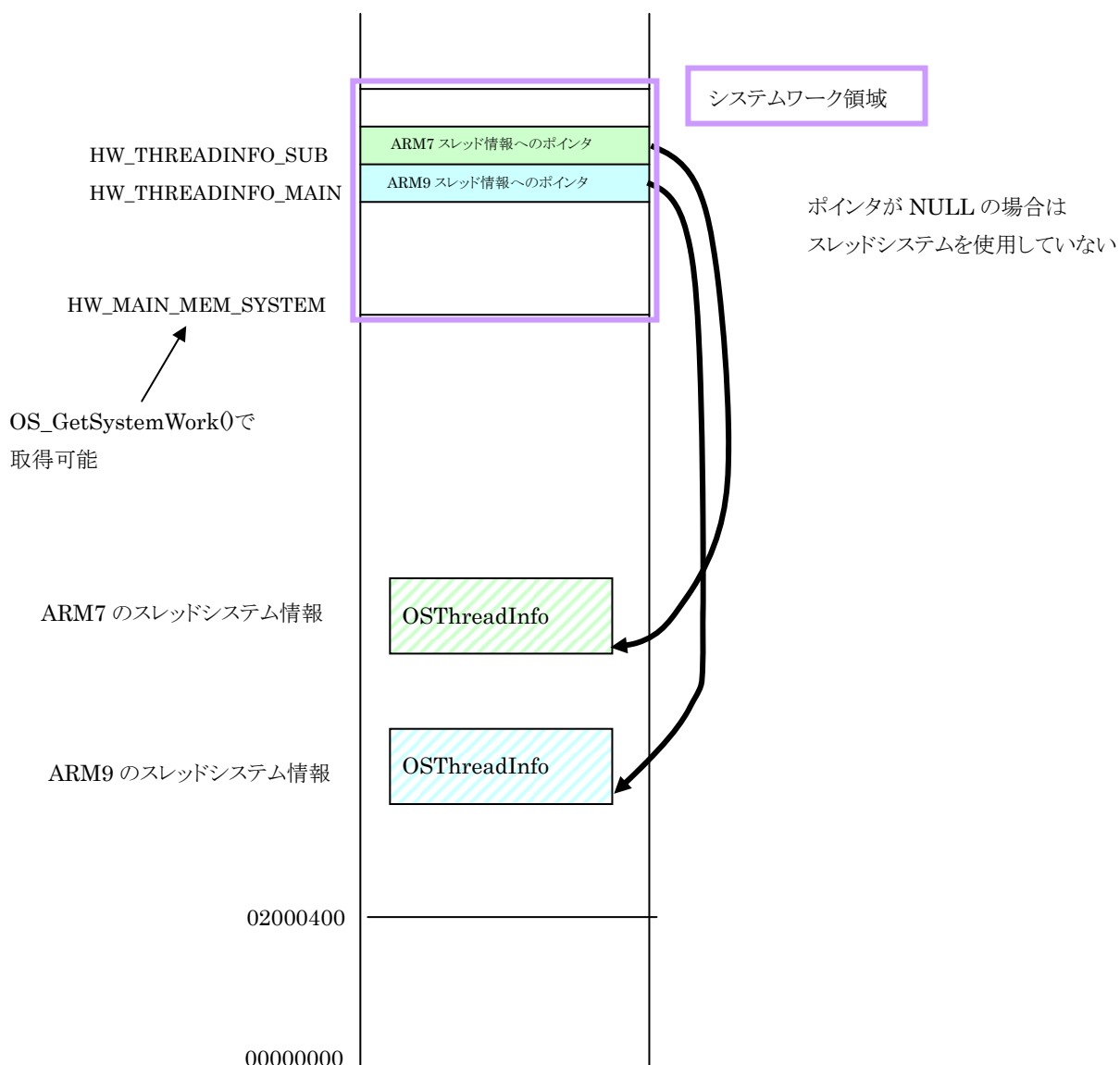
1 スレッド情報が格納されている場所

スレッド情報 `OSThreadInfo` はメインメモリ上に確保されています。その情報が格納されているアドレスを格納した領域が、ARM9/7 の両方からアクセス可能なメインメモリの一部であるシステムワーク領域に存在します。システムワークの開始アドレスは $\text{HW_MAIN_MEM_SYSTEM} = \text{HW_MAIN_MEM} + 0\text{xFFFC00} = 0\text{x2FFFC00}$ です。ユーザのプログラムからはこのアドレスは `OS_GetSystemWork0` として取得出来ます。

ARM9 の場合、 $\text{HW_THREADINFO_MAIN} = \text{HW_MAIN_MEM} + 0\text{x00FFFA0} = 0\text{x2FFFA0}$ にポインタが格納されています。このポインタ格納アドレスは `OS_GetSystemWork0->threadinfo_mainp` としても取得出来ます。

ARM7 の場合、 $\text{HW_THREADINFO_SUB} = \text{HW_MAIN_MEM} + 0\text{x00FFFA4} = 0\text{x2FFFA4}$ にポインタが格納されています。このポインタ格納アドレスは `OS_GetSystemWork0->threadinfo_subp` としても取得出来ます。

ポインタが `NULL` であれば、そのプロセッサではスレッドシステムを使用していないことになります。



2 OSThreadInfo スレッドシステム情報

```
// ----- Thread & context packed structure
typedef struct OSThreadInfo
{
    u16                isNeedRescheduling;
    u16                irqDepth;
    OSThread*          current;
    OSThread*          list;
    void*              switchCallback;
} OSThreadInfo;
```

OSThreadInfo 構造体の各メンバについて説明します。

isNeedRescheduling

IRQ 割り込み時等にスレッドの切り替え要求が発生し IRQ 割り込み終了時にリスケジュールする必要があるかどうかを記憶するためのフラグで TRUE, FALSE の2値をとります。OS 内部で使用される値なので触らないようにしてください。

irqDepth

IRQ の割り込みレベルを記録します。多重割り込みで活用されますが、OS 内部で使用される値なので触らないようにしてください。

current

カレントスレッドのスレッド情報へのポインタです。

list

スレッドリストへのポインタです。優先度の高いものから OSThread の next メンバを使い順番に繋がっています。最後は next=NULL となっています。スレッドが全く登録されていない場合は list は NULL となります。

switchCallback

スレッドを切り替える際のコールバックが格納されます。設定されていない場合は NULL となります。

3 OSThread スレッド構造体

```
// ----- Thread structure
typedef struct _OSThread OSThread;
struct _OSThread
{
    OSContext          context;
    OSThreadState      state;
    OSThread*          next;
    u32                id;
    u32                priority;

    void*              profiler;

    OSThreadQueue*     queue;
    OSThreadLink       link;

    OSMutex*           mutex;
    OSMutexQueue       mutexQueue;

    u32                stackTop;        // for stack overflow
    u32                stackBottom;    // for stack underflow
    u32                stackWarningOffset;

    OSThreadQueue       joinQueue;

    void*              specific[OS_THREAD_SPECIFIC_MAX];
    OSAlarm*           alarmForSleep;
    OSThreadDestructor destructor;
    void*              userParameter;

    int                systemError;
};
```

OSThread 構造体の各メンバについて説明します。

context

スレッドが切り替わっている間、コンテキストを格納しておく場所です。

state

スレッドの状態を表します。OS_THREAD_STATE_WAITING (=0) は停止中、OS_THREAD_STATE_READY (=1) は実行可能を表します。終了したスレッドでは、OS_THREAD_STATE_TERMINATED となります。

next

スレッドリストを構成する際の、次のスレッドへのポインタです。NULL の場合は最後尾となります。

id

スレッド id を表します。0~0x7ffffff の値をとります。この値はスレッドが作成される度に増やされていきます。

priority

スレッドの優先度を表します。0～31 の値をとり、0 が最も優先度の高いスレッドとなります。スレッドリストはこのスレッド優先度の順に並べられています。但し、OS_InitThread0 で作成されるアイドルスレッドは優先度 32 の値を取ります。アイドルスレッドの優先度は後から変更出来ません。

profile

関数コールトレースや関数コスト計測といった profile 機能のルーチンがスレッドごとの情報を保持するために使用するポインタです。profile 機能を使用しないときは何も機能しません。

queue, link

スレッドキューのための領域です。queue はスレッドがスリープする際に指定されたスレッドキューへのポインタが格納されます。link は同じスレッドキューに対してスリープしたスレッド同士をリストで繋ぐためのリンク情報です。

mutex, mutexQueue

スレッド終了時に mutex の解放を自動的に行うためのパラメータです。OS が内部で使用しますので触らないようにしてください。

stackTop, stackBottom, stackWarningOffset

スタック溢れチェックに使用されるパラメータです。OS が内部で使用しますので触らないようにして下さい。参照するのは構いません。

JoinQueue

スレッドが終了したときに sleep している他のスレッドを起こすためのキューです。

specific

システム内部で使います。

alarmForSleep

スレッドがスリープするときのアラームへのポインタです。

destructor

スレッドデストラクタです。スレッドが終了する際に呼ばれる関数を設定します。

userParameter

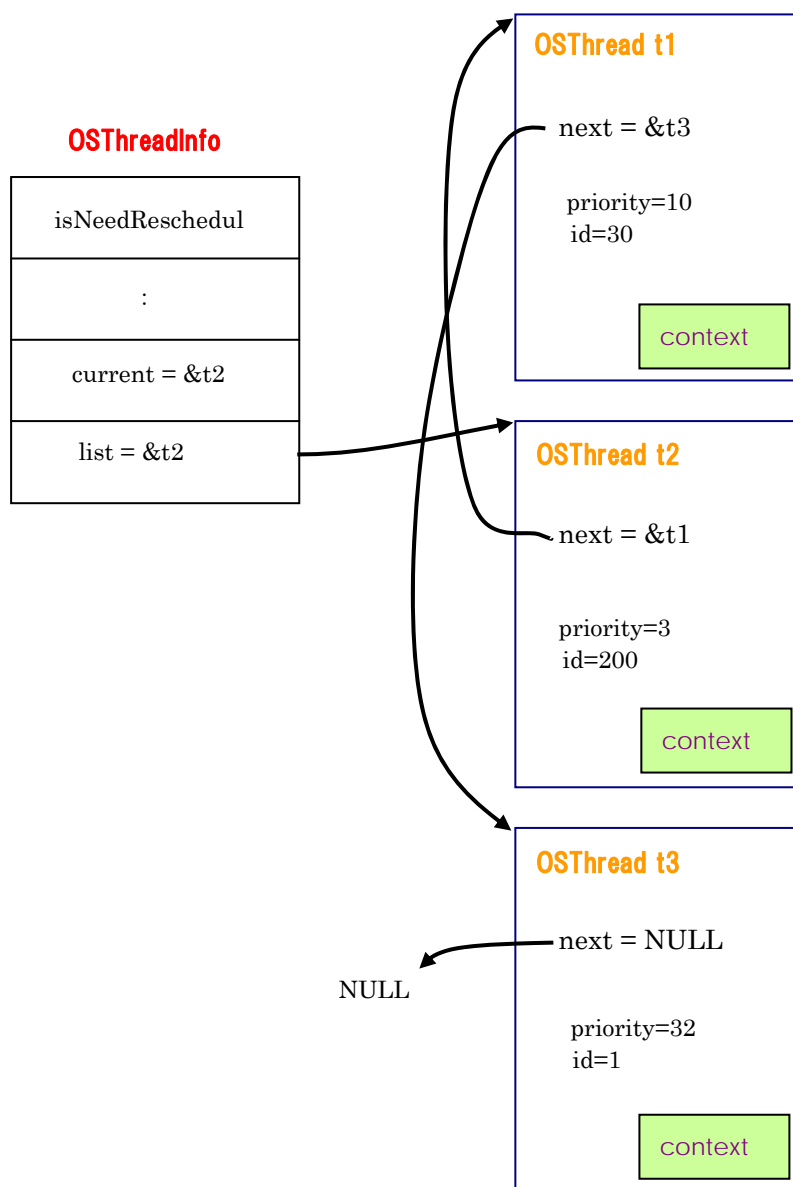
ユーザパラメータです。この領域はユーザが自由に使用することができます。システム側では一切変更しませんし、参照もしません。

systemError

システムエラー値です。システム内部で使います。

スレッド情報の例

以下の例では、スレッド t1, t2, t3 が存在し、t2 がカレントスレッドになっています。



ARM9 ではアイドルスレッド(優先度 32)のスレッドがリストの最後にあるはずですが、(ここでは t3 と書きましたが、OSThread 構造体は os_thread.c 内の OSi_IdleThread です)
なお、ARM7 にはアイドルスレッドがありません。

スレッドキューの例

以下の例では、スレッドキュー tq にスレッド t1, t2, t4 が繋がっています。

