

## 1.2. OS 可読マニュアル

### (1) OS のファイル

本研究 OS のファイルは全部で 77 個となる。(1-1)~(1-3)に順に示す。

#### (1-1) kernel ファイル

kernel ファイルは(1-1-1)~(1-1-3)の 44 個となる。

##### (1-1-1) kernel 主要ファイル

- ・ defines.h

[説明]

型の定義やシステムコールのエラーコード, カーネルオブジェクト資源数の定義や実装しているアルゴリズムの定義をしているヘッダファイルである。

- ・ interrupt.h

[説明]

H8 専用の割込み周りの CPU 依存コードをマクロ定義とソフトウェアベクタ設定関数プロトタイプを定義しているヘッダファイルである。

- ・ interrupt.c

[説明]

ソフトウェアベクタ設定処理を行う C ファイルである。

- ・ intr.h

[説明]

ソフトウェアベクタ番号をマクロ定義しているヘッダファイルである。

- ・ kernel.h

[説明]

kernel のヘインヘッダファイルとなる。各 ISR やカーネルが使用するマクロの定義, タスク情報, ディスパッチャ情報, タスクコンテキスト用システムコールのプロトタイプ, 非タスクコンテキスト用システムコールのプロトタイプ, サービスコールのプロトタイプ, ユーザタスクのプロトタイプ, ISR チェック関数のプロトタイプ, 他の定義をしているヘッダファイルである。

- ・ kernel.c

[説明]

kernel のメイン C ファイルとなる。本論文の 3.3 節の図 3-3 にある ID 変換処理, システムコールハンドラ, サービスコールハンドラ, 例外, 非タスクコンテキスト用システムコール前処理, タスクコンテキスト用システムコール前処理, サービスコール前処理, OS, init タスク処理(kernel 初期化処理), OS ダウン処理, OS フリーズ処理, 待ち要因解除処理, 取得情報自動解放処理を記述している。

- ・ ld.scr

[説明]

kernel メモリマップイメージとメモリセグメントを記述しているリンカスクリプトファイルとなる。

- ・ main.c

[説明]

kernel(OS)のメインファイルとなる。このファイルではカーネルオブジェクト資源数, init タスク, main 関数, シリアル割込みハンドラを記述している。

- ・ memory.h

[説明]

kernel 動的メモリ管理機構の関数プロトタイプを定義しているヘッダファイルである.

- ・ memory.c

[説明]

kernel 動的メモリ管理機構の関数を記述しているCファイルである

- ・ nmi.h

[説明]

NMI 割込みハンドラのプロトタイプを定義しているヘッダファイルである.

- ・ nmi.c

[説明]

NMI 割込みハンドラの記述しているCファイルである.

- ・ printrmutex.h

[説明]

各優先度逆転防止機構群つき ISR 関数のプロトタイプを定義しているヘッダファイルである.

- ・ printrmutex.c

[説明]

各優先度逆転防止機構群つき ISR 関数を記述しているCファイルである.

- ・ ready.h

[説明]

レディー管理情報とレディー管理操作関数のプロトタイプを定義しているヘッダファイルである.

- ・ ready.c

[説明]

レディー管理操作関数を記述しているCファイルである.

- ・ scheduler.h

[説明]

スケジューラ管理情報とスケジューラ関数のプロトタイプを定義しているヘッダファイルである.

- ・ scheduler.c

[説明]

スケジューラを記述しているCファイルである.

- ・ startup.s

[説明]

OSのスタートアップとタスクディスパッチャが記述されているアセンブラファイルである.

- ・ srvcall.c

[説明]

サービスコール本体が記述されているCファイルである.

- ・ syscall.h

[説明]

システムコールテーブルとシステムコール構造体が定義されているヘッダファイルである.

- ・ syscall.c

[説明]

システムコール本体が記述されているCファイルである.

#### (1-1-2) カーネルオブジェクト及び ISR ファイル

- ・ task\_manage.h

[説明]

acre\_tsk(), del\_tsk(), sta\_tsk(), run\_tsk(), ext\_tsk(), exd\_tsk(), ter\_tsk(), get\_pri(), chg\_pri(), get\_id(), chg\_slk(), get\_slk()システムコールのISR関数のプロトタイプが定義されているヘッダファイルである.

- ・ task\_manage.c

[説明]

acre\_tsk(), del\_tsk(), sta\_tsk(), run\_tsk(), ext\_tsk(), exd\_tsk(), ter\_tsk(), get\_pri(), chg\_pri(), get\_id(), chg\_slk(), get\_slk()システムコールのISR関数が記述されているCファイルである.

- ・ task\_sync.h

[説明]

slp\_tsk(), tslp\_tsk(), wup\_tsk(), rel\_wai(), dly\_tsk()システムコールのISR関数のプロトタイプが定義されているヘッダファイルである.

- ・ task\_sync.c

[説明]

slp\_tsk(), tslp\_tsk(), wup\_tsk(), rel\_wai(), dly\_tsk()システムコールのISR関数が記述されているCファイルである.

- ・ semaphore.h

[説明]

semaphore 情報と acre\_sem(), del\_sem(), wai\_sem(), pwai\_sem(), twai\_sem(), sig\_sem()システムコールのISR関数プロトタイプを定義しているヘッダファイルである.

- ・ semaphore.c

[説明]

acre\_sem(), del\_sem(), wai\_sem(), pwai\_sem(), twai\_sem(), sig\_sem()システムコールのISR関数を記述しているCファイルである.

- ・ mailbox.h

[説明]

mailbox 情報と acre\_mbx(), del\_mbx(), rcv\_mbx(), prcv\_mbx(), trcv\_mbx(), snd\_mbx()システムコールのISR関数プロトタイプを定義しているヘッダファイルである.

- ・ mailbox.c

[説明]

acre\_mbx(), del\_mbx(), rcv\_mbx(), prcv\_mbx(), trcv\_mbx(), snd\_mbx()システムコールのISR関数を記述しているCファイルである.

・ mutex.h

[説明]

mutex 情報と acre\_mtx(), del\_mtx(), loc\_mtx(), unl\_mtx(), ploc\_mtx(), tloc\_mtx() システムコールの ISR 関数プロトタイプを定義しているヘッダファイルである.

・ mutex.c

[説明]

acre\_mtx(), del\_mtx(), loc\_mtx(), unl\_mtx(), ploc\_mtx(), tloc\_mtx() システムコールの ISR 関数を記述している C ファイルである.

・ time\_manage.h

[説明]

周期ハンドラ情報及びアラームハンドラ情報と acre\_cyc(), del\_cyc(), sta\_cyc(), stp\_cyc(), acre\_alm(), del\_alm(), sta\_alm(), stp\_alm() システムコールの ISR 関数のプロトタイプが定義されているヘッダファイルである.

・ time\_manage.c

[説明]

acre\_cyc(), del\_cyc(), sta\_cyc(), stp\_cyc(), acre\_alm(), del\_alm(), sta\_alm(), stp\_alm() システムコールの ISR 関数が記述されている C ファイルである.

・ timer\_callrte.h

[説明]

タイマコールバックルーチンのプロトタイプが定義されているヘッダファイルである.

・ timer\_callrte.c

[説明]

タイマコールバックルーチンが記述されている C ファイルである.

・ intr\_manage.h

[説明]

def\_inf() システムコールの ISR 関数プロトタイプを定義しているヘッダファイルである.

・ intr\_manage.c

[説明]

def\_inf() システムコールの ISR 関数を記述している C ファイルである.

・ system\_manage.h

[説明]

システム情報と

rot\_rdq(), get\_id(), dis\_dsp(), ena\_dsp(), sns\_dsp(), set\_pow(), rol\_sys() システムコールの ISR 関数のプロトタイプが定義されているヘッダファイルである.

・ system\_manage.c

[説明]

rot\_rdq(), get\_id(), dis\_dsp(), ena\_dsp(), sns\_dsp(), set\_pow(), rol\_sys() システムコールの ISR 関数が記述されている C ファイルである.

### (1-1-3) ドライバ

・ serial\_driver.h

#### [説明]

デバイス制御関数のプロトタイプが定義されているヘッダファイルである.

・ serial\_driver.c

#### [説明]

デバイス制御関数が記述されている C ファイルである.

・ timer\_driver.h

#### [説明]

ソフトタイマ情報とタイマ制御関数のプロトタイプ, タイマ割込みハンドラのプロトタイプが定義されているヘッダファイルである.

・ timer\_driver.c

#### [説明]

タイマ制御関数とタイマ割込みハンドラが記述されている C ファイルである.

### (1-3) 簡易ユーザタスクライブラリ

簡易ユーザライブラリ構成を下表 1 に示す. これらのユーザタスクは本研究 OS の実装している機能を網羅したものとなり, 実行する事によって, レスポンスを体験できる.

表 1 簡易ユーザタスクライブラリ構成

タスク数	環境下	シナリオ
3	－	タスク管理システムコール郡の使用
3	－	タスク付属同期システムコール郡の使用
2	－	システム管理システムコール郡の使用
1	－	カーネルオブジェクト自動解放機能
3	－	周期ハンドラを使用した周期タスク制御
1	－	周期ハンドラを使用したポーリングチェック
1	－	アラームハンドラ使用
2	－	セマフォを使用した同期
3	－	セマフォを使用したクレジットトラック同期
2	－	セマフォを使用した排他
3	－	セマフォを使用したゼネラル排他
2	－	mutex を使用した完全排他
2	－	mutex を使用したデッドロック
2	－	タイムアウト付き mutex を使用したデッドロックの回避
2	－	ポーリング付き mutex を使用したデッドロックの回避
3	priority ceiling	デッドロックの予防

	protocol	
3	virtual inheritance muextx	デッドロックの予防
3		優先度逆転現象
3	priority inheritance protocol	デッドロックの誘発
3	priority inheritance protocol	ネストブロッキング現象(推移的優先度継承)
3	priority inheritance protocol	連続ブロッキング現象
3		ロックフリープロトコルを使用した優先度逆転防止
3	優先度スケジューリング	μ ITRON 型ラウンドロビンスケジューリング
3	Rate Monotonic	展開スケジューリングミス
3	Deadline Monotonic	展開スケジューリングミス
3	Earliest Deadline First	デッドラインミス(将棋倒し現象防止)
3	Least Laxity First	float time ミス

#### (1-4) 簡易Cライブラリ

Cライブラリは最小限の関数を実装している。Cライブラリはブートローダ側とOS側の両方に備わる実装とした。構成を下表2に示す。

表2 簡易Cライブラリ構成

関数名	機能概要	実装対象
memset	メモリに値をセットする	OSとブートローダ
memcpy	メモリの値をコピーする	OSとブートローダ
memcmp	メモリの値を比較する	OSとブートローダ
strlen	文字列の長さを取得する	OSとブートローダ
strcpy	文字列をコピーする	OSとブートローダ
strcmp	文字列を比較する	OSとブートローダ
strncmp	文字列を制限付きで比較する	OSとブートローダ
putc	文字を表示	OSとブートローダ
getc	文字を取得	OSとブートローダ
puts	文字列を表示	OSとブートローダ
gets	文字列を取得	OSとブートローダ
putxval	値を16進表示	(オリジナル)OSとブートローダ
atoi	数値へ変換	OS

#### (2) 可読性向上ツール類

コード可読によって、doxygen, graphvizがあると便利である。doxygen, graphvizはコード構

造を可視化するツールである.

#### (2-1) doxygen のインストール

参考文献[41]のサイトへ行き, ダウンロードし展開する.

```
.....>%tar xvzf doxygen-1.2.17.src.tar.gz
```

ディレクトリに移る

```
.....>%cd doxygen-1.2.17
```

実行する

```
.....>%./configure --prefix /usr/local
```

make ビルドする

```
.....>%make
```

ドキュメントを make ビルドする

```
.....>%make docs
```

インストールする

```
.....>%sudo make install
```

```
.....>%make install_docs
```

#### (2-2) graphviz のインストール

参考文献[42]のサイトへ行き, ダウンロードし展開する.

```
.....>%tar xvzf graphviz-1.8.9.tar.gz
```

ディレクトリに移る

```
.....>%cd graphviz-1.8.9
```

実行する

```
.....>%./configure
```

make ビルドする

```
.....>%make
```

インストールする

```
.....>%make install
```

#### (3) doxygen と graphviz の実行

GUI 環境で設定すると楽である.

```
.....>%doxywizard
```

GUI 環境の使い方, 設定の仕方は [http://www.doxygen.jp/doxywizard\\_usage.html](http://www.doxygen.jp/doxywizard_usage.html) を参考に  
していただきたい. コード可視化グラフのサンプルを 1.3. に示す.

### 1.3. コード可視化グラフのサンプル

トップページを図 1 に示す.



図1 トップページ画面

データ構造を選択すると，本研究 OS で使用されている全データ構造が参照できる．図 2 に示す．



図2 全データ構造表示画面

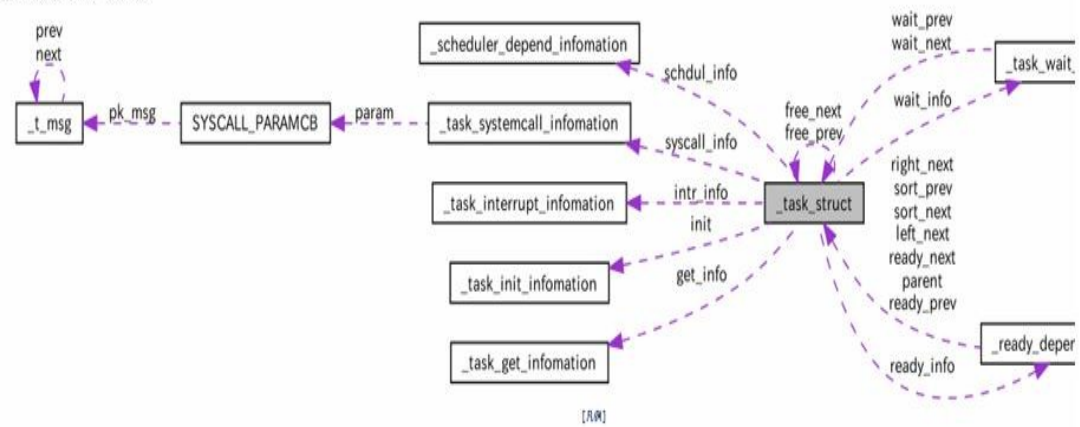
図 2 にあるデータ構造を選択すると，コラボレーション図や各コメントを参照できる．task\_struct を例に図 3 に示す．



# 構造体 \_task\_struct

```
#include <kernel.h>
```

\_task\_structのコラボレーション図



変数

struct _task_struct *	free_next
struct _task_struct *	free_prev
READY_DEP_INFOCB *	ready_info
int	priority
char *	stack
UINT16	state
TSK_INITCB	init
TSK_WAIT_INFOCB	wait_info
TSK_GET_INFOCB	get_info
TSK_INTR_INFOCB	intr_info
TSK_SYSCALL_INFOCB	syscall_info
SCHDUL_DEP_INFOCB *	schedul_info

説明

タスクコントロールブロック(TCB) free listにREADY\_DEP\_INFOCBのポインタを使用するとごちゃごちゃになるので、別途free list専用メンバを追加

## 図3 task\_structデータ構造コラボレーション関係

データ構造は検索できる。検索画面を図4に示す。

メインページ	データ構造	ファイル
データ構造	データ構造索引	データフィールド
データ構造索引		
A   H   R   S   _		
<b>A</b>	<a href="#">scheduler_infomation</a>	<a href="#">_dispatcher_infomation</a>
<a href="#">access</a>	<a href="#">SYSCALL_PARAMCB</a>	<a href="#">_mailbox_infomation</a>
<b>H</b>	<a href="#">system_infomation</a>	<a href="#">_mailbox_struct</a>
<a href="#">h8_3069f_sci</a>	<a href="#">_</a>	<a href="#">_mem_block</a>
<a href="#">h8_3069f_tmr</a>	<a href="#">_alarm_handler_infomation</a>	<a href="#">_mem_pool</a>
<b>R</b>	<a href="#">_alarm_struct</a>	<a href="#">_mutex_infomation</a>
<a href="#">RDYCB</a>	<a href="#">_binary_tree_ready_infomation</a>	<a href="#">_mutex_protocol_infomation</a>
<b>S</b>	<a href="#">_cycle_handler_infomation</a>	<a href="#">_mutex_struct</a>
<a href="#">SCHDULCB</a>	<a href="#">_cycle_struct</a>	<a href="#">_priority_ready_queue_infomation</a>
A   H   R   S   _		

図4 データ構造検索画面

また、データフィールドから変数や構造体、共用体をアルファベット順で確認できる。図5に示す。

メインページ	データ構造	ファイル
データ構造	データ構造索引	データフィールド
全て	変数	
a	b	c
d	e	f
g	h	i
k	l	m
n	o	p
q	r	s
<b>t</b>	u	v
w		
これはフィールドの一覧でそれぞれが属している構造体/共用体の説明へリンクしています。		
- t -		
<ul style="list-style-type: none"> <li>tail : <a href="#">_ready_queue_infomation</a></li> <li>tcnt : <a href="#">h8_3069f_tmr</a></li> <li>tcora0 : <a href="#">h8_3069f_tmr</a></li> <li>tcora1 : <a href="#">h8_3069f_tmr</a></li> <li>tcorb0 : <a href="#">h8_3069f_tmr</a></li> <li>tcorb1 : <a href="#">h8_3069f_tmr</a></li> <li>tcro : <a href="#">h8_3069f_tmr</a></li> <li>tcrl : <a href="#">h8_3069f_tmr</a></li> <li>tcsr0 : <a href="#">h8_3069f_tmr</a></li> <li>tcsr1 : <a href="#">h8_3069f_tmr</a></li> <li>tdr : <a href="#">h8_3069f_sci</a></li> <li>ter_tsk : <a href="#">SYSCALL_PARAMCB</a></li> <li>tlcc_mtx : <a href="#">SYSCALL_PARAMCB</a></li> <li>tm_slice : <a href="#">_scheduler_depend_infomation</a></li> <li>tmout : <a href="#">SYSCALL_PARAMCB</a> , <a href="#">SCHDULCB</a></li> <li>tmout_pri : <a href="#">RDYCB</a></li> <li>tmrhead : <a href="#">_timer_queue</a></li> <li>tobjp : <a href="#">_task_wait_infomation</a> , <a href="#">_scheduler_depend_infomation</a> , <a href="#">SCHDULCB</a> , <a href="#">_cycle_struct</a> , <a href="#">_al</a></li> <li>trcv_mbx : <a href="#">SYSCALL_PARAMCB</a></li> <li>tree_head : <a href="#">_binary_tree_ready_infomation</a></li> <li>tsk_num : <a href="#">_binary_tree_ready_infomation</a></li> <li>tskatr : <a href="#">_task_init_infomation</a></li> <li>tskid : <a href="#">_task_init_infomation</a> , <a href="#">SYSCALL_PARAMCB</a></li> <li>tskpri : <a href="#">SYSCALL_PARAMCB</a></li> <li>tslp_tsk : <a href="#">SYSCALL_PARAMCB</a></li> <li>twai_sem : <a href="#">SYSCALL_PARAMCB</a></li> <li>type : <a href="#">SYSCALL_PARAMCB</a> , <a href="#">_task_systemcall_infomation</a> , <a href="#">_mailbox_struct</a> , <a href="#">_semaphore_struc</a></li> <li><a href="#">SYSCALL_PARAMCB</a> , <a href="#">scheduler_infomation</a></li> </ul>		

図5 データフィールド表示画面

本研究 OS のファイルの一覧を見ることができる。図6に示す。

メインページ	データ構造	ファイル
ファイル一覧	グローバル	

### ファイル一覧

これはファイル一覧です。

os/src/os6.4/defines.h [コード]	
os/src/os6.4/interrupt.c [コード]	
os/src/os6.4/interrupt.h [コード]	
os/src/os6.4/intr.h [コード]	
os/src/os6.4/intr_manage.c [コード]	
os/src/os6.4/intr_manage.h [コード]	
os/src/os6.4/kernel.c [コード]	
os/src/os6.4/kernel.h [コード]	
os/src/os6.4/lib.c [コード]	
os/src/os6.4/lib.h [コード]	
os/src/os6.4/mailbox.c [コード]	
os/src/os6.4/mailbox.h [コード]	
os/src/os6.4/main.c [コード]	
os/src/os6.4/memory.c [コード]	
os/src/os6.4/memory.h [コード]	
os/src/os6.4/mutex.c [コード]	
os/src/os6.4/mutex.h [コード]	
os/src/os6.4/nmi.c [コード]	
os/src/os6.4/nmi.h [コード]	
os/src/os6.4/prinvermutex.c [コード]	
os/src/os6.4/prinvermutex.h [コード]	
os/src/os6.4/ready.c [コード]	
os/src/os6.4/ready.h [コード]	
os/src/os6.4/scheduler.c [コード]	
os/src/os6.4/scheduler.h [コード]	
os/src/os6.4/semaphore.c [コード]	
os/src/os6.4/semaphore.h [コード]	
os/src/os6.4/serial_driver.c [コード]	

図6 ファイル一覧画面

図6に画面からヘッダファイルのインクルード依存関係，そのファイルで使用されているデータ構造のコメントを参照できる。