

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 91 個となる．(1-1)～(1-4)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 初期化处理)，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.

- prinvermutex.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.

- prinvermutex.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(), s
tp_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.

(1-2) ブートローダ

ブートローダファイルは14個となる.

- defines.h

[説明]

ブートローダで使用する型やマクロを定義しているヘッダファイルである.

- dram.h

[説明]

RAM周りの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.

- dram.c

[説明]

RAM周りの関数を記述しているCファイルである.

- elf.h

[説明]

elfファイル解析周りの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.

- elf.c

[説明]

elfファイル解析周りの関数を記述しているCファイルである.

- interrupt.h

[説明]

ブートローダ側で使用する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]

ブートローダ側で使用する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]

ブートローダ側で使用する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.

- intr.S

[説明]

割込み出入り口処理が記述されているアセンブラファイルである.

- ld.scr

[説明]

bootloader 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]

ブートローダの main ファイルとなる。ブートローダ初期化関数，dump コマンド処理関数，main 関数等が記述されている。

- serial_driver. h

[説明]

ブートローダの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]

ブートローダのデバイス制御関数が記述されている C ファイルである。

- startup. s

[説明]

ブートローダのスタートアップが記述されているアセンブラファイルである。

- vector. c

[説明]

ベクタテーブルを記述している C ファイルである。

- xmodem. h

[説明]

xmodem 周りの処理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。

- xmodem. c

[説明]

xmodem 周りの処理関数(主に受信処理)とマクロが記述されている 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 protocol	デッドロックの予防
3	virtual inheritance mutex	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 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．

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

データ構造の説明です。

_alarm_handler_infomation	
_alarm_struct	
_binary_tree_ready_infomation	
_cycle_handler_infomation	
_cycle_struct	
_dispatcher_infomation	
_mailbox_infomation	
_mailbox_struct	
_mem_block	
_mem_pool	
_mutex_infomation	
_mutex_protocol_infomation	
_mutex_struct	
_priority_ready_queue_infomation	
_ready_depend_infomation	
_ready_queue_infomation	
_readyque_infomation	
_scheduler_depend_infomation	
_semaphore_infomation	
_semaphore_struct	
_t_msg	
task_get_infomation	

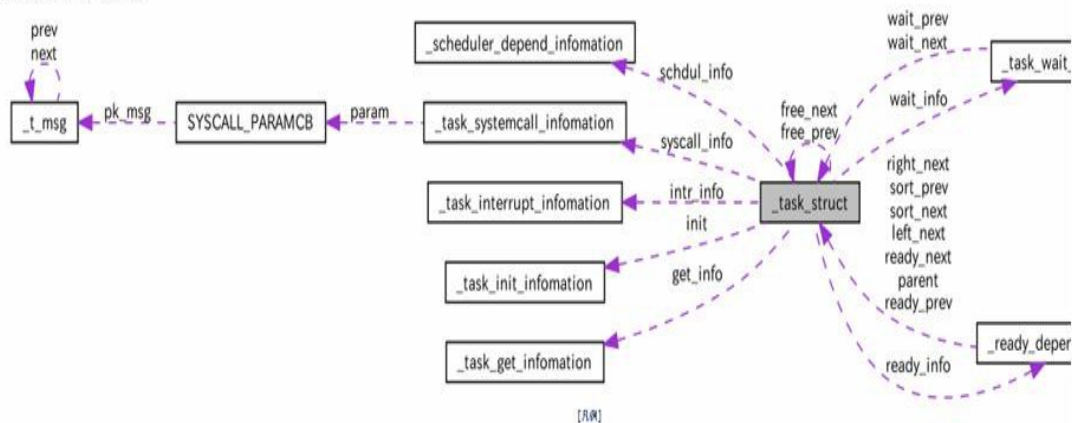
図2 全データ構造表示画面

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専用メモリを追加

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 _		
A	scheduler_infomation	_dispatcher_infomation
access	SYSCALL_PARAMCB	_mailbox_infomation
H	system_infomation	_mailbox_struct
h8_3069f_sci	_	_mem_block
h8_3069f_tmr	_alarm_handler_infomation	_mem_pool
R	_alarm_struct	_mutex_infomation
RDYCB	_binary_tree_ready_infomation	_mutex_protocol_infomation
S	_cycle_handler_infomation	_mutex_struct
SCHDULCB	_cycle_struct	_priority_ready_queue_infomation
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	実数	
a	b	c
d	e	f
g	h	i
k	l	m
n	o	p
q	r	s
t	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"> tail : _ready_queue_infomation tcnt : h8_3069f_tmr tcora0 : h8_3069f_tmr tcora1 : h8_3069f_tmr tcorb0 : h8_3069f_tmr tcorb1 : h8_3069f_tmr tcr0 : h8_3069f_tmr tcr1 : h8_3069f_tmr tcsr0 : h8_3069f_tmr tcsr1 : h8_3069f_tmr tdr : h8_3069f_sci ter_tsk : SYSCALL_PARAMCB tloc_mtx : SYSCALL_PARAMCB tm_slice : _scheduler_depend_infomation tmout : SYSCALL_PARAMCB , SCHDULCB tmout_pri : RDYCB tmrhead : _timer_queue tobjp : _task_wait_infomation , _scheduler_depend_infomation , SCHDULCB , _cycle_struct , _al trcv_mbx : SYSCALL_PARAMCB tree_head : _binary_tree_ready_infomation tsk_num : _binary_tree_ready_infomation tskatr : _task_init_infomation tskid : _task_init_infomation , SYSCALL_PARAMCB tskpri : SYSCALL_PARAMCB tslp_tsk : SYSCALL_PARAMCB twai_sem : SYSCALL_PARAMCB type : SYSCALL_PARAMCB , _task_systemcall_infomation , _mailbox_struct , _semaphore_struct SYSCALL_PARAMCB , scheduler_infomation 		

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/special_defines.h [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．semaphore.h を例に図 7 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

os/src/os6.4/semaphore.h

```
#include "defines.h"
```

semaphore.hのインクルード依存関係図

```

graph TD
    A[os/src/os6.4/semaphore.h] --> B[defines.h]
    B --> C[lib.h]
    D[os/src/os6.4/kernel.c] --> E[os/src/os6.4/semaphore.h]
    F[os/src/os6.4/semaphore.c] --> E
    G[os/src/os6.4/srvcall.c] --> E
    H[os/src/os6.4/system_manage.c] --> E
    I[os/] --> E
  
```

このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。

[ソースコードを見る。](#)

データ構造

```

struct _semaphore_struct
struct _semaphore_infomation

```

型定義

```
typedef struct _semaphore_struct SEMCB
```

関数

```

ER sem_init(void)
void nut sem_wait(sk (SEMCB *sch)

```

図7 semaphore.hのインクルード依存関係

図 6 の画面から C ファイルのインクルード依存関係を参照できる． semaphore.c を例に図 8 に示す．

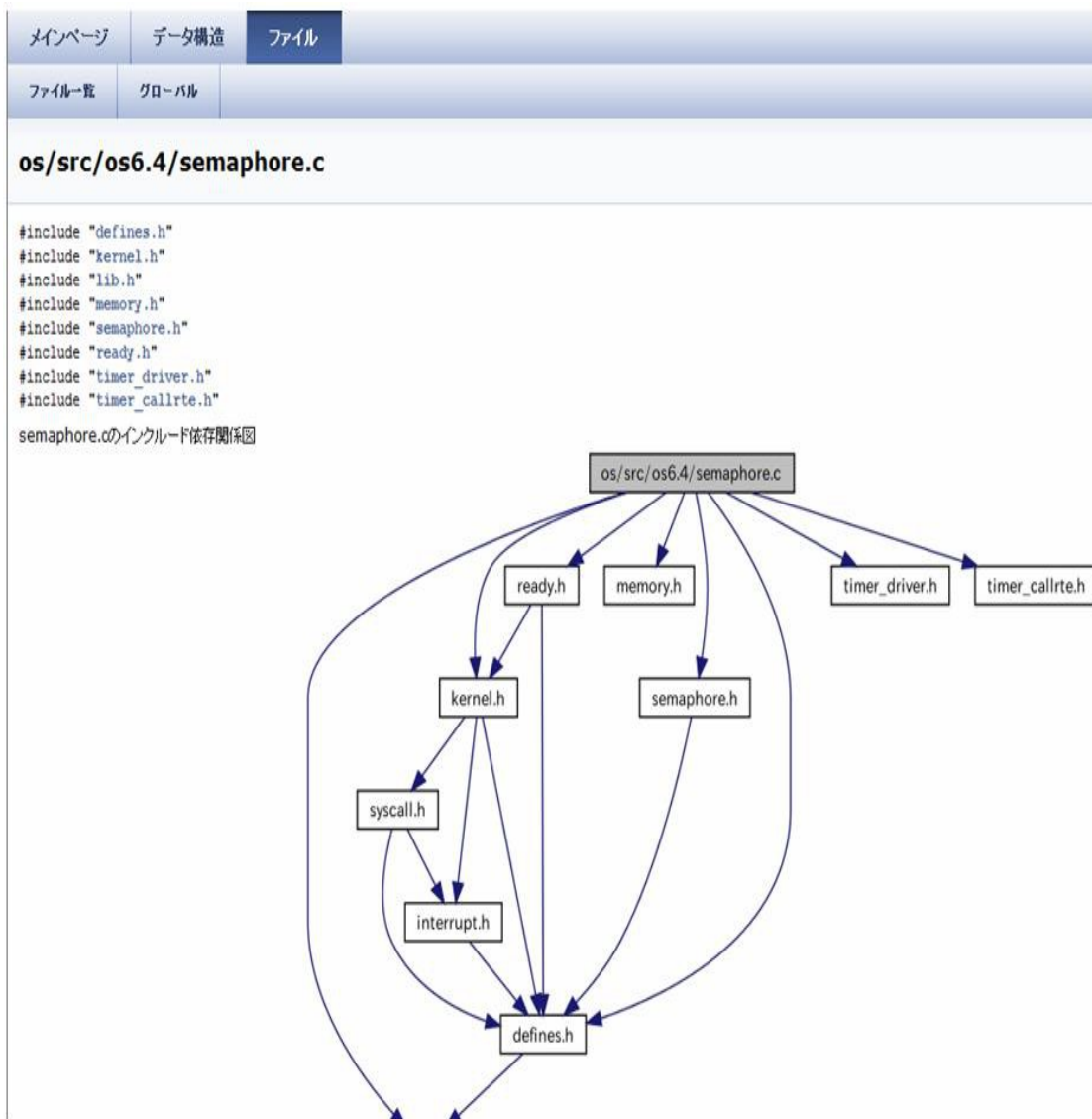


図8 semaphore.cのインクルード依存関係

また、図6の画面のコードを選択すると、コードが参照できる。

ファイルのグローバルから関数、変数、型定義、列挙型、列挙型の値、マクロ定義がアルファベット順で参照できる。図9に関数一覧、図10に列挙型一覧、図11にマクロ定義一覧を示す。

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						
全て	関数	変数	型定義	列挙型	列挙型の値	マクロ定義	
a	c	d	e	f	g	i	k
						m	n
							p
							r
							s
							t
							u
							w
- m -							
<ul style="list-style-type: none"> main() : main.c mbx_init() : mailbox.c , mailbox.h mem_init() : memory.c , memory.h memcmp() : lib.c , lib.h memcpy() : lib.c , lib.h memset() : lib.c , lib.h mtx_init() : mutex.c , mutex.h mtx_priver_init() : mutex.c , mutex.h mv_acre_alm() : kernel.h , srvcall.c mv_acre_cyc() : kernel.h , srvcall.c mv_acre_mbx() : kernel.h , srvcall.c mv_acre_mtx() : kernel.h , srvcall.c mv_acre_sem() : kernel.h , srvcall.c mv_acre_tsk() : kernel.h , srvcall.c mv_chg_slk() : kernel.h , srvcall.c mv_def_inh() : kernel.h , srvcall.c mv_del_alm() : kernel.h , srvcall.c mv_del_cyc() : kernel.h , srvcall.c mv_del_mbx() : kernel.h , srvcall.c mv_del_mtx() : kernel.h , srvcall.c mv_del_sem() : kernel.h , srvcall.c mv_del_tsk() : kernel.h , srvcall.c mv_dis_dsp() : kernel.h , srvcall.c mv_ena_dsp() : kernel.h , srvcall.c mv_get_pri() : kernel.h , srvcall.c mv_get_slk() : kernel.h , srvcall.c mv_get_tid() : kernel.h , srvcall.c mv_rol_sys() : kernel.h , srvcall.c mv_rot_rdq() : kernel.h , srvcall.c mv_sel_schedul() : kernel.h , srvcall.c mv_set_pow() : kernel.h , srvcall.c mv_sns_dsp() : kernel.h , srvcall.c mv_sta_alm() : kernel.h , srvcall.c mv_sta_cyc() : kernel.h , srvcall.c mv_stp_alm() : kernel.h , srvcall.c mv_stp_cyc() : kernel.h , srvcall.c mv_ter_tsk() : kernel.h , srvcall.c mz_acre_alm() : kernel.h , syscall.c mz_acre_cyc() : kernel.h , syscall.c 							

図9 関数一覧画面(アルファベット順)

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						
全て	関数	変数	型定義	列挙型	列挙型の値	マクロ定義	
<ul style="list-style-type: none"> ALM_TYPE : defines.h BTREE_ATR : ready.h CYC_TYPE : defines.h INTR_TYPE : defines.h ISR_TYPE : syscall.h MBX_ATR : defines.h MBX_TYPE : defines.h MSG_ATR : defines.h MTX_ATR : defines.h MTX_TYPE : defines.h PIVER_TYPE : defines.h READY_TYPE : defines.h ROL_ATR : defines.h SCHDUL_TYPE : <u>defines.h</u> SEM_ATR : defines.h SEM_TYPE : defines.h SYSCALL_TYPE : defines.h TSK_TYPE : defines.h 							

図10 列挙型一覧画面(アルファベット順)

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																							
全て				関数				変数				型定義				列挙型				列挙型の値				マクロ定義			
-	a	b	c	d	e	f	h	i	k	l	m	n	o	p	r	s	t	w									

- _ -	<ul style="list-style-type: none">• _INTR_MNAGE_H_ : intr_manage.h• _SYSTEM_MNAGE_H_ : system_manage.h
- a -	<ul style="list-style-type: none">• ACTIV_READY : ready.h• ALARM_ID_NUM : defines.h
- b -	<ul style="list-style-type: none">• BIT_SERCH_NUNBER : ready.h
- c -	<ul style="list-style-type: none">• CHECK_CCR : interrupt.h• CYCLE_ID_NUM : defines.h
- d -	<ul style="list-style-type: none">• DEBUG : defines.h• DEBUG_OUTMSG : defines.h• DEBUG_OUTVLE : defines.h• DYNAMIC : defines.h
- e -	<ul style="list-style-type: none">• E_CTX : defines.h• E_ID : defines.h• E_ILUSE : defines.h• E_NG : defines.h• E_NOEXS : defines.h• E_NOID : defines.h• E_NOMEM : defines.h• E_NOSPT : defines.h• E_OBJ : defines.h• E_OK : defines.h• E_PAR : defines.h• E_QOVR : defines.h• E_RLWAI : defines.h• E_TMOUT : defines.h• EV_NDL : defines.h

図11 マクロ定義一覧画面(アルファベット順)

なお、関数一覧の各関数を選択すると、関数呼び出しグラフとコメントが参照できる。
rel_mpf_isr()を例に図12に示す。

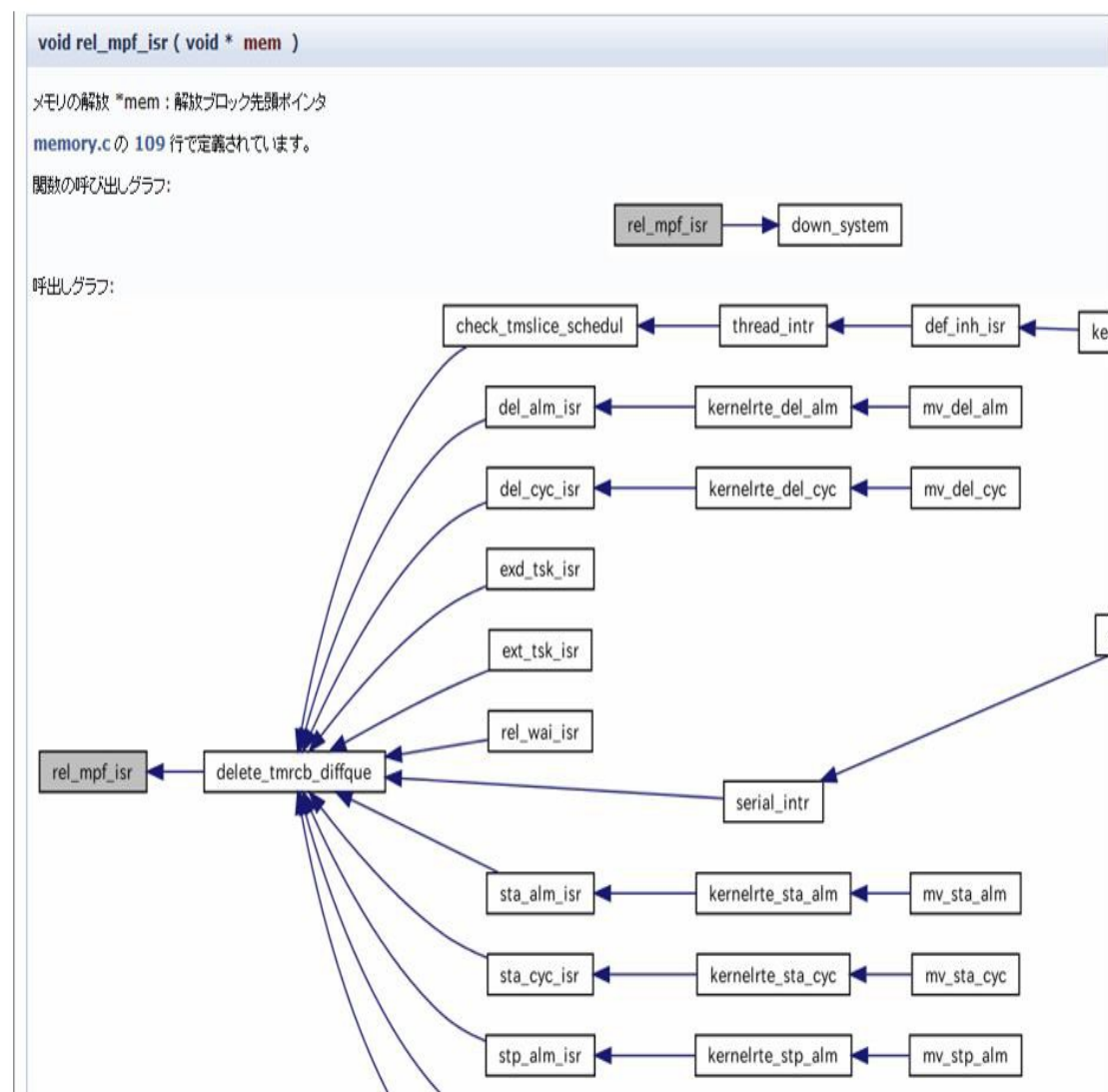


図12 rel_mpf_isr()関数呼び出しグラフ

列挙型一覧の各列挙型を選択すると，列挙型の値とコメントが参照できる．
SCHDUL_TYPE を例に図 13 に示す．

enum SCHEDUL_TYPE

スケジューリングの種類

列挙型の値:

<i>FCFS_SCHEDULING</i>	
<i>RR_SCHEDULING</i>	FCFSスケジューリング ~既存の方式~
<i>ITRON_RR_SCHEDULING</i>	ラウンドロビンスケジューリング ~既存の方式~
<i>PRI_SCHEDULING</i>	ITRON型ラウンドロビンスケジューリング(toppersカーネル参考) ~既存の方式~
<i>RR_PRI_SCHEDULING</i>	優先度スケジューリング ~既存の方式~
<i>MFQ_SCHEDULING</i>	ラウンドロビン×優先度スケジューリング ~既存の方式~
<i>ODRONE_SCHEDULING</i>	Multilevel Feedback Queue(BSDスケジューラ参考) ~既存の方式~
<i>FR_SCHEDULING</i>	簡易O(1)スケジューリング(Linuxカーネル2.6.10参考) ~既存の方式~
<i>PFR_SCHEDULING</i>	公平配分スケジューリング(Linuxカーネル2.6.23参考(赤黒木ではない ver.)) ~既存の方式~
<i>RM_SCHEDULING</i>	優先度公平配分スケジューリング ~新規の方式~
<i>DM_SCHEDULING</i>	Rate Monotonicスケジューリング ~既存の方式~
<i>EDF_SCHEDULING</i>	Deadline Monotonicスケジューリング ~既存の方式~
<i>LLF_SCHEDULING</i>	Earliest Deadline Firstスケジューリング ~既存の方式~

`defines.h` の 131 行で定義されています。

図13 SCHEDUL_TYPEの値とコメント