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.

・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_slt(), get_slt()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_slt(), get_slt()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 muetx	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	μITRON 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．

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．

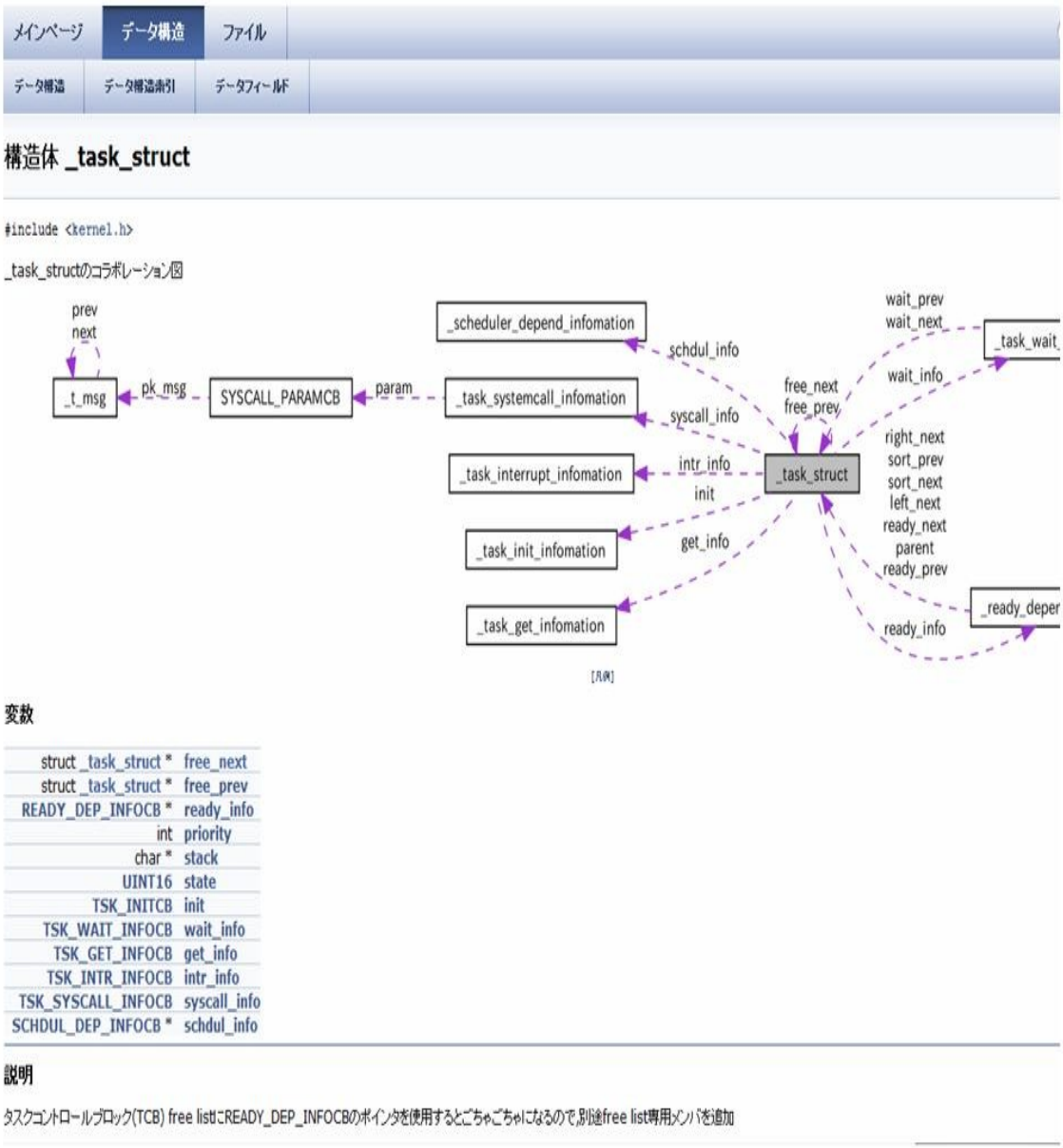

図3 task_structデータ構造コラボレーション関係

データ構造は検索できる．検索画面を図４に示す．

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．

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．

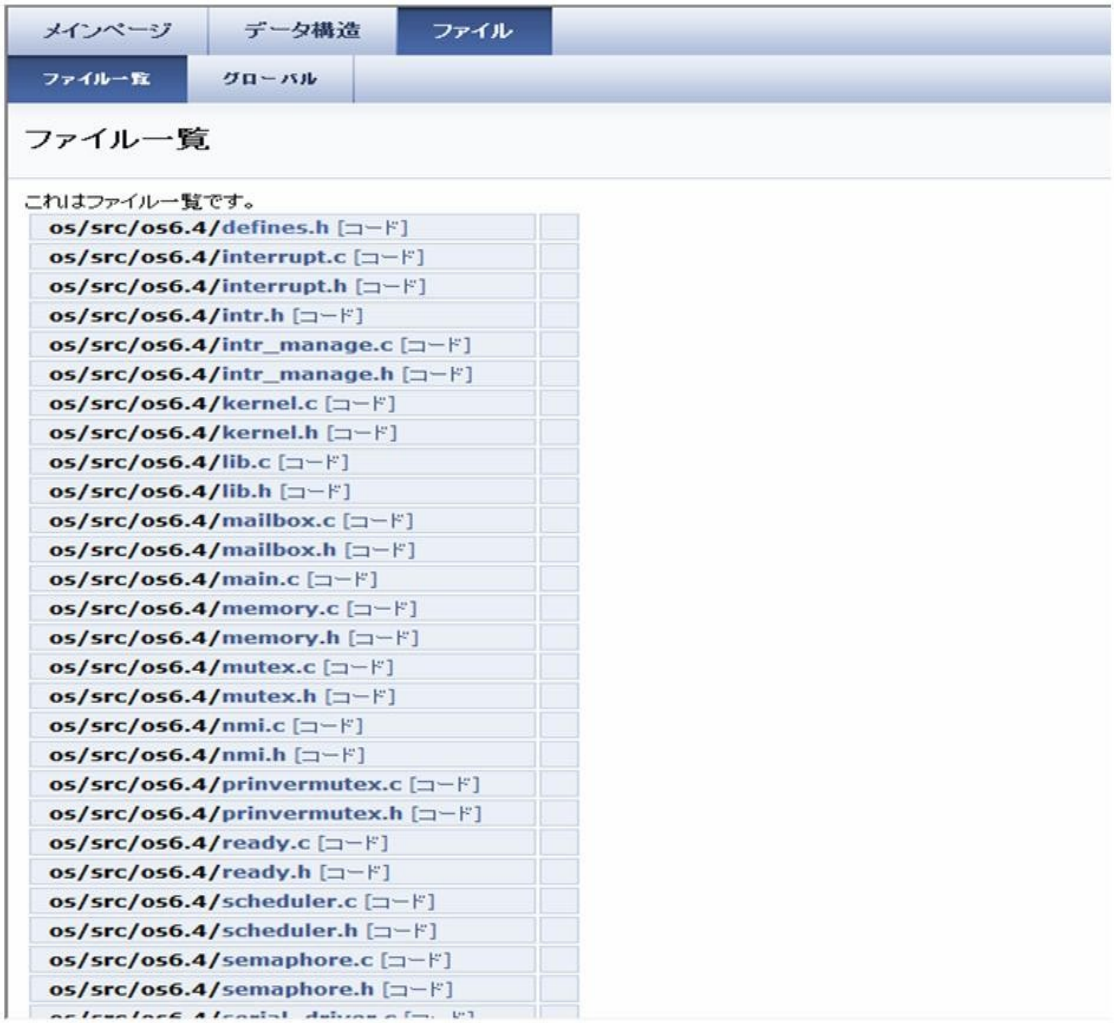

図6 ファイル一覧画面

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．