

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付	'09/12/20 T.S.
		改定日付	'09/01/10 T.S.

組込みソフトウェア科
第4システム
応用課題

スロットマシンを作ろう!!

簡易設計書

第1.5版

平成22年1月10日

組込みソフトウェア科(一般)

学籍番号	氏名
8番	佐藤 敏充

改定履歴

No.	版数	摘要	日付	改定者
1	1	新規作成	'09/12/20	T.S.
2	1.5	実装後追記	'09/01/10	T.S.
3				
4				
5				
6				
7				
8				
9				
10				

目次

スロットマシンを作ろう!!	1
改定履歴	1
目次	2
1 本書について	4
2 訓練内容について	4
3 習得内容測定課題「簡易スロットマシン」について	4
3-1 機能概要	4
3-2 前提条件	4
3-3 制約事項	4
3-4 システムのハードウェア、ソフトウェア構成	5
3-5 開発工数	6
3-6 開発実績	6
3-7 本システムのアピールポイント	6
3-8 プロジェクトを進める上での課題・留意事項	6
4 外部設計	7
5 システムユースケース(略式)	9
5-1 ユースケース全体像	9
5-2 個別ユースケース	10
5-2-1 UC001「スロットを回し、停める。」	10
5-2-2 UC002「3つのリールの目が揃った場合。」	11
5-2-3 UC010「一般状態」	11
5-2-4 UC011「一般アタリ判定」	11
5-2-5 UC012「一般回転方法」	12
5-2-6 UC013「一般演出方法」	12
5-2-7 UC014「一般アタリ表示」	12
5-2-8 UC020「確変状態」	13
5-2-9 UC021「確変アタリ判定」	13
5-2-10 UC022「確変回転方法」	13
5-2-11 UC023「確変演出方法」	14
5-2-12 UC024「確変アタリ表示」	14
5-2-13 UC100「プログラムを実行する。」	14
5-2-14 UC110「プログラムをデーモンとして起動する。」	15
5-2-15 UC120「プログラムをコントローラとして起動する。」	16
5-2-16 UC121「デーモンの生死にかかる優先度の高い命令がバースされた場合」	17
5-2-17 UC131「設定をコマンドラインオプションで変える。」	17
5-2-18 UC132「設定をメニューから変える」	18

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

6 状態図	20
7 处理シーケンス	21
7-1 システム起動とスレッドプール	21
7-2 待ち受け中 (ENTRY)	22
7-3 待ち受け中 (稼動)	23
7-4 待ち受け中 (EXIT)	23
7-5 待ち受け中からベット中への遷移..	25
7-6 ベット中 (ENTRY)	26
7-7 ベット中 (稼動)	27
7-8 ベット中 (EXIT) から回転中 3への遷移.....	28
7-9 回転中 3 (ENTRY)	29
7-10 回転中 3 (稼動)	29
7-11 回転中 3 (EXIT) から回転中 2への遷移	30
7-12 回転中 2 (ENTRY)、(稼動)、(EXIT) および遷移.....	30
7-13 回転中 1 (ENTRY)、(稼動)、(EXIT) および遷移.....	30
7-14 全リール停止 (ENTRY)、(稼動)、(EXIT) および遷移	30
7-15 アタリ保留中 (ENTRY)、(稼動)、(EXIT) および遷移	30
7-16 リーチ中 (ENTRY)、(稼動)、(EXIT) および遷移.....	31
7-17 アタリ動作中 (ENTRY)、(稼動)、(EXIT) および遷移	31
8 クラス間関連.....	31
8-1 スロットマシンゲーム.....	32
8-2 スロットマシンゲーム状態.....	33
9 スタブ実装環境.....	34
10 所感	36
11 卷末付録	37
11-1 第4システム・カリキュラム詳細.....	37
11-1-1 組込み Linux キャラクタデバイス開発	37
11-1-2 組込み Linux システムコール開発	39
11-2 参考文献	42

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

1 本書について

本書は雇用・能力開発機構大阪センター（通称ポリテクセンター関西）における訓練成果を示すための資料である。

また本書は、組込みソフトウェア科一般 6 ヶ月コース（以下本コース）の第4システム、組込み Linux 開発（以下本訓練）における習得度を測るための課題への成果物である。本書の内容はその訓練実績を簡易設計書として起こし説明するものである。

2 訓練内容について

本訓練のカリキュラム内容は大きく分けて二つの単元からなる。詳細は本書巻末付録参照。

- ・組込み Linux ・キャラクタデバイスドライバ開発
- ・組込み Linux ・システムコール開発

3 習得内容測定課題「簡易スロットマシン」について

3-1 機能概要

簡易スロットマシン（以下本システム）は、SH-4 マイコンボードと 7 セグメント LED 表示器拡張ボードを用いたスロットマシンである。スロットマシンのリールを模した 7 セグメント LED を 3 枚用意し、システムがその数字を規則に沿って変化させる。

同じくボード上のボタンをプレイヤーが押下することにより、コイン投入・スロット開始・リール停止をそれぞれ操作できる。3 枚のリールの目がそろった場合、その倍率によってプレイヤーへコインが返却される。ボード上の LED はそのときのゲームの状態によって投入コイン枚数の表示機能またはゲームの演出機能となる。

また、リールの更新間隔と更新パターンは設定として変更することができる。変更是本システム起動時および実行中に行なうことができる。

3-2 前提条件

本システムはマイコンボード上に実装された Linux OS 上で動作するものとする。本カリキュラムでは OS 自体の実装は行わない。

本システムのサンプルソースは本訓練側から提供されない。

本システムの成果物ソースコードは課題レポートとともに本訓練へ提出する。

本システムの LCD 表示・LED 点滅・スイッチ状態取得には、同じく本訓練前半で作成したデバイスドライバを使用する。またドライバは必要に応じて改善する。

3-3 制約事項

スロットに投入できるコイン枚数を LED 表示器で表示できるだけの枚数に制限する。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付	'09/12/20
		改定日付	'09/01/10 T.S.

3-4 システムのハードウェア、ソフトウェア構成

本システムを構成するハードウェア、ソフトウェアは以下の通りである。

・ハードウェア

本システムはSH-4マイコンボードCAT760 (SH7760仕様)を中心に、そのシリアル拡張ボードMB760とLINUX学習ボードの上で動作する。拡張ボードMB760にはSH-4のGPIO(General Parallel I/O)コネクタが備えられており、LINUX学習ボードを接続することができる(以下まとめてターゲットと呼ぶ)。

開発環境とターゲットマイコンの接続にはRS232Cシリアル接続およびイーサネット接続が利用できる。ここではWindows上のTAP-Win32トンネルデバイスにて仮想ブリッジを作り、coLinux仮想マシンの仮想NICとターゲットマイコンのNICを接続した。

LINUX学習ボードには、LCD 1基(16文字2段)、赤色LED 8基、7セグメントLED 3基、押しボタンスイッチ(ブルアッピ)5基が備わっている。図はLINUX学習ボードの外観である。

・ソフトウェア

本システムはCAT760用にコンフィギュレートされたLinuxカーネル2.6.15で動作する。ユーザランドはdebian sargeのミニマムセットになっている。

使用するコンパイラはSH-4クロスコンパイル可能なgcc version 3.4.4を使用する。

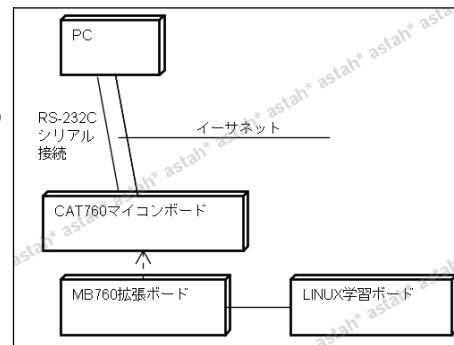


図1 システムのハードウェア構成

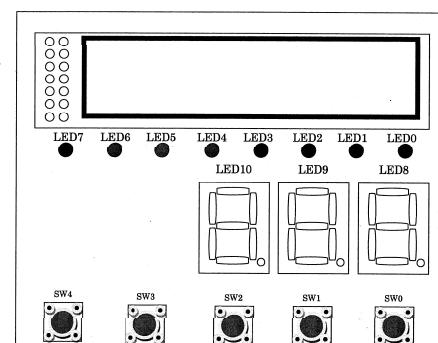


図2 LINUX学習ボード外観

本システムのターゲットおよび開発環境は以下のようになる。

表1 本システムのターゲットおよび開発環境

ターゲットマイコン	✓ CAT760 + MB760 + LINUX学習ボード
ターゲットOS	✓ Linux 2.6.15-cat CAT760専用カーネル
開発環境	✓ DELL OptiPlex755 Intel Core2Duo 2.33GHz 2GB mem
開発OS	✓ WindowsXP SP3上のcoLinux仮想マシン(debian r3.1 sarge)
クロスコンパイラ	✓ gcc version 3.4.4 20050314 (prerelease) (Debian 3.4.3-13)
ドキュメント作成	✓ Microsoft Word 2007

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付	'09/12/20
		改定日付	'09/01/10 T.S.

3-5 開発工数

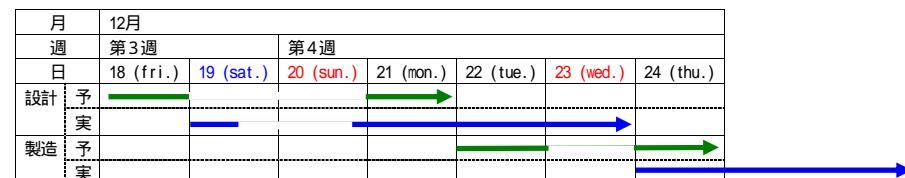
本カリキュラムは1ヶ月かけて行われた。本システムはその仕上げの課題である。その内本システムの開発自体に与えられた工数は、設計2人日、製造2人日である。これにテスト工数は含まれない。

3-6 開発実績

前述の開発工数に基づいて行われた開発実績は以下の通り。

スケジュールは訓練進捗の遅れに伴い開始日が送れることになった。

なお、予定から大きく外れることになったが、製造は1/5までおこなわれた。実質的な実装工数は8日、その後ドキュメント修正に4日費やした。予定が外れた原因是、もともとの工数が安定した動作を求める、設計工数を含めないことを前提にしていた為である。



3-7 本システムのアピールポイント

本システムのアピールポイントとして以下を挙げる。

- 状態遷移を明確にすることでプログラムは、演出や判定方法を柔軟に設計できるよう仕上がって いる。演出の変更に伴う工数の削減が期待できる。たとえば内部では、一般的のアタリ判定と確変 のアタリ判定を動的に入れ替えている。**期間短縮のため割愛** 入れ替えはモジュール化されているため、もうひとつ別のアタリ判定を追加するにも対応できる。
- デーモンとコントローラを分割することで、設定変更の反映を現在の次のスロットスタートから 反映することができる。**期間短縮のため割愛**

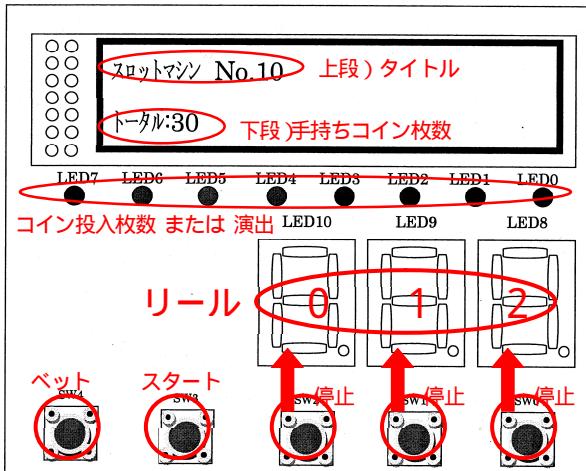
3-8 プロジェクトを進める上での課題・留意事項

本システムの仕様書は必要最低限に留める。短納期なので書きすぎに注意する。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付	'09/12/20
		改定日付	'09/01/10

4 外部設計

以下に各デバイスの機能概要を示す。



LED10、LED9、LED8 : スロットのリール
それぞれ '0' から '9' の数字を表示する。

SW4 : ベットボタン

コインを入れる代わりにこの押しボタンをコインの投入枚数だけ押す。3枚以上投入することにより、スロット開始可能状態となる。

SW3 : スロット開始ボタン

3桁の7セグメントLEDがすべて停止中のときのみ開始可能。

SW2 : LED10の停止ボタン

SW1 : LED9の停止ボタン

SW0 : LED8の停止ボタン

リールがすべて止められたときにはアタリ判定を行う。その条件は、

0以外の偶数で三桁が揃ったときに小当たり

1, 5, 9で三桁が揃ったときに中当たり

3, 7で三桁が揃ったときに大当たり

とする。当たりが出たときの動作は、任意とする。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付	'09/12/20
		改定日付	'09/01/10

以下にスロットマシンで変更可能な設定の一覧を示す。

設定はコマンドラインまたはシステムが提供するメニュー画面から変更できる。**期間短縮のため割愛**

・手持ちコイン枚数はユーザが現在持っているコインの枚数。デフォルトの初期値は30枚。

・更新間隔は各リールの数字が変化する時間(ミリ秒単位)

・更新パターンは各リールの数字が次になになるかを決めるアルゴリズム。

各桁毎にインクリメント、ディクリメント、ランダムを指定できる。

・動作モードはスロットの更新間隔と更新パターンを一括変更する。

0: 全く当たらないモード、1: 通常モード

2: 比較的当たりやすいモード、3: とても当たりやすいモード**期間短縮のため割愛**

コマンドラインまたはメニュー画面からの設定変更を実現するために、本システムはゲーム動作を実行するデーモンと、デーモンに設定を反映させるためのコントローラから構成することとする。

期間短縮のためパイプへの quit コマンドへの応答のみ実装。ただし拡張可能な形で実装

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

5 システムユースケース（略式）

以下の章では本システムの利用シナリオを列挙・概略する。

はじめに全体像としてユースケース図を示し、その後で個別のユースケースのシナリオを記述していく。

5-1 ユースケース全体像

以下に本システムのユースケース全体像を図示する。

本システムにおけるシステム外利用者は二人（Player と GameMaster）で、それぞれの役割は、

- ・ Player はゲームを行う。
- ・ GameMaster はゲームの起動停止と設定を行う。

である。そのため本ユースケース図の記述も大きく二つに分かれていることが見て取れる。

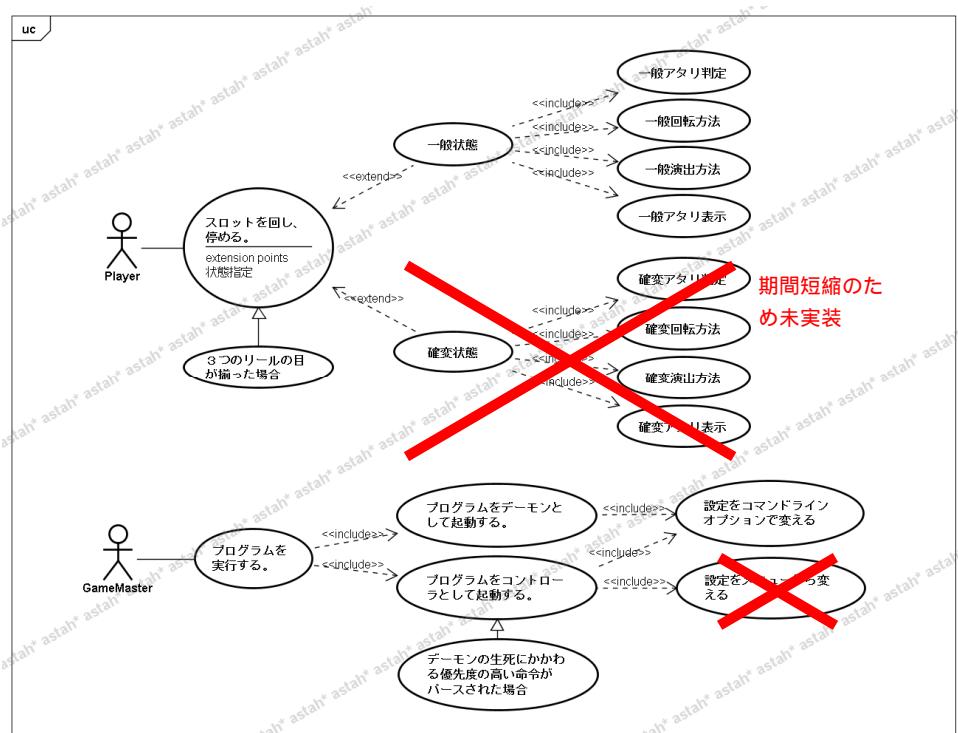


図 3 簡易スロットマシン・略式ユースケース図

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

5-2 個別ユースケース

5-2-1 UC001 「スロットを回し、停める。」

名前：スロットを回し、停める。

アクタ（ロール）： Player

事前条件：

1. スロットが回っていない。

事後条件（主シナリオ終了時の状態の条件）：

1. スロットが回っていない。

拡張点：

1. 状態指定

主シナリオ：

1. Player が手持コインを投入する。
2. Player が3つのリールを回し始める。
3. システムは状態指定毎の演出と回転方法を行う。
4. Player が3つのリールをすべて停める。
5. システムは状態指定毎のアタリを判定する。

表 2 リール停止状態表

リール	停止状態		
	1	2	3
			ptn. 0
•			ptn. 1
•	•		ptn. 2
•		•	ptn. 3
	•		ptn. 4
		•	ptn. 5
		•	ptn. 6
•	•	•	ptn. 7

バリエーション：投入不可

手順1で、投入コイン枚数が最大数(3*6)枚に達したら
ユーザにその旨伝えて手順1を行わない。

バリエーション：開始不可

手順2で、投入コイン枚数がまだ3未満なら手順2を行わない。

バリエーション：停止手順

手順4で、3つのリールの停止順序は問わない。
ただし停止順序を覚えておいてアタリ判定に利用できる。

バリエーション：ゲームオーバー

手順5で、3つのリールの目が揃わなかったときに、
Player の手持コインが3未満だったならゲームを終了する。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

5-2-2 UC002 「3つのリールの目が揃った場合。」

名前：3つのリールの目が揃った場合。

アクタ（ロール）： Player

主シナリオ：

UC001「スロットを回し、停める。」のバリエーションのひとつ。
手順5「システムがアタリを判定する。」で3つのリールの目が揃った場合、

5-1 システムは状態指定毎のアタリ表示を行う。

5-2 システムは状態指定毎の当たり表の中から該当の目を探り、
それを投入コイン枚数3に乗じた枚数のコインをPlayerに返却する。

5-2-3 UC010 「一般状態」

名前：一般状態

アクタ（ロール）： Player

主シナリオ：

UC001「スロットを回し、停める。」の拡張点「状態指定」に対するオプション
include 一般アタリ判定
include 一般回転方法
include 一般演出方法
include 一般アタリ表示

5-2-4 UC011 「一般アタリ判定」

名前：一般アタリ判定

アクタ（ロール）： Player

主シナリオ：

0以外の偶数で三桁が揃ったときに小当たり
1, 5, 9で三桁が揃ったときに中当たり
3, 7で三桁が揃ったときに大当たり

バリエーション：

7で三桁を3回連続で当てたら確変状態に移行する。

リール			当たり		
1	2	3	小	中	大
n					0
0	0	0			0
1	1	1	•		3
2	2	2	•		2
3	3	3		•	5
4	4	4	•		2
5	5	5	•		3
6	6	6	•		2
7	7	7		•	5
8	8	8	•		2

表 3 一般アタリ判定利率表

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

5-2-5 UC012 「一般回転方法」

名前：一般回転方法

アクタ（ロール）： Player

主シナリオ：

1. リールは設定の通り順番に回る。

5-2-6 UC013 「一般演出方法」

名前：一般演出方法

アクタ（ロール）： Player

主シナリオ：

1. LEDは順番に回る。

5-2-7 UC014 「一般アタリ表示」

名前：一般アタリ表示

アクタ（ロール）： Player

主シナリオ：

1. LEDは全点滅する。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

5-2-8 UC020「確変状態」

名前：確変状態

アクタ（ロール）： Player

主シナリオ：

UC001「スロットを回し、停める。」の拡張点「状態指定」に対するオプション

```
include 確変アタリ判定
include 確変回転方法
include 確変演出方法
include 確変アタリ表示
```

5-2-9 UC021「確変アタリ判定」

名前：確変アタリ判定

アクタ（ロール）： Player

主シナリオ：

三桁が揃わないときは等倍
0で三桁が揃ったときに等倍
それ以外は一般的のとき+1倍

表 4 確変アタリ判定率表

リール			当たり		
1	2	3	小	中	大
1	m	n			1
0	0	0			1
1	1	1	.		4
2	2	2	.		3
3	3	3		.	6
4	4	4	.		3
5	5	5	.		4
6	6	6	.		3
7	7	7		.	6
8	8	8	.		3

5-2-10 UC022「確変回転方法」

名前：確変回転方法

アクタ（ロール）： Player

主シナリオ：

- リールは設定の通り順番に回る。
- 最後のリールを停めた時、ほかの二つが揃っていたら、
最後のリールを2コマまで、その数値まで勝手にすすめてやる。それから判定に進む。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

5-2-11 UC023「確変演出方法」

名前：確変演出方法

アクタ（ロール）： Player

主シナリオ：

- LED は全点滅する。

5-2-12 UC024「確変アタリ表示」

名前：確変アタリ表示

アクタ（ロール）： Player

主シナリオ：

- LED は全点滅する。

5-2-13 UC100「プログラムを実行する。」

名前：プログラムを実行する。

アクタ（ロール）： GameMaster

事前条件：

- シェルからコマンドを実行する。

事後条件（主シナリオ終了時の状態の条件）：

- プログラムが実行されプロンプトが返る。

主シナリオ：

- GameMaster は目的の機能ごとに名前を変えてプログラムを実行する。
- プログラム名がデーモンを示す名前の場合、
include プログラムをデーモンとして起動する。
以降、これをシステムインスタンスまたは単にデーモンと呼ぶ。
- プログラム名がコマンダーを示すリンクの場合、
include プログラムをコマンダーとして起動する。
以降、これをシステムコントローラまたは単にコントローラと呼ぶ。
- プログラム名がそれ以外の場合、
訂正情報を示して即座に終了する。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

5-2-14 UC110 「プログラムをデーモンとして起動する。」

名前：プログラムをデーモンとして起動する。

アクタ（ロール）： GameMaster

事前条件：

1. デーモンが起動していない。

事後条件（主シナリオ終了時の状態の条件）：

1. システムが終了する。

主シナリオ：

1. システムはインスタンスのデーモン化を行う。
2. システムはプログラムの複数インスタンスを制限する。
3. システムはインスタンスのコマンド入出力となる FIFO を開く。
4. システムはインスタンスの出力となるログファイルを開く。
5. include 設定をコマンドラインオプションで変える。
6. システムはアプリケーションオブジェクト群を生成する。
7. システムは入力 FIFO を監視するループに入る。
8. ループを抜けた場合、システムはオブジェクト群を破棄する。
9. システムは終了する。

バリエーション：複数インスタンスで制限された場合

手順1でプログラムの実行が制限される場合、

システムは GameMaster に訂正情報を提示してただちに終了する。

バリエーション：FIFO のオープンエラー

手順2で FIFO が開けない場合、

システムは GameMaster に訂正情報を提示してただちに終了する。

バリエーション：ログファイルのオープンエラー

手順3でログファイルが開けない場合、

システムはすでに開かれている FIFO を閉じる。

システムは GameMaster に訂正情報を提示してただちに終了する。

バリエーション：デーモン化のエラー

手順4でデーモン化できない場合、

システムはすでに開かれている FIFO とログファイルを閉じる。

システムは GameMaster に訂正情報を提示してただちに終了する。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

バリエーション：不明なオプション

手順5で、コマンドラインオプションのパースの中斷が起きた場合、
システムはすでに開かれている FIFO とログファイルを閉じる。
システムは GameMaster に訂正情報を提示してただちに終了する。

バリエーション：アプリケーションオブジェクト生成のエラー

手順6でオブジェクト生成できない場合、
システムはすでに開かれている FIFO とログファイルを閉じる。
また、必要ならシステムは、デーモン化の手順を解除する。
システムは GameMaster に訂正情報を提示してただちに終了する。

5-2-15 UC120 「プログラムをコントローラとして起動する。」

名前：プログラムをコントローラとして起動する。

アクタ（ロール）： GameMaster

事前条件：

1. デーモンが起動している。
2. コントローラが起動していない。

事後条件（主シナリオ終了時の状態の条件）：

1. コントローラが起動していない。

主シナリオ：

1. コントローラはプログラムの複数インスタンスを確認する。
2. コントローラはコマンドラインオプションから
デーモンの生死にかかる優先度の高い命令をパースする。
3. コントローラはデーモンとのコマンド入出力となる FIFO を開く。
4. コントローラはデーモンから設定値群のコピーを得る。
5. include 設定をコマンドラインオプションで変える。
6. include 設定をメニューから変える
7. コントローラは終了する。

バリエーション：別インスタンスの不在

手順1で別インスタンスがない場合、
プログラムは訂正情報を示して即座に終了する。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 改定日付	'09/12/20 '09/01/10 T.S. T.S.

バリエーション：FIFOのオープンエラー

手順2で FIFO が開けない場合、

プログラムは GameMaster に訂正情報を提示してただちに終了する。

バリエーション：設定値群取得の失敗

手順3で、システムから設定値群のコピーが得られない場合、

プログラムはすでに開かれている FIFO を閉じる。

プログラムは GameMaster に訂正情報を提示してただちに終了する。

バリエーション：不明なオプション

手順4で、コマンドラインオプションのパースの中断が起きた場合、

プログラムはすでに開かれている FIFO を閉じる。

プログラムは GameMaster に訂正情報を提示してただちに終了する。

5-2-16 UC121 「デーモンの生死にかかわる優先度の高い命令がパースされた場合」

名前：デーモンの生死にかかわる優先度の高い命令がパースされた場合

アクタ（ロール）： GameMaster

主シナリオ：

ユースケース「プログラムをコントローラとして起動する。」のバリエーションのひとつ。

手順3「コントローラはコマンドラインオプションから

「デーモンの生死にかかわる優先度の高い命令をパースする。」で

デーモンの生死にかかわる優先度の高い命令がパースされた場合、

3-1 パースされた命令が終了命令だった場合、

コントローラはデーモンに SIGTERM シグナルを送る。

コントローラはただちに終了する。

3-2 パースされた命令が設定の即時反映命令だった場合、

コントローラはデーモンに設定を反映する。

また、コントローラはただちに終了する。

5-2-17 UC131 「設定をコマンドラインオプションで変える。」

名前：設定をコマンドラインオプションで変える。

アクタ（ロール）： GameMaster

主シナリオ：

1. プログラムはコマンドライン文字列から設定値群をパースする。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 改定日付	'09/12/20 '09/01/10 T.S. T.S.

2. プログラムは設定値群を反映する。

バリエーション：不明なオプション

手順1で、不明なオプションがあった場合メッセージを出して中断を返す。

5-2-18 UC132 「設定をメニューから変える。」

名前：設定をメニューから変える。

アクタ（ロール）： GameMaster

主シナリオ：

1. システムがシナリオ開始時点の設定値群のコピーを作る。

2. システムは以下を表示する。

- ・変更可能な設定の一覧
- ・設定反映
- ・設定終了

3. GameMaster はいずれかを選択する。

4. システムはバリエーション動作を行う。

5. GameMaster は動作の中で目的の値や動作命令を入力する。

6. システムは動作の結果としてキャンセル命令を受けたら手順2に返る。

7. システムは動作の結果として終了命令を受けたらシナリオを終了する。

8. システムは動作の結果として継続命令を受けたら手順2に返る。

バリエーション：変更可能な設定の一覧

4-1 GameMaster が変更可能な設定の一覧から設定のひとつを選んだ場合

4-1-1 システムは設定値の選択肢が定まるなら選択肢を表示する。

もしくはシステムは設定値の入力を待つ。

選択肢または入力にはキャンセル用の値が含まれる。

5-1-1 GameMaster は設定値を選択または入力する。

5-1-3 入力が範囲外の場合、システムは訂正情報を表示する。手順は 4-1-1 に返す。

5-1-2 入力がキャンセル値の場合、システムは動作の結果としてキャンセル命令を返す。

5-1-4 入力が設定値の場合、システムは手順1で作ったコピーの設定値群に値を反映する。

また、システムは動作の結果として継続命令を返す。

バリエーション：設定反映

4-2 GameMaster が一覧から設定反映を選んだ場合

4-2-1 システムは設定反映を本当に反映するか問う。

選択肢は反映する場合とキャンセルする場合の2択。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 改定日付	作成者 改定者 T.S.

5 -2-1 GameMaster は本当に反映するかを答える。

5-2-2 入力が範囲外の場合、システムは訂正情報を表示する。手順は4-2-1に返す。

5-2-3 キャンセルする場合、システムは動作の結果としてキャンセル命令を返す。

5-2-4 設定を反映する場合、システムは手順1で作ったコピーの設定値群を

本来の設定値群に反映する。

システムは動作の結果として継続命令を返す

バリエーション：設定終了

4-3 GameMaster が一覧から設定終了を選んだ場合

4-3-1 システムはコピーの設定値群と本来の設定値群の差異を探る。

4-3-2 差異がある場合、システムは未反映の設定を反映するか問う。

選択肢は反映する場合と反映しない場合と終了をキャンセルする場合の3択。

5 -3-1 GameMaster は反映するかを答える。

5-3-2 入力が範囲外の場合、システムはエラーメッセージを表示する。手順は4-3-2に返す。

5-3-3 キャンセルする場合、システムは動作結果としてキャンセル命令を返す

5-2-4 設定を反映する場合、システムは手順1で作ったコピーの設定値群を

本来の設定値群に反映する。その後、システムは動作の結果として終了命令を返す

5-2-4 設定を反映しない場合、システムは未反映の設定を本当に破棄するか問う

選択肢は破棄する場合と終了をキャンセルする場合の2択。

GameMaster は破棄するかを答える。

5-2-2 入力が範囲外の場合 システムは訂

5-2-3 キャンセルする場合 システムは動作の結果としてキャンセル命令を返す

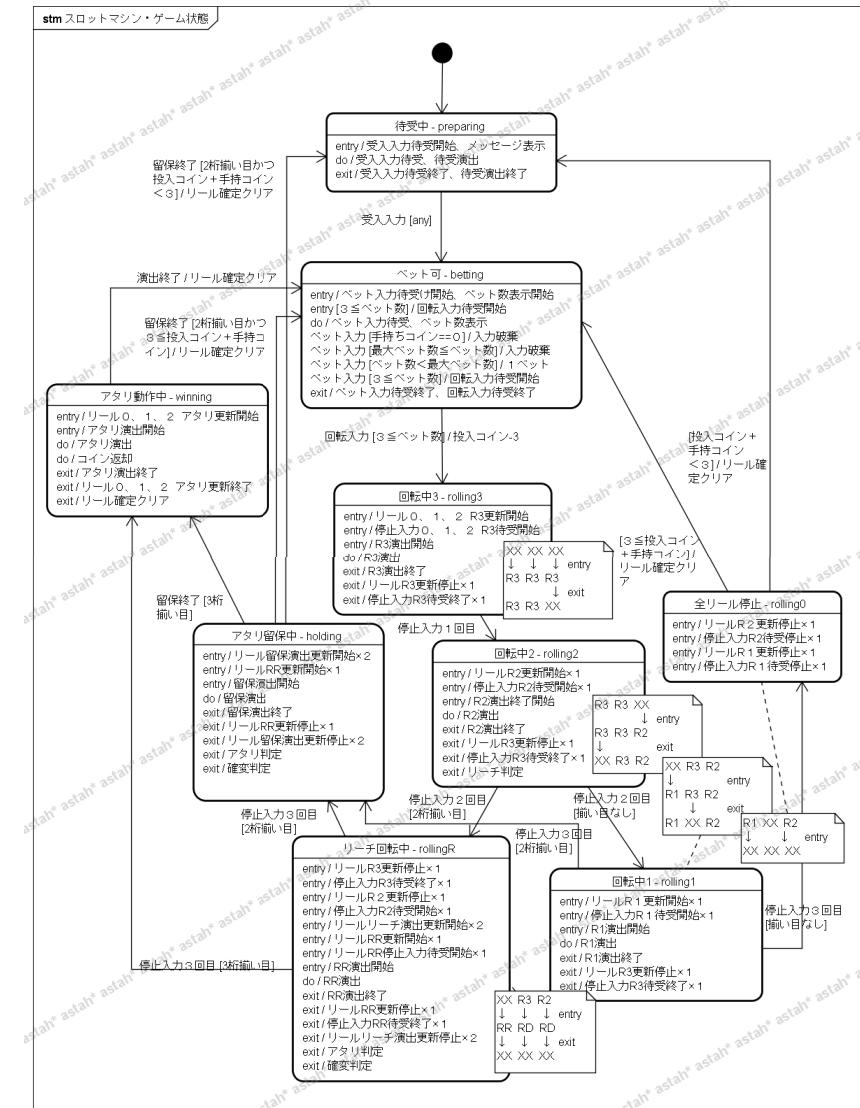
5.2.4 未反映の設定を破棄する場合、システムは動作の結果として終了命令を返す。

3-2-4 不反映的假定を破棄する場合、システムの運動の結果として終了部分を選択。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科（一般）
スロットマシンを作ろう！ 簡易設計書		作成日付 改定日付	'09/12/20 '09/01/10 作成者 改定者 T.S. T.S.

6 状態図

以下に本システムゲーム中の状態図を示す。注)図背景のグレー文字はツールが試用版の為で特に意味はありません。



ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 改定日付	作成者 T.S. 改定者 T.S.

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！	簡易設計書	作成日付 改定日付	'09/12/20 '09/01/10 作成者 T.S. 改定者 T.S.

7 処理シーケンス

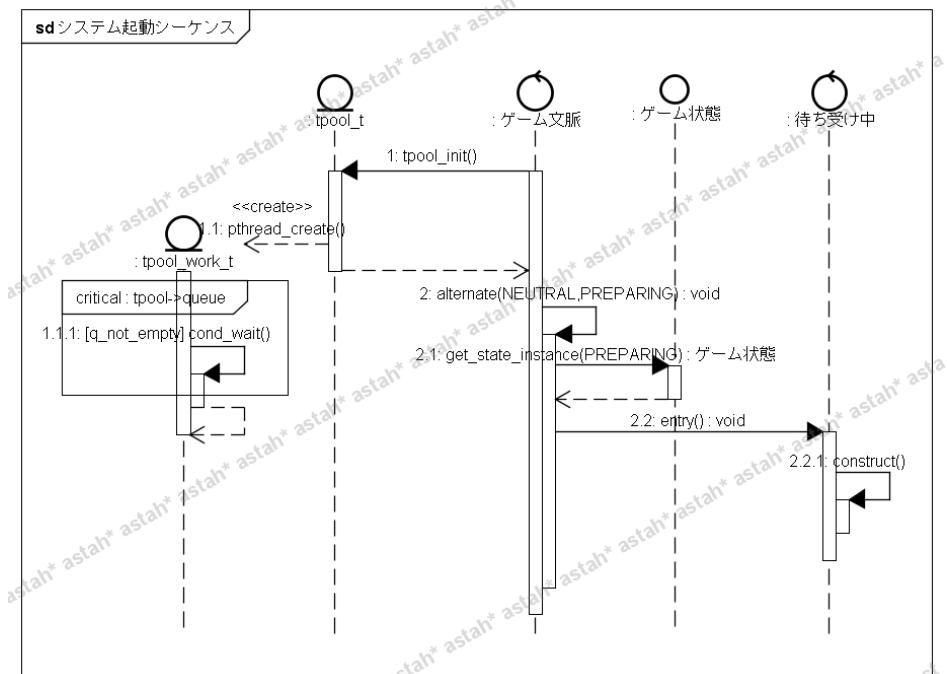
以下の章では本システムの処理シーケンスを列挙・概略する。

7-1 システム起動とスレッドプール

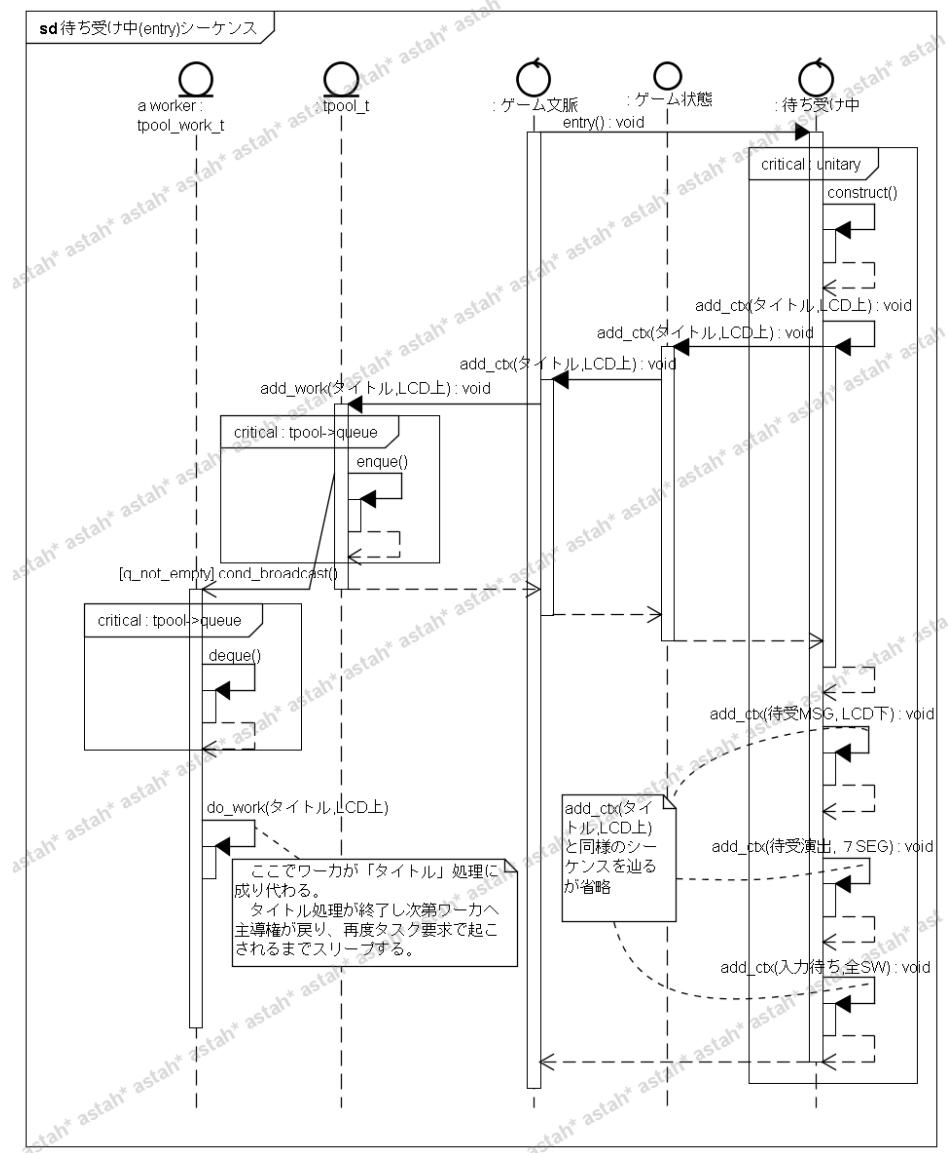
ゲーム文脈はゲームの流れを司るオブジェクトで、ここをゲームの起点とする。

ゲームの設計者は必要に応じて、ときどきの状態が必要なコンテンツを文脈に追加する。このためゲーム文脈には着脱可能な機能実装が必要で、今回はスレッドプールとして実装することとした。スレッドプールはゲーム文脈の生成と同時に初期化され利用待ち状態に入る。プールが初期化されると、複数のワーカスレッド（デフォルト 30 本）も生成され、初期動作としてそのままスリープする。これはワーカがプールのタスクキュエンプティを示す状態変数によって起こされるまで続く。

また、各ゲーム状態への遷移はゲーム文脈を通して行われる。ゲーム文脈の生成時は単に待ち受け中状態に遷移するのみである。遷移の初期段階が終わったところでゲーム文脈の生成は成功し、プログラムコンテキストはゲーム文脈を生成しようとしたコンテキストに返されることになる。この元コンテキストのなかでは無限ループが行われ、遷移先のゲーム状態がゲーム文脈を通じてその終了フラグをセットするまで待ち受けることになる。

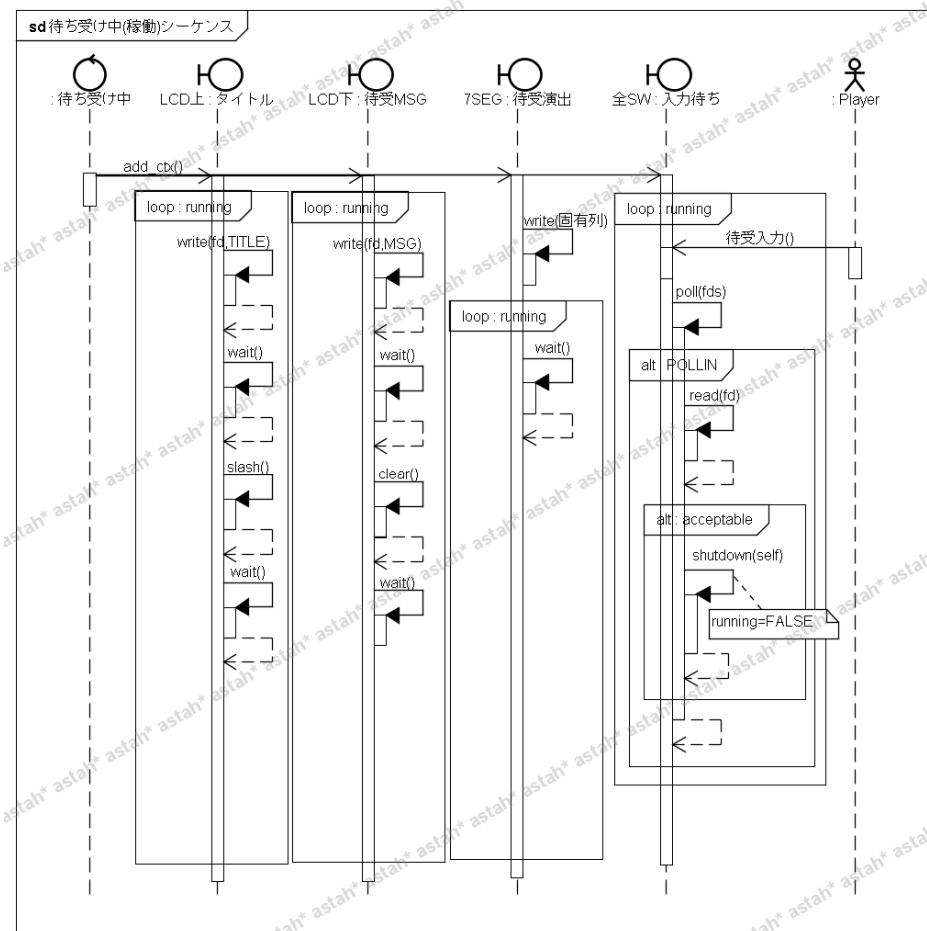


7-2 待ち受け中 (entry)



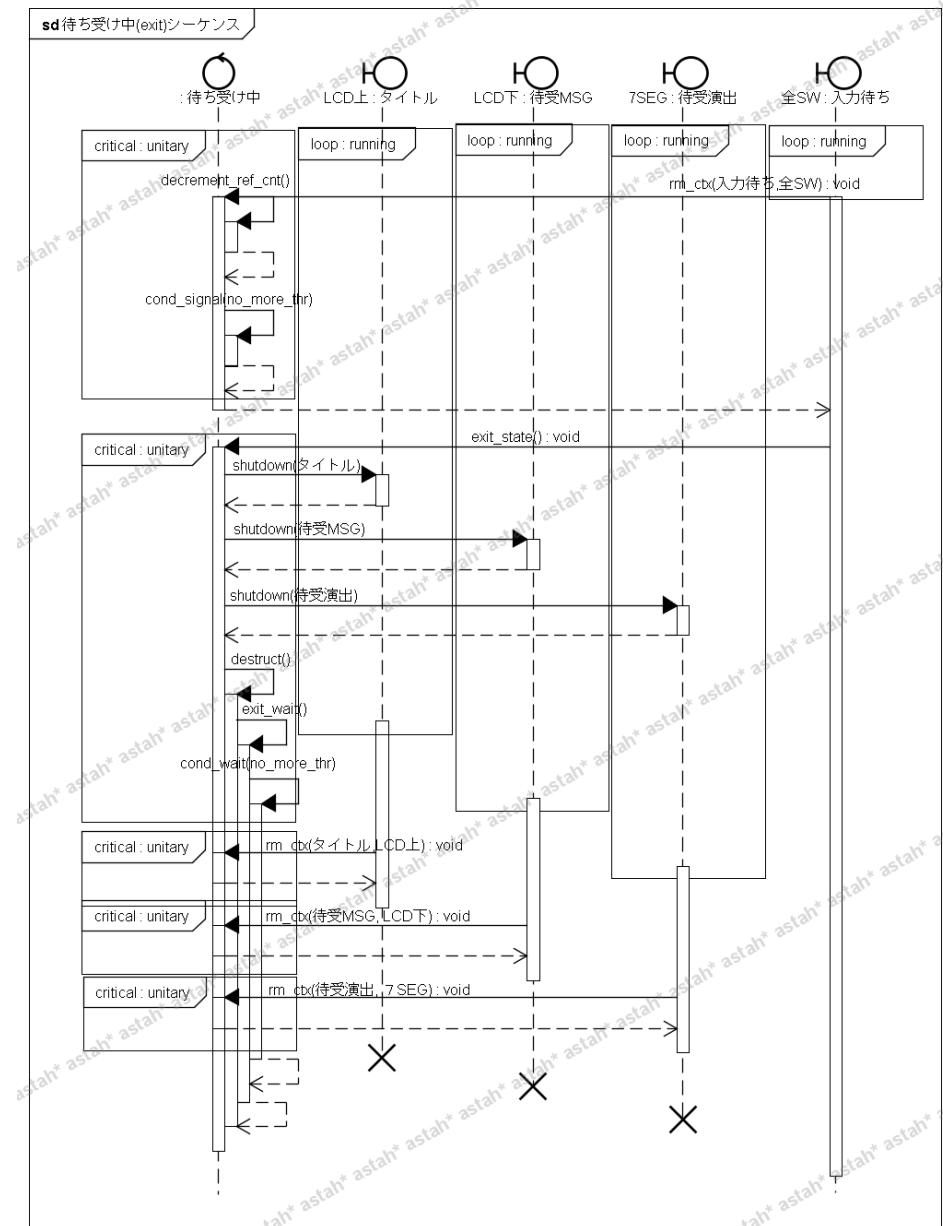
ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
作成日付	'09/12/20	作成者	T.S.
改定日付	'09/01/10	改定者	T.S.

7-3 待ち受け中(稼動)



7-4 待ち受け中 (exit)

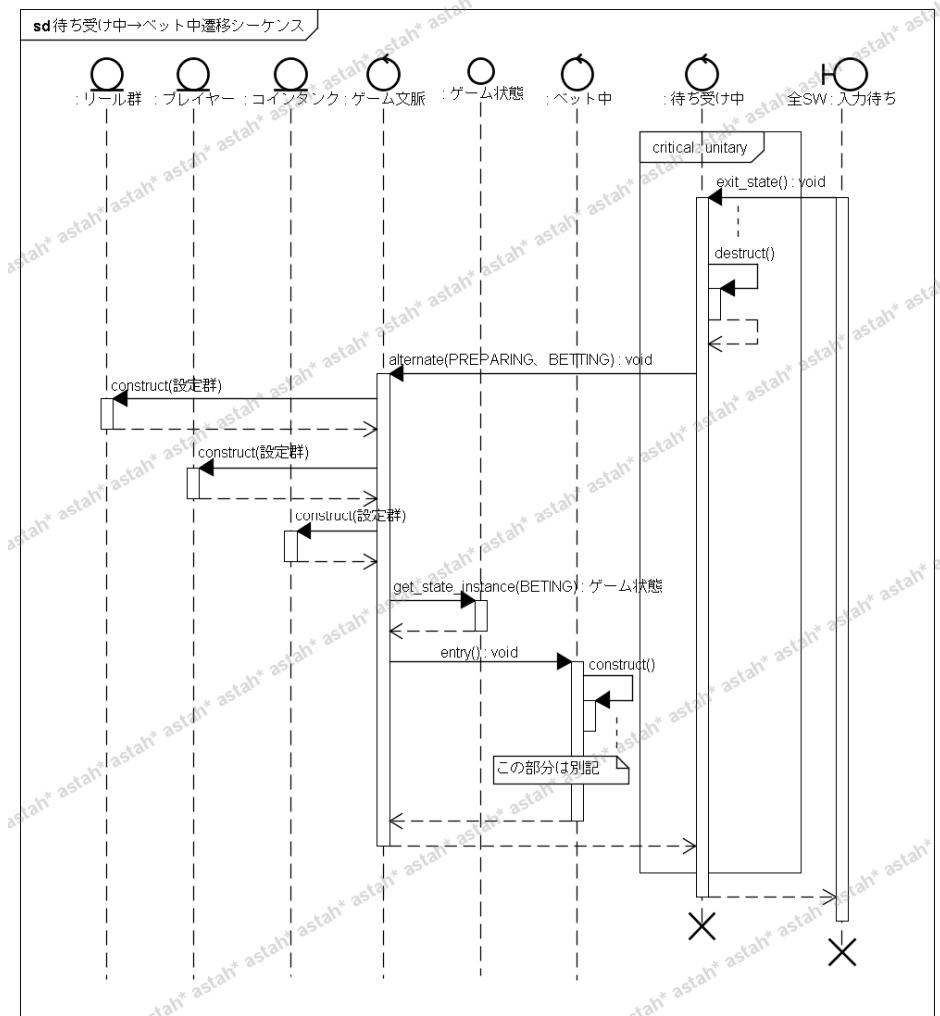
ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
作成日付	'09/12/20	作成者	T.S.
改定日付	'09/01/10	改定者	T.S.



ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科（一般）
スロットマシンを作ろう！ 簡易設計書		作成日付 改定日付	'09/12/20 '09/01/10 作成者 改定者 T.S. T.S.

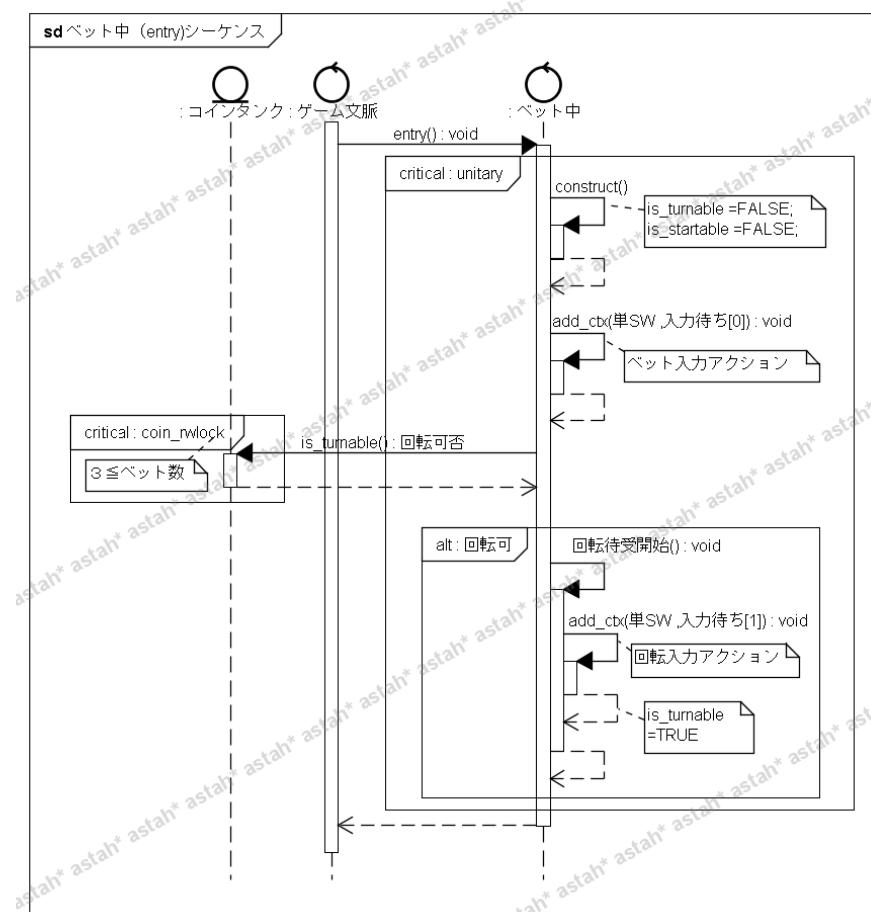
ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 改定日付	'09/12/20 '09/01/10 作成者 改定者 T.S. T.S.

7-5 待ち受け中からベット中への遷移



ゲーム状態を終了させたスレッドがスコープ外に抜けるタイミングが他のスレッドと異なる点は特に変則的なため注意する。遷移先の entry 遷移前の exit_state 状態を終了させたスレッドの順でスコープを抜ける。ロックを持ったまま遷移先でロックを取ろうとしてしまうことで、容易にdeadlockする。

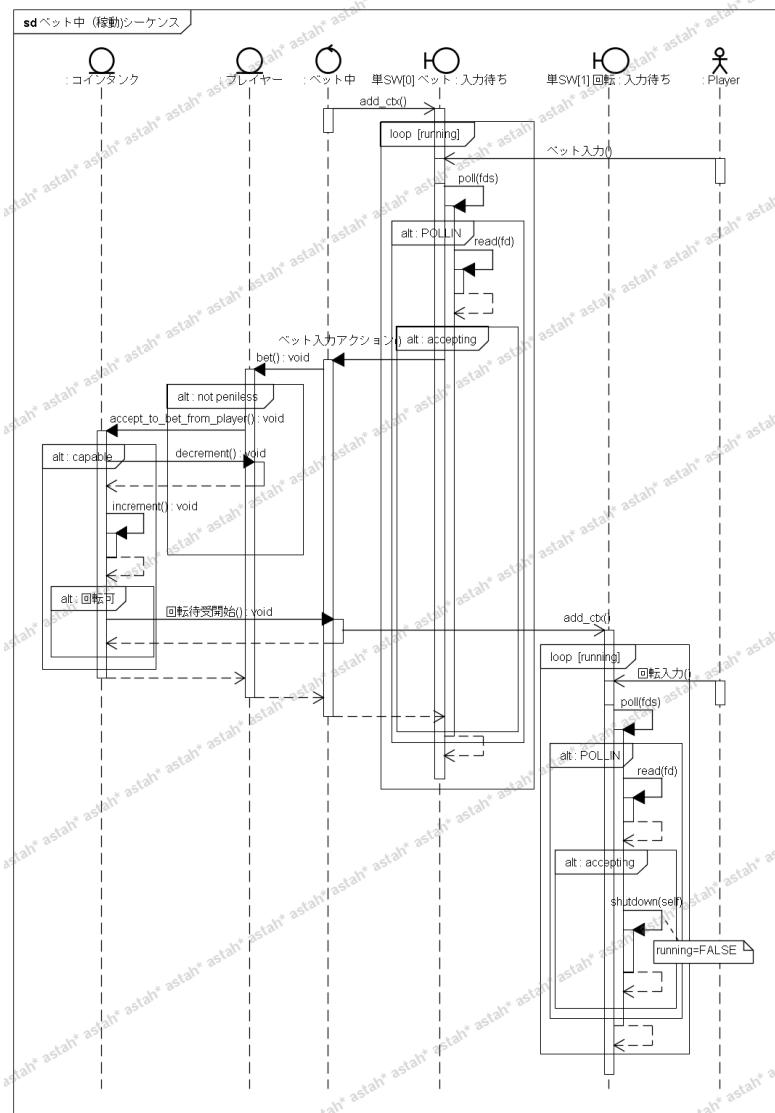
7-6 ベット中 (entry)



ペットを受け付け始めるにあたり、これまでの操作でコインタンクに回転可能な投入枚数ぶんのコインをすでに貯め込んでいた場合は即座に回転入力を受け付けるよう遷移する。

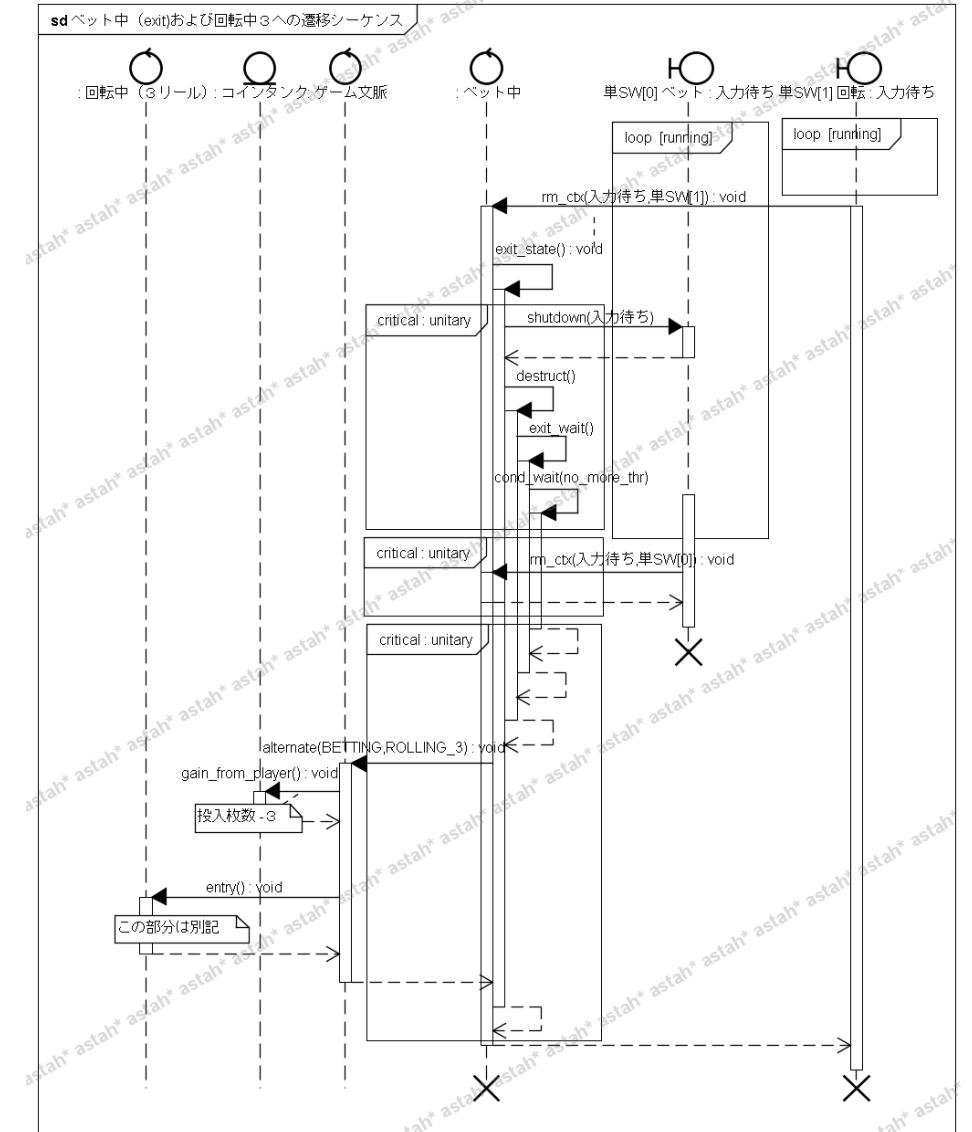
ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
作成日付	'09/12/20	作成者	T.S.
改定日付	'09/01/10	改定者	T.S.

7-7 ベット中（稼動）



ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
作成日付	'09/12/20	作成者	T.S.
改定日付	'09/01/10	改定者	T.S.

7-8 ベット中 (exit) から回転中 3 への遷移

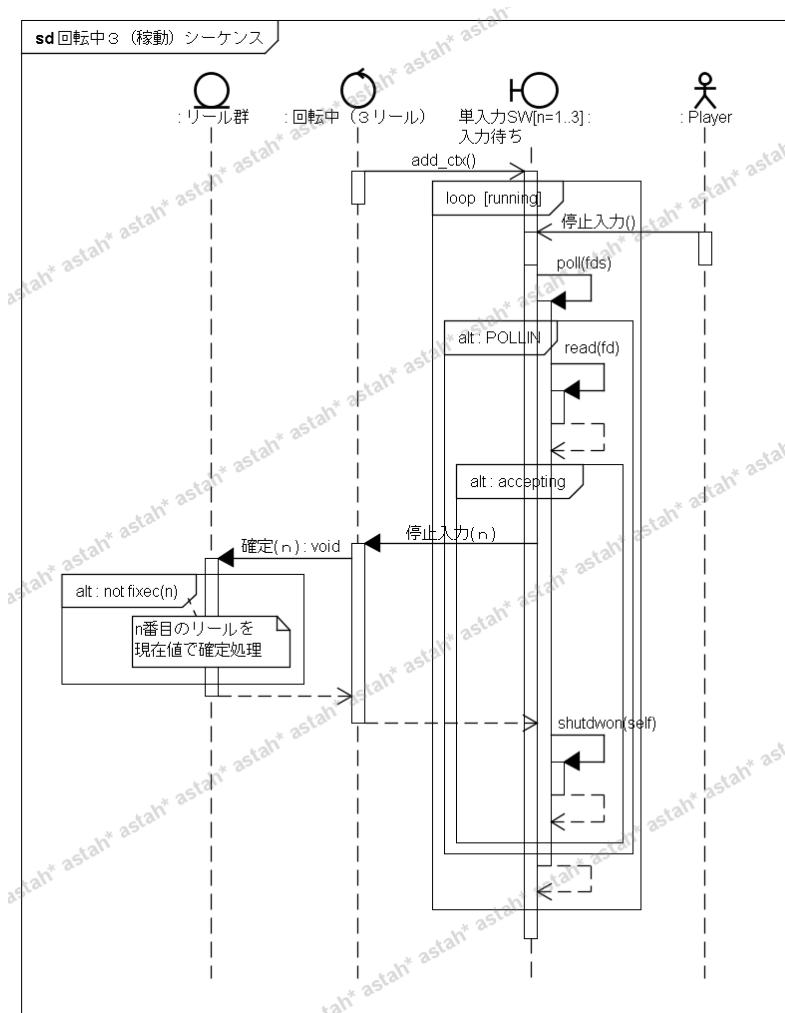


ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

7-9 回転中3 (entry)

回転中3のentryシーケンスはこれまで同様のため省略。

7-10 回転中3 (稼動)



回転中は未確定のリール各々に対して停止入力を待ち、入力があればリールを確定する。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

7-11 回転中3 (exit) から回転中2への遷移

回転中(3リール)からの遷移は確定されたリールの停止入力待ちが終了した時点で開始される。シーケンスについてはこれまで同様のため省略。

7-12 回転中2 (entry)、(稼動)、(exit) および遷移

回転中(2リール)のシーケンスは確定済みのリールで停止入力を受け付けないこと、および演出の表示方法が表面上異なることを除きこれまで同様。

回転中2からの遷移は確定されたリールの停止入力待ちが終了した時点で開始される。遷移の際にリール群に確定状態を問い合わせ、リールの目が二つ揃っている(つまりリーチしている)場合のみリーチ中状態へ遷移する。リーチしていない場合は回転中(1リール)へ遷移する。

処理の流れ自体はこれまで同様のため省略。

7-13 回転中1 (entry)、(稼動)、(exit) および遷移

回転中(1リール)のシーケンスは確定済みのリールで停止入力を受け付けないことを除いてこれまで同様。

回転中1からの遷移は確定されたリールの停止入力待ちが終了した時点で開始される。遷移の際にリール群の確定状態を問い合わせ、リールの目が二つ揃っている(つまり惜しかった)場合のみアタリ保留中状態へ遷移する。揃っていない場合は全停止状態へ遷移する(ゲーム性の話題になるが、保留状態への遷移があることで、ゲームを楽しみ易いものにする)。

処理の流れ自体はこれまで同様のため省略。

7-14 全リール停止 (entry)、(稼動)、(exit) および遷移

全リール停止ではゲームの継続判定と若干のクリーンナップを行う。稼動は実質的に処理をしておらずダミーを介す。ダミーはすぐに終了する。

全リール停止からの遷移はダミーの終了した時点で開始される。遷移の際にプレイヤーの手持ちコインとコインタンクの残枚数を確認し、ゲームが継続可能ならばベット中状態へ遷移する。継続不可能なら待受中状態へ遷移する。

7-15 アタリ保留中 (entry)、(稼動)、(exit) および遷移

アタリ保留中状態では最終的に二つの目まで揃えることができた場合(含リーチ)に、ボーナスとして一定回転数リールをズラすゲーム演出を行う(これを滑らすと呼ぶらしい)。この状態はゲーム性のために追加した。シーケンス自体はこれまで同様である。

アタリ保留中からの遷移は演出表示が終了した時点で開始される。演出によってリールの三つの目がそろった場合アタリ演出中状態へ、そろわなかった場合はプレイヤーの手持ちコインとコインタンクの残枚数を確認し、ゲームが継続可能ならばベット中状態へ、継続不可能ならば待受中状態へ遷移する。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

7-1 6 リーチ中 (entry)、(稼動)、(exit) および遷移

リーチ中状態のシーケンスは確定済みのリールで停止入力を受け付けないこと、および、確定済みのリールで演出的な表示がされることを除いてこれまで同様。

リーチ中からの遷移は確定されたりールの停止入力待ちが終了した時点で開始される。遷移の際にリール群の確定状態を問い合わせ、リールの目がすべて揃っている場合にアタリ演出状態へ遷移する。そろっていない場合はアタリ保留中状態へ遷移する。

処理の流れ自体はこれまで同様のため省略。

7-1 7 アタリ動作中 (entry)、(稼動)、(exit) および遷移

アタリ動作中のシーケンスはアタリ時のプレイヤーへのコイン返却の計算と演出的な表示だけを行っている。稼働状態は実質的に処理をしておらず表示処理を介すのみである。表示処理は一定時間で終了する。

アタリ動作中からの遷移は演出表示が終了した時点で開始される。遷移はベット状態へのみ行われる。

処理の流れ自体はこれまで同様のため省略。

8 クラス間関連

以下の章では本システムのユースケース、状態図、処理シーケンスから導かれたクラス間関連を列举・概略する。

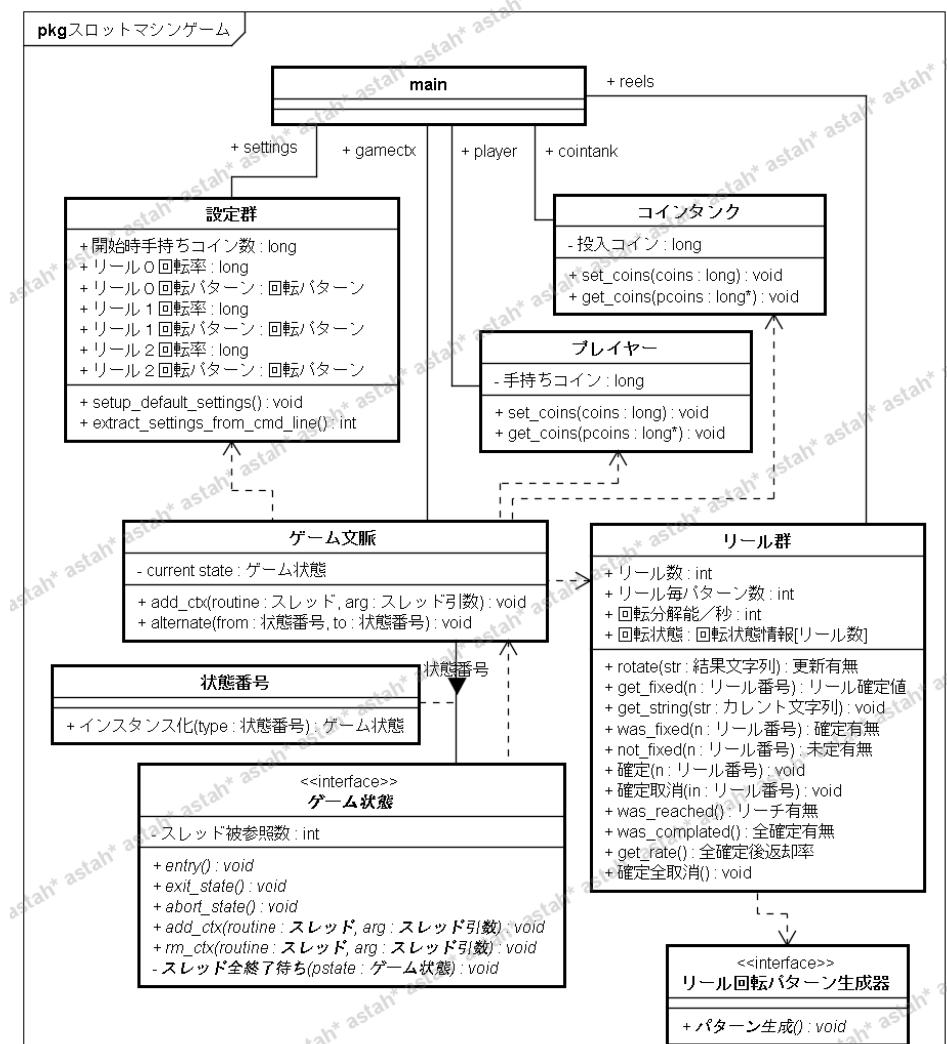
本システムのクラスは大きく二つ、ゲームシステム自体とゲームシステムの実行時間における各時点での状態に分類される。

ゲームシステムでは、main とそこから生成されるプレイヤー、コインタンク、設定群、リール群の各機能を、ゲーム文脈（コンテキスト）が依存・操作する。

ゲーム状態は、ゲーム文脈によってゲームの流れにそった適切な状態として選択・実行される。各状態は自分の状態と自分の次の状態およびその遷移条件を知っており、各自の終了時に次の状態へ適切にバトンタッチする。この実装のために GoF のステートパターンを利用する。

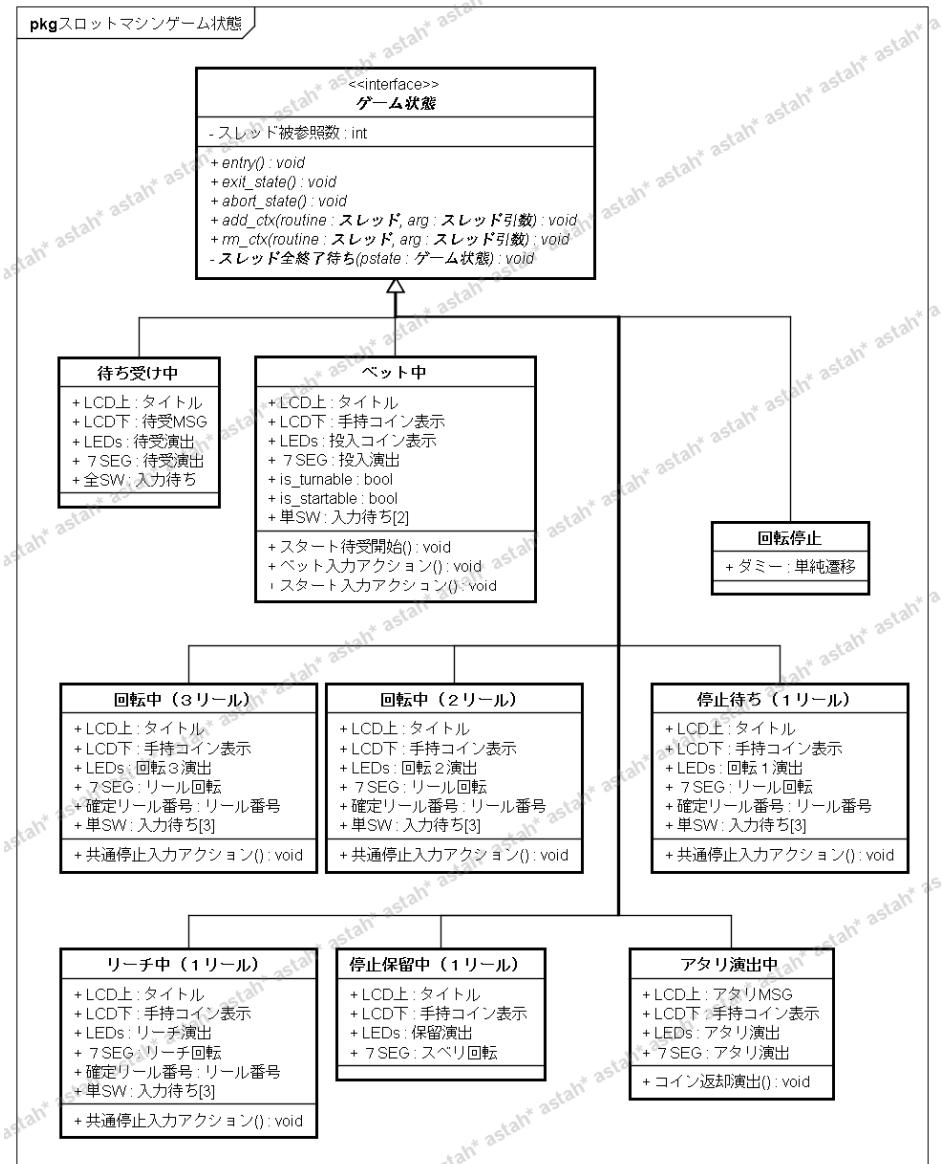
ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

8-1 スロットマシンゲーム



ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
作成日付	'09/12/20	作成者	T.S.
改定日付	'09/01/10	改定者	T.S.

8-2 スロットマシンゲーム状態



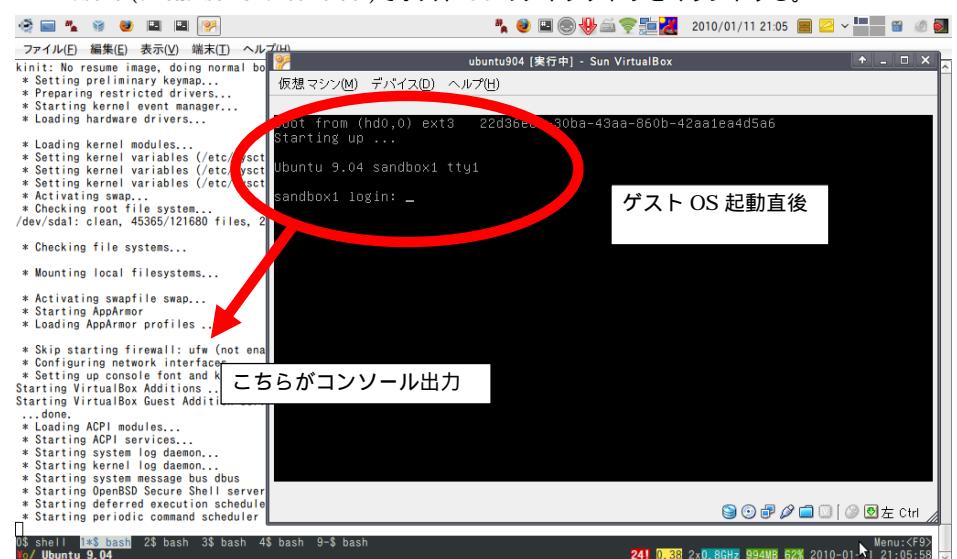
ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
作成日付	'09/12/20	作成者	T.S.
改定日付	'09/01/10	改定者	T.S.

9 スタブ実装環境

本課題の実装段階での実行テストには本来実機が必要なのだが、期間の都合上今回は実機環境でのテストができなかった。やむなく自宅での実装・テストを行うため、ドライバのスタブ動作を実装し実機無しでの開発環境を実現した。スロットマシンはすべてスタブ上で実装され、実機の利用は速度調整のみで済ませることができた（当り前なことではあるのですが、これにはとても驚きました）。

スタブを利用した実行テストの概略は以下の通り。テスト環境は Intel ATOM 1.6GHz メモリ 1G 上の Ubuntu 9.04、およびその上で動く Sun VirtualBox バーチャルマシンのサンドボックスである。

- 1) ゲスト側の VM 設定でシリアルポートを UNIX ソケットに関連付ける。
この際 UNIX ソケットファイルを VM が作成しない、既存のものを利用する設定とする。
- 2) ホスト側で socat(socket cat)を実行
 - socat は UNIX ソケットファイルを作成し、入力内容をダンプする。
 - "socat unix-listen:/vbox_tty,reuseaddr,fork -"
 - ゲスト OS からシリアルコンソールが出力されるための FIFO が作られる。
- 3) VirtualBox から sandbox を起動
 - ゲスト OS が FIFO にシリアルコンソール出力を始めると、
 - vboxsf(virtualbox shared folder)でホスト OS のディレクトリをマウントする。



ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 改定日付	作成者 T.S. 改定者 T.S.

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！	簡易設計書	作成日付 改定日付	T.S. T.S.

4) シャープ構築を有効にしてデバイスドライバをビルドする。

- Makefile でスタブ構築を指定する。
 - debug.h で IN_LOCAL を define する。
 - H/W を迂回するモジュールが作られる。
 - モジュールが vboxsf で参照できる。

5) ゲスト OS に ssh 接続する。

- ゲスト OS で root になることでドライバのテストを行う。

6) テスト準備スクリプトを実行する。

- テスト用デバイスノードを作成する（udev 未使用）。
 - スタブドライバを insmod する。

7) シャブ構築を有効にしてプログラムを構築する。

- Makefile でスタブ構築を指定する。
 - ローカルデバッグ出力が有効なモジュールが作られる。
 - Makefile に NDEBUG を指定すればデーモン化しないバージョンが作られる。
そのモジュールならば gdb でトレスすることもできる。
 - モジュールが vboxsf で参照できる。

8) ゲスト OS でプログラムを実行、テストする。

```
[28075, 270827] lcd mod:lcd write() write <- '0$0$slotmachine-hine No.'8'(19)
[28075, 365475] lcd mod:lcd write() write <- '0$0$slotmac-in-e No.'(19)
[28075, 408688] lcd mod:lcd write() write <- '0$139'(5)
[28075, 427193] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28075, 518205] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28075, 578463] ddsp mod:dynamic_write (ddsp) buf"
[28075, 593906] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28075, 675670] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28075, 733165] lcd mod:lcd write() write <- '0$139'(5)
[28075, 757869] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28075, 837525] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28075, 919383] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28076, 006415] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28076, 040988] lcd mod:lcd write() write <- '0$139'(5)
[28076, 090088] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28076, 118811] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28076, 210831] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28076, 295103] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28076, 358429] lcd mod:lcd write() write <- '0$139'(5)
[28076, 380551] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28076, 463488] lcd mod:lcd write() write <- '0$0$slotach-ne'
[28076, 550220] lcd mod:lcd write() write <- '0$0$slotach-ne'
[28076, 596567] ddsp mod:dynamic_write (ddsp) buf"
[28076, 637987] lcd mod:lcd write() write <- '0$0$slotmachine-hine'
[28076, 701846] lcd mod:lcd write() write <- '0$139'(5)
[28076, 717755] lcd mod:lcd write() write <- '0$0$slotmac-in-e'
[28076, 803297] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28076, 895610] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28076, 967362] lcd mod:lcd write() write <- '0$0$slotmach-ne'
[28077, 016960] lcd mod:lcd write() write <- '0$139'(5)
[28077, 049103] lcd mod:lcd write() write <- '0$0$slotmach-ne-No.'8'(19)
[28077, 149103] lcd mod:lcd write() write <- '0$0$slotmach-ne-No.'(19)
[28077, 240618] lcd mod:lcd write() write <- '0$0$slotmach-ne-'(19)
[28077, 299071] lcd mod:lcd write() write <- '0$139'(5)
[28077, 324411] lcd mod:lcd write() write <- '0$0$slotmachne No.'8'(19)
[28077, 405619] lcd mod:lcd write() write <- '0$0$slotmachne No.'(19)
[28077, 496496] lcd mod:lcd write() write <- '0$0$slotmachne No.'8'(19)

0$ shell [4$ bash 25 bash 38 bash 4$ bash 9$ bash
$_/ Ubuntu 9.04
```

10 所感

今回の訓練課題では自分の未経験技術への新しい試みを行いました。すなわちオブジェクト指向分析・設計とモデリングへの挑戦です。結果、経験不足のみえた荒削りなドキュメントになりましたが、今後の足がかりになるものを得られたと考えます。特にユースケース シーケンス クラス間関連と分析する過程は、自分にとって自然なシステム認識の流れに感じられ、今後の洗練を経ることでより確実な設計方法になると感じています。

本資料は以上です。

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

1.1 卷末付録

1.1-1 第4システム・カリキュラム詳細

1.1-2 参考文献

1.1-1 第4システム・カリキュラム詳細

1.1-1-1 組込み Linux キャラクタデバイス開発

1. キャラクタデバイスを開発するための準備

1.1. 概要

1.2. Linux のメモリモデル

1.2.1. 仮想アドレス空間

1.2.2. 空間レイアウト

1.3. 開発環境

1.3.1. ターゲット機へのイメージのロード

1.3.2. zImage のロードを含むカーネル起動処理

1.3.3. ホストとターゲットの接続

1.3.4. シリアルポートとターミナルエミュレータ

1.3.5. ブートとログイン

1.3.6. ファイルシステム

1.3.7. ネットワーク環境の編集

1.3.8. 時計のセット

1.3.9. エイリアスの作成

1.3.10. ターゲットへの coLinux の/home/kaihatsu のマウント

1.4. カーネルの再構築

1.4.1. カーネルイメージの圧縮ファイルを解凍

1.4.2. カーネルのコンフィグレーション

1.4.3. Makefile の変更

1.4.4. カーネルのメイク

1.4.5. カーネルイメージを FLASH MEMORY に書き込む

2. キャラクタ型デバイスドライバ開発

2.1.1. モジュールの初期化、後処理（クリーンアップ）関数

2.1.2. モジュールのメイク

2.1.3. モジュールのロードと削除

2.2. 最初のキャラクタデバイス

2.2.1. メジャー番号とマイナー番号

2.2.2. ファイル操作構造体

2.2.3. キャラクタ型デバイスの登録

2.2.4. file 構造体と inode 構造体

2.2.5. open メソッドと release メソッド

2.2.6. read メソッドと write メソッド

2.2.7. 最初のキャラクタデバイス leddrvO.c

2.2.8. 最初のキャラクタデバイスの実装と動作確認

2.2.9. 最初のキャラクタデバイスとアプリケーションプログラム

2.3. マイナー番号の応用例

2.3.1. キャラクタデバイス leddrv1.c

2.3.2. filp->private_data

2.3.3. コマンドを利用しての動作確認

2.3.4. leddrv1 とアプリケーションプログラム

2.4. スイッチにもマイナー番号を振る

2.4.1. キャラクタデバイス ledswdrvO.c

2.4.2. モジュールの動作確認

2.4.3. アプリケーションによる動作確認

2.5. ioctl を使って、新しい機能を追加

2.5.1. ioctl システムコールと ioctl メソッド

2.5.2. ioctl 番号

2.5.3. get_user と put_user マクロ関数

2.5.4. leddrv2

2.5.5. LED テストコマンドの作成

2.6. スイッチからの割り込み

2.6.1. 割り込みハンドラの登録・解放

2.6.2. 待ち列ヘッド

2.6.3. wait_event、wake_up マacro

2.6.4. sw_irq.c

2.6.5. 課題 7 セグメント LED のダイナミック点灯

2.7. 排他制御 - LCD に文字を表示させる -

2.7.1. セマフォの初期化

2.7.2. down、up 関数

2.7.3. lcddrv.c

2.8. 完成したドライバの組込みと自動ロード

2.8.1. 完成したドライバの組込み

2.8.2. モジュールの自動ロード

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

1.1.1-2 組込み Linux システムコール開発

1. 組込み Linux

1.1. 概要

- 1.1.1. サポートされている CPU アーキテクチャ
- 1.1.2. なぜ、組込み Linux が利用されるか？

1.2. 組込み Linux システム

- 1.2.1. 開発に必要となるソフトウェア
- 1.2.2. 組込み機器への Linux を実装する場合の手順

1.3. 組込み Linux 開発環境

- 1.3.1. 開発環境
- 1.3.2. 開発ツール
- 1.3.3. ライブラリ
- 1.3.4. ブートローダ
- 1.3.5. ブートローダの処理
- 1.3.6. ロードされるイメージ
- 1.3.7. zImage のロードを含むカーネル起動処理

2. Linux カーネルの基本性能

- 2.1. プロセス
- 2.2. スレッド
- 2.3. Linux のメモリモデル
 - 2.3.1. 仮想アドレス空間
 - 2.3.2. 空間レイアウト
- 2.4. ファイルシステム
 - 2.4.1. ファイル形式
- 2.5. ネットワーク機能
- 2.6. クロス開発の確認

- 2.6.1. ターゲット機へのイメージのロード
- 2.6.2. ホストとターゲットの接続
- 2.6.3. シリアルポートとターミナルエミュレータ
- 2.6.4. ブートとログイン
- 2.6.5. ネットワーク環境の編集
- 2.6.6. サンプルプログラムの作成 (catex1_1.c)
- 2.6.7. 実行ファイルの生成とファイルタイプ
- 2.6.8. ターゲット機での実行
- 2.6.9. gdb サーバによるリモートデバッグ
- 2.6.10. 課題 gdb サーバによるリモートデバッグの確認 (catex1_2.c)

3. システムコール

3.1. システムコールとは

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 '09/12/20 改定日付 '09/01/10	作成者 T.S. 改定者 T.S.

3.2. システムコールのエラー処理

- 3.2.1. 課題 エラー番号の表示 (catex2_2.c)
- 3.2.2. 課題 エラー内容の表示 (catex2_4.c)

4. プロセス

- 4.1. プロセスとは
 - 4.1.1. シングルプログラミング OS とマルチプログラミング OS について
 - 4.1.2. TSS 系 OS (マルチプログラミング OS)
 - 4.1.3. プロセス管理
 - 4.1.4. プロセスの複製、置換の流れ
- 4.2. プロセスの状態遷移
- 4.3. プロセス関係システムコール
 - 4.3.1. fork()系 - プロセスの複製 -
 - 4.3.2. fork()系のサンプル 1 (catex3_1.c)
 - 4.3.3. fork()系のサンプル 1 (catex3_2.c)
 - 4.3.4. プロセス実行時の仮想空間について
 - 4.3.5. exec 系 - プロセスの置換 -
 - 4.3.6. 課題プロセスの置換 (catsk3_1.c)
 - 4.3.7. 課題プロセスの置換その 2 (catsk3_2.c)
 - 4.3.8. exec 後のプロセスの状態
 - 4.3.9. その他の exec 系処理
 - 4.3.10. 課題プロセスの置換その 3 (catsk3_3.c)
 - 4.3.11. exit() コールについて
 - 4.3.12. ゾンビプロセスの確認
 - 4.3.13. プロセス複製の限界

5. シグナル

- 5.1. シグナルの概要
- 5.2. シグナルの種類
- 5.3. シグナルによる動作
 - 5.3.1. signal システムコール
 - 5.3.2. シグナルハンドラ
 - 5.3.3. 自プロセスにシグナル送信
 - 5.3.4. 他のプロセスにシグナルを送信 (親 子プロセス)
 - 5.3.5. 子プロセスの終了シグナルを受け取る

6. プロセス間通信 (IPC)

- 6.1. 名前無しパイプ (パイプ機能)
 - 6.1.1. パイプとは
 - 6.1.2. pipe システムコールによるパイプ

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 改定日付	'09/12/20 '09/01/10 T.S. T.S.

ドキュメント名	組込みソフトウェア科 第4システム応用課題	カリキュラム	組込みソフトウェア科(一般)
スロットマシンを作ろう！ 簡易設計書		作成日付 改定日付	'09/12/20 '09/01/10 T.S. T.S.

- 6.1.3. パイプのテスト (catex5_1.c)
- 6.1.4. 親子間のプロセス通信 (catex5_2.c)
- 6.1.5. 親子間のプロセス通信その2 (execされた子プロセス) (catex5_3.c)
- 6.2. 名前つきパイプ (FIFO 機能)
 - 6.2.1. FIFO について
 - 6.2.2. FIFO の生成
 - 6.2.3. FIFO を使ったプロセス通信実習 (catex5_5w.c と catex5_5r.c)
- 6.3. System V 系の IPC
 - 6.3.1. 共有メモリ
 - 6.3.2. 共有メモリを使ったプロセス間通信実習 (catex5_6w.c と catex5_6r.c)

7. スレッド

- 7.1. スレッドを利用したアプリケーション
 - 7.1.1. スレッドの利点と欠点
 - 7.1.2. スレッドの生成とスレッド終了の待ち合わせ
 - 7.1.1. スレッドの同時実行とスレッドのキャンセル
- 7.2. 課題 catsk6_1.c
- 7.3. 課題 catsk6_2.c
- 7.4. 課題 catsk6_2.c
- 7.5. 課題メニュー表示用のプログラム LED_SW3.c

8. フレームバッファ

- 8.1. 事前準備
- 8.2. フレームバッファとは
- 8.3. デバイスに対するリクエスト ioctl コール
- 8.4. 描画手順
- 8.5. サンプルプログラム (STN0.c)
 - 8.5.1. 課題 GDB によるデバッグ (STN0.c & STN01.c)
 - 8.5.2. 課題 STN02.c
 - 8.5.3. 課題 STN03.c
 - 8.5.4. 課題 STN04.c
 - 8.5.5. 課題 STN05.c
 - 8.5.6. 課題 STN06.c
 - 8.5.7. 課題 STN07.c
 - 8.5.8. 応用課題 1 (STN08.c)
 - 8.5.9. 応用課題 2 (STN09.c)

11-2 参考文献

LINUX デバイスドライバ 第3版 カーネル2.6対応
Linux Device Drivers, 3rd Edition: Chapter 5: Enhanced Char Driver Operations
オライリー・ジャパン

<http://www.ogis-ri.co.jp/otc/hiroba/index.html>
<http://www.ogis-ri.co.jp/otc/hiroba/UMLTutorial/index.html>
オージス総研 オブジェクトの広場
UML チュートリアル

UML モデリングレッスン / 同 モデリング入門
日経BP社

組込みプレス Selection 組込みシステムの設計手法
技術評論社

http://www002.upp.so-net.ne.jp/ys_oota/mdp/
デザインパターンの骸骨達

<http://www.netzmafia.de/skripten/unix/linux-daemon-howto.html>
Linux Daemon Writing HOWTO

<http://www.linux.or.jp/JF/JFdocs/RFC-HOWTO-find-oops-location.txt>
[RFC] HOWTO find oops location, v2

Pthread プログラミング
オライリー・ジャパン

ChangeVision Astah* Community
UML 解析ツール (本システムのモデリングに利用)
<http://astah.change-vision.com/ja/index.htm>