Ｈ１９年度成果のまとめ

JASA 設計ワーキンググループ

社団法人
組込みシステム技術協会
Japan Embedded Systems Technology Association

～ 目次 ～

0. はじめに

ソフトウェア設計では今まで様々な設計手法が考案され、用いられてきた。しかしながら、なかなか世の中の標準、あるいは共通になる手法が出てきていないのが現状であろう。これは、一口にソフトウェアと言っても、情報（エンタープライズ）系、組込み（エンベデッド）系、あるいは、アプリケーション、ミドルウェア、オペレーティングシステム、ドライバ、など様々な分野が存在する為とも言える。ある分野で使いやすい手法が考案されても、別な分野では全く使えないような場合があり、全ての分野に最適な手法が存在しなかったとも言えるのではないだろうか。

近年、ソフトウェアの規模は年々巨大化し、２０～３０年前には数万ステップと言われた組込み系ソフトウェアも今や百倍を越す規模に迫ろうとするシステムまで出てきている。また、ソフトウェアの開発量が増加するに従って、開発するエンジニアが不足しても、分野をまたがると設計手法が異なってしまう為、ソフトウェアエンジニアの転換が妨げられている。たとえ分野が同じでも異なる設計手法で設計されたソフトウェアであると、再利用がされ難い状態にある。

このような状況の中、日本では１９９５年にそれまで限られた分野でしか使われていなかった状態遷移設計に関して、ＪＩＳ　Ｘ０１３１「ソフトウェアの状態遷移の構成及びその表記方法」が制定された。このＪＩＳは当時の世界標準ＩＳＯ／ＩＥＣ１１４１１を翻訳したものであるが、その後、更に使い易い、あるいは使える分野を広げるような見直しは行われていないようである。

一方、開発現場では同時期にこの状態遷移設計手法の欠点である追加、修正が大変であることを克服して使いやすくする為にツールにして、この手法を広める動きが出てきていた。しかしながら、このようなツールも技術標準とは独立して進化を始めた為に、共通な設計手法になっていないのが現状である。

今回、本設計WGでこのような現状を整理して、状態遷移設計の欠点を改善、改良されてきている手法をより幅広いソフトウェア開発分野に広げられるよう、提案して行きたいと考え、活動を開始した。具体的には、設計の考え方や設計手順を示し、使用する用語を統一し、具体的なサンプルなどを提供していく。また、開発スピードやメンテナンス性を落とさない為にツール利用を促進し、ツールへのフィードバック提案も行っていく。

これを機会に、少しでもソフトウェア設計に携わるエンジニアが本設計手法に興味を持ち、実際の開発に使う事によって、本設計手法が世の中に広がると共に、日本のソフトウェアエンジニアがより高度で先端的な開発に着手でき、国際競争力が高まる事を期待したい。

1. 背景

組込み系のシステムにおけるソフトウェアの規模が増大すると、リアルタイム制御システムで必要な「事象（イベント）」、「状態（ステート）」、「動作（アクション）」、「遷移（トランジション）」をビジュアルにわかりやすく表現できることが必要になってくる。

状態遷移表を設計に使用するメリットは、モデル設計している段階で「事象」、「状態」、「動作」、「遷移」の漏れや矛盾を把握することができることである。すなわち状態遷移表ベースに

よるテストケースの抽出を容易に行うことができるので、設計段階でモデルの検証が容易となる。

設計手法はプログラム言語に依存しないのが理想であり、同じ設計手法で開発されたソフトウェアがソースコードになる前に設計評価が行われて上流で品質が確保される。さらにツールによるソースコード自動生成機能によって各種言語に変換されれば、ソフトウェアの品質やメンテナンス性も上がり、再利用が盛んになり、トータルの開発スピードがアップすることになる。

このようなことから、状態遷移表による組込みソフトウェアの設計は有用であることは明らかではあるが、その表記方法が統一されていないことや使い方が標準化されていないことにより、実際の開発現場では広く活用されているとは言えない。

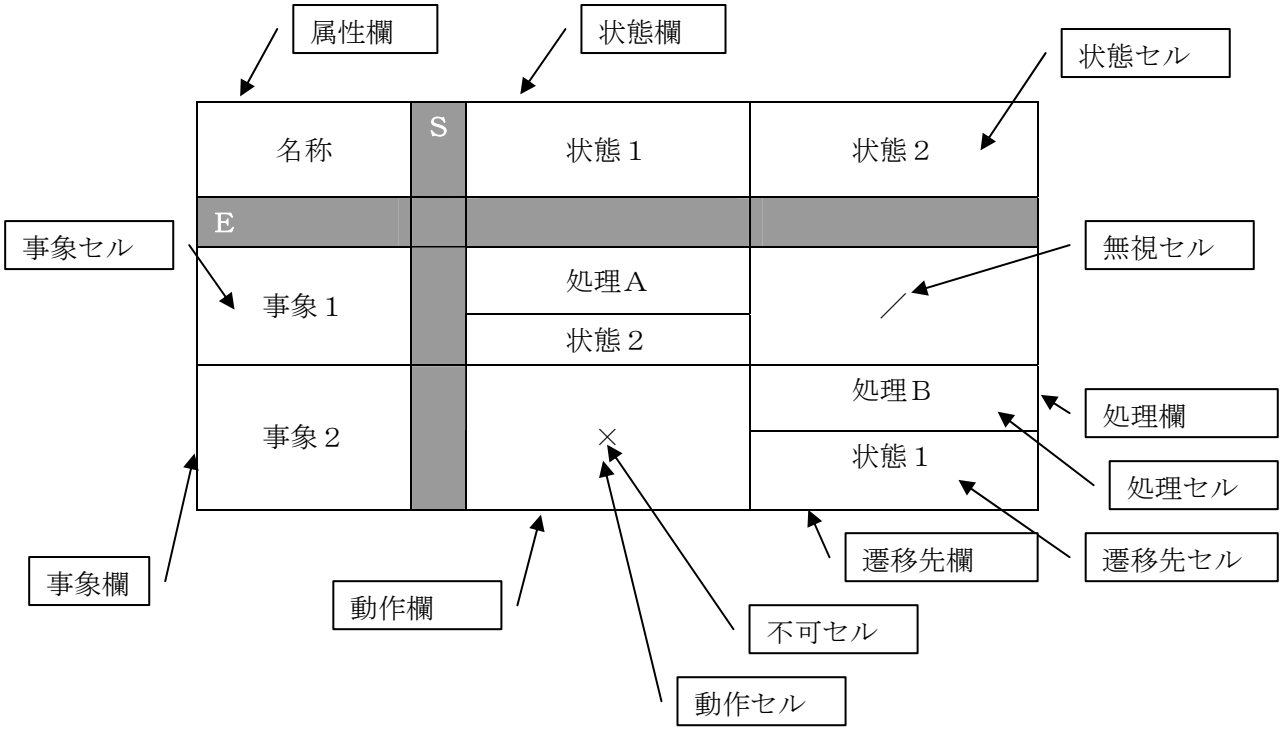
このことは巻末に示す当設計ワーキングのメンバーが ET2007 で実施したアンケートの結果にも現れており、「6.6.1 使用している表記法（設計書）について」で、未だに状態遷移図、フローチャートが上位を占めているにもかかわらず、状態遷移表は 4 位となっている。また、「6.6.2 今後採用してみたいと考えている表記法（設計書）について」で、UML に続いて状態遷移表が挙がっている点も見逃せない。

2. 基本編

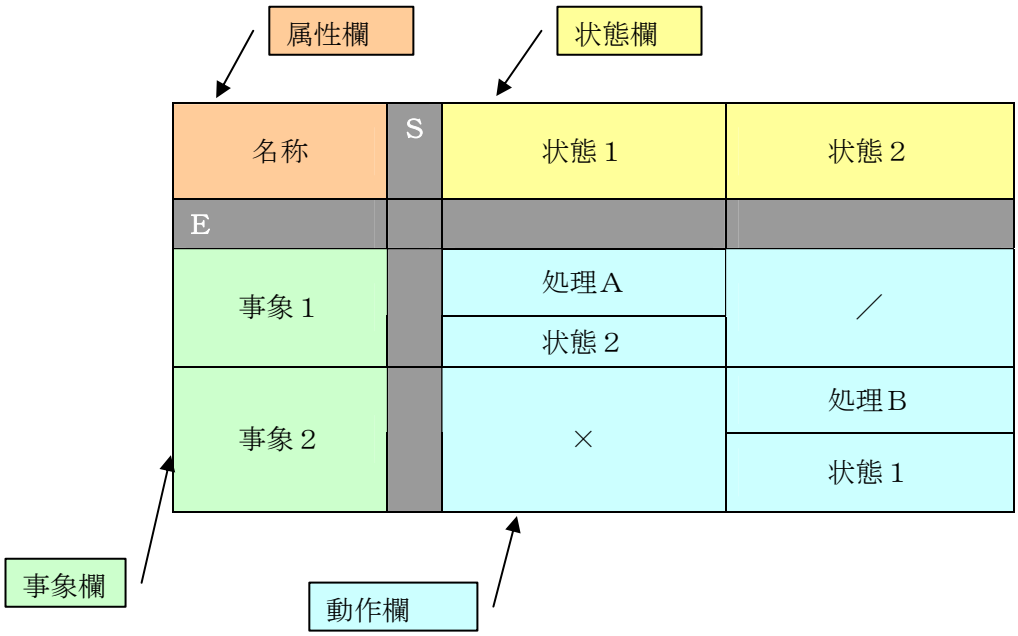
この章では、状態遷移表の表記法および名称について定義する。表記法が複数ある場合は、どちらが良いか議論した理由を明記する。本文の説明では、複数の表記法を示した後、最初に記載したもので表記を統一する。

2.1 表記法について

以下に、基本的な状態遷移表の表記方法を示す。

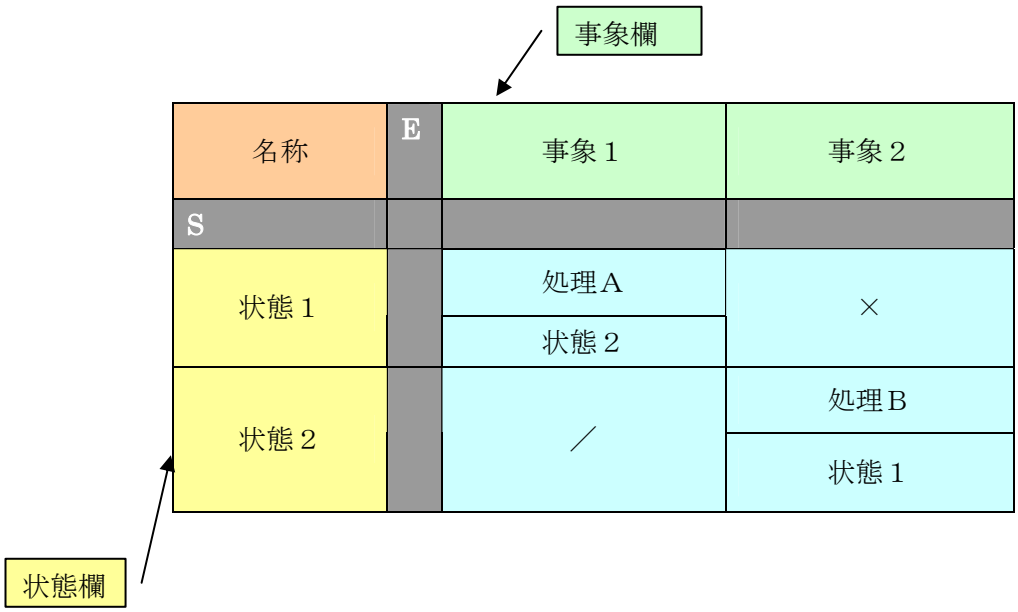


状態遷移表は、大きく４つの欄（「属性欄」、「状態欄」、「事象欄」、「動作欄」）に分類される。



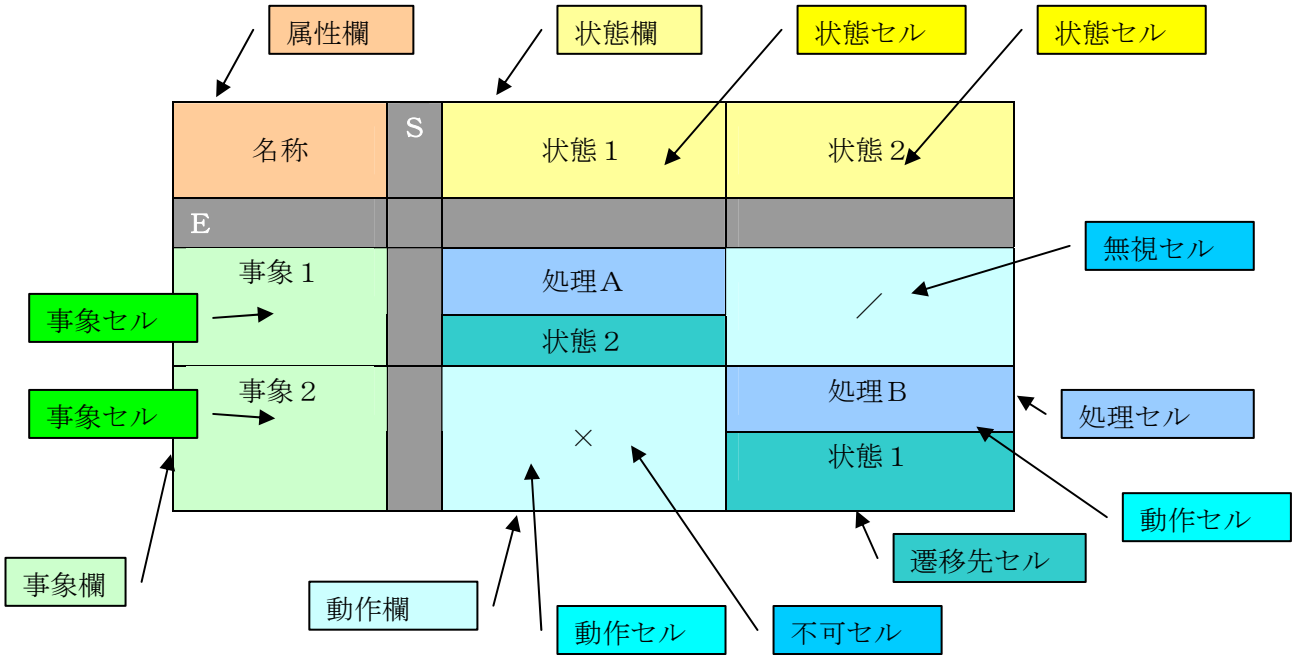
「状態欄」と「事象欄」の各セル数によって、「状態欄」と「事象欄」の行と列を入れ替えても良い。ただし、設計書全体で表記は統一されるべきである。

＜行と列が逆の表記例＞



「属性欄」は、状態遷移表を機能分類単位にする場合や階層構成にする場合を識別するため、“名称”を利用する。このため、設計書全体でユニーク(一つ)になるように定義する。

※階層構成などの表記法は、次年度で議論した内容をまとめた“応用編”で解説する。



「状態欄」は、「状態セル」をまとめた領域である。「状態セル」は、状態を示すマス目の一つを表す。

「事象欄」は、「事象セル」をまとめた領域である。「事象セル」は、事象を示すマス目の一つを表す。

「動作欄」は、「動作セル」をまとめた領域である。「動作セル」は、動作を示すマス目の一つを表す。「動作セル」には、「処理セル」と「遷移先セル」が内在する場合と、「不可セル」(“×”表記）または「無視セル」(“／”表記）で示す場合がある。

「不可セル」は、不可(“Invalid”)を指しプログラム中でありえない事象と状態の組合せを示す。この動作セルに遷移する場合は、設計不具合であることを示す。

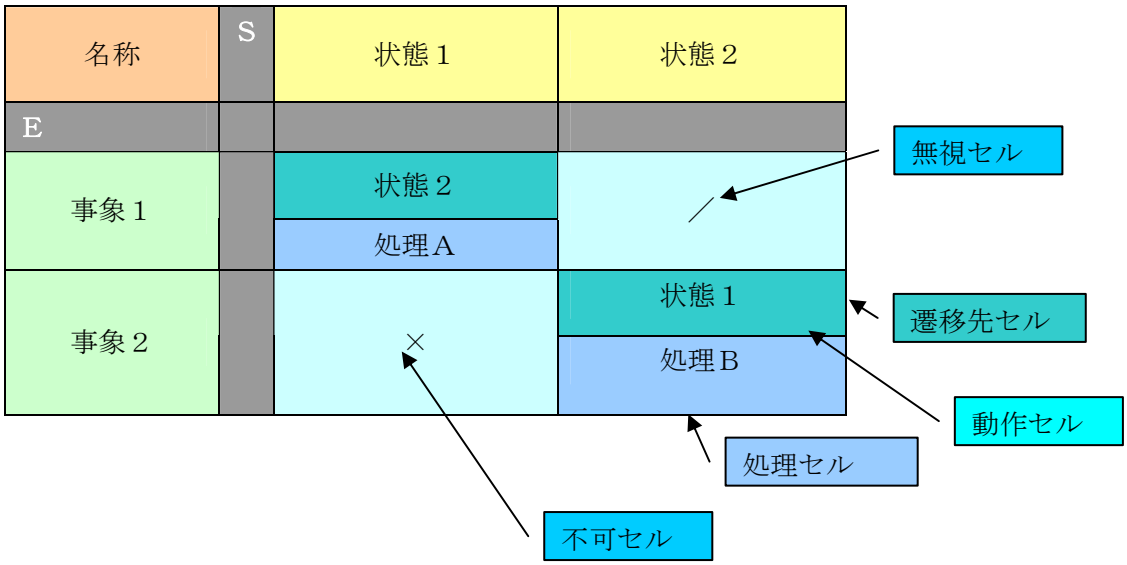
「無視セル」は、無視(“don’t care”)を指しプログラム中で発生するケースが存在するが、何も処理せず、遷移もしないことを示す。

「処理欄」は、「処理セル」をまとめた領域である。「遷移先欄」は、「遷移先セル」をまとめた領域である。

※「動作セル」内の「処理セル」や「遷移先セル」の分割についての表記法は、次年度で議論した内容をまとめた“応用編”で解説する。

表記法によっては、「処理セル」と「遷移先セル」の上下が逆になっている場合もあるが、以下に示す設計ポリシーに従って変更すると良い。ただし、設計書全体で表記は統一されるべきである。

＜上下が逆の表記例＞



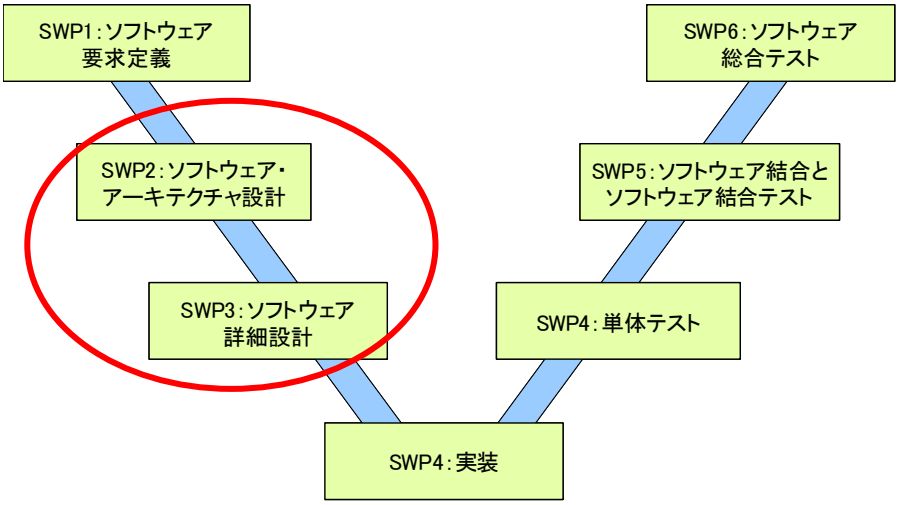
・ 設計ポリシー

状態および事象から遷移先を先に設計する手法を用いた場合、上記のように「動作セル」に記載する「遷移先セル」と「処理セル」を上下逆にすると見易くなる。

後に、「処理セル」内の処理設計は、「遷移先セル」間で調整することも可能である。

3. 設計への適用

以下に組み込みソフトウェア向け開発プロセスガイド（ESPR）に記載されている V 字開発プロセスを示す。



状態遷移設計は、主にソフトウェア・アーキテクチャ設計とソフトウェア詳細設計に適用する。

ここでは、ユーザ操作のあるシステムの設計へ適用する場合について具体的に説明する。

3.1 ソフトウェア要求定義

要求仕様の列挙と操作シナリオの作成を行う。

3.1.1 要求仕様の列挙

要求仕様の列挙では、これから開発するシステムがどのようなものなのか、システムの持つ機能を明らかにし、要求仕様の洗い出しを行う。

最初に以下のようなシステムの制約を明確にする。

・システムの概観
・ハードウエアの構造
・ユーザインターフェース
・制御対象
・制限事項

これらの制約を考慮しながら、システムがユーザに提供する機能という視点から要求仕様を列挙していく。

3.1.2 操作シナリオの作成

次に、要求仕様のなかで「～できる」「～する」といった機能を実現する手順を明らかにし、ユーザの操作に対するシステムの応答・振る舞いを明確にする。

操作の内容は、目的レベルで検討し、ユーザインターフェースが決まっている場合は具体的な操作内容でシナリオを作成する。

操作シナリオは操作の単位ごとに、ユーザの操作に対するシステムの応答・振る舞いを「～すると、～になる」のような形式で書き出す。

操作シナリオはひとつだけでなく、異なる操作シナリオをいくつか作成したほうがシステムの振る舞いを正確に表すことができる。

3.2 ソフトウェア・アーキテクチャ設計およびソフトウェア詳細設計

以下の作業はソフトウェア・アーキテクチャ設計とソフトウェア詳細設計の各々のプロセスで行うが、今回の基本編ではまとめて説明する。

手順として、事象・状態の洗い出しと正常系の状態遷移図の作成、異常系を含む状態遷移表の作成の順に行う。

3.2.1 事象・状態の洗い出し

作成した操作シナリオから、以下のように「事象」や「状態」の洗い出しを行う。

「事象」とは外部からの刺激を示し、操作シナリオにおける「～すると」の部分に相当するため、ユーザの操作を「事象」として抽出するが、事象名としてはユーザが何をしたいかが表された名称とするようにする。

「状態」とは過去の事象の発生を記憶するために用意された棚を示し、ユーザの操作によりシステムが移り変わる状況を「状態」として抽出する。

ユーザ操作以外にも、システム内部のなんらかの「きっかけ」（例えばセンサ）によりシステムが移り変わる場合も同様にして「事象」と「状態」を抽出する。

次に、抽出された「事象」と「状態」を分かり易く関連づけるため、状態遷移図の作成を行なう。（状態遷移表に慣れた設計者であれば状態遷移表を作成してもよい。）

3.2.2 正常系の状態遷移図の作成

作成した操作シナリオと洗い出した「事象」「状態」から、状態遷移図を作成しシステムの全体像を確認する。

初期状態を黒丸（●），事象を遷移線（→），状態を箱で表し、操作シナリオに従って書き出し、全体の流れを検討する。遷移先を検討した結果、処理の内容によりシステム内部の状態を分割したほうが分かりやすくなることもある。

なお、ここでは正常系の状態遷移のみを検討する。異常系を含むと状態遷移図の可読性が落ち、状態遷移図のメリットが失われる可能性がある。

3.2.3 異常系を含む状態遷移表の作成

作成した正常系の状態遷移図から、状態遷移表を作成する。

「2.1 表記法について」に示すとおり、「事象欄」の「事象セル」に洗い出した「事象」を、「状態欄」の「状態セル」に洗い出した「状態」を書き出し、状態遷移図の遷移線をもとに「動作欄」に処理内容と遷移先を書き出していく。

状態遷移図をもとに作成した状態遷移表では、空白の動作欄が残ることがある。この空白の動作欄が、仕様の「抜け」，「漏れ」を示しているため、空欄の処理を検討し、処理が必要な欄，処理の必要のない欄，遷移が起こらない欄に分類し、空白の欄がなくなるようにする。

また、異常時の「事象」「状態」を追加する。

4. 例題

キッチンタイマを例として説明する。

4.1 ソフトウェア要求定義

要求仕様の列挙と操作シナリオの作成を行う。

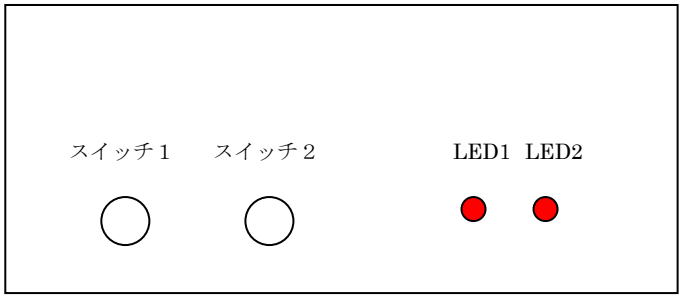
4.1.1 要求仕様の列挙

まず、要求仕様をまとめる。

システムの制約は、入力がスイッチ 2 個、表示が LED2 個とする。

【仕様】

- スイッチ１で時間を設定し、スイッチ２で計時を開始する
- 設定時間の表示は２進数で表示し、設定時間は１～３分とする
- 計時中は、LED を交互に点滅する
- 設定時間経過後 LED を３秒間同時点滅してユーザに通知する
- 計時中にスイッチ２を押すと一時停止する
- 計時中にスイッチ１を押すと計時を停止して待機する



4.1.2 操作シナリオの作成

要求仕様から操作シナリオを作成する。

【操作シナリオ】

１．キッチンタイマを起動する

プログラムを起動すると LED は全消灯して、操作待ちになっている。

２．時間を設定する

スイッチ１を押して時間を設定する。設定時間は LED で２進数表示される。

スイッチ１を押すたびに０，１，２，３，０のように時間設定が循環する。

３．計時を開始する

スイッチ 2 を押すとカウントダウンを開始して、LED が交互に点滅する。

４．一時停止する

計時中にスイッチ 2 を押すと一時停止し、LED が消灯する。

５．時間が経過してアラーム表示になる

設定時間が経過すると LED が３秒間全点滅した後、消灯し、

操作待ちになる。

4.2 ソフトウェア・アーキテクチャ設計およびソフトウェア詳細設計

4.2.1 事象・状態の洗い出し

状態の洗い出し

・待機

・数値入力

・カウントダウン

・一時停止

事象の洗い出し

・時間設定：スイッチ１の操作

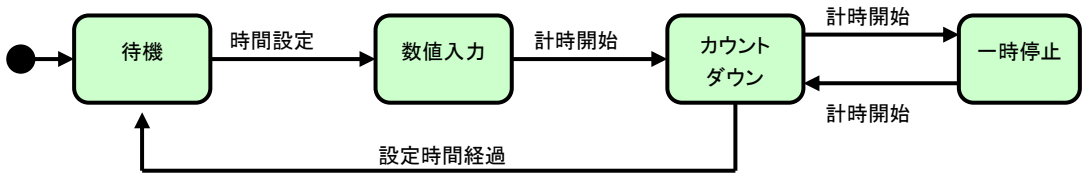
・計時開始：スイッチ２の操作（一時停止を含む）

・時間経過

4.2.2 正常系の状態遷移図の作成

操作シナリオに従って正常系の状態遷移図を作成する。

【状態遷移図】



4.2.3 異常系を含む状態遷移表の作成

状態遷移図から状態遷移表を作成すると、下図のようになり、空白の部分があることが分かる。この部分が仕様書や状態遷移図に記載が無い部分であり、漏れや抜けなのである。

【状態遷移表】(遷移先を上に書いた例)

□0 Timer	S	待機	数値入力	カウントダウン	一時停止
E		0	1	2	3
時間設定	0	数値入力 数値表示を更新する	- 数値表示を更新する	①	②
計時開始	1		カウントダウン 計時を開始する	一時停止 計時を停止する	カウントダウン 計時を開始する
設定時間経過	2			待機 時間経過を伝える	

仕様として、あり得ない所は不可（×）とし、何もしないところは無視（／）としても、まだ、①や②の部分の仕様が不明確である。

このように、状態遷移表を用いると仕様の不備を容易に発見することが出来る。

①、②の部分に、計時を停止して待機へ遷移すると記入すれば、以下のように状態遷移表が完成する。

【状態遷移表】

□0 Timer	S	待機	数値入力	カウントダウン	一時停止
E		0	1	2	3
時間設定	0	数値入力 数値表示を更新する	- 数値表示を更新する	待機 計時を停止する	待機 計時を停止する
計時開始	1	／	カウントダウン 計時を開始する	一時停止 計時を停止する	カウントダウン 計時を開始する
設定時間経過	2	×	×	待機 時間経過を伝える	×

5. 付録

5.1 用語集

属性欄: Department Column
　状態遷移表の名称など属性を書く領域

状態欄: State Column
　状態を書く領域

状態セル: State Cell
　状態欄にある個々の枠内

事象欄: Event Column
　事象を書く領域

事象セル: Event Cell
　事象欄にある個々の枠内

動作欄: Action Column
　操作を書く領域で、遷移先欄や処理欄を含む

動作セル: Action Cell
　操作欄にある個々の枠内

無視セル: Don't Care Cell
　動作セルのうちで無視のセル
　事象と状態の組合せ上、ありえるが処理はしないことを示す。無視は／マークで示される。

不可セル: Invalid Cell
　動作セルのうちで不可のセル
　事象と状態の組合せ上、絶対にありえないことを示す。不可(×）がアクションセルに記述される。

遷移先欄: Transition Column

遷移先を書く領域

遷移先セル: Transition Destination Cell

遷移先欄にある個々の枠内

処理欄: Activity Column

処理を書く領域

処理セル: Activity Cell

処理欄にある個々の枠内

事象: Event

事象とは外部からの刺激です。この刺激に対してどのように反応するかを設計するのが状態遷移表設計。

割込み型事象: Interrupt Event

CPU がプログラムに対して、外部からの刺激があったことをダイレクトに通知するもの。

状態: State

状態とは過去の事象の発生を記憶するために用意された棚

動作セル: Action Cell

特定状態下で特定自称が発生した際に実行される機能または処理。動作セルは事象セルと状態セルのクロスしたセルである。What(機能)を記述するが、How(処理)を記述することもできる。

遷移: Transition

ある状態から別の状態に移ることを示す