修 士 論 文

動的モデル ObTS の
大規模組込みシステム記述に関する研究

指導教官　片山 卓也教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

久保秋 真

1998 年 2 月 13 日

目 次

図 目 次

表 目 次

第 1 章

はじめに

各種の機器に組み込まれてその制御を担うコンピュータシステムのことを，一般的には機器組込みシステムと呼ぶ (以下簡単のために組込みシステムと呼ぶ)．本研究は，組込みシステムの一種である OA 機器の開発現場における「泥臭さ」を減らしたいということが動機になっている．

組込みシステムの例としては，以下のような製品を挙げることができる．

- ビデオ機器や電子レンジなどの民生機器
- 複写機や FAX などの OA 機器
- 交換機などの通信機器
- 自動車や飛行機などの機器制御，管制制御に用いるもの
- 工業用ロボット，生産ライン監視装置など

マイクロプロセッサを始めとする電子デバイスの発展によって，これらの機器は大規模で複雑化なものになっている．また組込みシステムの利用分野も拡大してきている．また，いずれの分野でも多機能化，ユーザーインターフェースの柔軟化などが製品の仕様に大きく影響するようになってきている．さらに，市場での製品寿命が短くなってきており，競争力を維持するためには，いかに短時間で開発しタイミングよく製品を市場に投入できるかということも重要な成功要因になってきている．このような状況に対応するため，組込みシステムの設計者からは，設計やコードの再利用を促す仕組みを求められ，詳細な仕様を記述し検証できるような科学的方法への期待が高まっている．

本研究の目的は，ObTS [12] に基づく分析/設計を計算機上で支援する環境を作成し，これを用いて実用規模の組込みシステムの事例を研究することで，実用規模の問題に対するObTS の有効性を明らかにすることである．

組込みシステムの設計では，オブジェクト指向方法論が盛んに議論される以前から状態遷移図や状態遷移表を非常によく利用している．このことは，組込みシステムの開発現場においてオブジェクト指向をシステムの設計に応用するには，動的モデルが重要であることを示している．そこで，実用規模のシステムのための動的モデルの記述法と，機能を検討する段階でも利用できるような計算機上の支援環境があれば，開発現場においてオブジェクト指向方法論が利用されるようになる機会が生まれるのではないかと考えた．

動的モデルに目を向けてみると，リアクティブシステムの設計 [8] やオブジェクト指向分析/設計論の動的モデルには，Statecharts [8] を採用しているものが多い．Statecharts は状態遷移図に階層構造/並行性/ブロードキャスト通信を持たせて拡張した記述モデルで，システムを記述するときに状態数と遷移数が爆発的に増加するのを抑えている．ObTS は，Statecharts の計算モデルに基づく記述モデルの提案で，動的モデルとオブジェクトモデルの構造を関連付けて，オブジェクトの階層構造でシステムを記述することに特徴がある．オブジェクトごとに状態遷移図とデータを持たせて局所性を強めているので，実用規模のシステムの場合にも可読性が損なわれにくいと考えられている．しかし，ObTS にはStatecharts に対する Statemate [9] に相当するような計算機による支援環境が十分ではなかったため，実用規模のシステム設計についての議論がなされていなかった．そこで，ObTS に基づく分析/設計を計算機上で支援する環境を作成し，これを用いて事例を研究することで，実用規模の問題に対する ObTS の有効性を明らかにしたいと考えた．

本論文は次のように構成する．第 2 章では本研究の背景を述べる．まず本研究の基礎として重要な Statecharts と ObTS について述べ，またその他の関連研究についても触れる．その上で ObTS を本研究の基礎とし仕様記述言語 ObCL を提案した理由を述べる．第 3 章では仕様記述言語 ObCL の概要と ObCL による記述例を示す．第 4 章ではシミュレーション環境 ObML の概要と ObML の利用例を示す．第 5 章では，事例研究として架空の複写機 CP-KT1 の操作部プログラムについて分析を実施し，ObCL による記述と ObML を用いた仕様確認の実行例を示す．第 6 章では ObCL および ObML を評価する．仕様確認環境としての表現力，機能に着目する．最後に，第 8 章に本研究では十分議論で

きていない問題を今後の課題として挙げる．巻末には付録を添付する．付録 A は ObCL の言語仕様，付録 B, C, D, E, F は，事例研究の要求仕様文書と事例研究で得た分析結果やコードである．

第 2 章

背景

本章では，本研究に関連する事項について簡単に説明する．

2.1 Statecharts

リアクティブシステムの設計 [8] やオブジェクト指向方法論の動的モデルで広く採用されているものにStatecharts [8] がある．例えば，Booch 法 [1], OMT 法 [16] などが Statecharts を採用している．

Statecharts は状態遷移図に階層構造/並行性/ブロードキャスト通信を持たせて拡張した記述モデルである．原始的な状態遷移図は状態を平面的に展開するため，システムの記述が複雑になるにつれて状態数や遷移数が爆発的に増加する傾向があるが，Statecharts は階層構造によってこれを抑制している．また，ひとつの状態遷移図の中に並行に動作する状態を記述できるよう，並行動作の表現を追加している．イベント通信には同期的なブロードキャスト通信を採用している．つまり，送信したイベントはシステム全体に同時に伝わり，このときの時系列のとらえ方がステップに区切ったクロック的なものになっている．なお，Statecharts のセマンティクスについては Harel [7] に詳しい．

Statecharts では，データは大域変数として扱いデータ処理は大域変数の副作用で表している．このため，記述の規模が大きくなると大域変数を多用した実用規模のプログラムが抱えるのと同じ問題 [14] が生じる．また，Statecharts の状態の階層はオブジェクトモデルの構造と対応がとれていないため，Statecharts のある状態がどのオブジェクトに関わるのか，Statecharts のある状態遷移と遷移時のアクションをどのオブジェクトが担当

するのかが明確になっていない.

2.2 ObTS

伊藤 [12] は Statecharts の計算モデルに基づく記述モデル ObTS を提案している. ObTS は，記述対象のシステムの構造をオブジェクトの階層構造で表し，システムの動作を状態遷移，関数的な属性計算，属性つきイベントのブロードキャスト通信で表すことでモデル化している.

ObTS のオブジェクトは属性と状態遷移図を持っている．オブジェクトは動作を委譲するために内部オブジェクトを持つことができる．内部オブジェクトも属性，状態遷移図をもつ．また，オブジェクトが最初に実行する状態遷移を初期遷移として定義しておく.

図 2.1 に ObTS のグラフィカルな表記によるシステム記述の概観を示す.

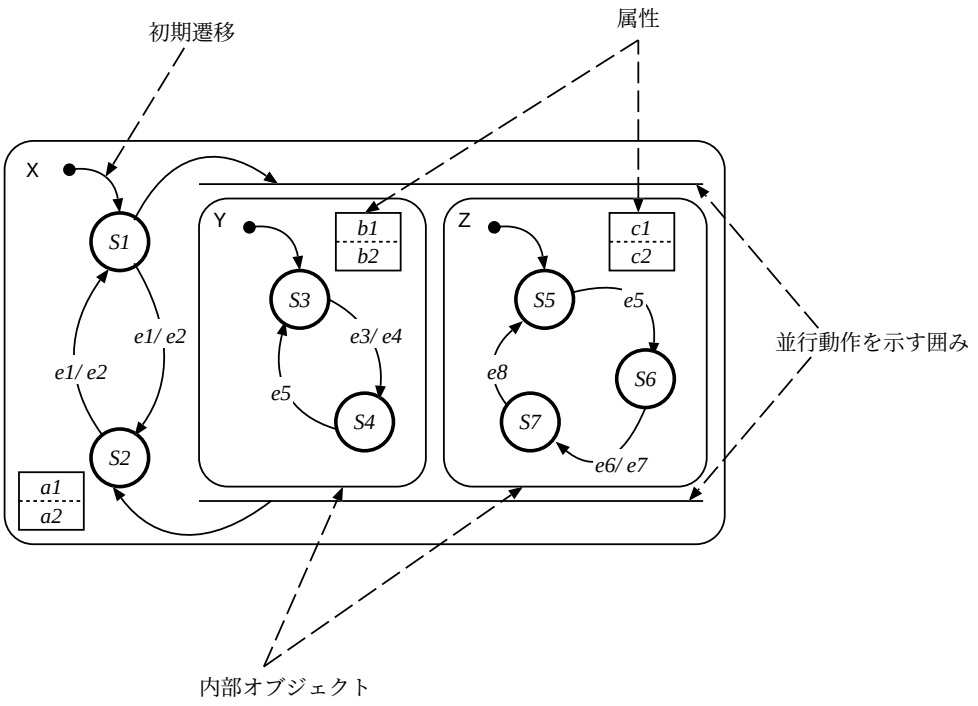


図 2.1: ObTS のグラフィカルな記述例

オブジェクトを角丸の四角，状態を円，状態遷移を矢印付きの弧，初期遷移は黒丸からの矢印付きの弧，属性を四角で記述する.

オブジェクトの階層関係と ObTS の対応を示すために，図 2.2 に図 2.1 のオブジェクト階層構造を OMT 法のモデル図に表したものを示す.

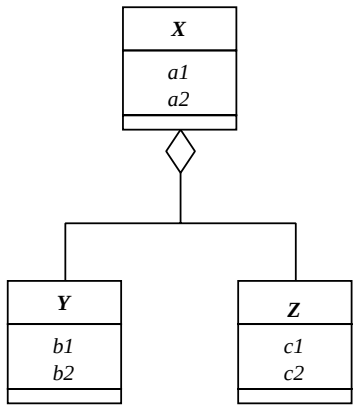


図 2.2: OMT 法の記法による ObTS の記述例に対応するモデル図

ObTS では，データ計算は関数的な属性計算によってなされる．あるオブジェクトの属性は，そのオブジェクトが状態遷移するときに計算する．イベントの属性は，そのイベントを発生する状態遷移において計算する．図 2.3 に ObTS のグラフィカルな表記による状態遷移の例を示す.

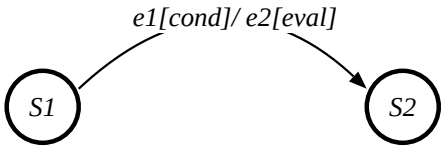


図 2.3: ObTS の状態遷移の例

この例は，状態 S1 で入力イベント e1 を受け取ったとき，条件 cond が成り立っていれば，状態 S2 に遷移し，イベント e2 を出力し eval の中の式を計算する状態遷移を表している.

2.3 その他の関連研究

ObTS 以外にも Objectcharts [2]，ObjChart [5]，O-Chart [6] や Object-Oriented State Machines [17] などの Statecharts に基づく研究が報告されている．

これらの状態遷移図に基づく手法では，クラスの継承をどのように扱うのかが問題となる場合が多いが，例えば Objectcharts の場合には継承は次のような場合に限定することで継承の複雑さを回避している：

- 新しいサービスの状態や遷移規則を追加するとき．
- 遷移の発火条件を弱める/事後条件を強めるとき．
- 不変関係を強化するとき．

Object-Oriented State Machines では，単純なステートマシンから複雑なステートマシンへの派生を許すような構築方法として，単純なステートマシンをインクリメンタルに修正/結合するためのサブクラス化，合成，委譲，総称の技術を紹介している．また C++で実装する場合の実例を用いてこの技術を説明している．

その他に，Statecharts に対してデータの局所性，階層性を導入したモデル AFSM (attributed finite state machines) [3] がある．AFSM では Statecharts と同様の状態の階層化をし，各状態に属性を持たせることでデータの局所性と階層性を導入している．データ計算は関数的な属性計算であり，並行動作部分の通信は属性付きイベントのブロードキャスト通信によっている．

2.4 なぜ ObTS を基礎としたのか

第 1 章において述べたように，組込みシステムの設計ではオブジェクト指向方法論が盛んに議論される以前から状態遷移図や状態遷移表を非常によく利用している．そして近年は，組込みシステムの設計者もオブジェクト指向分析/設計方法論に注目するようになってきている．以上のことは，開発現場においてオブジェクト指向の考え方を組込みシステムの設計に応用するには，動的モデルが重要であることを示している．

ObTS は，状態遷移図にオブジェクトの階層構造を持ち込み，またデータの局所性を強める工夫をしているなど，オブジェクト指向分析/設計方法論の動的モデルとしての有用

な特性を有している．本研究の基礎として ObTS を採用すれば，このような ObTS の特性が組込みシステムの分析/設計の際に有効に作用することが期待できるため，ObTS を本研究の基礎とすることにした．

2.5 仕様記述言語 ObCL の必要性

ObTS には，Statecharts に対する Statemate [9] に相当するような計算機による支援環境が十分ではなかった．そこで，ObTS による分析/設計を計算機上で支援する環境を作成し，これを用いて事例を研究することで，実用規模の問題に対する ObTS の有効性を明らかにしたいと考えた．

まず，ObTS は計算モデルとして与えられているため，システムを記述し，支援環境を用意するために ObTS に基づく具体的な構文が必要である．さらに，本研究の目指す支援環境は規模の大きなシステムの分析を支援することを目標にしているので，再利用のための工夫や実用規模の問題を効果的に記述する特性を有していることが望ましい．そこで，これらの要求をかなえるために仕様記述言語 ObCL を提案する．仕様記述言語 ObCL の言語設計の方針として，記述の解析性を損なわずに実用規模の問題や複雑性の高い問題を効果的に記述できることを掲げた．また，これまでの ObTS の研究の成果として ObTS のセマンティクスの定義，通信モデルについての議論，ObTS の シミュレーション環境である ObML などが得られており，これらの成果を有効に活用したいと考えた．したがって，今回提案する仕様記述言語 ObCL は，クラス/ クラスの継承/ フィールド / イベントクラスなどの特性を備えたマクロ言語的なものとし，ObCL で記述したシステムは ObTS モデルのみに基づくシステム記述に変換可能であるとした．

仕様記述言語 ObCL の特性については，第 3 章で述べる．

第 3 章

仕様記述言語 ObCL

本章では仕様記述言語 ObCL の概要と記述例を示す.

3.1 ObCL の概要

3.1.1 ObCL の設計方針

ObTS は計算モデルとして与えられているため，具体的なシステムを記述できるよう，構文を与える必要がある．また，本研究の目指す支援環境は規模の大きなシステムの分析を支援することを目標にしているので，再利用のための工夫や実用規模の問題を効果的に記述する特性を有していることが望ましい．そこでいくつかの事例を ObTS で検討した結果，ObTS を現実的な仕様の記述に用いるためには若干の拡張を施すべきであるという結論に達した．その方法としては，ObTS を補強するいくつかのアイディアを追加した仕様記述言語 ObCL を用意する方法を選択した．これは，これまでの ObTS の研究の成果として ObTS のセマンティクスの定義，通信モデルについての議論，ObTS の シミュレーション環境である ObML などが得られており，これらの成果を有効に活用したかったためである.

以上のことから，ObCL は ObTS モデルのみに基づくシステム記述に変換可能な言語とし，ObTS の計算モデルにシステム記述のための構文と，再利用や実用規模のための特性を与えるマクロ言語的なものとなった．また，ObCL が再利用や規模の問題に対応するため備える特性を以下に示す.

システム　システムは，クラス/フィールド/イベントクラスの集まりとして記述する．

クラス　オブジェクトはクラスのインスタンスとみなす．

クラスの継承　振舞いの似たクラス間で仕様記述を共有する．

フィールド　ブロードキャスト通信を制限する．

イベントクラス　イベントに操作を持たせる．

インスタンスの生成と破棄　インスタンスの寿命を定める．

次節以降ではこれらの特性について述べる．なお，ObCL 構文の詳細は付録 A に示す．

3.1.2　システム

ObCL のシステムの記述は，クラス/フィールド/イベントクラス記述と，これらからインスタンス化したオブジェクトの記述からなる．

以下はクラス階層の最上位クラスのサンプルコードである．

```
-- ObCL では大文字と小文字を区別しない.
-- "--" 以降行末までがコメント.
system      -- システム名となるクラス名を書く.
    SystemName
  inner     -- 内部オブジェクトの記述.
      Para1:{ ObjectA:ClassA; ObjectB:ClassB }
  transition
  Start is
    source
        init    -- 初期遷移の場合の定められた記述.
    destination -- 遷移先を書く.
        Para1
  end
end -- SystemName
```

3.1.3　クラス

ObTS ではシステムはオブジェクトの集合と捉えている．しかし，実用規模になると同じ性質あるいは似たような性質のオブジェクトが多数存在しうるので，複数のオブジェク

トを生成するテンプレートしてのクラスが不可欠である．また，仕様記述を再利用するにはクラスの概念を利用するのが自然である．よって，ObCL ではクラスとインスタンスの区別を持ち込むことにする．ObCL による記述中のクラスからインスタンス化したオブジェクトが ObTS のオブジェクトに対応する．

クラスはフィールド宣言/属性/状態/内部オブジェクト宣言/操作/遷移規則の各記述節からなる．以下は ObCL におけるクラス定義の記述を説明したサンプルコードである．

```
class      -- クラス名を書く.
    ClassName
  inherit        -- 継承する場合には親クラス名を書く (省略可).
      ParentClass
  field          -- このクラスが参加するフィールド名を書く.
      FieldName1,FieldName2
  attribute      -- このクラスの属性名を書く (省略可).
      Attr1:Int;  Attr2:String
  state          -- このクラスの状態名を書く (省略可).
      State1,State2
  inner          -- 内部オブジェクト名を書く (省略可).
      Object1:Class1;  Object2:Class2;
      -- 内部オブジェクトが並行動作する場合の記述 (省略可).
      Para1:{  ObjectA:ClassA;  ObjectB:ClassB  }
  transition
  T0  is         -- "T0" が遷移規則名
    source
        init     -- 初期遷移の場合の定められた記述.
    do           -- 遷移のときに実行する属性計算を書く (省略可).
        Attr1  := 0;
        Attr2  := ""
    destination  -- 遷移先の状態名を書く.
      State1
    output       -- 出力イベントを書く (省略可).
        FieldName1.Ev3
  end;
  T1  is
    source       -- 遷移元の状態名を書く.
        State1
    input        -- 入力イベントを書く.
        FieldName1.Ev1
    when         -- 遷移条件を書く (省略可).
```

```
        FieldName1.Ev1.val > 0
    do          -- 遷移のときに実行する属性計算を書く (省略可).
        Attr1 := FieldName1.Ev1.val
    destination -- 遷移先の状態名を書く.
        State2
    output      -- 出力イベントを書く (省略可).
        FieldName2.Ev2
  end
end -- ClassName
```

3.1.4 クラスの継承

振舞いの似たクラス間で仕様記述を共有するために，クラスの記述を継承できるようにしたいと考えた．ここでいう継承とは，記述のテキストそのものを再利用するための工夫を意味している．ObCL では，状態と遷移規則の追加に限定する継承を導入した．この方式は限られた場合にしか利用できないが，実用規模のシステムを想定した場合には理解しやすい簡潔な方式が効果的であると判断し，この方式に決定した．また，Coleman [2] などが状態遷移図の継承方法を扱っているが，いずれも継承できる場合の条件を定めた限定的なものである．

ObCL のクラスの継承では，継承したクラスは継承元の全ての節を引き継ぎ，フィールド宣言/属性/状態/操作/遷移規則を追加できる．以下は ObCL におけるクラス継承の記述を説明したサンプルコードである．

```
class
    Parent      -- 親クラス
  field
      -- この例では フィールド Fd1 に
      -- イベント Ev1,Ev2,Ev3 が宣言されているとする.
      Fd1
  attribute
      Attr1:Int; Attr2:String
  state
      State1
  transition
  T0 is
    source
        init
```

```
    do
        Attr1 := 0;
        Attr2 := ""
    destination
      State1
    output
        Fd1.Ev1
  end
end -- Parent

class
    Child      -- 子クラス
  inherit
      Parent -- クラス Parent の記述に以下の記述を追加することを表す.

      -- ここに field, attribute 記述を追加すれば,
      -- 親クラスの記述とともに利用できる.
  state
      -- 追加した状態は親クラスの状態とともに存在する.
      State2
  transition
  -- 追加した遷移規則は親クラスの遷移規則とともに利用できる.
  T2 is
    source
        State1
    input
        Fd1.Ev2
    destination
        State2
    output
        Fd1.Ev3
  end
end
```

3.1.5 フィールド

ObTS のイベント通信は Statecharts と同様にブロードキャスト通信である．しかし，現実のシステムではより関連の深い特定のオブジェクト同士がイベント通信をすることが多い．またシステムの規模が大きいときには，イベントの波及範囲が明確で限定されてい

る方が安全でわかりやすい．そこで，ObCL ではシステムグローバルなブロードキャスト通信を避けて，関連を持つクラス同士のイベント通信に限定するために，フィールドという概念を導入した．

図 3.1 にイベントがフィールドを伝わる様子を示す．

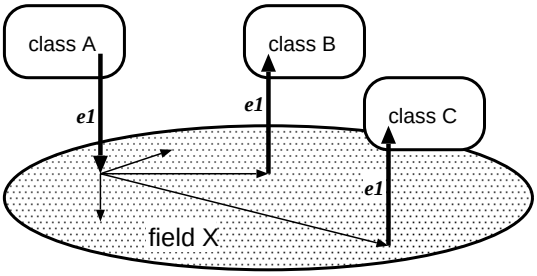


図 3.1: イベントがフィールドを伝わる様子

1. クラス A,B,C はフィールド X に所属する．
2. いまクラス A の出力イベントが $e1$ であったとすると，イベント $e1$ はフィールド X に送られる．
3. フィールド X はイベント $e1$ を自分のフィールドに所属するクラスにブロードキャストする．
4. クラス B,C はイベント $e1$ を受け取る．

以下は ObCL におけるフィールドの記述を説明したサンプルコードである．

```
field
    Fd1    -- フィールド名を書く.
event
    -- イベントインスタンス名：イベントクラス名 の形式で書く.
    in : Input_Event;
    out : Output_Event
end -- Fd1

-- クラスの記述中でフィールド中のイベントを参照するには,
    Fd1.in
    Fd1.out
-- のように書く.
```

3.1.6 イベントクラスとイベント

ObTS のイベントは属性を持つが，ObCL ではこの属性に対する操作を記述できるようにし，イベントの属性には操作を通じてのみアクセスできるように定めた．それ以外にもイベントにできる操作はイベントに持たせておくことにする．これによって，イベントの属性の評価方法がイベントを受け取るクラスによって異なるという事態を避けることが可能になる．そして記述の再利用のためにイベントクラスを用意し，イベントはそのインスタンスして扱う．属性を持たない単純なイベントは，基本イベントクラス (GENERIC_EVENT) を用意し，このクラスのインスタンスとして定義する．イベントクラスの継承は，基本イベントクラスを継承したイベントクラスを定義して属性/操作を追加する．

以下は ObCL におけるイベントクラスの記述を説明したサンプルコードである．

```
event  -- イベントクラス名を書く.
    Sample_Event
  inherit  -- 継承する場合は，親イベントクラスの名前を書く.
      generic_event
  attribute      -- イベント属性名：イベント属性の型 の形式で書く.
      attr1 : Int;
      attr2 : String
end -- Sample_Event
```

3.1.7 インスタンスの生成と破棄

ObCL ではシステム全体は各クラスのインスタンスの集約関係で表す．内部のインスタンスは，クラスのインスタンスを生成するときにクラス記述中の内部オブジェクト記述にしたがって生成する．この仕組みによって，クラス階層の最上位のクラスのインスタンスを生成から，連鎖的にシステム全体のインスタンスの生成がなされる．ObTS では各オブジェクトの寿命はシステム全体のそれと同一であるので，ObCL も原則としてシステム初期化時以外にインスタンスを動的に生成/破棄するメカニズムは提供しない．

3.1.8 均質な複数オブジェクトの扱い

動的に要素数の変わる均質なインスタンスの集まりを表現したい場合には，同じ構造や振舞いを持つインスタンスを集合として構造化し，これらの要素数を指定せずにシステムが初期化できる仕組みを利用できるようにする．この仕組みでは，集合のメンバ (インス

タンス) へのフィールドを通した同報通信や，各メンバからのレスポンスを収集するメカニズムを用意する予定である．

3.2 ObCL による記述例

ObCL を用いた仕様記述の例として，ごく簡単な CD プレーヤーを取りあげる．

3.2.1 CD プレーヤーの仕様

ここで例として取り上げるの CD プレーヤーは，図 3.2 のような外観を持ち，停止/再生/一時停止ができる．3 つのボタンとターンテーブルと LCD を持っていて，次のように振舞う．

1. 最初 CD プレーヤーは LCD に停止のメッセージを表示して停止している．
2. 再生ボタンを押すと再生を始め，LCD に再生中のメッセージを表示する．
3. 再生中に停止ボタンを押すと停止する．
4. 再生中に一時停止ボタンを押すと，LCD に一時停止中のメッセージを表示して一時停止する．
5. 一時停止中に一時停止ボタンを押すと再生する．
6. 一時停止中に停止ボタンを押すと停止する．

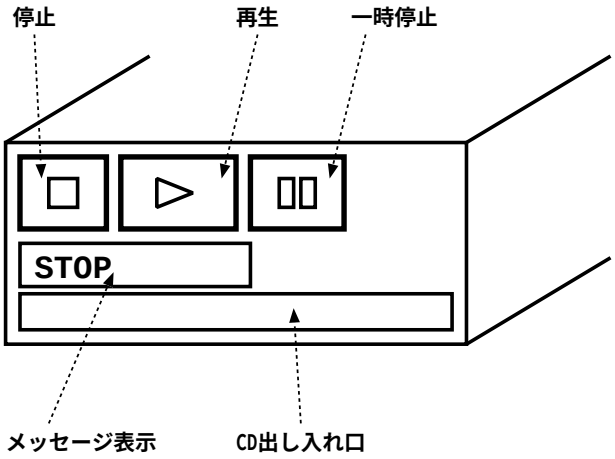


図 3.2: CD プレーヤーの操作部

3.2.2 CD プレーヤーのオブジェクトモデル

CD プレーヤークラスは，各々のボタン/ターンテーブル/LCD のクラスを集約したものとしてモデル化できる．図 3.3 に OMT 法の記法に準じた記法でオブジェクトモデルを示す．図には 3.1 節で説明したフィールドを直感的に理解できるよう追記してある．また，図 3.4 に ObTS のグラフィカル表記で表したでオブジェクトモデルを示す．

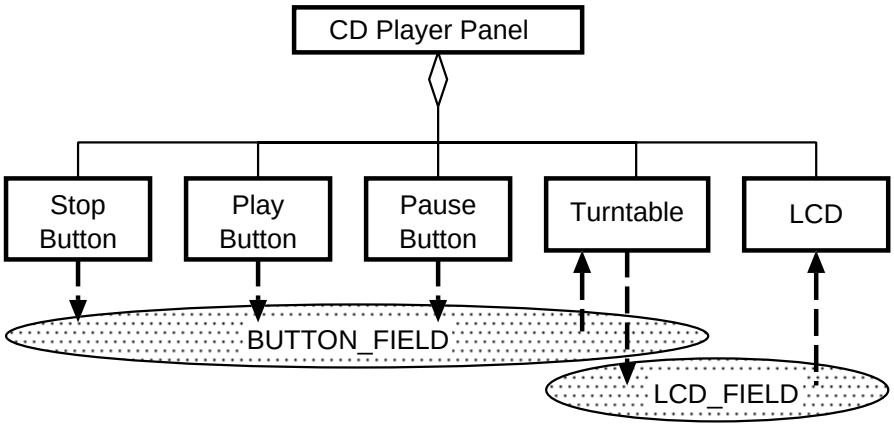


図 3.3: CD プレーヤーのオブジェクトモデル

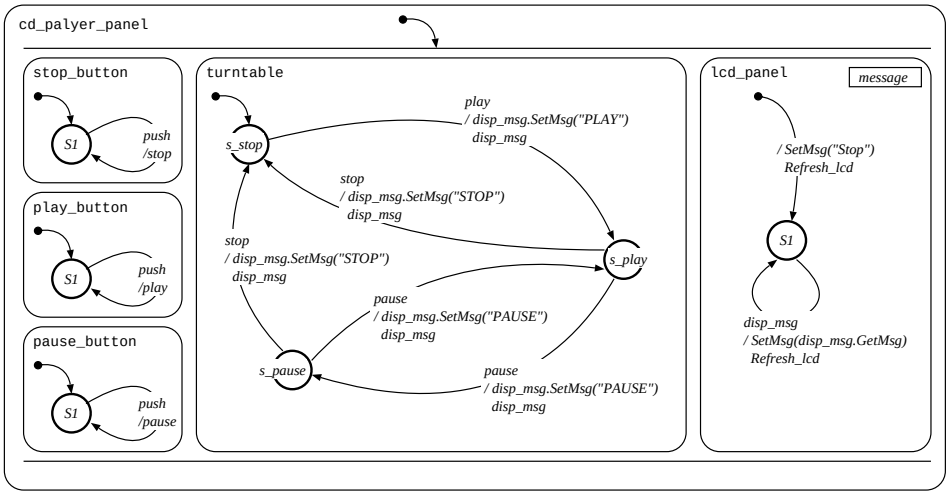


図 3.4: CD プレーヤーの ObTS によるモデル図

3.2.3 CD プレーヤーの ObCL コード

3.2.3.1 CD プレーヤーのイベントとフィールドの定義

まず GENERIC_EVENT を継承して DISPLAY_MSG_EVENT クラスを定義する．DISPLAY_MSG_EVENT クラスを定義するコードは図 3.5 のようになる．

```
event
    DISPLAY_MSG_EVENT
inherit
    GENERIC_EVENT
attribute
    message
operation
    SetMsg(msg) is
      do
10        message := msg
      end;
    GetMsg is
      do
        Result := message
    end
end -- DISPLAY_MSG_EVENT
```

図 3.5: DISPLAY_MSG_EVENT クラスの定義

BUTTON_FIELD, LCD_FIELD の各フィールドを定義した記述コードを図 3.6 に示す．

ボタンとターンテーブルの間にフィールド BUTTON_FIELD を設定する．BUTTON_FIELD には GENERIC_EVENT である push, stop, play, pause イベントを定義している．ターンテーブルと LCD の間にはフィールド LCD_FIELD を設定する．LCD_FIELD には DISPLAY_MSG_EVENT クラスの disp_msg イベントを定義している．

このように，フィールドによってイベントをブロードキャストせずに利用範囲を限定して記述できるようになっている．

```
field
    BUTTON_FIELD
event
    push : GENERIC_EVENT;
    stop : GENERIC_EVENT;
    play : GENERIC_EVENT;
    pause : GENERIC_EVENT
end -- BUTTON_FIELD

field
    LCD_FIELD
event
    disp_msg : DISPLAY_MSG_EVENT
end -- LCD_FIELD
```

図 3.6: BUTTON_FIELD，LCD_FIELD の定義

3.2.3.2 CD プレーヤーのクラスの定義

停止/再生中/一時停止の状態を持つターンテーブルクラスを例にあげる．図 3.7 がクラス TURNTABLE_CLASS の ObTS によるモデル図である．

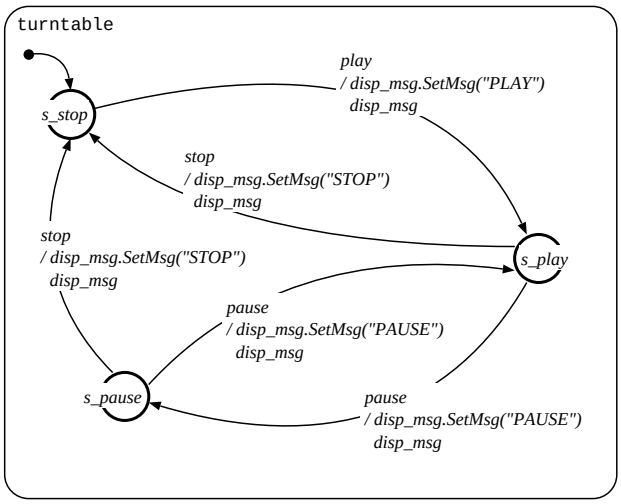


図 3.7: TURNTABLE_CLASS の ObTS によるモデル図

図 3.3 に示したようにクラス TURNTABLE_CLASS は フィールド BUTTON_FIELD と
LCD_FIELD に属しており，BUTTON_FIELD から stop, play, pause イベントを受け取ると遷移が起こり，disp_msg イベントを LCD_FIELD へ送る．

クラス TURNTABLE_CLASS のクラス定義の記述コードを図 3.8 に示す．

図 3.8 のコード中で，transition 節が遷移規則を記述する節である．遷移規則 T1 の source 節に記述した init が初期遷移を示している．遷移規則 T2 の input 節には BUTTON_FIELD からの入力イベント play が，output 節には LCD_FIELD への出力イベント disp_msg が記述してある．

もうひとつの例としてストップボタンをあげる．ストップボタンは BUTTON_FIELD から push イベントを受け取ると，stop イベントを BUTTON_FIELD へ送る．

ストップボタンを記述したクラス STOP_BUTTON_CLASS の定義の記述コードを
図 3.9 に示す．

```
class
    TURNTABLE_CLASS
field
    BUTTON_FIELD, LCD_FIELD
state
    s_stop, s_play, s_pause
transition
    T1 is
      source
        init
      do
        LCD_FIELD.disp_msg.SetMsg("STOP")
      destination
        s_stop
      output
        LCD_FIELD.disp_msg
      end;
    T2 is
      source
        s_stop
      input
        BUTTON_FIELD.play
      do
        LCD_FIELD.disp_msg.SetMsg("PLAY")
      destination
        s_play
      output
        LCD_FIELD.disp_msg
      end

    -- (以下同様につき省略)

end -- TURNTABLE_CLASS
```

図 3.8: TURNTABLE_CLASS の定義

```
class
    STOP_BUTTON_CLASS
field
    BUTTON_FIELD
state
    S1
transition
    T1 is
      source
        init
      destination
        S1
      end;
    T2 is
      source
        S1
      input
        BUTTON_FIELD.push
      destination
        S1
      output
        BUTTON_FIELD.stop
      end
end -- STOP_BUTTON_CLASS
```

図 3.9: STOP_BUTTON_CLASS の定義

3.2.3.3 CD プレーヤーのクラスの継承

初期版のクラス TURNTABLE_CLASS (図 3.7) を継承して再生中の早送り機能を追加した改良版を例にあげる.

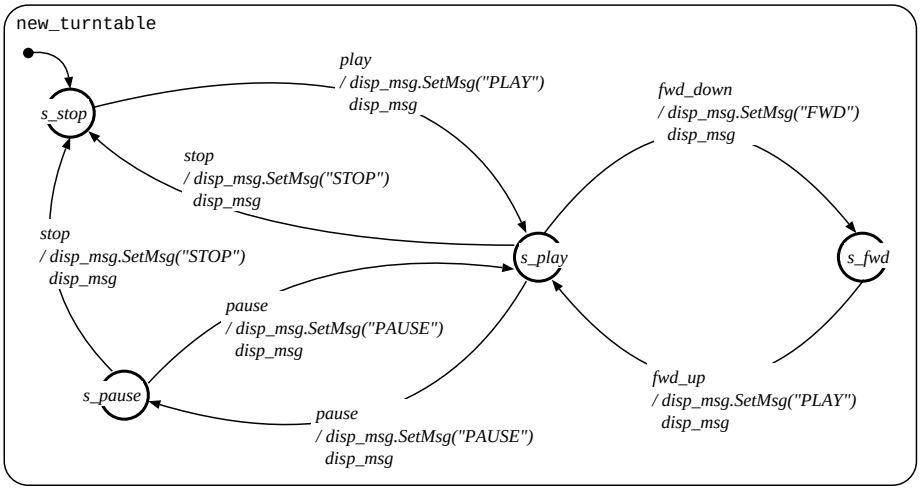


図 3.10: NEW_TURNTABLE_CLASS の ObTS によるモデル図

図 3.10 に示すように，NEW_TURNTABLE_CLASS は，状態 s_fwd とイベント fwd_down, fwd_down による遷移を TURNTABLE_CLASS に追加したものである. これらは TURNTABLE_CLASS の遷移規則と干渉していないので, NEW_TURNTABLE_CLASS は TURNTABLE_CLASS から継承できる.

図 3.11 に NEW_TURNTABLE_CLASS のコードを示す.

```
class
    NEW_TURNTABLE_CLASS
inherit
    TURNTABLE_CLASS
      -- TURNTABLE_CLASS に以下の記述を追加することを表す.
state
    s_fwd
transition
    Tfd is
      source
        s_play
      input
          BUTTON_FIELD.fwd_down
      do
          LCD_FIELD.disp_msg.SetMsg("FWD")
      destination
          s_fwd
      output
          LCD_FIELD.disp_msg
      end;
    Tfu is
      source
        s_fwd
      input
        BUTTON_FIELD.fwd_up
      do
        LCD_FIELD.disp_msg.SetMsg("PLAY")
      destination
        s_play
      output
        LCD_FIELD.disp_msg
      end
end -- NEW_TURNTABLE_CLASS
```

図 3.11: NEW_TURNTABLE_CLASS の定義

第 4 章

ObCL 記述支援環境とシミュレーション環境

本章では ObCL 記述支援環境とシミュレーション環境について説明し、利用例を示す.

4.1 ObCL 記述支援環境の概要

仕様記述言語 ObCL を用いて記述した仕様を計算機上で検討できるよう、ObCL 記述支援環境とシミュレーション環境を構築した。

この環境は,

- ObCL → ObML コンバータ
- ObTS シミュレーション環境 ObML
- ObCL 記述環境

からなる.

4.1.1 ObCL → ObML コンバータ

これは，仕様記述言語 ObCL によって記述した仕様を，ObTS シミュレーション環境 ObML 上で実行できるプログラムコードに変換するものである．コンバータは単独のプログラムで，シミュレーション環境である ObML や ObCL 記述支援環境とは独立して使用できる.

4.1.2 ObTS シミュレーション環境 ObML

シミュレーション環境 ObML の原型は伊藤 [13] によって報告されたものである．ObML は関数型言語 Standard ML で記述されている．また ObTS モデルも ML の関数群として実装されており，シミュレーション機能の関数群と共に動作する．

シミュレーションは，ObTS モデルでのステップ動作にしたがって進行する．シミュレータは，各ステップごとのオブジェクトの動作状態とイベントに関するスナップショットのリスト (システムコンフィギュレーションと呼んでいる) を動作結果として返す．通常，イベント列を与えてそのシミュレーション結果を見る場合は，システムコンフィギュレーションをプリティプリント関数を用いて読みやすいかたちに整形して使用する．後述 4.2.1 節に示した例ではこのようにして得られた出力を使用している．また，仕様の確認のために動作の検査をしたい場合には，システムコンフィギュレーションを検査用 ML 関数に与えて処理する．後述 4.2.2 節において，検査用 ML 関数を用いた動作チェックの例を示す．

図 4.1 にシミュレーションに使用する主要な型の ML コードを示す．

```
(*** Data and Type declaration ***)
type Name = string
(*
 属性に新しい型を追加するために必要な作業は
    1. Attribute_type に追加する。
    2. eqCommon 関数に等価判定を追加する。
    3. ppAttrVal 関数に表示機能を追加する
*)

datatype Attribute_type =
     Int of int
   | String of string
   | Null_of_Attr

type Attribute = Name * Attribute_type
type Event = Name * (Attribute list)
type Env = (Name * Attribute list) list
datatype X =
     Initial
   | State of string
   | Single of ((X * Attribute list * Event list * Env) ->
                    (X * Attribute list * Event list))
   | Parallel of ((X * Attribute list * Event list * Env) ->
                      (X * Attribute list * Event list)) list
   | Null
   | Name
type Object =
     (X * Attribute list * Event list * Env) -> (X * Attribute list * Event list)
type ObjectConf = Object * X * Attribute list
type SysConf = ObjectConf list * Event list
```

図 4.1: ObML でシミュレーションに使用する主要な型

4.1.3 ObCL 記述環境

設計者が設計の途中でどのような仕様とすべきかを検討している段階では，妥当な仕様を見出すまでにかなりの試行錯誤が必要である．経験的には，仕様を検討するときにはいろいろな場合を想定して一番良かろうというものに決め，波及する影響を考え…，という作業を頭のなか (あるいは机上) で繰り返している．とりわけ組込みシステムの場合には，実機あるいはクロス環境ができていない段階では設計者の経験に照らして判断するしかないことが多い．このようなときに，仕様を少し変更し，計算機上少し動かして確認し，また少し修正し…，という作業を繰り返すことができれば，仕様検討の効率を上げることができる．しかし，4.1.1節に示した ObCL → ObML コンバータを用いてこの作業を行うと，小さな変更のたびに ObCL と ML の間を往復しなければならず，やや不便である．

ObCL 記述環境は，そのような作業を支援するための対話的な環境で，関数型言語 Standard ML 上に構築されている．以下のような機能が用意してある．

- ML のデータとして記述したクラス/フィールド/イベントクラスなどを対話的あるいはバッチ的に処理するための関数群.
- ML 関数として記述されたクラス/フィールド/イベントクラスなどを ObCL 記述に変換する.
- 対話環境上の ObCL 記述をシミュレーション可能なコードに変換する.

これらの機能を用いて，記述する→シミュレーションする→記述する→…という作業手順でシステムの仕様を検討しながら記述を進めることができる.

4.2 ObML の利用例

ObML 環境での簡単な実行例を示す.

4.2.1 シミュレーションの実行例

以下は，3.2 節で ObCL の記述例として与えた CD プレーヤーの実行例である.

この例では，オブジェクト turntbl に イベント列 (play → pause → pause) を与えたときに，turntbl オブジェクト, lcd オブジェクトがどのような状態になり，どのような属性値に変化するのかを観察することができている.

```
- ppSystem( system' [turntbl] [ [],[],[("button.play",[])],
                              [("button.pause",[])],[("button.pause",[])]] 7);
Step:0
  turntbl:Initial,()
Step:1
  turntbl:State s_stop,()
  lcd.disp_msg(msg=0)
Step:2
  turntbl:State s_stop,()
  button.play
Step:3
  turntbl:State s_play,()
  lcd.disp_msg(msg=1),button.pause
Step:4
  turntbl:State s_pause,()
  lcd.disp_msg(msg=2),button.pause
Step:5
  turntbl:State s_play,()
  lcd.disp_msg(msg=1)
Step:6
  turntbl:State s_play,()
val it = () : unit
-
```

4.2.2 シミュレータ上での動作チェックの例

以下は，3.2 節で ObCL の記述例として与えた CD プレーヤーの動作チェック例である．

設計者がターンテーブルが止まっている時に LCD に”PAUSE” メッセージが送られないことを確認したいとしよう．ObML 上でこのようなことを調べるためには次の手順を踏む．

1. ターンテーブルが止まっている時に LCD に”PAUSE” メッセージが送られないことをチェックする検査用 ML 関数を記述する．
2. シミュレーション実行結果を検査用 ML 関数に与えて確認する．

以下は，この上記の手順にしたがって動作確認を行った例である．ここで，searchEv 関数は述語と event list を受け取り，event list の中から，述語を満たすイベントを探して返す関数で，sysevlist 関数は，system configuration から event list list だけを抜き出す関数である．

```
(* system configuration から event list list だけ抜き出す関数 *)
fun sysevlist [] = []
  | sysevlist ((objconf,es)::ss) =
    es::sysevlist ss;

(* ターンテーブルが止まっている時は， LCD に “PAUSE” メッセージが
   送られないことをチェックする関数 *)

(* 2 つの関数 inplay,instop を使って書いた場合 *)
fun check1 ss =
    let
        fun checkev ("LCD_FIELD.disp_msg",[("message",String "PAUSE")]) = true
          | checkev ("BUTTON_FIELD.play",[]) = true
          | checkev ("BUTTON_FIELD.stop",[]) = true
          | checkev _ = false;
        fun instop ([]:Event list list) = true
          | instop (e::es) =
            case (searchEv checkev e) of
                ("BUTTON_FIELD.play",[]) => inplay es
              | ("LCD_FIELD.disp_msg",[("message",String "PAUSE")]) =>
                    false
```

```
                | _ => instop es
        and inplay ([]:Event list list) = true
          | inplay (e::es) =
              case (searchEv checkev e) of
                  ("BUTTON_FIELD.stop",[]) => instop es
                | _ => inplay es
    in
        instop (sysevlist ss)
    end;
```

(* ターンテーブルが止まっている時は,
LCD に “PAUSE” メッセージが送られないことをチェックする *)

```
- check1 (system' [CD_PLAYER_cd_player_panel]
        [[],[("BUTTON_FIELD.play",[])],[("BUTTON_FIELD.pause",[])],
        [("BUTTON_FIELD.play",[])]] 5);
val it = true : bool
-
```

4.3 設計プロセスの支援

これまでに，仕様を記述するために記述言語 ObCL を提案し，記述支援環境や作成した仕様を確認するためのシミュレーション環境を構築した．第 5 章の事例研究は，主に ObCL → ObML コンバータを利用したバッチ処理的な作業手順を用いる．それは，ObCL コードを記述したファイルとイベントリストを記述したファイルを編集してはシミュレーション環境に与えるという手順である．一方，4.1.3 節に紹介した ObCL 記述支援環境は，関数型言語 Standard ML 上での対話的な環境である．これを積極的に利用すれば，ObCL 記述支援環境と設計者とのやりとりを設計作業の過程として再利用できる環境を構築できる可能性がある．ここでは，便宜上設計者と環境のやりとりの過程の支援を設計プロセスの支援と呼ぶことにする．

設計者がクラスやイベントに着目して仕様を検討しているときには，図 4.2 に示すような作業手順をとっていることが多い．このとき設計者としては，仕様の検討が不十分で少

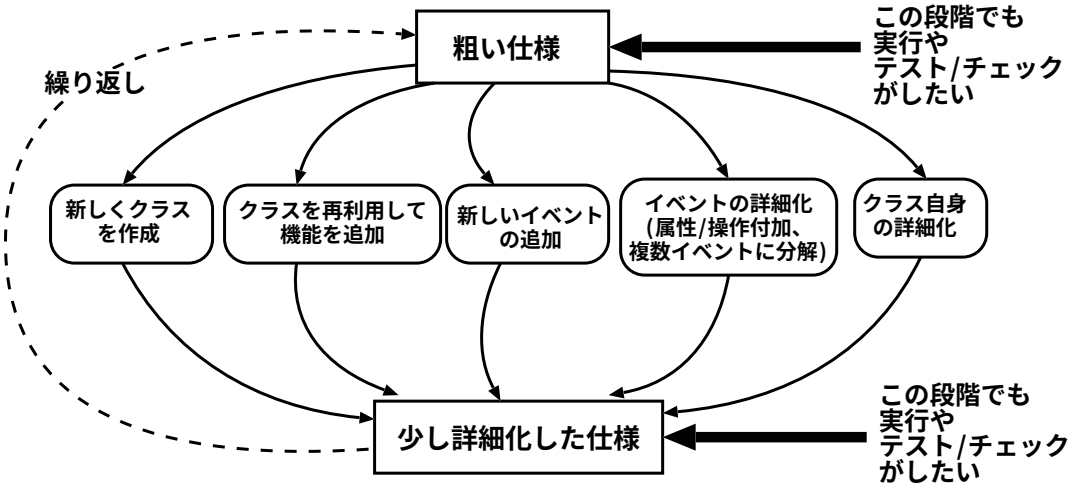


図 4.2: 設計作業の進め方の例

し粗い段階でも，シミュレーションして確認したい点は存在している．また，少し検討を加えて詳細化した場合にも，修正点が妥当であるか，他への影響はないかなどの確認をしたいところである．

例えば，図 4.3 に示すような場合を考えてみる．粗い段階では Timer オブジェクトが timeout イベントを Buzzer オブジェクトに送ることだけが決まっていたとし，詳細化の

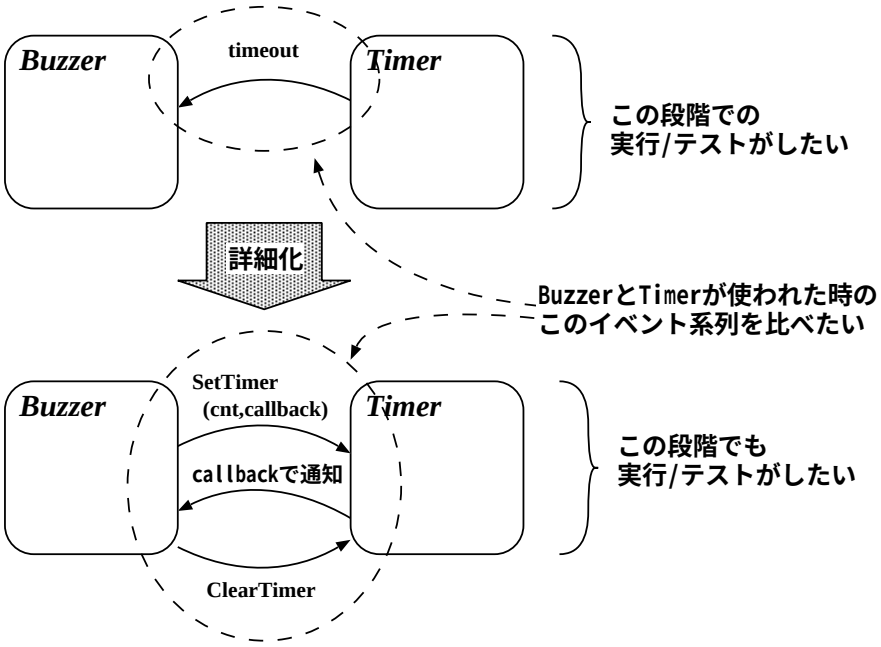


図 4.3: ブザーとタイマーがやりとりするイベントの詳細化の追跡

過程で，Timer オブジェクトの利用手順が次のように決まったとしよう.

1. SetTimer でタイムアウト時間とそのとき呼び出す callback 関数を指定しておく.
2. SetTimer 実施後，タイムアウト時間ごとに callback 関数を呼び出す.
3. ClearTimer で callback 関数呼び出しを繰り返すのを止める.

このとき，設計者としては 粗い段階で Buzzer オブジェクト中の timeout イベントが関わっていた部分が，詳細化のした段階で新しい Timer オブジェクトの利用手順に従うようになっているかを調べたいところである.

この例のような作業過程について次のような支援が考えられる.

- 新旧オブジェクトの比較
- オブジェクト間のやりとりの新旧の比較
- 設計プロセスの新旧両局面での実行やテスト，その結果の比較
- 設計プロセスそのものの保存，再利用

ObCL 記述支援環境とシミュレーション用の関数群を改良することで，上記の支援を実現できる．

4.4 ObCL Workbench

本研究では．支援の環境の基盤として Standard ML の対話的環境をそのまま利用しているが，この環境をさらに拡充することで分析/設計支援の統合環境として発展させることができる．図 4.4 に そのような統合環境 ObCL Workbench の構成概略図を示す．以下に ObCL Workbench の特徴を挙げる．

- Standard ML 環境上に構築
- ML 環境のもつ柔軟性を享受できる
- ObTS/ObCL に基づく設計支援環境
- プロトタイプ開発による設計プロセス支援
- 設計プロセスの保存/再利用が可能
- 柔軟な実行/テスト/各種チェックが可能

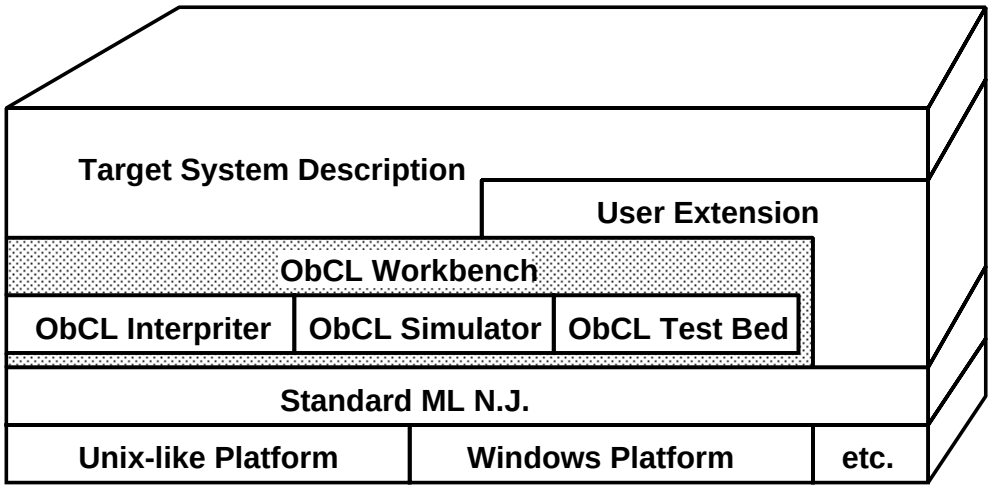


図 4.4: ObCL Workbench の構成概略図

第 5 章

事例研究

本章では ObCL/ObML を用いた事例研究について述べる．
本事例は，架空の複写機 CP-KT1 の操作部の分析を扱っている．

5.1 複写機 CP-KT1 の操作部プログラムの要求仕様の概略

複写機の操作部とは，ユーザーの操作を受け付ける，プリンタ/スキャナなどの制御機器に指示を送る，複写機の動作状態を表示する，などの機能を受け持つ部分のことである．本件の目的は，この CP-KT1 の操作部のプログラムを開発することであるが，本事例としてはこのうち分析過程だけを扱う．

この機種の製品化に向けていくつかの部門が開発を進めており，これまでのところ，

- 製品の外部仕様 (ハードウェア部門)
- 提供する機能 (製品企画部門)
- 操作パネルの意匠と基本操作方法 (デザイン部門)

などを各部門で作成している．

操作部は，制御部との通信および操作者との対話的なやりとりによって動作する．制御部は，スキャナ部，プリンタ部，給紙トレイ部を制御する部分で操作部はこれらとは別の組み込みプログラムによって制御するようになっている．制御部と各部との通信は操作部には見えない．デザイン部門の要求によって操作パネルの意匠は決まっている．また，基本的な操作手順についてもデザイン部門の要求によって決まっている．

本事例では，要求文書として以下の文書を使用した．

- CP-KT1 の外部仕様
- CP-KT1 が提供する機能のリスト
- CP-KT1 の機器構成
- 操作部の構成
- 操作パネルの構成
- 操作部の基本操作方法仕様
- 操作部-制御部間コマンド インターフェース仕様

上記文書は付録として添付する．付録 B を参照されたい．

5.2 分析結果

本事例の分析には方法論として OOSE [10] を採用した．採用の理由は，本事例のシステムが事象駆動型のシステムであり，基本操作仕様という use-case に転用可能な文書が要求文書として用意されているので，OOSE の use-case 駆動による分析/設計の手法によく適合するであろうと考えたからである．

分析作業は以下の手順にしたがって実施した．

1. 要求モデルの作成
   (a) アクタの識別
   (b) インターフェース記述の作成
   (c) use-case の識別
   (d) 問題ドメインオブジェクトモデルの作成
2. 分析モデルの作成
   (a) 分析オブジェクトの識別
   (b) 分析モデルの作成
   (c) インタラクション図の作成

(d) ObTS モデル図の作成

3. ObCL コードの作成
4. ObCL/ObML を用いた仕様確認

分析作業の成果物は付録として添付する．付録 C が要求モデル，付録 D が分析モデル，付録 E が ObCL コード，付録 F が ObML コードである．

5.3 ObML を用いた仕様確認の実行例

5.3.1 システム全体のシミュレーション実行

分析結果が use-case にしたがっているかどうかを調べるには，以下の手順によってシミュレーションを実行すればよい．

1. 5.2 節で作成した ObCL コードを 4.1.1 節で述べた ObCL → ObML コンバータによって ObML コードに変換する．
2. use-case ごとに外部からのイベントを記述したイベントリストを用意する．
3. Standard ML を起動し，インタプリタ上で変換した ObML コードを読み込む．
4. ひき続いて確認したいイベントリストを読み込む．

例えば「use-case 通常コピーの場合」について用意するイベントリストは以下に示すリストのようになる．この use-case は，スタンバイ状態でコピースタートキーを押してコピーを実行し，コピー終了後再びスタンバイ状態に戻るという，本事例で最も典型的な動作を表したもので，シミュレータのステップ数にして 50 ステップ分のスクリプトである．ここで，system' 関数がイベントリストを与えてシミュレーションを実行する関数である．イベントリストの途中にある [ ] は，外部からはイベントを与えないステップを表しており，システム内部での処理時間を見込んだものである．また，(* ··· *) はコメント，ppSystem 関数はシミュレーション結果を読みやすいかたちに整形して表示する関数である．

```
(* スタンバイ → コピースタート → コピー中 → コピー終り → スタンバイ *)

ppSystem ( system' [Sys_op_panel]
        [
        [],                             (* シミュレーションシステムの 初期化ステップ *)
        [],                             (* panel システムの 初期化ステップ *)
        [("Tm.i400m",[])],              (* 400ms タイマイベントを与える *)
        [],                             (* Cm.Send *)
        [],                             (* Pp.Send, Tm.o1s_Start *)
10      [("Pp.Ack",[("param2",Int 0x01),("param3",Int 0x00),("param4",Int 0x24),
                    ("param5",Int 0x24),("param6",Int 0x24)])],
        [],                             (* Cm.Ack, Tm.o1s_Stop *)
        [],                             (* St.Wait, Ms.Size "おまちください"*)
        [("Tm.i400m",[])],              (* 400ms タイマイベントを与える *)
        [],                             (* Cm.Send *)
        [],                             (* Pp.Send, Tm.o1s_Start *)
        [("Pp.Ack",[("param2",Int 0x02),("param3",Int 0x00),("param4",Int 0x24),
                    ("param5",Int 0x24),("param6",Int 0x24)])],
        [],                             (* Cm.Ack, Tm.o1s_Stop *)
20      [],                             (* St.Standby, Ms.Standby "コピーできます"*)
        [],                             (* Tm.o60s_Start, Ms.Count ,Ms.Select *)
        [("Key.Start",[])],             (* スタートキーを押すイベントを与える *)
        [],                             (* Cm.Send *)
        [],                             (* Pp.Send, Tm.o1s_Start *)
        [("Pp.Ack",[])],

        (* ... 省略 ... *)

        [],                             (* Cm.Ack, Tm.o1s_Stop *)
30      [],                     (* St.Standby, Ms.Standby, Tm.i200m_Stop "コピーできます"*)
        [],                             (* Tm.o60s_Start, Ms.Count, Ms.Select *)
        []
        ] 100 );
```

実行結果は以下に示すリストのようになる．実際には Step:0 から Step:50 まで，70 桁/行に表示して 1815 行の出力である．その各ステップごとに全てのオブジェクトについての状態と属性値を報告し，各ステップの最後にそのステップで発生したイベントを報告する．

```
Step:0
    Sys_op_panel:Initial ()
Step:1
    Sys_op_panel:Parallel [Sys_op_panel_copy_ctrl Sys_op_panel_status_ctrl
        Sys_op_panel_timer_ctrl Sys_op_panel_key_if Sys_op_panel_lcd_if
        Sys_op_panel_comport_if Sys_op_panel_cur_settings
        Sys_op_panel_def_settings] ()
    Sys_op_panel_copy_ctrl:State S_ready (key_input_start_flag=0)
    Sys_op_panel_status_ctrl:State S_ready (current_paper_select_position=1
        current_paper_count=1 default_paper_select_position=1
        default_paper_count=1 tray_bottom_old=255 tray_middle_old=255
        tray_top_old=255 copy_sub_status_old=0 copy_status_old=0)
    Sys_op_panel_timer_ctrl:Parallel [Sys_op_panel_timer_ctrl_tm_400m_ctrl
        Sys_op_panel_timer_ctrl_tm_200m_ctrl
        Sys_op_panel_timer_ctrl_tm_60s_ctrl Sys_op_panel_timer_ctrl_tm_1s_ctrl]
        ()
    Sys_op_panel_timer_ctrl_tm_400m_ctrl:State S_off ()
    Sys_op_panel_timer_ctrl_tm_200m_ctrl:State S_off ()
    Sys_op_panel_timer_ctrl_tm_60s_ctrl:State S_off ()
    Sys_op_panel_timer_ctrl_tm_1s_ctrl:State S_off ()
    Sys_op_panel_key_if:State S1 ()
    Sys_op_panel_lcd_if:Parallel [Sys_op_panel_lcd_if_mess_lcd
        Sys_op_panel_lcd_if_psel_lcd Sys_op_panel_lcd_if_pcnt_lcd
        Sys_op_panel_lcd_if_jam_lcd] ()
    Sys_op_panel_lcd_if_mess_lcd:State S1 (nak_code=0
        message="しばらくおまちください。")
    Sys_op_panel_lcd_if_psel_lcd:State S1 ()
    Sys_op_panel_lcd_if_pcnt_lcd:State S1 (count=1)
    Sys_op_panel_lcd_if_jam_lcd:State S1 ()
    Sys_op_panel_comport_if:State S_ready ()
    Sys_op_panel_cur_settings:State S1 (paper_select_position=1
        paper_count=1)
    Sys_op_panel_def_settings:State S1 (paper_select_position=1
        paper_count=1)
    Tm.i400m_Start Si.CurrentChange(paper_count=1 paper_select_position=1)
```

```
        Si.DefaultChange(paper_count=1  paper_select_position=1)
  Step:2

    ... 省略 ...
40
  Step:50

    ... 省略 ...

  val it = () : unit
  -
```

5.3.2 特定のオブジェクトに着目したシミュレーション実行

特定のオブジェクトの動作だけに着目したシミュレーション結果を得るためには，以下に示すリストのようなイベントリストを用意すればよい．この例は，copy_ctrl オブジェクトに着目した，シミュレータのステップ数にして 50 ステップ分のスクリプトである．ここで，system' 関数がイベントリストを与えてシミュレーションを実行する関数である．ppSubSys 関数は着目したいオブジェクトのリストを受け取り，そのリストに含まれるオブジェクトに関するシミュレーション結果だけを表示する関数である．

```
(* スタンバイ → コピースタート → コピー中 → コピー終り → スタンバイ *)
(* Sys_op_panel_copy_ctrl だけに着目した場合 *)

ppSubSys ( system' [Sys_op_panel]
        [
        [],                          (* シミュレーションシステムの 初期化ステップ *)
        [],                          (* panel システムの 初期化ステップ *)
        [("Tm.i400m",[])],           (* 400ms タイマイベントを与える *)

        (* ... 省略 ... *)

        [],                          (* St.Standby, Ms.Standby, Tm.i200m_Stop *)
        [],                          (* Tm.o60s_Start, Ms.Count, Ms.Select *)
        []
        ] 100 )
        [Sys_op_panel_copy_ctrl]; (* 着目するオブジェクトをこのリストに記述する *)
```

実行結果は以下に示すリストのようになる．実際には Step:0 から Step:50 まで，70 桁/行に表示して 262 行の出力である．各ステップごとに copy_ctrl オブジェクトににについて状態と属性値を報告し，ステップの最後にそのステップで発生したイベントを報告する．

```
Step:0
Step:1
    Sys_op_panel_copy_ctrl:State S_ready (key_input_start_flag=0)
    Tm.i400m_Start Si.CurrentChange(paper_count=1 paper_select_position=1)
        Si.DefaultChange(paper_count=1 paper_select_position=1)

  ... 省略 ...

Step:50
    Sys_op_panel_copy_ctrl:State S_ready (key_input_start_flag=0
        current_paper_select_position=1 current_paper_count=1
        default_paper_select_position=1 default_paper_count=1)
val it = () : unit
```

5.4 事例研究のまとめ

5.4.1 作成した ObCL コードの規模

本事例で作成した ObCL/ObML コードの規模を表 5.1 に示す．このうち，ObML のコードは全体で 90837 バイトで，一行が非常に長いため 70 桁/行に編集した場合には 1695 行になる．また，全てのオブジェクトの動作を報告するよう指定して 50 ステップ分のシミュレーションを実行すると，952 行，86770 バイトの出力があり，70 桁/行に編集した場合には 1815 行になった．

クラス数	18
オブジェクト数	17
フィールド数	7
イベントクラス数	16
イベント数	58
状態数	34
遷移規則数	153
ObCL コード行数	3359
ObML コード行数	915

表 5.1: 事例研究の ObCL コードの規模

5.4.2 規模とコードの可読性

ObCL コードは，本事例のように 3000 行を越える規模になっても ObTS モデル図やインタラクション図との対応がとりやすい．これは，ObTS 図のひとつの遷移の弧が ObCL コードのひとつの遷移規則句 (transition 節のなかのひとつの遷移規則記述) に対応しているからである．

例えば「use-case リセット操作」におけるリセットキーが押されてからスタンバイになるまでの Copy_Control クラスの動作に着目してみる．以下のコードは，Copy_Control クラスの ObCL コードの抜粋である．遷移規則 T6 が，状態 S_ready でリセットキーが押されたことを通知されたら，オートクリア用 60s タイマを止めるイベントとリセットコマ

ンドを送っているコードである．遷移規則 T7 が，Ack 通知を受け取ったときのコード，遷移規則 T7_2 が，スタンバイ通知を受け取ったら，オートクリア用 60s タイマを起動し，再描画用のイベントを送り， S_ready 状態に戻るコードである．

```
class
    Copy_Control

  -- 省略 ---------

  T6 is
    source
        S_ready
    input
10      Key.Reset or Tm.o60s
    do
        Cm.Send.cid := CID_Reset;
        Cm.Send.clen := CLEN_Reset
    destination
        S_reset
    output
        { Cm.Send, Tm.o60s_Stop }
  end;
  T7 is
20  source
        S_reset
    input
        Cm.Ack
    destination
        S_reset
  end;
  T7_2 is
    source
        S_reset
30  input
        St.Stanby
    do
      current_paper_count := default_paper_count;
      current_paper_select_position := default_paper_select_position
    destination
        S_ready
```

```
    output
        { Tm.o60s_Start, Ms.Count, Ms.Select }
  end;

40
  -- 省略 ---------

end -- Copy_Control
```

この部分についての ObTS 図の抜粋を図 5.1 にインタラクション図の抜粋を図 5.2 に示す．それぞれが，先に示した ObCL コードと一対一に対応しているのがわかる．

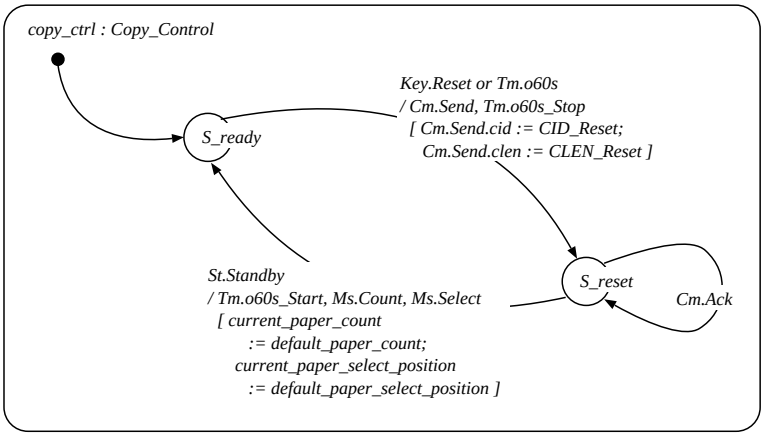


図 5.1: リセット操作部分の ObTS モデル図 (copy_ctrl:Copy_Control) の抜粋

シミュレーション結果は，インタラクション図に示したイベントのやりとりと対応づけできるので，期待した動作をしているかどうかを判断しやすい．しかし，この対応づけは ObML の実装上の都合からまだ自動化できていない．

本事例のステータス制御オブジェクト (status_ctrl:Status_Control : 付録 E の ObCL コード，図 D.41 の ObTS モデル図を参照) のように遷移条件の計算によって複雑な場合分けをするようなケースでは，シミュレーションの際に対象システムの動作に不具合を発見しても，それが仕様の問題なのか記述の不具合なのかを判断することが困難になってくる．このような事態を，要求仕様文書が与えている仕様が不適切であると捉えるか，やむを得ずこのまま開発をすすめると考えるかは，開発現場でも判断の難しいところである．

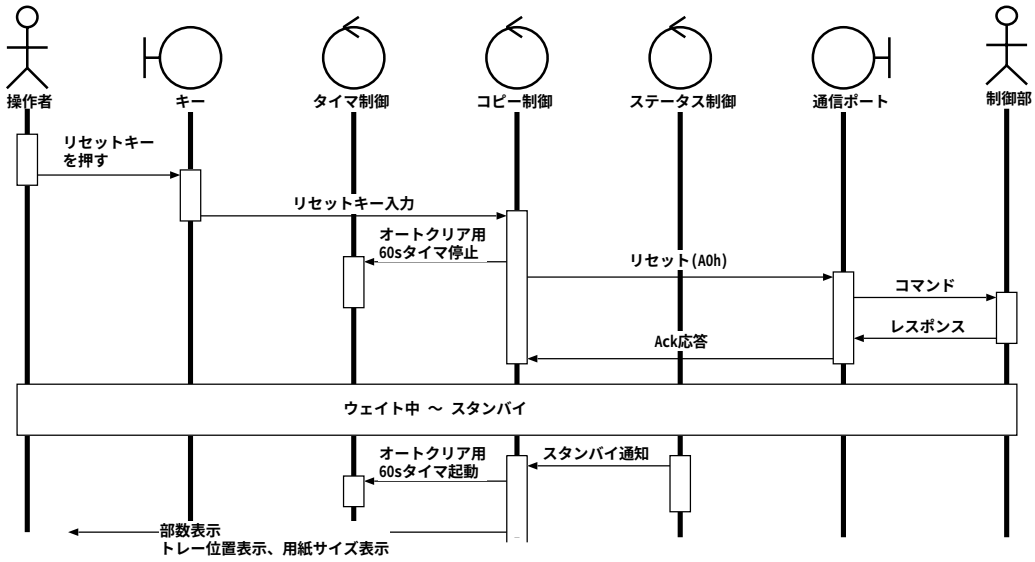


図 5.2: リセット操作部分のインタラクション図の抜粋

5.4.3 記述の再利用性

ObCL が記述の再利用性のために持ち込んだクラスの継承とイベントクラスの継承は，本事例ではほとんど利用しておらず，これらの有効性を確認することができなかった．それは， ObCL で提案した継承は制約が多く，再利用できる条件が満たされる場合がなかったためである．例えば，本事例の Default_Setting クラスと Current_Setting クラスには，共通の親クラスを用意して継承できそうであった，しかし継承する場合には既存の遷移規則を書き換える必要が生じていまい，継承をあきらめることとなった．

一方，本事例の仕様にキーやメッセージが追加となった場合には，既存の遷移規則に影響なく追加が可能なので，クラスの継承を利用可能である．図 5.3 にキー入力インターフェースオブジェクトに新しいキー New を追加した場合の ObTS モデル図を示す．図内で $Key.In[Key.In.code = 20]/Key.New$ が追加された遷移である．区別のために継承元の遷移規則の記述を網かけとし，継承元の遷移の弧を破線とした．

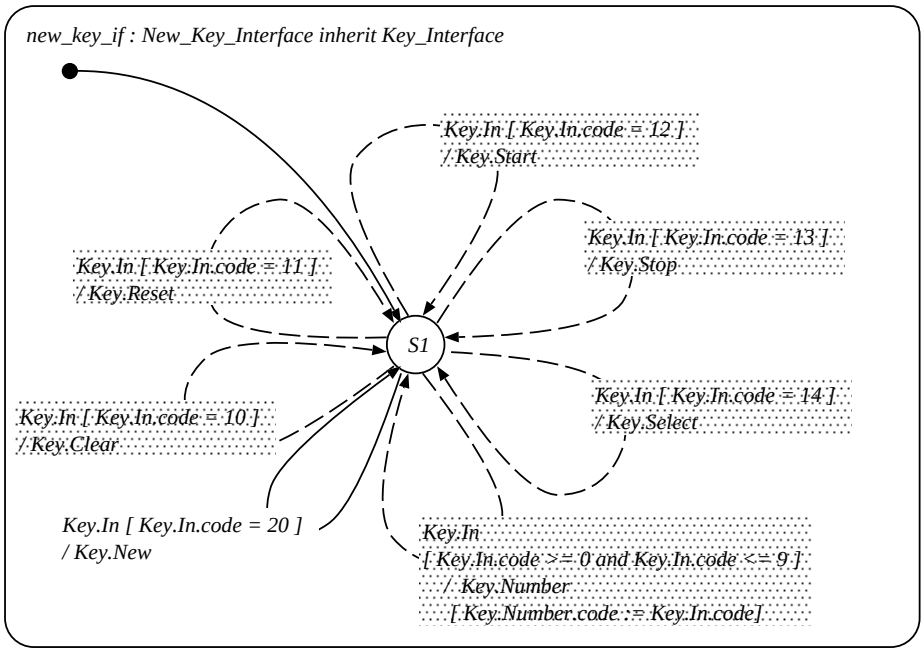


図 5.3: キー入力インターフェースオブジェクトの継承

5.4.4 疑似外部オブジェクトの活用

本事例では，操作部に含まれるオブジェクトだけをシステム内部のオブジェクトとして記述し，システムの外部—本事例ではアクタである制御部や操作者—についてはオブジェクトを記述しなかった．分析作業中はシステムの境界をはっきりさせておきたかったからである．このような方法をとった場合，イベントリストには外部オブジェクトからのイベントを書いておけばよい．またこの方法には外部からのイベントを与えるタイミングやイベント属性の値を自由に設定できる柔軟性がある．しかし，この方法では全ての外部イベントを用意しなければならず，仕様確認作業で着目したいオブジェクトやイベント以外にも気を配らねばならない．use-case が複雑あるいは長い場合にはこの問題は設計者をかなり煩わせる．

打開策として，システム外部オブジェクトあるいはアクタに相当する疑似オブジェクトを用意し，これらに適宜外部イベントに応答させる方法がある．本事例の場合には，制御部の振舞いを単純化したものを用意することが考えられる．たとえば，図 5.4 は，ステータス取得要求 (GetStatus) の応答を自動化することを狙った外部疑似オブジェクトの例である．

注意しなければならないのは，疑似外部オブジェクトはシステム境界の外側の存在であり，分析の過程でこれらのオブジェクトをシステム内部のオブジェクトと密に結合させるような記述を持ち込んではならないことである．疑似オブジェクトは，仕様確認作業においてアクタの単純な振舞いをシミュレートするためだけに用いるべきである．

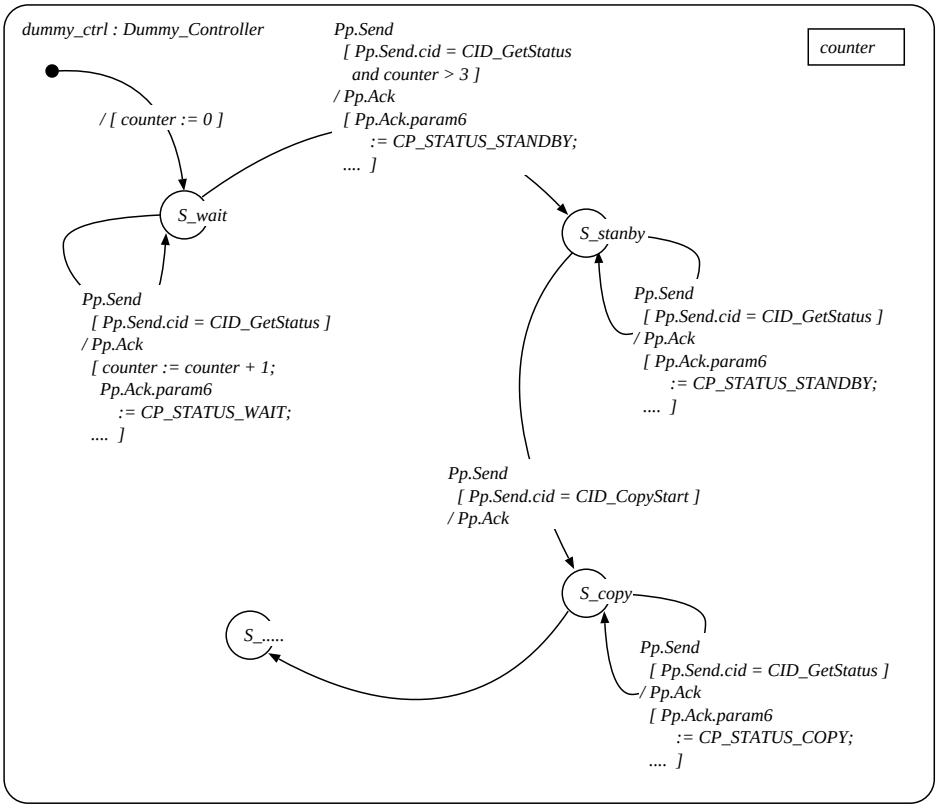


図 5.4: 制御部の疑似外部オブジェクトの例

第 6 章

ObCL/ObML の評価

本章では ObCL/ObML が実用規模の記述に関して，記述の可読性，記述の再利用性について有効であるかを評価する．

6.1 コードの可読性

6.1.1 ObCL コードの可読性

5.4.2 節に示したように，ObCL コードは ObTS モデル図やインタラクション図との対応がとりやすい．

また，インタラクション図との対応では，イベントがフィールドに属していることが効果をあげている．フィールドによって，イベントが関与するオブジェクトが限定されるので，シミュレーション出力や ObCL コード上でイベントの動きを追う場合に，インタラクション図上でどのオブジェクトに着目すれば良いのかが容易に判断できる．

規模が大きくなると，ObCL でも記述するコード量が多くなり，シミュレーションの際に対象システムの動作に不具合を発見しても，それが仕様の問題なのか記述の不具合なのかを判断することが困難になってくる．回避手段としては，着目したいオブジェクトを関係するフィールドでシステムから切り離すことが容易できるようにし，部分的なシミュレーションを実施しやすくする方法がある．

6.1.2 ObML コードの可読性

ObML コードは，ObCL コードの記述によっては付録 F に示すようにかなり複雑になり，そのままでは人が読むのは困難である．これは，現在の ObML の実装が，人間が直接 ML で記述する規模の問題を想定して実装されたものであり，複雑で規模の大きい問題に対しては可読性を維持できるよう作られていないためである．改善のためには ObML 内部でのデータの保持方式を変更する必要がある．

6.1.3 シミュレーション出力の可読性

シミュレーション結果はインタラクション図のイベントのやりとりと対応するので，期待した動作をしているかどうかを判断しやすい．逆に，ObTS モデル図/インタラクション図の助けなしでは見通しが悪い場合もある．それは，遷移規則がイベント自体の違いではなく，遷移条件を計算することで場合分けをする場合である．第 5 章の事例研究の場合では，同じ状態間の同じイベントに対する遷移規則が遷移条件の場合分けによって 60 通り以上に及ぶ場合があった．

6.2 記述の再利用性

6.2.1 記述の再利用性のための ObCL の特性の評価

ObCL が記述の再利用性のために持ち込んだものは，

- クラスの継承
- イベントクラスの継承

であったが，第 5 章の事例研究ではほとんど利用しておらず，これらの有効性は確認できなかった．ObCL のクラスの継承は，状態遷移図や ObTS モデル図での動作の継承が基本になっているが，これらは制限されたものであるため限られた局面でなければ利用できない．このため，事例研究でも継承が利用できそうな局面があったが使うことができなかった．したがって，上記の ObCL の再利用性のための特性については効果に疑問が残る．

6.2.2 再利用のレベルによる評価

Yordon [18] によれば，再利用には以下のような段階がある．

1. コードの再利用．
   ソースコードのカット・アンド・ペースト，ソースコードのインクルード，ライブラリ・リンク，実行時呼出し，シンボリックなパラメータのインクルードなどがこれにあたる．
2. 設計の再利用．
   コードの再利用はシステム設計階層の末端でなされるだけであるが，設計の再利用は再利用対象の木構造の全ての枝で行うことができる．
3. 仕様の再利用．
   分析結果の再利用のこと．仕様モデルがCASE リポジトリ上で維持できるようになり実用的になりつつある．
4. その他の再利用．
   テスト・ケース，テスト手順，テスト動作環境など

ObCL の場合，記述をコードとして見た場合にはコードの再利用に挙げた方法が利用できる．たとえば，分析結果として仕様上の定数を整理した場合には，シンボリックなパラメータのインクルードが利用できる．イベントやフィールドの定義についてもコードのインクルードによる再利用は可能である．現在の ObCL → ObML コンバータ自身にはインクルード機能はないが，プリプロセッサを用いれば実現は容易である．

設計/仕様の再利用としては，ObCL の継承の有効性には 6.2.1節 に述べたように疑問が残る．

テスト・ケースの再利用としては，シミュレーションに用いたイベントリストなどがこれにあたる．テスト・ケースのデータとして再利用可能である．

第 7 章

組込みシステムへの対応

組込みシステムは，多くの場合実時間システムとしての側面を持つ．ObCL/ObML を実時間システムに適用するには，システムの設計/実装の段階の実時間性に関する情報を使用する必要があるが，実現方法が確定していない段階ではこれらの正確な把握は難しい．したがって本章では，タスクマッピングの方法と実行時モデルについての調査と議論の結果を述べるに留める．

7.1 タスクへのオブジェクトのマップ方法

リアルタイム OS (RTOS) のタスクに ObCL/ObTS オブジェクトをどのように割り付けるのかについては，いくつかの研究を調査し，試験的に適用してみるに留まった．タスクへのマッピングの設計の手法については，Gomma [4] や川口 [11] などがある．

川口の研究では μ トランザクションに基づくタスクのマッピング方法を提案している．μ トランザクションとは，システムに対して非同期にイベントを発生するオブジェクト—通常はアクタとみなされるもの—からのイベントと，それに続くクラス間のメッセージの連鎖のことである．第 5 章の事例にこの方法を当てはめると，次のようになる．ここでは「use-case コピー中に中断 (ストップ) する場合」について検討する．

1. 「use-case コピー中に中断 (ストップ) する場合」についてのインタラクション図 (一部を抜粋したもの) に対して μ トランザクションとなる部分を識別する．
2. 識別結果として図 7.1 を得る．

3. 識別した μ トランザクションについてタスクマッピングを施す (図 7.2).
4. インターフェースオブジェクトであるキー，通信ポートは，必要な場面受信するイベントの実時間性が異なるので独立したタスクとする.
5. コピー制御は，制御オブジェクトなので拡張性を考慮して独立したタスクとする.
6. 結果として，キータスク，通信ポートタスク，コピー制御タスクを割り当てることができた.

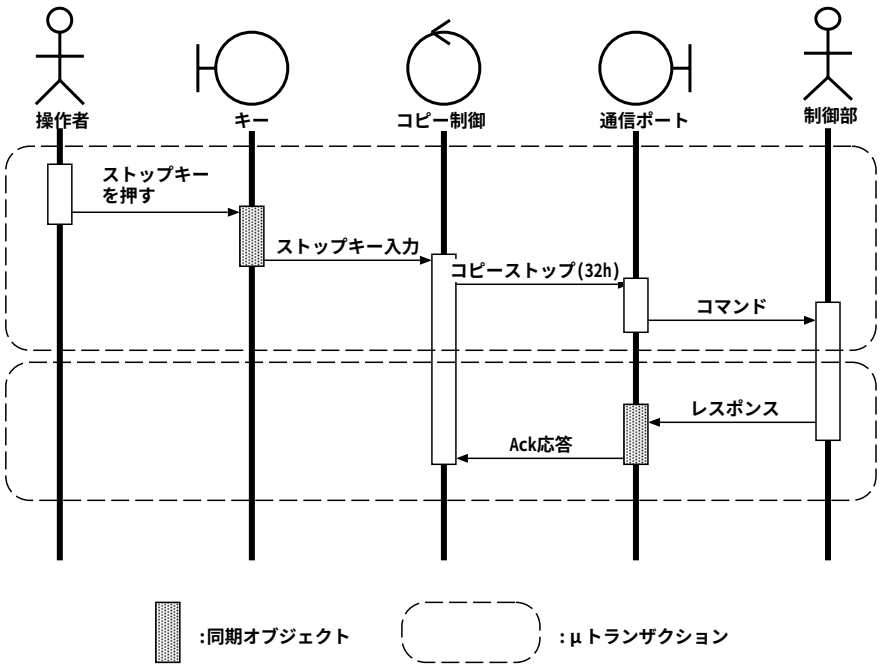


図 7.1: μトランザクションの例

以上の手順ですべての use-case について検討すれば，システム全体のタスクマッピングが可能である.

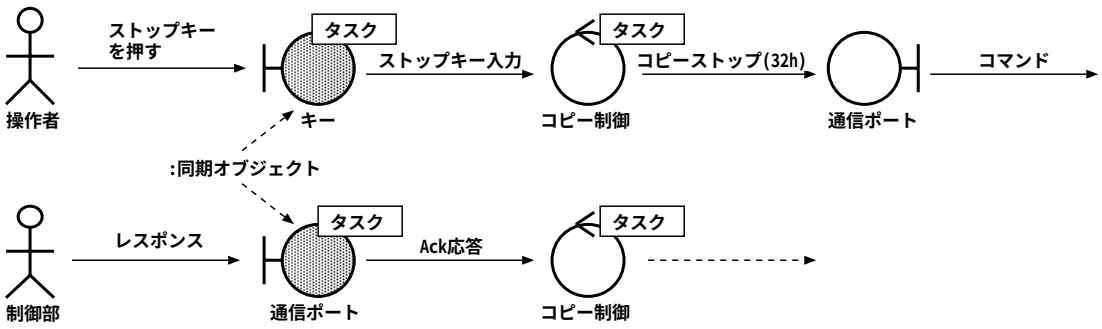


図 7.2: μトランザクションへのタスクマッピングの例

7.2 実行時モデルの提案

ObCL によるシステムの記述を分析段階から設計段階へと移行するには，実装上の情報を加味する仕組みが必要である．それには現場で利用している RTOS の機能と ObTS モデルの記述に明確な対応関係を持たせるために実行時モデルが必要となる．しかし，実行時モデルについては議論に留まり，提案できるだけの研究成果はあげることができなかった．

議論としてがあがっているのは，図 7.3 に示すように，上位レベルで得られたオブジェクトを, 実装レベルからボトムアップで整理したパーツオブジェクトを利用するように構成すればどうかというものである．しかし，これは RTOS がもつ機能あるいはサービスを ObTS モデルとして整理し，これをビルディングブロックとして利用するということであり，実行時モデルの提案にはなっていない．

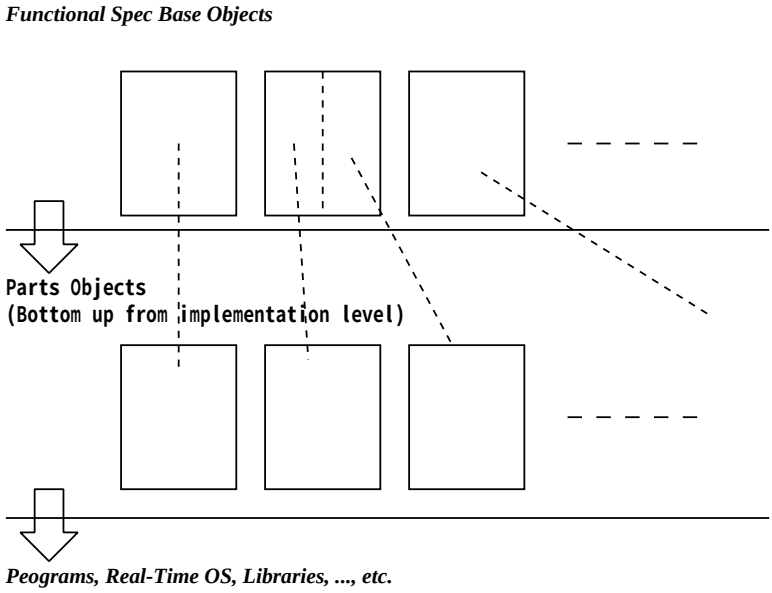


図 7.3: ビルディングブロックによるシステムの構成

第 8 章

今後の課題

本章では本研究に関連して今後なすべき課題について述べる．

リアルタイム OS (RTOS) のタスクへのマッピングの設計の手法については，Gomma [4] や川口 [11] を調査し，川口の方法を試験的に適用してみるに留まった．この課題の今後の進め方としては，以下の方針が考えられる．

- 具体的な実装環境を定めて実装時の情報を抽出し，川口の方法を実際に適用する．
- さらに広範にタスクマッピングの手法を調査し，本研究固有の方法を見出す．

ObCL によるシステムの記述を分析段階から設計段階へと移行するには，実装上の情報を加味するための実行時モデルが必要となる．しかし，実行時モデルについては調査自体がまだ不十分である．したがって，広範な調査から始める必要がある．

T.T. (Technology Transfer) の手順を整備するための課題としては，以下のようなことが挙げられる．

- T.T. の手順そのものの整備
- T.T. のために必要となるライブラリやツールの整備

ここでライブラリとは，RTOS の資源，サービスについて 各々を ObTS モデル化したパーツオブジェクト群などを指す．ツールとは，第 4 に示した ObCL → ObML コンバータ，ObTS シミュレーション環境 ObML，ObCL 記述環境であるが，これらを開発現場

の設計者が使いこなせせるようユーザーインターフェースを改善し，マニュアルやチュートリアル，教育手段を整備する必要があることを意味する．

統合環境 ObCL Workbench を構築し，ObCL 記述支援環境と設計者とのやりとりを設計作業の過程として再利用できる環境が用意できることを 4.3 節および 4.4 節で提案したが，そのためには以下のことができていなければならない．

- 設計プロセスを保存し，再利用するための仕組み．
- 設計者の試行錯誤の過程とシミュレーション結果などを峻別し見やすく表示するユーザーインターフェース．
- 設計プロセスが進んだ際に参照可能な実行時モデル．
- 実装上の情報を集積したパーツオブジェクト・ライブラリ．

第 9 章

まとめ

組込みシステムの開発現場においてオブジェクト指向方法論が利用されるようにするために，実用規模のシステムのための動的モデルの記述法と，仕様を検討する段階でも利用できるような計算機上の支援環境があればよいのではないかと考えた．ObTS は，オブジェクト指向分析/設計方法論の動的モデルとしての有用な特性を有しているので，本研究の基礎として ObTS を採用した．

計算モデルである ObTS に，具体的なシステムを記述できるよう構文を与え，記述の再利用の工夫や実用規模の問題を効果的に記述する特性付加した仕様記述言語 ObCL を用意した．

仕様記述言語 ObCL を用いて記述した仕様を計算機上で検討できるよう、ObCL → ObML コンバータ，ObCL 記述支援環境，シミュレーション環境 ObML を構築した。

ObTS の 実用規模での有効性を確認するために，架空の複写機 CP-KT1 の操作部の要求仕様を分析し，分析の結果得た ObCL/ObML コードをシミュレーションした．

事例研究の結果から，ObCL のコードは可読性が高いことがわかった．また，ObCL コードや，シミュレーション結果は，ObTS モデル図やインタラクション図との対応がとりやすいことがわかった．

クラスの継承，イベントクラスの継承は，記述の再利用の方法としては活用できず，再利用の工夫としての効果は明らかにできなかった．

ObCL/ObML にはまだ解決すべき課題が多いものの，ObTS を利用する支援環境としては規模が大きくなっても有用であることがわかった．

謝辞

本研究を行なうに当たり, 終始御指導を賜わった片山 卓也 教授に深謝いたします．また副指導教員である中島 達夫 助教授，副テーマ指導教官である Ho Tu Bao 先生には，多くの示唆に富む助言を頂戴しましたことに感謝いたします.

片山研究室の鈴木 正人 助手には日頃から有益な御助言をいただき, 多面に渡って励ましていただいきました．感謝いたします.

本論文をまとめるに当たって御協力いただいた片山研究室の諸兄，特に本論文の基礎となる考えである ObTS を提案された伊藤 恵氏には格別の御支援を頂戴いたしました．厚く御礼申し上げます.

筆者の派遣元であるキヤノンソフトウェア株式会社には，大学院への就学機会を与えていただきました．2 年間もの間現場を離れることを認めていただいたことに感謝いたします.

最後に，筆者の就学に快く賛成し，無限大の協力と励ましを与えてくれた妻と子供たちに心からの感謝の気持を捧げます.

参考文献

[1] Grady Booch. *Object-Oriented Analysis and Design with Applications.* Benjamin/Cummings Publishing Co., Menlo Park, Calif., 2nd edition, 1994.

[2] Derek Coleman, Fiona Hayes, and Stephen Bear. Introducing Objectcharts or How to Use Statecharts in Object-Oriented Design. *IEEE Transactions on Software Engineering*, Vol. 18, No. 1, pp. 9–18, Jan. 1992.

[3] Shiyuan Ding and Takuya Katayama. Specifying Reactive Systems with Attributed Finite State Machines. In *Proc. of International Workshop on Software Specification and Design.*

[4] H. Gomma. *Software Design Methods for Concurrent and Real-Time Systems.* Addison-Wesley, 1993.

[5] D. Grangopadhyay and S. Mitra. ObjChart: Tangible Specification of Reactive Object Behavior. In *ECOOP '93*, 1993.

[6] D. Harel and E. Gray. Executable Object Modeling with Statecharts. In *ICSE-18*, pp. 246–, 1996.

[7] D. Harel, A. Pnueli, J. P Schmid, and R. Sherman. On the Formal Semantics of Statecharts. In *Proc of 2nd IEEE Symposium on Logic in Computer Science*, pp. 54–64, 1987.

[8] David Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, pp. 231–274, 1987.

[9] i-Logix Inc., Andover, Massachusetts. *Statemate MAGNUM User Guide Version 1.01*, 1996.

[10] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Övergaard. *Object-Oriented Software Engineering A Use Case Driven Approach.* Addison-Wesley, 1992.

[11] 川口晃, 岸知二, 門田浩. 組込みシステムの設計手法—オブジェクト指向設計を中心にして—. 情報処理, Vol. 38, No. 10, pp. 879–885, Oct. 1997.

[12] 伊藤恵, 片山卓也. オブジェクト指向方法論のための動的モデル ObTS. In 大蒔和仁（編）, *FOSE '95*, レクチャーノート/ソフトウェア学, No. 15, pp. 51–60. 日本ソフトウェア科学会, 近代科学社, 1996.

[13] 伊藤恵, 片山卓也. オブジェクト指向方法論のための動的モデル ObTS. コンピュータソフトウェア, Vol. 14, No. 2, pp. 22–37, Mar. 1997.

[14] Brian W. Kernighan and Dennis M. Ritiche. *The C Pprogramming Language*, chapter 1, pp. 33–34. Prentice-Hall, Englewood Cliffs, N.J., 2nd edition, 1988.

[15] 久保秋真, 伊藤恵, 片山卓也. ObTS モデルに基づくオブジェクト指向仕様記述言語と支援環境. 日本ソフトウェア科学会 第 14 回 全国大会論文集, pp. 589–592. 日本ソフトウェア科学会, 1997.

[16] James Rumbaugh, Michel Blaha, William Premerlani, Frederick Eddy, and William Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall, Englewood Cliffs, N.J., 1991.

[17] Aamod Sane and Roy Campbell. Object-Oriented State Machines: Subclassing, Composition, Delegation, and Genericity. In Rebecca Wirfs-Brock, editor, *OOPSLA '95 Conference Proceedings*, Vol. 30, pp. 17–32. ACM, ACM press, 1995.

[18] Edoward Yordon. *Object-Orineted Systems Design An Integrated Approach.* Prentice-Hall, Englewood Cliffs, N.J., 1994.

第 A 章

ObCL の言語仕様

仕様記述言語 ObCL の予約語，特殊記号，演算子，構文規則を示す。

A.1 予約語

表 A.1 に ObCL の予約語をアルファベット順に示す．キーワードをボールド体で，それ以外の予約語 (基本型や基底のイベント名など) をイタリック体で示してある．

and	**attribute**	**class**	**creation**
destination	**do**	**else**	**elsif**
end	**event**	**false**	**field**
from	*generic_event*	**if**	**indexing**
inherit	**init**	**inner**	**input**
int	**is**	**local**	**loop**
not	**operation**	**or**	**output**
result	**source**	**state**	*string*
system	**then**	**transition**	**true**
until	**when**	**xor**	

表 A.1: 予約語一覧表

ObCL では大文字と小文字の区別をしないので，表中では全てを小文字で書きあらわし

た場合の表記だけを示してある．

これらの予約語は，クラス名，イベント名，フィールド名，オペレーション名，遷移規則名，属性名などに使用できない．

A.2 特殊記号

表 A.2に特殊記号を列挙し，呼称と使途を示す．

記号	呼称	使途
--	double dash	コメントのはじまり．
;	semicolon	宣言の区切り，ブロックの区切りなど．
,	comma	引数の区切り，宣言の中の実体の並びの区切りなど．
:	colon	宣言中の型指定のはじまり．
.	dot	イベントの参照でのフィールド指定の区切り，イベントのオペレーション呼出しの区切り．
= /=	Equal, not-equal signs	等しい，等しくない．
( )	Parentheses	部分式のグループ化，実引数，仮引数の括り．
{ }	Braces	内部オブジェクトのグループ化．
:=	Receives	代入記号．
’	single quote	文字定数の括り．
"	double quote	文字列定数の括り．
+ -	Signs	整数，実数の符号．

表 A.2: 特殊記号一覧表

A.3 演算子と優先順位

表 A.3 に演算子の優先順位を示す．表中の順位が大きい程優先順位が高い．

順位	記号
9	.
8	**not**　unary +　unary -
7	^(power)
6	*　/
5	binary +　binary -
4	=　/=(not equal)　<　>　<=　>=
3	**and**　**and then**
2	**or**　**or else**　**xor**
1	;

表 A.3: 演算子とその優先順位の一覧表

A.4　構文規則

本節では ObCL の構文規則を示す.

ここでの構文規則の表記法は，次のようなものとする.

- 非終端記号は大文字で始まる.
- + や − のような終端記号は二重引用符で囲む.
- **class** のような肉太書体で書いた終端記号はそのままとなる.
- 垂直線 | は選択を表す.
- 丸カッコ (と) はグループ化に用いる.
- $(\cdots)?$ はオプショナルな (0 回か 1 回の) 構造を表す.
- $(\cdots)*$ は 0 回以上の繰り返しを表す.
- $(\cdots)+$ は 1 回以上の繰り返しを表す.

A.4.1 Program

Program ::= ((Indexing)?
(Class_declaration
| Field_declaration
| Event_class_declaration)
)* EOF

A.4.2 Indexing

Indexing ::= **indexing** (Index_list)?
Index_list ::= Index_clause (";" Index_clause)*
Index_clause ::= Index (Index_term ("," Index_term)*)?
Index ::= Identifire ":"
Index_term ::= Identifire | Manifest_constant

A.4.3 Class_declaration

Class_declaration ::= Class_header (Parent)? (Features)? **end**

A.4.4 Class_header

Class_header ::= **class** Class_name

A.4.5 Class_name

Class_name ::= Identifire

A.4.6 Class Inheritnce

Parent ::= **inherit** Class_name

A.4.7 Features

Features ::= (Fields_ref)?
(Attributes)?
(States)?
(Inners)?
(Creations)?
(Operations)?
Transitions

A.4.8 Fields_ref

Fields_ref ::= **field** (Field_ref_list)?
Field_ref_list ::= Field_name (”,” Field_name)∗
Field_name ::= Identifire

A.4.9 Attributes

Attributes ::= **attribute** (Attribute_list)?
Attribute_list ::= Attribute_name (”,” Attribute_name)∗
Attribute_name ::= Identifire

A.4.10 States

States ::= **state** (State_list)?
State_list ::= State_name (”,” State_name)∗
State_name ::= Identifire

A.4.11 Inners

```
                Inners ::= inner ( Inner_list )?
            Inner_list ::= ( Inner_declaration_group_list )+
Inner_declaration_group_list ::= Inner_declaration_group
                                 ( ";" Inner_declaration_group )*
  Inner_declaration_group ::= Identifire ( "," Identifire )* ":"
                              ( Class_name
                              | "{" Inner_declaration_group_list "}" )
```

A.4.12 Class_mark

```
Class_mark ::= ":" Class_name
```

A.4.13 Creations

```
                Creations ::= creation ( Creation_operation_name_list )?
Creation_operation_name_list ::= Operation_name ( "," Operation_name )*
```

A.4.14 Operations

```
                Operations ::= operation ( Operation_declaration_list )?
Operation_declaration_list ::= Operation_declaration
                               ( ( ";" )? Operation_declaration )*
```

A.4.15 Operation_declaration

```
Operation_declaration ::= Operation_name ( Operation_argument_list )?
                          ( Type_mark )? Operation_value_mark
  Operation_value_mark ::= is ( Manifest_constant | Routine )
```

A.4.16 Operation_argument_list

```
Operation_argument_list ::= "(" Operation_arguments ")"
     Operation_arguments ::= Operation_argument ( ";" Operation_argument )*
      Operation_argument ::= ( Identifire ( "," Identifire )* ) ( Type_mark )?
```

A.4.17 Operation_name

Operation_name ::= Identifire

A.4.18 Entity_declaration_list

Entity_declaration_list ::= Identifire (” ,” Identifire)∗

A.4.19 Type_mark

Type_mark ::= Class_mark

A.4.20 Routine

Routine ::= (Local_declarations)? Routine_body **end**

Local_declarations ::= **local** (Entity_declaration_list)?

A.4.21 Routine_body

Routine_body ::= **do** (Compound)?

A.4.22 Compound

Compound ::= Instruction (” ;” Instruction)∗

Instruction ::= Assignment | Conditional | Loop | Call

A.4.23 Boolean_expression

Boolean_expression ::= Expression

A.4.24 Assignment

Assignment ::= Writable ” :=” Expression

A.4.25 Call

Call ::= (Parenthesised_qualifier)? Call_chain

Parenthesised_qualifier ::= Parenthesised "."

Call_chain ::= Unquantified_call ("." Unquantified_call)*

A.4.26 Unquantified_call

Unquantified_call ::= Operation_name (Actuals)?

A.4.27 Actuals

Actuals ::= "(" Actual_list ")"

Actual_list ::= Actual ("," Actual)*

Actual ::= Expression

A.4.28 Expression

Expression ::= (Unary Expression
| Parenthesised
| Call
| Local
| Constant)
(Expression_d)?

Expression_d ::= (Comparison | Binary) Expression (Expression_d)?

A.4.29 Parenthesised

Parenthesised ::= "(" Expression ")"

Comparison ::= "=" | "/="

A.4.30 Unary

Unary ::= **not** | "+" | "-"

A.4.31 Binary

Binary ::= "+" | "-" | "*" | "/"
| "<" | "<=" | ">" | ">="
| "^"
| **and** (**then**)? | **or** (**else**)? | **xor**

A.4.32 Conditional

Conditional ::= **if** Then_part_list (Else_part)? **end**
Then_part_list ::= Then_part (**elsif** Then_part)*
Then_part ::= Boolean_expression **then** Compound

A.4.33 Else_part

Else_part ::= **else** Compound

A.4.34 Loop

Loop ::= Loop_initial Loop_body **end**
Loop_initial ::= **from** Compound
Loop_body ::= Exit_clause **loop** Compound
Exit_clause ::= **until** Boolean_expression

A.4.35 Constant

Constant ::= Manifest_constant
Constant_attribute ::= Identifire

A.4.36 Manifest_constant

Manifest_constant ::= Boolean_constant
| Manifest_string
| Integer_constant
| Character_constant
Boolean_constant ::= **true** | **false**

A.4.37 Sign

Sign ::= "+" | "-"

A.4.38 Manifest_string

Manifest_string ::= String_literal

A.4.39 Integer_constant

Integer_constant ::= (Sign)? Integer

A.4.40 Integer

Integer ::= Plain_integer | Delimited_integer

A.4.41 Character_constant

Character_constant ::= Character_literal

A.4.42 Writable

Writable ::= Identifire | **result**

A.4.43 Local

Local ::= Identifire | **result**

A.4.44 Transitions

Transitions ::= **transition** (Transition_declaration_list)?

Transition_declaration_list ::= Transition_declaration
((";")? Transition_declaration)*

A.4.45 Transition_declaration

Transition_declaration ::= Transition_name Transition_value_mark
Transition_name ::= Identifire
Transition_value_mark ::= **is** Tratisition_routine

A.4.46 Transition_routine

Tratisition_routine ::= Source_clause
(Transition_routine_body)?
Destination_clause **end**

A.4.47 Source_clause

Source_clause ::= **source** (**init** | State_name) (Source_body)?

A.4.48 Source_body

Source_body ::= **input** Event_list (**when** Boolean_expression)?

A.4.49 Transition_routine_body

Transition_routine_body ::= (Local_declarations)? Routine_body

A.4.50 Destination_body

Destination_clause ::= **destination** State_name (**output** Event_list)?

A.4.51 Event_list

Event_list ::= Qualified_event_name
("," Qualified_event_name)*
Qualified_event_name ::= Field_name "." Event_name
Event_name ::= Identifire

A.4.52 Event_class_declaration

Event_class_declaration	::=	Event_class_header
		(Event_class_parent)?
		(Attributes)?
		(Operations)?
		end
Event_class_header	::=	**event** Event_class_name
Event_class_name	::=	Identifire

A.4.53 Event Inheritnce

Event_class_parent	::=	**inherit** Event_class_name

A.4.54 Field_declaration

Field_declaration	::=	Field_header
		(Included_events)?
		end
Field_header	::=	**field** Field_name
Included_events	::=	**event** (Event_declaration_list)?
Event_declaration_list	::=	Event_declaration_group
		((";")? Event_declaration_group)*

A.4.55 Event_declaration_group

Event_declaration_group	::=	Event_name
		("," Event_name)* Event_mark
Event_mark	::=	":" Event_class_name

第 B 章

複写機 CP-KT1 の操作部プログラムの要求文書

B.1 はじめに

付録として，本章以降に ObCL および ObTS を実用規模の問題に適応した事例の全体を掲載する．本事例は，架空の複写機 CP-KT1 の操作部の分析を扱っている．

複写機の操作部とは，ユーザーの操作を受け付ける，プリンタ/スキャナなどの制御機器に指示を送る，複写機の動作状態を表示する，などの機能を受け持つ部分のことである．操作部のことを操作パネル[1]あるいはコントロールパネル (コンパネ) とも呼ぶこともある．

複写機の操作部を対象としたのは以下の理由による．

第一の理由は，架空の機種とはいえ複写機のような OA 機器の問題を実用規模に近いレベルで扱うことは，現実的な問題の事例とするために有用なことである．さらに現実的な問題とするために，コマンド インターフェース仕様とユーザー操作の仕様が操作部の開発現場に他部門から提示されたという設定にした．これらの仕様は，OOSE では要求モデルでプロトタイプを用意すべきところのものであるが，本事例では現実的な事情を加味した．

第二の理由は，コマンド インターフェース仕様とユーザー操作の仕様があれば，制御

[1] 原則として操作部と呼ぶ時は機器としての操作部全体を指し，操作パネルと呼ぶ時は操作者が見ることができる操作部のボタン類，LCD などの全体を指すことにする．

のシーケンスと操作者の非同期な操作の両方を扱う問題と捉えることができることである．今日の事務用複写機を構成するプリンタ/スキャナを始めとする各機器は，おのおのが電子的に制御できるようブラックボックス化されており，制御用のインターフェース仕様が決まっていれば，内部の詳細を知らなくとも制御できるようになっている．そして操作部はユーザ操作対応と機器制御の両方の側面を有している．したがって，制御インターフェース仕様とユーザー操作の仕様があれば，複写機のシステム制御の実際の開発状況に近い状況を想定できることになる．

第三の理由は，複写機は多くの機能を持つため，その全ての機器について個人が分析/設計するのは本研究の範囲では困難であると判断したためである．むろん，各機器が電子制御可能なブラックボックスとなっているので，本事例の手順を個別に各機器について実施すれば，複写機全体を対象とした事例が可能である．

なお本事例は，まず本章に要求文書を示し，分析を実施した結果については，付録 C に要求モデル，付録 D に分析モデル，付録 E に ObCL コード，付録 F に ObML コードを割り当てた構成とした．

B.2　要求仕様文書

CP-KT1 は，デスクトップ型 (オフィスに据えつけて使うタイプ) の中型複写機である．本件の目的は，この CP-KT1 の操作部のプログラムを開発することである．この機種の製品化に向けていくつかの部門が開発を進めており，これまでのところ，

- 製品の外部仕様 (ハードウェア部門)
- 提供する機能 (製品企画部門)
- 操作パネルの意匠と基本操作方法 (デザイン部門)

などを各部門が決めている (各仕様の内容については後述する)．

操作部は，CP-KT1 の目指す機能や操作性を実現するために，上記既出の各仕様を満たすように設計され，実装されなければならない．

操作部はまた，操作者が予想外の操作をすることを前提としたシステムとしなければならない．このため各種動作中の受け付け可能な入力については十分に検討する必要がある．

B.3　複写機 CP-KT1 の仕様

B.3.1　外部仕様

本事例で取り上げる複写機 CP-KT1 の外部仕様を以下に示す.

- 名称： CP-KT1
- 形式： デスクトップ型
- 原稿台方式： 原稿台固定型
- コピー用紙： 普通紙
- 複写原稿： シート，本，立体物 (2kg まで)
- 最大原稿サイズ： A3
- 複写サイズ： A3(横方向)，B4(横方向)，A4(縦方向)，A4R (横方向)
- 給紙方式： 3 段カセット式給紙 (250 枚× 3 カセット)
- 排紙方法： フェイスアップトレイ
- 連続複写： 最大 100 枚

B.3.2 提供する機能

本事例で取り上げる複写機 CP-KT1 が提供する機能を以下に示す.

- 普通紙コピー (1〜99 部)
- ストップキーによるコピーの中断
- プリンタ部/給紙トレー部の用紙詰まり (ジャムと呼ぶ) の検出 (リカバリ機能[2]なし)
- コピー中の用紙なしの検出 (給紙後のコピー継続可，給紙段の中途変更不可)
- コピー中でないときの用紙なしの検出
- 用紙 (給紙段) の選択 (3 段式トレイのみの構成)
- リセットキーによる部数および給紙段の初期化
- 1 分間の無操作タイムアウトによる部数および給紙段の自動的な初期化
- ウォーミングアップ中表示
- 給紙用トレーの用紙サイズ変更 (A3，B4，A4，A4R のいずれか)

[2] ジャムを起こした用紙を全て取り除いた後，ジャムが発生した時点からコピーを再開する機能.

B.3.3 提供しない機能

一般的な複写機が持つ機能で CP-KT1 が提供していない機能を以下に示す.

- 手差しトレイからの給紙
- 両面コピー機能
- オートシートフィーダによる連続スキャン
- 拡大縮小コピー機能
- その他 (応用操作となるもの)

B.3.4　機器構成

図 B.1に CP-KT1 の概観図を示す.

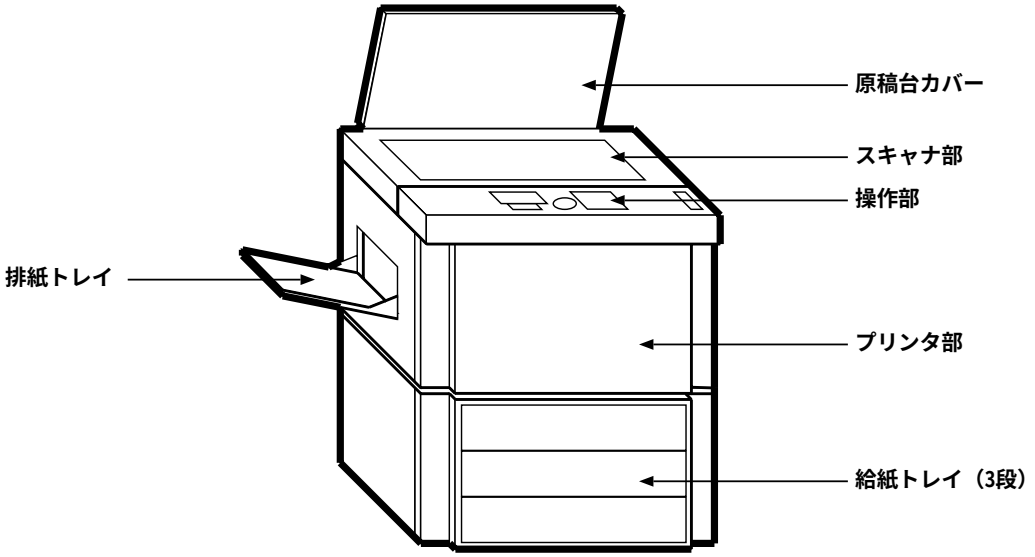


図 B.1: CP-KT1 の概観図

それぞれの部分の主な役割は次の通り.

スキャナ部　原稿台の上に置いた原稿を読み込む.

プリンタ部　スキャン部が読み込んだ原稿の画像を用紙に転写する.

給紙トレイ部 用紙を給紙する.

操作部　操作者が CP-KT1 を操作するためのボタン群とメッセージ表示用 LCD.

図 B.2に CP-KT1 の機器構成を示す.

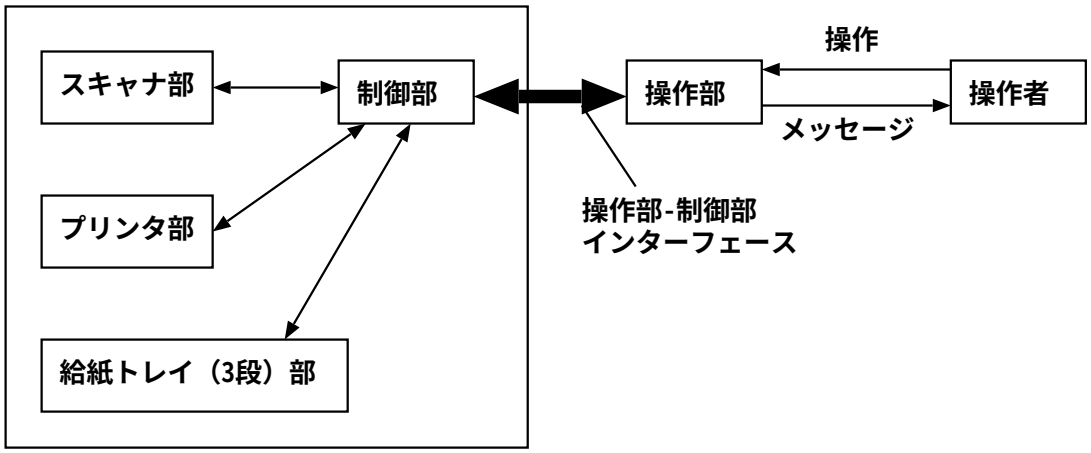


図 B.2: CP-KT1 の機器構成図

図 B.2に示すように， CP-KT1 は操作部とスキャナ部，プリンタ部，給紙トレイ部を制御する制御部からなる．各部は個別に組み込みプログラムによって制御されており，制御部は制御対象各部の組込みプログラムと通信することによって各機器を制御するようになっている．また，制御部と各部との通信は操作部には見えない．そして，操作部はこれらとは別の組み込みプログラムによって制御するようになっている．操作部は，制御部との通信および操作者との対話的なやりとりによって動作する．

B.4　複写機 CP-KT1 の操作部の概要

B.4.1　操作部の構成

図 B.3に CP-KT1 の操作部の構成図を示す.

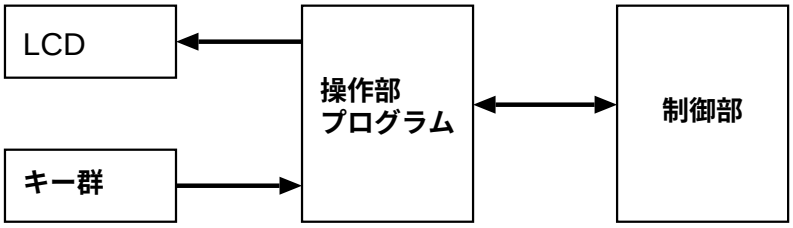


図 B.3: CP-KT1 の操作部の構成図

それぞれの部分の主な役割は次の通り.

LCD　表示用 LCD パネル．操作部プログラムからの出力を表示する．

キー群　操作用のキー群．操作者からの操作を操作部プログラムへの入力とする．

操作部プログラム　操作部の中心．CPU，RAM，ROM，クロック，その他で構成する．

制御部　プリンタ部，スキャナ部，給紙トレー部の制御をつかさどる．

B.4.2　操作パネルの構成

操作パネルの意匠と基本操作はあらかじめ策定されているとする．
図 B.4に CP-KT1 の操作パネルの概観図を示す．

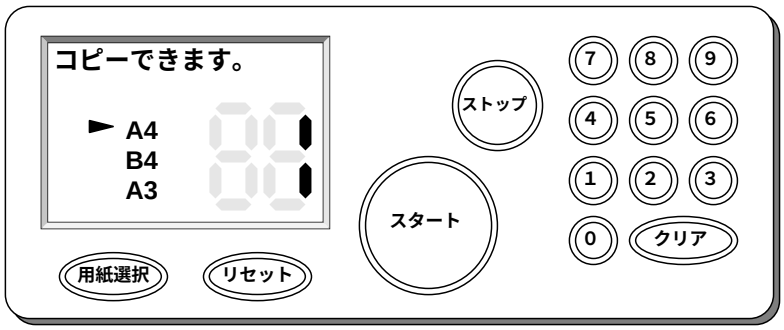


図 B.4: CP-KT1 の操作パネルの概観図

それぞれの部分の主な役割は次の通り．

メッセージ表示部　1/4VGA(320 ドット× 240 ドット) サイズの LCD パネルを使用する．ここには用紙サイズや部数の設定を表示する．また，操作者に伝えるメッセージを表示する．

スタートキー　コピーを開始する時押すキー．

ストップキー　コピー中の時にコピーを中断するキー．

用紙選択キー　コピーに使用する用紙のサイズ (用紙の入ったトレーの段) を選択する．

リセットキー　用紙選択や部数の設定を初期化する．

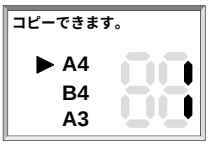
テンキー部　部数を設定する数字キー，部数設定初期化するクリアキーからなる．

B.5　操作部の基本操作方法仕様

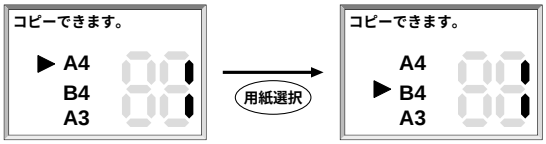
操作パネルの意匠と基本操作はあらかじめ策定されているとする.

B.5.1　通常のコピーの場合

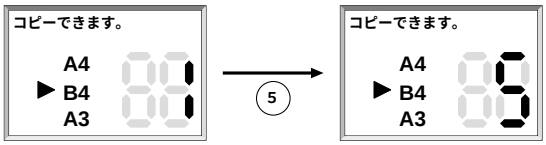
1. コピー可能な状態で待機中とする.「コピーできます。」のメッセージを表示.



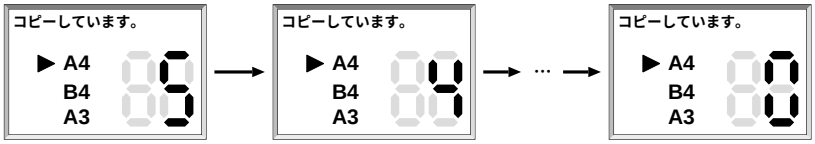
2. 原稿を原稿台に載せる.
3. 原稿台カバーを閉める.
4. 用紙選択キーを押してコピー用紙を選択する (標準は給紙トレー部最上段のトレーの用紙となる).



5. テンキーから部数を置数する (クリアキーを押した場合は 1 部となる).



6. スタートキーを押す.
7. コピー動作が始まる.「コピーしています。」のメッセージ表示.



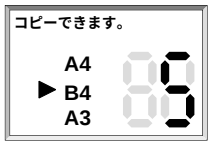
8. コピーされた用紙が一枚排紙されるごとに，置数されている部数の値を 1 ずつ減少し，残り部数を表示する．

9. 残り部数が 0 となり，全てコピーが完了したら，コピー動作をやめる．

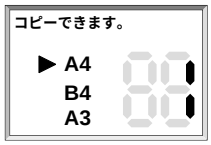
10. 部数の設定はコピー開始時に置数した値のままとする．

11. 用紙選択の設定はコピー開始時に選択した値のままとする．

12. そのままコピー可能な状態で待機する．「コピーできます。」のメッセージを表示．

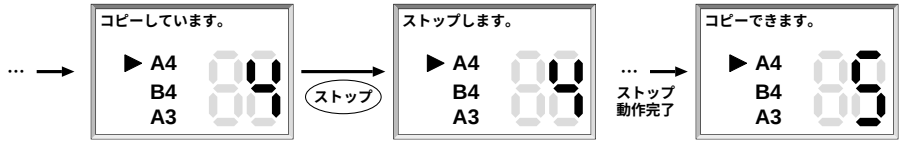


13. 1 分間操作がなかった場合には，置数を 1 とし，用紙選択を最上段のトレーにする (リセットキーを押した場合と同様の動作)．



B.5.2　コピー中に中断 (ストップ) する場合

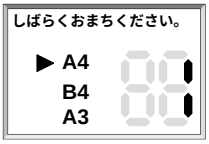
1. コピー動作までは通常の場合の操作方法に同じ.
2. 中断したいときはストップキーを押す.
3. 制御部がストップ動作を開始したら,「ストップします。」を表示する.



4. やがてコピー動作が止まる.
5. 部数の設定はコピー開始時に置数した値のままとする.
6. 用紙選択の設定はコピー開始時に選択した値のままとする.
7. そのままコピー可能な状態で待機する.「コピーできます。」のメッセージを表示.
8. 1 分間操作がなかった場合には，置数を 1 とし，用紙選択を最上段のトレーにする (リセットキーを押した場合と同様の動作).

B.5.3　ウォーミングアップ中表示

1. 電源投入後，制御部がコピー可能な状態になるまでは,「しばらくおまちください。」のメッセージを表示する．



2. 用紙サイズの表示は，トレー情報 (各給紙用トレーの用紙サイズ、用紙のあり/なしなど) を取得でき次第表示する．この場合もメッセージは「しばらくおまちください。」のままとする．

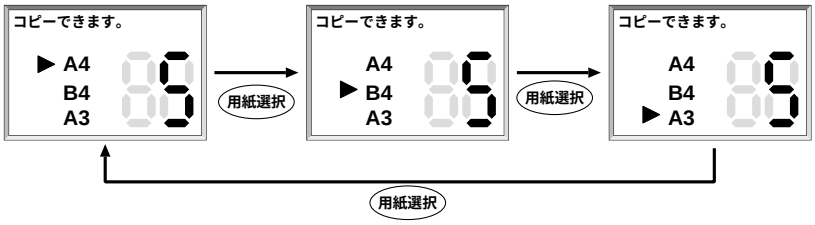
3. ウォーミングアップ中は、いずれのキーも操作できない．

4. ウォーミングアップが完了したら、通常のコピーの操作方法に移る．

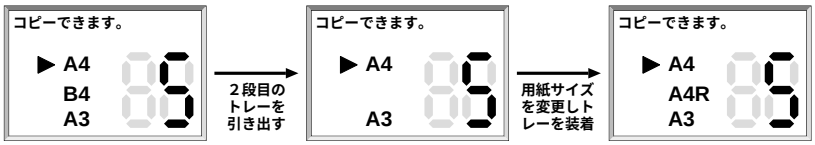
B.5.4　用紙選択操作，給紙操作

1. 用紙選択は，ウォーミングアップ中かコピー可能状態で待機中でなければ操作できない．(「しばらくおまちください。」か「コピーできます。」のメッセージを表示している．)

2. 一度押すと選択しているトレーは一段下に変わり，最下段でもう一度押すと，最上段を選択する．



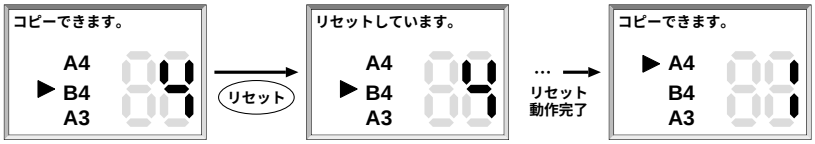
3. 給紙用トレーの用紙サイズを変更し，再び装着すると，LCD 上にも新しいサイズを表示する．コピー動作中に給紙用トレーを引き出すと給紙トレー部に (場合によってはプリンタ部も) ジャムが発生する (コピー中に用紙なしのメッセージによって給紙を促されている場合を除く).



B.5.5　リセット操作

1. リセットキーは、コピー可能状態で待機中でなければ操作できない．(「コピーできます。」のメッセージを表示している．)

2. リセットキーを押す.

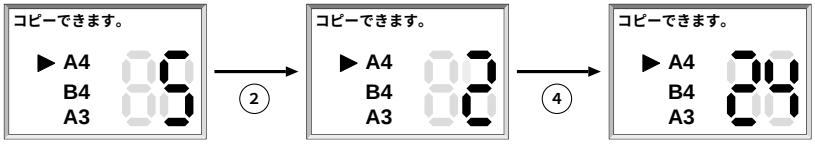


3. 制御部がリセット動作を開始したら,「リセットしています。」を表示する.

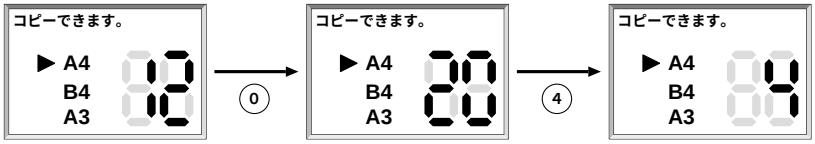
4. リセット動作が完了したら，コピー可能状態で待機中になる．(「コピーできます。」のメッセージを表示している．)

B.5.6　テンキー操作

1. コピー/リセット/クリアキー入力などの直後の最初の数字キー入力時には，現在の設定値を置き換えるように表示が変化する．

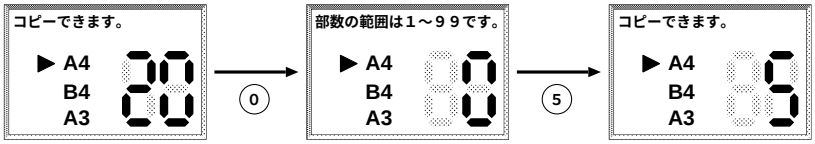


2. ひとたびテンキーの入力を始めたら，数字キーを 1 度押すたびに 1 桁ずつ右端に表示され，1 回前の入力値は 1 桁左へ送られて表示される．これで 2 桁の数値を表す．



3. 左へ送る桁が 0 の場合は，表示せずに今回の入力だけを表示する．

4. コピー可能な状態で設定値が 1 から 99 の範囲では，スタートキーを押すとコピーを開始できる．それ以外 (0 のとき) は「部数の範囲は 0〜99 です。」というメッセージを表示し，設定値が 1 から 99 の範囲になるまでスタートキーは押せない．



B.5.7　ジャム (用紙詰まり) 発生時の操作

1. ジャム発生箇所として以下の 3 箇所を報告する.

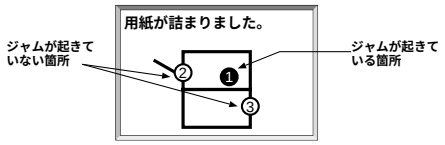
   (a) プリンタ部の排紙口部分

   (b) プリンタ部の用紙引き込みおよび転写部分

   (c) 給紙トレー部の用紙引き出し部分

2. ジャム発生時には「用紙が詰まりました。」のメッセージを表示し，発生箇所を示す図を表示する．ジャム中は、いずれのキーも操作できない.

3. 詰まった用紙を取り除くには CP-KT1 の各部のドアを開けなければならない．安全対策のため，いずれかのドアを開けたときにいったん電源が切れる．全てのドアを閉じると自動的に再び電源が入る.

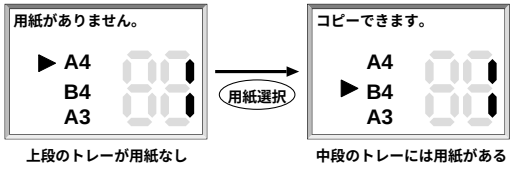


4. 電源投入時にジャム中かどうかをしらべ，全ての発生箇所の用紙詰まりが解消されるまで，ジャム箇所の番号付き印を更新しながら上記のメッセージを表示し続ける.

5. 全ての発生箇所の用紙詰まりが解消されると，コピー可能状態になる．リカバリ機能は提供しないので，リセット操作をした場合と同じ状態になる.

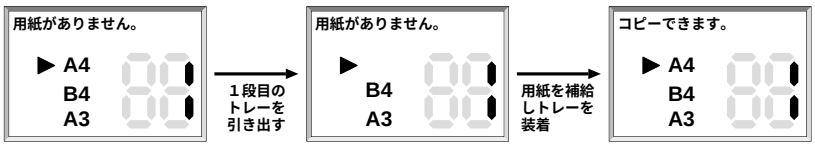
B.5.8　コピー中でないときの用紙なしの場合の操作

1. 用紙選択で用紙なしの状態にある給紙用トレーを選択すると「用紙がありません。」というメッセージを表示する．このときスタートキーは操作できない．



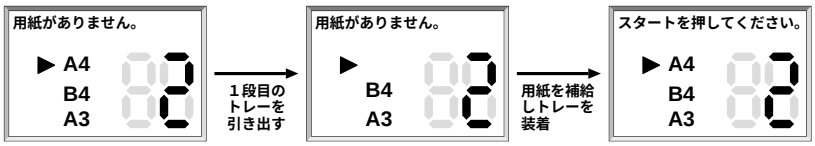
2. 用紙の入っている給紙用トレーを選択すれば,「コピーできます。」のメッセージを表示し、スタートキーを操作してコピーできる．

3. 紙なしになった給紙用トレーに用紙を補給し，再び装着すると,、「コピーできます。」のメッセージを表示し、スタートキーを操作してコピーできる．



B.5.9　コピー中の用紙なしの場合の操作

1. コピー中に使用中の給紙用トレーが用紙なしの状態になると「用紙がありません。」というメッセージを表示し，コピー動作が中断する．



2. 部数の設定は用紙なしになった時点の値のままとする．数字キー，クリアキーなどは操作できない．

3. 用紙選択の設定は用紙なしになった時点の値のままとする．用紙選択キーは操作できない．

4. 当該給紙用トレーに用紙を補給し，装着すると「スタートを押してください。」というメッセージを表示する．

5. ここでスタートキーを操作すると，用紙なしの時点からコピーを継続できる．

6. ストップキーを操作すると，コピー中の中断動作を実行する．ストップ動作完了後は，部数，用紙選択がコピー開始時点の値に戻り，コピー中でないときの用紙なしの場合と同様に操作できるようになる．

B.6　操作部-制御部間インターフェース仕様

操作部と制御部の間で情報を交換するためのコマンドインターフェースを定めた仕様を以下に示す．これは，CP-KT1 制御部の開発部門によって提案された仕様案である．操作部の分析/設計の際に必要が生じれば，制御部の開発部門に変更/追加/ 修正を提案できることになっている．

B.6.1　操作部-制御部間インターフェースの概要

図 B.5 に CP-KT1 の操作部-制御部間インターフェースの構成概略図を示す．CP-KT1 の操作部-制御部間インターフェースは，ハードウェア的には RS-232C インターフェースによるシリアル通信 (9600bps) となっている． CP-KT1 の操作部-制御部間の通信では，操作部が主となってコマンドを送り制御部がレスポンスを返す方法をとる．

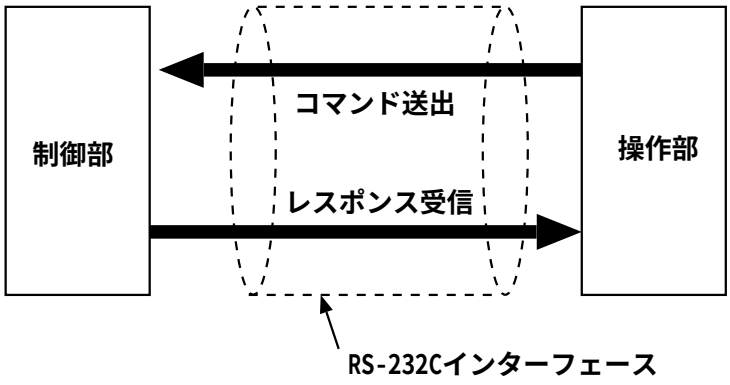


図 B.5: 操作部-制御部間インターフェースの構成概略図

CP-KT1 は，操作者の指示を操作部が受け付け必要な指示を制御部に送ることで基本的な動作をする．この動作のために必要な操作部-制御部間インターフェースは，操作部からコマンドを送出することで実現する．一方，操作部は何らかの方法で制御部の動作状態情報 (ステータス情報) を取得し，これを元にメッセージを表示して操作者に動作状態を伝えなければならない． CP-KT1 では，ステータス情報の取得はステータス要求用のコマンドを送出し，そのレスポンスとしてステータス情報を得る方法をとる．操作部は，定期的にステータス要求コマンドを送出し，制御部の動作状態を監視しなければならない．

B.6.2　操作部-制御部間コマンド インターフェースの概要

操作部の送出するコマンドには，大別して以下の 4 つのものがある．個々のコマンドの詳細については次節で述べる．

コピー設定用　　コピー部数設定，給紙段選択など．

コピー操作用　　コピースタート，コピーストップ，リセットなど．

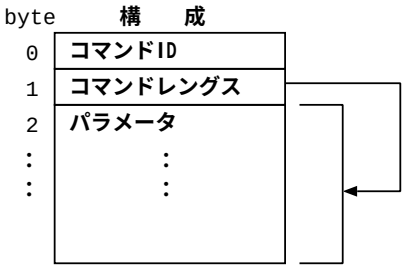
ステータス取得要求用　各種動作状態取得用．

コピー中動作状態表示用　コピー中の部数進捗表示用．

コマンドの基本的な構成を以下に示す．

B.6.2.1　コマンドの構成

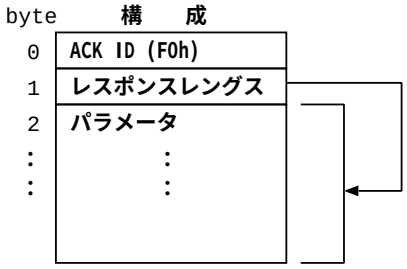
コマンドは，以下の図に示すような構造を持つ．



- 先頭バイトにコマンド ID を格納する．
- 2 バイト目に 3 バイト目以降に続くコマンドパラメータ全体の長さ (コマンドレングス) を格納する．
- パラメータのないコマンドではコマンドレングスは 0 とする．

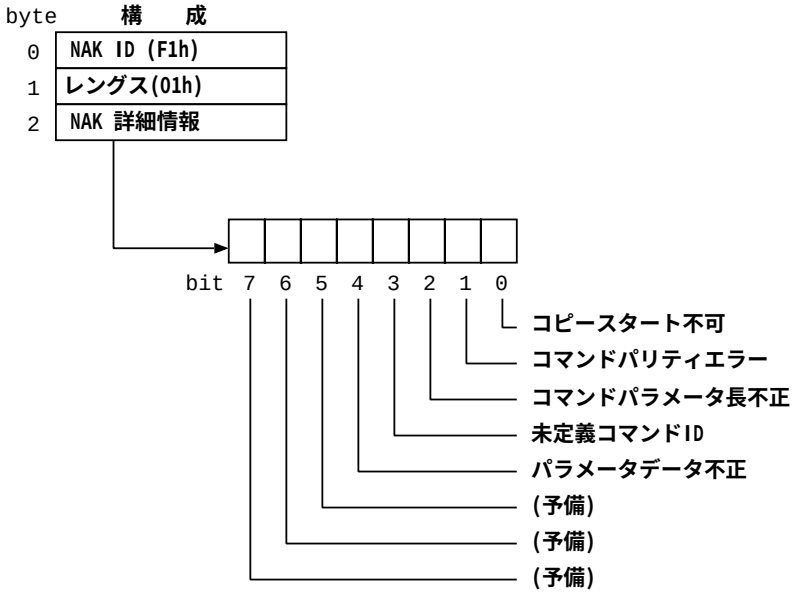
B.6.2.2 コマンド レスポンスの構成

1. 制御部からの正常な応答 (ACK 応答) の場合.



   - 先頭バイトに ACK ID を格納する.
   - 2 バイト目に 3 バイト目以降に続くレスポンスパラメータ全体の長さ (レスポンスレングス) を格納する.
   - パラメータのないレスポンスではレスポンスレングスは 0 とする.

2. 制御部からの不正な応答 (NAK 応答) の場合.



   - 先頭バイトに NAK ID を格納する.
   - 2 バイト目に 3 バイト目以降に続くレスポンスパラメータ全体の長さ (レスポンスレングス) を格納する.

- NAK レスポンスでは，レスポンスパラメータは 1 バイトの NAK 詳細情報を示すビット列である．また，レスポンスレングスは 1 である．
- NAK 詳細情報の内容は，制御部が当該コマンドの要求した動作ができない状態にある場合のエラー，コマンド送信時のパリティエラー，コマンドパラメータ長の不正，未定義コマンド ID の場合のエラー，設定パラメータ内容の不正エラーを示す各ビットである．各ビットとも ON(=1) のときにその状態にあることを示す．

3. 一定時間待っても制御部から応答がない場合．

- 送信したコマンドに対して 1000ms(=1s) 経過しても応答がない場合については，操作部側で判断し，通信タイムアウトエラーの処理をすること．

B.6.3　操作部-制御部間コマンド インターフェースの詳細

B.6.3.1　ステータス取得要求 (11h)

1. コマンド

byte	構　成
0	11h
1	00h

2. レスポンス

byte	構　成
0	F0h
1	05h
2	コピー状態
3	コピーサブ状態1
4	トレー情報1
5	トレー状態2
6	トレー状態3

3. 詳細

- コピー状態

byte	
2	コピー状態

00h (予備)
01h ウェイト中
02h スタンバイ
04h コピー中
08h ジャム(用紙詰まり)中
14h コピー中の紙無し状態
FFh リセット中
以下 (上記の値以外は予備とする)

- コピーサブ状態

byte

3	コピーサブ状態

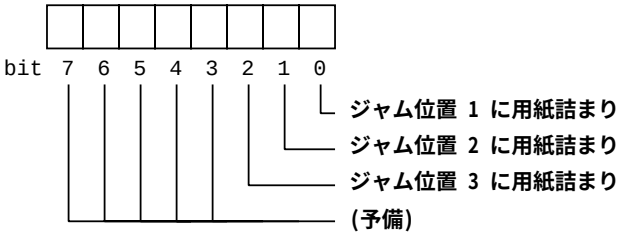
コピー状態が 04h(コピー中)のとき

08h ストップ動作中("ストップします。")

07h コピー動作中("コピーしています。")

コピー状態が 08h(用紙詰まり中)のとき

各ビットが ON(=1)のときにその状態にあることを示す.

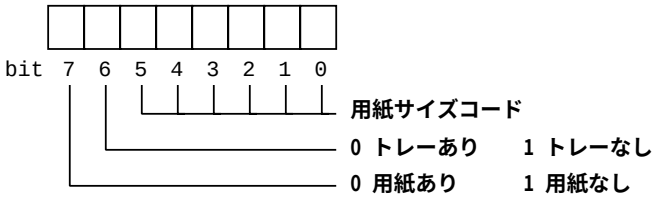


例えばビット並びが 00000101 のとき ジャム位置 1 と 3 に用紙詰まりが発生している.

コピー状態が 03h(コピー中)、08h(用紙詰まり中)以外のとき

80h 用紙無し状態("用紙がありません")

- トレー情報 1, 2, 3



用紙サイズコード

```
100100  A4
010100  B4
101100  A4R
100011  A3
```

その他の値には上記以外の用紙のコードが割り当てられている.

トレー情報のバイト全体で FFh を示しているときは,
当該トレーが無い(抜けている)状態で,かつ用紙無しの状態を示している.
トレーの有無を用紙の有無に優先して評価すること.

4. 本コマンドは操作部が恒常的に制御部を監視するために使用する．

5. 本コマンドを 400ms 程度の間隔をもって送信すれば，制御部の動作状態を監視できる．制御部は，操作部が 400ms 程度の間隔で本コマンドを送信することを前提に，制御部の内部状態を管理し，変化させることとする．

B.6.3.2 プリンタ設定 (21h)

1. コマンド

byte	構　成
0	21h
1	02h
2	プリント枚数
3	給紙段

給紙段の指定は以下の通り.

01h トレー位置 1 (上段)
02h トレー位置 2 (中段)
03h トレー位置 3 (下段)

2. レスポンス

byte	構　成
0	F0h
1	00h

3. 詳細

- コピースタートコマンド (31h) を送信する前に本コマンドを送信し，制御部に部数と給紙段 (トレー位置) を確定させること。

B.6.3.3 コピースタート (31h)

1. コマンド

byte	構　成
0	31h
1	00h

2. レスポンス

byte	構　成
0	F0h
1	00h

3. 詳細

- 本コマンド送信直前にプリンタ設コマンド (21h) を送信し ACK を受け取ること．同コマンドが NAK であれば本コマンドも実行できない．
- 本コマンドによって制御部はコピー動作を開始する．
- 本コマンドによってコピー動作が正常に開始されると，コピー状態はスタンバイ (02h) →コピー中 (04h) と変化し，コピーサブ状態 1 はコピー動作中 (03h) となる．
- コピー状態が中断中 (14h) のとき，再開には本コマンドを送信する．このときコピー状態は中断中 (14h) →コピー中 (04h) と変化する．

B.6.3.4 コピーストップ (32h)

1. コマンド

byte	構　成
0	32h
1	00h

2. レスポンス

byte	構　成
0	F0h
1	00h

3. 詳細

- コピー状態がコピー中 (04h) か中断中 (14h) のときに本コマンドを送信すると，制御部はストップ動作を試みようとする．
- 本コマンド受理後の制御部の動作は，ストップキー押下のタイミングの違いによって異なる．(これはすでに給紙トレーから送り出された用紙についてはコピーを完了することを原則としたためである．)
  (a) すでにコピーが終りかけているなどのため本コマンドの効果がないとき．この場合はコピー状態はコピー中 (04h) のままで，コピーサブ状態はストップ動作中 (08h) にならず，コピー動作中 (07h) のままである．
  (b) 本コマンドの効果があるとき．コピー状態はコピー中 (04h) のままで，コピーサブ状態はストップ動作中 (08h) になる．

B.6.3.5 表示情報取得 (41h)

1. コマンド

byte	構　成
0	41h
1	00h

2. レスポンス

byte	構　成
0	F0h
1	01h
2	**表示用置数**

表示用置数には、コピー中の表示すべき置数が格納されている。

3. 詳細

- コピー中の表示すべき部数の値 (コピーの残り部数) を取得する.
- 詳細ステータスのコピー状態がコピー中 (04h), コピーサブ状態 1 がコピー動作中 (07h) かストップ動作中 (08h) のときに本コマンドを送る.
- おおむね 200ms ごとに送れば給排紙動作に同期した値が維持できる.

B.6.3.6 トレー情報取得 (42h)

1. コマンド

byte	構　成
0	42h
1	00h

2. レスポンス

byte	構　成
0	F0h
1	01h
2	トレー選択情報

トレー選択情報には、制御部側が設定したトレー位置が格納されている。

01h　トレー位置 1 (上段)
02h　トレー位置 2 (中段)
03h　トレー位置 3 (下段)

3. 詳細

- 用紙選択キーが押されたときに，現在選択中のトレー位置を取得する．
- 用紙選択キーを押す動作と同期させるには，本コマンドを数 100ms ごとに送る必要がある．
- 本コマンドを 1 回送ると，次のトレーを選択する動作を制御部が実行し，その結果をレスポンスに返す．操作部はこのときのレスポンスの値を評価して選択されているトレーを表示する．

B.6.3.7 リセット (A0h)

1. コマンド

byte	構　成
0	A0h
1	00h

2. レスポンス

byte	構　成
0	F0h
1	00h

3. 詳細

- 本コマンドの実行には数 100ms 程度の時間がかかる．操作部は，コマンド送信とともにリセット中のメッセージを表示すること．
- 本コマンド送信後，制御部は ACK 応答を返すまでにコピー状態をリセット中 (FFh) にする．
- 再びコピー状態がスタンバイ (02h) になったことをもってリセット動作が完了したとみなすこと．
- 本コマンドの実行によって，トレー選択位置と部数の設定は標準値に戻る．
- 操作部は部数の現在設定値を標準値に戻すこと．

第 C 章

複写機 CP-KT1 の操作部プログラムの要求モデル

本章では，付録 B に示した複写機 CP-KT1 の操作部の要求仕様文書をもっとよく理解し，要求を分析するために要求モデルを作成する．

C.1 アクタの識別

まず，複写機 CP-KT1 の操作部のアクタを識別する．図 B.2 ，図 B.3 から，制御部がアクタと識別できる．また， B.5 節の基本操作仕様の記述から，操作部を使用する操作者がアクタと識別できる．したがって，操作部と対話するアクタは下記のものになる．

- 操作者
- 制御部

したがって，最初のシステムの概観図は図 C.1のようなものになった．
また，システム内部の直感的な概観図は，図 C.2のようなものになった．

図 C.1: 操作部の最初のシステム概観図

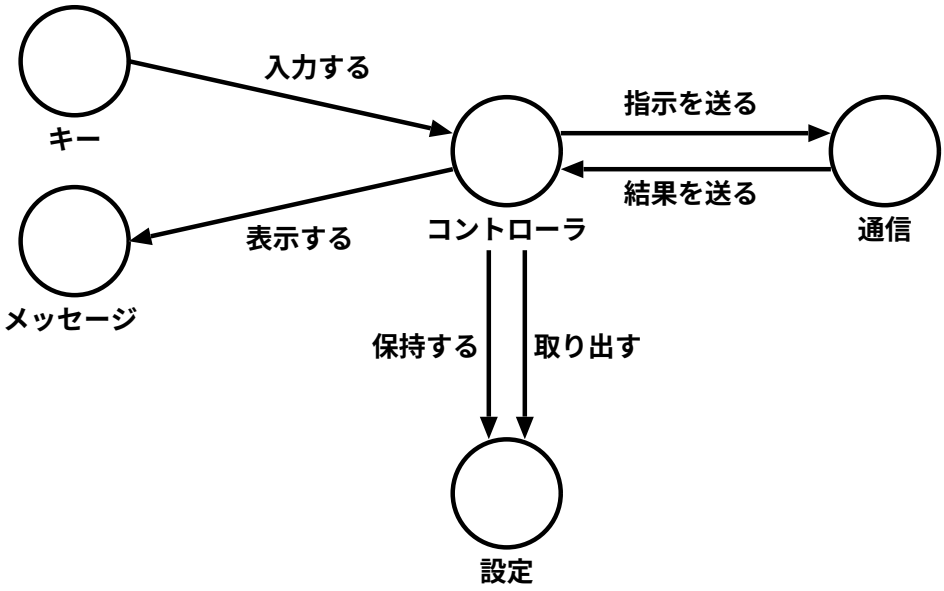


図 C.2: 操作部のシステム内部の最初の概観図

C.2　インターフェース記述

本事例では，B.4.2の図 B.4 に示したように， CP-KT1 の操作パネルの意匠はデザイン部門の要求によって決まっている．このユーザーインターフェースは，プロトタイプとして十分なものであると判断した．したがって use-case を記述し各部門と議論するために，インターフェース記述としてデザイン部門の要求したユーザーインターフェースの仕様を用いる．また，B.6.1の図 B.5 に示した CP-KT1 の操作部-制御部間インターフェースの構成概略図から， CP-KT1 の操作部-制御部間の通信のためのインターフェースが識別できる．

要求仕様から識別できたインターフェース記述を下記に示す.

1. **表示に関わるインターフェース記述**
   (a) メッセージ表示
   (b) 用紙選択表示
   (c) 部数表示
2. **ユーザー入力に関わるインターフェース記述**
   (a) 用紙選択キー
   (b) リセットキー
   (c) スタートキー
   (d) ストップキー
   (e) テンキー
   (f) クリアキー
3. **制御部との通信に関わるインターフェース記述**
   (a) 通信ポート

C.3　use-case の識別

C.1節において識別したアクタについて考察し，システムとしての操作部に対してどのように働きかけるのかを調査する．

C.3.1　操作者をアクタとした use-case

まず，操作者をアクタとした場合のインターフェス画面は要求文書として図 B.4に示してある．これをユーザーインターフェース (ここでは操作者のユーザーインターフェース) のプロトタイプとみなす．

操作者をアクタとした use-case は，B.5節に記述してある操作部の基本操作方法仕様から得ることができる．基本操作仕様には下記の操作を記述してある．

1. **通常のコピーの場合**
2. **コピー中に中断 (ストップ) する場合**
3. **ウォーミングアップ中表示**
4. **用紙選択操作，給紙操作**
5. **リセット操作**
6. **テンキー操作**
7. **ジャム (用紙詰まり) 発生時の操作**
8. **コピー中でないときの用紙なしの場合の操作**
9. **コピー中の用紙なしの場合の操作**

これらは，各操作の中の 1 操作段階ずつを箇条書にして記述してあるので，ほぼこのまま use-case として利用できる．

整理すると下記のようになる．

1. **電源投入からスタンバイ状態までの初期化段階**
   (a) ウォーミングアップ中表示

2. **コピースタート前の段階**

   (a) 用紙選択

   (b) リセット操作

   (c) テンキー操作 (= 部数設定操作とみなす)

   (d) コピー中でないときの用紙なしの場合の操作

3. **コピースタート後の段階**

   (a) 通常コピーの場合

   (b) コピー中に中断 (ストップ) する場合

   (c) ジャム (用紙詰まり) 発生時の操作

   (d) コピー中の用紙なしの場合

「通常コピーの場合」のうち,「用紙選択操作」,「部数設定操作」は, コピースタート前の段階に含まれる記述を付け足しただけのものなので,「通常コピーの場合」から「用紙選択操作」,「部数設定操作」の記述を省いてよい. あるいは、「通常コピーの場合」use-case のなかで「用紙選択操作」,「部数設定操作」の 2 つの use-case が利用されていると考えることもできる。また,「コピー中に中断 (ストップ) する場合」,「ジャム (用紙詰まり) 発生時の操作」,「コピー中の用紙なしの場合」は,「通常のコピーの場合」の代替系列とみなすこともできる.

C.3.2 制御部をアクタとした use-case

制御部をアクタとした use-case は, B.6節に記述してある操作部-制御部間インターフェース仕様から得ることができる. 操作部-制御部間インターフェース仕様には下記のコマンドが記述してある.

1. **ステータス取得要求 (11h)**

2. **プリンタ設定 (21h)**

3. **コピースタート (31h)**

4. **コピーストップ (32h)**

5. **表示情報取得 (41h)**

6. **トレー情報取得 (42h)**

7. **リセット (A0h)**

制御部とのやりとりでは，上記各コマンドとそれらのレスポンスの組がひとつの単位になっているので，ほぼこのまま use-case として利用できる．

整理すると下記のようになる．

1. **恒常的にやりとりするコマンド**

   (a) ステータス取得要求 (11h)

2. **コピースタート前のコマンド**

   (a) リセット (A0h)

   (b) トレー情報取得 (42h)

   (c) プリンタ設定 (21h)

3. **コピースタート以後のコマンド**

   (a) コピースタート (31h)

   (b) コピーストップ (32h)

   (c) 表示情報取得 (41h)

C.4 問題ドメインオブジェクトモデル

これまでに，操作部の環境を反映したオブジェクトや use-case を識別した．これらの情報をもとにまとめた問題ドメインオブジェクトモデルを図 C.3 に示す.

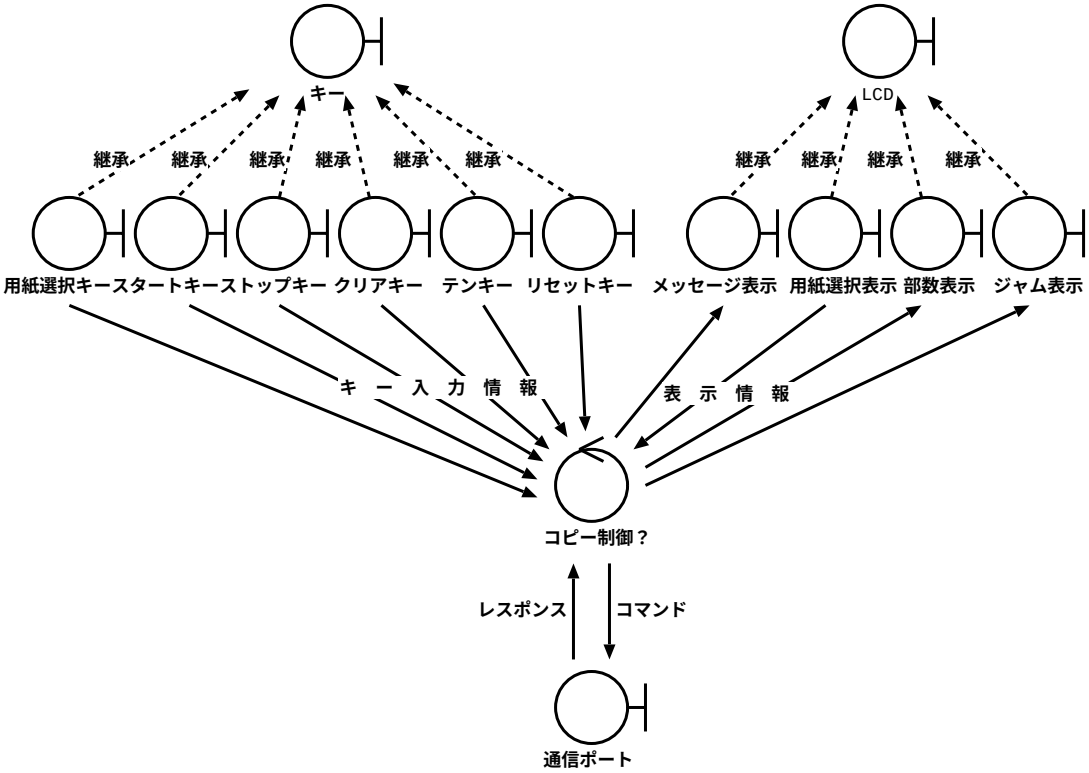


図 C.3: 最初の問題ドメインオブジェクトモデル

図 C.3 には次のオブジェクトを表現してある.

- 操作者からの入力を担当するインターフェース記述から識別した抽象的なオブジェクト「キー」.
  これを継承し各々の役割に特化した「用紙選択キー」,「リセットキー」,「スタートキー」,「ストップキー」,「テンキー」,「クリアキー」.
- 操作者への情報表示出力を担当するインターフェース記述から識別した抽象的なオブジェクト「LCD」.

これを継承し各々の役割に特化した「メッセージ表示」,「用紙選択表示」,「部数表示」,「ジャム (用紙詰まり) 表示」.

- 制御部との通信を担当する「通信ポート」.
- 上記のインターフェース記述を介してコピー動作の制御を司る制御オブジェクト「コピー制御」.
  ただしこの段階での「コピー制御」オブジェクトの洗練は十分ではなく，異なるオブジェクトが担う役割であるとか，複数の制御オブジェクトに分割すべきものであるといった可能性が残っている.

この問題ドメインオブジェクトモデルは，今後の本事例のシステムに関する議論のために引続き用いることができる.

第 D 章

複写機 CP-KT1 の操作部プログラムの分析モデル

本章では，付録 C に示した複写機 CP-KT1 の操作部の要求モデルを基に，分析モデルを作成する．

D.1 分析モデル作成の進め方

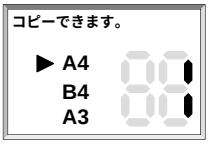
分析モデルでは， use-case に記述した振舞いを分析モデルのオブジェクトに割り当て，どの use-case にどのオブジェクトが責任を持つのかを明確にする．

分析作業は，use-case のひとつずつについて，インターフェースオブジェクト，実態オブジェクト，制御オブジェクトを識別することを繰り返すことで実施する．

D.2 use-case 通常コピーの場合

それぞれの操作段階について識別できるオブジェクトについて議論する．

1. コピー可能な状態で待機中とする．「コピーできます。」のメッセージを表示．



(a) 操作者にメッセージを表示するためにインターフェースオブジェクト「メッセージ表示」が必要である.

(b) メッセージ「コピーできます。」を表示するためには，制御部の動作状態を取得し，スタンバイ状態であることを確認する必要がある．このために制御部をアクタとした use-case「ステータス取得要求 (11h)」を使用し，現在の動作状態を取得する.

(c) 制御部の動作状態を解析するのはインターフェースオブジェクト「メッセージ表示」の責任とするのは適切ではない．この操作はインターフェースオブジェクトの種類によらず実行できるべきである．そこで制御オブジェクト「ステータス制御」を導入し，動作状態の解析を担当させることにする.

(d) この操作段落の検討から得られたオブジェクトを図 D.1 に示す.

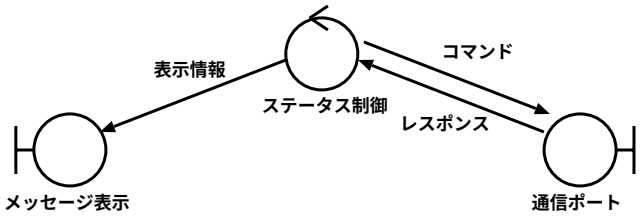


図 D.1: 通常コピーの場合から得られた分析オブジェクト (1)

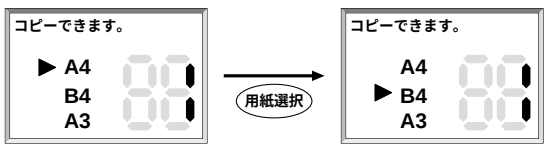
2. 原稿を原稿台に載せる.

(a) この操作は，操作部の動作には無関係なので議論しない.

3. 原稿台カバーを閉める.

(a) この操作は，操作部の動作には無関係なので議論しない.

4. 用紙選択キーを押してコピー用紙を選択する (標準は給紙トレー部最上段のトレーの用紙となる).



(a) 操作者の入力のためにインターフェースオブジェクト「用紙選択キー」が必要と考えられるが，用紙選択キー以外のキーの存在も考慮し，ここではインターフェースオブジェクト「キー」とした．操作者が用紙選択キーを押したときにはインターフェースオブジェクト「キー」が用紙選択キーが押されたことを識別する責任を持つことにする．

(b) 各トレーの用紙サイズを表示するには、インターフェースオブジェクト「用紙選択表示」が必要である．

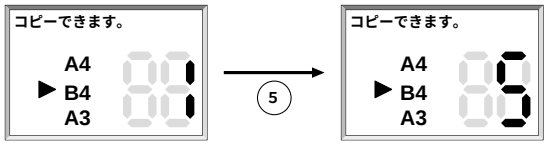
(c) 操作者の入力を制御部に伝えるには，制御部をアクタとした use-case 「トレー情報取得 (42h)」を使用する．

(d) use-case「トレー情報取得 (42h)」を実行するのをインターフェースオブジェクト「用紙選択キー」の責任にするのは適切ではない．この操作の結果は，インターフェースオブジェクト「用紙選択表示」に反映しなければならないからである．そこで制御オブジェクト「コピー制御」を導入し，コピー操作の制御部への伝達と結果の反映を担当させることにする．

(e) 各トレーの用紙サイズ情報は制御部から取得する必要がある．このために制御部をアクタとした use-case「ステータス取得要求 (11h)」を使用する．制御オブジェクト「ステータス制御」が動作状態を解析し，トレー情報をインターフェースオブジェクト「用紙選択表示」に伝えることとする．

(f) この操作段落の検討から得られたオブジェクトを図 D.2 に示す.

5. テンキーから部数を置数する (クリアキーを押した場合は 1 部となる).



(a) この操作段落の検討から得られたオブジェクトを図 D.3 に示す.

6. スタートキーを押す.

(a) この操作段落の検討から得られたオブジェクトを図 D.4 に示す.

7. コピー動作が始まる.「コピーしています。」のメッセージ表示.

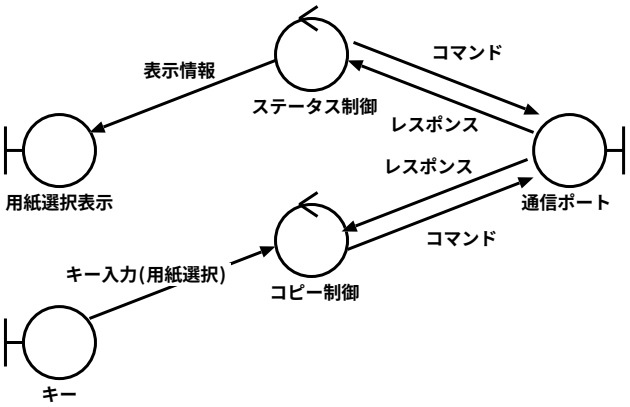


図 D.2: 通常コピーの場合から得られた分析オブジェクト (2)

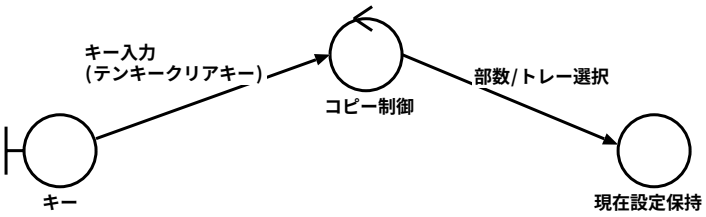
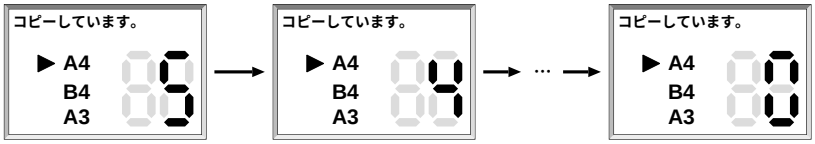


図 D.3: 通常コピーの場合から得られた分析オブジェクト (3)

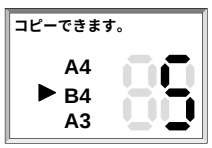


8. コピーされた用紙が一枚排紙されるごとに，置数されている部数の値を 1 ずつ減少し，残り部数を表示する．

9. 残り部数が 0 となり，全てコピーが完了したら，コピー動作をやめる．

   (a) この操作段落の検討から得られたオブジェクトを図 D.5 に示す．

10. 部数の設定はコピー開始時に置数した値のままとする．



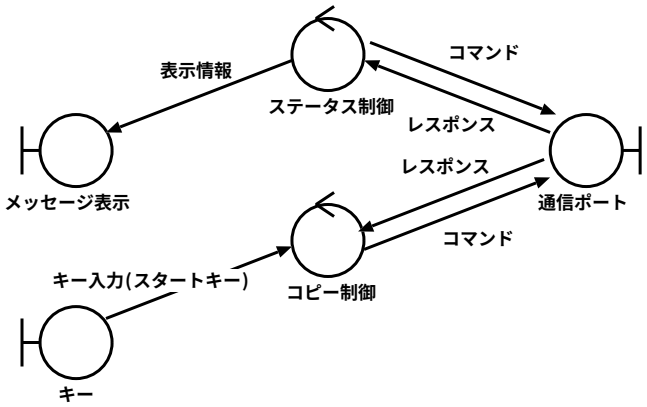


図 D.4: 通常コピーの場合から得られた分析オブジェクト (4)

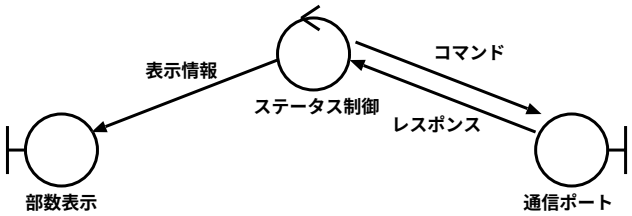


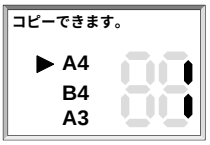
図 D.5: 通常コピーの場合から得られた分析オブジェクト (5)

11. 用紙選択の設定はコピー開始時に選択した値のままとする.

12. そのままコピー可能な状態で待機する.「コピーできます。」のメッセージを表示.

    (a) この操作段落の検討から得られたオブジェクトを図 D.6 に示す.

13. 1 分間操作がなかった場合には，置数を 1 とし，用紙選択を最上段のトレーにする (リセットキーを押した場合と同様の動作).



    (a) この操作段落の検討から得られたオブジェクトを図 D.7 に示す.

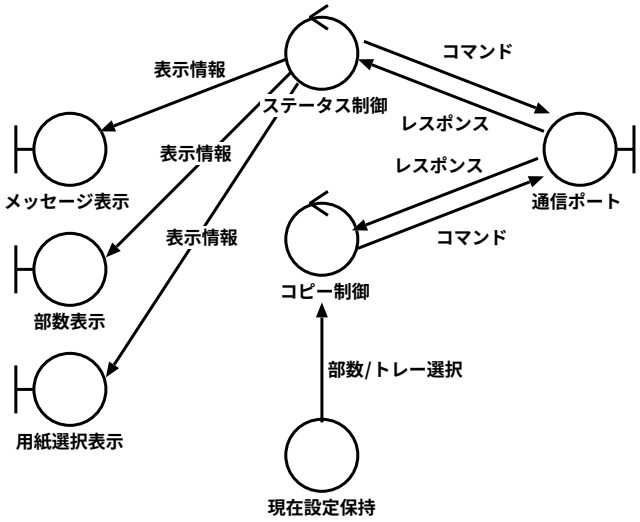


図 D.6: 通常コピーの場合から得られた分析オブジェクト (6)

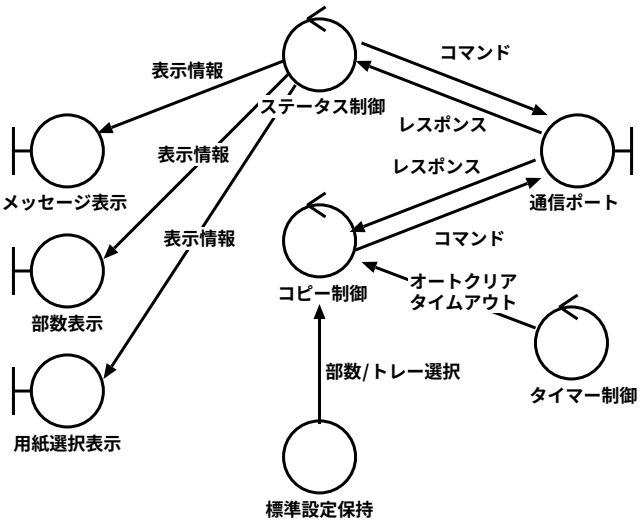


図 D.7: 通常コピーの場合から得られた分析オブジェクト (7)

D.3 use-case コピー中に中断 (ストップ) する場合

それぞれの操作段階について識別できるオブジェクトについて議論する.

1. コピー動作までは通常の場合の操作方法に同じ.

2. 中断したいときはストップキーを押す.

   (a) この操作段落の検討から得られたオブジェクトを図 D.8 に示す.

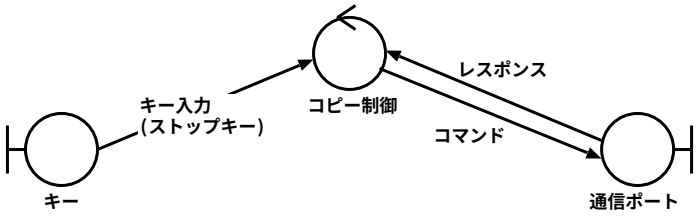
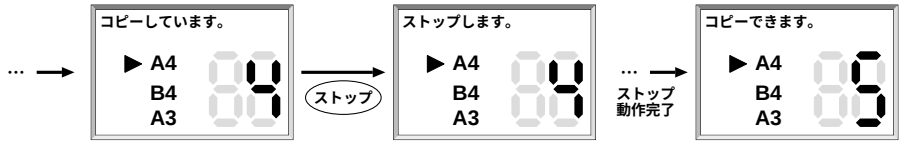


図 D.8: コピー中に中断する場合から得られた分析オブジェクト (1)

3. 制御部がストップ動作を開始したら,「ストップします。」を表示する.



4. やがてコピー動作が止まる.

   (a) この操作段落の検討から得られたオブジェクトを図 D.8 に示す.

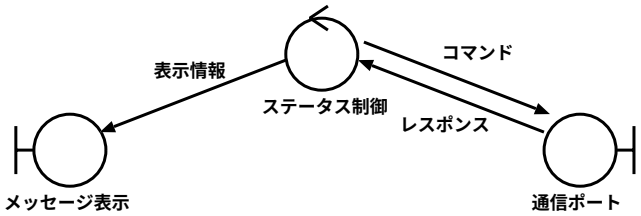


図 D.9: コピー中に中断する場合から得られた分析オブジェクト (2)

5. 部数の設定はコピー開始時に置数した値のままとする.

6. 用紙選択の設定はコピー開始時に選択した値のままとする.

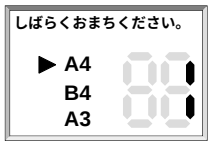
7. そのままコピー可能な状態で待機する.「コピーできます。」のメッセージを表示.

8. 1 分間操作がなかった場合には，置数を 1 とし，用紙選択を最上段のトレーにする (リセットキーを押した場合と同様の動作).

D.4 use-case ウォーミングアップ中表示

それぞれの操作段階について識別できるオブジェクトについて議論する.

1. 電源投入後, 制御部がコピー可能な状態になるまでは,「しばらくおまちください。」のメッセージを表示する.



2. 用紙サイズの表示は, トレー情報 (各給紙用トレーの用紙サイズ、用紙のあり/なしなど) を取得でき次第表示する. この場合もメッセージは「しばらくおまちください。」のままとする.
3. ウォーミングアップ中は、いずれのキーも操作できない.
4. ウォーミングアップが完了したら、通常のコピーの操作方法に移る.

   (a) この操作段落の検討から得られたオブジェクトを図 D.10 に示す.

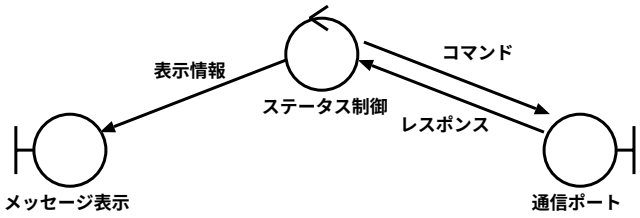


図 D.10: ウォーミングアップ中表示から得られた分析オブジェクト (1)

D.5 use-case 用紙選択操作，給紙操作

それぞれの操作段階について識別できるオブジェクトについて議論する．

1. 用紙選択は，ウォーミングアップ中かコピー可能状態で待機中でなければ操作できない．(「しばらくおまちください。」か「コピーできます。」のメッセージを表示している．)

   (a) この操作段落の検討から得られたオブジェクトを図 D.11 に示す．

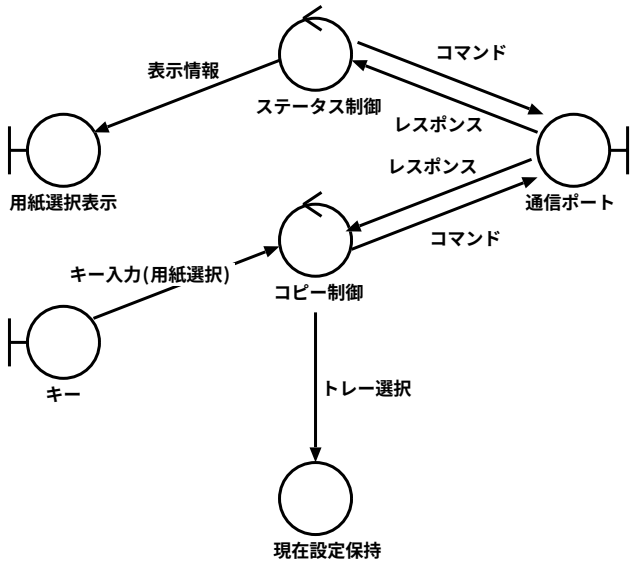
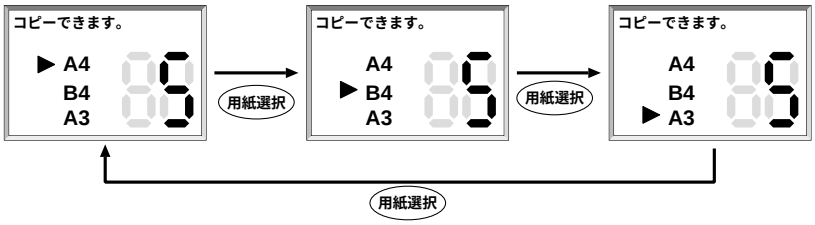


図 D.11: 用紙選択操作，給紙操作から得られた分析オブジェクト (1)

2. 一度押すと選択しているトレーは一段下に変わり，最下段でもう一度押すと，最上段を選択する．



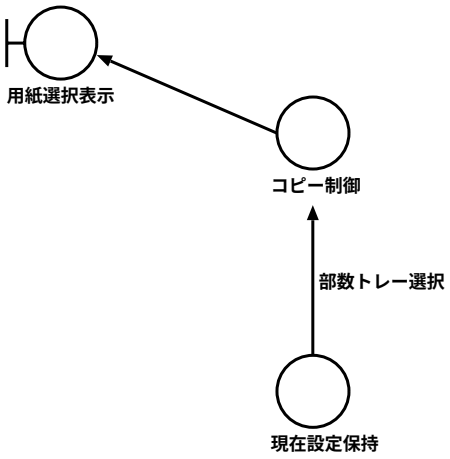
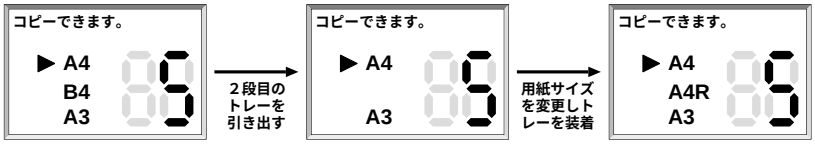


図 D.12: 用紙選択操作，給紙操作から得られた分析オブジェクト (2)

(a) この操作段落の検討から得られたオブジェクトを図 D.12 に示す.

3. 給紙用トレーの用紙サイズを変更し，再び装着すると，LCD 上にも新しいサイズを表示する．コピー動作中に給紙用トレーを引き出すと給紙トレー部に (場合によってはプリンタ部も) ジャムが発生する (コピー中に用紙なしのメッセージによって給紙を促されている場合を除く).



(a) この操作段落の検討から得られたオブジェクトを図 D.13 に示す.

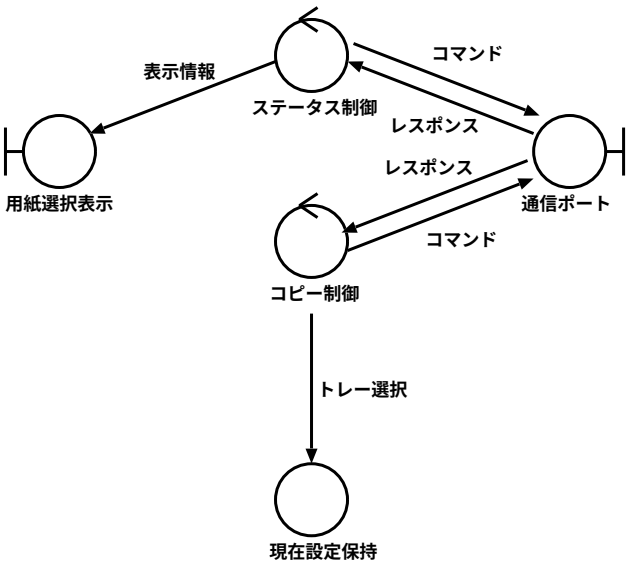


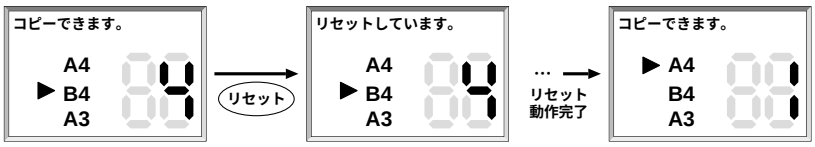
図 D.13: 用紙選択操作，給紙操作から得られた分析オブジェクト (3)

D.6 use-case リセット操作

それぞれの操作段階について識別できるオブジェクトについて議論する.

1. リセットキーは、コピー可能状態で待機中でなければ操作できない.(「コピーできます。」のメッセージを表示している.)

2. リセットキーを押す.



3. 制御部がリセット動作を開始したら,「リセットしています。」を表示する.

   (a) この操作段落の検討から得られたオブジェクトを図 D.14 に示す.

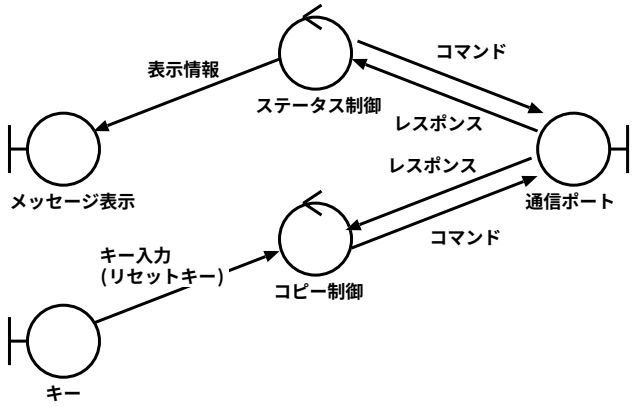


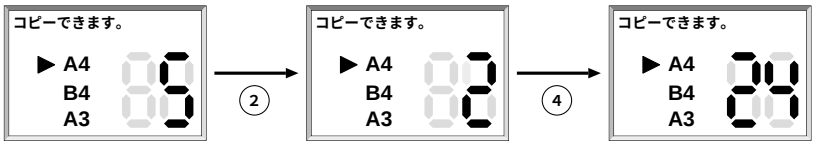
図 D.14: リセット操作から得られた分析オブジェクト (1)

4. リセット動作が完了したら，コピー可能状態で待機中になる.(「コピーできます。」のメッセージを表示している.)

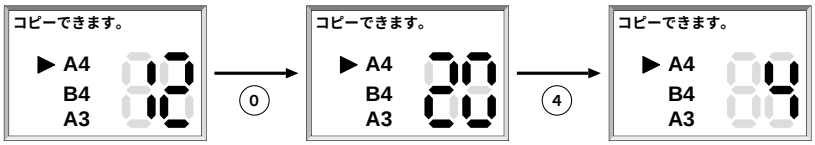
D.7　use-case テンキー操作

それぞれの操作段階について識別できるオブジェクトについて議論する.

1. コピー/リセット/クリアキー入力などの直後の最初の数字キー入力時には，現在の設定値を置き換えるように表示が変化する.

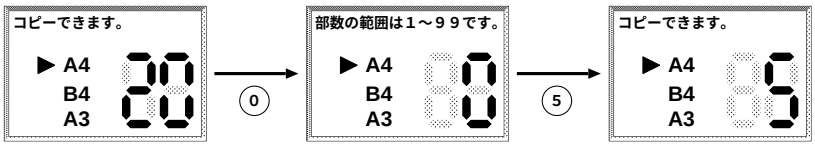


2. ひとたびテンキーの入力を始めたら，数字キーを 1 度押すたびに 1 桁ずつ右端に表示され，1 回前の入力値は 1 桁左へ送られて表示される．これで 2 桁の数値を表す.



3. 左へ送る桁が 0 の場合は，表示せずに今回の入力だけを表示する.

4. コピー可能な状態で設定値が 1 から 99 の範囲では，スタートキーを押すとコピーを開始できる．それ以外 (0 のとき) は「部数の範囲は 0〜99 です。」というメッセージを表示し，設定値が 1 から 99 の範囲になるまでスタートキーは押せない.



(a) この操作段落の検討から得られたオブジェクトを図 D.15 に示す.

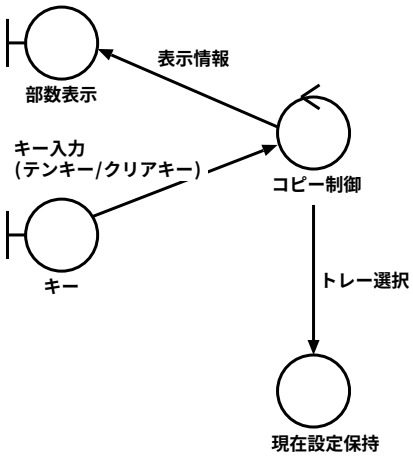


図 D.15: テンキー操作から得られた分析オブジェクト (1)

D.8 use-case ジャム (用紙詰まり) 発生時の操作

それぞれの操作段階について識別できるオブジェクトについて議論する.

1. ジャム発生箇所として以下の 3 箇所を報告する.
   (a) プリンタ部の排紙口部分
   (b) プリンタ部の用紙引き込みおよび転写部分
   (c) 給紙トレー部の用紙引き出し部分
2. ジャム発生時には「用紙が詰まりました。」のメッセージを表示し，発生箇所を示す図を表示する．ジャム中は、いずれのキーも操作できない.
   (a) この操作段落の検討から得られたオブジェクトを図 D.16 に示す.

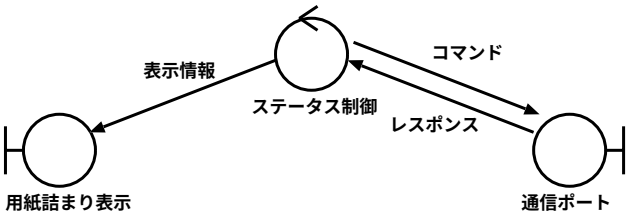
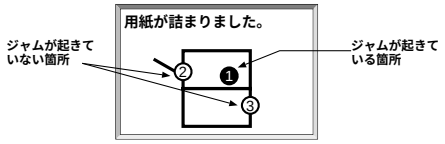


図 D.16: ジャム (用紙詰まり) 発生時の操作から得られた分析オブジェクト (1)

3. 詰まった用紙を取り除くには CP-KT1 の各部のドアを開けなければならない．安全対策のため，いずれかのドアを開けたときにいったん電源が切れる．全てのドアを閉じると自動的に再び電源が入る.



4. 電源投入時にジャム中かどうかをしらべ，全ての発生箇所の用紙詰まりが解消されるまで，ジャム箇所の番号付き印を更新しながら上記のメッセージを表示し続ける.
5. 全ての発生箇所の用紙詰まりが解消されると，コピー可能状態になる．リカバリ機能は提供しないので，リセット操作をした場合と同じ状態になる.

(a) この操作段落の検討から得られたオブジェクトを図 D.17 に示す.

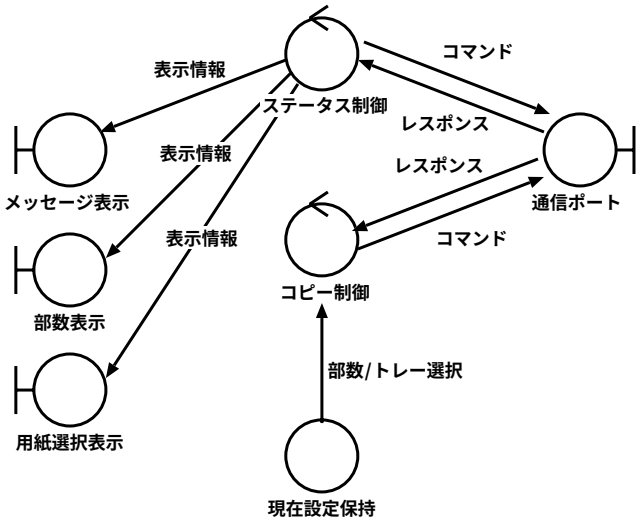
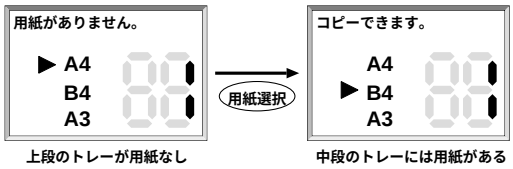


図 D.17: ジャム (用紙詰まり) 発生時の操作から得られた分析オブジェクト (2)

D.9 use-case コピー中でないときの用紙なしの場合の操作

それぞれの操作段階について識別できるオブジェクトについて議論する.

1. 用紙選択で用紙なしの状態にある給紙用トレーを選択すると「用紙がありません。」というメッセージを表示する．このときスタートキーは操作できない.



(a) この操作段落の検討から得られたオブジェクトを図 D.18 に示す.

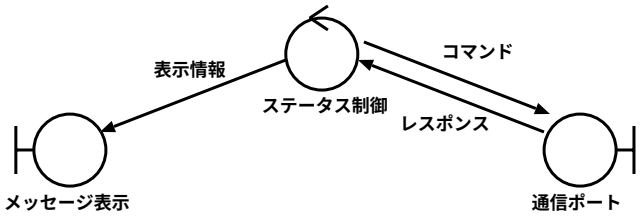
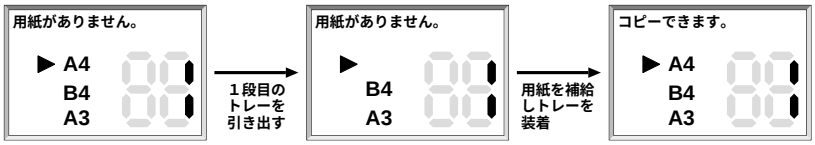


図 D.18: コピー中でないときの用紙なしの場合の操作から得られた分析オブジェクト (1)

2. 用紙の入っている給紙用トレーを選択すれば,「コピーできます。」のメッセージを表示し、スタートキーを操作してコピーできる.

(a) この操作段落の検討から得られたオブジェクトを図 D.19 に示す.

3. 紙なしになった給紙用トレーに用紙を補給し，再び装着すると,、「コピーできます。」のメッセージを表示し、スタートキーを操作してコピーできる.



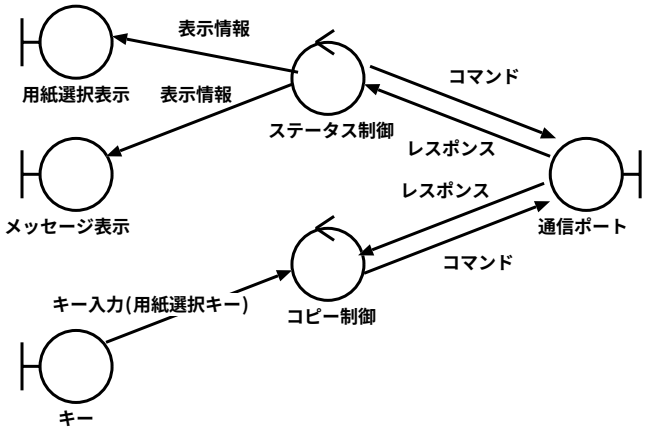
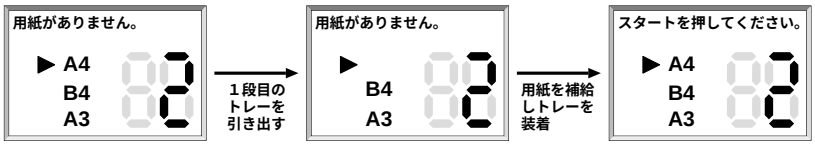


図 D.19: コピー中でないときの用紙なしの場合の操作から得られた分析オブジェクト (2)

D.10 use-case コピー中の用紙なしの場合の操作

それぞれの操作段階について識別できるオブジェクトについて議論する.

1. コピー中に使用中の給紙用トレーが用紙なしの状態になると「用紙がありません。」というメッセージを表示し，コピー動作が中断する.



(a) この操作段落の検討から得られたオブジェクトを図 D.20 に示す.

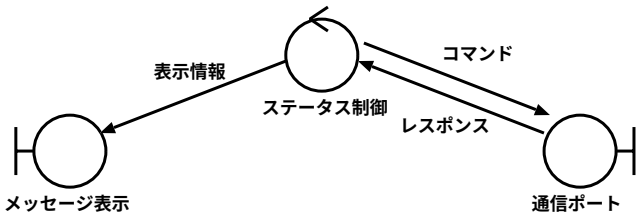


図 D.20: コピー中の用紙なしの場合の操作から得られた分析オブジェクト (1)

2. 部数の設定は用紙なしになった時点の値のままとする．数字キー，クリアキーなどは操作できない.

3. 用紙選択の設定は用紙なしになった時点の値のままとする．用紙選択キーは操作できない.

4. 当該給紙用トレーに用紙を補給し，装着すると「スタートを押してください。」というメッセージを表示する.

(a) この操作段落の検討から得られたオブジェクトを図 D.21 に示す.

5. ここでスタートキーを操作すると，用紙なしの時点からコピーを継続できる.

(a) この操作段落の検討から得られたオブジェクトを図 D.22 に示す.

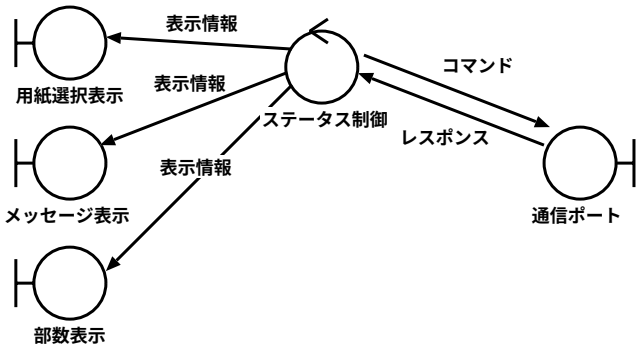


図 D.21: コピー中の用紙なしの場合の操作から得られた分析オブジェクト (2)

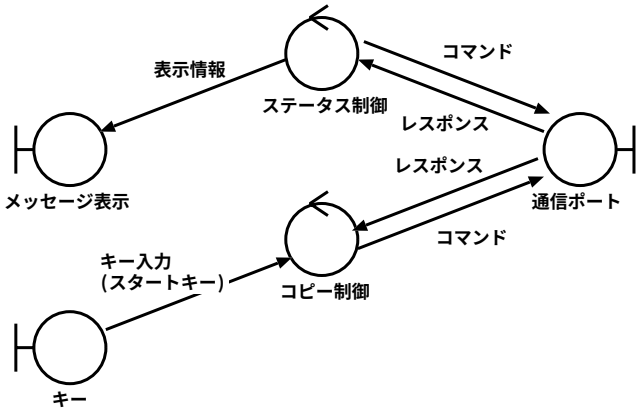


図 D.22: コピー中の用紙なしの場合の操作から得られた分析オブジェクト (3)

6. ストップキーを操作すると，コピー中の中断動作を実行する．ストップ動作完了後は，部数，用紙選択がコピー開始時点の値に戻り，コピー中でないときの用紙なしの場合と同様に操作できるようになる．

   (a) この操作段落の検討から得られたオブジェクトを図 D.23 に示す.

   (a) この操作段落の検討から得られたオブジェクトを図 D.24 に示す.

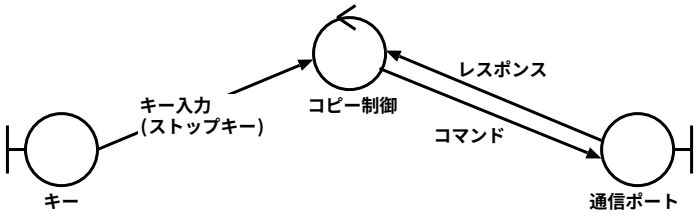


図 D.23: コピー中の用紙なしの場合の操作から得られた分析オブジェクト (4)

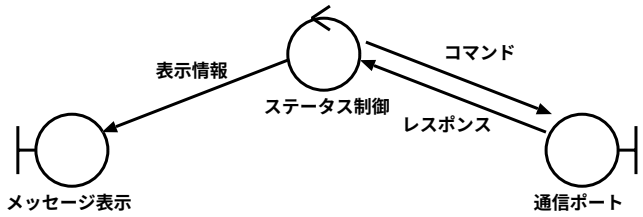


図 D.24: コピー中の用紙なしの場合の操作から得られた分析オブジェクト (5)

D.11　分析モデルの作成

これまでの分析作業で，use-case ごとの分析オブジェクトを得た．これら全ての分析オブジェクトをまとめれば，分析モデルが得られる．また，このときに要求モデル作成の結果得た問題ドメインオブジェクトモデルを参照し，要求分析の結果と大きく異なる点がないかどうか確認する．あれば，それらについて再び議論する必要がある．

図 D.25 が，そのようにしてまとめた OOSE 法に準ずる記法によってる分析モデル図である．

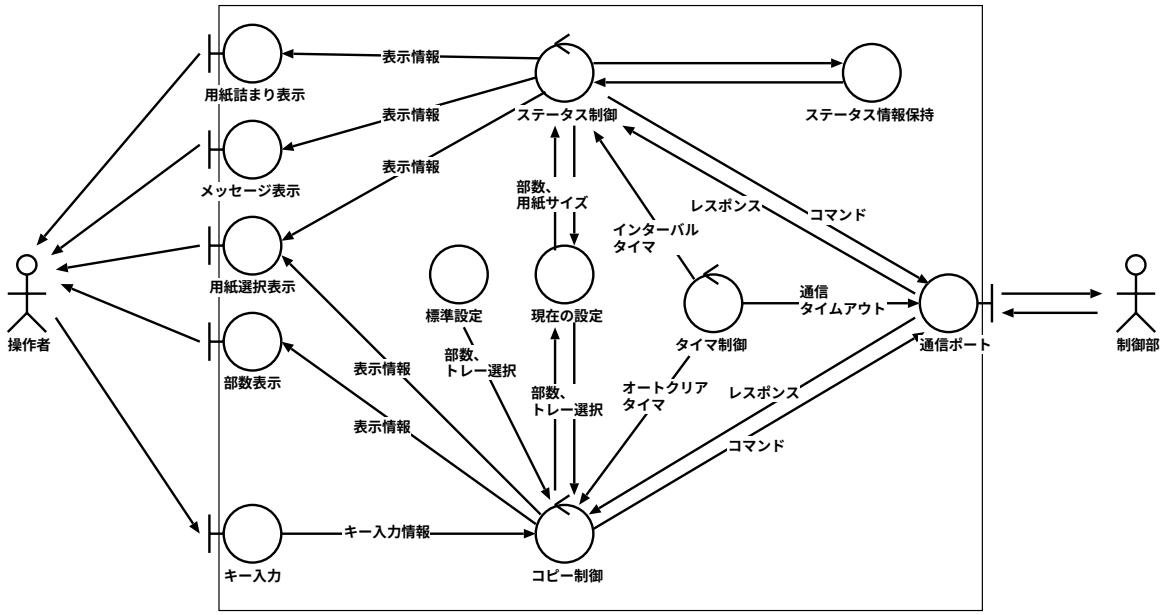


図 D.25: OOSE 法に準ずる記法による分析モデル図 (まとめ)

D.12　インタラクション図の作成

D.11節で得られたオブジェクトについて，それぞれの間でやりとりされるイベントの動きを use-case に沿って整理すれば，個々のオブジェクトがどのようなイベントを受け取ったときにどのように動作するかを詳細に理解できるようになる．整理方法としては，オブジェクトメッセージ図/ インタラクション図 (メッセージトレース図) などの手法があるが，ここではインタラクション図にまとめることにする．

図 D.26 から図 D.37 に本事例の分析結果のインタラクション図を示す．

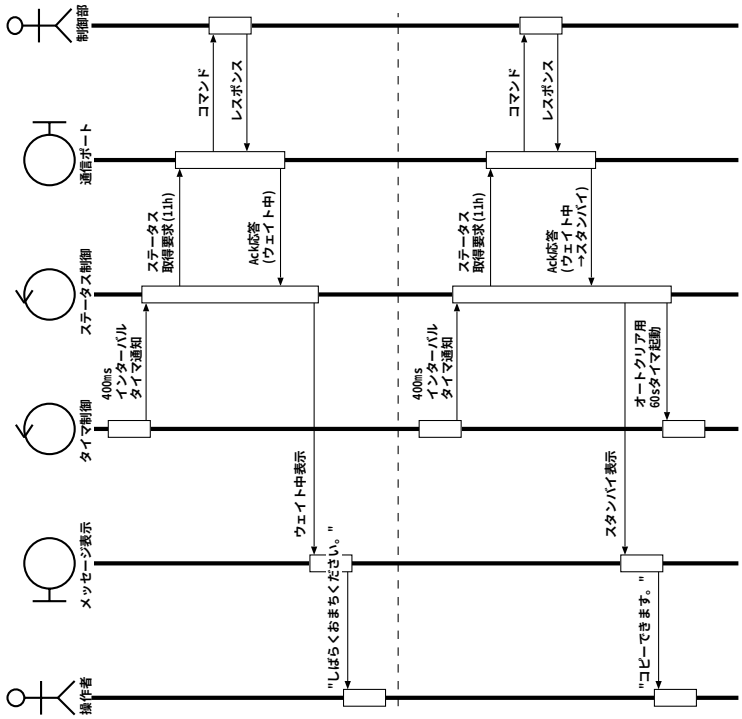


図 D.26: ウェイト中からスタンバイまでのインタラクション図

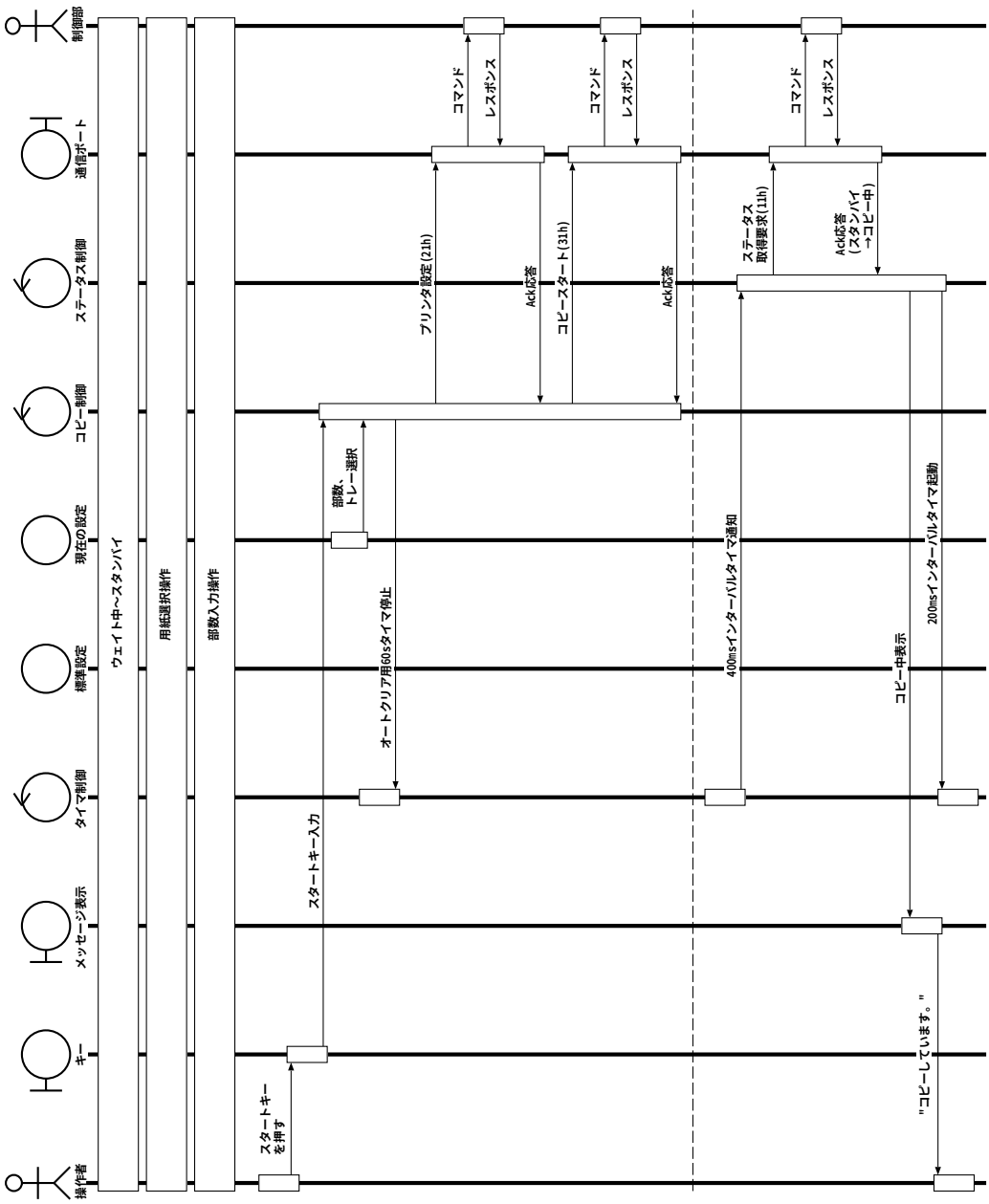


図 D.27: スタンバイからコピー中までのインタラクション図

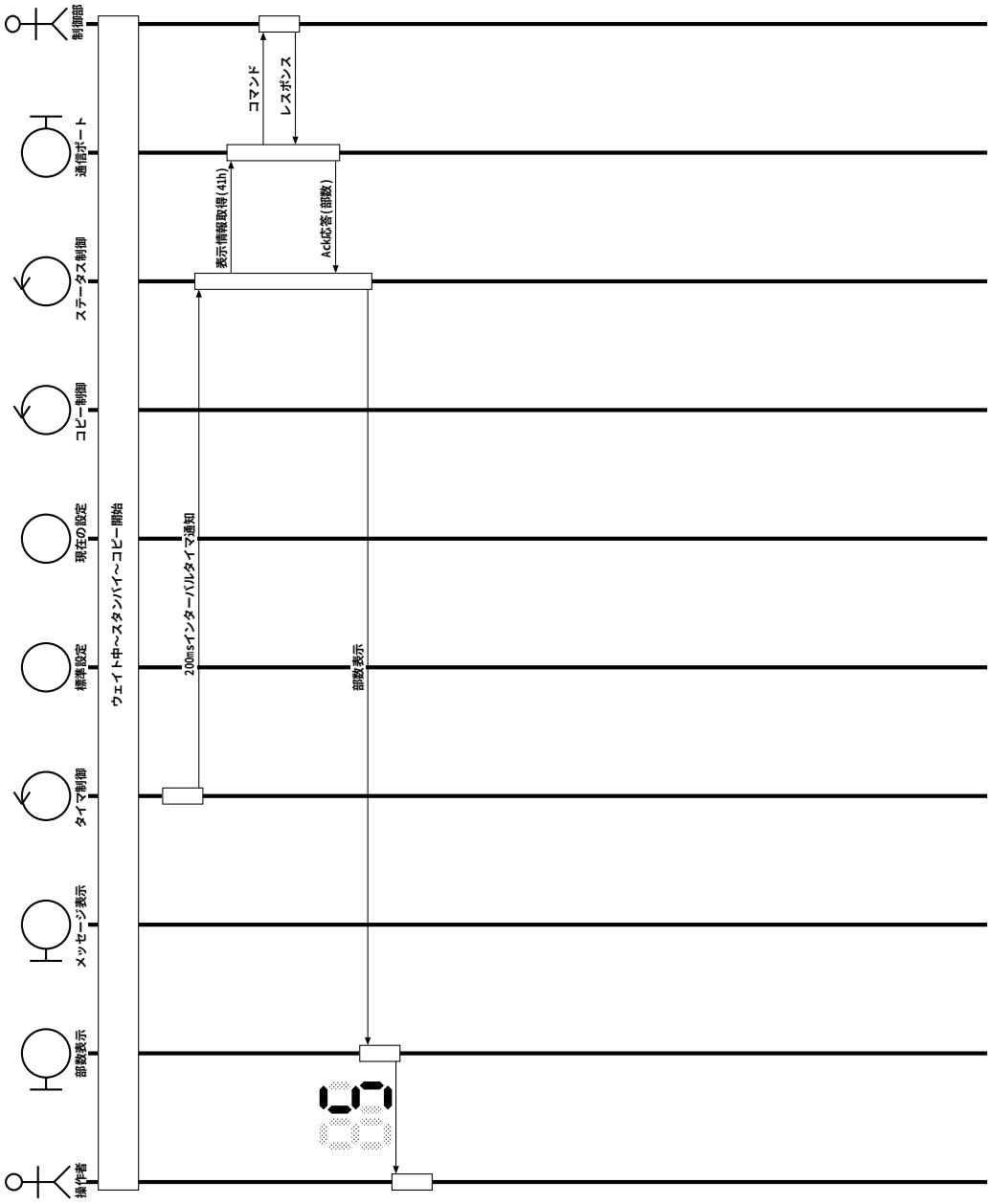


図 D.28: コピー中のインタラクション図

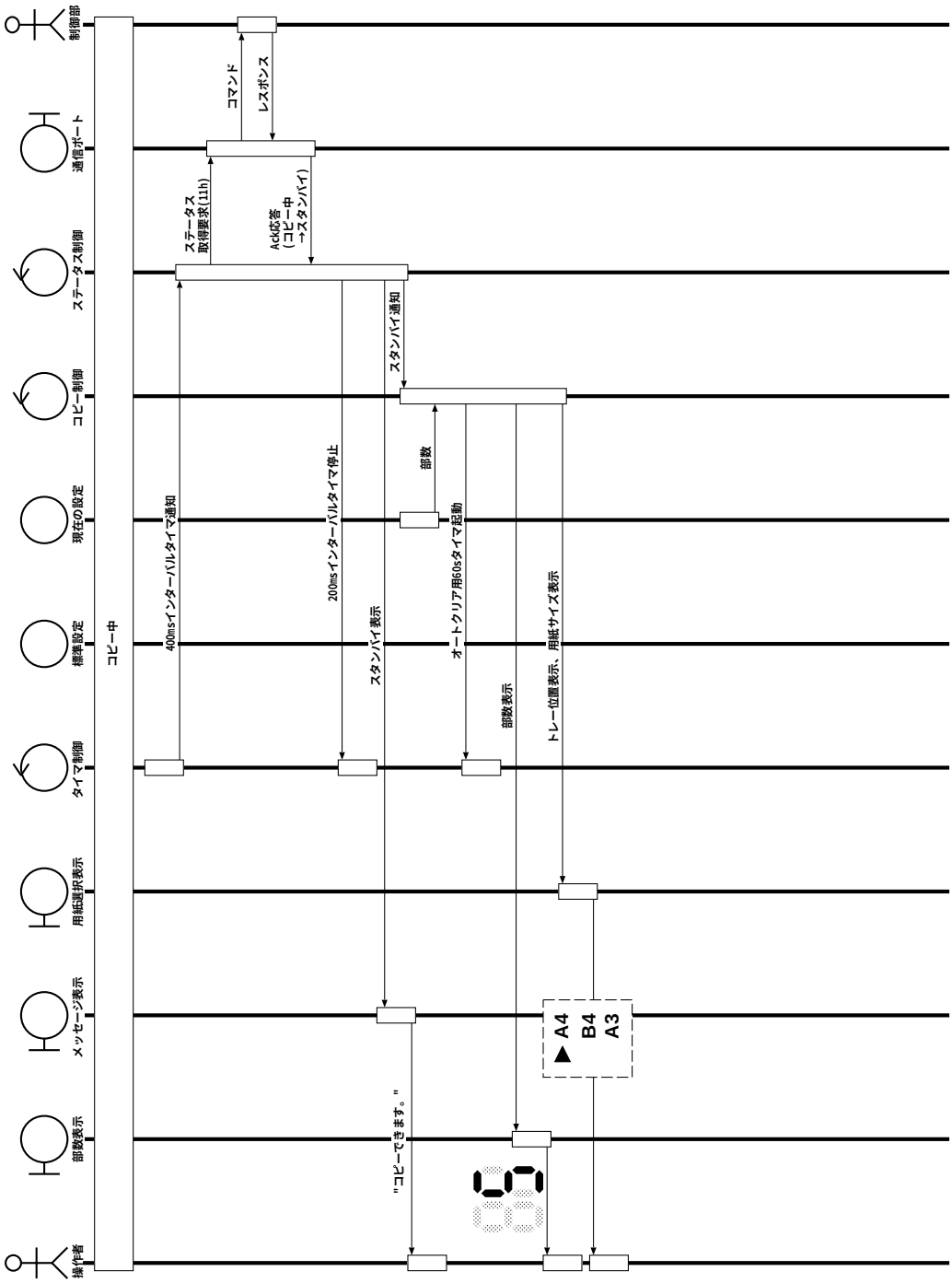


図 D.29: コピー終了後スタンバイまでのインタラクション図

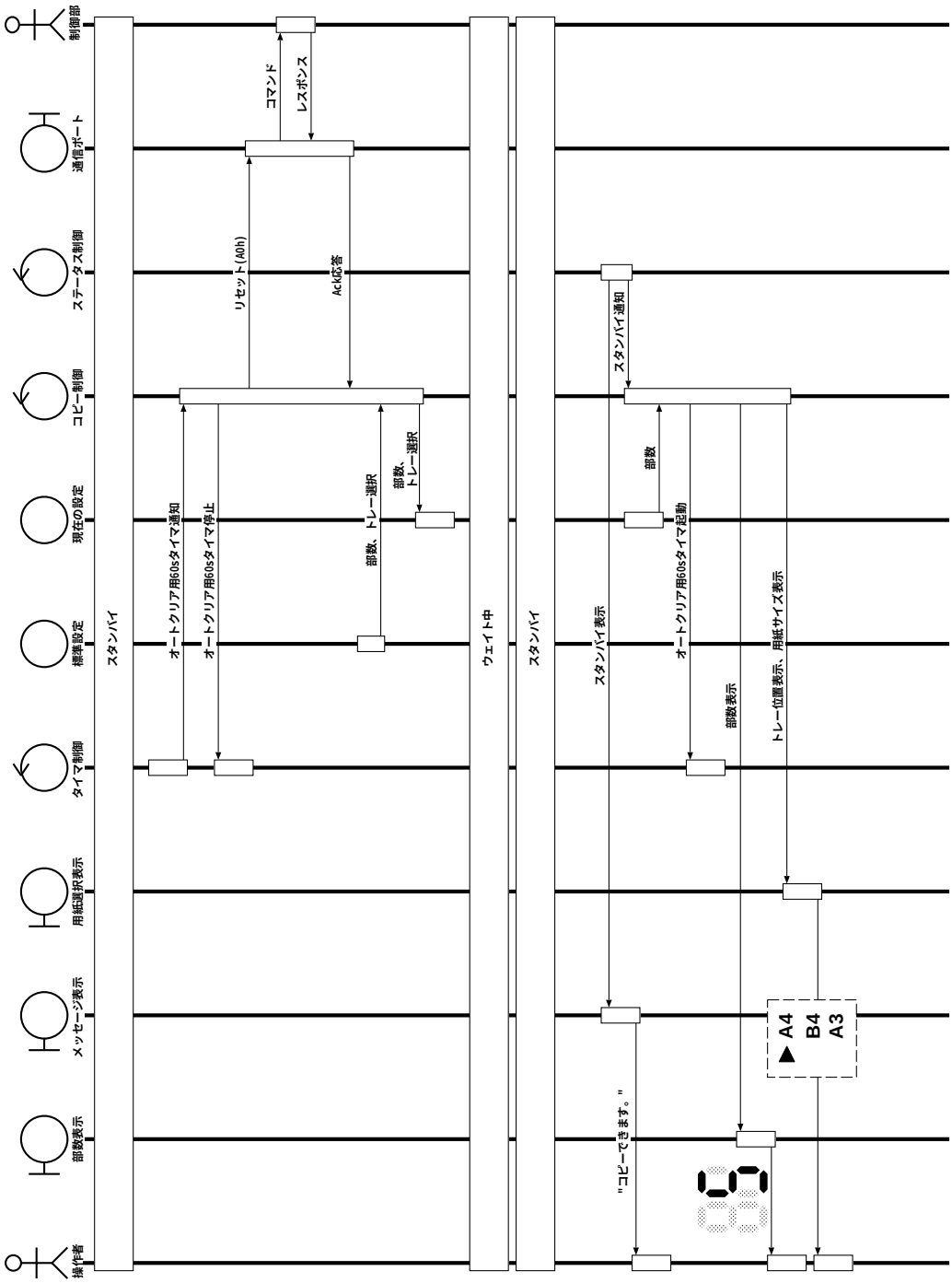


図 D.30: スタンバイからオートクリア動作までのインタラクション図

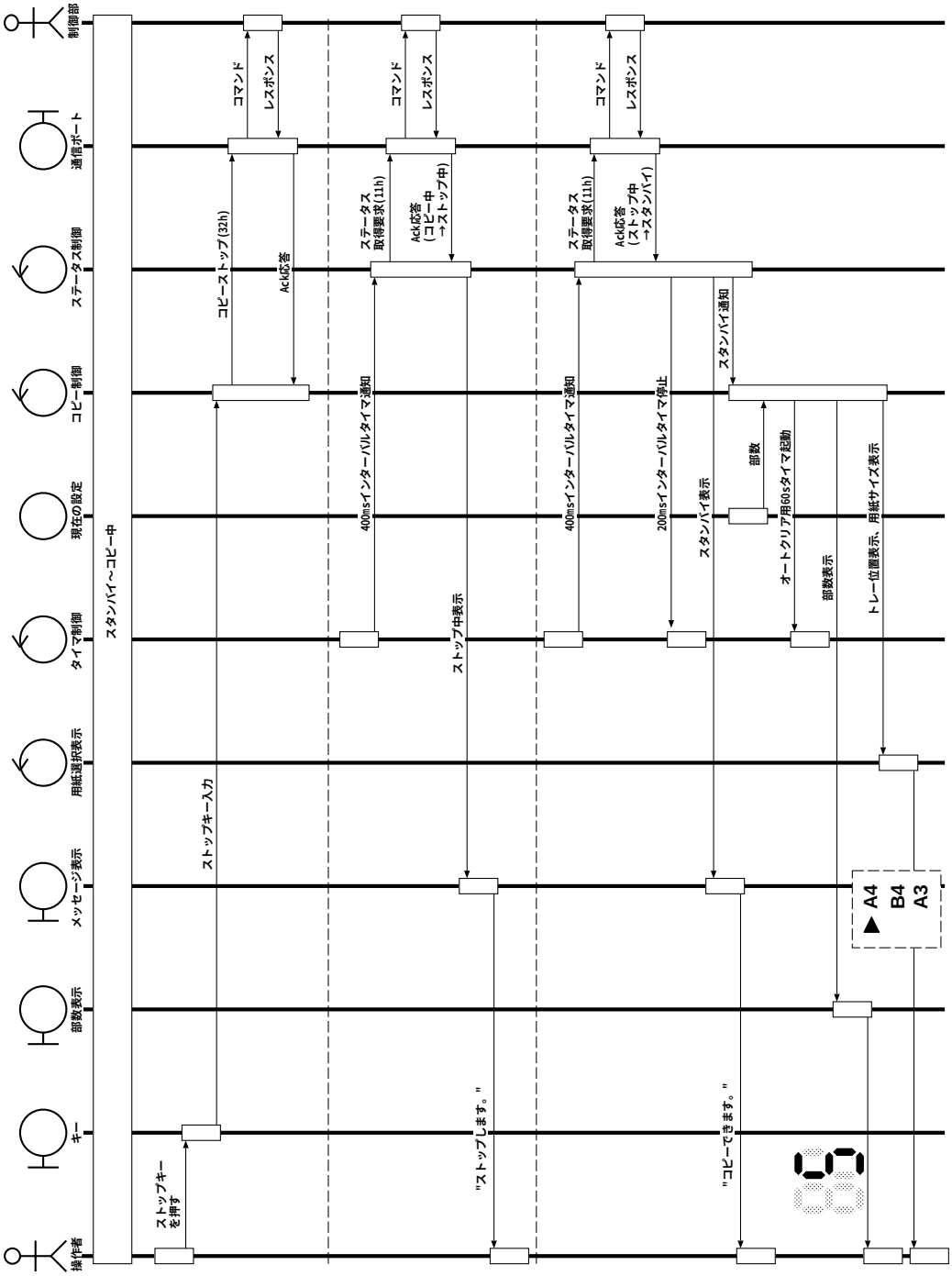


図 D.31: コピー中からコピー中断動作までのインタラクション図

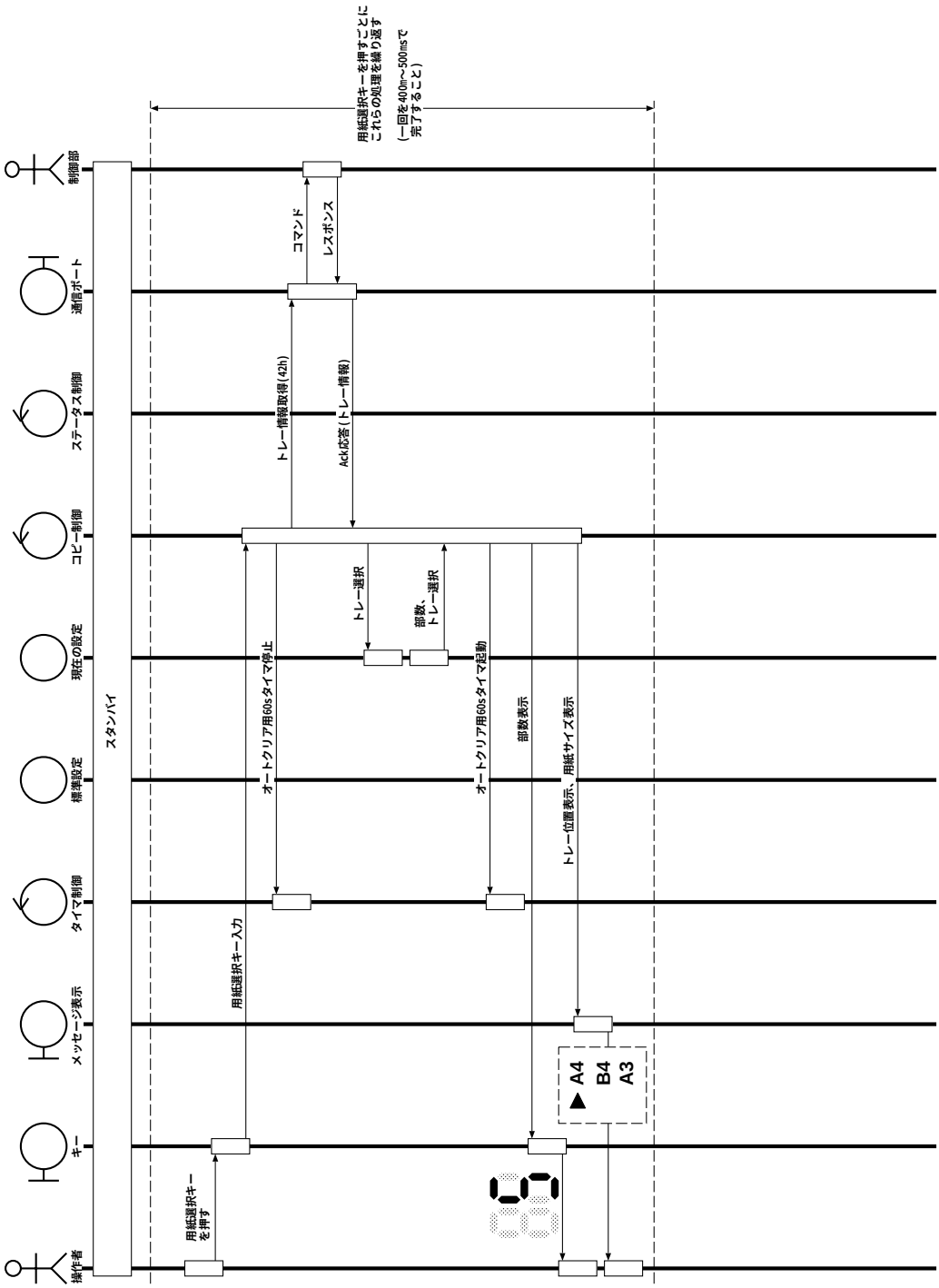


図 D.32: 用紙選択操作のインタラクション図

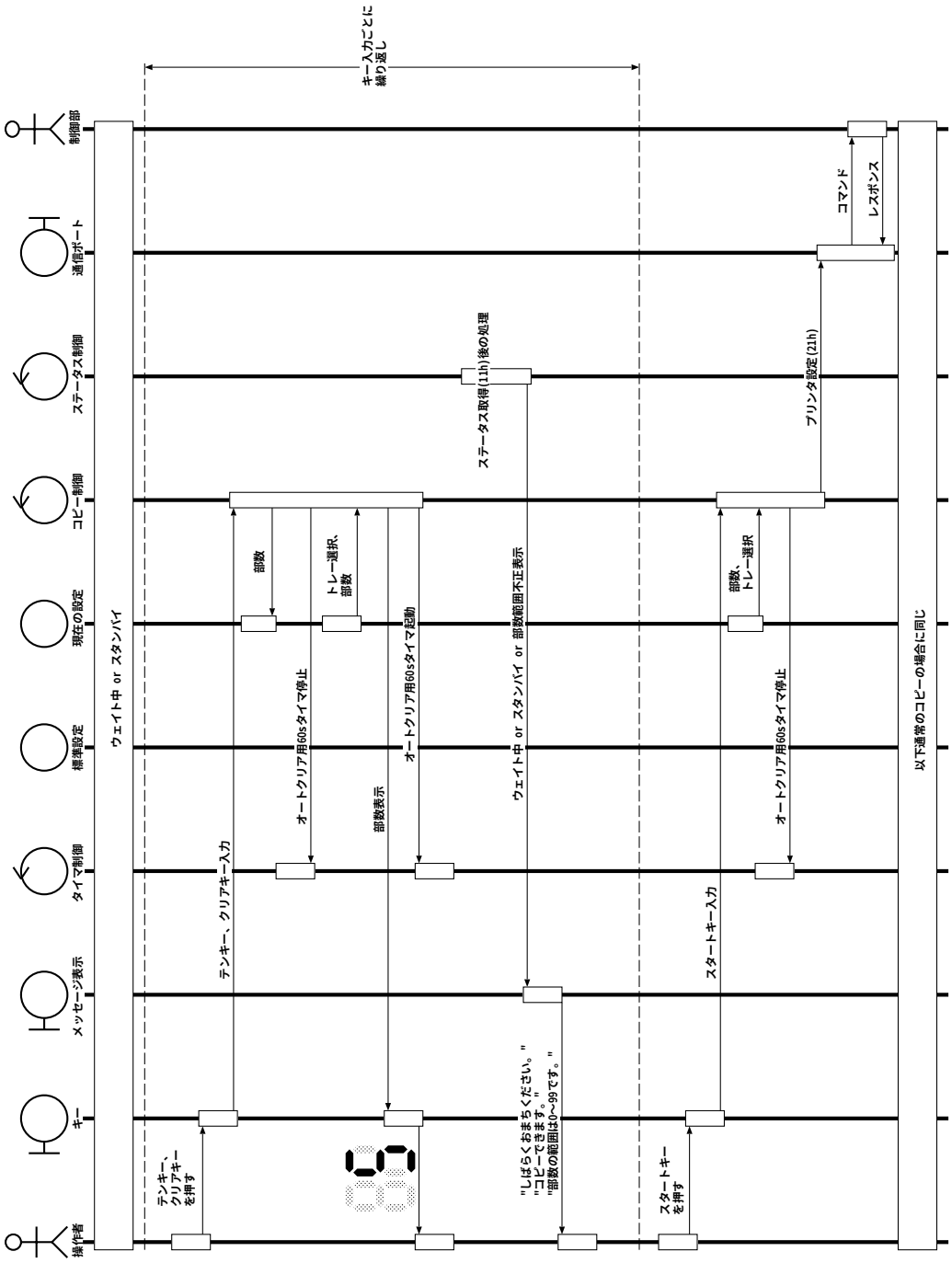


図 D.33: 部数設定操作のインタラクション図

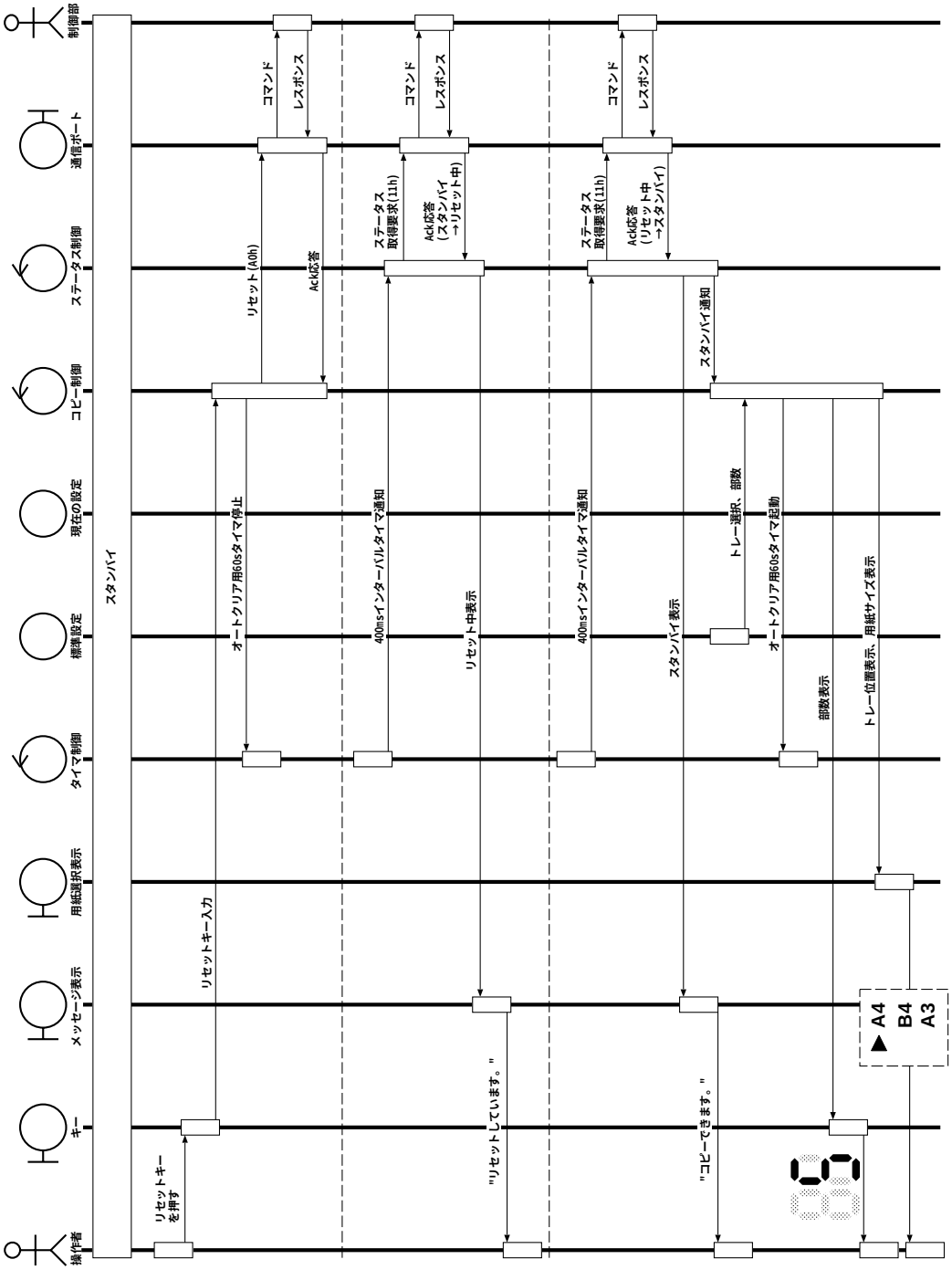


図 D.34: リセット操作のインタラクション図

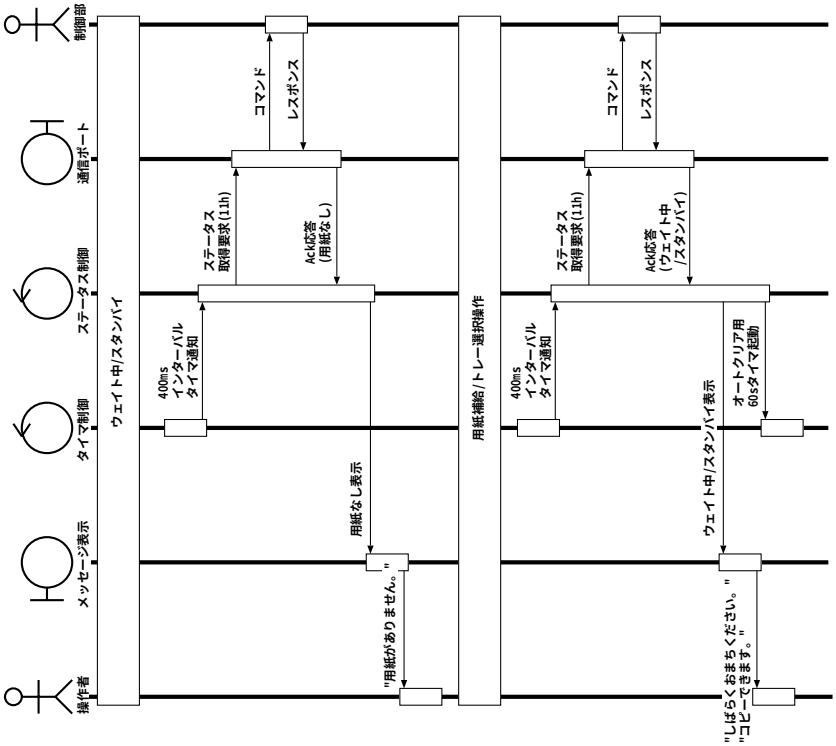


図 D.35: コピー中でない用紙なしが発生したときのインタラクション図

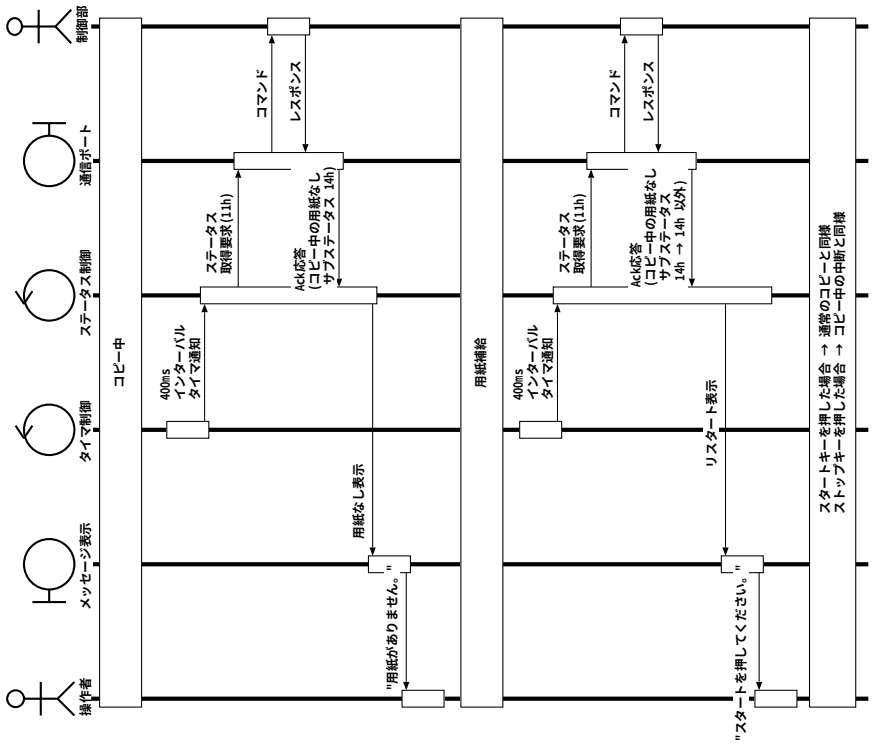


図 D.36: コピー中に用紙なしが発生したときのインタラクション図

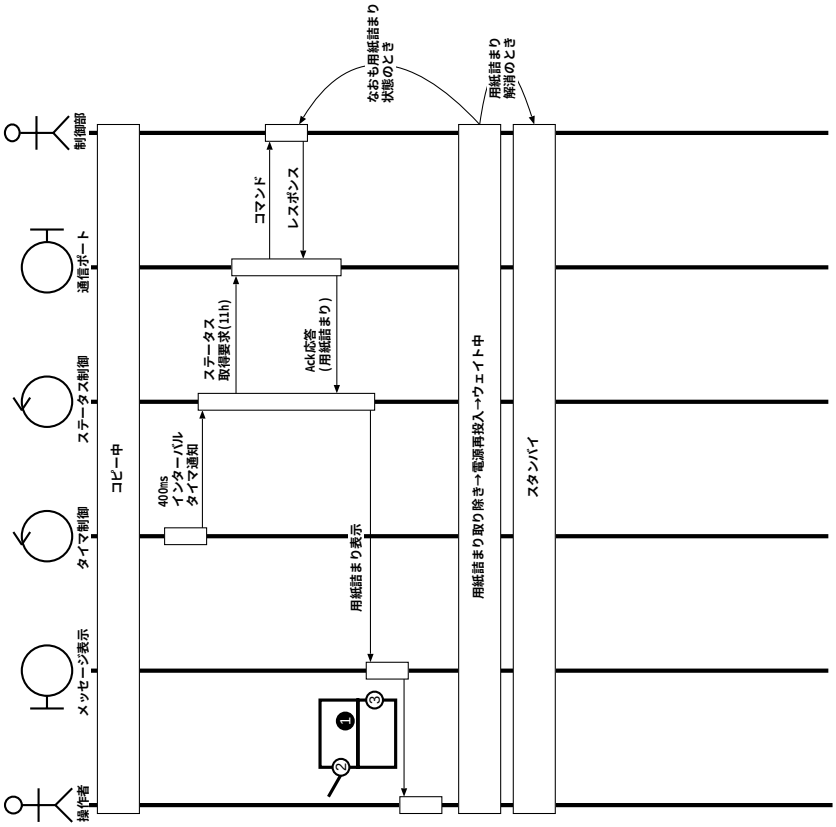


図 D.37: コピー中に用紙詰まりが発生したときのインタラクション図

D.13　ObTS モデル図の作成

これまでのところ，要求文書/要求モデル作成/分析モデル作成の各過程から以下の成果物を得た．

- use-case 文書 (本事例では要求文書の一部を転用している)
- 操作部の基本操作方法仕様
- 操作部 - 制御部間インターフェース仕様
- インターフェース記述
- 問題ドメインオブジェクトモデル
- 分析モデル図
- インタラクション図

以上で ObCL コードを書く準備が整った．

オブジェクトの階層構造を把握するために，オブジェクトモデル図を構造に着目して整理しておくとよい．このために，図 D.38 を作成した．

また，インタラクション図で得たオブジェクトの動的な振舞いは，動的モデル図として ObTS モデル図に整理しておくと動作理解の助けとなる．

図 D.39 から図 D.51 に動的モデル図として ObTS モデル図を示す．だだしここに示した図は，後述の ObCL コードとの対応がとりやすいように ObTS のグラフィカルな記法の原則を若干変え，ObCL のコードを記述している部分がある．

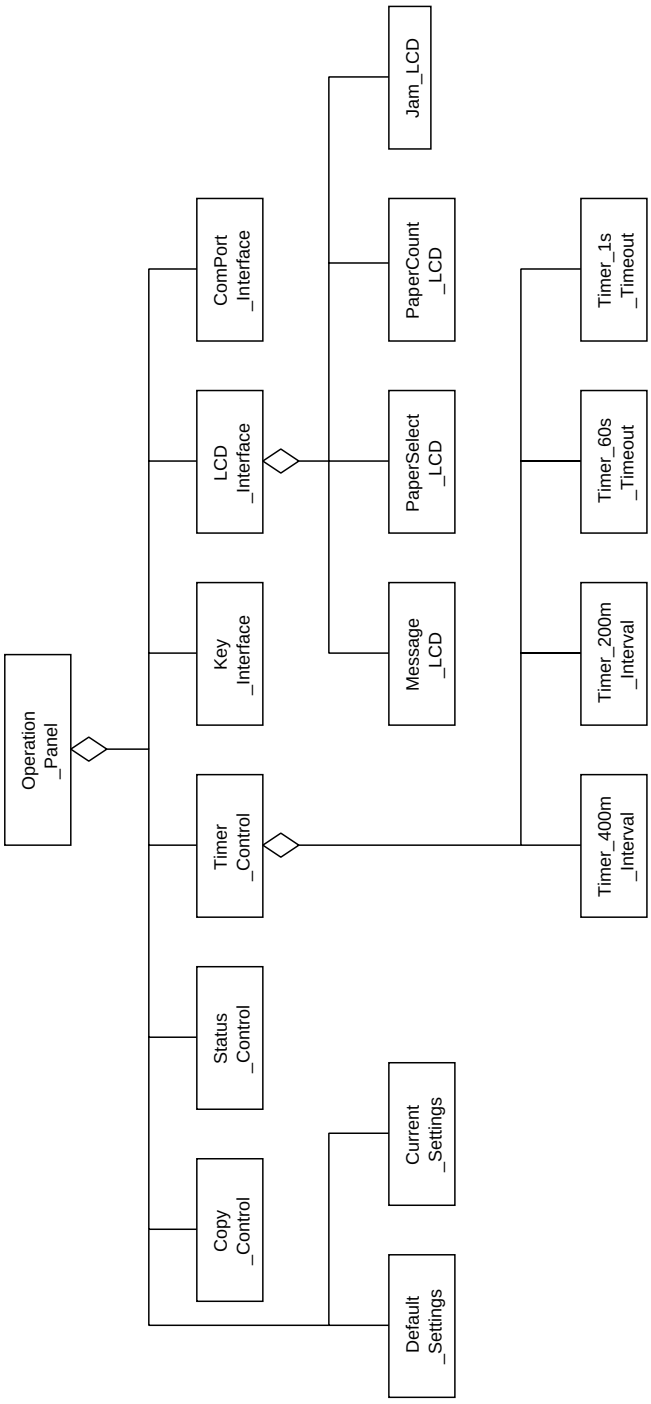


図 D.38: OMT 法に準ずる記法によるオブジェクトモデル図

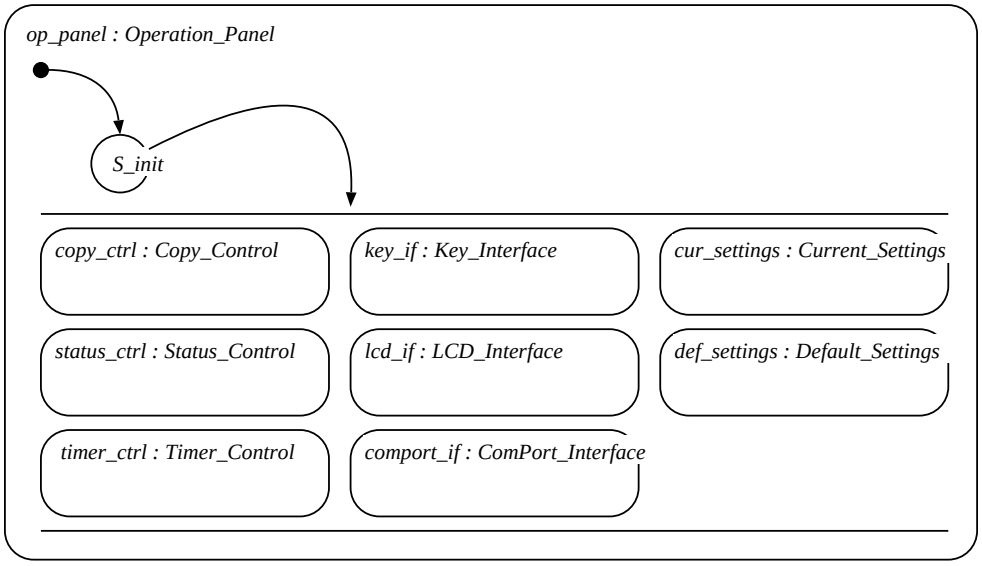


図 D.39: ObTS モデル図 (op_panel:Operation_Panel)

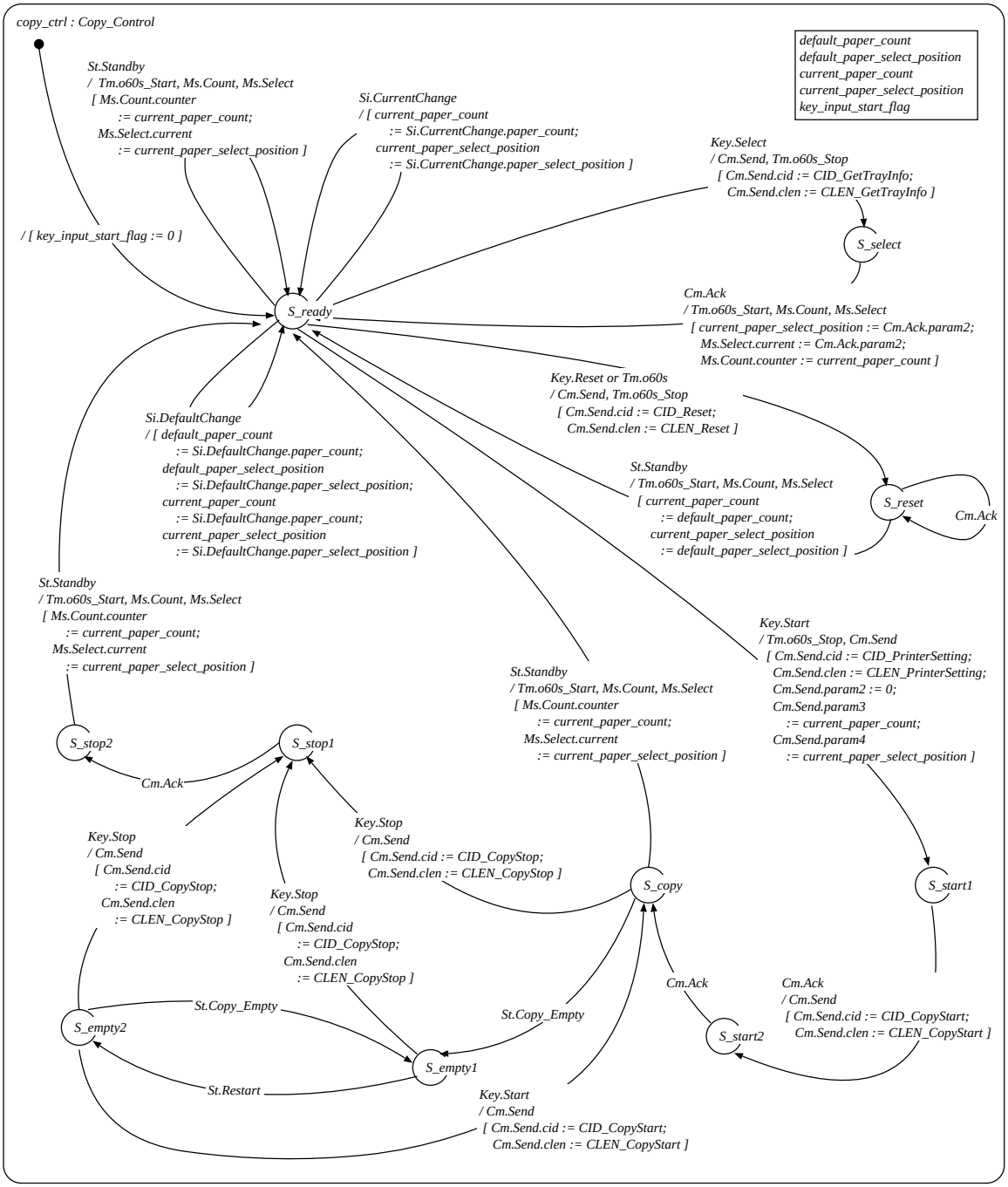


図 D.40: ObTS モデル図 (copy_ctrl:Copy_Control)

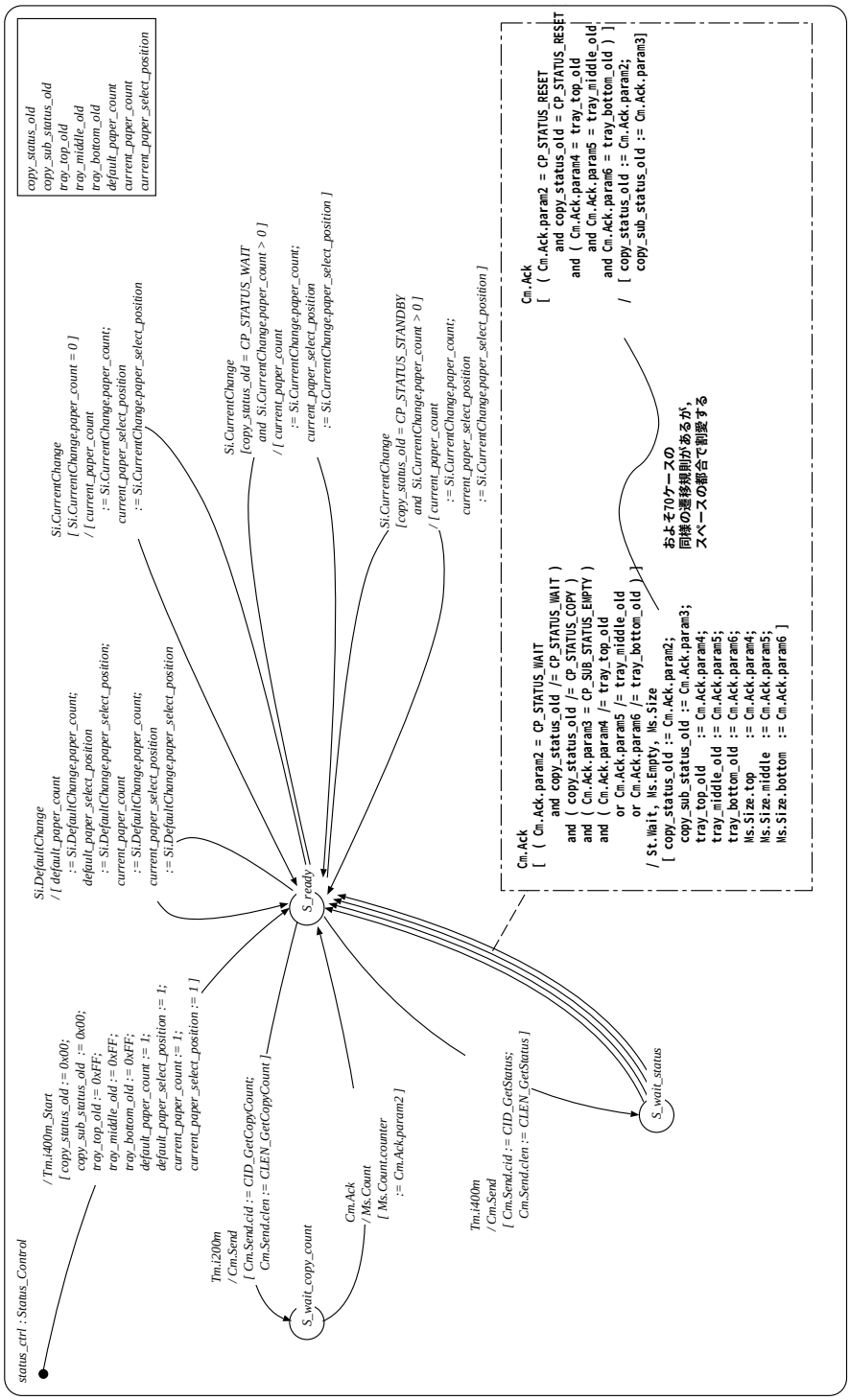


図 D.41: ObTS モデル図 (status_ctrl:Status_Control)

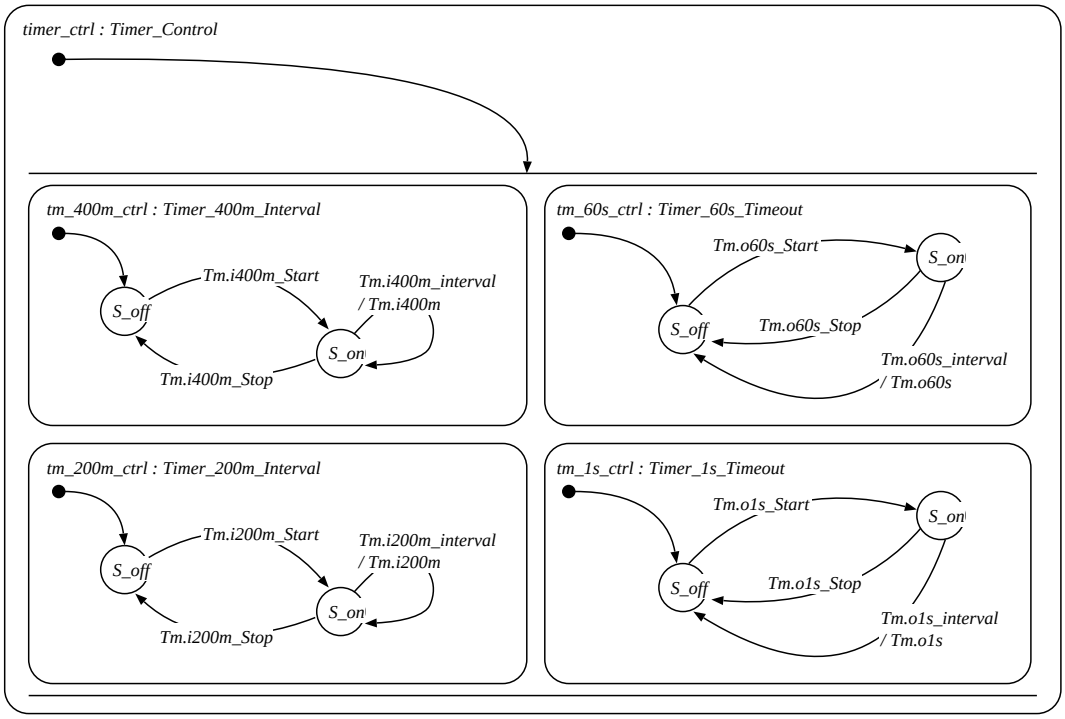


図 D.42: ObTS モデル図 (timer_ctrl:Timer_Control)

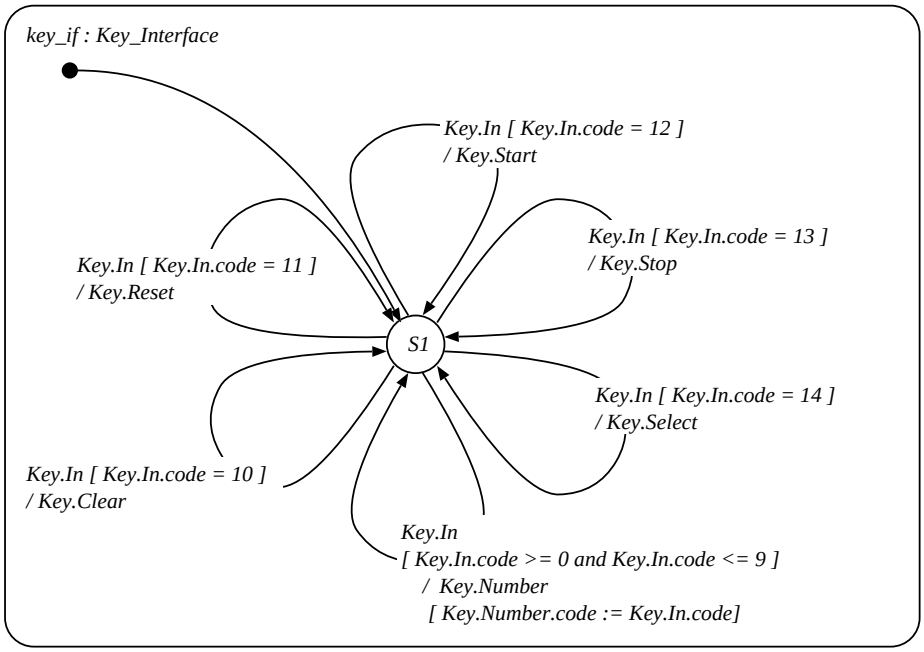


図 D.43: ObTS モデル図 (key_if:Key_Interface)

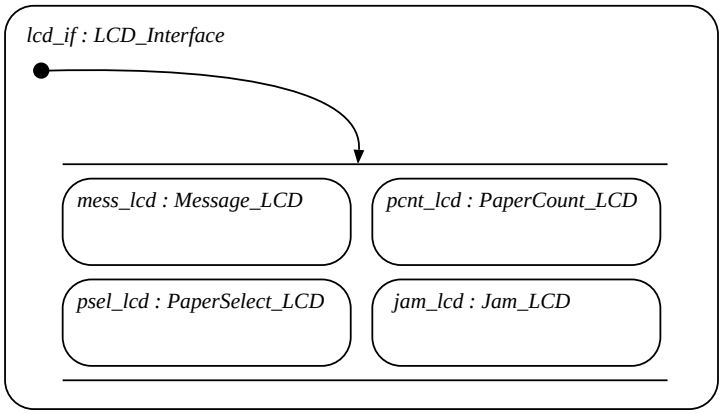


図 D.44: ObTS モデル図 (lcd_if:LCD_Interface)

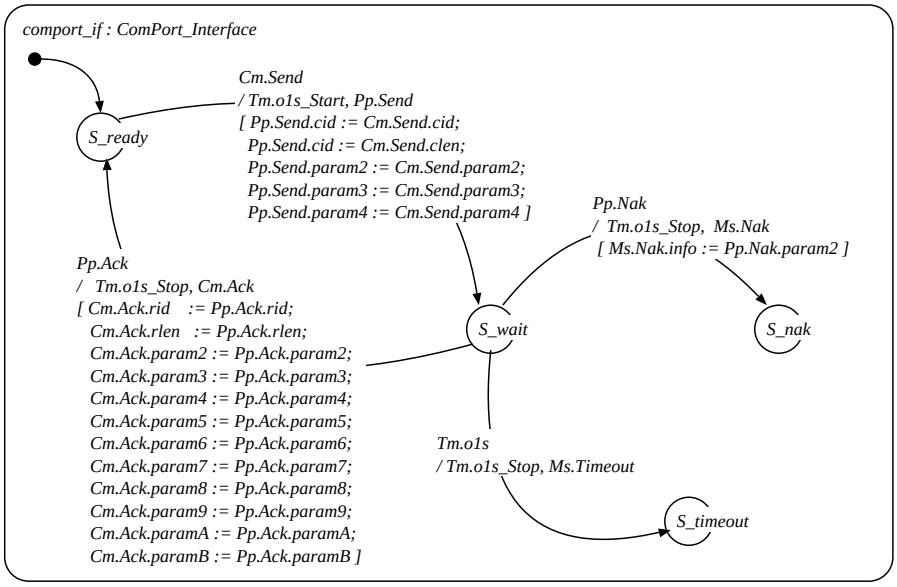


図 D.45: ObTS モデル図 (comport_if:ComPort_Interface)

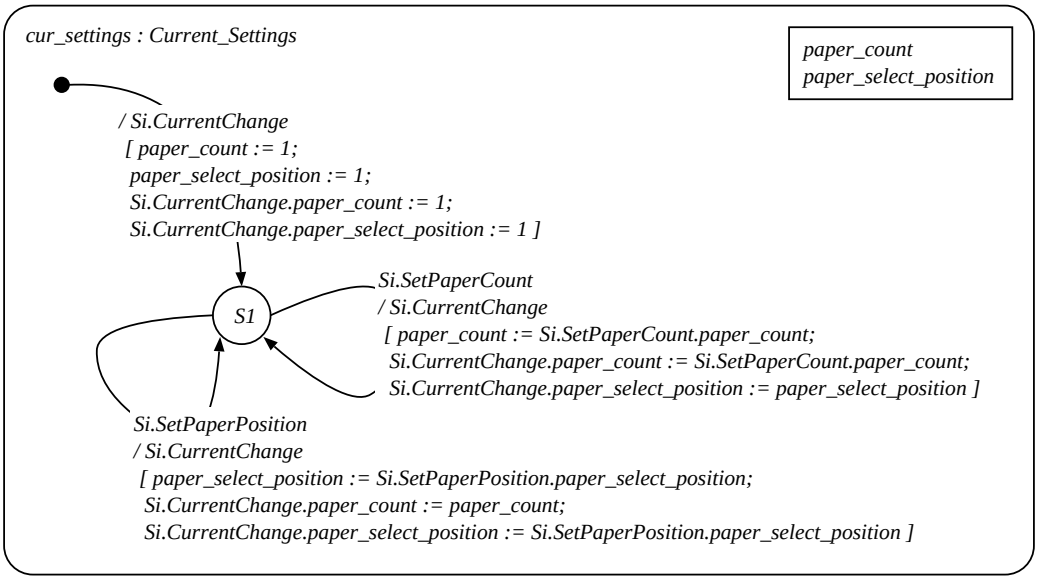


図 D.46: ObTS モデル図 (cur_settings:Current_Settings)

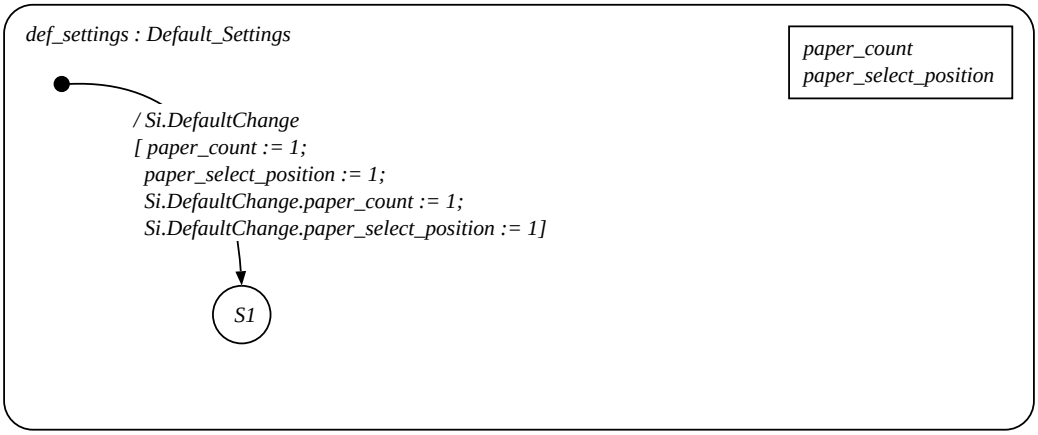


図 D.47: ObTS モデル図 (def_settings:Default_Settings)

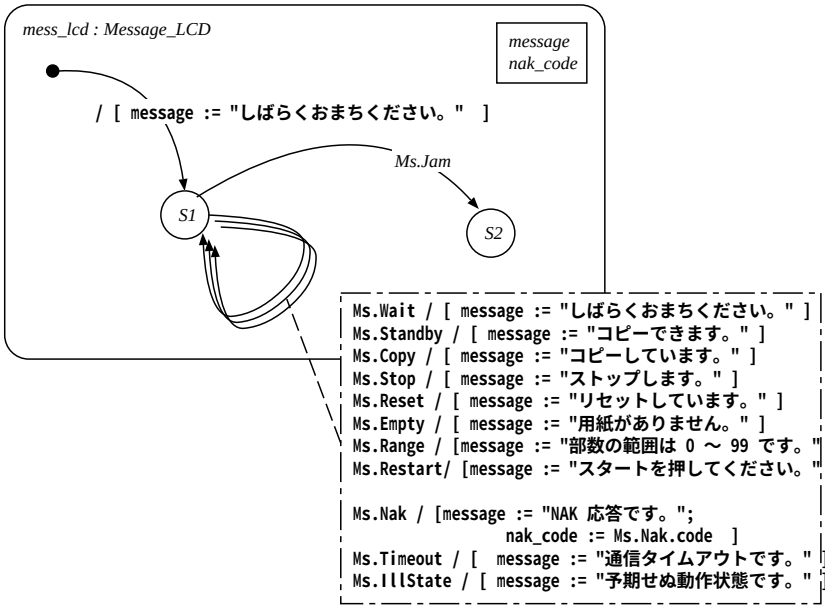


図 D.48: ObTS モデル図 (mess_lcd:Message_LCD)

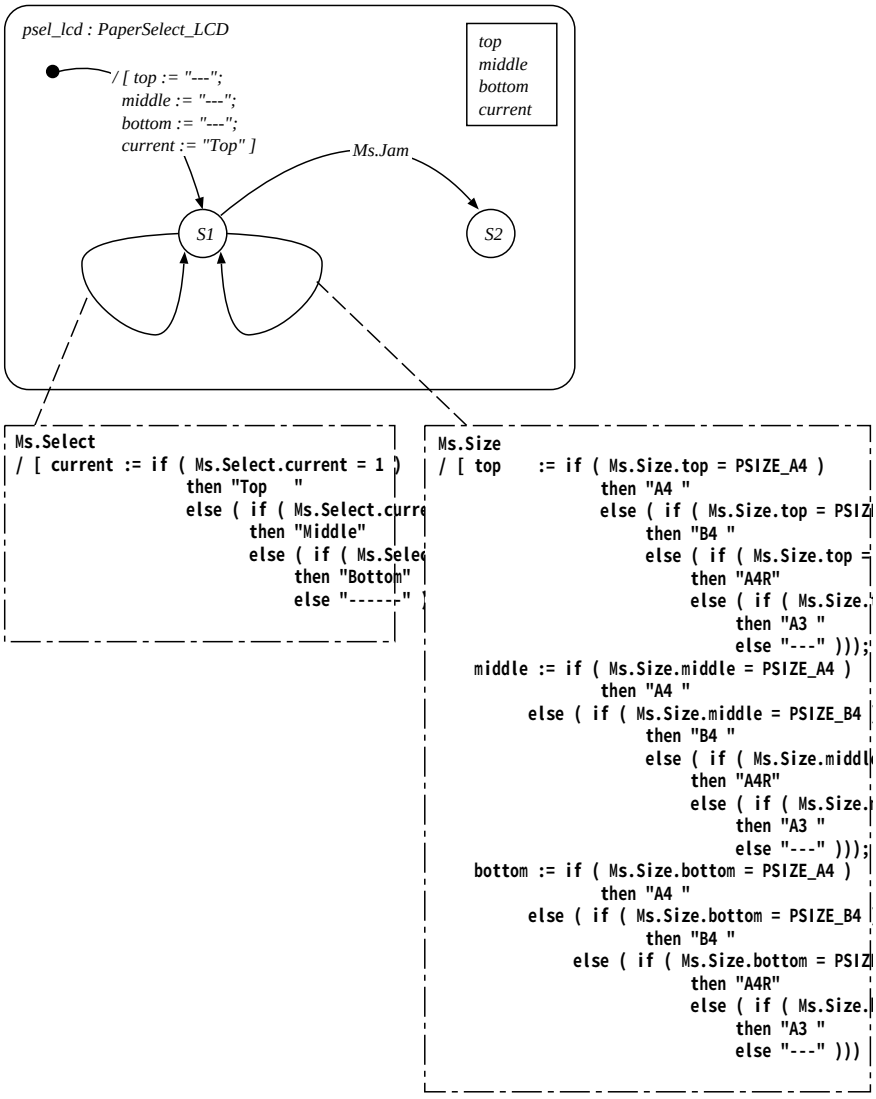


図 D.49: ObTS モデル図 (psel_lcd:PaperSelect_LCD)

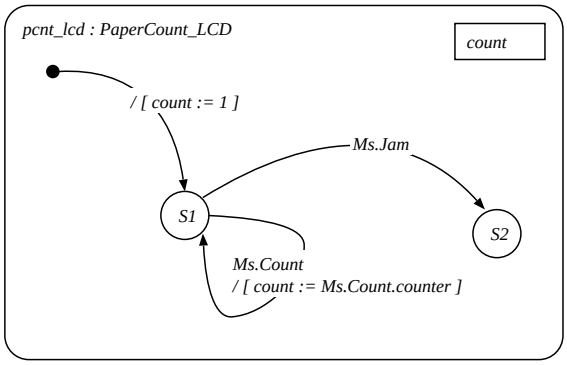


図 D.50: ObTS モデル図 (pcnt_lcd:PaperCount_LCD)

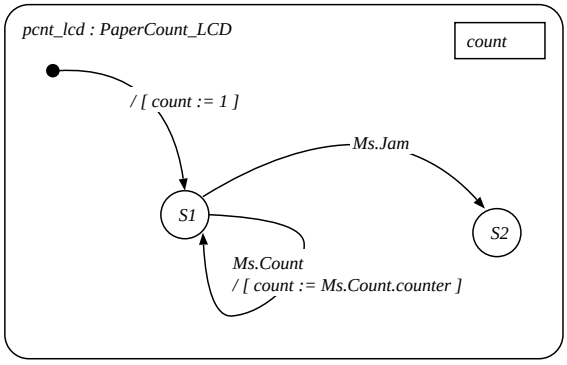


図 D.51: ObTS モデル図 (jam_lcd:Jam_LCD)

第 E 章

複写機 CP-KT1 の操作部プログラムの ObCL コード

本章では, 付録 D に示した複写機 CP-KT1 の操作部の分析モデルを基に作成した ObCL コードを示す.

```
-- -*- Mode : obcl -*- ------------------------------
--indexing
--    description : "Copy Machine CP-KT1 操作部の仕様.";
--    author      : "$Author: kuboaki $";
--    date        : "$Date: 1998/02/18 10:18:46 $";
--    filename    : "$RCSfile: panel.obcl,v $";
--    revision    : "$Revision: 1.49 $";
--    state       : "$State: Exp $"
-------------------------------------------------------
-- Event Definition ---------------------------------
-------------------------------------------------------
event
    Paper_Size_Display_Event
  inherit
    generic_event
  attribute
    top, middle, bottom : Int
end --  Paper_Size_Display_Event
-------------------------------------------------------
event
    Paper_Select_Display_Event
```

```
    inherit
        generic_event
    attribute
        current : Int
end -- Paper_Select_Display_Event
-------------------------------------------------------
event
    Paper_Counter_Display_Event
  inherit
        generic_event
    attribute
        counter : Int
end -- Paper_Counter_Display_Event
-------------------------------------------------------
event
    Nak_Info_Display_Event
inherit
    generic_event
attribute
    info : Int
end -- Nak_Info_Display_Event
-------------------------------------------------------
event
    Jam_Display_Event
  inherit
        generic_event
    attribute
        pos1, pos2, pos3 : Int
end -- Jam_Display_Event
-------------------------------------------------------
event
    Scan_Code_Event
  inherit
        generic_event
    attribute
        code : Int
end -- Scan_Code_Event
-------------------------------------------------------
event
    Number_Key_Event
```

```
  inherit
      generic_event
  attribute
      code : Int
end -- Number_Key_Event
-------------------------------------------------------
-- Command ID constants
-- for Command_Send_Event, Physical_Command_Send_Event
#define CID_GetStatus          0x11
#define CID_PrinterSetting     0x21
#define CID_CopyStart          0x31
#define CID_CopyStop           0x32
#define CID_GetCopyCount       0x41
#define CID_GetTrayInfo        0x42
#define CID_Reset              0xA0
-- Command Length constants
-- for Command_Send_Event, Physical_Command_Send_Event
#define CLEN_GetStatus         0x00
#define CLEN_GetErrorInfo      0x00
#define CLEN_PrinterSetting    0x02
#define CLEN_CopyStart         0x00
#define CLEN_CopyStop          0x00
#define CLEN_GetCopyCount      0x00
#define CLEN_GetTrayInfo       0x00
#define CLEN_Reset             0x00
-------------------------------------------------------
event
    Command_Send_Event
  inherit
      generic_event
  attribute
      cid    : Int;
      clen   : Int;
      param2 : Int;
      param3 : Int;
      param4 : Int
end -- Command_Send_Event
-------------------------------------------------------
event
    Ack_Receive_Event
```

```
    inherit
        generic_event
    attribute
        rid      : Int;
        rlen     : Int;
        param2   : Int;
        param3   : Int;
        param4   : Int;
        param5   : Int;
        param6   : Int;
        param7   : Int;
        param8   : Int
end -- Ack_Receive_Event
-------------------------------------------------------
event
    Physical_Command_Send_Event
    inherit
        generic_event
    attribute
        cid  : Int;
        clen : Int;
        param2 : Int;
        param3 : Int;
        param4 : Int
end -- Physical_Command_Send_Event
-------------------------------------------------------
event
    Physical_Ack_Receive_Event
    inherit
        generic_event
    attribute
        rid      : Int;
        rlen     : Int;
        param2   : Int;
        param3   : Int;
        param4   : Int;
        param5   : Int;
        param6   : Int;
        param7   : Int;
        param8   : Int
```

```
end -- Physical_Ack_Receive_Event
-------------------------------------------------------
event
    Physical_Nak_Receive_Event
  inherit
      generic_event
  attribute
      rid     : Int;
      rlen    : Int;
      param2  : Int
end -- Physical_Nak_Receive_Event
-------------------------------------------------------
event
    Set_Paper_Count_Event
  inherit
      generic_event
  attribute
      paper_count : Int
end -- Set_Paper_Count_Event
-------------------------------------------------------
event
    Set_Paper_Position_Event
  inherit
      generic_event
  attribute
      paper_pos : Int
end -- Set_Paper_Position_Event
-------------------------------------------------------
event
    Setting_Change_Event
  inherit
      generic_event
  attribute
      paper_count : Int;
      paper_select_position : Int
end -- Setting_Change_Event
-------------------------------------------------------
-- Field Definition ---------------------------
-------------------------------------------------------
field
```

```
        Ms -- Lcd_Field
    event
        -- for Messeage Part
        Wait, Standby, Copy, Stop,
        Reset, Empty, Restart, Range : generic_event;

        -- for Papaer Size Part
        Size : Paper_Size_Display_Event;
        Select : Paper_Select_Display_Event;

        -- for Paper Count Part
        Count : Paper_Counter_Display_Event;

        -- for Jam Display
        Jam : Jam_Display_Event;

        -- Nak 詳細情報の表示 ( debug用 )
        Nak : Nak_Info_Display_Event;
        -- 通信タイムアウトエラーの表示 ( debug用 )
        Timeout : generic_event;
        -- 予期せぬコピー動作状態の検出 ( debug用 )
        IllState : generic_event

    end -- Lcd_Field
    ------------------------------------------------------------
    field
        Key -- Key_Field
    event
        In : Scan_Code_Event;
        Number : Number_Key_Event;
        Clear, Reset, Start,
        Stop, Select : generic_event

    end -- Key_Field
    ------------------------------------------------------------
    field
        St  -- Status_Field
    event
        -- for Major Status "Waiting"
        Wait, Wait_Empty, Wait_Range : generic_event;
```

```
        -- for Major Status "Standby"
        Standby, Standby_Empty, Standby_Range : generic_event;

        -- for Major Status "Copying"
        Copy, Stop : generic_event;

        -- for Major Status "Jam"
        Jam : generic_event;

        -- for Major Status "Copying-Empty"
        Copy_Empty, Restart : generic_event;

        -- for Major Status "Reset"
        Reset : generic_event

    end -- Status_Field
    ------------------------------------------------------------
    field
        Cm -- ComPort_Field
    event
        Send : Command_Send_Event;
        Ack : Ack_Receive_Event

    end -- ComPort_Field
    ------------------------------------------------------------
    field
        Pp -- PhysicalPort_Field
    event
        Send : Physical_Command_Send_Event;
        Ack : Physical_Ack_Receive_Event;
        Nak : Physical_Nak_Receive_Event

    end -- PhysicalPort_Field
    ------------------------------------------------------------
    field
        Tm -- TimerEvent_Field
    event
        -- 400ms Interval Timer Control
        i400m, i400m_interval,
```

```
    i400m_Start, i400m_Stop : generic_event;

    -- 200ms Interval Timer Control
    i200, i200m_interval,
    i200m_Start, i200m_Stop : generic_event;

    -- 60s Timeout Timer Control
    o60s, o60s_timeout,
    o60s_Start, o60s_Stop : generic_event;

    -- 1s Timeout Timer Control
    o1s, o1s_timeout,
    o1s_Start, o1s_Stop : generic_event

end -- TimerEvent_Field
-------------------------------------------------------
field
    Si -- SettingInfo_Field
event
    DefaultChange : Setting_Change_Event;
    CurrentChange : Setting_Change_Event;
    SetPaperCount : Set_Paper_Count_Event;
    SetPaperPos   : Set_Paper_Position_Event
end -- SettingInfo_Field
-------------------------------------------------------
-- Class Definition ------------------------------
-------------------------------------------------------
class
    Default_Settings
field
    Si
  state
    S1
  attribute
    paper_count : Int;
    paper_select_position : Int
  transition
  Start is
    source
      init
```

```
    do
        paper_count := 1;
        paper_select_position := 1;
        Si.DefaultChange.paper_count := 1;
        Si.DefaultChange.paper_select_position := 1
    destination
        S1
    output
310     Si.DefaultChange
  end
end -- Default_Settings
-------------------------------------------------------------
class
    Current_Settings
field
    Si
  state
      S1
320 attribute
      paper_count : Int;
      paper_select_position : Int
  transition
  Start is
    source
        init
    do
        paper_count := 1;
        paper_select_position := 1;
330     Si.CurrentChange.paper_count := 1;
        Si.CurrentChange.paper_select_position := 1
    destination
        S1
    output
        Si.CurrentChange
  end;
  T1 is
    source
        S1
340 input
        Si.SetPaperCount
```

```
    do
        paper_count := Si.SetPaperCount.paper_count;
        Si.CurrentChange.paper_count := Si.SetPaperCount.paper_count;
        Si.CurrentChange.paper_select_position := paper_select_position
    destination
        S1
    output
        Si.CurrentChange
  end;
  T2 is
    source
        S1
    input
        Si.SetPaperPosition
    do
        paper_select_position := Si.SetPaperPosition.paper_select_position;
        Si.CurrentChange.paper_count := paper_count;
        Si.CurrentChange.paper_select_position := Si.SetPaperPosition.paper_select_position
    destination
        S1
    output
        Si.CurrentChange
  end
end -- Current_Settings
-------------------------------------------------------------
class
    Copy_Control
field
    St, Ms, Tm, Si, Key
  state
      S_ready, S_copy,
      S_start1, S_start2,
      S_stop1, S_stop2,
      S_empty1, S_empty2,
      S_reset, S_select
  attribute
      default_paper_count,
      default_paper_select_position : Int;
      current_paper_count,
      current_paper_select_position : Int;
```

```
        key_input_start_flag : Int
    transition
    Start is
      source
          init
      do
          key_input_start_flag := 0
      destination
          S_ready
    end;
    T1 is
      source
          S_ready
      input
          Si.DefaultChange
      do
        default_paper_count := Si.DefaultChange.paper_count;
        default_paper_select_position := Si.DefaultChange.paper_select_position;
        current_paper_count := Si.DefaultChange.paper_count;
        current_paper_select_position := Si.DefaultChange.paper_select_position
      destination
          S_ready
    end;
    T2 is
      source
          S_ready
      input
          Si.CurrentChange
      do
        current_paper_count := Si.CurrentChange.paper_count;
        current_paper_select_position := Si.CurrentChange.paper_select_position
      destination
          S_ready
    end;
    T3 is
      source
          S_ready
      input
          St.Standby
      do
```

```
        Ms.Count.counter := current_paper_count;
        Ms.Select.current := current_paper_select_position
    destination
        S_ready
    output
        { Tm.o60s_Start, Ms.Count, Ms.Select }
end;
T4 is
    source
        S_ready
    input
        Key.Select
    do
        Cm.Send.cid := CID_GetTrayInfo;
        Cm.Send.clen := CLEN_GetTrayInfo
    destination
        S_select
    output
        { Cm.Send, Tm.o60s_Stop }
end;
T5 is
    source
        S_select
    input
        Cm.Ack
    do
        current_paper_select_position := Cm.Ack.param2;
        Ms.Select.current := Cm.Ack.param2;
        Ms.Count.counter := current_paper_count
    destination
        S_ready
    output
        { Tm.o60s_Start, Ms.Count, Ms.Select }
end;
T6 is
    source
        S_ready
    input
        Key.Reset or Tm.o60s
    do
```

```
        Cm.Send.cid := CID_Reset;
        Cm.Send.clen := CLEN_Reset
    destination
        S_reset
    output
        { Cm.Send, Tm.o60s_Stop }
end;
T7 is
    source
        S_reset
    input
        Cm.Ack
    destination
        S_reset
end;
T7_2 is
    source
        S_reset
    input
        St.Standby
    do
      current_paper_count := default_paper_count;
      current_paper_select_position := default_paper_select_position
    destination
        S_ready
    output
        { Tm.o60s_Start, Ms.Count, Ms.Select }
end;
T8 is
    source
        S_ready
    input
        Key.Start
    do
        Cm.Send.cid := CID_PrinterSetting;
        Cm.Send.clen := CLEN_PrinterSetting;
        Cm.Send.param2 := current_paper_count;
        Cm.Send.param3 := current_paper_select_position
    destination
        S_start1
```

```
    output
        { Tm.o60s_Stop, Cm.Send }
  end;
  T9 is
    source
        S_start1
    input
        Cm.Ack
    do
        Cm.Send.cid := CID_CopyStart;
        Cm.Send.clen := CLEN_CopyStart
    destination
        S_start2
    output
        Cm.Send
  end;
  T10 is
    source
        S_start2
    input
        Cm.Ack
    destination
        S_copy
  end;
  T11 is
    source
        S_copy
    input
        St.Standby
    do
        Ms.Count.counter := current_paper_count;
        Ms.Select.current := current_paper_select_position
    destination
        S_ready
    output
        { Tm.o60s_Start, Ms.Count, Ms.Select }
  end;
  T12 is
    source
        S_copy
```

```
    input
        Key.Stop
    do
        Cm.Send.cid := CID_CopyStop;
        Cm.Send.clen := CLEN_CopyStop
    destination
        S_stop1
    output
        Cm.Send
end;
T13 is
    source
        S_stop1
    input
        Cm.Ack
    destination
        S_stop2
end;
T14 is
    source
        S_stop2
    input
        St.Standby
    do
        Ms.Count.counter := current_paper_count;
        Ms.Select.current := current_paper_select_position
    destination
        S_ready
    output
        { Tm.o60s_Start, Ms.Count, Ms.Select }
end;

-- T15 ～ T20 コピー中の用紙なしと
-- 給紙後の "スタートを押してください" のための追加

T15 is
    source
        S_copy
    input
        St.Copy_Empty
```

```
    destination
        S_empty1
end;
T16 is
    source
        S_empty1
    input
        Key.Stop
    do
        Cm.Send.cid := CID_CopyStop;
        Cm.Send.clen := CLEN_CopyStop
    destination
        S_stop1
    output
        Cm.Send
end;
T17 is
    source
        S_empty1
    input
        St.Restart
    destination
        S_empty2
end;
T18 is
    source
        S_empty2
    input
        St.Copy_Empty
    destination
        S_empty1
end;
T19 is
    source
        S_empty2
    input
        Key.Stop
    do
        Cm.Send.cid := CID_CopyStop;
        Cm.Send.clen := CLEN_CopyStop
```

```
    destination
        S_stop1
    output
        Cm.Send
  end;
  T20 is
    source
        S_empty2
    input
        Key.Start
    do
        Cm.Send.cid := CID_CopyStart;
        Cm.Send.clen := CLEN_CopyStart
    destination
        S_copy
    output
        Cm.Send
  end

end -- Copy_Control
-------------------------------------------------------
-- Status Sub-Status constants
#define CP_STATUS_WAIT          0x01
#define CP_STATUS_STANDBY       0x02
#define CP_STATUS_COPY          0x04
#define CP_STATUS_JAM           0x08
#define CP_STATUS_EMPTY         0x14
#define CP_STATUS_RESET         0xFF

#define CP_SUB_STATUS_COPY      0x07
#define CP_SUB_STATUS_STOP      0x08
#define CP_SUB_STATUS_EMPTY     0x80
-------------------------------------------------------
class
    Status_Control
field
    St, Ms, Tm, Si
  attribute
      copy_status_old : Int;
      copy_sub_status_old  : Int;
```

```
        tray_top_old : Int;
        tray_middle_old : Int;
        tray_bottom_old : Int;
        default_paper_count,
        default_paper_select_position : Int;
        current_paper_count,
        current_paper_select_position : Int
    state
        S_ready, S_wait_status, S_wait_copy_count
    transition
    Start is
      source
          init
      do
          copy_status_old := 0x00;
          copy_sub_status_old  := 0x00;
          tray_top_old := 0xFF;
          tray_middle_old := 0xFF;
          tray_bottom_old := 0xFF;
          default_paper_count := 1;
          default_paper_select_position := 1;
          current_paper_count := 1;
          current_paper_select_position := 1
      destination
          S_ready
      output
          Tm.i400m_Start
    end;
    T1 is
      source
          S_ready
      input
          Tm.i200m
      do
          Cm.Send.cid := CID_GetCopyCount;
          Cm.Send.clen := CLEN_GetCopyCount
      destination
          S_wait_copy_count
      output
          Cm.Send
```

```
    end;
    T2 is
      source
          S_wait_copy_count
      input
          Cm.Ack
      do
          Ms.Count.counter := Cm.Ack.param2
      destination
          S_ready
      output
          Ms.Count
    end;
    T3 is
      source
          S_ready
      input
          Si.DefaultChange
      do
        default_paper_count := Si.DefaultChange.paper_count;
        default_paper_select_position
                  := Si.DefaultChange.paper_select_position;
        current_paper_count := Si.DefaultChange.paper_count;
        current_paper_select_position
                  := Si.DefaultChange.paper_select_position
      destination
          S_ready
    end;
    T4 is
      source
          S_ready
      input
          Si.CurrentChange
      when
          Si.CurrentChange.paper_count = 0
      do
        current_paper_count := Si.CurrentChange.paper_count;
        current_paper_select_position
                  := Si.CurrentChange.paper_select_position
      destination
```

```
        S_ready
    -- output
        -- Ms.Range
end;
T5 is
    source
        S_ready
    input
        Si.CurrentChange
    when
        copy_status_old = CP_STATUS_WAIT
                and Si.CurrentChange.paper_count /= 0
    do
      current_paper_count := Si.CurrentChange.paper_count;
      current_paper_select_position
              := Si.CurrentChange.paper_select_position
    destination
        S_ready
    -- output
        -- Ms.Wait
end;
T6 is
    source
        S_ready
    input
        Si.CurrentChange
    when
        copy_status_old = CP_STATUS_STANDBY
                and Si.CurrentChange.paper_count /= 0
    do
      current_paper_count := Si.CurrentChange.paper_count;
      current_paper_select_position
              := Si.CurrentChange.paper_select_position
    destination
        S_ready
    -- output
        -- Ms.Standby
end;
T7 is
    source
```

```
        S_ready
    input
        Tm.i400m
    do
        Cm.Send.cid := CID_GetStatus;
        Cm.Send.clen := CLEN_GetStatus
    destination
        S_wait_status
    output
        Cm.Send
end;


-- 用紙サイズの変化あり ----------------------------------------

T8 is
-- コピーステータスがウェイト中になった かつ コピーサブステータスが用紙なし
-- かつ 直前がコピー中でなかった かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old /= CP_STATUS_WAIT )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
```

```
        S_ready
    output
        { St.Wait, Ms.Empty, Ms.Size }
end;

T9 is
-- コピーステータスがウェイト中になった かつ コピーサブステータスが用紙なしでなく
-- かつ 直前がコピー中でなかった かつ 現在の部数 =0 かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old /= CP_STATUS_WAIT )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count = 0 )
        and ( Cm.Ack.param4 /= tray_top_old
           or Cm.Ack.param5 /= tray_middle_old
           or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Wait, Ms.Range, Ms.Size }
end;

T10 is
-- コピーステータスがウェイト中になった かつ コピーサブステータスが用紙なしでなく
-- かつ 直前がコピー中でなかった かつ 現在の部数 >0 かつ 用紙サイズの変化あり
    source
```

```
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old /= CP_STATUS_WAIT )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count /= 0 )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Wait, Ms.Size }
end;

T11 is
-- コピーステータスがウェイト中になった かつ コピーサブステータスが用紙なし
-- かつ 直前がコピー中だった ( 200m タイマ停止 ) かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
```

```
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Wait, Ms.Empty, Tm.i200m_Stop, Ms.Size }
end;

T12 is
-- コピーステータスがウェイト中になった かつ コピーサブステータスが用紙なしでなく
-- かつ 現在の部数=0  かつ 直前がコピー中だった ( 200m タイマ停止 )
-- かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count = 0 )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
```

```
        Ms.Size.bottom   := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Wait, Ms.Range, Tm.i200m_Stop, Ms.Size }
end;

T13 is
-- コピーステータスがウェイト中になった かつ コピーサブステータスが用紙なしでなく
-- かつ 現在の部数>0  かつ 直前がコピー中だった ( 200m タイマ停止 )
-- かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count /= 0 )
        and ( Cm.Ack.param4 /= tray_top_old
           or Cm.Ack.param5 /= tray_middle_old
           or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Wait, Ms.Wait, Tm.i200m_Stop, Ms.Size }
end;

--------------------------------------------------
```

```
T14 is
-- コピーステータスがウェイト中のまま かつ コピーサブステータスが用紙なしになった
-- かつ 用紙サイズの変化あり
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_WAIT
              and copy_status_old = CP_STATUS_WAIT )
      and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
                  and copy_sub_status_old /= CP_SUB_STATUS_EMPTY )
      and ( Cm.Ack.param4 /= tray_top_old
         or Cm.Ack.param5 /= tray_middle_old
         or Cm.Ack.param6 /= tray_bottom_old )
  do
      copy_status_old := Cm.Ack.param2;
      copy_sub_status_old := Cm.Ack.param3;
      tray_top_old    := Cm.Ack.param4;
      tray_middle_old := Cm.Ack.param5;
      tray_bottom_old := Cm.Ack.param6;
      Ms.Size.top     := Cm.Ack.param4;
      Ms.Size.middle  := Cm.Ack.param5;
      Ms.Size.bottom  := Cm.Ack.param6
  destination
      S_ready
  output
      { Ms.Empty, Ms.Size }
end;

T15 is
-- コピーステータスがウェイト中のまま かつ コピーサブステータスが用紙なしでなくなった
-- かつ 現在の部数 =0 かつ 用紙サイズの変化あり
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_WAIT
              and copy_status_old = CP_STATUS_WAIT )
```

```
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                    and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
        and ( current_paper_count = 0 )
        and ( Cm.Ack.param4 /= tray_top_old
           or Cm.Ack.param5 /= tray_middle_old
           or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { Ms.Range, Ms.Size }
end;

T16 is
-- コピーステータスがウェイト中のまま かつ コピーサブステータスが用紙なしでなくなった
-- かつ 現在の部数 >0 かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old = CP_STATUS_WAIT )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                    and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
        and ( current_paper_count /= 0 )
        and ( Cm.Ack.param4 /= tray_top_old
           or Cm.Ack.param5 /= tray_middle_old
           or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
```

```
        tray_top_old      := Cm.Ack.param4;
        tray_middle_old   := Cm.Ack.param5;
        tray_bottom_old   := Cm.Ack.param6;
        Ms.Size.top       := Cm.Ack.param4;
        Ms.Size.middle    := Cm.Ack.param5;
        Ms.Size.bottom    := Cm.Ack.param6
    destination
        S_ready
    output
        { Ms.Wait, Ms.Size }
end;

T17 is
-- コピーステータスがウェイト中のまま かつ コピーサブステータスが用紙なしのまま
-- かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old = CP_STATUS_WAIT )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
                    and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 /= tray_top_old
           or Cm.Ack.param5 /= tray_middle_old
           or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old   := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old      := Cm.Ack.param4;
        tray_middle_old   := Cm.Ack.param5;
        tray_bottom_old   := Cm.Ack.param6;
        Ms.Size.top       := Cm.Ack.param4;
        Ms.Size.middle    := Cm.Ack.param5;
        Ms.Size.bottom    := Cm.Ack.param6
    destination
        S_ready
    output
        { Ms.Size }
```

```
end;

T18 is
-- コピーステータスがウェイト中のまま
-- かつ コピーサブステータスが用紙なしでないまま
-- かつ 用紙サイズの変化あり
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_WAIT
              and copy_status_old = CP_STATUS_WAIT )
      and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                  and copy_sub_status_old /= CP_SUB_STATUS_EMPTY )
      and ( Cm.Ack.param4 /= tray_top_old
         or Cm.Ack.param5 /= tray_middle_old
         or Cm.Ack.param6 /= tray_bottom_old )
  do
      copy_status_old := Cm.Ack.param2;
      copy_sub_status_old := Cm.Ack.param3;
      tray_top_old    := Cm.Ack.param4;
      tray_middle_old := Cm.Ack.param5;
      tray_bottom_old := Cm.Ack.param6;
      Ms.Size.top     := Cm.Ack.param4;
      Ms.Size.middle  := Cm.Ack.param5;
      Ms.Size.bottom  := Cm.Ack.param6
  destination
      S_ready
  output
      { Ms.Size }
end;

--------------------------------------------------

T19 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なし
-- かつ 直前がコピー中でなかった かつ 用紙サイズの変化あり
  source
      S_wait_status
```

```
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old /= CP_STATUS_STANDBY )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Standby, Ms.Empty, Ms.Size }
end;

T20 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なしでなく
-- かつ 直前がコピー中でなかった かつ 現在の部数 =0 かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old /= CP_STATUS_STANDBY )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count = 0 )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
```

```
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Standby, Ms.Range, Ms.Size }
end;

T21 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なしでなく
-- かつ 直前がコピー中でなかった かつ 現在の部数 >0 かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old /= CP_STATUS_STANDBY )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count /= 0 )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
```

```
        Ms.Size.bottom   := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Standby, Ms.Standby, Ms.Size }
end;

T22 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なし
-- かつ 直前がコピー中だった ( 200m タイマ停止 ) かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 /= tray_top_old
           or Cm.Ack.param5 /= tray_middle_old
           or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old      := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top       := Cm.Ack.param4;
        Ms.Size.middle   := Cm.Ack.param5;
        Ms.Size.bottom   := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Standby, Ms.Empty, Tm.i200m_Stop, Ms.Size }
end;

T23 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なしでなく
-- かつ 現在の部数=0  かつ 直前がコピー中だった ( 200m タイマ停止 )
-- かつ 用紙サイズの変化あり
```

```
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count = 0 )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Standby, Ms.Range, Tm.i200m_Stop, Ms.Size }
end;

T24 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なしでなく
-- かつ 現在の部数>0  かつ 直前がコピー中だった ( 200m タイマ停止 )
-- かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count /= 0 )
```

```
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Standby, Ms.Standby, Tm.i200m_Stop, Ms.Size }
end;

------------------------------------------

T25 is
-- コピーステータスがスタンバイのまま かつ コピーサブステータスが用紙なしになった
-- かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_STANDBY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
                    and copy_sub_status_old /= CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
```

```
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { Ms.Empty, Ms.Size }
end;

T26 is
-- コピーステータスがスタンバイのまま かつ コピーサブステータスが用紙なしでなくなった
--  かつ 現在の部数 =0 かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_STANDBY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                    and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
        and ( current_paper_count = 0 )
        and ( Cm.Ack.param4 /= tray_top_old
           or Cm.Ack.param5 /= tray_middle_old
           or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { Ms.Range, Ms.Size }
end;
```

```
T27 is
-- コピーステータスがスタンバイのまま かつ コピーサブステータスが用紙なしでなくなった
-- かつ 現在の部数 >0 かつ 用紙サイズの変化あり
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_STANDBY
              and copy_status_old = CP_STATUS_STANDBY )
      and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                  and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
      and ( current_paper_count /= 0 )
      and ( Cm.Ack.param4 /= tray_top_old
         or Cm.Ack.param5 /= tray_middle_old
         or Cm.Ack.param6 /= tray_bottom_old )
  do
      copy_status_old := Cm.Ack.param2;
      copy_sub_status_old := Cm.Ack.param3;
      tray_top_old    := Cm.Ack.param4;
      tray_middle_old := Cm.Ack.param5;
      tray_bottom_old := Cm.Ack.param6;
      Ms.Size.top     := Cm.Ack.param4;
      Ms.Size.middle  := Cm.Ack.param5;
      Ms.Size.bottom  := Cm.Ack.param6
  destination
      S_ready
  output
      { Ms.Standby, Ms.Size }
end;

T28 is
-- コピーステータスがスタンバイのまま かつ コピーサブステータスが用紙なしのまま
-- かつ 用紙サイズの変化あり
  source
      S_wait_status
  input
      Cm.Ack
  when
```

```
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_STANDBY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
                and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 /= tray_top_old
          or Cm.Ack.param5 /= tray_middle_old
          or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { Ms.Size }
end;

T29 is
-- コピーステータスがスタンバイのまま かつ コピーサブステータスが用紙なしでないまま
-- かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_STANDBY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                and copy_sub_status_old /= CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 /= tray_top_old
          or Cm.Ack.param5 /= tray_middle_old
          or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
```

```
        tray_top_old      := Cm.Ack.param4;
        tray_middle_old   := Cm.Ack.param5;
        tray_bottom_old   := Cm.Ack.param6;
        Ms.Size.top       := Cm.Ack.param4;
        Ms.Size.middle    := Cm.Ack.param5;
        Ms.Size.bottom    := Cm.Ack.param6
    destination
        S_ready
    output
        { Ms.Size }
end;

------------------------------------

T30 is
-- コピーステータスがコピー中になった ( 200m タイマ起動 )
-- かつ コピーサブステータスがコピー中 かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_COPY
                and copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_COPY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old   := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old      := Cm.Ack.param4;
        tray_middle_old   := Cm.Ack.param5;
        tray_bottom_old   := Cm.Ack.param6;
        Ms.Size.top       := Cm.Ack.param4;
        Ms.Size.middle    := Cm.Ack.param5;
        Ms.Size.bottom    := Cm.Ack.param6
    destination
        S_ready
    output
```

```
        { St.Copy, Tm.i200m_Start, Ms.Copy, Ms.Size }
    end;

    T31 is
    -- コピーステータスがコピー中になった ( 200m タイマ起動 )
    -- かつ コピーサブステータスがストップ中 かつ 用紙サイズの変化あり
      source
        S_wait_status
      input
        Cm.Ack
      when
        ( Cm.Ack.param2 = CP_STATUS_COPY
                and copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_STOP )
        and ( Cm.Ack.param4 /= tray_top_old
           or Cm.Ack.param5 /= tray_middle_old
           or Cm.Ack.param6 /= tray_bottom_old )
      do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
      destination
        S_ready
      output
        { St.Copy, Tm.i200m_Start, Ms.Stop, Ms.Size }
    end;

    ------------------------------------------------

    T32 is
    -- コピーステータスがコピー中のまま かつ コピーサブステータスがコピー中になった
    -- かつ 用紙サイズの変化あり
      source
        S_wait_status
      input
```

```
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_COPY
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_COPY
                    and copy_sub_status_old /= CP_SUB_STATUS_COPY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { Ms.Copy, Ms.Size }
end;

T33 is
-- コピーステータスがコピー中のまま かつ コピーサブステータスがストップ中になった
-- かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_COPY
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_STOP
                    and copy_sub_status_old /= CP_SUB_STATUS_STOP )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
```

```
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { Ms.Stop, Ms.Size }
end;

-------------------------------------------

T34 is
-- コピーステータスが用紙詰まり(JAM)になった
-- かつ 直前がコピー中でなかった かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_JAM
                and copy_status_old /= CP_STATUS_JAM )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Jam, Ms.Jam } -- { Ms.Size }
end

T35 is
```

```
-- コピーステータスが用紙詰まり(JAM)になった
-- かつ 直前がコピー中だった( 200m タイマ停止 )かつ 用紙サイズの変化あり
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_JAM
              and copy_status_old = CP_STATUS_COPY )
      and ( Cm.Ack.param4 /= tray_top_old
         or Cm.Ack.param5 /= tray_middle_old
         or Cm.Ack.param6 /= tray_bottom_old )
  do
      copy_status_old := Cm.Ack.param2;
      copy_sub_status_old := Cm.Ack.param3
  destination
      S_ready
  output
      { St.Jam, Ms.Jam, Tm.i200m_Stop } -- { Ms.Size }
end;

-----------------------------------------

T36 is
-- コピーステータスが用紙詰まり(JAM)のまま かつ 用紙サイズの変化あり
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_JAM
              and copy_status_old = CP_STATUS_JAM )
      and ( Cm.Ack.param4 /= tray_top_old
         or Cm.Ack.param5 /= tray_middle_old
         or Cm.Ack.param6 /= tray_bottom_old )
  do
      copy_status_old := Cm.Ack.param2;
      copy_sub_status_old := Cm.Ack.param3
  destination
      S_ready
```

```
    -- output
        -- { Ms.Size }
end;

-----------------------------------------

T37 is
-- コピーステータスがコピー中の用紙なしになった
-- かつ 直前がコピー中でなかった かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_EMPTY
                and copy_status_old /= CP_STATUS_EMPTY )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Copy_Empty, Ms.Empty, Ms.Size }
end;

T38 is
-- コピーステータスがコピー中の用紙なしになった
-- かつ 直前がコピー中だった ( 200m タイマ停止 ) かつ 用紙サイズの変化あり
    source
        S_wait_status
```

```
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_EMPTY
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Copy_Empty, Ms.Empty, Tm.i200m_Stop, Ms.Size }
end;

------------------------------------

T39 is
-- コピーステータスがコピー中の用紙なしのまま
-- かつ コピーサブステータスが用紙なしのまま
-- かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_EMPTY
                and copy_status_old = CP_STATUS_EMPTY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
                and copy_status_old = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
```

```
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { Ms.Size }
end;

T39_2 is
-- コピーステータスがコピー中の用紙なしのまま
-- かつ コピーサブステータスが用紙なしでなくなった
-- かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_EMPTY
                and copy_status_old = CP_STATUS_EMPTY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                and copy_status_old = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
```

```
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Restart, Ms.Restart, Ms.Size }
end;
T39_3 is
-- コピーステータスがコピー中の用紙なしのまま
-- かつ コピーサブステータスが用紙なしになった
-- かつ 用紙サイズの変化あり
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_EMPTY
                and copy_status_old = CP_STATUS_EMPTY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
                and copy_status_old /= CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 /= tray_top_old
            or Cm.Ack.param5 /= tray_middle_old
            or Cm.Ack.param6 /= tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old    := Cm.Ack.param4;
        tray_middle_old := Cm.Ack.param5;
        tray_bottom_old := Cm.Ack.param6;
        Ms.Size.top     := Cm.Ack.param4;
        Ms.Size.middle  := Cm.Ack.param5;
        Ms.Size.bottom  := Cm.Ack.param6
    destination
        S_ready
    output
        { St.Copy_Empty, Ms.Empty, Ms.Size }
end;

-- 用紙サイズの変化なし ----------------------------------------

T40 is
```

```
-- コピーステータスがウェイト中になった かつ コピーサブステータスが用紙なし
-- かつ 直前がコピー中でなかった かつ 用紙サイズの変化なし
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_WAIT
              and copy_status_old /= CP_STATUS_WAIT )
      and ( copy_status_old /= CP_STATUS_COPY )
      and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY )
      and ( Cm.Ack.param4 = tray_top_old
          and Cm.Ack.param5 = tray_middle_old
          and Cm.Ack.param6 = tray_bottom_old )
  do
      copy_status_old := Cm.Ack.param2;
      copy_sub_status_old := Cm.Ack.param3
  destination
      S_ready
  output
      { St.Wait, Ms.Empty }
end;

T41 is
-- コピーステータスがウェイト中になった かつ コピーサブステータスが用紙なしでなく
-- かつ 直前がコピー中でなかった かつ 現在の部数 =0  かつ 用紙サイズの変化なし
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_WAIT and copy_status_old /= CP_STATUS_WAIT )
      and ( copy_status_old /= CP_STATUS_COPY )
      and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
      and ( current_paper_count = 0 )
      and ( Cm.Ack.param4 = tray_top_old
          and Cm.Ack.param5 = tray_middle_old
          and Cm.Ack.param6 = tray_bottom_old )
  do
      copy_status_old := Cm.Ack.param2;
```

```
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Wait, Ms.Range }
end;

T42 is
-- コピーステータスがウェイト中になった かつ コピーサブステータスが用紙なしでなく
-- かつ 直前がコピー中でなかった かつ 現在の部数 >0  かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old /= CP_STATUS_WAIT )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count /= 0 )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Wait }
end;

T43 is
-- コピーステータスがウェイト中になった かつ コピーサブステータスが用紙なし
-- かつ 直前がコピー中だった ( 200m タイマ停止 ) かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
```

```
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Wait, Ms.Empty, Tm.i200m_Stop }
end;

T44 is
-- コピーステータスがウェイト中になった かつ コピーサブステータスが用紙なしでなく
-- かつ 現在の部数=0  かつ 直前がコピー中だった ( 200m タイマ停止 )
-- かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count = 0 )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Wait, Ms.Range, Tm.i200m_Stop }
end;
```

```
T45 is
-- コピーステータスがウェイト中になった
-- かつ コピーサブステータスが用紙なしでなく かつ 現在の部数>0
-- かつ 直前がコピー中だった ( 200m タイマ停止 )  かつ 用紙サイズの変化なし
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_WAIT
              and copy_status_old = CP_STATUS_COPY )
      and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
      and ( current_paper_count /= 0 )
      and ( Cm.Ack.param4 = tray_top_old
         and Cm.Ack.param5 = tray_middle_old
         and Cm.Ack.param6 = tray_bottom_old )
  do
      copy_status_old := Cm.Ack.param2;
      copy_sub_status_old := Cm.Ack.param3
  destination
      S_ready
  output
      { St.Wait, Ms.Wait, Tm.i200m_Stop }
end;

------------------------------------------

T46 is
-- コピーステータスがウェイト中のまま
-- かつ コピーサブステータスが用紙なしになった  かつ 用紙サイズの変化なし
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_WAIT
              and copy_status_old = CP_STATUS_WAIT )
      and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
                  and copy_sub_status_old /= CP_SUB_STATUS_EMPTY )
      and ( Cm.Ack.param4 = tray_top_old
```

```
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { Ms.Empty }
end;

T47 is
-- コピーステータスがウェイト中のまま かつ コピーサブステータスが用紙なしでなくなった
-- かつ 現在の部数 =0  かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old = CP_STATUS_WAIT )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                    and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
        and ( current_paper_count = 0 )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { Ms.Range }
end;

T48 is
-- コピーステータスがウェイト中のまま かつ コピーサブステータスが用紙なしでなくなった
-- かつ 現在の部数 >0  かつ 用紙サイズの変化なし
    source
```

```
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old = CP_STATUS_WAIT )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                    and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
        and ( current_paper_count /= 0 )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { Ms.Wait }
end;


T49 is
-- コピーステータスがウェイト中のまま
-- かつ コピーサブステータスが用紙なしのまま　かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old = CP_STATUS_WAIT )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
                    and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
```

```
    destination
        S_ready
    -- output
        -- None
end;

T50 is
-- コピーステータスがウェイト中のまま
-- かつ コピーサブステータスが用紙なしでないまま　かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_WAIT
                and copy_status_old = CP_STATUS_WAIT )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                    and copy_sub_status_old /= CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    -- output
        -- None
end;

-------------------------------------

T51 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なし
-- かつ 直前がコピー中でなかった　かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
```

```
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old /= CP_STATUS_STANDBY )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Standby, Ms.Empty }
end;


T52 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なしでなく
-- かつ 直前がコピー中でなかった かつ 現在の部数 =0  かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old /= CP_STATUS_STANDBY )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count = 0 )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Standby, Ms.Range }
```

```
end;

T53 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なしでなく
-- かつ 直前がコピー中でなかった かつ 現在の部数 >0  かつ 用紙サイズの変化なし
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_STANDBY
              and copy_status_old /= CP_STATUS_STANDBY )
      and ( copy_status_old /= CP_STATUS_COPY )
      and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
      and ( current_paper_count /= 0 )
      and ( Cm.Ack.param4 = tray_top_old
         and Cm.Ack.param5 = tray_middle_old
         and Cm.Ack.param6 = tray_bottom_old )
  do
      copy_status_old := Cm.Ack.param2;
      copy_sub_status_old := Cm.Ack.param3
  destination
      S_ready
  output
      { St.Standby, Ms.Standby }
end;

T54 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なし
--  かつ 直前がコピー中だった( 200m タイマ停止 )  かつ 用紙サイズの変化なし
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_STANDBY
              and copy_status_old = CP_STATUS_COPY )
      and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY )
      and ( Cm.Ack.param4 = tray_top_old
         and Cm.Ack.param5 = tray_middle_old
```

```
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Standby, Ms.Empty, Tm.i200m_Stop }
end;

T55 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なしでなく
-- かつ 現在の部数=0　かつ 直前がコピー中だった ( 200m タイマ停止 )
-- かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count = 0 )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Standby, Ms.Range, Tm.i200m_Stop }
end;


T56 is
-- コピーステータスがスタンバイになった かつ コピーサブステータスが用紙なしでなく
-- かつ 現在の部数>0　かつ 直前がコピー中だった ( 200m タイマ停止 )
-- かつ 用紙サイズの変化なし
```

```
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY )
        and ( current_paper_count /= 0 )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Standby, Ms.Standby, Tm.i200m_Stop }
end;

-----------------------------------------

T57 is
-- コピーステータスがスタンバイのまま
-- かつ コピーサブステータスが用紙なしになった　かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_STANDBY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
                    and copy_sub_status_old /= CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
```

```
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { Ms.Empty }
end;

T58 is
-- コピーステータスがスタンバイのまま かつ コピーサブステータスが用紙なしでなくなった
-- かつ 現在の部数 =0  かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_STANDBY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                    and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
        and ( current_paper_count = 0 )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { Ms.Range }
end;

T59 is
-- コピーステータスがスタンバイのまま かつ コピーサブステータスが用紙なしでなくなった
-- かつ 現在の部数 >0  かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
```

```
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_STANDBY )
        and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
        and ( current_paper_count /= 0 )
        and ( Cm.Ack.param4 = tray_top_old
           and Cm.Ack.param5 = tray_middle_old
           and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { Ms.Standby }
end;

T60 is
-- コピーステータスがスタンバイのまま
-- かつ コピーサブステータスが用紙なしのまま　かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_STANDBY
                and copy_status_old = CP_STATUS_STANDBY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
                and copy_sub_status_old = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 = tray_top_old
           and Cm.Ack.param5 = tray_middle_old
           and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    -- output
        -- None
end;
```

```
T61 is
-- コピーステータスがスタンバイのまま
-- かつ コピーサブステータスが用紙なしでないまま　かつ 用紙サイズの変化なし
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_STANDBY
              and copy_status_old = CP_STATUS_STANDBY )
      and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
                  and copy_sub_status_old /= CP_SUB_STATUS_EMPTY )
      and ( Cm.Ack.param4 = tray_top_old
         and Cm.Ack.param5 = tray_middle_old
         and Cm.Ack.param6 = tray_bottom_old )
  do
      copy_status_old := Cm.Ack.param2;
      copy_sub_status_old := Cm.Ack.param3
  destination
      S_ready
  -- output
      -- None
end;

-----------------------------------------

T62 is
-- コピーステータスがコピー中になった ( 200m タイマ起動 )
-- かつ コピーサブステータスがコピー中　かつ 用紙サイズの変化なし
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_COPY
              and copy_status_old /= CP_STATUS_COPY )
      and ( Cm.Ack.param3 = CP_SUB_STATUS_COPY )
      and ( Cm.Ack.param4 = tray_top_old
         and Cm.Ack.param5 = tray_middle_old
```

```
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Copy, Tm.i200m_Start, Ms.Copy }
end;

T63 is
-- コピーステータスがコピー中になった ( 200m タイマ起動 )
-- かつ コピーサブステータスがストップ中　かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_COPY
                and copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_STOP )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Copy, Tm.i200m_Start, Ms.Stop }
end;

--------------------------------------------------

T64 is
-- コピーステータスがコピー中のまま
-- かつ コピーサブステータスがコピー中になった　かつ 用紙サイズの変化なし
    source
        S_wait_status
```

```
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_COPY
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_COPY
                    and copy_sub_status_old /= CP_SUB_STATUS_COPY )
        and ( Cm.Ack.param4 = tray_top_old
           and Cm.Ack.param5 = tray_middle_old
           and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { Ms.Copy }
end;

T65 is
-- コピーステータスがコピー中のまま
-- かつ コピーサブステータスがストップ中になった　かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_COPY
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_STOP
                    and copy_sub_status_old /= CP_SUB_STATUS_STOP )
        and ( Cm.Ack.param4 = tray_top_old
           and Cm.Ack.param5 = tray_middle_old
           and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
```

```
        { Ms.Stop }
    end;

    ----------------------------------------

    T66 is
    -- コピーステータスが用紙詰まり(JAM)になった
    -- かつ 直前がコピー中でなかった　かつ 用紙サイズの変化なし
      source
          S_wait_status
      input
          Cm.Ack
      when
          ( Cm.Ack.param2 = CP_STATUS_JAM
                  and copy_status_old /= CP_STATUS_JAM )
          and ( copy_status_old /= CP_STATUS_COPY )
          and ( Cm.Ack.param4 = tray_top_old
              and Cm.Ack.param5 = tray_middle_old
              and Cm.Ack.param6 = tray_bottom_old )
      do
          copy_status_old := Cm.Ack.param2;
          copy_sub_status_old := Cm.Ack.param3
      destination
          S_ready
      output
          { St.Jam, Ms.Jam }
    end;

    T67 is
    -- コピーステータスが用紙詰まり(JAM)になった
    -- かつ 直前がコピー中だった( 200m タイマ停止 )　かつ 用紙サイズの変化なし
      source
          S_wait_status
      input
          Cm.Ack
      when
          ( Cm.Ack.param2 = CP_STATUS_JAM
                  and copy_status_old = CP_STATUS_COPY )
          and ( Cm.Ack.param4 = tray_top_old
              and Cm.Ack.param5 = tray_middle_old
```

```
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Jam, Ms.Jam, Tm.i200m_Stop }
end;

-------------------------------------------
T68 is
-- コピーステータスが用紙詰まり(JAM)のまま
-- かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_JAM
                and copy_status_old = CP_STATUS_JAM )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    -- output
        -- None
end;

-------------------------------------------

T69 is
-- コピーステータスがコピー中の用紙なしになった
-- かつ 直前がコピー中でなかった　かつ 用紙サイズの変化なし
    source
        S_wait_status
```

```
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_EMPTY
                and copy_status_old /= CP_STATUS_EMPTY )
        and ( copy_status_old /= CP_STATUS_COPY )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Copy_Empty, Ms.Empty }
end;


T70 is
-- コピーステータスがコピー中の用紙なしになった
-- かつ 直前がコピー中だった ( 200m タイマ停止 )  かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_EMPTY
                and copy_status_old = CP_STATUS_COPY )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Copy_Empty, Ms.Empty, Tm.i200m_Stop }
end;
```

```
-----------------------------------------

T71 is
-- コピーステータスがコピー中の用紙なしのまま
-- かつ コピーサブステータスが用紙なしのまま
-- かつ 用紙サイズの変化なし
  source
2590      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_EMPTY
              and copy_status_old = CP_STATUS_EMPTY )
      and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
              and copy_status_old = CP_SUB_STATUS_EMPTY )
      and ( Cm.Ack.param4 = tray_top_old
         and Cm.Ack.param5 = tray_middle_old
2600     and Cm.Ack.param6 = tray_bottom_old )
  do
      copy_status_old := Cm.Ack.param2;
      copy_sub_status_old := Cm.Ack.param3
  destination
      S_ready
  -- output
      -- None
end

2610 T71_2 is
-- コピーステータスがコピー中の用紙なしのまま
-- かつ コピーサブステータスが用紙なしでなくなった
-- かつ 用紙サイズの変化なし
  source
      S_wait_status
  input
      Cm.Ack
  when
      ( Cm.Ack.param2 = CP_STATUS_EMPTY
2620          and copy_status_old = CP_STATUS_EMPTY )
      and ( Cm.Ack.param3 /= CP_SUB_STATUS_EMPTY
```

```
                and copy_status_old = CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Restart, Ms.Restart }
end

T71_3 is
-- コピーステータスがコピー中の用紙なしのまま
-- かつ コピーサブステータスが用紙なしになった
-- かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_EMPTY
                and copy_status_old = CP_STATUS_EMPTY )
        and ( Cm.Ack.param3 = CP_SUB_STATUS_EMPTY
                and copy_status_old /= CP_SUB_STATUS_EMPTY )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Copy_Empty, Ms.Empty }
end;

T72 is
-- コピーステータスがリセット中になった
```

```
        -- かつ 用紙サイズの変化あり
          source
              S_wait_status
          input
              Cm.Ack
          when
              ( Cm.Ack.param2 = CP_STATUS_RESET
                        and copy_status_old /= CP_STATUS_RESET )
2670          and ( Cm.Ack.param4 /= tray_top_old
                  or Cm.Ack.param5 /= tray_middle_old
                  or Cm.Ack.param6 /= tray_bottom_old )
          do
              copy_status_old := Cm.Ack.param2;
              copy_sub_status_old := Cm.Ack.param3;
              tray_top_old    := Cm.Ack.param4;
              tray_middle_old := Cm.Ack.param5;
              tray_bottom_old := Cm.Ack.param6;
              Ms.Size.top     := Cm.Ack.param4;
2680          Ms.Size.middle  := Cm.Ack.param5;
              Ms.Size.bottom  := Cm.Ack.param6
          destination
              S_ready
          output
              { St.Reset, Ms.Reset, Ms.Size }
        end;

        T72_2 is
        -- コピーステータスがリセット中のまま
2690    -- かつ 用紙サイズの変化あり
          source
              S_wait_status
          input
              Cm.Ack
          when
              ( Cm.Ack.param2 = CP_STATUS_RESET
                        and copy_status_old = CP_STATUS_RESET )
              and ( Cm.Ack.param4 /= tray_top_old
                  or Cm.Ack.param5 /= tray_middle_old
2700              or Cm.Ack.param6 /= tray_bottom_old )
          do
```

```
        copy_status_old  := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3;
        tray_top_old     := Cm.Ack.param4;
        tray_middle_old  := Cm.Ack.param5;
        tray_bottom_old  := Cm.Ack.param6;
        Ms.Size.top      := Cm.Ack.param4;
        Ms.Size.middle   := Cm.Ack.param5;
        Ms.Size.bottom   := Cm.Ack.param6
    destination
        S_ready
    output
        Ms.Size
end;

T73 is
-- コピーステータスがリセット中になった
-- かつ 用紙サイズの変化なし
    source
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_RESET
                and copy_status_old /= CP_STATUS_RESET )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
    output
        { St.Reset, Ms.Reset }
end;

T73_2 is
-- コピーステータスがリセット中のまま
-- かつ 用紙サイズの変化なし
    source
```

```
        S_wait_status
    input
        Cm.Ack
    when
        ( Cm.Ack.param2 = CP_STATUS_RESET
                and copy_status_old = CP_STATUS_RESET )
        and ( Cm.Ack.param4 = tray_top_old
            and Cm.Ack.param5 = tray_middle_old
            and Cm.Ack.param6 = tray_bottom_old )
    do
        copy_status_old := Cm.Ack.param2;
        copy_sub_status_old := Cm.Ack.param3
    destination
        S_ready
  end

end -- Status_Control
-------------------------------------------------------------
-------------------------------------------------------------
class
    Timer_400m_Interval
field
    Tm
  state
      S_on, S_off
  transition
  Start is
  source
      init
  destination
      S_off
  end;
  T1 is
  source
      S_off
  input
      Tm.i400m_Start
  destination
      S_on
  end;
```

```
    T2 is
      source
          S_on
      input
          Tm.i400m_Stop
      destination
          S_off
    end;
    T3 is
      source
          S_on
      input
          Tm.i400m_interval
      destination
          S_on
    end
  end -- Timer_400m_Interval
  ------------------------------------------------------------
  class
      Timer_200m_Interval
  field
      Tm
    state
        S_on, S_off
    transition
    Start is
      source
          init
      destination
          S_off
    end;
    T1 is
      source
          S_off
      input
          Tm.i200m_Start
      destination
          S_on
    end;
    T2 is
```

```
        source
            S_on
        input
            Tm.i200m_Stop
        destination
            S_off
    end;
    T3 is
        source
            S_on
        input
            Tm.i200m_interval
        destination
            S_on
    end
end -- Timer_200m_Interval
-------------------------------------------------------------
class
    Timer_60s_Timeout
field
    Tm
    state
        S_on, S_off
    transition
    Start is
        source
            init
        destination
            S_off
    end;
    T1 is
        source
            S_off
        input
            Tm.o60s_Start
        destination
            S_on
    end;
    T2 is
        source
```

```
        S_on
    input
        Tm.o60s_Stop
    destination
        S_off
  end;
  T3 is
    source
        S_on
    input
        Tm.o60s_timeout
    destination
        S_off
  end
end -- Timer_60s_Timeout
--------------------------------------------------
class
    Timer_1s_Timeout
field
    Tm
  state
      S_on, S_off
  transition
  Start is
    source
        init
    destination
        S_off
  end;
  T1 is
    source
        S_off
    input
        Tm.o1s_Start
    destination
        S_on
  end;
  T2 is
    source
        S_on
```

```
    input
        Tm.o1s_Stop
    destination
        S_off
  end;
  T3 is
    source
        S_on
    input
        Tm.o1s_timeout
    destination
        S_off
  end
end -- Timer_1s_Timeout
-----------------------------------------------------
class
    Timer_Control
  inner
      timer_ctrl_children : {
      tm_400m_ctrl : Timer_400m_Interval;
      tm_200m_ctrl : Timer_200m_Interval;
      tm_60s_ctrl : Timer_60s_Timeout;
      tm_1s_ctrl : Timer_1s_Timeout
      }
  transition
  Start is
    source
        init
    destination
        timer_ctrl_children
  end
end -- Timer_Control
-----------------------------------------------------
class
    Key_Interface
field
    Key, Ms
  state
      S1, S2
  transition
```

Start **is**
source
init
destination
S1
end;
T10 **is**
source
S1
input
Key.In
when
Key.In.ScanCode = 10
destination
S1
output
Key.Clear
end;
T11 **is**
source
S1
input
Key.In
when
Key.In.ScanCode = 11
destination
S1
output
Key.Reset
end;
T12 **is**
source
S1
input
Key.In
when
Key.In.ScanCode = 12
destination
S1
output

```
        Key.Start
    end;
    T13 is
      source
        S1
      input
        Key.In
      when
2990    Key.In.ScanCode = 13
      destination
        S1
      output
        Key.Stop
    end;
    T14 is
      source
        S1
      input
3000    Key.In
      when
        Key.In.ScanCode = 14
      destination
        S1
      output
        Key.Select
    end;
    Tn is
      source
3010    S1
      input
        Key.In
      when
        ( Key.In.code = 0 or Key.In.code > 0 )
        and
        ( Key.In.code = 9 or Key.In.code < 9 )
      do
        Key.Number.code := Key.In.code
      destination
3020    S1
      output
```

```
        Key.Number
    end;
    Td is
      source
        S1
      input
        Ms.Jam
      destination
3030    S2
    end
  end -- Key_Interface
  ------------------------------------------------------------
  class
    ComPort_Interface
  field
    Cm,  -- ComPort_Field
    Pp,  -- PhysicalPort_Field
    Tm,  -- Timer_Field
3040    Ms   -- LCD_Field
    state
      S_ready, S_wait, S_nak, S_timeout
    transition
    Start is
      source
        init
      destination
        S_ready
    end;
3050  T1 is
      source
        S_ready
      input
        Cm.Send
      do
        -- シリアル送信バッファへの転送
        Pp.Send.cid := Cm.Send.cid;
        Pp.Send.cid := Cm.Send.clen;
        Pp.Send.param2 := Cm.Send.param2;
3060    Pp.Send.param3 := Cm.Send.param3;
        Pp.Send.param4 := Cm.Send.param4
```

```
    destination
        S_wait
    output
        -- 疑似送信
        -- 通信タイムアウト用タイマの起動
        { Pp.Send, Tm.o1s_Start }
end;
T2 is
3070  source
        S_wait
    input
        -- 疑似受信割り込み
        Pp.Ack
    do
        -- 割り込み側からの受信バッファの転送
        Cm.Ack.rid    := Pp.Ack.rid;
        Cm.Ack.rlen   := Pp.Ack.rlen;
        Cm.Ack.param2 := Pp.Ack.param2;
3080    Cm.Ack.param3 := Pp.Ack.param3;
        Cm.Ack.param4 := Pp.Ack.param4;
        Cm.Ack.param5 := Pp.Ack.param5;
        Cm.Ack.param6 := Pp.Ack.param6
    destination
        S_ready
    output
        -- Ack レスポンス受信イベント
        -- 通信タイムアウト用タイマの停止
        { Cm.Ack, Tm.o1s_Stop }
3090 end;
T3 is
    source
        S_wait
    input
        -- 疑似受信割り込み
        Pp.Nak
    do
        -- Nak 詳細情報の転送
        Ms.Nak.info := Pp.Nak.param2
3100  destination
        S_nak
```

```
      output
          -- Nak 詳細情報の表示( debug用 )
          -- 通信タイムアウト用タイマの停止
          { Ms.Nak, Tm.o1s_Stop }
    end;
    T4 is
      source
          S_wait
      input
          -- 通信タイムアウトイベント
          Tm.o1s
      destination
          S_timeout
      output
          -- 通信タイムアウトエラーの表示( debug用 )
          -- 通信タイムアウト用タイマの停止
          { Ms.Timeout, Tm.o1s_Stop }
    end
end -- ComPort_Interface
-------------------------------------------------------------
class
    Message_LCD
field
    Ms
  attribute
      message : String;
      nak_code : Int
state
      S1
transition
  Start is
    source
        init
    do
        message := "しばらくおまちください。";
        nak_code := 0
    destination
        S1
  end;
  T1 is
```

```
    source
        S1
    input
        Ms.Wait
    do
        message := "しばらくおまちください。"
    destination
        S1
end;
T2 is
    source
        S1
    input
        Ms.Standby
    do
        message := "コピーできます。"
    destination
        S1
end;
T3 is
    source
        S1
    input
        Ms.Copy
    do
        message := "コピーしています。"
    destination
        S1
end;
T4 is
    source
        S1
    input
        Ms.Stop
    do
        message := "ストップします。"
    destination
        S1
end;
T5 is
```

```
    source
        S1
    input
        Ms.Reset
    do
        message := "リセットしています。"
    destination
        S1
end;
T6 is
    source
        S1
    input
        Ms.Empty
    do
        message := "用紙がありません。"
    destination
        S1
end;
T7 is
    source
        S1
    input
        Ms.Range
    do
        message := "部数の範囲は 0〜99 です。"
    destination
        S1
end;
T8 is
    source
        S1
    input
        Ms.Restart
    do
        message := "スタートを押してください。"
    destination
        S1
end;
```

```
    ------------------------------------------
    -- for debug -----
    ------------------------------------------

    T91 is
      source
          S1
      input
3230      Ms.Nak
      do
          message := "NAK 応答です。";
          nak_code := Ms.Nak.info
      destination
          S1
    end;
    T92 is
      source
          S1
3240  input
          Ms.Timeout
      do
          message := "通信タイムアウトです。"
      destination
          S1
    end;
    T93 is
      source
          S1
3250  input
          Ms.IllState
      do
          message := "予期せぬ動作状態です。"
      destination
          S1
    end
  end -- Message_LCD
  ------------------------------------------------------------
  #define PSIZE_A4     0x24
3260 #define PSIZE_B4     0x14
  #define PSIZE_A4R    0x2C
```

```
#define PSIZE_A3      0x23
-------------------------------------------------------------
class
    PaperSelect_LCD
field
    Ms
state
    S1
attribute
    top : String;
    middle : String;
    bottom : String;
    current : String
transition
  Start is
    source
        init
    do
        top     := "---";
        middle  := "---";
        bottom  := "---";
        current := "Top    "
    destination
        S1
  end;
  T1 is
    source
        S1
    input
        Ms.Size
    do
        top     := if ( Ms.Size.top = PSIZE_A4 )
                   then "A4 "
                   else ( if ( Ms.Size.top = PSIZE_B4 )
                          then "B4 "
                          else ( if ( Ms.Size.top = PSIZE_A4R )
                                 then "A4R"
                                 else ( if ( Ms.Size.top = PSIZE_A3 )
                                        then "A3 "
                                        else "---" )));
```

```
        middle := if ( Ms.Size.middle = PSIZE_A4 )
                  then "A4 "
                  else ( if ( Ms.Size.middle = PSIZE_B4 )
                       then "B4 "
                       else ( if ( Ms.Size.middle = PSIZE_A4R )
                            then "A4R"
                            else ( if ( Ms.Size.middle = PSIZE_A3 )
                                 then "A3 "
                                 else "---" )));
        bottom := if ( Ms.Size.bottom = PSIZE_A4 )
                  then "A4 "
                  else ( if ( Ms.Size.bottom = PSIZE_B4 )
                       then "B4 "
                       else ( if ( Ms.Size.bottom = PSIZE_A4R )
                            then "A4R"
                            else ( if ( Ms.Size.bottom = PSIZE_A3 )
                                 then "A3 "
                                 else "---" )))
    destination
        S1
  end
  T2 is
    source
        S1
    input
        Ms.Select
    do
        current := if ( Ms.Select.current = 1 )
                   then "Top    "
                   else ( if ( Ms.Select.current = 2 )
                        then "Middle"
                        else ( if ( Ms.Select.current = 3 )
                             then "Bottom"
                             else "------" ))
    destination
        S1
  end
end -- PaperSelect_LCD
-----------------------------------------------------------------
class
```

```
        PaperCount_LCD
    field
        Ms
    state
        S1
    attribute
        count : Int
    transition
3350    Start is
          source
              init
          do
              count := 1
          destination
              S1
        end;
        T2 is
          source
3360          S1
          input
              Ms.Count
          do
              count := Ms.Count.counter
          destination
              S1
        end
    end -- PaperCount_LCD
    ------------------------------------------------------------
3370 class
        Jam_LCD
    field
        Ms
    state
        S1, S2
    attribute
        message : String;
        pos1 : Int;
        pos2 : Int;
3380    pos3 : Int
    transition
```

```
    Start is
      source
          init
      destination
          S1
    end;
    T1 is
      source
3390      S1
      input
          Ms.Jam
      do
          message := "用紙が詰まりました。";
          pos1 := Ms.Jam.pos1;
          pos2 := Ms.Jam.pos2;
          pos3 := Ms.Jam.pos3
      destination
          S2
3400  end
  end -- Jam_LCD
  ------------------------------------------------------------
  class
      LCD_Interface
  field
      St
  inner
      lcd_if_child : {
      mess_lcd : Message_LCD;
3410  psel_lcd : PaperSelect_LCD;
      pcnt_lcd : PaperCount_LCD;
      jam_lcd : Jam_LCD
      }
  transition
    Start is
      source
          init
      destination
          lcd_if_child
3420  end
  end -- LCD_Interface
```

```
-----------------------------------------------------------
class
    Operation_Panel
  inner
      panel_children : {
      copy_ctrl : Copy_Control;
      status_ctrl : Status_Control;
      timer_ctrl : Timer_Control;
      key_if : Key_Interface;
      lcd_if : LCD_Interface;
      comport_if : ComPort_Interface;
      cur_settings : Current_Settings;
      def_settings : Default_Settings
      }
  transition
  Start is
    source
        init
    destination
        panel_children
  end
end -- Operation_Panel
-----------------------------------------------------------
-----------------------------------------------------------
system
    Sys
  object
      op_panel : Operation_Panel
  transition
  Start is
    source
        init
    destination
        op_panel
  end
end -- Sys
-----------------------------------------------------------
```

第 F 章

複写機 CP-KT1 の操作部プログラムの ObML コード

本章では，付録 E に示した複写機 CP-KT1 の操作部の ObCL コードから生成した ObML コードを示す.

```
(* output type for SML/NJ version 110 *)
datatype Attribute_type =
        Int of int
    | String of string
    | Null_of_Attr;
(*
fun eqCommon (Int a) (Int b) = (a=b)
    | eqCommon (String a) (String b) = (a=b)
    | eqCommon Null_of_Attr Null_of_Attr = true;
fun ppAttrVal (Int n) = Int.toString n
    | ppAttrVal (String s) = "\"" ^ s ^ "\""
    | ppAttrVal Null_of_Attr = "Null";
*)
(* cat obts8-110.sml *)
(* ObTS Simulator for SML/NJ 109.* *)

(* System.Print.printDepth := 10; *)

use "typedef.sml";
use "attrtype-110.sml";
use "tools.sml";
```

```
use “main.sml”;
use “pp.sml”;
(*
  use "sample.sml";
  *)

(* Sys_op_panel_timer_ctrl_tm_400m_ctrl *)
fun Sys_op_panel_timer_ctrl_tm_400m_ctrl (Name _ _ _) = (Name []
        [(“Sys_op_panel_timer_ctrl_tm_400m_ctrl” [])])
    | Sys_op_panel_timer_ctrl_tm_400m_ctrl (Initial a e env) =
        (State “S_off” a [])
    | Sys_op_panel_timer_ctrl_tm_400m_ctrl (State “S_on” a e env) =
        if (memberEv “Tm.i400m_Stop” e) then
                (State “S_off” a [])
        else if (memberEv “Tm.i400m_interval” e) then
                (State “S_on” a [])
        else
                (Null [] [])
    | Sys_op_panel_timer_ctrl_tm_400m_ctrl (State “S_off” a e env) =
        if (memberEv “Tm.i400m_Start” e) then
                (State “S_on” a [])
        else
                (Null [] [])
    | Sys_op_panel_timer_ctrl_tm_400m_ctrl (_ _ _ _) = (Null [] []);
(* Sys_op_panel_timer_ctrl_tm_200m_ctrl *)
fun Sys_op_panel_timer_ctrl_tm_200m_ctrl (Name _ _ _) = (Name []
        [(“Sys_op_panel_timer_ctrl_tm_200m_ctrl” [])])
    | Sys_op_panel_timer_ctrl_tm_200m_ctrl (Initial a e env) =
        (State “S_off” a [])
    | Sys_op_panel_timer_ctrl_tm_200m_ctrl (State “S_on” a e env) =
        if (memberEv “Tm.i200m_Stop” e) then
                (State “S_off” a [])
        else if (memberEv “Tm.i200m_interval” e) then
                (State “S_on” a [])
        else
                (Null [] [])
    | Sys_op_panel_timer_ctrl_tm_200m_ctrl (State “S_off” a e env) =
        if (memberEv “Tm.i200m_Start” e) then
                (State “S_on” a [])
        else
```

```
                (Null [] [])
    | Sys_op_panel_timer_ctrl_tm_200m_ctrl (_ _ _ _) = (Null [] []);
(* Sys_op_panel_timer_ctrl_tm_60s_ctrl *)
fun Sys_op_panel_timer_ctrl_tm_60s_ctrl (Name _ _ _) = (Name []
        [("Sys_op_panel_timer_ctrl_tm_60s_ctrl" [])])
    | Sys_op_panel_timer_ctrl_tm_60s_ctrl (Initial a e env) =
        (State "S_off" a [])
    | Sys_op_panel_timer_ctrl_tm_60s_ctrl (State "S_on" a e env) =
        if (memberEv "Tm.o60s_Stop" e) then
                (State "S_off" a [])
        else if (memberEv "Tm.o60s_timeout" e) then
                (State "S_off" a [])
        else
                (Null [] [])
    | Sys_op_panel_timer_ctrl_tm_60s_ctrl (State "S_off" a e env) =
        if (memberEv "Tm.o60s_Start" e) then
                (State "S_on" a [])
        else
                (Null [] [])
    | Sys_op_panel_timer_ctrl_tm_60s_ctrl (_ _ _ _) = (Null [] []);
(* Sys_op_panel_timer_ctrl_tm_1s_ctrl *)
fun Sys_op_panel_timer_ctrl_tm_1s_ctrl (Name _ _ _) = (Name []
        [("Sys_op_panel_timer_ctrl_tm_1s_ctrl" [])])
    | Sys_op_panel_timer_ctrl_tm_1s_ctrl (Initial a e env) =
        (State "S_off" a [])
    | Sys_op_panel_timer_ctrl_tm_1s_ctrl (State "S_on" a e env) =
        if (memberEv "Tm.o1s_Stop" e) then
                (State "S_off" a [])
        else if (memberEv "Tm.o1s_timeout" e) then
                (State "S_off" a [])
        else
                (Null [] [])
    | Sys_op_panel_timer_ctrl_tm_1s_ctrl (State "S_off" a e env) =
        if (memberEv "Tm.o1s_Start" e) then
                (State "S_on" a [])
        else
                (Null [] [])
    | Sys_op_panel_timer_ctrl_tm_1s_ctrl (_ _ _ _) = (Null [] []);
(* Sys_op_panel_lcd_if_mess_lcd *)
fun Sys_op_panel_lcd_if_mess_lcd (Name _ _ _) = (Name []
```

```
        [("Sys_op_panel_lcd_if_mess_lcd" [])])
    | Sys_op_panel_lcd_if_mess_lcd (Initial a e env) =
        (State "S1" changeAttr "message" (String "しばらくおまちください。")
        (changeAttr "nak_code" (Int 0) a) [])
    | Sys_op_panel_lcd_if_mess_lcd (State "S1" a e env) =
        if (memberEv "Ms.Wait" e) then
                (State "S1" changeAttr "message" (String "しばらくおまちください。") a [])
        else if (memberEv "Ms.Standby" e) then
                (State "S1" changeAttr "message" (String "コピーできます。") a [])
        else if (memberEv "Ms.Copy" e) then
                (State "S1" changeAttr "message" (String "コピーしています。") a [])
        else if (memberEv "Ms.Stop" e) then
                (State "S1" changeAttr "message" (String "ストップします。") a [])
        else if (memberEv "Ms.Reset" e) then
                (State "S1" changeAttr "message" (String "リセットしています。") a [])
        else if (memberEv "Ms.Empty" e) then
                (State "S1" changeAttr "message" (String "用紙がありません。") a [])
        else if (memberEv "Ms.Range" e) then
                (State "S1" changeAttr "message" (String "部数の範囲は0〜99です。") a [])
        else if (memberEv "Ms.Restart" e) then
                (State "S1" changeAttr "message" (String "スタートを押してください。") a [])
        else if (memberEv "Ms.Nak" e) then
                (State "S1" changeAttr "message" (String "NAK 応答です。") (changeAttr
        "nak_code" ((eventAttr "Ms.Nak" "info" e)) a) [])
        else if (memberEv "Ms.Timeout" e) then
                (State "S1" changeAttr "message" (String "通信タイムアウトです。") a [])
        else if (memberEv "Ms.IllState" e) then
                (State "S1" changeAttr "message" (String "予期せぬ動作状態です。") a [])
        else
                (Null [] [])
    | Sys_op_panel_lcd_if_mess_lcd (_ _ _ _) = (Null [] []);
(* Sys_op_panel_lcd_if_psel_lcd *)
fun Sys_op_panel_lcd_if_psel_lcd (Name _ _ _) = (Name []
        [("Sys_op_panel_lcd_if_psel_lcd" [])])
    | Sys_op_panel_lcd_if_psel_lcd (Initial a e env) =
        (State "S1" changeAttr "top" (String "—") (changeAttr "middle" (String "—")
        (changeAttr "bottom" (String "—") (changeAttr "current" (String "Top ") a)))
        [])
    | Sys_op_panel_lcd_if_psel_lcd (State "S1" a e env) =
        if (memberEv "Ms.Size" e) then
```

```
                (State “S1” changeAttr “top” (if ((eventAttr “Ms.Size” “top” e)=Int 36) then
        String “A4 ” else (if ((eventAttr “Ms.Size” “top” e)=Int 20) then String “B4 ” else
        (if ((eventAttr “Ms.Size” “top” e)=Int 44) then String “A4R” else (if ((eventAttr
        “Ms.Size” “top” e)=Int 35) then String “A3 ” else String “—”)))) (changeAttr
        “middle” (if ((eventAttr “Ms.Size” “middle” e)=Int 36) then String “A4 ” else (if
        ((eventAttr “Ms.Size” “middle” e)=Int 20) then String “B4 ” else (if ((eventAttr
        “Ms.Size” “middle” e)=Int 44) then String “A4R” else (if ((eventAttr “Ms.Size”
        “middle” e)=Int 35) then String “A3 ” else String “—”)))) (changeAttr “bottom”
        (if ((eventAttr “Ms.Size” “bottom” e)=Int 36) then String “A4 ” else (if
        ((eventAttr “Ms.Size” “bottom” e)=Int 20) then String “B4 ” else (if ((eventAttr
        “Ms.Size” “bottom” e)=Int 44) then String “A4R” else (if ((eventAttr “Ms.Size”
        “bottom” e)=Int 35) then String “A3 ” else String “—”)))) a)) [])
        else if (memberEv “Ms.Select” e) then
                (State “S1” changeAttr “current” (if ((eventAttr “Ms.Select” “current” e)=Int
        1) then String “Top ” else (if ((eventAttr “Ms.Select” “current” e)=Int 2) then
        String “Middle” else (if ((eventAttr “Ms.Select” “current” e)=Int 3) then String
        “Bottom” else String “——”))) a [])
        else
                (Null [] [])
    | Sys_op_panel_lcd_if_psel_lcd (_ _ _ _) = (Null [] []);
(* Sys_op_panel_lcd_if_pcnt_lcd *)
fun Sys_op_panel_lcd_if_pcnt_lcd (Name _ _ _) = (Name []
        [(“Sys_op_panel_lcd_if_pcnt_lcd” [])])
    | Sys_op_panel_lcd_if_pcnt_lcd (Initial a e env) =
        (State “S1” changeAttr “count” (Int 1) a [])
    | Sys_op_panel_lcd_if_pcnt_lcd (State “S1” a e env) =
        if (memberEv “Ms.Count” e) then
                (State “S1” changeAttr “count” ((eventAttr “Ms.Count” “counter” e)) a [])
        else
                (Null [] [])
    | Sys_op_panel_lcd_if_pcnt_lcd (_ _ _ _) = (Null [] []);
(* Sys_op_panel_lcd_if_jam_lcd *)
fun Sys_op_panel_lcd_if_jam_lcd (Name _ _ _) = (Name []
        [(“Sys_op_panel_lcd_if_jam_lcd” [])])
    | Sys_op_panel_lcd_if_jam_lcd (Initial a e env) =
        (State “S1” a [])
    | Sys_op_panel_lcd_if_jam_lcd (State “S1” a e env) =
        if (memberEv “Ms.Jam” e) then
                (State “S2” changeAttr “message” (String “用紙が詰まりました。”) (changeAttr
        “pos1” ((eventAttr “Ms.Jam” “pos1” e)) (changeAttr “pos2” ((eventAttr “Ms.Jam”
```

```
        "pos2" e)) (changeAttr "pos3" ((eventAttr "Ms.Jam" "pos3" e)) a))) [])
        else
                (Null [] [])
    | Sys_op_panel_lcd_if_jam_lcd (_ _ _ _) = (Null [] []);
(* Sys_op_panel_copy_ctrl *)
fun Sys_op_panel_copy_ctrl (Name _ _ _) = (Name [] [("Sys_op_panel_copy_ctrl"
        [])])
    | Sys_op_panel_copy_ctrl (Initial a e env) =
        (State "S_ready" changeAttr "key_input_start_flag" (Int 0) a [])
    | Sys_op_panel_copy_ctrl (State "S_ready" a e env) =
        if (memberEv "Si.DefaultChange" e) then
                (State "S_ready" changeAttr "default_paper_count" ((eventAttr
        "Si.DefaultChange" "paper_count" e)) (changeAttr
        "default_paper_select_position" ((eventAttr "Si.DefaultChange"
        "paper_select_position" e)) (changeAttr "current_paper_count" ((eventAttr
        "Si.DefaultChange" "paper_count" e)) (changeAttr
        "current_paper_select_position" ((eventAttr "Si.DefaultChange"
        "paper_select_position" e)) a))) [])
        else if (memberEv "Si.CurrentChange" e) then
                (State "S_ready" changeAttr "current_paper_count" ((eventAttr
        "Si.CurrentChange" "paper_count" e)) (changeAttr
        "current_paper_select_position" ((eventAttr "Si.CurrentChange"
        "paper_select_position" e)) a) [])
        else if (memberEv "St.Standby" e) then
                (State "S_ready" a [("Tm.o60s_Start" []) ("Ms.Count" [("counter" (pickupAttr
        "current_paper_count" a))]) ("Ms.Select" [("current" (pickupAttr
        "current_paper_select_position" a))])])
        else if (memberEv "Key.Select" e) then
                (State "S_select" a [("Cm.Send" [("cid" Int 66) ("clen" Int 0)]) ("Tm.o60s_Stop"
        [])])
        else if (memberEv "Key.Reset" e) orelse (memberEv "Tm.o60s" e) then
                (State "S_reset" a [("Cm.Send" [("cid" Int 160) ("clen" Int 0)]) ("Tm.o60s_Stop"
        [])])
        else if (memberEv "Key.Start" e) then
                (State "S_start1" a [("Tm.o60s_Stop" []) ("Cm.Send" [("cid" Int 33) ("clen" Int
        2) ("param2" (pickupAttr "current_paper_count" a)) ("param3" (pickupAttr
        "current_paper_select_position" a))])])
        else
                (Null [] [])
    | Sys_op_panel_copy_ctrl (State "S_copy" a e env) =
```

```
        if (memberEv “St.Standby” e) then
                (State “S_ready” a [(“Tm.o60s_Start” []) (“Ms.Count” [(“counter” (pickupAttr
        “current_paper_count” a))]) (“Ms.Select” [(“current” (pickupAttr
        “current_paper_select_position” a))])])
        else if (memberEv “Key.Stop” e) then
                (State “S_stop1” a [(“Cm.Send” [(“cid” Int 50) (“clen” Int 0)])])
        else if (memberEv “St.Copy_Empty” e) then
                (State “S_empty1” a [])
        else
                (Null [] [])
    | Sys_op_panel_copy_ctrl (State “S_start1” a e env) =
        if (memberEv “Cm.Ack” e) then
                (State “S_start2” a [(“Cm.Send” [(“cid” Int 49) (“clen” Int 0)])])
        else
                (Null [] [])
    | Sys_op_panel_copy_ctrl (State “S_start2” a e env) =
        if (memberEv “Cm.Ack” e) then
                (State “S_copy” a [])
        else
                (Null [] [])
    | Sys_op_panel_copy_ctrl (State “S_stop1” a e env) =
        if (memberEv “Cm.Ack” e) then
                (State “S_stop2” a [])
        else
                (Null [] [])
    | Sys_op_panel_copy_ctrl (State “S_stop2” a e env) =
        if (memberEv “St.Standby” e) then
                (State “S_ready” a [(“Tm.o60s_Start” []) (“Ms.Count” [(“counter” (pickupAttr
        “current_paper_count” a))]) (“Ms.Select” [(“current” (pickupAttr
        “current_paper_select_position” a))])])
        else
                (Null [] [])
    | Sys_op_panel_copy_ctrl (State “S_empty1” a e env) =
        if (memberEv “Key.Stop” e) then
                (State “S_stop1” a [(“Cm.Send” [(“cid” Int 50) (“clen” Int 0)])])
        else if (memberEv “St.Restart” e) then
                (State “S_empty2” a [])
        else
                (Null [] [])
    | Sys_op_panel_copy_ctrl (State “S_empty2” a e env) =
```

```
        if (memberEv “St.Copy_Empty” e) then
                (State “S_empty1” a [])
        else if (memberEv “Key.Stop” e) then
                (State “S_stop1” a [(“Cm.Send” [(“cid” Int 50) (“clen” Int 0)])])
        else if (memberEv “Key.Start” e) then
                (State “S_copy” a [(“Cm.Send” [(“cid” Int 49) (“clen” Int 0)])])
        else
                (Null [] [])
    | Sys_op_panel_copy_ctrl (State “S_reset” a e env) =
        if (memberEv “Cm.Ack” e) then
                (State “S_reset” a [])
        else if (memberEv “St.Standby” e) then
                (State “S_ready” changeAttr “current_paper_count” ((pickupAttr
        “default_paper_count” a)) (changeAttr “current_paper_select_position”
        ((pickupAttr “default_paper_select_position” a)) a) [(“Tm.o60s_Start” [])
        (“Ms.Count” []) (“Ms.Select” [])])
        else
                (Null [] [])
    | Sys_op_panel_copy_ctrl (State “S_select” a e env) =
        if (memberEv “Cm.Ack” e) then
                (State “S_ready” changeAttr “current_paper_select_position” ((eventAttr
        “Cm.Ack” “param2” e)) a [(“Tm.o60s_Start” []) (“Ms.Count” [(“counter”
        (pickupAttr “current_paper_count” a))]) (“Ms.Select” [(“current” (eventAttr
        “Cm.Ack” “param2” e))])])
        else
                (Null [] [])
    | Sys_op_panel_copy_ctrl (_ _ _ _) = (Null [] []);
(* Sys_op_panel_status_ctrl *)
fun Sys_op_panel_status_ctrl (Name _ _ _) = (Name []
        [(“Sys_op_panel_status_ctrl” [])])
    | Sys_op_panel_status_ctrl (Initial a e env) =
        (State “S_ready” changeAttr “copy_status_old” (Int 0) (changeAttr
        “copy_sub_status_old” (Int 0) (changeAttr “tray_top_old” (Int 255)
        (changeAttr “tray_middle_old” (Int 255) (changeAttr “tray_bottom_old” (Int
        255) (changeAttr “default_paper_count” (Int 1) (changeAttr
        “default_paper_select_position” (Int 1) (changeAttr “current_paper_count”
        (Int 1) (changeAttr “current_paper_select_position” (Int 1) a)))))))
        [(“Tm.i400m_Start” [])])
    | Sys_op_panel_status_ctrl (State “S_ready” a e env) =
        if (memberEv “Tm.i200m” e) then
```

```
                (State “S_wait_copy_count” a [(“Cm.Send” [(“cid” Int 65) (“clen” Int 0)])])
        else if (memberEv “Si.DefaultChange” e) then
                (State “S_ready” changeAttr “default_paper_count” ((eventAttr
        “Si.DefaultChange” “paper_count” e)) (changeAttr
        “default_paper_select_position” ((eventAttr “Si.DefaultChange”
        “paper_select_position” e)) (changeAttr “current_paper_count” ((eventAttr
        “Si.DefaultChange” “paper_count” e)) (changeAttr
        “current_paper_select_position” ((eventAttr “Si.DefaultChange”
310     “paper_select_position” e)) a))) [])
        else if (memberEv “Si.CurrentChange” e) andalso ((eventAttr
        “Si.CurrentChange” “paper_count” e)=Int 0) then
                (State “S_ready” changeAttr “current_paper_count” ((eventAttr
        “Si.CurrentChange” “paper_count” e)) (changeAttr
        “current_paper_select_position” ((eventAttr “Si.CurrentChange”
        “paper_select_position” e)) a) [])
        else if (memberEv “Si.CurrentChange” e) andalso ((pickupAttr
        “copy_status_old” a)=Int 1 andalso (eventAttr “Si.CurrentChange”
        “paper_count” e)<>Int 0) then
320             (State “S_ready” changeAttr “current_paper_count” ((eventAttr
        “Si.CurrentChange” “paper_count” e)) (changeAttr
        “current_paper_select_position” ((eventAttr “Si.CurrentChange”
        “paper_select_position” e)) a) [])
        else if (memberEv “Si.CurrentChange” e) andalso ((pickupAttr
        “copy_status_old” a)=Int 2 andalso (eventAttr “Si.CurrentChange”
        “paper_count” e)<>Int 0) then
                (State “S_ready” changeAttr “current_paper_count” ((eventAttr
        “Si.CurrentChange” “paper_count” e)) (changeAttr
        “current_paper_select_position” ((eventAttr “Si.CurrentChange”
330     “paper_select_position” e)) a) [])
        else if (memberEv “Tm.i400m” e) then
                (State “S_wait_status” a [(“Cm.Send” [(“cid” Int 17) (“clen” Int 0)])])
        else
                (Null [] [])
    | Sys_op_panel_status_ctrl (State “S_wait_status” a e env) =
        if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1 andalso
        (pickupAttr “copy_status_old” a)<>Int 1) andalso ((pickupAttr
        “copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack” “param3” e)=Int 128)
        andalso ((eventAttr “Cm.Ack” “param4” e)<>(pickupAttr “tray_top_old” a)
340     orelse (eventAttr “Cm.Ack” “param5” e)<>(pickupAttr “tray_middle_old” a)
        orelse (eventAttr “Cm.Ack” “param6” e)<>(pickupAttr “tray_bottom_old” a)))
```

```
then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“St.Wait” [])
(“Ms.Empty” []) (“Ms.Size” [(“top” (eventAttr “Cm.Ack” “param4” e)) (“middle”
(eventAttr “Cm.Ack” “param5” e)) (“bottom” (eventAttr “Cm.Ack” “param6”
e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1
andalso (pickupAttr “copy_status_old” a)<>Int 1) andalso ((pickupAttr
“copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack” “param3” e)<>Int
128) andalso ((pickupAttr “current_paper_count” a)=Int 0) andalso ((eventAttr
“Cm.Ack” “param4” e)<>(pickupAttr “tray_top_old” a) orelse (eventAttr
“Cm.Ack” “param5” e)<>(pickupAttr “tray_middle_old” a) orelse (eventAttr
“Cm.Ack” “param6” e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“St.Wait” [])
(“Ms.Range” []) (“Ms.Size” [(“top” (eventAttr “Cm.Ack” “param4” e)) (“middle”
(eventAttr “Cm.Ack” “param5” e)) (“bottom” (eventAttr “Cm.Ack” “param6”
e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1
andalso (pickupAttr “copy_status_old” a)<>Int 1) andalso ((pickupAttr
“copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack” “param3” e)<>Int
128) andalso ((pickupAttr “current_paper_count” a)<>Int 0) andalso
((eventAttr “Cm.Ack” “param4” e)<>(pickupAttr “tray_top_old” a) orelse
(eventAttr “Cm.Ack” “param5” e)<>(pickupAttr “tray_middle_old” a) orelse
(eventAttr “Cm.Ack” “param6” e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“St.Wait” [])
(“Ms.Size” [(“top” (eventAttr “Cm.Ack” “param4” e)) (“middle” (eventAttr
“Cm.Ack” “param5” e)) (“bottom” (eventAttr “Cm.Ack” “param6” e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1
andalso (pickupAttr “copy_status_old” a)=Int 4) andalso ((eventAttr “Cm.Ack”
```

```
"param3" e)=Int 128) andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr
"tray_top_old" a) orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr
"tray_middle_old" a) orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr
"tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("St.Wait" [])
("Ms.Empty" []) ("Tm.i200m_Stop" []) ("Ms.Size" [("top" (eventAttr "Cm.Ack"
"param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e)) ("bottom" (eventAttr
"Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 1
andalso (pickupAttr "copy_status_old" a)=Int 4) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128) andalso ((pickupAttr "current_paper_count" a)=Int 0)
andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a)
orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a)
orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("St.Wait" [])
("Ms.Range" []) ("Tm.i200m_Stop" []) ("Ms.Size" [("top" (eventAttr "Cm.Ack"
"param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e)) ("bottom" (eventAttr
"Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 1
andalso (pickupAttr "copy_status_old" a)=Int 4) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128) andalso ((pickupAttr "current_paper_count" a)<>Int 0)
andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a)
orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a)
orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("St.Wait" [])
("Ms.Wait" []) ("Tm.i200m_Stop" []) ("Ms.Size" [("top" (eventAttr "Cm.Ack"
```

```
"param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e)) ("bottom" (eventAttr
"Cm.Ack" "param6" e))])]
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 1
andalso (pickupAttr "copy_status_old" a)=Int 1) andalso ((eventAttr "Cm.Ack"
"param3" e)=Int 128 andalso (pickupAttr "copy_sub_status_old" a)<>Int 128)
andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a)
orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a)
orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("Ms.Empty" [])
("Ms.Size" [("top" (eventAttr "Cm.Ack" "param4" e)) ("middle" (eventAttr
"Cm.Ack" "param5" e)) ("bottom" (eventAttr "Cm.Ack" "param6" e))])]
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 1
andalso (pickupAttr "copy_status_old" a)=Int 1) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128 andalso (pickupAttr "copy_sub_status_old" a)=Int 128)
andalso ((pickupAttr "current_paper_count" a)=Int 0) andalso ((eventAttr
"Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a) orelse (eventAttr
"Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a) orelse (eventAttr
"Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("Ms.Range" [])
("Ms.Size" [("top" (eventAttr "Cm.Ack" "param4" e)) ("middle" (eventAttr
"Cm.Ack" "param5" e)) ("bottom" (eventAttr "Cm.Ack" "param6" e))])]
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 1
andalso (pickupAttr "copy_status_old" a)=Int 1) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128 andalso (pickupAttr "copy_sub_status_old" a)=Int 128)
andalso ((pickupAttr "current_paper_count" a)<>Int 0) andalso ((eventAttr
"Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a) orelse (eventAttr
"Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a) orelse (eventAttr
"Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
```

```
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("Ms.Wait" [])
("Ms.Size" [("top" (eventAttr "Cm.Ack" "param4" e)) ("middle" (eventAttr
"Cm.Ack" "param5" e)) ("bottom" (eventAttr "Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 1
andalso (pickupAttr "copy_status_old" a)=Int 1) andalso ((eventAttr "Cm.Ack"
"param3" e)=Int 128 andalso (pickupAttr "copy_sub_status_old" a)=Int 128)
andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a)
orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a)
orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("Ms.Size" [("top"
(eventAttr "Cm.Ack" "param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e))
("bottom" (eventAttr "Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 1
andalso (pickupAttr "copy_status_old" a)=Int 1) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128 andalso (pickupAttr "copy_sub_status_old" a)<>Int 128)
andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a)
orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a)
orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("Ms.Size" [("top"
(eventAttr "Cm.Ack" "param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e))
("bottom" (eventAttr "Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)<>Int 2) andalso ((pickupAttr
"copy_status_old" a)<>Int 4) andalso ((eventAttr "Cm.Ack" "param3" e)=Int 128)
andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a)
orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a)
orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
```

```
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“St.Standby” [])
(“Ms.Empty” []) (“Ms.Size” [(“top” (eventAttr “Cm.Ack” “param4” e)) (“middle”
(eventAttr “Cm.Ack” “param5” e)) (“bottom” (eventAttr “Cm.Ack” “param6”
e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 2
andalso (pickupAttr “copy_status_old” a)<>Int 2) andalso ((pickupAttr
“copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack” “param3” e)<>Int
128) andalso ((pickupAttr “current_paper_count” a)=Int 0) andalso ((eventAttr
“Cm.Ack” “param4” e)<>(pickupAttr “tray_top_old” a) orelse (eventAttr
“Cm.Ack” “param5” e)<>(pickupAttr “tray_middle_old” a) orelse (eventAttr
“Cm.Ack” “param6” e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“St.Standby” [])
(“Ms.Range” []) (“Ms.Size” [(“top” (eventAttr “Cm.Ack” “param4” e)) (“middle”
(eventAttr “Cm.Ack” “param5” e)) (“bottom” (eventAttr “Cm.Ack” “param6”
e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 2
andalso (pickupAttr “copy_status_old” a)<>Int 2) andalso ((pickupAttr
“copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack” “param3” e)<>Int
128) andalso ((pickupAttr “current_paper_count” a)<>Int 0) andalso
((eventAttr “Cm.Ack” “param4” e)<>(pickupAttr “tray_top_old” a) orelse
(eventAttr “Cm.Ack” “param5” e)<>(pickupAttr “tray_middle_old” a) orelse
(eventAttr “Cm.Ack” “param6” e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“St.Standby” [])
(“Ms.Standby” []) (“Ms.Size” [(“top” (eventAttr “Cm.Ack” “param4” e))
(“middle” (eventAttr “Cm.Ack” “param5” e)) (“bottom” (eventAttr “Cm.Ack”
“param6” e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 2
andalso (pickupAttr “copy_status_old” a)=Int 4) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 128) andalso ((eventAttr “Cm.Ack” “param4” e)<>(pickupAttr
```

```
"tray_top_old" a) orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr
"tray_middle_old" a) orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr
"tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("St.Standby" [])
("Ms.Empty" []) ("Tm.i200m_Stop" []) ("Ms.Size" [("top" (eventAttr "Cm.Ack"
"param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e)) ("bottom" (eventAttr
"Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)=Int 4) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128) andalso ((pickupAttr "current_paper_count" a)=Int 0)
andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a)
orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a)
orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("St.Standby" [])
("Ms.Range" []) ("Tm.i200m_Stop" []) ("Ms.Size" [("top" (eventAttr "Cm.Ack"
"param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e)) ("bottom" (eventAttr
"Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)=Int 4) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128) andalso ((pickupAttr "current_paper_count" a)<>Int 0)
andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a)
orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a)
orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("St.Standby" [])
("Ms.Standby" []) ("Tm.i200m_Stop" []) ("Ms.Size" [("top" (eventAttr "Cm.Ack"
"param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e)) ("bottom" (eventAttr
```

```
"Cm.Ack" "param6" e))]])]
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)=Int 2) andalso ((eventAttr "Cm.Ack"
"param3" e)=Int 128 andalso (pickupAttr "copy_sub_status_old" a)<>Int 128)
andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a)
orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a)
orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("Ms.Empty" [])
("Ms.Size" [("top" (eventAttr "Cm.Ack" "param4" e)) ("middle" (eventAttr
"Cm.Ack" "param5" e)) ("bottom" (eventAttr "Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)=Int 2) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128 andalso (pickupAttr "copy_sub_status_old" a)=Int 128)
andalso ((pickupAttr "current_paper_count" a)=Int 0) andalso ((eventAttr
"Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a) orelse (eventAttr
"Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a) orelse (eventAttr
"Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("Ms.Range" [])
("Ms.Size" [("top" (eventAttr "Cm.Ack" "param4" e)) ("middle" (eventAttr
"Cm.Ack" "param5" e)) ("bottom" (eventAttr "Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)=Int 2) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128 andalso (pickupAttr "copy_sub_status_old" a)=Int 128)
andalso ((pickupAttr "current_paper_count" a)<>Int 0) andalso ((eventAttr
"Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a) orelse (eventAttr
"Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a) orelse (eventAttr
"Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
```

```
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("Ms.Standby" [])
("Ms.Size" [("top" (eventAttr "Cm.Ack" "param4" e)) ("middle" (eventAttr
"Cm.Ack" "param5" e)) ("bottom" (eventAttr "Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)=Int 2) andalso ((eventAttr "Cm.Ack"
"param3" e)=Int 128 andalso (pickupAttr "copy_sub_status_old" a)=Int 128)
andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a)
orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a)
orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("Ms.Size" [("top"
(eventAttr "Cm.Ack" "param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e))
("bottom" (eventAttr "Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)=Int 2) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128 andalso (pickupAttr "copy_sub_status_old" a)<>Int 128)
andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a)
orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a)
orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("Ms.Size" [("top"
(eventAttr "Cm.Ack" "param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e))
("bottom" (eventAttr "Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 4
andalso (pickupAttr "copy_status_old" a)<>Int 4) andalso ((eventAttr "Cm.Ack"
"param3" e)=Int 7) andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr
"tray_top_old" a) orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr
"tray_middle_old" a) orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr
"tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
```

```
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("St.Copy" [])
("Tm.i200m_Start" []) ("Ms.Copy" []) ("Ms.Size" [("top" (eventAttr "Cm.Ack"
"param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e)) ("bottom" (eventAttr
"Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 4
andalso (pickupAttr "copy_status_old" a)<>Int 4) andalso ((eventAttr "Cm.Ack"
"param3" e)=Int 8) andalso ((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr
"tray_top_old" a) orelse (eventAttr "Cm.Ack" "param5" e)<>(pickupAttr
"tray_middle_old" a) orelse (eventAttr "Cm.Ack" "param6" e)<>(pickupAttr
"tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("St.Copy" [])
("Tm.i200m_Start" []) ("Ms.Stop" []) ("Ms.Size" [("top" (eventAttr "Cm.Ack"
"param4" e)) ("middle" (eventAttr "Cm.Ack" "param5" e)) ("bottom" (eventAttr
"Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 4
andalso (pickupAttr "copy_status_old" a)=Int 4) andalso ((eventAttr "Cm.Ack"
"param3" e)=Int 7 andalso (pickupAttr "copy_sub_status_old" a)<>Int 7) andalso
((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a) orelse
(eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a) orelse
(eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("Ms.Copy" [])
("Ms.Size" [("top" (eventAttr "Cm.Ack" "param4" e)) ("middle" (eventAttr
"Cm.Ack" "param5" e)) ("bottom" (eventAttr "Cm.Ack" "param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 4
andalso (pickupAttr "copy_status_old" a)=Int 4) andalso ((eventAttr "Cm.Ack"
"param3" e)=Int 8 andalso (pickupAttr "copy_sub_status_old" a)<>Int 8) andalso
((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a) orelse
(eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a) orelse
(eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
```

```
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“Ms.Stop” [])
(“Ms.Size” [(“top” (eventAttr “Cm.Ack” “param4” e)) (“middle” (eventAttr
“Cm.Ack” “param5” e)) (“bottom” (eventAttr “Cm.Ack” “param6” e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 8
andalso (pickupAttr “copy_status_old” a)<>Int 8) andalso ((pickupAttr
“copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack” “param4”
e)<>(pickupAttr “tray_top_old” a) orelse (eventAttr “Cm.Ack” “param5”
e)<>(pickupAttr “tray_middle_old” a) orelse (eventAttr “Cm.Ack” “param6”
e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Jam” []) (“Ms.Jam” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 8
andalso (pickupAttr “copy_status_old” a)=Int 4) andalso ((eventAttr “Cm.Ack”
“param4” e)<>(pickupAttr “tray_top_old” a) orelse (eventAttr “Cm.Ack”
“param5” e)<>(pickupAttr “tray_middle_old” a) orelse (eventAttr “Cm.Ack”
“param6” e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Jam” []) (“Ms.Jam” []) (“Tm.i200m_Stop” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 8
andalso (pickupAttr “copy_status_old” a)=Int 8) andalso ((eventAttr “Cm.Ack”
“param4” e)<>(pickupAttr “tray_top_old” a) orelse (eventAttr “Cm.Ack”
“param5” e)<>(pickupAttr “tray_middle_old” a) orelse (eventAttr “Cm.Ack”
“param6” e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a) [])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 20
andalso (pickupAttr “copy_status_old” a)<>Int 20) andalso ((pickupAttr
“copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack” “param4”
e)<>(pickupAttr “tray_top_old” a) orelse (eventAttr “Cm.Ack” “param5”
e)<>(pickupAttr “tray_middle_old” a) orelse (eventAttr “Cm.Ack” “param6”
e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“St.Copy_Empty”
```

```
[]) (“Ms.Empty” []) (“Ms.Size” [(“top” (eventAttr “Cm.Ack” “param4” e))
(“middle” (eventAttr “Cm.Ack” “param5” e)) (“bottom” (eventAttr “Cm.Ack”
“param6” e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 20
andalso (pickupAttr “copy_status_old” a)=Int 4) andalso ((eventAttr “Cm.Ack”
“param4” e)<>(pickupAttr “tray_top_old” a) orelse (eventAttr “Cm.Ack”
“param5” e)<>(pickupAttr “tray_middle_old” a) orelse (eventAttr “Cm.Ack”
“param6” e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“St.Copy_Empty”
[]) (“Ms.Empty” []) (“Tm.i200m_Stop” []) (“Ms.Size” [(“top” (eventAttr
“Cm.Ack” “param4” e)) (“middle” (eventAttr “Cm.Ack” “param5” e)) (“bottom”
(eventAttr “Cm.Ack” “param6” e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 20
andalso (pickupAttr “copy_status_old” a)=Int 20) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 128 andalso (pickupAttr “copy_status_old” a)=Int 128) andalso
((eventAttr “Cm.Ack” “param4” e)<>(pickupAttr “tray_top_old” a) orelse
(eventAttr “Cm.Ack” “param5” e)<>(pickupAttr “tray_middle_old” a) orelse
(eventAttr “Cm.Ack” “param6” e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“Ms.Size” [(“top”
(eventAttr “Cm.Ack” “param4” e)) (“middle” (eventAttr “Cm.Ack” “param5” e))
(“bottom” (eventAttr “Cm.Ack” “param6” e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 20
andalso (pickupAttr “copy_status_old” a)=Int 20) andalso ((eventAttr “Cm.Ack”
“param3” e)<>Int 128 andalso (pickupAttr “copy_status_old” a)=Int 128) andalso
((eventAttr “Cm.Ack” “param4” e)<>(pickupAttr “tray_top_old” a) orelse
(eventAttr “Cm.Ack” “param5” e)<>(pickupAttr “tray_middle_old” a) orelse
(eventAttr “Cm.Ack” “param6” e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“St.Restart” [])
```

```
("Ms.Restart" []) ("Ms.Size" [("top" (eventAttr "Cm.Ack" "param4" e))
("middle" (eventAttr "Cm.Ack" "param5" e)) ("bottom" (eventAttr "Cm.Ack"
"param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 20
andalso (pickupAttr "copy_status_old" a)=Int 20) andalso ((eventAttr "Cm.Ack"
"param3" e)=Int 128 andalso (pickupAttr "copy_status_old" a)<>Int 128) andalso
((eventAttr "Cm.Ack" "param4" e)<>(pickupAttr "tray_top_old" a) orelse
(eventAttr "Cm.Ack" "param5" e)<>(pickupAttr "tray_middle_old" a) orelse
(eventAttr "Cm.Ack" "param6" e)<>(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e))
(changeAttr "tray_top_old" ((eventAttr "Cm.Ack" "param4" e)) (changeAttr
"tray_middle_old" ((eventAttr "Cm.Ack" "param5" e)) (changeAttr
"tray_bottom_old" ((eventAttr "Cm.Ack" "param6" e)) a)))) [("St.Copy_Empty"
[]) ("Ms.Empty" []) ("Ms.Size" [("top" (eventAttr "Cm.Ack" "param4" e))
("middle" (eventAttr "Cm.Ack" "param5" e)) ("bottom" (eventAttr "Cm.Ack"
"param6" e))])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 1
andalso (pickupAttr "copy_status_old" a)<>Int 1) andalso ((pickupAttr
"copy_status_old" a)<>Int 4) andalso ((eventAttr "Cm.Ack" "param3" e)=Int 128)
andalso ((eventAttr "Cm.Ack" "param4" e)=(pickupAttr "tray_top_old" a)
andalso (eventAttr "Cm.Ack" "param5" e)=(pickupAttr "tray_middle_old" a)
andalso (eventAttr "Cm.Ack" "param6" e)=(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a)
[("St.Wait" []) ("Ms.Empty" [])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 1
andalso (pickupAttr "copy_status_old" a)<>Int 1) andalso ((pickupAttr
"copy_status_old" a)<>Int 4) andalso ((eventAttr "Cm.Ack" "param3" e)<>Int
128) andalso ((pickupAttr "current_paper_count" a)=Int 0) andalso ((eventAttr
"Cm.Ack" "param4" e)=(pickupAttr "tray_top_old" a) andalso (eventAttr
"Cm.Ack" "param5" e)=(pickupAttr "tray_middle_old" a) andalso (eventAttr
"Cm.Ack" "param6" e)=(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a)
[("St.Wait" []) ("Ms.Range" [])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 1
andalso (pickupAttr "copy_status_old" a)<>Int 1) andalso ((pickupAttr
"copy_status_old" a)<>Int 4) andalso ((eventAttr "Cm.Ack" "param3" e)<>Int
```

```
128) andalso ((pickupAttr “current_paper_count” a)<>Int 0) andalso
((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a) andalso
(eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a) andalso
(eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Wait” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1
andalso (pickupAttr “copy_status_old” a)=Int 4) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 128) andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr
“tray_top_old” a) andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr
“tray_middle_old” a) andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr
“tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Wait” []) (“Ms.Empty” []) (“Tm.i200m_Stop” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1
andalso (pickupAttr “copy_status_old” a)=Int 4) andalso ((eventAttr “Cm.Ack”
“param3” e)<>Int 128) andalso ((pickupAttr “current_paper_count” a)=Int 0)
andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a)
andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a)
andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a)))
then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Wait” []) (“Ms.Range” []) (“Tm.i200m_Stop” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1
andalso (pickupAttr “copy_status_old” a)=Int 4) andalso ((eventAttr “Cm.Ack”
“param3” e)<>Int 128) andalso ((pickupAttr “current_paper_count” a)<>Int 0)
andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a)
andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a)
andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a)))
then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Wait” []) (“Ms.Wait” []) (“Tm.i200m_Stop” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1
andalso (pickupAttr “copy_status_old” a)=Int 1) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 128 andalso (pickupAttr “copy_sub_status_old” a)<>Int 128)
andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a)
```

```
andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a)
andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a)))
then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“Ms.Empty” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1
andalso (pickupAttr “copy_status_old” a)=Int 1) andalso ((eventAttr “Cm.Ack”
“param3” e)<>Int 128 andalso (pickupAttr “copy_sub_status_old” a)=Int 128)
andalso ((pickupAttr “current_paper_count” a)=Int 0) andalso ((eventAttr
“Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a) andalso (eventAttr
“Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a) andalso (eventAttr
“Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“Ms.Range” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1
andalso (pickupAttr “copy_status_old” a)=Int 1) andalso ((eventAttr “Cm.Ack”
“param3” e)<>Int 128 andalso (pickupAttr “copy_sub_status_old” a)=Int 128)
andalso ((pickupAttr “current_paper_count” a)<>Int 0) andalso ((eventAttr
“Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a) andalso (eventAttr
“Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a) andalso (eventAttr
“Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“Ms.Wait” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1
andalso (pickupAttr “copy_status_old” a)=Int 1) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 128 andalso (pickupAttr “copy_sub_status_old” a)=Int 128)
andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a)
andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a)
andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a)))
then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a) [])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 1
andalso (pickupAttr “copy_status_old” a)=Int 1) andalso ((eventAttr “Cm.Ack”
“param3” e)<>Int 128 andalso (pickupAttr “copy_sub_status_old” a)<>Int 128)
andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a)
andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a)
```

```
andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a)))
then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a) [])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 2
andalso (pickupAttr “copy_status_old” a)<>Int 2) andalso ((pickupAttr
“copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack” “param3” e)=Int 128)
andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a)
andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a)
andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a)))
then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Standby” []) (“Ms.Empty” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 2
andalso (pickupAttr “copy_status_old” a)<>Int 2) andalso ((pickupAttr
“copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack” “param3” e)<>Int
128) andalso ((pickupAttr “current_paper_count” a)=Int 0) andalso ((eventAttr
“Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a) andalso (eventAttr
“Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a) andalso (eventAttr
“Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Standby” []) (“Ms.Range” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 2
andalso (pickupAttr “copy_status_old” a)<>Int 2) andalso ((pickupAttr
“copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack” “param3” e)<>Int
128) andalso ((pickupAttr “current_paper_count” a)<>Int 0) andalso
((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a) andalso
(eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a) andalso
(eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Standby” []) (“Ms.Standby” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 2
andalso (pickupAttr “copy_status_old” a)=Int 4) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 128) andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr
“tray_top_old” a) andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr
“tray_middle_old” a) andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr
“tray_bottom_old” a))) then
```

```
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a)
[("St.Standby" []) ("Ms.Empty" []) ("Tm.i200m_Stop" [])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)=Int 4) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128) andalso ((pickupAttr "current_paper_count" a)=Int 0)
andalso ((eventAttr "Cm.Ack" "param4" e)=(pickupAttr "tray_top_old" a)
andalso (eventAttr "Cm.Ack" "param5" e)=(pickupAttr "tray_middle_old" a)
andalso (eventAttr "Cm.Ack" "param6" e)=(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a)
[("St.Standby" []) ("Ms.Range" []) ("Tm.i200m_Stop" [])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)=Int 4) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128) andalso ((pickupAttr "current_paper_count" a)<>Int 0)
andalso ((eventAttr "Cm.Ack" "param4" e)=(pickupAttr "tray_top_old" a)
andalso (eventAttr "Cm.Ack" "param5" e)=(pickupAttr "tray_middle_old" a)
andalso (eventAttr "Cm.Ack" "param6" e)=(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a)
[("St.Standby" []) ("Ms.Standby" []) ("Tm.i200m_Stop" [])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)=Int 2) andalso ((eventAttr "Cm.Ack"
"param3" e)=Int 128 andalso (pickupAttr "copy_sub_status_old" a)<>Int 128)
andalso ((eventAttr "Cm.Ack" "param4" e)=(pickupAttr "tray_top_old" a)
andalso (eventAttr "Cm.Ack" "param5" e)=(pickupAttr "tray_middle_old" a)
andalso (eventAttr "Cm.Ack" "param6" e)=(pickupAttr "tray_bottom_old" a)))
then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a)
[("Ms.Empty" [])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 2
andalso (pickupAttr "copy_status_old" a)=Int 2) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128 andalso (pickupAttr "copy_sub_status_old" a)=Int 128)
andalso ((pickupAttr "current_paper_count" a)=Int 0) andalso ((eventAttr
"Cm.Ack" "param4" e)=(pickupAttr "tray_top_old" a) andalso (eventAttr
"Cm.Ack" "param5" e)=(pickupAttr "tray_middle_old" a) andalso (eventAttr
"Cm.Ack" "param6" e)=(pickupAttr "tray_bottom_old" a))) then
```

```
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“Ms.Range” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 2
andalso (pickupAttr “copy_status_old” a)=Int 2) andalso ((eventAttr “Cm.Ack”
“param3” e)<>Int 128 andalso (pickupAttr “copy_sub_status_old” a)=Int 128)
andalso ((pickupAttr “current_paper_count” a)<>Int 0) andalso ((eventAttr
“Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a) andalso (eventAttr
“Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a) andalso (eventAttr
“Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“Ms.Standby” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 2
andalso (pickupAttr “copy_status_old” a)=Int 2) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 128 andalso (pickupAttr “copy_sub_status_old” a)=Int 128)
andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a)
andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a)
andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a)))
then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a) [])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 2
andalso (pickupAttr “copy_status_old” a)=Int 2) andalso ((eventAttr “Cm.Ack”
“param3” e)<>Int 128 andalso (pickupAttr “copy_sub_status_old” a)<>Int 128)
andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a)
andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a)
andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a)))
then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a) [])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 4
andalso (pickupAttr “copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 7) andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr
“tray_top_old” a) andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr
“tray_middle_old” a) andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr
“tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Copy” []) (“Tm.i200m_Start” []) (“Ms.Copy” [])])
```

```
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 4
andalso (pickupAttr “copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 8) andalso ((eventAttr “Cm.Ack” “param4” e)=(pickupAttr
“tray_top_old” a) andalso (eventAttr “Cm.Ack” “param5” e)=(pickupAttr
“tray_middle_old” a) andalso (eventAttr “Cm.Ack” “param6” e)=(pickupAttr
“tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Copy” []) (“Tm.i200m_Start” []) (“Ms.Stop” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 4
andalso (pickupAttr “copy_status_old” a)=Int 4) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 7 andalso (pickupAttr “copy_sub_status_old” a)<>Int 7) andalso
((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a) andalso
(eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a) andalso
(eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“Ms.Copy” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 4
andalso (pickupAttr “copy_status_old” a)=Int 4) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 8 andalso (pickupAttr “copy_sub_status_old” a)<>Int 8) andalso
((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a) andalso
(eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a) andalso
(eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“Ms.Stop” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 8
andalso (pickupAttr “copy_status_old” a)<>Int 8) andalso ((pickupAttr
“copy_status_old” a)<>Int 4) andalso ((eventAttr “Cm.Ack” “param4”
e)=(pickupAttr “tray_top_old” a) andalso (eventAttr “Cm.Ack” “param5”
e)=(pickupAttr “tray_middle_old” a) andalso (eventAttr “Cm.Ack” “param6”
e)=(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Jam” []) (“Ms.Jam” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 8
andalso (pickupAttr “copy_status_old” a)=Int 4) andalso ((eventAttr “Cm.Ack”
“param4” e)=(pickupAttr “tray_top_old” a) andalso (eventAttr “Cm.Ack”
“param5” e)=(pickupAttr “tray_middle_old” a) andalso (eventAttr “Cm.Ack”
```

```
"param6" e)=(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a)
[("St.Jam" []) ("Ms.Jam" []) ("Tm.i200m_Stop" [])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 8
andalso (pickupAttr "copy_status_old" a)=Int 8) andalso ((eventAttr "Cm.Ack"
"param4" e)=(pickupAttr "tray_top_old" a) andalso (eventAttr "Cm.Ack"
"param5" e)=(pickupAttr "tray_middle_old" a) andalso (eventAttr "Cm.Ack"
"param6" e)=(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a) [])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 20
andalso (pickupAttr "copy_status_old" a)<>Int 20) andalso ((pickupAttr
"copy_status_old" a)<>Int 4) andalso ((eventAttr "Cm.Ack" "param4"
e)=(pickupAttr "tray_top_old" a) andalso (eventAttr "Cm.Ack" "param5"
e)=(pickupAttr "tray_middle_old" a) andalso (eventAttr "Cm.Ack" "param6"
e)=(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a)
[("St.Copy_Empty" []) ("Ms.Empty" [])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 20
andalso (pickupAttr "copy_status_old" a)=Int 4) andalso ((eventAttr "Cm.Ack"
"param4" e)=(pickupAttr "tray_top_old" a) andalso (eventAttr "Cm.Ack"
"param5" e)=(pickupAttr "tray_middle_old" a) andalso (eventAttr "Cm.Ack"
"param6" e)=(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a)
[("St.Copy_Empty" []) ("Ms.Empty" []) ("Tm.i200m_Stop" [])])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 20
andalso (pickupAttr "copy_status_old" a)=Int 20) andalso ((eventAttr "Cm.Ack"
"param3" e)=Int 128 andalso (pickupAttr "copy_status_old" a)=Int 128) andalso
((eventAttr "Cm.Ack" "param4" e)=(pickupAttr "tray_top_old" a) andalso
(eventAttr "Cm.Ack" "param5" e)=(pickupAttr "tray_middle_old" a) andalso
(eventAttr "Cm.Ack" "param6" e)=(pickupAttr "tray_bottom_old" a))) then
        (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a) [])
else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 20
andalso (pickupAttr "copy_status_old" a)=Int 20) andalso ((eventAttr "Cm.Ack"
"param3" e)<>Int 128 andalso (pickupAttr "copy_status_old" a)=Int 128) andalso
((eventAttr "Cm.Ack" "param4" e)=(pickupAttr "tray_top_old" a) andalso
```

```
(eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a) andalso
(eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Restart” []) (“Ms.Restart” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 20
andalso (pickupAttr “copy_status_old” a)=Int 20) andalso ((eventAttr “Cm.Ack”
“param3” e)=Int 128 andalso (pickupAttr “copy_status_old” a)<>Int 128) andalso
((eventAttr “Cm.Ack” “param4” e)=(pickupAttr “tray_top_old” a) andalso
(eventAttr “Cm.Ack” “param5” e)=(pickupAttr “tray_middle_old” a) andalso
(eventAttr “Cm.Ack” “param6” e)=(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e)) a)
[(“St.Copy_Empty” []) (“Ms.Empty” [])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 255
andalso (pickupAttr “copy_status_old” a)<>Int 255) andalso ((eventAttr
“Cm.Ack” “param4” e)<>(pickupAttr “tray_top_old” a) orelse (eventAttr
“Cm.Ack” “param5” e)<>(pickupAttr “tray_middle_old” a) orelse (eventAttr
“Cm.Ack” “param6” e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“St.Reset” [])
(“Ms.Reset” []) (“Ms.Size” [(“top” (eventAttr “Cm.Ack” “param4” e)) (“middle”
(eventAttr “Cm.Ack” “param5” e)) (“bottom” (eventAttr “Cm.Ack” “param6”
e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 255
andalso (pickupAttr “copy_status_old” a)=Int 255) andalso ((eventAttr
“Cm.Ack” “param4” e)<>(pickupAttr “tray_top_old” a) orelse (eventAttr
“Cm.Ack” “param5” e)<>(pickupAttr “tray_middle_old” a) orelse (eventAttr
“Cm.Ack” “param6” e)<>(pickupAttr “tray_bottom_old” a))) then
        (State “S_ready” changeAttr “copy_status_old” ((eventAttr “Cm.Ack” “param2”
e)) (changeAttr “copy_sub_status_old” ((eventAttr “Cm.Ack” “param3” e))
(changeAttr “tray_top_old” ((eventAttr “Cm.Ack” “param4” e)) (changeAttr
“tray_middle_old” ((eventAttr “Cm.Ack” “param5” e)) (changeAttr
“tray_bottom_old” ((eventAttr “Cm.Ack” “param6” e)) a)))) [(“Ms.Size” [(“top”
(eventAttr “Cm.Ack” “param4” e)) (“middle” (eventAttr “Cm.Ack” “param5” e))
(“bottom” (eventAttr “Cm.Ack” “param6” e))])])
else if (memberEv “Cm.Ack” e) andalso (((eventAttr “Cm.Ack” “param2” e)=Int 255
```

```
            andalso (pickupAttr "copy_status_old" a)<>Int 255) andalso ((eventAttr
            "Cm.Ack" "param4" e)=(pickupAttr "tray_top_old" a) andalso (eventAttr
            "Cm.Ack" "param5" e)=(pickupAttr "tray_middle_old" a) andalso (eventAttr
            "Cm.Ack" "param6" e)=(pickupAttr "tray_bottom_old" a))) then
                    (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
            e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a)
            [("St.Reset" []) ("Ms.Reset" [])])
            else if (memberEv "Cm.Ack" e) andalso (((eventAttr "Cm.Ack" "param2" e)=Int 255
            andalso (pickupAttr "copy_status_old" a)=Int 255) andalso ((eventAttr
            "Cm.Ack" "param4" e)=(pickupAttr "tray_top_old" a) andalso (eventAttr
            "Cm.Ack" "param5" e)=(pickupAttr "tray_middle_old" a) andalso (eventAttr
            "Cm.Ack" "param6" e)=(pickupAttr "tray_bottom_old" a))) then
                    (State "S_ready" changeAttr "copy_status_old" ((eventAttr "Cm.Ack" "param2"
            e)) (changeAttr "copy_sub_status_old" ((eventAttr "Cm.Ack" "param3" e)) a) [])
            else
                    (Null [] [])
        | Sys_op_panel_status_ctrl (State "S_wait_copy_count" a e env) =
            if (memberEv "Cm.Ack" e) then
                    (State "S_ready" a [("Ms.Count" [("counter" (eventAttr "Cm.Ack" "param2"
            e))])])
            else
                    (Null [] [])
        | Sys_op_panel_status_ctrl (_ _ _ _) = (Null [] []);
    (* Sys_op_panel_timer_ctrl *)
    fun Sys_op_panel_timer_ctrl (Name _ _ _) = (Name [] [("Sys_op_panel_timer_ctrl"
            [])])
        | Sys_op_panel_timer_ctrl (Initial a e env) =
            (Parallel [Sys_op_panel_timer_ctrl_tm_400m_ctrl
            Sys_op_panel_timer_ctrl_tm_200m_ctrl
            Sys_op_panel_timer_ctrl_tm_60s_ctrl Sys_op_panel_timer_ctrl_tm_1s_ctrl]
            a [])
        | Sys_op_panel_timer_ctrl (_ _ _ _) = (Null [] []);
    (* Sys_op_panel_key_if *)
    fun Sys_op_panel_key_if (Name _ _ _) = (Name [] [("Sys_op_panel_key_if" [])])
        | Sys_op_panel_key_if (Initial a e env) =
            (State "S1" a [])
        | Sys_op_panel_key_if (State "S1" a e env) =
            if (memberEv "Key.In" e) andalso ((eventAttr "Key.In" "ScanCode" e)=Int 10) then
                    (State "S1" a [("Key.Clear" [])])
            else if (memberEv "Key.In" e) andalso ((eventAttr "Key.In" "ScanCode" e)=Int 11)
```

```
            then
                    (State "S1" a [("Key.Reset" [])])
            else if (memberEv "Key.In" e) andalso ((eventAttr "Key.In" "ScanCode" e)=Int 12)
            then
                    (State "S1" a [("Key.Start" [])])
            else if (memberEv "Key.In" e) andalso ((eventAttr "Key.In" "ScanCode" e)=Int 13)
            then
                    (State "S1" a [("Key.Stop" [])])
            else if (memberEv "Key.In" e) andalso ((eventAttr "Key.In" "ScanCode" e)=Int 14)
            then
                    (State "S1" a [("Key.Select" [])])
            else if (memberEv "Key.In" e) andalso (((eventAttr "Key.In" "code" e)=Int 0
            orelse (eventAttr "Key.In" "code" e)>Int 0) andalso ((eventAttr "Key.In" "code"
            e)=Int 9 orelse (eventAttr "Key.In" "code" e)<Int 9)) then
                    (State "S1" a [("Key.Number" [("code" (eventAttr "Key.In" "code" e))])])
            else if (memberEv "Ms.Jam" e) then
                    (State "S2" a [])
            else
                    (Null [] [])
        | Sys_op_panel_key_if (_ _ _ _) = (Null [] []);
    (* Sys_op_panel_lcd_if *)
    fun Sys_op_panel_lcd_if (Name _ _ _) = (Name [] [("Sys_op_panel_lcd_if" [])])
        | Sys_op_panel_lcd_if (Initial a e env) =
            (Parallel [Sys_op_panel_lcd_if_mess_lcd Sys_op_panel_lcd_if_psel_lcd
            Sys_op_panel_lcd_if_pcnt_lcd Sys_op_panel_lcd_if_jam_lcd] a [])
        | Sys_op_panel_lcd_if (_ _ _ _) = (Null [] []);
    (* Sys_op_panel_comport_if *)
    fun Sys_op_panel_comport_if (Name _ _ _) = (Name [] [("Sys_op_panel_comport_if"
            [])])
        | Sys_op_panel_comport_if (Initial a e env) =
            (State "S_ready" a [])
        | Sys_op_panel_comport_if (State "S_ready" a e env) =
            if (memberEv "Cm.Send" e) then
                    (State "S_wait" a [("Pp.Send" [("cid" (eventAttr "Cm.Send" "cid" e)) ("cid"
            (eventAttr "Cm.Send" "clen" e)) ("param2" (eventAttr "Cm.Send" "param2" e))
            ("param3" (eventAttr "Cm.Send" "param3" e)) ("param4" (eventAttr "Cm.Send"
            "param4" e))]) ("Tm.o1s_Start" [])])
            else
                    (Null [] [])
        | Sys_op_panel_comport_if (State "S_wait" a e env) =
```

```
            if (memberEv "Pp.Ack" e) then
                    (State "S_ready" a [("Cm.Ack" [("rid" (eventAttr "Pp.Ack" "rid" e)) ("rlen"
            (eventAttr "Pp.Ack" "rlen" e)) ("param2" (eventAttr "Pp.Ack" "param2" e))
            ("param3" (eventAttr "Pp.Ack" "param3" e)) ("param4" (eventAttr "Pp.Ack"
            "param4" e)) ("param5" (eventAttr "Pp.Ack" "param5" e)) ("param6" (eventAttr
            "Pp.Ack" "param6" e))]) ("Tm.o1s_Stop" [])])
            else if (memberEv "Pp.Nak" e) then
                    (State "S_nak" a [("Ms.Nak" [("info" (eventAttr "Pp.Nak" "param2" e))])
1230        ("Tm.o1s_Stop" [])])
            else if (memberEv "Tm.o1s" e) then
                    (State "S_timeout" a [("Ms.Timeout" []) ("Tm.o1s_Stop" [])])
            else
                    (Null [] [])
        | Sys_op_panel_comport_if (_ _ _ _) = (Null [] []);
    (* Sys_op_panel_cur_settings *)
    fun Sys_op_panel_cur_settings (Name _ _ _) = (Name []
            [("Sys_op_panel_cur_settings" [])])
        | Sys_op_panel_cur_settings (Initial a e env) =
1240        (State "S1" changeAttr "paper_count" (Int 1) (changeAttr
            "paper_select_position" (Int 1) a) [("Si.CurrentChange" [("paper_count" Int
            1) ("paper_select_position" Int 1)])])
        | Sys_op_panel_cur_settings (State "S1" a e env) =
            if (memberEv "Si.SetPaperCount" e) then
                    (State "S1" changeAttr "paper_count" ((eventAttr "Si.SetPaperCount"
            "paper_count" e)) a [("Si.CurrentChange" [("paper_count" (eventAttr
            "Si.SetPaperCount" "paper_count" e)) ("paper_select_position" (pickupAttr
            "paper_select_position" a))])])
            else if (memberEv "Si.SetPaperPosition" e) then
1250                (State "S1" changeAttr "paper_select_position" ((eventAttr
            "Si.SetPaperPosition" "paper_select_position" e)) a [("Si.CurrentChange"
            [("paper_count" (pickupAttr "paper_count" a)) ("paper_select_position"
            (eventAttr "Si.SetPaperPosition" "paper_select_position" e))])])
            else
                    (Null [] [])
        | Sys_op_panel_cur_settings (_ _ _ _) = (Null [] []);
    (* Sys_op_panel_def_settings *)
    fun Sys_op_panel_def_settings (Name _ _ _) = (Name []
            [("Sys_op_panel_def_settings" [])])
1260    | Sys_op_panel_def_settings (Initial a e env) =
            (State "S1" changeAttr "paper_count" (Int 1) (changeAttr
```

```
            "paper_select_position" (Int 1) a) [("Si.DefaultChange" [("paper_count" Int
            1) ("paper_select_position" Int 1)])])
      | Sys_op_panel_def_settings (_ _ _ _) = (Null [] []);
(* Sys_op_panel *)
fun Sys_op_panel (Name _ _ _) = (Name [] [("Sys_op_panel" [])])
      | Sys_op_panel (Initial a e env) =
            (Parallel [Sys_op_panel_copy_ctrl Sys_op_panel_status_ctrl
            Sys_op_panel_timer_ctrl Sys_op_panel_key_if Sys_op_panel_lcd_if
            Sys_op_panel_comport_if Sys_op_panel_cur_settings
            Sys_op_panel_def_settings] a [])
      | Sys_op_panel (_ _ _ _) = (Null [] []);
(*
symbol table of class
  0: Default_Settings (CLASS)
  1:    Si (FIELD)
  2:    S1 (STATE(state))
  3:    paper_count (ATTRIBUTE)
  4:    paper_select_position (ATTRIBUTE)
  5:    Start (TRANSITION)
  6: Current_Settings (CLASS)
  7:    Si (FIELD)
  8:    S1 (STATE(state))
  9:    paper_count (ATTRIBUTE)
  10:    paper_select_position (ATTRIBUTE)
  11:    Start (TRANSITION)
  12:    T1 (TRANSITION)
  13:    T2 (TRANSITION)
  14: Copy_Control (CLASS)
  15:    St (FIELD)
  16:    Ms (FIELD)
  17:    Tm (FIELD)
  18:    Si (FIELD)
  19:    Key (FIELD)
  20:    S_ready (STATE(state))
  21:    S_copy (STATE(state))
  22:    S_start1 (STATE(state))
  23:    S_start2 (STATE(state))
  24:    S_stop1 (STATE(state))
  25:    S_stop2 (STATE(state))
  26:    S_empty1 (STATE(state))
```

27: *S_empty2* (*STATE*(*state*))
28: *S_reset* (*STATE*(*state*))
29: *S_select* (*STATE*(*state*))
30: *default_paper_count* (*ATTRIBUTE*)
31: *default_paper_select_position* (*ATTRIBUTE*)
32: *current_paper_count* (*ATTRIBUTE*)
33: *current_paper_select_position* (*ATTRIBUTE*)
34: *key_input_start_flag* (*ATTRIBUTE*)
35: *Start* (*TRANSITION*)
36: *T1* (*TRANSITION*)
37: *T2* (*TRANSITION*)
38: *T3* (*TRANSITION*)
39: *T4* (*TRANSITION*)
40: *T5* (*TRANSITION*)
41: *T6* (*TRANSITION*)
42: *T7* (*TRANSITION*)
43: *T7_2* (*TRANSITION*)
44: *T8* (*TRANSITION*)
45: *T9* (*TRANSITION*)
46: *T10* (*TRANSITION*)
47: *T11* (*TRANSITION*)
48: *T12* (*TRANSITION*)
49: *T13* (*TRANSITION*)
50: *T14* (*TRANSITION*)
51: *T15* (*TRANSITION*)
52: *T16* (*TRANSITION*)
53: *T17* (*TRANSITION*)
54: *T18* (*TRANSITION*)
55: *T19* (*TRANSITION*)
56: *T20* (*TRANSITION*)
57: *Status_Control* (*CLASS*)
58: *St* (*FIELD*)
59: *Ms* (*FIELD*)
60: *Tm* (*FIELD*)
61: *Si* (*FIELD*)
62: *copy_status_old* (*ATTRIBUTE*)
63: *copy_sub_status_old* (*ATTRIBUTE*)
64: *tray_top_old* (*ATTRIBUTE*)
65: *tray_middle_old* (*ATTRIBUTE*)
66: *tray_bottom_old* (*ATTRIBUTE*)

67: *default_paper_count* (*ATTRIBUTE*)
68: *default_paper_select_position* (*ATTRIBUTE*)
69: *current_paper_count* (*ATTRIBUTE*)
70: *current_paper_select_position* (*ATTRIBUTE*)
71: *S_ready* (*STATE*(*state*))
72: *S_wait_status* (*STATE*(*state*))
73: *S_wait_copy_count* (*STATE*(*state*))
74: *Start* (*TRANSITION*)
75: *T1* (*TRANSITION*)
76: *T2* (*TRANSITION*)
77: *T3* (*TRANSITION*)
78: *T4* (*TRANSITION*)
79: *T5* (*TRANSITION*)
80: *T6* (*TRANSITION*)
81: *T7* (*TRANSITION*)
82: *T8* (*TRANSITION*)
83: *T9* (*TRANSITION*)
84: *T10* (*TRANSITION*)
85: *T11* (*TRANSITION*)
86: *T12* (*TRANSITION*)
87: *T13* (*TRANSITION*)
88: *T14* (*TRANSITION*)
89: *T15* (*TRANSITION*)
90: *T16* (*TRANSITION*)
91: *T17* (*TRANSITION*)
92: *T18* (*TRANSITION*)
93: *T19* (*TRANSITION*)
94: *T20* (*TRANSITION*)
95: *T21* (*TRANSITION*)
96: *T22* (*TRANSITION*)
97: *T23* (*TRANSITION*)
98: *T24* (*TRANSITION*)
99: *T25* (*TRANSITION*)
100: *T26* (*TRANSITION*)
101: *T27* (*TRANSITION*)
102: *T28* (*TRANSITION*)
103: *T29* (*TRANSITION*)
104: *T30* (*TRANSITION*)
105: *T31* (*TRANSITION*)
106: *T32* (*TRANSITION*)

107: *T33* (*TRANSITION*)
108: *T34* (*TRANSITION*)
109: *T35* (*TRANSITION*)
110: *T36* (*TRANSITION*)
111: *T37* (*TRANSITION*)
112: *T38* (*TRANSITION*)
113: *T39* (*TRANSITION*)
114: *T39_2* (*TRANSITION*)
115: *T39_3* (*TRANSITION*)
116: *T40* (*TRANSITION*)
117: *T41* (*TRANSITION*)
118: *T42* (*TRANSITION*)
119: *T43* (*TRANSITION*)
120: *T44* (*TRANSITION*)
121: *T45* (*TRANSITION*)
122: *T46* (*TRANSITION*)
123: *T47* (*TRANSITION*)
124: *T48* (*TRANSITION*)
125: *T49* (*TRANSITION*)
126: *T50* (*TRANSITION*)
127: *T51* (*TRANSITION*)
128: *T52* (*TRANSITION*)
129: *T53* (*TRANSITION*)
130: *T54* (*TRANSITION*)
131: *T55* (*TRANSITION*)
132: *T56* (*TRANSITION*)
133: *T57* (*TRANSITION*)
134: *T58* (*TRANSITION*)
135: *T59* (*TRANSITION*)
136: *T60* (*TRANSITION*)
137: *T61* (*TRANSITION*)
138: *T62* (*TRANSITION*)
139: *T63* (*TRANSITION*)
140: *T64* (*TRANSITION*)
141: *T65* (*TRANSITION*)
142: *T66* (*TRANSITION*)
143: *T67* (*TRANSITION*)
144: *T68* (*TRANSITION*)
145: *T69* (*TRANSITION*)
146: *T70* (*TRANSITION*)

147: *T71* (*TRANSITION*)
148: *T71_2* (*TRANSITION*)
149: *T71_3* (*TRANSITION*)
150: *T72* (*TRANSITION*)
151: *T72_2* (*TRANSITION*)
152: *T73* (*TRANSITION*)
153: *T73_2* (*TRANSITION*)
154: *Timer_400m_Interval* (*CLASS*)
155: *Tm* (*FIELD*)
156: *S_on* (*STATE*(*state*))
157: *S_off* (*STATE*(*state*))
158: *Start* (*TRANSITION*)
159: *T1* (*TRANSITION*)
160: *T2* (*TRANSITION*)
161: *T3* (*TRANSITION*)
162: *Timer_200m_Interval* (*CLASS*)
163: *Tm* (*FIELD*)
164: *S_on* (*STATE*(*state*))
165: *S_off* (*STATE*(*state*))
166: *Start* (*TRANSITION*)
167: *T1* (*TRANSITION*)
168: *T2* (*TRANSITION*)
169: *T3* (*TRANSITION*)
170: *Timer_60s_Timeout* (*CLASS*)
171: *Tm* (*FIELD*)
172: *S_on* (*STATE*(*state*))
173: *S_off* (*STATE*(*state*))
174: *Start* (*TRANSITION*)
175: *T1* (*TRANSITION*)
176: *T2* (*TRANSITION*)
177: *T3* (*TRANSITION*)
178: *Timer_1s_Timeout* (*CLASS*)
179: *Tm* (*FIELD*)
180: *S_on* (*STATE*(*state*))
181: *S_off* (*STATE*(*state*))
182: *Start* (*TRANSITION*)
183: *T1* (*TRANSITION*)
184: *T2* (*TRANSITION*)
185: *T3* (*TRANSITION*)
186: *Timer_Control* (*CLASS*)

187: *tm_400m_ctrl* (*OBJECT*)
188: *tm_200m_ctrl* (*OBJECT*)
189: *tm_60s_ctrl* (*OBJECT*)
190: *tm_1s_ctrl* (*OBJECT*)
191: *timer_ctrl_children* (*STATE*(*parallel object*))
192: *Start* (*TRANSITION*)
193: *Key_Interface* (*CLASS*)
194: *Key* (*FIELD*)
1470 195: *Ms* (*FIELD*)
196: *S1* (*STATE*(*state*))
197: *S2* (*STATE*(*state*))
198: *Start* (*TRANSITION*)
199: *T10* (*TRANSITION*)
200: *T11* (*TRANSITION*)
201: *T12* (*TRANSITION*)
202: *T13* (*TRANSITION*)
203: *T14* (*TRANSITION*)
204: *Tn* (*TRANSITION*)
1480 205: *Td* (*TRANSITION*)
206: *ComPort_Interface* (*CLASS*)
207: *Cm* (*FIELD*)
208: *Pp* (*FIELD*)
209: *Tm* (*FIELD*)
210: *Ms* (*FIELD*)
211: *S_ready* (*STATE*(*state*))
212: *S_wait* (*STATE*(*state*))
213: *S_nak* (*STATE*(*state*))
214: *S_timeout* (*STATE*(*state*))
1490 215: *Start* (*TRANSITION*)
216: *T1* (*TRANSITION*)
217: *T2* (*TRANSITION*)
218: *T3* (*TRANSITION*)
219: *T4* (*TRANSITION*)
220: *Message_LCD* (*CLASS*)
221: *Ms* (*FIELD*)
222: *message* (*ATTRIBUTE*)
223: *nak_code* (*ATTRIBUTE*)
224: *S1* (*STATE*(*state*))
1500 225: *Start* (*TRANSITION*)
226: *T1* (*TRANSITION*)

227: *T2* (*TRANSITION*)
228: *T3* (*TRANSITION*)
229: *T4* (*TRANSITION*)
230: *T5* (*TRANSITION*)
231: *T6* (*TRANSITION*)
232: *T7* (*TRANSITION*)
233: *T8* (*TRANSITION*)
234: *T91* (*TRANSITION*)
235: *T92* (*TRANSITION*)
236: *T93* (*TRANSITION*)
237: *PaperSelect_LCD* (*CLASS*)
238: *Ms* (*FIELD*)
239: *S1* (*STATE*(*state*))
240: *top* (*ATTRIBUTE*)
241: *middle* (*ATTRIBUTE*)
242: *bottom* (*ATTRIBUTE*)
243: *current* (*ATTRIBUTE*)
244: *Start* (*TRANSITION*)
245: *T1* (*TRANSITION*)
246: *T2* (*TRANSITION*)
247: *PaperCount_LCD* (*CLASS*)
248: *Ms* (*FIELD*)
249: *S1* (*STATE*(*state*))
250: *count* (*ATTRIBUTE*)
251: *Start* (*TRANSITION*)
252: *T2* (*TRANSITION*)
253: *Jam_LCD* (*CLASS*)
254: *Ms* (*FIELD*)
255: *S1* (*STATE*(*state*))
256: *S2* (*STATE*(*state*))
257: *message* (*ATTRIBUTE*)
258: *pos1* (*ATTRIBUTE*)
259: *pos2* (*ATTRIBUTE*)
260: *pos3* (*ATTRIBUTE*)
261: *Start* (*TRANSITION*)
262: *T1* (*TRANSITION*)
263: *LCD_Interface* (*CLASS*)
264: *St* (*FIELD*)
265: *mess_lcd* (*OBJECT*)
266: *psel_lcd* (*OBJECT*)

267: *pcnt_lcd* (*OBJECT*)
268: *jam_lcd* (*OBJECT*)
269: *lcd_if_child* (*STATE*(*parallel object*))
270: *Start* (*TRANSITION*)
271: *Operation_Panel* (*CLASS*)
272: *copy_ctrl* (*OBJECT*)
273: *status_ctrl* (*OBJECT*)
274: *timer_ctrl* (*OBJECT*)
275: *key_if* (*OBJECT*)
276: *lcd_if* (*OBJECT*)
277: *comport_if* (*OBJECT*)
278: *cur_settings* (*OBJECT*)
279: *def_settings* (*OBJECT*)
280: *panel_children* (*STATE*(*parallel object*))
281: *Start* (*TRANSITION*)
282: *Sys* (*CLASS*)
283: *op_panel* (*OBJECT*)
284: *op_panel* (*STATE*(*single object*))
285: *Start* (*TRANSITION*)

symbol table of field

0: *Ms* (*FIELD*)
1: *Wait* (*EVENT*)
2: *Standby* (*EVENT*)
3: *Copy* (*EVENT*)
4: *Stop* (*EVENT*)
5: *Reset* (*EVENT*)
6: *Empty* (*EVENT*)
7: *Restart* (*EVENT*)
8: *Range* (*EVENT*)
9: *Size* (*EVENT*)
10: *Select* (*EVENT*)
11: *Count* (*EVENT*)
12: *Jam* (*EVENT*)
13: *Nak* (*EVENT*)
14: *Timeout* (*EVENT*)
15: *IllState* (*EVENT*)
16: *Key* (*FIELD*)
17: *In* (*EVENT*)
18: *Number* (*EVENT*)
19: *Clear* (*EVENT*)

20: *Reset* (*EVENT*)
21: *Start* (*EVENT*)
22: *Stop* (*EVENT*)
23: *Select* (*EVENT*)
24: *St* (*FIELD*)
25: *Wait* (*EVENT*)
26: *Wait_Empty* (*EVENT*)
27: *Wait_Range* (*EVENT*)
28: *Standby* (*EVENT*)
29: *Standby_Empty* (*EVENT*)
30: *Standby_Range* (*EVENT*)
31: *Copy* (*EVENT*)
32: *Stop* (*EVENT*)
33: *Jam* (*EVENT*)
34: *Copy_Empty* (*EVENT*)
35: *Restart* (*EVENT*)
36: *Reset* (*EVENT*)
37: *Cm* (*FIELD*)
38: *Send* (*EVENT*)
39: *Ack* (*EVENT*)
40: *Pp* (*FIELD*)
41: *Send* (*EVENT*)
42: *Ack* (*EVENT*)
43: *Nak* (*EVENT*)
44: *Tm* (*FIELD*)
45: *i400m* (*EVENT*)
46: *i400m_interval* (*EVENT*)
47: *i400m_Start* (*EVENT*)
48: *i400m_Stop* (*EVENT*)
49: *i200* (*EVENT*)
50: *i200m_interval* (*EVENT*)
51: *i200m_Start* (*EVENT*)
52: *i200m_Stop* (*EVENT*)
53: *o60s* (*EVENT*)
54: *o60s_timeout* (*EVENT*)
55: *o60s_Start* (*EVENT*)
56: *o60s_Stop* (*EVENT*)
57: *o1s* (*EVENT*)
58: *o1s_timeout* (*EVENT*)
59: *o1s_Start* (*EVENT*)

60: *o1s_Stop* (*EVENT*)
61: *Si* (*FIELD*)
62: *DefaultChange* (*EVENT*)
63: *CurrentChange* (*EVENT*)
64: *SetPaperCount* (*EVENT*)
65: *SetPaperPos* (*EVENT*)

symbol table of event class

0: *GENERIC_EVENT* (*CLASS*)
1: *Paper_Size_Display_Event* (*CLASS*)
2: *top* (*ATTRIBUTE*)
3: *middle* (*ATTRIBUTE*)
4: *bottom* (*ATTRIBUTE*)
5: *Paper_Select_Display_Event* (*CLASS*)
6: *current* (*ATTRIBUTE*)
7: *Paper_Counter_Display_Event* (*CLASS*)
8: *counter* (*ATTRIBUTE*)
9: *Nak_Info_Display_Event* (*CLASS*)
10: *info* (*ATTRIBUTE*)
11: *Jam_Display_Event* (*CLASS*)
12: *pos1* (*ATTRIBUTE*)
13: *pos2* (*ATTRIBUTE*)
14: *pos3* (*ATTRIBUTE*)
15: *Scan_Code_Event* (*CLASS*)
16: *code* (*ATTRIBUTE*)
17: *Number_Key_Event* (*CLASS*)
18: *code* (*ATTRIBUTE*)
19: *Command_Send_Event* (*CLASS*)
20: *cid* (*ATTRIBUTE*)
21: *clen* (*ATTRIBUTE*)
22: *param2* (*ATTRIBUTE*)
23: *param3* (*ATTRIBUTE*)
24: *param4* (*ATTRIBUTE*)
25: *Ack_Receive_Event* (*CLASS*)
26: *rid* (*ATTRIBUTE*)
27: *rlen* (*ATTRIBUTE*)
28: *param2* (*ATTRIBUTE*)
29: *param3* (*ATTRIBUTE*)
30: *param4* (*ATTRIBUTE*)
31: *param5* (*ATTRIBUTE*)
32: *param6* (*ATTRIBUTE*)

33: *param7* (*ATTRIBUTE*)
34: *param8* (*ATTRIBUTE*)
35: *Physical_Command_Send_Event* (*CLASS*)
36: *cid* (*ATTRIBUTE*)
37: *clen* (*ATTRIBUTE*)
38: *param2* (*ATTRIBUTE*)
39: *param3* (*ATTRIBUTE*)
40: *param4* (*ATTRIBUTE*)
41: *Physical_Ack_Receive_Event* (*CLASS*)
42: *rid* (*ATTRIBUTE*)
43: *rlen* (*ATTRIBUTE*)
44: *param2* (*ATTRIBUTE*)
45: *param3* (*ATTRIBUTE*)
46: *param4* (*ATTRIBUTE*)
47: *param5* (*ATTRIBUTE*)
48: *param6* (*ATTRIBUTE*)
49: *param7* (*ATTRIBUTE*)
50: *param8* (*ATTRIBUTE*)
51: *Physical_Nak_Receive_Event* (*CLASS*)
52: *rid* (*ATTRIBUTE*)
53: *rlen* (*ATTRIBUTE*)
54: *param2* (*ATTRIBUTE*)
55: *Set_Paper_Count_Event* (*CLASS*)
56: *paper_count* (*ATTRIBUTE*)
57: *Set_Paper_Position_Event* (*CLASS*)
58: *paper_pos* (*ATTRIBUTE*)
59: *Setting_Change_Event* (*CLASS*)
60: *paper_count* (*ATTRIBUTE*)
61: *paper_select_position* (*ATTRIBUTE*)

instance list

tm_400m_ctrl : *Timer_400m_Interval*
tm_200m_ctrl : *Timer_200m_Interval*
tm_60s_ctrl : *Timer_60s_Timeout*
tm_1s_ctrl : *Timer_1s_Timeout*
mess_lcd : *Message_LCD*
psel_lcd : *PaperSelect_LCD*
pcnt_lcd : *PaperCount_LCD*
jam_lcd : *Jam_LCD*
copy_ctrl : *Copy_Control*
status_ctrl : *Status_Control*

timer_ctrl : *Timer_Control*
key_if : *Key_Interface*
lcd_if : *LCD_Interface*
comport_if : *ComPort_Interface*
cur_settings : *Current_Settings*
def_settings : *Default_Settings*
op_panel : *Operation_Panel*
*)
