筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

携帯電話を活用する音声応答家電の開発
－家電組み込み用赤外線通信・制御ファームウェアの開発－

劉　俊鵬

(コンピュータサイエンス専攻)

指導教員　田中　二郎

２０１２年３月

概要

本プロジェクトは、筑波大学システム情報工学研究科コンピュータサイエンス専攻、高度 IT 人材育成のための実践的ソフトウェア開発専修プログラムにおける研究開発プロジェクトとして、同大学院に所属する教員から受注した研究開発である。
本研究開発プロジェクトでは目の不自由な方でも利用できる音声応答家電を開発する。音声合成機能つき携帯電話を活用し、家電の音声応答を低コストで実現するモデルを提案するため。赤外線通信用の通信ボードを作り、市販の家電に埋め込むことでこの実証機を製作する。
筆者は、携帯電話を活用する音声応答家電の開発のうち、とくに家電組み込み用赤外線通信制御ファームウェアの開発を担当した。すなわち通信ボードのプロトタイピングに適したマイコンボードを選択し、赤外線通信を行うプログラムを書き込み、携帯電話と通信の上、家電の状態をチェックしながら、携帯の命令によって家電を制御する組み込み通信・コントロールモジュールの開発を行った。
本プロジェクトは 8 ヶ月の開発がかかって、携帯電話を活用する音声応答家電を作る目標を達成した。

目次

第 1 章　はじめに ……1
1.1　研究の背景 ……1
1.2　研究の目的 ……1
1.3　本報告書の構成 ……1
第 2 章　関連研究 ……2
2.1　UPnP ……2
2.2　URC ……2
2.3　HAVi ……2
2.4　ECHONET ……3
2.5　Jini ……4
2.6　赤外線通信方式の調査 ……4
第 3 章　システムの構成および開発計画 ……5
3.1　機能要件 ……5
3.2　非機能要件 ……6
3.3　機材の選定 ……6
3.3.1　端末の選定 ……6
3.3.2　通信方式の選定 ……6
3.3.3　家電用マイコンの選定 ……7
3.3.4　実証用家電の選定 ……7
3.4　全体設計 ……8
3.4.1　総体的なユースケース図 ……8
3.5　開発体制 ……8
3.6　プロジェクトマネージメント ……9
3.6.1　プロジェクトマネージメントの役割分担 ……9
3.6.2　プロジェクトマネージメントの方針 ……9
3.7　スケジュール ……10
第 4 章　家電側のソフトウェア設計 ……12
4.1　家電側のユースケース ……12
4.2　家電側のクラス ……12
4.3　家電側のシーケンス ……14
4.3.1　タイマーシーケンス ……14
4.3.2　CIR 受信シーケンス ……14
4.3.3　ボタンコントロールシーケンス図 ……15
第 5 章　赤外線通信の実装方式の検討 ……17
5.1　通信原理 ……17
5.2　IrOBEX の実装方式の検討 ……17
5.3　試作環境のハードウェアの選択 ……18
5.4　ボード側の開発ツール ……19
第 6 章　家電用赤外線通信モジュールの開発 ……20

6.1 IrDA による赤外線通信の実装 ……20
6.1.1 IrDA OBEX 通信用 API ……20
6.1.2 電話帳オブジェクトの送受信 ……20
6.1.3 ユーザ定義オブジェクトの送受信 ……20
6.1.4 Microchip Standard IrDA Stack の問題点 ……20
6.2 CIR による赤外線通信の実装 ……21
6.2.1 CIR とは ……21
6.2.2 モジュレーション(変調) ……21
6.2.3 NEC フォーマット ……21
6.2.4 ボード側ＣＩＲ受信のハードウェア ……23
6.2.5 PPM 変調のデコード ……25
6.2.6 ボード側ＣＩＲ受信のソフトウェア ……28
6.2.7 赤外線通信モジュールのテスト ……28
6.2.8 新規家電への組み込み ……28
第 7 章 家電音声化システム・家電操作の実証 ……29
7.1 家電用コントロールモジュール ……29
7.1.1 メーカー用素直なモデル ……29
7.1.2 試作モデル ……29
7.2 実証用家電(おまかせミールさん)について ……30
7.2.1 実証用家電の LED/SW Board ……31
7.2.2 ダイナミックの調査 ……33
7.3 ボタンの入力のエミュレーション ……34
7.3.1 Switch Emulation Board ……35
7.3.2 Analog Multiplexers の活用 ……36
7.3.3 ボタン入力エミュレーションソフトウェアの概要 ……37
7.3.4 ボタン入力エミュレーションソフトウェアの実装 ……37
7.4 家電 LED の読み取り目的と原理 ……37
7.4.1 １つ LED の点灯原理 ……38
7.4.2 LED/SW ボードと GL-3P404 のダイナミック読み取りの関係 ……39
7.5 LED 読み取り部分の実装の事前調査 ……41
7.5.1 4 つ Digit Segment の切り替える周期と順番 ……41
7.5.2 LED 読み取りの実装方針 ……41
7.5.3 ハードウェアによるノイズの低減 ……42
7.6 LED 読み取り部分の実装 ……43
7.6.1 ソフトウェア実装の事前準備 ……43
7.6.2 GL-3P404 の読み取り処理 ……44
7.6.3 ソフトウェアによるノイズの除去 ……45
7.7 CIR 受信、疑似ボタン入力、LED 読み取りの統合 ……45
7.7.1 3 部分連携用のハードウェア ……45
7.7.2 タイマー割り込みとソフトウェアタイマーの活用 ……46
7.7.3 CIR コマンドからボタン入力シーケンスへの変換 ……47
7.7.4 実証用家電の LED 状態による疑似ボタン入力 ……47

7.7.5　点滅している LED の時間の読み取りによる疑似ボタン入力 ……48
7.7.6　統合した後の強化バージョンの疑似ボタン入力処理 ……49
第 8 章　家電側のソフトウェアの検証 ……51
8.1　家電側のテスト環境 ……51
8.2　家電側の単体テスト ……52
8.2.1　CIR 受信部分の単体テスト ……52
8.2.2　疑似ボタン入力部分の単体テスト ……53
8.2.3　LED 読み取り部分の単体テスト ……53
8.3　3 部分統合の結合テスト ……53
8.3.1　リモコンのボタンを押す送信で、携帯の CIR 送信にふりをするアプリ ……53
8.3.2　テスト内容と結果 ……53
8.4　被験者試験について ……54
第 9 章　まとめ ……55
謝辞 ……57
参考文献 ……58

図目次

図 2-1　ECHONET クラスグループ例 …… 3
図 3-1　システム構成図 …… 5
図 3-2　おまかせミールさん …… 7
図 3-3　総体的なユースケース図 …… 8
図 3-4　仕事分担と筆者の担当 …… 9
図 4-1　家電側のユースケース図 …… 12
図 4-2　クラス図 …… 13
図 4-3　タイマーシーケンス図 …… 14
図 4-4　CIR 受信シーケンス図 …… 15
図 4-5　ボタンコントロールシーケンス図 …… 16
図 5-1　赤外線通信の原理 …… 17
図 5-2　IrOBEX の実装方式 …… 18
図 5-3　開発環境 Explorer16 …… 19
図 6-1　Frame の構成 …… 22
図 6-2　キャリア変調した後のロジック 1 と 0 …… 23
図 6-3　PIC24FJ64GA002 …… 23
図 6-4　PIC24FJXXGA002(28 ピン版)のピンアサイン …… 23
図 6-6　PIC24FJ120GA010 …… 24
図 6-6　PIC24FJXXGA010 と PIC24FJXXXGA010 (100 ピン版)のピンアサイン … 24
図 6-7　デコードフローチャート …… 26
図 6-8　CIR 受信テスト …… 28
図 7-1　素直なモデル …… 29
図 7-2　試作モデル …… 30
図 7-3　通信ボードの実装方針 …… 31
図 7-4　解析した LED/SW Board …… 32
図 7-5　GL-3P404 の略図 …… 32
図 7-6　GL-3P404 の Interal connection diagram …… 33
図 7-7　ダイナミックスキャンの周期と電圧 …… 33
図 7-8　ダイナミックスキャンの概要 …… 34
図 7-9　ハードウェア …… 35
図 7-10　Switch Emulation Board …… 35
図 7-11　Analog Multiplexers でボタン制御 …… 36
図 7-12　１つ LED の点灯原理 …… 38
図 7-13　フラットケーブルのピンと GL-3P404 の繋がり方 …… 40
図 7-14　ピン 9～12 の電圧シーケンス図 …… 41
図 7-15　ハードウェアで電圧の抑え方 …… 43
図 7-16　GL-3P404 読み取りフローチャート …… 44
図 7-17　3 部分連携のハードウェア環境 …… 45
図 7-18　LED Switch Emulation Board …… 46

図 7-19　実証用家電の時間設定の流れ……48
図 7-20　各疑似ボタン入力コードに対する処理フローチャート ……50
図 8-1　家電側のテスト環境……51
図 8-2　NEC フォーマットを送信できる赤外線リモコン(KTRA-109) ……52

表目次

表 2.1　URC の定義ファイルの種類 2
表 2.2　HAVi で取り扱われる情報とその概要 3
表 3.1　対象端末に対応するプロファイル 6
表 3.2　通信方式と特徴 6
表 3.3　赤外線通信の方式と特徴 6
表 3.4　携帯側の開発スケジュール 10
表 3.5　家電側の開発スケジュール 11
表 5.1　サンプルプログラムとマイコンのリソース 19
表 6.1　各部分の意味 22
表 6.2　送信情報 1 と 0 の定義 22
表 7.1　入力ピンとボタン対照表 37
表 7.2　LED 入力 Port 38
表 7.3　LED 入力 Port の値と表示数字の対照表 38
表 7.4　LED 入力 Port の値と部分ローマ字の対照表 39
表 7.5　PIC24FJ64GA002 の入力電圧情報 42
表 9.1　携帯側の開発スケジュール(実績) 55
表 9.2　家電側の開発スケジュール(実績) 56

第1章　はじめに

現在、洗濯機やオーブンレンジなど多くの家電製品は高機能化及び複雑化していて、目の不自由な方が使いこなすことは困難である。また、家電に音声応答機能を持たせるためには、スピーカーやアンプ等のハードウェア、規則音声合成のためのソフトウェア等を家電に内蔵させる必要があり、コストが高くなる。そのため、音声合成家電は一般の家電と比較して高価であるので、音声合成機能を家電製品にあまねく持たせることはコストの面から困難である。そこで、目の不自由な方の間でシェアが高い音声読み上げ機能付きの携帯電話を用いて、家電と連携させることで、音声応答での操作が可能となる家電を製作し実働させる。

1.1　研究の背景

現在家電の高機能化が進んでおり、視覚障碍者には使いにくくなっている問題がある。功能が複雑な家電に発声機能が付けないと、視覚障碍者は使用する時に色々な不便である。白物家電にはコストの制約が厳しくで、発声機能を付くのは困難である。この課題を解決するソリューションが求められる。

1.2　研究の目的

個々の家電に音声をつけるようとすると、高性能プロセッサ、アンプ、スピーカーが要る。値段が高くなる。ところで、視覚障碍者はメールやウェブブラウジングのために音声合成機能のついた携帯を常用している、この携帯と家電を連携させて低コストで家電の音声応答を実現する。

1.3　本報告書の構成

本段落で本報告書の構成を簡単に説明する。

第 1~3 章でシステム総体的な内容を説明する。

第 4 章~第 8 章は筆者が担当する内容を説明する。

第 4 章は筆者が担当するソフトウェアの設計を説明する。

第 5 章は開発する前の赤外線通信に関する調査を説明する。

第 6 章は通信機能の実装を説明する。

第 7 章はコントロール機能の実装を説明する。

第 8 章は筆者が担当する部分のテストを説明する。

第 9 章は筆者の感想と本システムに対するまとめである。

第2章　関連研究

本システムを開発する内に、幾つ関連研究を調べた。

2.1　UPnP

我々のシステムは家電をコントロールする時､まず家電の仕様を知る必要がある｡これで、家電の仕様記述を作るため、同チームの章は UPnP を調べた。UPnP(Universal Plug and Play)とは、パソコンや周辺機器、AV 機器、などの家電製品がネットワークで相互に機能の内容を転送する時の技術仕様である。(詳細は同チームの章の報告者の関連研究にある)

2.2　URC

URC(Universal Remote Console)[1]とは, 携帯電話や PDA 等にソフトを組み込むことで, 様々な機器を操作可能にするために，V2 という規格に準拠した端末である．機器との通信には無線 LAN や Bluetooth を使い，制御情報は XML でやり取りする．

URC では，以下の 4 つのファイルを用いて制御情報をやり取りしている．

表 2.1　URC の定義ファイルの種類

ファイルの種類	概要
ターゲットディスクリプション	全ての機器のソケットファイルの位置を定義する.
ソケットファイル	機器を操作するために使用する信号とそのデータ形式について定義する.
プレゼンテーションテンプレート	機器を制御するために必要な情報を定義する.
リソースディスクリプションフレームワーク	機器を制御するために必要な UI 情報を定義する.

(本段落は同チームの持田の調査結果である)

2.3　HAVi

HAVi(Home Audio Video Interoperability)[2]とは，グルンディッヒ，日立，松下，フィリップス，シャープ，ソニー，トムソン，東芝の 8 社によって策定された情報家電のためのミドルウェアである．HAVi は，IEEE1394 を使って，AV 機器をメーカーや機種にとらわれることなく，相互に接続するための共通基盤である．

互いに機器同士で制御しあうことができる．

HAVi では SDD(Self Describing Device)データと呼ばれる HAVi 機器としてのプロフィールや制御情報を含んだデータが Configulation ROM に置かれる．SDD データに含まれる情報は以下の 4 つである．

表 2.2　HAVi で取り扱われる情報とその概要

情報	概要
HAVi Device Profile	メーカーなどの機器全体に関する情報と，機器に含まれる機能に関する情報を定義
HAVi User Preferred Name	HAVi に準拠する機器ではユーザが任意に機器に名前を付けることができる．この名前に関して定義
HAVi DCM, HAVi DCM Profile, HAVi DCM Reference	制御プログラムとその属性情報, URL などを定義
HAVi Device Icon Bitmap	HAVi 機器を表すアイコンのデータ

（本段落は同チームの持田の調査結果である）

2.4　ECHONET

ECHONET（Energy Conservation and Homecare Network）[3]は、オブジェクト指向により家電機器をモデル化して、相互接続実現に必要となる共通的・基本的機能をサービスミドルウェアとして規格化される。規格の中で、家電機器を制御するためのオブジェクト定義は制御方法を細かく定義している。それは機器オブジェクトと呼ばれる[4]。機器の制御部分はクラス内に定義する。

オブジェクトの構造について、機器オブジェクトをいくつのクラスグループに分けて、この下にクラスがある。クラス内にプロパティがあって、制御内容又は機器状態を定義する。クラスグループコード、クラスコード及びプロパティにより、どんな設備であるか、どのような機能を持っているかを知る。例えば、下図のようにセンサ関連機器クラスグループ、空調関連機器クラスグループなどいくつのグループに分ける。電気ポット、冷凍冷蔵庫、オーブンレンジなどは調理・家事関連機器クラスグループの下にある。電気ポットの場合、蓋開閉状態、湯切れ警告などのプロパティを含む。

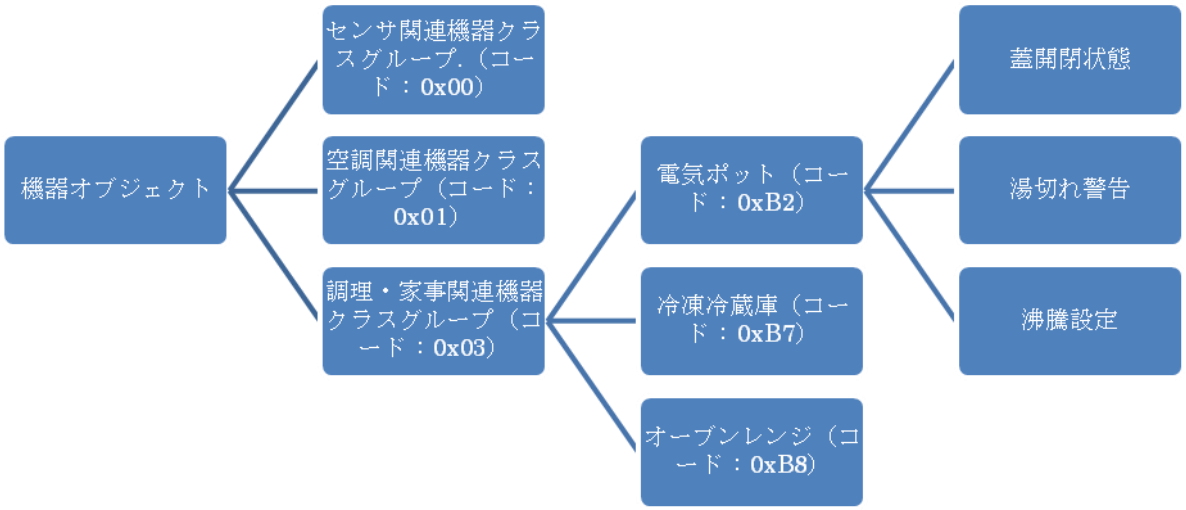


図 2-1　ECHONET クラスグループ例

機器を制御する時、目的機器の ECHONET アドレス、オブジェクト ID（クラスグループコード)、プロパティ ID（クラスコード)、及びサービス内容（SET、GET など）を指定したコマンドを送信することで制御が行われる[4]。例えば、電気ポットを用いて、沸騰させようと思う時、機器の ECHONET アドレス (Ox03)、オブジェクト ID(OxB2)、沸騰設定(Ox41)を送信することで、水を沸騰させる。

しかし、プロパティ内容が事前に決められているので、拡張性が弱く、今回の開発に適していない。

(本段落は同チームの徐の調査結果である)

2.5 Jini

Jini（Java Intelligent Network Infrastructure）[5]とは、デジタル機器をネットワークに接続するだけで、機器間での協調動作を実現するための Java 分散オブジェクト応用技術である。Jini のポイントは，ハードウェアもソフトウェアも区別せず，全部オブジェクトとして扱う点と，互いの情報を持たないオブジェクト同士でも通信できる点である。

Jini はサービスの提供側である「サービスプロバイダ」，サービスの利用側である「クライアント」、ネットワーク内のすべてのサービスを管理する「Lookup サービス」から構成される[6]。Discovery、Join、Lookup の 3 つ過程でプラグアンドプレイが実現される。

まずサービスプロバイダは Lookup サービスを見つける（Discovery)。そして、サービスプロバイダは自サービスのプロキシーオブジェクトを Lookup サービスに登録する（Join)。クライアントはサービスプロバイダを使用しようと思う時、Lookup サービスから探して、そのプロキシーをダウンロードする。それで、クライアントとサービスプロバイダはプロキシーを介して通信することで実現できる。

例えば、プリントとコンピュータと繋ぐ時、プリンタが Lookup サービスを発見して、自サービスのプロキシーオブジェクトを Lookup サービスに登録する。そして、コンピュータは Lookup サービスからプリンタを発見して、そのプロキシーをダウンロードする。それで、コンピュータはプロキシーを介してプリントと通信し、プリントを利用する[7]。

Jini では RMI[8]にて家電製品を連携させるということで、家電同士のやり取りはオブジェクトを使って行われる。しかし、携帯電話で RMI が使用できないので、今回の開発に適していない。

(本段落は同チームの徐の調査結果である)

2.6 赤外線通信方式の調査

筆者の担当する部分を実装する前に、赤外線通信方式は関連研究として調べた。調べた内容の詳細は本報告書の第 5 章に説明がある。

第3章　　システムの構成および開発計画

音声読み上げ機能付き携帯電話で、家電を音声応答可能にするシステム(以下、家電音声化システム)を設計、実装する。そして、家電音声化システムを用いて、実際に動作の実証を行うために、家電(岩谷産業社製「おまかせミールさん」IOC-125)を音声応答可能にする。
実証を行う家電として、おまかせミールさんを選定した。多くの家電に汎用的に利用できるようにするためには、ある程度の複雑な機能を持った家電で実証する必要がある。また、実証用家電には、開発したシステムを搭載し、家電を制御する必要があるため、集積度が低い方が良い。そのため、ある程度の複雑な機能を持ち、集積度が低い家電を選定した。

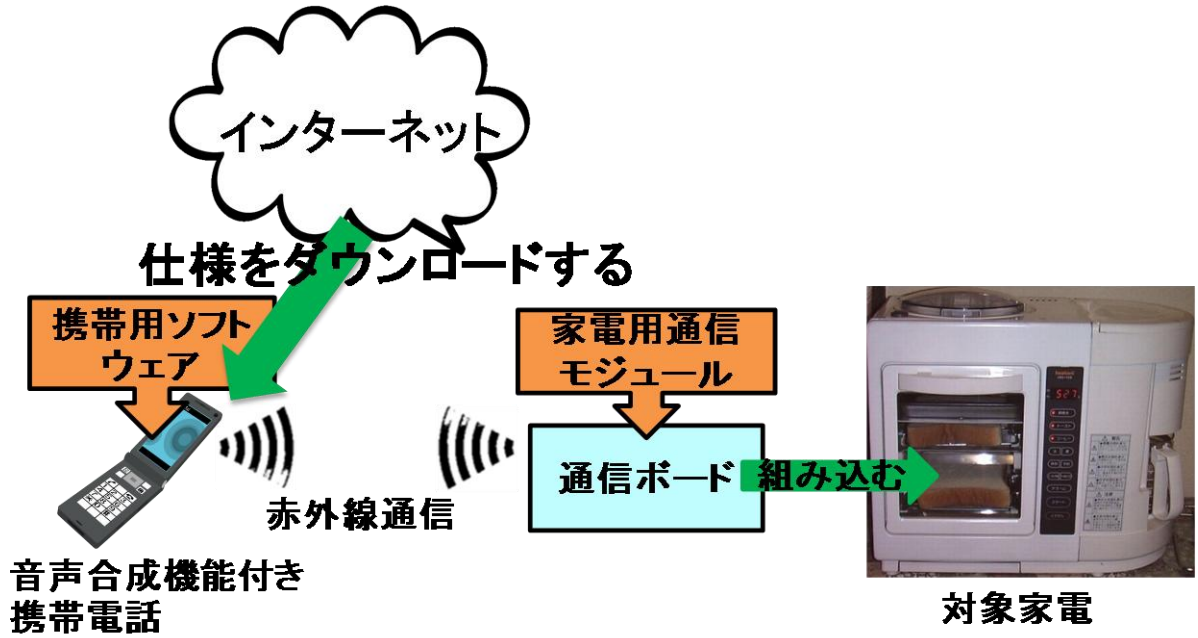


図 3-1　　システム構成図

家電音声化システムは、家電側に搭載する家電用通信モジュールと携帯電話側に搭載する携帯電話用ソフトウェアである。本プロジェクトの開発は 2 段階、まず、様々な家電に汎用に使えるものをつくり、次にそれを使って実際の家電の音声応答を実証する。

3.1　　機能要件

本システムの機能要件は以下である、各部分の詳しい内容は付録の「要件定義書」の機能要件の部分に示す。

携帯側
・家電の操作が行えること
・家電の仕様を読み取って様々な家電が操作可能であること
・操作に関しては音声のガイドがあること
家電側

・携帯の命令を受けること
・携帯の命令による家電のコントロール
・家電の状態を読み取ること

3.2 非機能要件

本システムの非機能要件は以下である。
効率性、機能性、安全性、移植性、使用性、信頼性、操作性
各部分の詳しい内容は付録の「要件定義書」の非機能要件の部分に示す。

3.3 機材の選定

各部分の機材の選定はソフトウェアを実装する前に決める必要があるので、これから検討する。

3.3.1 端末の選定

今回、使用する携帯電話は、発声機能の API を利用するため、らくらくホンシリーズのうち、DoJa5.1 プロファイルに対応したものを選定する。そして、文献[9]より 2011 年 9 月現在、DoJa5.1 に対応しているらくらくホンは、下表の通りである。

表 3.1 対象端末に対応するプロファイル

型番	製品名	対応プロファイル
F-08C	らくらくホンベーシックⅢ	DoJa5.1LE
F884i	らくらくホンプレミアム	DoJa5.1LE
F884iES	らくらくホンⅤ	DoJa5.1LE
F-10A	らくらくホン 6	DoJa5.1LE
F-09B	らくらくホン 7	DoJa5.1

3.3.2 通信方式の選定

携帯が通常の通信方式色々、この中に本システムに対する適切な通信方式を選択する。下表は一般的な通信方式とらくらくホンの対応状況である。

表 3.2 通信方式と特徴

通信方式	端末対応	価格
Bluetooth 通信	非対応	
パケット通信	対応	高価
赤外線通信	対応	安価

しかし、使用する携帯電話はらくらくホンであるため、それに対応していて、かつ安価に実現できる赤外線通信を採用した。

赤外線の通信方式には，IrDA 通信と CIR 通信の 2 種類がある．ここでは各通信方式について，実在する家電と連携させた場合を例に挙げて説明する

表 3.3 赤外線通信の方式と特徴

赤外線通信方式	通信	速度	価格

CIR	単方向	低速	安価
IrDA	双方向	高速	高価

IrDA 通信と CIR 通信の詳細は第 5 章に示す。

3.3.3 家電用マイコンの選定

家電用のマイコンとは、家電側の通信ボードのマイコンである。家電側の通信ボードは携帯と通信機能を付けるので、家電用マイコンの選択もこれを中心にして検討するべき。

(1) 赤外線の通信方式 IrDA と CIR 両方とも付けるマイコンを選択する。

(2) CIR はマイコンの選択に対してあまり厳しくないが、IrDA は IrOBEX まで実装するなら、工数が多い。従って、IrOBEX を通信できるライブラリを利用できるマイコンを選択する。

(3) できるだけ安価なマイコンを選択する。

以上の理由で Microchip 社の PIC24FJ64GA002(本番の環境のマイコン 28 ピン)と PIC24FJ128GA010(試作環境のマイコン 100 ピン)を採用した。

(IrOBEX のライブラリとマイコンの詳細は 5.3 と 6.1.4 に説明である)

3.3.4 実証用家電の選定

本システムは 1 つ家電で限りシステムではなく、家電の汎用モデルである。しかし、最初の実証用家電として、1 つ家電を選択する必要がある。将来汎用しやすいため、この家電は様々の機能を持ち低集積度の家電である方がいいので、集積度が低い家電・イワタニ IOC-125(おまかせミールさん)を選択した。

図 3-2　おまかせミールさん

おまかせミールさんではトースト、卵焼き、コーヒーの調理が可能である。卵焼きとトーストは最大 2 枚まで設定でき、そして焼き色も選べる。選べられる焼き色は淡、普通、濃の 3 種類のみだが、細かく焼き加減を調整したい場合は調理時間の指定もできる。複数のメニューを選択して調理することもでき、この場合同時に仕上がるよう逆算してそれぞれの調理が開始される。時間設定、アラーム設定と予約機能も付いている。

1 つ家電で以上の多い機能を持ちは選択する理由である。

(おまかせミールさんの内部構造は 7.2 に説明がある)

3.4　　全体設計

本段落はシステムの要件によって、利用者の立場から総体的なユースケース図を書いた。

3.4.1　　総体的なユースケース図

本システムは 2 つのサブシステムとして構成する。携帯側のシステムと家電側のシステムである。

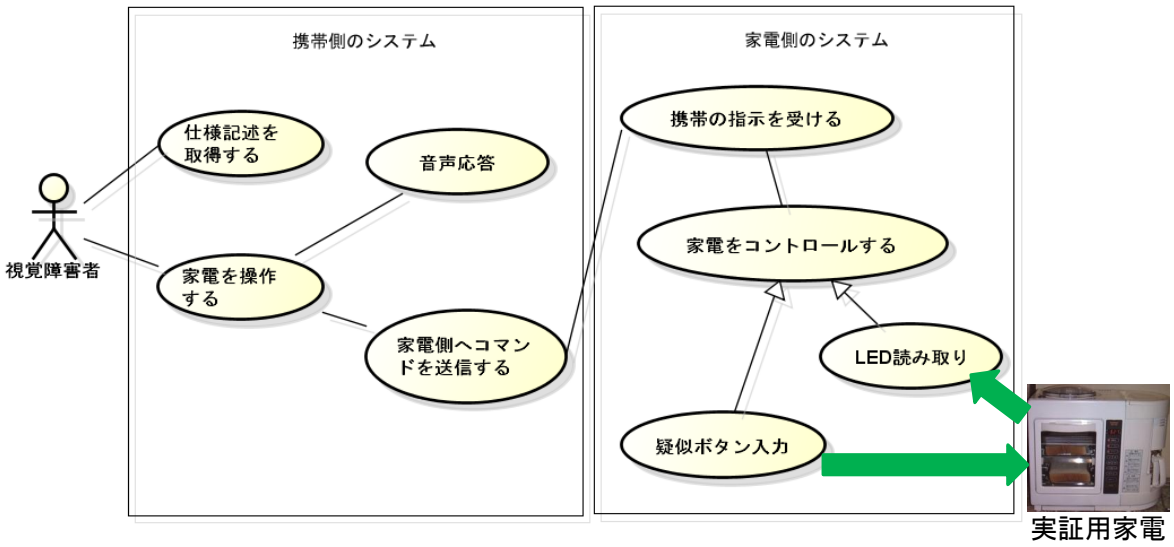


図 3-3　　総体的なユースケース図

本システムのユーザが視覚障害者に想定して、上図を書いた。ユーザは携帯に家電の仕様を取得して、携帯に家電を操作する命令を指示する。携帯側は家電側に命令を送信しながら、音声応答をする。家電側は命令を受け取った、家電をコントロールする。

3.5　　開発体制

本段落で全体的な仕事分担と筆者の担当を図として示す。

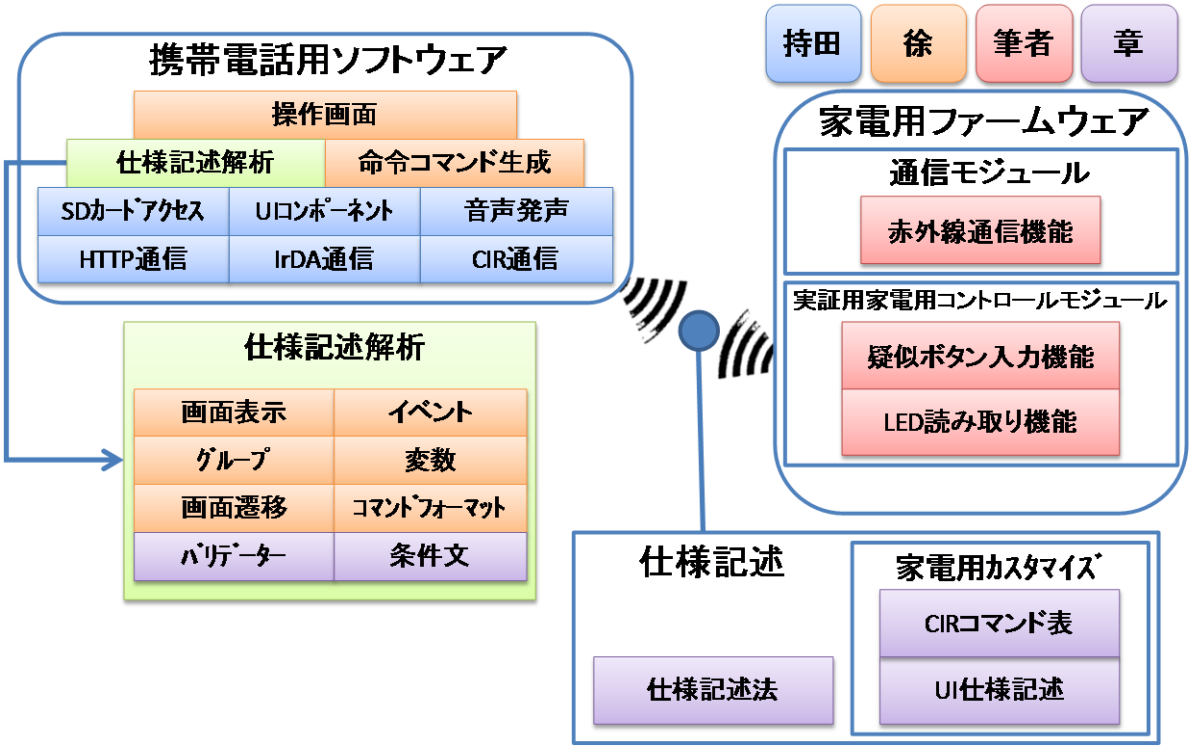


図 3-4　仕事分担と筆者の担当

家電側に搭載する家電用通信モジュールと実証用家電のコントロールモジュールを実装することが筆者の担当である。

実証用家電のコントロールモジュールに関しては、カスタマイズのために開発することである。

3.6　プロジェクトマネージメント

システム開発の品質を高めるため、本システムはプロジェクトマネージメントを実施している。

3.6.1　プロジェクトマネージメントの役割分担

本プロジェクトの開発チームは図 3-4 の 4 つチームメンバーがいる。チーム名は「Omakase」である。プロジェクトマネージメントの役割分担以下である。

チームリーダ：　持田(プロジェクトの総体的な管理)
サブリーダ：　徐(チームリーダを補佐する)
議事録係：　章(会議の内容を記録する)
タイムキーパー：　筆者(会議の時間とプロジェクトの進捗の管理)

3.6.2　プロジェクトマネージメントの方針

うちのチームのプロジェクトマネージメントは以下の方針で実施している。

(1) 週 2 回に会議(2 時間程度)をする。(課題担当教員も参加する)半週間の仕事内容と進捗を報告する、技術の問題と進捗の問題をチーム内で検討する。
(2) 週 2 回コアタイム(5 時間程度)を設定する。チームメンバーがお互いに相談しやすいように、一緒に作業する時間とする。

(3) 週 2 回に報告書を提出する。(会議の日の前に提出する)半週間の仕事内容のまとめである。会議の時、チームに発表する。

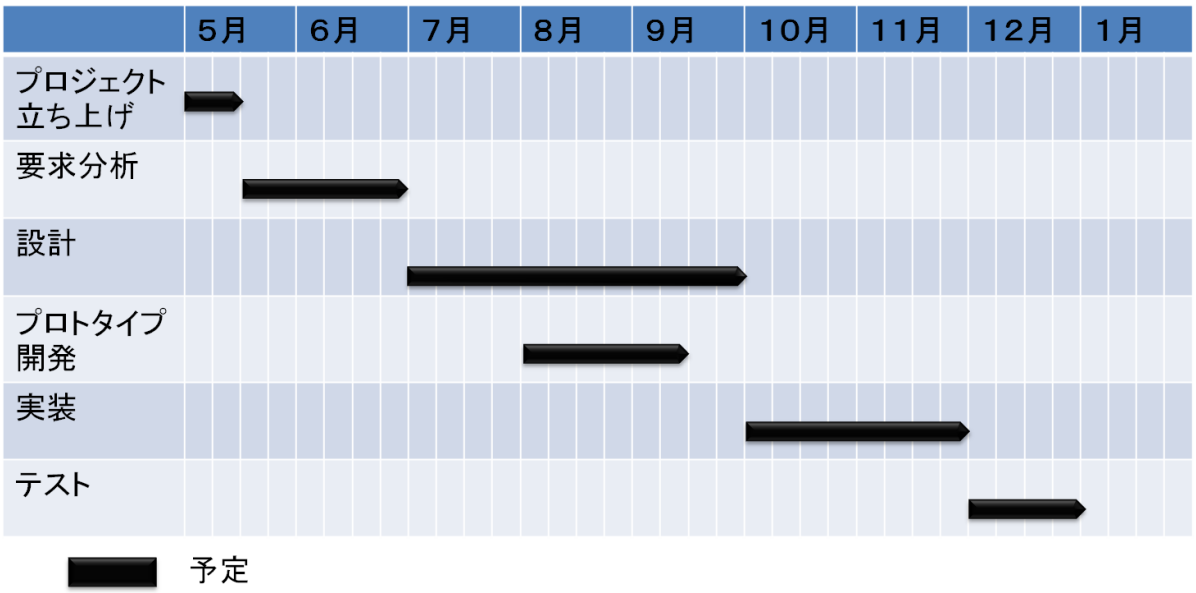
(4) Dropbox でチーム内に情報を共有する。

プロジェクトマネージメントを実施した効果について第 9 章のまとめで揃う。

3.7 スケジュール

本システムは主に 2 つ子システムで分ける。携帯側のシステムと家電側のシステムである。携帯側のシステムはチームの他の 3 人が担当する、下表は携帯側のシステムの開発スケジュールである。

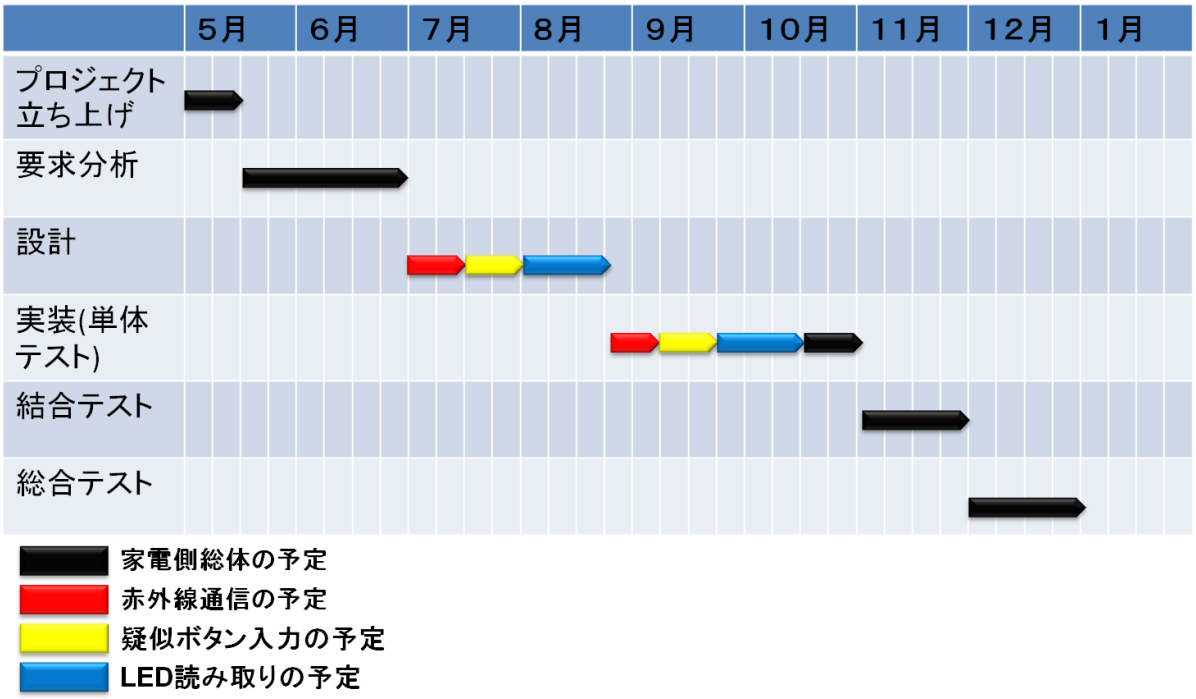
表 3.4 携帯側の開発スケジュール



上表で予定のスケジュールと実績のスケジュールを比べる。

筆者は担当する家電側のシステムの開発スケジュールは下表で示す。

表 3.5　家電側の開発スケジュール



家電側のスケジュールは設計の部分から、3 部分で分ける。(赤外線通信、疑似ボタン入力、LED 読み取り)

第 4 章から家電側のスケジュールによって、家電側のシステムを実装する。

第4章　　家電側のソフトウェア設計

本章で家電側のシステムのソフトウェアの設計を説明する。本システムは C 言語(文献[10][11] [12])で開発するが、できるだけオブジェクト指向(文献[13])の設計思想で設計する。さらに説明しやすいように UML(文献[14])も使用した。

4.1　　家電側のユースケース

ソフトウェアの設計はユースケース図を書くことから始める。下図は家電側のユースケースである。携帯側システムの内容は図 3-3 に書いてある。

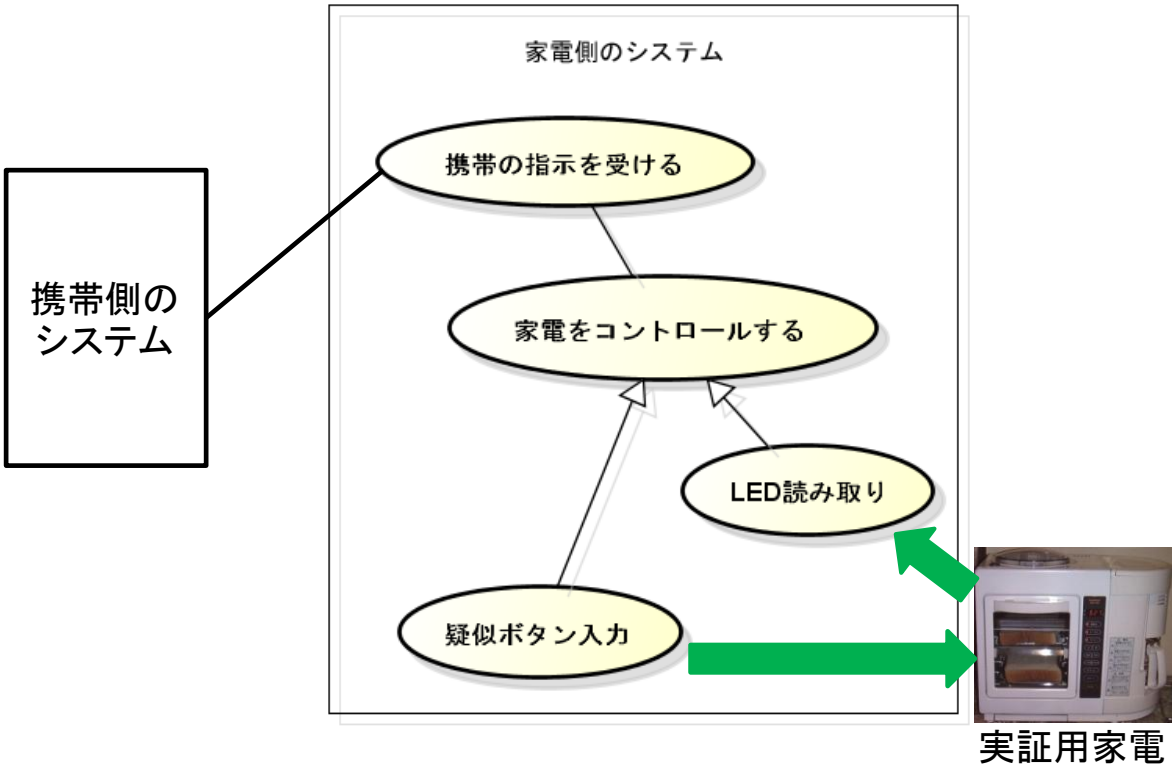


図 4-1　　家電側のユースケース図

家電側のシステムは主に 2 つの機能がある。赤外線通信を通して携帯の指示を受ける機能と家電コントロール機能である。(実証用家電に対しては、家電コントロール機能が LED 読み取りと疑似ボタン入力で分ける)

4.2　　家電側のクラス

厳格的に言うと、C 言語はクラス図を書けない。しかし、UML の図で表示できると、人に納得しやすい。従って、各機能はクラスとして設計するが、実装上はクラスの記述能力をもたない C 言語の範囲で記述する。

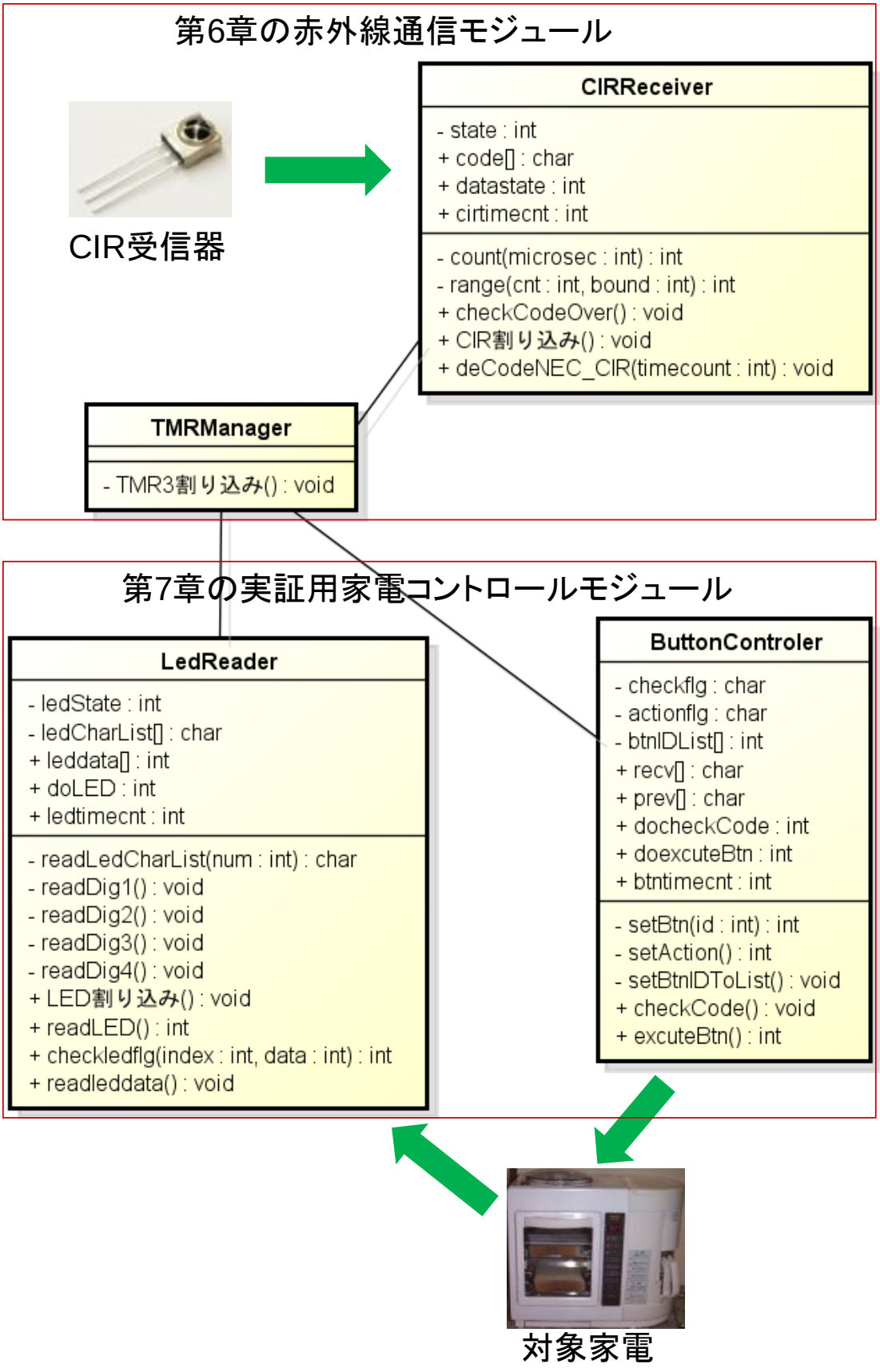


図 4-2　クラス図

家電側のクラスは主に 4 つがある。

TMRManger：家電側の各機能の時間を計る。
CIRReceiver：CIR 受信器が受信した赤外線信号を解析する。
ButtonControler：家電をコントロールする。
LedReader：家電の LED 状態を読み取る。

4.3 家電側のシーケンス

家電側のクラスの間のやり取りは本段落の 3 つシーケンスで説明する。

4.3.1 タイマーシーケンス

クラス図で見ると、TMRManger は他の 3 つクラスと全部に関連があるので、下のシーケンス図で TMRManger と他のクラスのやり取りを示す。

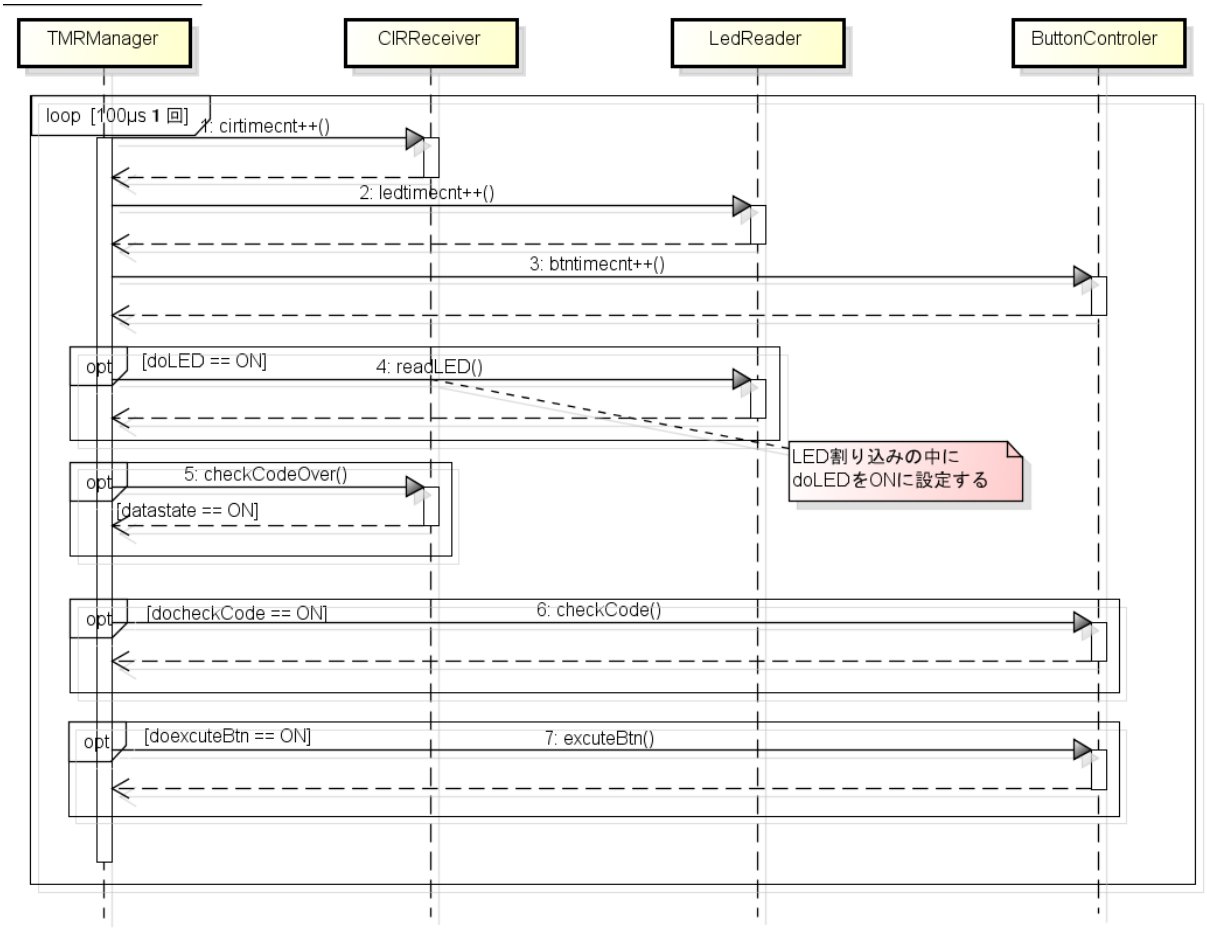


図 4-3 タイマーシーケンス図

本シーケンス 100μs に 1 回起動する。シーケンスの中に各処理の起動フラグ(doLED、datastate、docheckCode、doexcuteBtn)をチェックして、起動するべき処理を起動する。

4.3.2 CIR 受信シーケンス

CIR 受信処理のシーケンスは下図で示す。

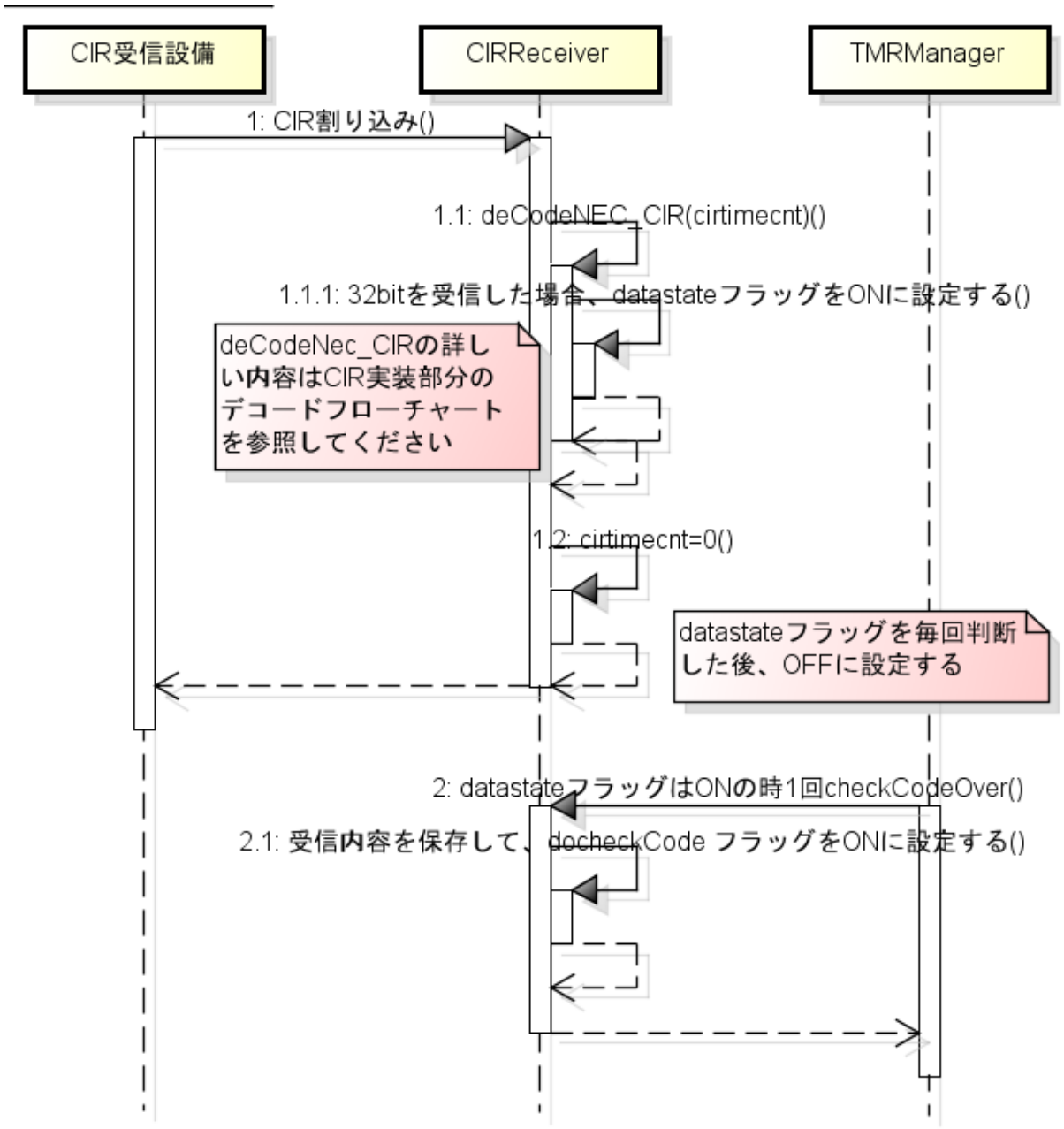


図 4-4　CIR 受信シーケンス図

本シーケンスは毎回 CIR 割り込み(6.2.6 で説明する)がかかる時に起動する。本シシーケンスの実装の詳細は 6.3.5 で説明する。

4.3.3　ボタンコントロールシーケンス図

ボタンコントロールシーケンスは下図で示す。

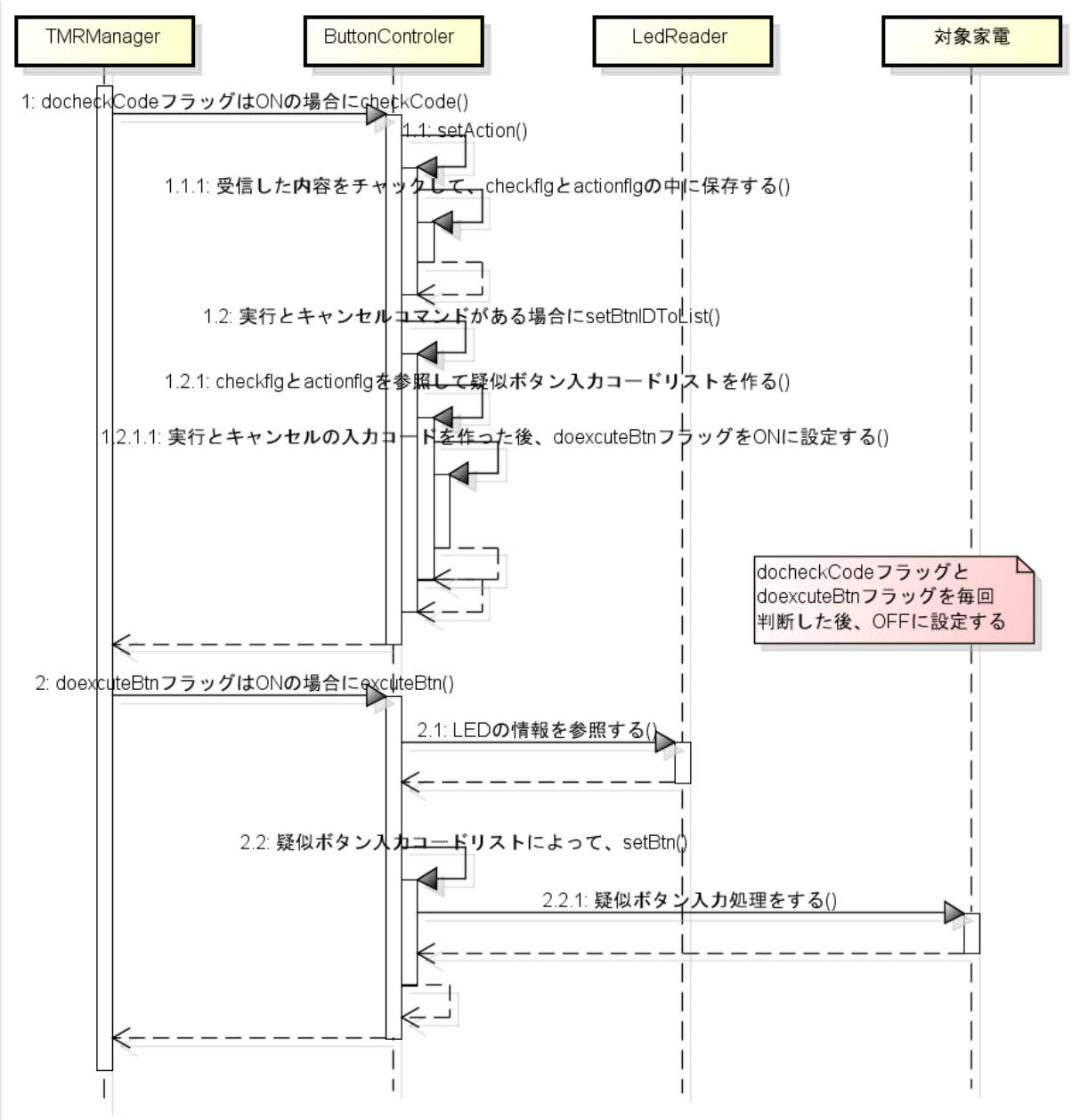


図 4-5　ボタンコントロールシーケンス図

本シーケンスはタイマーシーケンスの起動フラグ(docheckCode)が ON の場合に起動する。本シーケンスは主に疑似ボタン入力と LED 読み取りをする。(詳しい内容は 7.7 に説明がある)

第5章　赤外線通信の実装方式の検討

赤外線は IR (infrared) と略称される。電波より波長が短く、可視光線の赤色より波長が長い電磁波のことである。ヒトの目では見ることができない光である。

5.1　通信原理

送信モジュールは電気信号を光信号に変換して送信する、受信モジュールは受信した光信号を電気信号に復元する。送受信する光信号はあり/なしの 2 状態である。

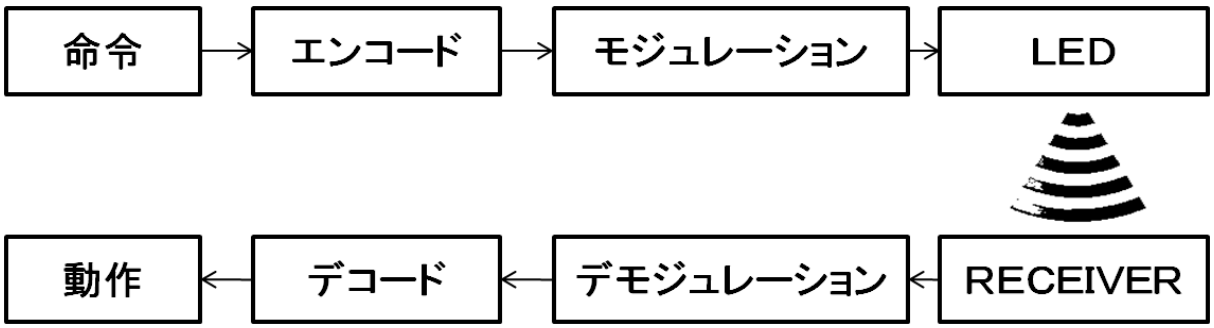


図 5-1　赤外線通信の原理

携帯電話が対応している赤外線通信方式には主に IrDA と CIR がある。

IrDA は Infrared Data Association によって策定された幾つかの赤外線通信プロトコルの総称である。本システムでドコモの携帯「らくらくホン」が対応する DoJa5.1LE プロファイルでは IrDA の通信プロトコルとして IrOBEX(Infrared Object Exchange)をサポートしている([15]の com.nttdocomo.io.ObexConnection を参照した)。

CIR とは異なり IrDA は双方向の赤外線通信プロトコルである。

5.2　IrOBEX の実装方式の検討

IrOBEX の実装方式には複数の方法があるため先に検討する。

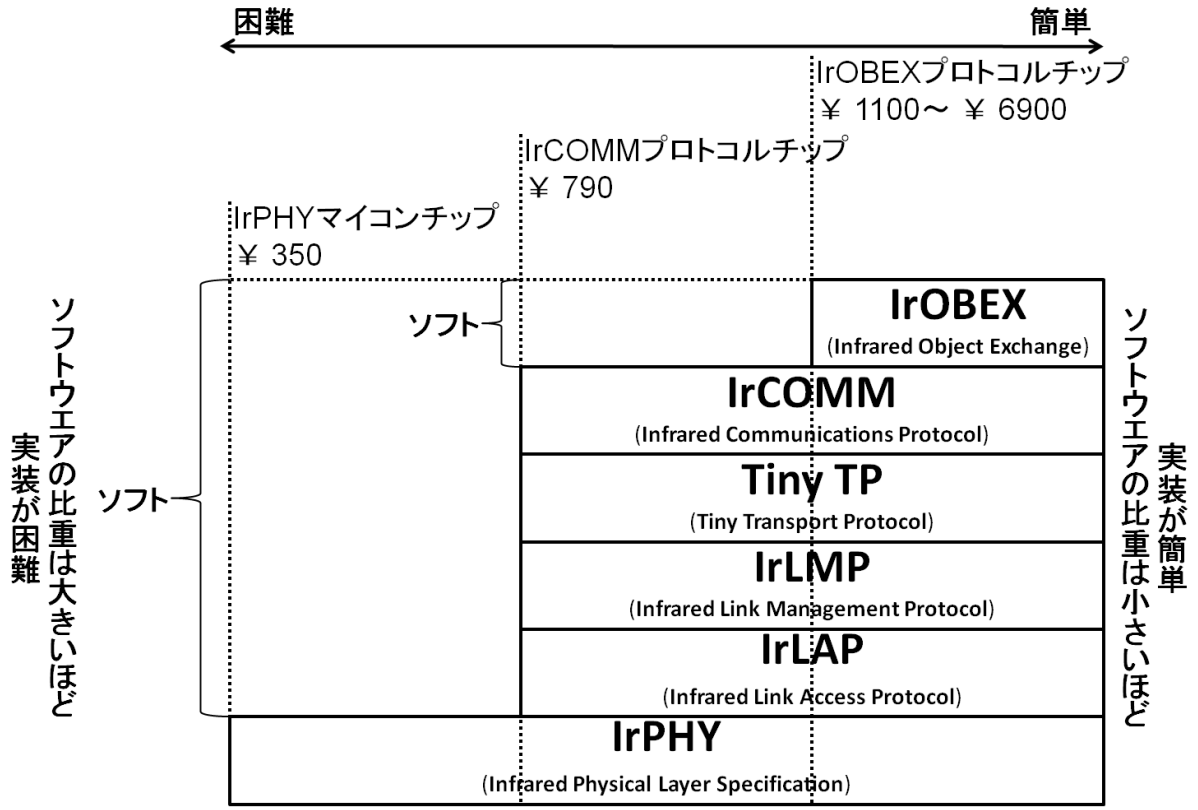


図 5-2　IrOBEX の実装方式

(上図の構成は文献[16]の IrDA 規格を参照した)できるだけ安価に実現することが目標なので、一番安い選択肢である IrPHY 内蔵マイコンを採用し、IrLAP 以上を全てソフトウェア実装する方法が望ましい。従って、IrPHY 内蔵マイコンを第 1 の選択肢とする。

ただし、この選択肢で開発することは、リスクが高いし、工数も大きい。スキルの問題などで実装できなかった場合はコストが上がっても実現することを優先して、第 2・第 3 の選択肢に切り替えることにする。

5.3　試作環境のハードウェアの選択

プロトコルスタックの実装は工数が大きいので、ライブラリを探した、Microchip Standard IrDA Stack(文献[17]を参照する)は PIC24、dsPIC33、PIC32 の 3 種類のマイコンをサポートするライブラリである。本システムはこのライブラリが対応している PIC24 および dsPIC33 マイコンの評価ボードである Explorer16(DV164037)と IrDA PICtail Plus(AC164124)を試作環境として採用する。

Explorer16 ボードには PIC24FJ128GA010 と dsPIC33F256GP710 の 2 種類が付属している。IrDA Standard Stack のマニュアル文献[18]によると、OBEX に必要となるリソースと PIC24FJ128GA010 の持つリソースは下の表の通りである。

表 5.1　サンプルプログラムとマイコンのリソース

プログラム	要求ROM	要求RAM
OBEXServer	23.2KB～23.5KB	1.85KB
OBEXPeer	33.7KB～34.1KB	2.15KB
マイコン	ROM	RAM
PIC24FJ128GA010	128KB	8KB

このことから、安価な PIC24F で充分と考えられるので、PIC24F を採用する。

5.4　ボード側の開発ツール

Explorer16 を使うにあたって開発ツール MPLAB IDE、Explorer16 とパソコンの通信アダプターMPLAB ICD3 のインストールを行った(文献[19]を参照する)。下図は Explorer16 と MPLAB ICD3 の写真である。

図 5-3　開発環境 Explorer16

C 言語をコンパイルするために、MPLAB IDE の Microchip Toolsuite で language toolsuite の指定を行った(C Compiler、Object Linker、Assembler、Archiver) 。

試作環境は採用するコンパイラが COFF(Executable and Linkable Format)と ELF(Common Object File Format)の両方のライブラリフォーマットをサポートしている。

IrDA OBEX の基本的な通信用 API はライブラリ「Microchip¥IrDA¥libIrDA_PIC24F_OP-elf.a」或いは「Microchip¥IrDA¥libIrDA_PIC24F_OP-coff.a」を参照することで使用可能となる。

第6章　家電用赤外線通信モジュールの開発

本章では第 5 章で検討した家電用赤外線通信モジュールの開発について説明する。

6.1　IrDA による赤外線通信の実装

IrDA 通信は双方向通信なので、低コストで実装できれば、優先的に実装するつもり。

6.1.1　IrDA OBEX 通信用 API

IrDA の通信モードを 2 つ持っている。
(1)プライマリモード(Primary mode はクライアント)
(2)セカンダリモード(Secondary mode はサーバ)
IrDA OBEX の通信処理は API「IrDA_SendOBEX」と「IrDA_ReceiveOBEX」を使える。

6.1.2　電話帳オブジェクトの送受信

携帯と Explorer16 の間で任意のデータを送受信することが送受信テストの目標である。しかし、携帯と Explorer16 は双方自作プログラムでの送受信になるので、エラーが出る場合、どちらの原因か判断しにくい。

そこでまずは信頼できる通信相手として、携帯が標準で対応している電話帳オブジェクトの送受信を行う。

電話帳オブジェクトは vCard 文献[20]として定義されている。Microchip Standard IrDA Stack のサンプルプログラムで通信が成功することが確認できた。

6.1.3　ユーザ定義オブジェクトの送受信

ユーザ定義オブジェクトにおいて送受信処理ができるプログラムを作成しテストしたところ通信に失敗した。携帯側とボード側での通信処理のくい違いをつきとめ、ユーザで定義オブジェクトの通信も可能となったが、コネクションを開いたまま、オブジェクトを交換することができなかった。

6.1.4　Microchip Standard IrDA Stack の問題点

問題を追跡していくと、Microchip Standard IrDA Stack が提供しているライブラリには、セカンダリモードのオブジェクト送信リクエストの処理に失敗する、このため、コネクションを開いたままオブジェクト(メッセージ)を交換できないことがわかった。

解決法は 2 つがあるが、両方とも難しいところがある。

解決法 1：家電側がプライマリモードになる。

問題：Microchip IrDA のプライマリモードにはオブジェクト受信機能がなく双方向通信できない。

解決法 2：コネクションを都度切断し、プライマリ/セカンダリを切り換えながら両方向通信を行う。

問題：コネクションを開き直すたびに携帯に警告が出る。

つまり、Microchip Standard IrDA Stack が提供しているライブラリの赤外線通信機能は本システムには適用できないことがわかった。

(1)メーカーへの障害報告

(2)オープンソースの IrOBEX Stack の調査

(3)IrOBEX Protocol Chip の調査

(4)CIR 受信ルーチンの先行実装

以上の 4 つ仕事を並行して実施することにした。

6.2 CIR による赤外線通信の実装

6.1 の IrDA 通信は短期間で実装できないので、家電側の CIR 受信部分を先に実装する。

6.2.1 CIR とは

CIR(Consumer InfraRed)はテレビのリモコンや色々な家電製品のリモコンとして最も幅広く応用されている単方向の赤外線通信プロトコルである。

本システムは NEC フォーマットの赤外線変調方式を使うものとする。さらに、NEC フォーマットがユーザに開放されている CIR フォーマットであるので、携帯電話と通信ボードの CIR 通信は NEC フォーマットを採用する。

NEC フォーマットの CIR 受信プログラムを開発するにあたり、以下赤外線変調方式と NEC フォーマットについて説明する。

6.2.2 モジュレーション(変調)

赤外線送信は PPM 変調したビット列をキャリア変調して送信される(キャリア変調と PPM 変調の概念は文献[21]を参照した)。

(1) PPM(Pulse-Position Modulation)変調
PPM 変調はキャリア変調の 2 状態のアウトプット (あり・なし)で状態の変換と保持する時間の変換を組み合わせて、ビット列を送信できるように、存在する変調方式である。

(2) キャリア変調
キャリア変調は環境光の影響を排除することに対して有効な方式である。
キャリア変調においては、「あり」は、特定の周波数での点滅であり、「なし」は真に消灯した状態である。

世の中の赤外線通信フォーマットはほとんど全てキャリア変調と PPM 変調をかけて定義してきた。NEC フォーマットもこの中の 1 つである、本システムの CIR 送受信はユーザに開放された NEC フォーマットを採用するので、簡単に紹介する。

6.2.3 NEC フォーマット

NEC フォーマットの Frame は Leader code、Custom Code、Data Code の 3 つ部分で構成されている。

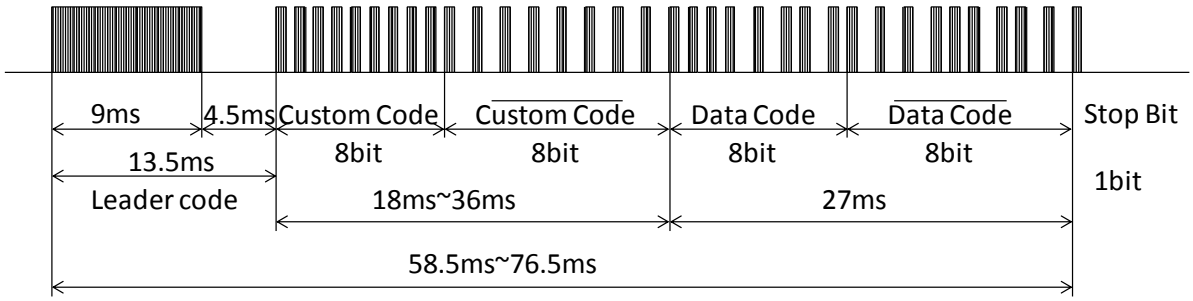


図 6-1　Frame の構成

(文献[22]を参照してキャリア変調もかけるように修正した)

Leader code、Custom Code、Data Code など各部分の意味と長さは下表で説明する。

表 6.1　各部分の意味

コード名	意味	サイズ	時間(マイクロ秒)
Leader code	開始フラグ	1bit	ON(9000) ⇒OFF (4500)
Custom Code	メーカー情報	8bit	0のビット ON(560) ⇒OFF (560) 1のビット ON(560) ⇒OFF (1690)
$\overline{\text{Custom Code}}$	メーカー情報反転	8bit	
Data Code	コマンドコード	8bit	
$\overline{\text{Data Code}}$	コマンドコード反転	8bit	
Stop Bit	終了フラグ	1bit	ON(560) ⇒OFF (残り時間)

Custom Code と Data Code は通信したいデータ部分である。
Frame をデコードする時、信号の状態と長さで 1 と 0 を区別できる。

表 6.2　送信情報 1 と 0 の定義

データ内容	サイズ	時間(マイクロ秒)
0	1bit	ON(560) ⇒OFF (560)
1	1bit	ON(560) ⇒OFF (1690)

560 マイクロ秒の ON 信号の次、560 マイクロ秒の OFF 信号の場合に 0 を送信したい。OFF 信号は 1690 マイクロ秒の場合は 1 である。下図はキャリア変調した後のロジック 1 と 0 を示す。

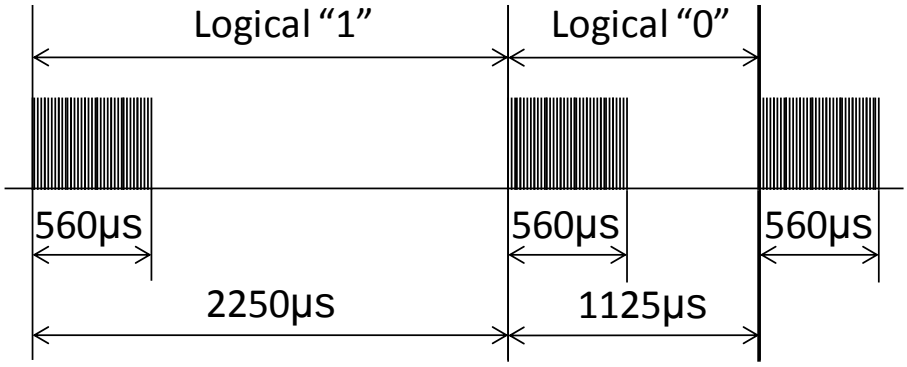


図 6-2　キャリア変調した後のロジック１と０

（文献[22]を参照してキャリア変調もかけるように修正した）

上図の ON の状態はキャリア変調した様子である。

6.2.4　ボード側ＣＩＲ受信のハードウェア

通信ボード本番環境:PIC24FJXXGA002 Microchip の 28 ピンマイコン。製品のソースコードのメモリサイズによって、マイコンのメモリサイズを決める。下図は 64MB の PIC24FJ64GA002 の写真である。

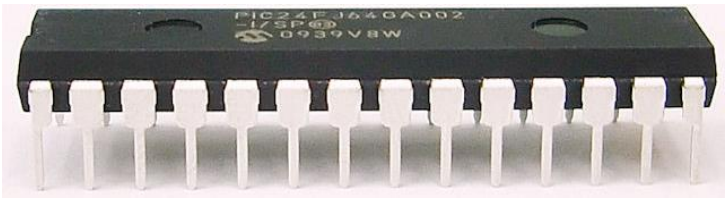


図 6-3　PIC24FJ64GA002

下図は文献[23]のマイコン PIC24FJXXGA002 のピンアサインである。

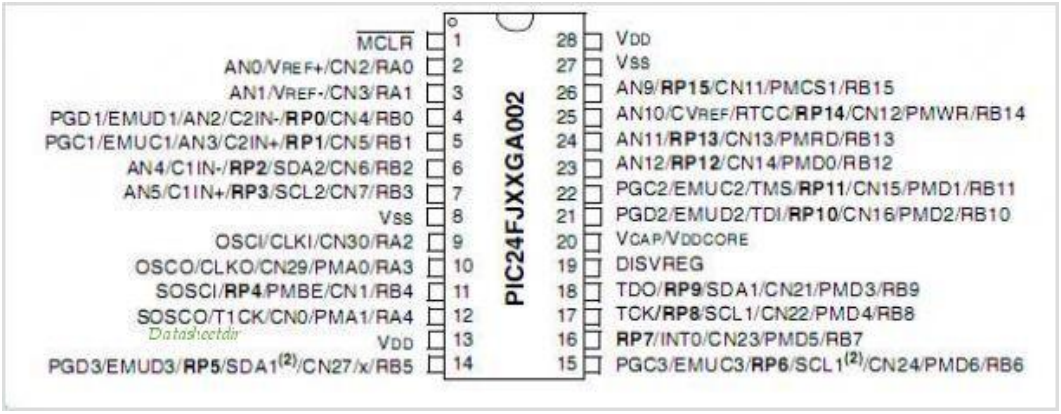


図 6-4　PIC24FJXXGA002(28 ピン版)のピンアサイン

（文献[23]の PIC24FJXXGA002 ピンアサイン）

通信ボードの試作は: PIC24FJ128GA010 Microchip の 100 ピンマイコン/Explorer16 ボード/Proto Expansion Board で行っている。下図は 128MB の PIC24FJ128GA010 の写真である。

図 6-5　PIC24FJ120GA010

下図は文献[24]のマイコン PIC24FJXXGA010 と PIC24FJXXXGA010 のピンアサインである。

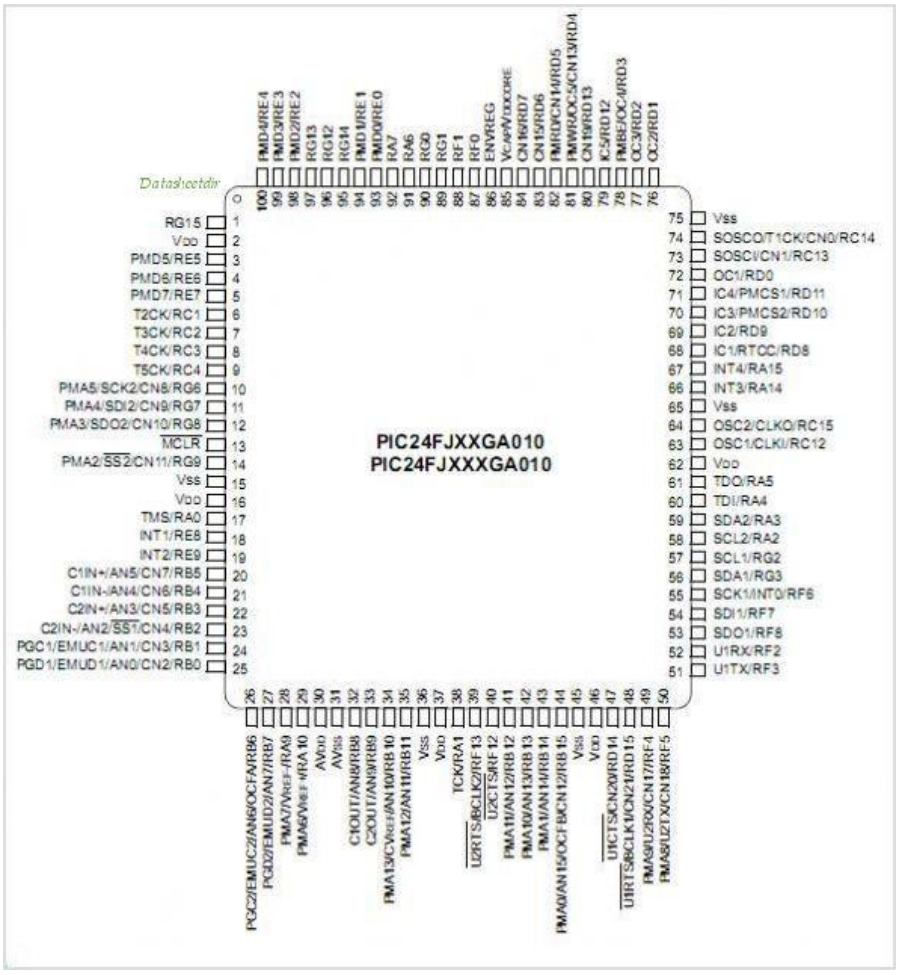


図 6-6　PIC24FJXXGA010 と PIC24FJXXXGA010 (100 ピン版)のピンアサイン

(文献[24]の PIC24FJXXGA010 と PIC24FJXXXGA010 ピンアサイン)

赤外線受信機能を実装する時に注意するべきことは、完成品で使用するマイコンに付いていないハードウェアリソースを使わないようにすることである。例えば、PIC24FJ128GA010 にはあるが PIC24FJ64GA002 にはないピンを使うと、将来 PIC24FJ64GA002 で実行することが難しくなる。(この注意点は赤外線受信の本質的な機能に対することである。テストの

ためにログを出すなどの本番のボードが使わない機能の実現はテスト環境特有のリソースを使っても問題ない)

6.2.5 PPM 変調のデコード

NEC フォーマットの CIR 通信は一つ要素がある、それは毎回の状態変換の時間である。

Leader code、Custom Code、Data Code、Stop Bit、さらに“1”と“0”の定義も時間の長さがある。つまり ON 信号と OFF 信号の時間を計って、受けた信号をデコードする。

タイマーで信号を計るために入力キャプチャを用いる(タイマーと入力キャプチャの概念と使い方は文献[25] [26])。

NEC フォーマットの場合具体的なデコード方法は以下の各 Frame のデコードフローチャートを描いた(本システムのフローチャートの書き方は文献[27]を参照した)。

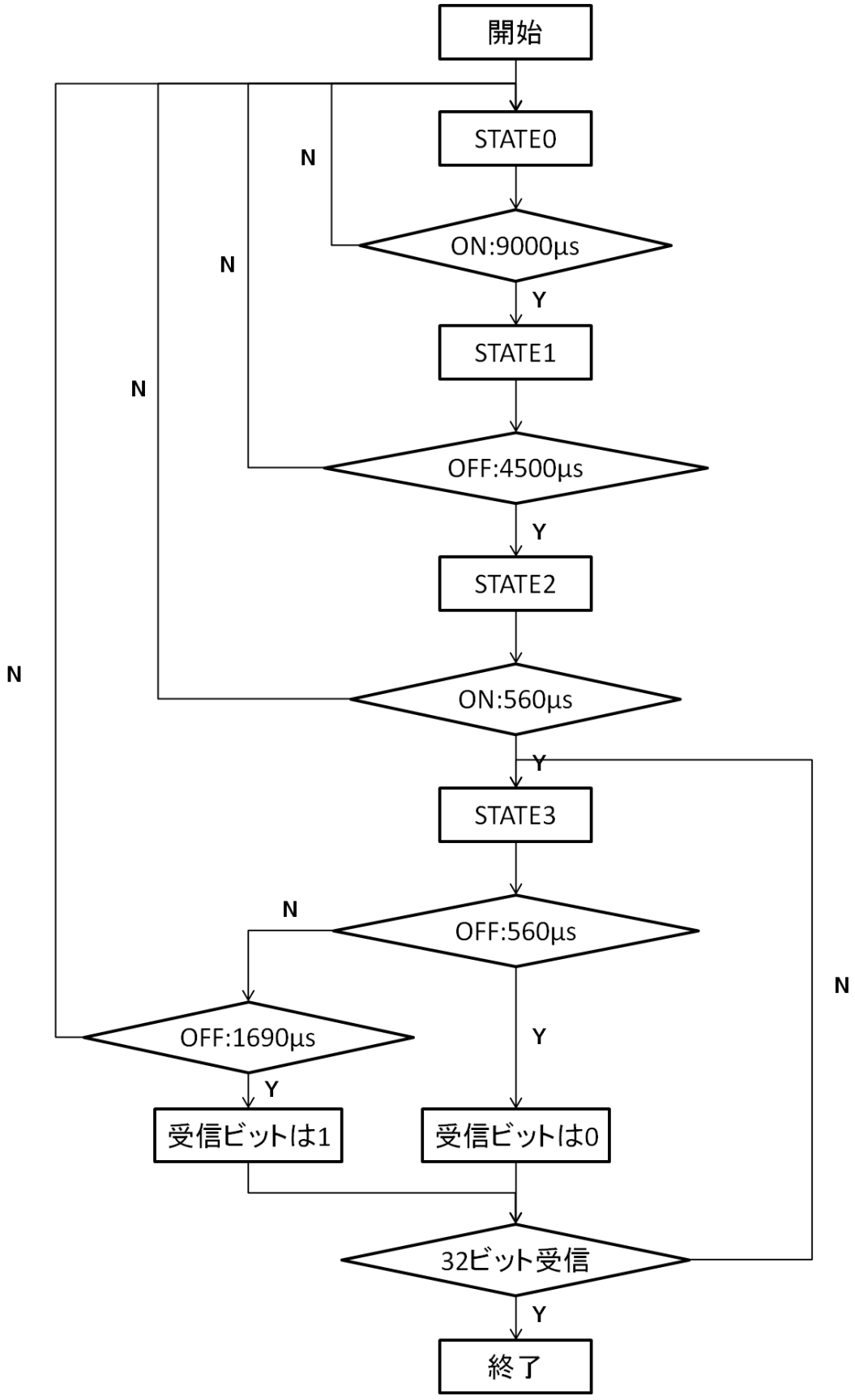


図 6-7　デコードフローチャート

タイマーで測った時間は毎回 IC1(IC は入力キャプチャの意味、本段落の下に説明がある)

の入力値を変化した時から次の変化の前までの時間である。この時間と NEC フォーマットで定義した各状態の時間に比べて、現時点の通信状態と受けるデータ情報を確認できる。

または上図の方法で各 Frame 毎の 32bit の情報を受信できる。

タイマー(TMR)：タイマーとは、一定周期に一回増やすカウンタである。使いたいタイマーのレジスターbit に開始 FLAG を設定すると、時間の計測が始まる。

大部分の PIC24F は 5 つタイマーが持っている、TMR1~5 である。この 5 つタイマーは 3 種類で分ける、TypeA、TypeB、TypeC である。TypeA は普通な 16bit のタイマーである。1 つ TypeB タイマー(16bit)と 1 つ TypeC タイマー(16bit)を組み合わせると、1 つ 32bit のタイマーになれる。(PIC24F のタイマーの種類は文献[28]を参照した)

PIC24F の 5 つタイマーの種類は以下である。

TMR1 ⇒ TypeA
TMR2 ⇒ TypeB
TMR3 ⇒ TypeC
TMR4 ⇒ TypeB
TMR5 ⇒ TypeC

TMR2 と TMR3 を組み合わせる、TMR4 と TMR5 を組み合わせる。

PIC24F の IrDA Stack は TMR2 を使っているので、TMR3 は 16bit タイマーとしてしか使わない。(文献[18]を参照した)

PIC24F の頻度と TMR1~5 のカウント頻度も調べて赤外線受信処理用タイマーは TMR1 と TMR3 のどちらでも使用できる。(TMR4 と TMR5 は 32bit のタイマーになれるので、将来他の機能は使用する可能性がある。)

PIC24FJXXGA002 の頻度は最大 8MHZ を設定できるが､デフォルト値は 4MHZ である。([23]を参照した)PIC24FJXXGA002 の頻度はデフォルトの時に TMR1~5 のカウント頻度は 4 つ選択肢である。

1:256 ⇒ 1 秒/(4 MHZ /256) = 1000000μs/(4000000/256) = 64μs
1:64 ⇒ 1 秒/(4 MHZ /64) = 1000000μs/(4000000/64) = 16μs
1:8 ⇒ 1 秒/(4 MHZ /8) = 1000000μs/(4000000/8) = 2μs
1:1 ⇒ 1 秒/(4 MHZ /1) = 1000000μs/(4000000/1) = 0.25μs

NEC フォーマットの赤外線受信する時に計りたい時間の最大値は NEC フォーマットの 1Frame の最大時間 108000μs([22]を参照した)である｡頻度の 4 つ選択肢のどちらで 9000μs を計れることについて検討する。

108000μs/64μs = 1687.5 < 16bit の最大カウント数(65536 回)
108000μs/16μs = 6750 < 16bit の最大カウント数(65536 回)
108000μs/2μs = 54000 < 16bit の最大カウント数(65536 回)
108000μs/0.25μs = 432000 > 16bit の最大カウント数(65536 回)

これで、16bit カウントで 108000μs を計れる 3 つ選択肢の中に一番早いタイマーのカウント頻度の選択肢 1:8 を選んだ。

入力キャプチャ(Input Capture)：入力キャプチャとは、フリーランでカウントする TMR の値をキャプチャ入力ピンの変化により自動的に読み取り、キャプチャレジスターに転記する機能である。

最終的に使用する PIC24FJXXGA002 では入力キャプチャは IC0、IC1、IC2 の 3 チャンネルが存在している｡入力キャプチャと連携して使えるタイマーは TMR2 と TMR3 である。TMR2 は IrDA Stack が使うので、TMR3 しか使えない。

6.2.6　　ボード側ＣＩＲ受信のソフトウェア

テストする時マイコンのピンを使いやすいように、Prototyping Extension Board を利用する。Explorer16 に Prototyping Extension Board を付けて、Prototyping Extension Board で作業して、Explorer16 のピンをコントロールできる。

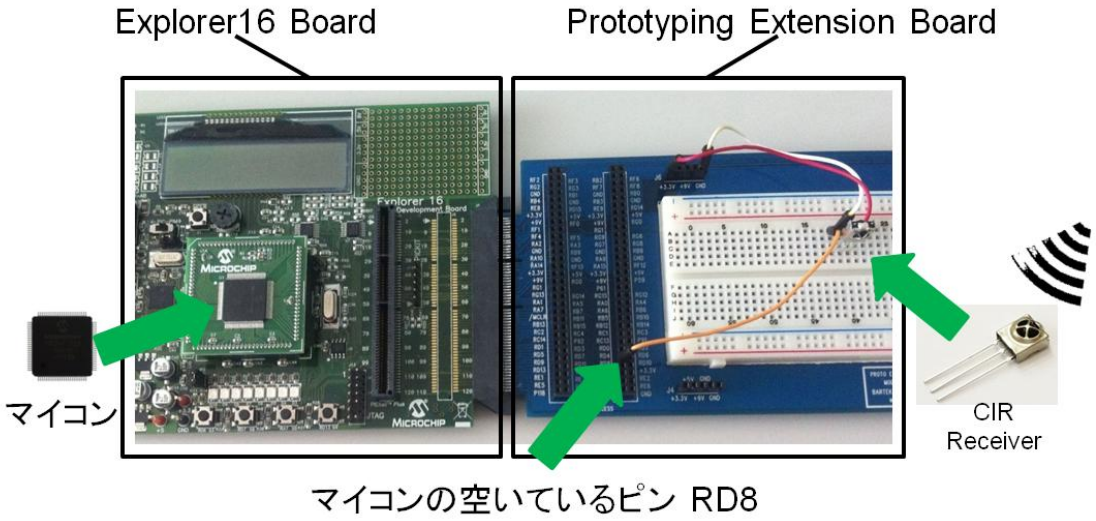


図 6-8　CIR 受信テスト

Prototyping Extension Board の上に CIR Receiver を乗せて CIR Receiver の出力信号をマイコンの空いているピンに設定する。

RD8 は IC1(入力キャプチャ 1)と同じピンを使っているので、RD8 に信号を設定すると、IC1 の状態も変化できる。IC1 の状態が変化すると、IC1 の割り込み関数呼び出せられる。

IC1 の割り込み関数の中に毎回 RD8 の入力値が変化するまでの時間を TMR で計って、CIR 受信の基本の部分を実現してくる。

割り込み関数 : 割り込み関数とは、マイコンの入力キャプチャ(読み書きできるメモリアドレス)を利用して、入力キャプチャの(1・0)状態を毎回変化する時、割り込み関数を呼び出せる。

ここまで赤外線通信モジュールの CIR 受信の部分を実装できた。

6.2.7　　赤外線通信モジュールのテスト

赤外線通信モジュールを実装した後、次の実装をやる前に、赤外線通信モジュール関するテストする必要がある。この部分のテストの詳細は 8.2.1 に説明がある。

6.2.8　　新規家電への組み込み

本章では PIC24FJ128GA010 を用いて通信モジュールを実装した。使用リソースは入力ピン 1 本と 16bit タイマー1ch であり、CIR 受信プロクラムのステップ数は 100 ぐらいであった。

CIR は省リソースで実装でき、家電ですでに多く使われている赤外線通信の方式でもあることから、メーカーが新規家電を設計する場合は、本章で開発したソフトウェアがその家電の CPU に内蔵可能であると考えられる。

第7章　　家電音声化システム・家電操作の実証

本章では家電音声化システムの動作や操作性を実際の家電で検証したい。家電音声化システムを検証用家電に組み込むための追加の実装について説明する。

7.1　　家電用コントロールモジュール

前章で実装した家電音声化システムで家電を制御すうための方法 2 つである。素直なモデルと試作モデルである。

7.1.1　メーカー用素直なモデル

素直なモデルは下図のような形である。

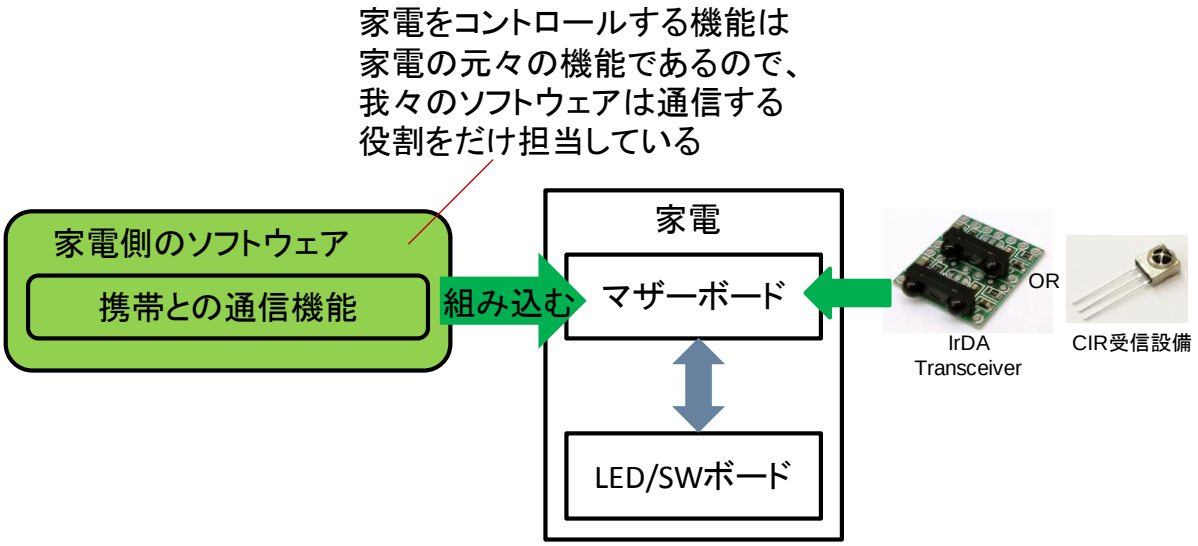


図 7-1　　素直なモデル

メーカーは本システムを採用する時、このモデルを採用できるが、我々は家電の製造者ではなく CPU のソフトウェアを改造できない。そこで、今回は 7.1.2 のように試作モデルによる実装した。

7.1.2　試作モデル

下図は試作モデルの実装方式である。

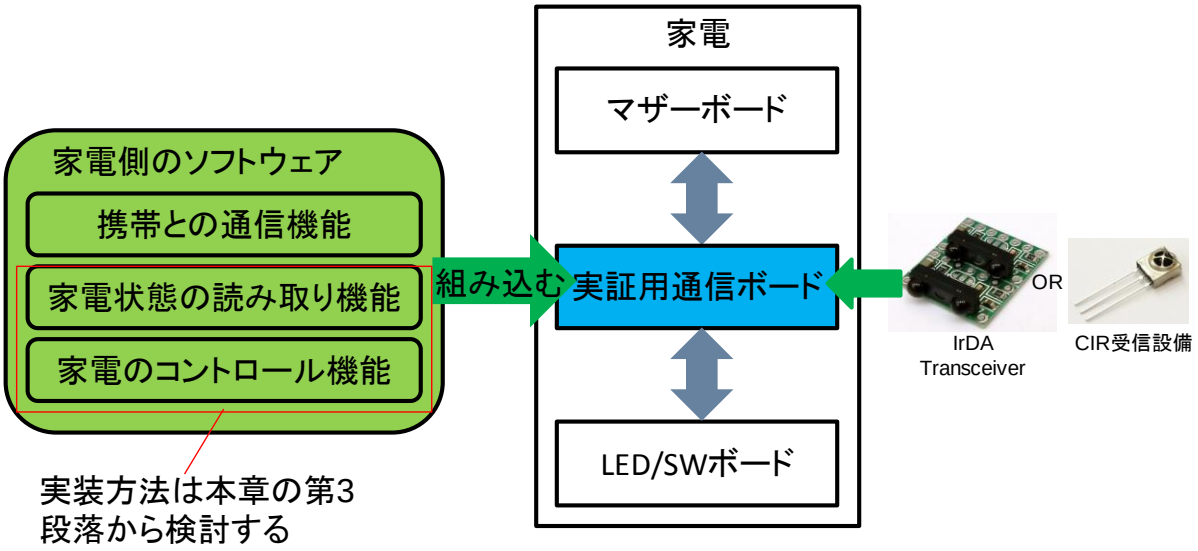


図 7-2　試作モデル

家電の信号線を解析し、表示の読み取りと疑似ボタン入力により、家電の制御を行う。

7.2　実証用家電(おまかせミールさん)について

目玉焼きとトーストとコーヒーの調理機能をもつイワタニ IOC-125(おまかせミールさん)を携帯電話からの制御対象とする。実証用家電であるおまかせミールさんには赤外線送受信器が付いておらず、基板上に赤外線送受信器を付ける場所もない。しかし、携帯の命令を受信するために、赤外線送受信器を乗せる基板が必要である。通信ボードの実装方針を下図に示す。

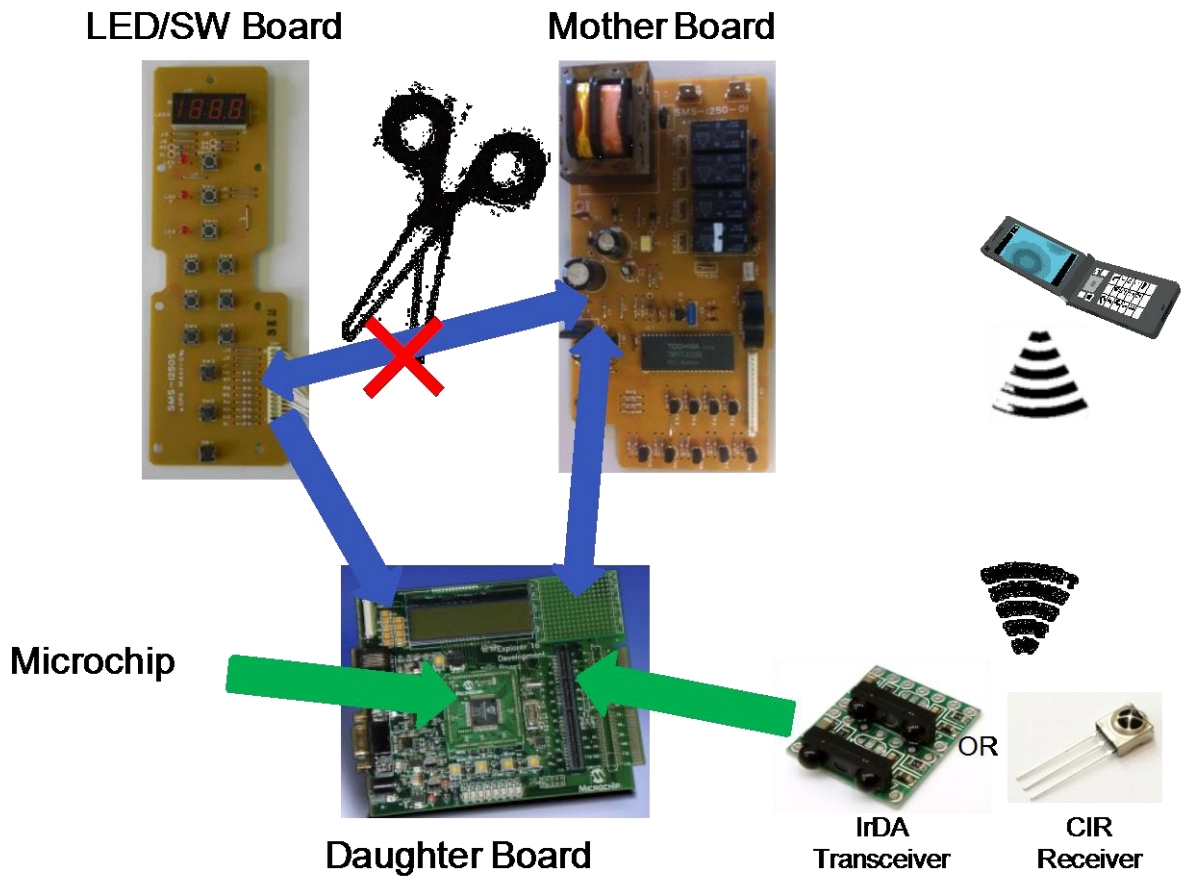


図 7-3　通信ボードの実装方針

実証用家電はマザーボードと LED/SW ボードが 15 芯のフラットケーブルで結線されている。これを赤外線で制御するために、マザーボードと LED/SW ボードの間に独自の通信ボードを挟んで解決する。通信ボードにはマザーボードから LED/SW ボードへの表示信号を読み取って外部に赤外線送信する能力と、外部からの赤外線信号を受信して LED/SW ボードのボタンが押されたかのような信号をマザーボードに送信する能力を持たせる。

これからの実装は実証用家電の LED/SW Board と関係が緊密であるので、LED/SW Board を先に解析する。

7.2.1　実証用家電の LED/SW Board

チーム全員一緒に解析(付録の「基板の解析」に説明がある)して、実証用家電 LED/SW Board の回路図を書いた。

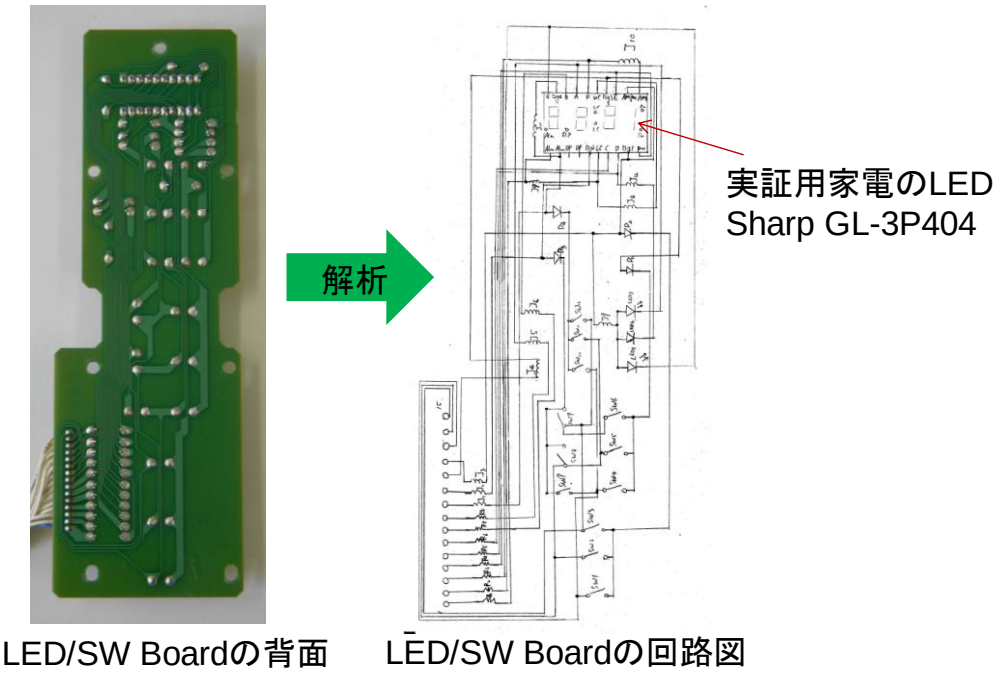


図 7-4　解析した LED/SW Board

LED/SW Board の回路図の詳細は付録の「LED/SW Board の回路図」に示す。この LED/SW Board は 12 個スイッチと LED(Sharp の GL-3P404)を持っている。

文献[29]を参照した GL-3P404 の略図を示す。

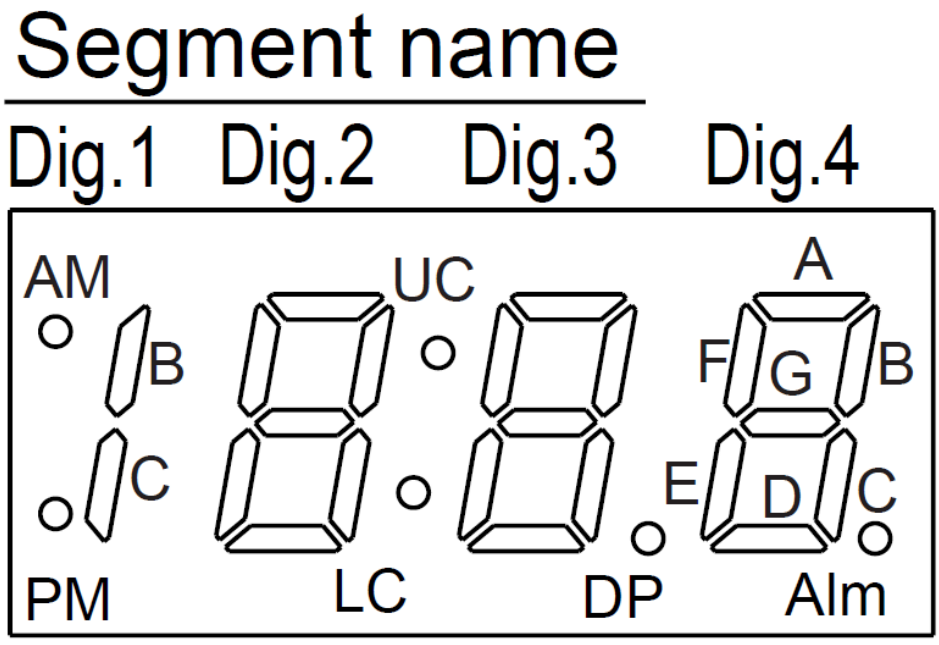


図 7-5　GL-3P404 の略図

(文献[29]の GL-3P404)

GL-3P404 は 29 ユニットを持つ LED である。

スイッチと LED のユニット数で計算して LED/SW Board は 44 個ユニット(12 個スイッチ+29 個 LED ユニット+3 個スイッチ LED)がある。従って、15 芯のフラットケーブルで 44 個ユニットのコントロール信号を転送したいなら、実証用家電はスイッチのダイナミックス

キャンと LED のダイナミック点灯をしているはずである。

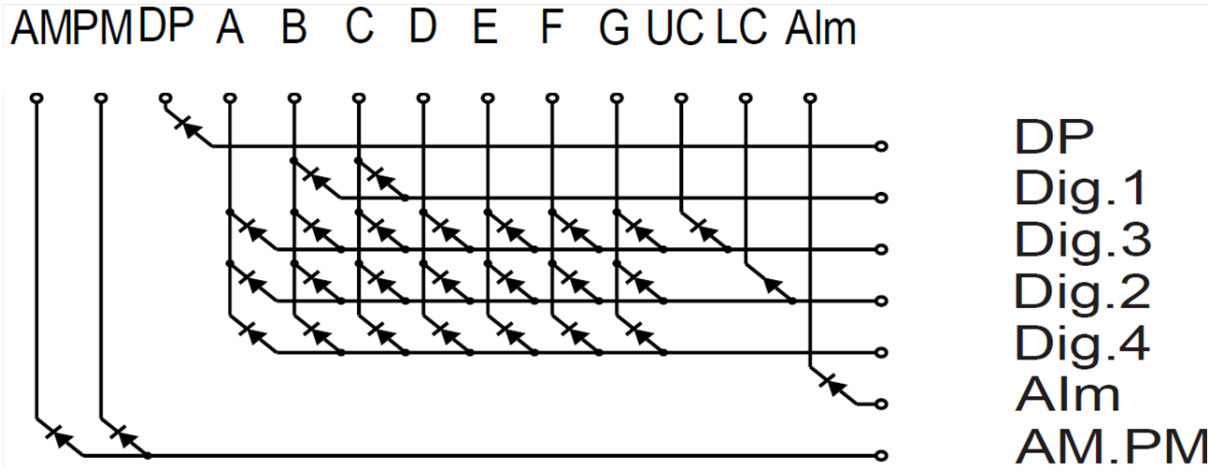


図 7-6　GL-3P404 の Interal connection diagram

(文献[29]の GL-3P404)

上図と付録の「LED/SW Board の回路図」を参照して、以下の結論を得た。

GL-3P404 はアノードコモンである。

フラットケーブルのピン 1~8 は LED 出力である。

フラットケーブルのピン 9~12 はスキャンラインである

フラットケーブルのピン 13~15 はキー入力である。

7.2.2　ダイナミックの調査

ダイナミックスキャンの詳しい内容を確認するため、Oscilloscope(SeeedStudio 社の DSO Quad)で下図のような情報を計った。

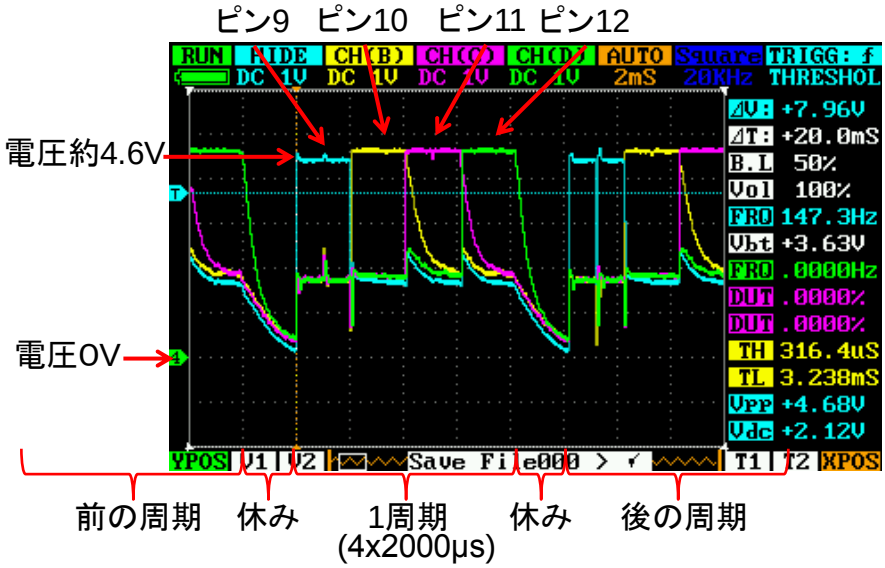


図 7-7　ダイナミックスキャンの周期と電圧

上図はピン 9~12 のダイナミックスキャンの周期と電圧状況である、ピン 9~12 は周期(1 周期 4*2000μs+休み 2000μs)に繰り返して ON に変わる。電圧状況は ON の時が 4.6V であ

る、OFF になると次第に下がる。ダイナミックスキャンの周期と電圧状況も家電用コントロールモジュールを実装する時に役立つである。

下図は実証用家電のピン 9~12 とピン 13~15 を活用してダイナミックスキャンで 12 個スイッチをコントロールする概要である。

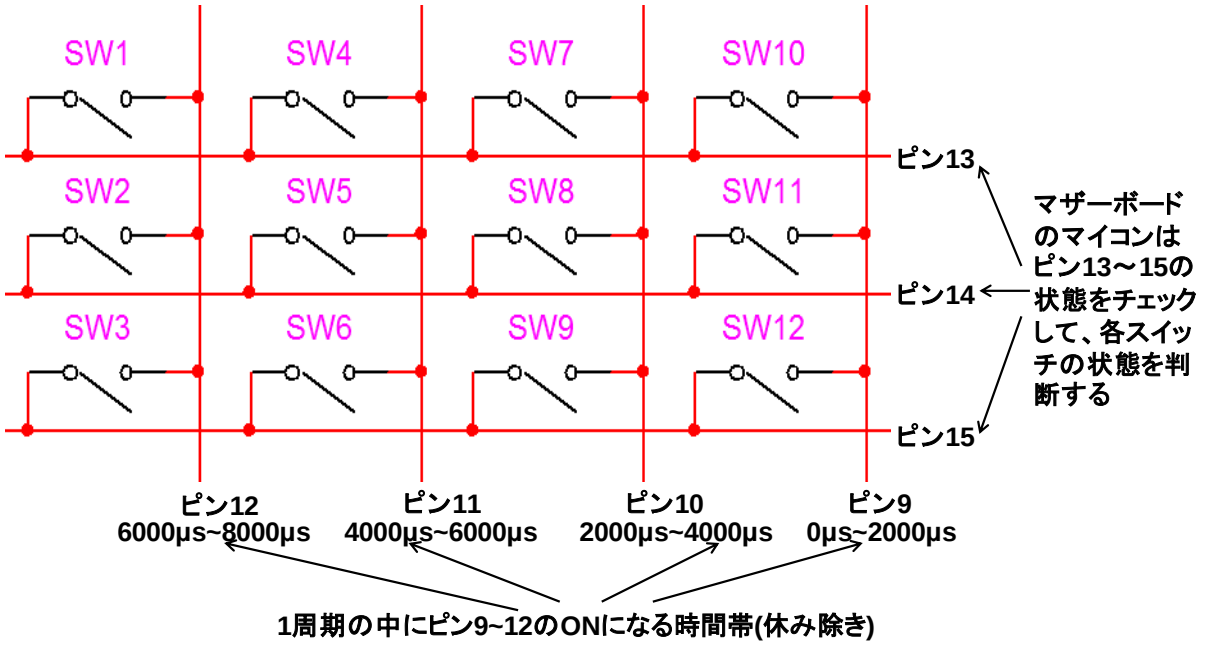


図 7-8　ダイナミックスキャンの概要

上図を見ると、例えばピン 10 は ON になる時間帯に SW8 を押す時、ピン 14 が ON になる。または、ピン 11 は ON になる時間帯に SW4 を押す時、ピン 13 が ON になる。これで、ピン 9~12 の各時間帯にピン 13~15 の状態をチェックすると、12 個スイッチがどちらか押下していることを判断できる、これはダイナミックスキャンである。

図 7-7 で見るとダイナミックスキャンの周期は 10000μs である、10000μs は人間に対して一瞬である。ユーザが実証用家電のスイッチを押して離すまでの時間は必ず 10000μs より長い。つまり、人がスイッチを押している間に、ピン 9~12 が少なくとも全て 1 回 ON になるため、スイッチの状態を読み取ることができる。

次はダイナミックスキャンとダイナミック点灯(7.5 に説明がある)の場合に家電用コントロールモジュールの試作モデルの実装を説明する。

7.3　ボタンの入力のエミュレーション

Explorer16 Board はフラットケーブルを通して LED/SW Board のボタン操作と同じ信号をマザーボードに送って、人のボタンを押す動作に代わりに実証用家電のボタンの入力のエミュレーションをする。

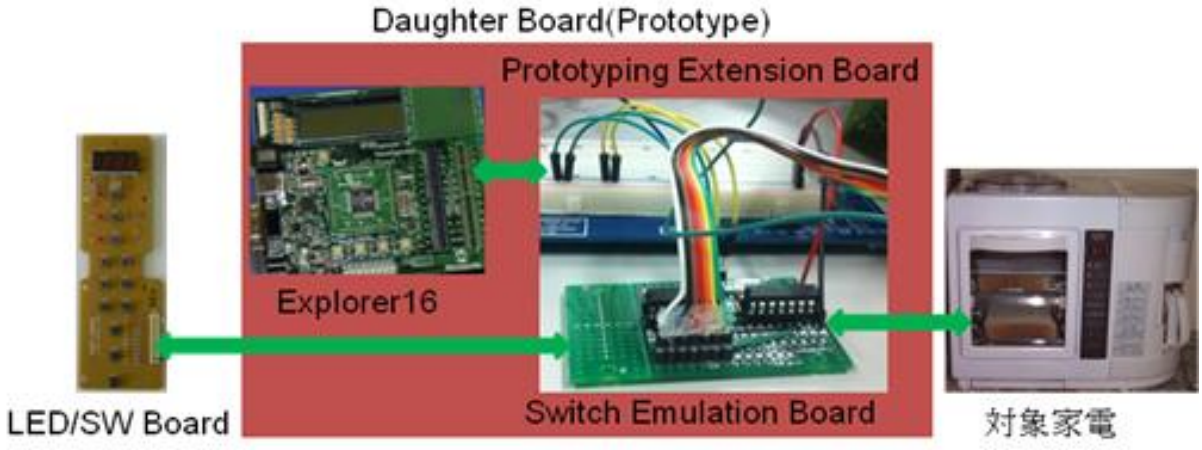


図 7-9　ハードウェア

今回のテスト環境に関して、Explorer16 は実証用家電の LED/SW Board に直接につけることできない。 Prototyping Extension Board と Switch Emulation Board を通して実証用家電の LED/SW Board をコントロールする。

7.3.1　Switch Emulation Board

ダイナミックスキャンの調査結果で見ると、ピン 9~12 は自動的に ON になることである。従って、ピン 9~12 の該当時間帯に操作したスイッチと繋がっているピン 13~15 のいずれか ON の電圧をかけると、入力のエミュレーションを実現できる。しかし、毎回入力のエミュレーションの時、家電の CPU のスキャンに同期して外部の CPU から信号を入力することは困難を伴う。非同期にキー入力を与えるためにキーと並列にスイッチング素子を設計の方法では 12 個のトランジスタが必要とするため現実的ではない。そのため、アナログスイッチを用いた Switch Emulation Board を利用した。

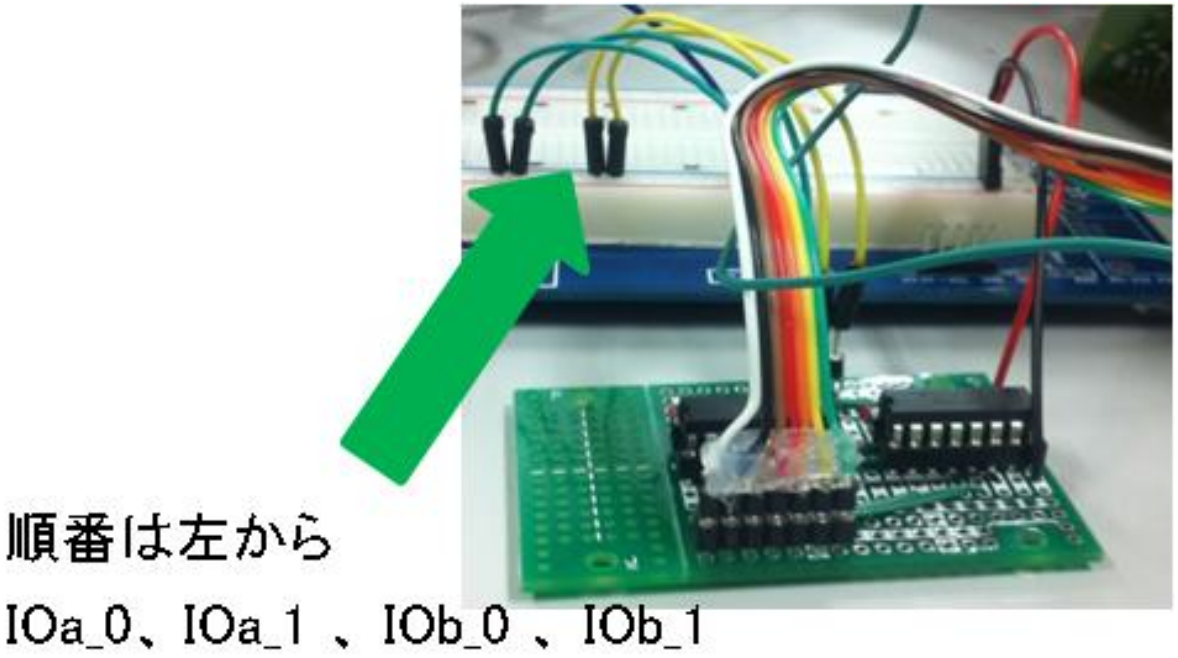


図 7-10　Switch Emulation Board

この Switch Emulation Board を使って 4 つピンで 12 個以上(最大 16 個)の状態を出力で

きる。

次はこのボードの構成を説明する。

7.3.2 Analog Multiplexers の活用

Switch Emulation Board の上に 2 つ Analog Multiplexers を乗せる。74HCT4051(文献[30]を参照した)を採用した。74HCT4051 は S0、S1、S2 の 3 つ入力により YN を Y0～Y7 のピンに接続できる SP8T(Single Pole 8 Throws)スイッチである。

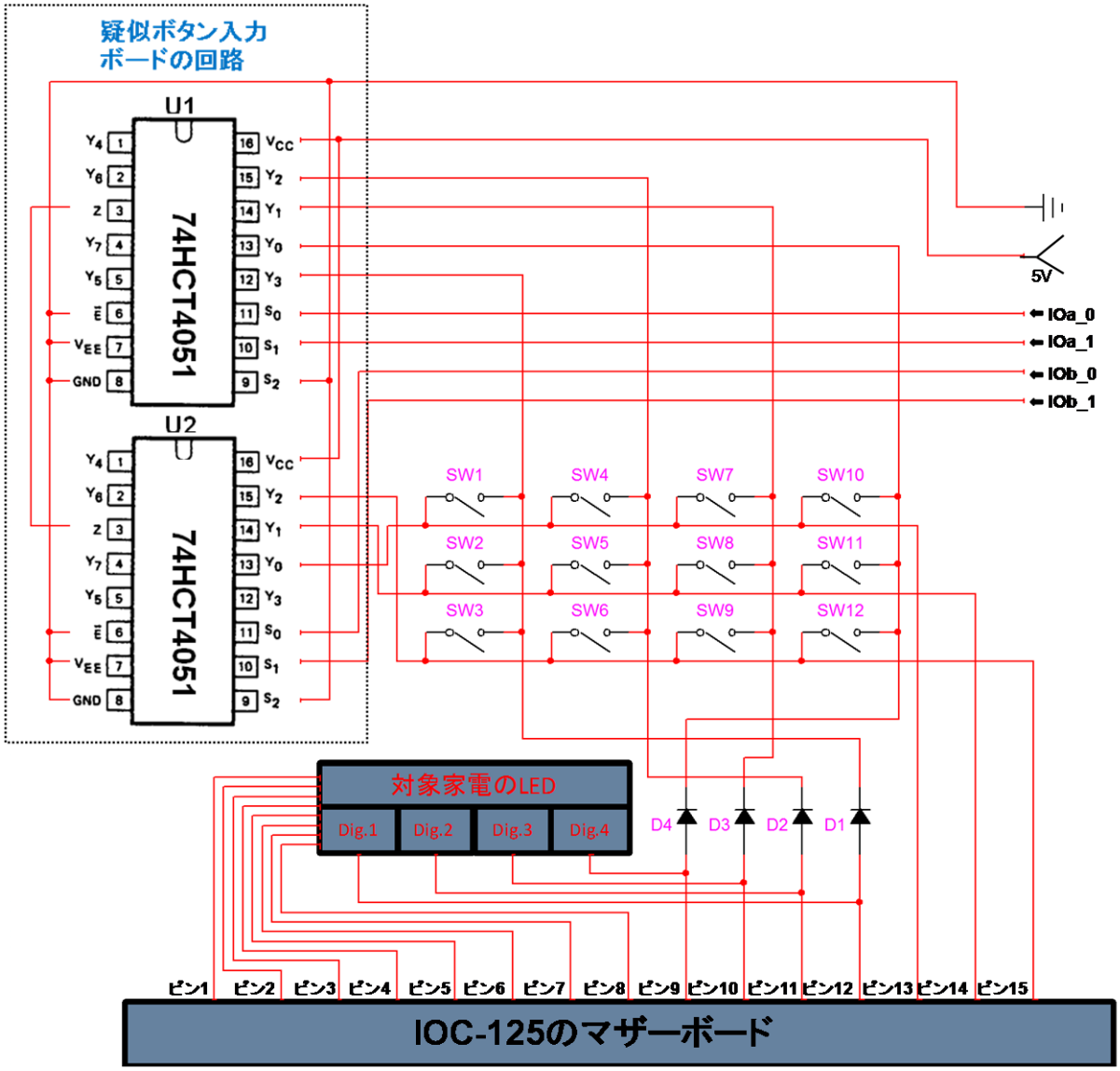


図 7-11 Analog Multiplexers でボタン制御

実証用家電のピンと実証用家電 LED の詳しい接続図 7-13 に示す。

上図のような形で(IOa_0、IOa_1 、IOb_0 、IOb_1)4 つピンの入力で 12 個のボタンをコントロールできる。例えば、U1 の B0 と U2 の B2 が ON の時は SW1 を押された場合である。

U1 と U2 の S2 は今回の実証用家電で使用しない、常に 0 を入力する。さらに複雑な機能がある家電を制御する時、S2 の入力も利用すると、最大 8*8=64 個出力状態をコントロールできる。

実証用家電のボタンをうまくコントロールできるように Switch Emulation Board を制御するソフトウェアを通信ボード上に実装した。

7.3.3 ボタン入力エミュレーションソフトウェアの概要

Explorer16 から電圧(ON⇒1、OFF⇒0)を入力して、ボタンの入力のエミュレーションを実現する。

IOa_0、IOa_1 、IOb_0 、IOb_1 の 4 つ線の各自の 2 つ状態を組み合わせて、2x2x2x2=16 状態までに出力できる。

下表は 4 つピンの入力とコントロールボタンの機能の対照表である。

表 7.1 入力ピンとボタン対照表

IOb・IOa	00	01	10	11
00	ボタンを離す	—	—	—
01	コーヒー(SW10)	10分(SW4)	時(SW7)	取り消し(SW1)
10	トースト(SW11)	予約(SW5)	時計(SW8)	スタート(SW2)
11	卵焼き(SW12)	濃(SW6)	淡(SW9)	アラーム(SW3)

横は IOa_0 と IOa_1 の組み合わせである。

縦は IOb_0 と IOb_1 の組み合わせである。

7.3.4 ボタン入力エミュレーションソフトウェアの実装

Explorer16 の 4 つ空いている出力 Port を選択して、Oa_0、IOa_1 、IOb_0 、IOb_1 に入力する。Explorer16 の出力をソフトウェアでコントロールして、ボタン入力のコントロール目的を達成する。

受信する命令をエミュレーションでボタンの操作のかわりをする時、ソフトウェアの実装に対して、注意するところがある。

(1)連続的な幾つかのボタン操作命令の場合、次のボタン操作のエミュレーションをする前に、実機（おまかせミールさん）は先のボタンを読み取るまでに、先のボタン動作状態をそのまま保持する必要がある。実機がボタンを読み取った後は、次のボタン操作のエミュレーションをしても大丈夫である。そうしないと、先のボタン操作を読み落とすことが発生するはず。

(2)次のボタンを押す前に、前のボタンを離す必要がある。ボタンを離すコードは 0000 である。

(3) IOa と IOb の 4bit は同時に変化されなければならない、設定する過程に間違った入力値は認識される可能性がある。つまり、4bit を一括設定する。

以上の注意点に従って、エミュレーションソフトウェアのソースを実装した。

7.4 家電 LED の読み取り目的と原理

家電をうまくコントロールするため、家電の状態をはっきり知る必要であるはず。この目的は家電の LED を表示している内容を読み取って判断することで達成できる。

7.4.1　　１つ LED の点灯原理

多 LED ダイナミック点灯原理を説明する前に１つ LED の点灯原理を説明する。

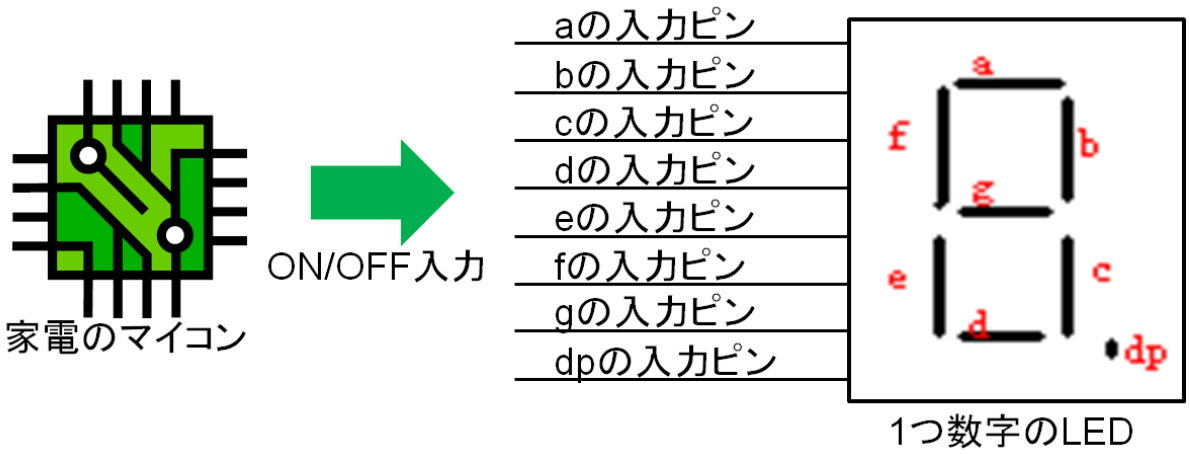


図 7-12　　１つ LED の点灯原理

上図は１つ数字を表示できる LED の点灯原理図である、家電のマイコンは ON と OFF の命令を LED の入力ピンに送って LED の表示をコントロールする。ON と OFF 命令については、ON=High/OFF=Low と ON=Low/OFF=High のような 2 パタンがある、これは実証用家電によって違うことである。

これからの LED 読み取り部分の説明は ON=Low/OFF=High の場合で例として行う。普段の１つ LED の入力に対して１つ 8Bit のマイコンの出力 Port(これから LED 入力 Port と呼ぶ)を用意する。LED 入力 Port は１つ int8 の変数として使うと、LED の点灯の内容をコントロールできる。下表は１つ LED 入力 Port 各 Bit の意味の一例である。

表 7.2　LED 入力 Port

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
a	b	c	d	e	f	g	dp

a~g の順番が逆にする並び方と dp を Bit7 に置く並び方もある。上表の並び方で考えると、ON=Low=0/OFF=High=1 の場合に LED 入力 Port の値と表示数字の関係は下表で示す。

表 7.3　LED 入力 Port の値と表示数字の対照表

表示数字	入力値(Bit7～Bit1)
0	0000001
1	1001111
2	0010010
3	0000110
4	1001100
5	0100100
6	0100000
7	0001111
8	0000000
9	0000100

上表のような数字の以外は、部分のローマ字も表示できる。下表で幾つの例を挙がる。

表 7.4　LED 入力 Port の値と部分ローマ字の対照表

ローマ字	入力値(Bit7～Bit1)
A	0001000
b	1100000
C	0110001
d	1000010
E	0110000
F	0111000
0	数字の 0 と同じ

ここ以上は 1 つ LED の点灯の原理である。しかし、1 つ LED だけ使っている家電が少ない、大部分の家電は LED を使う目的は時間の表示である。時間を表示するため、4 つ LED を同時に使用する家電が多い。今回の実証用家電の LED(図 7-5Sharp の GL-3P404)もこのような形である。

GL-3P404 は 4 つ Digit Segment で構造する LED である。こらから、GL-3P404 の Digit Segment は Dig.1~Dig4 を呼ぶ。次は実証用家電の中に GL-3P404 のダイナミック方式を説明する。

7.4.2　LED/SW ボードと GL-3P404 のダイナミック読み取りの関係

実証用家電は GL-3P404 と直接に繋がる部分が LED/SW ボードである、LED/SW ボードとマザーボードが 15 芯のフラットケーブルで結線されているので。このフラットケーブルの LED をコントロールしているピンの出力値を読むと、GL-3P404 のダイナミックを読み取りできる。フラットケーブルのピンと GL-3P404 の繋がり方を下図で示す。

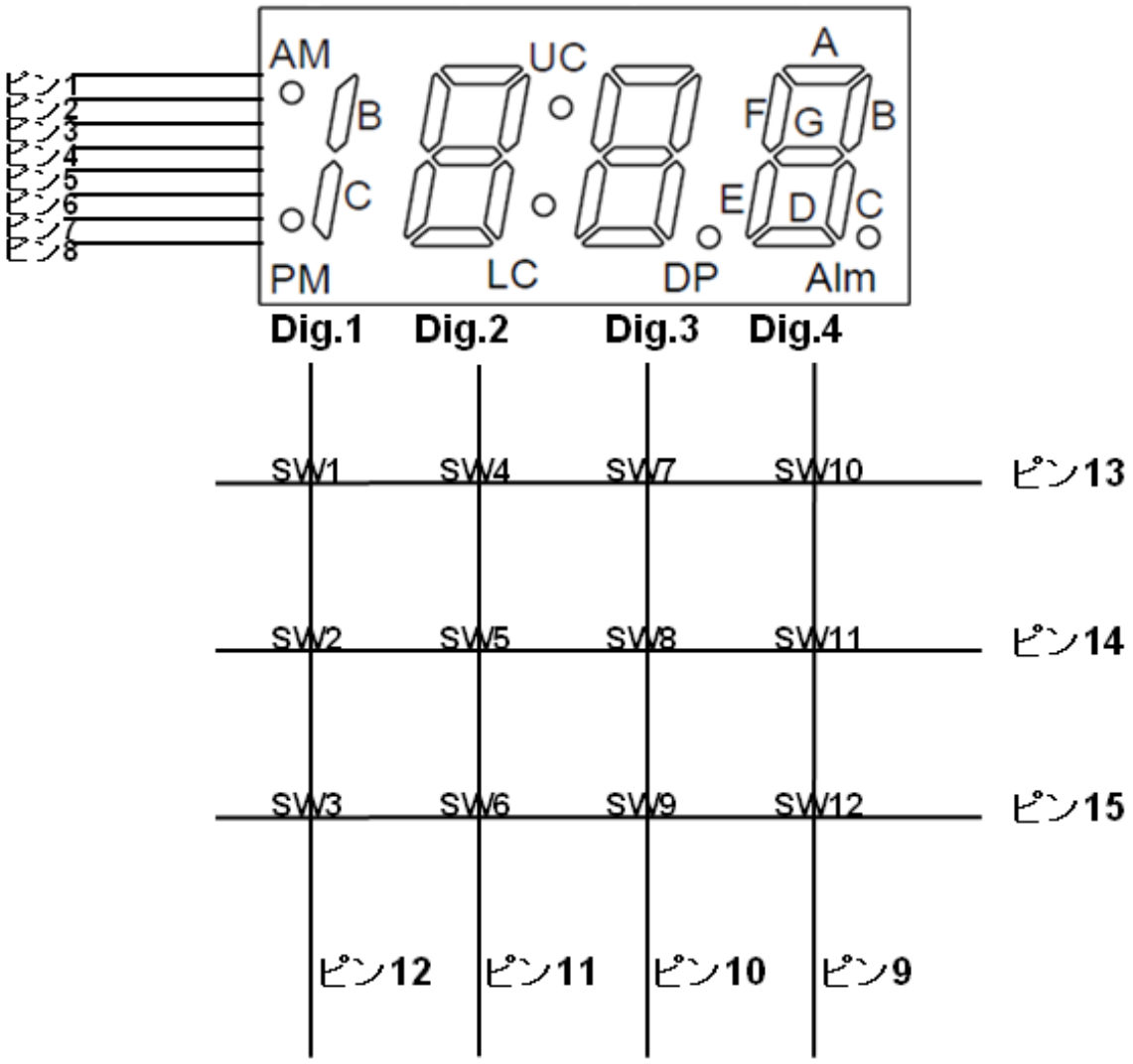


図 7-13　フラットケーブルのピンと GL-3P404 の繋がり方

実証用家電のフラットケーブルはピン 1～8 は GL-3P404 の各 Digit Segment の表示内容をコントロールしている。ピン 9~15 はボタン入力が使っているピンである(7.3 に説明がある)。しかし、ピン 1～8 の 8Bit の入力で 4 つ Digit Segment をうまくコントロールすることは、ピン 9～12 の入力も考える必要がある。

ピン 9 は ON の時　⇒　ピン 1～8 は Dig.4 をコントロールする
ピン 10 は ON の時　⇒　ピン 1～8 は Dig.3 をコントロールする
ピン 11 は ON の時　⇒　ピン 1～8 は Dig.2 をコントロールする
ピン 12 は ON の時　⇒　ピン 1～8 は Dig.1 をコントロールする

ピン 9~12 は周期に繰り返して ON に変わる。この周期は十分速いなら、人は LED を見ると、4 つ Digit Segment は全部表示している様子に成れる。

ここまでは実証用家電の LED ダイナミック点灯原理を紹介したが、4 つ Digit Segment の表示をうまく読み取る前に、ピン 9～12 の入力も考える必要があるので、実装する前の事前調査で調べた。

7.5　　LED 読み取り部分の実装の事前調査

LED ダイナミック点灯原理によって、実証用家電の LED を読み取る前に、実証用家電は特有の色々な情報を調べる必要がある。そして、実証用家電は特有のことで実装に対して障害があるかどうかも調べる必要がある。

7.5.1　　4 つ Digit Segment の切り替える周期と順番

7.4.1 と 7.4.2 は実証用家電の LED ダイナミック点灯原理を紹介したが、4 つ Digit Segment の表示をうまく読み取る前に、ピン 9～12 の入力順番と切り替える周期を測る必要である。下図は Oscilloscope(SeeedStudio 社の DSO Quad)で測ったピン 9～12 の電圧シーケンス図である。

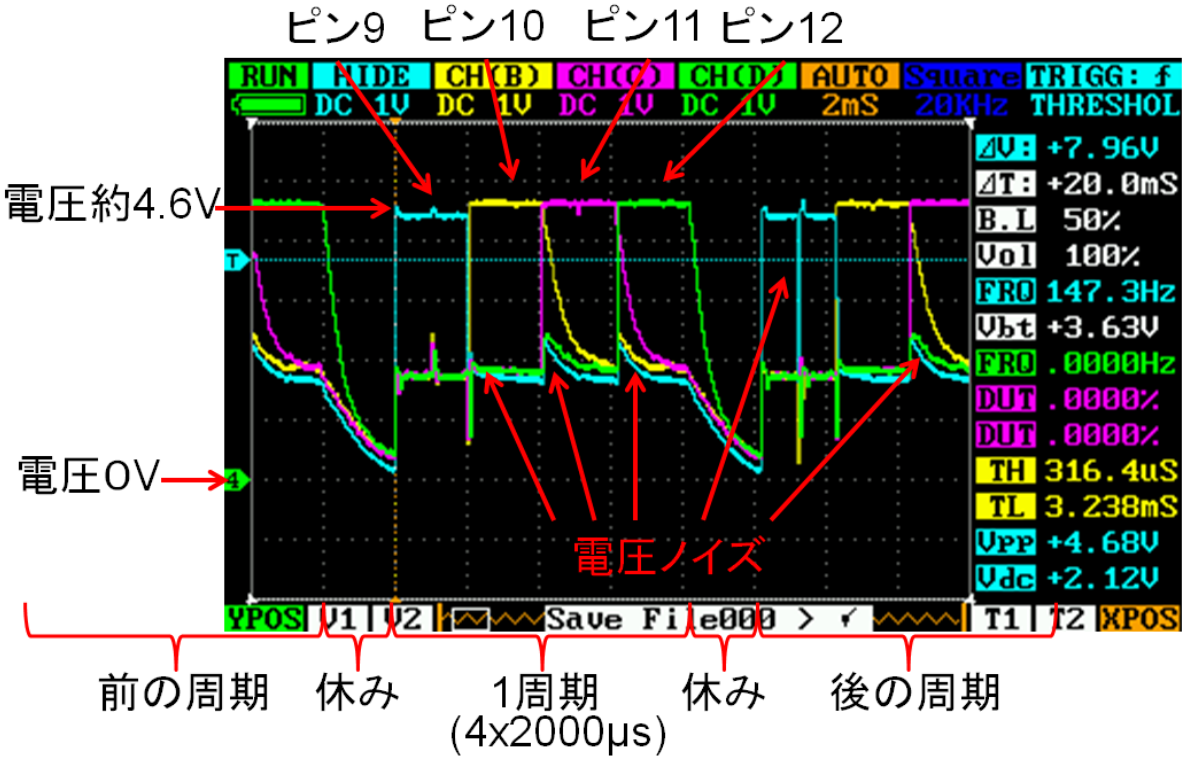


図 7-14　　ピン 9～12 の電圧シーケンス図

青、黄、紫、緑の 4 つ色のラインはピン 9、10、11、12 の出力電圧である(色とピン番号の順番は左⇒右、1 対 1 である)。縦方向は電圧である(1 枠は 1V)。横方向は時間である(1 枠は 2000μs)。

上図を見ると、ピン 9~12 の電圧シーケンスはピン 9⇒ON、10⇒ON、11⇒ON、12⇒ON、休みの順番で繰り返して交替する(休みの時間と各ピンの ON の時間は 2000μs である、同じ時間帯に ON の状態のピンは 1 つだけである)。

7.5.2　　LED 読み取りの実装方針

7.5.1 の調査結果で考えて、LED 読み取り部分の実装方針を作った。

(1)ピン 9 は ON になる瞬間にかかる割り込みを使う。(LED 割り込みと呼ぶ)

(2) LED 割り込みがかかった瞬間から、4 つ段取りを分ける。State1(0～2000μs)ピン 9 は ON の時間、State2(2000～4000μs) ピン 10 は ON の時間、State3(4000～

6000μs）ピン 11 は ON の時間、State4(6000～8000μs）ピン 12 は ON の時間。

(3) State1　⇒　Dig.4 の内容を読み取る。

(4) State2　⇒　Dig.3 の内容を読み取る。

(5) State3　⇒　Dig.2 の内容を読み取る。

(6) State4　⇒　Dig.1 の内容を読み取る。

この実装方針でうまく LED を読み取るため、2 つ条件である。

(1) LED 割り込みのかかる時点は安定的に毎回ちゃんと守る、さらに 1 周期 5*2000μs(休みも含む)以内にピン 9 は 2 回 ON になることがない。

(2) タイマーで測る時間は正確である。つまり、毎回の State 変更は間違いない。

(2)番目の条件は後のソフトウェアを実装する時に考えるので。まず、(1)番目の条件を検討する。

図 6-7 を見ると、ピン 9 の電圧は結構不安定である。色々なところにノイズがある。次はソフトウェアを実装する前のピン 9 の電圧ノイズを除く理由と除き方を説明する。

7.5.3　　ハードウェアによるノイズの低減

(1)除き理由：ピン 9 の理想的な電圧状態は、ON の時 4.6V、OFF の時 0V になるはずである。しかし、実証用家電は OFF の時が 0V ではなく、大部分の時間は 1.8V～2.6V 間を示している。4.6V が High、1.8V～2.6V が Low となるように外部回路を構成する方式で、電圧ノイズを除き方針を検討する。文献[23]を参照して下表のような PIC24FJ64GA002 の入力電圧情報である。

表 7.5　PIC24FJ64GA002 の入力電圧情報

名前	電圧範囲
V_{IL}(Input Low Voltage)	V_{SS}～0.2 V_{DD}
V_{IH}(Input High Voltage)	0.8 V_{DD}～5.5 V

V_{SS}=0V、V_{DD}=3.3V。V_{IL} の間の電圧は OFF を認めされる、V_{IL} の間の電圧は ON を認めされる。V_{IL} の最大値~V_{IH} の最小値の間の電圧は High か Low が保証されない。

0.2 V_{DD}～0.8 V_{DD}　=　0.2*3.3V～0.8*3.3V　=　0.66V～2.64V

この間の入力電圧は High か Low が保証されない。

1.8V～2.6V 間のノイズはちょうどこの保証されない範囲の電圧なので､除かなければならない。

(2)除き方と注意点：電圧ノイズを除き方は主に 2 つ種類である。ソフトウェアで除く、ハードウェアで除く。ソフトウェアの除き方は 6.4 で説明するので、まずハードウェアの除き方を説明する。抵抗を使って 1.8V～2.6V 間のノイズをできるだけ 0.66V 以下に抑える、ただ、抑える時に元々の 4.6V の電圧も下に抑えたので、この電圧は 2.64V 以上に保証することが必要である。

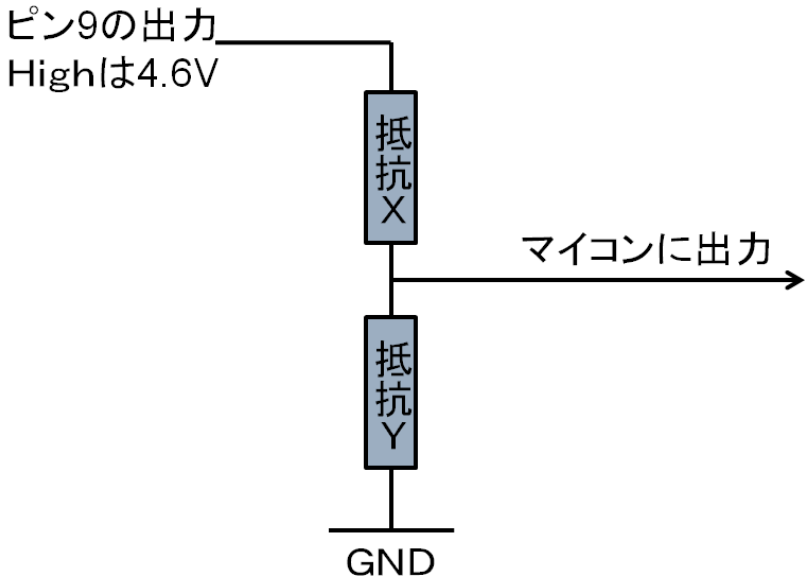


図 7-15　ハードウェアで電圧の抑え方

上図のような配線仕方で、仮設：抵抗 X が x　kΩ、抵抗 Y が y　kΩの場合に、ピン 9 の出力電圧 z は 4.6V(ON の電圧)と 2.6V(ハードウェアで除きたいノイズの最大値)で計算して抵抗 X と Y の最優オーム値を求める。

(1)計算式：(y/(x+y))*z　= マイコンに出力電圧

(2)下式を満足できる抵抗を採用する

x+y=10 kΩ

(y/(x+y))* 4.6 > 2.64V　　⇒　　y > 1.35*x

(y/(x+y))* 2.6 < 0.66V　　⇒　　y < 0.34*x

y > 1.35*x 且 y < 0.34*x はx > 0 且 y > 0 の場合に 解答がない

(3) 購入した抵抗の中に(2)の式の一部が満足できる抵抗を探して、一番近い方は抵抗 X が 3.9 kΩ、抵抗 Y が 6.2 kΩを採用する。

(6.2/(3.9+6.2))*4.6 = 2.82V > 2.64V　　○

(6.2/(3.9+6.2))*2.6 = 1.59V < 0.66V　　×

このような 2 つ抵抗を使って、全部のノイズを除けないが、ノイズの数量を大幅に削減できるはず。

残ったハードウェアで除けないノイズをソフトウェアで除く。

7.6　LED 読み取り部分の実装

家電 LED 読み取り原理の調査と実装の事前調査をしてから、実証用家電の LED 読み取り部分を実装する。(実証用家電の時間設定する時に LED の表示は点滅しているので、点滅している場合の LED 読み取りは 7.7.5 で検討する。本段落は点滅以外の場合の LED 読み取りを検討する。)

7.6.1　ソフトウェア実装の事前準備

(1) GL-3P404 を読み取るため、マイコンの 9 個空いているピンを用意する必要がある。8 個は実証用家電のピン 1~8 の入力を受ける(これから LED ピン 1~8 と呼ぶ)

1 個は実証用家電のピン 9 の入力を受ける(これから LED ピン 9 と呼ぶ)

(2) LED ピン 9 の割り込みを設定する、割り込みは LED ピン 9 が毎回 ON になる瞬間でかかる。

(3) 7.5.2 の各 State の時間を計れるタイマーを用意する(これから LEDTMR と呼ぶ)。

7.6.2 GL-3P404 の読み取り処理

下図は図 7-14 の各周期の GL-3P404 の 4 つ Digit Segment の読み取りフローチャートである。

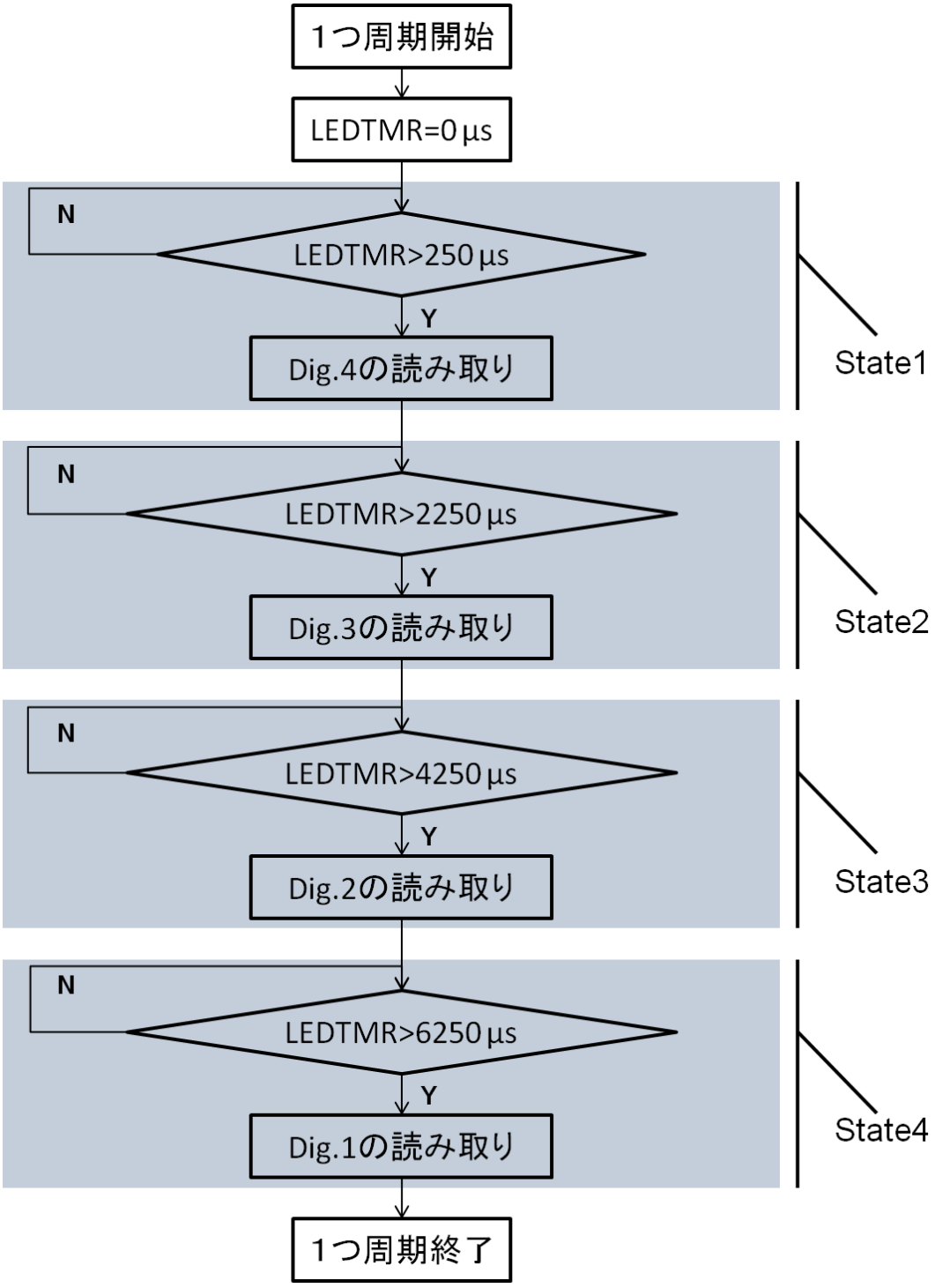


図 7-16 GL-3P404 読み取りフローチャート

毎回の LED 割り込みがかかる瞬間で上図の処理を行う。

実は State1、2、3、4 の時間帯の境界値は 2000μs、4000μs、6000μs である(7.6.2 に説明がある)。上図の中に LEDTMR と比べる時間は各境界値より 250μs 遅いである。これをする原因は読み取る時点が境界値により近過ぎると、LEDTMR の誤差があれは、違う Digit Segment の値を読み取ってしまう可能性が高い。または各 Digit Segment の読み取り処理も少しい時間がかかるので、各 State の真中より 250μs 早い時点で各 Digit Segment の読み取り処理を始めることを設定した。

具体的な各 Digit Segment の読み取る処理は 7.4.1 に説明である。

7.6.3 ソフトウェアによるノイズの除去

7.5.3 のようなやり方で電圧ノイズを大幅に削減したが、また残っているしつこい電圧ノイズもある。このような電圧ノイズを除きたいと、ソフトウェアの手段を使用しなければいけない。

LED 割り込みに 1 つ周期全部終わらないなら、次の割り込みがかからない制限をつけると、LED の間違い読み取りは大幅に削減できる。または読み取ったデータは正しいかどうかチェック処理を追加して、変なデータを捨てて、最新の正しいデータしか保存しない。これで電圧ノイズの原因で LED をうまく読み取らない障害を解決できた。

7.7 CIR 受信、疑似ボタン入力、LED 読み取りの統合

本システムで自分の担当する部分は家電側の開発である。開発は複雑の問題をモジュールに分ける原則によって、家電側の開発は主に 3 部分(CIR 受信、疑似ボタン入力、LED 読み取り)で分けた。開発は 1 つずつで行った、3 部分が別々の機能として作ったので、最後この 3 部分の連携も実装する必要がある。

7.7.1 3 部分連携用のハードウェア

3 部分連携の実装する前に、3 部分が使っているハードウェア資源を統一のハードウェア環境に纏める必要がある。

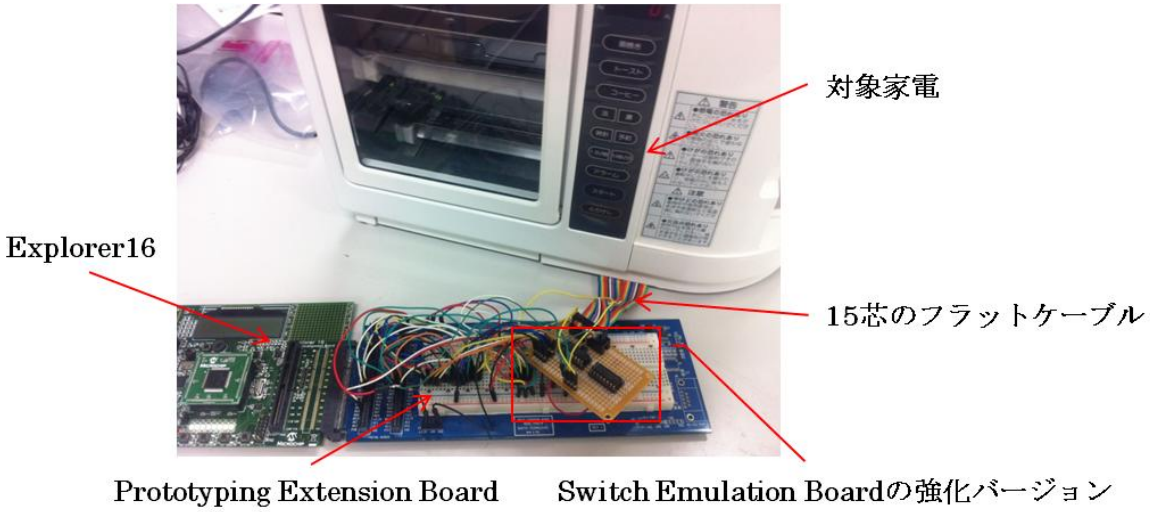


図 7-17 3 部分連携のハードウェア環境

CIR 受信部分のハードウェア(6.2.6 に説明がある)と疑似ボタン入力部分のハードウェア

(7.3 に説明がある)の上に LED 読み取りのため、Switch Emulation Board の強化バージョンを作った(これから LED Switch Emulation Board と呼ぶ)。

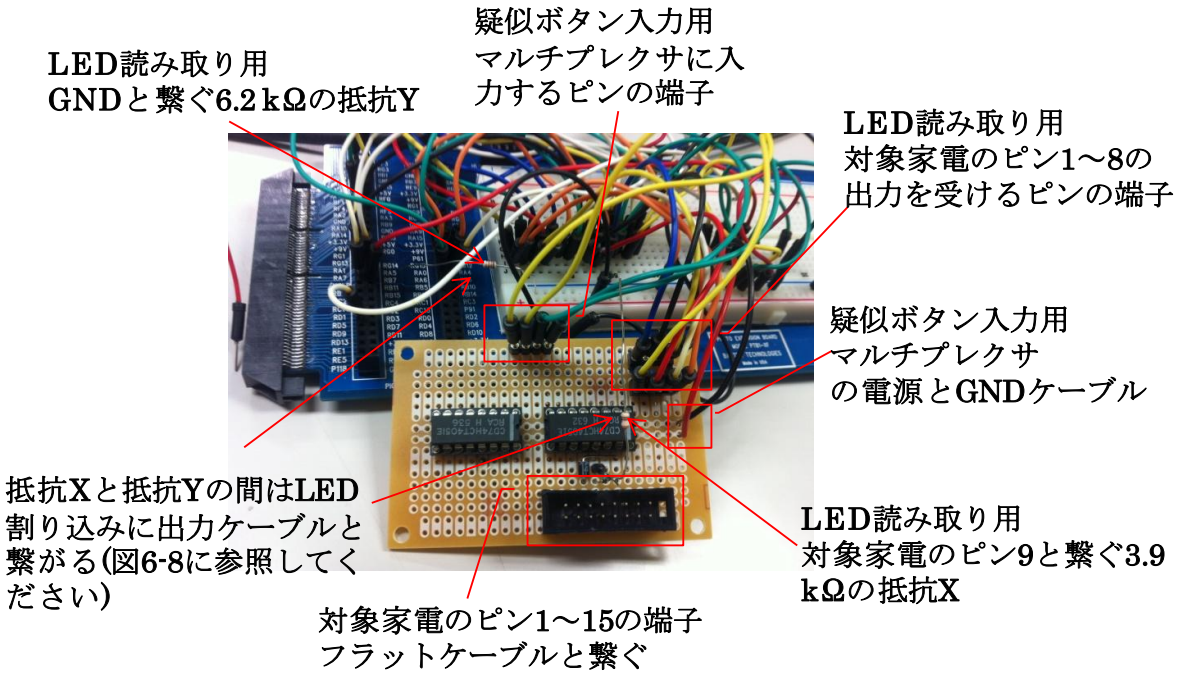


図 7-18　LED Switch Emulation Board

Explorer16 のマイコンの上に実装した 3 部分のソフトウェアを組み込んで、以上のハードウェアを使って、連携のソフトウェアを実装する。

7.7.2　タイマー割り込みとソフトウェアタイマーの活用

組み込みソフトウェア開発の時マイコンのタイマー(これからハードウェアタイマーと呼ぶ)は珍しい資源なので、できるだけ節約するべき。各時間を計る処理に対して 1 つハードウェアタイマーを使用するやり方をしたくないので、 ソフトウェアタイマーを利用した。

タイマー割り込みとは、ハードウェアタイマー の割り込み関数である。ハードウェアタイマーの毎回のカウントする時間を設定して、毎回この時間が経つと、このタイマーの割り込み関数がかかる。割り込み関数の中に時間の数えることと軽い判断処理を置く。しかし、重い処理をタイマー割り込みの中に置くと、計りたい時間は正確でなくなる可能性がある。

ソフトウェアタイマーとは、1 つハードウェアタイマーの割り込みの中に作った幾つカウントである。割り込みがかかる時、ソフトウェアタイマーを増やして時間を数える。

例えば：本システムは 3 つ処理の時間を計りたいので、以下の 3 つソフトウェアタイマーを作った。

CIR 受信用のタイマー	⇒	cirtimecnt
疑似ボタン入力用のタイマー	⇒	btntimecnt
LED 読み取り用のタイマー	⇒	ledtimecnt

1 つハードウェアタイマー(PIC24FJ の TMR3)は各 100μs に割り込みがかかることを設定する。

ハードウェアタイマーの周期を設定する原則は計りたい時間を計れる程度の最大値である。100μs を採用する原因もこの通りである。

このハードウェアタイマーのタイマー割り込みの中に 3 つソフトウェアタイマーを

増やして。各ソフトウェアタイマーも各 100μs にカウントできることになった。

各処理は自分のソフトウェアタイマーしか読み書かないと、1 つハードウェアタイマーで多処理の時間を計る目的を達成できる。

7.7.3 CIR コマンドからボタン入力シーケンスへの変換

(本段落の詳しい内容は家電側の家電コントロールシーケンス図に示す)家電側は CIR コマンド(付録の「コマンド転送規則」に説明がある)を受信すると、すぐにこのコマンドの指示によって、疑似ボタン入力処理をすることではない。原因は普段にユーザが携帯で送信した指示は 1 つ CIR コマンドで表示できない、毎回の指示は幾つの CIR コマンドをクループして送信する形になっている。つまり、各指示の CIR コマンドを全部に受信する前に、疑似ボタン入力処理を行えない。

これに従って、各 CIR コマンドを受信すると、疑似ボタン入力のために用意したデータバッファー(これから命令バッファーと呼ぶ)に保存して、疑似ボタン入力コード(7.3.3 と 7.3.4 に説明がある)リストを作る時使う。1 回指示を全部終わってから(作業開始コマンドを受けた時) 、疑似ボタン入力処理を行う。

疑似ボタン入力処理の中に、命令バッファーを参照して、疑似ボタン入力コードリストを作る。

さらに疑似ボタン入力コードリストを作る時に命令バッファーだけを参照することは不十分である。命令バッファーの中に特別な命令がある時に、LED の状態を読みながら、疑似ボタン入力コードを作る場合もある。これは次の段落で説明する。

7.7.4 実証用家電の LED 状態による疑似ボタン入力

実証用家電は幾つ状態がある。ある状態はある命令を受けないことがある。CIR が単方向通信なので、家電側は当状態で受けない命令を受信すると、携帯に「命令を受けない」ことを返事できない。さらに、家電の状態を読まないなら、家電側の Daughter Board(図 7-3 に説明がある)でも家電の状態がわからない。それで家電をうまくコントロールできない。

従って、LED の状態を読みながら、疑似ボタン入力コードリストを作る必要がある。

特に実証用家電の現在時間、予約時間、アラーム時間などの時間を設定する操作がある場合に LED の状態の読み取りはとても重要である。普通の組み込みシステムの時間設定処理は家電のメモリ中の時間バッファーの値を修正する形であるが。今回の実証用家電に対しては他の方法しなければならない、原因は今回の実証用家電をコントロールする方式が直接に実証用家電の元々のプログラムが提供している API を呼び出すことではなく、疑似ボタン入力である(7.3 に説明がある)。つまり、毎回の時間設定は人の操作のように特定の回数の分/時ボタンと秒/分ボタンの疑似入力をすることである。

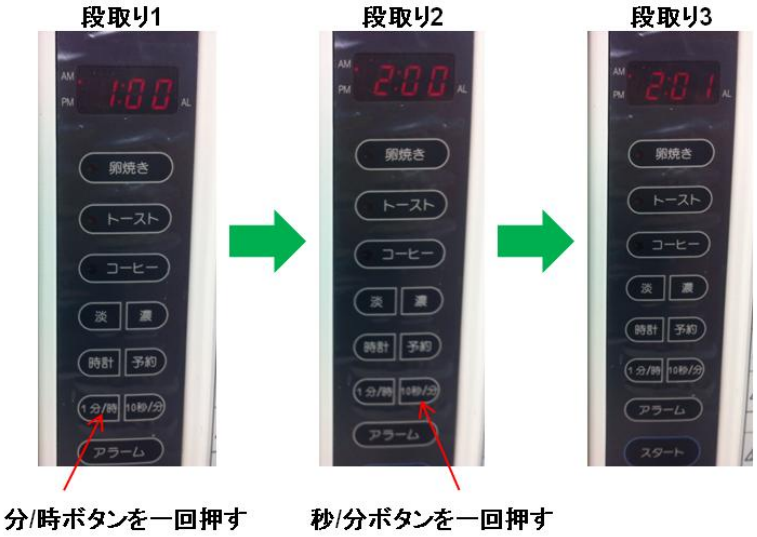


図 7-19　実証用家電の時間設定の流れ

上図は実証用家電の時間設定の流れである(設定したい時間と表示している時間の差別によって各段取りのボタン押す回数を決める)。実証用家電の時間設定は 3 つ段取りである。

段取り 1：表示している時間の確認、分/時ボタンの押す

段取り 2：秒/分ボタンの押す

段取り 3：設定した時間の確認、完了

人のボタン入力と疑似ボタン入力の流れは同じである。つまり、この部分のソフトウェアの設計も人のボタン入力の流れにエミュレーションをするべき。次は時間設定がある時の疑似ボタン入力の流れ。

(1) 表示している時間を読み取る。

(2) 読み取った時間が設定したい時間と比べて、分/時ボタンの押す回数を計算する。

(3) 読み取った時間が設定したい時間と比べて、秒/分ボタンの押す回数を計算する。

(4) 分/時ボタンの回数によって、疑似ボタン入力処理をする。

(5) 秒/分ボタンの回数によって、疑似ボタン入力処理をする。

ここまで、理論的に 3 部分の統合実装も終わったが、まだ 1 つ問題を解決する必要がある。次の段落で説明する。

7.7.5　点滅している LED の時間の読み取りによる疑似ボタン入力

実証用家電の時間設定する時に LED の表示は点滅している。LED 読み取りは最後の有効な LED の表示をメーモしているので、理論的に LED は点滅しても、最後にメーモした値を使っても計算してくれた時間ボタンの押す回数が間違いない。しかし、実証用家電のフラットケーブルの出力電圧ノイズの原因で LED の読み取りは時々変なデータがある、このようなデータを捨てる処理(7.5 と 7.6.3 に説明がある)をしているので、LED を消す直前の参照したいデータを捨てる可能性もある。

従って、家電 LED を参照して疑似入力をする前に点滅している時間を確認するための待ちが必要である。待ち時間の長さは家電 LED の点滅周期決める。筆者の考え方は下式のような待ち時間で十分であると思う。

待ち時間 = 家電 LED の点滅周期 + 100000μs;

そうすると、待ちしている間に災厄の状況は LED の消すことが待ち時間の真中の場合である、この場合の LED 読み取る回数を下式で検討する。

LED を表示している時間 = 消す前 50000μs + 消した後 50000μs;
LED 読み取り周期(7.5 に説明がある) = 5 * 2000μs = 10000μs;
消す前の読み取り回数 = 50000μs / 10000μs = 5(災厄の場合は 4)回;
消した後の読み取り回数 = 50000μs / 10000μs = 5(災厄の場合は 4)回;
災厄の場合に待ち時間の読み取り回数 = 8 回;

電圧ノイズの災厄の場合でも、連続的な 8 回の不正な読み取りがないので、以上のような待ちことをしてから､必ず 1 回正しい読み取るデータを保存できる｡このデータを参照して、疑似ボタン入力処理をする。疑似ボタン入力部分の時間の測りは 7.7.2 のソフトウェアタイマーbtntimecnt を使う。

7.7.6 統合した後の強化バージョンの疑似ボタン入力処理

7.3 まで実現した疑似ボタン入力処理の単位は毎回 1 つボタン処理である｡3 部分を統合する時、7.3 を基ついて疑似ボタン入力処理は疑似ボタン入力コードリストが単位として修正した。

(1)修正する理由：

理由 1：疑似ボタン入力処理はボタン押す操作だけではなく、押す状態から離れる操作も疑似入力する必要がある。さらに毎回疑似ボタン入力の後、次回疑似ボタン入力の前に実証用家電の反応時間も必要なので、ボタン操作の間に待ち時間(テストしてもらった値は 35000μs でちょうどいい)が必要である。

理由 2：統合した後の各単位の疑似ボタン入力回数は毎回の命令バッファーの情報で決めることである、毎回の疑似ボタン入力コードリストのサイズの大値は何百回の場合もある。

理由 3：7.3 のまま、疑似ボタン入力機能を強化しないと、理由 2 場合の多回の疑似ボタン入力操作の場合に、毎回の疑似入力する時、理由 1 のことを考えられなければならない。

以上の理由で疑似ボタン入力処理を強化するつもりである。

(2)強化バージョンの疑似ボタン入力処理詳細：

処理 1：強化した疑似ボタン入力処理は命令バッファーと LED 読み取りのアウトプットによって、疑似ボタン入力コードリストを作る。

処理 2：タイマー割り込みで周期的に疑似ボタン入力コードリストの内容を一個ずつにトラバーサルをすることである。下図は各疑似ボタン入力コードに対する処理フローチャートである。

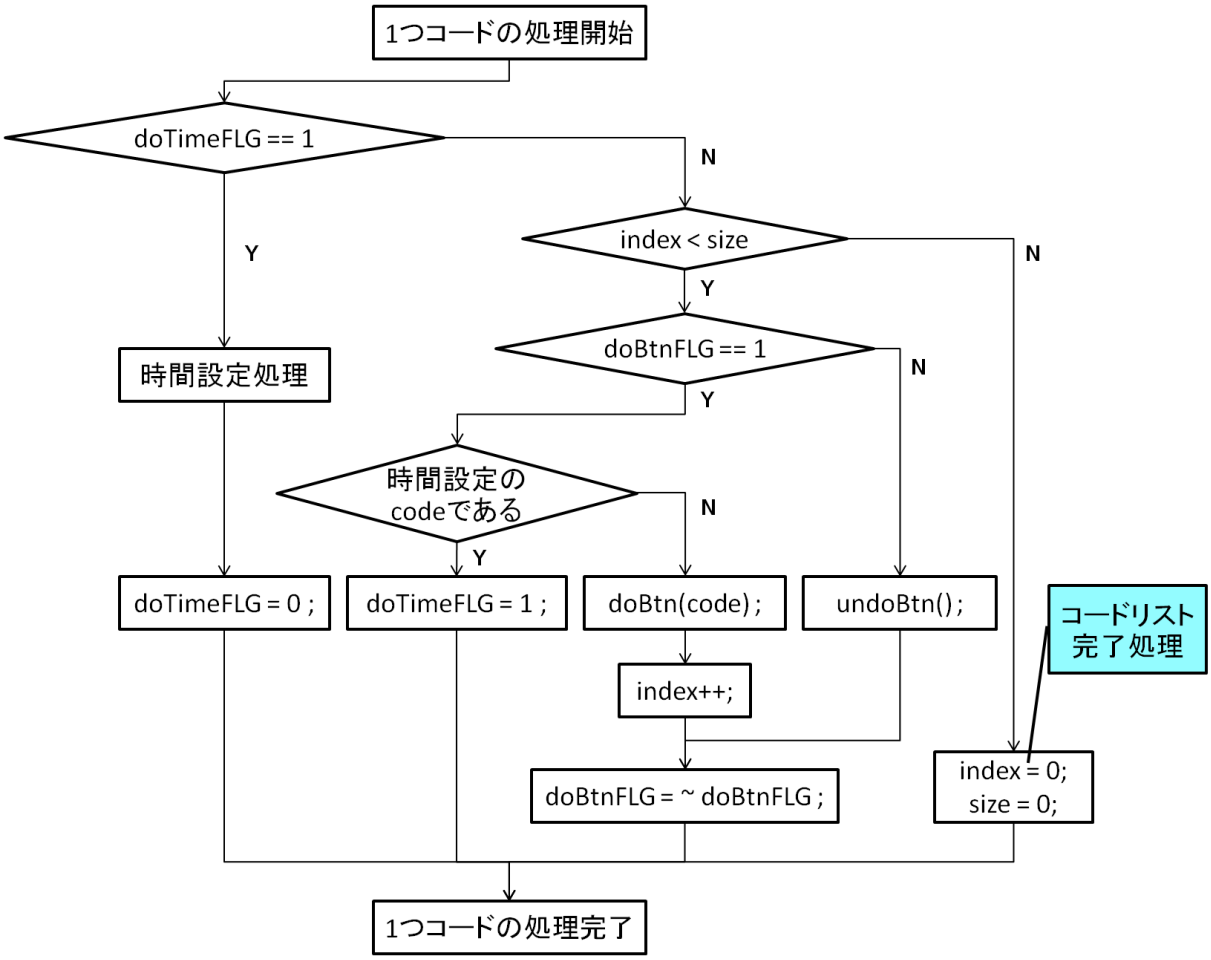


図 7-20　各疑似ボタン入力コードに対する処理フローチャート

上図の時間設定処理は 7.7.4 と 7.7.5 に説明がある。

上図を使っている変数と関数は下の注釈 1~6 に説明がある。

注釈 1：index は疑似ボタン入力コードリストのトラバーサルをしているコードの索引である。

注釈 2：size は疑似ボタン入力コードリストが保存しているコード数である。

注釈 3：code はトラバーサルをしている疑似ボタン入力コードである。

注釈 4：doTimeFLG==1 の場合、時間設定の疑似ボタン入力処理をする。doTimeFLG==0 の場合、普通な疑似ボタン入力処理をする。

注釈 5：doBtnFLG ==1 の場合、当コードのボタン押す操作をする。doBtnFLG ==0 の場合、押す状態から離れる操作をする。

注釈 6：ボタン押す操作と押す状態から離れる操作を 2 つ関数の中にカプセル化をする(上図は doBtn(code)と undoBtn()で表示する)。

ここまで、CIR 受信、疑似ボタン入力、LED 読み取り 3 部分統合の実装が終わった。

第8章　　家電側のソフトウェアの検証

第 7 章までは家電側のソフトウェアの実装もう完了したが、本システムの他の部分と連携する前に、家電側のソフトウェアをテストする必要がある。

8.1　　家電側のテスト環境

筆者が担当する部分の開発は組み込み開発なので、実装とテストは補助的なツールを使用したことがある。テストの内容を説明する前に、テスト環境を簡単に紹介する。

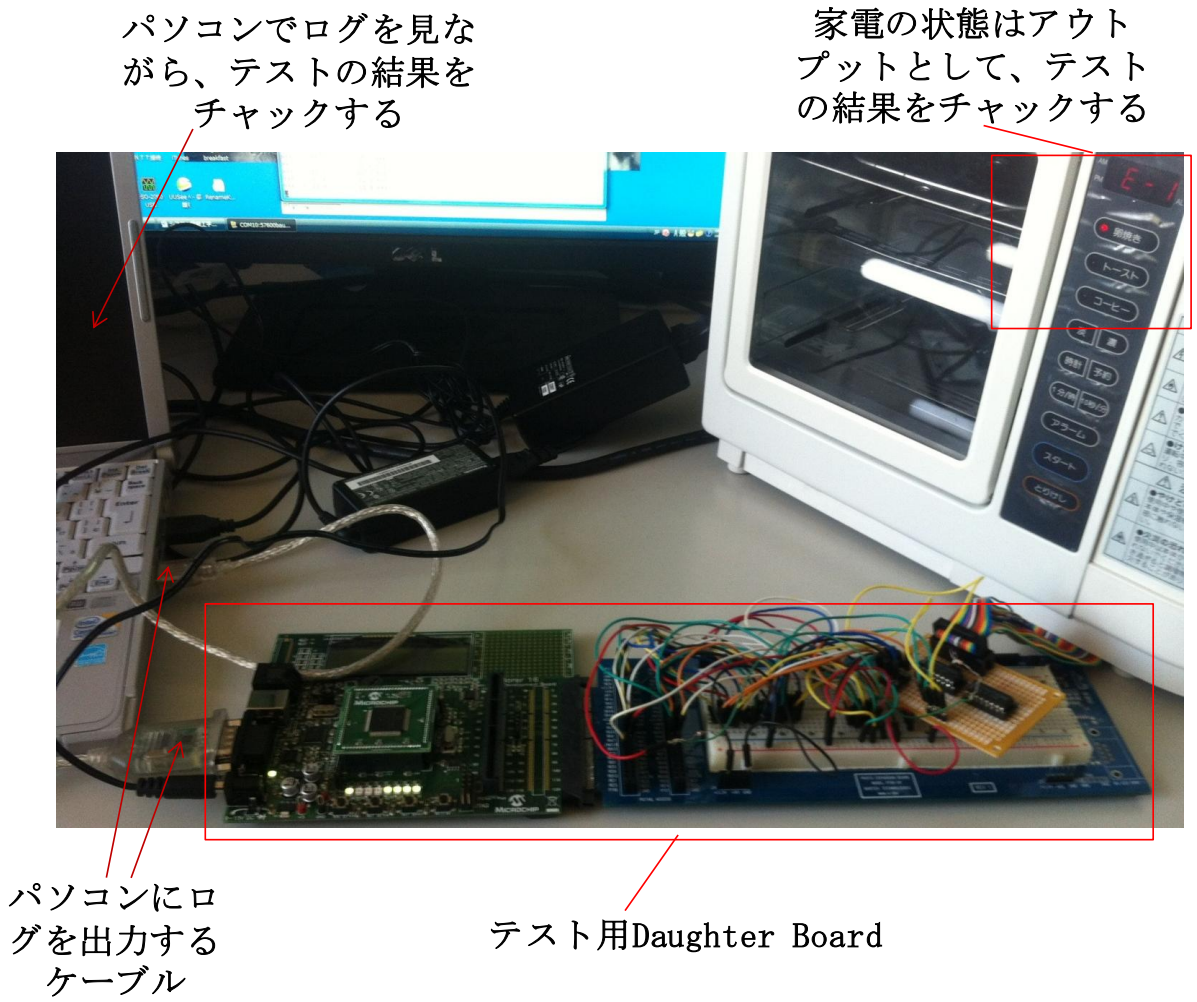


図 8-1　　家電側のテスト環境

テスト用 Daughter Board(7.7.1 に説明がある)と実証用家電を繋がって、テスト結果は実証用家電の反応で確認する。また、実証用家電の反応で判断できないテストパタンはテスト用 Daughter Board からのログで確認する。

携帯の CIR 送信部分の実装が終わっていない時、家電側の CIR 受信のテストはシーネッ

ト社の NEC フォーマットを送信できるテレビリモコンを利用する。

図 8-2　NEC フォーマットを送信できる赤外線リモコン(KTRA-109)

NEC フォーマットは標準なので、携帯と家電側が両方ともこの標準を守ると、必ず通信できる。従って、携帯側の送信部分ができていない内に KTRA-109 を使用するテスト仕方は問題ないはず。

8.2　家電側の単体テスト

筆者が担当する 3 部分の結合テストする前に、各部分を単体テストする必要がある。次はこの 3 部分の単体テストの内容を説明する。

8.2.1　CIR 受信部分の単体テスト

CIR 受信部分のテストのやり方について、KTRA-109 が送信器として、家電側の CIR 受信部分のソフトウェアが NEC フォーマットの赤外線コマンドを受信できることを確認する(受信したコマンドをログで出力する)。

KTRA-109 を使うテストは実装したソフトウェアが NEC フォーマットの赤外線コマンドを受信できることを確認した。

携帯側の送信機能をできた後、また携帯は送信器としてテストした。このテストは携帯が指定するコマンドを正しく受信できることを確認した。

KTRA-109 と携帯でテストする時、ログで確認する内容は以下である

(1) 毎回の送信コマンドを全部受信できること。
(2) 各コマンドの Custom Code と Custom Code の反転があっていること。
(3) 各コマンドの Data Code と Data Code の反転があっていること。

8.2.2 疑似ボタン入力部分の単体テスト

実証用家電は 12 個ボタンがある。ボタン押す状態から離れ操作も含めて 13 個入力状態がある。この 13 個入力状態に対する、4 ビットの入力バッファーを使った(7.3.3 に説明がある)。疑似ボタン入力部分のテストは、この 4 ビットの全部の 16 個入力パタン(表 7.1 に示す)でテストして、実証用家電の反応でボタン疑似入力操作を確認する。

このテストで実証用家電の任意のボタンに対する疑似入力できることを確認した。

8.2.3 LED 読み取り部分の単体テスト

実証用家電の LED 読み取りは主に表示している時間の数字と AM/PM に対する読み取りである。

これに対する確認するべきパタンは以下である。

Dig.1 ⇒ AM、PM、1 を読み取れること
Dig.2 ⇒ 表 7.3 と表 7.4 の内容を読み取れること
Dig.3 ⇒ 表 7.3 と表 7.4 の内容を読み取れること
Dig.4 ⇒ 表 7.3 と表 7.4 の内容を読み取れること

実証用家電を操作して、変化した実証用家電 LED の表示と読み取りソフトウェアから出るログを比べて、以上のテストパタンを確認する。

このテストで実装した LED 読み取り部分のソフトウェアが実証用家電をコントロールするための情報を読み取れることを確認した。

8.3 3 部分統合の結合テスト

家電側のソフトウェアとは、携帯から CIR コマンドを受信してから、コマンドによって実証用家電をコントロールする。従って、携帯側のソフトウェアの実装を完了する前に、家電側のソフトウェアをテストしたいなら、携帯の送信をふりするテスト方式を検討する必要である。

8.3.1 リモコンのボタンを押す送信で、携帯の CIR 送信にふりをするアプリ

NEC ファーマットの CIR コマンドを送信できるリモコン(KTRA-109)を利用して、携帯の CIR 送信にふりをするアプリを作って、このテストアプリを利用してテストする。

テストアプリの構成は主に 2 部分がある。

(1) テスト用 CIR コマンド配列を作る。この配列は各テストパタンに対する CIR コマンドグループを保存する(テストパタンの内容は付録の「家電側結合テスト用コマンド表」用コマンド表に示す)。
(2) リモコンから受信した時、リモコンの押下したボタンによって、(1)の配列の中の押下したボタンと対応しているテストパタンの CIR コマンドを受信したようにする。

このテストアプリを利用して 3 部分統合の結合テストをした。

8.3.2 テスト内容と結果

結合テストの内容と結果は付録の「家電側結合テスト項目表」に示す。

8.4　　被験者試験について

本システムの性能を計るために、視覚障碍者は本システムを利用して、アンケート(付録の「アンケート」を参照する)を書いた。

試験の流れは以下である。

本システムの内容を説明する。

おまかせミールさんの機能を説明する。

本システムを利用しないで、直接におまかせミールさんを操作する。

本システムを利用して、おまかせミールさんを操作する。

アンケートを答える。

以下は被験者が本システムを利用した感想である。

本システムは視覚障碍者に対するとても役立つであるところは、家電の時間に関する設定の場合である。これは視覚障碍者が家電を操作する時の非常に困難なところである。または、システムはよくてきていると思うが、初めてのユーザに対して、携帯のアプリは対象家電の全体的な説明ができた方がいいと思う。前回の設定を覚えるなら、便利になる。音声応答の内容はわかりにくい場合がある。

頂いた意見による、システムの性能を高めることは今後の課題である。

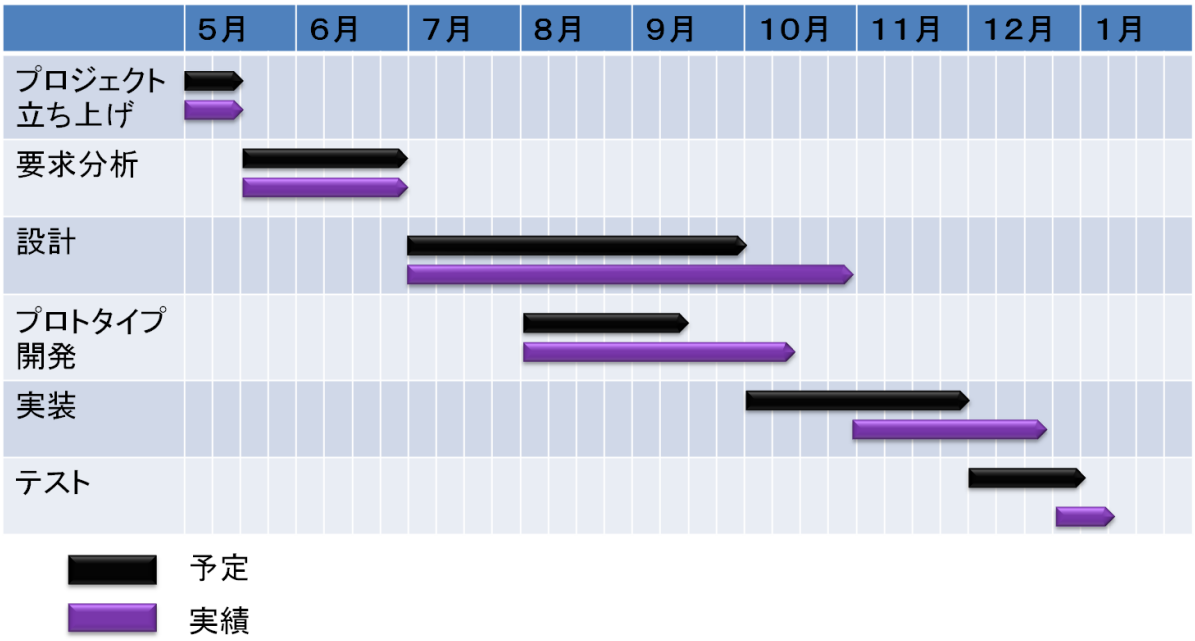
第9章　　まとめ

本システムは、委託元教員の要求を受け、「携帯電話を活用する音声応答家電の開発－家電組み込み用赤外線通信・制御ファームウェアの開発－」を行った。

本システムの開発要件を決めるにあたり、委託元教員にヒアリングとインタヒューをし、委託元の要求を元に、機能がより充実なシステムを提案した。

システム開発する前にスケジュールを作ったが、実際に開発する時、実績と予定の差別を下表で示す。

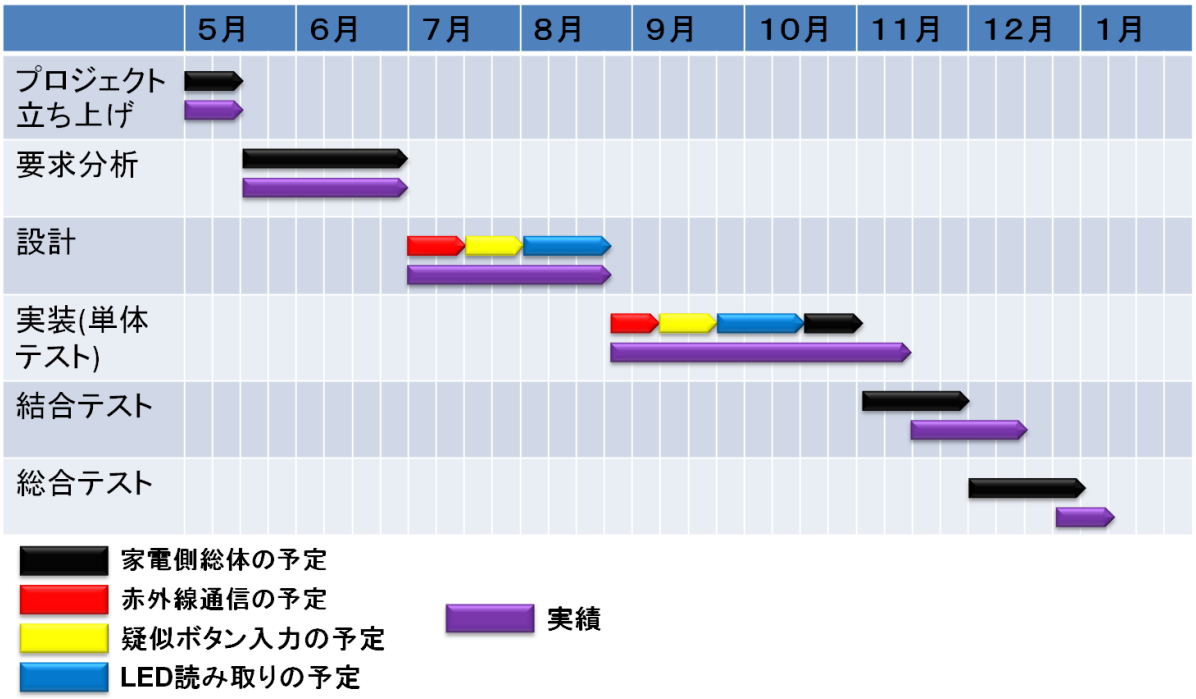
表 9.1　携帯側の開発スケジュール(実績)



上表は携帯側の開発スケジュールの予定と実績を比べることである。
携帯側の設計の 7、8、9 月に夏休みがある、その時の開発は予定よりとても遅れたので、毎週の会議を週 1 回から週 2 回に変更した、会議を週 2 回にしたのは、中間発表 2 の前あたり。

筆者は担当する家電側のシステムの開発スケジュールの実績と予定の差別は下表で示す。

表 9.2　家電側の開発スケジュール(実績)



家電側の開発は組み込み開発である、筆者は開発しながら、色々なハードウェア知識も身に付けなければならない、進捗は予想より遅いである。中間発表 2 の後、筆者はシステムの進捗と残る時間を考えると、開発目標を見直した。家電側の双方向赤外線通信 IrDA の部分をカットすることになった。

筆者が先導 IT プログラムに入る目的は組み込み開発技術を身に付けたいなので、本システムの家電側の開発を担当して実装したことはよかった、開発の過程で色々な組み込み開発知識を勉強してきた、特に委託元教員からに教えてもらった組み込み知識は家電側の開発に対してとても重要である。

謝辞

本プロジェクトを行うにあたり、委託元教員である秋川友宏准教授から大変貴重なご助言、ご指導を頂きました。心より感謝いたします。

指導教員である田中二郎教授には、様々なご指摘と助言を頂きました。研究生時代から指導してくださった 2 年間半、本当にお世話になりました。深く感謝いたします。

筆者が最初に田中二郎教授の研究室の研究生になったことは菊池純男教授から紹介して頂いたので、菊池純男教授にも心より感謝いたします。

Omakase チームのメンバーである。徐偉さん、章程君、持田辰弥君 3 名からも多くの助力と意見を頂きました。有難うございます。

最後には、自分が日本にきてから、ずっと支えてくれた両親そしてすべての友人に心より感謝いたします。

参考文献

[1]URC コンソーシアム,
http://myurc.org/.

[2]樋口正生,森岡隆一郎,稲垣勝利,戸崎明宏, 家庭内 AV ネットワーク HAVi の概要,
PIONEER R&D, Vol11, No.2, pp.39-49.

[3] ECHONET CONSORTIUM, APPENDIX ECHONET, 機器オブジェクト詳細規定,
Ver.3.21 Release b, 2005/10/13.

[4]多鹿陽介, 門間信行, 一色正男, ネットワーク家電の標準化技術―ECHONETTM 技術とその応用, 日本ロボット学会誌, Vol.21, No.6, pp.626～631, 2003.

[5] Sun Microsystems, Jini Architecture Sepecification, Version 1.2, 2001/12.

[6]門脇恒平, 早川裕志, 小板隆浩, 佐藤健哉, インタフェースレス志向による Jini システム構築, FIT2006(第 5 回情報科学技術フォーラム).

[7] RBBTODAY, Jini(ジニー。Java Intelligent Network Infrastructure),
http://dictionary.rbbtoday.com/Details/term1570.html
(2000) .

[8] William Grosso, Java RMI, オライリー・ジャパン, 2002/06.

[9] NTT ドコモ株式会社:対応コンテンツ・機能一覧,
http://www.nttdocomo.co.jp/binary/pdf/service/developer/make/content/spec/imode_spec.pdf
(参照 2011/10/04) .

[10] Rex Jaeschke, THE DICTIONARY OF STANDARD C, HBJ, May 1994.

[11] Kernighan, Ritchie, The C Programming Language, PRENTICE HALL SOFTWARE SERIES, Second Edition.

[12]石田晴久, C プログラミング, 岩波書店, 1984 年 3 月.

[13] Peter Coad, Edward Yourdon, Object Oriented Design, Prentice Hall, Vol.7, Vol.8, 1991.

[14]浅井麻衣, 重田正俊, 橋本大輔, 浜口弘志, 藤井啓詞, 現場の UML, モデルベース開発のすべて, ソーテック社, 第 1~5 章, 2006 年 12 月.

[15] NTT DoCoMo, DoJa-5.1 (API リファレンス),

http://logicalwave.jp/docomo/overview-summary.html
(参照 2011/12/1) .

[16]日本アイ・ビー・エム PC カード開発, 赤外線通信プログラミングガイド IR, ソフトバンク株式会社　出版事業部　ハードウェア活用書編集部, 第 1 章, 1995 年 11 月.

[17] Microchip, AN1071:IrDA Standard Stack for Microchip 16-Bit and 32-bit MCUs, http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en528001.

[18] Microchip, IrDA Standard Library Help, Release Notes, November 2008.

[19] Microchip, Explorer 16 Dev. Board User's Guide_DS51589a.

[20] A versit Consortium Specification, The Electronic Calendaring and Scheduling Exchange Format Version 1.0, September 1996.

[21]志田晟, ディジタル・データ転送技術入門, CQ 出版社, 第 2~3 章, 2006 年 1 月.

[22]高木弘之, 中塚重行, 赤外線リモコンを理解する, トランジスタ技術, 第 4 章, P261~277, 1996 年 11 月.

[23] Microchip, PIC24FJ64GA002 Family Data Sheet.

[24] Microchip, PIC24FJ128GA010 Family Data Sheet.

[25] ハラパン・メディアテック 宇野俊夫, インタフェース組み込み I/O, 翔泳社, 第 5~6 章, 2006 年 9 月.

[26] ロブウィリアムズ, リアルタイム組み込みシステム, 翔泳社, 第 3 章, 第 12 章, 2006 年 11 月.

[27]江村潤朗, 野津昭, プログラム流れ図の作成技法, オーム社, 第 1~3 章, 1981 年 8 月.

[28] Microchip, PIC24F Reference Manuals, Section 14. Timers, TIMER VARIANTS, 2006.

[29] Sharp, Numeric LED Data Sheet.

[30] PHILIPS, 74HC/HCT4051 8-channel analog multiplexer/emultiplexer.

付録目次

付録 A：要件定義書
付録 B：IOC-125　基板の解析
付録 C：IOC-125　LEDSW Board の回路図
付録 D：ユースケース記述
付録 E：画面イメージ図
付録 F：画面設計書
付録 G：メッセージ定義書
付録 H：チェック項目定義書
付録 I：IOC-125 用 CIR コマンド・コマンド転送規則
付録 J：仕様記述法
付録 K：IOC-125 用仕様記述 1.1
付録 L：クラス図
付録 M：シーケンス図
付録 N：プログラム設計書
付録 O：単体テスト項目表
付録 P：結合テスト項目表
付録 Q：家電側結合テスト項目表
付録 R：総合テスト項目表
付録 S：被験者実験の評価アンケート

付録 A

要件定義書

要件定義書

ver.1.7

改変番号	改変内容	作成者
1.0	初版	章程
1.1	検討の部分を削除 不足の部分を追加	章程
1.2	家電に関する共通部分追加 導入するシステムの機能の説明追加	章程
1.3	文章全体を書きなおす	章程
1.4	1.4 節削除 2 章内容修正 6.1 節、7 章内容追加 5 章図交換	チーム OMAKASE
1.5	システム構成図変更 2 章と 3 章合併	チーム OMAKASE
1.6	日本語の修正を含め全体の修正	持田辰弥

チーム OMAKASE

2011/09/18

目次

1. システムの概要 4
   1.1. 背景 4
   1.2. 目的 4
   1.3. 想定する利用者 4
2. システム構成 5
3. システム要件 6
4. 機能要件 7
   4.1. 携帯電話用端末ソフトウェア 7
   4.1.1. 通信仕様解析機能 7
   4.1.2. 命令コマンド作成機能 7
   4.1.3. 画面描画機能 7
   4.1.4. 音声読み上げ機能 7
   4.1.5. IrDA 送受信機能 7
   4.1.6. CIR 送信機能 8
   4.1.7. HTTP 通信機能 8
   4.2. 家電用通信ファームウェア 8
   4.2.1. IrDA 送受信機能 8
   4.2.2. CIR 受信機能 8
   4.2.3. 家電制御機能 8
   4.3. おまかせミールさん用カスタマイズ 6
   4.3.1. 調理機能 6
   4.3.2. 予約機能 6
   4.3.3. 現在時刻設定機能 6
   4.3.4. 目覚まし機能 7
   4.3.5. 制約条件 7
   4.3.6. ユースケース図 8
5. 非機能要件 9
   5.1. 効率性 9
   5.2. 機能性 9
   5.3. 安全性 9
   5.4. 移植性 9
   5.5. 使用性 9
   5.6. 信頼性 9
   5.7. 保守性 9
   5.8. 操作性 9
6. システム導入後の流れ 10

6.1. IrDA 10
6.2. CIR 11
7. 開発推進体制 12
7.1. 開発環境 12
7.1.1. 携帯側 12
7.1.2. 家電 Daughter Board 側 12
7.2. 開発大日程 12
8. 初期導入費、維持費 12
9. 保守・運用について 12

1. システムの概要

1.1. 背景

洗濯機やオーブンレンジなどの家電の操作メニューは複雑化しており、目の不自由な方が使いこなすことは困難である。解決策としては音声応答機能を家電に付けることだが、音声応答機能を付けるにはスピーカやアンプ、規則音声合成用のためのソフトウェアなどが必要となっているためコストが高い。そのため、現在音声応答機能は一部のフラグシップモデルにしか備わっておらず、家電の種類によっては音声応答可能な機種が存在しない。

一方、多くの目の不自由な方の間では、らくらくホンのシェアが高い。この携帯電話は音声応答機能があるため、目の不自由な方でも利用できる。

そこで、らくらくホンと家電を連携させることで、音声応答での操作が可能となる家電を実際に製作し稼働させる。

1.2. 目的

本研究が成功したら、低コストで音声応答機能による家電の操作を実現する方式になる。この方式により、低コストで音声応答家電を実現することができれば、家電メーカーの参入も容易にすることができる。

そこで、本研究は目の不自由な方たちが簡単に家電を使えること、そして、低価格で実現できる機能を実機の開発を通じて実証する事が本研究の目的である。

1.3. 想定する利用者

目の不自由な方が本研究の利用者と想定する。

2. システム構成

システム構成図

システム構成は上の図のようになる。

対象家電の LED/SW ボードとマザーボードの間に携帯からの命令を受け取るための家電用 CIR/IrDA 通信ボードを設置する。家電用 CIR/IrDA 通信ボードは、携帯から受信した命令をマイコンで解析し、命令を LED/SW ボードとマザーボードに送信、LED/SW ボードでは命令内容を表示させる。マザーボードでは命令を実行させる。

家電用 CIR/IrDA 通信ボードの上には IrDA（Infrared Data Association）用の送受信機と CIR(Consumer Infrared)用の受信機を備え付ける。IrDA の場合、送受信が可能なため、家電の通信仕様は家電から直接ダウンロードする。CIR の場合、単方向通信であり、家電側から仕様をダウンロードできないため、インターネット上にサーバーを設置し、サーバー上からパケット通信により家電の通信仕様をダウンロードする。

3. システム要件

3.1. 携帯電話の要件

- 携帯については、音声応答機能付きの携帯を前提条件とする。また、らくらくホンシリーズのうち、ユーザープログラムが利用できる DoJa-5.1 プロファイルと DoJa-5.1LE プロファイルに対応した機種とする。2011 年 9 月現在、以下の表＊の機種となる。

製品名	型番	Java プロファイル
らくらくホンプレミアム	F884i	DoJa 5.1LE
らくらくホンベーシックⅢ	F-08C	DoJa 5.1LE
らくらくホンV	F884i-ES	DoJa 5.1LE
らくらくホン 6	F-10A	DoJa 5.1LE
らくらくホン 7	F-09B	DoJa 5.1

- CIR（単方向）送信機能および IrDA（双方向）送受信機能があること。
- パケット通信機能があること。
- ユーザインターフェースは､ユニバーサルデザインが十分に考慮されたものであること。

3.2. 家電の要件

- ボタンが 63 個以内であること。なお、ボタンが 63 個以上の場合、一部の機能が再現できなくなる可能性がある。

3.3. おまかせミールさん用の要件

本研究の研究対象であるおまかせミールさんの機能、制約条件は以下のようになる。

3.3.1. 調理機能

おまかせミールさんには目玉焼き（0～2 個)、トースト（0～2 枚)、コーヒーが一度にできる。一品だけ調理したい時、調理時間や焼き加減なども設定できる。

3.3.2. 予約機能

仕上がり時間の予約も可能である。全ての料理を同時に仕上がるよう、それぞれの個数に応じて、各ユニットが逆算して調理を開始できる。

3.3.3. 現在時刻設定機能

電源を入れるたびに現在時刻を設定する必要がある。設定しない場合、予約機能とアラーム機能が使えなくなる。

3.3.4. 目覚まし機能

おまかせミールさんには目覚まし機能が付いている。料理が仕上がる前後にアラームなどが設定できる。

3.3.5. 制約条件

おまかせミールさんには四つ制約条件がある。

- 現在時刻を覚える機能がないため、電源を入れた後に必ず現在時刻を設定し直さなければならない。
- 焼き加減、目玉焼きやトーストの枚数が同時に設定できるが、調理時間はこれらと一緒に設定できない。
- 目玉焼き、トーストを一緒に作る時、調理時間が設定できない。焼き加減が同じになる。
- 仕上がり時間が現在時間プラス調理所要時間より前に設定した時、仕上がりが翌日の設定時間となる。

3.4. その他要件

- サーバーはiアプリ、通信仕様をアップロード、ダウンロードできること。

4. 機能要件

4.1. 携帯電話用端末ソフトウェア

携帯電話に搭載すべき機能は以下の通りとなる。

4.1.1. 通信仕様解析機能

家電、またはサーバー上からダウンロードした通信仕様を読み込み、解析するための機能。通信仕様には家電のメニュー画面を描画するための情報が記載されており、その情報を解析することで、家電に対応したメニュー画面を描画するための情報を取得する。

4.1.2. 命令コマンド作成機能

家電を操作するための命令コマンドを作成するための機能。通信仕様にはメニュー画面を描画するための情報の他、家電の操作メニューに対応した命令コマンドが記載されている。その情報から選択した操作メニューに応じた命令コマンドを作成する。

4.1.3. 画面描画機能

家電のメニュー画面を描画するための機能。4.1.1 の機能から解析した情報を用いて、家電のメニュー画面を描画する。

4.1.4. 音声読み上げ機能

家電のメニュー等を音声読み上げさせるための機能。目の不自由な方でも操作できるように適切な音声を読み上げさせる。

4.1.5. IrDA 送受信機能

家電と IrDA 通信にてオブジェクトの送受信を行うための機能。家電とやり取りするオブ

ジェクトは家電についての情報・通信仕様・家電を操作するための命令コマンドである。

4.1.6. CIR 送信機能

家電と CIR 通信にて家電を操作するための命令コードを送信するための機能。命令コマンドは NEC フォーマットにて 32bit のデータを送信する。

4.1.7. HTTP 通信機能

サーバー上から通信仕様をダウンロードするための機能。ダウンロードした通信仕様は 4.1.1 の機能に渡される。

4.2. 家電用通信ファームウェア

家電に搭載すべき機能は以下の通りとなる。

4.2.1. IrDA 送受信機能

携帯電話と IrDA 通信にてオブジェクトの送受信を行うための機能。携帯電話とやり取りするオブジェクトは家電についての情報・通信仕様・家電を操作するための命令コマンドである。

4.2.2. CIR 受信機能

携帯電話から CIR 通信にて家電を操作するための命令コードを受信するための機能。命令コマンドは NEC フォーマットにて 32bit のデータを受信する。

4.3. おまかせミールさん用カスタマイズ

4.3.1. おまかせミールさんの状態読み取り機能

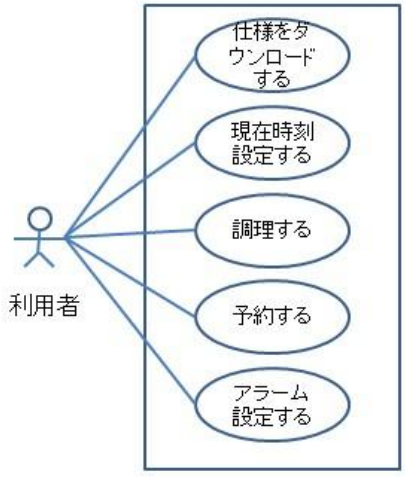
現在のおまかせミールさんの情報（LED の点灯）を取得するための機能を持つ。

4.3.2. おまかせミールさん制御機能

携帯電話から受け取った命令コマンドを解釈し、それを疑似ボタンとして入力するための機能を持つ。

4.3.3. ユースケース図

おまかせミールさんのユースケースは下の図に示します。おまかせミールさんが持っている機能の他、携帯から操作するために家電の仕様をダウンロードする機能もユースケース図に入れる。



5. 非機能要件

5.1. 効率性

赤外線通信速度

主記憶使用量

5.2. 機能性

目の不自由な人のことを十分考慮し、システムを設計する。そして、家電の機能を全て携帯上で再現できるシステムを作る。

5.3. 安全性

家電が誤動作しないように、コマンド表を十分考えて設計する。こうしたことで、システム設計により誤動作の可能性を最低まで下がる。家電が暴走しないようにシステムを開発する。

5.4. 移植性

本研究は研究対象であるおまかせミールさん以外の家電でも使えるための移植性を持たせる。携帯側のソフトウェアについては、対象家電が変わった際にも初手から作り直すことがないように、対象家電の仕様（UI 仕様、制御仕様）を携帯に読み込ませることで、汎用的に制御することができるものを開発する。家電側については、家電の制御機能はおまかせミールさん用にカスタマイズされたものを開発するが、将来的にはこの制御機能については、家電側マイコンのソフトウェアに統合されることを想定している。このため、通信（CIR 受信、IrDA 送受信）に関する機能はどの家電でも汎用的に利用できるものを開発し、またそのリソース（ROM/RAM 容量、入出力ポート数、割り込みチャネル数及び割り込みレベル、タイマー）を小さく保つように設計し、開発する。

5.5. 使用性

コンセプト明快性（UI 設計）

デモ、ソフトウェア装備率

マニュアル使いやすさ

オペレーション習熟時間

誤操作

5.6. 信頼性

安全性向上のために、IrDA（双方向）の場合、家電に送るコマンドはちゃんと確認して実行する。CIR（単方向）の場合、コマンドの漏れのないようにコマンドを 3 回送信する。

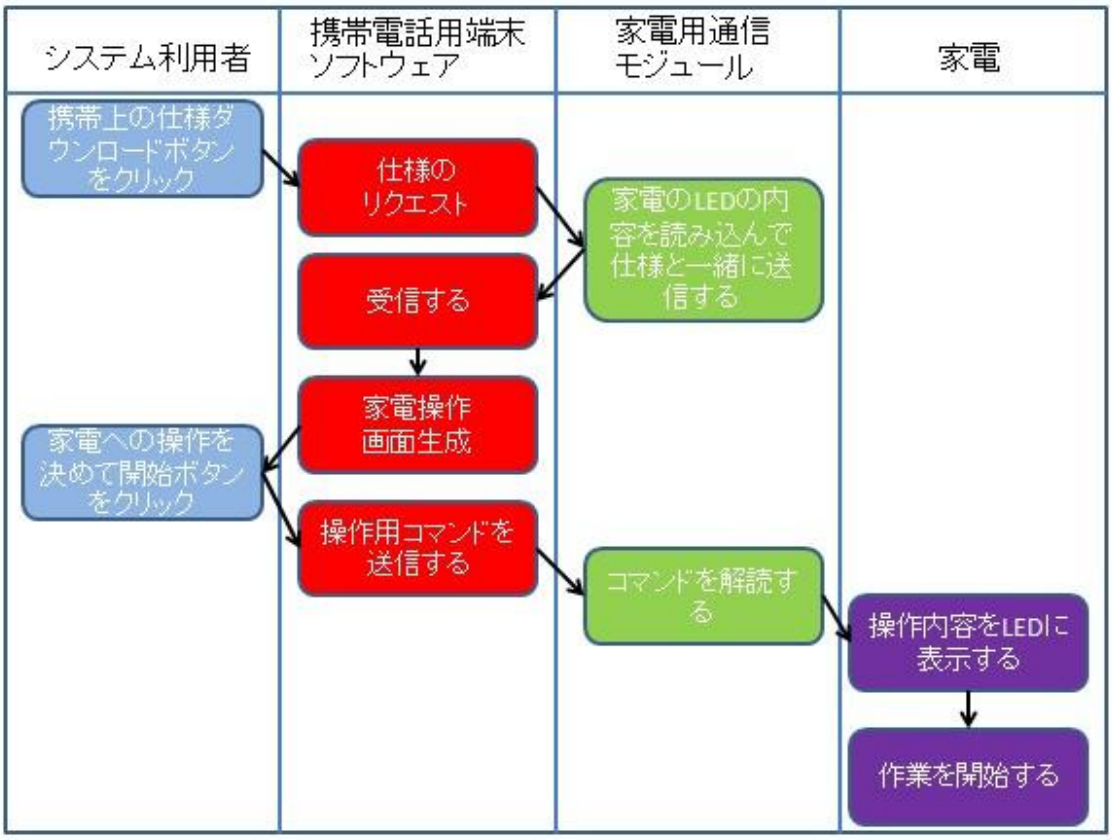
5.7. 操作性

ユーザインターフェースは目の不自由な方にもわかりやすく操作できるように十分考慮したものにする

6. システム導入後の流れ

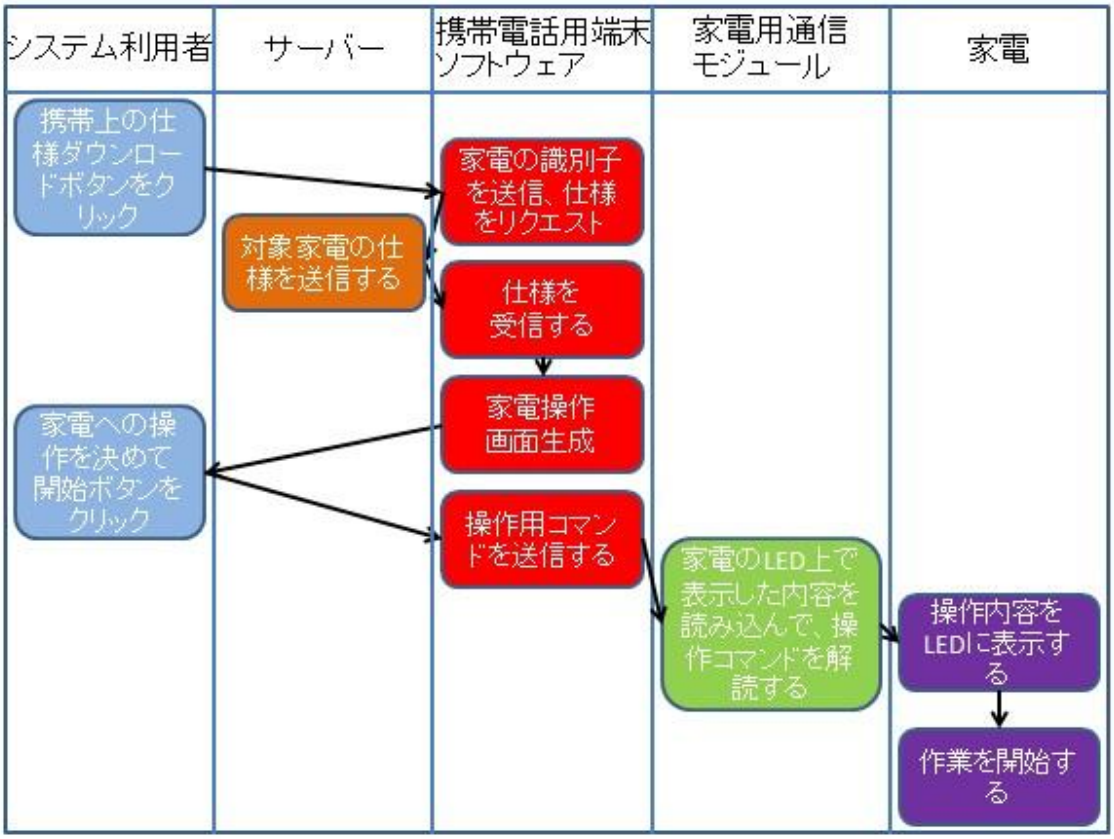
6.1. IrDA

IrDA で家電をコントロールする時の流れを下の図に示します。図で表示してないが、家電用通信モジュールと携帯電話用端末ソフトウェアで送受信する時、内容の確認を行ってから作業を開始する。



6.2. CIR

CIR で家電をコントロールする時の流れを下の図に示します。IrDA と違って、CIR は単方向通信のため、家電の仕様はインターネットに設置するサーバーからダウンロードする。図で表示してないだが、携帯電話用端末ソフトウェアから家電用通信モジュールにコマンドを送る時、ちゃんとコマンドが届くため、3 回同じ内容を送信し、3 回の中の 2 回の内容が同じでしたら作業を開始する。



7. 開発推進体制

7.1. 開発環境

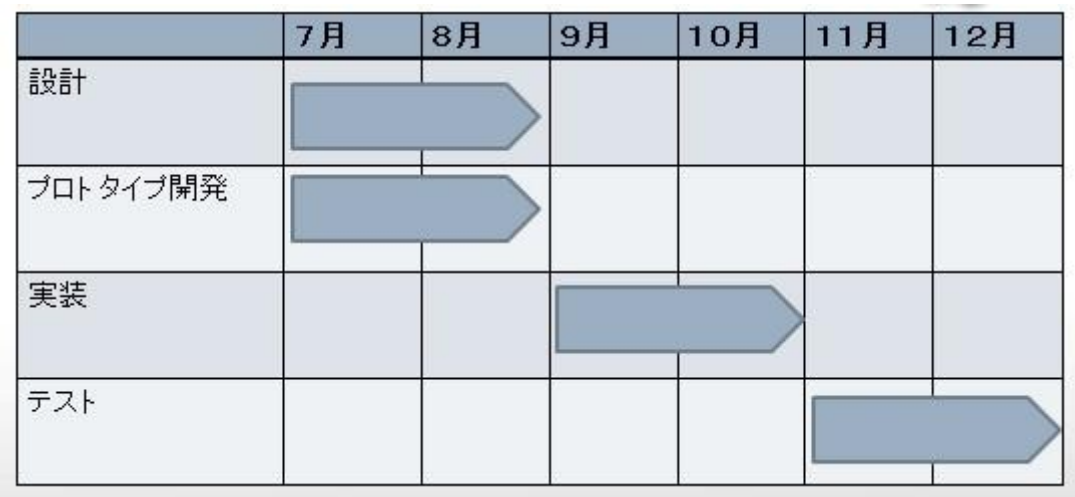
7.1.1. 携帯側

- 端末は F-10A を使用する
- 開発言語は JAVA 言語を使用する。
- 開発環境には Eclipse3.6.2 Helios Service Release 2(jre6/Jdk1.6)および DoJa-5.1 Environment を利用する。

7.1.2. 家電 Daughter Board 側

- マイコンは PIC24FJXXGA002 を使用する。
- 開発言語は C 言語を使用する。
- 開発環境として MPLAB IDE を利用する。

7.2. 開発大日程



8. 初期導入費、維持費

・ボード製作費

・携帯電話（らくらくホン）

仕様をダウンロードするサーバーには無料サーバーを使う。

一方で、家電と連携用の携帯、基板、CIR 受信機、IrDA 送受信機などは祓川先生に負担していただくようお願いいたします。

また、開発に関わる人件費等の費用は学習を目的としているため、無料とさせていただきます。

9. 保守・運用について

本システムの納入から私たちが卒業するまでの間は不具合の修正やトラブルの対応などの保守・運用を行っていきます。この間は私たちチーム OMAKASE のメンバの在学期間であり、各メンバが卒業した後は保守・運用をすることが出来なくなってしまいます。そのため、卒業後は、祓川先生自身もしくは他の業者に保守・運用をお任せする形になります。

付録 B

IOC-125　基板の解析

IOC-125　プリント基盤の解析

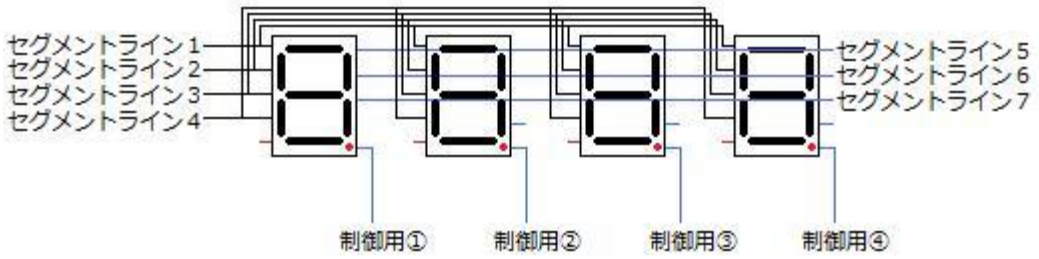
- LED のダイナミック点灯とは何か。

　７セグ LED を例に挙げて説明する。

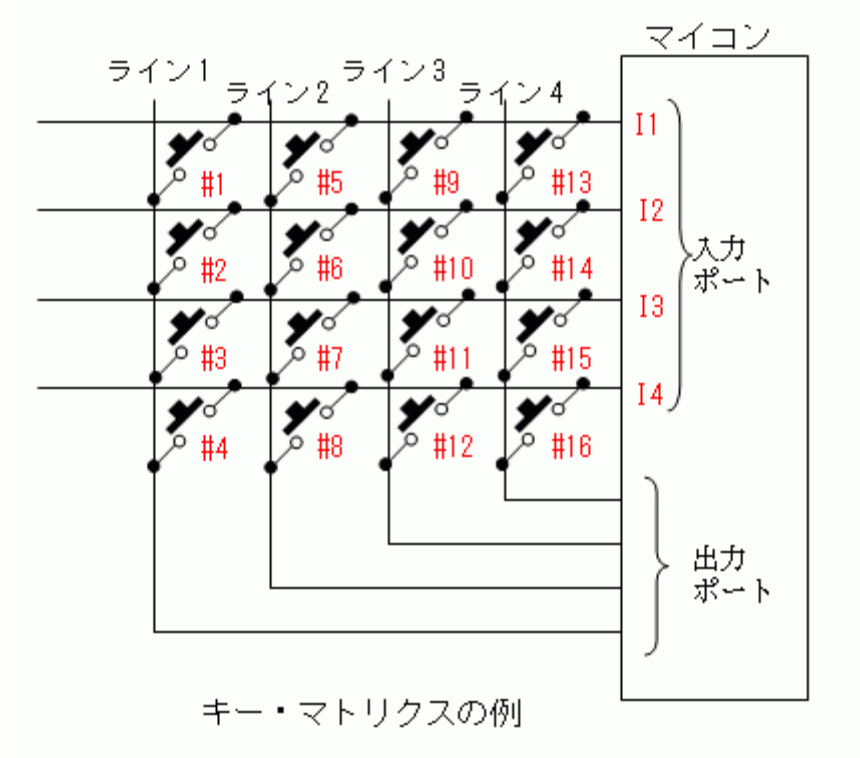
　７セグ LED をスタティック点灯で制御すると、桁数×７個の電源ラインが必要になり、また制御用 IC の数も増え無駄が生じてしまいます。そこで、人間の目の錯覚（人間の目には残像現象があり、一度光を見ると、50msec～100msec の間はその光を連続してみているように錯覚します）を利用して、１つの LED を一瞬だけ点灯させて、次から次へと点灯させます。これをダイナミック点灯方式と呼びます。

　具体例を挙げると、４桁の７セグ LED をダイナミック点灯方式で点灯する場合、左の桁から１桁ずつ必要とする数字を一瞬表示させて一度消したら、右の桁に移り一瞬表示、一度消すと次から次へと表示させ消すという動作を行います。そして、一番右の桁までいったら一番左の桁にもどり、また 1 桁ずつ表示させていくことで 4 桁の７セグ LED すべてを表示させているように見せます。

　この方式を用いることにより、７セグ LED 制御用の７つのラインとそれぞれの桁数を表示させるためのライン（桁数分。具体例でいうと、４つ）があれば十分であるため、とても経済的である。



- キーマトリクスにおけるダイオードの役割はなにか



キー・マトリクスの例

①キーを２つ同時に押したときに出力同士が衝突するのを防ぐ。

＃１と＃５が同時に押された場合、ライン１とライン２が＃１と＃５を介して接続された状態になる。そのため、出力ポートがＣＭＯＳ出力である場合、ハイレベルの出力とロウレベルの出力がぶつかることになるため、デバイスにダメージを与えたり、キー・スイッチに定格以上に電流が流れたりする可能性がある。

②キーを３つ以上同時に押したときの誤動作を防ぐ。

＃１と＃５、＃６が同時に押された場合、ライン１をスキャンしているときに＃１と＃５を介して、ライン２もロウレベルとなる。この状態でライン２に接続されている＃６を押すと、Ｉ２入力もロウレベルになります。これにより＃２も押されたと間違われます。

①、②を防ぐのがダイオードの役割となる。

- インターフェース基盤の工夫
  - 時刻表示の最上位桁が「８」でないこと。

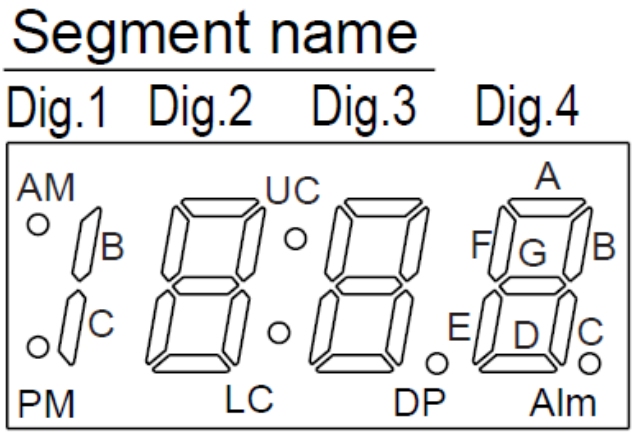
    ＡＭ／ＰＭ表示があるため、時刻の最大表示は１２時。また、調理に必要な時間も１９分５９秒以内で十分であるため。
  - 外付けのＬＥＤが卵焼き、トースト、コーヒーの３つであること。

    調理中にＬＥＤを点灯させ、調理終了時にＬＥＤが消えるようにしている。
  - 抵抗が８個であること。

    ７セグ LED の制御を安定させるために、８個の抵抗が使われている。

    それぞれの抵抗は以下の LED に接続されている

    R1→
    AM/PM/Alm/UC/LC
    R2→G
    R3→F
    R4→E
    R5→D
    R6→C
    R7→B
    R8→A



  - ダイオードが４個であること。

    ①７セグ LED の桁数の切り替えの制御を安定させるため

    ②スイッチに流れる電流の逆流を防ぐため

    以上の２つの理由で４つのダイオードが使われている。

    また、それぞれのダイオードに接続されている７セグ LED の桁数、スイッチを以下に示す。

    D1→Dig.3　　D1→SW4/SW5/SW6
    D2→Dig.1　　D2→SW1/SW2/SW3
    D3→Dig.2　　D3→SW7/SW8/SW9
    D4→Dig.4　　D4→SW10/SW11/SW12
  - タクトスイッチが１２個であること。

    IOC-125 の制御に１２個のボタンが使われているため。

    以下の１２個のボタンである。

    卵焼き／トースト／コーヒー／淡・濃／時計・予約／１分/時・１０秒/分／アラーム／スタート／とりけし

- フラットケーブルが１５ピンであること。

  ７セグ LED の制御に８ピン、スイッチの制御に７ピン使用されている。

  ７セグ LED の制御に８ピン使用する理由は、上記の抵抗が８個の理由と同様のため説明は省略する。

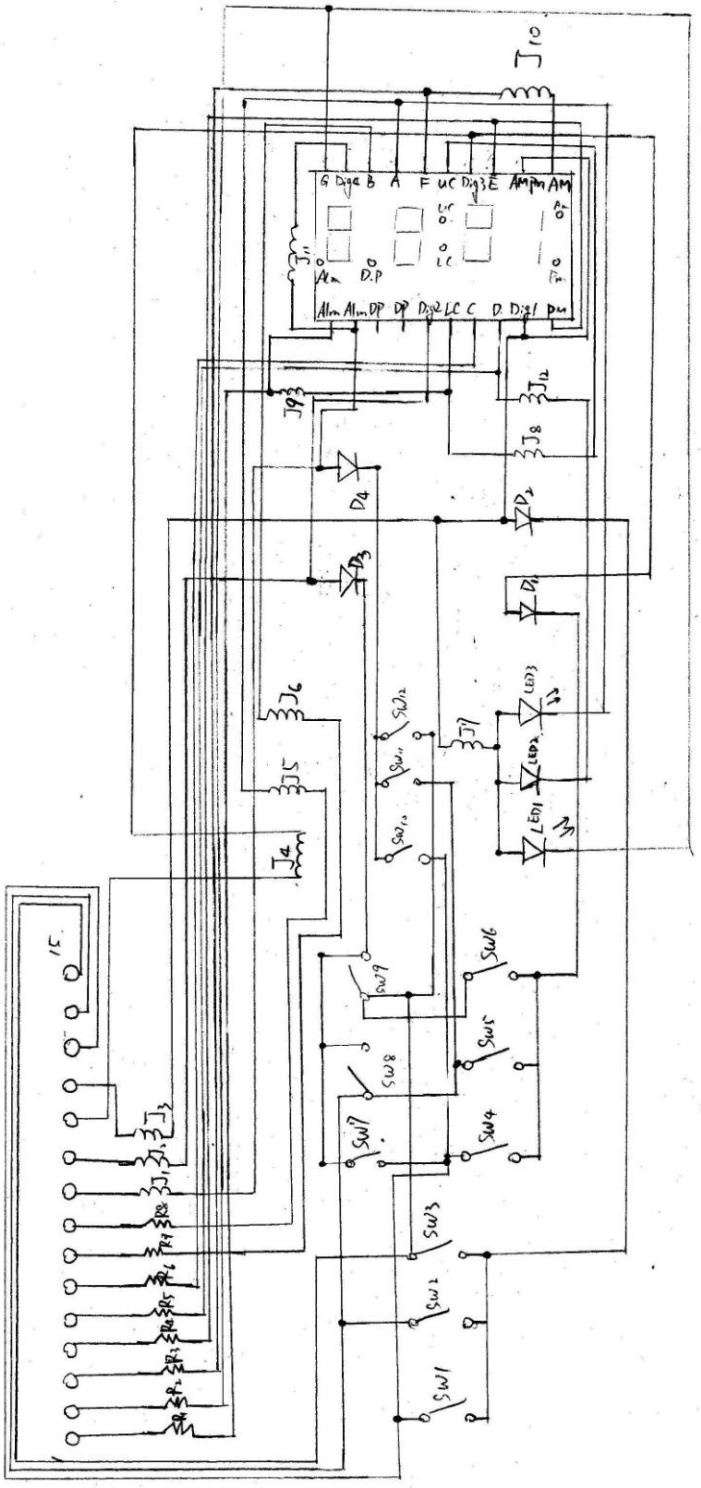
  スイッチの制御に７ピン必要な理由は、１２個のタクトスイッチを４×３のキーマトリクスで制御しているためである。

  つまり、以下の図のような形式でタクトスイッチ１２個を７ピンで制御している。

	１３ピン	１４ピン	１５ピン
１２ピン（D2）	SW1	SW2	SW3
１１ピン（D1）	SW4	SW5	SW6
１０ピン（D3）	SW7	SW8	SW9
９ピン（D4）	SW10	SW11	SW12

付録 C
IOC-125
LEDSW Board の回路図

解析した IOC-125 の LED/SW ボードの回路図を示す。



付録 D

ユースケース記述

ユースケース名	調理する
アクター	ユーザ
概要	ユーザが携帯を使って調理する
前提条件	仕様がプログラムに読み込まれた状態
基本フロー	1. ユーザが「料理を作る」ボタンをクリックする 2. 携帯が調理メニュー設定画面を表示する 3. ユーザが調理メニューを設定し、「次へ」ボタンをクリックする 4. 携帯が設定内容を記録し、焼き加減設定画面を表示する 5. ユーザが焼き加減を設定し、「次へ」ボタンをクリックする 6. 携帯が設定内容を記録し、設定確認画面を表示する 7. ユーザが「完了ボタン」をクリックする 8. 携帯が最終画面を表示し、設定内容を家電に送る 9. 家電が設定内容を受信する 10. 家電が調理を開始する
代替フロー	6a.ユーザが入力した焼き時間に入力エラーが発生した場合 1. 焼き時間以外の設定内容を記録したまま、もう一度焼き加減設定画面を表示する。エラーの内容も画面上に表示する。
事後条件	なし

ユースケース名	調理を予約する
アクター	ユーザ
概要	ユーザが携帯を使って調理の予約をする
前提条件	仕様がプログラムに読み込まれた状態
基本フロー	1. ユーザが「料理を作る」ボタンをクリックする 2. 携帯が調理メニュー設定画面を表示する 3. ユーザが調理メニューを設定し、「次へ」ボタンをクリックする 4. 携帯が設定内容を記録し、焼き加減設定画面を表示する 5. ユーザが焼き加減を設定し、「次へ」ボタンをクリックする 6. 携帯が設定内容を記録し、予約・目覚まし画面を表示する 7. ユーザが予約・目覚ましを設定し、「次へ」ボタンをクリックする 8. 携帯が設定内容を記録し、設定確認画面を表示する 9. ユーザが「完了ボタン」をクリックする 10. 携帯が最終画面を表示し、設定内容を家電に送る 11. 家電が設定内容を受信する 12. 家電が調理の予約を開始する
代替フロー	6a.ユーザが入力した焼き時間に入力エラーが発生した場合 1. 焼き時間以外の設定内容を記録したまま、もう一度焼き加減設定画面を表示する。エラーの内容も画面上に表示する。 8a.ユーザが入力した目覚まし時間に入力エラーが発生した場合 1. 時間以外の設定内容を記録したまま、もう一度予約・目覚まし設定画面を表示する。エラーの内容も画面上に表示する。
事後条件	なし

ユースケース名	現在時刻設定
アクター	ユーザ
概要	ユーザが携帯を使って現在時刻を設定する
前提条件	仕様がプログラムに読み込まれた状態
基本フロー	1. ユーザが「時間を設定する」ボタンをクリックする 2. 携帯が時間設定画面を表示する 3. ユーザが「現在時刻を設定する」ボタンをクリックする 4. 携帯が携帯の時刻を家電に送る 5. 家電が LED を読み取り、現在時刻を設定する
代替フロー	
事後条件	なし

ユースケース名	目覚まし設定
アクター	ユーザ
概要	ユーザが携帯を使って目覚ましを設定する
前提条件	仕様がプログラムに読み込まれた状態
基本フロー	1. ユーザが「時間を設定する」ボタンをクリックする 2. 携帯が時間設定画面を表示する 3. ユーザが「目覚ましを設定する」ボタンをクリックする 4. 携帯が目覚まし設定画面を表示する 5. ユーザが目覚ましの時刻を入力し、「設定する」ボタンをクリックする 6. 携帯が最終画面に表示し、設定内容を家電に送る 7. 家電が LED を読み取り、目覚ましを設定する
代替フロー	6a.ユーザが入力した目覚まし時間に入力エラーが発生した場合 1. 携帯がもう一度目覚まし設定画面を表示する。エラーの内容も画面上に表示する。
事後条件	なし

ユースケース名	仕様をダウンロードする
アクター	ユーザ
概要	ユーザが携帯を使って仕様をダウンロードする
前提条件	
基本フロー	1. ユーザがプログラムを起動する。 2. 携帯がメイン画面を表示する。 3. ユーザは「仕様を取得」を選択する。 4. 携帯が仕様のリストを表示する。 5. ユーザは取得したい仕様を選択する。 6. 携帯は仕様をダウンロードして、家電の操作画面を表示する。さらに、取得した仕様は SD カードに保存する。
代替フロー	6a. 既にユーザが取得したい仕様がダウンロードされている場合、SD カードから仕様を読み込み、家電の操作画面を表示する。
事後条件	仕様が携帯に保存されている

ユースケース名	家電の ID を取得する
アクター	ユーザ
概要	ユーザが携帯を使って家電の ID を取得
前提条件	
基本フロー	1. ユーザがプログラムを起動する。 2. 携帯がメイン画面を表示する。 3. ユーザは「家電の情報を取得」をクリックする。 4. 携帯は ID を家電に要求する。 5. 家電は ID を携帯に返す。 6. 携帯は仕様の取得確認画面を表示する。 7. ユーザは「はい」を選択する。 8. 家電と IrDA 通信を行い、仕様が保存されている ID を取得する。 9. 携帯は取得した ID から仕様をダウンロードして、家電の操作画面を表示する。さらに、取得した仕様を SD カードに保存する。
代替フロー	7a. ユーザは「いいえ」を選択する。 1. 基本フローの 2 に戻る。 9a. 既にユーザが取得したい仕様がダウンロードされている場合、SD カードから仕様を読み込み、家電の操作画面を表示する。
事後条件	仕様が携帯に保存されている

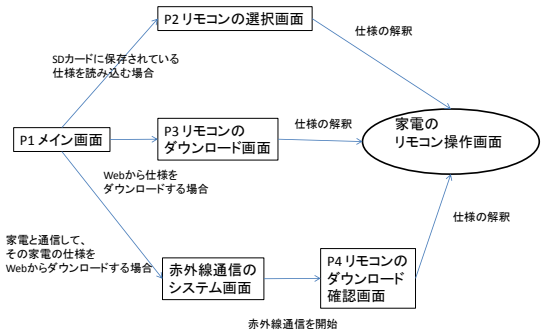
ユースケース名	取り消す
アクター	ユーザ
概要	ユーザが携帯を使って調理と予約の取り消しをする
前提条件	仕様がプログラムに読み込まれた状態
基本フロー	1. ユーザが「現作業中止」ボタンをクリックする 2. 携帯が最終画面を表示し、設定内容を家電に送る 3. 家電が設定内容を受信する 4. 家電が調理と予約を取り消す
代替フロー	
事後条件	

ユースケース名	仕様を読み込む
アクター	ユーザ
概要	ユーザが携帯を使っておまかせミールさんの仕様を読む
前提条件	仕様がダウンロードされた状態
基本フロー	1. ユーザがプログラムを開く 2. 携帯がメイン画面を表示する 3. ユーザは「家電の操作」を選択する。 4. 携帯が操作できる家電の一覧を表示する 5. ユーザがおまかせミールさんを選ぶ 6. 携帯がおまかせミールさんの仕様を読み、おまかせミールさんの操作画面を表示する。
代替フロー	
事後条件	仕様がプログラムに読み込まれる

付録 E

画面イメージ図

画面イメージ図
- TOP画面-



P1

メインメニュー

リモコンの選択

リモコンの
ダウンロード

家電情報の取得

P2

リモコンの選択

①おまかせ
ミールさん

②電子レンジ

③洗濯機

④HDDレコーダ

P3

リモコンの
ダウンロード

①おまかせ
ミールさん
②電子レンジ
③洗濯機
④HDDレコーダ

α赤外線
通信しますか？

①通信する
②通信しない

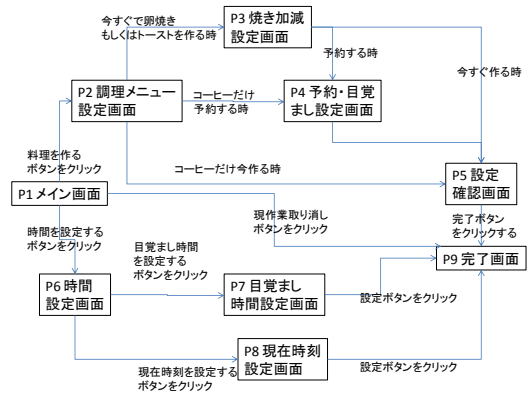
携帯の
システム側で描画

P4

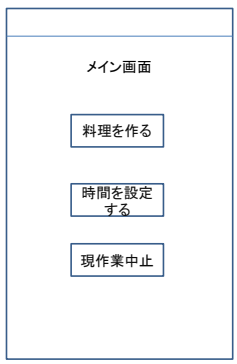
IOC-125 おまかせミールさんのリモコンをダウンロードしますか？

①はい
②いいえ

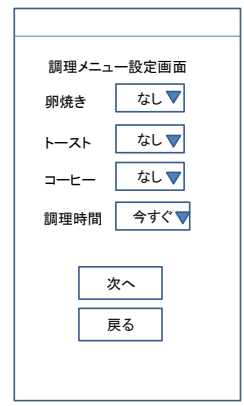
画面イメージ図
- おまかせミールさん -



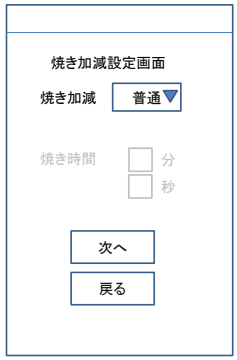
P1



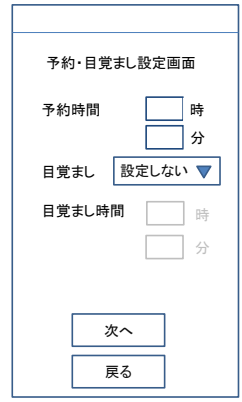
P2



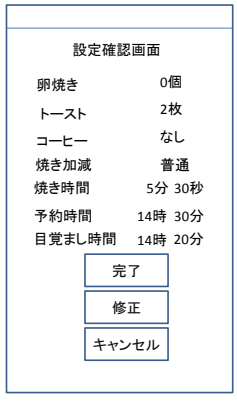
P3


焼き加減設定画面
焼き加減
普通
焼き時間
分
秒
次へ
戻る

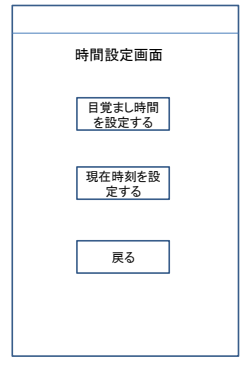
P4


予約・目覚まし設定画面
予約時間
時
分
目覚まし
設定しない
目覚まし時間
時
分
次へ
戻る

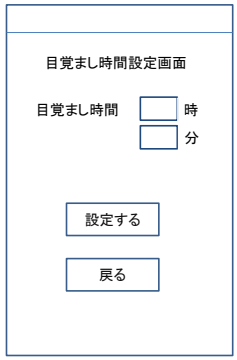
P5


設定確認画面
卵焼き 0個
トースト 2枚
コーヒー なし
焼き加減 普通
焼き時間 5分 30秒
予約時間 14時 30分
目覚まし時間 14時 20分
完了
修正
キャンセル

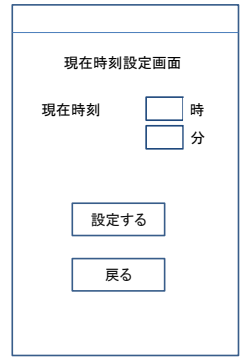
P6


時間設定画面
目覚まし時間を設定する
現在時刻を設定する
戻る

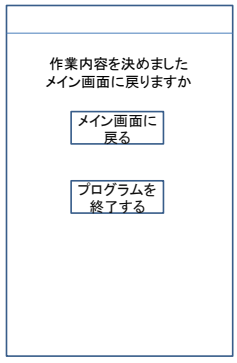
P7


目覚まし時間設定画面
目覚まし時間
時
分
設定する
戻る

P8


現在時刻設定画面
現在時刻
時
分
設定する
戻る

P9


作業内容を決めました
メイン画面に戻りますか
メイン画面に戻る
プログラムを終了する

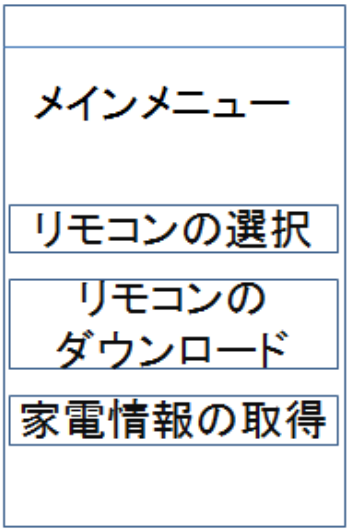
付録 F

画面設計書

画面ID　P1
画面名　メイン画面
画面概要　リモコンを新たに取得するか、保存してあるリモコンを呼び出すか決める

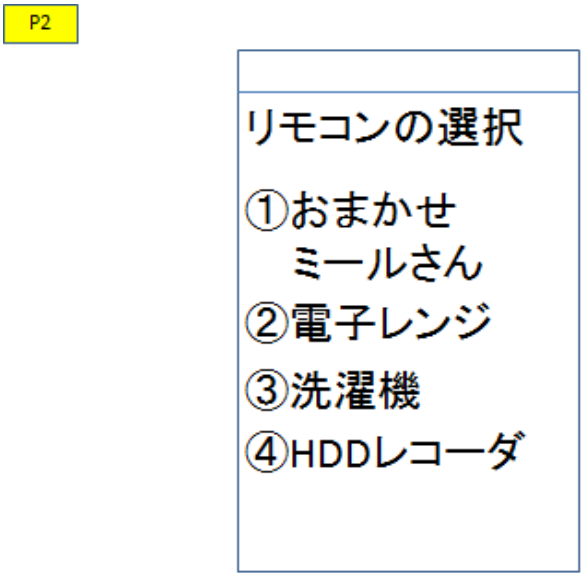
項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	リモコンの選択	ボタン	−	なし	P2に遷移	
2	リモコンのダウンロード	ボタン	−	なし	P3に遷移	
3	家電情報の取得	ボタン	−	なし	P4に遷移	

P1



画面ID　P2
画面名　リモコンの選択画面
画面概要　リモコンを選択する

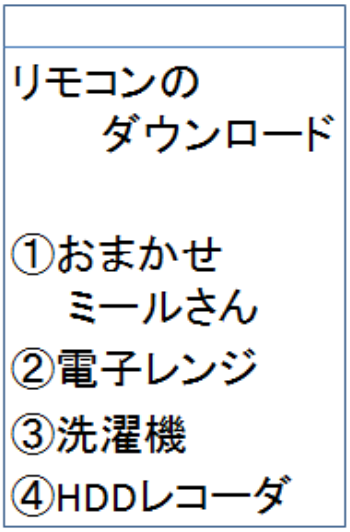
項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	家電1	リスト	－	なし	家電1のリモコン	リモコンの操作画面は、
2	家電2	リスト	－	なし	家電2のリモコン	リモコンの操作画面は、
3	家電3	リスト	－	なし	家電3のリモコン	リモコンの操作画面は、
・	・	・	・	・	・	・
・	・	・	・	・	・	・
・	・	・	・	・	・	・



画面ID　P3
画面名　リモコンのダウンロード画面
画面概要　リモコンをダウンロードする

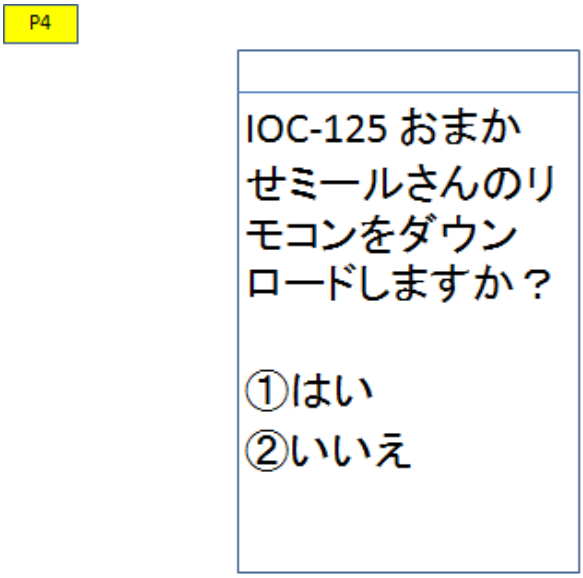
項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	家電1	リスト	−	なし	家電1の仕様をSDカードに保存。 家電1のリモコン	リモコンの操作画面は、 仕様を読み込んで出力
2	家電2	リスト	−	なし	家電2の仕様をSDカードに保存。 家電2のリモコン	リモコンの操作画面は、 仕様を読み込んで出力
3	家電3	リスト	−	なし	家電3の仕様をSDカードに保存。 家電3のリモコン	リモコンの操作画面は、 仕様を読み込んで出力
・	・	・	・	・	・	・
・	・	・	・	・	・	・
・	・	・	・	・	・	・
・	・	・	・	・	・	・

P3



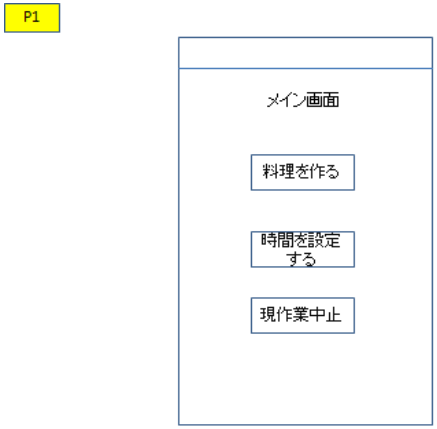
画面ID P4
画面名 仕様のダウンロード確認画面
画面概要 家電とIrDA通信後に、Webから仕様をダウンロードする際の確認画面

項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	家電1のリモコンをダウンロードします	ラベル	–	–	–	
2	はい	ボタン	–	なし	家電1の仕様をSDカードに保存。	リモコンの操作画面は、仕様を読み込んで出力
3	いいえ	ボタン	–	なし	家電1のリモコンP1に遷移	



画面ID　P1
画面名　メイン画面
画面概要　料理を作るか時間を設定するかを決める

項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	料理を作る	ボタン	-	なし	P2に遷移	
2	時間を設定する	ボタン	-	なし	P6に遷移	
3	現作業中止	ボタン	-	なし	作業をキャンセルするためのコマンドを出力する P9に遷移	



画面ID　P2
画面名　調理メニュー設定画面
画面概要　料理の具体設定ページ

項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	卵焼き	ドロップダウンリスト	0	なし	なし	0，1，2が選べる
2	トースト	ドロップダウンリスト	0	なし	なし	0，1，2が選べる
3	コーヒー	ドロップダウンリスト	なし	なし	なし	なし、ありが選べる
4	調理時間	ドロップダウンリスト	今すぐ	なし	なし	今すぐ、予約が選べる
5	次へ	ボタン	-	なし	コーヒーだけ今すぐ作る場合P5に遷移 コーヒーだけ予約する場合P4に遷移 それ以外の場合P3に遷移	卵焼き、トースト、コーヒーが共に設定されていない場合にdisabled 背景色を灰色にする
6	戻る	ボタン	-	なし	P1に遷移	

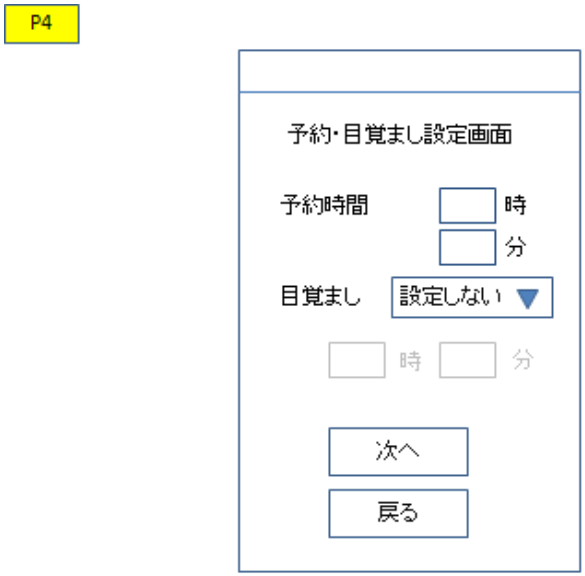
P2

画面ID　P3
画面名　焼き加減設定画面
画面概要　料理の焼き加減を設定する

項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	焼き加減	ドロップダウンリスト	普通	なし	なし	普通、濃、淡、調理時間設定が選べる
2	焼き時間	グループ	–	–	–	予約する時、またはトーストと卵焼き焼き加減が普通、濃、淡のいずれかにした時disabled、背景色が灰色に
3	焼き時間:分	テキストボックス	–	卵焼きの場合 4–10の整数 トーストの場合 1–10の整数	なし	
4	焼き時間:秒	テキストボックス	–	卵焼きの場合 0–50の10の倍数 トーストの場合 0–50の10の倍数	なし	
5	次へ	ボタン	–	–	予約の場合P4に遷移 今すぐ作る場合P5に遷移	
6	戻る	ボタン	–	–	P2に遷移	

画面ID　P4
画面名　予約・目覚まし設定画面
画面概要　料理を仕上げる時間と目覚まし時間を設定する

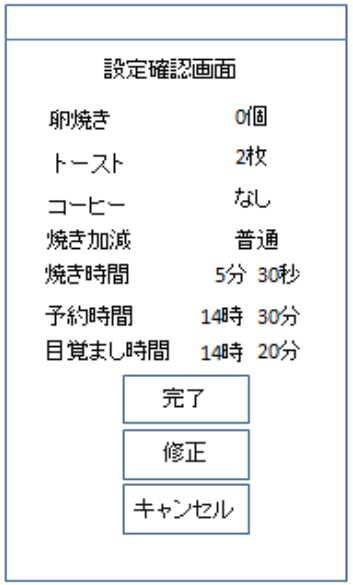
項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	予約時間	グループ	−	−	−	
2	予約時間：時	テキストボックス	−	0−23の整数	−	
3	予約時間：分	テキストボックス	−	0−59の整数	−	
4	目覚まし	ドロップダウンリスト	設定しない	なし	なし	設定しない、設定するが選べる
5	目覚まし時間	グループ	−	−	−	
6	目覚まし時間：時	テキストボックス	−	0−23の整数	なし	
7	目覚まし時間：分	テキストボックス	−	0−59の整数	なし	
8	次へ	ボタン	−	−	P5に遷移	予約時間を設定しないとdisabled 背景色を灰色にする
9	戻る	ボタン	−	−	P3に遷移	



画面ID　P5
画面名　設定確認画面
画面概要　設定した内容を確認する画面

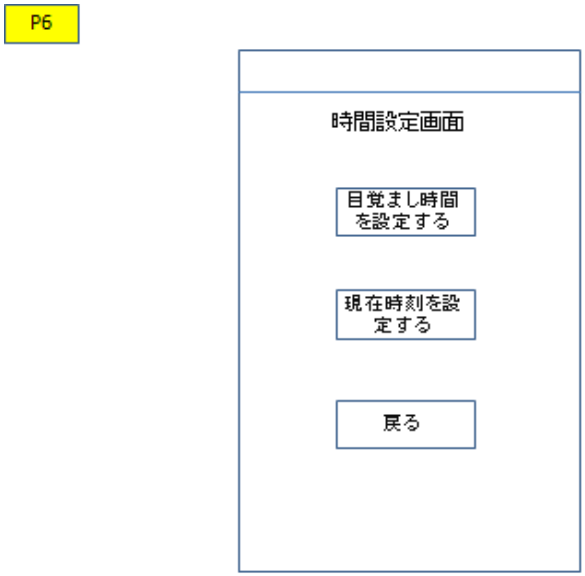
項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	卵焼き	ラベル	−	−	−	0枚の場合、非表示
2	トースト	ラベル	−	−	−	0個の場合、非表示
3	コーヒー	ラベル	−	−	−	なしの場合、非表示
4	焼き加減	ラベル	−	−	−	設定されてない場合、非表示
5	焼き時間	グループ	−	−	−	設定されてない場合、非表示
6	焼き時間:分	ラベル	−	−	−	
7	焼き時間:秒	ラベル	−	−	−	
8	予約時間	グループ	−	−	−	設定されてない場合、非表示
9	予約時間:時	ラベル	−	−	−	
10	予約時間:分	ラベル	−	−	−	
11	目覚まし時間	グループ	−	−	−	設定されてない場合、非表示
12	目覚まし時間:時	ラベル	−	−	−	
13	目覚まし時間:分	ラベル	−	−	−	
14	完了	ボタン	−	−	コマンドを出力する	
15	修正	ボタン	−	−	P2に遷移	
16	キャンセル	ボタン	−	−	P1に遷移	

P5



画面ID　P6
画面名　時間設定画面
画面概要　アラームを設定するか、現在時刻を設定するかを決める

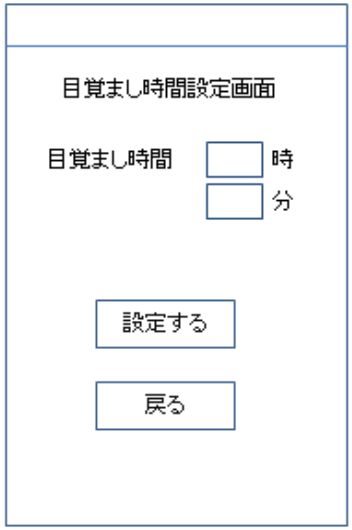
項目ID	項目名	項目種別	初期値	カチェック	出力	備考
1	目覚まし時間を設定する	ボタン	−	−	P7に遷移	
2	現在時刻を設定する	ボタン	−	−	P8に遷移	
3	戻る	ボタン	−	−	P1に遷移	



画面ID　P7
画面名　目覚まし時間設定画面
画面概要　目覚まし時間を決める

項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	目覚まし時間	グループ	-	-	-	
2	目覚まし時間:時	テキストボックス	-	0-23の整数	-	
3	目覚まし時間:分	テキストボックス	-	0-59の整数	-	
4	設定する	ボタン	-	-	目覚まし時間を設定するコマンドを出力する	目覚まし時間を設定しないとdisabled背景色を灰色にす
5	戻る	ボタン	-	-	P1に遷移	

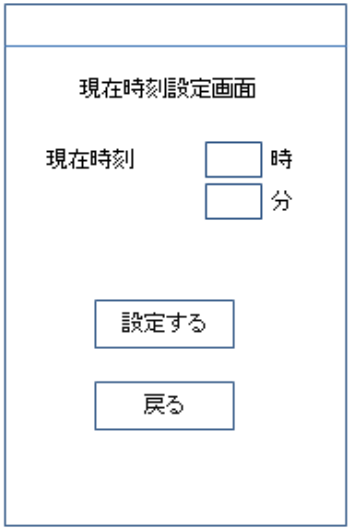
P7



画面ID　P8
画面名　現在時刻設定画面
画面概要　現在時刻を決める

項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	現在時刻	グループ	−	−	−	
2	現在時刻:時	テキストボックス	−	0-23の整数	−	
3	現在時刻:分	テキストボックス	−	0-59の整数	−	
4	設定する	ボタン	−	−	現在時刻設定のコマンドを出力する	現在時刻を設定しないとdisabled 背景色を灰色にする
5	戻る	ボタン	−	−	P1に遷移	

P8



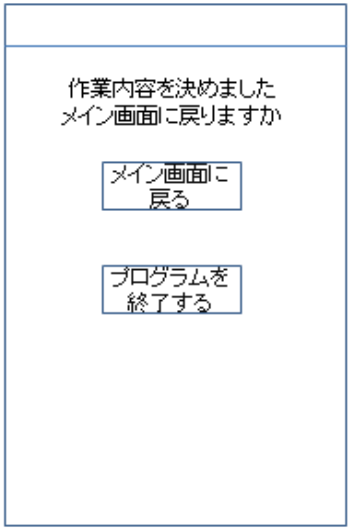
画面ID　P9

画面名　終了画面

画面概要　メイン画面に戻るか、プログラムを終了するかを決める

項目ID	項目名	項目種別	初期値	入力チェック	出力	備考
1	メイン画面に戻る	ボタン	-	なし	P1に遷移	
2	プログラムを終了する	ボタン	-	なし	プログラム終了	

P9



付録 G

メッセージ定義書

メッセージID	メッセージ名	内容	置換文字列の説明	画面ID	備考
EMSG01	焼き時間入力エラー	%a~%bの整数を入力してくださ	%a:最小値、%b：最大値	P3	
EMSG02	焼き時間入力エラー	%a~%bの%cの倍数を入力してください	%a:最小値、%b：最大値、%c:increment	P3	
EMSG03	予約時間入力エラー	%a~%bの整数を入力してくださ	%a:最小値、%b：最大値	P4	
EMSG04	予約時間入力エラー	%a~%bの整数を入力してくださ	%a:最小値、%b：最大値	P4	
EMSG05	目覚まし時間入力エラー	%a~%bの整数を入力してくださ	%a:最小値、%b：最大値	P4	
EMSG06	目覚まし時間入力エラー	%a~%bの整数を入力してくださ	%a:最小値、%b：最大値	P4	
EMSG07	アラーム入力エラー	%a~%bの整数を入力してくださ	%a:最小値、%b：最大値	P7	
EMSG08	アラーム入力エラー	%a~%bの整数を入力してくださ	%a:最小値、%b：最大値	P7	
EMSG09	現在時刻入力エラー	%a~%bの整数を入力してくださ	%a:最小値、%b：最大値	P8	
EMSG10	現在時刻入力エラー	%a~%bの整数を入力してくださ	%a:最小値、%b：最大値	P8	

付録 H
チェック項目定義書

チェック項目ID	画面ID	項目ID	チェック内容	備考
CH001	P3	3	4-10の整数を入力したか	EMSG01
CH002	P3	3	1-10の整数を入力したか	EMSG01
CH003	P3	4	0-50の10の倍数を入力したか	EMSG03
CH004	P4	2	0-23の整数を入力したか	EMSG04
CH005	P4	3	0-59の整数を入力したか	EMSG05
CH006	P4	6	0-23の整数を入力したか	EMSG06
CH007	P4	7	0-59の整数を入力したか	EMSG07
CH008	P7	2	0-23の整数を入力したか	EMSG08
CH009	P7	3	0-59の整数を入力したか	EMSG09
CH010	P8	2	0-23の整数を入力したか	EMSG10
CH011	P8	3	0-59の整数を入力したか	EMSG11

付録 I

IOC-125 用

CIR コマンド・コマンド転送規則

家電側プログラム用の制約

C1

7	6	5	4	3	2	1	0
0	0	目玉焼き なし0 あり1	トースト なし0 あり1	コーヒー なし0 あり1	普00 淡01 濃10 調理時間11		現在0 予約1

C2

7	6	5	4	3	2	1	0
0	1	目玉焼き0 トースト1	分0 秒1 枚数1	1分～10分 0001～1010(トースト) 4分～10分 0100～1010(目玉焼き) 00秒～50秒 0000~0101 枚数1：1001 枚数2：1010			

C3(24 時間制に修正した)

7	6	5	4	3	2	1	0
1	0	0	0時～23時　00000～10111				

C4

7	6	5	4	3	2	1	0
1	1	00分～59分　000000～111011					

C5

現在時刻設定：10111110

アラーム設定：10111111

C6(コマンド列の最後)

作業開始：11111111

取り消し：00000000

(C6)(C5C3C4(C6))(C1(C2)*n(C3C4)(C5C3C4)C6)　　0<=n<=2

1. 最初に来るコマンドは C1,C5 または C6（00000000）である。
   C5 が来る場合、次に C3,C4 が来る。
2. 次に来るコマンドは C1、または C6(11111111)である。
3. C1 が来る場合、C1 の 5 ビット目が 1 であれば 010 で始まりの C2 が来る。
   C1 の 1，2 ビット目が 11 であれば C2 が 2 回来る
4. C1 が来る場合、C1 の 4 ビット目が 1 であれば 011 で始まりの C2 が来る。
   C1 の 1，2 ビット目が 11 であれば C2 が 2 回来る

5. C1 が来る場合、C1 の 0 ビット目が 1 であれば C3,C4 が来る。C2 も一緒に来るなら、C3,C4 は C2 の後に来る。
6. 次に 10111111（アラーム）があれば C3,C4 が来る。
7. 最後に 11111111（開始）が来る

付録 I

IOC-125 用

CIR コマンド・コマンド転送規則

仕様記述法

仕様は 2 部分に分ける。

1. 引数初期化部分
2. 画面定義部分

例：

BEGIN:Definition

BEGIN:Parameter

・・・引数定義部分・・・

END:Parameter

BEGIN:PageList

・・・画面定義部分・・・

END:PageList

END:Definition

第一部 引数初期化部分

引数リストの開始と終了：

BEGIN:Parameter

引数の定義部分

ENDParameter

引数の定義方法

型　引数名

例：INT eggPiece

型　引数名＝初期値

例：INT eggPiece = 0;

第二部 画面定義部分

画面の開始と終了：

BEGIN:PageList

BEGIN:Page

・・・ページ１の定義・・・

END:Page

BEGIN:Page

・・・ページ 2 の定義・・・

END:Page

…

…

…

END:PageList

画面基本情報の定義

画面の ID、タイトル、音声案内例：

ID:1

TITLE:メインメニュー

SOUND:料理を作るか、時間を設定するかを決めてください

画面上で使用できるパーツの定義

ボタン

コンポーネントの種類	BEGIN:Button
コンポーネント名前	NAME:ボタン名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
ボタン上の文字	TEXT:文字
Disabled の初期値、 イベント	例： DISABLED:true ACTION:A.checked → B.disabled
イベント処理 例：ページ移動	BEGIN:Action DO:clicked->move=ページ数 END：Action

テキストボックス

コンポーネントの種類	BEGIN:TextBox
コンポーネント名前	NAME:テキストボックス名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
タイトル	TITLE：タイトル
コンポーネントの横のサイズも指定	SIZE：横のサイズ
制約： 条件 1 条件 2 条件 3	CONDITION:項目名 COND1： COND2: COND3:
Disabled の初期値、 イベント	例：　DISABLED:true ACTION:A.checked　→　B.disabled

テキストボックスに入力するデータをチェックする必要があります。チェックできる項目は以下のようになります。

項目	意味	条件 1	条件 2	条件 3
Required	必須チェック	－	－	－
Int	Int 型チェック	－	－	－
Double	Double 型チェック	－	－	－
LengthLessThan	入力値の長さチェック　<a	a	－	－
LengthGreaterThan	入力値の長さチェック　>a	a	－	－
LengthLessOrEqual	入力値の長さチェック　<=a	a	－	－
LengthGreaterOrEqual	入力値の長さチェック　>=a	a	－	－
LengthEqual	入力値の長さチェック　==a	a	－	－
lessThan	入力した値の大きさチェック　<a	a	－	
greaterThan	入力した値の大きさチェック　>a	a	－	－
lessOrEqual	入力した値の大きさチェック　<=a	a	－	－
greaterOrEqual	入力した値の大きさチェック　>=a	a	－	－
Equal	入力した値の大きさチェック　==a	a	－	－
lengthBetween	入力値の長さの範囲チェック　a~b	a	b	－
Between	入力した値の範囲チェック　a~b	a	b	－
betweenWithIncrement	入力した値の範囲チェック　a~b,間隔 c	a	b	c

他にも、文字チェック、英数字チェック、URL チェックやメールアドレスチェックなどを

考えたが、家電操作には必要ないもしくは必要とした家電が少ないため、優先度的には先に他の仕事をします。

リストボックス

コンポーネントの種類	BEGIN:ListBox
コンポーネント名前	NAME:リストボックス名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
テキスト	TEXT：表示内容
Disabled の初期値、 イベント	例：　DISABLED:true DO:A.checked　→　B.disabled
初期値	DEFAULTCHOICE：項目番号

チェックボックス

コンポーネントの種類	BEGIN:CheckBox
コンポーネント名前	NAME:チェックボックス名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
テキスト	TEXT：表示内容
Disabled の初期値、 イベント	例：　DISABLED:true DO:A.checked　→　B.disabled
初期値	CHECKED:true

ラジオボタン

コンポーネントの種類	BEGIN:RadioButton
コンポーネント名前	NAME:ラジオボタン名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
選択肢具体設定	BEGIN：Choice ID：項目番号 TEXT：表示内容 ACTION：イベント END：Choice
Disabled の初期値、 イベント	例：　Disabled:true DO:A.checked　→　B.disabled
初期値	DEFAULT（項目番号）

ラベル

コンポーネントの種類	BEGIN:Label
コンポーネント名前	NAME:ラベル名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
ラベル内容	TEXT：ラベル
改行	{EOL}

グループ

コンポーネントの種類	BEGIN:Group（階層設定）
コンポーネント名前	NAME:グループ名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
コンポーネントの大きさ	W：横の長さ　　H：縦の長さ
タイトル	TITLE:タイトル
階層設定 例： A 　B 　　C　　D 　E	例：Group A（型 B；型 E） 　　GroupB（型 C；型 D）

グループに属するコンポーネントの定義はグループの外に書く。

基本構文

コメント識別子

/*　コメント　*/

変数

データ型　変数名　［＝初期値］;

[ ]内は省略可

基本データ型

String 型

d = 文字列

演算子とその優先順位

優先順位	演算子	結合性
高い	. [] () {}	右 → 左
	#	左 → 右
	* / %	左 → 右
	+ -	左 → 右
	< > <= >=	左 → 右
	== !=	左 → 右
	&&	左 → 右
	||	左 → 右
	;	左 → 右
	=	右 → 左
低い	—>	左 → 右

“{}”とは、引数を引用する時に使う演算子になります。

例：

IFNOT:｛cookMinuteMIN｝から 10 までの整数を入力してください。

“#”とは、2 進数に変更する演算子になります。

例：

16#8　＝０００１０００８

“；”とは、優先順位が同じの二つの作業を連結する演算子になります。

例：

Output(11111111;01010101)

“—>”とは判定により結果が決まる演算子になります。

例：
A.checked->B.disabled

制御文一覧

If-else
BEGIN：IF
IF：条件
DO：処理
ELSEIF：条件
DO：処理
ELSE：処理
END：IF

キーワード
データ型定義：BOOLEAN、BYTE、INT、LONG、SHORT、FLOAT、DOUBLE、CHAR
制御文：IF、ELSEIF、ELSE、CASE、FOR、WHILE、BREAK、CONTINUE、RETURN

コンポーネントの種類：Label, Button, TextBox, ListBox, RadioButton, CheckBox、Group
コンポーネント定義：ID、NAME、ACTION、LABEL、EPOSITION（エラーメッセージの位置）、X、Y、W、H、SIZE（テキストボックスの横長さ）、INPUTSIZE（テキストボックスが入力できる最大文字数）、TITLE、TEXT、DEFAULTCHOISE、CHECKED、Choise（リストボックスの選択肢）
タグ：
制約：CONDITION、DEFAULT、TYPE、MAXIMUM、MINIMUM、INCREMENT（間隔）、IFNOT
イベント：Clicked、Inputed、Pressed
非表示：DISABLED
その他：BEGIN、END、EMSG（エラーメッセージ）、EMSGLIST（エラーメッセージのリスト）、Definition、FIRSTPAGE（最初に表示する画面）、Parameter、True、False、Page、OutputCommand、OutputError、{EOL}、ALLDATA、RTD（Return To Default デフォルト値に戻る）、CLEAR（データをクリアする）、QUIT（プログラム終了）
など

付録 J

仕様記述法

仕様記述法

仕様は 2 部分に分ける。

1. 引数初期化部分
2. 画面定義部分

例：

BEGIN:Definition

BEGIN:Parameter

・・・引数定義部分・・・

END:Parameter

BEGIN:PageList

・・・画面定義部分・・・

END:PageList

END:Definition

第一部 引数初期化部分

引数リストの開始と終了：

BEGIN:Parameter

引数の定義部分

ENDParameter

引数の定義方法

型　引数名

例：INT eggPiece

型　引数名＝初期値

例：INT eggPiece = 0;

第二部 画面定義部分

画面の開始と終了：

BEGIN:PageList

BEGIN:Page

・・・ページ１の定義・・・

END:Page

BEGIN:Page

・・・ページ 2 の定義・・・

END:Page

…

…

…

END:PageList

画面基本情報の定義

画面の ID、タイトル、音声案内例：

ID:1

TITLE:メインメニュー

SOUND:料理を作るか、時間を設定するかを決めてください

画面上で使用できるパーツの定義

ボタン

コンポーネントの種類	BEGIN:Button
コンポーネント名前	NAME:ボタン名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
ボタン上の文字	TEXT:文字
Disabled の初期値、 イベント	例： DISABLED:true ACTION:A.checked → B.disabled
イベント処理 例：ページ移動	BEGIN:Action DO:clicked->move=ページ数 END：Action

テキストボックス

コンポーネントの種類	BEGIN:TextBox
コンポーネント名前	NAME:テキストボックス名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
タイトル	TITLE：タイトル
コンポーネントの横のサイズも指定	SIZE：横のサイズ
制約： 条件 1 条件 2 条件 3	CONDITION:項目名 COND1： COND2: COND3:
Disabled の初期値、 イベント	例：　DISABLED:true ACTION:A.checked　→　B.disabled

テキストボックスに入力するデータをチェックする必要があります。チェックできる項目は以下のようになります。

項目	意味	条件 1	条件 2	条件 3
Required	必須チェック	－	－	－
Int	Int 型チェック	－	－	－
Double	Double 型チェック	－	－	－
LengthLessThan	入力値の長さチェック　<a	a	－	－
LengthGreaterThan	入力値の長さチェック　>a	a	－	－
LengthLessOrEqual	入力値の長さチェック　<=a	a	－	－
LengthGreaterOrEqual	入力値の長さチェック　>=a	a	－	－
LengthEqual	入力値の長さチェック　==a	a	－	－
lessThan	入力した値の大きさチェック　<a	a	－	
greaterThan	入力した値の大きさチェック　>a	a	－	－
lessOrEqual	入力した値の大きさチェック　<=a	a	－	－
greaterOrEqual	入力した値の大きさチェック　>=a	a	－	－
Equal	入力した値の大きさチェック　==a	a	－	－
lengthBetween	入力値の長さの範囲チェック　a~b	a	b	－
Between	入力した値の範囲チェック　a~b	a	b	－
betweenWithIncrement	入力した値の範囲チェック a~b,間隔c	a	b	c

他にも、文字チェック、英数字チェック、URL チェックやメールアドレスチェックなどを

考えたが、家電操作には必要ないもしくは必要とした家電が少ないため、優先度的には先に他の仕事をします。

リストボックス

コンポーネントの種類	BEGIN:ListBox
コンポーネント名前	NAME:リストボックス名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
テキスト	TEXT：表示内容
Disabled の初期値、 イベント	例：　DISABLED:true DO:A.checked　→　B.disabled
初期値	DEFAULTCHOICE：項目番号

チェックボックス

コンポーネントの種類	BEGIN:CheckBox
コンポーネント名前	NAME:チェックボックス名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
テキスト	TEXT：表示内容
Disabled の初期値、 イベント	例：　DISABLED:true DO:A.checked　→　B.disabled
初期値	CHECKED:true

ラジオボタン

コンポーネントの種類	BEGIN:RadioButton
コンポーネント名前	NAME:ラジオボタン名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
選択肢具体設定	BEGIN：Choice ID：項目番号 TEXT：表示内容 ACTION：イベント END：Choice
Disabled の初期値、 イベント	例：　Disabled:true DO:A.checked　→　B.disabled
初期値	DEFAULT（項目番号）

ラベル

コンポーネントの種類	BEGIN:Label
コンポーネント名前	NAME:ラベル名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
ラベル内容	TEXT：ラベル
改行	{EOL}

グループ

コンポーネントの種類	BEGIN:Group（階層設定）
コンポーネント名前	NAME:グループ名
音声案内	SOUND:音声内容
コンポーネントの描画を開始する位置	X：X 座標　　Y：Y 座標
コンポーネントの大きさ	W：横の長さ　　H：縦の長さ
タイトル	TITLE:タイトル
階層設定 例： A 　B 　　C　　D 　E	例：Group A（型 B；型 E） 　　GroupB（型 C；型 D）

グループに属するコンポーネントの定義はグループの外に書く。

基本構文

コメント識別子

/*　コメント　*/

変数

データ型　変数名　［＝初期値］;

[ ]内は省略可

基本データ型

String 型

d = 文字列

演算子とその優先順位

優先順位	演算子	結合性
高い	. [] () {}	右 → 左
	#	左 → 右
	* / %	左 → 右
	+ -	左 → 右
	< > <= >=	左 → 右
	== !=	左 → 右
	&&	左 → 右
	||	左 → 右
	;	左 → 右
	=	右 → 左
低い	—>	左 → 右

“{}”とは、引数を引用する時に使う演算子になります。

例：

IFNOT:｛cookMinuteMIN｝から 10 までの整数を入力してください。

“#”とは、2 進数に変更する演算子になります。

例：

16#8　＝０００１０００８

“；”とは、優先順位が同じの二つの作業を連結する演算子になります。

例：

Output(11111111;01010101)

“—>”とは判定により結果が決まる演算子になります。

例：
A.checked->B.disabled

制御文一覧

If-else
BEGIN：IF
IF：条件
DO：処理
ELSEIF：条件
DO：処理
ELSE：処理
END：IF

キーワード
データ型定義：BOOLEAN、BYTE、INT、LONG、SHORT、FLOAT、DOUBLE、CHAR
制御文：IF、ELSEIF、ELSE、CASE、FOR、WHILE、BREAK、CONTINUE、RETURN

コンポーネントの種類：Label, Button, TextBox, ListBox, RadioButton, CheckBox、Group
コンポーネント定義：ID、NAME、ACTION、LABEL、EPOSITION（エラーメッセージの位置）、X、Y、W、H、SIZE（テキストボックスの横長さ）、INPUTSIZE（テキストボックスが入力できる最大文字数）、TITLE、TEXT、DEFAULTCHOISE、CHECKED、Choise（リストボックスの選択肢）
タグ：
制約：CONDITION、DEFAULT、TYPE、MAXIMUM、MINIMUM、INCREMENT（間隔）、IFNOT
イベント：Clicked、Inputed、Pressed
非表示：DISABLED
その他：BEGIN、END、EMSG（エラーメッセージ）、EMSGLIST（エラーメッセージのリスト）、Definition、FIRSTPAGE（最初に表示する画面）、Parameter、True、False、Page、OutputCommand、OutputError、{EOL}、ALLDATA、RTD（Return To Default デフォルト値に戻る）、CLEAR（データをクリアする）、QUIT（プログラム終了）
など

付録 K

IOC-125 用仕様記述 1.1

```
BEGIN:PARAMETER
 move=1
 commandSend=0
 isAlarm=0
 eggPiece=0
 toastPiece=0
 isEgg=0
 isToast=0
 isCoffee=0
 isNow
 color=0
 isCookMinuteSet=0
 isCookSecondSet=0
 isReservationHourSet=0
 isReservationMinuteSet=0
 isAlarmHourSet=0
 isAlarmMinuteSet=0
 isNowHourSet=0
 isNowMinuteSet=0
 cookMinuteMIN
 cookMinute=0
 cookSecond=0
 reservationHour=0
 reservationMinute=0
 alarmHour=0
 alarmMinute=0
 nowHour=0
 nowMinute=0
END:PARAMETER
BEGIN:PAGE
 ID:1
 TITLE:メインメニュー
 SOUND:メインメニュー
 BEGIN:BUTTON
  ID:make
  NAME:料理を作る
```

```
  XPOSITION:10
  YPOSITION:50
  SOUND:料理を作る
  ACTION:move=2
 END:BUTTON
 BEGIN:BUTTON
  ID:setTime
  NAME:時間設定
  XPOSITION:10
  YPOSITION:90
  SOUND:時間を設定する
  ACTION:move=6
 END:BUTTON
 BEGIN:BUTTON
  ID:cancel
  NAME:現作業中止
  XPOSITION:10
  YPOSITION:130
  SOUND:現作業を中止する
  ACTION:output=00000000;move=9
 END:BUTTON
END:PAGE
BEGIN:PAGE
 ID:2
 TITLE:調理メニュー
 SOUND:調理メニュー
 BEGIN:LABEL
  NAME:卵焼き
  XPOSITION:10
  YPOSITION:50
  SOUND:卵焼き
 END:LABEL
 BEGIN:LISTBOX
  SIZE:70
  XPOSITION:140
  YPOSITION:50
```

```
 ITEM:なし,eggPiece=0;isEgg=0
 ITEM:1 個,eggPiece=1;isEgg=1;cookMinuteMIN=4
 ITEM:2 個,eggPiece=2;isEgg=1;cookMinuteMIN=4
END:LISTBOX
BEGIN:LABEL
 NAME:トースト
 XPOSITION:10
 YPOSITION:90
 SOUND:トースト
END:LABEL
BEGIN:LISTBOX
 SIZE:70
 XPOSITION:140
 YPOSITION:90
 ITEM:なし,toastPiece=0;isToast=0
 ITEM:1 枚,toastPiece=1;isToast=1;cookMinuteMIN=1
 ITEM:2 枚,toastPiece=2;isToast=1;cookMinuteMIN=1
END:LISTBOX
BEGIN:LABEL
 NAME:コーヒー
 XPOSITION:10
 YPOSITION:130
 SOUND:コーヒー
END:LABEL
BEGIN:LISTBOX
 SIZE:70
 XPOSITION:140
 YPOSITION:130
 ITEM:なし,isCoffee=0
 ITEM:あり,isCoffee=1
END:LISTBOX
BEGIN:LABEL
 NAME:調理時間
 XPOSITION:10
 YPOSITION:170
 SOUND:調理時間
```

```
 END:LABEL
 BEGIN:LISTBOX
  SIZE:100
  XPOSITION:140
  YPOSITION:170
  ITEM:今すぐ,isNow=0
  ITEM:予約,isNow=1
 END:LISTBOX
 BEGIN:BUTTON
  ID:next
  NAME:次へ
  XPOSITION:80
  YPOSITION:220
  SOUND:次へ
  ACTION:move=3
 END:BUTTON
 BEGIN:BUTTON
  ID:return
  NAME:戻る
  XPOSITION:80
  YPOSITION:260
  SOUND:戻る
  ACTION:move=1
 END:BUTTON
END:PAGE
BEGIN:PAGE
 ID:3
 TITLE:焼き加減設定
 SOUND:焼き加減設定
 BEGIN:LABEL
  NAME:焼き加減
  XPOSITION:10
  YPOSITION:50
  SOUND:焼き加減
 END:LABEL
 BEGIN:LISTBOX
```

SIZE:70
XPOSITION:140
YPOSITION:50
ITEM:普通,color=0
ITEM:淡,color=1
ITEM:濃,color=2
ITEM:時間,color=3
END:LISTBOX
BEGIN:LABEL
NAME:焼き時間
XPOSITION:10
YPOSITION:100
SOUND:焼き時間
END:LABEL
BEGIN:LISTBOX
SIZE:70
XPOSITION:60
YPOSITION:140
ITEM:1 分,cookMinute=1;isCookMinuteSet=1
ITEM:2 分,cookMinute=2;isCookMinuteSet=1
ITEM:3 分,cookMinute=3;isCookMinuteSet=1
ITEM:4 分,cookMinute=4;isCookMinuteSet=1
ITEM:5 分,cookMinute=5;isCookMinuteSet=1
ITEM:6 分,cookMinute=6;isCookMinuteSet=1
ITEM:7 分,cookMinute=7;isCookMinuteSet=1
ITEM:8 分,cookMinute=8;isCookMinuteSet=1
ITEM:9 分,cookMinute=9;isCookMinuteSet=1
ITEM:10 分,cookMinute=10;isCookMinuteSet=1
END:LISTBOX
BEGIN:LISTBOX
SIZE:70
XPOSITION:140
YPOSITION:140
ITEM:0 秒,cookSecond=0;isCookSecondSet=1
ITEM:10 秒,cookSecond=10;isCookSecondSet=1
ITEM:20 秒,cookSecond=20;isCookSecondSet=1

ITEM:30 秒,cookSecond=30;isCookSecondSet=1
ITEM:40 秒,cookSecond=40;isCookSecondSet=1
ITEM:50 秒,cookSecond=50;isCookSecondSet=1
END:LISTBOX
BEGIN:BUTTON
ID:next
NAME:次へ
XPOSITION:80
YPOSITION:190
SOUND:次へ
ACTION:move=4
END:BUTTON
BEGIN:BUTTON
ID:return
NAME:戻る
XPOSITION:80
YPOSITION:230
SOUND:戻る
ACTION:move=1
END:BUTTON
END:PAGE
BEGIN:PAGE
ID:4
TITLE:予約目覚し設定
SOUND:予約目覚まし時間を設定する
BEGIN:LABEL
NAME:予約時間
XPOSITION:10
YPOSITION:60
SOUND:予約時間
END:LABEL
BEGIN:TEXTBOX
SIZE:40
XPOSITION:50
YPOSITION:100
ACTION:reservationHour;isReservationHourSet=1

END:TEXTBOX
BEGIN:LABEL
 NAME:時
 XPOSITION:90
 YPOSITION:100
 SOUND:時
END:LABEL
BEGIN:TEXTBOX
 SIZE:40
 XPOSITION:140
 YPOSITION:100
 ACTION:reservationMinute;isReservationMinuteSet=1
END:TEXTBOX
BEGIN:LABEL
 NAME:分
 XPOSITION:180
 YPOSITION:100
 SOUND:分
END:LABEL
BEGIN:LABEL
 NAME:目覚し
 XPOSITION:10
 YPOSITION:150
 SOUND:目覚まし時間
END:LABEL
BEGIN:LISTBOX
 SIZE:90
 XPOSITION:110
 YPOSITION:150
 ITEM:する,isAlarm=0
 ITEM:しない,isAlarm=1
END:LISTBOX
BEGIN:TEXTBOX
 SIZE:40
 XPOSITION:50
 YPOSITION:190

```
 ACTION:alarmHour;isAlarmHourSet=1
END:TEXTBOX
BEGIN:LABEL
 NAME:時
 XPOSITION:90
 YPOSITION:190
 SOUND:時
END:LABEL
BEGIN:TEXTBOX
 SIZE:40
 XPOSITION:140
 YPOSITION:190
 ACTION:alarmMinute;isAlarmMinuteSet=1
END:TEXTBOX
BEGIN:LABEL
 NAME:分
 XPOSITION:180
 YPOSITION:190
 SOUND:分
END:LABEL
BEGIN:BUTTON
 ID:next
 NAME:次へ
 XPOSITION:80
 YPOSITION:250
 SOUND:次へ
 ACTION:move=5
END:BUTTON
BEGIN:BUTTON
 ID:return
 NAME:戻る
 XPOSITION:80
 YPOSITION:290
 SOUND:戻る
 ACTION:move=1
END:BUTTON
```

```
END:PAGE
BEGIN:PAGE
 ID:5
 TITLE:設定確認
 SOUND:設定確認画面
 BEGIN:LABEL
  NAME:卵焼き
  XPOSITION:5
  YPOSITION:60
  SOUND:卵焼き
 END:LABEL
 BEGIN:LABEL
  NAME:{eggPiece}#個
  XPOSITION:170
  YPOSITION:60
 END:LABEL
 BEGIN:LABEL
  NAME:トースト
  XPOSITION:5
  YPOSITION:100
  SOUND:トースト
 END:LABEL
 BEGIN:LABEL
  NAME:{toastPiece}#枚
  XPOSITION:170
  YPOSITION:100
 END:LABEL
 BEGIN:LABEL
  NAME:コーヒー
  XPOSITION:5
  YPOSITION:140
  SOUND:コーヒー
 END:LABEL
 BEGIN:LABEL
  NAME:{isCoffee}
  XPOSITION:170
```

YPOSITION:140
END:LABEL
BEGIN:LABEL
NAME:焼き加減
XPOSITION:5
YPOSITION:180
SOUND:焼き加減
END:LABEL
BEGIN:LABEL
NAME:{color}
XPOSITION:170
YPOSITION:180
END:LABEL
BEGIN:LABEL
NAME:焼き時間
XPOSITION:5
YPOSITION:220
SOUND:焼き時間
END:LABEL
BEGIN:LABEL
NAME:{cookMinute}#分#{cookSecond}#秒
XPOSITION:130
YPOSITION:220
END:LABEL
BEGIN:LABEL
NAME:予約時間
XPOSITION:5
YPOSITION:260
SOUND:予約時間
END:LABEL
BEGIN:LABEL
NAME:{reservationHour}#時#{reservationMinute}#分
XPOSITION:130
YPOSITION:260
END:LABEL
BEGIN:LABEL

```
    NAME:目覚時間
    XPOSITION:5
    YPOSITION:300
    SOUND:目覚まし時間
  END:LABEL
  BEGIN:LABEL
    NAME:{alarmHour}#時#{alarmMinute}#分
    XPOSITION:130
    YPOSITION:300
  END:LABEL
  BEGIN:BUTTON
    ID:finish
    NAME:完了
    XPOSITION:80
    YPOSITION:340
    SOUND:完了
    ACTION:move=9
  END:BUTTON
  BEGIN:BUTTON
    ID:modify
    NAME:修正
    XPOSITION:80
    YPOSITION:380
    SOUND:修正
    ACTION:move=2
  END:BUTTON
  BEGIN:BUTTON
    ID:cancel
    NAME:キャンセル
    XPOSITION:30
    YPOSITION:420
    SOUND:キャンセル
    ACTION:move=1
  END:BUTTON
END:PAGE
BEGIN:PAGE
```

```
 ID:6
 TITLE:時間設定
 SOUND:時間設定画面
 BEGIN:BUTTON
  ID:setAlarm
  NAME:目覚時間設定
  XPOSITION:10
  YPOSITION:50
  SOUND:目覚まし時間を設定する
  ACTION:move=7
 END:BUTTON
 BEGIN:BUTTON
  ID:setNowTime
  NAME:現在時刻設定
  XPOSITION:10
  YPOSITION:90
  SOUND:現在時刻を設定する
  ACTION:move=8
 END:BUTTON
 BEGIN:BUTTON
  ID:return
  NAME:戻る
  XPOSITION:80
  YPOSITION:130
  SOUND:戻る
  ACTION:move=1
 END:BUTTON
END:PAGE
BEGIN:PAGE
 ID:7
 TITLE:目覚時間設定
 SOUND:目覚まし時間設定画面
 BEGIN:LABEL
  NAME:目覚時間
  XPOSITION:10
  YPOSITION:60
```

```
 SOUND:目覚まし時間
END:LABEL
BEGIN:TEXTBOX
 SIZE:40
 XPOSITION:50
 YPOSITION:100
 ACTION:alarmHour;isAlarmHourSet=1
END:TEXTBOX
BEGIN:LABEL
 NAME:時
 XPOSITION:90
 YPOSITION:100
 SOUND:時
END:LABEL
BEGIN:TEXTBOX
 SIZE:40
 XPOSITION:140
 YPOSITION:100
 ACTION:alarmMinute;isAlarmMinuteSet=1
END:TEXTBOX
BEGIN:LABEL
 NAME:分
 XPOSITION:180
 YPOSITION:100
 SOUND:分
END:LABEL
BEGIN:BUTTON
 ID:alarmSet
 NAME:設定する
 XPOSITION:50
 YPOSITION:150
 SOUND:設定する
 ACTION:move=9
END:BUTTON
BEGIN:BUTTON
 ID:return
```

```
    NAME:戻る
    XPOSITION:80
    YPOSITION:190
    SOUND:戻る
    ACTION:move=6
  END:BUTTON
END:PAGE
BEGIN:PAGE
  ID:8
  TITLE:現在時刻設定
  SOUND:現在時刻設定画面
  BEGIN:LABEL
    NAME:現在時刻
    XPOSITION:10
    YPOSITION:60
    SOUND:現在時刻
  END:LABEL
  BEGIN:TEXTBOX
    SIZE:40
    XPOSITION:50
    YPOSITION:100
    ACTION:nowHour;isNowHourSet=1
  END:TEXTBOX
  BEGIN:LABEL
    NAME:時
    XPOSITION:90
    YPOSITION:100
    SOUND:時
  END:LABEL
  BEGIN:TEXTBOX
    SIZE:40
    XPOSITION:140
    YPOSITION:100
    ACTION:nowMinute;isNowMinuteSet=1
  END:TEXTBOX
  BEGIN:LABEL
```

```
    NAME:分
    XPOSITION:180
    YPOSITION:100
    SOUND:分
  END:LABEL
  BEGIN:BUTTON
    ID:nowSet
    NAME:設定する
    XPOSITION:50
    YPOSITION:150
    SOUND:設定する
    ACTION:move=9
  END:BUTTON
  BEGIN:BUTTON
    ID:return
    NAME:戻る
    XPOSITION:80
    YPOSITION:190
    SOUND:戻る
    ACTION:move=6
  END:BUTTON
END:PAGE
BEGIN:PAGE
  ID:9
  TITLE:作業内容終了
  SOUND:メイン画面に戻るか、プログラムを終了するかを決める
  BEGIN:IF
  IF:isEgg==1||isToast==1||isCoffee==1
  DO:output=00#{isEgg:1}#{isToast:1}#{isCoffee:1}#{color:2}#{isNow:1};commandSend=1
  END:IF
  BEGIN:IF
  IF:isEgg==1&&isCookMinuteSet==0
  DO:output=010110#{eggPiece:2};commandSend=1
  END:IF
  BEGIN:IF
  IF:isEgg==1&&isCookMinuteSet==1
```

DO:output=0100#{cookMinute:4};output=0101#{cookSecond / 10:4};commandSend=1
END:IF
BEGIN:IF
IF:isToast==1&&isCookMinuteSet==0
DO:output=011110#{toastPiece:2};commandSend=1
END:IF
BEGIN:IF
IF:isToast==1&&isCookMinuteSet==1
DO:output=0110#{toastMinute:4};output=0111#{toastSecond / 10:4};commandSend=1
END:IF
BEGIN:IF
IF:isNow==1

DO:output=100#{reservationHour:5};output=11#{reservationMinute:6};commandSend=
1
END:IF
BEGIN:IF
IF:isAlarmHourSet==1

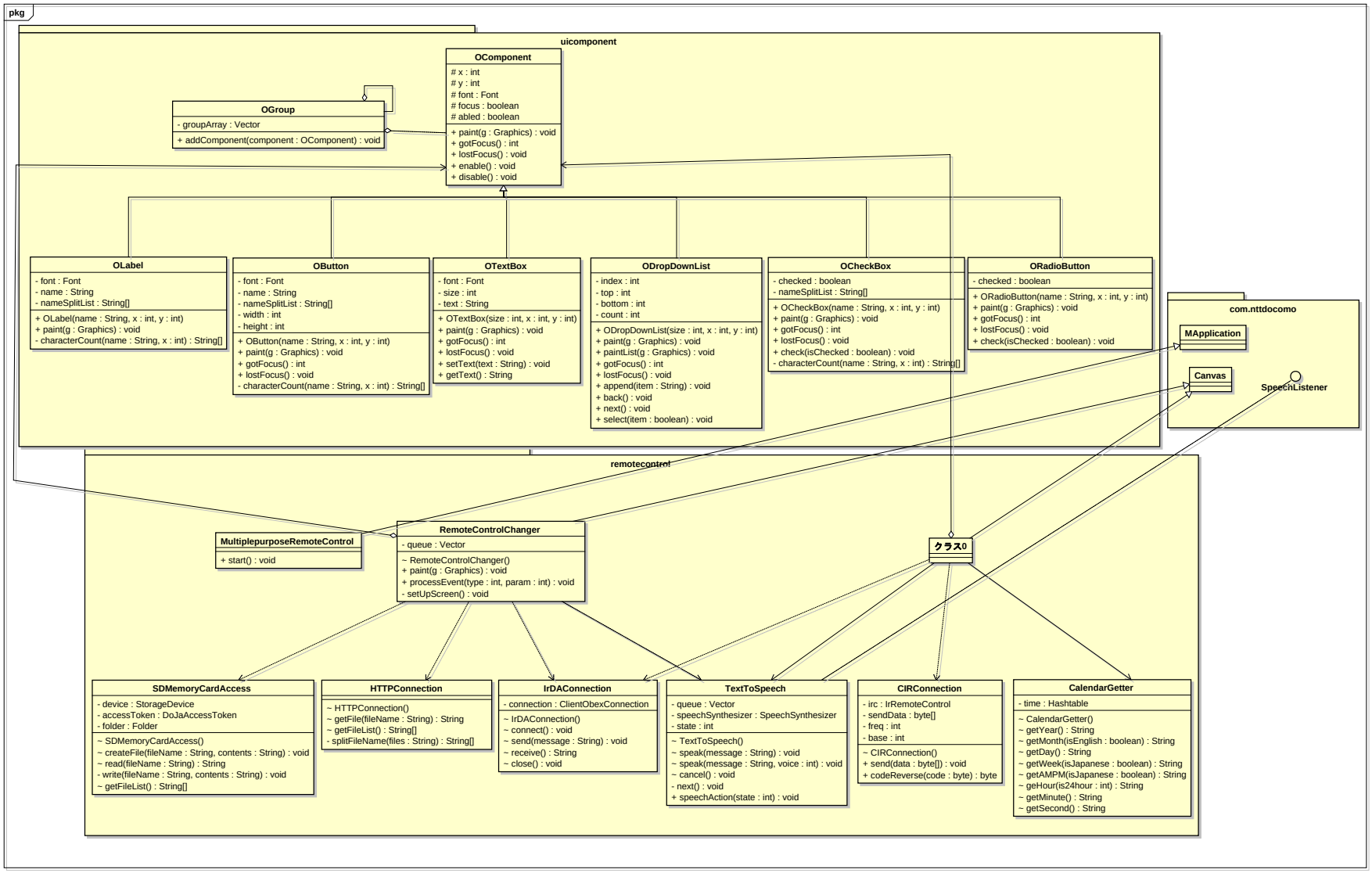
DO:output=10111111;output=100#{alarmHour:5};output=11#{alarmMinute:6};command
Send=1
END:IF
BEGIN:IF
IF:isNowHourSet==1

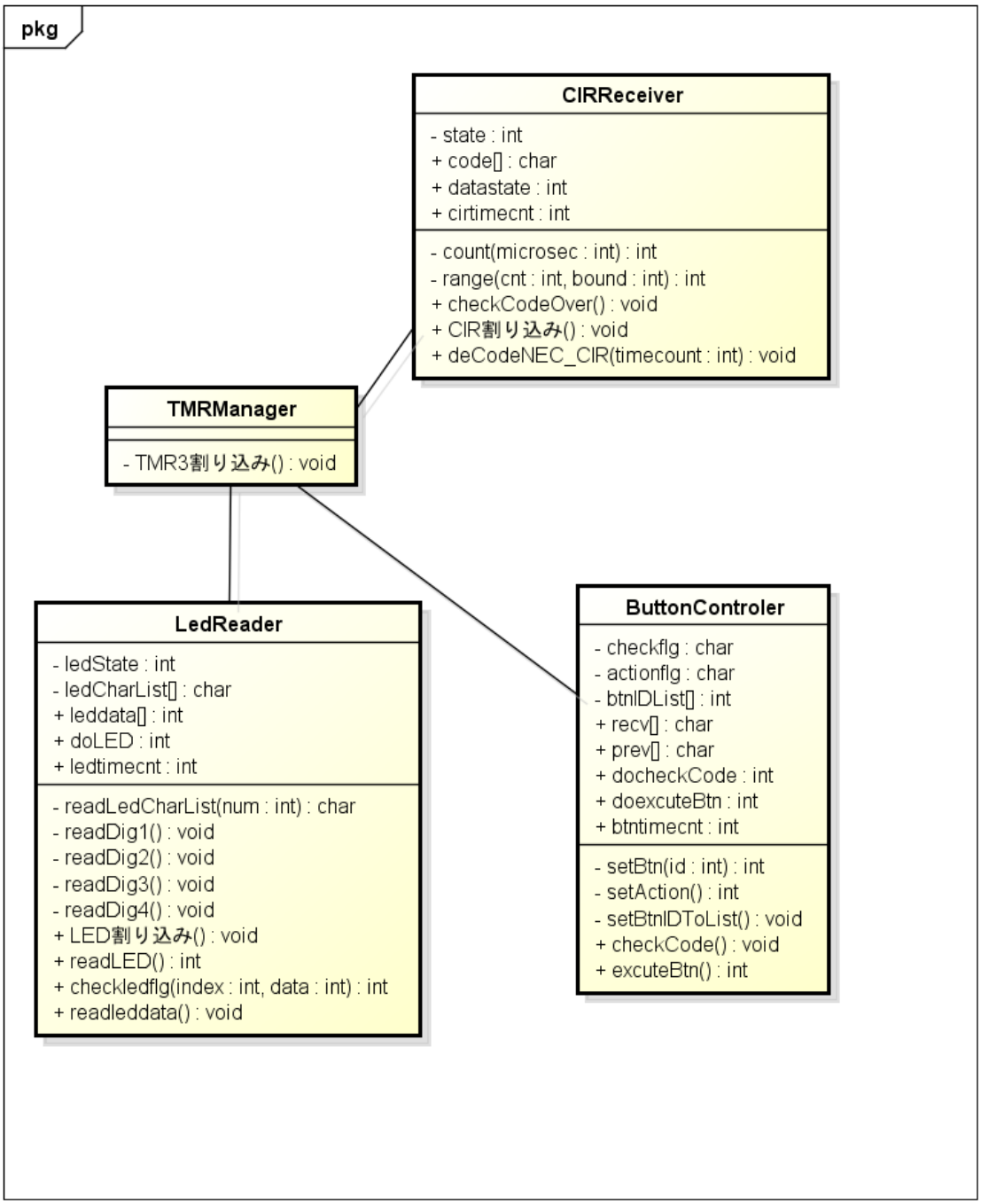
DO:output=10111110;output=100#{nowHour:5};output=11#{nowMinute:6};commandSe
nd=1
END:IF
BEGIN:IF
IF:commandSend==1
DO:output=11111111
END:IF
BEGIN:BUTTON
ID:return
NAME:メインに戻る
XPOSITION:10

```
    YPOSITION:50
    SOUND:メイン画面に戻る
    ACTION:move=1
  END:BUTTON
END:PAGE
```

付録 L

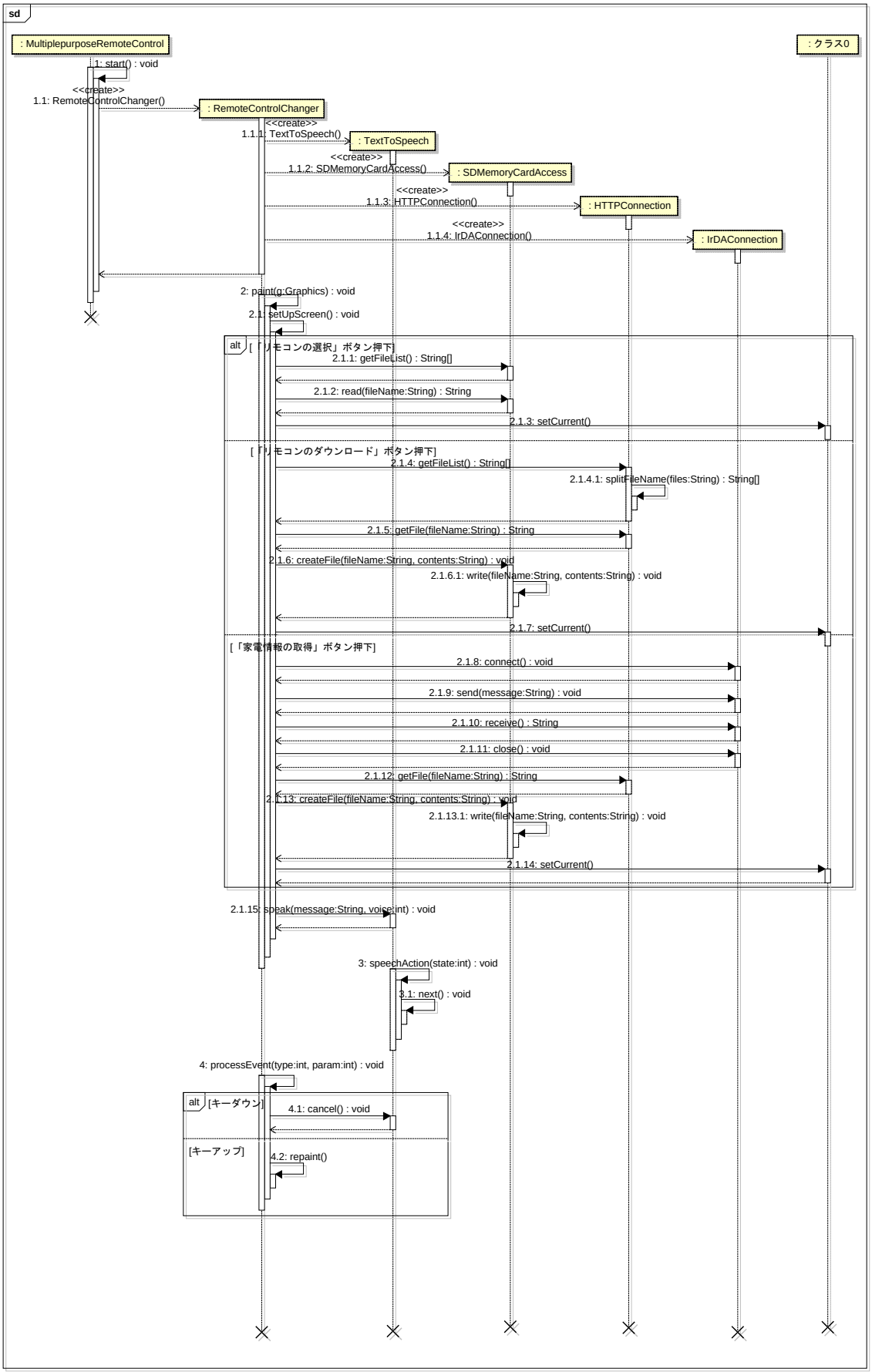
クラス図

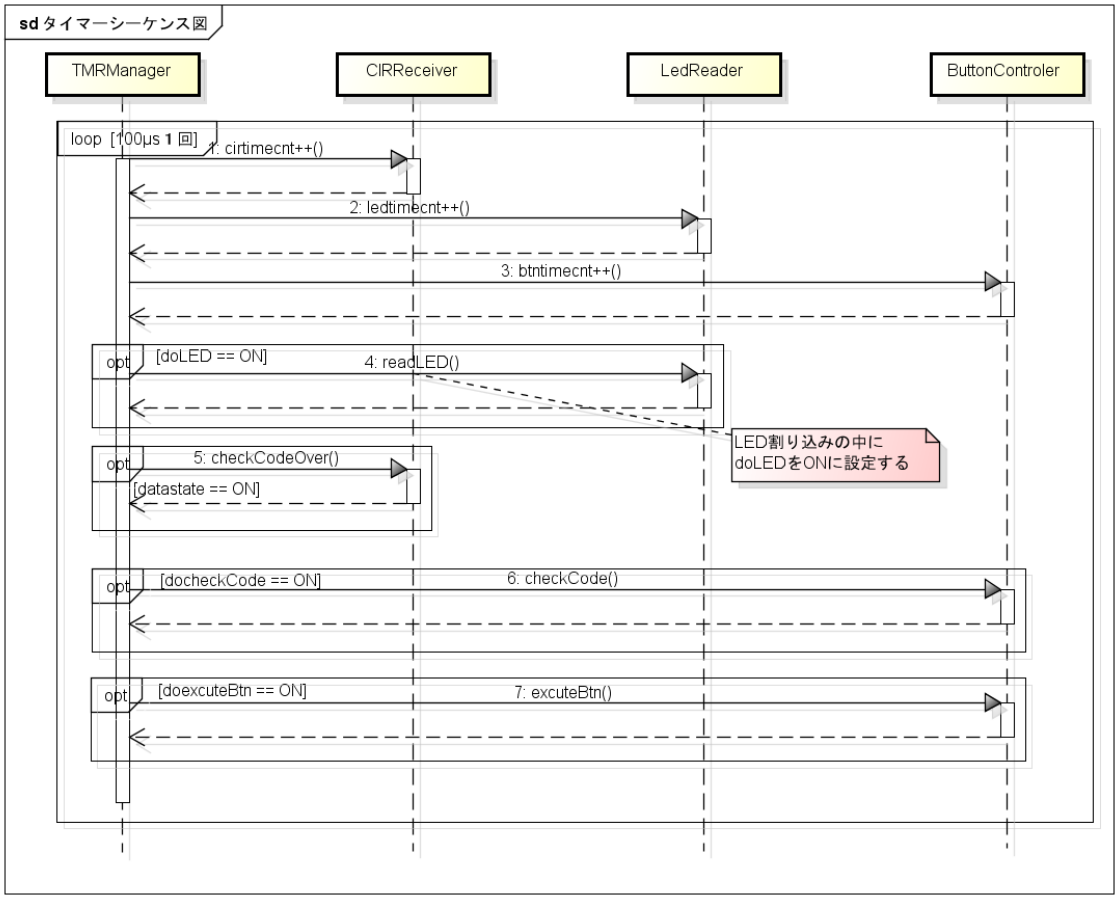

pkg
uicomponent
OComponent
x : int
y : int
font : Font
focus : boolean
abled : boolean
+ paint(g : Graphics) : void
+ gotFocus() : int
+ lostFocus() : void
+ enable() : void
+ disable() : void
OGroup
- groupArray : Vector
+ addComponent(component : OComponent) : void
OLabel
- font : Font
- name : String
- nameSplitList : String[]
+ OLabel(name : String, x : int, y : int)
+ paint(g : Graphics) : void
- characterCount(name : String, x : int) : String[]
OButton
- font : Font
- name : String
- nameSplitList : String[]
- width : int
- height : int
+ OButton(name : String, x : int, y : int)
+ paint(g : Graphics) : void
+ gotFocus() : int
+ lostFocus() : void
- characterCount(name : String, x : int) : String[]
OTextBox
- font : Font
- size : int
- text : String
+ OTextBox(size : int, x : int, y : int)
+ paint(g : Graphics) : void
+ gotFocus() : int
+ lostFocus() : void
+ setText(text : String) : void
+ getText() : String
ODropDownList
- index : int
- top : int
- bottom : int
- count : int
+ ODropDownList(size : int, x : int, y : int)
+ paint(g : Graphics) : void
+ paintList(g : Graphics) : void
+ gotFocus() : int
+ lostFocus() : void
+ append(item : String) : void
+ back() : void
+ next() : void
+ select(item : boolean) : void
OCheckBox
- checked : boolean
- nameSplitList : String[]
+ OCheckBox(name : String, x : int, y : int)
+ paint(g : Graphics) : void
+ gotFocus() : int
+ lostFocus() : void
+ check(isChecked : boolean) : void
- characterCount(name : String, x : int) : String[]
ORadioButton
- checked : boolean
+ ORadioButton(name : String, x : int, y : int)
+ paint(g : Graphics) : void
+ gotFocus() : int
+ lostFocus() : void
+ check(isChecked : boolean) : void
com.nttdocomo
MApplication
Canvas
SpeechListener
remotecontrol
MultiplepurposeRemoteControl
+ start() : void
RemoteControlChanger
- queue : Vector
~ RemoteControlChanger()
+ paint(g : Graphics) : void
+ processEvent(type : int, param : int) : void
- setUpScreen() : void
クラス0
SDMemoryCardAccess
- device : StorageDevice
- accessToken : DoJaAccessToken
- folder : Folder
~ SDMemoryCardAccess()
~ createFile(fileName : String, contents : String) : void
~ read(fileName : String) : String
- write(fileName : String, contents : String) : void
~ getFileList() : String[]
HTTPConnection
~ HTTPConnection()
~ getFile(fileName : String) : String
~ getFileList() : String[]
- splitFileName(files : String) : String[]
IrDAConnection
- connection : ClientObexConnection
~ IrDAConnection()
~ connect() : void
~ send(message : String) : void
~ receive() : String
~ close() : void
TextToSpeech
- queue : Vector
- speechSynthesizer : SpeechSynthesizer
- state : int
~ TextToSpeech()
~ speak(message : String) : void
~ speak(message : String, voice : int) : void
~ cancel() : void
- next() : void
+ speechAction(state : int) : void
CIRConnection
- irc : IrRemoteControl
- sendData : byte[]
- freq : int
- base : int
~ CIRConnection()
+ send(data : byte[]) : void
+ codeReverse(code : byte) : byte
CalendarGetter
- time : Hashtable
~ CalendarGetter()
~ getYear() : String
~ getMonth(isEnglish : boolean) : String
~ getDay() : String
~ getWeek(isJapanese : boolean) : String
~ getAMPM(isJapanese : boolean) : String
~ geHour(is24hour : int) : String
~ getMinute() : String
~ getSecond() : String

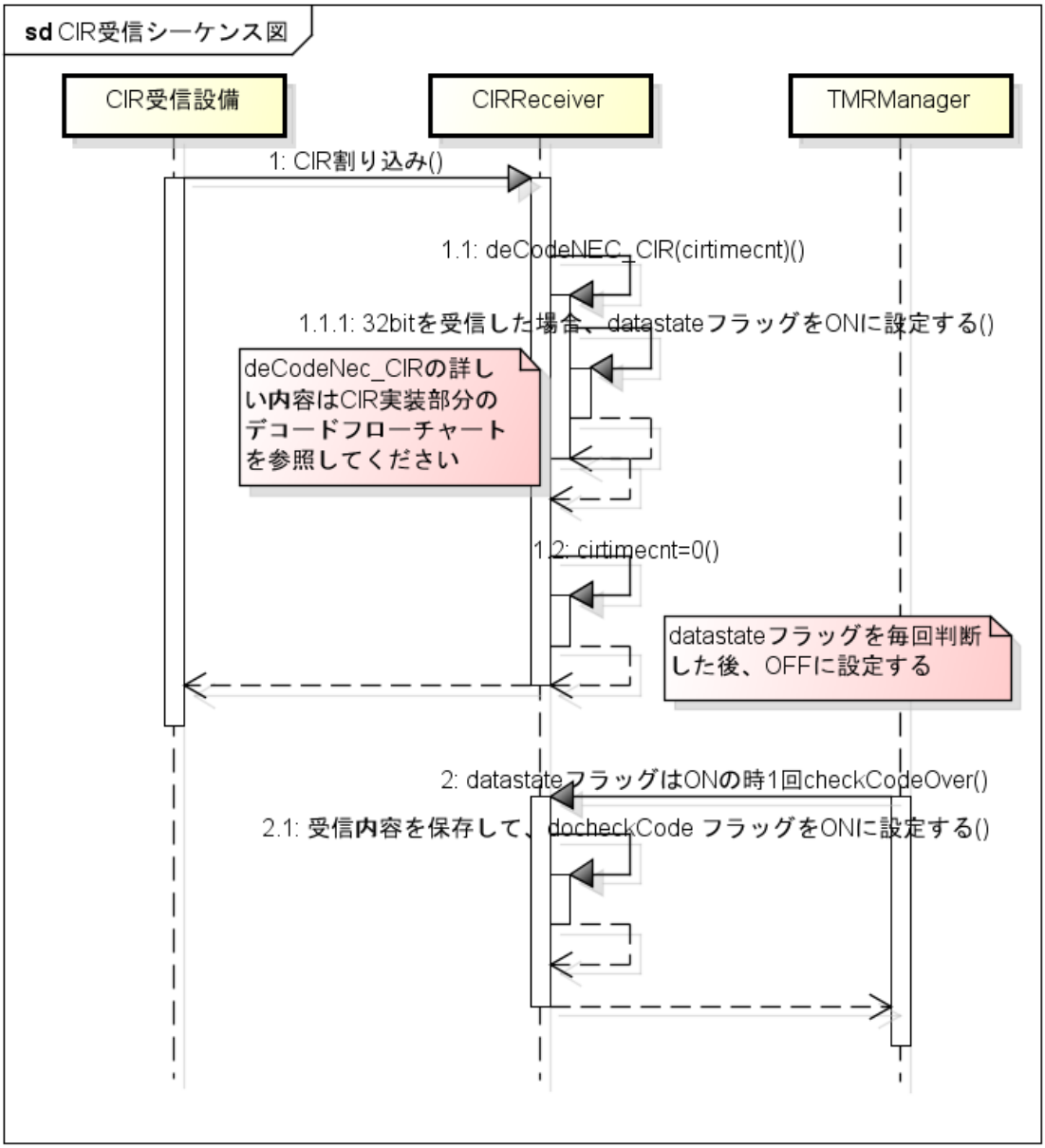

pkg
CIRReceiver
- state : int
+ code[] : char
+ datastate : int
+ cirtimecnt : int
- count(microsec : int) : int
- range(cnt : int, bound : int) : int
+ checkCodeOver() : void
+ CIR割り込み() : void
+ deCodeNEC_CIR(timecount : int) : void
TMRManager
- TMR3割り込み() : void
LedReader
- ledState : int
- ledCharList[] : char
+ leddata[] : int
+ doLED : int
+ ledtimecnt : int
- readLedCharList(num : int) : char
- readDig1() : void
- readDig2() : void
- readDig3() : void
- readDig4() : void
+ LED割り込み() : void
+ readLED() : int
+ checkledflg(index : int, data : int) : int
+ readleddata() : void
ButtonControler
- checkflg : char
- actionflg : char
- btnIDList[] : int
+ recv[] : char
+ prev[] : char
+ docheckCode : int
+ doexcuteBtn : int
+ btntimecnt : int
- setBtn(id : int) : int
- setAction() : int
- setBtnIDToList() : void
+ checkCode() : void
+ excuteBtn() : int

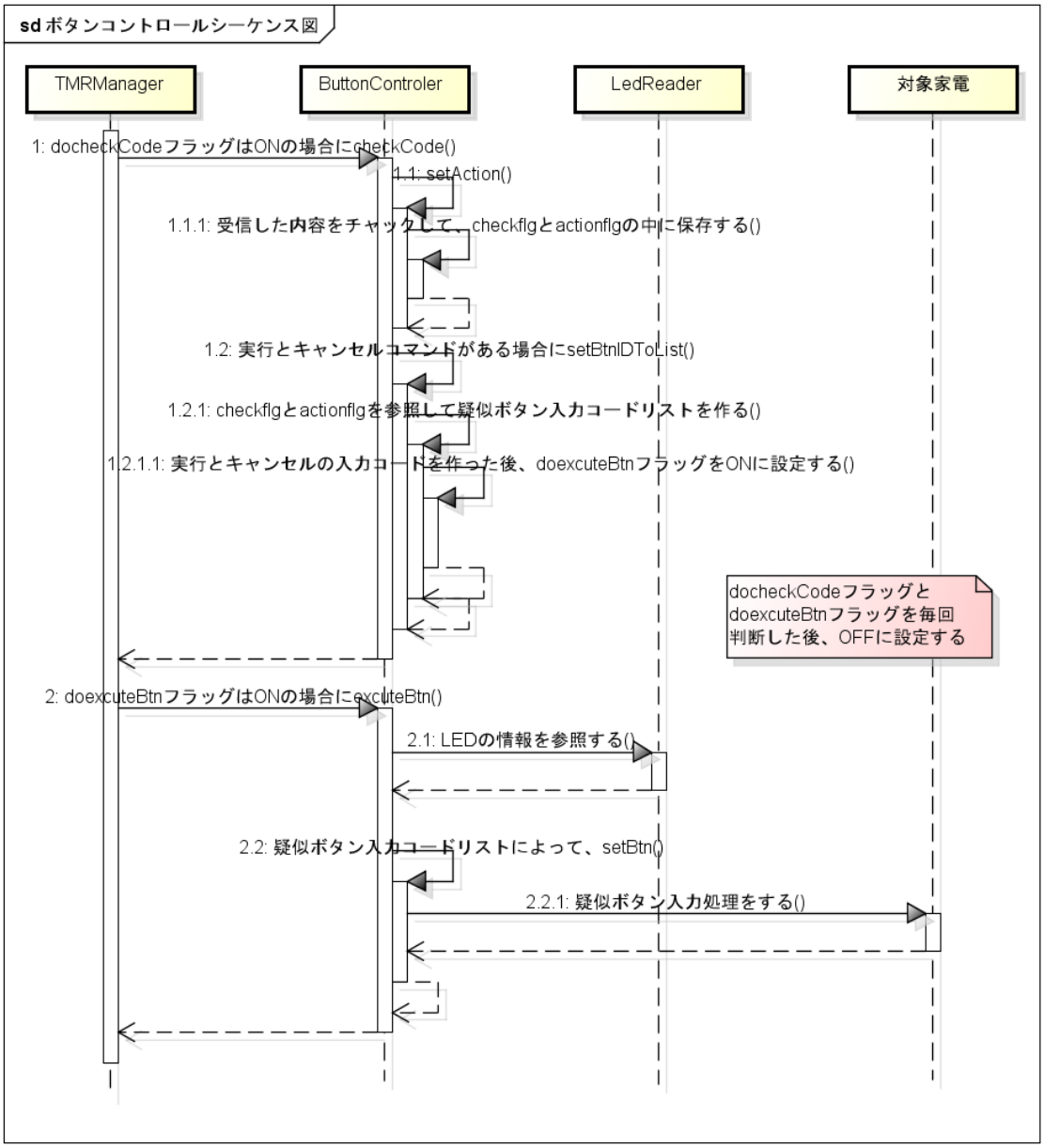
付録 M

シーケンス図


sd
: MultiplepurposeRemoteControl
: クラス0
1: start() : void
<<create>>
1.1: RemoteControlChanger()
: RemoteControlChanger
<<create>>
1.1.1: TextToSpeech()
: TextToSpeech
<<create>>
1.1.2: SDMemoryCardAccess()
: SDMemoryCardAccess
<<create>>
1.1.3: HTTPConnection()
: HTTPConnection
<<create>>
1.1.4: IrDAConnection()
: IrDAConnection
2: paint(g:Graphics) : void
2.1: setUpScreen() : void
alt
[「リモコンの選択」ボタン押下]
2.1.1: getFileList() : String[]
2.1.2: read(fileName:String) : String
2.1.3: setCurrent()
[「リモコンのダウンロード」ボタン押下]
2.1.4: getFileList() : String[]
2.1.4.1: splitFileName(files:String) : String[]
2.1.5: getFile(fileName:String) : String
2.1.6: createFile(fileName:String, contents:String) : void
2.1.6.1: write(fileName:String, contents:String) : void
2.1.7: setCurrent()
[「家電情報の取得」ボタン押下]
2.1.8: connect() : void
2.1.9: send(message:String) : void
2.1.10: receive() : String
2.1.11: close() : void
2.1.12: getFile(fileName:String) : String
2.1.13: createFile(fileName:String, contents:String) : void
2.1.13.1: write(fileName:String, contents:String) : void
2.1.14: setCurrent()
2.1.15: speak(message:String, voice:int) : void
3: speechAction(state:int) : void
3.1: next() : void
4: processEvent(type:int, param:int) : void
alt
[キーダウン]
4.1: cancel() : void
[キーアップ]
4.2: repaint()


sd タイマーシーケンス図
TMRManager
CIRReceiver
LedReader
ButtonControler
loop [100μs 1 回]
1: cirtimecnt++()
2: ledtimecnt++()
3: btntimecnt++()
opt
[doLED == ON]
4: readLED()
LED割り込みの中に
doLEDをONに設定する
opt
5: checkCodeOver()
[datastate == ON]
opt
[docheckCode == ON]
6: checkCode()
opt
[doexcuteBtn == ON]
7: excuteBtn()


sd CIR受信シーケンス図
CIR受信設備
CIRReceiver
TMRManager
1: CIR割り込み()
1.1: deCodeNEC_CIR(cirtimecnt)()
1.1.1: 32bitを受信した場合、datastateフラッグをONに設定する()
deCodeNec_CIRの詳しい内容はCIR実装部分のデコードフローチャートを参照してください
1.2: cirtimecnt=0()
datastateフラッグを毎回判断した後、OFFに設定する
2: datastateフラッグはONの時1回checkCodeOver()
2.1: 受信内容を保存して、docheckCode フラッグをONに設定する()


sd ボタンコントロールシーケンス図
TMRManager
ButtonControler
LedReader
対象家電
1: docheckCodeフラッグはONの場合にcheckCode()
1.1: setAction()
1.1.1: 受信した内容をチャックして、checkflgとactionflgの中に保存する()
1.2: 実行とキャンセルコマンドがある場合にsetBtnIDToList()
1.2.1: checkflgとactionflgを参照して疑似ボタン入力コードリストを作る()
1.2.1.1: 実行とキャンセルの入力コードを作った後、doexcuteBtnフラッグをONに設定する()
docheckCodeフラッグと
doexcuteBtnフラッグを毎回
判断した後、OFFに設定する
2: doexcuteBtnフラッグはONの場合にexcuteBtn()
2.1: LEDの情報を参照する()
2.2: 疑似ボタン入力コードリストによって、setBtn()
2.2.1: 疑似ボタン入力処理をする()

付録 N
プログラム設計書

パッケージ	クラス
uicomponent	OComponent
	OGroup
	OLabel
	OButton
	OTextBox
	ODropDownListBox
	OCheckBox
	ORadioButton
remotecontrol	MultiplepurposeRemoteContro
	RemoteControlChanger
	SDMemoryCardAccess
	HTTPConnection
	CIRConnection
	IrDAConnection
	TextToSpeech
	CalendarGetter

パッケージ名　uicomponent
クラス名　Ocomponent

メソッド名	paint
引数	Graphics g
返り値	void
機能	UIコンポーネントを描画する
処理内容	

メソッド名	gotFocus
引数	void
返り値	int
機能	UIコンポーネントにフォーカスを当てる
処理内容	1. -1を返す

メソッド名	lostFocus
引数	void
返り値	void
機能	UIコンポーネントからフォーカスを外す
処理内容	

メソッド名	enable
引数	void
返り値	void
機能	UIコンポーネントを使用可能にする

処理内容	1. コンポーネントを使用可能にする

メソッド名	disable
引数	void
返り値	void
機能	UIコンポーネントを使用不可能にする
処理内容	1. コンポーネントを使用不可能にする

パッケージ名　uicomponent
クラス名　OLabel

メソッド名	paint
引数	Graphics g
返り値	void
機能	描画する
処理内容	1. ラベルが使用可能であるかチェック 1.1. 使用可能な場合 1.1.1. 描画する 1.2. 使用不可能な場合 1.2.1. グレーアウトして描画する

パッケージ名　uicomponent
クラス名　OButton

メソッド名	paint
引数	Graphics g
返り値	void
機能	描画する
処理内容	1. ボタンが使用可能であるかチェック 1.1. 使用可能な場合 1.1.1. 描画する 1.2. 使用不可能な場合 1.2.1. グレーアウトして描画する 2. フォーカスが当てられているかチェック 2.1. 当てられている場合 2.1.1. フォーカスの描画を行う

メソッド名	gotFocus
引数	void
返り値	int
機能	フォーカスを当てる
処理内容	1. ボタンが使用可能であるかチェック 1.1. 使用可能な場合 1.1.1. フォーカスを当てる 1.1. 2 1を返す 1.2. 使用不可能な場合 1.2.1. -1を返す

メソッド名	lostFocus
引数	void
返り値	void
機能	フォーカスを外す
処理内容	1. ボタンが使用可能であるかチェック 1.1. 使用可能な場合 1.1.1. フォーカスを外す

パッケージ名　uicomponent
クラス名　OTextBox

メソッド名	paint
引数	Graphics g
返り値	void
機能	描画する
処理内容	1. テキストボックスが使用可能であるかチェック 1.1. 使用可能な場合 1.1.1. 描画する 1.2. 使用不可能な場合 1.2.1. グレーアウトして描画する 2. フォーカスが当てられているかチェック 2.1. 当てられいている場合 2.1.1. フォーカスを描画する 3. 入力があるかチェック 3.1. 入力がある場合 3.1.1. テキストボックスに文字を描画する

メソッド名	gotFocus
引数	void
返り値	int
機能	フォーカスを当てる
処理内容	1. テキストボックスが使用可能であるかチェック 1.1. 使用可能である場合 1.1.1. フォーカスを当てる 1.1.2. 1を返す 1.2. 使用不可能である場合 1.2.1. -1を返す

メソッド名	lostFocus
引数	void
返り値	void
機能	フォーカスを外す
処理内容	1. テキストボックスが使用可能であるかチェック 1.1. 使用可能である場合 1.1.1. フォーカスを外す

メソッド名	setText
引数	String text
返り値	void
機能	テキストボックスに文字をセットする
処理内容	1. 受け取った引数(文字列)をセットする

メソッド名	getText
引数	void
返り値	String text
機能	テキストボックスに入力された文字を取得する
処理内容	1. 入力された文字列を返す

パッケージ名　uicomponent
クラス名　ODropDownList

メソッド名	paint
引数	Graphics g
返り値	void
機能	描画する
処理内容	1. ドロップダウンリストが使用可能であるかチェック 1.1. 使用可能な場合 1.1.1. 描画する 1.2. 使用不可能な場合 1.2.1. グレーアウトして描画する 2. フォーカスが当てられているかチェック 2.1. 当てれている場合 2.1.1. フォーカスを描画する 3. ドロップダウンリストの項目が選択されているかチェック 3.1. 選択されている場合 3.1.1. 選択されている項目を描画

メソッド名	paintList
引数	Graphics g
返り値	void
機能	ドロップダウンリストの項目を描画する
処理内容	1. 指定されたページ内の項目を描画する 2. 選択されている項目があるかチェック 2.1. 選択されている項目がある場合 2.1.1. 項目の背景色と文字の色を変える 2.1.2. 指定されている項目にフォーカスを当てる

メソッド名	gotFocus
引数	void
返り値	int
機能	フォーカスを当てる
処理内容	1. ドロップダウンリストが使用可能であるかチェック 1.1. 使用可能な場合

	1.1.1. フォーカスを取得する 1.1.2. 1を返す 1.2. 使用不可能な場合 1.2.1. -1を返す

メソッド名	lostFocus
引数	void
返り値	void
機能	フォーカスを外す
処理内容	1. ドロップダウンリストが使用可能であるかチェック 1.1. 使用可能である場合 1.1.1. フォーカスを外す

メソッド名	append
引数	String item
返り値	void
機能	ドロップダウンリストに項目を追加する
処理内容	1. リストに項目を追加する

メソッド名	select
引数	String item
返り値	void
機能	項目を選択する/選択を解除する
処理内容	1. フォーカスが当てられているかチェック 1.1. 当てらている場合 1.1.1. itemをドロップダウンリストで選択された項目として登録する

メソッド名	back
引数	void

返り値	void
機能	項目を表示するページを前のページにする
処理内容	1. 現在指定されている項目の前の項目が0以上であるかチェック 1.1. 0以上の場合 1.1.1. 前のページに移動する 1.1.2. インデックスを前のページの末尾に移動させる

メソッド名	next
引数	void
返り値	void
機能	項目を表示するページを次のページにする
処理内容	1. 現在指定されている項目の次の項目がリストの最大サイズ以内であるかチェック 1.1. 最大サイズ以内の場合 1.1.1. 次のページに移動する 1.1.2. インデックスを次のページの先頭に移動させる

パッケージ名　uicomponent
クラス名　OCheckBox

メソッド名	paint
引数	Graphics g
返り値	void
機能	描画する
処理内容	1. チェックボックスが使用可能であるかチェック 1.1. 使用可能な場合 1.1.1. 描画する 1.2. 使用不可能な場合 1.2.1. グレーアウトして描画する 2. フォーカスが当てられているかチェック 2.1. 当てられいている場合 2.1.1. フォーカスを描画する 3. チェックマークが付いているかチェック 3.1. 付いている場合 3.1.1. チェックマークを描画する

メソッド名	gotFocus
引数	void
返り値	int
機能	フォーカスを当てる
処理内容	1. チェックボックスが使用可能であるかチェック 1.1. 使用可能である場合 1.1.1. フォーカスを当てる 1.1.2. 1を返す 1.2. 使用不可能である場合 1.2.1. -1を返す

メソッド名	lostFocus
引数	void
返り値	void
機能	フォーカスを外す
処理内容	1. チェックボックスが使用可能であるかチェック 1.1. 使用可能である場合 1.1.1. フォーカスを外す

メソッド名	check
引数	boolean checked
返り値	void
機能	チェックマークを付ける/外す
処理内容	1. フォーカスが当てられているかチェック 1.1. 当てられている場合 1.1.1. 引数として受け取った値をチェックマーク有無フラグに代入する

パッケージ名　uicomponent
クラス名　ORadioButtonBox

メソッド名	paint
引数	Graphics g
返り値	void
機能	描画する
処理内容	1. ラジオボタンが使用可能であるかチェック 1.1. 使用可能な場合 1.1.1. 描画する 1.2. 使用不可能な場合 1.2.1. グレーアウトして描画する 2. フォーカスが当てられているかチェック 2.1. 当てられいている場合 2.1.1. フォーカスを描画する 3. チェックマークが付いているかチェック 3.1. 付いている場合 3.1.1. チェックマークを描画する

メソッド名	gotFocus
引数	void
返り値	int
機能	フォーカスを当てる
処理内容	1. ラジオボタンが使用可能であるかチェック 1.1. 使用可能である場合 1.1.1. フォーカスを当てる 1.1.2. 1を返す 1.2. 使用不可能である場合 1.2.1. -1を返す

メソッド名	lostFocus
引数	void
返り値	void
機能	フォーカスを外す
処理内容	1. ラジオボタンが使用可能であるかチェック 1.1. 使用可能である場合 1.1.1. フォーカスを外す

メソッド名	check
引数	boolean checked
返り値	void
機能	チェックマークを付ける/外す
処理内容	1. フォーカスが当てられているかチェック 1.1. 当てられている場合 1.1.1. 引数として受け取った値をチェックマーク有無フラグに代入する

パッケージ名　　remotecontrol
クラス名　　MultiplepurposeRemoteControl

メソッド名	start
引数	void
返り値	void
機能	iアプリの起動
処理内容	1. RemoteControlChangerクラスのインスタンス化 2. RemoteControlChangerオブジェクトを画面に設定する

パッケージ名　　remotecontrol
クラス名　　RemoteControlChanger

メソッド名	paint
引数	Graphics g
返り値	void
機能	画面を描画する
処理内容	1. setUpScreen()メソッドを呼び出す 2. リスト表示かどうかチェック 2.1. リスト表示の場合 2.1.1. リストの項目を描画する 2.2. リスト表示でない場合 2.2.1. コンポーネントを描画する

メソッド名	processEvent
引数	int type, int param
返り値	void
機能	イベント処理を行う
処理内容	1. イベントの種類をチェック 1.1. キーアップイベントの場合 1.1.1.　イベントのパラメーターをチェック 1.1.1.1. 上方向キーの場合 1.1.1.1.1. インデックスを1減らす 1.1.1.2　下方向キーの場合 1.1.1.2.1. インデックスを1増やす 1.1.1.3. 左方向キーの場合 1.1.1.3.1. ページを1減らす 1.1.1.4. 右方向キーの場合 1.1.1.4.1. ページを1増やす 1.1.1.5. 決定キーの場合 1.1.1.5.1. リスト表示かチェック 1.1.1.5.1.1. リスト表示の場合 1.1.1.5.1.1.1. selectedとリストの選択されている項目番号をキューに格納 1.1.1.5.1.2. リスト表示でない場合 1.1.1.5.1.2. pressedと選択されているコンポーネントの番号をキューに格納

メソッド名	setUpScreen
引数	void
返り値	void
機能	画面の設定を行う
処理内容	

パッケージ名　　remotecontrol
クラス名　　SDMemoryCardAccess

メソッド名	createFile
引数	String Filename, String contents
返り値	void
機能	ファイルの新規作成を行う
処理内容	1. ファイルの作成 2. write()メソッドを呼び出す

メソッド名	read
引数	String filename
返り値	void
機能	ファイルの読み込み
処理内容	1. ファイルのインスタンスを取得 2. ファイルを読み込み専用で開く 3. ファイルを読み込むためのデータ入力インスタンスを取得 4. ファイルからデータを読み込む 5. データ入力を閉じる 6. ファイルを閉じる

メソッド名	write
引数	String Filename, String contents
返り値	void
機能	ファイルの書き込み
処理内容	1. ファイルのインスタンスを取得 2. ファイルを書き込み専用で開く 3. ファイルを書き込むためのデータ出力インスタンスを取得 4. ファイルにデータを書き込む 5. データ出力をフラッシュする 6. データ出力を閉じる 7. ファイルを閉じる

メソッド名	getFileList
引数	void
返り値	String[] list
機能	ファイル一覧の取得
処理内容	1. ファイル一覧を取得 2. String配列にファイルのパスを格納

パッケージ名　　remotecontrol
クラス名　　HTTPConnection

メソッド名	getFile
引数	String fileName
返り値	String
機能	ファイルの読み込み
処理内容	1. HTTP接続オブジェクトの取得 2. リクエストメソッドの設定 3. 接続する 4. HTTP_OKをレスポンスとして受け取ったかチェック 4.1. 受け取った場合 4.1.1. 入力ストリームを取得 4.1.2. 入力ストリームリーダーを取得 4.1.3. データを取得 4.1.4. 入力ストリームリーダーを閉じる 4.1.5. 入力ストリームを閉じる 5. 接続を閉じる 6. データを返す

メソッド名	getFileList
引数	void
返り値	void
機能	ファイル一覧の作成
処理内容	1. HTTP接続オブジェクトの取得 2. リクエストメソッドの設定 3. 接続する 4. HTTP_OKをレスポンスとして受け取ったかチェック 4.1. 受け取った場合 4.1.1. 入力ストリームを取得 4.1.2. 入力ストリームリーダーを取得 4.1.3. データを取得 4.1.4. 入力ストリームリーダーを閉じる 4.1.5. 入力ストリームを閉じる 5. 接続を閉じる 6. splitFileName()メソッドを呼び出す 6. ファイル一覧を返す

メソッド名	splitFileName
引数	String data
返り値	String[]
機能	ファイル名ごとに分割してリストに格納
処理内容	1. 受け取った引数をファイル名で分割してVectorに格納 2. Vectorから要素を取り出し、String配列に格納 3. ファイル一覧を返す

パッケージ名　　remotecontrol
クラス名　　IrDAConnection

メソッド名	connect
引数	void
返り値	void
機能	赤外線通信のコネクションを張る
処理内容	1. 赤外線通信用のクライアントとして、接続をオープンする 2. 接続を確立する

メソッド名	send
引数	String message
返り値	void
機能	メッセージを送信する
処理内容	1. PUTオペレーションコードを設定する 2. ネームヘッダの設定 3. データ出力ストリームを開く 4. データを書き込む 5. データ出力ストリームを閉じる 6. リクエストを送信する 7. レスポンスを受け取る

メソッド名	receive
引数	void
返り値	String
機能	メッセージを受信する
処理内容	1. GETオペレーションコードを設定すr 2. ネームヘッダの設定 3. リクエストを送信する 4. レスポンスを受け取る 5. データ入力ストリームを開く 6. データを読み込む 7. データ入力ストリームを閉じる

メソッド名	close
引数	void
返り値	void
機能	赤外線通信のコネクションを閉じる
処理内容	1. コネクションを閉じる

パッケージ名　　remotecontrol
クラス名　　TextToSpeech

メソッド名	speak
引数	String message
返り値	void
機能	メッセージを読み上げる
処理内容	1. デフォルトヴォイスを設定して、speak（String, int）メソッドを呼び出す

メソッド名	speak
引数	String message, int voice
返り値	void
機能	メッセージを読み上げる
処理内容	1. メッセージと指定した声を読み上げキューに格納 2. next()メソッドを呼び出す

メソッド名	cancel
引数	void
返り値	void
機能	読み上げを止める
処理内容	1. 読み上げキューを空にする 2. 読み上げを停止する

メソッド名	next
引数	void
返り値	void
機能	次のメッセージを読み上げる
処理内容	1. 読み上げ状態変数が0以上かチェック 1.1. 0以上の場合 1.1.1. 読み上げ状態変数を-1にする 1.1.2. キューの最初の要素を取り出す 1.1.3. キューの0番目の要素を空にする 1.1.4. 読み上げ属性の設定 1.1.5. メッセージを読み上げる

メソッド名	speechAction
引数	int state
返り値	void
機能	読み上げ機能のイベントを処理する
処理内容	1. 引数stateのチェック 1.1. stateが読み上げ停止の場合 1.1.1. キューを空にする 2. stateを引数stateで上書きする 3. next()メソッドを呼び出す

パッケージ名　　remotecontrol
クラス名　　CIRConnection

メソッド名	send
引数	byte[] data
返り値	void
機能	データを送信する
処理内容	1. IrRemoteControlFrameインスタンスの配列を生成 2. codeReverse()メソッドを呼び出す 3. データコードをビット反転する 4. IrRemoteControlFrameインスタンスを生成し、配列に格納 5. 送信するデータとbit長を設定 6. リーダコードのhigh時間を設定 7. リーダコードのlow時間を設定 8. ストップビットのhigh時間を設定 9. フレーム時間を設定 10. リピート回数を設定 11. データを送信する

メソッド名	codeReverse
引数	byte code
返り値	byte
機能	8bitのコードをMSB→LSBをLSB→MSBに書き変える
処理内容	1. MSB→LSBをLSB→MSBに入れ替える 2. 入れ替えた値を戻す

パッケージ名　remotecontrol
クラス名　CalendarGetter

メソッド名	getYear
引数	void
返り値	String
機能	携帯電話で設定されている現在の西暦を取得する
処理内容	1. 現在の西暦を取得 2. 西暦を文字列型に変換して返す

メソッド名	getMonth
引数	boolean isEnglish
返り値	String
機能	携帯電話で設定されている現在の月を取得する
処理内容	1. 現在の月を取得 2.1. isEnglishをチェック 2.1.1. isEnglishがtrueの場合 2.1.1.1. 月を英語表記に変換する 3. 月を文字列型に変換して返す

メソッド名	getDay
引数	void
返り値	String
機能	携帯電話で設定されている現在の日付を取得する
処理内容	1. 現在の日付を取得 2. 日付を文字列型に変換して返す

メソッド名	getWeek
引数	boolean isJapanese
返り値	String
機能	携帯電話で設定されている現在の曜日を取得する
処理内容	1. 現在の曜日を取得 2.1. isJapaneseをチェック 2.1.1. isJapaneseがtrueの場合 2.1.1.1. 曜日を日本語表記に変更 2.1.2. isJapaneseがfalseの場合 2.1.2.1. 曜日を英語表記に変更 3. 曜日を返す

メソッド名	getAMPM
引数	boolean isJapanese
返り値	String
機能	携帯電話で設定されている現在の時刻のAM/PMを取得する
処理内容	1. 現在の時刻のAM/PMを取得 2.1. isJapaneseをチェック 2.1.1. isJapaneseがtrueの場合 2.1.1.1. AM/PMを日本語表記に変更 2.1.2. isJapaneseがfalseの場合 2.1.2.1. AM/PMを英語表記に変更

	3. AM/PMを返す

メソッド名	getHour
引数	boolean is24hour
返り値	String
機能	携帯電話で設定されている現在の時を取得する
処理内容	1. 現在の時を取得 2.1. is24hourをチェック 2.1.1. is24hourがtrueの場合 2.1.1.1. 現在の時を24時間表記に変換 2.1.2. is24hourがfalseの場合 2.1.2.1. 現在の時を12時間表記に変換 3. 現在の時を返す

メソッド名	getMinute
引数	void
返り値	String
機能	携帯電話で設定されている現在の分を取得する
処理内容	1. 現在の分を取得 2. 現在の分を文字列型で返す

メソッド名	getSecond
引数	void
返り値	String
機能	携帯電話で設定されている現在の秒を取得する
処理内容	1. 現在の秒を取得 2. 現在の秒を文字列型で返す

付録 O

単体テスト項目表

uicomponentパッケージ

OLabelクラス

ID	テスト項目	結果
1	abled	○
2	disabled	○
3	縦範囲外に描画　描画されない	○
4	横範囲外に描画　描画されない	○

OButtonクラス

ID	テスト項目	結果
1	abled	○
2	disabled	○
3	focus	○
4	フォーカスを外す	○
5	縦範囲外に描画　描画されない	○
6	横範囲外に描画　描画されない	○

OTextBoxクラス

ID	テスト項目	結果
1	abled	○
2	disabled	○
3	focus	○
4	フォーカスを外す	○
5	縦範囲外に描画　描画されない	○
6	横範囲外に描画　描画されない	○
7	テキストのセット	○
8	テキストの取得	○

ODropDownListクラス

ID	テスト項目	結果
1	abled	○
2	disabled	○
3	focus	○
4	フォーカスを外す	○
5	選択状態にする/しない	○
6	選択状態の取得	○
7	縦範囲外に描画　描画されない	○
8	横範囲外に描画　描画されない	○
9	項目の描画	○
10	フォーカスの描画	○
11	フォーカスを外す	○
12	選択状態の描画	○
13	選択状態を解除	○
14	項目ががない場合	×
15	項目名が長い場合	○
16	前のページへ移動	○
17	次のページへ移動	○

OCheckBoxクラス

ID	テスト項目	結果
1	abled	○
2	disabled	○
3	focus	○
4	checked	○
5	フォーカスを外す	○
6	縦範囲外に描画　描画されない	○
7	横範囲外に描画　描画されない	○

ORadioButtonクラス

ID	テスト項目	結果
1	abled	○
2	disabled	○
3	focus	○
4	checked	○
5	フォーカスを外す	○
6	縦範囲外に描画　描画されない	○
7	横範囲外に描画　描画されない	○

remotecontrolパッケージ

CIRConnectionクラス

ID	テスト項目	結果
1	コマンドの送信	○
2	コマンドの反転	○

IrDAConnectionクラス

ID	テスト項目	結果
1	接続	○
2	切断	○
3	文字列送信	○
4	文字列受信	○

TextToSpeechクラス

ID	テスト項目	結果
1	発声開始	○
2	発声停止	○
3	声変更	○

HTTPConnectionクラス

ID	テスト項目	結果
1	data.txtがある場合	○
2	ない場合	○
3	指定したファイルがある場合	○
4	ない場合	○
5	ファイルの最後に改行コードがある場合	○
6	ない場合	○
7	改行コードが複数ある場合	○

SDMemoryCardAccessクラス

ID	テスト項目	結果
1	ファイル一覧表示	○
2	ファイルあり	○
3	ファイルなし	○
4	ファイル読み込み（ファイルあり）	○
5	ファイル読み込み（ファイルなし）	○
6	ファイル新規作	○
7	ファイル新規作成（既にファイルがある）	○
8	ファイル書き込み（ファイルあり）	×
9	ファイル書き込み（ファイルなし）	○

項番	前提条件	動作	期待結果	実測結果
1	入力は「BEGIN:PAGE」	キーと値を正しく取得した	キーは「BEGIN」、値は	
2	入力は BEGIN:PARAMETER move=1 END:PARAMETER	変数は配列に保存した	キーは「move」、値は「1」として変数保存用配列に保存する	
3	入力は BEGIN:PARAMETER isNow END:PARAMETER	変数は配列に保存した	キーは「isNow」、値は""として変数保存用配列に保存する	
4	入力は BEGIN:PAGE 　TITLE:メインメニュー …… END:PAGE	画面のタイトルを取得した	ページのタイトルを「メインメニュー」と設定した	
5	入力は BEGIN:PAGE 　SOUND:メインメニュー …… END:PAGE	画面の音声内容を取得した	ページの音声読み上げ内容を「メインメニュー」と設定した	
6	入力は BEGIN:BUTTON 　ID:make …… END:BUTTON	ボタンのIDを取得した	ボタンのIDを「make」と設定した	
7	入力は BEGIN:BUTTON 　NAME:料理を作る …… END:BUTTON	ボタンのテキストを取得した	ボタンのテキストを「料理を作る」と設定した	
8	入力は BEGIN:BUTTON 　XPOSITION:10 …… END:BUTTON	ボタンのX座標を取得した	ボタンのX座標を「１０」と設定した	
9	入力は BEGIN:BUTTON 　YPOSITION:50 …… END:BUTTON	ボタンのY座標を取得した	ボタンのY座標を「50」と設定した	
10	入力は BEGIN:BUTTON 　SOUND:料理を作る …… END:BUTTON	ボタンの音声内容を取得した	ボタンの音声内容を「料理を作る」と設定した	
11	入力は BEGIN:BUTTON 　ACTION:move=2 …… END:BUTTON	ボタンのアクション内容を取得した	ボタンのアクション内容を「move=2」と設定した	
12	入力は BEGIN:BUTTON 　ENABLED:false …… END:BUTTON	ボタンの初期状態を取得した	ボタンの初期状態を「false」と設定した	

13	入力は BEGIN:LABEL 　NAME:卵焼き …… END:LABEL	ラベルのテキストを取得した	ラベルのテキストを「卵焼き」と設定した	
14	入力は BEGIN:LABEL 　XPOSITION:10 …… END:LABEL	ラベルのX座標を取得した	ラベルのX座標を「１０」と設定した	
15	入力は BEGIN:LABEL 　YPOSITION:50 …… END:LABEL	ラベルのY座標を取得した	ラベルのY座標を「50」と設定した	
16	入力は BEGIN:LABEL 　SOUND:卵焼き …… END:LABEL	ラベルの音声内容を取得した	ラベルの音声内容を「卵焼き」と設定した	
17	入力は BEGIN:LISTBOX 　SIZE:70 …… END:LISTBOX	リストボックスのサイズを取得した	リストボックスのサイズを「70」と設定した	
18	入力は BEGIN:LISTBOX 　XPOSITION:140 …… END:LISTBOX	リストボックスのX座標を取得した	リストボックスのX座標を「140」と設定した	
19	入力は BEGIN:LISTBOX 　YPOSITION:50 …… END:LISTBOX	リストボックスのY座標を取得した	リストボックスのY座標を「50」と設定した	
20	入力は BEGIN:LISTBOX ITEM:1個,eggPiece=1;isEgg=1;cookMinuteMIN=4 …… END:LISTBOX	リストボックスの選択リストを取得した	リストボックスの選択リストを「1個」と設定し、アクションは「eggPiece=1;isEgg=1;cookMinuteMIN=4」と設定した	
21	入力は BEGIN:LISTBOX 　ENABLED:false …… END:LISTBOX	リストボックスの初期状態を取得した	リストボックスの初期状態を「false」と設定した	
22	入力は BEGIN:TEXTBOX 　SIZE:40 …… END:TEXTBOX	テキストボックスのサイズを取得した	テキストボックスのサイズを「40」と設定した	
23	入力は BEGIN:TEXTBOX 　XPOSITION:140 …… END:TEXTBOX	テキストボックスのX座標を取得した	テキストボックスのX座標を「140」と設定した	

24	入力は BEGIN:TEXTBOX 　YPOSITION:100 …… END:TEXTBOX	テキストボックスのY座標を取得した	テキストボックスのY座標を「100」と設定した	
25	入力は BEGIN:TEXTBOX ACTION:alarmMinute;isAlarmMinuteSet=1 …… END:TEXTBOX	テキストボックスの選択リストを取得した	テキストボックスのアクションは「alarmMinute;isAlarmMinuteSet=1」と設定した	
26	入力は BEGIN:TEXTBOX 　ENABLED:false …… END:TEXTBOX	テキストボックスの初期状態を取得した	テキストボックスの初期状態は「false」と設定した	
27	入力は BEGIN:CHECKBOX 　ID:make …… END:CHECKBOX	チェックボックスのIDを取得した	チェックボックスのIDを「make」と設定した	
28	入力は BEGIN:CHECKBOX 　NAME:料理を作る …… END:CHECKBOX	チェックボックスのテキストを取得した	チェックボックスのテキストを「料理を作る」と設定した	
29	入力は BEGIN:CHECKBOX 　XPOSITION:10 …… END:CHECKBOX	チェックボックスのX座標を取得した	チェックボックスのX座標を「１０」と設定した	
30	入力は BEGIN:CHECKBOX 　YPOSITION:50 …… END:CHECKBOX	チェックボックスのY座標を取得した	チェックボックスのY座標を「50」と設定した	
31	入力は BEGIN:CHECKBOX 　SOUND:料理を作る …… END:CHECKBOX	チェックボックスの音声内容を取得した	チェックボックスの音声内容を「料理を作る」と設定した	
32	入力は BEGIN:CHECKBOX 　ACTION:move=2 …… END:CHECKBOX	チェックボックスのアクション内容を取得した	チェックボックスのアクション内容を「move=2」と設定した	
33	入力は BEGIN:CHECKBOX 　ENABLED:false …… END:CHECKBOX	チェックボックスの初期状態を取得した	チェックボックスの初期状態を「false」と設定した	
34	入力は BEGIN:RADIOBUTTON 　ID:make …… END:RADIOBUTTON	ラジオボタンのIDを取得した	ラジオボタンのIDを「make」と設定した	

35	入力は BEGIN:RADIOBUTTON 　NAME:料理を作る …… END:RADIOBUTTON	ラジオボタンのテキストを取得した	ラジオボタンのテキストを「料理を作る」と設定した	
36	入力は BEGIN:RADIOBUTTON 　XPOSITION:10 …… END:RADIOBUTTON	ラジオボタンのX座標を取得した	ラジオボタンのX座標を「１０」と設定した	
37	入力は BEGIN:RADIOBUTTON 　YPOSITION:50 …… END:RADIOBUTTON	ラジオボタンのY座標を取得した	ラジオボタンのY座標を「50」と設定した	
38	入力は BEGIN:RADIOBUTTON 　SOUND:料理を作る …… END:RADIOBUTTON	ラジオボタンの音声内容を取得した	ラジオボタンの音声内容を「料理を作る」と設定した	
39	入力は BEGIN:RADIOBUTTON 　ACTION:move=2 …… END:RADIOBUTTON	ラジオボタンのアクション内容を取得した	ラジオボタンのアクション内容を「move=2」と設定した	
40	入力は BEGIN:RADIOBUTTON 　ENABLED:false …… END:RADIOBUTTON	ラジオボタンの初期状態を取得した	ラジオボタンの初期状態を「false」と設定した	
41	入力は BEGIN:GROUP 　NAME:eggGroup …… END:GROUP	グループの名前を取得した	グループの名前を「eggGroup」と設定した	
42	入力は BEGIN:GROUP MEMBER:eggPiece;eggBurntColor;eggMinute;eggSecond …… END:GROUP	グループのメンバーを取得した	グループのメンバーを「eggPiece;eggBurntColor;eggMinute;eggSecond」と設定した	
43	入力は BEGIN:IF IF:isEgg==1||isToast==1||isCoffee==1 …… END:IF	IFの内容を取得した	IFの内容を「isEgg==1||isToast==1||isCoffee==1」と設定した	
44	入力は BEGIN:IF DO:output=00#{isEgg:1}#{isToast:1}#{isCoffee:1}#{color:2}#{isNow:1};commandSend=1 …… END:IF	DOの内容を取得した	DOの内容を「output=00#{isEgg:1}#{isToast:1}#{isCoffee:1}#{color:2}#{isNow:1};commandSend=1」と設定した	
45	codeArrはnullではない	コマンド送信関数を呼び出して、コマンド送信用配列をクリア 変数保存用配列の値を元の	commandSendを呼び出す codeArrをクリア paraTblの値を元の値にセットする	

項番	前提条件	動作	期待結果	実測結果
1	引数は「卵焼き」	そのまま返す	卵焼き	
2	引数は 「{cookMinute}#分 #{cookSecond}#秒」	変数と結合して返す	変数cookMinuteの値＋分＋変数cookSecondの値＋秒	

項番	前提条件	動作	期待結果	実測結果
1	引数は BEGIN:GROUP NAME:eggGroup MEMBER:eggPiece;eggBurntColor;eggMinute;eggSecond END:GROUP	次の行を正しく取得できる	最後の行まで一行ずつ読み込む	
2	引数は BEGIN:GROUP NAME:breadGroup MEMBER:breadMinute;breadSecond BEGIN:GROUP NAME:amount MEMBER:onePiece;twoPiece END:GROUP	次の行を正しく取得できる	最後の行まで一行ずつ読み込む	

項番	前提条件	動作	期待結果	実測結果
1	引数は BEGIN:GROUP NAME:eggGroup MEMBER:eggPiece;eggBurntColor;eggMinute;eggSecond END:GROUP	コンポーネントは全部グループに入れる	IDは「eggPiece;eggBurntColor;eggMinute;eggSecond」のコンポーネントを全部グループに入れた	
2	引数は BEGIN:GROUP NAME:breadGroup MEMBER:breadMinute;breadSecond BEGIN:GROUP NAME:amount MEMBER:onePiece;twoPiece END:GROUP	子グループとコンポーネントは全部グループに入れる	IDは「breadMinute;breadSecond」のコンポーネントを全部グループに入れた 子グループも入れた	

項番	前提条件	動作	期待結果	実測結果
1	引数は 「eggGroup, eggPiece」	グループ名を返す	「eggGroup」を返す	
2	引数は 「breadGroup, onePiece」	子グループ名を返す	「amount」を返す	

項番	前提条件	動作	期待結果	実測結果
1	引数は「A.checked||B.checked->T.enabled」	フラグはor関係で設定する	A或はBはチェックされた時Tが使える	
2	引数は「C.checked&&D.checked->E.enabled」	フラグはor関係で設定する	CかつDはチェックされた時Eが使える	
3	引数は「C.checked->E.enabled」	フラグはCの状態より設定する	Cはチェックされた時Eが使える	
4	引数は「C.checked->E.enabled」	フラグはCの状態より設定する	CかつDはチェックされた時Eが使える	
5	引数は「C.checked->E.enabled」、Eはグループ名	グループE中のコンポーネントを全部設定する	Cはチェックされた時、groupSetを呼び出す	
6	引数は「output=11111111」	コマンド作成関数を呼び出して、コマンド配列に入れる	codeMakeを呼び出して、codeArrに入れる	
7	引数は「isCoffee=0」	変数を設定する	変数isCoffeeを「0」に設定する	
8	引数は「move=1」	変数を設定して、コンポーネント配列をクリア	変数moveを「１」に設定して、componentをクリアす	

項番	前提条件	動作	期待結果	実測結果
1	引数は 「11111111」	Byteに変換して、コマンド送信関数を呼び出す	chgBytesとcirSend.sendを呼び出す	

項番	前提条件	動作	期待結果	実測結果
1	引数は 「11111111」	byteの11111111を返す	byteのFFを返す	

項番	前提条件	動作	期待結果	実測結果
1	引数は「00#{isEgg:1}#{isToast:1}#{isCoffee:1}#{color:2}#{isNow:1};commandSend=」	計算関数を呼び出す、桁数不足の場合は「0」で埋める	CalcStringを呼び出す、桁数不足の場合は「0」で埋める	
2	引数は「11111111」	そのまま返す	「11111111」を返す	

項番	前提条件	動作	期待結果	実測結果
1	引数は 「11111111」	そのまま返す	「11111111」を返す	
2	引数は 「cookSecond + 10」	計算した結果を返す	変数cookSecondの値を 用いて計算した結果を返	
3	引数は 「cookSecond − 10」	計算した結果を返す	変数cookSecondの値を 用いて計算した結果を返	
4	引数は 「cookSecond * 10」	計算した結果を返す	変数cookSecondの値を 用いて計算した結果を返	
5	引数は 「cookSecond / 10」	計算した結果を返す	変数cookSecondの値を 用いて計算した結果を返	
6	引数は 「(hour − minute / 10) * 3」	計算した結果を返す	変数hourとminuteの値を 用いて計算した結果を返	
7	引数は 「(hour − minute) * 3」	計算した結果を返す	変数hourとminuteの値を 用いて計算した結果を返	

項番	前提条件	動作	期待結果	実測結果
1	引数は 「isCookMinuteSet=0」	変数の値をセットする	変数isCookMinuteSetの 値は「0」とセットする	

項番	前提条件	動作	期待結果	実測結果
1	引数は 「「A, checked」	Aのchecked状態を返す	Aのchecked状態を返す	

項番	前提条件	動作	期待結果	実測結果
1	引数は「eggGroup, eggGroup,″checked″, false」	グループ中のコンポーネントの状態を変更する	グループ中のコンポーネントのcheckedをfalseにする	
2	引数は「breadGroup, breadGroup,″checked″, false」	グループ中の全部コンポーネント（子グループにも）の状態を変更する	グループ中の全部コンポーネント（子グループにも）のcheckedをfalseにす	

項番	前提条件	動作	期待結果	実測結果
1	引数は 「A,″checked″, true」 Aのコンポーネント種類は CheckBox	Aのchecked状態を変更する	Aはチェックされた	
2	引数は 「A,″checked″, true」 Aのコンポーネント種類は RadioButton	Aのchecked状態を変更する	Aはチェックされた	
3	引数は 「A,″enabled″, false」 Aのコンポーネント種類は CheckBox	Aのenabled状態を変更する	Aは使用できなくなった	
4	引数は 「A,″enabled″, false」 Aのコンポーネント種類は RadioButton	Aのenabled状態を変更する	Aは使用できなくなった	
5	引数は 「A,″enabled″, false」 Aのコンポーネント種類は TextBox	Aのenabled状態を変更する	Aは使用できなくなった	
6	引数は 「A,″enabled″, false」 Aのコンポーネント種類は	Aのenabled状態を変更する	Aは使用できなくなった	

項番	前提条件	動作	期待結果	実測結果
1	引数は「isNow==1」	isNowの値によりtrue或はfalseを返す	isNowは「１」と等しい場合、trueを返す、等しくない場合、falseを返す	
2	引数は「isEgg==1&&isCookMinuteSet==0」	isEggかつisCookMinuteSetの値によりtrue或はfalseを返す	isEggは「１」と等しいかつisCookMinuteSetは「0」と等しい場合、trueを返す、片方は等しくない場合、falseを返す	
3	引数は「isEgg==1||isToast==1」	isEggかつisToastの値によりtrue或はfalseを返す	isEggは「１」と等しい或はisToastは「１」と等しい場合、trueを返す、両方は等しくない場合、falseを	

項番	前提条件	動作	期待結果	実測結果
1	引数は 「卵焼き」	compSpk.speakを呼び出す	compSpk.speakを呼び出す際、「卵焼き」は引数としてcompSpk.speakに渡	
2	引数は 「」	なし	なし	

項番	前提条件	動作	期待結果	実測結果
1	引数は 「setTime」	コンポーネントのオブジェクトを返す	IDは「setTime」のコンポーネントのオブジェクト	
2	引数は 「XXXXXX」	nullを返す	nullを返す	

付録 P

結合テスト項目表

ID	正常/異常	対象画面名	手順	確認項目	実施日	担当者	結果	修正日	修正したかどうか	修正確認	備考
1	正常	P1	「リモコンの選択」をクリック	P2に遷移	2012.1.4	持田・徐	○				
2	正常	P1	「リモコンのダウンロード」をクリック	P3に遷移	2012.1.4	持田・徐	○				
3	正常	P1	「家電情報の取得」をクリック	家電とIrDA通信して、仕様記述のファイル名を取得。その後、P1に遷移。	2012.1.4	持田・徐	○				
4	正常	P2	SDカードにファイルが保存されて	画面上の表示はなしで、「SDカードにファイルは保存されていません。」を発声。	2012.1.4	持田・徐	○				
5	正常	P2	項目を選択し、決定キー押下	家電の操作画面（おまかせミールさん等）に遷移	2012.1.4	持田・徐	○				
6	正常	P2	キャンセルキー押下	P1に遷移	2012.1.4	持田・徐	○				
7	正常	P3	ファイル一覧表がWeb上にない状態	画面上の表示はなしで、「ファイル一覧表が見つかりません。」を発声。	2012.1.4	持田・徐	○				
8	正常	P3	項目を選択し、決定キー押下	家電の仕様記述をWebからダウンロードし、P1に遷移	2012.1.4	持田・徐	○				
9	正常	P3	キャンセルキー押下	P1に遷移	2012.1.4	持田・徐	○				
10	異常	P3	選択したファイルがWeb上にない状	「ファイルが見つかりませんでした。」を発声。	2012.1.4	持田・徐	○				

手順	確認項目	実施日	担当者	結果	修正日	修正したかどうか	修正確認	備考
「料理を作る」をクリック	P2に遷移	2012.1.4	持田・徐	○				
「時間を設定する」をクリック	P6に遷移	2012.1.4	持田・徐	○				
「現作業中止」をクリック	P9に遷移	2012.1.4	持田・徐	○				
卵焼きが「1枚」、トーストが「なし」、コーヒーが「なし」、調理時間を「今すぐ」を選び、「次へ」ボタンをクリック	P3に遷移	2012.1.4	持田・徐	○				
卵焼きが「なし」、トーストが「1枚」、コーヒーが「なし」、調理時間を「今すぐ」を選び、「次へ」ボタンをクリック	P3に遷移	2012.1.4	持田・徐	○				
卵焼きが「なし」、トーストが「なし」、コーヒーが「あり」、調理時間を「今すぐ」を選び、「次へ」ボタンをクリック	P5に遷移	―	―	―	―	―	―	実装しなかったためテストしない
卵焼きが「1枚」、トーストが「なし」、コーヒーが「なし」、調理時間を「予約」を選び、「次へ」ボタンをクリック	P3に遷移	2012.1.4	持田・徐	○				
卵焼きが「なし」、トーストが「1枚」、コーヒーが「なし」、調理時間を「予約」を選び、「次へ」ボタンをクリック	P3に遷移	2012.1.4	持田・徐	○				
卵焼きが「なし」、トーストが「なし」、コーヒーが「あり」、調理時間を「予約」を選び、「次へ」ボタンをクリック	P4に遷移	―	―	―	―	―	―	実装しなかったためテストしない
卵焼きが「なし」、トーストが「なし」、コーヒーが「なし」、調理時間を「今すぐ」を選び、「次へ」ボタンをクリック	エラーアラートが出る	―	―	―	―	―	―	実装しなかったためテストしない
卵焼きが「なし」、トーストが「なし」、コーヒーが「なし」、調理時間を「予約」を選び、「次へ」ボタンをクリック	エラーアラートが出る	―	―	―	―	―	―	実装しなかったためテストしない
「戻る」ボタンをクリック	P1に遷移	2012.1.4	持田・徐	○				
P2で「今すぐ」を選んだ状態で、焼き加減が「普通」を選び、「次へ」ボタンをクリック	P5に遷移	―	―	―	―	―	―	実装しなかったためテストしない
P2で「予約」を選んだ状態で、焼き加減が「普通」を選び、「次へ」ボタンをクリック	P4に遷移	2012.1.4	持田・徐	○				
焼き加減が「時間」を選ぶ	焼き時間がEnabled	―	―	―	―	―	―	実装しなかったためテストしない

P2で「今すぐ」を選んだ状態で、焼き加減が「時間」を選び、焼き時間に「1分30秒」を選び、「次へ」ボタンをクリック	P5に遷移	―	―	―	―	―	―	実装しなかったためテストしない
P2で「予約」を選んだ状態で、焼き加減が「時間」を選び、焼き時間に「1分30秒」を選び、「次へ」ボタンをクリック	P4に遷移	2012.1.4	持田・徐	○				
焼き加減が「時間」を選び、焼き時間の分の部分を選ばない	エラーアラートが出る	―	―	―	―	―	―	実装しなかったためテストしない
焼き加減が「時間」を選び、焼き時間の秒の部分を選ばない	エラーアラートが出る	―	―	―	―	―	―	実装しなかったためテストしない
焼き加減が「時間」を選び、焼き時間の分と秒の部分を選ばない	エラーアラートが出る	―	―	―	―	―	―	実装しなかったためテストしない
「戻る」ボタンをクリック	P2に遷移	2012.1.4	持田・徐	×	2012.1.13	○	章	
予約時間を入力し「次へ」をクリック	P5に遷移	2012.1.4	持田・徐	○				
目覚ましを「設定する」をクリック	目覚まし時間がEnabled	―	―	―	―	―	―	実装しなかったためテストしない
予約時間を入力しないで「次へ」をクリック	エラーアラートが出る	―	―	―	―	―	―	実装しなかったためテストしない
目覚ましを「設定する」をクリックし、目覚まし時間を入力しないで「次へ」をクリック	エラーアラートが出る	―	―	―	―	―	―	実装しなかったためテストしない
「戻る」ボタンをクリック	P2に遷移	2012.1.4	持田・徐	×	2012.1.13	○	章	
「完了」ボタンをクリック	コマンドが送るし、全引数がデフォルト値に戻り、P9に遷移	2012.1.4	持田・徐	○				
「修正」ボタンをクリック	全引数がデフォルト値に戻り、P2に遷移	2012.1.4	持田・徐	○				
「キャンセル」ボタンをクリック	全引数がデフォルト値に戻り、P1に遷移	2012.1.4	持田・徐	○				
「目覚まし時間を設定する」ボタンをクリック	P7に遷移	2012.1.4	持田・徐	○				
「現在時刻を設定する」ボタンをクリック	P8に遷移	2012.1.4	持田・徐	○				
「戻る」ボタンをクリック	P1に遷移	2012.1.4	持田・徐	○				
「時」に入力する	入力した内容を画面に表示させ	2012.1.4	持田・徐	○				
「分」に入力する	入力した内容を画面に表示させ	2012.1.4	持田・徐	○				
「設定する」ボタンをクリック	P9に遷移	2012.1.4	持田・徐	○				
「戻る」ボタンをクリック	P6に遷移	2012.1.4	持田・徐	○				
「時」に入力する	入力した内容を画面に表示させ	2012.1.4	持田・徐	○				
「分」に入力する	入力した内容を画面に表示させ	2012.1.4	持田・徐	○				

「設定する」ボタンをクリック	P9に遷移	2012.1.4	持田・徐	○				
「メイン画面に戻る」ボタンをクリック	P1に遷移	2012.1.4	持田・徐	○				

付録Q

家電側結合テスト項目表

ID	テスト対象	前提条件	手順	確認項目	実施日	担当者	結果	修正日
1.1	家電側	調理を作っている状態	別紙−結合テスト用コマンド表のパタン番号1のコマンドを受ける	全部の操作を取り消しできる	12/21/2011	劉	○	
1.1	家電側	予約がある状態	別紙−結合テスト用コマンド表のパタン番号1のコマンドを受ける	全部の操作を取り消しできる	12/21/2011	劉	×（取り消しできない）	12/21/2011
1.1	家電側	アラムがある状態	別紙−結合テスト用コマンド表のパタン番号1のコマンドを受ける	全部の操作を取り消しできる	12/21/2011	劉	○	
2	家電側	なし	別紙−結合テスト用コマンド表のパタン番号2のコマンドを受ける	別紙−結合テスト用コマンド表のパタン番号2のコマンドの命令を全部に実行する	12/21/2011	劉	×（濃を設定できない）	12/21/2011
3	家電側	なし	別紙−結合テスト用コマンド表のパタン番号3のコマンドを受ける	別紙−結合テスト用コマンド表のパタン番号3のコマンドの命令を全部に実行する	12/22/2011	劉	○	
4	家電側	なし	別紙−結合テスト用コマンド表のパタン番号4のコマンドを受ける	別紙−結合テスト用コマンド表のパタン番号4のコマンドの命令を全部に実行する	12/22/2011	劉	○	
5	家電側	なし	別紙−結合テスト用コマンド表のパタン番号5のコマンドを受ける	別紙−結合テスト用コマンド表のパタン番号5のコマンドの命令を全部に実行する	12/22/2011	劉	○	
6	家電側	なし	別紙−結合テスト用コマンド表のパタン番号6のコマンドを受ける	別紙−結合テスト用コマンド表のパタン番号6のコマンドの命令を全部に実行する	12/22/2011	劉	×（予約の時間設定が不正である）	12/23/2011
7	家電側	なし	別紙−結合テスト用コマンド表のパタン番号7のコマンドを受ける	別紙−結合テスト用コマンド表のパタン番号7のコマンドの命令を全部に実行する	12/23/2011	劉	○	
8	家電側	なし	別紙−結合テスト用コマンド表のパタン番号8のコマンドを受ける	別紙−結合テスト用コマンド表のパタン番号8のコマンドの命令を全部に実行する	12/23/2011	劉	○	
9	家電側	なし	別紙−結合テスト用コマンド表のパタン番号9のコマンドを受ける	別紙−結合テスト用コマンド表のパタン番号9のコマンドの命令を全部に実行する	12/23/2011	劉	○	
10	家電側	なし	別紙−結合テスト用コマンド表のパタン番号10のコマンドを受ける	別紙−結合テスト用コマンド表のパタン番号10のコマンドの命令を全部に実行する	12/23/2011	劉	○	

11	家電側	なし	別紙−結合テスト用コマンド表のパタン番号11のコマンドを受ける	別紙−結合テスト用コマンド表のパタン番号11のコマンドの命令を全部に実行する	12/23/2011	劉	○
12	家電側	なし	別紙−結合テスト用コマンド表のパタン番号12のコマンドを受ける	別紙−結合テスト用コマンド表のパタン番号12のコマンドの命令を全部に実行する	12/23/2011	劉	○

パタン番号	コマンド系	2進数	16進数	反転2進数	反転16進数	意味(コードIDは「コマンド転送規則」を
1	c6	00000000	00	11111111	ff	C6(取消)
2	c1	00100100	24	11011011	db	C1(卵、濃、現在)
	c2	01011001	59	10100110	a6	C2(卵、1枚)
	c6	11111111	ff	00000000	00	C6(作業開始)
3	c1	00110010	32	11001101	ad	C1(卵、ト、淡、現在)
	c2	01011001	59	10100110	a6	C2(卵、1枚)
	c2	01111010	7a	10000101	85	C2(ト、2枚)
	c6	11111111	ff	00000000	00	C6(作業開始)
4	c1	00100110	26	11011001	b9	C1(卵、調理時間あり、現在)
	c2	01001010	4a	10110101	b5	C2(卵、10分)
	c2	01010101	55	10101010	aa	C2(卵、50秒)
	c6	11111111	ff	00000000	00	C6(作業開始)
5	c1	00001000	08	11110111	f7	C1(コ、普通、現在)
	c6	11111111	ff	00000000	00	C6(作業開始)
6	c5	10111110	be	01000001	41	C5(現時刻設定)
	c3	10001111	8f	01110000	70	C3(AM、15時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c1	00100001	21	11011110	de	C1(卵、普通、予約)
	c2	01011001	59	10100110	a6	C2(卵、1枚)
	c3	10000000	80	01111111	7f	C3(PM、0時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c6	11111111	ff	00000000	00	C6(作業開始)
7	c5	10111110	be	01000001	41	C5(現時刻設定)
	c3	10001111	8f	01110000	70	C3(AM、15時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c1	00110001	31	11001110	ce	C1(卵、ト、普通、予約)
	c2	01011001	59	10100110	a6	C2(卵、1枚)
	c2	01111010	7a	10000101	85	C2(ト、2枚)
	c3	10000000	80	01111111	7f	C3(PM、0時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c6	11111111	ff	00000000	00	C6(作業開始)
8	c5	10111110	be	01000001	41	C5(現時刻設定)
	c3	10001111	8f	01110000	70	C3(AM、15時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c1	00001001	09	11110110	f6	C1(コ、普通、予約)
	c3	10000000	80	01111111	7f	C3(PM、0時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c6	11111111	ff	00000000	00	C6(作業開始)
9	c5	10111110	be	01000001	41	C5(現時刻設定)
	c3	10001111	8f	01110000	70	C3(AM、15時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c1	00100001	21	11011110	de	C1(卵、普通、予約)
	c2	01011001	59	10100110	a6	C2(卵、1枚)
	c3	10000000	80	01111111	7f	C3(PM、0時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c5	10111111	bf	01000000	40	C5(アラーム設定)
	c3	10000000	80	01111111	7f	C3(PM、0時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c6	11111111	ff	00000000	00	C6(作業開始)
10	c5	10111110	be	01000001	41	C5(現時刻設定)
	c3	10001111	8f	01110000	70	C3(AM、15時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c1	00110101	35	11001110	ca	C1(卵、ト、濃、予約)
	c2	01011001	59	10100110	a6	C2(卵、1枚)
	c2	01111010	7a	10000101	85	C2(ト、2枚)
	c3	10000000	80	01111111	7f	C3(PM、0時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c5	10111111	bf	01000000	40	C5(アラーム設定)

	c3	10000000	80	01111111	7f	C3(PM、0時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c6	11111111	ff	00000000	00	C6(作業開始)
11	c5	10111110	be	01000001	41	C5(現時刻設定)
	c3	10001111	8f	01110000	70	C3(AM、15時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c1	00001001	09	11110110	f6	C1(コ、普通、予約)
	c3	10000000	80	01111111	7f	C3(PM、0時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c5	10111111	bf	01000000	40	C5(アラーム設定)
	c3	10000000	80	01111111	7f	C3(PM、0時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c6	11111111	ff	00000000	00	C6(作業開始)
12	c5	10111110	be	01000001	41	C5(現時刻設定)
	c3	10001111	8f	01110000	70	C3(AM、15時)
	c4	11000000	c0	00111111	3f	C4(0分)
	c6	11111111	ff	00000000	00	C6(作業開始)

付録 R

総合テスト項目表

項目番号	テスト名	正常/異常	手順	確認項目	実施日	担当者	結果	修正日	修正リビジョン	修正したかどうか	修正確認	備考
	携帯側											
1	HTTP通信機能	正常	1. 携帯の中の全ての仕様を削除 2. TOP画面のP1から「リモコンのダウンロード」を選択 3. 「IOC－125」を選択	携帯のSDカードに仕様がダウンロードされたことを確認	2012.1.5	章・徐	○					
2	画面描画機能と通信仕様解釈機能	正常	1. TOP画面のP1に「リモコンの選択」を選択 2. 「IOC－125」を選択	おまかせミールさんのメイン画面が生成される	2012.1.5	章・徐	○					
3	音声読み上げ機能	正常	1. おまかせミールさんのメイン画面を開く	おまかせミールさんのメイン画面で音声が出せることを確認	2012.1.5	章・徐	○					
4	調理機能（卵だけ）	正常	1. 卵焼きが「1枚」、トーストが「なし」、コーヒーが「なし」、調理時間が「今すぐ」、焼き加減が「濃」と設定して「完了」ボタンをクリック	おまかせミールさんが調理始めることを確認	2012.1.5	章・徐	○					
5	調理機能(卵、トースト)	正常	1. 卵焼きが「1枚」、トーストが「2枚」、コーヒーが「なし」、調理時間が「今すぐ」、焼き加減が「普通」と設定して「完了」ボタンをクリック	おまかせミールさんが調理始めることを確認	2012.1.5	章・徐	○					
6	調理機能（卵、調理時間）	正常	1. 卵焼きが「1枚」、トーストが「なし」、コーヒーが「なし」、調理時間が「今すぐ」、焼き加減が「時間」、分が「5」、秒が「30」と設定して「完了」ボタンをクリック	おまかせミールさんが調理始めることを確認	2012.1.5	章・徐	○					

7	調理機能（コーヒー）	正常	1. 卵焼きが「なし」、トーストが「なし」、コーヒーが「あり」、調理時間が「今すぐ」と設定して「完了」ボタンをクリック	おまかせミールさんが調理始めることを確認	2012.1.5	章・徐	○					
8	予約機能（卵）	正常	1. 卵焼きが「1枚」、トーストが「なし」、コーヒーが「なし」、調理時間が「予約」、予約時間が「3時50分」と設定して「完了」ボタンをクリック ＊15時16分	おまかせミールさんが調理を予約したことを確認	2012.1.5	章・徐	○					
9	予約機能（卵、トースト）	正常	1. 卵焼きが「1枚」、トーストが「2枚」、コーヒーが「なし」、調理時間が「予約」、予約時間が「3時50分」と設定して「完了」ボタンをクリック ＊15時30分	おまかせミールさんが調理を予約したことを確認	2012.1.5	章・徐	○					
10	予約機能（コーヒー）	正常	1. 卵焼きが「なし」、トーストが「なし」、コーヒーが「あり」、調理時間が「予約」、予約時間が「3時50分」と設定して「完了」ボタンをクリック	おまかせミールさんが調理を予約したことを確認	2012.1.5	章・徐	○					
11	予約機能（卵、アラーム）	正常	1. 卵焼きが「1枚」、トーストが「なし」、コーヒーが「なし」、調理時間が「予約」、予約時間が「3時50分」、アラームが「3時40分」と設定して「完了」ボタンをクリック ＊15時30分　15時20分	おまかせミールさんが調理を予約したことを確認	2012.1.5	章・徐	○					

12	予約機能（卵、トースト、アラーム）	正常	1. 卵焼きが「1枚」、トーストが「2枚」、コーヒーが「なし」、調理時間が「予約」、予約時間が「3時50分」、アラームが「3時40分」と設定して「完了」ボタンをクリック ＊15時30分　15時25分	おまかせミールさんが調理を予約したことを確認	2012.1.5	章・徐	○					
13	予約機能（コーヒー、アラーム）	正常	1. 卵焼きが「なし」、トーストが「なし」、コーヒーが「あり」、調理時間が「予約」、予約時間が「3時50分」、アラームが「3時40分」と設定して「完了」ボタンをクリック ＊15時30分　15時25分	おまかせミールさんが調理を予約したことを確認	2012.1.5	章・徐	○					
14	現在時刻設定機能	正常	現在時刻を「2時30分」と設定して「完了」ボタン をクリック ＊15時14分	おまかせミールさんが時間を設定したことを確認	2012.1.5	章・徐	○					
15	アラーム設定機能	正常	アラームを「2時30分」と設定して「完了」ボタン をクリック ＊15時30分	おまかせミールさんがアラームを設定したことを確認	2012.1.5	章・徐	○					
16	取り消し機能	正常	おまかせミールさんが作業を居ている状態で「現作業中止」をクリック ＊上記設定に関して全部OK	おまかせミールさんが現作業中止したことを確認	2012.1.5	章・徐	○					
	命令コマンド作成機能	正常	―	―	―	―	―	―	―	―	―	テスト済み
	CIR送信機能	正常	―	―	―	―	―	―	―	―	―	テスト済み
	IrDA送受信機能	正常	―	―	―	―	―	―	―	―	―	実装しないためテストしない
	家電側											

	IrDA送受信機能	正常	ー	ー	ー	ー	ー	ー	ー	ー	ー	実装しないためテストしない
	CIR受信機能	正常	ー	ー	ー	ー	ー	ー	ー	ー	ー	テスト済み
	おまかせミールさん状態読み取り機能	正常	ー	ー	ー	ー	ー	ー	ー	ー	ー	テスト済み
	おまかせミールさん制御機能	正常	ー	ー	ー	ー	ー	ー	ー	ー	ー	テスト済み

付録 S

被験者実験の評価アンケート

家電音声化システムの実証実験アンケート

本日は私たちの実証実験にご参加頂き誠にありがとうございます。今後の改善の参考にさせて頂きますので、お手数ですが下記アンケートへご協力をお願いいたします。

Q1.音声応答家電を所持していますか？

A. **はい** B. **いいえ**

Q2. 音声応答家電について、どのように思いますか？

A, 高価 B.商品が少ない又はない C. その他(　　)

Q3. 安価でいろいろな家電に対して、音声応答機能がついたらどのように思いますか？

(　　)

もし付けるとしたら、値段は原価の何％又は何円以内に収めて欲しいですか？

(　)％ 又は (　)円

Q4. 普段よく利用する家電と音声応答機能をつけたい家電を記入してください。

よく利用する家電　　(　　)

音声応答機能をつけたい家電　　(　　)

Q5　下記について、操作内容はわかりやすかったですか？

①家電情報のダウンロード

５ － ４ － ３ － ２ － １

②メニューの選択

５ － ４ － ３ － ２ － １

③確認画面

５ － ４ － ３ － ２ － １

④アナウンスの内容はわかりやすい

５ － ４ － ３ － ２ － １

Q6. 家電を直接操作するより本システムを利用した操作は使いやすかったですか？

５ － ４ － ３ － ２ － １

Q7. 使いやすかったと感じた点は何ですか？

（　　）

Q8. いままで家電の時間設定は困難だったと思いますが、本システムの時間操作は使いやすかったですか？

５ － ４ － ３ － ２ － １

Q9. 操作内容に対してストレスを感じましたか？

５ － ４ － ３ － ２ － １

Q10. Q9について具体的にどんなことにストレスを感じましたか？

具体的な内容を記入してください。

（　　）

Q11. 将来本システムを利用していろいろな家電を操作したいと思いますか？

５ － ４ － ３ － ２ － １

Q12. その他、本システムに対するご意見、ご感想等があればご記入ください

（　　）

以上でアンケートは終わりです。

ご協力、ありがとうございました。