平成 17 年度(2005 年度)

信州大学大学院工学系研究科 修士学位論文

ホームセキュリティシステムの検討と実装

信州大学工学研究科 情報工学専攻

03TA603C 山下祐司郎

2006 年 1 月 16 日

目次

図目次

表目次

数式目次

第1章　　序論

ピッキングに代表されるような侵入犯罪はここ数年増加の一途をたどっている。特に最近は、不在時間を調査した上での犯行や侵入後の短時間での犯行など、犯罪の多様化や巧妙化が進んでおり、家庭におけるさらなる防犯の意識・セキュリティシステムへの関心が高まってきている。

そのような背景から、最近ではインターネットを用いて遠隔地より監視できるネットワークカメラ等の個人ユースな防犯機器をはじめ、セキュリティ会社による家庭用をターゲットにしたセキュリティサービス等、様々な防犯対策が提案されており、需要は日増しに高まってきている。

そこで本論文では、手軽・安価に実現できるホームセキュリティシステムの検討・提案を行い、実際に設計・実装し有用性の確認・評価した結果を報告する。

第2章　　研究の背景

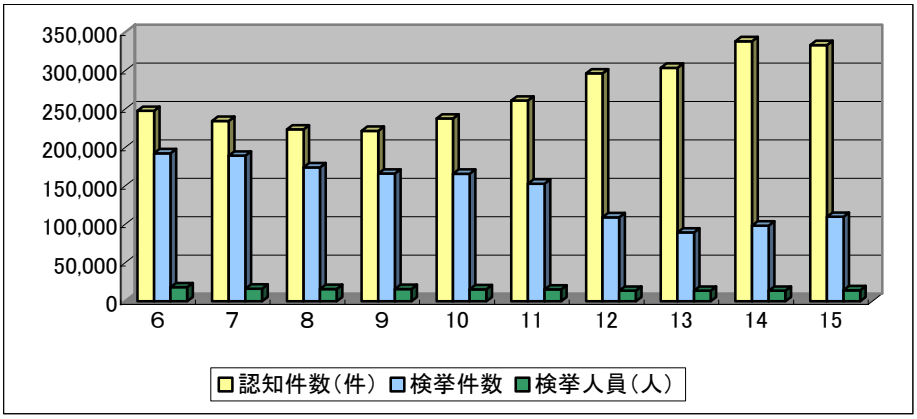


図 2-1 侵入盗の認知・検挙状況の推移（平成 6～15 年）[i]

図 2-1 に示したように昨今の侵入犯罪の増加は目覚しいものがある。侵入方法もピッキング、サムターン回し、カム送りといったドアを破壊するもの。窓枠や窓ガラスを破壊するものと、様々な手段がいたちごっこの様に出現している現状に加えて、そのような様々な防犯対策にもかなりの出費を必要とし、全てにおいて完璧に対策された状態というのは無いと言い切れるのではないだろうか。

そのような背景からか、最近はコンシューマ向けのセキュリティ鍵や侵入を検知するカメラ、ライト等防犯対策機器が様々なメーカーから販売されるようになっている。また、家庭をターゲットにしたセキュリティサービスも各警備会社でラインナップされており、ホームセキュリティに対する需要の多さを物語っている。

実際に近所を見渡しても、監視カメラや人感知ライト等のセキュリティシステムを導入している家庭が多く見られるようになった。しかしながら、そのような装備は家に工事が必要であったり、まだまだ価格も安価とは言い難く、導入に関しては抵抗がある人が多いのも事実である。

このような状況であるので、安価で手軽に導入できる実用的なホームセキュリティシステムに対しては相当な需要があると思われる。

[i] 平成 16 年度 警察白書より（http://www.npa.go.jp/hakusyo/h16/index.html)[1]

第3章　　研究の動機

侵入犯罪の防止には、家の鍵を各種犯罪対策用に強化する等の侵入そのものを困難にする方法がまず考えられる。しかしながら完璧な防御など無いといっても過言ではない。そこで、侵入の防止だけでなく、万が一侵入された時の早期発見・通知が被害を最小限に食い止める方法であるといえる。その早期発見・通知を目的としたホームセキュリティシステムに注目した。

弊信州大学大学院工学系研究科においても、カリキュラムの中で PIC を用いた簡易的なホームセキュリティシステムの演習があるように、ドアの開け閉めを感知するというシンプルなものから、各種センサを用いたもの、カメラによる常時監視等々家庭用のホームセキュリティシステムには、多種多様な製品が現在販売されている。しかしながらどの製品も一長一短であり、単体のみで完全に防犯出来るといったものは無いに等しい。また、値段も安いものでも 2、3 万円以上かかるだけでなく、取り付けに際し工事が必要であるものも多く、導入し難い商品であることは否めない。

導入のしやすさを一番のポイントとし、現在最も手軽に導入できる方法として注目しているものにネットワークカメラがある。ネットワークカメラとはインターネットを利用し遠隔地より自宅に設置されたカメラ画像を確認できるといったカメラである。

従来のこのようなカメラは常時稼動するパソコンが必要であり、そのパソコンをサーバに用いて、カメラの画像を外部クライアントに表示するといったような形態が多かった。しかし最近のネットワークカメラは単体でイーサネットに接続でき、家庭では常時接続のインターネット回線があれば良いという非常に手軽な形となっており、加えて部屋の中に置くという設置スタイルは複雑な工事も不要で、導入の手軽さにおいては群を抜いている製品である。但し、安価なものは見ることしかできず、動くものの感知となると相当高価なものとなってしまう。また、赤外線や超音波等の様々なセンサを用い、侵入を検知して知らせるといった製品もあるが、通知が来たからと言って誤動作なのかどうかの判断が非常に難しいといった事もあるために、即通報というのもなかなか勇気の要るものではないかと思われる。

そこで、それらの長所を組み合わせた実用的で手軽に導入でき、且つ安価なものがあれば相当魅力的な商品になるのではないか。加えて、犯罪とは別に火事等の留守宅における災害も気がかりの一つであり、ホームセキュリティシステムで家庭を監視するという手段は、災害の監視にも共用でき、実際に多くのホームセキュリティシステムがそういった災害の検知も機能に含まれているので、その機能もコスト増にならず含んでいればより魅力は増すのではないか。ということから、手軽・安価・侵入検知・災害検知・通知機能という特徴をもった商品を検討・提案するというのが本研究の動機である。

第4章　　研究の目的

本研究を通して、実際の商品を開発することをターゲットに考え、異常感知且つ自動通知型という機能をこのシステムの一番の特徴とする安価で実用的な製品の検討・提案を行うと共に、実際にシステムの設計・実装そして運用を行い、その実用性、優位性を実際に確かめ評価することを目的とする。

第5章　　関連研究

本大学大学院におけるホームセキュリティ関連の研究で 2003 年度以降の研究には以下のものがある。

- 2003 年度 修士論文
  - マイクロ波センサ利用による防犯・セキュリティーシステム (02TA588B 馬場智也氏)[2]
- 2004 年度 修士論文
  - 各種センサを活用したホームセキュリティシステムの遠隔操作 (03TA532A 河野志行氏)[3]

第6章　　システム概要

6.1　　前提

セキュリティシステムを構築するにあたり、商品として開発する際のターゲットを絞る為に以下に示す前提条件を決めた。

- 安価

コストをかければ多機能・高性能な物が出来るのは当たり前である。本研究では安価であるということを最重視し、その中でより実用的なものを目指すことにする。よって、既製品の組み合わせではコストがかかるだけでなく消費電力というランニングコストの面でも不利になるので、ワンチップマイコンを用いた単独でのシステムを開発することとした。

- 手軽

ワンチップマイコンを用いたシステムとするには、まずは部品の入手性が重要である。

本システムの製作においては個人レベルでの部品入手・実装を前提とせざるを得ない状況であるために、多数の販売を見込める企業であれば手に入るが個人では入手困難という部品は使用し難い。仮に入手できたとしても部品の形状の問題で実装が困難であるという部品も使用し難い。やはり秋葉原に代表される個人顧客を対象とする電子部品販売店や通販等で容易に入手できる部品であれば使いやすいので、そのような入手の容易性、実装の容易性を重視した部品を極力使用することとする。

- 環境

イニシャルコスト、省電力によるランニングコストを考えて単体で動作する製品とする。また、異常の通知には、国内で最大の端末数を誇る携帯電話のメール機能(E-Mail)を利用することとし、インフラとして常時接続型のインターネット環境があることを前提とする。ブロードバンドが復旧している現在では常時接続型のインターネット環境は多くの場合は問題ないと思われる。

6.2　　特徴

本システムの特徴を以下に示す。

- 留守を前提とした家の中の監視を常時行い、異常を感知した際には自動で通知を行うというプッシュ型を特徴とする。
- 異常の監視方法は、侵入犯を想定したカメラによる画像と火災を想定した温度センサによる室温の監視の 2 種類とする。
- 異常の検知方法は、カメラ画像の変化によるものと、温度の急激な変化によるものの 2 種類を用いる。
- 外部への通知は、SMTP(RFC[ii]-2821[4])を用いたE-Mailによる方法を用いる。

[ii] Request for Comments

- HTTP サーバを並行して動作させることで、WEB ブラウザでのアクセスにより外部からの状態モニタを可能とする。
- HTTP サーバによる状態の表示だけでなく CGI によりクライアント側からシステムへ簡易的なメッセージ通知機能を設ける。

このために LCD と LED を設置し、LCD には表示領域に表示できる範囲で外部からメッセージを書き込めるという機能と LED の点灯状態を制御するという 2 種類の方法で、留守宅に先に帰宅する家人への簡易的なメッセージ機能とすることを想定する。

- システム単体での異常状態がわかるように、異常時には LED 点滅を行うこととする。

インターネットに関する技術の標準を定める団体である IETF が正式に発行する文書
IETF…Internet Engineering Task Force (http://www.ietf.org/home.html)

第7章　　システムの検討

前章で示したシステムの構築を本章で検討する。

本システムにおいては、おおまかにネットワーク通信、画像取り込み、画像処理、温度値取り込みという処理を行う。これらの処理をリーズナブル且つ低消費電力で実現するにはワンチップマイコンに周辺機能を加えたシステムが最適と思われる。

以下にシステムのハードウェア各部についての検討を示す。

7.1　　メインプロセッサ

メインプロセッサを検討する前に、まずシステムとしてのワークフローを考える。

主に行うタスクとしては、温度測定、カメラ処理（撮影＋画像解析）のメインタスクとHTTP サーバ、CGI による各種ハードウェア制御である。

温度測定に関しては、一定間隔（間隔は設定可能とする）で行い記録する。

カメラ処理に関しては、記録まで行うとメモリやストレージに対する要求が高くなり、コストに大きく反映するので、現在の画像と異常を検知した時の画像のみメモリに保存することにする。カメラ撮影は間隔を長く取ると消費電力に対しては有利になるが、不感時間が増えセキュリティが甘くなる欠点があるので、可能な限りカメラは撮り続け監視を行うこととする。

以上これらの処理はメインプログラムで動かすのでメインタスクと呼ぶこととする。このメインタスクはシステム稼動中は常に動作している状態である。

このメインタスクに加えて不定期に発生するタスクとして、ネットワークからの外部アクセスがある。そのネットワークからのアクセスを受け持つのが HTTP サーバである。HTTP サーバではインターネット上で WEB ブラウザと呼ばれる HTTP クライアントからのアクセスを検知し、ステータス表示、CGI 処理を行う。このタスクは外部からのアクセス時にのみ発生するので不定期である。

これらの主要なタスクを考慮し、必要最小限に処理できる能力と、必要な機能（ネットワーク機能、カメラ接続I/F、温度センサ接続I/F）を持ち合わせたプロセッサが必須であるが、残念ながらこれら全ての機能を内蔵した安価で入手可能なワンチップマイコンが見当たらなかったので、ネットワーク機能（イーサネットコントローラ）を外付けLSIに頼ることにし、ルネサス テクノロジ[iii]社の 16 ビット CISCマイコンであるH8 を選択した。

ネットワークを外付けということであれば、他にもいくつか候補になるマイコンはあるのだが、C言語で開発でき、プロトコルスタックを備えたマルチタスクOSとその開発環境が整っており、また、マイコンキットを多く手がけている 秋月電子通商[iv]にて、H8 に加えてイーサネットコントローラ、DRAMが実装されるキット基板が販売されており入手可能という面を重視した。

H8 はマイコンのシリーズの名前でありいくつか種類があるのだが、値段が大きく違いが無かったので、内蔵メモリ量の大きなH8/3069F[5]を用いているAKI-H8/3069F LANボード（図 7-1）というキットをシステムの核として用いることとした（回路図は付録 [A]参照）。

[iii] 株式会社 ルネサス テクノロジ（ http://japan.renesas.com/ ）

[iv] 東京秋葉原にある電子部品の小売商(http://akizukidenshi.com/)

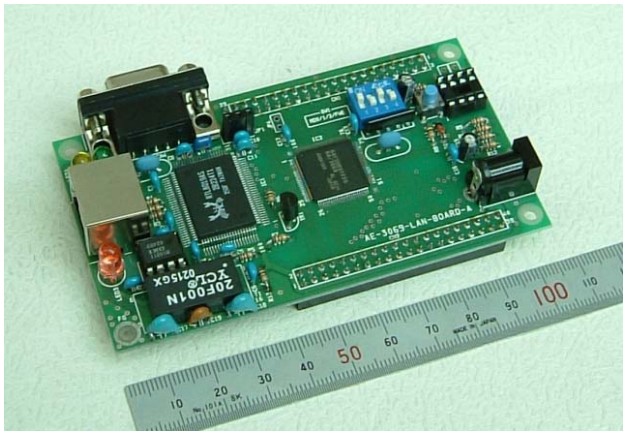


図 7-1 AKI-H8/3069F LAN ボード

このキットの特徴を以下に示す。

- H8/3069F 搭載
- NE2000 互換ネットワークコントローラ(RTL8019AS)搭載
- オンボードにて H8 内蔵フラッシュ ROM に書き込み可能
- 16MbitDRAM 搭載(8bit 接続)
- RS232C ドライバ搭載
- IO ポート用コネクタ実装

また、H8/3069F の特徴を以下に示す。

- 汎用レジスタマシン
  - 汎用レジスタ 16bit×16 本 (8bit×16 本+16bit×8 本、32bit×8 本としても使用可能)
- 高速動作
  - 最大動作周波数 25MHz
  - 加減算 80ns
  - 乗除算 560ns
- アドレス空間
  - 16MByte
- 内蔵メモリ
  - FlashROM 512kByte
  - RAM 16kByte
- 割り込みコントローラ
  - 外部割込み端子 7 本(NMI、IRQ0-5)
  - 内部割込み要因 36 要因
  - 3 レベルの割り込み優先順位が設定可能
- バスコントローラ
  - アドレス空間を 8 エリアに分割し、エリアごとに独立してバス仕様を設定可能
  - エリアごとにチップセレクト出力可能

- エリアごとに 8 ビットアクセス／16 ビットアクセスを設定可能
- エリアごとにウェイトのステート数を設定可能
- DRAM コントローラ内蔵により DRAM 直接接続可能

- DMA コントローラ
  - 最大 4 チャネル
- タイマ
  - 16bit Timer　　3ch
  - 8bit Timer　　4ch
- プログラマブルタイミングパターンコントローラ
  - 16bitTImer をベースとした最大 16 ビットのパルス出力が可能
- ウォッチドッグタイマ
  - 1 チャネル
- シリアルコミュニケーションインターフェース
  - 3 チャネル
- A/D 変換器
  - 10bit
  - 8ch
  - サンプル&ホールド機能付き
- D/A 変換器
  - 8bit
  - 2ch
- I/O ポート
  - 入出力端子　　70 本
  - 入力端子　　9 本

7.2　ネットワークコントローラ（RTL8019AS [6]）

RTL8019ASは Realtek[v]社のマークを称して俗にカニチップと呼ばれるI/Fが容易な非常にメジャーなLSIであり、先述した 秋月電子通商のAKI-H8/3069F LANボードキットに実装されている。

特徴を以下に示す。

- 製造　Realtek Semiconductor Corp.[v]
- 特徴
  - 100-pin PQFP
  - 16K byte SRAM built in
  - Compliant to Ethernet II and IEEE802.3 10Base5, 10Base2, 10BaseT
  - Supports UTP, AUI & BNC auto-detect
  - Software compatible with NE2000 on both 8 and 16-bit slots

[v] Realtek Semiconductor Corp.（http://www.realtek.com.tw/）

- Supports jumper, jumperless and PnP modes
- Supports PnP auto detect mode
- Supports Full-Duplex Ethernet function to double channel bandwidth
- Supports three level power down modes:
  - Sleep
  - Power down with internal clock running clock
  - Power down with internal halted
- Built-in data prefetch function to improve performance
- Supports auto polarity correction for 10BaseT
- Support 8 IRQ lines
- Supports IO address fully decode mode
- Supports 16K, 32K, 64K and 16K-page mode access to BROM (up to 256 pages with 16K bytes/page), and Flash memory read/write
- Supports BROM disable command to release memory after remote boot
- Supports flash memory read/write
- 16k byte SRAM built in
- Use 9346(64*16-bit EEPROM) to store resource configurations and ID parameters
- Capable of programming blank 9346 on board for manufacturing convenience
- Support 4 diagnostic LED pins with programmable outputs
- Improves the blinking problem of CRS LED when the cable is not connected

7.3 DRAM

HM5117805CFT-60という東芝セミコンダクター[vi]社製の16Mbit（2kword×8bit）のDRAMであり、8ビットモードでH8と接続されキットに実装される。

既に東芝においてはディスコンになっているようであるが、標準的なDRAMであり、H8がサポートしているDRAMであれば他社製品に置き換え可能である。

特徴を以下に示す。

- 16Mbit （2k-Word × 8-bit）
- FastPageモード搭載
- 5V単電源にて動作可能
- 60nsのアクセスタイム
- 各種リフレッシュモード搭載
  (RAS only refresh, CAS before RAS refresh, Hidden refresh, self refresh)
- 全I/Oピン TTLレベルコンパチブル

[vi] http://www.semicon.toshiba.co.jp/

7.4　温度センサ（LM35DZ[7]）

温度センサは NationalSemiconductor[vii]社のLM35DZを用いた。

以下に特徴を示す。

- 摂氏(℃) 温度に直接較正されている
- 温度係数はリニアで＋10.0mV/℃
- ＋25℃において 0.5℃の精度を保証
- －55℃～＋150℃の温度範囲
- リモート・アプリケーションに最適
- ウェハ・レベル・トリミングによる低コスト化
- 4 ～ 30V の動作電源電圧範囲
- 60μA 以下の電流ドレイン
- 低自己発熱、静止空気で 0.08℃
- ±1/4℃以下の非直線性(代表値)
- 低出力インピーダンス、1mA 負荷で 0.1Ω

アナログ出力ではあるが、H8/3069F に A/D 変換機能が備わっているので、それにて値を読み取ることとする。また、温度係数がリニアであるというのが最大の特徴で、温度係数がリニアであれば H8 マイコンに大きな負荷をかけずに温度値の算出が可能となるので、このデバイスが最適であると思われる。

7.5　リアルタイムクロック（RTC）

キット単体で、ネットワークに接続でき、TCP、UDP の実装ができているので、UDP による NTP (Network Time Protocol)を実装すれば正確な時刻情報は得られることができる。

しかし、正確な時間を維持するためには H8 のタイマ割り込み等を用いる必要があり、その動作は H8 にとって大きな負荷となるばかりか誤差も発生する。そこで、安価な RTC を用いることにした。

RTC を用いれば誤差は無視できる精度にはなるのだが、RTC への設定の手間、電源 OFF 時のバックアップの問題も考慮し、ネットワークを用いた時刻合わせである NTP を実装。RTC はバッテリーバックアップせずに電源投入時には NTP にて RTC へ設定を行うこととする。また定期的に NTP を用いて RTC を再設定し、誤差をより少ないものへ補正することとする。

RTCには、エプソントヨコム[viii]社の RTC-8564NB[ix][8]を使用し、使用しやすいように 8pinDIPサイズに変換されたリアルタイムクロックモジュールが 秋月電子通商にて販売されているので、それを用いることにした。

以下にこのデバイスの特徴を示す。

- 32.768kHz の水晶振動子を内蔵

[vii] http://www.national.com/
[viii] http://www.epsontoyocom.co.jp/
[ix] http://www.epsontoyocom.co.jp/product/RTC/RTC-8564JE_NB/RTC-8564JE_NB.html

- I^2C[x]高速バス規格(400kHz)対応
- アラーム機能、タイマ機能、周波数出力機能 (32.768kHz、1024Hz、32Hz、1Hz)
- 低消費電流 275nA/ 3V (Typ.)
- 低電圧計時保持(1.0～5.5V)

このようにインターフェースは I^2Cである。H8/3069F自体は I^2C機能は有していないが、汎用ポートをソフトウェアから制御することで I^2C機能を実現することとする。

7.6 カメラ (BT656 カメラ、Treva)

室内の画像を監視し、画像の変化から異常を検出する為には、CCD カメラや CMOS イメージセンサ等の画像データを取得できるデバイスが必要である。

画像の異常の検出は、カメラの画像をマイコンが処理することにより得ることとするので、カメラのデータはコンポジット出力のようなアナログではなく、デジタル出力の方が親和性が高い。画像のデジタルデータのインタフェースでは ITU-R[xi] BT.656[xii]規格に沿ったフォーマットが多く用いられており、実際に 秋月電子通商にてこのBT.656 規格に準拠したデータを出力するカメラユニットが販売されていたので、これを購入し検討を行った。

このカメラの仕様を以下に示す。

- 出力 NTSC ITU-R BT.656
- 画素 542x496 (26 万画素)
- 最低照度 0.5lux F1.2 5600° K
- S/N 60dB
- 電子シャッター 1/60～1/120,000sec
- デジタル出力 ITU-R BT.656 (8bit)
- 消費電力 DC12V 100mA

しかし、このカメラは性能的には十分であるものの、27MHzクロックとそれに同期して8bitデータが出力されるために、CPUで直接取り込むことが出来ずにデータを仲介させる外部回路が必要となる。図 7-2 にその仲介部分の検討したブロック図を示す。

---

[x] Philips Semiconductors 社が提唱しているシリアル・バス
http://www.jp.semiconductors.philips.com/markets/mms/protocols/i2c/index.html

[xi] International Telecommunication Union (http://www.itu.int/home/index.html)内の ITU Radiocommunication Sector
(http://www.itu.int/ITU-R/index.html)

[xii] Interfaces for digital component video signals in 525-line and 625-line television systems operating at the 4:2:2 level of Recommendation ITU-R BT.601 (Part A)

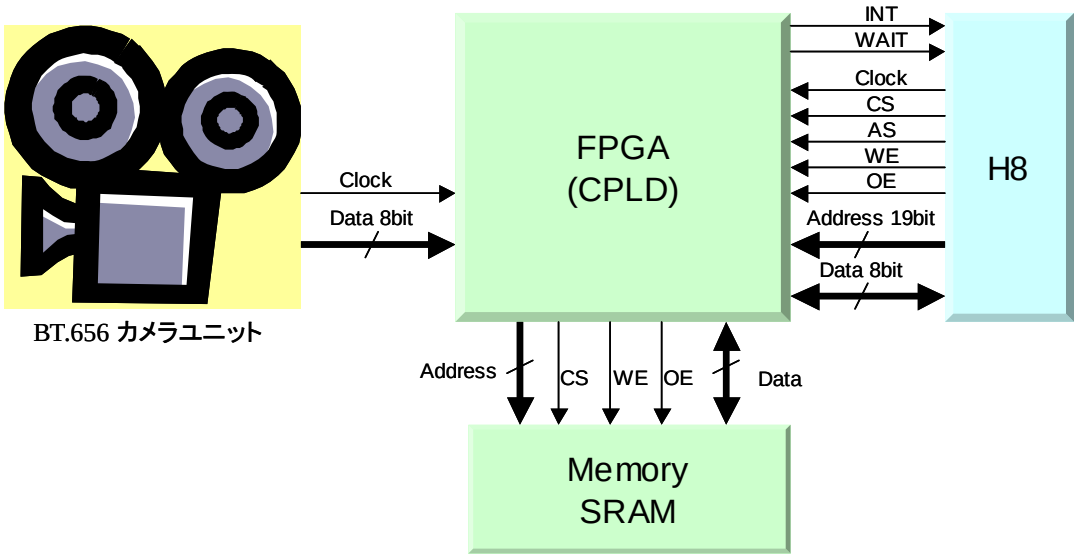


図 7-2 BT.656 カメラユニット接続回路ブロック図

取り込み、画像のスタート検出等を考えると、ディスクリートな部品のよる回路を組むよりは、CPLD や FPGA を用いる方が現実的である。加えて、BT.656 カメラは常にデータを送り出してくるので、画像の貯蔵用のワークメモリとの組み合わせを行い、CPU からはインターリーブして読み出す必要がある。この仲介を行う機能の FPGA(CPLD)の回路は複雑ではなく、大規模な回路にはならないが、ワークメモリへの読み書きと、H8 へのメモリにマッピングを行うようにする接続を考慮するとピン数が非常に多くなってしまう(100 ピン程度)。またメモリや FPGA の入手性、値段、実装の困難さ等の問題、そして何よりもこのカメラ自体の 8200 円という高価な値段であり、それらを総合的に検討した結果、システムのコンセプトにあわないと判断し、このカメラは採用を見送った。

そこで、より安価で容易にH8 に接続可能なカメラを探して見つけたのが、今回用いたCMOSイメージセンサを利用したPHS用カメラユニットTrevaである(図 7-3)。Trevaとは 京セラ[xiii]社が開発したPHS(feel H“)用のデジタルカメラユニットである。しかし既に販売中止となっており、本システムのコンセプトである入手性に関しては疑問があるといえるのだが、他に適当なデバイスが見つけられなかったということ、ネットオークションを使えば現在でも比較的容易且つ千円以下程度の安価に入手できるということからこのデバイスを用いることとした。また、このデバイスはPHS用ということで電気的な仕様が公開されていないのだが、有志による解析が行われており仕様は明らかとなっている。(参考 AAFぱ研[xiv][9])

図 7-3 Treva

上記サイト内の CMOSカメラユニット「Treva」の解析にて公開されているTrevaの仕様を以下に

[xiii] http://www.kyocera.co.jp/
[xiv] http://www.paken.org/

示す。

- 撮像素子　　1/4 インチ 10 万画素 CMOSイメージセンサ OV6630[xv][10]
- 出力画像解像度　96(H)×72(V) pixel
- レンズ　　固定焦点(30cm～∞)
- 露出　　自動制御
- ホワイトバランス　自動調整
- 電源　　3.3V
- 画像フォーマット　16bitYUV シリアル出力

このように、解像度が低くカメラとしての性能は落ちるのだが、シリアル出力であるということで、マイコンとの親和性は良いといえる。残念ながら3.3V動作のデバイスであるために、5V動作であるH8へは直結は出来ない。よって、H8 への接続にはレベル変換を考慮する必要がある。

次に Treva の画像フォーマットの仕様を以下に示す。

- I/F部（図 7-4 参照 ）
  - Vcc　　電源 3.3V
  - GND　　電源 GND
  - CLK　　Input　クロック入力
  - DOUT　　Output　シリアルデータ出力
- 出力フォーマット

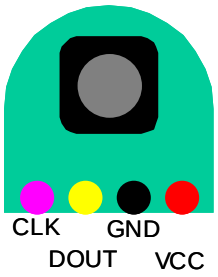


図 7-4 Treva I/F 部

表 7-1 にヘッダ部のデータ列を、また、表 7-2 に画像データの構成を示す

表 7-1 Treva 出力フォーマット（ヘッダ部）

AA 55	ダミーコード
FF D8	フレームスタートマーカ
1C	ヘッダバイト数（28バイト）
F0	モデルタイプ（この場合はカメラ）
F1	F1がバージョン（この場合は1）
00 60	1ラインの画素数（96 画素）
00 48	ライン数（78 ライン）
81	8:8 ビット階調、1:YUV タイプ
31	3:UYVY データ出力、1:MSB スタート
20	2:「4:2:2」タイプ、0:非圧縮データ
4B 43 23 38 42 50 2E 2E	メーカ情報
00 00 00 00 00 00 00 00	ユーザ情報
AA 55	データスタート認識マーク

[xv] OmniVision 社（ http://www.ovt.com/ ）製

表 7-2 Treva 画像データフォーマット

Y0	Y1	Y2	Y3	Y4	Y5	…
U0 / V0		U2 / V2		U4 / V4		…

Treva の画像サンプリングは 4:2:2 という方式であるので、Y(輝度)は各ピクセル毎、U と V（色差）は 2 ピクセル単位でデータが構成されている。これが

V0-Y0-U0-Y1-V2-Y2-U2-Y3-V4-Y4-U4-Y5-…

という順番でヘッダ部のデータの後に 96×72 ピクセル分連続して出力される。よって、ヘッダの検出を行い、その後のデータをピクセル数分取り込むことで画像データを得ることができる。

7.7 LCD、LED

システム本体での動作の確認、メッセージの掲示、またデバッグ等の多様な用途のために、単体での表示用としてキャラクタ型の LCD と LED を用意することにする。

キットと共に用いる拡張基板にはポート 4 の 7 ビット目と 6 ビット目に赤色と緑色の LED が実装されているが、これだけではあまりにも表現能力が低いし H8 のポートの未使用部分も多いので、8 ビットポートを全て利用する 8 本の LED（赤）を用意。また、それ以外に異常検知状態を知らせる LED、外部からのアクセスによるメッセージの有無を知らせる LED と 2 色の LED（赤、緑）を用意することとする。

メッセージ表示用に LCD が必要だが、拡張基板には 16 文字×2 行用の 4 ビットインターフェース用 LCD の接続が可能な端子が用意されているので、その端子に接続可能な LCD を用いることとする。

7.8 スイッチ

単体での設定を行うために最低限のスイッチがあった方が都合がよい。

機能としては以下にあげる 4 点である。

- メール通知 ON/OFF
- カメラ画像以上検知 ON/OFF
- 温度検知 ON/OFF
- 開発時のデバッグ用途

よって 4 ビット以上の DIP スイッチを用いることとする。拡張基板には既に 4 ビットの DIP スイッチが実装されているのでこれを流用することとする。

第8章　　システム構築

図 8-1 に、前章で検討した各部品を用いて構成したこのシステムのブロック図を示す。

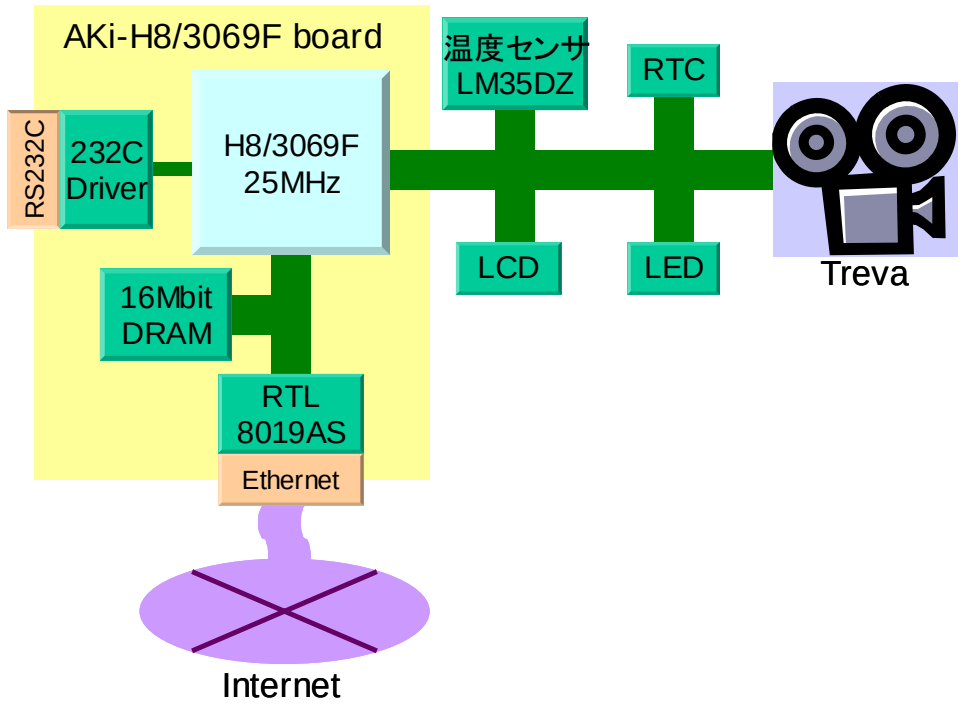


図 8-1　システムブロック図

図の中で黄色い枠で囲んだ部分は秋月のキットの部分である。キットの拡張端子からIOポートを取り出し、LCD、LED、RTC、Trevaを接続。また拡張端子のAD変換用入力へ温度センサの出力を接続という拡張を行った。実際の回路はこれに加えてAD変換器用リファレンス電圧、Treva用3.3V電源生成、TrevaとのI/Fの電圧変換回路(3.3V、5V間)が必要であるが、ブロック図上は割愛している。

このシステムブロック図を元に各部の設計を行った。次章に設計内容を示す。

第9章　　システムの設計 ― ハードウェア

この章ではシステムのハードウェア部分についての設計について示す。この設計回路は実際に組み立て・実装しているということで、抵抗やコンデンサ等の電子部品の定数に関しては設計計算上の最適なものではなく、手持ち或いは入手の容易な部品の中で最適なものを選んでいる。よって、多少の誤差やオーバースペック等の使用形態もある点を考慮願いたい。

実際の回路図を、AKI-H8/3069F LANボードを付録 [A]に、拡張基板部分の今回新たに設計した回路図を付録 [B]に添付した。

9.1　　マイコン部

マイコン周辺部は AKI-H8/3069F LAN ボードに実装されており、主には 16MbitDRAM、イーサネットコントローラ、RS232C ドライバと拡張コネクタ類が実装されている。拡張コネクタには全ての I/O ポートが接続されているので、拡張コネクタを介して各 I/O を接続する。

また、H8 自体はモード 5 で動作させるが、そのモード 5 の動作時のアドレスマップを 図 9-1 に示す。

外部に接続されるバスは全てH8/3069F 側にて 8 ビットモードで使うように設定してあり、その外部バスのエリア 1 には RTL8019AS が、エリア 2 には 16MbitDRAM(TC5117805CFT)がそれぞれ 8 ビット幅で接続されている。

拡張は全て IO ポートに接続するので、メモリマップ上にはこれ以外にはマッピングされない。

拡張が必要な部品は以下に示す 3 点であり、これらについての各部の設計に次項以降に示す。

- 温度センサ
- RTC
- カメラ(CMOS イメーセンサ)

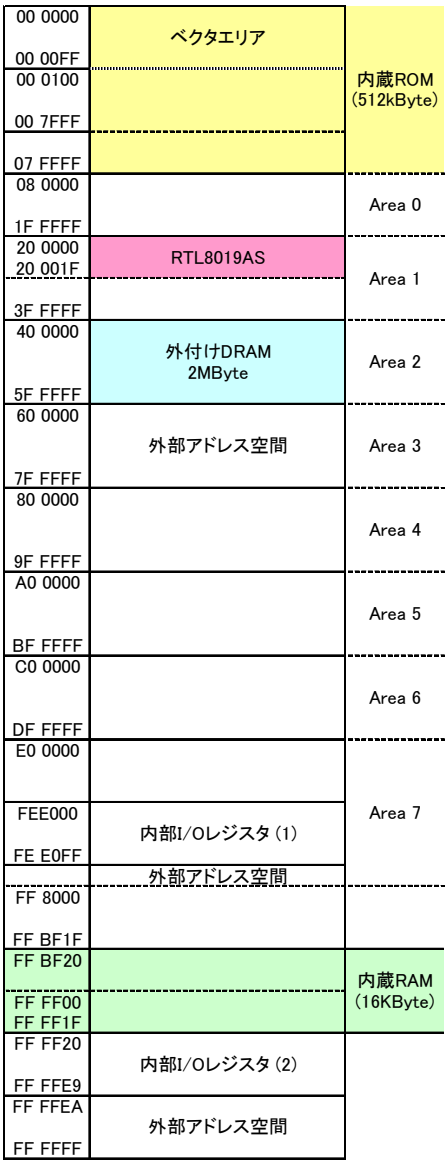


図 9-1　H8 AddressMap (mode5)

9.2　温度センサ

温度センサは、NationalSemiconductor[xvi]社 LM35 シリーズ [xvii]のプラスチックTO-92 パッケージタイプであるLM35DZ[7]を用いた。このデバイスは出力が摂氏温度に対しリニアに比例した電圧を出力する高精度のIC温度センサである。以下に簡単な仕様を示す。

- 摂氏温度に直接較正済
- 温度計数はリニアで+10.0mV/℃
- +25℃において 0.5℃の精度を保証
- -55℃～+150℃の温度範囲
- 4～30V の動作電源範囲
- 60μA 以下の電流ドレイン
- 低自己発熱、静止空気で 0.08℃
- ±1/4℃以下の非直線性
- 低出力インピーダンス

この仕様から、室温を測定するという目的としては十分な性能を有しているといえる。しかも、10mV/℃という温度係数は温度値を容易に計算できる係数であり、特に乗除算に負担のかかる非力なマイコンに向いているといえる。

次に 図 9-2 に温度センサ部の回路図を示す。

動作用電源電圧には H8 ボード上から供給される 5V の AVccを用いた。電源-GND 間にはノイズ防止のためのコンデンサを2種類(高周波用、低周波用)接続。またLM35DZのアナログ出力には変動を極力抑えるために抵抗とコンデンサを用いて安定化させた。この出力をH8のA/D変換器用入力 AN1 に接続した。

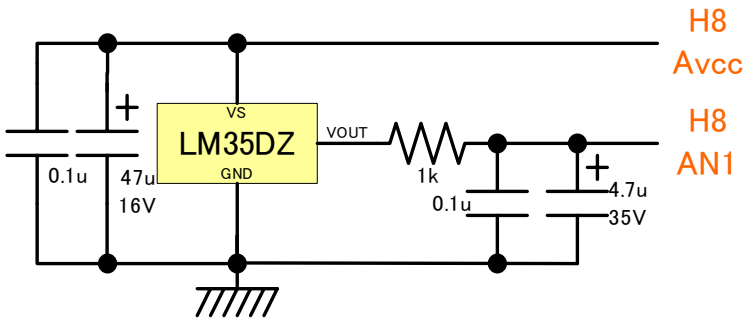


図 9-2 温度センサ部回路

LM35DZ 自体の動作温度範囲は-55℃～+150℃とかなり広いのだが、室温という限定された用途においては、ここまでダイナミックレンジを広く取る必要はなく、ダイナミックレンジを抑えることでノイズ等による誤差を少なくし、より精度を上げる方が望ましい。

そこで、室温ということより想定温度範囲を 2～50℃と限定し、この温度範囲を正確に測定するべ

[xvi] http://www.national.com/
[xvii] http://www.national.com/pf/LM/LM35.html

く回路を設計した。最低想定室温を 2℃としたのは、この IC センサで 2℃以下を測定する為にはマイナス電源が必要となるからである。2～50℃の温度範囲であれば、LM35DZ より出力される電圧は 2mV～0.5V となる。これを H8 の AD 変換の端子へ入力するわけだが、10 ビットの A/D 変換機能を有意義に使うためには、10 ビットフルレンジに極力近づける方が良い。そのアプローチには LM35DZ の出力を OP アンプ等で増幅させ、A/D 変換のレンジは H8 の電源と同じ 5V にするという方法と、A/D 変換のレンジそのものを小さくする方法がある。

ここでは、手持ちの部品の問題、H8 基板で準備される Vref 電源の 5V はデジタル電源と同じであるのでノイズが多いだろうという事、また AD 変換によって得られた値の計算のしやすさを考慮して、AD 変換用のリファレンス電源(Vref)を低く且つ計算しやすい値として生成する方法を選択した。
図 9-3 にA/D変換器用のリファレンス電源(Vref)生成部の回路図を示す。

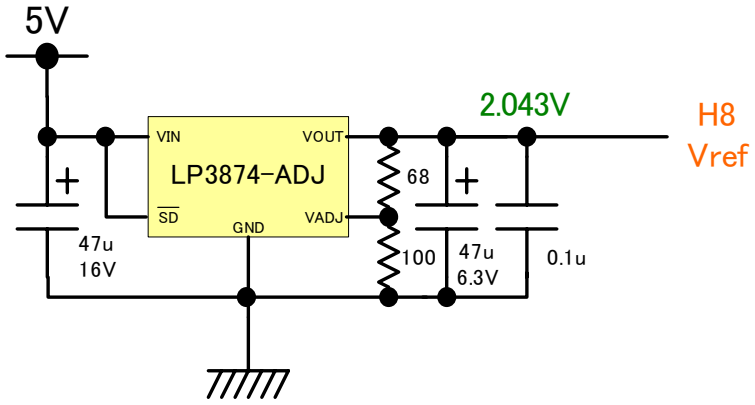


図 9-3 A/D 変換器用リファレンス電源生成部回路

具体的には、まずA/D変換用のリファレンス電源をデジタル電源と切り離す為に、デジタル電源の 5Vから三端子レギュレータを用いて安定した電圧を生成。この三端子レギュレータには出力電圧可変タイプの LP3874-ADJ[11](NationalSemiconductor[xvi]社)を用いた [xviii]。この可変タイプは、外付け抵抗によって所望の電圧を生成するのだが、ここはE系列 [xix]も考慮して 100Ωと 68Ωを選択。この値より算出される出力電圧は 2.043Vとなる。よって、このA/D変換器の 1LSBはリファレンス電圧 2.043Vの$1/1024 \approx 2mV$ となり、非常に区切りのよい数字に出来、後々の計算が容易になる。

ここで Vref を小さくしたことによる誤差の拡大という問題があるが、1LSB である 2mV は温度に換算すると 0.2℃であり、これであれば、室温を測るという用途においては1℃以下の数 LSB の誤差はまず問題にはならないと判断した。

AD 変換によって得られる値は 10 ビットであり、前述したように 1LSB は 0.2℃であるので、実際の温度値を得る為には、AD 変換値÷5 の計算をする必要がある。ここで、少数点 1 位まで表示することを考えるが、ワンチップマイコンで浮動小数点を扱うのは負担が大きいので、0.1 を1として実際の値の 10 倍を整数値で扱うすることとする。よって、得られた値を 2 倍すれば、10 倍の値が得られることとなり、1 ビットの左シフト演算で済むので乗除算に負担の大きいマイコンに取っては非常に効率

[xviii] 手持ち部品の関係でこれを用いた。定格 0.8A 出力なのでかなりのオーバースペックである。
[xix] 誤差を考え、隣の値とオーバーラップすることなく合理的な数値の並びになるように、等比数列で決められた値。JIS により E3、E6、E12、E24 のように系列が定められている。

が良くなる。

また、アナログ値の測定には誤差は付き物である。特にノイズの多く発生するデジタル回路との混合であればなおさらである。そのノイズによる誤差へのソフトウェア側からの対策であるが、A/D変換に要する時間が数μsと他の処理を考えても特別に大きな負荷になるような時間ではないので複数回測定による誤差の縮小と測定エラーの除去を図ることとした。具体的には10回連続して測定し、その10回の測定結果の最大値と最小値を除いた8個の値の平均をとるものとする。8個の平均は合計値を8で割る必要があるが、これも3ビット分右シフトするというマイコンにとって非常に軽い負荷で値を得ることができる。

実際の運用においては、リファレンス電源(Vref)の電圧値を決める各部品の誤差等使用している部品の誤差があるので、別の正確な温度計にて得た値からあらかじめ補正値を算出しておき、その値を測定結果に一律に補正するという形でより正確な値となるようにした。

9.3 リアルタイムクロック(RTC)

RTCはエプソントヨコム社のRTC-8564NB[8]を使用し、使いやすいように8ピンDIPパッケージになっている秋月電子のRTCモジュールを用いる。

このRTCの仕様を以下に示す。

- 32.768kHz水晶振動子内蔵
- I^2C-Busインターフェース
- 低電圧計時保持(1.0～5.5V)
- I^2C -Bus Slave Address Read $A3_H$/Write $A2_H$

I^2Cインターフェースとは、クロックとデータの2線のインターフェース形式であるが、H8にはI^2Cインターフェースは実装されていないので、2本のI/Oポートを用いてI^2C規格の動作を行うことでインターフェースを行うこととする。

また、今回用いたRTCそのものには割り込み機能等様々な機能があるが、本システムではレジスタへの現在日時書き込み(設定)機能と、レジスタからの現在日時読み出し機能のみ使用する。

RTC部の回路を 図 9-4 に示す。

実際のRTCであるRTC-8564NBはSON-22pinパッケージであるが、実際に使用するピンは 図 9-4 に示した5本の信号線と2本の電源である。このICを用いて 図 9-5 のように8pinDIPパッケージにモジュール化してあるRTCモジュールを今回は用いた。このRTCモジュールは 図 9-4 内の枠で示した様にRTCに加えて、周辺の3つの抵抗とLEDが既に実装されている部品である。よってこのモジュールからの接続は電源とI^2Cインターフェースの信号だけという非常にシンプルなものとなる。

今回はI^2CインターフェースのクロックであるSCLをH8のP61ポートに、データであるSDAをH8のP60ポートに接続した。よって、ポート6の1ビット目、0ビット目を用いてI^2Cインターフェースの動作を実現すればRTCへの読み書きが可能となる。

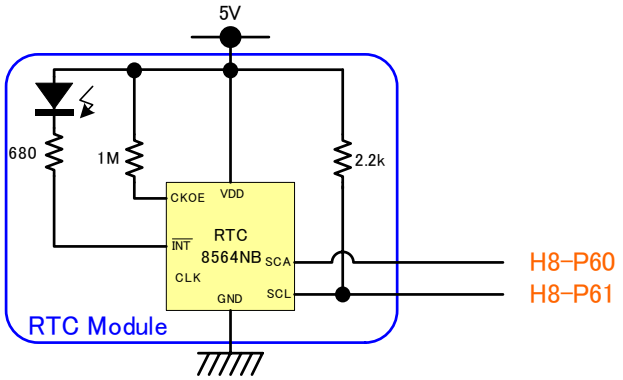


図 9-4 RTC 部回路

図 9-5 RTC モジュール

9.4 カメラ (Treva)

図 9-6 にカメラ部の回路図を示す。

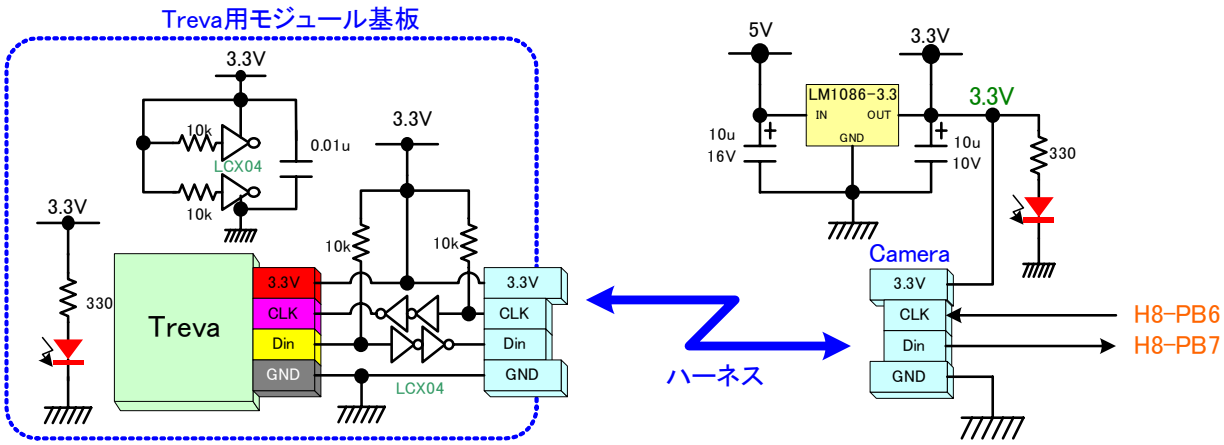


図 9-6 カメラ部回路

この図 9-6 内で枠で囲んだ部分は、Treva用モジュール基板を示し、本体とは別の基板とした。これは、水平に設置されるH8 基板に対してカメラアングルの自由度を得る為である。実際にこのモジュール基板とH8 基板とはL字アングルを用いて直角になるように設置している。この基板間の電気的な接続には、2 本の信号と 2 本の電源を接続する為 4 ピンのハーネスを用いた。

H8 側との接続にはポートBを用い、そのポートBの 6 ビット目にカメラ用CLK出力を、7ビット目にはカメラからのデータ入力を接続した。また、Trevaの電源電圧は 3.3Vであるため、H8 基板上にて三端子レギュレータ(LM1086-3.3[12][xx])を用いて 3.3Vを生成。その電圧をハーネスを通して供給した。

次に信号の電圧レベルであるがTrevaの信号は電源電圧と同じ 3.3V、H8 は 5Vである為、お互いの接続にはレベル変換が必要となる。そこで、東芝セミコンダクター社製汎用低電圧CMOSロジック

[xx] 手持ち部品の関係でこの型番を用いたが、定格 1.5A 出力なのでかなりのオーバースペックである。

であるLCXシリーズのインバータ TC74LCX04F[xxi][13]を 2 段接続することでレベル変換のバッファとして用いた。このLCXシリーズの特徴は電源電圧が 3.3Vで動作し、入力が 5Vトレラントということである。よってカメラへのCLK信号送出に関しては、H8 からの 5V出力を 3.3V出力に変換してTreva へ入力することができる。逆にTrevaからのデータ出力は 3.3Vであり、これもLCX04 を通してバッファリングしている。しかしこの使い方ではただのバッファであり 5Vには変換されない。これは、LCXの V_{OH}が最低でも 3.1V以上ということと、H8 のV_{IH}が最低 2.0Vであるということから、H8 への入力に関しては電圧レベルを無変換でも十分規格を満足しているためである。

また Treva や H8 からの入力が無いことも想定し、カメラ基板側 LCX04 の入力と、TC74LCX04F 内の使用していない 2 つのインバータの入力には、貫通電流防止の為のプルアップ抵抗を接続している。

9.5 LCD、LED

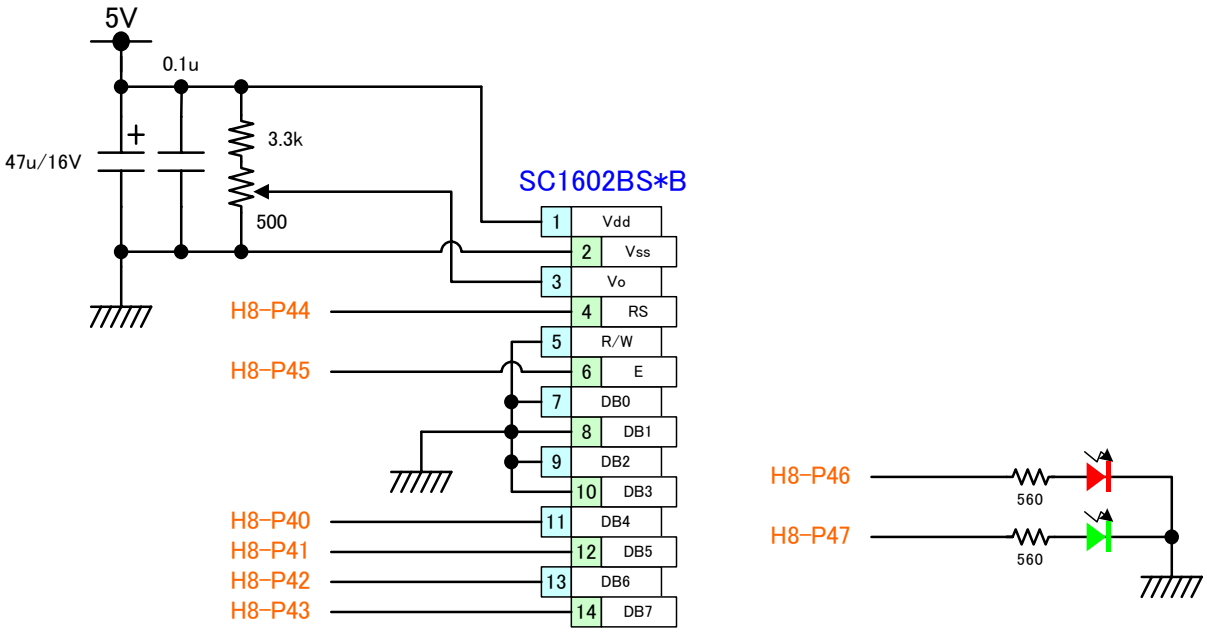


図 9-7 LCD、LED 部回路（拡張基板分）

拡張基板に実装されているLCDとLEDの回路図を 図 9-7 に示す。

LCDには、拡張基板に用意されている端子にそのまま接続できる Sunlike Display Tech.[xxii] 社製のSTN型 16 文字×2 行キャラクタLCDモジュール(SC1602BS*B-SO-GB-K)[14]を用いた。このモジュールは 4 ビット、8 ビット両方のインターフェースを備えているが、今回用いた拡張基板が 4 ビット用に設計されているので、そのまま 4 ビットインターフェースにて用いた。

LED は赤と緑の 2 色とし、それぞれ 560Ω の電流制限抵抗を介してポートと接続されるように実装した。

[xxi] http://www.semicon.toshiba.co.jp/openb2b/websearch/productDetails.jsp?partKey=TC74LCX04F
[xxii] http://www.sunlikedisplay.com/

次に 図 9-8 に拡張したLED部の回路図を示す。

この 10 個のLEDのうち 8 個の赤色LEDはデバックを主目的とする汎用の用途とし、ポートAの 8 ビットに接続した。LEDには実装効率を考え、シチズン電子 [xxiii]社製のチップ型である CL-170UR[15]を使用した。このLED用の電流制限抵抗は高輝度型ということと最高の明るさは必要ないということを考慮し、順電流を数mAと設定、そこから計算して妥当な 1.1kΩとした。

残りの 2 個の LED は、緑色の方をメッセージ用に、赤色の方を異常検知状態を示すのにそれぞれ用いる。その目的から考えて周囲に知らせる必要があるので、チップ型でなく高輝度の 3Φの丸型タイプを用いた(手持ち部品の為型番不明)。このLED の電流制限抵抗は実際に光らせてみて妥当だと思われる値であった 220Ωを用いた。接続に関しては、点灯・消滅だけでなく点滅状態の使用も想定していることから、CPU のソフトウェアに余分な負担をかけぬようタイマー機能を用いることを想定し、タイマー出力も兼ねているポート B の 0 ビット目に赤色 LED、2 ビット目に緑色 LED を実装した。

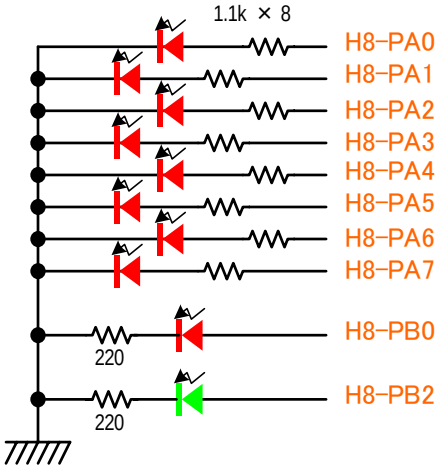


図 9-8 LED 回路（追加分）

9.6 スイッチ

拡張基板に実装されているDIPスイッチ部の回路図を 図 9-9 に示す。

この回路図に示すように、この DIP スイッチは ON にすると Low を示す。回路図だけでは OFF にすると浮いてしまい不安定になるように見えるが、ポート 5 の 4 ビットは H8 内部での設定により H8 に内蔵されているプルアップ抵抗を使用することができる。この機能を使用することにより、DIP スイッチが OFF 時には内蔵プルアップ抵抗により High を示すこととなり、ON/OFF を示す 2 値スイッチとして問題なく機能することとなる。

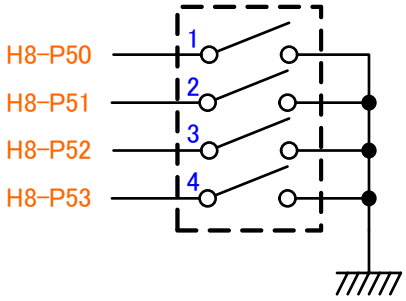


図 9-9 DIP スイッチ部回路

[xxiii] http://www.c-e.co.jp/

第10章　　システムの設計 ― ソフトウェア

この章ではシステムにおけるソフトウェアの設計・実装について説明する。

10.1　OS (MES 1.0b5 [16])

メインタスクとは別にネットワークから HTTP サーバへの不定期なアクセスを前提とすると、マルチタスクの OS を準備し、その上でメインのタスクと HTTP サーバが動いているという形態の方がシステムを構築しやすい。しかし、TCP/IP のプロトコルスタックをサポートしたマルチタスク OS を一から作成し、それに対応する開発環境を整備するというのはそれだけで一つの研究論文の題材になってしまう位の大きな作業となる為、今回は開発環境が整備されている既存の H8 用の OS を使用することとした。

既存のH8 用のOSとしてはいくつか存在するが、UDP、TCPのネットワークプロトコルをサポート。マルチタスクをサポート。使用に関しての費用が無料が安価。インターネット上若しくは書籍にて情報が入りやすい。実績がある。安定している。等々の条件を考慮してWEB公開されているフリーの三岩氏作成の MES 1.0b5 を用いることとした。この MESVersion1.0 は次にあげる特徴をもつ。

- マルチタスクをサポート。ラウンドロビン方式で複数のプログラムを実行可能
- シリアルポートドライバの搭載
- キャラクタ LCD ドライバの搭載
- パソコンからマイコンのファイルシステムへファイルをダウンロード可能
- 標準入出力の搭載
- システムライブラリにより、各種 printf/scanf 関数が使用可能(浮動少数は除く)
- メモリ管理機能搭載により、malloc/free 関数により動的にメモリ確保が可能
- NE2000 互換 10BaseT 有線 LAN ネットワークデバイスをサポート
- ARP プロトコルのサポート
- IP プロトコルのサポート
- ICMP プロトコル(要求/応答のみ)のサポート
- UDP プロトコルのサポート
- TCP プロトコルのサポート
- WEB ブラウザのサポート
- ダイアグラムソケット API 提供により、UDP クライアント/サーバに対応
- ストリームソケット API 提供により、TCP クライアント/サーバに対応
- IP フラグメントに対応、NIC の内蔵メモリ量限界までの IP パケットをサポート
- DHCP[xxiv]プロトコルのサポート
- TFTP サーバーから RAM ディスク上へのプログラムダウンロードが可能
- コマンドインタープリタに TFTP クライアント機能を追加
- WEB サーバでの HTML 文書/画像ファイル/CGI プログラムへの対応

[xxiv] Dynamic Host Configuration Protocol の略。動的にホストの設定を行うための UDP 上で動作するインターネットアプリケーションプロトコル。RFC-2131[17]と RFC-2132[18]で規定される。

10.2　システムフロー

プログラムの各部を検討する前に全体のフローを検討する。

このシステムのフロー図を 図 10-1 に示す。

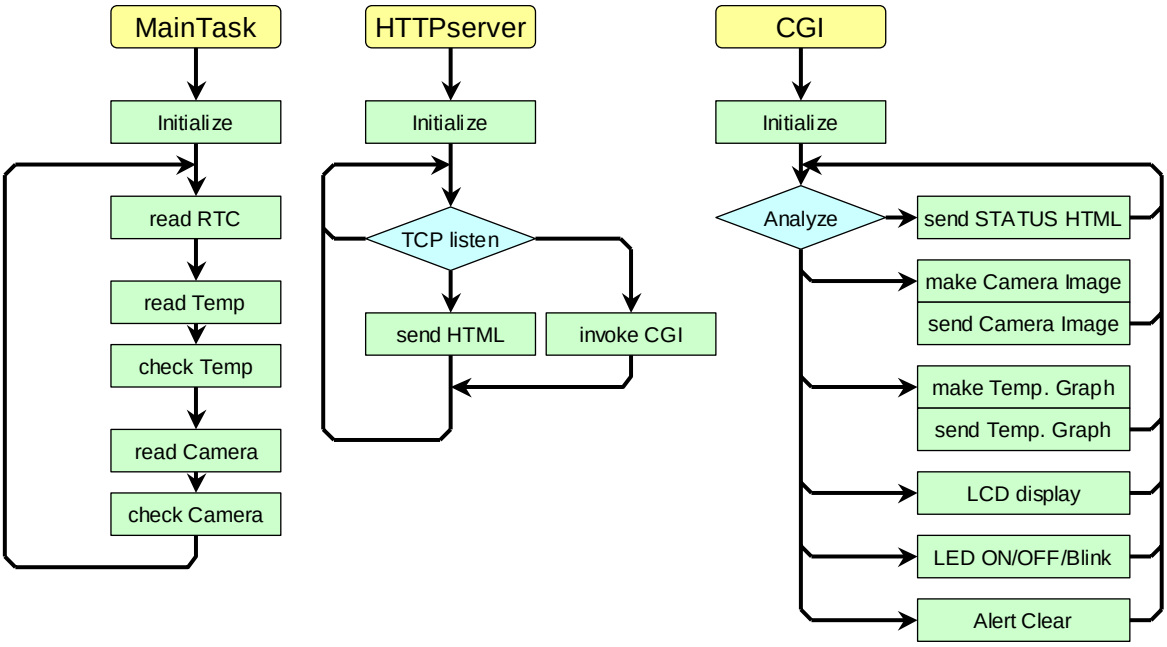


図 10-1 システムフロー

処理としては温度監視、画像監視と異常時の通知を主に行うメインタスクと、ネットワークからの HTTP アクセスに対する WEB サーバのタスクと大きく 2 つに分かれ、OS の上でマルチタスクとして両方同時に実行されている形態をなす。また HTTP サーバへのアクセスによる状態表示、制御に関しては全て外部 CGI プログラムを起動することで実現する。CGI プログラムは CGI 起動時の引数の内容により各種ステータスの表示、カメラデータより画像データの生成・表示、温度データより温度グラフの生成・表示、LCD への表示制御、LED の表示制御、Alert 状態のクリアの処理を行う。

次項以降にそれぞれの処理ブロック単位での説明を示す。

10.3　メインタスク

10.3.1　Initialize（イニシャライズ）

メインタスクのイニシャライズルーチンでは、以下の処理を行う。

- 共有メモリの確保、初期化

OS 上でメインタスクと並行して動作する HTTP サーバ、また HTTP サーバから呼び出される CGI、これら複数のタスクで必要とするデータには共有メモリを用いる。共有メモリの確保は OS の機能として用意されているので、それに基づき確保と初期化を行う。

排他制御を考え、共有メモリへの書き込みはメインタスクのみとし、他のタスクからは読み出しのみを行う。

- 各変数の確保、初期化

共有メモリ以外のメインタスクでのみ用いるローカル変数の確保と初期化を行う。

- H8 の各ポートの設定・初期化

H8/3069F の各ポートは様々な使い方ができる為、使用にあたっては設定と初期化が必要である。このシステムでは以下のように設定した。

  - ポート 4 （LED、LCD 接続）　全て出力、プルアップ抵抗 OFF
  - ポート 5 （DIP スイッチ）　全て入力、プルアップ抵抗 ON
  - ポート 6 （I^2C）　P60 入力、P61 出力
  - ポート A （LED）　全て出力
  - ポート B （Treva、LED）　PB7 入力、PB6～PB0 出力

- AD 変換器の初期化

AN1、シングルモードに設定

- RTC の初期化

リピートモード、アラーム無効、INT 有効イネーブル(LED 点灯のみ)、クロック出力無効

- H8 内蔵タイマの初期化

CH0/CH1、CH2/CH3 のカスケード接続、周期 1 秒、Duty50%設定

- ネットワークの初期化

OS によってサポートされる DHCP による IP 設定

- LCD の初期化

LCD のシステムコールによる open と表示クリア

- NTP による時刻の取得と RTC への時刻設定

NTP により現時刻を取得し、RTC へ設定。その時刻を起動時刻として共有変数に格納。

- 起動メッセージの表示

LED 点滅による起動メッセージを行う。

共有変数として確保した変数 init に、これら一連の初期化処理に突入した時点で1を、初期化処理が終了した時点でに 2 を格納する。この変数の値を他のタスクから確認することで、メインタスクが起動しているのかどうかの判断を行う。

10.3.2　read RTC（RTC 読み込み、再設定）

メインタスクの無限ループの最初では RTC から現在時刻を読む処理を行う。この得られた時刻によって次の温度測定の処理を行うかスキップするか判断を行う。温度測定の設定は 1 分、3 分、5 分、10 分と 4 種類設定可能とし、前回の測定時刻より設定時間経過していれば次に温度測定を行い、経過していなければ次の温度測定の処理はスキップする。当然ながら起動後最初の処理時は温度測定を行う。

また RTC の誤差防止のために、1時間毎(正時)に NTP にて補正を行い RTC に再設定を行う。

10.3.3　read Temp.（温度センサ読み込み）

RTC 読み出しの結果、温度測定を行う場合にこの処理を行う。

温度測定は 9.2温度センサ にて説明したようにAD変換機能を用いて温度センサからの値を得る。

AD変換による温度値取得のフローを 図 10-2 に示す。このように温度測定はノイズ等の誤差を考慮し 10 回連続して測定を行い、その最大値と最小値を省いた 8 個の値の平均を求めることでより正確な値とする。また、これら 8 個の平均を求める際にはシフト演算を用いることでCPUにかかる負荷の低減を図る。

AD 変換によって得られる値は 10 ビットであり、2.043V を Vref とする場合の1LSB は約 2mV。即ち 0.2℃を表す。また AD 変換の結果をシフト演算により2倍した値で使用することで、1LSB は 0.1℃を表し、結果 10 倍された値を格納していることとなる。よってこのシステムにおいては温度値の扱いは 0.1℃単位で行うこととする。

また、得られた値は 10 分単位で読む際に 24 時間分の推移がわかるように 144 ポイント分格納できる変数領域を用意し、そこに時系列で、またあふれ時にはオーバーラップする形で格納することとする。HTTP サーバからこの推移値を利用することで温度推移グラフを生成し表示する機能を有することとする。

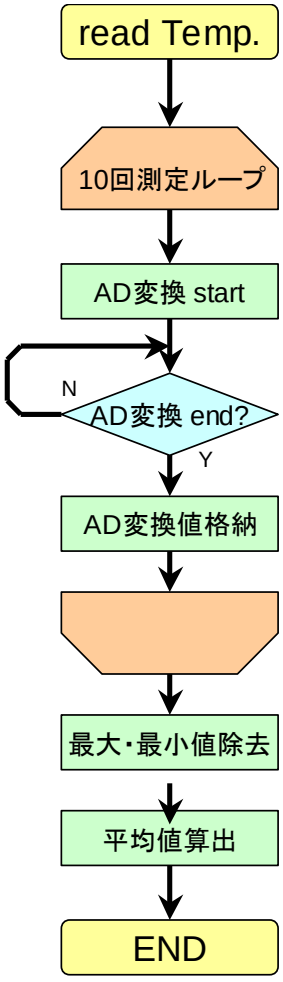


図 10-2 AD 変換フロー

10.3.4　check Temp.（温度異常検出）

温度センサより得られた現在の温度値と前回測定した際の温度値との比較を行い、その差が閾値以上ある場合に異常と判断を行う。測定には最大で 10 分の間隔があるが、10 分で変化する温度値で自然の温度現象ではまず無いであろう 5℃を閾値として用いた。

異常時にはまずAlert用赤色LEDを遅い間隔にて点滅をさせ、図 10-3 に示す様な異常検出時の温度を示した異常の旨の内容にて、SMTPによるメール通知をあらかじめ設定されている送り先に対し行う。但し既に異常時となっている場合には再度のメール通知は行わない。

赤色 LED の点滅には H8 内蔵のタイマー機能を用いることで CPU の負荷の低減を図る。またメール通知には SMTP を用いる。

```
From: H8 <m03603@std.int-univ.com>
To: test@localhost
Date: Fri, 14 Jan 2006 22:56:45 +0900
Subject: AlertMail

Alert : Temperature
  38.4 degree Celsius
http://192.168.0.13/cgi
```

図 10-3 温度異常時の Alert メール内容例

10.3.5 read Camera（カメラデータ読み込み）

7.6カメラ（BT656 カメラ、Treva）で仕様を、9.4カメラ（Treva）で回路を示したが、このTrevaから画像を読み出す処理を行う。

Trevaからの読み込みフローを図 10-4 に示す。TrevaはH8 側からクロックを与えることでCCDの値を読み出し出力することができる。しかし 1 ビット単位の読み込みでは 1 バイトの区別がつかない。加えて画像データはフレーム単位で読み出す等の高度な処理は出来ないので、フレームの最初を検出する必要がある。よってまずは 1 ビット単位で読み出し、シフト演算を施しながらフレームのスタートバイトになる部分を探す。これが見つかれば、フレームの最初とバイトの区切りの両方が検出できたことになる。フレームの最初が検出できたら、余分なヘッダ分空読みを行い、その後に続く画像データをピクセル数分読み出す。

Treva の画像データは YUV 形式で 4:2:2 のサンプリングであるので、Y はピクセル数分、U と V はピクセル数の半分が読み出せることとなる。

画像をHTTPサーバ経由で表示させる際にはRGB形式である必要があるので 数式 10-1 に示す式にてYUV値をRGB値へ色変換を行い、サンプリング 4:4:4 のRGBデータとメモリへの格納を行う。但し、書き込みの最中に読み出しが起こること

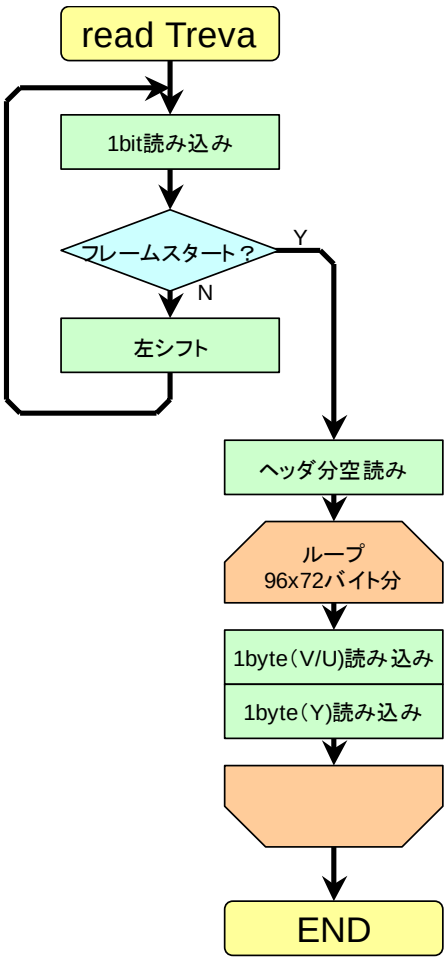


図 10-4 Treva 読み込みフロー

もありえるので、その時の排他性を考え画像格納用バッファはオーバーラップが起こらない様に 3 画面分用意してリングバッファ的な使用を行う。加えて現在の最新画像のindexを格納する変数も用意し、1画像を格納終了するたびにindexを更新する。

数式 10-1 YUV→RGB 変換式

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.164 & 0 & 1.596 \\ 1.164 & -0.391 & -0.813 \\ 1.164 & 2.018 & 0 \end{bmatrix} \begin{bmatrix} Y \\ V \\ U \end{bmatrix}$$

10.3.6 check Camera（画像異常検出）

得られた画像の変化から異常を検出する。

横 96 ピクセル、縦 72 ピクセルで構成されるTreva画像データと異常検出に注目する中央部分の 8×8 ピクセルの並びを 図 10-5 に示す。

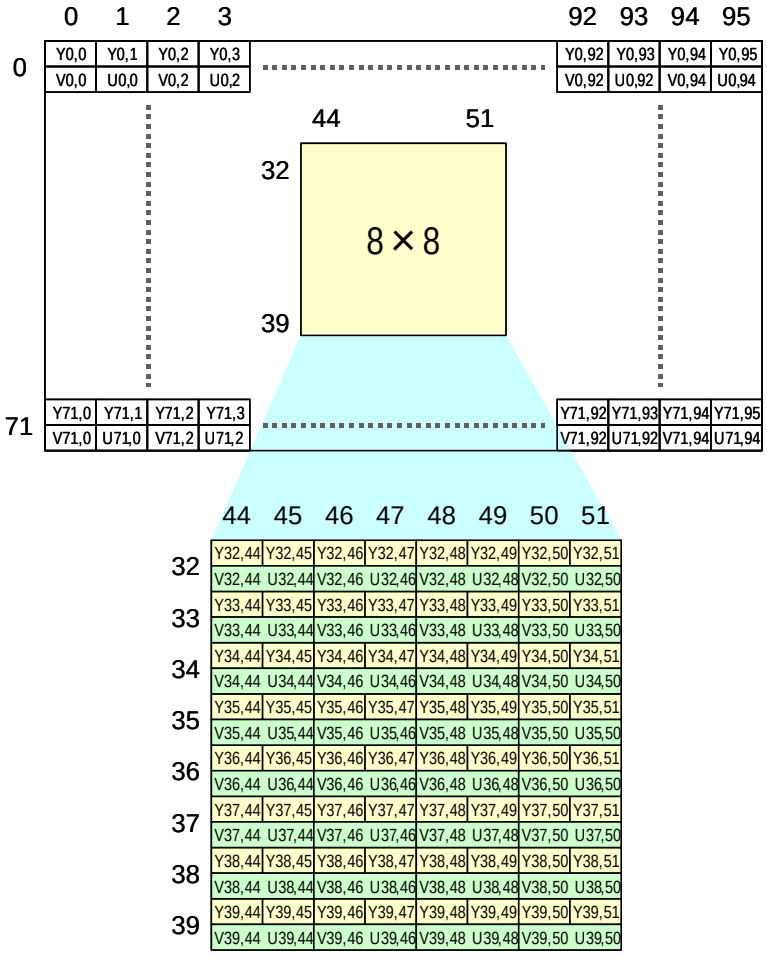


図 10-5 Treva 画像データ

画像の異常検出方法については、留守宅の撮影を想定しているので通常状態においては画像の変化は無いはずである。よって前回の画像データと今回の画像データの変化により異常を検出するものとする。しかし 96×72 と低解像度のカメラとはいえ、この全てのピクセルについて変化を検出するには H8 には負荷が重過ぎるので、図に示したように画像中央部のピクセル座標で示す(44,32)～(51,39)の 8×8 ピクセル分にのみ注目し、この 64 ピクセル分のデータの変化により異常を検出することとした。

データの変化であるが、カメラの精度や誤差の問題、環境光の変化等による画像データの変化は常に生じている。その変化を異常と検知しないように一定のロバスト性をもたせる必要がある。そこで、実際にカメラからの画像データを測定しその結果からアルゴリズムを検討することとした。

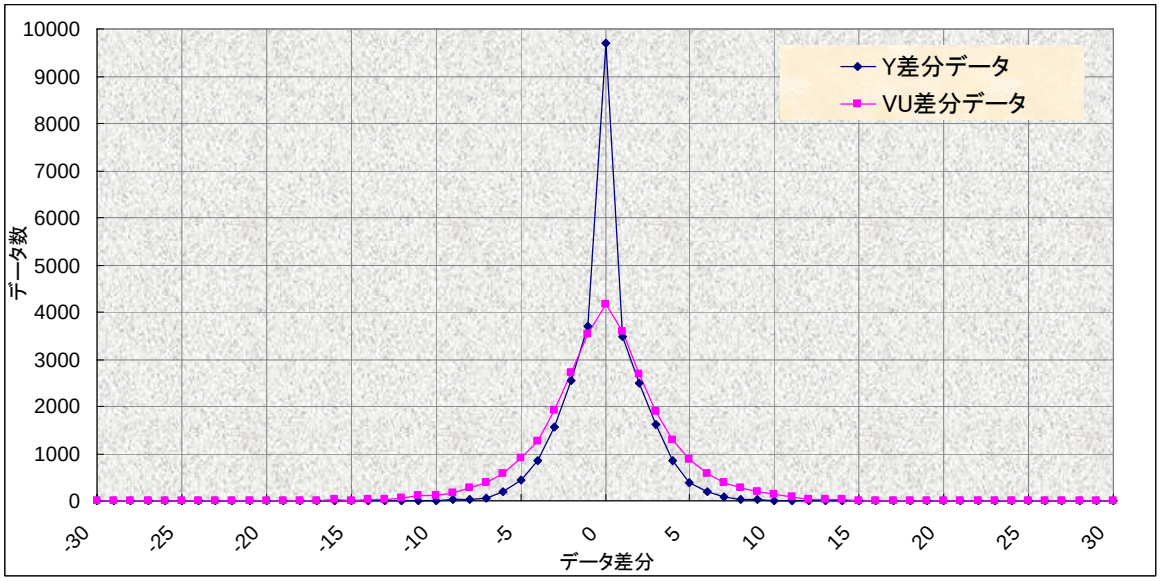


図 10-6 カメラ画像差分データ分布図

444点の連続する画像データを採取し前画像との差分データを 1 画像に付き 64 ポイント(8×8 画素分)をYとVUそれぞれに計算した値のうち、データの差分を横軸に、その差分値となったポイント数を縦軸にとったグラフを 図 10-6 に示した。また、このデータの統計結果を 表 10-1 に示した。

表 10-1 カメラ画像差分データ統計表

	総数	平均	最小	最大	標準偏差	2σ	3σ
Y差分データ	28352	0.03	-28	110	2.82	5.64	8.46
VU差分データ	28352	0.00	-33	36	3.78	7.55	11.33

図に示したように Y も VU も平均をほぼ 0 とする正規分布の形となった。また、Y よりも VU の方がぶれが大きい結果となった。

この結果からまず差分を計算し、その値からぶれと思われるデータを除去、残った値の大きさから異常を検出するという方法を考案した。

まずぶれの除去方法だが、先にあげた統計データより、2σ と 3σ の間の 8 に注目し、Y、VU それ

ぞれの差分値より 8 未満の値を除去することにした。8 という値に注目したのは 2 のべき乗であるという事にも依存する。2 のべき乗であればシフト演算にて簡単に除去ができるからである。また VU の方が Y よりもぶれは大きいのであるがその倍の 16 を用いるまでの違いは無いので同じく 8 を用いることとした。

よってアルゴリズムは、まず右 3 ビット算術シフトにより差分データを 8 で除算。その商の絶対値を 64 ピクセル分加算する。その値とあらかじめ決めた閾値を比較し、閾値を超えていれば異常と判断する。という方法をカメラで得られた画像毎に Y と VU 別々に行うこととする。

閾値は全てのピクセルが閾値を超えていた場合として 64 と設定した。

このアルゴリズムと 64 という閾値にてテスト運用を行ってみた結果、誤動作にもいたらず、且つわずかにカメラを動かしただけの異常時ですら 64 よりはるかに大きな差分値が算出されたので、問題は無いと判断した。

異常時にはまずAlert用赤色LEDを早く点滅をさせ、図 10-7 に示す様な異常画像検出の旨の内容でSMTPによりメール通知をあらかじめ設定されている送り先に対し行う。但し既に異常時となっている場合には再度のメール通知は行わない。

```
From: H8 <m03603@std.int-univ.com>
To: test@localhost
Date: Fri, 14 Jan 2006 22:26:29 +0900
Subject: AlertMail

Alert : Image
http://192.168.0.13/cgi
```

図 10-7 画像異常時の Alert メール内容例

10.3.7 send Mail（異常通知）

このシステムの最大の特徴であるのはメールによる自動通知である。

温度や画像の異常検知時にはメールにてその旨を通知をする。

セキュリティも考慮し、本システムでは送り元アカウント、送り先メールアドレスはシステムに既に設定され稼動状態では変更不可能とする。

メールの送信には標準的な SMTP(Simple Mail Transfer Protocol)というプロトコルを用いる。システム単体では配送できないので、システムの接続されているネットワーク上にてアクセス可能な SMTP サーバと接続用アカウントがあることを前提とする。

一般家庭で用いることを想定しているこのシステムの SMTP サーバとしてはプロバイダの用意する SMTP サーバを用いるというのが普通であろう。しかし、最近の SMTP サーバはセキュリティの為に SMTPS(SMTP over SSL)等の高度な認証システムが導入されているものが多く、いまや何の制限も

無い SMTP サーバは殆ど無いという状況である。そこで、SMTP サーバ利用における様々な認証の中でこのシステムで使用できる一番現実的なものが POP before SMTP であろう。これは SMTP アクセスの前に POP(Post Office Protocol)アクセスを行うことで SMTP サーバが一定時間使用できるようになるという認証システムである。このために SMTP 以外にも POP3(POP ver.3)というメール受信プロトコルを実装する必要があるが、POP3 も SMTP 同様に非常にシンプルなプロトコルなので問題は無い。

実際の運用に関しては、無料でアカウントを取得でき、そのアカウントを用いてPOP before SMTPでSMTPサーバを利用できる Yahoo[xxv]のSMTPサーバを利用することで実現した。

10.4 HTTP サーバ

HTTPサーバは新たに作成せずにOS（MES 1.0b5）用としてOS開発者である三岩氏作成で公開されているものを用いた(http.c [19])。

この HTTP サーバはシンプルな機能であるが HTML 表示だけでなく CGI による外部プログラムの呼び出しにも対応しており今回の用途では十分である。

10.5 CGI プログラム

HTTP サーバではスタティックな HTML ファイルを準備することはせずに、ステータスの表示、コマンド、画像の表示すべてに一つの CGI プログラムで対応する。何を表示するかは CGI を呼び出すときの引数によって判断するものとする。

実際のCGIによるWEB画面とその説明を 図 10-8 示す。この図に示すように、上部にはステータス表示部、そして下部にコマンド制御部を配置した。細長い画面にしてあるのは、携帯電話からのアクセスも想定し、見やすいように配慮したためである。

[xxv] http://www.yahoo.co.jp/

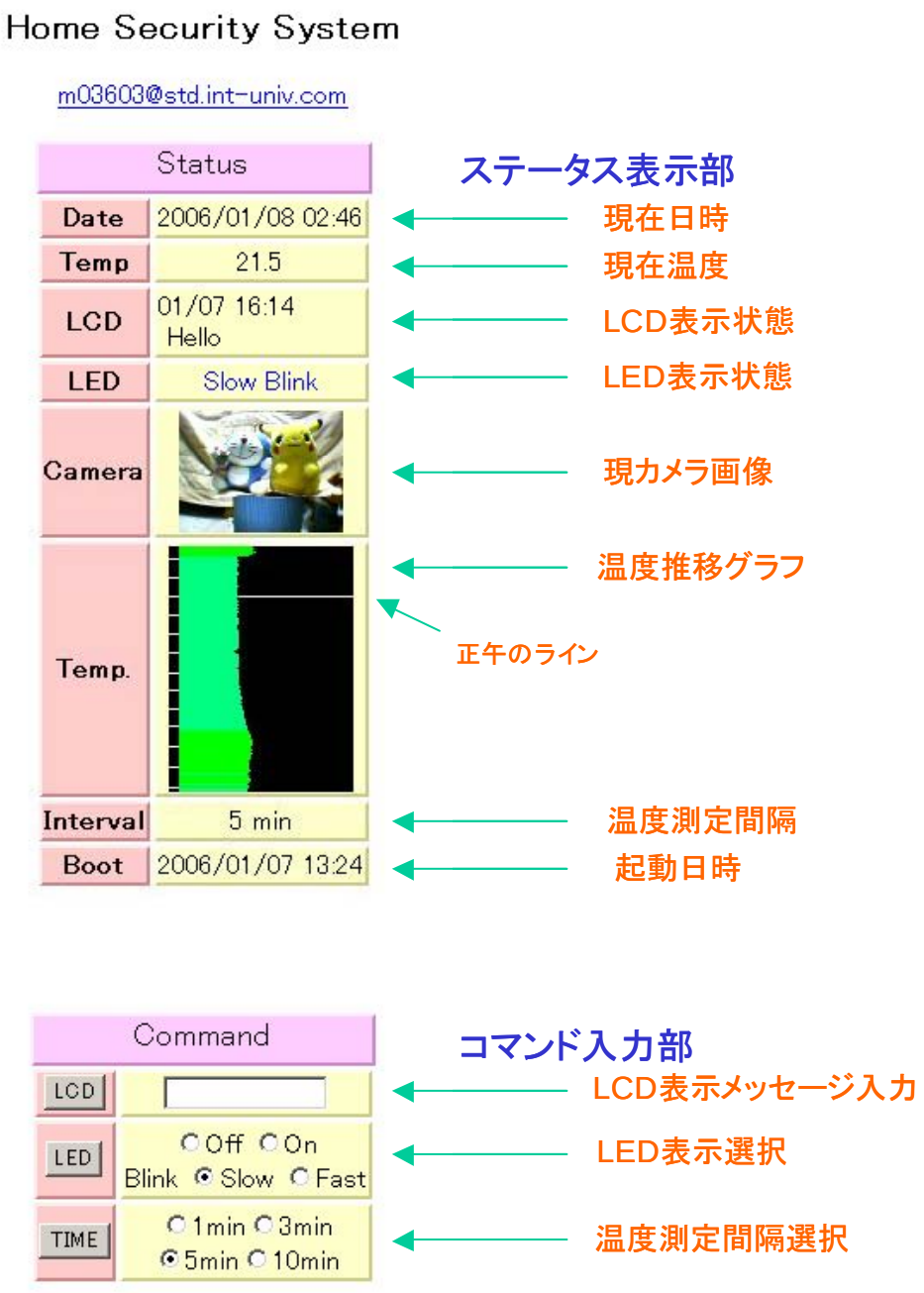


図 10-8 CGI による WEB 画面

異常検知時にはこのCGI画面はステータス表示の上の部分に異常が表示される。画像異常時は図 10-9、温度異常時は 図 10-10 に異常時の表示の例を示した。この様に異常時には異常状態をクリアできるように、『Alert Clear』ボタンが表示され、これを押すことで異常状態を解除し通常状態へと戻すことが出来る。

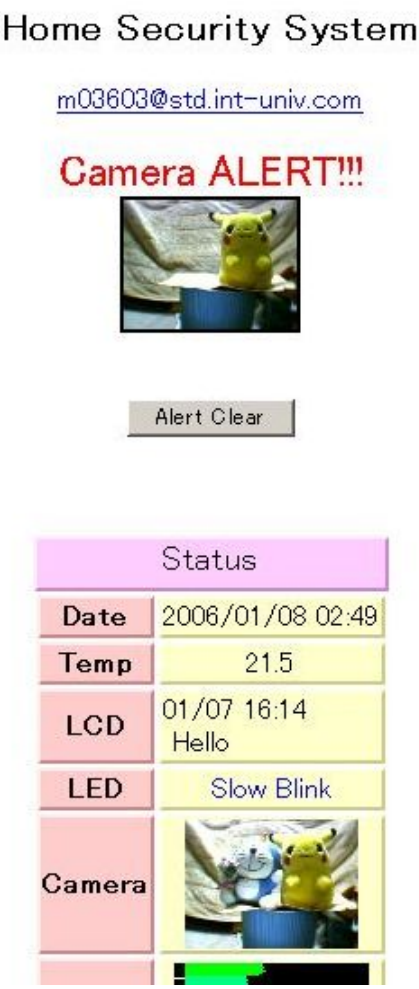


図 10-9 CGI による WEB 画面(画像異常時)

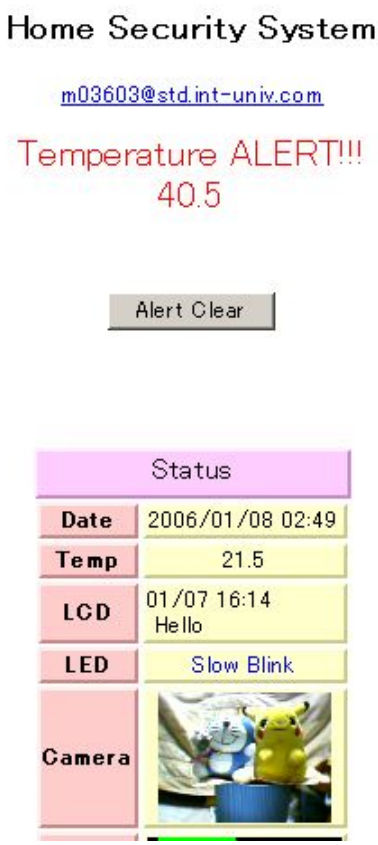


図 10-10 CGI による WEB 画面（温度異常時）

次にこの図に示したように CGI によって表示すべき項目を列挙する。

- Alert 表示
- 現在の日時
- 現在の温度
- LCD 表示状態
- メッセージ用 LED 表示状態
- 現在のカメラ画像
- 温度推移グラフ
- 温度測定間隔
- 起動日時

また、CGI によって設定できるコマンド項目を次に列挙する。

- Alert 解除コマンド
- LCD 表示コマンド
- LED 制御コマンド
- 温度測定時間設定コマンド

以下にこれら各項目について説明する。

10.5.1　Alert 表示

通常状態時では表示の必要が無いが、異常時には Alert 状態である旨を表示する。

温度異常時であれば Alert 表示と共に異常時の温度値を表示。画像異常時であれば Alert 表示と共に異常時の画像を表示する。異常画像は画像のバッファとは別の領域に確保されているので、異常状態がクリアされない限り保持されている。

10.5.2　現在の日時

現在の日時はメインタスクでのループの最初に RTC を読んで共有変数へ格納した値を用いている。実際には CGI タスクが RTC を直接読むわけではないので、メインタスクの処理の位置によって最大十数秒程度の誤差が最大で生じる可能性があるが、その程度の誤差は運用上重要ではないので、RTC の排他制御を考慮しこの値を用いることとした。

10.5.3　現在の温度

メインタスク側で設定間隔毎に測定された値が共有変数へ格納されており、その値を用いることで現在の温度とする。

10.5.4　LCD 表示状態

CGI のコマンド操作により LCD に簡易メッセージを表示する機能があるが、ステータス画面からでも現在の表示されている内容（メッセージと書き込まれた日時）がわかるように表示する。

10.5.5　メッセージ用 LED 表示状態

CGI のコマンド操作により LED を使った簡易メッセージを表示する機能があるが、ステータス画面からでも現在の LED の表示の内容（点灯、早い点滅、遅い点滅、消滅）がわかるように表示する。

10.5.6　現在のカメラ画像

カメラ画像はメインタスク側での処理で常に更新され格納されている。格納エリアは共有メモリ内で 3 画像分あり、現在の最新の番号を示すインデックス(0～2)も共有変数にて持っている。このインデックスの示す最新画像を読み出し、表示することで最新画像とする。

表示に際してはRGBのROWデータはフォーマットとして認識されないので、生成の容易さを考慮しBMP[xxvi]フォーマットを用いることとした。BMPフォーマットにはランレングスによる圧縮もサポートしているが、カメラ画像のような自然画像はランレングス圧縮には向かないので、フォーマットは非圧縮とした。またカラーはTrevaからのYUVをRGBに変換したカラーである 24 ビットをそのまま用いた。

BMPフォーマットの作成方法の詳細は 11.10BMPフォーマット にて説明している。

[xxvi] Windows Device Independent Bitmap　Windows 環境で標準的に使用される画像形式

10.5.7　温度推移グラフ

メインタスクにて 144 ポイント分の温度を共有メモリに記録している。この値を用いて時間軸による温度推移グラフを生成し表示を行う。

温度推移グラフの画像の画面設計図を 図 10-11 に示す。

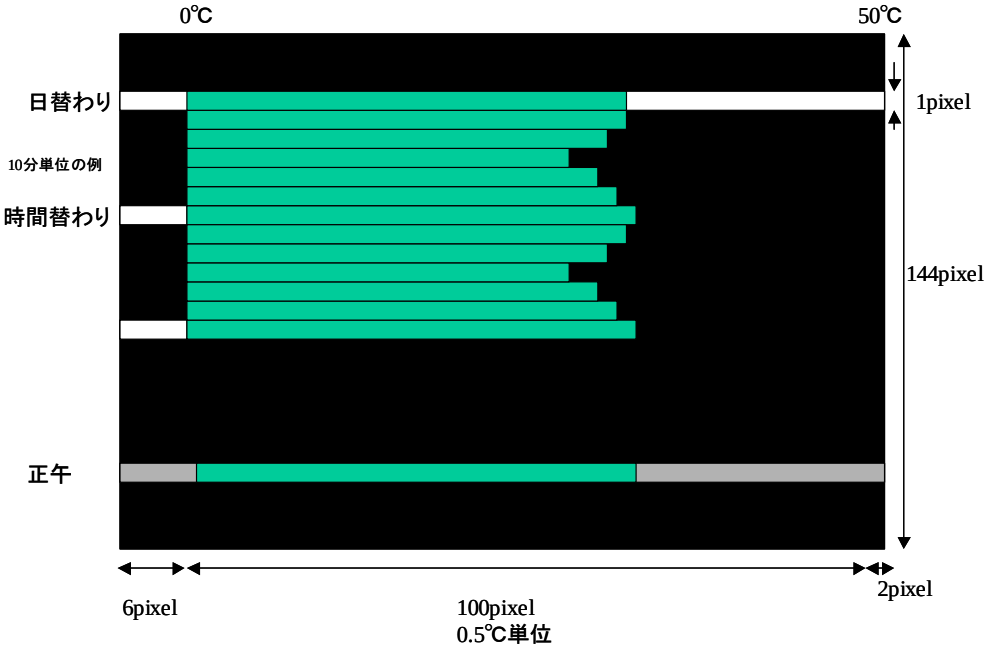


図 10-11　温度推移グラフ設計図

BMP ファイルのランレングス圧縮を用いることを想定し、縦軸が時間、横軸に温度という形にした。左端 6 ピクセルと右端 2 ピクセルは時間属性を示すこととし、真中の 100 ピクセルは 0.5℃を 1 ピクセルとする温度用に使用、また縦は1つの温度値に対して1ピクセルを使い、横 108 ピクセル縦 144 ピクセルの BMP フォーマット画像とした。

グラフは一番上のラインが現在の温度値を示し、それから下にいくにしたがって過去の温度値を示す。この際に時間がわかるように午前 0 時には白(RGB=FFFFFF)のラインを正午にはグレー(RGB=808080)のラインを横幅いっぱいにバックグラウンド色として用いる。また 1 時間単位の正時には左の 6 ピクセル分にのみ白(RGB=FFFFFF)のラインを表示させる。これにより現時刻からさかのぼればグラフの時刻が算出できることとなる。

温度の棒グラフは温度値がわかりにくいのだが、図 10-12 に示したようなパレットカラーを用いて温度の範囲を色でわかるようにし

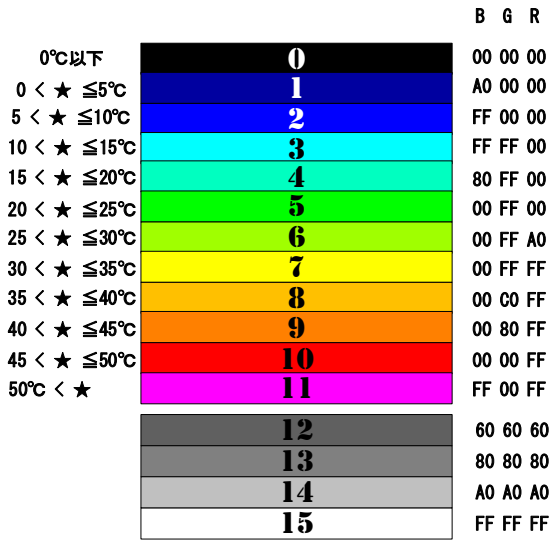


図 10-12　温度グラフ用パレット

た。この図の中での 0～15 の番号はBMPフォーマットの 16 色カラーを用いる際のパレット番号である。このパレットのうち温度表示には 0～11 を用いた。

画像としては、横方向の棒グラフとし、16 色パレットとランレングス圧縮を用いることで、温度グラフはわずか 1272 バイトに抑えることができた。

10.5.8 温度測定間隔

温度測定の間隔には 1 分、3 分、5 分、10 分（デフォルト）と CGI プログラムのコマンドによって変更することができる。その現在の設定値を表示する。

10.5.9 起動日時

電源を投入若しくはリセットされメインタスクのイニシャライズルーチンによって設定された時刻が起動時刻として共有変数に格納されている。その値を起動時刻として表示する。

このシステムはバッテリーバックアップはしていない。よって停電があると全ての情報が失われてしまう。よって、起動時刻が変わっていればリセットされたか電源が落ちたかの判断ができ、簡易的な停電検出機能となる。

10.5.10 Alert 解除コマンド

異常検知時には CGI によるステータス画面には Alert 状態となり、システム本体は警告用赤 LED が点滅している状態となっている。この状態を解除するためのコマンドである。このコマンドを実行すると通常状態へと戻り、警告用赤 LED も消灯する。

10.5.11 LCD 表示コマンド

システム本体に実装されている LCD に表示を行うコマンドである。LCD は 16 文字 2 行あるが、メッセージは 16 文字までとし、残りの 1 行にはメッセージが書き込まれた日時を表示する。

10.5.12 LED 制御コマンド

システム本体に実装されているメッセージ用緑 LED を制御するコマンドである。ラジオボタンにより、OFF（消灯）、ON（点灯）、Blink-Slow(遅い点滅)、Blink-Fast(早い点滅)と 4 種類選択可能であり、現設定値がデフォルトとして表示されている。

実際の実装は OFF と ON はポートを汎用出力として制御し、点滅はタイマー機能を用いており、早い点滅は周期を 0.25 秒、Duty25%とし、遅い点滅は周期を 1 秒、Duty40％としている。

10.5.13 温度測定時間設定コマンド

メインタスクで行っている温度測定の間隔を設定するコマンドである。ラジオボタンにより 1 分、3 分、5 分、10 分が選択可能であり、現設定値がデフォルトとして表示される。

第11章　　要素技術とその実装

このシステム開発に用いた要素技術とその実装についてこの章にて説明する。

11.1　IP (Internet Protocol)

OSI参照モデルの第3層であるネットワーク層に位置し、インターネットの実現の基となっているプロトコルである。ヘッダを用いたデータのカプセル化、分散処理、ベストエフォート型という特徴がある。

IP は今回用いた OS である MES にてサポートされている。

11.2　TCP (Transmission Control Protocol)

OSI 参照モデルのトランスポート層に位置し、もっとも広く用いられているネットワークプロトコルの1つであり、ネットワーク層の IP と、セッション層以上のプロトコルとの橋渡しをする役割を担う。

TCP はコネクション型で、データの送信順、受信順を保証し、信頼性が高い通信プロトコルである。

TCP は今回用いた OS である MES にてサポートされている。

11.3　UDP (User Datagram Protocol)

OSI 参照モデルのトランスポート層に位置し、IP をほとんどそのまま利用した通信プロトコルである。

コネクションレス型であり、TCPより信頼性は低いが反面転送速度が高い、1対多の通信にも向いているとの特徴がある。

UDP は今回用いた OS である MES にてサポートされている。

11.4　NTP (Network Time Protocol)

ネットワークを介して正しい時刻を得るプロトコルであり、RFC-1305[20]で規定される。また、NTPの簡易版であるSNTP (Simple Network Time Protocol)はRFC-2030[21]にて規定される。

本システムでは SNTP を用いてネットワーク上の NTP サーバから正確な時刻を得て RTC への初期時刻の設定を行う。また定期的にアクセスを行い RTC の誤差を補正するために用いる。

以下に UDP クライアントである SNTP による時間取得の手順を示す。

① ソケット生成

UDP においてはコネクションを張る必要が無いので、ネットワークプログラムにおいてはソケットを生成するだけでよい。

② パケット準備

図 11-1 にNTP/SNTPバージョン 4 のメッセージフォーマットを示す。このメッセージはIPヘッダ、UDPヘッダに続くものである。

このパケットに初期データを設定してNTPサーバにUDP送信を行う。

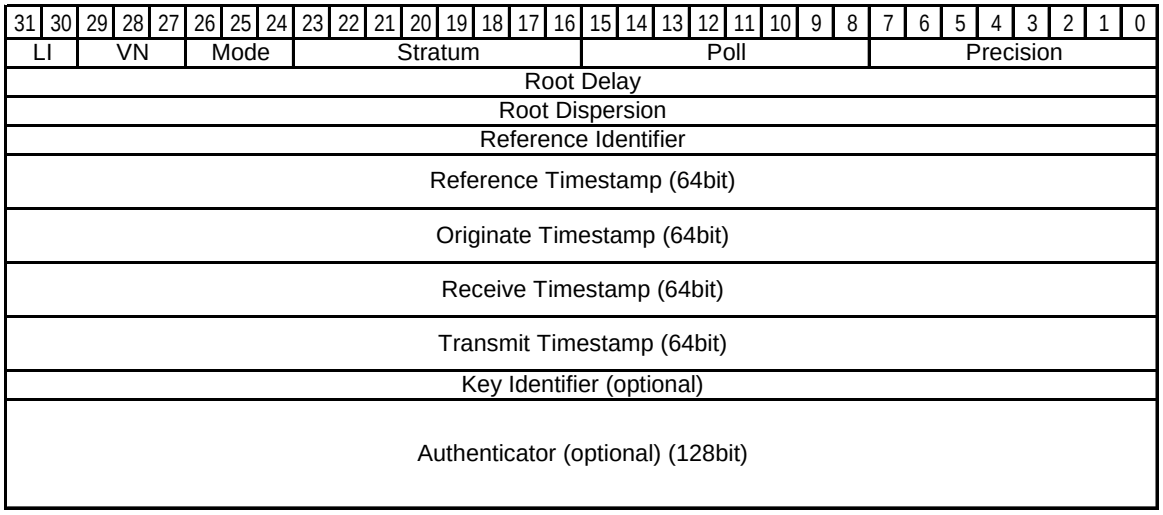


図 11-1 NTP/SNTP メッセージフォーマット Version4

以下にオプションを除く各項目について説明する。

- LI　閏秒指示子
  - 00　警告なし
  - 01　最後の 1 分が 61 秒
  - 10　最後の 1 分が 59 秒
  - 11　警告状態(時計が同期していない)
- VN　NTP/SNTP バージョン番号を示す。
- Mode　モード
  - 000　予約
  - 001　対称能動
  - 010　対称受動
  - 011　クライアント
  - 100　サーバー
  - 101　ブロードキャスト
  - 110　NTP 制御メッセージのため予約
  - 111　私的使用のため予約
- Stratum　階層、ローカル時計の階層レベルを示す。
  - 00000000　不明または有効でない階級
  - 00000001　1 次参照(電波時計等の時刻を参照)
  - ～00001111　2 次参照
  - ～11111111　予約
- Poll　連続するメッセージの最大間隔を示し最も近い 2 のべき乗で表される。

- Precision　ローカル時計の精度を示し最も近い 2 のべき乗で表される。
- Root Delay　1 次参照源までの往復遅延の合計を示す。
- Root Dispersion　1 次参照源との名目的な相対誤差を示し、16 ビットの整数部と 16 ビットの小数部の固定小数点で示される。
- Reference Identifier　固有の参照源を確認するための 32 ビットのビット列である。
- Reference Timestamp　ローカル時計が最後に設定あるいは、修正された時刻であり、64 ビットタイムスタンプフォーマットで表される。
- Originate Timestamp　クライアントからサーバーへリクエストが発信された時間であり、64 ビットタイムスタンプフォーマットで表される。
- Receive Timestamp　サーバーへリクエストが到着した時間を示している。64 ビットタイムスタンプフォーマットで表される。
- Transmit Timestamp　サーバーからクライアントに応答が発信された時間である。64 ビットタイムスタンプフォーマットで表される。

ここでいう 64 ビットタイムスタンプフォーマットというのは、整数部 32 ビット、小数部 32 ビットの固定小数点のフォーマットを言う。

このメッセージパケットのうち、VN に 011 を、Mode に 011 を設定。残りのフィールドは全て 0 に設定したメッセージパケットを送信する。

ここで、本来はネットワーク遅延を考慮して時刻のずれを計算し、より正確な値を求めるのが普通であるが、この得られた値の処理にかなりの時間がかかるので正確な時間を得ても結局は誤差が出てしまうということと、そこまでの正確さは問わないという使用形態からネットワークのディレイの誤差は含むものとした。

③　パケット受信

送信したメッセージパケットに対し NTP サーバが値を設定して返信が行われる。

④　パケット解析・時刻取得

受信したメッセージパケットを解析する。

まずは LI フィールド調べ、LI=3 即ちサーバが同期していない場合は時刻取得を諦め、エラーとすることとする。

次に時刻の取得である。前述したようにネットワークディレイ、オフセットを考慮した正確な時刻までは計算せずに、サーバの送出時刻をそのまま用いることとしたので、時刻としては Transmit Timestamp のフィールド値を小数点以下を無視する形で用いることとした。

このフィールドの整数部 32 ビットは 1900 年 1 月 1 日 0 時からの経過時間が秒単位で刻まれている。

その方法であるが、今回開発に使った H8 の C 言語では 32 ビットの符号なし整数を扱うのが出来なかったために、あらかじめ計算しておいた 2005 年 1 月 1 日 0 時(C57FE7F0)を引き、そこからの計算で現在の日時を得るようにした。それでもかなりの計算量であり、H8 には重い処理であると思われるので、先の小数点以下の無視、ネットワークディレイの無視等の誤差を考慮して、計算値に 1 秒を加えている。正確な根拠は無いが、何もしないよりは誤差が少なくなっ

ていると思われる。

この SNTP の実装と RTC の実装によりいつでも約 1 秒程度の誤差で正確な時刻を知ることが出来、RTC のバッテリーバックアップ等の処置をする必要が無くなった。

11.5 POP3 (Post Office Protocol version 3)

メールを送る際、SMTPサーバを利用する時の認証方法としてPOP before SMTPを用いることにしたのでそのためのPOP実装である。POP3 とはRFC(Request for Comments)1939[22] に定義されているインターネット上のメールを読み込む際のプロトコルである。

POPクライアントであるH8 側とPOPサーバとの間のPOP3 プロトコルによるフローを 図 11-2 に示す。

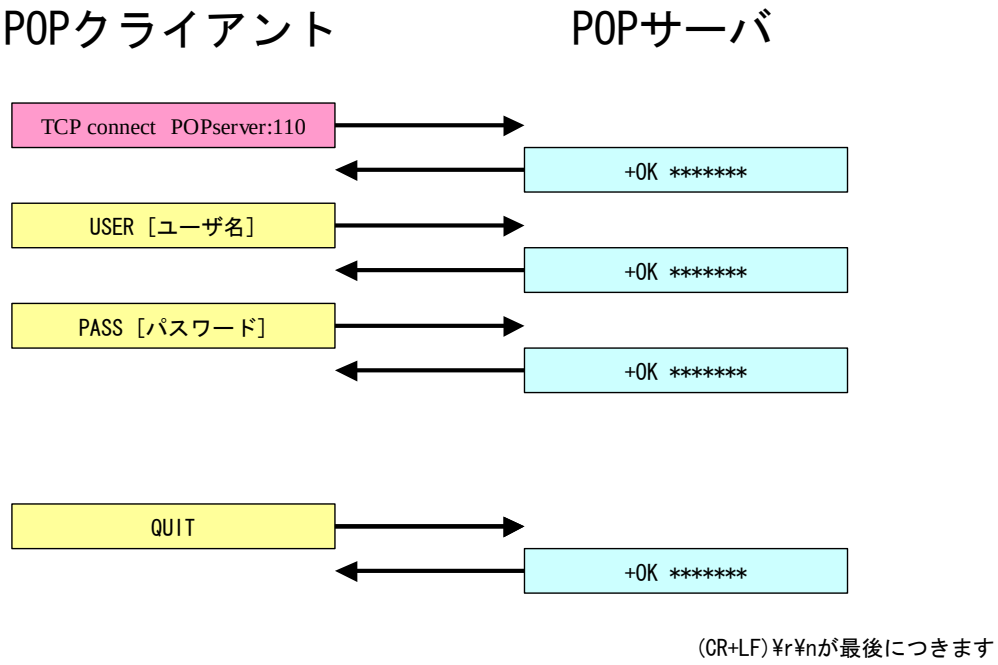


図 11-2 POP アクセスフロー

POP3 は TCP 上に実装されるので、まずは POP サーバへコネクションを行う。コネクションが確立されたら +OK を先頭とするメッセージが返ってくるので、これを確認しながら、USER でユーザ名を、PASS でパスワードを送信。本来はその後にメッセージの受信を行うのが普通であるが、今回の POP3 はユーザ認証が通りさえすれば完了なので、このまま QUIT にてコマンドは終了。そしてコネクションを切ることで POP3 アクセスは終了となる。この後に時間を開けることなく SMTP アクセスすることで POP before SMTP 認証となる。

11.6 SMTP (Simple Mail Transfer Protocol)

SMTPとはRFC(Request for Comments) - 821[23] に定義されているインターネット上でメールを送信する際のプロトコルである。

SMTPクライアントであるH8 側とSMTPサーバとの間のSMTPプロトコルによるフローを 図 11-3 に示す。

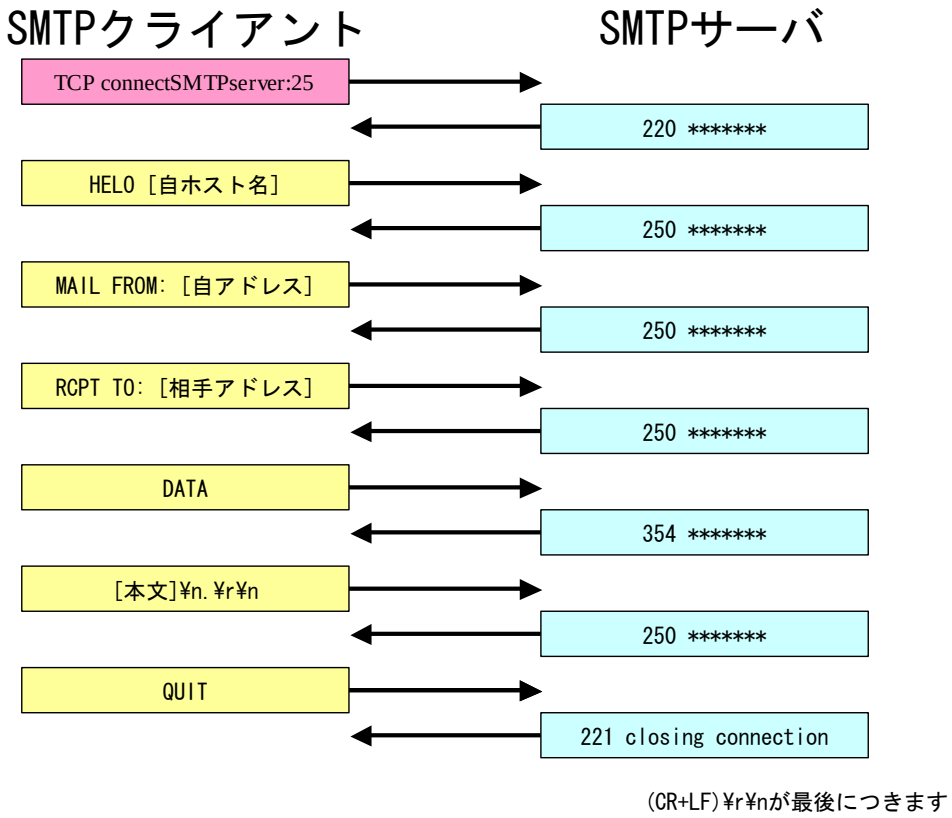


図 11-3 SMTP アクセスフロー

SMTP も POP3 と同じく TCP 上に実装されるので、まずは SMTP サーバへコネクションを行う。コネクションが確立されると 220 というリターンコードを先頭とするメッセージが返ってくる。以降 SMTP サーバからのメッセージには頭に 3 桁の数字のリターンコードがあるので、この数字で意味の判断を行うことができる。このリターンコードを確認しながら、HELO で SMTP セッションの開始を宣言。自分のアドレス、送り先のアドレス、そしてメールに表示するヘッダ部とメッセージを含んだ本文を送り、最後に本文の終わりの印である.を送りメールの送信は終了となる。このまま QUIT にてコマンドは終了。そしてコネクションを切ることで SMTP アクセスは終了となる。これでメールが送れたこととなる。

11.7 I²C (Inter Integrated Circuit)[24]

Philips Semiconductors[xxvii]社が提唱しているシリアル・バスで以下の特徴をもつ。

- 必要なバスラインはデータ(SDA)とクロック(SCL)の 2 本のみで構成。
- バスに接続される各デバイスは、デバイス固有のアドレスを持ち、それをもとにソフトウェアにより各デバイスにアクセス可能。その際にシンプルなマスター／スレーブシステムを形成。
- 衝突検出、調停機能を備えたマルチマスターをサポート。
- シリアルで 8 ビット単位の両方向のデータ転送をスタンダードモードでは最高 100kbit/s、ファーストモードでは最高 400kbit/s、ハイスピードモードでは最高 3.4Mbit/s で実行可能。
- バスの静電容量が 400pF 以下であればバス上にいくつでもデバイスを接続することが可能。

[xxvii] http://www.semiconductors.philips.com/

また、これらSDA、SCLは双方向のラインであり、図 11-4 に示すように並列抵抗を介して正の電源電圧に接続される。よって、バスが開放されている時はどちらのラインも"H"の状態になる。バスに接続されているデバイスの出力段にはAND接続機能を実現する為にオープンドレイン(オープンコレクタ)が必要となる。

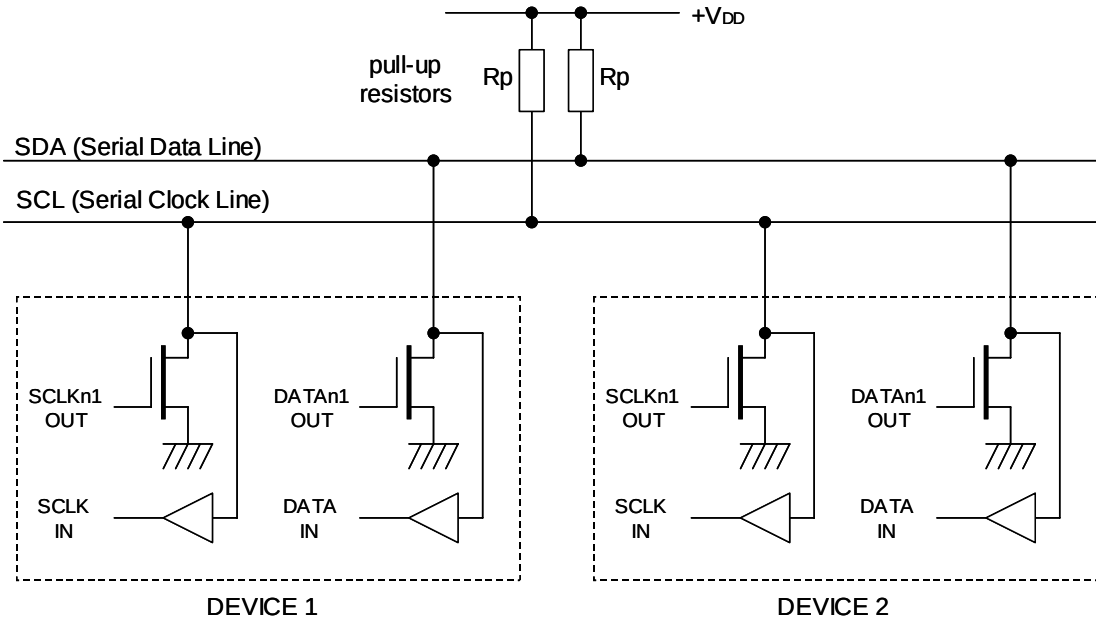


図 11-4 I²C デバイスの接続

本システムではポート 6 の P60 に SCL を P61 に SDA を接続したが、P60 の SCL の方は設定にてプルアップ抵抗をイネーブルにした。P61 の SDA は RTC モジュール内にてプルアップ接続がされているので H8 側はディスエーブルにしている。よって H8 側では入力若しくは H 出力の時には Hi-Z で制御、L 出力の際には Low 出力で制御と 2 種類の制御方法で I^2C 機能を実現した。

次に簡単に I^2C による転送を説明する。

今まで説明したように I^2C は双方向バスである SCL、SDA の 2 本を用いて通信を行う。その基本的なコンディションとして、START、STOP、データ通信、アクノリッジがある。

START、STOP は常にマスタ側が制御し、I^2C 通信の始まりと終了を示す。データの通信は 1 バイト(8 ビット)単位で区切られ、そのたびにデータの受け取りを確認するアクノリッジのフェーズがあり、STOP コンディションにより通信が終了するまで、この繰り返しが行われる。よって一度のアクセスで多バイトの通信も可能であり、双方向混在も可能となる。

使用の詳細は参考文献であるI^2C仕様書 56[24]に譲るとして、ここでは今回使用したRTCの使用に関して説明する。

本システムにおいては、H8 がマスタであり、RTC がスレーブデバイスとなる。

RTC は I^2C アドレスとして Read $A2_H$、Write $A3_H$ という固有のアドレスを持つ。I^2C 通信においては、

H8 による START コンディションの後の 8 ビットにてアドレスが示される。そのアドレスが $A2_H$ であれば、RTC からの読み出し命令、$A3_H$ であれば RTC への書き込み命令ということになる。

RTCへのI^2Cアドレス通知の後はRead時、Write時どちらの場合でもRTC内のレジスタに対するアドレスが通知される。RTC内のレジスタの内容を 表 11-1 に示す。

表 11-1 RTC-8564NB レジスタ

00_H	Control1
01_H	Control2
02_H	Seconds
03_H	Minutes
04_H	Hours
05_H	Days
06_H	Day of Week
07_H	Months / Century
08_H	Years
09_H	Minutes Alarm
$0A_H$	Hour Alarm
$0B_H$	Day Alarm
$0C_H$	Weekday Alarm
$0D_H$	CLKOUT frequency
$0E_H$	Timer control
$0F_H$	Timer

Read 時であればこれらのレジスタを読み出し、Write であればこれらのレジスタに設定を行う。

つまり、Read 時には

START→Read($A2_H$)コマンド書込み→Address 書込み→Data 読込み→STOP

という流れになり、Data 読み込みフェーズのみが RTC→H8 への方向となる。

また、Write 時には

START→Write($A3_H$)コマンド書込み→Address 書込み→Data 書込み→STOP

となり、全てのフェーズが H8→RTC への方向である。

また、アドレスインクリメント機能によるバースト転送もサポートされているので、

START→Read($A2_H$)コマンド書込み→Address 書込み→Data 読込み→Data 読込み→Data 読込み→…→STOP

START→Write($A3_H$)コマンド書込み→Address 書込み→Data 書込み→Data 書込み→Data 書込み→…→STOP

のような連続した Read/Write アクセスも可能となる。但し、この RTC の規格として START から STOP まで 1 秒以内という制約があるので、その制約を守る必要がある。

本システムにおけるRTCは 表 11-1 で示した 02_H～08_Hの日時のレジスタの設定、読み出し、また日時設定後の再スタートの為の 00_HのControl1レジスタへの設定の用途でしか使用せず、それ以

外のレジスタは未使用若しくは初期設定のみとなる。よって、日時の読み出しに関しては、02_Hからの 7 バイトの読み出しアクセスによる実現。また日時の設定に関しては 00_HのControl1 レジスタによるクロックの停止後に 02_Hからの 7 バイトの書き込み、そして再度 00_HのControl1 レジスタによるクロックのスタートという 2 種類のインタフェースのみを実装することでRTCのI^2Cでの使用を可能とした。

11.8 ITU-R BT.656[25]

ITU-R BT.656 とは 国際電気通信連合（International Telecommunication Union）[xxviii]内の 無線通信標準化部門(ITU Radiocommunication Sector)[xxix]による勧告で、テレビ放送向けコンポーネント信号のインタフェースに関する規格を定めたものである。

ITU-R BT.601[26] に示される 16 ビットの色差信号YCbCrのデータを 8 ビットに多重化して伝送するためのフォーマットで、同期信号情報も含まれているので 8 ビット信号だけで画像信号の伝送が可能となる。

本システムでは 図 11-5 に示すBT.656 規格に準拠したデータを出力するCCDカメラを購入し、データの実測を行い、7.6カメラ（BT656 カメラ、Treva）にて説明したように実装の検討はしたものの、コストの問題、実装の問題から使用には至らなかった。

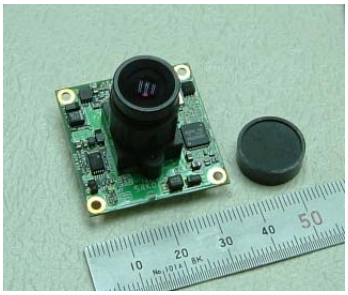


図 11-5 BT.656 データ出力 CCD カメラ

11.9 IEEE 802.3i (10BASE-T)

1980 年に設立された IEEE 802 LAN/MAN Standard Committee により標準化された 10Mbps のイーサネット規格。セグメント長は 100m、ハブ段数は 4 段までとなっており、カテゴリー3 以上のツイストペアケーブルの両端に RJ-45 コネクタを付けたケーブルにより接続される。

本システムでは、H8 キットの基板上にイーサネットコントローラとして Realtek[xxx] 社のRTL8019AS と 10BASE-T Filter PTH moduleである YCL[xxxi]社の 20F001N[27] 、そしてMACアドレス書込済の1kbitシリアルEEPROMが実装されており、H8 からRTL8019ASのレジスタを制御することで簡単に10BASE-Tのイーサネットが使用できるようになっている。またこのネットワークの制御においてもTCP/UDPレベルまでOSでサポートされているので、物理層やデータリンク層であるこの下位のレイヤーを意識する必要が無くネットワークプログラムが開発可能である。

11.10 BMPフォーマット [28]

Windows Device Independent Bitmap フォーマット。Windows 環境で標準的に使用される画像形式である。本システムではカメラ画像、温度グラフを HTTP クライアント側に表示させる為にこのフォ

[xxviii] http://www.itu.int/home/index.html
[xxix] http://www.itu.int/ITU-R/index.html
[xxx] Realtek Semiconductor Corp.（ http://www.realtek.com.tw/ ）
[xxxi] YCL Electronics Co., Ltd.（ http://www.ycl.com.tw/ ）

ーマットを用いている。

基本的な構造はヘッダと画像データというシンプルな構造である。ヘッダの種類によって Windows Bitmap と OS/2 Bitmap の 2 種類があるのだが、本システムではより一般的である Windows Bitmap フォーマットを用いた。

ヘッダはその内容により、表 11-2 に示す 14 バイト固定サイズのファイルヘッダ、表 11-3 に示す 40 バイト固定(Windows Bitmap時)サイズの情報ヘッダがあり、その後に必要であれば 表 11-4 に示すカラーパレットの情報が付属する。

また、BMP データはリトルエンディアンであるが H8 はビックエンディアンであるので、BMP データを生成するときにはエンディアンを考慮する必要がある。

表 11-2 BMP-ファイルヘッダ

サイズ	内容	備考
2 byte	ファイルタイプ	'BM'固定
4 byte	ファイルサイズ	バイト数
2 byte	予約領域	未使用
2 byte	予約領域	未使用
4 byte	画像データまでのオフセット	

表 11-3 BMP-情報ヘッダ

サイズ	内容	備考
4 byte	情報ヘッダのサイズ(byte)	40 固定
4 byte	画像の幅	ピクセル単位
4 byte	画像の高さ	ピクセル単位
2 byte	プレーン数	常に 1
2 byte	1 画素あたりのデータサイズ	1: 2 色ビットマップ 4: 16 色ビットマップ 8: 256 色ビットマップ 24: 1677 万色ビットマップ 32: 1677 万色ビットマップ
4 byte	圧縮形式	0: 無圧縮 1: ランレングス(8bit/pixel) 2: ランレングス(4bit/pixel) 3: ビットフィールド
4 byte	画像データ部のサイズ(byte)	96dpi 時 3780、0 時デフォルト
4 byte	横方向解像度	96dpi 時 3780、0 時デフォルト
4 byte	縦方向解像度	96dpi 時 3780、0 時デフォルト
4 byte	格納パレット数	0 時デフォルト
4 byte	パレットインデックス	0 時デフォルト

表 11-4 BMP-カラーパレット

サイズ	内容	備考
1 byte	青	0～255
1 byte	緑	0～255
1 byte	赤	0～255
1 byte	予約領域	0

これらのヘッダの後に画像データが格納される。

本システムで用いた BMP フォーマットはカメラ画像用と温度推移グラフ用の 2 種類で、各々のフォーマットの詳細は以下のとおりである。

- カメラ画像用 BMP フォーマット
  - 横 96×縦 72 ピクセル
  - 24bit/pixel 無圧縮
- 温度推移グラフ用 BMP フォーマット
  - 横 108×縦 288 ピクセル
  - 8bit/pixel ランレングス圧縮 （16 色パレット使用）

カメラ画像用の 24 ビット無圧縮の画像データは非常にシンプルで、前述したヘッダの後（オフセットで指定した位置）に、青、緑、赤と 1 ピクセルに付き 3 バイトのデータを総ピクセル数分書き込むことでカメラ画像の BMP データとなる。この BMP データのサイズはヘッダ 54 バイト、画像データ 20736 バイト（96×72×3）の合計 20790 バイトとなる。

温度推移グラフは 16 色パレットを用いた 8 ビットのランレングス圧縮である。図 11-6 に今回用いた 16 色パレットを示す。温度を表す棒グラフに 1～11 のパレットを用い、バックグランド、時間メモリにそれ以外のパレットを用いる。パレットデータは 4 バイト×16 色で 64 バイト必要である。温度の棒グラフは 1 ラインに付き目盛り部、棒グラフ部、残りのバックグランド若しくは目盛り部の 3 ブロックに分け、各ブロック毎にピクセル数、パレット番号の 2 バイトを書き込む。そして 1 ラインの終わりに区切りとして識別コード 00、00 の 2 バイトを書き込む。それを全ライン数分繰り返し、最後に画像の終わりとしての識別コード 01、00 の 2 バイトを書き込んでBMPデータとなる。このBMPデータのサイズはヘッダ 54 バイト、パレットデータ 64 バイト、画像データ 1154 バイト（（2 バイト×3 ブロック＋識別コード 2 バイト）×144 ライン+識別コード 2 バイト）の合計1272バイトとなり、ランレングス圧縮で非常に小さなサイズとなっている。

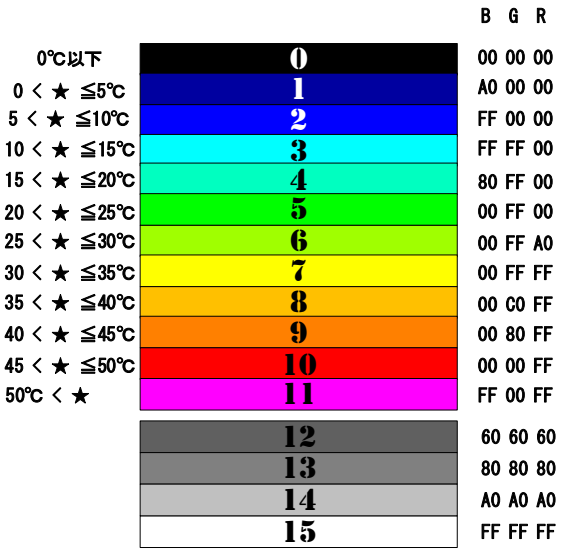


図 11-6 温度グラフ用パレット

11.11　色空間変換 (YUV→RGB変換) [29][30]

カメラの画像データは YUV というY（輝度）、U（赤の色差）、V（青の色差）の信号で定義される色空間によって表現される。一方コンピュータ画像は RGB というR（赤）、G（緑）、B（青）の信号で定義される色空間によって表現されることが多く、しばしばこの空間同士の変換が必要となる。

本システムではYUV、RGBともに 8 ビットで表す。この時Yの範囲は 16～235 であり、UとVは 128 を中心とする 16～240 である。また、RGBは全て 0～255 である。この変換式を 数式 11-1 に示す。この式を用いてRGBへの変換を行う。この際、YUV信号は 422 というサンプリング係数であるので、UとVに関しては 2 ピクセル同じ値を用いることで、RGB側は 444 というサンプリング係数に変換されることとなる。

数式 11-1 YUV→RGB 変換式

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.164 & 0 & 1.596 \\ 1.164 & -0.391 & -0.813 \\ 1.164 & 2.018 & 0 \end{bmatrix} \begin{bmatrix} Y \\ V \\ U \end{bmatrix}$$

第12章　　研究結果の評価と課題

図 12-1 ホームセキュリティシステム外観

以上に説明した内容に沿って構築した本システム(図 12-1)を用い実際に稼動させてその優位性を確認した。

温度異常の検知においては火災を実際に起こす等の検知される条件を実践するのは困難であるので温度センサの横でライターを用いて意図的に温度上昇を起こしたことによる検知を行った。それ以外の自然現象での確認は出来ていないが、稼動状態で誤動作は一度も起こらなかったので、異常温度検知方法については妥当であるのではないかと推測する。

また、画像異常の検知については、実際に部屋に自分がいる場合には、カメラの真中の検知部分に変化がないように見えても、影や明るさの変化から僅かな動きでも検知でき、予想以上の結果であった。ただし、直射日光が写りこむような場所においては光の具合を誤検知するときがあった。また、夜の暗い場合は明るさの変化での画像検知機能は役立つが、暗い画像では外部から確認が困難であるという欠点が見られた。直射日光に関してはシステムの内部での解決は困難であるが、留守宅はカーテンが閉まってる。外部から光の入る余地が無いはず。ということを前提に置けば問題にはならないのではないかと思われる。また、暗い状況においての画像確認は CCD センサを赤外線タイプに変更し、暗い場合には赤外線 LED にて照射するようにすれば、暗い状況で画像を確認できるようになる。このとき色情報は失われてしまうが、本システムの目的からすれば問題はないであろう。研究当初は実際に赤外線カメラを検討していたのだが、アナログ信号出力タイプのカメラしか見つけられず、その場合デジタルデータへの変換には費用と工数の負担が大きくなるということで断念した。但し、今回用いた Treva のようなマイコンとの親和性の良いデジタルデータ出力型の赤外線カメラが安価で入手できるのであれば、積極的に用いた方が良いと思う。その際の画像検知方法については現在の Y と UV の変化を見るのと同様なアルゴリズムで問題ないと思われる。

また、今回用いたOSは俗に言う枯れた状態であるため安定性は良いのだがパフォーマンスという面

では弱いという欠点がある。今回用いたMESの最新版はVer.2.1[xxxii]となっており途中で移行を試みたのだが、パフォーマンスについてはかなり改善されており魅力的ではあったものの、当時公開されていたβバージョンでは安定性に欠けるという欠点があり移行を断念した経緯がある。但し、現在(2006/1/15)公開されているバージョンではより安定化されているようであるので、このバージョンを用いれば現システムよりも画像読み込みやHTTPサーバのパフォーマンスはかなり改善されるはずである。

また、費用であるが、本システムを開発するのにかかった主な部品の費用を以下に示す。

- H8 マイコンボード、拡張基板　　8700 円(現在は 5700 円)
- LCD　　700 円
- RTC モジュール　　500 円
- 温度センサ　　150 円
- Treva(オークション)　　約 1000 円

これ以外にも三端子レギュレータ、汎用ロジックである LCX04 等の IC やコンデンサ、抵抗等の部品をいくらか使用しているが、これらは自分の手持ち部品から捻出したものであり、具体的な購入費用は発生していないのだが、これらを全て新たに購入したとしても数百円程度かと思われる。

以上のことから全費用を計算すると、9000 円程度[xxxiii]になる。但しこのうちの大半は部品代ではなく、製品としての売値である。実際に商品化を前提として量産をする場合の具体的な部品代の計算は困難だが、H8 マイコンボード上に実装されているだけのものを実現するのには部品代は大雑把に見積もっても数百円程度に収まるのではないかと思われる。このことから実際にこのシステムを商品化・製造・販売するとしたときの原価は基板代も含めて 2000 円程度にはなるのではないかと思われる。これに加えてケース等の外装費がかかるが、それら全てを考慮したとしても、希望小売価格 1 万円前後で販売できる商品になるのではないかと思われる。

加えてランニングコストについても測定器具が無いので正確にはわからないが、H8 マイコン、DRAM、イーサネットコントローラ以外の大きな部品も無いことと、常時稼動を続けていても殆ど熱を発しないということから、消費電力に関してもかなり抑えられているのではないかと思われる。

このことから、当初の目的であった安価についても十分に満足できるものではないかと思われる。加えてこのシステムは全て自宅にて製作したものであるので、手軽という部分についても問題ないと思われる。また環境に関しても ADSL 等の常時接続ネットワークへの接続と電源が必要というだけで、設置はただ置くだけでよく、非常にシンプルに実現できた。

但し、100%満足しているわけではない。今後の課題としては、前述したカメラの暗所対応に加えて動画への対応というカメラの性能をあげるという方法と、SSL 等のセキュアなシステムへの対応によるより広い範囲での使用が望まれる。また、インターネットからの制御を利用してカメラの方向制御、家庭内の様々なリモコンの制御等より便利にする方向への対応も需要があるのではないかと思われる。加えて商品化を前提とした場合には、H8 のプログラムを各環境に合わせて書き込むという方法は現

[xxxii] Micro Embeded System/MES Ver2.1　http://mes.sourceforge.jp/mes2/index-j.html
[xxxiii] H8 マイコンキットは購入価格でなく値下がりしている現在の価格を用いた。

実的ではないので、メールアドレス、アカウント、IP アドレス、画像検知エリアの位置、サイズ、検知感度等々の設定を WEB から設定可能とすべきだと思うし、今回殆ど実装していない各種プロトコルのエラー処理も実装すべきであろう。
これらの課題を現在のコストと殆ど変わらない中で克服することで、より魅力ある製品になるのではないかと思われる。

最後に、今度このようなセキュリティシステムや製品が増えてくると思うが、このような製品が必要無い世の中になることを祈っている。

謝辞

本研究においては、多大なご助言を頂きました担当教官である信州大学大学院工学系研究科 Pauline Naomi Kawamoto 教官に厚くお礼申し上げます。

また、CAI にて様々な基礎技術の習得に際し、ご指導いただきました教官方に感謝いたします。

そして、社会人でありながら大学院の勉強、研究が出来るように、このような環境をご準備くださりその維持・発展に努力されている教官方、職員の皆様に感謝するとともに、ますますの発展をお祈りいたします。

参考文献、参考 WEB

[1] 平成 16 年度 警察白書
http://www.npa.go.jp/hakusyo/h16/index.html

[2] マイクロ波センサ利用による防犯・セキュリティーシステム
2003 年度 信州大学大学院工学系研究科修士論文 02TA588B 馬場智也著

[3] 各種センサを活用したホームセキュリティシステムの遠隔操作
2004 年度 信州大学大学院工学系研究科修士論文 03TA532A 河野志行著

[4] RFC2821 Simple Mail Transfer Protocol
ftp://ftp.rfc-editor.org/in-notes/rfc2821.txt

[5] H8/3069R F-ZTAT™ ハードウェアマニュアル Rev. 6.00
(株)ルネサステクノロジ http://japan.renesas.com/
http://documentation.renesas.com/jpn/products/mpumcu/rjj09b0165_h83069rf.pdf

[6] RTL8019AS ISA Full-Duplex Ethernet Controller with Plug and Play Function
Realtek Semiconductor Corp.
http://www.realtek.com.tw/downloads/downloads1-3.aspx?spec=True&compamodel=RTL8019AS

[7] LM35DZ Precision Centigrade Temperature Sensor
National Semiconductor Corp. http://www.national.com/
http://www.national.com/pf/LM/LM35.html

[8] RTC-8564NB I2C-Busインタフェースリアルタイムクロックモジュール
エプソントヨコム株式会社 http://www.epsontoyocom.co.jp/
http://www.epsondevice.com/www/PDFS/epdoc_qd.nsf/WJ_rtc_sif/93B6F3C1E6CE88D54925707C002FA580?OpenDocument
RTC-8564JE/NBアプリケーションマニュアル
http://www.epsondevice.com/www/PDFS/epdoc_qd.nsf/WJ_rtc_sif/1DAFDA8D3C66A9C94925707D003E5F7D?OpenDocument

[9] AAFぱ研 http://www.paken.org/
CMOSカメラユニット「Treva」の解析 http://www.paken.org/aaf/treva/index.html

[10] OV6630 Single-Chip CMOS CIF Color Digital Camera
OmniVision Technologies, Inc. http://www.ovt.com/
http://www.paken.org/aaf/treva/ov6630dsdl6.pdf

[11] LP3874-ADJ 0.8A Fast Ultra Low Dropout Linear Regulators
National Semiconductor Corp. http://www.national.com/
http://www.national.com/ds.cgi/LP/LP3874-ADJ.pdf

[12] LM1086-3.3 3.3V fixed output 1.5A Low Dropout Positive Regulators
National Semiconductor Corp. http://www.national.com/
http://www.national.com/JPN/ds/LM/LM1086.pdf

[13] TC74LCX04F Low Voltage Hex Inverter with 5 V Tolerant Inputs and Outputs
(株)東芝セミコンダクター http://www.semicon.toshiba.co.jp/
http://www.semicon.toshiba.co.jp/docget.jsp?pid=TC74LCX04F&lang=jp

[14] SC1602BS*B-SO-GB-K LCD module (16character × 2 lines、STN、No Backlight)
Sunlike Display Tech. http://www.sunlikedisplay.com/
http://www.sunlikedisplay.com/data/SC/Sc1602b.pdf

[15] CL-170UR Ultra brightness red Miniature Chip LED
シチズン電子株式会社 http://www.c-e.co.jp/
http://www.c-e.co.jp/products/pdf/citiled/cl-170.pdf

[16] MES（Micro Embeded System）
http://mes.sourceforge.jp/mes/index.html

[17] RFC-2131 Dynamic Host Configuration Protocol
ftp://ftp.rfc-editor.org/in-notes/rfc2131.txt

[18] RFC-2132 DHCP Options and BOOTP Vendor Extensions
ftp://ftp.rfc-editor.org/in-notes/rfc2132.txt

[19] マイコンWEBサーバ構築 http://mes.sourceforge.jp/mes/cgiprog.html
WEBサーバのソースファイル http.c

[20] RFC-1305 Network Time Protocol (Version 3)
ftp://ftp.rfc-editor.org/in-notes/rfc1305.txt

[21] RFC-2030 Simple Network Time Protocol (SNTP)
ftp://ftp.rfc-editor.org/in-notes/rfc2030.txt

[22] RFC-1939 Post Office Protocol – Version 3
ftp://ftp.rfc-editor.org/in-notes/rfc1939.txt

[23] RFC-821 Simple Mail Transfer Protocol
ftp://ftp.rfc-editor.org/in-notes/rfc821.txt

[24] I²C Inter Integrated Circuit
Philips Semiconductors I²C-bus Information
http://www.semiconductors.philips.com/products/interface_control/i2c/index.html
The I²C-bus specification
http://www.semiconductors.philips.com/acrobat/literature/9398/39340011.pdf

[25] ITU-R BT.656 Interfaces for digital component video signals in 525-line and 625-line television systems operating at the 4:2:2 level of Recommendation ITU-R BT.601[26] (Part A)
http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=R-REC-BT.656

[26] ITU-R BT.601 Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios
http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=R-REC-BT.601

[27] 20F001N 10BASE-T Filter PTH module
YCL Electronics Co., Ltd. http://www.ycl.com.tw/
http://www.ycl.com.tw/YCLWEB/CMS/files/20F001N.pdf?id=20

[28] BMP ファイルフォーマット http://www.kk.iij4u.or.jp/~kondo/bmp/

[29] YUVフォーマット及び YUV<->RGB変換
http://vision.kuee.kyoto-u.ac.jp/~hiroaki/firewire/yuv.html

[30] YcbCr http://image-d.isp.jp/commentary/color_cformula/YCbCr.html

付録

[A] Circuit_A.pdf AKI-H8/3069F LANボード回路図

[B] Circuit_B.pdf 拡張基板部分回路図

[C] プログラムソースリスト – メインタスク用ソースファイル（main.c）

[D] プログラムソースリスト – 共通用定義ファイル（define.h）

[E] プログラムソースリスト – CGIプログラム用ソースファイル（cgi.c）

[F] プログラムソースリスト – 各種サブルーチン

[G] プログラムソースリスト – httpサーバ用ソースファイル（http.c）
MESのサイトにサンプルとして公開されているプログラムをそのまま利用している。
http://mes.sourceforge.jp/mes/http.c

[H] H8_3069R F-ZTATハードウェアマニュアル.pdf
H8/3069R F-ZTAT™ ハードウェアマニュアル Rev. 6.00

[I] RTL8019AS.pdf RTL8019AS Datasheet

[J] LM35.pdf LM35series Datasheet

[K] RTC-8564JENB.pdf RTC-8564JE/NB simple Datasheet

[L] RTC-8564JENBアプリケーションマニュアル.pdf RTC-8564JE/NBアプリケーションマニュアル

[M] ov6630dsdl6.pdf OV6630/OV6130 Datasheet

[N] LP3874-ADJ.pdf LP3874-ADJ Datasheet

[O] LM1086.pdf LM1086 Datasheet

[P] SC1602B.pdf SC1602B Datasheet

[Q] cl-170.pdf CL-170 シリーズDatasheet

[R] 20F001N.pdf 20F001N Datasheet

[S] 20011006_TC74LCX04F_datasheet.pdf TC74LCX04F/FT Datasheet

[T] rfc821.txt RFC-821 SIMPLE MAIL TRANSFER PROTOCOL

[U] rfc1305.txt RFC-1305 Network Time Protocol (Version 3)

[V] rfc1939.txt RFC-1939 Post Office Protocol - Version 3

[W] rfc2030.txt RFC-2030 Simple Network Time Protocol (SNTP) Version 4

[X] rfc2131.txt RFC-2131 Dynamic Host Configuration Protocol

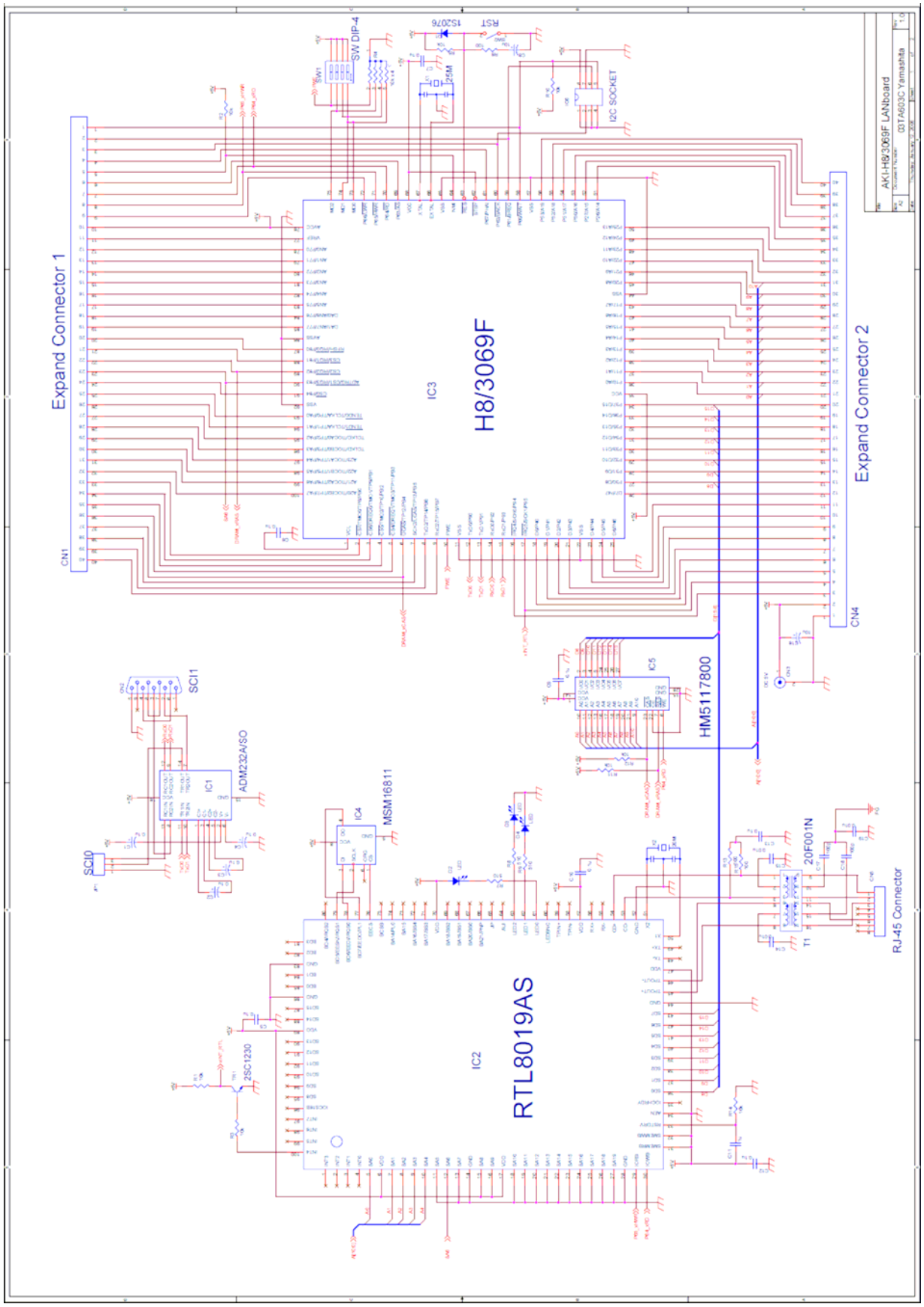
[Y] rfc2132.txt RFC-2132 DHCP Options and BOOTP Vendor Extensions

[Z] rfc2821.txt RFC-2821 Simple Mail Transfer Protocol

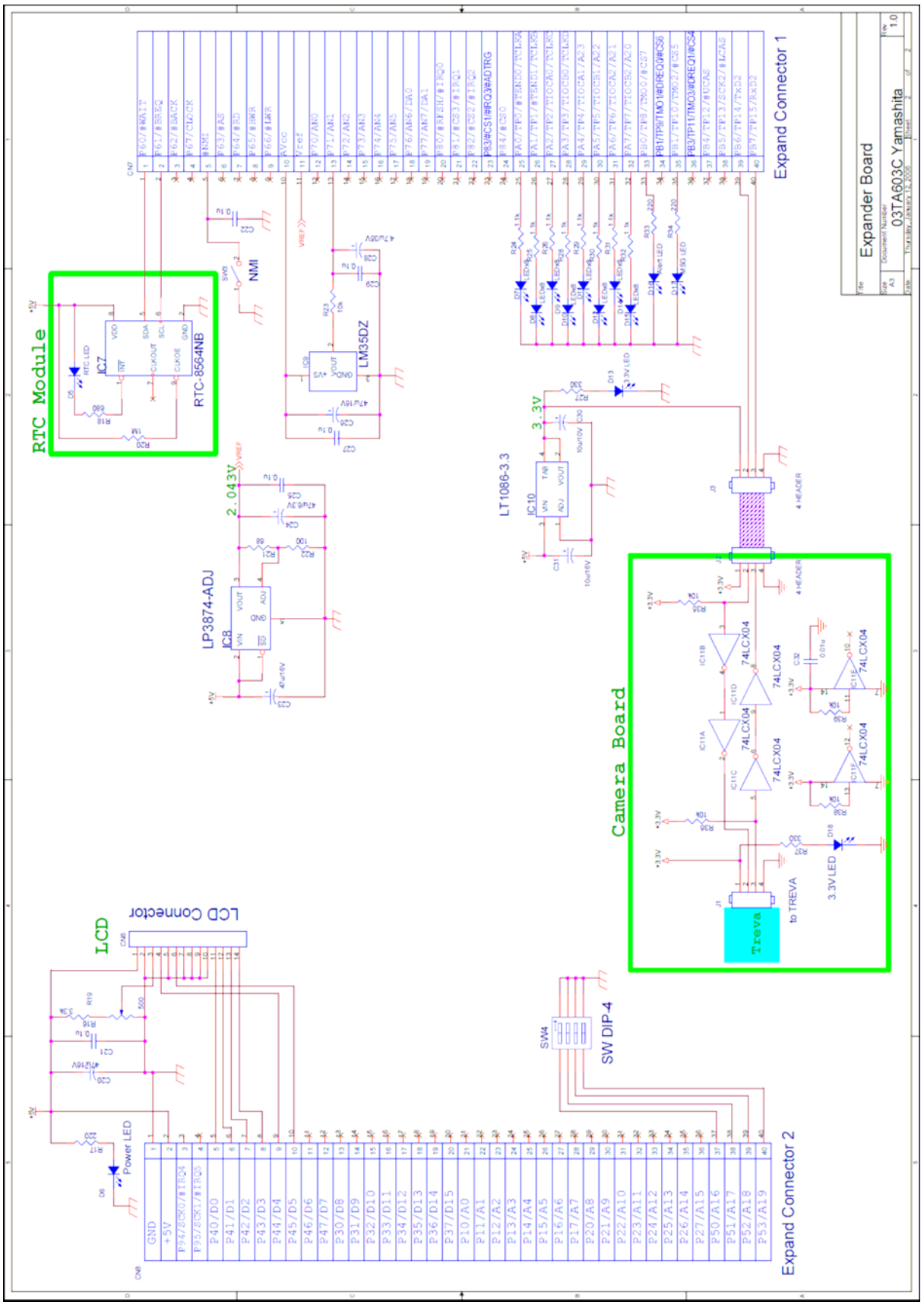
[AA] The I2C-bus specification.pdf THE I²C-BUS SPECIFICATION VERSION 2.1

[A][B] [C][D][E][F]の資料については次項以降に添付するとともに電子ファイルとして提出する。
[G]以降の資料については電子ファイルとして提出する。

付録 [A]　AKI-H8/3069F LANボード回路図



付録 [B]　拡張基板部分回路図



付録[C] プログラムソースリスト – メインタスク用ソースファイル

（注意）メインタスクの作成ソースファイルはいくつかのファイルに分かれるが、開発ツールの都合上 main.c 内で全てを一つにまとめる形で作成している。

main.c　　メインプログラムファイル

```
//
// Home Security System for AKI-H8/3069F
//      Yamashita ( 03TA603C @ SUGSI )
//      Copyright (C) 2005-2006 All Rights Reserved
//
//      --- DHCP version ---
//
//      2006/01/15      Revision 1.0
//



// H8 Information
//       char   1 byte
//       short  2 byte
//       int    4 byte
//       long   4 byte
//       float  4 byte
//       double 4 byte
//

#include <mes.h>
#include <string.h>
#include "Lib/define.h"


// ****************
// * Define Macro *
// ****************
#define CLIP(a)   ({a = ( a > 255 ) ? 255 : ((a < 0 ) ? 0 : a);})


// ****************************
// * Define Seting Parameter *
// ****************************
#define INTERVAL_TEMP   2000    // Intervaltime for A/D
#define INTERVAL        10      // Loop Interval Time (Default:10min)
#define TREVA_WAIT      1       // Wait value to get Treva-Image


// ****************************
// * Define Alert Parameter *
// ****************************
#define ALERT_TEMP      50
#define ALERT_YUV       64


// *************
// * for DEBUG *
// *************
#undef  WRITEYUVFILE
#undef  WRITEPPMFILE
#undef  WRITEBMPFILE
```

```
// ***************************
// * for POP3 / SMTP access *
// ***************************
#define FROM_ADDRESS    "m03603@std.int-univ.com"
#define TO_ADDRESS      "test@localhost"

#define POPbeforeSMTP   1
#define SMTPSERVER      IPADDR(192,168,0,7)
#define SMTP_PORT       25
#define POP3SERVER      IPADDR(192,168,0,7)
#define POP3_PORT       110
#define POP3_USER       "test"
#define POP3_PASS       "test"

//      **************
//      * yahoo mail *
//      **************
//
// smtp.mail.yahoo.co.jp
//      #undef  POP3SERVER
//      #define SMTPSERVER      IPADDR(202,93,83,190)
// pop.mail.yahoo.co.jp
//      #undef  POP3SERVER
//      #define POP3SERVER      IPADDR(202,93,83,191)
//      #undef  FROM_ADDRESS
//      #define FROM_ADDRESS    "******@yahoo.co.jp"
//      #undef  POP3_USER
//      #define POP3_USER       "******"
//      #undef  POP3_PASS
//      #define POP3_PASS       "******"


// ****************************
// * DIPswitch info & define *
// ****************************
// 1    Mailsend                ON:no-mailsend  / OFF:alert-mail send
// 2    Alert Image             ON:skip alert   / OFF:keep lookout image
// 3    Alert Temp              ON:skip alert   / OFF:keep lookout temperature
// 4    Exit                    ON:no-loop exit / OFF:loop continue
#define DIPSW1  (!(P5DR & 0x01))
#define DIPSW2  (!(P5DR & 0x02))
#define DIPSW3  (!(P5DR & 0x04))
#define DIPSW4  (!(P5DR & 0x08))



// ************************
// * Define Shared Memory *
// ************************
#include "Sub/share_memory.h"


// ****************************
// * Define Static Variable *
// ****************************
struct sockaddr myaddr;         // My IP-Address
int *Y, *UV;                    // Treva ImageData pointer


// ********************
// * Define Prototype *
// ********************
void    init_port( void );
int     init_network(char *, unsigned int, unsigned int, int );
```

```
void    init_timer8( void );
void    opening( int, unsigned int );
#include "Sub/treva.h"
int     readtemp( void );
#include "Sub/misc.h"
int     writePPMfile( char *, int *, int *);
int     writeBMPfile( char *, int *, int *);
int     ntp( dateformat * );
int     mailsend( int , char *, dateformat * );
int     timer8x2( int, int );
#include "Sub/rtc.h"
#include "Sub/i2c.h"


// ******************
// *                *
// *  Main Routine  *
// *                *
// ******************
int main( int argc, char *argv[] ) {
  int id;
  int i, j, k, p;
  unsigned int old_d, old_hh, old_mm;
  int old_temp;
  int frame;
  long_char rgb;
  int oldYUV[128], YUV[128];    // 異常画像検出用ワークエリア 8*8*2
  oldYUV[0] = -1;               // 初期値設定


  //
  // 共有メモリ確保
  //
#include "Sub/share_memory.c"                   // 共有メモリ確保

  //
  // 共有メモリ Initialize
  //
  *init = 1;                                    // Init 開始
  bzero(lcdmes, sizeof(dateformat)+20);         // LCDmes clear
  *interval = INTERVAL;                         // Interval Time 1min
  *led = 0;                                     // LED off
  for(i=0; i<144; i++)                          // bmptemp[0-144] clear
    bmptemp[i]=0;
  bmptemp[144] = -1;
  *imageid = -1;                                // Imageid = -1
  bmpA[0] = (char)0x00;                         // normal condition
  *tempA = -1;                                  // normal condition

  //
  // 起動時のパラメータ処理              起動時に INTERVAL 値を指定できる
  //
  if( argc == 2 ) {
    if( argv[1][0] == '1' ) { *interval = 1; }
    if( argv[1][0] == '3' ) { *interval = 3; }
    if( argv[1][0] == '5' ) { *interval = 5; }
    printf("¥n*** Interval time [%d min]¥n¥n", *interval);
  }


  //
  // Port Initialize
  //
  init_port();

  //
```

```
// RTC Initialize
//
init_rtc();

//
// 8bit Timer Initialize
//
init_timer8();

//
// Network Initialize         ( DHCP )
//
int network_index;
network_index = init_network("ne0", (unsigned int)0, (unsigned int)0, 1);
if( network_index == -1 ) {
  exit;
}


//
// LCD Open
//
*lcdfid = open("lcd0", DEVICE );
if( *lcdfid == -1 ) {
  printf("¥n LCD open Error!¥n");
  exit;
}
if( ioctl(*lcdfid, 0, LCD_CLEAR) == -1 ) {
  printf("¥n LCD clear Error!¥n");
  exit;
}


//
// Allocate Treva Image memory --- Y[96*72] / UV[96*72]
//
Y  = (int *)malloc( 27648 );  // 4(int) * 96 * 72 = 27648
if( Y == 0 ) {
  printf("malloc error(Y)¥n");
  exit;
}
UV = (int *)malloc( 27648 );
if( UV == 0 ) {
  printf("malloc error(UV)¥n");
  exit;
}


//
// read & set  Initial temperature value
//
*avetemp = readtemp();


//
// Initial Time & copy to BOOTTIME
//
i = ntp( datetime );
while( i < 0 ) {              // Error 時には再取得を試みる
  sleep(500);
  i = ntp( datetime );
  printf("NTP retry¥n");
}

//
// ntp によって得られた時刻を RTC にセット
```

```
//
set_rtc( datetime );

//
// boottime に現時刻をブート時刻として格納
//
bcopy( datetime, boottime, sizeof( dateformat ) );


//
// Opening Message to LCD     LCD に IP アドレスとブート時刻を表示
//
fprintf(*lcdfid, "%d.%d.%d.%d¥nBt:",// IPAddress,Boottime => LCDout
        ((myaddr.sin_addr>>24)&0xff), ((myaddr.sin_addr>>16)&0xff),
        ((myaddr.sin_addr>>8) &0xff), ((myaddr.sin_addr)&(0xff) ) );
fprintf( *lcdfid, "%02d/", boottime->m );
fprintf( *lcdfid, "%02d ", boottime->d );
fprintf( *lcdfid, "%02d:", boottime->hh );
fprintf( *lcdfid, "%02d", boottime->mm );


//
// Opening Messsage to LED    いわゆるオープニングデモ
//
opening( 3, 50 );                    // Opening Indicater (3times, wait 50)


//
// Initialize End
//
*init = 2;


//
// ループに入る前の初期値設定
//
old_d = 100;
frame = 0;


//
// Mainloop
//
//    DIPsw4  ON: LoopEnd / OFF:continue

while( !DIPSW4 ) {
  printf("¥nFrame[%d]¥t%x%x%x%x¥n",frame, DIPSW1, DIPSW2, DIPSW3, DIPSW4 );

  //
  // Loop Start              LED : ●●ーーーーーー
  //
  PADR &= 0x7F;
  PADR |= 0xC0;

  //
  // 時刻を読み出して、先の温度測定から interval 時間過ぎてたら測定する
  //
  read_rtc( datetime );
  printf("¥t%02d/%02d:%02d ==== %02d/%02d:%02d¥n", datetime->d, datetime->hh, datetime->mm,
         old_d, old_hh, old_mm );

  if( (old_d == 100 )                // 最初は必ず測定
      || (old_d != datetime->d)      // interval 時間過ぎてたら温度測定
      || (old_hh != datetime->hh)
      || ((old_mm + *interval) <= datetime->mm) ) {
```

```
//
// 温度測定                LED : ●○－－－－－－－
//
PADR &= 0x7F;

// 温度異常の比較値の表示
if( (bmptemp[144]) >= 0 ) {
  printf("¥tTEMP(%d)[%d]", bmptemp[144], old_temp);
}else{
  printf("¥t            ");               // 一番最初の場合は表示しない
  old_temp = -1;
}

// 温度測定  →  bmptemp[i]に格納
*avetemp = readtemp();
i = ( (bmptemp[144]+1) % 144 );
bmptemp[ i ] = *avetemp;
bmptemp[144] = i;
printf(" -> (%d)[%d]¥n", bmptemp[144], bmptemp[ i ] );

//
// 異常温度検出
//                                        最初、DIPSW3 時、温度 Alert 中は比較しない
if( (old_temp >= 0 ) && (!DIPSW3) && ( *tempA < 0 ) ) {
  int diff = ( *avetemp - old_temp);
  if( diff < 0 ) diff = -diff;
  if( diff >= ALERT_TEMP ) {              // 差が ALERT_TEMP 以上だったら異常と判断
    printf("¥t***¥n");
    printf("¥t***Alert TEMP [%d]¥n", bmptemp[ bmptemp[144] ] );
    printf("¥t***¥n");

    *tempA = bmptemp[ bmptemp[144] ];     // 異常値は*tempA にコピー
    timer8x2( 0, 3 );                     // 異常時は赤 LED を点滅させる

    //
    // Alert Mail を送る
    //
    if( !DIPSW1 ) {       // DIPSW1 on ならメール送らない
      char mes[256];
      sprintf( mes , "Alert : Temperature¥r¥n %d.%d degree Celsius¥r¥nhttp://%d.%d.%d.%d/cgi¥r¥n",
               (*tempA)/10, ((*tempA+5)%10),
               (myaddr.sin_addr & 0xFF000000)>>24,
               (myaddr.sin_addr & 0x00FF0000)>>16,
               (myaddr.sin_addr & 0x0000FF00)>>8,
               (myaddr.sin_addr & 0x000000FF) );
      mailsend( POPbeforeSMTP, mes, datetime );
    }else{
      printf(" ------ skip ... alert-mail¥n");
    }
    //
  }
}else{
  if( DIPSW3 ) {                  // DIPSW3 on の時 Alert OFF にする
    if( *tempA > 0 ) {            // 温度 Alert 中？
      *tempA = -1;                        // Temp Alert OFF
      if( bmpA[0] == 0x00 ) {             // Image Alert 中だったら LED の点滅は消さない
        timer8x2( 0, 0 );                 // Image Alert 中でなかったら LED 消灯
      }
    }
  }
}
old_temp = *avetemp;      // 現温度値を次の比較用に格納

//
// 温度値に時間属性をつける
//
```

```
  if( old_d != 100 ) {              // 最初だったら属性つけません
    if( old_d != datetime->d ) {
      bmptemp[ bmptemp[144] ] |= 0x040000;          // 日付が変わった時(0時) (04)
    }else{
      if( old_hh != datetime->hh ) {                // 時間が変わった時
        if( datetime->hh == 12 ) {
          bmptemp[ bmptemp[144] ] |= 0x020000;              // 正午 (02)
        }else{
          bmptemp[ bmptemp[144] ] |= 0x010000;              // 毎時 (01)
        }
      }
    }
  }

  // 属性が付いた時(=毎時ごと)に ntp して RTC の狂いを修正します
  if( (bmptemp[ bmptemp[144] ] & 0xff0000 ) != 0 ) { // 毎時間毎に ntp
    i = ntp( datetime );
    while( i < 0 ) {        // Error 時には再取得を試みる
      sleep(100);
      i = ntp( datetime );
      printf("NTP retry¥n");
    }
    set_rtc( datetime );
  }

  // 次に温度測定する時間を知るために、現在の温度測定した時間を格納
  old_d  = datetime->d;
  old_hh = datetime->hh;
  old_mm = datetime->mm;
}
//
// 温度測定  ー  終了
//




//
// 画像読み込み             LED : ○●ーーーーーー
//
PADR &= 0x3F;
PADR |= 0x40;


// 現在の画像格納 Index を調べて
k = *imageid;
// 次の Index へ格納するようにして画像取り込み
treva(  (k+1)%3, frame, YUV );

//
// 異常画像検出      調査座標:[46:34]-[49:37]
//                                        最初、DIPSW2 時、画像 Alert 中は比較しない
if( (oldYUV[0] > 0) && (!DIPSW2) && ( bmpA[0] == 0x00) ) {
  //
  //        Y の差分計算
  //
  unsigned long sumY = 0;           // Y の差分合計用変数クリア
  // printf("¥nY:¥t");
  p = 0;
  for(j=0; j<8; j++) {
    for(i=0; i<8; i++) {
      signed int x = oldYUV[p]-YUV[p];
      // printf("%4d ", x );
      if( x < 0 ) x = -x;           // 誤差を絶対値にする
      x >>= 3;                      // 8 以下の誤差を省く
      sumY += x;
```

```
      p++;
    }
    // printf(" : %5d¥n", sumY);
  }
  // printf(" Y:[ %5d ] ¥n", sumY);

  //
  //        VU の差分計算
  //
  unsigned long sumVU=0;            // VU の差分合計用変数クリア
  // printf("VU:¥t");
  for(j=0; j<8; j++) {
    for(i=0; i<8; i++) {
      signed int x = oldYUV[p]-YUV[p];
      // printf("%4d ", x );
      if( x < 0 ) x = -x;           // 誤差を絶対値にする
      x >>= 3;                      // 8 以下の誤差を省く
      sumVU += x;
      p++;
    }
    // printf(" : %5d¥n", sumVU);
  }
  // printf( " VU:[ %5d ]¥n¥n", sumVU);

  printf("¥n¥tY:[ %5d ]  VU:[ %5d ] ¥n", sumY, sumVU);


  if( ( sumY >= ALERT_YUV ) || ( sumVU >= ALERT_YUV ) ) {   // ALERT_YUV 以上なら異常と判断
    printf("¥t***¥n");
    printf("¥t***Alert Image¥n" );
    printf("¥t***¥n");
    switch( k ) {                           // 異常画像を Alert 画像として格納
    case 0 : bcopy( bmp1, bmpA, IMG_SIZE ); break;
    case 1 : bcopy( bmp2, bmpA, IMG_SIZE ); break;
    case 2 : bcopy( bmp0, bmpA, IMG_SIZE ); break;
    }
    timer8x2( 0, 3 );                       // 異常時は赤 LED を点滅させる

    //
    // Alert Mail を送る
    //
    if( !DIPSW1 ) { // DIPSW1 on ならメール送らない
      char mes[256];
      sprintf( mes , "Alert : Image¥r¥nhttp://%d.%d.%d.%d/cgi¥r¥n",
               (myaddr.sin_addr & 0xFF000000)>>24,
               (myaddr.sin_addr & 0x00FF0000)>>16,
               (myaddr.sin_addr & 0x0000FF00)>>8,
               (myaddr.sin_addr & 0x000000FF) );
      mailsend( POPbeforeSMTP, mes, datetime );
    }else{
      printf(" ------ skip ... alert-mail¥n");
    }
    //
  }
}else{
  if( DIPSW2 ) {                    // DIPSW2 on の時 Alert OFF にする
    if( bmpA[0] != 0x00 ) {         // ImageAlert 中？
      bmpA[0] = (char)0x00;
      if( *tempA < -0 ) {                   // Temp Alert 中だったら LED の点滅は消さない
        timer8x2( 0, 0 );                   // Temp Alert 中でなかったら LED 消灯
      }
    }
  }
}

// 比較用画像として新画像を格納
```

```
    bcopy( YUV, oldYUV, 512 );  // 4(int) * 8 * 8 * 2(Y,UV) = 512

    (*imageid) = (k+1)%3;       // 画像 Index 更新



    //
    // 画像処理修了
    //
    PADR &= 0x0F;       //      LED : ○○－－－－－－

    P4DR |= 0x80;
    sleep( 2000 );      // とりあえずちょっとだけおやすみをいれてみる
    P4DR &= 0x3F;

    frame++;
  }

  close( *lcdfid );
  exit();
}

#include "Sub/misc.c"
#include "Sub/init_port.c"
#include "Sub/init_network.c"
#include "Sub/init_timer8.c"
#include "Sub/opening.c"
#include "Sub/treva.c"
#include "Sub/readtemp.c"
#include "Sub/writeyuvfile.c"
#include "Sub/writeppmfile.c"
#include "Sub/writebmpfile.c"
#include "Sub/ntp.c"
#include "Sub/timer8x2.c"
#include "Sub/pop3.c"
#include "Sub/mailsend.c"

#include "Sub/i2c.c"
#include "Sub/rtc.c"
```

付録[D] プログラムソースリスト － 共通用定義ファイル

Lib/define.h　　各種定義ヘッダファイル

```
/* port address of internal registers */

#define MDCR            (*(volatile unsigned char *)0xfee011)
#define SYSCR           (*(volatile unsigned char *)0xfee012)
#define BRCR            (*(volatile unsigned char *)0xfee013)
#define ISCR            (*(volatile unsigned char *)0xfee014)
#define IER             (*(volatile unsigned char *)0xfee015)
#define ISR             (*(volatile unsigned char *)0xfee016)
#define IPRA            (*(volatile unsigned char *)0xfee018)
#define IPRB            (*(volatile unsigned char *)0xfee019)
#define CSCR            (*(volatile unsigned char *)0xfee01f)
#define ABWCR           (*(volatile unsigned char *)0xfee020)
#define ASTCR           (*(volatile unsigned char *)0xfee021)
#define WCRH            (*(volatile unsigned char *)0xfee022)
#define WCRL            (*(volatile unsigned char *)0xfee023)
#define BCR             (*(volatile unsigned char *)0xfee024)
#define RAMCR           (*(volatile unsigned char *)0xfee077)
#define FLMCR           (*(volatile unsigned char *)0xfee030)
#define FLMCR1          (*(volatile unsigned char *)0xfee030)
#define FLMCR2          (*(volatile unsigned char *)0xfee031)
#define EBR             (*(volatile unsigned char *)0xfee032)
#define EBR1            (*(volatile unsigned char *)0xfee032)
#define EBR2            (*(volatile unsigned char *)0xfee033)
#define FLMSR           (*(volatile unsigned char *)0xfee07d)
#define TSTR            (*(volatile unsigned char *)0xffff60)
#define TSNC            (*(volatile unsigned char *)0xffff61)
#define TMDR            (*(volatile unsigned char *)0xffff62)
#define TOLR            (*(volatile unsigned char *)0xffff63)
#define TISRA           (*(volatile unsigned char *)0xffff64)
#define TISRB           (*(volatile unsigned char *)0xffff65)
#define TISRC           (*(volatile unsigned char *)0xffff66)
#define TCR0            (*(volatile unsigned char *)0xffff68)
#define TIOR0           (*(volatile unsigned char *)0xffff69)
#define TCNT0           (*(volatile unsigned short *)0xffff6a)
#define GRA0            (*(volatile unsigned short *)0xffff6c)
#define GRB0            (*(volatile unsigned short *)0xffff6e)
#define TCR1            (*(volatile unsigned char *)0xffff70)
#define TIOR1           (*(volatile unsigned char *)0xffff71)
#define TCNT1           (*(volatile unsigned short *)0xffff72)
#define GRA1            (*(volatile unsigned short *)0xffff74)
#define GRB1            (*(volatile unsigned short *)0xffff76)
#define TCR2            (*(volatile unsigned char *)0xffff78)
#define TIOR2           (*(volatile unsigned char *)0xffff79)
#define TCNT2           (*(volatile unsigned short *)0xffff7a)
#define GRA2            (*(volatile unsigned short *)0xffff7c)
#define GRB2            (*(volatile unsigned short *)0xffff7e)
#define TCSR            (*(volatile unsigned short *)0xffff8c)
#define TCNT            (*(volatile unsigned short *)0xffff8d)

#define T8TCR0          (*(volatile unsigned char *)0xffff80)
#define T8TCR1          (*(volatile unsigned char *)0xffff81)

#define SMR0            (*(volatile unsigned char *)0xffffb0)
#define BRR0            (*(volatile unsigned char *)0xffffb1)
#define SCR0            (*(volatile unsigned char *)0xffffb2)
#define TDR0            (*(volatile unsigned char *)0xffffb3)
#define SSR0            (*(volatile unsigned char *)0xffffb4)
#define RDR0            (*(volatile unsigned char *)0xffffb5)
```

```
#define SMR1            (*(volatile unsigned char *)0xffffb8)
#define BRR1            (*(volatile unsigned char *)0xffffb9)
#define SCR1            (*(volatile unsigned char *)0xffffba)
#define TDR1            (*(volatile unsigned char *)0xffffbb)
#define SSR1            (*(volatile unsigned char *)0xffffbc)
#define RDR1            (*(volatile unsigned char *)0xffffbd)
#define SMR1            (*(volatile unsigned char *)0xffffb8)
#define SMR2            (*(volatile unsigned char *)0xffffc0)
#define BRR2            (*(volatile unsigned char *)0xffffc1)
#define SCR2            (*(volatile unsigned char *)0xffffc2)
#define TDR2            (*(volatile unsigned char *)0xffffc3)
#define SSR2            (*(volatile unsigned char *)0xffffc4)
#define RDR2            (*(volatile unsigned char *)0xffffc5)

#define P1DDR           (*(volatile unsigned char *)0xfee000)
#define P2DDR           (*(volatile unsigned char *)0xfee001)
#define P3DDR           (*(volatile unsigned char *)0xfee002)
#define P4DDR           (*(volatile unsigned char *)0xfee003)
#define P5DDR           (*(volatile unsigned char *)0xfee004)
#define P6DDR           (*(volatile unsigned char *)0xfee005)
#define P7DDR           (*(volatile unsigned char *)0xfee006)
#define P8DDR           (*(volatile unsigned char *)0xfee007)
#define P9DDR           (*(volatile unsigned char *)0xfee008)
#define PADDR           (*(volatile unsigned char *)0xfee009)
#define PBDDR           (*(volatile unsigned char *)0xfee00a)
#define P2PCR           (*(volatile unsigned char *)0xfee03c)
#define P4PCR           (*(volatile unsigned char *)0xfee03e)
#define P5PCR           (*(volatile unsigned char *)0xfee03f)
#define P1DR            (*(volatile unsigned char *)0xffffd0)
#define P2DR            (*(volatile unsigned char *)0xffffd1)
#define P3DR            (*(volatile unsigned char *)0xffffd2)
#define P4DR            (*(volatile unsigned char *)0xffffd3)
#define P5DR            (*(volatile unsigned char *)0xffffd4)
#define P6DR            (*(volatile unsigned char *)0xffffd5)
#define P7DR            (*(volatile unsigned char *)0xffffd6)
#define P8DR            (*(volatile unsigned char *)0xffffd7)
#define P9DR            (*(volatile unsigned char *)0xffffd8)
#define PADR            (*(volatile unsigned char *)0xffffd9)
#define PBDR            (*(volatile unsigned char *)0xffffda)

#define ADDRA           (*(volatile unsigned short *)0xfffffe0)
#define ADDRB           (*(volatile unsigned short *)0xfffffe2)
#define ADDRC           (*(volatile unsigned short *)0xfffffe4)
#define ADDRD           (*(volatile unsigned short *)0xfffffe6)
#define ADDRAH          (*(volatile unsigned char *)0xfffffe0)
#define ADDRAL          (*(volatile unsigned char *)0xfffffe1)
#define ADDRBH          (*(volatile unsigned char *)0xfffffe2)
#define ADDRBL          (*(volatile unsigned char *)0xfffffe3)
#define ADDRCH          (*(volatile unsigned char *)0xfffffe4)
#define ADDRCL          (*(volatile unsigned char *)0xfffffe5)
#define ADDRDH          (*(volatile unsigned char *)0xfffffe6)
#define ADDRDL          (*(volatile unsigned char *)0xfffffe7)
#define ADCSR           (*(volatile unsigned char *)0xfffffe8)
#define ADCR            (*(volatile unsigned char *)0xfffffe9)

#define RTCOR           (*(volatile unsigned char *)0xfee02a)
#define RTMCSR          (*(volatile unsigned char *)0xfee028)
#define DRCRA           (*(volatile unsigned char *)0xfee026)
#define DRCRB           (*(volatile unsigned char *)0xfee027)

#define TCR0_8          (*(volatile unsigned char *)0xffff80)
#define TCR1_8          (*(volatile unsigned char *)0xffff81)
#define TCSR0_8         (*(volatile unsigned char *)0xffff82)
#define TCSR1_8         (*(volatile unsigned char *)0xffff83)
#define TCORA0_8        (*(volatile unsigned char *)0xffff84)
#define TCORA1_8        (*(volatile unsigned char *)0xffff85)
```

```
#define TCORB0_8        (*(volatile unsigned char *)0xffff86)
#define TCORB1_8        (*(volatile unsigned char *)0xffff87)
#define TCNT0_8         (*(volatile unsigned char *)0xffff88)
#define TCNT1_8         (*(volatile unsigned char *)0xffff89)
#define TCR2_8          (*(volatile unsigned char *)0xffff90)
#define TCR3_8          (*(volatile unsigned char *)0xffff91)
#define TCSR2_8         (*(volatile unsigned char *)0xffff92)
#define TCSR3_8         (*(volatile unsigned char *)0xffff93)
#define TCORA2_8        (*(volatile unsigned char *)0xffff94)
#define TCORA3_8        (*(volatile unsigned char *)0xffff95)
#define TCORB2_8        (*(volatile unsigned char *)0xffff96)
#define TCORB3_8        (*(volatile unsigned char *)0xffff97)
#define TCNT2_8         (*(volatile unsigned char *)0xffff98)
#define TCNT3_8         (*(volatile unsigned char *)0xffff99)

#define MAR0AR          (*(volatile unsigned char *)0xffff20)
#define MAR0AE          (*(volatile unsigned char *)0xffff21)
#define MAR0AH          (*(volatile unsigned char *)0xffff22)
#define MAR0AL          (*(volatile unsigned char *)0xffff23)
#define IOAR0A          (*(volatile unsigned char *)0xffff26)
#define ETCR0AH         (*(volatile unsigned char *)0xffff24)
#define ETCR0AL         (*(volatile unsigned char *)0xffff25)
#define DTCR0A          (*(volatile unsigned char *)0xffff27)
#define MAR0BR          (*(volatile unsigned char *)0xffff28)
#define MAR0BE          (*(volatile unsigned char *)0xffff29)
#define MAR0BH          (*(volatile unsigned char *)0xffff2a)
#define MAR0BL          (*(volatile unsigned char *)0xffff2b)
#define IOAR0B          (*(volatile unsigned char *)0xffff2e)
#define ETCR0BH         (*(volatile unsigned char *)0xffff2c)
#define ETCR0BL         (*(volatile unsigned char *)0xffff2d)
#define DTCR0B          (*(volatile unsigned char *)0xffff2f)
#define MAR1AR          (*(volatile unsigned char *)0xffff30)
#define MAR1AE          (*(volatile unsigned char *)0xffff31)
#define MAR1AH          (*(volatile unsigned char *)0xffff32)
#define MAR1AL          (*(volatile unsigned char *)0xffff33)
#define IOAR1A          (*(volatile unsigned char *)0xffff36)
#define ETCR1AH         (*(volatile unsigned char *)0xffff34)
#define ETCR1AL         (*(volatile unsigned char *)0xffff35)
#define DTCR1A          (*(volatile unsigned char *)0xffff37)
#define MAR1BR          (*(volatile unsigned char *)0xffff38)
#define MAR1BE          (*(volatile unsigned char *)0xffff39)
#define MAR1BH          (*(volatile unsigned char *)0xffff3a)
#define MAR1BL          (*(volatile unsigned char *)0xffff3b)
#define IOAR1B          (*(volatile unsigned char *)0xffff3e)
#define ETCR1BH         (*(volatile unsigned char *)0xffff3c)
#define ETCR1BL         (*(volatile unsigned char *)0xffff3d)
#define DTCR1B          (*(volatile unsigned char *)0xffff3f)


// TPC
#define TPMR            (*(volatile unsigned char *)0xFFFFA0)
#define TPCR            (*(volatile unsigned char *)0xFFFFA1)
#define NDERB           (*(volatile unsigned char *)0xFFFFA2)
#define NDERA           (*(volatile unsigned char *)0xFFFFA3)
#define NDRB            (*(volatile unsigned char *)0xFFFFA4)
#define NDRA            (*(volatile unsigned char *)0xFFFFA5)


//
#define DEVICE 0
// NTPSERVER = ntp.nc.u-tokyo.ac.jp
#define NTP_SERVER      IPADDR(130,69,251,23)
#define NTP_PORT        123

typedef unsigned long time_t;
```

```
typedef struct {
  unsigned int l_ui;
  unsigned int l_uf;
} l_fp;

struct ntp_pkt {
  unsigned char li_vn_mode;
  unsigned char stratum;
  unsigned char ppoll;
  unsigned char precision;
  unsigned int  rootdelay;
  unsigned int  rootdispersion;
  unsigned int  refid;
  l_fp          reftime;
  l_fp          org;
  l_fp          rec;
  l_fp          xmt;
};

typedef struct {
  unsigned int y;
  unsigned int m;
  unsigned int d;
  unsigned int w;
  unsigned int hh;
  unsigned int mm;
  unsigned int ss;
} dateformat;

typedef union{
  unsigned short stime[2];              // s[0] s[1]
  unsigned int   ltime;                 // l
} long_short ;

typedef union{
  char  c[4];
  long  l;
} long_char;

typedef union{
  char   c[2];
  short  s;
} short_char;

#define STDOUT          0
#define STDIN           1

#define IMG_WIDTH       96
#define IMG_HEIGHT      72
#define IMG_SIZE        20736   // 96*72*3

#define BMP_WIDTH       108
#define BMP_HEIGHT      144

#define TEMP_REVISE     (-15)   //温度補正値 -1.5deg
#define TIME_REVISE     1       //ntp 補正値  +1sec

#define SHKEY_INIT      1
#define SHKEY_BOOT      2
#define SHKEY_DATE      3
#define SHKEY_INTERVAL  4

#define SHKEY_LCDMES    5
#define SHKEY_LED       6
#define SHKEY_LCDFID    7
#define SHKEY_AVETEMP   8
```

```
#define SHKEY_TEMP      9

#define SHKEY_IMAGEID   10
#define SHKEY_BMP0      11
#define SHKEY_BMP1      12
#define SHKEY_BMP2      13
#define SHKEY_BMPA      14      // Alert Image
#define SHKEY_TEMPA     15      // Alert Temperature


#define TCNT0_8_short           (*(volatile unsigned short *)0xffff88)
#define TCNT2_8_short           (*(volatile unsigned short *)0xffff98)

#define TIMEOVER        180

#define RTC_WRITE       0xA2
#define RTC_READ        0xA3


#define RTC_CTRL1       0x00
#define RTC_CTRL2       0x01
#define RTC_SEC         0x02
#define RTC_MIN         0x03
#define RTC_HOUR        0x04
#define RTC_DAY         0x05
#define RTC_WEEK        0x06
#define RTC_MONTH       0x07
#define RTC_YEAR        0x08
#define RTC_ALM_M       0x09
#define RTC_ALM_H       0x0A
#define RTC_ALM_D       0x0B
#define RTC_ALM_W       0x0C
#define RTC_CLKOUT      0x0D
#define RTC_TIMERCTL    0x0E
#define RTC_TIMER       0x0F
```

付録[E] プログラムソースリスト – CGI プログラム用ソースファイル

cgi.c	CGI 処理用プログラム

```
#include <mes.h>
#include <string.h>
#include "Lib/define.h"

#include "Sub/endian.h"
int     timer8x2( int, int );

//
// CGI 処理プログラム
//
//
//

int main( int argc, char *argv[] ) {

#include "Sub/share_memory.h"

  unsigned char oldP4DR;

  int i;
  int id;
  //
  //    共有メモリ設定
  //
#include "Sub/share_memory.c"
  int *bmptemp_idx;
  bmptemp_idx = bmptemp+144;

  if( *init != 2 ) {    // イニシャライズされてる？
    printf("<HTML><BODY>");
    printf("Alert : MainTask not running...");
    printf("</BODY></HTML>");
    return;
  }                     // イニシャライズされてないと動かせません

    unsigned short BITMAP_ID = 0x424d;
    struct {
      unsigned long  fSize;
      unsigned short Reserved1;
      unsigned short Reserved2;
      unsigned long  OffBits;
      unsigned long  Size;
      long           Width;
      long           Height;
      unsigned short Planes;
      unsigned short BitCount;
      unsigned long  Compression;
      unsigned long  SizeImage;
      long           XPixPerMeter;
      long           YPixPerMeter;
      unsigned long  ClrUsed;
      unsigned long  ClrImporant;
    } BITMAP;

  char tmp[32];

  //
  // Treva 画像表示
```

```
//
if( cgi_value( argv[1], "BMP", tmp, 16 ) == 0 ) {
  //                                                  96*72*3+54 = 20790
  BITMAP.fSize        = toLittle4(20790);            // 0000 5136
  BITMAP.Reserved1    = (short)0;
  BITMAP.Reserved2    = (short)0;
  BITMAP.OffBits      = toLittle4(54);               // 0000 0036

  BITMAP.Size         = toLittle4(40);               // 0000 0028
  BITMAP.Width        = toLittle4(IMG_WIDTH);        // 0000 0060
  BITMAP.Height       = toLittle4(IMG_HEIGHT);       // 0000 0048
  BITMAP.Planes       = toLittle2(1);
  BITMAP.BitCount     = toLittle2(24);
  BITMAP.Compression  = 0;
  BITMAP.SizeImage    = toLittle4(IMG_SIZE);
  BITMAP.XPixPerMeter = 0;
  BITMAP.YPixPerMeter = 0;
  BITMAP.ClrUsed      = 0;
  BITMAP.ClrImporant  = 0;

  write( STDOUT, (char *)&BITMAP_ID,  2 );
  write( STDOUT, (char *)&BITMAP, sizeof(BITMAP) );

  char zebra[6] = { 0x00, 0x00, 0x00, 0xff, 0xff, 0xff };

  if( tmp[0] == 'A' ) {               // Alert 画像表示
    write( STDOUT, bmpA , IMG_SIZE );
  }else {
    switch( *imageid ) {
    case -1 :
      for(i=0; i<3456; i++) { // まだ準備できてないときはしましま模様
        write( STDOUT, zebra , 6 );
      }
      break;
    case 0:
      write( STDOUT, bmp0 , IMG_SIZE );
      break;
    case 1:
      write( STDOUT, bmp1 , IMG_SIZE );
      break;
    case 2:
      write( STDOUT, bmp2 , IMG_SIZE );
      break;
    }
  }
  //    oldP4DR = (P4DR & 0xc0);
  //    fprintf( *lcdfid, "C:%d ", *imageid );
  //    P4DR |= oldP4DR;
  return;
}

//
// 温度値表示
//
if( cgi_value( argv[1], "TEMP", tmp, 16 ) == 0 ) {
                                                      // 54 + 4*16 + 144*10 + 2 = 1560
                                                      // 54 + 4*16 + 144*8  + 2 = 1272
                                                      // 54 + 4*16 + 144*6  + 2 = 984
                                                      // 54 + 4*16 + 144*4  + 2 = 696
  BITMAP.fSize        = toLittle4(1272);             // 0000 0618
  BITMAP.Reserved1    = (short)0;
  BITMAP.Reserved2    = (short)0;
  BITMAP.OffBits      = toLittle4(118);              // 0000 0076  (54+4*16)

  BITMAP.Size         = toLittle4(40);               // 0000 0028
  BITMAP.Width        = toLittle4(BMP_WIDTH);        // 0000 006C
```

```
BITMAP.Height       = toLittle4(BMP_HEIGHT);        // 0000 0090
BITMAP.Planes       = toLittle2(1);
BITMAP.BitCount     = toLittle2(8);                 // 256color
BITMAP.Compression  = toLittle4(1);                 // BI_RLE8
BITMAP.SizeImage    = toLittle4(1154);              // 144*4+2= 578= 0242
                                                    // 144*6+2= 866= 0362
                                                    // 144*8+2=1154= 0482
BITMAP.XPixPerMeter = 0;
BITMAP.YPixPerMeter = 0;
BITMAP.ClrUsed      = toLittle4(16);                // Palette
BITMAP.ClrImporant  = 0;

char palette[] = {
  0x00, 0x00, 0x00, 0,      //  0 Black
  0xa0, 0x00, 0x00, 0,      //  1 DarkBlue
  0xFF, 0x00, 0x00, 0,      //  2 Blue
  0xFF, 0xFF, 0x00, 0,      //  3 Cyan
  0x80, 0xFF, 0x00, 0,      //  4
  0x00, 0xFF, 0x00, 0,      //  5 Green
  0x00, 0xFF, 0xA0, 0,      //  6
  0x00, 0xFF, 0xFF, 0,      //  7 Yellow
  0x00, 0xC0, 0xFF, 0,      //  8 Orange1
  0x00, 0x80, 0xFF, 0,      //  9 Orange2
  0x00, 0x00, 0xFF, 0,      // 10 Red
  0xFF, 0x00, 0xFF, 0,      // 11 Magenta
  0x60, 0x60, 0x60, 0,      // 12 Gray1
  0x80, 0x80, 0x80, 0,      // 13 Gray2
  0xA0, 0xA0, 0xA0, 0,      // 14 Gray3
  0xff, 0xff, 0xff, 0};     // 15 White

int fp;

if( tmp[0] == '0' ) {               // table 表示    ファイルにも落としておく
  fp = open("temp.bmp", 0);
  write( fp, (char *)&BITMAP_ID,  2 );
  write( fp, (char *)&BITMAP, sizeof(BITMAP) );
  write( fp, palette,  36 );
  write( fp, palette,  28 );        // Dummy Palette
}else{                              // グラフ表示
  write( STDOUT, (char *)&BITMAP_ID,  2 );
  write( STDOUT, (char *)&BITMAP, sizeof(BITMAP) );
  write( STDOUT, palette,  64 );
}

id = *bmptemp_idx;
if( tmp[0] == '0' ) {       // Table 表示    Header
  printf("<html><body>id = %d<br><table border=1>", id);
}

for(i=1; i<145; i++) {
  int t    = bmptemp[ ((id+i)%144) ];
  int tid  = t & 0xff0000;
  int temp = t & 0x00FFFF;  // value(temp*10)
  int tc;
  switch(tid) {
  case 0x040000:
  case 0x010000: tc = 15; break;    // day/hour --- White
  case 0x020000: tc = 14; break;    // noon     --- Gray
  default :      tc = 0;  break;    // other    --- Black
  }
if( tmp[0] == '0' ) {       // Table 表示
  printf("<tr><th>%d(%d)</th><th>%d</th><td>%d</td><td>%d</td>",
         i, (id+i)%144, temp, 6, tc );
  fprintf(fp, "%c%c", 6, tc);
}else{                      // グラフ表示  時間表示用 6 ピクセル分
  printf("%c%c", 6, tc);
```

```
    }

      //温度をバーグラフ化
      temp = (temp+3)/5;        // 0-50deg => 0-100pixel 四捨五入

      t = ((temp+9)/10);        // パレット割り当て
      if( t>10 ) t = 11;

      if( temp < 1 ) temp = 1;
      else if(temp > 100 ) temp = 100;

      if( tmp[0] == '0' ) {     // Table 表示    温度値表示
        printf("<td>%d</td><td>%d</td>", temp, t);
        fprintf(fp, "%c%c", temp, t );
      }else{                    // グラフ表示   温度バー
        printf( "%c%c", temp, t );
      }

      // 右半分の残りバー
      if( tid == 0x010000 ) tc = 0;
      if( tmp[0] == '0' ) {     // Table 表示    参考までに属性を表示
        printf("<td>%d</td><td>%d</td></tr>", 102-temp, tc);
        fprintf(fp, "%c%c%c%c", (102-temp), tc, 0, 0 );
      }else{                    // グラフ表示   右の残りを表示
        printf( "%c%c%c%c", (102-temp), tc, 0, 0 );
      }
    }

    if( tmp[0] == '0' ) {       // Table 表示    HTML footer
      printf("</table></body></html>");
      fprintf(fp, "%c%c", 0, 1 );
      close(fp);
    }else{
      printf("%c%c", 0, 1 );    // グラフ表示   終了
    }

    return;
  }


  dateformat *lcddate;
  lcddate = (dateformat *)(lcdmes+20);

  int myip;
  myip = getip( 0 );

  //
  // LCD 表示
  //
  if( cgi_value( argv[1], "LCD", tmp, 16) == 0 ) {      // LCD
    strcpy(lcdmes, tmp);
    bcopy( datetime, lcddate, sizeof( dateformat ));
    ioctl(*lcdfid, 0, LCD_CLEAR);
    oldP4DR = (P4DR & 0xc0);
    fprintf( *lcdfid, "¥n%02d/%02d %02d:%02d¥n%s", lcddate->m, lcddate->d,
             lcddate->hh, lcddate->mm, lcdmes);
    P4DR |= oldP4DR;
    printf("<html><head><meta http-equiv=¥"Refresh¥"
content=¥"0;URL=http://%d.%d.%d.%d/cgi¥"></head></html>",
           ((myip>>24)&0xff), ((myip>>16)&0xff),((myip>>8) &0xff),((myip)&(0xff)));
    return;

  } else {
    //
    // LED 制御 ( Green LED )
    //
```

```
    if( cgi_value( argv[1], "LED", tmp, 16) == 0 ) {    // LED
      if( ('0' <= tmp[0] ) && (tmp[0] <= '3' ) ) {
        *led = tmp[0]-'0';
        timer8x2( 2, *led );
      }
      printf("<html><head><meta http-equiv=¥"Refresh¥"
content=¥"0;URL=http://%d.%d.%d.%d/cgi¥"></head></html>",
             ((myip>>24)&0xff), ((myip>>16)&0xff),((myip>>8) &0xff),((myip)&(0xff)));
      return;
    } else {
      //
      // Interval time 変更
      //
      if( cgi_value( argv[1], "TIME", tmp, 16) == 0 ) { // Interval Time
        switch(tmp[0]) {
        case '0':       *interval =  1;break;           // 0 :  1min
        case '1':       *interval =  3;break;           // 1 :  3min
        case '2':       *interval =  5;break;           // 2 :  5min
        case '3':       *interval = 10;break;           // 3 : 10min
        }
        printf("<html><head><meta http-equiv=¥"Refresh¥"
content=¥"0;URL=http://%d.%d.%d.%d/cgi¥"></head></html>",
               ((myip>>24)&0xff), ((myip>>16)&0xff),((myip>>8) &0xff),((myip)&(0xff)));
        return;
      } else {
        //
        // Alert Clear
        //
        if( cgi_value( argv[1], "CLEAR", tmp, 16) == 0 ) {      // Alert Clear
          bmpA[0] = (char)0x00; // Status Clear
          *tempA = -1;
          timer8x2(0, 0);               // LED Off

          printf("<html><head><meta http-equiv=¥"Refresh¥"
content=¥"0;URL=http://%d.%d.%d.%d/cgi¥"></head></html>",
                 ((myip>>24)&0xff), ((myip>>16)&0xff),((myip>>8) &0xff),((myip)&(0xff)));
          return;

        }
      }
    }
  }

  dateformat now;
  bcopy( datetime, &now, sizeof( dateformat ));


  //
  // Status/Control 画面表示
  //
  printf("<html><head><style type=¥"text/css¥">¥n<!--¥n");
  printf("*  { text-align : center;}¥n");
  printf("h1 { font-size : 140%%}¥n");
  printf("caption { background-color: #fcf;¥nborder: 2px solid;¥nborder-color: #fef #c8c #c8c
#fef;¥nfont-size:120%%;}¥n");
  printf("th { background-color: #fcc;¥nborder: 2px solid;¥nborder-color: #fee #c88 #c88 #fee;}¥n");
  printf("td { background-color: #ffc;¥nborder: 2px solid;¥nborder-color: #ffe #cc8 #cc8
#ffe;}¥n-->¥n</style></head>¥n");

  printf("<body><h1>Home Security System</h1>");
  printf("<a href=¥"mailto:m03603@std.int-univ.com¥">m03603@std.int-univ.com</a><br><br>¥n");
  if( bmpA[0] != 0x00 ) {       // Image Alert
    printf("<font size=+2 color=¥"red¥">Camera ALERT!!!</font><br>");
    printf("<img border=2 src=¥"cgi?BMP=A¥"><br>¥n");
  }
  if( *tempA != -1 ) {          // temp Alert
```

```
  printf("<font size=+2 color=¥"red¥">Temperature ALERT!!!<br>");
  printf("%d.%d<br>¥n",(*tempA)/10, ((*tempA+5)%10) );
}
if( (bmpA[0] != 0x00 ) || ( *tempA != -1 ) ) {
  printf("<form action=cgi>¥n<input type=hidden name=CMD value=CLEAR><BR>¥n");
  printf("<input type=hidden name=CLEAR value=0>");
  printf("<input type=submit value=¥"Alert Clear¥"></form><br><br>");
}

printf("<table border=0><caption>Status</caption>¥n");
printf("<tr><th>Date</th><td>%04d/%02d/%02d %02d:%02d</td></tr>",
       now.y, now.m, now.d, now.hh, now.mm );
// Display Temp
i = *avetemp;
printf("<tr><th>Temp</th><td>%d.%d</td></tr>¥n",i/10, (i%10) );
// Display LCD
printf("<tr><th>LCD</th><td nowrap style=¥"{text-align:left;}¥">");
if( lcddate->y != 0 ) {
  printf("%02d/%02d %02d:%02d<BR> %s",
         lcddate->m, lcddate->d, lcddate->hh, lcddate->mm, lcdmes);
}
printf("</td></tr>¥n");
// Display LED
printf("<tr><th>LED</th><td>");
switch(*led) {
case 0:
  printf("<font color=black>Off</font>");
  break;
case 1:
  printf("<font color=green>On</font>");
  break;
case 2:
  printf("<font color=blue>Slow Blink</font>");
  break;
case 3:
  printf("<font color=blue>Fast Blink</font>");
  break;
default:
  printf("Unknown:");
}
printf("</td></tr>¥n");
// Display Camera Image
printf("<tr><th>Camera</th><td><img src=¥"cgi?BMP=0¥"></td></tr>");
// Display Temperture Graph
printf("<tr><th>Temp.</th><td><img src=¥"cgi?TEMP=1¥"></td></tr>");

// Display IntervalTime
printf("<tr><th>Interval</th><td>%d min</td></tr>", *interval);
// Display BootTime
printf("<tr><th>Boot</th><td>%04d/%02d/%02d %02d:%02d</td></tr></table>",
       boottime->y, boottime->m, boottime->d,
       boottime->hh, boottime->mm );

printf("<BR><BR><table border=0><caption>Command</caption><BR>");
// Command LCD
printf("<form action=cgi>¥n<input type=hidden name=CMD value=LCD><BR>¥n");
printf("<tr><th><input type=submit value=LCD></th>");
printf("<td><input type=text name=LCD size=16 maxlength=16 style=¥"{text-align:left;}¥"></td></tr></form>");
// Command LED
printf("<form action=cgi>¥n<tr><th><input type=submit value=LED></th><td>");
if( *led == 0 ) strcpy(tmp,"checked"); else strcpy(tmp,"");
printf("<input type=radio name=LED value=0 %s>Off¥n", tmp);
if( *led == 1 ) strcpy(tmp,"checked"); else strcpy(tmp,"");
printf("<input type=radio name=LED value=1 %s>On<BR>¥n", tmp);
if( *led == 2 ) strcpy(tmp,"checked"); else strcpy(tmp,"");
printf("Blink <input type=radio name=LED value=2 %s>Slow¥n", tmp);
```

```
  if( *led == 3 ) strcpy(tmp,"checked"); else strcpy(tmp,"");
  printf("<input type=radio name=LED value=3 %s>Fast</td></tr></form>¥n", tmp);
  // Command Interval Time
  printf("<form action=cgi>¥n<tr><th><input type=submit value=TIME></th><td>");
  if( *interval == 1 ) strcpy(tmp,"checked"); else strcpy(tmp,"");
  printf("<input type=radio name=TIME value=0 %s>1min", tmp);
  if( *interval == 3 ) strcpy(tmp,"checked"); else strcpy(tmp,"");
  printf("<input type=radio name=TIME value=1 %s>3min", tmp);
  if( *interval == 5 ) strcpy(tmp,"checked"); else strcpy(tmp,"");
  printf("<BR><input type=radio name=TIME value=2 %s>5min", tmp);
  if( *interval == 10) strcpy(tmp,"checked"); else strcpy(tmp,"");
  printf("<input type=radio name=TIME value=3 %s>10min</td></tr></table></form>¥n", tmp);

  printf("<BR><BR><a href=¥"cgi¥">Reload</a><BR><a href=¥"cgi?BMP=2¥">Treva Image</a><BR><a
href=¥"cgi?TEMP=1¥">Temp Image</a><BR><a href=¥"cgi?TEMP=0¥">Temp Table</a><BR>");

  /*
    for(i=0; i<144; i++) {
    printf("%d   %d<BR>", i, bmptemp[i] );
    }
  */

  printf("</body></html>");

}

#include "Sub/endian.c"
#include "Sub/timer8x2.c"
```

付録[F] プログラムソースリスト – 各種サブルーチン

Sub/share_memory.h　　共有変数定義ヘッダ

```
int *init;                          // shared memory : Init flag
                                    //        1 : now initializing
                                    //        2 : init done
                                    //        * : unknown
dateformat *boottime;               // shared memory : BOOT Time
dateformat *datetime;               // shared memory : now DateTime
int *interval;                      // shared memory : Interval time (min)
int *lcdfid;                        // shared memory : LCD fid
char *lcdmes;                       // shared memory : LCD settime & LCD Message
int *led;                           // shared memory : LED Status
unsigned int *avetemp;              // shared memory : Average tempature (*10)
int *bmptemp;                       // shared memory : Temp for BMP (*10)
int *imageid;                       // shared memory : ImageID -1 0 1 2
char *bmp0,*bmp1,*bmp2,*bmpA;       // shared memory : BMP Image #0/#1/#2/Alert
int *tempA;                         // shared memory : Alert Temperature
```

Sub/treva.h　　Sub/treva.c プロトタイプ宣言ヘッダ

```
int read_treva_bit( void );
int read_treva_byte( void );
int treva( int, int, int * );
```

Sub/misc.h　　Sub/misc.c プロトタイプ宣言ヘッダ

```
int   bcdtobin( int );
int   bintobcd( int );
long  toLittle4( long );
short toLittle2( short );
int   YUVtoRGB(int, int, int);
int   YUVtoRGB2(int, int, int);
```

Sub/rtc.h　　Sub/rtc.c プロトタイプ宣言ヘッダ

```
int     init_rtc( void );
int     set_rtc(  dateformat * );
int     read_rtc( dateformat * );
int     rtc_read( int );
int     rtc_write( int , int );
int     rtc_readmulti(int, int, int *);
int     rtc_writemulti(int, int, int *);
```

Sub/i2c.h　　Sub/i2c.c プロトタイプ宣言ヘッダ

```
int  i2c_start( int );
void i2c_stop( void );
int  i2c_write( int );
int  i2c_read( void );
void i2c_nak( void );
void i2c_ack( void );
```

Sub/share_memory.c　　共有変数確保プログラム

```
//
// Allocate shared memory
//

//
//      int *init
//      Initialize flag
//              1 : now initializing
//              2 : init done
//              * : unknown
id = shmget( SHKEY_INIT , sizeof(int) );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:init]¥n");
  return(-1);
}

init = (int *)shmat( id );
if( (int)init == -1 ) {
  printf("Shared memory allocate error [SHMAT:init]¥n");
  return(-1);
}

//      dateformat *boottime
//      Boot Time
//
//
id = shmget( SHKEY_BOOT , sizeof( dateformat ) );
  if( id == -1 ) {
    printf("Shared memory allocate error [SHMGET:BootTime]¥n");
    return(-1);
  }
boottime = (dateformat *)shmat( id );
if( (int)boottime == -1 ) {
  printf("Shared memory allocate error [SHMAT:BootTime¥n");
  return(-1);
}

//
//      dateformat *datetime
//      Not Time
//
id = shmget( SHKEY_DATE , sizeof( dateformat ) );
  if( id == -1 ) {
    printf("Shared memory allocate error [SHMGET:DateTime]¥n");
    return(-1);
  }
datetime = (dateformat *)shmat( id );
if( (int)datetime == -1 ) {
  printf("Shared memory allocate error [SHMAT:DateTime¥n");
  return(-1);
}

//
//      int *interval
//      Interval Time
//
id = shmget( SHKEY_INTERVAL, sizeof(int) );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:interval]¥n");
  return(-1);
}
```

```
interval = (int *)shmat( id );
if( (int)interval == -1 ) {
  printf("Shared memory allocate error [SHMAT:interval]¥n");
  return(-1);
}

//
//      int *lcdfid
//      FileID for LCD
//
id = shmget( SHKEY_LCDFID , sizeof(int) );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:lcdfid]¥n");
  return(-1);
}
lcdfid = (int *)shmat( id );
if( (int)lcdfid == -1 ) {
  printf("Shared memory allocate error [SHMAT:lcdfid]¥n");
  return(-1);
}

//
//      char *lcdmes
//              char lcdmes[20]         LCD Message
//              + dateformat            LCD Message Time
//
id = shmget( SHKEY_LCDMES , 20+sizeof(dateformat) );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:LCDmes]¥n");
  return(-1);
}
lcdmes = (char *)shmat( id );
if( (int)lcdmes == -1 ) {
  printf("Shared memory allocate error [SHMAT:LCDmes]¥n");
  return(-1);
}

//
//      int *led
//      LED Status      0 Off
//                      1 on
//                      2 Blink (Comming soon)
//
id = shmget( SHKEY_LED , sizeof(int) );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:LED]¥n");
  return(-1);
}
led = (int *)shmat( id );
if( (int)led == -1 ) {
  printf("Shared memory allocate error [SHMAT:LED]¥n");
  return(-1);
}

//
//      unsigned int *avetemp
//      Now Temperture  (x10)
//
id = shmget( SHKEY_AVETEMP , sizeof(unsigned int) );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:avetemp]¥n");
  return(-1);
}
avetemp = (int *)shmat( id );
if( (int)avetemp == -1 ) {
  printf("Shared memory allocate error [SHMAT:avetemp]¥n");
```

```
  return(-1);
}

//
//      int *bmptemp[145]                       Temperture Log
//              [0...143] : Temp log
//              [144]     : index
//
id = shmget( SHKEY_TEMP , 580 );        // 144*4=576
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:BMPtemp]¥n");
  return(-1);
}
bmptemp = (int *)shmat( id );
if( (int)bmptemp == -1 ) {
  printf("Shared memory allocate error [SHMAT:BMPtemp]¥n");
  return(-1);
}


//
//      int *imageid
//      Ready Image No  -1      not ready
//                      0       0:done
//                      1       1:done
//                      2       2:done
//
id = shmget( SHKEY_IMAGEID , sizeof(int) );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:imageid]¥n");
  return(-1);
}
imageid = (int *)shmat( id );
if( (int)imageid == -1 ) {
  printf("Shared memory allocate error [SHMAT:imageid]¥n");
  return(-1);
}

//
//      char *bmp0,*bmp1,*bmp2,*bmpA
//      Image Buffer
//              W96 x H72 x RGB3 = 20736(IMG_SIZE)
//              bmp0    : Image #0
//              bmp1    : Image #1
//              bmp2    : Image #2
//              bmpA    : Image for Alert
id = shmget( SHKEY_BMP0 , IMG_SIZE );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:BMP0]¥n");
  return(-1);
}
bmp0 = (char *)shmat( id );
if( (int)bmp0 == -1 ) {
  printf("Shared memory allocate error [SHMAT:BMP0]¥n");
  return(-1);
}
//      Image #1 : BMP1
id = shmget( SHKEY_BMP1 , IMG_SIZE );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:BMP1]¥n");
  return(-1);
}
bmp1 = (char *)shmat( id );
if( (int)bmp1 == -1 ) {
  printf("Shared memory allocate error [SHMAT:BMP1]¥n");
  return(-1);
```

```
}
//      Image #2 : BMP2
id = shmget( SHKEY_BMP2 , IMG_SIZE );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:BMP2]¥n");
  return(-1);
}
bmp2 = (char *)shmat( id );
if( (int)bmp2 == -1 ) {
  printf("Shared memory allocate error [SHMAT:BMP2]¥n");
  return(-1);
}
//      Image for ALERT : BMPA
id = shmget( SHKEY_BMPA , IMG_SIZE );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:BMPA]¥n");
  return(-1);
}
bmpA = (char *)shmat( id );
if( (int)bmpA == -1 ) {
  printf("Shared memory allocate error [SHMAT:BMPA]¥n");
  return(-1);
}

//
//      int *tempA
//      Alert Temperture        (x10)
//
id = shmget( SHKEY_TEMPA , sizeof(unsigned int) );
if( id == -1 ) {
  printf("Shared memory allocate error [SHMGET:tempA]¥n");
  return(-1);
}
tempA = (int *)shmat( id );
if( (int)tempA == -1 ) {
  printf("Shared memory allocate error [SHMAT:tempA]¥n");
  return(-1);
}
```

```
//
// BCD to Binary
//
int bcdtobin( int bcd ) {
  int bin;
  bin = ((bcd >> 4)&0x0f);
  bin = (bin<<3)+(bin<<1) + (bcd & 0x0f);
  return(bin);
}


//
// Binary to BCD
//
int bintobcd( int bin ) {
  int bcd;
  bcd = (bin / 10)<<4;
  bcd += (bin % 10);
  return( bcd );
}


//
// endian conv (long)
//
long toLittle4( long l ) {
  long_char lc, ret;

  lc.l = l;

  ret.c[0] = lc.c[3];
  ret.c[1] = lc.c[2];
  ret.c[2] = lc.c[1];
  ret.c[3] = lc.c[0];

  return(ret.l);
}


//
// endian conv (short)
//
short toLittle2( short s ) {
  short_char sc, ret;

  sc.s = s;

  ret.c[0] = sc.c[1];
  ret.c[1] = sc.c[0];

  return(ret.s);
}


int YUVtoRGB(int YY, int UU, int VV) {
  short Y, U, V;
  short R, B;
  int G;

  long_char ret;

  Y = (short)YY;
```

```
  U = (short)UU - 128;
  V = (short)VV - 128;

  if( U <-127 ) U = -128;
  if( U > 127 ) U =  127;

  if( V <-127 ) V = -128;
  if( V > 127 ) V =  127;

  R = U + Y;
  if( R < 0 )      R = 0x00;
  if( R & 0xff00 ) R = 0xff;

  B = V + Y;
  if( B < 0 )      B = 0x00;
  if( B & 0xff00 ) B = 0xff;

  // G = 0.98*Y - 0.53*(U-128) - 0.19*(V-128)
  //   = FA*Y   - 88*(U-128)   - 31*(V-128)
  // G = Y*FA = Y*100 - Y*8 + Y*2
  G = (Y<<8) - (Y<<3) + (Y<<1);

  // G -= 88*(U-128) = -(U*80 + U*8)
  G = G - (U<<7) - (U<<3);

  // G -= 31*(V-128) = -(V*32 - V)
  G = G - (V<<5) + V;
  G = (G>>8);

  if( G < 0 )          G = 0x00;
  if( G & 0xffffff00 ) G = 0xff;

  ret.c[0] = (char)R;
  ret.c[1] = (char)G;
  ret.c[2] = (char)B;

  return( ret.l );
}


int YUVtoRGB2(int YY, int UU, int VV) {
  //
  // Y Cr(U) Cb(V)
  //
  long Y, U, V;
  long R, B;
  long G;

  long_char ret;

  Y = YY - 16;
  U = UU - 128;
  V = VV - 128;

  //
  // clip Y U V
  //
  if( Y < 0 )   Y = 0;
  if( U <-127 ) U = -128;
  if( U > 127 ) U =  127;
  if( V <-127 ) V = -128;
  if( V > 127 ) V =  127;



  //
```

```
  // pre Calc
  //
  long Y12A; // 1.164*Y*100H
  Y12A = (Y<<8) + (Y<<5) + (Y<<3) + (Y<<1);

  // R = 1.164*Y + 1.596Cr(U);
  //     1.164=(12A/100)*Y
  //     1.596=(199/100)*U
  R = Y12A + (U<<8) + (U<<7) + (U<<4) + (U<<3) + U;
  if( R < 0 )         R = 0x00;
  R>>=8;
  if( R & 0xffff00 ) R = 0xff;

  // G = 1.164*Y - 0.391*Cb - 0.813*Cr
  //     1.164=(12A/100)*Y
  //     0.391=(064/100)*V
  //     0.813=(0D0/100)*U
  G = Y12A - (V<<6) - (V<<5) - (V<<2) - (U<<7) - (U<<6) - (U<<4);
  if( G < 0 )         G = 0x00;
  G>>=8;
  if( G & 0xffff00 ) G = 0xff;

  // B = 1.164*Y + 2.018*Cb
  //     1.164=(12A/100)*Y
  //     2.018=(204/100)*V
  B = Y12A + (V<<9) + (V<<6);
  if( B < 0 )       B = 0x00;
  B>>=8;
  if( B & 0xff00 ) B = 0xff;

  ret.c[0] = (char)R;
  ret.c[1] = (char)G;
  ret.c[2] = (char)B;

  return( ret.l );
}
```

Sub/init_port.c　　H8 各ポート初期設定用プログラム

```
// Port Initialize
//      void portinit( void )
//
void init_port( void ) {

  // Port 4(LED,LCD)
  P4DDR = 0xFF;         // all output
  P4PCR = 0x00;         // Pullup-R off

  // Port 5(DIPSW[4:1])
  P5DDR = 0xF0;         // all input
  P5PCR = 0xFF;         // Pullup-R on

  // Port 6
  //  P67 Input         I       0
  //  P66-63 none       -       111 1
  //  P62 I2C-A1        0       1
  //  P61 I2C-CLK       0       1
  //  P60 I2C-Data      I       0
  P6DDR = 0x7E;
  P6DR  = 0x00;

  // Port 8 (Treva)
  //  DRCRA.DRAS2 = 0
  //  DRCRA.DRAS1 = 0
  //  DRCRA.DRAS0 = 1   CS2=DRAM
  //  DRCRA.BE    = 1   for TC5117805
  //  DRCRA.RDM   = 0
  //  DRCRA.SRFMD = 1
  //  DRCRA.RFSHE = 0
//  DRCRA = 0x3A;

  // Port A (LED[7:0])
  PADDR = 0xFF;         // all output
  PADR  = 0x00;         // LED off

  // Port B
  // PB7 0 Input        Treva Din
  // PB6 1 Output       Treva CLKout
  PBDR   = 0x00;
  //  PBDDR &= 0x7F;    // PortB7 input  Treva Din
  //  PBDDR |= 0x40;    // PortB6 output Treva CLKout
  PBDDR  = 0x7F;

  // A/D Converter
  ADCSR = 0x01;         // Single-mode / AN1
  ADCR  = 0x7E;

  // LED(Green, Red) turn off
  P4DR &= 0x3f;

  return;
}
```

Sub/init_timer8.c　　H8 8bit タイマ機能初期設定用プログラム

```
//
// 8bit Timer Initialize
//
//    void init_timer8( void )
//              8bit timer 0/1 2/3 Cascade PWM-mode
//
//              Default : clk off
//

void init_timer8( void ) {

  //
  // 8bit Timer CH0/CH1 Cascade                --- LED [RED]
  //
  T8TCR0 = 0x0C;// CMIEB:0          Inhibit Interrupt via CMFB
                          // CMIEA:0          Inhibit Interrupt via CMFA
                          // OVIE :0          Inhibit Interrupt via OVF
                          // CCLR1 CCLR0 : 01  Clear by CMFB
                          // CKS2-0 : 100               Clock select : Ch0 16bit mode (CH0/CH1)

  T8TCR1 = 0x08;// CMIEB:0          Inhibit Interrupt via CMFB
                          // CMIEA:0          Inhibit Interrupt via CMFA
                          // OVIE :0          Inhibit Interrupt via OVF
                          // CCLR1 CCLR0 : 01  Clear by CMFB
                          // CKS2-0 : 000               Clock none
                          //            011             Clock select : Φ/8 = 2.4kHz(0.4096ms)

  TCSR0_8 = 0xE6;         // CMFB: 1          set when 8TCNT=TCORB
                          // CMFA: 1          set when 8TCNT=TCORA
                          // OVF:1            set whenc FF->00
                          // ADTE:0           AD Trigger inhibit
                          // OIS3-0 : 0110    CMFA:1 - CMFB:0

  TCSR1_8 = 0xE6;         // CMFB: 1          set when 8TCNT=TCORB
                          // CMFA: 1          set when 8TCNT=TCORA
                          // OVF:1            set whenc FF->00
                          // ICE:0            Compare match
                          // OIS3-0 : 0110    CMFA:1 - CMFB:0

  TCORA0_8 = 0x09;        // TCORA0/1 = 0989H(2441 = 1sec)
  TCORA1_8 = 0x89;

  TCORB0_8 = 0x04;        // TCORB0/1 = 04C4H(1220 = Duty 50%)
  TCORB1_8 = 0xC4;

  TCNT0_8_short = 0x0000;

  //
  // 8bit Timer CH2/CH3 Cascade                --- LED [GREEN]
  //
  TCR2_8 = 0x0C;// CMIEB:0          Inhibit Interrupt via CMFB
                          // CMIEA:0          Inhibit Interrupt via CMFA
                          // OVIE :0          Inhibit Interrupt via OVF
                          // CCLR1 CCLR0 : 01  Clear by CMFB
                          // CKS2-0 : 100               Clock select : Ch0 16bit mode (CH0/CH1)

  TCR3_8 = 0x08;// CMIEB:0          Inhibit Interrupt via CMFB
                          // CMIEA:0          Inhibit Interrupt via CMFA
                          // OVIE :0          Inhibit Interrupt via OVF
                          // CCLR1 CCLR0 : 01  Clear by CMFB
```

```
                        // CKS2-0 : 000              Clock none
                        //            011            Clock select : Φ/8 = 2.4kHz(0.4096ms)

  TCSR2_8 = 0xE6;       // CMFB: 1          set when 8TCNT=TCORB
                        // CMFA: 1          set when 8TCNT=TCORA
                        // OVF:1            set whenc FF->00
                        // ADTE:0           AD Trigger inhibit
                        // OIS3-0 : 0110    CMFA:1 - CMFB:0

  TCSR3_8 = 0xE6;       //CMFB: 1          set when 8TCNT=TCORB
                        // CMFA: 1          set when 8TCNT=TCORA
                        // OVF:1            set whenc FF->00
                        // ICE:0            Compare match
                        // OIS3-0 : 0110    CMFA:1 - CMFB:0

  TCORA2_8 = 0x09;      // TCORA0/1 = 0989H(2441 = 1sec)
  TCORA3_8 = 0x89;

  TCORB2_8 = 0x04;      // TCORB0/1 = 04C4H(1220 = Duty 50%)
  TCORB3_8 = 0xC4;

  TCNT2_8_short = 0x0000;

  return;
}
```

Sub/opening.c　　起動デモ用プログラム

```
//
// Opening DEMO
//      void opening ( int num )
//
void opening( int num, unsigned int wait ) {
  unsigned int  i;

  for(i=0; i<8; i++) {
    PADR = (0x01)<<i;
    sleep( wait );
  }
  for(i=1; i<8; i++) {
    PADR = (0x80)>>i;
    sleep( wait );
  }
  for(i=0; i<num; i++ ) {
    PADR = 0xFF;
    sleep( wait );    sleep( wait );
    PADR = 0x00;
    sleep( wait );
  }
  return;
}
```

```
//
// Read 1bit from Treva
//
//      int read_treva_bit ( void )
//              return (read bit)
//
int read_treva_bit( void ) {
  int j, d;
  PBDR &= 0xBF;         // CLK = 0
  for(j=0;j<TREVA_WAIT; j++) ;
  d = (PBDR & 0x80);
  PBDR |= 0x40;         // CLK = 1
  for(j=0;j<TREVA_WAIT; j++) ;
  if( d == 0 ) return(0);
  else return(1);
}

//
// Read 1byte from Treva
//
//      int read_treva_byte ( void )
//              return (read byte)
//
int read_treva_byte( void ) {
  int i, j, d;
  d =  0x00;

  for(i=0; i<8; i++) {
    d <<= 1;
    PBDR &= 0xBF;               // CLK = 0
    for(j=0;j<TREVA_WAIT; j++) ;
    if(PBDR & 0x80) d |= 0x01;
    PBDR |= 0x40;               // CLK = 1
    for(j=0;j<TREVA_WAIT; j++) ;
  }
  return( d );
}


//
//
//
int treva( int ID, int frame, int *YUV ) {
  int i, j, k, p;
  char fname[16];
  long_char rgb;

  asm("orc #0xc0, ccr");                // 割り込み禁止
  //
  // find ID
  //
  i = 0;
  while( (i & 0xffffffff) != 0xAA55FFD8 ) {
    i <<= 1;
    if( read_treva_bit()) i |= 0x01;
  }
  i = read_treva_byte();                // Byte of Header

  //
  // HeaderByte分 空読み
  //
  for(j=1; j<i; j++) {
```

```
  read_treva_byte();
}

//
// Read ImageData     V-Y-U-Y-V...
//
int Ymax=0, Ymin=255;
p = 0;
for(j=0; j<IMG_HEIGHT; j++) {
  for(i=0; i<IMG_WIDTH; i++) {
    UV[p] = read_treva_byte();        // V-U
    Y[p]  = read_treva_byte();        // Y
    if(Y[p] < Ymin) Ymin = Y[p];
    if(Y[p] > Ymax) Ymax = Y[p];
    p++;
  }
}
asm("andc #0x3f, ccr");               // 割り込み許可

k = 0;
//  p = 3310; // 3310 = 34 * 96 + 46
p = 3116;     // 3116 = 32 * 96 + 44
for(j=0; j<8; j++) {
  for(i=0; i<8; i++) {
    YUV[k++] = Y[p+i];
  }
  p += 88;    // 96 - 8
}

//  p = 3310; // 3310 = 34 * 96 + 46
p = 3116;     // 3116 = 32 * 96 + 44
for(j=0; j<8; j++) {
  for(i=0; i<8; i++) {
    YUV[k++] = UV[p+i];
  }
  p += 88;    // 96 - 8
}
printf("¥tY[%x]-[%x]¥n", Ymin, Ymax);

//
// Convert to BMPRGB
//
char *bmp;
if( ID == 0 ) bmp = bmp0;
else if(ID == 1 ) bmp = bmp1;
else if(ID == 2 ) bmp = bmp2;
else {
  printf("¥tTreva:ID Error [%d]¥n", ID);
  return(-1);
}

// BMP フォーマットって変な並びだからしゃーないねん
p = 6816; // 96*(72-1)
for(j=0; j<IMG_HEIGHT; j++) {
  for(i=0; i<IMG_WIDTH; i++) {
    if( (p & 0x1)==0 ) {
      rgb.l = YUVtoRGB2( Y[p], UV[(p+1)], UV[p] );
    }else{
      rgb.l = YUVtoRGB2( Y[p], UV[p], UV[(p-1)] );
    }
    *bmp = rgb.c[2];          bmp++;
    *bmp = rgb.c[1];          bmp++;
    *bmp = rgb.c[0];          bmp++;
    p++;
  }
  p -=  192; //96*2
```

```
  }

#ifdef WRITEYUVFILE
  sprintf(fname, "/ram0/%d.yuv", frame);
  if( writeYUVfile( fname, Y, UV ) < 0 ) {
    exit;
  }
#endif

#ifdef WRITEPPMFILE
  sprintf(fname, "/ram0/img%d.ppm", frame);
  // strcpy(fname,"/ram0/img.ppm");
  // fname[9] = 0x30+ID;
  if( writePPMfile( fname, Y, UV ) < 0 ) {
    exit;
  }
#endif

#ifdef WRITEBMPFILE
  strcpy( fname, "/ram0/img*.bmp" );
  fname[9] = 0x30+ID;
  if( writeBMPfile( fname, Y, UV ) < 0 ) {
    exit;
  }
#endif
  printf("¥tTreva write ImageID[%d]¥n", ID );



  return(0);
}
```

Sub/readtemp.c　　温度センサ読み取りプログラム

```
//
// int readtemp( void )
//
// 1度に10回測定して、最大値と最低値を除いた8個の平均値を温度として return する。
// *avetemp にも格納
//      return は x10 の値
//

int readtemp( void ) {
  int temp10[10];
  int i, min, max, min_i, max_i;
  int ret;

  // Measure
  for(i=0; i<10; i++) {
    ADCSR = 0x21;
    while( (ADCSR & 0x80)==0 );
    ADCSR = 0x01;
    temp10[i] = (  (((ADDRBH & 0xFF)<<3) + ((ADDRBL & 0xC0)>>5) ) + TEMP_REVISE );
  }

  // reject Min & Max
  min = temp10[0]; min_i = 0;
  max = temp10[0]; max_i = 0;
  for(i=1; i<10; i++) {
    if( temp10[i] < min ) {
      min = temp10[i];
      min_i = i;
    }
    if( temp10[i] > max ) {
      max = temp10[i];
      max_i = i;
    }
  }



  ret = 0;
  // Calc ave-temp
  temp10[min_i] = 0;
  temp10[max_i] = 0;
  for(i=0; i<10; i++) {
    ret += temp10[i];
  }
  ret = (ret>>3);       // ave = ret÷8

  *avetemp = ret;

  return( ret );
}
```

Sub/writeyuvfile.c　　[デバッグ用]画像 YUV ファイル書き出しプログラム

```
int writeYUVfile( char *fname, int *Y, int *UV) {
  int fd, i, j, p;

  printf("open YUV Filename : %s ..... ", fname );
  fd = open( fname, 0 );
  if( fd < 0 ) {
    printf("File Open ERROR¥n");
    return(-1);
  }
  p = 0;

  for(j=0; j<IMG_HEIGHT; j++) {
    for(i=0; i<IMG_WIDTH; i++) {
      fprintf(fd, "%d %d¥n", Y[p], UV[p]);
      p++;
    }
  }
  close( fd );
  printf(" end¥n");

  return(0);
}
```

Sub/writeppmfile.c　　[デバッグ用]画像 PPM フォーマット書き出しプログラム

```
//
// writePPMfile( char * filename, int *Y, int *UV )
//
//      write file as PPMformat
//
int writePPMfile( char *fname, int *Y, int *UV ) {
  int fd, i, j, p;
  long_char rgb;

  printf("open PPM Filename : %s ..... ", fname);
  fd = open( fname, 0 );
  if( fd < 0 ) {
    printf("File Open ERROR¥n");
    return(-1);
  }
  fprintf(fd, "P6¥n%d %d¥n255¥n#treva image¥n", IMG_WIDTH, IMG_HEIGHT);
  p = 0;
  for(j=0; j<IMG_HEIGHT; j++) {
    for(i=0; i<IMG_WIDTH; i++) {
      if( (i%2)==0 ) {
        rgb.l = YUVtoRGB( Y[p], UV[(p+1)], UV[p] );
      }else{
        rgb.l = YUVtoRGB( Y[p], UV[p], UV[(p-1)] );
      }
      p++;
      write( fd, (char *)&(rgb.c[0]), 1);
      write( fd, (char *)&(rgb.c[1]), 1);
      write( fd, (char *)&(rgb.c[2]), 1);
      // fprintf(fd, "%d %d %d¥n", rgb.c[0], rgb.c[1], rgb.c[2] );
    }
  }
  close( fd );
  printf(" end¥n");
  return(0);
}
```

Sub/writebpmfile.c　　[デバッグ用]画像 BMP フォーマット書き出しプログラム

```
// writeBMPfile( char * filename, int *Y, int *UV )
//
//      write file as BMPformat
//
int writeBMPfile( char *fname, int *Y, int *UV ) {
  int fd, i, j, p;
  long_char rgb;

  unsigned short BITMAP_ID = 0x424d;
  struct {
    unsigned long  fSize;
    unsigned short Reserved1;
    unsigned short Reserved2;
    unsigned long  OffBits;
    unsigned long  Size;
    long           Width;
    long           Height;
    unsigned short Planes;
    unsigned short BitCount;
    unsigned long  Compression;
    unsigned long  SizeImage;
    long           XPixPerMeter;
    long           YPixPerMeter;
    unsigned long  ClrUsed;
    unsigned long  ClrImporant;
  } BITMAP;

  //                                                    96*72*3+54 = 20790
  BITMAP.fSize          = toLittle4(20790);            // 0000 5136
  BITMAP.Reserved1      = (short)0;
  BITMAP.Reserved2      = (short)0;
  BITMAP.OffBits        = toLittle4(54);               // 0000 0036

  BITMAP.Size           = toLittle4(40);               // 0000 0028
  BITMAP.Width          = toLittle4(IMG_WIDTH);        // 0000 0060
  BITMAP.Height         = toLittle4(IMG_HEIGHT);               // 0000 0048
  BITMAP.Planes         = toLittle2(1);
  BITMAP.BitCount       = toLittle2(24);
  BITMAP.Compression    = 0;
  BITMAP.SizeImage      = toLittle4(20736);
  BITMAP.XPixPerMeter   = 0;
  BITMAP.YPixPerMeter   = 0;
  BITMAP.ClrUsed        = 0;
  BITMAP.ClrImporant    = 0;

  printf("open BMP Filename : %s ..... ", fname);
  fd = open( fname, 0 );
  if( fd < 0 ) {
    printf("File Open ERROR¥n");
    return(-1);
  }
  write( fd, (char *)&BITMAP_ID,  2 );
  write( fd, (char *)&BITMAP, sizeof(BITMAP) );

  p = 6816; // 96*(72-1)
  // BMP フォーマットって変な並びだからしゃーないねん
  for(j=0; j<IMG_HEIGHT; j++) {
    for(i=0; i<IMG_WIDTH; i++) {
      if( (p & 0x1)==0 ) {
        rgb.l = YUVtoRGB( Y[p], UV[(p+1)], UV[p] );
      }else{
```

```
        rgb.l = YUVtoRGB( Y[p], UV[p], UV[(p-1)] );
      }
      p++;
      write( fd, (char *)&(rgb.c[2]), 1);
      write( fd, (char *)&(rgb.c[1]), 1);
      write( fd, (char *)&(rgb.c[0]), 1);
    }
    p -=  192; //96*2
  }
  close(fd);
  printf(" end¥n");
  return(0);
}
```

Sub/ntp.c	簡易 SNTP クライアントプログラム

```
//
// int ntp( void )
//

int ntp( dateformat *datetime ) {
  struct sockaddr fromadd, ntpserver;
  int sock, size;
  struct ntp_pkt pkt;

  PADR |= 0x01;
  sock = udp_socket();
  // myaddr.sin_addr = IPADDR(192,168,0,12);
  myaddr.sin_port = 10000;
  size = udp_bind(sock, &myaddr);               // いらない

  ntpserver.sin_addr = NTP_SERVER;
  ntpserver.sin_port = NTP_PORT;

  bzero(&pkt, sizeof(pkt));
  pkt.li_vn_mode = 0x1B;

  size = sendto(sock,   (char *)&pkt, sizeof(struct ntp_pkt), &ntpserver);
  size = recvfrom(sock, (char *)&pkt, sizeof(struct ntp_pkt), &fromadd);

  udp_free(sock);

  //
  if( DIPSW1 ) {
    printf("%02x %02x %02x %02x¥n", pkt.li_vn_mode, pkt.stratum, pkt.ppoll, pkt.precision);
    printf("%08x¥n", pkt.rootdelay );
    printf("%08x¥n", pkt.rootdispersion );
    printf("%08x¥n", pkt.refid );

    printf("%08x %08x¥n", pkt.reftime.l_ui, pkt.reftime.l_uf );
    printf("%08x %08x¥n", pkt.org.l_ui, pkt.org.l_uf );
    printf("%08x %08x¥n", pkt.rec.l_ui, pkt.rec.l_uf );
    printf("%08x %08x¥n", pkt.xmt.l_ui, pkt.xmt.l_uf );
  }




  //


  // LI check (LI=3 error)
  if ( ( pkt.li_vn_mode & 0xc0) == 0xc0 ) {
    return(-2);
  }

  long_short time;

  // - C57F E7F0 (3313494000)   2005.1.1
  time.ltime = pkt.xmt.l_ui;

  //  UP - C57F
  unsigned long up = (unsigned long)time.stime[0];
  up -= 0xC57F;

  // Low - E7F0
  signed long lo = (unsigned long)time.stime[1] - 0xE7F0;
```

```
if( lo < 0 ) {
  lo = 0x1810 + (unsigned long)time.stime[1];
  up--;
}

long_short ls;
ls.stime[0] = up;
ls.stime[1] = lo;
signed long serialtime = ls.ltime+ TIME_REVISE;


//
// Calc-Year
//
unsigned short Y = 2005;
unsigned short W = 6;                  // 2005.1.1 = Saturday
unsigned long  l;
do {
  if( ((Y%4==0)&&(Y%100!=0))||(Y%400==0) ) { // うるう年
    l = 0x01E28500; // 24*3600*366
  }else{
    l = 0x01E13380; // 24*3600*365
  }
  // printf("%d - %d", serialtime, l);
  serialtime -= l;
  // printf(" = %d¥n", serialtime);
  Y++;

  //    printf("Y:%d (Remain:%d)¥n", Y, serialtime);
}while( serialtime >= 0 );
serialtime += l;
Y--;
// printf("Y:%d (Remain:%d)¥n", Y, serialtime);

// Illegal Data Check
if( (Y < 2005) || ( Y > 2030) ) {
  return(-2);
}

//
// Calc-Month
//
unsigned short MONTD[12]={31,28,31,30,31,30,31,31,30,31,30,31};
unsigned long  MONTH[12]={0x0028DE80, 0x0024EA00, 0x0028DE80, 0x00278D00,
                          0x0028DE80, 0x00278D00, 0x0028DE80, 0x0028DE80,
                          0x00278D00, 0x0028DE80, 0x00278D00, 0x0028DE80};
unsigned short M = 0;
int i;

if( l == 0x01E28500 ) {
  MONTD[1] = 29;
  MONTH[1] = 0x00263B80; // 29*24*3600
}
for( i=0; i<12; i++ ) {
  serialtime -= MONTH[i];
  if( serialtime < 0 ) break;
  W += MONTD[i];
}
serialtime += MONTH[i];
M = i + 1;

//
// Calc-Day
//
unsigned short D;
unsigned long lm;
```

```
  D = 0;
  lm = 0x00015180;
  do{
    // printf("D:%d %d - %d ", D, serialtime, lm);
    serialtime -= lm;
    // printf(" = %d¥n", serialtime);
    D++;
  }while(serialtime >= 0);
  serialtime += lm;
  // printf("D:%d (Remain:%d)¥n", D, serialtime);
  W = W + D - 1;

  //
  // Calc-Hour
  //
  unsigned short HH;
  HH = 0;
  lm = 0x00000E10;
  do{
    // printf("H:%d %d - %d ", HH, serialtime, lm);
    serialtime -= lm;
    // printf(" = %d¥n", serialtime);
    HH++;
  }while(serialtime >= 0);
  HH--;
  serialtime += lm;
  // printf("HH:%d (Remain:%d)¥n", HH, serialtime);

  //
  // Calc-Min
  //
  unsigned long MM,SS;
  MM = 0;
  lm = 0x0000003C;
  do{
    // printf("M:%d %d - %d ", MM, serialtime, lm);
    serialtime -= lm;
    // printf(" = %d¥n", serialtime);
    MM++;
  }while(serialtime >= 0);
  MM--;
  serialtime += lm;
  // printf("MM:%d (Remain:%d)¥n", MM, serialtime);
  SS = serialtime;

  dateformat date;
  date.y  = (unsigned int)Y;
  date.m  = (unsigned int)M;
  date.d  = (unsigned int)D;
  date.w  = (unsigned int)(W % 7);
  date.hh = (unsigned int)HH;
  date.mm = (unsigned int)MM;
  date.ss = (unsigned int)SS;
  bcopy( &date, datetime, sizeof( dateformat ) );

  printf("¥n¥nNTP¥t%04d/%02d/%02d(%d) %02d:%02d:%02d¥n", Y, M, D, (W%7), HH, MM, SS);

  PADR &= 0xFE;
  return(0);
}
```

Sub/timer8x2.c　　H8 8bit タイマー制御プログラム

```
// int timer8x2( int ID, int status )
//
//      return :  0 normal
//             : -1 error
//
//      int ID          0 : Timer0/1    Red   LED
//                      2 : Timer2/3    Green LED
//
//      int status      0 : OFF
//                      1 : ON
//                      2 : Slow Blink
//                      3 : Fast Blink

int timer8x2( int ID, int status ) {

  switch( ID ) {
  case 0 :      // Timer 0/1    LED [RED]
    TCNT0_8_short = 0x0000;
    T8TCR1   = 0x08;            // Clock OFF

    switch( status ) {
    case 0:             // LED OFF
      // PortB re-setting
      TCSR0_8 = 0xE0;
      PBDR &= 0xFE;             // PB0 off
      break;

    case 1:             // LED ON
      // PortB re-setting
      TCSR0_8 = 0xE0;
      PBDR |= 0x01;             // PB0 on
      break;

    case 2 :            // LED Blink Slow (1sec)
      TCSR0_8 = 0xE6;
      TCORA0_8 = 0x09;          // Period : 1sec
      TCORA1_8 = 0x89;
      TCORB0_8 = 0x03;
      TCORB1_8 = 0xD0;          // Duty 40%
      T8TCR1   = 0x0B;          // Clock On
      break;

    case 3 :            // LED Blink Fast (0.25sec)
      TCSR0_8 = 0xE6;
      TCORA0_8 = 0x02;          // Period : 0.25sec
      TCORA1_8 = 0x62;
      TCORB0_8 = 0x00;
      TCORB1_8 = 0x98;          // Duty 25%
      T8TCR1   = 0x0B;          // Clock On
      break;

    default :
      return(-1);
    }
    break;

  case 2 :      // Timer 2/3    LED [GREEN]
    TCNT2_8_short = 0x0000;
    TCR3_8   = 0x08;    //      Clock off
    switch( status ) {
    case 0:             // LED OFF
```

```
      // PortB re-setting
      TCSR2_8 = 0xE0;
      PBDR &= 0xFB;             // PB2 off
      break;

    case 1:             // LED ON
      // PortB re-setting
      TCSR2_8 = 0xE0;
      PBDR |= 0x04;             // PB2 on
      break;

    case 2 :            // LED Blink Slow (1sec)
      TCSR2_8 = 0xE6;
      TCORA2_8 = 0x09;  //      Period : 1sec
      TCORA3_8 = 0x89;
      TCORB2_8 = 0x03;
      TCORB3_8 = 0xD0;  //      Duty 40%
      TCR3_8   = 0x0B;  //      Clock on        Φ/8 = 2.4kHz(0.4096ms)
      break;

    case 3 :            // LED Blink Fast (0.25sec)
      TCSR2_8 = 0xE6;
      TCORA2_8 = 0x02;  //      Period : 0.25sec
      TCORA3_8 = 0x62;
      TCORB2_8 = 0x00;
      TCORB3_8 = 0x7A;  //      Duty 20%
      TCR3_8   = 0x0B;  //      Clock on        Φ/8 = 2.4kHz(0.4096ms)
      break;

    default :
      return(-1);
    }
    break;
  default :
    return(-1);
  }

  return(0);
}
```

Sub/pop3.c　　POP3 クライアントプログラム

```
// POP before SMTP
//
// int pop( void )
//


int pop3 ( void ) {
  char mes[256];
  int sock, ret;
  struct sockaddr pop3server;

  myaddr.sin_port = 12346;
  pop3server.sin_addr = POP3SERVER;
  pop3server.sin_port = POP3_PORT;

  /* POP3 server 表示
  printf("POP3 : %d.%d.%d.%d : %d¥n¥n",
         ((pop3server.sin_addr>>24)&0xff),       ((pop3server.sin_addr>>16)&0xff),
         ((pop3server.sin_addr>>8) &0xff),       ((pop3server.sin_addr)&(0xff) ),
         pop3server.sin_port );
  */


  //
  // TCP コネクション
  //

  // Socket 取得
  sock = tcp_socket();
  printf("¥tPOP3 : SOCK(%d)¥n", sock);

  // Connect
  ret = tcp_connect( sock, &pop3server );
  if( ret != 0 ) {
    printf("¥tPOP3 : tcp_connect(%d)¥n", ret );
    tcp_free(sock);
    return(-1);
  }
  printf("¥tPOP3 : Connect¥n");

  sleep(200);
  bzero( mes,  250 );
  ret = tcp_recv(sock, mes, 250 );
  if( ret == -1 ) {
    printf("¥tPOP3: R 0 :Disconnect¥n");
    tcp_free(sock);
    return(-1);
  }
  printf("¥tPOP3: R 0 :[%d]%s", ret, mes);
  if( strncmp(mes, "+OK", 3) != 0 ) {
    printf("¥tDisconnect (%s)******¥n", mes);
    tcp_free(sock);
    return(-1);
  }
  printf("¥n");

  //
  // send command
  //

  //
```

```
  // 1: USER
  //
  sleep(200);
  bzero( mes,  250 );
  sprintf(mes, "USER %s¥r¥n", POP3_USER);
  ret = tcp_send( sock, mes, strlen(mes) );
  if( ret == -1 ) {
    printf("¥tPOP3: S 1 :Disconnect ******¥n");
    tcp_free(sock);
    return(-1);
  }
  printf("¥tPOP3: S 1 :[%d]%s", ret, mes);

  sleep(200);
  bzero( mes,  250 );
  ret = tcp_recv(sock, mes, 250 );
  printf("¥tPOP3: R 1 :(%d)%s¥n", strlen(mes), mes);


  //
  // 2: Pass
  //
  sleep(200);
  bzero( mes,  250 );
  sprintf(mes, "PASS %s¥r¥n", POP3_PASS);
  ret = tcp_send( sock, mes, strlen(mes) );
  if( ret == -1 ) {
    printf("¥tPOP3: S 2 :Disconnect ******¥n");
    tcp_free(sock);
    return(-1);
  }
  printf("¥tPOP3: S 2 :[%d]%s", ret, mes);

  sleep(200);
  bzero( mes,  250 );
  ret = tcp_recv(sock, mes, 250 );
  printf("¥tPOP3: R 2 :(%d)%s¥n", strlen(mes), mes);


  //
  // 6: QUIT
  //
  sleep(200);
  bzero( mes,  250 );
  strcat(mes, "QUIT¥r¥n");
  ret = tcp_send( sock, mes, strlen(mes) );
  if( ret == -1 ) {
    printf("¥tPOP3: S 6 :Disconnect ******¥n");
    tcp_free(sock);
    return(-1);
  }
  printf("¥tPOP3: S 6 :[%d]%s", ret, mes);

  bzero( mes,  250 );
  ret = tcp_recv(sock, mes, 250 );
  printf("¥tPOP3: R 6 :(%d)%s¥n", strlen(mes), mes);



  //
  // DisConnect
  //
  printf("¥tPOP3 : Disconnect¥n");
  tcp_free(sock);
  return(0);
}
```

Sub/mailsend.c　　SMTP メール送信プログラム

```
//
// メールを送る
//
// int mailsend( int popbeforesmtp, char *mes, dateformat *date )
//      popbeforesmtp : POP before SMTP flag   0 .. No / 1 .. Yes
//      mes : メール本文へのポインタ
//      date : 現在時刻へのポインタ
//

int mailsend( int popbeforesmtp, char *body, dateformat *date ) {
  char mes[256];

  int sock, ret;
  struct sockaddr smtpserver;

  if( strlen(body) > 250 ) {
    printf("¥tSendmail:body too big¥n");
    return(-1);
  }

  //
  // POP before SMTP
  //
  if( popbeforesmtp == 1 ) {
    pop3();
  }

  //
  // TCP コネクション
  //
  sock = tcp_socket();
  printf("¥tSendmail : SOCK(%d)¥n", sock);

  myaddr.sin_port = SMTP_PORT ;
  smtpserver.sin_addr = SMTPSERVER;
  smtpserver.sin_port = SMTP_PORT;

  // Connect
  ret = tcp_connect( sock, &smtpserver );
  if( ret != 0 ) {
    printf("¥tSendmail : tcp_connect(%d)¥n", ret );
    tcp_free(sock);
    return(-1);
  }
  printf("¥tSendmail : Connect OK¥n");

  sleep(200);
  bzero( mes,  250 );
  ret = tcp_recv(sock, mes, 250 );
  if( ret == -1 ) {
    printf("¥tSendmail: R 0 :Disconnect¥n");
    tcp_free(sock);
    return(-1);
  }
  printf("¥tSendmail: R 0 :[%d]%s", ret, mes);
  if( strncmp(mes, "220", 3) != 0 ) {
    printf("¥tDisconnect ******");
    tcp_free(sock);
    return(-1);
  }
  printf("¥n");
```

```
//
// send command
//

//
// 1: HELLO
//
sleep(200);
bzero( mes,  250 );
sprintf(mes, "HELO %d.%d.%d.%d¥r¥n",
        (myaddr.sin_addr & 0xFF000000)>>24,
        (myaddr.sin_addr & 0x00FF0000)>>16,
        (myaddr.sin_addr & 0x0000FF00)>>8,
        (myaddr.sin_addr & 0x000000FF));
ret = tcp_send( sock, mes, strlen(mes) );
if( ret == -1 ) {
  printf("¥tSendmail: S 1 :Disconnect ******¥n");
  tcp_free(sock);
  return(-1);
}
printf("¥tSendmail: S 1 :[%d]%s¥n", ret, mes);

sleep(200);
bzero( mes,  250 );
ret = tcp_recv(sock, mes, 250 );
if( strncmp(mes, "250", 3) != 0 ) {
  printf("¥tSendmail: R 1 :Disconnect(%s) ******¥n", mes);
  tcp_free(sock);
  return(-1);
}
printf("¥tSendmail: R 1 :(%d)%s¥n", strlen(mes), mes);

//
// 2: MAIL FROM
//
sleep(200);
bzero( mes,  250 );
sprintf(mes, "MAIL FROM:<%s>¥r¥n", FROM_ADDRESS);
ret = tcp_send( sock, mes, strlen(mes) );
if( ret == -1 ) {
  printf("¥tSendmail: S 2 :Disconnect ******¥n");
  tcp_free(sock);
  return(-1);
}
printf("¥tSendmail: S 2 :[%d]%s¥n", ret, mes);

sleep(200);
bzero( mes,  250 );
ret = tcp_recv(sock, mes, 250 );
if( strncmp(mes, "250", 3) != 0 ) {
  printf("¥tSendmail: R 2 :Disconnect(%s) ******¥n", mes);
  //    tcp_free(sock);
  //    return(-1);
}
printf("¥tSendmail: R 2 :(%d)%s¥n", strlen(mes), mes);

//
// 3: RCPT TO
//
sleep(200);
bzero( mes, 250 );
sprintf(mes, "RCPT TO:<%s>¥r¥n", TO_ADDRESS);
ret = tcp_send( sock, mes, strlen(mes) );
if( ret == -1 ) {
  printf("¥tSendmail: S 3 :Disconnect ******¥n");
```

```
    tcp_free(sock);
    return(-1);
  }
  printf("¥tSendmail: S 3 :[%d]%s", ret, mes);

  sleep(200);
  bzero( mes, 250 );
  ret = tcp_recv(sock, mes, 250 );
  if( strncmp(mes, "250", 3) != 0 ) {
    printf("¥tSendmail: R 3 :Disconnect(%s) ******¥n", mes);
    //    tcp_free(sock);
    //    return(-1);
  }
  printf("¥tSendmail: R 3 :(%d)%s¥n", strlen(mes), mes);

  //
  // 4: DATA
  //
  sleep(200);
  bzero( mes, 250 );
  strcpy(mes, "DATA¥r¥n");
  ret = tcp_send( sock, mes, strlen(mes) );
  if( ret == -1 ) {
    printf("¥tSendmail:tcp_send:4:Disconnect ******¥n");
    //    tcp_free(sock);
    //    return(-1);
  }
  printf("¥tSendmail: S 4 :(%d)%s¥n", strlen(mes), mes);
  sleep(200);
  bzero( mes, 250 );
  ret = tcp_recv(sock, mes, 250 );
  if( strncmp(mes, "354", 3) != 0 ) {
    printf("¥tSendmail: R 4 :4:Disconnect(%s) ******¥n", mes);
    //    tcp_free(sock);
    //    return(-1);
  }
  printf("¥tSendmail: R 4 :(%d)%s¥n", strlen(mes), mes);

  //
  // 5: Mail Body
  //
  sleep(200);
  bzero( mes, 250 );
  char *week[7]   = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };
  char *month[12] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };
  sprintf(mes, "From: H8 <%s>¥r¥nTo: %s¥r¥nDate: %s, %d %s %04d %02d:%02d:%02d +0900¥r¥nSubject:
AlertMail¥r¥n¥r¥n",
          FROM_ADDRESS,
          TO_ADDRESS,
          week[ date->w ], date->d, month[(date->m)-1], date->y, date->hh, date->mm, date->ss );
  strcat(mes, body);
  strcat(mes, "¥r¥n.¥r¥n");
  ret = tcp_send( sock, mes, strlen(mes) );
  if( ret == -1 ) {
    printf("¥tSendmail:tcp_send:5:Disconnect ******¥n");
    //    tcp_free(sock);
    //return(-1);
  }
  printf("¥tSendmail: S 5 :(%d)%s¥n", strlen(mes), mes);
  sleep(200);
  bzero( mes, 250 );
  ret = tcp_recv(sock, mes, 250 );
  if( strncmp(mes, "250", 3) != 0 ) {
    printf("¥tSendmail:tcp_rcv:5:Disconnect(%s) ******¥n", mes);
    //    tcp_free(sock);
    //    return(-1);
```

```
  }
  printf("¥tSendmail: R 5 :(%d)%s¥n", strlen(mes), mes);

  //
  // 6: QUIT
  //
  sleep(200);
  bzero( mes, 250 );
  strcat(mes, "QUIT¥r¥n");
  ret = tcp_send( sock, mes, strlen(mes) );
  printf("¥tSendmail: S 6 :(%d)%s¥n", strlen(mes), mes);

  sleep(200);
  bzero( mes, 250 );
  ret = tcp_recv(sock, mes, 250 );
  if( strncmp(mes, "221", 3) != 0 ) {
    printf("¥tSendmail: R 6 :(%d)%s¥n", strlen(mes), mes);
  }
  printf("¥tSendmail: R 6 :(%d)%s¥n", strlen(mes), mes);

  tcp_free(sock);
  return(0);
}
```

Sub/i2c.c	I2C 制御プログラム

```
//
// I2C
//
//
//

#define SCL     (0x02)
#define SDA     (0x01)

#define I2CDDR  P6DDR
#define I2CDR   P6DR

#define NAK     SDA
#define ACK     0

#define SDA_READ        (I2CDR  &   SDA)

#define SDA_OPEN        (I2CDDR &= ~SDA)
#define SDA_OUT         (I2CDDR |=  SDA)
#define SCL_OUT         (I2CDDR |=  SCL)
#define SCL_LOW         (I2CDR  &= ~SCL)
#define SCL_HIGH        (I2CDR  |=  SCL)
#define SDA_LOW         (I2CDR  &= ~SDA)
#define SDA_HIGH        (I2CDR  |=  SDA)

int i2c_start( int addr ) {
  int ret, count;
  count = 0;
  do {
    SDA_OUT;
    SCL_LOW;
    SDA_HIGH;
    SCL_HIGH;
    SDA_LOW;    // Start Condition
    SCL_LOW;
    SDA_OPEN;

    ret = i2c_write( addr );
    if( count++ > TIMEOVER )    return(-1);
  }while(ret == NAK);
  return(0);
}

void i2c_stop( void ) {
  SDA_OUT;
  SCL_LOW;
  SDA_LOW;
  SCL_HIGH;
  SDA_HIGH;     // Stop Condition
  SCL_LOW;
  SDA_OPEN;
  return;
}

int i2c_write( int data ) {
  int i, mask;

  SDA_OUT;
  mask = 0x80;
  for(i=0; i<8; i++) {
    if( data & mask ) SDA_HIGH;
```

```
    else              SDA_LOW;
    SCL_HIGH;
    SCL_LOW;
    mask >>= 1;
  }
  SDA_OPEN;
  SDA_OPEN;  SDA_OPEN;  SDA_OPEN;       //wait
  SCL_HIGH;
  SCL_HIGH;  SCL_HIGH;  SCL_HIGH;
  i = (I2CDR & SDA);
  SCL_LOW;
  return(i);
}

int i2c_read( void ) {
  int i, data;

  data = 0;
  for(i=0; i<8; i++) {
    data <<= 1;
    SCL_HIGH;
    if(  SDA_READ ) data++;
    SCL_LOW;
  }
  return(data);
}

void i2c_nak( void ) {
  SDA_OUT;
  SDA_HIGH;
  SCL_HIGH;
  SCL_LOW;
  SDA_OPEN;
}

void i2c_ack( void ) {
  SDA_OUT;
  SDA_LOW;
  SCL_HIGH;
  SCL_LOW;
  SDA_OPEN;
}
```

Sub/rtc.c　　RTC 制御プログラム

```
// int init_rtc ( void )
//      return  -1 : Error
//               0 : Normal
//
int init_rtc( void ) {
  int i, ret;

  ret = rtc_write( RTC_CTRL1, 0x20 );           // Timer Init & Stop
  if( ret == -1 ) {printf("*** init_rtc error #0 ***¥n"); return(-1);}

  ret = rtc_write( RTC_CTRL2, 0x11 );           // Repeat mode
                                                // Alarm disable / xINT enable
  if( ret == -1 ) {printf("*** init_rtc error #1 ***¥n"); return(-1);}

  for(i=2; i<=8; i++) {
    ret = rtc_write( i, 0x00 );                 // Y/M/D(W) H:M:S = All-0 clear
    if( ret == -1 ) {printf("*** init_rtc error #%x ***¥n", i); return(-1);}
  }

  for(i=9; i<=0x0c; i++) {
    ret = rtc_write( i, 0x00 );                 // Alarm All-0 clear
    if( ret == -1 ) {printf("*** init_rtc error #%x ***¥n", i); return(-1);}
  }

  ret = rtc_write( RTC_CLKOUT, 0x03 );          // CLKOUT 1Hz no-output
  if( ret == -1 ) {printf("*** init_rtc error #D ***¥n"); return(-1);}

  ret = rtc_write( RTC_TIMERCTL, 0x82 );        // Timer 1sec enable
  if( ret == -1 ) {printf("*** init_rtc error #E ***¥n"); return(-1);}

  ret = rtc_write( RTC_TIMER, 0x01 );           // Timer 1sec
  if( ret == -1 ) {printf("*** init_rtc error #E ***¥n"); return(-1);}

  return(0);
}

//
// set RTC from datetime
//
int set_rtc( dateformat *datetime ) {
  int i;
  rtc_write( RTC_CTRL1, 0x20 );         // stop RTC

  rtc_write( RTC_SEC,   bintobcd(datetime->ss) );
  rtc_write( RTC_MIN,   bintobcd(datetime->mm) );
  rtc_write( RTC_HOUR,  bintobcd(datetime->hh) );
  rtc_write( RTC_DAY,   bintobcd(datetime->d)  );
  rtc_write( RTC_WEEK, datetime->w  );
  rtc_write( RTC_MONTH, bintobcd(datetime->m)  );
  i = (datetime->y)%100;
  rtc_write( RTC_YEAR,  bintobcd( i )  );

  rtc_write(RTC_CTRL1, 0x00);           // RTC start
}

//
// Read RTC & set to datetime
//
int read_rtc( dateformat *datetime ) {
  int date[7];
  do{
```

```
    rtc_readmulti( RTC_SEC, 7, date );  // read Timedate from RTC
    datetime->ss = bcdtobin(date[0] & 0x7F);
    datetime->mm = bcdtobin(date[1] & 0x7F);
    datetime->hh = bcdtobin(date[2] & 0x3F);
    datetime->d  = bcdtobin(date[3] & 0x3F);
    datetime->w  = bcdtobin(date[4] & 0x07);
    datetime->m  = bcdtobin(date[5] & 0x1F);
    datetime->y  = 2000 + bcdtobin(date[6]);
  }while( datetime->y > 2010 );
  //  printf("¥tRTC %d/%d/%d(%d) %d:%d:%d¥n",
  //datetime->y, datetime->m, datetime->d, datetime->w,
  //datetime->hh, datetime->mm, datetime->ss);
  return(0);
}



int rtc_write( int addr, int data ) {
  int ret;

  ret = i2c_start( RTC_WRITE );
  if( ret == -1 ) return(-1);

  ret = i2c_write( addr );
  ret = i2c_write( data );
  i2c_stop();
  return(0);
}

int rtc_read( int addr ) {
  int   data, ret;

  ret = i2c_start( RTC_WRITE ); // Address 指定読み出し
  if(ret == -1) {printf("*** rtc_read error #1 ***¥n"); return(-1);}

  i2c_write( addr );            // Address 指定

  ret = i2c_start( RTC_READ );  // Start & SlaveAddress Read
  if(ret == -1) {printf("*** rtc_read error #2 ***¥n"); return(-1);}

  data = i2c_read();            // Data Read

  i2c_nak();
  i2c_stop();

  return( data );
}



int rtc_readmulti(int addr, int num, int *rdata ) {
  int   data, ret, i;
  ret = i2c_start( RTC_WRITE ); // I2C SlaveAddress & Write
  if(ret == -1) {printf("*** rtc_readmulti error #1 ***¥n"); return(-1);}

  ret = i2c_write( addr );              // Command Address Write
  if( ret != 0 ) {
    i2c_stop();                         //      Error
    printf("¥nERROR readmulti¥n");
    return(-1);
  }

  ret = i2c_start( RTC_READ );  // I2C SlaveAddress & Read
  if(ret == -1) {printf("*** rtc_readmulti error #2 ***¥n"); return(-1);}
```

```
  for(i=0; i<num; i++) {
    rdata[i] = i2c_read();
    i2c_ack();
  }
  i2c_nak();
  i2c_stop();
}


int rtc_writemulti(int addr, int num, int *wdata ) {
  int i, ret;

  ret = i2c_start( RTC_WRITE );
  if( ret == -1 ) return(-1);

  ret = i2c_write( addr );
  if( ret != 0 ) {
    //i2c_stop();                                //      Error
    printf("¥nERROR writemulti[%x]¥n", ret);
    //return(-1);
  }

  int err = 0;
  for( i=0; i<num; i++) {
    ret = i2c_write( wdata[i] );
    if( ret != 0 ) {
      err++;
      //i2c_stop();                              //      Error
      //printf("¥nERROR writemulti[%x]:[%x]¥n", i, ret);
      //return(-1);
    }
  }
  i2c_stop();
  printf("¥nERROR writemulti total :%d¥n", err);

  return 0 ;
}
```

Sub/endian.c　　エンディアン変換プログラム

```
//
// endian.c
//


//
//
//
long toLittle4( long l ) {
  long_char lc, ret;

  lc.l = l;

  ret.c[0] = lc.c[3];
  ret.c[1] = lc.c[2];
  ret.c[2] = lc.c[1];
  ret.c[3] = lc.c[0];

  return(ret.l);
}


short toLittle2( short s ) {
  short_char sc, ret;

  sc.s = s;

  ret.c[0] = sc.c[3];
  ret.c[1] = sc.c[2];

  return(ret.s);
}
```