巻　末　資　料

資料 1

生産電子情報システム技術科

標準カリキュラム（案）

生産システム技術系 【生産電子情報システム技術科】

【専攻学科】 標準カリキュラム（案）

授 業 科 目 カ リ キ ュ ラ ム 表

科　　名	生産電子情報システム技術科	教科の区分	専攻学科
教 科 の 科 目	複合電子回路設計		
授　業　科　目	アナログ電子回路設計	単　　位	2
教 育 訓 練 目 標	無線通信機器に用いられる回路として増幅器、スイッチ、フィルタを中心に、アナログ電子回路の設計手法について学習する。		

授業科目の細目	授　業　科　目　の　内　容	訓 練 時 間
1. アンプ回路の設計	(1) ローノイズアンプ（LNA）の設計 ① LNAの役割 ② 雑音指数と増幅度 (2) LNAに使える半導体デバイス	6H
2. ミキサの設計	(1) ミキサの役割 (2) 周波数変換の原理 (3) ミキサの種類と特徴 ① パッシブミキサ ② アクティブミキサ	８H
3. フィルタ回路の設計	(1) フィルタ回路の種類と役割 ① ローパスフィルタ ② バンドパスフィルタ ③ アクティブフィルタ ④ その他	８H
4. 検波回路の設計	(1) ショットキー・バリア・ダイオード (2) 検波回路の種類	６H
5. 発振回路設計	(1) 発振回路の基礎 (2) LC発振回路 (3) VCO（Voltage Controlled Oscillator）の基礎	８H
		合計３６H
使 用 す る 機 械 器 具 等		

授業科目カリキュラム表

1／1

科名	生産電子情報システム技術科	教科の区分	専攻学科
教科の科目	複合電子回路設計		
授業科目	デジタルデバイス設計	単位	2
教育訓練目標	ハードウェア記述言語（HDL）による大規模デジタル回路の実践的かつ効率的な設計の手法について学習する。		

授業科目の細目	授業科目の内容	訓練時間
1. 論理回路	(1) 論理回路設計 ① 組み合わせ回路 ② フリップフロップ回路 ③ 順序回路 (2) 各種論理回路設計演習	１０H
2. ハードウェア記述言語	(1) HDLによる回路設計 (2) 論理シミュレータ (3) 論理合成ツール	１０H
3. カスタムIC	(1) CPLDの種類と特徴、応用例 (2) FPGAの種類と特徴、応用例 (3) ASICの種類と特徴、応用例	１０H
4. 論理回路の実装	(1) HDLの実装テスト ① 論理合成 ② 配置配線 ③ デバイスコンフィギュレーション	６H
		合計３６H
使用する機械器具等		

授 業 科 目 カ リ キ ュ ラ ム 表

科　　名	生産電子情報システム技術科	教科の区分	専攻学科
教 科 の 科 目	複合電子回路設計		
授　業　科　目	センサ応用技術	単　　位	2
教 育 訓 練 目 標	磁気センサ、光センサ、温度センサ、生体センサ等のセンサ応用回路とセンシング技術の産業への活用について学習する。		

授業科目の細目	授　業　科　目　の　内　容	訓 練 時 間
1. センサ活用	(1) センシング技術 ① システムでの位置づけと物理基準（精度、分解能、特性） ② センシング技術と産業	6 H
2. 磁気センサ	(1) 磁気センサ応用回路 ① 金属探知 ② マグネスケール ③ 磁気と電流の検出	6 H
3. 光センサ	(1) 光センサ応用回路 ① 光コード読み取り ② 輝度、照度、日照 ③ 赤外領域での活用とレーザ応用	6 H
4. 物理センサ	(1) 物理量をセンシングするセンサ応用回路 ① 温度、湿度、変位、加速度	6 H
5. 生体センサ	(1) 生体センサ技術 ① 生体認証における生体情報とセンサ	6 H
６．A/D変換	(1) A/D変換技術 ① A/D変換の原理とA/D変換チップ ② A/D変換回路	6 H
		合計３６H
使　用　す　る 機 械 器 具 等		

授業科目カリキュラム表

1／1

科名	生産電子情報システム技術科	教科の区分	専攻学科
教科の科目	複合電子回路設計		
授業科目	複合電子回路技術	単位	2
教育訓練目標	デジタルICとアナログ回路を同一システム上に実装するための技術を学習する。		

授業科目の細目	授業科目の内容	訓練時間
1. デジタル・アナログ混在回路	(1) デジタル・アナログ混在回路 ① デジタル回路とアナログ回路の基礎 ② デジタル回路とアナログ回路の混在システム例	8H
2. 電磁ノイズ対策	(1) ノイズ対策 ① 電磁ノイズの基礎と種類 ② グランドの重要性 ③ デジタルICがアナログ回路に与える影響 ④ アナログ回路のS/N比劣化対策	8H
3. A/D・D/A変換	(1) A/D変換回路 (2) D/A変換回路 ① D/A変換回路の方式 ② 映像と音声のD/A変換技術	10H
4. デジタル信号処理とデジタルフィルタ	(1) デジタル信号処理の概要 (2) DSPの概要 ① 音響信号処理 ② デジタル画像処理 ③ 音声処理 (3) デジタルフィルタの基礎 ① 線形／因果性／時不変 ② 安定／不安定フィルタ ③ 有限インパルス応答（FIR）と無限インパルス応答（IIR）	10H 合計36H
使用する機械器具等		

授 業 科 目 カ リ キ ュ ラ ム 表

科　　名	生産電子情報システム技術科	教科の区分	専攻学科
教 科 の 科 目	複合電子回路設計		
授　業　科　目	デジタル通信技術	単　　位	2
教 育 訓 練 目 標	デジタル通信システムの主要技術であるデジタル変復調技術、伝送技術を学習する。		

授業科目の細目	授　業　科　目　の　内　容	訓 練 時 間
1. 通信システムの概要	(1) 通信システムの方式とその構成 (2) 通信システムの構成要素	2H
2. デジタル変復調技術	(1) アナログ変調とデジタル変調 (2) 標本化・量子化・符号化 (3) パルス符号変調（PCM） (4) デジタル無線通信 ① 送信側の回路構成 ② 受信側の回路構成 ③ 周波数変換 ④ フェージング ⑤ マルチパス (5) ビットエラーレート	8H
3. 変復調方式	(1) 振幅偏移変調（ASK） (2) 周波数偏移変調（FSK） (3) 位相偏移変調（PSK） (4) 直交振幅変調（QAM）	１０H
4. 多重化・多重伝送	(1) 周波数分割多元接続（FDMA） (2) 時分割多元接続（TDMA） (3) 符号分割多元接続（CDMA） (4) 直交周波数分割多重（OFDM） (5) 波長分割多重（WDM）	１２H

授業科目カリキュラム表

授業科目の細目	授業科目の内容	訓練時間
5. その他の無線データ伝送	(1) Bluetooth (2) ZigBee (3) Global Positioning System (GPS)	4H 合計３６H
使用する機械器具等		

授業科目カリキュラム表

科名	生産電子情報システム技術科	教科の区分	専攻学科
教科の科目	セキュア通信システム設計		
授業科目	通信プロトコル実装設計	単位	2
教育訓練目標	ハードウェアの通信制御とプロトコルスタックを利用してデータを送受信する仕組みを理解し、組込み機器のプロトコル実装設計について学習する。		

授業科目の細目	授業科目の内容	訓練時間
1. LANとプロトコル	(1) LANプロトコルの知識と仕組み (2) 下位層プロトコル (3) 上位層プロトコル	4H
2. プロトコルスタック	(1) NICの役割とEthernetコントローラの仕組み (2) プロトコルとEthernet/IEEE802.3フレーム (3) パケット送受信の仕組み (4) パケットモニタによるプロトコル解析	4H
3. プロトコルの実装設計	(1) TCP/IPによるハードウェア制御の仕組み (2) IP、UDP、ARP、ICMP要件分析 (3) ネットワークアプリケーションの設計 (4) 信頼性を考慮した設計技法	16H
4. 無線LANの仕組み	(1) 無線LANの知識と仕組み (2) 無線LANのマイコン実装について	4H
5. その他のネットワーク	(1) 車載ネットワークのメカニズム (2) LINプロトコルと設計法 (3) CANプロトコルと設計法	8H
		合計36H
使用する機械器具等		

授業科目カリキュラム表

1／1

科名	生産電子情報システム技術科	教科の区分	専攻学科
教科の科目	セキュア通信システム設計		
授業科目	セキュアシステム設計	単位	2
教育訓練目標	通信機器やインフラに合わせたセキュリティの現状と対策を理解し、セキュアなネットワーク設計及びそのシステム構築・運用・管理について学習する。		

授業科目の細目	授業科目の内容	訓練時間
1. ネットワークシステム概要	(1) 情報通信ネットワークの基礎知識 (2) デジタル伝送技術と光通信技術 (3) ネットワークシステムの形態	2H
2. ネットワーク設計概要	(1) ネットワークシステム設計の概要 (2) ネットワークシステム設計手順	6H
3. 要件分析フェーズ	(1) 対象領域の決定 (2) 性能・機能・セキュリティの目標設定 (3) 調査・事前確認項目の把握	6H
4. 概要設計フェーズ	(1) ネットワークシステム概要設計 (2) ネットワークシステム構成設計 (3) 機器選定 (4) トラフィックの見積り (5) 回線容量設計、冗長構成 (6) セキュリティ確保 (7) 無線機器の使用法	8H
5. 詳細設計フェーズ	(1) レイアウト設計 (2) ネットワーク機器構成の設計 (3) セキュリティを考慮した機器の設置条件	8H
6. 導入と運用管理	(1) 運用の基礎知識 (2) 障害時の対処方法	6H 合計36H
使用する機械器具等		

授業科目カリキュラム表

科名	生産電子情報システム技術科	教科の区分	専攻学科
教科の科目	組込みシステム設計		
授業科目	組込みシステム設計	単位	4
教育訓練目標	組込みリアルタイムシステムの概要を理解し、組込みOSの機能及び製造現場の用途に応じた組込みシステムの構築技法を学習する。		

授業科目の細目	授業科目の内容	訓練時間
1. リアルタイムシステム概論	(1) リアルタイムシステムの概要 (2) リアルタイムシステムの特徴と適用例	4H
2. 組込みOSの基礎	(1) リアルタイムOSの概要と特徴 (2) リアルタイムOSの開発環境	4H
3. 組込みOSの機能とAPI	(1) タスク管理／タスク間同期・通信 ① タスク・スケジューリング ② セマフォ ③ デッドロックの発生とその回避法 ④ メールボックス (2) 割込み管理 (3) 資源管理 (4) 通信・ネットワーク	32H
4. 組込みシステム構築技法	(1) 組込みシステム開発フロー ① システム要求分析 ② 概要設計・詳細設計 ③ 実装技法 ④ テスト手法 ⑤ 運用・管理技法 (2) 組込みシステムの設計演習 ① システム要求分析 ② 概要設計	32H 合計72H
使用する機械器具等		

生産システム技術系 【生産電子情報システム技術科】

【専攻実技】 標準カリキュラム（案）

授業科目カリキュラム表

科名	生産電子情報システム技術科	教科の区分	専攻実技
教科の科目	機械工作実習		
授業科目	機械工作・組立実習	単位	4
教育訓練目標	筐体設計に必要とされる機械図面の読み方と加工図面にそった機械部品の加工、組立及び検査の方法を習得する。		

授業科目の細目	授業科目の内容	訓練時間
1. 加工図面の読み方	(1) 加工図面の読み方	2H
2. 旋盤加工	(1) 機械の取り扱い ① 旋盤による外形加工、段付き加工等による部品加工	24H
3. フライス盤加工	(1) 機械の取り扱い ① エンドミル加工、ボーリング加工等による部品加工	24H
4. その他の加工	(1) ボール盤作業 (2) 手仕上げ加工	12H
5. 組立	(1) 伝達機構の組立 ① 組立と加工精度 ② 伝達機構の組立	8H
6. 安全作業	(1) 危険防止、メンテナンス	2H
		合計72H
使用する機械器具等	旋盤、フライス盤、その他関連器工具	

授業科目カリキュラム表

科　　名	生産電子情報システム技術科	教科の区分	専攻実技
教科の科目	設計プロセス応用実習		
授業科目	実装設計製作実習	単　位	4
教育訓練目標	プリント基板の設計・製作に必要とされる作成全般について学び、部品の実装方法や配線設計方法を習得する。		

授業科目の細目	授業科目の内容	訓練時間
1. CADシステム	(1) CADシステム ① システムの概要 ② 基本操作実習	２０Ｈ
2. 回路図入力	(1) 回路図入力 ① 回路図作成 ② パーツリスト作成と追加 ③ 回路チェックとネットリスト ④ シミュレータによる解析	１６Ｈ
3. 伝送回路設計	(1) 伝送回路設計 ① 伝送回路の模擬実験 ② パターン回路設計のノウハウ	１６Ｈ
4. プリント基板設計	(1) 基板設計 ① 基板外形図 ② 部品配置、回路図の自動配置 (2) 配線設計 ① グランドパターンの設計 ② ベタパターンの活用 ③ アートワーク設計とバイパスコンデンサ ④ 配線の検証	２０Ｈ 合計７２Ｈ
使用する機械器具等	CAD／CAMシステム	

授業科目カリキュラム表

科名	生産電子情報システム技術科	教科の区分	専攻実技
教科の科目	設計プロセス応用実習		
授業科目	EMC応用実習	単位	4
教育訓練目標	部品の形状や特性を考慮した選定法、配線材料選定法、放熱や信号干渉等を考慮した実装設計法、組立法及び検査法を習得する。		

授業科目の細目	授業科目の内容	訓練時間
1. EMC設計	(1) EMC設計の概要 ① 設計仕様に基づくコンセプト確認	8H
2. CAM	(1) CAMによるプリント基板加工 ① 加工機用データフォーマット ② CAM操作による加工法 ③ パターンチェック ④ 部品実装、基板製作	20H
3. 回路シミュレーション	(1) 回路シミュレーションソフトウェアの活用法 ① クロストーク解析 ② リンギング解析 (2) 電界強度シミュレーション ① 電磁ノイズ解析、近傍電界強度解析	16H
4. 測定実験	(1) 試作基板の測定実験 ① クロストーク測定 ② リンギング測定 (2) 試作基板のEMC測定 ① 電磁ノイズ、近傍電界強度の測定	20H
5. 評価	(1) 製品の評価 ① 部品実装精度、はんだ付け、配線余長等 ② 問題点とその対策	8H 合計72H
使用する機械器具等	CAD/CAMシステム、基板加工機またはエッチング装置等、ボール盤、各種測定器	

授業科目カリキュラム表

1／1

科　　名	生産電子情報システム技術科	教科の区分	専攻実技
教科の科目	複合電子回路設計応用実習		
授業科目	複合電子回路設計製作実習	単　　位	6
教育訓練目標	アナログ回路、デジタル回路、高周波回路、通信回路技術を基に、移動体通信に関連する回路コンポーネントの設計手法と利用方法を習得する。		

授業科目の細目	授業科目の内容	訓練時間
1. 回路コンポーネント	(1) パッシブコンポーネントとアクティブコンポーネント ① バンドパスフィルタ、増幅器、発振器、減衰器、ミキサ、検波器、アンテナ	１２H
2. 測定器の使用法	(1) ファンクションジェネレータとシグナルジェネレータ (2) FFTアナライザとスペクトラムアナライザ (3) ネットワークアナライザ	２４H
3. 回路シミュレーション	(1) AC解析 (2) トランジェント解析 (3) Sパラメータ解析	３６H
4. 回路コンポーネントの利用と評価	(1) フィルタの反射損と挿入損 (2) アンプの入出力反射損と増幅度 (3) VCOの発振周波数特性 (4) アンテナ特性	２４H
5. システム動作実験・評価	(1) システム動作実験 ① 回路コンポーネントの接続試験 ② 不要発振抑制・スプリアス抑制 (2) システムの評価 ① 通信実験 ② 実波形による評価 ③ スペクトルによる評価 (3) 問題点とその対策	１２H 合計108H
使用する機械器具等	コンピュータシステム、高周波回路コンポーネント、高周波回路シミュレータ、各種測定器	

授業科目カリキュラム表

科名	生産電子情報システム技術科	教科の区分	専攻実技
教科の科目	複合電子回路設計応用実習		
授業科目	電子装置設計製作実習	単位	4
教育訓練目標	電子装置の設計・製作・評価を行い、ものづくりに関する基本的な手順を理解し、製品化技術を習得する。		

授業科目の細目	授業科目の内容	訓練時間
1. 設計	(1) 設計手法 ①設計コンセプト ②設計仕様に基づく電子回路設計 ③配線設計、筐体設計	6H
2. 回路製作	(1) 電子回路製作 ①電源回路 ②表示回路 ③回路実装 ④総合調整、動作試験	36H
3. 筐体加工・組立	(1) 筐体加工・組立 ①筐体加工 ②部品取付け、配線 ③総合調整、動作試験	24H
4. 評価	(1) 製品の評価 ①設計仕様との比較と完成度 ②問題点とその対策	6H
		合計72H
使用する機械器具等	CAD/CAMシステム、基板加工機またはエッチング装置等、ボール盤、各種測定器	

授業科目カリキュラム表

科名	生産電子情報システム技術科	教科の区分	専攻実技
教科の科目	複合電子回路設計応用実習		
授業科目	電子制御技術応用実習	単位	4
教育訓練目標	モータ駆動回路の設計・製作およびモータ制御プログラミングを学習し、モータの速度制御および位置決め制御方法を習得する。		

授業科目の細目	授業科目の内容	訓練時間
1. 設計手法	(1) 設計手法 ① 要求仕様に基づく設計 ② 評価項目の設定	6H
2. モータ制御ハードウェア	(1) インタフェース回路の設計・製作 (2) 電力変換回路の設計・製作 (3) モータ駆動回路の設計・製作	24H
3. モータ制御ソフトウェア	(1) I/O制御プログラミング (2) モータ制御基本プログラミング (3) 速度制御プログラミング (4) 位置決め制御プログラミング	24H
4. モータ制御システムの構築	(1) システムの統合 (2) システム動作テスト (3) 問題点とその対策	18H
		合計72H
使用する機械器具等	制御機器実験装置、マイコン開発環境一式、各種測定器	

授 業 科 目 カ リ キ ュ ラ ム 表

科　　名	生産電子情報システム技術科	教科の区分	専攻実技
教 科 の 科 目	セキュア通信システム構築応用実習		
授 業 科 目	通信プロトコル実装実習	単　　位	4
教 育 訓 練 目 標	リモートからの監視・制御を可能とするのに必要となるマイコンへのプロトコルスタック実装技術を習得する。		

授業科目の細目	授 業 科 目 の 内 容	訓 練 時 間
1. プロトコルスタック	(1) TCP/IPプロトコルスタックの概要 (2) TCP/IPプロトコルとヘッダの構造 (3) ソケットシステムコールによる処理 (4) パケットモニタリング実験 (5) TCP及びUDP通信プログラム	8H
2. 物理層	(1) イーサネットコントローラの動作原理 (2) イーサネットコントローラのプログラミング	8H
3. データリンク層	(1) MACフレーム実装プログラミング (2) パケットモニタリング (3) 電気信号計測（プリアンブル、コリジョン検出等）	8H
4. ネットワーク層	(1) ARPプロトコル実装と動作確認 (2) ICMPプロトコル実装と動作確認 (3) パケット分割 (4) ルーティング	１６H
5. トランスポート層	(1) UDPプロトコル実装（echoサーバ） (2) TCPプロトコル実装（echoサーバ、Webサーバ）	１２H
6. アプリケーション層	(1) リモート制御（LEDの点灯制御等） (2) リモート監視（スイッチ入力や温度・湿度の取得等）	２０H
		合計７２H

使 用 す る 機 械 器 具 等	ネットワーク接続機器一式、マイコン開発環境一式、各種測定器

授業科目カリキュラム表

科　　名	生産電子情報システム技術科	教科の区分	専攻実技
教科の科目	セキュア通信システム構築応用実習		
授業科目	セキュアシステム構築実習	単　　位	4
教育訓練目標	ユビキタス社会におけるネットワークシステムの構築及び運用管理を通して、セキュアなシステム構築技術を習得する。		

授業科目の細目	授業科目の内容	訓練時間
1. ネットワークシステム構築概要	(1) ネットワークの構成 (2) OSのインストール (3) ネットワーク機器の設定	8H
2. セキュリティの基本	(1) セキュリティポリシーの概要と設計 (2) 不要サービスの無効化 (3) アクセス制御とユーザ管理	12H
3. クライアントのセキュリティ	(1) ポートの制限 (2) ファイアウォールの考え方と設定 (3) 各種ウィルスソフトの導入	8H
4. サーバのセキュリティ	(1) ファイアウォール（ルータ）の考え方と設定 (2) 各種サーバサービスのセキュリティの考え方と設定 ① メールサーバ（暗号化、スパム） ② ファイルサーバ ③ Webサーバ（SSL） ④ リモートログイン（暗号化）	24H
5. 無線通信のセキュリティ	(1) アクセスポイントの構築 (2) SSID・MACフィルタリングの設定 (3) 暗号化機能(WEP、TKIP、AES)を設定	12H
6. 運用・管理	(1) ログファイルの監視と検証 (2) 侵入・検知とその対応	8H 合計72H
使用する機械器具等	ネットワーク接続機器一式、各種サーバ一式、クライアントPC一式、各種ソフトウェア一式	

授 業 科 目 カ リ キ ュ ラ ム 表

科　　名	生産電子情報システム技術科	教科の区分	専攻実技
教 科 の 科 目	組込みシステム構築応用実習		
授 業 科 目	組込みシステム構築実習	単　　位	6
教 育 訓 練 目 標	組込みOSの活用及びネットワークに対応した組込みソフトウェア技術を習得する。		

授業科目の細目	授 業 科 目 の 内 容	訓 練 時 間
1. ハードウェアとクロス開発環境	(1) ターゲットボードの概要 ① ハードウェアの仕様と動作 (2) クロス開発環境 ① 開発環境の構築 ② プログラミングデバッグ環境の習得	１２H
2. 組込みOSプログラミング	(1) 組込みOSシステムプログラミング ① システムコールとプロセス間通信 ② マルチタスクプログラミング ③ セマフォ、ミューテックス、タスク処理 (2) インタフェースプログラミング ① CF、SDメモリカードまたはUSBメモリの接続 ② 周辺デバイスからの時刻及び状態取得 (3) デバイスドライバの作成 ① USB機器の接続	４８H
3. マイコンネットワークプログラミング	(1) マイコンネットワークプログラミング実習 ① WWWサーバ構築 ② CGIアプリケーション制作 ③ 演習課題と評価	４８H
		合計108H
使 用 す る 機 械 器 具 等	ネットワーク接続機器一式、マイコン開発環境一式	

授業科目カリキュラム表

1／1

科　　名	生産電子情報システム技術科	教科の区分	専攻実技
教科の科目	組込みシステム構築応用実習		
授業科目	組込みデバイス設計実習	単　位	4
教育訓練目標	FPGA/CPLDを用いたデジタル回路の一連の開発フローを学習し、HDLによるデジタル回路設計技法を習得する。		

授業科目の細目	授業科目の内容	訓練時間
1. 開発環境	(1) 開発環境 ① 統合開発環境 ② シミュレータ	１２H
2. 開発フロー	(1) 回路デザインと論理検証 (2) 論理合成と配置配線 (3) タイミング検証	１２H
3. HDLによる回路設計	(1) HDLによる回路設計 ① 記述スタイル ② 組み合わせ回路 ③ 順序回路・演算回路・同期回路	１８H
4. 回路実装	(1) カウンタの製作 (2) マルチプレクサの製作 (3) シフトレジスタの製作 (4) デコーダの製作 (5) IPモジュールの活用	１８H
5. 評価と検証	(1) 各種回路の接続試験 (2) 評価と検証 (3) 問題点とその対策	１２H
		合計７２H

使用する機械器具等	コンピュータシステム、ターゲットボード、FPGA/CPLD開発環境一式

授 業 科 目 カ リ キ ュ ラ ム 表

科　　名	生産電子情報システム技術科	教科の区分	専攻実技
教 科 の 科 目	無線通信機器設計製作応用実習（標準課題実習）		
授 業 科 目	電子通信機器設計製作課題実習	単　　位	１０
教育訓練目標	無線通信機能を有した温度・湿度計測データロガー装置の設計・製作を通して、電子通信機器設計製作に必要な製品化技術を習得する。		

授業科目の細目	授 業 科 目 の 内 容	訓練時間
1. 基本設計	(1) 製作計画 ① 装置概要と機能 ② ハードウェア・ソフトウェア構成 ③ 製作手順・計画と役割分担 (2) ハードウェア（入出力）設計 ①入力デバイス、各種センサ ② 記憶デバイス ③ 表示装置 ④ 筐体設計 ⑤ 無線通信 (3) ソフトウェア設計 ① クロス開発環境の構築 ② CPUボード動作確認 ③ 機能分割設計 ④ テスト・サンプルプログラムの調査 (4) プレゼンテーション	２４H
2. 回路試作と動作実験	(1) 試作と実験 ①センサ周辺回路 ②表示回路 ③A/D変換回路 ④動作確認	２４H

授業科目カリキュラム表

授業科目の細目	授業科目の内容	訓練時間
3. ソフトウェア設計制作テスト	(1) 制御プログラムモジュールの制作 ① スタートアップ処理 ② リモート操作用モニタ ③ 外部メモリへのアクセス ④ 温度・湿度等取得処理 ⑤ 通信モジュール (2) 各プログラムのテスト	３０Ｈ
4. 回路設計製作	(1) プリント基板の設計製作 ① CAD/CAMによるPCB設計 ② プリント基板製作 ③ PCBの評価試験 ④ 部品実装	３０Ｈ
5. 筐体設計製作	(1) 筐体設計製作 ① 筐体選定 ② 筐体設計 ③ 筐体加工	１２Ｈ
6. 総合組立・試験調整	(1) 総合組立調整 ① 総合組立調整	８Ｈ
7. 性能試験	(1) 性能試験と検査表の作成 ① 動作試験と各部調整 ② 信頼性試験 ③ 検査表作成	２０Ｈ
8. マニュアル作成	(1) 製品マニュアルの作成 (2) 製品仕様書の作成	１６Ｈ
9. 報告・発表	(1) 報告書の作成 (2) 発表用資料作成 (3) 発表会の実施	１６Ｈ 合計180Ｈ
使用する機械器具等	コンピュータシステム、CAD/CAMシステム、マイコン開発環境一式、各種測定器、プレゼンテーション機器一式	

授 業 科 目 カ リ キ ュ ラ ム 表

科　　名	生産電子情報システム技術科	教科の区分	専攻実技
教 科 の 科 目	無線通信機器設計製作応用実習（標準課題実習）		
授 業 科 目	組込みシステム構築課題実習	単　　位	１０
教 育 訓 練 目 標	データ収集機能やセキュアなネットワーク機能を実装した組込みシステムの構築を通して、組込みソフトウェア開発、センサ制御、負荷装置制御等の統合的な技術を習得する。		

授業科目の細目	授　業　科　目　の　内　容	訓練時間
1. システム要件と製作計画	(1) システムの概要 (2) システム要件書の作成 ① システム構成と仕様の検討 ② 制御用端末の検討 ③ セキュリティと運用管理の検討 (3) 製作手順と役割分担	１２H
2. システム概要設計	(1) マイコンと周辺回路及び負荷装置の概要設計 ① インタフェースの検討 ② センサの検討 ③ 負荷装置の選択 (2) 制御用端末の概要設計 ① 0Sの検討 ② サーバサービスの検討 ③ セキュリティの検討 (3) アプリケーション概要設計 ① マイコン側ソフトウェア ② 制御用端末側ソフトウェア (4) 概要設計レビュー	１２H
3. システム詳細設計	(1) マイコンと周辺回路及び負荷装置の詳細設計 (2) 制御用端末の詳細設計 (3) アプリケーションの詳細設計 (4) 詳細設計レビュー	１２H

授業科目カリキュラム表

授業科目の細目	授業科目の内容	訓練時間
4. マイコン周辺機器の設計	(1) 周辺回路の設計 ① センサ ② 周辺装置 (2) 基板設計 (3) 基板加工 (4) 筐体設計製作	４０H
5. 制御用端末の構築	(1) OSインストール (2) 各種サーバサービスの設定 (3) セキュリティの設計構築	１８H
6. ソフトウェアの制作	(1) 開発環境の構築 (2) マイコン側プログラミング ① 単体テストとデバック (3) 制御用端末側プログラミング ① 単体テストとデバック (4) 通信プログラミング ① 単体テストとデバック	３６H
7. 性能試験	(1) 性能試験と検査表の作成 ① 動作試験と各部調整 ② 信頼性試験 ③ 検査表作成	１８H
8. マニュアル作成	(1) 製品マニュアルの作成 (2) 製品仕様書の作成	１６H
9. 報告・発表	(1) 報告書の作成 (2) 発表用資料作成 (3) 発表会の実施	１６H 合計180H
使用する機械器具等	コンピュータシステム、CAD/CAMシステム、マイコン開発環境一式、各種測定器、プレゼンテーション機器一式	

授業科目カリキュラム表

科名	生産電子情報システム技術科	教科の区分	応用
教科の科目	自動化機器等企画開発、生産システム設計・製作等実習(開発課題)		
授業科目	電子装置設計製作応用課題実習、組込みシステム応用課題実習、通信システム応用課題実習	単位	５４
教育訓練目標	生産現場を意識した「ものづくり」全工程の生産管理を主体的に行うことにより複合した技能・技術及びその活用能力（応用力、創造的能力、問題解決能力、管理的能力）を習得する。		

授業科目の細目	授業科目の内容	訓練時間
1. 開発課題の概要	(1) 開発課題の概要と基本方針 (2) 生産現場の工程管理（労務・コスト・納期等）	
2. 調査・企画	(1) 製品開発のためのニーズ調査 (2) 専門分野ごとの技術要素編成の設定 (3) 企画書の作成 (4) 企画書発表・修正	
3. 基本設計	(1) 基本設計書の作成 ①電子部の要求に対する仕様書の作成 ②仕様書に基づいたシステム設計 ③システムに基づいたブロック図の作成 (2) 基本工程表・基本見積書の作成 (3) 基本設計発表・修正	
4. 詳細設計	(1) 詳細設計書の作成 ①ブロックごとの機能設計 ②ブロックごとのインタフェース設計 ③ハードウェア・ソフトウェアの詳細設計 (2) 詳細工程表・詳細見積書の作成 (3) 詳細設計発表・修正	
5. 製作	(1) 各部の製作 ①ハードウェアの製作 ②ソフトウェアの制作	

授業科目カリキュラム表

授業科目の細目	授業科目の内容	訓練時間
6. 単体テスト	(1) 電子部の単体テスト・検査	
7. 統合テスト	(1) 機械部・電子部・情報部の統合組立 (2) 総合動作試験	
8. 製品評価・改善	(1) 製品の評価 (2) 製品の改善	
9. マニュアル作成	(1) 製品マニュアルの作成 (2) 製品仕様書の作成	
10. 報告・発表	(1) 報告書の作成 (2) 発表用資料作成 (3) 発表会の実施	合計972H
使用する機械器具等	コンピュータシステム、CAD/CAMシステム、マイコン開発環境一式、各種測定器、プレゼンテーション機器一式、その他	

資料 2

生産電子情報システム技術科

標準訓練支援計画書（案）

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	機械工学概論	必須	3、4期	2	2
教科の区分	専攻学科					
教科の科目	機械工学概論					

担当教員	内線電話番号	電子メールアドレス	

授業科目に対応する業界・仕事・技術

機械製造業における設計・加工・組み立て業務や加工オペレータ

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
機械要素、メカニズム、機械材料、製図、工作機械等の機械加工に関する基礎的な知識とCAD／CAMへの展開方法について学習します。	①	機械工学の概要について知っている。
	②	機械製図（図形の表し方、図示法、寸法、各種記号等）について知っている。
	③	機械要素（ねじ、締結部品、軸、軸受け、ばね、歯車、ベルト、プーリー、カム、リンク、クランク等）について知っている。
	④	機械材料（機械材料の性質、材料試験法、鉄鋼材料、非金属材料、プラスチック）について知っている。
	⑤	材料力学（荷重・応力・ひずみ、はりの曲げと応力、はりの曲げモーメント、はりの変形等）について知っている。
	⑥	機械加工法（切削加工、研削加工、塑性加工、溶接加工、特殊加工等）について知っている。
	⑦	油圧・空気圧制御（油圧・空気圧の原理、機器、図記号、回路等）について知っている。
	⑧	生産システム（CAD／CAM／CAE、自動計測・検査システム、自動組立システムおよびFMS等）について知っている。

授業科目受講に向けた助言

予備知識・技能技術	専門課程で学習した「物理」を復習しておいてください。
授業科目についての助言	現在の機械は制御や情報などの技術が融合し構成され、自動化やネットワーク化に対応したものとなっており、機械技術について学ぶことは、電子技術者や情報技術者にとっても重要である。この授業で、機械全般の工学について知識を習得する。
教科書および参考書	テキスト：「 図解入門よくわかる最新機械工学の基本 」 秀和システム
授業科目の発展性	機械工学概論 → 開発課題

評価の割合

評価方法 ／ 指標・評価割合		試験	小テスト	レポート	制作物	成果発表	その他	合計
		40		50			10	100
評価割合	授業内容の理解度	30		50				
	技能・技術の習得度							
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力	10						
	取り組む姿勢・意欲						10	
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス (1)シラバスの提示と説明 (2)オリエンテーション	講義、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認してください。
2週	2．機械の概要	講義、質疑	機械の要素について復習してください。
3週		講義、質疑	機械の要素について復習してください。
4週	3．機械材料 (1)金属材料 (2)非金属材料	講義、質疑	金属材料及び非金属材料の種類と特徴について復習してください。
5週	(3)複合材料	講義、質疑	複合材料の種類と特徴について復習してください。
6週	(4)材料試験 (5)熱処理	講義、質疑	材料試験、熱処理について復習してください。
7週	4．材料力学 (1)応力 (2)ひずみ (3)機械的性質	講義、質疑	各種材料の応力、ひずみ、機械的性質について復習してください。
8週	(4)はり (5)断面係数 (6)伸び	講義、質疑	各種材料のはり、断面係数、伸びについて復習してください。
9週	5．機械要素 (1)締結用部品 (2)伝動装置 (3)流体装置 (4)その他	講義、質疑	締結用部品の種類や、電動装置、流体装置などの特徴について復習してください。
10週	6．機械設計 (1)機械の設計法 (2)設計事例	講義、質疑	機械の設計法について復習してください。
11週	7．機械製図 (1)三角法、線、図記号、断面、その他	講義、質疑	機械製図の基本について復習してください。
12週	8．機械加工 (1)切削加工 (2)研作加工 (3)特殊加工	講義、質疑	切削加工、研削加工、特殊加工の基本について復習してください。
13週	(4)塑性加工 (5)接合加工法 (6)その他	講義、質疑	塑性加工、接合加工法の基本について復習してください。
14週	9．空気圧制御 (1)空気圧の原理 (2)空気圧機器 (3)図記号 (4)回路	講義、質疑	空気圧制御の基本について復習してください。
15週	10．油圧制御 (1)油圧の原理 (2)油圧機器 (3)図記号 (4)回路	講義、質疑	油圧制御の基本について復習してください。
16週	11．生産システム	講義、質疑	生産システムの基本について復習してください。
17週		講義、質疑	生産システムの基本について復習してください。
18週	12．期末試験	試験	試験に備え、これまで学習してきた内容について再確認して臨んでください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	アナログ電子回路設計	必修	1、2期	2	2
教科の区分	専攻学科					
教科の科目	複合電子回路設計					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

電子情報関連企業、電子回路設計技術者、アナログ電子回路設計技術。

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
無線通信機等に用いられる増幅回路、発振回路、スイッチ回路、フィルタ回路を中心に、アナログ電子回路の設計手法について学習する。	①	受動部品、能動部品の特性について知っている。
	②	半導体デバイスの取り扱いについて知っている。
	③	雑音に関する知識について知っている。
	④	増幅の原理と回路設計について知っている。
	⑤	周波数変換の原理と回路設計について知っている。
	⑥	フィルタ回路の伝達関数と設計について知っている。
	⑦	検波の原理と回路設計について知っている。
	⑧	発振の原理と回路設計について知っている。
	⑨	
	⑩	

授業科目受講に向けた助言

予備知識・技能技術	専門課程で学んだ、電気回路基礎、電子回路基礎、半導体基礎を前提知識とします。
授業科目についての助言	電気回路論をベースに、電子デバイスの活用技術と電子回路設計へと展開します。
教科書および参考書	教科書:わかるアナログ電子回路(日新出版) 参考書:アナログ電子回路(培風館)
授業科目の発展性	アナログ電子回路設計 ― 電子装置設計製作実習

評価の割合

評価方法 / 指標・評価割合		試験	小テスト	レポート	制作物	成果発表	その他	合計
			100		0	0	0	100
評価割合	授業内容の理解度		30					
	技能・技術の習得度		30					
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力		30					
	取り組む姿勢・意欲		10					
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1. ガイダンス (1)シラバスの提示と説明 2. 増幅回路の設計 (1) 受動部品の役割	講義、質疑	周波数特性、インピーダンスについて復習してください。
2週	(2)電子デバイスの役割	講義、質疑	直流バイアスと交流信号について復習してください。
3週	(3)雑音と増幅	講義、質疑	熱抵抗雑音、雑音指数について復習してください。
4週	(4)小テスト	講義、質疑	1～3週授業内容の理解度をチェックします。授業内容を復習しておいてください。
5週	3. 周波数変換回路の設計 (1)周波数変換の原理	講義、質疑	混合、変調の基本について復習してください。
6週	(2)周波数変換回路の役割	講義、質疑	中間周波数、AM、FM、PMについて復習してください。
7週	(3)周波数変換回路の種類	講義、質疑	パッシブミキサ、アクティブミキサについて復習してください。
8週	(4)小テスト	講義、質疑	5～7週授業内容の理解度をチェックします。授業内容を復習しておいてください。
9週	4. フィルタ回路の設計 (1)フィルタの原理	講義、質疑	伝達関数について復習してください。
10週	(2)フィルタ回路の種類	講義、質疑	ローパス、ハイパス、バンドパスについて復習してください。
11週	(3)周波数特性	講義、質疑	振幅特性、位相特性について復習してください。
12週	(4)小テスト	講義、質疑	9～11週授業内容の理解度をチェックします。授業内容を復習しておいてください。
13週	5. 検波回路の設計 (1)検波用デバイスと原理	講義、質疑	ショットキーダイオードについて復習してください。
14週	(2)検波回路の種類	講義、質疑	AM、FM、PM検波回路について復習してください。
15週	6. 発振回路の設計 (1)発振の原理	講義、質疑	正帰還と発振条件について復習してください。
16週	(2)発振回路の種類	講義、質疑	CR発振回路、LC発振回路について復習してください。
17週	(3)制御発振回路	講義、質疑	VCOの基礎について復習してください。
18週	(4)小テスト	講義、質疑	13～17週授業内容の理解度をチェックします。授業内容を復習しておいてください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	デジタルデバイス設計	必須	5、6期	2	2
教科の区分	専攻学科					
教科の科目	複合電子回路設計					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

ハードウェア記述言語によるデジタル回路設計
組込み機器に関する業界全般

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
ハードウェア記述言語（HDL）による大規模デジタル回路の実践的かつ効率的な設計の手法について学習します。	①	論理回路設計について知っている。
	②	ハードウェア記述言語(HDL)による回路設計について知っている。
	③	論理シミュレータについて知っている。
	④	論理合成ツールについて知っている。
	⑤	カスタムICの種類と特徴、応用例について知っている。
	⑥	HDLの実装テストについて知っている。
	⑦	
	⑧	
	⑨	
	⑩	

授業科目受講に向けた助言	
予備知識・技能技術	専門課程の「デジタル回路技術」を理解していることを前提としています。
授業科目についての助言	この授業では、組合わせ回路や順序回路の論理回路を、ハードウェア記述言語で設計する方法を学びます。ハードウェア記述言語は、組込みデバイス設計実習で利用します。毎回の授業をしっかり受講し、遅刻・欠席をしないようにしてください。また、わからないことは積極的に質問して積み残さないようにしてください。
教科書および参考書	教科書：自作テキスト
授業科目の発展性	組込みシステム設計 → 組込みシステム構築実習 → 組込みシステム構築課題実習 デジタルデバイス設計 → 組込みデバイス設計実習 → 組込みシステム構築課題実習

評価の割合

評価方法 ＼ 指標・評価割合	試験	小テスト	レポート	制作物	成果発表	その他	合計
	80					20	100
評価割合 授業内容の理解度	60						
技能・技術の習得度							
コミュニケーション能力							
プレゼンテーション能力							
論理的な思考力、推論能力	20						
取り組む姿勢・意欲						20	
主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス （1）シラバスの提示と説明 2．カスタムIC概要 （1）CPLDの種類と特徴、応用例 （2）FPGADの種類と特徴、応用例 （3）ASICの種類と特徴、応用例	講義、質疑	CPLD、FPGA、ASICとは何か調べてきてください。
2週	3．論理回路設計の基礎 （1）組合せ回路	講義、質疑	専門課程で学んだ「デジタル回路技術」の基本ゲート回路を復習してください。
3週	（2）組合せ回路	演習、質疑	専門課程で学んだ「デジタル回路技術」の組合わせ論理回路を復習してください。
4週	（3）フリップフロップ回路	講義、演習	専門課程で学んだ「デジタル回路技術」のフリップフロップ回路を復習してください。
5週	（4）順序回路	講義、質疑	専門課程で学んだ「デジタル回路技術」のカウンタ回路を復習しておいてください。
6週		演習、質疑	専門課程で学んだ「デジタル回路技術」のシフトレジスタ回路を復習してください。
7週	4．ハードウェア記述言語（HDL)を用いた回路設計 （1)マルチプレクサ	講義、演習、質疑	マルチプレクサについて復習してください。
8週		講義、演習、質疑	マルチプレクサについて復習してください。
9週	（2）デコーダ	講義、演習、質疑	デコーダについて復習してください。
10週		講義、演習、質疑	デコーダについて復習してください。
11週	（3）カウンタ	講義、演習、質疑	カウンタについて復習してください。
12週		講義、演習、質疑	カウンタについて復習してください。
13週	（4）シフトレジスタ	講義、演習、質疑	シフトレジスタについて復習してください。
14週		講義、演習、質疑	シフトレジスタについて復習してください。
15週	5．HDLの実装テスト （1）論理合成 （2）配置配線 （3）デバイスコンフィギュレーション	講義、演習、質疑	IPモジュールを活用した回路について復習してください。
16週		講義、演習、質疑	IPモジュールを活用した回路について復習してください。
17週		講義、演習、質疑	IPモジュールを活用した回路について復習してください。
18週	6．試験	試験	試験に備え、これまで学習してきた内容について再確認して臨んでください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	センサ応用技術	必須	1、2期	2	2
教科の区分	専攻学科					
教科の科目	複合電子回路設計					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

電気・電子機器製造、FAシステム製造に関連する開発・設計・生産・保守等の業務

授業科目の訓練目標

授業科目の目標	No	
磁気センサ、光センサ、温度センサ、生体センサ等のセンサ応用回路とセンシング技術の産業への活用について学習します。	①	センシング技術 システムでの位置づけと物理基準(精度、分解能、特性)について知っている。
	②	磁気センサ　磁気センサ応用回路について知っている。
	③	光センサ　光センサ応用回路について知っている。
	④	物理センサ 物理量をセンシングするセンサ応用回路について知っている。
	⑤	温度センサ　温度センサ応用回路について知っている。
	⑥	湿度センサ　湿度センサ応用回路について知っている。
	⑦	変位センサ　変位センサ応用回路について知っている。
	⑧	加速度センサ　加速度センサ応用回路について知っている。
	⑨	生体センサ 生体センサ技術について知っている。
	⑩	A/D変換技術 A/D変換回路について知っている。

授業科目受講に向けた助言

予備知識・技能技術	専門課程で学んだ電気・電子回路および計測制御について理解しておく必要があります。
授業科目についての助言	センサを効果的に使用するための電子回路やその構成方法などについて、実用回路事例を学習します。
教科書および参考書	教科書:図解使えるセンサ回路設計法(総合電子出版)　参考書:センサのしくみ(電波新聞社)
授業科目の発展性	アナログ電子回路設計 センサ応用技術 デジタルデバイス設計 → 電子通信機器設計製作課題実習(標準課題) → 開発課題

評価の割合

評価方法 ＼ 指標・評価割合		試験	小テスト	レポート	制作物	成果発表	その他	合計
		80					20	100
評価割合	授業内容の理解度	60						
	技能・技術の習得度							
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力	20						
	取り組む姿勢・意欲						20	
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1. ガイダンス (1)シラバスの提示と説明 2. センシング技術 (1)システムの位置づけ	講義、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認して下さい。
2週	(2)物理基準 ①精度 ②分解能 ③特性	講義、質疑	物理基準について復習してください。
3週	(3)センシング技術と産業	講義、質疑	センシング技術と産業について復習してください。
4週	3. 各種センサ応用回路 (1)磁気センサ応用回路	講義、質疑	磁気センサ応用回路の活用方法について復習してください。
5週		講義、質疑	磁気センサ応用回路の活用方法について復習してください。
6週		講義、質疑	磁気センサ応用回路の活用方法について復習してください。
7週	(2)光センサ応用回路	講義、質疑	光センサ応用回路の活用方法について復習してください。
8週		講義、質疑	光センサ応用回路の活用方法について復習してください。
9週		講義、質疑	光センサ応用回路の活用方法について復習してください。
10週	(3)温度、湿度、変位、加速度センサ応用回路	講義、質疑	物理センサを活用した回路の活用方法について復習してください。
11週		講義、質疑	物理センサを活用した回路の活用方法について復習してください。
12週		講義、質疑	物理センサを活用した回路の活用方法について復習してください。
13週		講義、質疑	物理センサを活用した回路の活用方法について復習してください。
14週	4. 生体センサ技術 (1)生体認証における生体情報とセンサ	講義、質疑	生体センサの種類と活用方法について復習してください。
15週		講義、質疑	生体センサの種類と活用方法について復習してください。
16週	5. A/D変換技術 (1)A／D変換の原理とA／D変換チップ	講義、質疑	センサを活用したA/D変換技術について復習してください。
17週	(2)A／D変換回路	講義、質疑	センサを活用したA/D変換回路について復習してください。
18週	6. 試験	講義、質疑	試験に備え、これまで学習してきた内容について再確認して臨んでください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	複合電子回路技術	必須	3、4期	2	2
教科の区分	専攻学科					
教科の科目	複合電子回路設計					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

電子回路を組込む電子機器製造業界

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
デジタルICとアナログ回路を同一システム上に実装するための技術を学習する。	①	デジタル回路とアナログ回路の混在システムについて知っている。
	②	電磁ノイズ対策について知っている。
	③	A/D変換回路とD/A変換回路について知っている。
	④	デジタル信号処理について知っている。
	⑤	デジタルフィルタについて知っている。
	⑥	
	⑦	
	⑧	
	⑨	
	⑩	

授業科目受講に向けた助言

予備知識・技能技術	専門課程の「電気回路」、「電子回路」、「電磁気学」で学んだ内容について理解しておく必要があります。
授業科目についての助言	アナログ、デジタル混在例、ノイズ対策、デジタル信号処理を通してノイズに強い回路設計手法を学習できます。
教科書および参考書	参考書:ディジタル信号処理（電機大出版局）
授業科目の発展性	アナログ電子回路設計／デジタルデバイス設計 → 複合電子回路技術 → 複合電子回路設計製作実習

評価の割合

評価方法 ＼ 指標・評価割合		試験	小テスト	レポート	制作物	成果発表	その他	合計
		100						100
評価割合	授業内容の理解度	50						
	技能・技術の習得度	10						
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力	20						
	取り組む姿勢・意欲	10						
	主体性・協調性	10						

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス （1）シラバスの提示と説明 2．デジタル・アナログ混在回路 （1）デジタル回路の基礎	講義、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認して下さい。
2週	（2）アナログ回路の基礎	講義、質疑	アナログ回路の基礎について復習してください。
3週	（3）デジタル回路とアナログ回路の混在システム例	講義、質疑	デジタル回路とアナログ回路の混在システムについて復習してください。
4週	3．電磁ノイズ対策 （1）電磁ノイズの基礎と種類 （2）電磁ノイズ対策	講義、質疑	電磁ノイズ対策について復習してください。
5週	4．信号のサンプリング	講義、質疑	信号のサンプリング手法について復習してください。
6週	5．A／D変換・D／A変換 （1）A／D変換回路の種類と特徴	講義、質疑	A／D変換回路の種類と特徴について復習してください。
7週		講義、質疑	A／D変換回路の種類と特徴について復習してください。
8週	（2）D／A変換回路の方式	講義、質疑	D／A変換回路の方式について復習してください。
9週	（3）D／A変換回路の活用事例	講義、質疑	D／A変換回路の活用事例について復習してください。
10週	6．デジタル信号処理 （1）デジタル信号処理の概要	講義、質疑	デジタル信号処理の概要について復習してください。
11週	（2）デジタルフィルタの基礎 ①線形システム ②時不変システムとたたみ込み	講義、質疑	デジタルフィルタの基礎について復習してください。
12週	③Z変換と逆Z変換	講義、質疑	Z変換と逆Z変換について復習してください。
13週	④有限インパルス応答（FIR）システム	講義、質疑	有限インパルス応答（FIR）システムについて復習してください。
14週	⑤無限インパルス応答（IIR）システム	講義、質疑	無限インパルス応答（IIR）システムについて復習してください。
15週	⑥システムの安定・不安定	講義、質疑	デジタルフィルタによるシステムの安定・不安定について復習してください。
16週	（3）フィルタ設計 ①FIRフィルタ設計	講義、質疑	FIRフィルタ設計について復習してください。
17週	②IIRフィルタ設計	講義、質疑	IIRフィルタ設計について復習してください。
18週	7．試験	講義、質疑	試験に備え、これまで学習してきた内容について再確認して臨んでください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	デジタル通信技術	必須	3、4期	2	2
教科の区分	専攻学科					
教科の科目	情報通信					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

デジタル通信回路を組込む電子機器製造業界

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
デジタル通信システムの主要技術であるデジタル変復調技術、伝送技術を習得します。	①	通信システムの概要について知っている。
	②	アナログ変調について知っている。
	③	デジタル変復調技術について知っている。
	④	変復調方式について知っている。
	⑤	多重化・多重伝送について知っている。
	⑥	
	⑦	
	⑧	
	⑨	
	⑩	

授業科目受講に向けた助言

予備知識・技能技術	アナログ無線通信の基本的な仕組みについて理解しておいてください。
授業科目についての助言	アナログ通信とデジタル通信の違い、そのシステムの主要技術であるデジタル変復調技術、伝送技術を学習します。
教科書および参考書	教科書：自 作テキスト
授業科目の発展性	デジタル通信技術 → 組込みシステム構築課題実習 → 開発課題

評価の割合

指標・評価割合 ＼ 評価方法		試験	小テスト	レポート	制作物	成果発表	その他	合計
		80	20					100
評価割合	授業内容の理解度	30						
	技能・技術の習得度							
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力	30	20					
	取り組む姿勢・意欲	20						
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス (1)シラバスの提示と説明 2．デジタル通信システムの概要 (1)通信システムの方式	講義、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認してください。
2週		講義、質疑	通信システムの方式について復習してください。
3週	(2)通信システムの構成	講義、質疑	通信システムの構成について復習してください。
4週	2．デジタル変復調技術 (1)アナログ変調の種類と特徴	講義、質疑	アナログ変調について、専門課程の授業内容を復習しておいてください。
5週	(2)デジタル変調の種類と特徴	講義、質疑	デジタル変調の種類と特徴について復習してください。
6週	(3)標本化・量子化・符号化	講義、質疑	標本化・量子化・符号化について復習してください。
7週	(4)パルス符号変調(PCM)	講義、質疑	パルス符号変調(PCM)について復習してください。
8週	(5)デジタル無線通信	講義、質疑	送信側の回路構成について復習してください。
9週		講義、質疑	受信側の回路構成について復習してください。
10週		講義、質疑	周波数変換、フェージング、マルチパスについて復習してください。
11週	(6)ビットエラーレート	講義、質疑	ビットエラーレートについて復習してください。
12週	3．変復調方式 (1)振幅偏移変調(ASK)	講義、質疑	振幅偏移変調(ASK)の特徴について復習してください。
13週	(2)周波数偏移変調(FSK)	講義、質疑	周波数偏移変調(FSK)の特徴について復習してください。
14週	(3)位相偏移変調(PSK)	講義、質疑	位相偏移変調(PSK)の特徴について復習してください。
15回	(4)直交振幅変調(QAM)	講義、質疑	直交振幅変調(QAM)の特徴について復習してください。
16週	4．多重化・多重伝送 (1)FDMA (2)TDMA (3)CDMA (4)OFDM (5)WDM	講義、質疑	多重化・多重伝送の種類と特徴について復習してください。
17週	5．まとめと評価	講義、質疑	小テストで理解度のチェックを行います。
18週	6．試験	講義、質疑	試験に備え、これまで実習してきた内容について再確認して臨んでください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	通信プロトコル実装設計	必須	3、4期	2	2
教科の区分	専攻学科					
教科の科目	セキュア通信システム設計					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

通信事業、組込み機器に関係する業界全般
ネットワークプログラミング技術

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
ハードウェアの通信制御とプロトコルスタックを利用してデータを送受信する仕組みを理解し、組込み機器のプロトコル実装設計について学習します。	①	LANとTCP/IPについて知っている。
	②	プロトコルスタックについて知っている。
	③	プロトコルの実装設計について知っている。
	④	無線LANの仕組みについて知っている。
	⑤	車載ネットワークについて知っている。
	⑥	
	⑦	
	⑧	
	⑨	
	⑩	

授業科目受講に向けた助言	
予備知識・技能技術	専門課程で学習した「ネットワーク技術」「移動体通信技術」について復習してください。
授業科目についての助言	本授業は専門課程で学んだ授業科目である「ネットワーク技術」「移動体通信技術」で学習した技術を更に深く学びます。特に、TCP/IP、プロトコル設計、無線LANのマイコン実装技術について学びます。
教科書および参考書	教科書：自作テキスト
授業科目の発展性	通信プロトコル実装設計 → 通信プロトコル実装実習 セキュアシステム設計 → セキュアシステム構築実習 → 組込みシステム構築課題実習

評価の割合

評価方法 ＼ 指標・評価割合		試験	小テスト	レポート	制作物	成果発表	その他	合計
		80					20	100
評価割合	授業内容の理解度	60						
	技能・技術の習得度							
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力	20						
	取り組む姿勢・意欲						20	
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス （1）シラバスの提示と説明 2．TCP/IP （1）物理層、リンク層、ARP	講義、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認してください。 専門課程での「ネットワーク技術」を復習してください。
2週	（2）ネットワーク層、経路制御、PING	講義、質疑	物理層、リンク層について復習してください。
3週	（3）ネットワーク層、パケット分割	講義、質疑	ネットワーク層、経路制御について復習してください。
4週	（4）トランスポート層、UDP、TFTP	講義、質疑	ネットワーク層、パケット分割について復習してください。
5週	（5）アプリケーション層、Webサーバ	講義、質疑	トランスポート層、UDPについて復習してください。
6週	3．TCP/IPによるハードウェア制御の仕組み	講義、質疑	アプリケーション層、Webサーバについて復習してください。
7週	4．プロトコル設計 （1）要求分析	講義、演習	TCP/IP全般を復習してください。
8週		講義、演習	TCP/IP全般を復習してください。
9週	（2）シーケンス図の作成	講義、演習	要求分析について復習してください。
10週	（3）状態遷移図の作成	講義、演習	要求分析について復習してください。
11週	（4）暗号技術の検討	講義、演習	要求分析について復習してください。
12週	（5）プロトコルの作成	講義、演習	要求分析について復習してください。
13週		講義、演習	要求分析について復習してください。
14週		講義、演習	要求分析について復習してください。
15週	5．無線LANの仕組み	講義、質疑	専門課程での「移動体通信技術」を復習してください。
16週	6．無線LANのマイコン実装	講義、質疑	専門課程での「移動体通信技術」を復習してください。
17週	7．車載ネットワーク	講義、質疑	専門課程での「移動体通信技術」を復習してください。
18週	8．試験	試験	試験に備え、これまで学習した内容について再確認して臨んでください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	セキュアシステム設計	必須	1、2期	2	2
教科の区分	専攻学科					
教科の科目	セキュア通信システム設計					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

電気通信事業や情報処理サービス事業におけるネットワークの構築・運用・設計に係わる技術
その他コンピュータシステムに係わる管理・運用技術

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
ネットワーク機器やインフラに合わせたセキュリティの現状と対策を理解し、セキュアなネットワーク設計及びそのシステム構築・運用・管理について学習します。	①	情報通信ネットワークの基礎について知っている。
	②	ネットワークシステム設計の概要について知っている。
	③	ネットワークシステム設計の手順について知っている。
	④	ネットワークシステムの性能・機能・セキュリティの目標設定について知っている。
	⑤	ネットワークシステムの概要設計について知っている。
	⑥	ネットワーク機器の選定について知っている。
	⑦	トラフィックの見積りと回線容量設計、冗長設計について知っている。
	⑧	セキュリティポリシーの作成について知っている。
	⑨	暗号技術について知っている。
	⑩	

授業科目受講に向けた助言

予備知識・技能技術	専門課程の「情報通信工学」「ネットワーク技術」で学習したコンピュータネットワークの各項目を復習してください。特にTCP/IPの構造と原理について再確認してください。
授業科目についての助言	専門課程の「情報通信工学」や「ネットワーク技術」で学習した基礎を更に深め、個々の技術を統合したネットワークシステムを設計します。特にインターネット利用が一般的となった企業内のセキュアなネットワーク設計について学習します。
教科書および参考書	教科書：自作テキスト 参考書：マスタリングTCP/IP 入門編（オーム社）
授業科目の発展性	通信プロトコル実装設計 → 通信プロトコル実装実習 セキュアシステム設計 → セキュアシステム構築実習 → 組込みシステム構築課題実習

評価の割合

評価方法 ／ 指標・評価割合		試験	小テスト	レポート	制作物	成果発表	その他	合計
		60		30			10	100
評価割合	授業内容の理解度	20		10				
	技能・技術の習得度	20		10				
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力	20		10				
	取り組む姿勢・意欲						10	
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス （1）シラバスの提示と説明 2．情報通信ネットワークの基礎 （1）LANの基礎	講義、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認してください。専門課程の「情報通信工学」「ネットワーク技術」について復習してください。
2週	（2）ネットワークの基礎	講義、質疑	専門課程の「情報通信工学」「ネットワーク技術」について復習してください。
3週	3．ネットワークシステム設計の概要	講義、質疑	ネットワークシステム設計の概要について復習してください。
4週		講義、質疑	ネットワークシステム設計の概要について復習してください。
5週	4．ネットワークシステム設計の手順	講義、質疑	ネットワークシステム設計の手順を復習してください。
6週		講義、質疑	ネットワークシステム設計の手順を復習してください。
7週	5．ネットワークシステムの性能・機能・セキュリティの目標設定	講義、質疑	LANの基礎やネットワークの基礎を復習してください。
8週		講義、質疑	LANの基礎やネットワークの基礎を復習してください。
9週	6．機器選定、トラフィック見積り	講義、質疑	機器選定、トラフィック見積りについて復習してください。
10週		講義、質疑	機器選定、トラフィック見積りについて復習してください。
11週	7．回線容量設計、冗長設計	講義、質疑	回線容量設計、冗長設計について復習してください。
12週	8．セキュリティポリシーの作成	講義、質疑	LANの基礎やネットワークの基礎を復習してください。
13週		講義、質疑	セキュリティポリシーの作成について復習してください。
14週	9．暗号技術 （1）RSA	講義、質疑	剰余計算について理解しておいてください。
15週		講義、質疑	暗号技術（RSA）について復習してください。
16週	（2）ハッシュ関数	講義、質疑	暗号技術（ハッシュ関数）について復習してください。
17週	（3）共通鍵暗号	講義、質疑	暗号技術（共通鍵暗号）について復習してください。
18週	10．試験	試験	試験に備え、これまで学習した内容について再確認して臨んでください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	組込みシステム設計	必須	1、2期	4	4
教科の区分	専攻学科					
教科の科目	組込みシステム設計					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

組込み機器に関係する業界全般
OSを用いた組込み機器開発技術
組込み機器のプログラミング技術

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
組込みシステムの概要を理解し、組込みOSの機能および製造現場の用途に応じた組込みシステムの構築技法を学習します。	①	組込みOSの概要と特徴について知っている。
	②	組込みOSの開発環境について知っている。
	③	プロセスについて知っている。
	④	シグナルについて知っている。
	⑤	メッセージキューについて知っている。
	⑥	共有メモリについて知っている。
	⑦	ソケットについて知っている。
	⑧	セマフォについて知っている。
	⑨	システム要求分析について知っている。
	⑩	概要設計・詳細設計について知っている。

授業科目受講に向けた助言

予備知識・技能技術	専門課程の「組込みシステム工学」「組込みソフトウェア応用実習」を理解していること。
授業科目についての助言	近年、多くの家電製品は組込みマイコンによって制御され、組込み技術者が不足しています。この授業では組込み技術者が必要とするマルチタスクプログラミングと設計手法について学習します。 この授業で学ぶ知識や技術は企業のみならず、標準課題や開発課題を受講する上でも必要です。将来、学習した知識を活用するためにも毎回の授業をしっかりと受講し、わからないことは積極的に質問して積み残さないようにしてください。
教科書および参考書	教科書:自作テキスト
授業科目の発展性	組込みシステム設計 → 組込みシステム構築実習 デジタルデバイス設計 → 組込みデバイス設計実習 組込みシステム構築実習・組込みデバイス設計実習 → 組込みシステム構築課題実習

評価の割合

評価方法 ＼ 指標・評価割合	試験	小テスト	レポート	制作物	成果発表	その他	合計
	80					20	100
評価割合 授業内容の理解度	80						
技能・技術の習得度							
コミュニケーション能力							
プレゼンテーション能力							
論理的な思考力、推論能力							
取り組む姿勢・意欲						20	
主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1. ガイダンス (1)シラバスの提示と説明 2. リアルタイムシステムの概要 (1)リアルタイムシステムの特徴と適用例	講義、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認してください。 専門課程の「組込みシステム工学」「組込みソフトウェア応用実習」を復習してください。
2週	3. リアルタイムOSの概要と特徴 (1)リアルタイムOSとLinux	講義、質疑	専門課程の「組込みシステム工学」「組込みソフトウェア応用実習」を復習してください。
3週	4. タスク管理／タスク間同期・通信 (1)プロセス	講義、質疑	プロセスの操作に関して復習してください。
4週		講義、質疑	プロセスを生成するシステムコールについて復習してください。
5週	(2)シグナル	講義、質疑	シグナルを操作する各種関数について復習してください。
6週		講義、質疑	マイクロ秒単位の処理について復習してください。
7週	(3)メッセージキュー	講義、質疑	メッセージキューの機能について復習してください。
8週		講義、質疑	メッセージキューの使い方について復習してください。
9週	(4)共有メモリ	講義、質疑	共有メモリの機能について復習してください。
10週		講義、質疑	共有メモリの使い方を復習してください。
11週	(5)ソケット	講義、質疑	ソケットの機能と使い方を復習してください。
12週	(6)セマフォ	講義、質疑	セマフォの機能について復習してください。
13週		講義、質疑	セマフォの使い方を復習してください。
14週	5. 組込みシステム開発フロー (1)システム要求分析	講義、質疑	要求仕様書の作成方法に関して復習してください。
15週	(2)概要設計・詳細設計	講義、質疑	システム仕様書に基づいた概要設計・詳細設計の方法に関して復習してください。
16週	(3)実装技法	講義、質疑	設計書に基づいた実装方法に関して復習してください。
17週	(4)テスト手法、概要設計・詳細設計	講義、質疑	検証(テスト)仕様書作成とテストの実施、組込みシステムの運用・管理に関して復習してください。
18週	6. 試験	試験	試験に備え、これまで学習してきた内容について再確認して臨んでください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	安全衛生管理	必須	3、4期	2	2
教科の区分	専攻学科					
教科の科目	安全衛生管理					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

安全衛生管理におけるすべての業務

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
機械設備の安全対策、作業者の安全対策、セーフティ・アセスメント、その他安全に関する規約と認証等について学びます。	①	安全管理の基本的なことについて知っている。
	②	機械設備の安全対策、作業者の安全対策について知っている。
	③	セーフティ・アセスメントについて知っている。
	④	製品安全について知っている。
	⑤	各種規約について知っている。
	⑥	認証について知っている。
	⑦	
	⑧	
	⑨	
	⑩	

授業科目受講に向けた助言	
予備知識・技能技術	専門課程で学習した「安全衛生工学」について復習しておいてください。
授業科目についての助言	安全管理の仕組みと安全な機器の使用方法および災害防止に配慮した設計・製作に関する安全管理対策を、ものづくり現場での事例より学び、ものづくりのリーダーとしての災害防止能力を学びます。
教科書および参考書	教科書：安全管理技術　工業調査会　(社)実践教育訓練研究協会　編
授業科目の発展性	安全衛生管理 → 各実習科目

評価の割合

指標・評価割合 ＼ 評価方法		試験	小テスト	レポート	制作物	成果発表	その他	合計
		60	10	20			10	100
評価割合	授業内容の理解度	50	10					
	技能・技術の習得度							
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力	10		20				
	取り組む姿勢・意欲						10	
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1. ガイダンス (1)シラバスの提示と説明 2. 安全管理の基本 (1)安全管理の意義と目標	講義、質疑	安全管理体制、労働災害の現状と問題点について理解すること
2週	(2)危険の防止	講義、質疑	災害発生メカニズム、不安全な状態と不安全な行動、災害原因分析について理解すること
3週	(3)企業経営と安全管理	講義、質疑	安全と生産、安全と経営損失の防止、経営者の安全責務、安全配慮義務について理解すること
4週	3. 危険の防止対策 (1)機械設備の安全対策 ①安全点検と安全対策	講義、質疑	機械設備のレイアウト、機械設備の安全条件、安全点検制度の検討、安全点検の結果に基づく欠陥の是正、安全点検の方法について理解すること
5週	②ケーススタディ『設置現場の環境を考慮したFA機器の安全設置』	講義、質疑	労働安全衛生法による機械設備基準、リスクアセスメントによる危険源の特定、そのリスクの見積と評価、リスク評価に基づく方策について理解すること
6週	(2)作業者の安全対策 ①作業方法の改善	講義、質疑	作業環境の安全化の進め方、作業分析の方法、作業標準の作り方について理解すること 例：運搬用等
7週	②作業環境・職場の改善	講義、質疑	個々の機械設備の安全化、人間工学からみた人間のエラー、安全装置、保護具等の意義と必要性、作業と適正な姿勢について理解すること
8週	③安全教育 4. 小テスト	講義、質疑	安全衛生教育(知識教育、技能教育、態度教育)、安全衛生法定教育(雇い入れ時の安全衛生教育ばど)、KYT(危険予知訓練)、安全衛生教育計画の立て方について理解すること。 試験範囲は第1週から第8週までです。テキストの内容を十分理解し不明な点を質問などで明らかにし、試験に臨んで下さい。
9週	5. セーフティ・アセスメント (1)セーフティ・アセスメントとは	講義、質疑	設計・レイアウト時の安全性審査、セーフティ・アセスメントの手順について理解すること　①機械の資料の整備、②定性的評価、③定性的評価を定量化、④危険性に対する対策検討、⑤④の再検討など
10週	(2)機械のセーフティ・アセスメント	講義、質疑	機械の 本質安全化、外観部の安全化(回転部分、突出部分、電気装置、機械の設備の色彩等)、機能面の安全化(フール・プルーフ、フェールセーフ、冗長回路等)について理解すること
11週	(2)機械のセーフティ・アセスメント	講義、質疑	機械設備のレイアウト時の安全性の検討、レイアウトの原則について理解すること
12週	6. 規約・認証 (1)製品設計における安全の考え方(規約) 安全を取り巻く国内外の動向、国際安全規格の解説	講義、質疑	安全を取り巻く国内外の動向、国際安全規格(ISO/IEC51)について理解すること ISO12100とISO14121の解説(A規格、B規格、C規格)について理解すること
13週	(2)安全に関する規格 ①安全の基本原則(A規格) ②共通に使える技術(B規格) ③個別規格(C規格) リスクアセスメントの概要	講義、質疑	厚生労働省発表「機械の包括的な安全基準に関する指針」について理解すること 機械の使用状況の特定、危険源の同定、リスクの見積、リスクの評価について理解すること
14週	(2)安全に関する規格 ①安全の基本原則(A規格) ②共通に使える技術(B規格) ③個別規格(C規格) リスクアセスメントの演習	講義、質疑	安全の基本(安全確認型システムと危険検出型システム)、確立安全と確定安全について理解すること
15週	(2)安全に関する規格 ①安全の基本原則(A規格) ②共通に使える技術(B規格) ③個別規格(C規格) リスクアセスメントの演習	講義、質疑	安全方策の検討、本質安全設計、設計によるリスクの低減について理解すること
16週	(4)適応事例	講義、質疑	安全防護及び追加安全方策、ガード(固定ガード、包囲ガード、隔離ガード、可動ガード、など)、インターロック付きガードなど、追加の保護方策(非常停止装置)について理解すること
17週	(5)認証	講義、質疑	使用上の情報(残留リスクについて)、安全監査の作り方について理解すること
18週	7. 試験	講義、質疑	試験に備え、これまで学習してきた内容について再確認して臨んでください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	機械工作・組立実習	必須	2期	4	4
教科の区分	専攻実技					
教科の科目	機械工学実験・実習					

担当教員	内線電話番号	電子メールアドレス	期

授業科目に対応する業界・仕事・技術

生産現場における加工・組み立て業務、生産現場における加工オペレータ、ラインオペレータ、製品開発における設計業務

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
製品製作（一軸テーブル）を通し、機械加工の基礎を実践的に習得する。具体的には・機械JIS規格製図及びCAD・機械加工機を使い材料特性と加工の重要性を習得する。	①	三角法の読み方、投影法、線の太さ、JIS規格ができる。
	②	JIS規格製図に基づく機械設計とCADによる製品設計ができる。
	③	熱処理、材料記号、はめ合い方式、幾何公差、表面粗さができる。
	④	測定具の使い方（ノギス、マイクロメータ、スケール、ダイヤルゲージ　等）ができる。
	⑤	安全作業方法について、旋盤の使い方（外径切削、内径切削）ができる。
	⑥	フライス盤の使い方（バイスの取付、工作物の取り付け方、6面体の削り方）ができる。
	⑦	エンドミル工具の特性、ドリルの特性、切削条件と切り粉の関係を知っている。
	⑧	ボール盤作業（穴あけ時の危険性）、面取りの重要性、ドリルの破損と欠損ができる。
	⑨	ケガキ作業、ハイとゲージの使い方、センターポンチの使い方、タップ立てができる。
	⑩	組立調整作業（製品検査、平行度の取付、締め付け調整　等）やすりのかけ方ができる。

授業科目受講に向けた助言

予備知識・技能技術	鋼及びALの材料特性について理解しておいて下さい。　工具材料別の切削条件の知識が必要です。
授業科目についての助言	実際の機械を使っての実習であるため、危険性があります。　不慣れからくる「赤チン」災害が想定されることから、指導者の助言を十分に聞いて作業に着手してください。また、工具の切削条件は非常に重要であり、工具の寿命に大きく影響することから、十分な検討するよう心がけてください。
教科書および参考書	教科書：自作テキスト
授業科目の発展性	機械工作・組立実習 → 電子装置設計製作実習 → 標準課題実習

評価の割合

評価方法 ＼ 指標・評価割合	試験	小テスト	レポート	制作物	成果発表	その他	合計
			20	80			100
評価割合 授業内容の理解度			20	10			
技能・技術の習得度				40			
コミュニケーション能力				20			
プレゼンテーション能力							
論理的な思考力、推論能力				10			
取り組む姿勢・意欲							
主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス （1）シラバスの提示と説明 2．JIS規格に基づく機械設計 （1）CADによる機械製図 ①一軸テーブルの仕様	実習、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認して下さい。
2週	②一軸テーブルのCAD ③基準面の取り方、寸法の取り方、寸法公差の入れ方	実習、質疑	2次元CAD操作に慣れて下さい。機械図面を描くルールを理解しましょう。
3週	④三角法の読み方 ⑤投影図 ⑥はめ合い記号　・幾何公差　・表面粗さの表示	実習、質疑	3次元形状から三面図に表すルールを理解しましょう。
4週	⑦材料記号　・熱処理　・溶接記号 ⑧組み立て図と部品図の照合	実習、質疑	材料の種類やそれを表す記号、熱処理や溶接といった加工法を理解し、適切な表し方を覚えましょう。
5週	⑨ノギス及びマイクロメータの読み方 ⑩スケール　・ダイヤルゲージの使い方	実習、質疑	長さ測定の基本であるノギス、マイクロメータの取り扱いや読み方を習熟しましょう。
6週	3．安全作業について （1）旋盤の使い方 ①外径切削加工 ②外径段付き加工 ③内径切削加工 ④切削条件の決定（送り量、切り込み量、回転数、切削油　等）	実習、質疑	旋盤のハンドル操作、保護めがね着用、正しい作業服の着用等遵守しなければならない最低限の安全作業法を理解し、習得しましょう。
7週	（2）フライス盤の使い方 ①バイスの取付方 ②工作物の取り付け方 ③正面フライスによる六面体切削 ④エンドミルによる溝削り	実習、質疑	フライス盤のハンドル操作、保護めがね着用、正しい作業服着用等遵守しなければならない最低限の安全作業法を理解し、習得しましょう。
8週	⑤加工面の取り方 ⑥工具の取扱（エンドミル） ⑦切削条件の吟味 ⑧自動送り機構	実習、質疑	材料の取り付け方法、工具の取り付け取り外し方法、切削条件の意味等を理解しましょう。
9週	（3）工具のアプローチについて ①寸法の決めかた ②六面体切削の基準面のとり方	実習、質疑	六面体加工の手順を理解し、工作物の取り付け方法と寸法の出し方を習熟しましょう。
10週	（4）ドリルによる穴あけ ①座ぐり加工 ②エンドミルによる段付加工 ③デップスマイクロメターによる測定	実習、質疑	安全作業に留意しながら加工しましょう。
11週	④ケガキ作業 ⑤ハイトゲージの使い方 ⑥センターポンチの打ち方 ⑦やすりによるバリ取り	実習、質疑	基準の取り方、ケガキ線の引き方、センタポンチ打ちの力加減等習熟しましょう。
12週	4．ボール盤作業 （1）センタードリルとドリル加工 ①加工作業 ②安全上の注意点 ③バリの除去	実習、質疑	ボール盤作業における安全作業法を確認して作業に取り組みましょう。
13週	5．精密旋盤による加工 （1）加工作業 ①溝入れ、面取り ②切削油剤の使い方	実習、質疑	旋盤作業の基本操作と安全作業法を再確認して作業に臨みましょう。
14週	6．タップ立て作業 （1）M2、M3、M4、M6、M8タップ立て作業	実習、質疑	径の細いタップ立ては慎重に行いましょう。常にタップが基準面に垂直になるよう注意しましょう。
15回	7．総部品の寸法検査と不具合部品の再加工	実習、質疑	各部品の寸法チェックと共に面取りが丁寧になされているか確認しましょう。
16週	8．組み立て調整作業	実習、質疑	各部品の勘合は無理な力をかけず慎重に行いましょう。
17週		実習、質疑	摺動部品は移動端までなめらかに動くよう、平行を確認するようにして下さい。
18週	9．確認試験	実習、質疑	動作確認と簡単な口頭試問を行います。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	実装設計製作実習	必修	1、2期	4	4
教科の区分	専攻実技					
教科の科目	設計プロセス応用実習					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

電子機器製造に関連する開発・設計・生産等の業務

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
プリント基板の設計・製作に必要とされる作成全般について学び、部品の実装方法や配線設計方法を習得します。	①	CADシステムを知り、CADの基本操作ができる。
	②	ライブラリへのシンボル・部品等の登録ができる。
	③	回路シミュレータが利用できる。
	④	回路検証とネットリストの抽出ができる。
	⑤	ガーバーフォーマットが生成できる。
	⑥	基板製作のための回路図入力、基板設計ができる。
	⑦	
	⑧	
	⑨	
	⑩	

授業科目受講に向けた助言

予備知識・技能技術	一般に使用される回路図用図記号とプリント基板製作の基本的な流れを理解しておいてください。
授業科目についての助言	プリント基板の設計と製作は、標準課題や開発課題において必須の技術です。CADシステムの基本操作を習得し、回路図入力から基板製作までを確実に習得してください。回路設計や基板設計の段階においても、基板製作を常に意識することが必要です。
教科書および参考書	教科書:自作テキスト 参考書:
授業科目の発展性	実装設計製作実習 ― 標準課題

評価の割合

指標・評価割合 ＼ 評価方法		試験	小テスト	レポート	制作物	成果発表	その他	合計
		0	0	20	60	0	20	100
評価割合	授業内容の理解度			20	20			
	技能・技術の習得度				40			
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力							
	取り組む姿勢・意欲						20	
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス (1)シラバスの提示と説明 2．CAD/CAMシステムの概要	講義、実習、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認してください。
2週	3．回路図の作成	実習、質疑	回路図の作成方法を復習してください。
3週	4．シンボルの登録と管理	実習、質疑	シンボルの登録とその管理方法を復習してください。
4週	5．回路チェック、パーツリスト生成、ネットリスト抽出	実習、質疑	回路チェック、パーツリスト生成、ネットリスト抽出方法を復習してください。
5週	6．基板外形、部品配置 7．配線設計	実習、質疑	基板外形、部品配置、配線設計方法を復習してください。
6週		実習、質疑	配線設計方法を復習してください。
7週	(1)グランドパターンの設計 ①ベタグランド、ベタパターンの活用	実習、質疑	グランドパターンの設計、ベタグランド、ベタパターンの活用方法を復習してください。
8週	(2)フォトデータ出力、ガーバーデータ出力	実習、質疑	フォトデータ出力、ガーバーデータ出力方法を復習してください。
9週	(3)部品ライブラリへのシンボル・部品の登録、部品管理	実習、質疑	部品ライブラリへのシンボル・部品の登録、部品管理方法を復習してください。
10週		実習、質疑	部品ライブラリへのシンボル・部品の登録、部品管理方法を復習してください。
11週		実習、質疑	部品ライブラリへのシンボル・部品の登録、部品管理方法を復習してください。
12週	(4)回路シミュレーション	実習、質疑	回路シミュレーションの方法を復習してください。
13週		実習、質疑	回路シミュレーションの方法を復習してください。
14週		実習、質疑	回路シミュレーションの方法を復習してください。
15週	8．課題演習	実習、質疑	回路図の作成、シンボルの登録、回路チェック、パーツリスト生成、ネットリスト抽出などの課題演習の復習してください。
16週		実習、質疑	基板外形、部品配置、配線設計などの課題演習の復習してください。
17週		実習、質疑	グランドパターンの設計、フォトデータ出力、ガーバーデータ出力、部品ライブラリへのシンボル・部品の登録、部品管理、回路シミュレーションなどの課題演習の復習してください。
18週	9．報告書作成	実習、質疑	報告書作成に関わる準備をしてください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	EMC応用実習	必須	5、6期	4	4
教科の区分	専攻実技					
教科の科目	設計プロセス応用実習					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

電子回路を組込む電子機器製造業界

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
部品の形状や特性を考慮した選定法、配線材料選定法、放熱や信号干渉等を考慮した実装設計法、組立法及び検査法を習得する。	①	ノイズを発生する原理を知っている。
	②	ノイズを考慮した回路・基板設計ができる。
	③	電磁界シミュレーションができる。
	④	伝送線路シミュレーションができる。
	⑤	基板製作と測定評価ができる。
	⑥	シミュレーション結果と実測結果の比較検討ができる。
	⑦	ノイズを考慮した基板設計、製作ができる。
	⑧	SI/EMC評価測定ができる。
	⑨	
	⑩	

授業科目受講に向けた助言

予備知識・技能技術	電子回路、複合電子回路、プリント基板設計に関する知識が必要です。
授業科目についての助言	シミュレーション、基板製作、測定と評価を通して、ノイズに強い回路設計法を習得します。
教科書および参考書	教科書:自作テキスト 参考書:高速ディジタル回路実装ノウハウ
授業科目の発展性	複合電子回路設計技術 複合電子回路設計製作実習 → EMC応用実習 → 開発課題

評価の割合

評価方法 ／ 指標・評価割合		試験	小テスト	レポート	制作物	成果発表	その他	合計
				50	50	0	0	100
評価割合	授業内容の理解度			20				
	技能・技術の習得度				20			
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力			10	10			
	取り組む姿勢・意欲			10	10			
	主体性・協調性			10	10			

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス (1)シラバスの提示と説明 2．ノイズの発生する原理	講義、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認してください。 ノイズの発生原理について復習してください。
2週	3．ノイズを考慮した回路・基板設計	講義、質疑	ノイズを考慮した回路設計上の注意点、基板設計上の注意点を復習してください。
3週	4．電磁界シミュレータの原理	講義、質疑	電磁界シミュレーションの必要性を復習してください。
4週	5．電磁界シミュレーション	実習、質疑	電磁界シミュレーションの操作について復習してください。また、ノイズとの関係についても復習してください。
5週	6．伝送線路シミュレータの原理	実習、質疑	伝送線路シミュレータの原理について復習してください。
6週	7．伝送線路シミュレーション	実習、質疑	伝送線路シミュレーションの操作について復習してください。また、シミュレーションの結果から伝送線路を評価する方法についても復習してください。
7週	8．評価基板の測定評価	実習、質疑	評価基板の測定評価方法について復習してください。
8週	9．シミュレーション結果と実測結果の比較検討	実習、質疑	シミュレーション結果と実測結果の比較検討について、問題点とシミュレーションの利用法について復習してください。
9週	10．ノイズを考慮した基板設計	実習、質疑	CADの操作について復習してください。 1週から8周までの内容を復習してください。
10週		実習、質疑	GNDの取り方について復習してください。 1週から8周までの内容を復習してください。
11週	11．ノイズを考慮した基板設計のシミュレーション	実習、質疑	電磁界シミュレーション、伝送線路シミュレーションについて復習してください。
12週		実習、質疑	電磁界シミュレーション、伝送線路シミュレーションについて復習してください。
13週	12．ノイズを考慮した基板製作	実習、質疑	実装設計製作における、基板製作について復習しておくこと。
14週		実習、質疑	実装設計製作における、基板製作について復習してください。
15週		実習、質疑	ノイズを考慮した基板製作の注意点等について要点をまとめてください。
16週	13．SI/EMC評価測定	実習、質疑	SI/EMCについて復習してください。
17週		実習、質疑	SI/EMCの評価測定についてまとめてください。
18週	14．報告書作成	実習、質疑	1週から17周までの内容をまとめてください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	複合電子回路設計製作実習	必須	3、4期	6	6
教科の区分	専攻実技					
教科の科目	複合電子回路設計応用実習					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

電子回路を組込む電子機器製造業界

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
アナログ回路、デジタル回路、高周波回路、通信回路技術を基に、移動体通信に関する回路コンポーネントの設計手法と利用方法を習得する。	①	回路コンポーネントについて知っている。
	②	測定器の使用法ができる。
	③	回路シミュレーションができる。
	④	電磁界シミュレーションができる。
	⑤	回路コンポーネントの利用と評価ができる。
	⑥	システムの動作実験と評価ができる。
	⑦	
	⑧	
	⑨	
	⑩	

授業科目受講に向けた助言

予備知識・技能技術	電子回路、複合電子回路に関する知識が必要です。
授業科目についての助言	シミュレーション、回路コンポーネントの利用と評価、システムの動作実験と評価を通して、電磁界の影響を考慮した高周波回路設計法を習得できます。
教科書および参考書	教科書:自作テキスト 参考書:高速ディジタル回路実装ノウハウ
授業科目の発展性	複合電子回路技術 ― 複合電子回路設計製作実習 ― EMC応用実習

評価の割合

評価方法 / 指標・評価割合		試験	小テスト	レポート	制作物	成果発表	その他	合計
				50	50	0	0	100
評価割合	授業内容の理解度			20				
	技能・技術の習得度				20			
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力			10	10			
	取り組む姿勢・意欲			10	10			
	主体性・協調性			10	10			

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1. ガイダンス (1)シラバスの提示と説明 (2)回路コンポーネントについて	講義、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認して下さい。
2週	2. 回路コンポーネントの動作測定	実習、質疑	回路コンポーネントの動作測定について復習してください。
3週		実習、質疑	回路コンポーネントの動作測定について復習してください。
4週	3. 電磁界シミュレーション	実習、質疑	電磁界シミュレーションについて復習してください。
5週	4. 測定器(パワーメータ)の使用法	実習、質疑	測定器(パワーメータ)の使用法について復習してください。
6週	5. 信号発生器を用いた各種変調波の発生	実習、質疑	各種変調波の発生について復習してください。
7週	6. 測定器(スペクトラムアナライザ)の使用法	実習、質疑	測定器(スペクトラムアナライザ)の使用法について復習してください。
8週	7. スペクトラムアナライザを用いた各種信号、スプリアス等の測定	実習、質疑	スペクトラムアナライザを用いた測定法について復習し、各種変調波の測定を行えるようにしてください。
9週	8. 測定器(ネットワークアナライザ)の活用法	実習、質疑	測定器(ネットワークアナライザ)の活用法について復習してください。
10週	9. ネットワークアナライザを用いたSパラメータ測定	実習、質疑	Sパラメータ、およびSパラメータの測定について復習してください。
11週	10. 回路シミュレーション	実習、質疑	回路シミュレーションについて復習してください。
12週		実習、質疑	回路シミュレーションについて復習してください。
13週		実習、質疑	回路シミュレーションについて復習してください。
14週	11. 回路コンポーネントの利用と評価	実習、質疑	回路コンポーネントの利用と評価について復習してください。
15週		実習、質疑	回路コンポーネントの利用と評価について復習してください。
16週		実習、質疑	回路コンポーネントの利用と評価について復習してください。
17週	12. システム動作実験・評価	実習、質疑	動作実験・評価と問題点への対策方法について復習してください。
18週	13. 報告書作成	実習、質疑	論理的な報告書の作成方法を復習し、これまでの実験結果をまとめてください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	電子装置設計製作実習	必須	3、4期	4	4
教科の区分	専攻実技					
教科の科目	複合電子回路設計応用実習					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

電子回路設計技術
電子制御装置設計製作技術

授業科目の訓練目標

授業科目の目標	No	
電子装置の設計・製作・評価を行い、ものづくりに関する基本的な手順を理解し、製品化技術を習得します。	①	設計手法　設計コンセプト　設計仕様に基づく電子回路設計ができる。
	②	配線設計　筐体設計ができる。
	③	電子回路製作ができる。
	④	電源回路、表示回路ができる。
	⑤	回路実装ができる。
	⑥	総合調整、動作試験ができる。
	⑦	筐体加工・組立ができる。
	⑧	部品取付け、配線ができる。
	⑨	製品の評価ができる。
	⑩	設計仕様との比較と完成度　問題点とその対策ができる。

授業科目受講に向けた助言

予備知識・技能技術	電気・電子回路および計測制御について理解しておく必要があります。また、プリント基板の設計、製作や筐体加工技術が必要です。
授業科目についての助言	電源回路の設計から評価に至る一連の過程を通して、電子装置の設計・製作技術や製品化技術が習得できます。
教科書および参考書	教科書:自作テキスト
授業科目の発展性	アナログ電子回路設計 実装設計製作実習 機械工作・組立実習 → 電子装置設計製作実習 → 開発課題

評価の割合

評価方法 ＼ 指標・評価割合		試験	小テスト	レポート	制作物	成果発表	その他	合計
				50	30	10	10	100
評価割合	授業内容の理解度			20				
	技能・技術の習得度			20	15			
	コミュニケーション能力							
	プレゼンテーション能力					10		
	論理的な思考力、推論能力			10	15			
	取り組む姿勢・意欲						5	
	主体性・協調性						5	

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス (1)シラバスの提示と説明 2．設計手法 (1)設計コンセプト (2)設計仕様に基づく電子回路設計	講義、実習、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認して下さい。
2週	(3)配線設計　筐体設計	実習、質疑	CADを用いたプリント基板設計方法並びに、基板や機器の保護防止としての筐体設計に関して復習してください。
3週	3．電子回路製作 (1)電源回路の製作	実習、質疑	設計仕様書に基づいた電源回路の製作についてよく復習してください。
4週		実習、質疑	設計仕様書に基づいた電源回路の製作についてよく復習してください。
5週	(2)表示回路の製作	実習、質疑	設計仕様書に基づいた表示回路の製作についてよく復習してください。
6週		実習、質疑	設計仕様書に基づいた表示回路の製作についてよく復習してください。
7週	(3)回路実装	実習、質疑	設計仕様書に基づいた回路実装について復習してください。
8週		実習、質疑	設計仕様書に基づいた回路実装について復習してください。
9週	(4)総合調整、動作試験	実習、質疑	設計仕様書に基づいて、総合調整、動作試験の方法を復習してください。
10週		実習、質疑	設計仕様書に基づいた総合調整、動作試験の方法を復習してください。
11週	4．筐体加工・組立 (1)筐体加工	実習、質疑	筐体加工に関して復習してください
12週		実習、質疑	筐体加工に関して復習してください
13週		実習、質疑	筐体加工に関して復習してください
14週	(2)部品取付け、配線	実習、質疑	部品取り付け、配線方法に関して復習してください。
15週		実習、質疑	部品取り付け、配線方法について復習してください。
16週	(3)部品取付け、配線	実習、質疑	部品取り付け、配線方法について復習してください。
17週	5．製品の評価 (1)設計仕様との比較と完成度 (2)問題点とその対策	実習、質疑	製品の評価と問題点への対策方法について復習してください。
18週	6．報告書作成	実習、質疑	電子装置の設計・製作・評価までの内容及び、問題点とその対策などを論理的に取りまとめて報告書を作成してください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	電子制御技術応用実習	必須	3、4期	4	4
教科の区分	専攻実技					
教科の科目	複合電子回路設計応用実習					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

生産現場での安全作業
制御システムの構築

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
モータ駆動回路の設計・製作およびモータ制御プログラミングを学習し、モータの速度制御および位置決め制御方法を習得する。	①	設計手法ができる。
	②	モータ制御ハードウェア構築ができる。
	③	モータ制御ソフトウェア作成ができる。
	④	モータ制御システム構築ができる。
	⑤	モータ制御回路の製作ができる。
	⑥	
	⑦	
	⑧	
	⑨	
	⑩	

授業科目受講に向けた助言

予備知識・技能技術	専門課程における「マイクロコンピュータ工学」「マイクロコンピュータ工学実習」で学習した各項目を理解しておいてください。
授業科目についての助言	モータ制御回路には、トランジスタ等の半導体を使用するので電子回路を復習しておいてください。
教科書および参考書	教科書：自作テキスト
授業科目の発展性	電子制御技術応用実習 ―― 開発課題

評価の割合

評価方法 ＼ 指標・評価割合		試験	小テスト	レポート	制作物	成果発表	その他	合計
				50	50			100
評価割合	授業内容の理解度				10			
	技能・技術の習得度			30	40			
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力			10				
	取り組む姿勢・意欲			10				
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス (1)シラバスの提示と説明 2．電子制御技術応用実習の概要	講義、実習、質疑	シラバスをよく読み、この科目の目標と授業の流れを確認してください。
2週	3．設計手法 (1)設計仕様に基づく設計	実習、質疑	モータ制御ロジック回路の内容を考察してください。
3週	(2)設計仕様に基づく評価	実習、質疑	モータ制御ロジック回路を復習し、用途に応じたモータの選定を考察してください。
4週	4．モータ制御ハードウェア (1)インターフェース回路の設計・製作	実習、質疑	モータ駆動制御回路の設計・動作実験方法を考察してください。
5週	(2)電力変換回路の設計・製作	実習、質疑	パワートランジスタによる電力制御方法を調べて下さい。
6週	(3)モータ駆動回路の設計・製作	実習、質疑	トランジスタを用いたモータ駆動回路の設計を考察してください。
7週	5．モータ制御ソフトウェア (1)I/O制御プログラミング	実習、質疑	マイコンにおけるパラレルI/Oの初期化及び入出力命令を考察してください。
8週	(2)モータ制御プログラミング	実習、質疑	マイコンにおけるパラレルI/Oの初期化及び入出力命令を復習してください。
9週	(3)速度制御プログラミング	実習、質疑	マイコンのI/Oを用いた速度制御法を調べて下さい。
10週	(4)位置決め制御プログラミング	実習、質疑	マイコンにセンサを接続し、入出力命令を用いた位置決めのセンシング方法を考察してください。
11週	6．モータ駆動回路製作	実習、質疑	ディスクリート部品を用いた製作工程を考察してください。
12週		実習、質疑	トランジスタ、抵抗、コンデンサ、ダイオード、入出力コネクタの回路配置を考察してください。
13週		実習、質疑	モータ駆動回路を復習し、回路を製作します。
14週	7．制御プログラミング	実習、質疑	制御プログラムを考察してください。
15週		実習、質疑	制御プログラムを復習し、速度制御を追加するプログラムを検討してください。
16週	8．モータ制御システムの構築 (1)システムの統合 (2)システム動作テスト (3)問題点とその対策	実習、質疑	製作した回路の動作試験、トラブル対策を検討してください。
17週	9．製作課題の評価	実習、質疑	制御プログラムに対する動作状態を確認し、評価してください。
18週	10．報告書作成	実習、質疑	報告書を作成してください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	通信プロトコル実装実習	必須	3、4期	4	4
教科の区分	専攻実技					
教科の科目	セキュア通信システム構築応用実習					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

通信事業、組込み機器に関係する業界全般
ネットワークプログラミング技術

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
マイコンを用いた遠隔監視・遠隔制御を可能とするプログラムスタック実装技術を習得します。	①	TCP/IPの構造について説明できる。
	②	ソケットを用いたネットワークプログラムが作成できる。
	③	負荷装置の操作について説明できる。
	④	イーサネットコントローラの原理について説明できる。
	⑤	ARPの実装ができる。
	⑥	PINGの実装ができる。
	⑦	TFTPの実装ができる。
	⑧	echoプログラムの実装ができる。
	⑨	簡易Webサーバの実装ができる。
	⑩	簡易な遠隔監視・制御システムが作成できる。

授業科目受講に向けた助言

予備知識・技能技術	TCP/IPの基礎を知っていること、C言語でプログラミングできることを前提とします。
授業科目についての助言	本授業は専門課程で学んだ授業科目である「ネットワーク技術」「情報通信工学実習」が関連します。この2つの科目内容について復習しておいてください。
教科書および参考書	教科書：自作テキスト
授業科目の発展性	通信プロトコル実装設計 → 通信プロトコル実装実習 セキュアシステム設計 → セキュアシステム構築実習 通信プロトコル実装実習、セキュアシステム構築実習 → 組込みシステム構築課題実習

評価の割合

指標・評価割合 ＼ 評価方法		試験	小テスト	レポート	制作物	成果発表	その他	合計
				40	60			100
評価割合	授業内容の理解度				20			
	技能・技術の習得度			20	20			
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力			20				
	取り組む姿勢・意欲				10			
	主体性・協調性				10			

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス （1）シラバスの提示と説明 2．TCP/IPの構造	実習	シラバスをよく読み、この科目の目標と授業の流れを確認してください。 専門課程での「ネットワーク技術」を復習してください。
2週	3．ソケット（UDP）を用いたechoプログラムの作成 （1）クライアントとサーバでの文字列の送受信	実習	TCP/IPの構造、C言語について復習してください。
3週		実習	TCP/IPの構造、C言語について復習してください。
4週	4．イーサネットコントローラ	実習	専門課程での「ネットワーク技術」を復習してください。
5週	5．LEDとスイッチの操作プログラム作成	実習	C言語について復習してください。
6週	6．ブザーと液晶ディスプレイの操作プログラム作成	実習	C言語について復習してください。
7週	7．ARPの実装 （1）イーサネットヘッダ、ARPリクエスト、ARPリプライの作成	実習	ARPについて復習してください。
8週	8．PINGの実装 （1）イーサネットヘッダ、IPヘッダ、ICMPメッセージの作成	実習	PINGについて復習してください。
9週	9．ルータを用いたネットワークでのPINGの動作	実習	経路制御について復習してください。
10週	10．ネットワーク構築 （1）パケット分割	実習	パケット分割について復習してください。
11週	11．UDPを用いたechoプログラムの作成	実習	UDPについて復習してください。
12週	12．TFTPプログラムの作成 （1）イーサネットヘッダ、IPヘッダ、UDPヘッダ、TFTPメッセージの作成	実習	TFTPについて復習してください。
13週		実習	TFTPについて復習してください。
14週	13．TCPを用いたechoプログラムの作成	実習	C言語について復習してください。
15週	14．簡易Webサーバの作成 （1）イーサネットヘッダ、IPヘッダ、TCPヘッダ、HTTPメッセージの作成	実習	Webサーバについて復習してください。
16週		実習	Webサーバについて復習してください。
17週	15．LEDとスイッチを用いた遠隔制御	実習	LEDとスイッチの操作法について復習してください。
18週		実習	LEDとスイッチの操作法について復習してください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	セキュアシステム構築実習	必修	1、2期	4	4
教科の区分	専攻実技					
教科の科目	セキュア通信システム構築応用実習					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

電気通信事業や情報処理サービス事業におけるネットワークの構築・運用・設計に係わる技術
その他コンピュータシステムに係わる管理技術

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
ユビキタス社会におけるネットワークシステムの構築および運用管理を通して、セキュアなシステム構築技術を習得します。	①	サーバ用OSのインストールができる。
	②	インターネットサーバの構築ができる。
	③	不要サービスの無効化とアクセス制御およびユーザ管理ができる。
	④	ネットワーク機器の設定ができる。
	⑤	静的経路制御と動的経路制御の設定ができる。
	⑥	ルータとファイアウォールの設定ができる。
	⑦	クライアントPCのポートの制限とファイアウォールの設定ができる。
	⑧	暗号化の設定ができる。
	⑨	無線通信の設定ができる。
	⑩	各種ログファイルの監視と検証、侵入・検知について対応できる。

授業科目受講に向けた助言

予備知識・技能技術	専門課程の「情報通信工学実習」で学習したTCP/IPについて復習してください。
授業科目についての助言	専門課程の「情報通信工学実習」で学習したTCP/IPを基に、UNIXのシステム管理、セキュアなインターネットサーバの構築やネットワーク機器の設定、運用管理技術を学びます。
教科書および参考書	教科書:自作プリント 参考書:マスタリングTCP/IP 入門編(オーム社)
授業科目の発展性	通信プロトコル実装設計 → 通信プロトコル実装実習 セキュアシステム設計 → セキュアシステム構築実習 → 組込みシステム構築課題実習

評価の割合

指標・評価割合 ＼ 評価方法		試験	小テスト	レポート	制作物	成果発表	その他	合計
		30		20	30		20	100
評価割合	授業内容の理解度	10		10	15			
	技能・技術の習得度	10		10	15			
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力	10						
	取り組む姿勢・意欲						20	
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1. ガイダンス (1)シラバスの提示と説明 2. ネットワークOSのインストール	実習	シラバスをよく読み、この科目の目標と授業の流れを確認してください。 専門課程の「情報通信工学実習」について復習しておいてください。
2週	3. UNIXシステム管理	実習	専門課程の「情報通信工学実習」について復習しておいてください。
3週	4. ネットワークの設定	実習	「セキュアシステム設計」のネットワークの基礎と合わせ復習してください。
4週	5. DNSサーバの構築	実習	DNSについて復習してください。
5週	6. メールサーバの構築	実習	メールサーバについて復習してください。
6週	7. Webサーバの構築	実習	Webサーバについて復習してください。
7週	8. PROXYサーバの構築	実習	PROXYサーバについて復習してください。
8週	9. POP/IMAPサーバの構築、Windowsのインストール	実習	3週から8週目までの内容を総合的に復習してください。
9週	10. 総合実習(1)	実習	3週から8週目までの内容を総合的に復習してください。
10週	11. 経路制御	実習	TCP/IPと経路制御について復習してください。
11週	12. ルータ設定	実習	ルータの設定について復習してください。
12週	13. ファイアウォール構築(DMZ)	実習	ファイアウォールについて復習してください。
13週	14. セキュリティポリシーに従ったポート制限	実習	ファイアウォールとポート制御について復習してください。
14週	15. ウィルス対策ソフトのインストール、暗号化(SSL、SSH)	実習	ウィルス対策について復習してください。
15週	16. アクセスポイントの構築、暗号化	実習	暗号技術について復習してください。
16週	17. システムの運用 (1)侵入検知、ログの確認	実習	システム運用の重要性と内容について復習してください。
17週	18. 総合演習(2)	実習	10週から17週目までの内容について総合的に復習してください。
18週	19. 実技試験	試験	試験に備え、これまで実習した内容について再確認して臨んでください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	組込みシステム構築実習	必須	1、2期	6	6
教科の区分	専攻実技					
教科の科目	組込みシステム構築応用実習					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

組込み機器に関係する業界全般
OSを用いた組込み機器開発
組込み機器のプログラミング

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
組込みOSの活用及びネットワークに対応した組込みソフトウェア技術を習得します。	①	ターゲットボード概要の説明ができる。
	②	クロス開発環境の構築ができる。
	③	プロセスを利用したプログラミングができる。
	④	シグナルを利用したプログラミングができる。
	⑤	メッセージキューを利用したプログラミングができる。
	⑥	共有メモリを利用したプログラミングができる。
	⑦	セマフォを利用したプログラミングができる。
	⑧	デバイスドライバを作成できる。
	⑨	LCDプログラミングができる。
	⑩	ネットワークプログラミングができる。

授業科目受講に向けた助言	
予備知識・技能技術	専門課程の「組込みシステム工学」、「組込みソフトウェア応用実習」を理解し、C言語のプログラミングとLinuxの操作ができること。
授業科目についての助言	近年ほとんどの家電製品は組込みマイコンによって制御されています。多くの企業において組込みプログラマは不足している状態です。本実習では組込みのマルチタスクプログラムの方法を習得していきます。 本実習で学ぶ知識や技術は企業のみならず、標準課題や開発課題を受講する上でも必要です。将来、習得した知識を活用するためにも毎回の授業をしっかりと受講し、わからないことは積極的に質問して積み残さないようにしてください。
教科書および参考書	教科書:自作テキスト
授業科目の発展性	組込みシステム設計 → 組込みシステム構築実習 デジタルデバイス設計 → 組込みデバイス設計実習 組込みシステム構築実習 / 組込みデバイス設計実習 → 組込みシステム構築課題実習

評価の割合

指標・評価割合 ＼ 評価方法		試験	小テスト	レポート	制作物	成果発表	その他	合計
				30	50		20	100
評価割合	授業内容の理解度			30	20			
	技能・技術の習得度				30			
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力							
	取り組む姿勢・意欲						20	
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1. ガイダンス (1)シラバスの提示と説明 2. ターゲットボードの概要	実習、質疑	専門課程の「組込みソフトウェア応用実習」を復習しておいてください。ターゲットボードの共通点、異なる点を理解してください。
2週	3. クロス開発環境	実習、質疑	開発環境を構築します。休まないようにしてください。終わらなかった部分については、次回実習までに終わらせるようにしてください。
3週	4. 組込みOSシステムプログラミング (1)プロセス	実習、質疑	「組込みシステム設計」のプロセスについて復習しておいてください。
4週		実習、質疑	「組込みシステム設計」のプロセスについて復習しておいてください。
5週	(2)シグナル	実習、質疑	「組込みシステム設計」のシグナルについて復習しておいてください。
6週		実習、質疑	「組込みシステム設計」のシグナルについて復習しておいてください。
7週	(3)メッセージキュー	実習、質疑	「組込みシステム設計」のメッセージキューについて復習しておいてください。
8週		実習、質疑	「組込みシステム設計」のメッセージキューについて復習しておいてください。
9週	(4)共有メモリ	実習、質疑	「組込みシステム設計」の共有メモリについて復習しておいてください。
10週		実習、質疑	「組込みシステム設計」の共有メモリについて復習しておいてください。
11週	(5)ソケット	実習、質疑	「組込みシステム設計」のソケットについて復習しておいてください。
12週	(6)セマフォ	実習、質疑	「組込みシステム設計」のセマフォについて復習しておいてください。
13週		実習、質疑	「組込みシステム設計」のセマフォについて復習しておいてください。
14週	5. デバイスドライバの作成	実習、質疑	デバイスドライバの作り方に関して学びます。
15週		実習、質疑	デバイスドライバを用いたプログラミングに関して学びます。
16週	6. LCDプログラミング	実習、質疑	DirectFBを用いたLCDプログラミングに関して学びます。
17週		実習、質疑	LCDに画像を表示する方法などに関して学びます。
18週	7. マイコンネットワークプログラミング	実習、質疑	ソケットプログラムを復習しておいてください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	組込みデバイス設計実習	必須	5、6期	4	4
教科の区分	専攻実技					
教科の科目	組込みシステム構築応用実習					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

ハードウェア記述言語によるデジタル回路設計
組込み機器に関する業界全般

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
FPGAを用いたデジタル回路の一連の開発フローを学習し、HDLによるデジタル回路設計技法を習得します。	①	組み込みデバイス開発環境の構築ができる。
	②	組み込みデバイスの開発フローについて説明できる。
	③	組み込みデバイスのHDLによる回路設計ができる。
	④	組み込みデバイスの回路実装ができる。
	⑤	組み込みデバイスの評価と検証ができる。
	⑥	
	⑦	
	⑧	
	⑨	
	⑩	

授業科目受講に向けた助言	
予備知識・技能技術	ハードウェア記述言語を知っていることを前提としています。
授業科目についての助言	この実習では、デジタルデバイス設計で学んだHDLを用いて、デジタル回路を仕様から設計し製作します。この実習で学ぶ技術は、組込みシステム構築課題実習で利用する技術です。毎回の授業をしっかり受講し、遅刻・欠席をしないようにしてください。また、わからないことは積極的に質問して積み残さないようにしてください。
教科書および参考書	教科書：自作テキスト
授業科目の発展性	組込みシステム設計 → 組込みシステム構築実習 → 組込みシステム構築課題実習 デジタルデバイス設計 → 組込みデバイス設計実習 → 組込みシステム構築課題実習

評価の割合

指標・評価割合 ＼ 評価方法		試験	小テスト	レポート	制作物	成果発表	その他	合計
				50	50			100
評価割合	授業内容の理解度			20	20			
	技能・技術の習得度				20			
	コミュニケーション能力							
	プレゼンテーション能力							
	論理的な思考力、推論能力			20				
	取り組む姿勢・意欲			10	10			
	主体性・協調性							

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス （1）シラバスの提示と説明 2．開発環境 （1）統合開発環境 （2）シュミレータ 3．開発フロー （1）回路デザインと論理検証 （2）論理合成と配置配線 （3）タイミング検証	実習、質疑	開発環境、開発フローについて復習してください。
2週	4．HDLによる回路設計 （1）記述スタイル （2）組合せ回路	実習、質疑	「デジタルデバイス設計」で学んだ基本ゲート回路とHDLの文法を復習しておいてください。
3週		実習、質疑	「デジタルデバイス設計」で学んだ基本ゲート回路とHDLの文法を復習しておいてください。
4週		実習、質疑	「デジタルデバイス設計」で学んだフリップフロップ回路とHDLの文法を復習しておいてください。
5週	（3）順序回路・演算回路・同期回路	実習、質疑	「デジタルデバイス設計」で学んだシフトレジスタ回路とHDLの文法を復習しておいてください。
6週		実習、質疑	「デジタルデバイス設計」で学んだカウンタ回路とHDLの文法を復習しておいてください。
7週	5．回路実装 （1）マルチプレクサの製作	実習、質疑	「デジタルデバイス設計」で学んだマルチプレクサ回路とHDLの文法を復習しておいてください。
8週		実習、質疑	「デジタルデバイス設計」で学んだマルチプレクサ回路とHDLの文法を復習しておいてください。
9週	（2）デコーダの製作	実習、質疑	「デジタルデバイス設計」で学んだデコーダ回路とHDLの文法を復習しておいてください。
10週		実習、質疑	「デジタルデバイス設計」で学んだデコーダ回路とHDLの文法を復習しておいてください。
11週	（3）カウンタの製作	実習、質疑	「デジタルデバイス設計」で学んだカウンタ回路とHDLの文法を復習しておいてください。
12週		実習、質疑	「デジタルデバイス設計」で学んだカウンタ回路とHDLの文法を復習しておいてください。
13週	（4）シフトレジスタの製作	実習、質疑	「デジタルデバイス設計」で学んだシフトレジスタ回路とHDLの文法を復習しておいてください。
14週		実習、質疑	「デジタルデバイス設計」で学んだシフトレジスタ回路とHDLの文法を復習しておいてください。
15週	6．評価と検証 （1）IPモジュールの活用 （2）各種回路の接続試験	実習、質疑	デジタルデバイス設計で学んだIPモジュールと実装について復習しておいてください。
16週		実習、質疑	デジタルデバイス設計で学んだIPモジュールと実装について復習しておいてください。
17週		実習、質疑	デジタルデバイス設計で学んだIPモジュールと実装について復習しておいてください。
18週		実習、質疑	デジタルデバイス設計で学んだIPモジュールと実装について復習しておいてください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	電子通信機器設計製作課題実習（標準課題）	必須	5、6期	10	10
教科の区分	専攻実技					
教科の科目	通信機器設計製作応用実習（標準課題）					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

電子回路設計技術
電子制御装置設計製作技術
マイコン制御装置設計製作技術

授業科目の訓練目標

授業科目の目標	No	
無線通信機能を有した温度・湿度計測データロガー装置の設計・製作を通して、電子通信機器設計製作に必要な製品化技術を習得します。	①	基本設計、製作計画、ハードウェア（入出力）設計ができる。
	②	ソフトウェア設計、プレゼンテーションができる。
	③	回路試作と動作実験（センサ周辺回路、表示回路、A/D変換回路）、動作確認ができる。
	④	ソフトウェア 設計制作テスト、制御プログラムモジュールの制作、各プログラムのテストができる。
	⑤	回路設計製作 、プリント基板の設計製作ができる。
	⑥	筐体設計製作ができる。
	⑦	総合組立・試験調整ができる。
	⑧	性能試験と検査表の作成ができる。
	⑨	製品マニュアルの作成ができる。
	⑩	報告・発表ができる。

授業科目受講に向けた助言

予備知識・技能技術	マイクロコンピュータについてのハードウェア及びソフトウェアの知識、無線通信についての基本的な知識が必要です。さらに、CAD/CAMによる回路図作成、アートワーク、基板製作、筐体加工技術も必要です。
授業科目についての助言	この授業は理論をもとに技能・技術を学ぶ実学融合教育訓練であり、課題をグループで取り組むグループワーキング方式で実施します。この授業では理論・技能・技術を習得するだけでなく、コミュニケーション能力の向上も図ります。
教科書および参考書	教科書：自作テキスト
授業科目の発展性	電子通信機器設計製作課題実習（標準課題） → 開発課題

評価方法 ／ 指標・評価割合		試験	小テスト	レポート	制作物	成果発表	その他	合計
				25	40	10	25	100
評価割合	授業内容の理解度			10	20			
	技能・技術の習得度			15	20			
	コミュニケーション能力						5	
	プレゼンテーション能力					10		
	論理的な思考力、推論能力						5	
	取り組む姿勢・意欲						10	
	主体性・協調性						5	

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス （1）シラバスの提示と説明 2．基本設計 （1）製作計画 （2）ハードウェア（入出力）設計	講義、実習、質疑	基本設計、製作計画を立て、ハードウェア（入出力）設計を復習してください。
2週	（3）ソフトウェア設計 （4）プレゼンテーション	実習、質疑	ソフトウェア設計を復習してください。
3週	3．回路試作と動作実験 （1）試作と実験 ①センサ周辺回路 ②表示回路 ③A/D変換回路 ④動作確認	実習、質疑	センサ周辺回路、表示回路、A/D変換回路を復習してください。
4週		実習、質疑	センサ周辺回路、表示回路、A/D変換回路を復習してください。
5週	4．ソフトウェア 設計制作テスト （1）制御プログラムモジュールの制作 （2）各プログラムのテスト	実習、質疑	制御プログラムモジュールを制作し、モジュールをテストしてください。
6週		実習、質疑	制御プログラムモジュールを制作し、モジュールをテストしてください。
7週		実習、質疑	制御プログラムモジュールを制作し、モジュールをテストしてください。
8週	5．回路設計製作 （1）プリント基板の設計製作	実習、質疑	プリント基板を設計・製作をしてください。
9週		実習、質疑	プリント基板を設計・製作をしてください。
10週		実習、質疑	プリント基板を設計・製作をしてください。
11週	6．筐体設計製作 （1）筐体設計製作	実習、質疑	筐体を設計・製作をしてください。
12週	7．総合組立・試験調整 （1）総合組立調整	実習、質疑	装置を組み立て、試験・調整してください。
13週	8．性能試験 （1）性能試験と検査表の作成	実習、質疑	性能試験と検査表を作成し、検証および考察してください。
14週		実習、質疑	性能試験と検査表を作成し、検証および考察してください。
15週	9．マニュアル作成 （1）製品マニュアルの作成 （2）製品仕様書の作成	実習、質疑	製品マニュアルと製品仕様書を作成してください。
16週		実習、質疑	製品マニュアルと製品仕様書を作成してください。
17週	10．報告・発表 （1）報告書の作成 （2）発表用資料の作成	実習、質疑	報告書と発表用資料を作成してください。
18週	（3）報告・発表会の実施	実習、質疑	報告書と発表用資料を作成してください。

訓練支援計画書

科名：生産電子情報システム技術科

訓練科目の区分		授業科目名	必須・選択	開講時期	単位	時間／週
教育訓練課程	応用課程	組込みシステム構築課題実習 （標準課題実習）	必須	3、4期	10	10
教科の区分	専攻実技					
教科の科目	無線通信機器設計製作応用実習					

担当教員	内線電話番号	電子メールアドレス	教室・実習場

授業科目に対応する業界・仕事・技術

OSを用いた組込み機器開発
組込み機器のプログラミング
通信事業、組込み機器に関係する業界全般

授業科目の訓練目標

授業科目の目標	No	授業科目のポイント
データ収集機能やセキュアなネットワーク機能を実装した組込みシステムの構築を通して、組込みソフトウェア開発、センサ制御、負荷装置制御等の統合的な技術を習得します。	①	システム要件と製作計画が作成できる。
	②	システムの概要設計ができる。
	③	システムの詳細設計ができる。
	④	マイコンの周辺機器設計ができる。
	⑤	制御用端末の構築ができる。
	⑥	ソフトウェアの制作ができる。
	⑦	性能試験ができる。
	⑧	マニュアルの作成ができる。
	⑨	報告書の作成ができる。
	⑩	発表ができる。

授業科目受講に向けた助言	
予備知識・技能技術	「組込みシステム設計」「組込みシステム構築実習」「通信プロトコル実装設計」「通信プロトコル実装実習」の内容について理解しておいてください。
授業科目についての助言	本実習では組込み技術、ネットワーク技術、ハードウェア技術を複合した課題として遠隔監視システムをグループで作成します。システムの設計・製作において、リーダを決め、スケジュールを立て、進捗を管理し、製品の設計・製造のプロセスを体験します。グループ学習では、技術的な知識だけでなく、対人関係能力や物事を概念的に捉える能力が重要です。
教科書および参考書	教科書：自作テキスト
授業科目の発展性	組込みシステム設計 → 組込みシステム構築実習 通信プロトコル実装設計 → 通信プロトコル実装実習 → 組込みシステム構築課題実習

評価の割合

評価方法 ＼ 指標・評価割合	試験	小テスト	レポート	制作物	成果発表	その他	合計
				40	20	40	100
評価割合 授業内容の理解度				20			
評価割合 技能・技術の習得度				20			
評価割合 コミュニケーション能力						10	
評価割合 プレゼンテーション能力					20		
評価割合 論理的な思考力、推論能力						10	
評価割合 取り組む姿勢・意欲						10	
評価割合 主体性・協調性						10	

回数	訓練の内容	運営方法	訓練課題　予習・復習
1週	1．ガイダンス （1）シラバスの提示と説明 2．製作計画の作成 （1）システム計画書の提示 （2）製作計画の作成	実習、質疑	「組込みシステム設計」で学んだ組込みシステム開発フローを復習してください。
2週	3．システム要求仕様書の作成 （1）要求分析 （2）システム要求仕様書の作成	実習、質疑	「システム計画書」を理解してくださいさい。
3週	（1）要求分析 （2）システム要求仕様書の作成	実習、質疑	要求モデリングを復習し、遠隔監視システムの機能を理解してください。
4週	4．システム概要設計書の作成 （1）概要設計 （2）システム概要設計書の作成 （3）システムを構成する部品の機能試験	実習、質疑	分析モデリングを復習し、雲台・ネットワーク・画像表示の機能を理解してください。
5週		実習、質疑	分析モデリングを復習し、雲台・ネットワーク・画像表示の機能を理解してください。
6週		実習、質疑	分析モデリングを復習し、雲台・ネットワーク・画像表示の機能を理解してください。
7週	5．詳細設計書及びハードウェア設計書の作成 （1）詳細設計 （2）周辺機器の設計 （3）ハードウェア設計書の作成	実習、質疑	設計モデリングを復習し、雲台・ネットワーク・画像表示の実装方法を理解してください。
8週		実習、質疑	設計モデリングを復習し、雲台・ネットワーク・画像表示の実装方法を理解してください。
9週		実習、質疑	設計モデリングを復習し、雲台・ネットワーク・画像表示の実装方法を理解してください。
10週	6．プログラミング及びハードウェアの実装 （1）プログラムの作成 （2）ハードウェアの実装	実習、質疑	プログラミング手法、ハードウェアの実装方法を理解してください。
11週		実習、質疑	プログラミング手法、ハードウェアの実装方法を理解してください。
12週		実習、質疑	プログラミング手法、ハードウェアの実装方法を理解してください。
13週		実習、質疑	プログラミング手法、ハードウェアの実装方法を理解してください。
14週	7．システム全体の組立、調整、機能試験、性能試験	実習、質疑	機能試験、性能試験の方法について復習してください。
15週		実習、質疑	機能試験、性能試験の方法について復習してください。
16週	8．発表会準備、報告書作成	実習、質疑	論理的な報告書の作成方法を復習してください。
17週		実習、質疑	論理的な報告書の作成方法を復習してください。
18週	9．発表会	実習、質疑	発表会のリサールをしてください。

資料 3

組込みシステム構築課題実習

自作テキスト等

組込みソフトウェア設計

職業能力開発総合大学校東京校
生産電子情報システム技術科
2011年10月18日

1

ソフトウェア設計の必要性

コンピュータの性能向上と普及

↓

システム開発の需要が拡大

↓

ソフトウェアの規模、需要が増大

↓

ソフトウェアのコスト増大、技術者不足

↓

ソフトウェアの生産性の低下、品質の低下

ソフトウェア設計法を考案

解決

解決

個人の経験に依存した作業

2

ソフトウェア設計の必要性

要求者の仕様が曖昧、設計者の理解も曖昧

↓

ソフトウェア設計手法を用いる

↓

要求仕様が明確になる
機能、構成が明確になる

↓

品質が向上する
保守が容易になる（生産性の向上）
技術者を教育できる

→

作業の手戻りが少なくなる
共同での作業効率が向上

↓

開発期間の短縮（コストの削減）

3

ソフトウェア設計のプロセスモデル

規模が大きく複雑であり、要求されている機能が
曖昧であるものを一度に解決することができない。

↓

問題を分割し段階的に作業する。

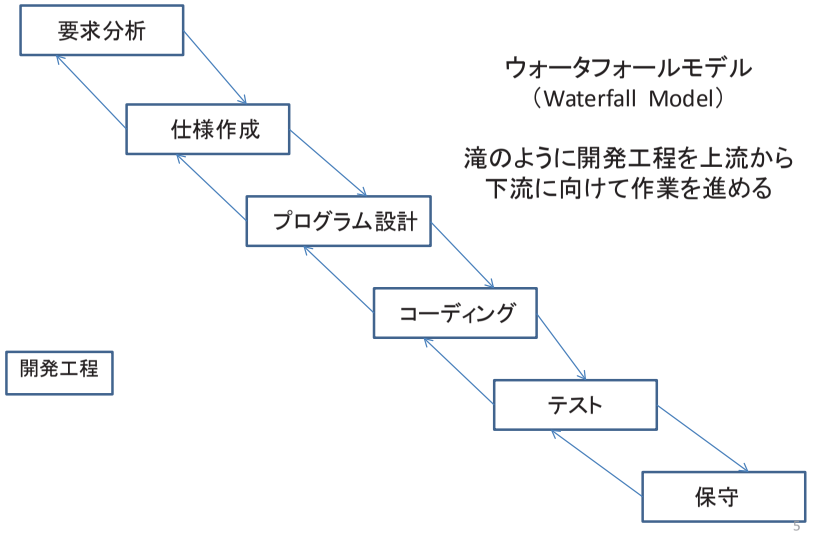
具体的な方法

・ウォータフォールモデル

・スパイラルモデル

4

ウォータフォールモデル



ウォータフォールモデル
（Waterfall Model）

滝のように開発工程を上流から下流に向けて作業を進める

5

ソフトウェア設計手法

構造化分析手法　　Yourdon、Constantine　1979年
DeMarco　1979年
が提案、その後多くの改良がなされた。

オブジェクト分析手法　　1980年代後半から現在まで多くの手法が考案された。現在では多くの手法で記法として統一モデリング言語（UML）を採用する。

6

構造化分析手法の特徴

上流工程から下流工程に向けて、トップダウンで設計する。

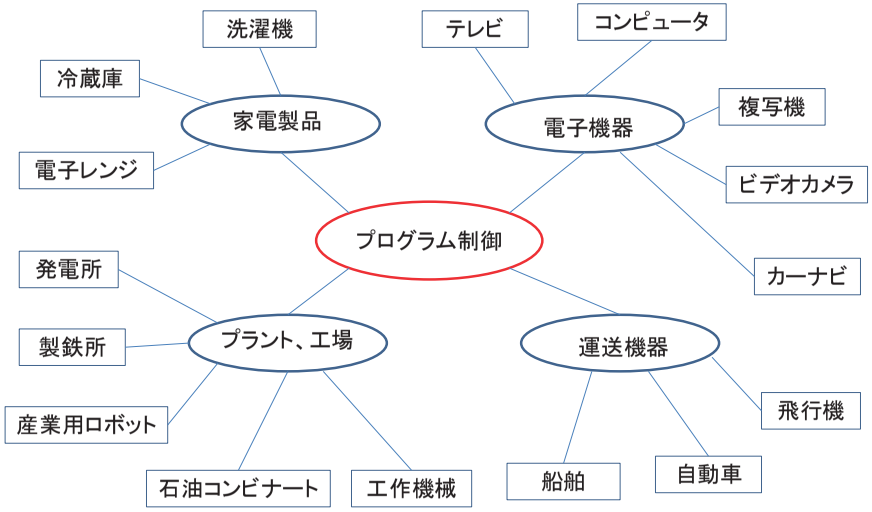
各工程で成果物（ドキュメント、プログラムリストなど）が得られ、検証できる。

工程のなかで設計ミスなど誤りを見つけた場合、前工程に戻れる。

顧客の要求が変わった場合、最初から設計し直す必要がある。

7

組込みシステム



8

組込みソフトウェアの特徴

ハードウェアを直接制御する

メモリなどに制約がある

リアルタイム性がある

OSを使用しない場合もある

高い信頼性、品質が要求される

出荷したらアップデートし難い

9

組込みシステム開発でのソフトウェア設計の必要性

ハードウェアの進歩、コンピュータの性能向上と普及

↓

組込みシステムのニーズが増大

↓

組込みソフトウェアの規模、需要が増大

↓

組込みソフトウェアのコスト増大、技術者不足

↓

組込みソフトウェアの生産性の低下、品質の低下

組込みソフトウェア開発での
ソフトウェア設計法の考案

解決

解決

↑

個人の経験に依存した作業

↓

10

組込みソフトウェア設計

組込みシステムの特徴

1. 自然法則を扱う
2. リアルタイム要求が厳しい
3. 制約事項が厳しい
4. 高い品質と信頼性

付加 ←

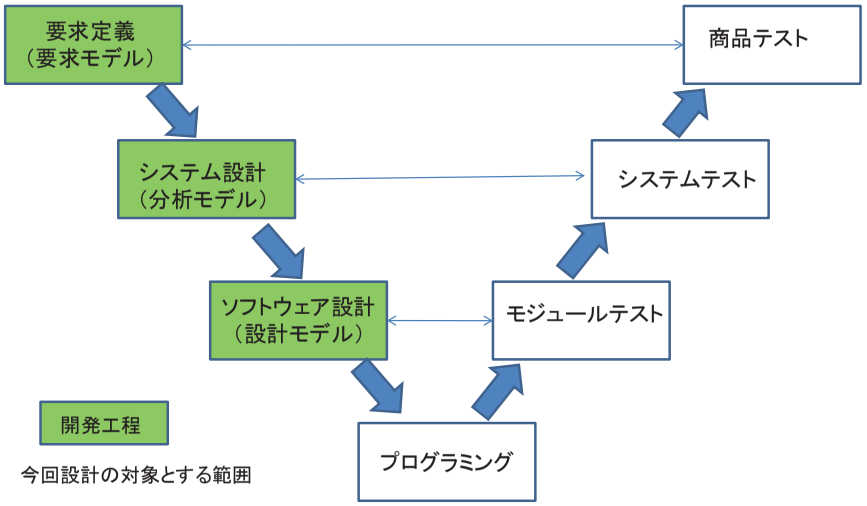
ソフトウェア開発のための
構造化分析・設計手法

↓

組込みシステム開発のための
構造化モデリング
（以下、構造化モデリングという）

11

構造化モデリング



今回設計の対象とする範囲

12

要求モデリング

職業能力開発総合大学校東京校
生産電子情報システム技術科
2011年10月25日

1

システム設計の流れ

要求モデリング　何がしてほしいか

↓

分析モデリング　何ができるか

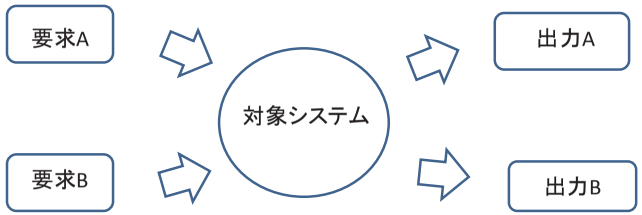
↓

設計モデリング　どう実現するか

2

要求モデリングとは

要求モデリングでは「システムに何をしてほしいか」、その結果「何が起こるか」を考え、システムに必要な機能要件を整理し、仕様書にまとめる。



3

問1　要求モデリングなしで設計するとどのような問題が起こるか。

解答例

・曖昧な仕様書では、書き手と読み手の理解が異なり、顧客の意図と異なる製品ができてしまう。

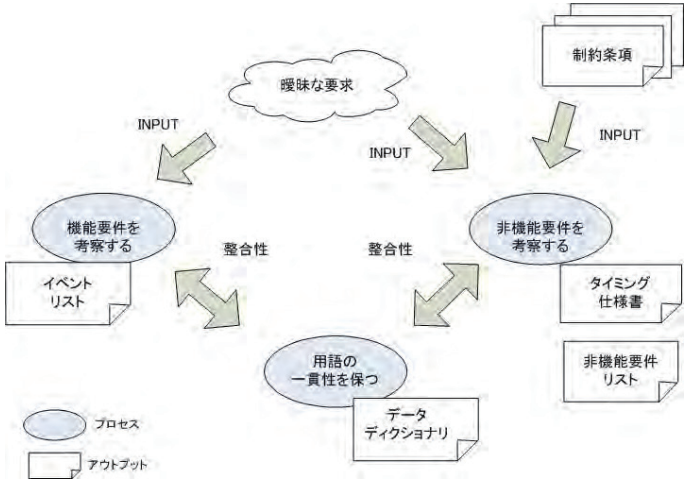
・仕様書の抜け漏れや曖昧さにより、手戻りが発生し納期が遅れる。

・品質が下がり、出荷後不具合が発生する可能性がある。

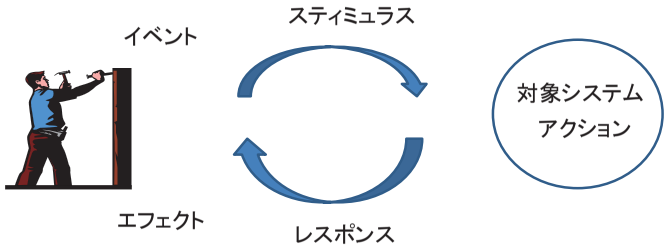
・設計者に「システムで何をしたいか」を正確に効率よく伝えられない。

4

要求モデリング



イベントリスト



イベント（事象）： 対象システムの外部で生じ、その発生をシステムが制御できない事象

スティミュラス（刺激）： イベントが発生したことを伝えるデータ

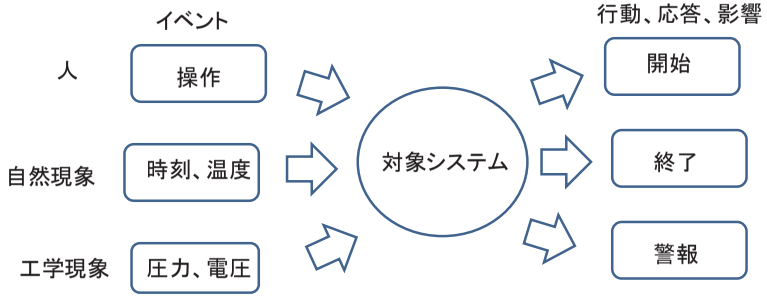
アクション（行動）： イベントが発生したときのシステムの果たすべき機能

レスポンス（応答）： イベントが発生したときのシステムの外部への反応

エフェクト（影響）： システムの反応による外部への影響

イベントリストの作成

システムに対する曖昧な要求から、システムの外部で発生するイベント（事象）に対してシステムの行動と応答、システム外への影響をイベントリストにまとめる。



話題沸騰ポット

テキスト：組込みソフトウェア開発のための構造化モデリング

P33～P38

話題沸騰ポットの機能

実現する機能	機能名称
ポット内の水を沸騰する	沸騰機能
ポット内の水を保温する	保温機能
ポット内の水を給湯する	給湯機能
利用者が指定した時間がきたらブザー鳴動	キッチンタイマー機能

イベントリスト（精査前）

イベント	スティミュラス	アクション	レスポンス	エフェクト
沸騰要求	（なし）	水を沸騰する 水を一定温度で保温する	ヒータの加熱操作 沸騰状態及び 保温状態	カルキの抜けたお湯が得られる
給湯要求	（なし）	ポット内のお湯を給湯する		カップにお湯が注がれる
給湯口のロック	（なし）	給湯口をロックする	ブザーオン ロックランプ点灯	お湯を出せなくできる
給湯口のロック解除	（なし）	給湯口のロックを解除する	ブザーオン ロックランプ点灯	お湯を出せるようになる
水が溢れる	満水	沸騰をやめる	ブザーオン エラー通知	危険な状態を回避できる
水が無くなる	水なし	沸騰をやめる	ブザーオン エラー通知	危険な状態を回避できる
水温が上がらない	（なし）	沸騰をやめる		ポットの異常がわかる
水温が下がらない	（なし）	保温をやめる		ポットの異常がわかる
…	…	…	…	…

9

イベントリスト（精査後）

記述の抽象度、各項目の粒度を揃え、対象性を考慮して理解し易くするため表を再考する。

利用者からのイベント

イベント	スティミュラス	アクション	レスポンス	エフェクト
沸騰要求	沸騰開始	沸騰行為をする カルキ抜きをする 目標の温度で保温行為をする	ヒータオン／オフ 動作状態	安全なお湯が得られる 適量のお湯が得られる ポットの動作状態がわかる
保温温度の設定	保温モード設定	目標の保温温度を設定する	保温モード	お湯を欲しい温度にできる
給湯の準備	給湯準備	給湯口のロックを解除する	ロック状態	ポットから給湯が可能となる
給湯要求	給湯開始	ポットのお湯を給湯する	ポンプの動作	お湯が注がれる
給湯の完了	給湯停止	お湯の給湯を停止する	ポンプの停止	適量のお湯が得られる
誤給湯の防止	給湯禁止	給湯口をロックする	ロック状態	誤ってお湯をこぼさなくなる

従属的な事象は省略

10

イベントリスト（精査後）

記述の抽象度、各項目の粒度を揃え、対象性を考慮して理解し易くするため表を再考する。

センサーからのイベント（外部からと解釈）

イベント	スティミュラス	アクション	レスポンス	エフェクト
水位変化	水位	水量の表示を更新する	水量	お湯の残量がわかる
水位異常	満水	沸騰行為または保温行為を中止する	水量 警報情報 ヒータオフ	危険な状態を回避できる 装置の異常を知る
水温変化	水温	水温の表示を更新する 沸騰行為または保温行為をする	水温 ヒータオン／オフ	お湯の温度がわかる 指定した温度のお湯を得る
水温異常	水温	沸騰行為または保温行為を中止する	水量、 警報情報 ヒータオフ	危険な状態を回避できる 装置の異常を知る

11

問2　イベントリストの作成において、水位変化、温度変化などポット内部で発生しているセンサーからのイベントを外部からと解釈する理由について考えなさい。

解答例

水位変化、温度変化などポットの中の水から発生している。
しかし、ポット内の水は、ポット自体の構成要素ではない。

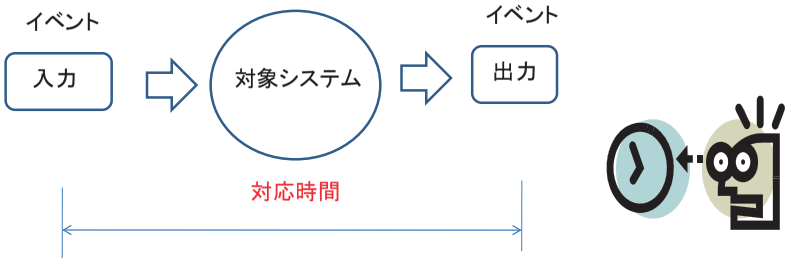
人体の消化器官を通る食物、液体は人体を構成している骨、筋肉、血液、体液とは異なり、人体の構成要素ではない。人体の構成要素ではないので、食物、液体は人体の外部のものと考えられる。

すなわち、消化器官の外と同様に、ポット内の水も外部にあるものと考えられる。

12

タイミング仕様書の作成

システムに対する曖昧な要求と制約条項から、システムのリアルタイム性に要求される時間的制約を整理し、タイミング仕様書を作成する。



13

タイミング仕様書

入力	イベント	出力	イベント	対応時間
水温	100度を超える	ブザー音	高温エラー	最大1秒
水温	前回検出した水温より低い	ブザー音	ヒータ動作異常	1分

タイミング仕様書は、システムの内部からでなく外部から見て、入力のイベントに対し、出力のイベントが発生するまでの対応時間を記述する。

14

非機能要件リストの作成

タイミング仕様書に盛り込まない安全性、信頼性、拡張性、保守性、ハードウェア制約などを非機能要件リストをしてまとめる。

ハードウェア制約

安全性

信頼性

15

非機能要件リスト

No	分類	内容	優先度	関係部署
1	環境	消費電力XXワット以下	高	商品企画
2	安全性	ポットのふたが開いてから100ms以内にヒータの加熱を停止すること	高	商品企画
3	保守性	途中で担当者が入れ替わっても作業に支障をきたさない	中	開発部署
4	信頼性	1年間連続動作を保証する	中	品質管理
5	拡張性	同系列商品にて、後継機のスケジュールが1/2に短縮できること	低	事業部長
6	ハードウェア	RAMサイズ128KB	低	開発部署

非機能要件リストにはシステムの機能以外の制約、品質に関するものをまとめる。

16

データディクショナリの作成

設計の各工程で用いられる用語を統一するため、データディクショナリを作成する。データディクショナリの用語は以下の書式を用いる。

【記号の定義】

A ＝ B	AをBで定義する
A ＝ “text”	Aはリテラル”text”である
A ＝ 型、値域、単位系	Aは値域の範囲の値をもつ
A ＝ B ＋ C	AはBとCをもつ
[A ｜ B]	AまたはB
{ A }	Aの繰返し
(A)	Aはオプション

17

データディクショナリの例

給湯準備 ＝ “給湯口をロック解除する”

給湯禁止 ＝ “給湯口をロックする”

ロック状態 ＝ [“ロック” ｜ “ロック解除”]

沸騰 ＝ “水温が100℃の状態”

保温 ＝ “水温が95℃から100℃未満の状態”

動作状態 ＝ [“沸騰” ｜ “保温”]

水温 ＝ 整数、単位： ℃

水位 ＝ 整数、下限：0 上限：4 単位：なし

18

分析モデリング

職業能力開発総合大学校東京校
生産電子情報システム技術科
2011年11月8日

2011年12月7日　改訂

1

システム設計の流れ

要求モデリング

⇩

分析モデリング

⇩

設計モデリング

2

分析モデリングとは

分析モデリングでは、要求モデリングで「システムに何をしてほしいか」という視点で整理した結果をもとに、「システムが何を実現するか」という視点で分析する。

3

分析モデリングの必要性

分析モデリングなしで設計するとどのようなことが起こるか。

1. システムが利用者の意図に合わない
2. 仕様書の矛盾に気づかない
3. モジュールのインタフェースが合わない

4

分析モデリングで実施すること

1. システムのスコープを決める
   システムの対象範囲を明確にする
2. 機能を分割する
   抽象的なプロセスから、より具体的なプロセスに機能を分割する
3. 制御構造を決める
   各プロセス間の制御関係を検討する
4. 制御バーの動作仕様を決める
   制御を束ねた制御バーの動作仕様を定義する
5. プロセスの動作仕様を決める
   各プロセスの動作仕様を定義する

5

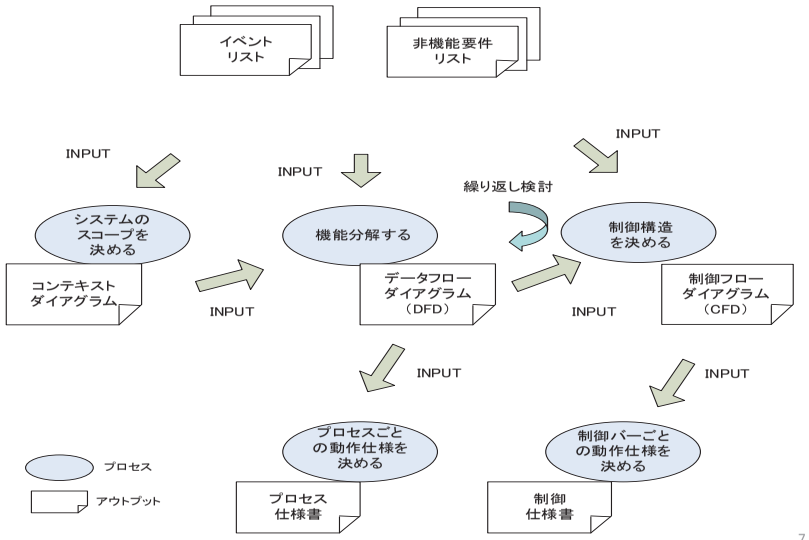
問3　分析モデリングではシステム機能をトップダウンで分割していく理由を考えなさい。

解答例

- ボトムアップでシステムを構築すると、詳細な事柄は分かるがシステムの全体像が明確にならない。
- システムの全体像が分からなければ、各機能の結びつきがどのようになるか明確にならない。
- 各機能の整合性がとりにくい。

6

分析モデリング



7

コンテキストダイアグラムの作成

要求モデル（イベントリスト、非機能要件リストのハードウェア制約）をもとにシステムの対象範囲を決め、コンテキストダイアグラムにまとめる。

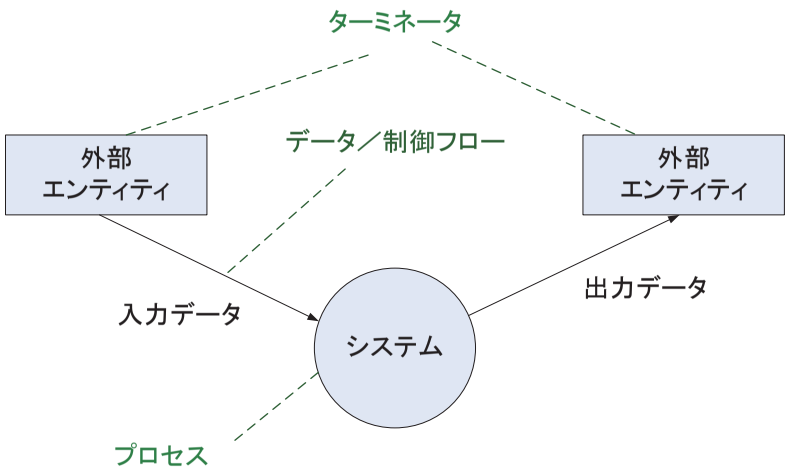
ターミネータ（外部エンティティ）：
システムの外部にありシステムに影響を与える、あるいはシステムから影響を受ける実体

プロセス：　開発対象システム

データ／制御フロー：　データまたは制御の流れ

8

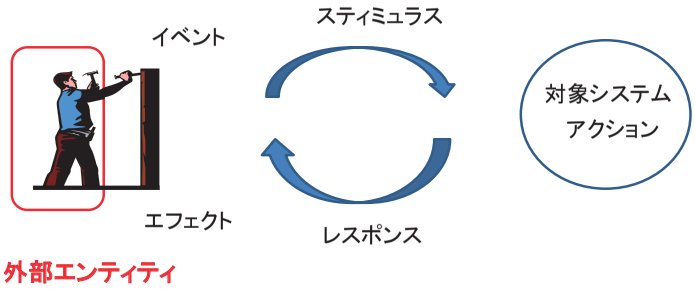
コンテキストダイアグラムの表記法



9

ターミネータの抽出

要求モデリングで作成したイベントリストからターミネータを抽出する。イベントリストの各イベントについて、その泉源、吸収となる外部エンティティを選ぶ。



外部エンティティ

10

イベントリスト（精査後）

利用者からのイベント

No	イベント	スティミュラス	アクション	レスポンス	エフェクト
1	沸騰要求	沸騰開始	沸騰行為をする カルキ抜きをする 目標の温度で保温行為をする	ヒータオン／オフ動作状態	安全なお湯が得られる 適量のお湯が得られる ポットの動作状態がわかる
2	保温温度の設定	保温モード設定	目標の保温温度を設定する	保温モード	お湯を欲しい温度にできる
3	給湯の準備	給湯準備	給湯口のロックを解除する	ロック状態	ポットから給湯が可能となる
4	給湯要求	給湯開始	ポットのお湯を給湯する	ポンプの作動	お湯が注がれる
5	給湯の完了	給湯停止	お湯の給湯を停止する	ポンプの停止	適量のお湯が得られる
6	誤給湯の防止	給湯禁止	給湯口をロックする	ロック状態	誤ってお湯をこぼさなくなる

データまたは制御

利用者からのイベント

よって、外部エンティティは利用者

11

イベントリスト（精査後）

利用者からのイベント（従属的な事象）

No	イベント	スティミュラス	アクション	レスポンス	エフェクト
1	時間計測要求	時間計測開始	残り時間を設定する タイマを起動する	タイマ残り時間	時間を計る手間が省ける
2	時間遅延要求	タイマ時間追加	残り時間を設定し直す	タイマ残り時間	意図した時間を指定できる
3	水の補充準備	ふた開	沸騰行為または保温行為を中断する	ヒータオフ動作状態	水を補充することが可能となる
4	水の補充完了	ふた閉	沸騰行為を開始する	ヒータオフ動作状態	安全なお湯が得られる

データまたは制御

利用者とふたセンサからのイベント

よって、外部エンティティは利用者とふたセンサ

12

イベントリスト（精査後）

センサなどからのイベント

No	イベント	スティミュラス	アクション	レスポンス	エフェクト
1	水位変化	水位	水量の表示を更新する	水量	お湯の残量がわかる
2	水位異常	満水 水なし	沸騰行為または保温行為を中止する	水量 警報情報 ヒータオフ	危険な状態を回避できる 装置の異常を知る
3	水温変化	水温	水温の表示を更新する 沸騰行為または保温行為をする	水温 ヒータオン／オフ	お湯の温度がわかる 指定した温度のお湯を得る
4	水温異常	水温	沸騰行為または保温行為を中止する	水量、 警報情報 ヒータオフ	危険な状態を回避できる 装置の異常を知る

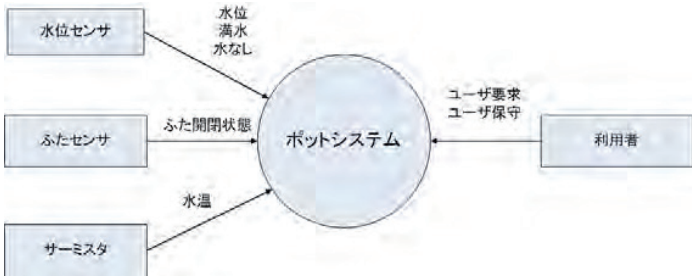
データまたは制御

センサーなどからのイベント

よって、外部エンティティは水位センサ、サーミスタ

13

話題沸騰ポットのコンテキストダイアグラム



ユーザ要求 ＝［ 沸騰開始 ｜ 給湯準備 ｜ 給湯開始 ｜ 給湯停止 ｜ 時間計測開始 ｜ タイマ時間追加 ］

ユーザ保守 ＝［ 給湯禁止 ｜ 保温モード設定 ］

ふた開閉状態 ＝［ “ふた開” ｜ “ふた閉” ］

➡ データディクショナリに追加

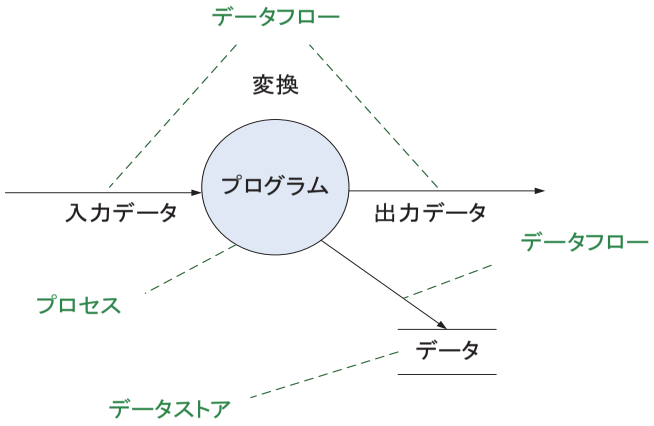
14

データフローダイアグラムの作成

要求モデルの結果（イベントリスト、非機能要件リスト）とコンテキストダイアグラムをもとに、システムの機能に着目し、抽象的なコンテキストダイアグラムから、より具体的なプロセスへ段階的に機能を分割する。このプロセスの分割は具体的な動作仕様が見えるまで行い、検討結果をデータフローダイアグラム（DFD: Data Flow Diagram）にまとめる。

15

データフローダイアグラムの表記法



データストア： 一時的なデータの保存

16

DFDOを作成

1. 主要機能（プロセス）を抽出する
   コンテキストダイアグラムのシステムを主要機能に分割する

2. 出力フローを検討する
   イベントリストのレスポンスから外部エンティティへの出力を考える

3. ターミネータを検討する
   出力フローの吸収先であるターミネータを抽出する

4. 入出力フローを書き込む
   主要プロセス、ターミネータ、入出力フローで結ぶ

17

イベントリスト（精査後）

利用者からのイベント

No	イベント	スティミュラス	アクション	レスポンス	エフェクト
1	沸騰要求	沸騰開始	沸騰行為をする カルキ抜きをする 目標の温度で保温行為をする	ヒータオン／オフ 動作状態	安全なお湯が得られる 適量のお湯が得られる ポットの動作状態がわかる
2	保温温度の設定	保温モード設定	目標の保温温度を設定する	保温モード	お湯を欲しい温度にできる
3	給湯の準備	給湯準備	給湯口のロックを解除する	ロック状態	ポットから給湯が可能となる
4	給湯要求	給湯開始	ポットのお湯を給湯する	ポンプの作動	お湯が注がれる
5	給湯の完了	給湯停止	お湯の給湯を停止する	ポンプの停止	適量のお湯が得られる
6	誤給湯の防止	給湯禁止	給湯口をロックする	ロック状態	誤ってお湯をこぼさなくなる

出力フロー

18

イベントリスト（精査後）

利用者からのイベント（従属的な事象）

No	イベント	スティミュラス	アクション	レスポンス	エフェクト
1	時間計測要求	時間計測開始	残り時間を設定する タイマを起動する	タイマ残り時間	時間を計る手間が省ける
2	時間遅延要求	タイマ時間追加	残り時間を設定し直す	タイマ残り時間	意図した時間を指定できる
3	水の補充準備	ふた開	沸騰行為または保温行為を中断する	ヒータオフ 動作状態	水を補充することが可能となる
4	水の補充完了	ふた閉	沸騰行為を開始する	ヒータオフ 動作状態	安全なお湯が得られる

出力フロー

19

イベントリスト（精査後）

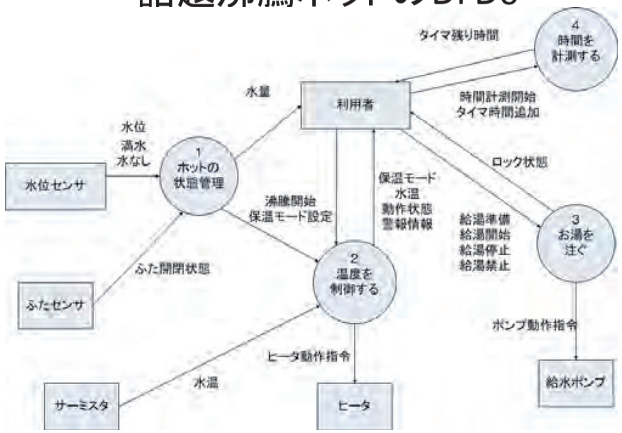
センサーなどからのイベント

No	イベント	スティミュラス	アクション	レスポンス	エフェクト
1	水位変化	水位	水量の表示を更新する	水量	お湯の残量がわかる
2	水位異常	満水 水なし	沸騰行為または保温行為を中止する	水量 警報情報 ヒータオフ	危険な状態を回避できる 装置の異常を知る
3	水温変化	水温	水温の表示を更新する 沸騰行為または保温行為をする	水温 ヒータオン／オフ	お湯の温度がわかる 指定した温度のお湯を得る
4	水温異常	水温	沸騰行為または保温行為を中止する	水量、 警報情報 ヒータオフ	危険な状態を回避できる 装置の異常を知る

出力フロー

20

話題沸騰ポットのDFD0



ヒータ動作指令 ＝［ ヒータオン ｜ ヒータオフ ］

ポンプ動作指令 ＝［ ポンプ作動 ｜ ポンプ停止 ］

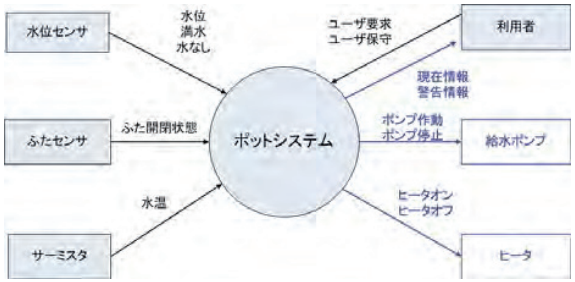
ふた開閉状態 ＝［ “ふた開” ｜ “ふた閉” ］

ロック状態 ＝ ［ “ロック” ｜ “ロック解除” ］

➡ **データディクショナリに追加**

コンテキストダイアグラムの見直し

追加したターミネータと出力フローをコンテキストダイアグラムに反映させる。



現在状態 ＝［ 動作状態 ｜ 保温モード ｜ 水温 ｜ 水位 ｜ ロック状態 ｜ タイマ残り時間 ］

動作状態 ＝［ “沸騰” ｜ “保温” ］

警報情報 ＝［ 温度異常 ｜ 水位異常 ］

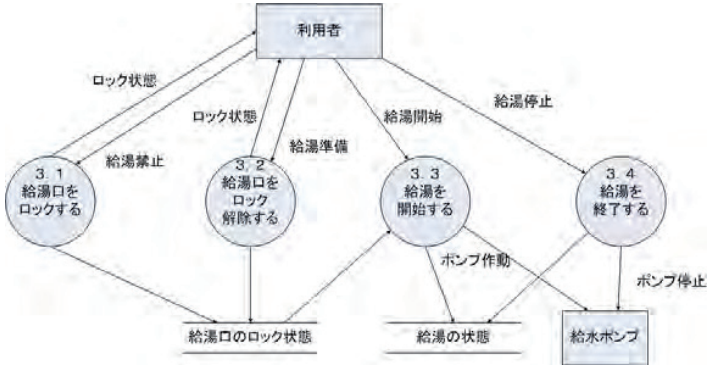
温度異常 ＝［ “高温異常” ｜ “温度上がらす異常” ］

水位異常 ＝ “水位の異常を表す”

➡ **データディクショナリに追加**

話題沸騰ポットのDFD3

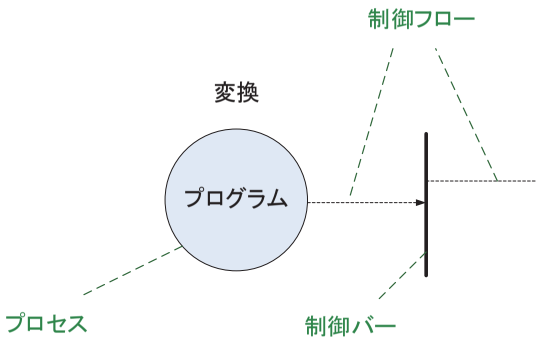
プロセス「3　お湯を注ぐ」をサブプロセスに分解する。



制御フローダイアグラムの作成

コンテキストダイアグラム、データフローダイアグラムについてプロセスがどの条件で起動／停止するかを検討する。検討結果を制御フローダイアグラム（CFD: Control Flow Diagram）にまとめる。

制御フローダイアグラムの表記法

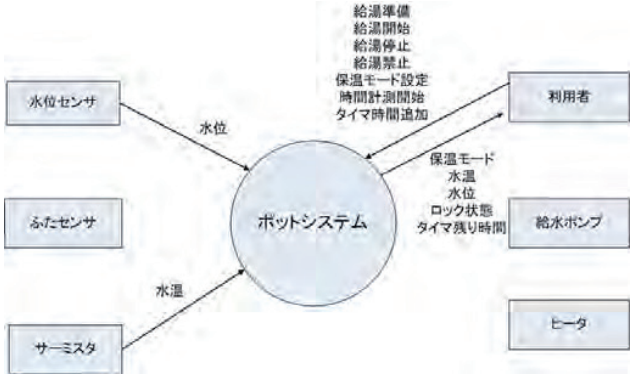


制御バー: 複数の制御フローを集約する

25

コンテキストダイアグラムの見直し

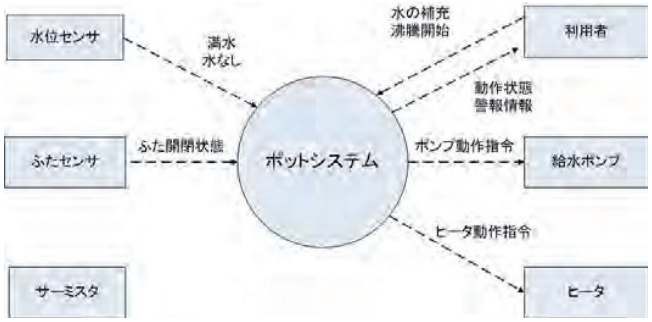
制御フローを検討した結果をコンテキストダイアグラム（データフロー）に反映させる。



26

コンテキストダイアグラムの見直し

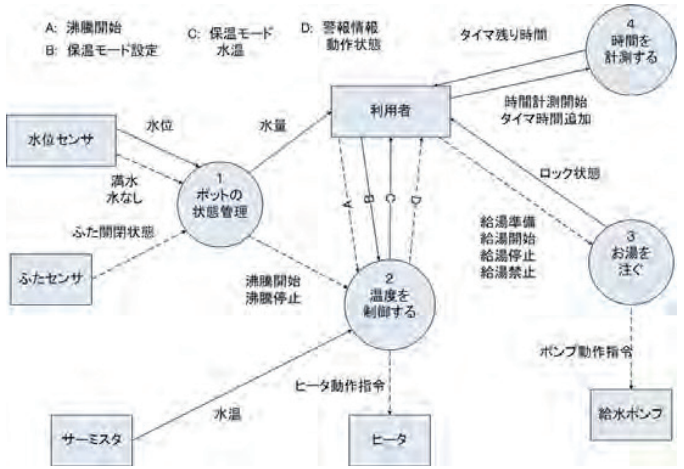
制御フローを検討した結果をコンテキストダイアグラム（制御フロー）に反映させる。



27

DFD0の見直し

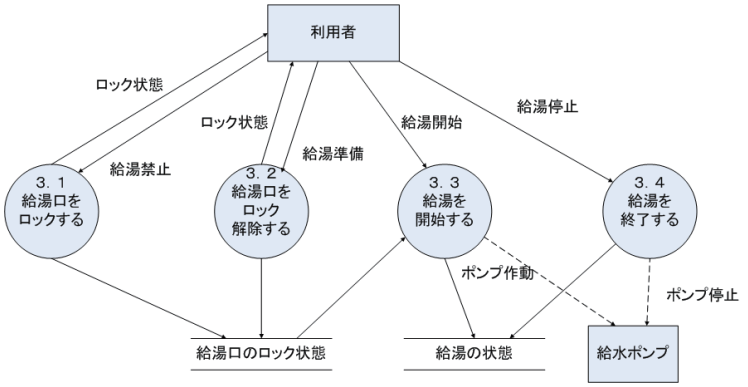
制御フローを検討した結果をDFD0に反映させる。



28

DFD3の見直し

制御フローを検討した結果をDFD3に反映させる。



29

制御仕様書の作成

制御フローダイアグラムをもとに、複数の制御を束ねた制御バーごとに動作仕様を定義し、 制御仕様書にまとめる。

記述方法

状態遷移図（STD：State Transition Diagram）

状態遷移表（STT：State Transition Table）

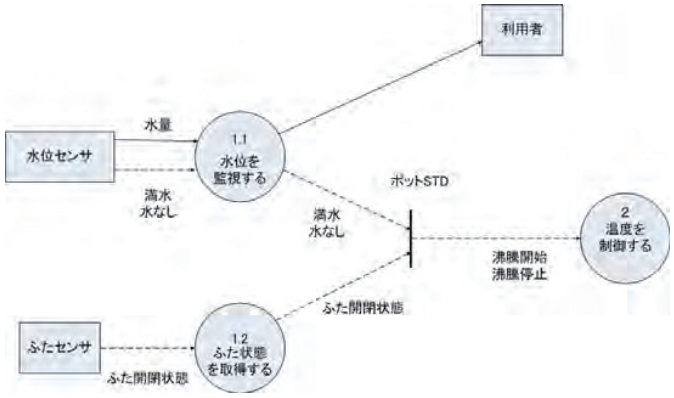
デシジョンテーブル（DT：Decision Table）

プロセス起動テーブル（PAT：Process Activate Table）

30

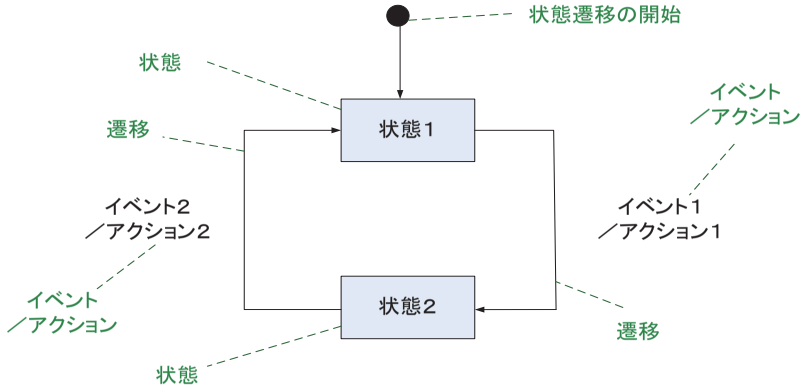
DFD1を分割

制御フローを考慮してDFD1を分割する



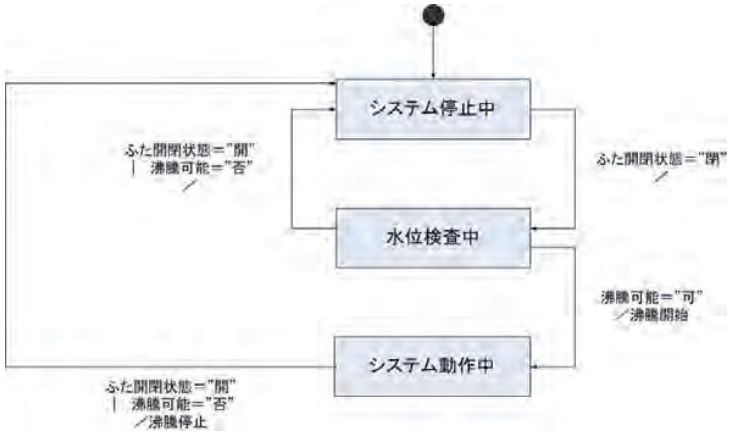
31

状態遷移図の表記法



32

制御仕様書：ポットSTD



状態遷移表の表記法

イベント / 状態	イベント１	・・・	イベントm
状態１	次の状態	・・・	次の状態
・・・	・・・	・・・	・・・
状態ｎ	次の状態	・・・	次の状態

×　＝　“起りえない（Can’t Happen）”

／　＝　“イベント無視（Don’t Care）”

次の状態　＝　［　状態１　｜　・・・　｜　状態n　｜　×　｜　／　］

制御仕様書：ポットSTT

イベント / 状態	ふた開閉状態＝“開”	ふた開閉状態＝“閉”	沸騰可能＝“可”	沸騰可能＝“否”
システム停止中	／	水位検査中	／	／
水位検査中	システム停止中	／	システム動作中 沸騰開始	システム停止中
システム動作中	システム停止中 沸騰停止	／	／	システム停止中 沸騰停止

デシジョンテーブルの表記法

条件１	Y	Y	・・・	Y	N
条件2	Y	Y	・・・	N	N
・・・	・・・	・・・	・・・	・・・	・・・
条件n-1	Y	Y	・・・	N	N
条件ｎ	Y	N	・・・	N	N
動作1	出力値	出力値	・・・	出力値	出力値
・・・	・・・	・・・	・・・	・・・	・・・
動作m	出力値	出力値	・・・	出力値	出力値

Y　＝　“条件を満たす”

N　＝　“条件を満たさない”

出力値　＝　［　○｜　－　］
○　＝　“動作を実行”
－　＝　“動作を実行しない”

制御仕様書：ポットDT

満水	Y	Y	N	N
水なし	Y	N	Y	N
沸騰可能	－（否）	－（否）	－（否）	○（可）

37

プロセス起動テーブルの表記法

入力	プロセス		
	プロセス番号１	・・・	プロセス番号m
イベント１	０または１	・・・	０または１
・・・	・・・	・・・	・・・
イベントｎ	０または１	・・・	０または１

０ ＝ “プロセス停止”

１ ＝ “プロセス起動”

38

制御仕様書：プロセス起動テーブル

入力	プロセス	
	３．３	３．４
給湯可	１	１
給湯不可	０	１

39

プロセス仕様書の作成

データフローダイアグラムの分割された最後のプロセスごとにプロセスの動作仕様を定義し、プロセス仕様書にまとめる。

40

プロセス仕様書:給湯を開始する

「3.3　給湯を開始する」のプロセス仕様書を作成する。

プロセス番号	3.3
プロセス名	給湯を開始する
入力	給湯口のロック状態
出力	給湯の状態
動作	// 給湯開始を受けて起動する １．if　給湯口のロック状態が解除 　　給水ポンプを作動する ２．給湯の状態を給湯中とする

41

データディクショナリの更新

分析モデリングにおいて新たに発生した用語をデータディクショナリに追加する。データディクショナリに存在するが、より詳細な定義ができる用語は修正する。

42

設計モデリング

職業能力開発総合大学校東京校
生産電子情報システム技術科
2011年12月6日

2011年12月7日　改訂

1

システム設計の流れ

要求モデリング

⇩

分析モデリング

⇩

設計モデリング

2

設計モデリングとは

設計モデリングでは、分析モデリングで「システムが何を実現するか」という視点で整理した結果をもとに、「どのようにしてシステムを実現するか」という視点で検討する。

3

設計モデリングの必要性

設計モデリングなしで設計するとどのようなことが起こるか。

1. モジュール間の呼び出しが分かりづらい
2. モジュール間の依存関係が分かりづらいので、ソフトウェアの保守が難しくなる

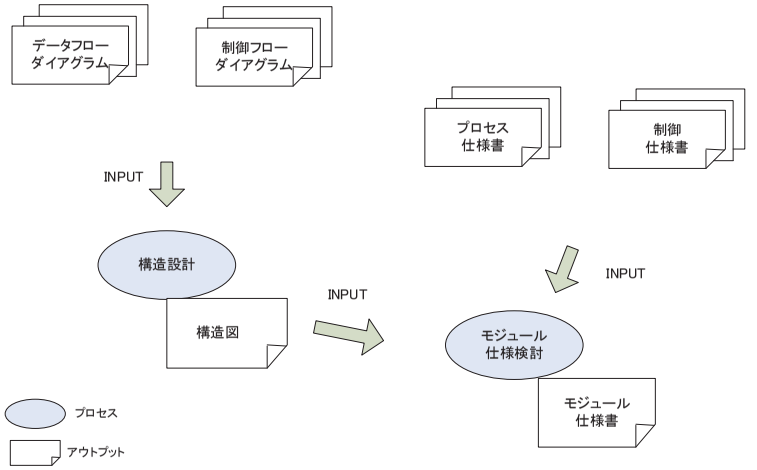
4

設計モデリングで実施すること

1. モジュール構造図を作成する
   機能分割に沿ってモジュールを分割しモジュール構造を設計する
2. モジュールの詳細仕様を決める
   各モジュールのインタフェース仕様と機能仕様を決める

5

設計モデリング



6

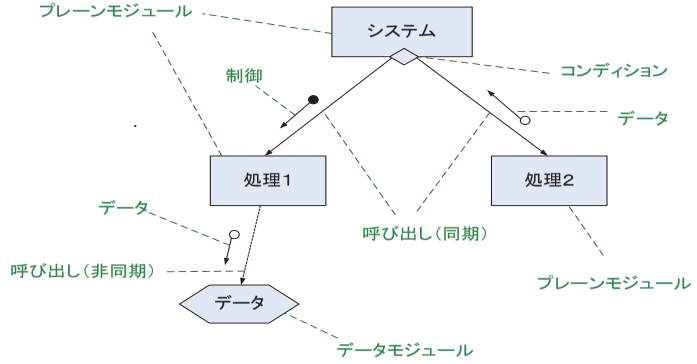
構造図の作成

データフローダイアグラムと制御フローダイアグラムをもとにシステムの機能を分割し、構造図（SC:Structure Chart）にまとめる。構造図は機能をまとめて制御するBOSSモジュールを定め、BOSSモジュールを中心に組み合わせて一つの構造図を作成する。

構造図を作成することにより、ソフトウェア設計の全体や、モジュールの依存関係が把握できる。

7

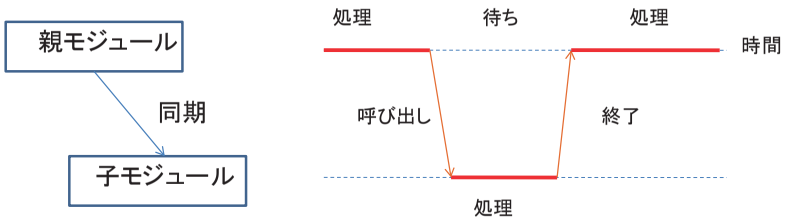
構造図の表記法



詳細は教科書P131図6.3を参照

8

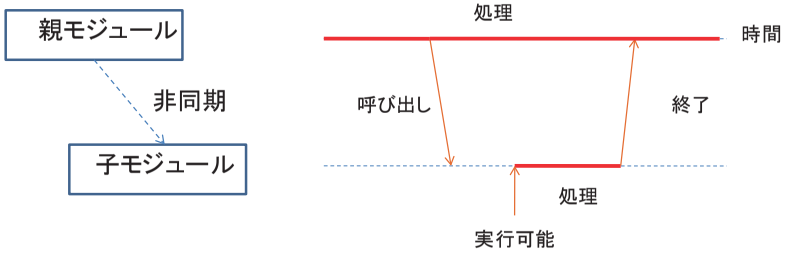
構造図の表記法　同期処理



親モジュールが子モジュールを呼び出すと直ちに子モジュールが実行される。親モジュールは子モジュールが終了するまで待ち、子モジュールが終了すると次の処理に移る。

9

構造図の表記法　非同期処理



親モジュールが子モジュールを呼び出すても直ちに子モジュールは実行されない。子モジュールが実行可能になったとき子モジュールは実行される。親モジュールは子モジュールの呼び出しに関係なく次の処理に移る。

10

BOSSモジュール

BOSSモジュール：　機能全体を制御するモジュール

BOSSモジュールを見つける方法

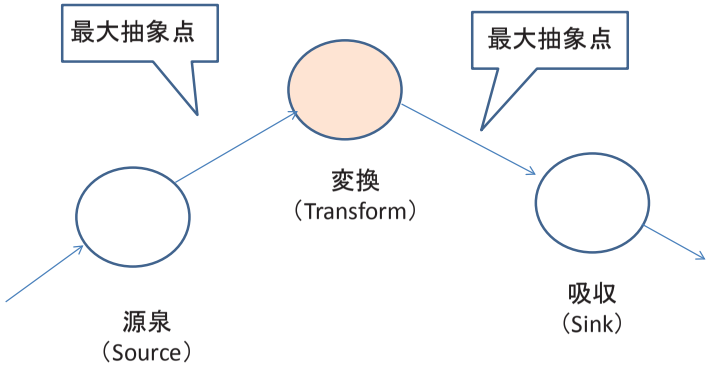
- 変換分析
- トランザクション分析

11

変換分析

・データが変換される箇所を探す

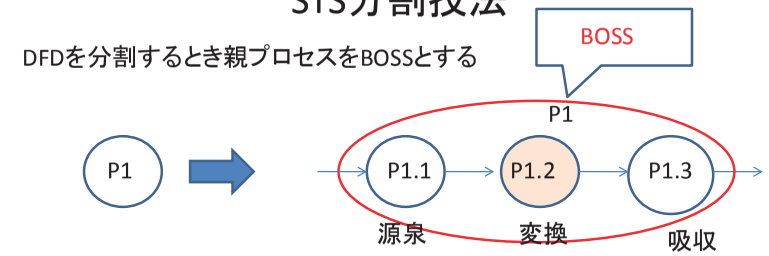
・すなわち最大抽象点を見つける



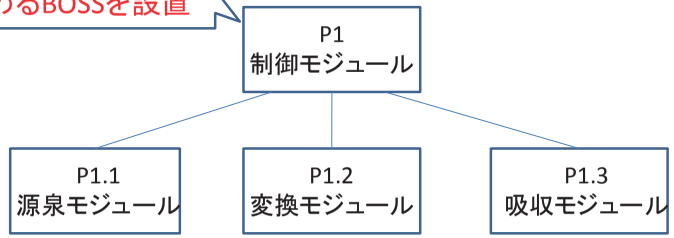
12

STS分割技法

DFDを分割するとき親プロセスをBOSSとする



源泉/変換/吸収をまとめるBOSSを設置

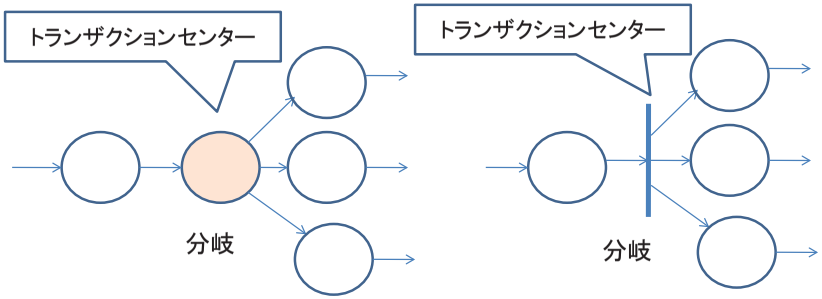


13

トランザクション分析

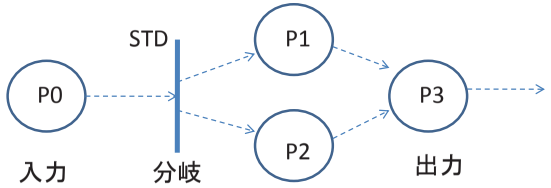
・データが交通整理される箇所

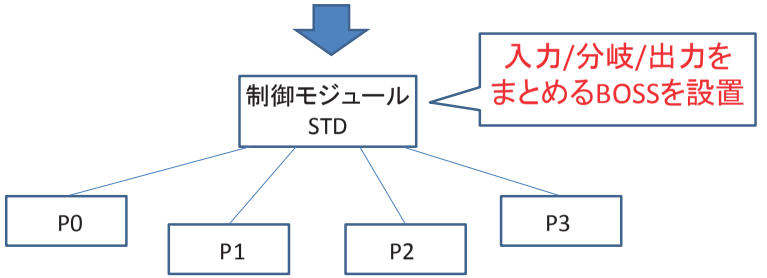
・すなわちトランザクションセンターを見つける



14

TR分割技法

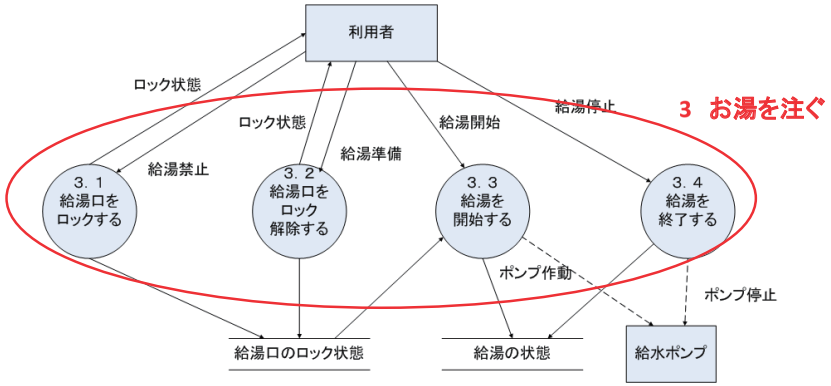




15

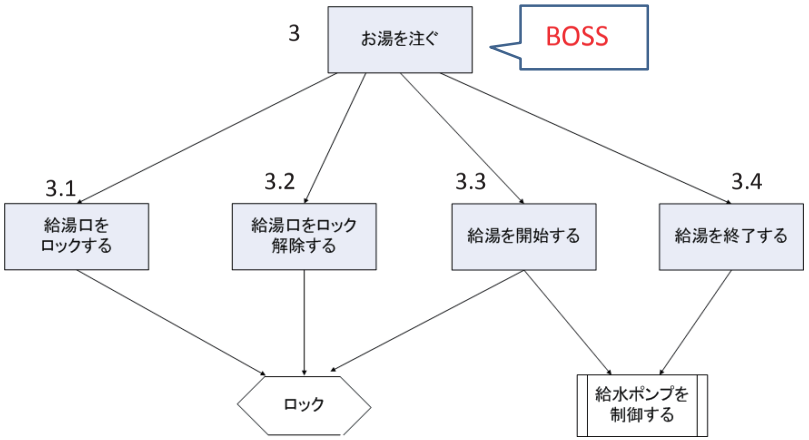
DFD3からBOSSモジュールを見つける

DFD3から最大抽象点を探す



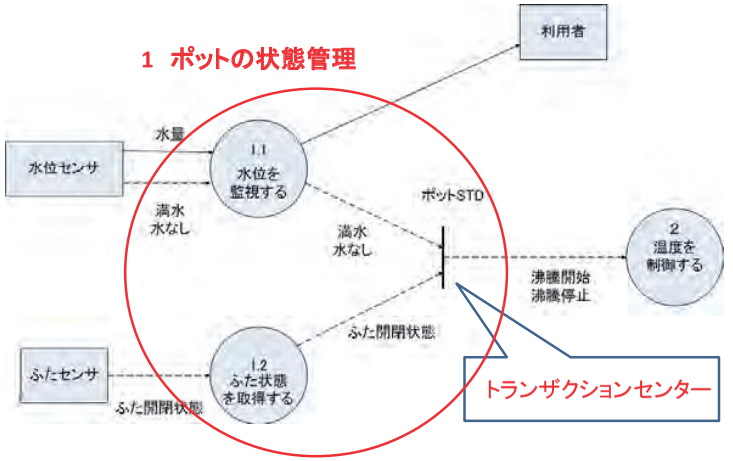
16

DFD3から作成した構造図



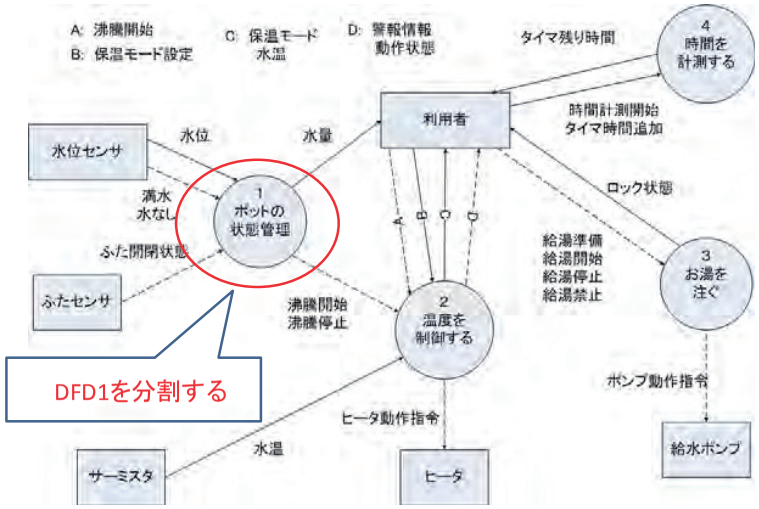
17

DFD1からBOSSモジュールを見つける



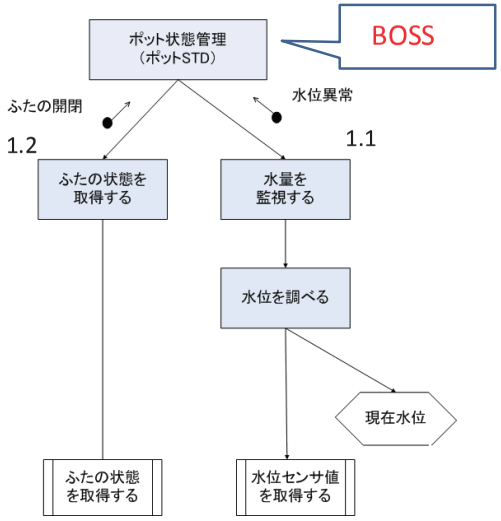
19

DFD1に着目



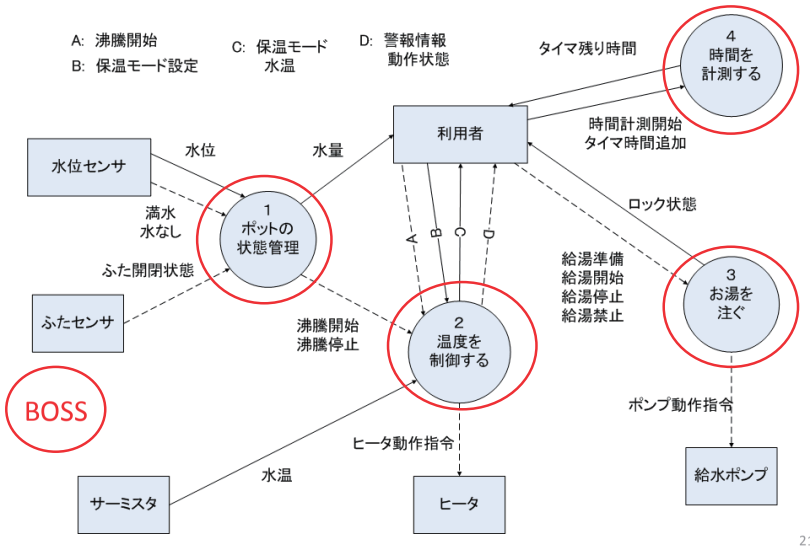
18

ポットSTDをBOSSとした構造図の部分



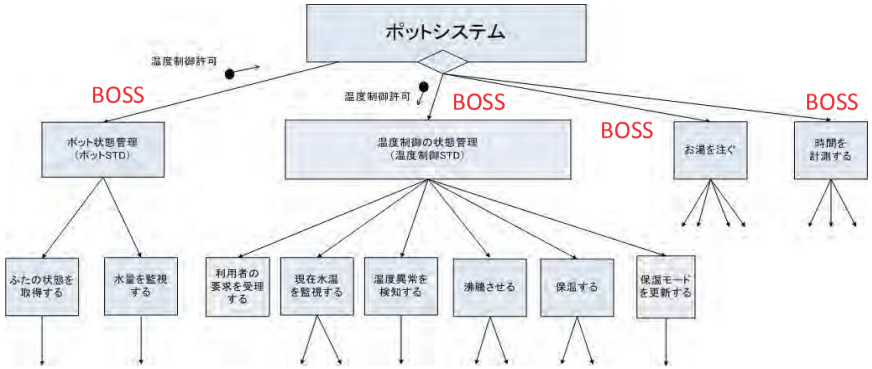
20

ポットシステムのBOSSモジュール



21

構造図



構造図全体は教科書P139図6.12を参照

22

モジュール仕様書の作成

プロセス仕様書、制御仕様書、構造図をもとに単一の機能を実現するモジュールに分割し、インタフェースの仕様、機能仕様をまとめる。

23

データディクショナリの更新

設計モデリングにおいて新たに発生した用語をデータディクショナリに追加する。データディクショナリに存在するが、より詳細な定義ができる用語は修正する。

24

2011/10/11

ネットワークプログラミング(その１)

UDP ソケット

UDP ソケットを用いて、クライアントからサーバに文字列”hello”を送信するプログラムを作成しなさい。クライアントからサーバへの送信は１回である。

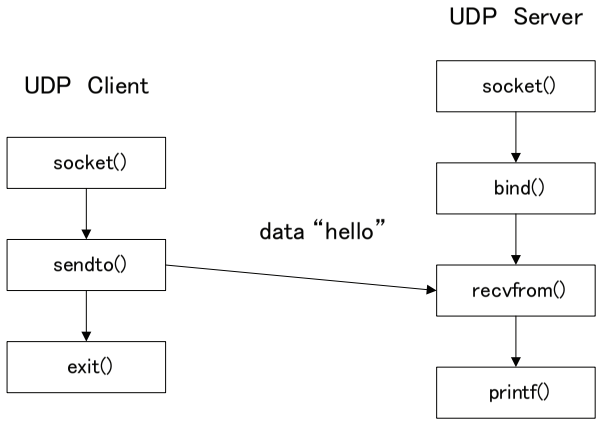


図１　UDP を用いたクライアントサーバプログラムの流れ

1　サーバプログラム

1.1　サーバプログラムを作成

```
# gedit  udpsrv01.c
```

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#define          MAXLINE         1024
#define          SERV_PORT       9877

int
main(int argc, char **argv)
{
        int             sockfd, n;
        struct  sockaddr_in     servaddr, cliaddr;
        socklen_t len;
        char    mesg[MAXLINE];

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(SERV_PORT);

        bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));

        len = sizeof(cliaddr);
        n = recvfrom(sockfd, mesg, MAXLINE, 0, (struct sockaddr *)&cliaddr, &len);
        mesg[len] = '¥0';
        printf("%s¥n",mesg);
}
```

1.2　サーバプログラムをコンパイル

```
# gcc  udpsrv01.c  -o  srv
```

2　クライアントプログラム

2.1　クライアントプログラムを作成

```
# gedit  udpcli01.c
```

```
#include        <stdio.h>
#include        <stdlib.h>
#include        <sys/types.h>
#include        <sys/socket.h>
#include        <netinet/in.h>
#include        <arpa/inet.h>
#include        <string.h>
#define         MAXLINE         4096
#define         SERV_PORT       9877

int
main(int argc, char **argv)
{
        int     sockfd, i, n;
        struct sockaddr_in      servaddr;
        char    sendline[MAXLINE] = "hello";

        if (argc != 2) {
                printf("usage: cli <IPaddress>¥n");
                exit(1);
        }

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(SERV_PORT);
        inet_pton(AF_INET, argv[1], &servaddr.sin_addr);

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);

        sendto(sockfd,sendline,strlen(sendline), 0,
                (struct sockaddr *)&servaddr,  sizeof(servaddr));
        exit(0);
}
```

2.2　クライアントプログラムをコンパイル

```
# gcc  udpcli01.c  -o  cli
```

3　実行１

１台のパソコンで実行する。端末を２枚開き、サーバとする端末を「端末 A」、クライアントとする端末を「端末 B」とする。

【端末 A】

3.1　サーバプログラムを実行

```
# ./srv  &
```

3.2　ポートが開いていることを確認

```
# netstat –na | more
```

【端末 B】

3.3　クライアントプログラムを実行

```
# ./cli  localhost
```

【端末 A】

3.4　“hello”が表示したことを確認

4　実行２

２台のパソコンで実行する。サーバとするパソコンを「HOSTA」、クライアントとするパソコンを「HOSTB」とする。

【HOSTA】

4.1　サーバプログラムを実行

```
# ./srv  &
```

4.2　ポートが開いていることを確認

```
# netstat –na | more
```

【HOSTB】

4.3　クライアントプログラムを実行

```
# ./cli  172.16.xx.xx          <-  サーバの IP アドレスを指定
```

【HOSTA】

4.4　“hello”が表示したことを確認

4.5　tcpdump を用いてパケットを観測

(IP アドレスが 172.16.12.200 のホストから 172.16.12.100 のホストに送信した例)

```
# tcpdump  -s  512  -X  src  or  dst  172.16.xx.xx
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 512 bytes
                    | | |
20:40:00.464407 IP 172.16.12.200.40628 > 172.16.12.100.9877: UDP, length 5
        0x0000:  4500 0021 0000 4000 4011 c97f ac10 0cc8  E..!..@.@.......
        0x0010:  ac10 0c64 9eb4 2695 000d 856b 6865 6c6c  ...d..&....khell
        0x0020:  6f00 0000 0000 0000 0000 0000 0000       o.............
```

5　ソケットアドレス構造体

```
struct in_addr {
    in_addr_t   s_addr ;              /*  32-bit IPv4 address              */
} ;                                   /*  network byte ordered             */

struct sockaddr_in {
    uint8_t             sin_len ;     /* length of structure (16)          */
    sa_family_t         sin_family ;  /* AF_INET                           */
    in_port_t           sin_port ;    /* 16-bit TCP or UDP port number     */
                                      /* network byte ordered              */
    struct in_addr      sin_addr ;    /* 32-bit IPv4 address               */
                                      /* network byte ordered              */
    char                sin_zero[8] ; /* unused                            */
} ;
```

Datatype	Description	Header
unit8_t	unsigned 8-bit integer	<sys/types.h>
unit16_t	unsigned 16-bit integer	<sys/types.h>
unit32_t	unsigned 32-bit integer	<sys/types.h>
sa_family_t	address family of socket address structure	<sys/socket.h>
socklen_t	length of socket address structure, normally uint32_t	<sys/socket.h>
in_addr_t	IPv4 address, normally uint32_t	<netinet/in.h>
in_port _t	TCP or UDP port, normally uint16_t	<netinet/in.h>

6　ソケットシステムコール

6.1　socket

socket function creates an endpoint for communication.

```
#include       <sys/socket.h>
int     socket(int  family, int  type, int  protocol) ;
        family : protocol family
                 AF_INET : IPv4 protocols
                 AF_INET6 : IPv6  protocols
        type : type of socket
                 SOCK_STREAM : stream socket
                 SOCK_DGRAM : datagram socket
                 SOCK_RAW : raw socket
        protocol : set to 0 except raw socket

        Return : nonnegative descriptor if OK, -1 on error
```

6.2　bind

bind function assigns a local protocol address to a socket.

```
#include       <sys/socket.h>
int     bind(int  sockfd, const  struct  sockaddr  *myaddr, socklen_t  addrlen) ;
        sockfd : descriptor of a socket
        myaddr : pointer to a protocol-specific address
        adderlen : size of the structure

        Retuen : 0 if OK, -1 on error
```

6.3　recvfrom

recvfrom function read messages from a socket. This function waits until data arraives from some client.

```
#include       <sys/socket.h>
ssize_t   recvfrom(int  sockfd, void  *buff, size_t  nbytes, int  flags,
                sturuct sockaddr  *from, socklen_t  *addrlen) ;
         sockfd : descriptor of a socket
         buff : pointer to buffer
         nbytes : number of bytes to read
         flags : set to zero
         from :  pointer to a socket address structure
         addrlen : pointer to an integer value (a value-result argument)

         Return : number of bytes read if OK, -1 on error
```

6.4　sendto

sendto function sends a message to a socket.

```
#include       <sys/socket.h>
ssize_t    sendto(int  sockfd, const void  *buff, size_t  nbytes, int  flags,
                const  struct  sockaddr  *to, socklen_t  addrlen) ;
         sockfd : descriptor of a socket
         buff : pointer to buffer
         nbytes : number of bytes to witer
         flags : set to zero
         ro :  pointer to a socket address structure
         addrlen : pointer to an integer value (a value-result argument)

        Return : number of bytes written if OK, -1 on error
```

ネットワークプログラミング(その２)

UDP ソケット

UDP ソケットを用いて、クライアントから英小文字の文字列を送信するとサーバは英大文字の文字列に変換して送り返すプログラムを作成しなさい。図１はこのプログラムの流れである。サーバは受信、変換、送信の処理を無限に繰り返し、クライアントは送信、受信の処理を Ctrl+c が押下されるまで繰り返す。socket()、bind()、sendto()、recvfrom()は、ソケットシステムコールであり、mytoupper()は英小文字から英大文字に文字列を変換する自作関数である。

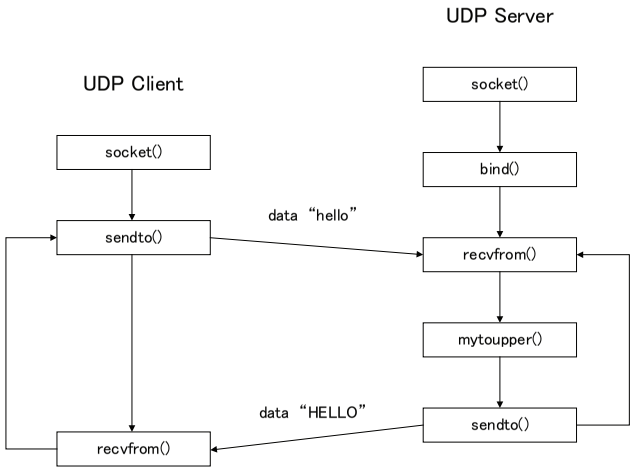


図１　UDP を用いたエコープログラムの流れ

7　その他の関数

7.1 bzero

bzero sets the specified number of bytes to 0 in the destination.

```
#include       <strings.h>

void    bzero(void  *dest, size_t  nbytes);
         dest:  pointer to destination
         nbyes:  number of bytes
```

7.2　inet_pton

inet_pton function converts a presentation format address(ASCII string) to network format(binary).

```
#include       <arpa/inet.h>
int     inet_pton(int  family, const char  *strptr, void  *addrptr);
         family: protocol family
                  AF_INET: IPv4 protocols
                  AF_INET6: IPv6  protocols
         strptr: pointer to a string
         addrptr: pointer to a binary data

                Return: 1 if OK, 0 if input not a valid presentation format, -1 on error
```

7.3　htons

htons function converts 16-bits value to 16-bits value in network byte order.

```
#include       <netinet/in.h>
uint16     htons(uint16_t  host16bitvalue);
           host16bitvalue: 16-bits value

                Return: value in network byte order
```

7.4　htonl

htonl function converts 32-bits value to 32-bits value in network byte order.

```
#include       <netinet/in.h>
unint132    htonl(uint32_t  host32bitvalue);
         host32bitvalue: 32-bits value

                Return: value in network byte order
```

1　実行１

１台のパソコンで実行する。端末を２枚開き、サーバとする端末を「端末 A」、クライアントとする端末を「端末 B」とする。

【端末 A】

1.1　サーバプログラムを実行

```
# ./srv  &
```

1.2　ポートが開いていることを確認

```
# netstat  -na | more
```

【端末 B】

1.3　クライアントプログラムを実行

```
# ./cli  localhost
hello
HELLO
Ctrl+c
```

2　実行２

２台のパソコンで実行する。サーバとするパソコンを「HOSTA」、クライアントとするパソコンを「HOSTB」とする。

【HOSTA】

2.1　サーバプログラムを実行

```
# ./srv  &
```

2.2　ポートが開いていることを確認

```
# netstat  -na | more
```

【HOSTB】

2.3　クライアントプログラムを実行

```
# ./cli  172.16.45.xx      <-  サーバの IP アドレスを指定
hello
HELLO
Ctrl+c
```


3　関数

3.1　fgets

```
#include   <stdio.h>
char   *fgets(char  *str, int  size, FILE  *stream);

The fgets() function reads at most one less than the number of characters
specified by size from the given stream and stores them in the string str.
Reading stops when a newline character is found, at end-of-file or error.  The
newline, if any, is retained.  If any characters are read and  there is no
error, a `¥0' character is appended to end the string.

Return : pointer to the string if OK , NULL on error
```

3.2　fputs

```
#include   <stdio.h>
Int    fputs(char  *str,  FILE  *stream);

The function fputs() writes the string pointed to by str to the stream pointed
to by stream.

Return : 0 if OK , EOF on error
```

4　プログラム例

4.1　サーバプログラム

gedit　udpsrv02.c

```
#include    <stdio.h>
#include    <sys/types.h>
#include    <sys/socket.h>
#include    <netinet/in.h>
#include    <arpa/inet.h>
#include    <string.h>
#define     MAXLINE     1024
#define     SERV_PORT   9877
int  mytoupper(char *m, int len);

int  main(int argc, char **argv)
{
     int     sockfd, n;
     struct  sockaddr_in servaddr, cliaddr;
     socklen_t len;
     char    mesg[MAXLINE];

     sockfd = socket(AF_INET, SOCK_DGRAM, 0);

     bzero(&servaddr, sizeof(servaddr));
     servaddr.sin_family = AF_INET;
     servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
     servaddr.sin_port = htons(SERV_PORT);

     bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));

     for ( ; ; ) {
         len = sizeof(cliaddr);
         n = recvfrom(sockfd, mesg, MAXLINE, 0, (struct sockaddr *)&cliaddr, &len);
         mytoupper(mesg, len);
         sendto(sockfd, mesg, n, 0, (struct sockaddr *)&cliaddr, len);
     }
}

int  mytoupper(char *m, int len)
{
     int     i;

     for (i = 0 ; i < len ; i++, m++)
         if (*m >= 'a' && *m <= 'z')
             *m -= 0x20;
 }
```

4.2　クライアントプグラム

gedit　udpcli02.c

```
#include    <stdio.h>
#include    <stdlib.h>
#include    <sys/types.h>
#include    <sys/socket.h>
#include    <netinet/in.h>
#include    <arpa/inet.h>
#include    <string.h>
#define     MAXLINE     4096
#define     SERV_PORT   9877

int     main(int argc, char **argv)
{
    int sockfd, i, n;
    struct sockaddr_in servaddr;
    char    sendline[MAXLINE], recvline[MAXLINE];

    if (argc != 2) {
        printf("usage: udpcli <IPaddress>¥n");
        exit(1);
    }

    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(SERV_PORT);
    inet_pton(AF_INET, argv[1], &servaddr.sin_addr);

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);

    while (fgets(sendline, MAXLINE, stdin) != NULL) {
        sendto(sockfd,sendline,strlen(sendline), 0,
                        (struct sockaddr *)&servaddr, sizeof(servaddr));
        n = recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);
        recvline[n] = 0;
        fputs(recvline, stdout);
    }
    exit(0);
}
```

ネットワークプログラミング(その３)

- ネットワークバイトオーダー、バッファ -

１．ネットワークバイトオーダ

２バイト以上の数値データをメモリに格納する場合、ビックエンディアン（Big Endian)、リトルエンディアン（Little Endian）と呼ばれる２種類の方法がある。ビックエンディアンは上位桁を下位バイトに格納し、リトルエンディアンは上位桁を上位バイトに格納する。ビックエンディアンを採用しているプロセッサには、SPARC、PowerPC、680x0 などがある。リトルエンディアンを採用しているプロセッサには、x86、Pentium などがある。ネットワークでは、データをビックエンディアンに変換し、上位桁から順に送信する。これをネットワークバイトオーダー（Network Byte Order）と呼ぶ。htons、htonl 関数はホストが使用しているバイトオーダーをネットワークバイトオーダーに変換する。

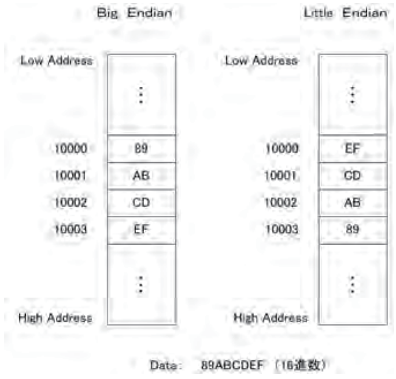


図１　リトルエンディアンとビックエンディアン

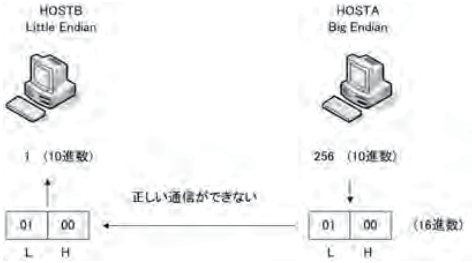


図２　コンピュータ間の通信（変換しない場合）

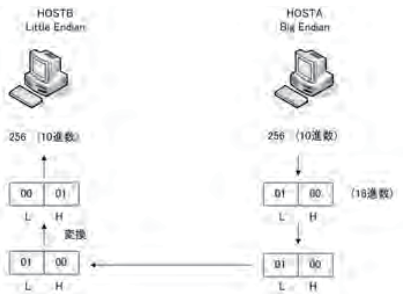


図３　コンピュータ間の通信（変換する場合）

1.1　エンディアンの確認プログラム

```
# gedit  byteorder.c
```

```
#include        <stdio.h>
#include        <stdlib.h>

int     main()
{
        union {
                short   s;
                char    c[sizeof(short)];
        } un;

        un.s = 0x0102;
        if (sizeof(short) == 2) {
                if (un.c[0] == 1 && un.c[1] == 2)
                        printf("big-endian¥n");
                else if (un.c[0] == 2 && un.c[1] == 1)
                        printf("little-endian¥n");
                else
                        printf("unknown¥n");
        } else
                printf("sizeof(short) = %d¥n", sizeof(short));
        exit(0);
}
```

1.2　実行例

```
# gcc  byteorder.c  -o byteorder
# ./byteorder
little-endian
#
```

２．バッファ

UDP ではアプリケーションのデータを保持する必要がないので、送信バッファは実在しないが、送信できるパケットの最大値は SO_SNDBUF で決められている。もし、送信バッファサイズより大きなデータをアプリケーションが送信しようとした場合、バッファの最大値を超えたデータは捨てられる。また、受信バッファは存在するが受信バッファの最大値を超えたデータが届いた場合、データは捨てられる。ここでは、送信バッファと受信バッファのデフォルトを調べる。

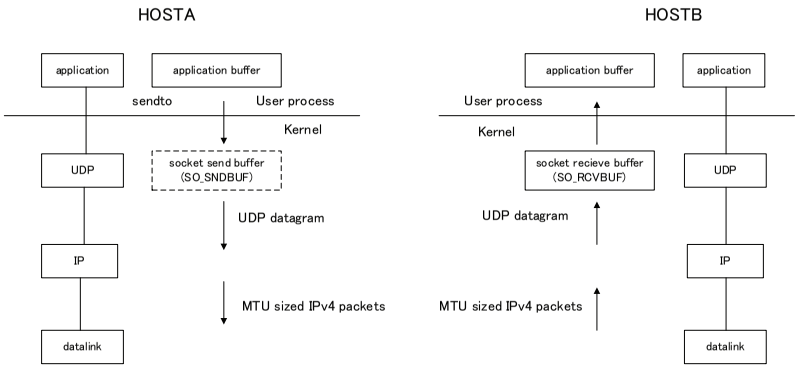


図４　UDP での送受信バッファ

2.1　sockopt.c をカレントディレクトリのダウンロード

2.2　送信バッファ、受信バッファサイズの確認プログラム

```
# gedit  checkopts.c
```

```
#include        <stdio.h>
#include        <stdlib.h>
#include        <netinet/tcp.h>
#include        <netinet/in.h>
#include        "sockopt.c"

int     main(int argc, char **argv)
{
        int     sockfd, optlen;

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);
        optlen = sizeof(val);

        getsockopt(sockfd, SOL_SOCKET, SO_RCVBUF, &val, &optlen);
        printf("SO_RCVBUF = %s¥n", sock_str_int(&val, optlen));

        getsockopt(sockfd, SOL_SOCKET, SO_SNDBUF, &val, &optlen);
        printf("SO_SNDBUF = %s¥n", sock_str_int(&val, optlen));
}
```

2.3　実行例

```
# gcc checkopts.c -o checkopts
# ./checkopts
SO_RCVBUF = 109568
SO_SNDBUF = 109568
#
```

2.4 sockopt.c プログラム

```
union val {
        int             i_val;
        long            l_val;
        char            c_val;
        struct linger   linger_val;
        struct  timeval timeval_val;
}val;

static char     *sock_str_flag(union val *, int);
static char     *sock_str_int(union val *, int);
static char     *sock_str_linger(union val *, int);
static char     *sock_str_timeval(union val *, int);

struct sock_opts {
        char    *opt_str;
        int     opt_level;
        int     opt_name;
        char    *(*opt_val_str)(union val *, int);
} sock_opts[] = {
        "SO_BROADCAST",  SOL_SOCKET,      SO_BROADCAST,    sock_str_flag,
        "SO_DEBUG",      SOL_SOCKET,      SO_DEBUG,        sock_str_flag,
        "SO_DONTROUTE",  SOL_SOCKET,      SO_DONTROUTE,    sock_str_flag,
        "SO_ERROR",      SOL_SOCKET,      SO_ERROR,        sock_str_int,
        "SO_KEEPALIVE",  SOL_SOCKET,      SO_KEEPALIVE,    sock_str_flag,
        "SO_LINGER",     SOL_SOCKET,      SO_LINGER,       sock_str_linger,
        "SO_OOBINLINE",  SOL_SOCKET,      SO_OOBINLINE,    sock_str_flag,
        "SO_RCVBUF",     SOL_SOCKET,      SO_RCVBUF,       sock_str_int,
        "SO_SNDBUF",     SOL_SOCKET,      SO_SNDBUF,       sock_str_int,
        "SO_RCVLOWAT",   SOL_SOCKET,      SO_RCVLOWAT,     sock_str_int,
        "SO_SNDLOWAT",   SOL_SOCKET,      SO_SNDLOWAT,     sock_str_int,
        "SO_RCVTIMEO",   SOL_SOCKET,      SO_RCVTIMEO,     sock_str_timeval,
        "SO_SNDTIMEO",   SOL_SOCKET,      SO_SNDTIMEO,     sock_str_timeval,
        "SO_REUSEADDR",  SOL_SOCKET,      SO_REUSEADDR,    sock_str_flag,
#ifdef  SO_REUSEPORT
        "SO_REUSEPORT",  SOL_SOCKET,      SO_REUSEPORT,    sock_str_flag,
#else
        "SO_REUSEPORT",  0,               0,               NULL,
#endif
        "SO_TYPE",       SOL_SOCKET,      SO_TYPE,         sock_str_int,
//      "SO_USELOOPBACK",SOL_SOCKET,      SO_USELOOPBACK,  sock_str_flag,  // BSD
        "IP_TOS",        IPPROTO_IP,      IP_TOS,          sock_str_int,
        "IP_TTL",        IPPROTO_IP,      IP_TTL,          sock_str_int,
        "TCP_MAXSEG",    IPPROTO_TCP,     TCP_MAXSEG,      sock_str_int,
        "TCP_NODELAY",   IPPROTO_TCP,     TCP_NODELAY,     sock_str_flag,
        NULL,            0,               0,               NULL
};

static char     strres[128];
```



```
static char     *sock_str_flag(union val *ptr, int len)
{
        if (len != sizeof(int))
                snprintf(strres, sizeof(strres), "size (%d) bit sizeof(int)", len);
        else
                snprintf(strres, sizeof(strres), "%s", (ptr->i_val == 0) ? "off" : "on");
        return(strres);
}

static char     *sock_str_linger(union val *ptr, int len)
{
        if (len != sizeof(int))
                snprintf(strres, sizeof(strres), "size (%d) bit sizeof(int)", len);
        else
                snprintf(strres, sizeof(strres), "%s", (ptr->i_val == 0) ? "off" : "on");
        return(strres);
}

static char     *sock_str_int(union val *ptr, int len)
{
        if (len != sizeof(int))
                snprintf(strres, sizeof(strres), "size (%d) bit sizeof(int)", len);
        else
                snprintf(strres, sizeof(strres), "%d", ptr->i_val);
        return(strres);
}
static char     *sock_str_timeval(union val *ptr, int len)
{
        if (len != sizeof(int))
                snprintf(strres, sizeof(strres), "size (%d) bit sizeof(int)", len);
        else
                snprintf(strres,sizeof(strres), %s",ctime(&(ptr->timeval_val.tv_sec)));
        return(strres);
}
```

2011/11/8

ネットワークプログラミング(その４)

パケットロス

UDP は通信の信頼性を保障しないので、送信したデータが受信者に届かない場合がある。通信中にデータを喪失するパケットロスの原因には、通信路のパケット量、送受信バッファのサイズが関係している。

１．パケットロスの頻度

図１はパケットロスの頻度を調べる実験である。HOSTB から HOSTA に 1024 バイトのパケットを連続して送信し、HOSTA で受信できたパケット数をカウントする。ただし、パケットロスは通信路のパケット量に依存するので、実験での結果はあくまでも目安である。

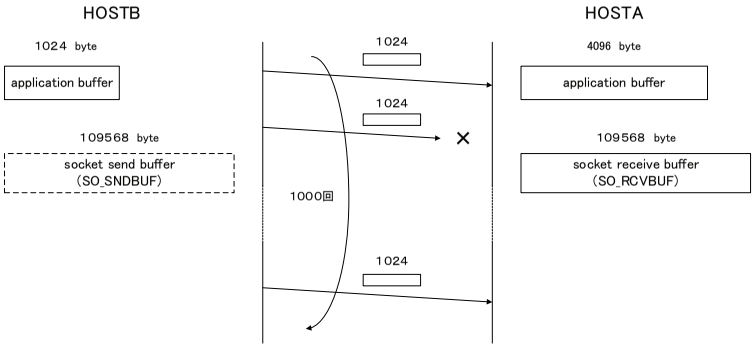


図１　パケットロスの実験

【HOSTA】

1.1　サーバプログラムの作成

```
# gedit  udpsrv03.c
```

```
#include        <stdio.h>
#include        <sys/types.h>
#include        <sys/socket.h>
#include        <netinet/in.h>
#include        <arpa/inet.h>
#include        <string.h>
#include        <signal.h>
#include        <stdlib.h>

#define         MAXLINE         4096
#define         SERV_PORT       9877
static  void    recvfrom_int(int);
static  int     count = 0;

int
main(int argc, char **argv)
{
        int             sockfd, n;
        struct  sockaddr_in     servaddr, cliaddr;
        socklen_t       len;
        char    mesg[MAXLINE];

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(SERV_PORT);

        signal(SIGINT, recvfrom_int);

        bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));

        for ( ; ; ) {
                len = sizeof(cliaddr);
                n = recvfrom(sockfd, mesg, MAXLINE, 0,
                        (struct sockaddr *)&cliaddr, &len);
                count++;
        }
}

static  void recvfrom_int(int signo)
{
        printf("¥nreceived %d datagrams¥n",count);
        exit(0);
}
```

1.2　サーバプログラムをコンパイル

```
# gcc  udpsrv03.c  -o  srv
```

【HOSTB】

1.3　クライアントプログラムの作成

```
# gedit  udpcli03.c
```

```
#include        <stdio.h>
#include        <stdlib.h>
#include        <sys/types.h>
#include        <sys/socket.h>
#include        <netinet/in.h>
#include        <arpa/inet.h>
#include        <string.h>
#define         SERV_PORT       9877
#define         NDG             1000
#define         SNDLEN          1024

int
main(int argc, char **argv)
{
        int     sockfd, i, n;
        struct sockaddr_in      servaddr;
        char    sendline[SNDLEN];

        if (argc != 2) {
                printf("usage: udpcli <IPaddress>¥n");
                exit(1);
        }

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(SERV_PORT);
        inet_pton(AF_INET, argv[1], &servaddr.sin_addr);

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);

        for (i = 0; i < NDG ; i++) {
                sendto(sockfd,sendline,SNDLEN, 0, (struct sockaddr *)&servaddr,
                        sizeof(servaddr));
        }
        exit(0);
}
```

1.4　クライアントプログラムをコンパイル

```
# gcc  udpcli03.c  -o  cli
```


【HOSTA】

1.5　サーバプログラムを実行

```
# ./srv
```

1.6　ポートが開いていることを確認

```
# netstat –na | more
```

【HOSTB】

1.7　クライアントプログラムを実行

```
# ./cli  172.16.45.xx          <-  サーバの IP アドレスを指定
```

【HOSTA】

1.8　受信したパケット数を確認

```
# ./srv                        <-  1.5 で入力済み
Ctrl+c
received 853 datagrams
```

【HOSTA, HOSTB】

1.9　「1.5」から「1.8」を 10 回繰り返す

```
# ./srv
Ctrl+c
received 853 datagrams
# ./srv
Ctrl+c
        ・・・
received 1000 datagrams
# ./srv
Ctrl+c
received 1000 datagrams
#
```

2　フラグメントの実験

MTU より大きいデータを送信するとフラグメントが発生する。

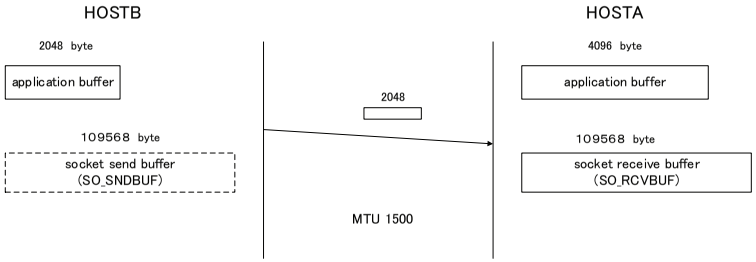


図２　フラグメントの実験

2.1　クライアントプログラムを変更

2048 バイトのデータを 1 回送信する。

```
# gedit  udpcli03.c
```

```
                ・・・
#define         NDG             1       <-  1000 を 1 に変更
#define         SNDLEN          2048    <-  1024 を 2048 に変更
                ・・・
```

2.2　クライアントプログラムをコンパイル

```
# gcc  udpcli03.c  -o  cli
```

【HOSTA】

2.3　端末 A から tcpdump を実行

(ethx は eth0 または eth1、172.16.45.xx は自ホストの IP アドレス)

```
# tcpdump  -i  ethx  src  or  dst 172.16.45.xx
```

2.4　端末 B からサーバプログラムを実行

```
# ./srv
```

【HOSTB】

2.5　クライアントプログラムを実行

```
# ./cli  172.16.45.xx          <-  サーバの IP アドレスを指定
```

【HOSTA】

2.6　パケットを観測

```
# tcpdump -i eth0 src or dst 172.16.45.xx
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
                        ・・・
03:28:29.675398 IP 172.16.45.yy.49478 > 172.16.45.xx.9877: UDP, length 2048
03:28:29.675441 IP 172.16.45.yy > 172.16.45.xx: udp
```

2.7　サーバプログラムを終了

```
#  Ctrl+c
```

3　バッファサイズを変える実験（その１）

アプリケーションの受信バッファより送信データが大きい場合、データを受信できるか調べる。送信データを 2048 バイト、アプリケーションの受信バッファを 1024 バイトとする。

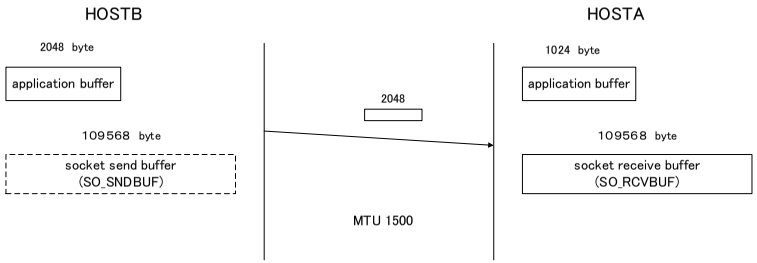


図３　アプリケーションの受信バッファサイズより大きいパケットの送信

3.1　アプリケーションの受信バッファを 1024 バイト、受信したデータのバイト数を表示

```
# gedit  udpcli03.c
```

```
                ・・・
#define         MAXLINE         1024    <-  4096 を 1024 に変更

                ・・・
                n = recvfrom(sockfd, mesg, MAXLINE, 0,
                        (struct sockaddr *)&cliaddr, &len);
                printf("n = %d¥n", n);          <-  追加
                ・・・
```

3.2　サーバプログラムをコンパイル

```
# gcc  udpsrv03.c  -o  srv
```

【HOSTA】

3.3　端末Ｂからサーバプログラムを実行

```
# ./srv
```

【HOSTB】

3.4　クライアントプログラムを実行

```
# ./cli  172.16.45.xx          <-  サーバの IP アドレスを指定
```

【HOSTA】

3.5　サーバプログラムを終了

```
# ./srv
n = 1024
ctrl+c
received 1 datagrams
#
```

4　バッファサイズを変える実験（その２）

受信バッファを1024バイト(実際は自動的に2倍の2048バイトとなる)、送信データを512、1024、2048バイトをした場合、受信のできたかどうか表１に記述する。udpsrv03.ｃをもとに受信バッファを変更するプログラム udpsrv04.ｃを作成しなさい。

（SO_SNDBUF 、SO_RCVBUF で指定した値は２倍に設定される）

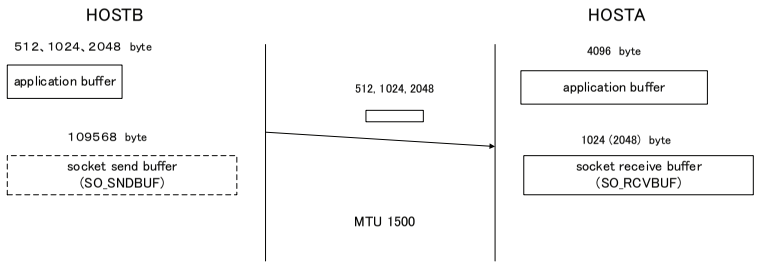


図４　受信バッファ（SO_RCVBUF）サイズと送信データサイズ

表１　受信バッファと送信データサイズ

受信バッファサイズ	1024	1024	1024
送信データサイズ	512	1024	2048
受信できたバイト数			

4.1　受信バッファサイズを変更

```
# gedit  udpcsrv04.c
```

```
#include        <stdio.h>
#include        <sys/types.h>
#include        <sys/socket.h>
#include        <netinet/in.h>
#include        <netinet/tcp.h>
#include        <arpa/inet.h>
#include        <string.h>
#include        <signal.h>
#include        <stdlib.h>
#include        "sockopt.c"

#define         MAXLINE         4096
#define         SERV_PORT       9877
static  void    recvfrom_int(int);
static  int     count = 0;

int
main(int argc, char **argv)
{
        int             sockfd, fd, n, rbufsize, optlen;
        struct  sockaddr_in     servaddr, cliaddr;
        socklen_t len;
        char    mesg[MAXLINE];

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(SERV_PORT);

        signal(SIGINT, recvfrom_int);

        rbufsize = 1024;
        setsockopt(sockfd, SOL_SOCKET, SO_RCVBUF, &rbufsize, sizeof(rbufsize));

        optlen = sizeof(val);
        getsockopt(sockfd, SOL_SOCKET, SO_RCVBUF, &val, &optlen);
        printf("SO_RCVBUF = %s¥n", sock_str_int(&val, optlen));

        bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));

        for ( ; ; ) {
                len = sizeof(cliaddr);
                n = recvfrom(sockfd, mesg, MAXLINE, 0,
                        (struct sockaddr *)&cliaddr, &len);
                printf("n = %d¥n", n);
                count++;
```

```
        }
}

static  void recvfrom_int(int signo)
{
        printf("¥nreceived %d datagrams¥n",count);
        exit(0);
}
```

4.2 サーバプログラムをコンパイル

```
# gcc  udpsrv04.c  -o  srv
```

4.3 送信データサイズを変更

```
# gedit  udpcli03.c
```

```
                ・・・
#define         SNDLEN          1024    <-  512,1024,2048 に変更
                ・・・
```

4.4 クライアントプログラムをコンパイル

```
# gcc  udpcli03.c  -o  cli
```

【HOSTA】

4.5 サーバプログラムを実行

```
# ./srv
```

【HOSTB】

4.6 クライアントプログラムを実行

```
# ./cli  172.16.45.xx          <-  サーバの IP アドレスを指定
```

【HOSTA】

4.7 サーバプログラムを終了

```
# ./srv
SO_RCVBUF = 2048
n = 1024
ctrl+c
received 1 datagrams
#
```

＜サンプルプログラム　１（checkopts）＞

```
#include<stdio.h>
#include<stdlib.h>
#include<netinet/tcp.h>
#include<netinet/in.h>

union val {
        int                     i_val;
        long                    l_val;
        char                    c_val;
        struct linger           linger_val;
        struct  timeval         timeval_val;
}val;

static char     *sock_str_int(union val *, int);

struct sock_opts {
        char    *opt_str;
        int     opt_level;
        int     opt_name;
        char    *(*opt_val_str)(union val *, int);
} sock_opts[] = {
        "SO_RCVBUF",            SOL_SOCKET,   SO_RCVBUF,              sock_str_int,
        "SO_SNDBUF",            SOL_SOCKET,   SO_SNDBUF,              sock_str_int
};

int     main(int argc, char **argv)
{
        int     fd, len, bufsize;
        struct  sock_opts *ptr;

        fd = socket(AF_INET, SOCK_DGRAM, 0);
        len = sizeof(val);
        if (getsockopt(fd, SOL_SOCKET, SO_RCVBUF, &val, &len) == -1) {
                printf("getsockopt error¥n");
                exit(0);
        }
        printf("SO_RCVBUF defalt = %s¥n", sock_str_int(&val, len));
        if (getsockopt(fd, SOL_SOCKET, SO_SNDBUF, &val, &len) == -1) {
                printf("getsockopt error¥n");
                exit(0);
        }
        printf("SO_SNDBUF defalt = %s¥n", sock_str_int(&val, len));

        bufsize = 1024;
        if (setsockopt(fd, SOL_SOCKET, SO_RCVBUF, (int *)&bufsize, sizeof(bufsize)) == -1) {
                printf("setsockopt error¥n");
                exit(0);
        }
        if (getsockopt(fd, SOL_SOCKET, SO_RCVBUF, &val, &len) == -1) {
                printf("getsockopt error¥n");
                exit(0);
        }
        printf("SO_RCVBUF defalt = %s¥n", sock_str_int(&val, len));
        if (getsockopt(fd, SOL_SOCKET, SO_SNDBUF, &val, &len) == -1) {
                printf("getsockopt error¥n");
                exit(0);
        }
        printf("SO_SNDBUF defalt = %s¥n", sock_str_int(&val, len));
        exit(0);
}

static char     strres[128];

static char     *sock_str_int(union val *ptr, int len)
{
        if (len != sizeof(int))
                snprintf(strres, sizeof(strres), "size (%d) bit sizeof(int)", len);
        else
                snprintf(strres, sizeof(strres), "%d", ptr->i_val);
        return(strres);
}
```

＜サンプルプログラム　2（udpcli01）＞

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<string.h>
#define  MAXLINE         4096
#define  SERV_PORT       9877
#define  SA          struct sockaddr

int
main(int argc, char **argv)
{
        int         sockfd, i, n;
        struct sockaddr_in          servaddr;
        char        sendline[MAXLINE] = "hello";

        if (argc != 2) {
                printf("usage: udpcli <IPaddress>¥n");
                exit(1);
        }

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(SERV_PORT);
        inet_pton(AF_INET, argv[1], &servaddr.sin_addr);

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);

        sendto(sockfd,sendline,strlen(sendline), 0, (struct sockaddr *)&servaddr,
                        sizeof(servaddr));
        exit(0);
}
```

＜サンプルプログラム　3（udpcli02）＞

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<string.h>
#define  MAXLINE         4096
#define  SERV_PORT       9877

int
main(int argc, char **argv)
{
        int         sockfd, i, n;
        struct sockaddr_in          servaddr;
        char        sendline[MAXLINE], recvline[MAXLINE];

        if (argc != 2) {
                printf("usage: udpcli <IPaddress>¥n");
                exit(1);
        }

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(SERV_PORT);
        inet_pton(AF_INET, argv[1], &servaddr.sin_addr);

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);

        while (fgets(sendline, MAXLINE, stdin) != NULL) {
                sendto(sockfd,sendline,strlen(sendline), 0, (struct sockaddr *)&servaddr,
                                sizeof(servaddr));
                n = recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);
                recvline[n] = 0;
                fputs(recvline, stdout);
        }
        exit(0);
}
```

＜サンプルプログラム　4（udpserv01）＞

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<string.h>
#define  MAXLINE         1024
#define  SERV_PORT       9877

int
main(int argc, char **argv)
{
        int                     sockfd, n;
        struct    sockaddr_in       servaddr, cliaddr;
        socklen_t len;
        char      mesg[MAXLINE];

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(SERV_PORT);

        bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));

        len = sizeof(cliaddr);
        n = recvfrom(sockfd, mesg, MAXLINE, 0, (struct sockaddr *)&cliaddr, &len);
        mesg[len] = '¥0';
        printf("%s¥n",mesg);
}
```


＜サンプルプログラム　5（udpserv02）＞

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<string.h>
#define  MAXLINE         1024
#define  SERV_PORT       9877

int       mytoupper(char *m, int len);

int
main(int argc, char **argv)
{
        int                     sockfd, n;
        struct    sockaddr_in       servaddr, cliaddr;
        socklen_t len;
        char      mesg[MAXLINE];

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(SERV_PORT);

        bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));

        for ( ; ; ) {
                len = sizeof(cliaddr);
                n = recvfrom(sockfd, mesg, MAXLINE, 0, (struct sockaddr *)&cliaddr, &len);
                mytoupper(mesg, len);
                sendto(sockfd, mesg, n, 0, (struct sockaddr *)&cliaddr, len);
        }
}

int       mytoupper(char *m, int len)
{
        int       i;

        for (i = 0 ; i < len ; i++, m++)
                if (*m >= 'a' && *m <= 'z')
                        *m -= 0x20;
}
```

2012年9月18日

DirectFBでの画像表示

1．MS104-LCD/AUDIO の構成

MS104-LCD/AUDIOはMS104-SH4AGの専用LCDオーディオボードであり、そのシステム構成は図１のとおりである。MS104-LCD/AUDIOのシステムは、ハードウェア、Linuxカーネル、DirectFB（Direct Frame Buffer）、ALSA（Advanced Linux Sound Architecture）ライブラリ、アプリケーションソフトウェアで構成する。ハードウェアはMS104-LCD/AUDIOとLinuxカーネルが動作するターゲットボードMS104-SH4AGである。Linuxカーネルには、LCD/タッチパネルデバイスドライバ、ALSAデバイスドライバが組み込まれている。DirectFBはLinux Frame Buffer Driver上で実装されたグラフィックAPI（Application Program Interface）である。ALSAは、オーディオやMIDI（Musical Instrument Digital Interface）機能を提供するAPIである。アプリーションソフトウェアからはDirectFBを用いて図形・画像表示、タッチパネル操作、ALSAを用いて音声出力を行うことができる。

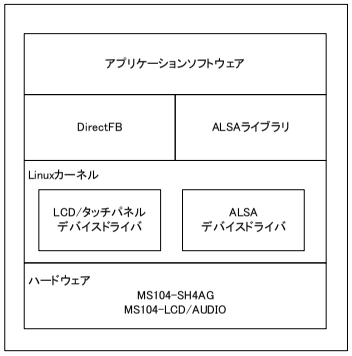


図１　MS104-LCD/AUDIO のシステム構成

2．DirectFB

DirectFB は少ないメモリでグラフィック処理を行い、入力デバイス管理、抽象レイヤーの作成、ウィンドウの統合をサポートし、カーネルを変更せずにフレームバッファ上に透過的なウィンドウと多層の描画レイヤーを提供する。DirectFB は使用するメモリが少ないため、メモリ容量に制限がある組込みシステムで用いられる。

2.1　DirectFB の機能

DirectFB のが提供するグラフィック機能は、以下のとおりである。

- 矩形（Rectangle）描画
- 三角形（Triangle）描画
- 直線（Line）描画
- ブリッティング（Blitting）
- 合成（Blending）
- 透過（Alpha Channel）
- 色（Colorizing）

ブリッティングは画像のコピー操作で、等倍コピーやフォーマット変換を行う。

画像は JPEG、PNG、GIF が、動画は video4linux、mpeg1/2、AVI、macromedia flash が扱える。

2.2　DirectFB での描画

DirectFB ではビットイメージの画像を保持するメモリ領域をサーフェイス（Surface）と呼ぶ。図形などの描画はサーフェイス上で行い、ビットブロック転送もサーフェイス間で行う。フルスクリーンモードでは、表示画面をプライマリサーフェイス（Primary Surface）と呼び、直接グラフィック操作が可能である。すべてのサーフェイスはダブルバッファリングが使用でき、ダブルバッファリングを用いるとちらつき（flickering）を防止できる。サーフェイスの一部をサブサーフェイス（SubSurface）と呼び、サーフェイスと同様に扱える。サブサーフェイスは新たにメモリ領域を割り当てることはせずに、サーフェイスの一部を使用する。

グラフィック装置の種類により、サーフェイスを複数もつことができる。各サーフェイスはメモリ上異なる領域を使用する。１画面に複数のサーフェイスを重ね、透過的に合成（Alpha Blending）して表示できるので、この重なったサーフェイスをレイヤ（Layer）と呼ぶ。

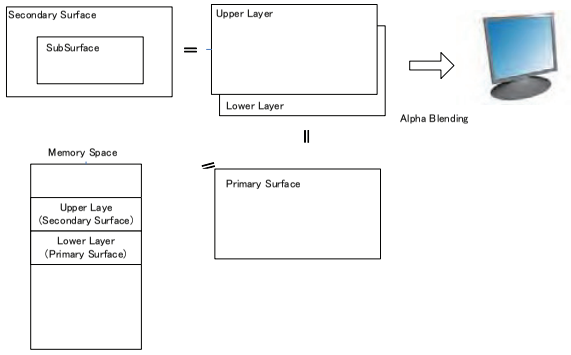


図２　サーフェイスとレイヤ

2.3　グラフィック表示

矩形、直線、三角形、画像などのグラフィック表示は、サーフェイス上で行う。表示画面に対応したサーフェイスであるプライマリサーフェイスではグラフィック表示を直接実行できる。矩形、直線、三角形は DrawRectangle、DrawLine、DrawTriangle 関数を用いて、サーフェイス上に直接描画する。JPEG、PNG、GIF などの画像は、ファイルからイメージプロバイダ（Image Provider）に取り込み、RendTo 関数を用いてスケーリング、カラーフォーマット変換を自動的に行う。文字列の表示は、フォントを生成し、DrawString 関数によりサーフェイスに出力する。

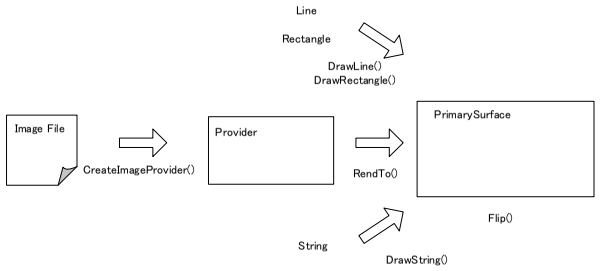


図３　グラフィック表示

2.4　インタフェースの関係

DirectFB を使用する場合、最初にスーパーインタフェースである IDirectFB を生成し、IDirectFB を基にして、サーフェイス IDirectFBSurface を生成する。画像を表示する場合、IDirectFBImageProvider を生成する。文字列を表示する場合、IDirectFBFont を生成する。

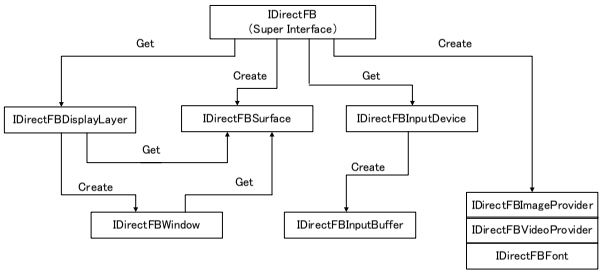


図４　DirectFB インタフェースの関係

2012/10/2

カメラ情報と画像ファイルの取得

ビューアソフト luvcview を用いて Web カメラの情報と画像ファイルを得る。カメラの情報には、画像フォーマット、画像の大きさである幅、高さがある。luvcview が保存できる画像は、PNM 形式（Portable aNyMap）であり、PNM 形式には以下の３形式がある。

- PBM 形式（portable bitmap format）：白黒２値のビットマップ
- PGM 形式（portable graymap format）：グレイスケール
- PPM 形式（portable pixmap format）：RGB フルカラー

PNM 形式の画像ファイルには、ヘッダに２バイトの File Descriptior があり、この File Descriptior により形式が分かる。Encoding はデータがテキストであるかバイナリであるかを示している。

File Descriptor	Type	Encoding
P1	Portable bitmap	ASCII
P2	Portable graymap	ASCII
P3	Portable pixmap	ASCII
P4	Portable bitmap	Binary
P5	Portable graymap	Binary
P6	Portable pixmap	Binary

1　ビューアソフトをインストール

1.1　CentOS5.3 を起動

1.2　Root でログイン

1.3　luvcview をダウンロード

1.4　ディレクトリを移動

```
# cd /usr/local/src
```

1.5　解凍

```
# gzip -d luvcview-20070512.tar.gz
```

1.6　tar ファイルを展開

```
# tar xvf luvcview-20070512.tar
```

1.7　プログラムをメイク

```
# cd luvcview-20070512
# make
```

1.8　実行ファイルをコピー

```
# cp luvcview /usr/local/bin
```

2　カメラ（Logicool）情報の取得

2.1　カメラの USB 端子をパソコンに接続

2.2　デバイス名を確認

```
# ls -l /dev/vi*
lrwxrwxrwx 1 root root     6  9月 21 00:21 /dev/video -> video0
crw------- 1 root root 81, 0  9月 21 00:21 /dev/video0
#
```

2.3　カメラの情報を得る

```
# luvcview  -L
luvcview version 0.2.1
Video driver: x11
A window manager is available
video /dev/video0
/dev/video0 does not support read i/o
{ pixelformat = 'YUYV', description = 'YUV 4:2:2 (YUYV)' }
{ discrete: width = 640, height = 480 }
       Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 160, height = 120 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 176, height = 144 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 320, height = 240 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 352, height = 288 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 640, height = 360 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 640, height = 400 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ pixelformat = 'MJPG', description = 'MJPEG' }
{ discrete: width = 640, height = 480 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 160, height = 120 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 176, height = 144 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 320, height = 240 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 352, height = 288 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 640, height = 360 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
{ discrete: width = 640, height = 400 }
   Time interval between frame: 1/30, 1/25, 1/20, 1/15, 1/10, 1/5,
#
```

3　画像ファイルを得る

3.1　作業用ディレクトリを作成

```
# cd
# mkdir  image
# cd  image
```

3.2　画像を表示

```
# luvcview  -s  320x240  -f  yuv
```

3.3　表示している画像をファイルに保存

中央の４ボタンの中で左上ボタンをクリック

3.4　ビューアを終了

中央の４ボタンの中で右下ボタンをクリック

3.5　ファイルサイズを確認

```
# ls -l
-rw-r--r-- 1 root root 230415  9 月 28 08:28 P-09:28:2012-08:28:49.pnm
```

ファイルサイズ：

横×縦×１ピクセル当りのビット数＋ヘッダサイズ　＝　320×240×3＋15　＝　230415

3.6　ヘッダと形式の確認

```
# od -x -c P-09:28:2012-08:28:49.pnm | more
0000000 3650 330a 3032 3220 3034 320a 3535 760a
          P    6   ¥n    3    2    0         2    4    0   ¥n    2    5    5   ¥n    v
0000020 696e 6d75 7468 656f 6e73 7364 656d 6d73
          n    i    u    m    h    t    o    e    s    n    d    s    m    e    s    m
```

ヘッダは“P6¥n320 240¥n255¥n”である。ヘッダの長さは 15 バイト、ヘッダの先頭から 2 バイトが File Descriptior である。File Descriptior が P6 なので PPM 形式である。

3.7 ファイル名を変更

```
# mv  P-09:28:2012-08:28:49.pnm  foo.ppm
```

2012/10/30

ビデオキャプチャ

カメラからの画像を Linux に取り込むために Video for Linux Two(V4L2) を用いる。V4L2 は Linux でビデオキャプチャ (Video Capture) するための API (Application Program Interface) である。ビデオキャプチャとは、カメラやビデオデッキなどの装置 (Device) から映像 (Video)、画像 (Image)、音声 (Audio) をデジタルデータ (Digital Data) として取り込むことである。

ビデオキャプチャでは、カメラなどのデバイスとアプリケーションの間でデータの読み書きにストリーミング I/O (Streaming Input Output) とメモリマッピング (MMAP:Memory Mapping) 用いる。ストリーミング I/O はデバイスを操作するドライバとアプリケーションとの間でデータをコピーせずに、バッファへのポインタのみで行う方法である。メモリマッピングは、カメラのビデオメモリ (Video Memory)、すなわちデバイスメモリ (Device Memory) をアプリケーションのメモリにマッピングさせる方法である。メモリマッピングを用いると、アプリケーションはデバイスメモリでなく、アプリケーションのメモリにアクセスすることでデータを読むことができる。

1　ビデオキャプチャの手順

V4L2 を用いて、カメラから画像を撮るビデオキャプチャの操作手順は、以下のとおりである。ここで、ビデオデバイスとはカメラのことである。この手順に従ったプログラム例を「2　ビデオキャプチャプログラム」に示す。ビデオデバイスへのパラメータのセット、パラメータの問い合わせは、ioctl を拡張した xioctl 関数で実行する。

(1)　ビデオデバイスをオープン
(2)　ビデオデバイスを初期化
(3)　ビデオキャプチャを開始
(4)　画像処理
(5)　ビデオキャプチャを停止
(6)　初期化前の状態に戻す
(7)　ビデオデバイスをクローズ

1.1　ビデオデバイスをオープン

ビデオキャプチャのデバイス名は「/dev/video0」である。ビデオデバイスのオープンはデバイス名「/dev/video0」とモードに「O_RDWR」「O_NONBLOCK」を指定し、open 関数を呼び出す。「O_RDWR」はビデオデバイスが読み書きともに使用できることを意味する。「O_NONBLOCK」を指定すると、出力キューにバッファがない場合、画像データの読み込みを待たず、すなわちブロックせずに EAGAIN エラーを返す。

1.2　ビデオデバイスを初期化

ビデオデバイスの初期化は画像の幅「IMG_WIDTH」、画像の高さ「IMG_HEIGHT」、ピクセルフォーマット「V4L2_PIX_FMT_YUYV」などビデオキャプチャに必要な情報を、リクエストコード「VIDIOC_S_FMT」を指定して xioctl でビデオデバイスに設定する。デバイスメモリから画像データを読む場合、メモリマッピングを用いる。ビデオキャプチャした画像データはカメラのビデオメモリ、すなわちデバイスメモリに格納される。メモリマッピングにより、デバイスメモリをアプリケーションのメモリにマッピングさせ、アプリケーションのメモリから画像データを読む。

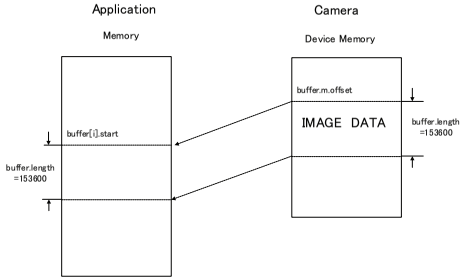


図 1　デバイスメモリとアプリケーションのメモリとのマッピング

メモリマッピングは次の手順で初期化する。

(1)　バッファ数、バッファタイプ「V4L2_BUF_TYPE_VIDEO_CAPTURE」、メモリマッピング「V4L2_MEMORY_MMAP」を、リクエストコード「VIDIOC_REQBUFS」でビデオデバイスに設定する。
(2)　バッファへのポインタとバッファサイズを保持する構造体「struct buffer」のメモリ領域を calloc 関数でヒープ (heap) 領域に確保する。
(3)　バッファタイプ「V4L2_BUF_TYPE_VIDEO_CAPTURE」、MMAP「V4L2_MEMORY_MMAP」、バッファ数を指定し、リクエストコード「VIDIOC_QUERYBUF」でビデオデバイスに問い合わせ、ビデオデバイスからバッファサイズ「buf.length」とオフセット「buf.m.offset」を得る。
(4)　バッファサイズ「buf.length」、ポートの READ/WRITE「PORT_READ | PORT_WRITE」、共有指定「MAP_SHARED」、ファイルハンドラ「fd」、バッファのオフセット「buf.m.offset」を指定し、mmap 関数を呼び出す。mmap 関数は割り当てたメモリの先頭アドレスを返す。
(5)　アプリケーションからアクセスできるように、mmap が割り当てたメモリの先頭アドレスを「buffers[n_buffers].start」に格納する。
(6)　バッファ数だけ(3)～(5)を繰り返す。

図 2 はバッファ数 4 のときのメモリマッピングの構造である。バッファサイズは、画面の幅が 320 ピクセル、画面の高さが 240 ピクセル、ピクセルフォーマットが YUYV 形式 (YUV422 形式) であり 1 ピクセル 2 バイトなので、153600 (320×240×2) バイトである。このサイズのバッファをデバイスメモリに 4 個確保する。構造体「struct buffer」には、バッファサイズ 153600 と mmap 関数が返したバッファの先頭アドレスを格納する。バッファが複数存在するので「struct buffer」を配列で管理し、その配列の先頭アドレスを変数「buffers」で保持する。

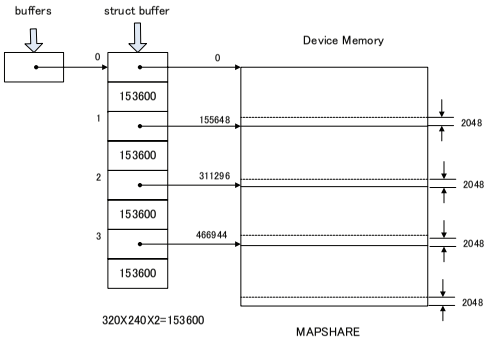


図２　メモリマッピングの構造

1.3　ビデオキャプチャの開始

図３はストリーミング I/O の動作を示している。ストリーミング I/O 機能をもつ装置、すなわちストリーミングデバイス（Streaming Device）は、入力キュー（Incoming Queue）と出力キュー（Outgoing Queue）をもつ。ストリーミング I/O の動作は、アプリケーションがバッファを入力キューに挿入（Enqueue）することから始める。次に入力キューからバッファを取り出し（Dequeue）、そのバッファにキャプチャした画像データを書き込む。書き込みが終わると、画像データが書かれたバッファを出力キューに挿入する。アプリケーションは出力キューからバッファを取り出し、画像データを読み、読み終えたバッファを再び入力キューに挿入する。

ビデオキャプチャを開始する場合、入力キューにすべてのバッファを挿入する。ただし、挿入されるバッファは、バッファ自身でなくバッファの情報を記述した構造体「struct v4l2_buffer」である。「strcut v4l2_buffer」にはバッファの識別子「index」があり、この識別子で複数のバッファを区別する。入力キューへの挿入は、リクエストコード「VIDIOC_QBUF」により実行する。ストリーミングデバイスは、リクエストコード「VIDIOC_STREAMON」によりストリーミングを開始する。

1.4　画像処理

ビデオキャプチャでは、入力キューから構造体「struct v4l2_buffer」を取り出し、その構造体「struct v4l2_buffer」の識別子で指定されたデバイスメモリに画像データを格納する。画像データを格納すると、出力キューに「struct v4l2_buffer」を挿入する。アプリケーションは出力キューから「struct v4l2_buffer」を取り出し、必要な処理を行う。もし、出力キューに「struct v4l2_buffer」が存在しない場合、エラーEAGAIN が返される。出力キューにバッファが存在するかどうか select 関数によりチェックする。

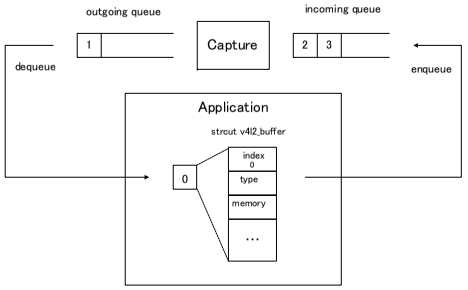


図３　ビデオキャプチャと入出力キュー

1.5　終了処理

ビデオキャプチャを終了する場合、初期化する以前の状態に戻して終了する。ビデオキャプチャの停止は、リクエストコード「IDIOC_STREAMOFF」をビデオデバイスに通知する。次にメモリマッピングを munmap 関数で解消し、calloc 関数で割り当てた領域を free 関数で解放する。最後にビデオデバイスを close 関数で閉じる。

2　ビデオキャプチャプログラム

$ **gedit capture.c**

```
/*
 *  V4L2 video capture example
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <getopt.h>             /* getopt_long() */
#include <fcntl.h>              /* low-level i/o */
#include <unistd.h>
#include <errno.h>
#include <malloc.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/mman.h>
#include <sys/ioctl.h>
#include <asm/types.h>          /* for videodev2.h */
#include <linux/videodev2.h>

#define CLEAR(x) memset(&(x), 0, sizeof (x))

#define IMG_WIDTH        320
#define IMG_HEIGHT       240

struct buffer {
    void         *start;
    size_t       length;
};

static char *dev_name     = "/dev/video0";          // ビデオデバイス名
static int  fd            = -1;                     // ビデオデバイスのハンドラ
struct buffer *buffers    = NULL;                   // バッファへの先頭アドレス
static unsigned int n_buffers = 0;                  // バッファ数
static char *output_file = "room.yuv";              // 出力ファイル

static void errno_exit(const char *s)
{
    fprintf(stderr,"%s error %d, %s¥n", s, errno, strerror (errno));
    exit (EXIT_FAILURE);
}

static int xioctl(int fd, int request, void * arg)
{
    int r;

    do r = ioctl(fd, request, arg);
    while (-1 == r && EINTR == errno);

    return r;
}

static void process_image(const void *p)
{
    // 画像を処理
}

static int read_frame(void)
{
    struct v4l2_buffer buf;

    CLEAR (buf);
    buf.type      = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    buf.memory    = V4L2_MEMORY_MMAP;
                  // outgoing queue から１フレーム分のバッファを取り出す
    if (xioctl(fd, VIDIOC_DQBUF, &buf) == -1) {
        switch (errno) {
            case EAGAIN:          // outgoing queue が空のとき
                return 0;
            default:
                errno_exit("VIDIOC_DQBUF");
        }
    }
    assert(buf.index < n_buffers);
    process_image(buffers[buf.index].start); // フレームを処理
    xioctl(fd, VIDIOC_QBUF, &buf);   // 使用したバッファを incoming queue に入れる

    return 1;
}

static void mainloop(void)
{
    unsigned int count;
    count = 1;                              // capture するフレーム数を指定

    while (count-- > 0) {
        for (;;) {
            fd_set fds;
            struct timeval tv;
            int r;

            FD_ZERO(&fds);
            FD_SET(fd, &fds);
            tv.tv_sec = 5;                  // タイムアウト時間を設定
            tv.tv_usec = 0;
            r = select(fd+1, &fds, NULL, NULL, &tv);  // USB カメラからのイベント待ち

            if (r == -1) {                  // エラー処理
                if (EINTR == errno)
                    continue;
                    errno_exit("select");
            }
            if (r == 0) {                   // タイムアウト処理
                fprintf (stderr, "select timeout¥n");
                    exit (EXIT_FAILURE);
            }
            if (read_frame()){    // 読み込み OK になったらフレームを１枚読み込む
                printf(".");
                break;
            } else {
                printf("EAGAIN¥n");
            }
            // EAGAIN : outgoing queue が空のとき繰り返す
```

```
        }
    }
}

static void stop_capturing(void)
{
    enum v4l2_buf_type type;

    type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    xioctl(fd, VIDIOC_STREAMOFF, &type);  // ストリーミングを停止
}

static void start_capturing(void)
{
    unsigned int i;
    enum v4l2_buf_type type;

    for (i = 0; i < n_buffers; ++i) {
        struct v4l2_buffer buf;
        CLEAR(buf);
        buf.type        = V4L2_BUF_TYPE_VIDEO_CAPTURE;
        buf.memory      = V4L2_MEMORY_MMAP;
        buf.index       = i;                // バッファの識別子
        xioctl(fd, VIDIOC_QBUF, &buf);    // incoming queueにバッファをエンキュー
    }
    type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    xioctl(fd, VIDIOC_STREAMON, &type);   //  ストリーミングを開始
}

static void uninit_device(void)
{
    unsigned int i;

    for (i = 0; i < n_buffers; ++i)
        munmap(buffers[i].start, buffers[i].length);        // アンマップ
        free(buffers);                                      // メモリを開放
}

static void init_mmap(void)
{
    struct v4l2_requestbuffers req;

    CLEAR (req);
    req.count    = 4;                                // バッファ数
    req.type     = V4L2_BUF_TYPE_VIDEO_CAPTURE;    // バッファタイプ
    req.memory   = V4L2_MEMORY_MMAP;                // MMAP を使用
    xioctl(fd, VIDIOC_REQBUFS, &req);               // メモリマッピングの設定

    buffers = calloc(req.count, sizeof(*buffers)); //  動的にメモリを確保

    for (n_buffers = 0; n_buffers < req.count; ++n_buffers) {
        struct v4l2_buffer buf;

        CLEAR(buf);
        buf.type    = V4L2_BUF_TYPE_VIDEO_CAPTURE;
        buf.memory  = V4L2_MEMORY_MMAP;
        buf.index   = n_buffers;
        xioctl(fd, VIDIOC_QUERYBUF, &buf);

        buffers[n_buffers].length = buf.length;
                //  デバイスメモリとアプリケーションのメモリをマッピング（共有）
        buffers[n_buffers].start = mmap(NULL,       // start anywhere
                buf.length,                         // バッファの長さ
                PROT_READ | PROT_WRITE,             // ポートの read/write
                MAP_SHARED,                         // 共有を指定
                fd, buf.m.offset);                  // ファイルハンドラ、オフセット
    }
}

static void init_device(void)
{
    struct v4l2_format fmt;

    CLEAR(fmt);
    fmt.type                = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    fmt.fmt.pix.width       = IMG_WIDTH;            // キャプチャ画像の解像度：幅
    fmt.fmt.pix.height      = IMG_HEIGHT;           // キャプチャ画像の解像度：高さ
    fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_YUYV;// ピクセルフォーマットの設定：YUYV
    fmt.fmt.pix.field  = V4L2_FIELD_INTERLACED; // 受信するビデオフォーマットの設定
    xioctl(fd, VIDIOC_S_FMT, &fmt);             // 映像フォーマットの設定

    init_mmap(); // メモリマッピングの初期化
}

static void close_device(void)
{
    close(fd);
}

static void open_device(void)
{
    struct stat st;

    fd = open(dev_name, O_RDWR | O_NONBLOCK, 0);
}

int main(void)
{
    open_device();              //  ビデオバイスをオープン
    init_device();              //  ビデオデバイスを初期化
    start_capturing();          //  画像キャプチャ開始
    mainloop();                 //  メインループ処理
    stop_capturing();           //  画像キャプチャ停止
    uninit_device();            //  初期化前の状態に戻す
    close_device();             //  ビデオデバイスをクローズ
    return 0;
}
```

3　課題

ビデオキャプチャプログラム「capture.c」を用いて、キャプチャした YUV422 形式の画像をファイル「room.yuv」に保存しなさい。ビデオキャプチャプログラムをダウンロードし、関数「process_image」の中にプログラムを追加すればよい。fwrite 関数を用いると簡潔に記述できる。

4　プログラム例

```
static void process_image(const void *p)
{
    FILE *fp;

    fp = fopen(output_file, "wb");
    fwrite(p, 1, IMG_HEIGHT*IMG_WIDTH*2, fp);
    fclose(fp);
}
```

2012/9/18

LCD プログラミング(その１)

― 解像度を調べる ―

MS104-LCD/AUDIO の LCD について、その解像度を調べる。DirectFB を使用ためには、DirectFB の初期化（DirectFBInit）、スーパインタフェースの生成（DirectFBCreate）、スパーインタフェースのフルスクリーン設定（SetCooperativeLevel）、プライマリーサーフェイスの生成（CreateSurface）が必要である。LCD の解像度は、プライマリサーフェイスの属性を得る関数 GetSize を用いる。

図１　LCD の解像度

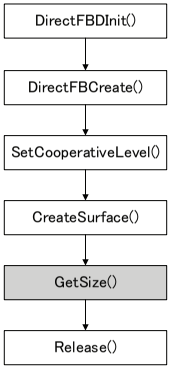


図２　解像度表示の流れ

１　準備１（ファイルのダウンロード）

1.1　ホスト OS（WindowsXP）を起動

1.2　「スタート」「ファイル名を指定して実行」を選択

1.3　「¥¥172.16.40.90」を入力し、「OK」をクリック

1.4　「public」をダブルクリック

1.5　フォルダ「manual」をホスト OS のデスクトップにコピー

1.6　ヘッダファイル「mydfb.h」、Make ファイル「Makefile」をホスト OS にコピー

２　準備２（作業用ディレクトリの作成）

2.1　ゲスト OS（Fedora Core 10）を起動

2.2　ゲスト OS に「guest」でログイン

2.3　ディレクトリ「lcd」を作成

2.4　ディレクトリ「lcd」に移る

2.5　「mydfb.h」「Makefile」をディレクトリ「lcd」にコピー

３　共通のファイル

3.1　ヘッダファイル「mydfb.h」

```
// DirectFB から呼び出されるエラーチェック用マクロ
#define DFBCHECK(x...)                          ¥
{                                               ¥
     DFBResult err = x;                         ¥
                                                ¥
     if (err != DFB_OK)                         ¥
     {                                          ¥
          fprintf( stderr, "%s <%d>:¥n¥t", __FILE__, __LINE__ );  ¥
          DirectFBErrorFatal( #x, err );                          ¥
     }                                          ¥
}                                               ¥
```

3.2　Make ファイル「Makefile」

```
CC = sh4-linux-gcc

export  SYSROOT=$(shell  readlink  -f  `$(CC)  -print-prog-name=gcc`  |  sed  -e
s!/usr/sh4-linux-uclibc/.*!!)

CFLAGS = -Wall `sh4-linux-directfb-config --cflags`
LDFLAGS = `sh4-linux-directfb-config --libs`

objs = getsize              <-  拡張子のないファイル名

all: $(objs)

taa:
        @echo $(SYSROOT)

clean:
        $(RM) $(objs)
```

4　LCD の解像度を調べる

4.1　解像度を調べるプログラムを作成

$ gedit getsize.c

```
#include <stdio.h>
#include <unistd.h>
#include <directfb.h>
#include  "mydfb.h "

// DirectFB の機能を実現するためのスーパサーフェイス
static IDirectFB *dfb = NULL;

// プライマリ・サーフェイス（フル画面のプライマリレイヤ）
static IDirectFBSurface *primary = NULL;

// 画面の解像度
static int screen_width  = 0;
static int screen_height = 0;

int main (int argc, char **argv)
{
     //サーフェイス
     DFBSurfaceDescription dsc;

     // DirectFB の初期化
     DFBCHECK (DirectFBInit (&argc, &argv));

     //スーパインターフェースの生成
     DFBCHECK (DirectFBCreate (&dfb));

     //スーパインターフェイスのフルスクリーン指定
     DFBCHECK (dfb->SetCooperativeLevel (dfb, DFSCL_FULLSCREEN));

     //サーフェイスの設定（プライマリサーフェイス）
     dsc.flags = DSDESC_CAPS;
     dsc.caps  = DSCAPS_PRIMARY | DSCAPS_FLIPPING;

     //プライマリサーフェイス生成
     DFBCHECK (dfb->CreateSurface( dfb, &dsc, &primary ));

     //解像度の取得
     DFBCHECK (primary->GetSize (primary, &screen_width, &screen_height));

     printf("screen_width = %d¥n", screen_width);
     printf("screen_height = %d¥n", screen_height);

     //リソースを解放
     primary->Release( primary );
     dfb->Release( dfb );

     return 23;
}
```

4.2　プログラムをコンパイル

$ **make**

4.3　実行ファイルをコピー

$ **cp -a getsize /nfs**

5　プログラムを実行

5.1　ターゲットボードにログイン

5.2　ゲスト OS の「/nfs」をターゲットボードの「/mnt/nfs」にマウントしていることを確認

5.3　実行

```
# ./getsize
~~~~~~~~~~~~~~~~~~~~~~~~~~~| DirectFB 1.2.8 |~~~~~~~~~~~~~~~~~~~~~~~~~~
     (c) 2001-2008  The world wide DirectFB Open Source Community
     (c) 2000-2004  Convergence (integrated media) GmbH
     (c) 2000-2004  Convergence (integrated media) GmbH

(*) DirectFB/Core: Single Application Core. (2012-07-20 18:55)
(*) Direct/Thread: Started 'VT Switcher' (972) [CRITICAL OTHER/OTHER 0/0] <20930
56>...
(*) Direct/Thread: Started 'Linux Input' (973) [INPUT OTHER/OTHER 0/0] <2093056>
...
(*) DirectFB/Input: TSC2007 Touchscreen 0.1 (directfb.org)
(*) DirectFB/Graphics: Generic Software Rasterizer 0.6 (directfb.org)
(*) DirectFB/Core/WM: Default 0.3 (directfb.org)
(*) FBDev/Surface: Allocated 480x272 16 bit RGB16 buffer (index 0) at offset 0
and pitch 960.
(*) FBDev/Surface: Allocated 480x272 16 bit RGB16 buffer (index 0) at offset 0
and pitch 960.
(*) FBDev/Mode: Setting 480x272 RGB16
(*) FBDev/Mode: Switched to 480x272 (virtual 480x544) at 16 bit (RGB16), pitch
960
(!) DirectFB/FBDev: Could not set gamma ramp    --> Invalid argument
(*) FBDev/Surface: Allocated 480x272 16 bit RGB16 buffer (index 1) at offset 261
120 and pitch 960.
screen_width = 480
screen_height = 272
(*) FBDev/Surface: Allocated 480x272 16 bit RGB16 buffer (index 0) at offset 0
and pitch 960.
#
```

2012/9/18

LCD プログラミング(その２)

― 直線、矩形を表示 ―

DirectFB は直線、矩形、三角形を表示できる。表示に用いる座標は、LCD の左上が(0,0)、右下が（479,271）である。直線を描画するためのプログラムの流れは、「LCD プログラミング（その１）」と同様に、DirectFB の初期化からプライマリサーフェイスの作成まで行い、直線を描画する関数（DrawLine）、矩形を描画する関数（DrawRectangle）などを用いて図形を表示する。図形の色は、関数 SetColor で指定できる。２節の「line.c」は、直線を(0,138)から（479,136）まで描くプログラムである。

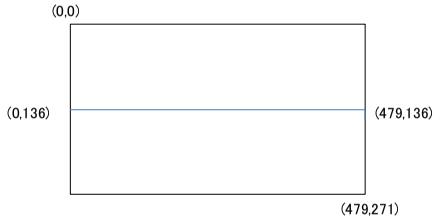


図１　座標と直線表示

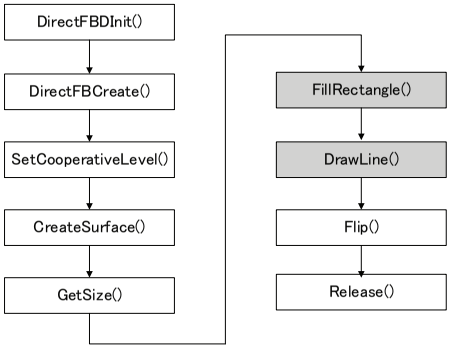


図２　直線、矩形表示の流れ

1　準備

1.1　ゲスト OS に「guest」でログイン

1.2　ディレクトリ「lcd」に移る

2　直線を表示

2.1　直線を表示するプログラムを作成

$ gedit line.c

```
#include <stdio.h>
#include <unistd.h>
#include <directfb.h>
#include "mydfb.h"

// DirectFB の機能を実現するためのスーパサーフェイス
static IDirectFB *dfb = NULL;

// プライマリ・サーフェイス（フル画面のプライマリレイヤ）
static IDirectFBSurface *primary = NULL;

// 画面の解像度
static int screen_width  = 0;
static int screen_height = 0;

int main (int argc, char **argv)
{
    //サーフェイス
    DFBSurfaceDescription dsc;

    // DirectFB の初期化
    DFBCHECK (DirectFBInit (&argc, &argv));

    //スーパインターフェースの生成
    DFBCHECK (DirectFBCreate (&dfb));

    //スーパインターフェイスのフルスクリーン指定
    DFBCHECK (dfb->SetCooperativeLevel (dfb, DFSCL_FULLSCREEN));

    //サーフェイスの設定（プライマリサーフェイス）
    dsc.flags = DSDESC_CAPS;
    dsc.caps  = DSCAPS_PRIMARY | DSCAPS_FLIPPING;

    //プライマリサーフェイス生成
    DFBCHECK (dfb->CreateSurface( dfb, &dsc, &primary ));

    //解像度の取得
    DFBCHECK (primary->GetSize (primary, &screen_width, &screen_height));

    //黒の矩形表示（バックカラー）
    DFBCHECK (primary->FillRectangle (primary, 0, 0, screen_width, screen_height));

    //色を変えて直線を表示する。
    DFBCHECK (primary->SetColor (primary, 0x80, 0x80, 0xff, 0xff));
    DFBCHECK (primary->DrawLine (primary, 0, screen_height / 2, screen_width - 1,
                                 screen_height / 2));

    //描画を反映
    DFBCHECK (primary->Flip (primary, NULL, 0));

    sleep (5);

    //リソースを解放
    primary->Release( primary );
    dfb->Release( dfb );
```

```
    return 23;
}
```

2.2　Make ファイルを変更

```
$ gedit  Makefile
```

```
        | | |

objs = line              <-  拡張子のないファイル名

        | | |
```

2.3　プログラムをコンパイル

```
$ make
```

2.4　実行ファイルをコピー

```
$ cp  -a  getsize  /nfs
```

3　プログラムを実行

3.1　ターゲットボードにログイン

3.2　ゲスト OS の「/nfs」をターゲットボードの「/mnt/nfs」にマウントしていることを確認

3.3　実行

```
# ./line
```

3.4　確認

LCD に直線を表示したことを確認

4　課題 1

図 3 のように矩形を 2 個表示するプログラムを作成しなさい。左側の矩形は赤の枠線で塗りつぶしなし、右側の矩形は緑で塗りつぶしなさい。

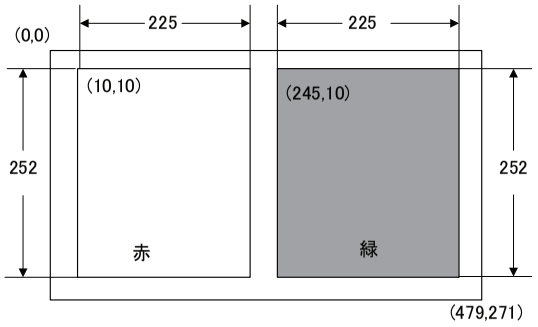


図 3　矩形の表示

4.1　サンプルプログラム

```
$ gedit  k01.c
```

```
#include <stdio.h>
#include <unistd.h>
#include <directfb.h>
#include "mydfb.h"

// DirectFB の機能を実現するためのスーパサーフェイス
static IDirectFB *dfb = NULL;

// プライマリ・サーフェイス（フル画面のプライマリレイヤ）
static IDirectFBSurface *primary = NULL;

int main (int argc, char **argv)
{
 //サーフェイス
  DFBSurfaceDescription dsc;

 // DirectFB の初期化
  DFBCHECK (DirectFBInit (&argc, &argv));

 //スーパインターフェースの生成
  DFBCHECK (DirectFBCreate (&dfb));

  //スーパインターフェイスのフルスクリーン指定
  DFBCHECK (dfb->SetCooperativeLevel (dfb, DFSCL_FULLSCREEN));

  //サーフェイスの設定（プライマリサーフェイス）
  dsc.flags = DSDESC_CAPS;
  dsc.caps  = DSCAPS_PRIMARY | DSCAPS_FLIPPING;

  //プライマリサーフェイス生成
  DFBCHECK (dfb->CreateSurface( dfb, &dsc, &primary ));

 //赤の矩形を表示
  DFBCHECK (primary->SetColor (primary, 0xff, 0x00, 0x00, 0xff));
  DFBCHECK (primary->DrawRectangle (primary, 10, 10, 225, 252));

 //緑に塗りつぶした矩形を表示
  DFBCHECK (primary->SetColor (primary, 0x00, 0xff, 0x00, 0xff));
  DFBCHECK (primary->FillRectangle (primary, 245, 10, 225, 252));

 //描画を反映
  DFBCHECK (primary->Flip (primary, NULL, 0));

  sleep (10);

 //リソースを解放
  primary->Release( primary );
  dfb->Release( dfb );

  return 23;
}
```

2012/9/18

LCD プログラミング(その３)

― 画像表示 ―

DirectFB は、JPEG、GIF、PNG 画像を表示できる。画像の表示は、「LCD プログラミング（その２）」と同様に、DirectFB の初期化から解像度の取得まで行い、イメージプロバイダを作成（Create ImageProvider）、画像ファイルの属性取得（GetSurfaceDescription）、スケーリング、ファイル変換（RendTo）により画像を表示する。また、Flip 関数を用いるとちらつきを防止できる。

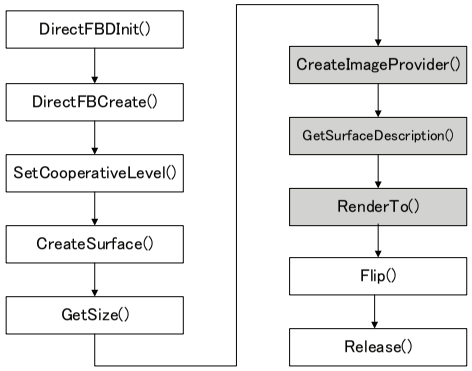


図１　画像表示の流れ

１　準備

1.1　ゲスト OS に「guest」でログイン

1.2　ディレクトリ「lcd」に移る

1.3　JPEG 画像ファイル「penguin.jpg」を得る

２　画像を表示

2.1　画像を表示するプログラムを作成

```
$ gedit  image.c
```

```
#include <stdio.h>
#include <unistd.h>
#include <directfb.h>
#include "mydfb.h"

// DirectFB の機能を実現するためのスーパサーフェイス
static IDirectFB *dfb = NULL;

// プライマリ・サーフェイス（フル画面のプライマリレイヤ）
static IDirectFBSurface *primary = NULL;

// 画面の解像度
static int screen_width  = 0;
static int screen_height = 0;

Int  main (int argc, char **argv)
{
    //サーフェイス
    DFBSurfaceDescription dsc;
    DFBRectangle r;

     //画像読み込み用イメージプロバイダ.
    IDirectFBImageProvider *provider;

    // DirectFB の初期化
    DFBCHECK (DirectFBInit (&argc, &argv));

    //スーパインターフェイスの生成
    DFBCHECK (DirectFBCreate (&dfb));

    //スーパインターフェイスのフルスクリーン指定
    DFBCHECK (dfb->SetCooperativeLevel (dfb, DFSCL_FULLSCREEN));

    //サーフェイスの設定（プライマリサーフェイス）
    dsc.flags = DSDESC_CAPS;
    dsc.caps  = DSCAPS_PRIMARY | DSCAPS_FLIPPING;

    //プライマリサーフェイス生成
    DFBCHECK (dfb->CreateSurface( dfb, &dsc, &primary ));

    //解像度の取得
    DFBCHECK (primary->GetSize (primary, &screen_width, &screen_height));

    //画像プロバイダを生成
    DFBCHECK (dfb->CreateImageProvider (dfb, "penguin.jpg", &provider));

    //画像プロバイダの設定値取得（画像ファイル情報、解像度、ピクセル当たりのビット数等）
    DFBCHECK (provider->GetSurfaceDescription (provider, &dsc));
    printf("dsc.width = %d¥n", dsc.width);
    printf("dsc.height = %d¥n", dsc.height);

    //画像の表示位置指定
    r.x = 0;
    r.y = 0;
    r.w = screen_width;
    r.h = screen_height;

    //
    DFBCHECK (provider->RenderTo (provider, primary, &r));

    //
    DFBCHECK (primary->Flip (primary, NULL, DSFLIP_WAITFORSYNC));

    sleep(10);

    //リソースを解放
    provider->Release (provider);
    primary->Release (primary);
    dfb->Release (dfb);

    return 23;
}
```

2.2 Makeファイルを変更

```
$ gedit  Makefile
```

```
        | | |

objs = image              <-  拡張子のないファイル名

        | | |
```

2.3 プログラムをコンパイル

```
$ make
```

2.4 実行ファイルをコピー

```
$ cp  -a  image  /nfs
```

2.5 画像「penguin.jpg」を「/nfs」にコピー

3 プログラムを実行

3.1 ターゲットボードにログイン

3.2 ゲストOSの「/nfs」をターゲットボードの「/mnt/nfs」にマウントしていることを確認

3.3 実行

```
# ./image
~~~~~~~~~~~~~~~~~~~~~~~~~| DirectFB 1.2.8 |~~~~~~~~~~~~~~~~~~~~~~~~~
(*) FBDev/Surface
        (c) 2001-2008  The world wide DirectFB Open Source Community
        (c) 2000-2004  Convergence (integrated media) GmbHce: Allocated 480x272 16 bit
         RGB16 buffer (index 0) at off
--------------------------------------------------------------------
056>...
(*) Direct/Thread: Started 'Linux Input' (1115) [INPUT OTHER/OTHER 0/0] <2093056
>...
(*) DirectFB/Input: TSC2007 Touchscreen 0.1 (directfb.org)
(*) DirectFB/Graphics: Generic Software Rasterizer 0.6 (directfb.org)
(*) DirectFB/Core/WM: Default 0.3 (directfb.org)
(*) FBDev/Surface: Allocated 480x272 16 bit RGB16 buffer (index 0) at offset 0 a
nd pitch 960.
(*) FBDev/Surface: Allocated 480x272 16 bit RGB16 buffer (index 0) at offset 0 a
nd pitch 960.
(*) FBDev/Mode: Setting 480x272 RGB16
(*) FBDev/Mode: Switched to 480x272 (virtual 480x544) at 16 bit (RGB16), pitch 9
60
(!) DirectFB/FBDev: Could not set gamma ramp    --> Invalid argument
(*) FBDev/Surface: Allocated 480x272 16 bit RGB16 buffer (index 1) at offset 261
120 and pitch 960.
(*) Direct/Interface: Loaded 'JPEG' implementation of 'IDirectFBImageProvider'.
dsc.width = 480
dsc.height = 272

(*) FBDev/Surface: Allocated 480x272 16 bit RGB16 buffer (index 0) at offset 0 a
nd pitch 960.
#
```

3.4 確認

LCDに画像を表示したことを確認

4 課題

4.1 JPEG画像「penguin.jpg」を縮小して、LCDに4枚表示しなさい。画像の縮小は表示の大きさを指定すると自動的に行われる。

図2　複数の画像表示

4.2 GIF画像「london.gif」を縦横ともに1/4縮小して、LCDに表示しなさい。画像の表示位置は、画像の左上を（80,8）としなさい。

図3　GIF画像

５．プログラム例

5.1　JPEG 画像を４枚表示するプログラム

gedit　k02.c

```
#include <stdio.h>
#include <unistd.h>
#include <directfb.h>
#include "mydfb.h"

// DirectFB の機能を実現するためのスーパサーフェイス（最上位のインターフェース）
 static IDirectFB *dfb = NULL;

// プライマリ・サーフェイス（フル画面のプライマリレイヤ）
 static IDirectFBSurface *primary = NULL;

// 画面の解像度
static int screen_width  = 0;
static int screen_height = 0;

//画像表示用サーフェイス（ロゴ画像）

int main (int argc, char **argv)
{
 //サーフェイス
  DFBSurfaceDescription dsc;
  DFBRectangle r;

  //画像読み込み用イメージプロバイダ.
  IDirectFBImageProvider *provider;

 // DirectFB の初期化
  DFBCHECK (DirectFBInit (&argc, &argv));

 //スーパインターフェイスの生成
  DFBCHECK (DirectFBCreate (&dfb));

  //スーパインターフェイスのフルスクリーン指定
  DFBCHECK (dfb->SetCooperativeLevel (dfb, DFSCL_FULLSCREEN));

  //サーフェイスの設定（プライマリサーフェイス）
  dsc.flags = DSDESC_CAPS;
  dsc.caps  = DSCAPS_PRIMARY | DSCAPS_FLIPPING;

  //プライマリサーフェイス生成
  DFBCHECK (dfb->CreateSurface( dfb, &dsc, &primary ));

  //解像度の取得
   DFBCHECK (primary->GetSize (primary, &screen_width, &screen_height));

  //画像プロバイダを生成
  DFBCHECK (dfb->CreateImageProvider (dfb, "penguin.jpg", &provider));

  //画像プロバイダの設定値取得（画像ファイル情報、解像度、ピクセル当たりのビット数等）
  DFBCHECK (provider->GetSurfaceDescription (provider, &dsc));
  printf("dsc.width = %d¥n", dsc.width);
  printf("dsc.height = %d¥n", dsc.height);

  r.x = 0;
  r.y = 0;
  r.w = screen_width/2;
  r.h = screen_height/2;
  DFBCHECK (provider->RenderTo (provider, primary, &r));
  r.x = screen_width/2;
  r.y = 0;
  DFBCHECK (provider->RenderTo (provider, primary, &r));
  r.x = 0;
  r.y = screen_height/2;;
  DFBCHECK (provider->RenderTo (provider, primary, &r));
  r.x = screen_width/2;
  r.y = screen_height/2;;
  DFBCHECK (provider->RenderTo (provider, primary, &r));

  DFBCHECK (primary->Flip (primary, NULL, DSFLIP_WAITFORSYNC));

  sleep(10);

  //リソースを解放
  provider->Release (provider);
  primary->Release (primary);
  dfb->Release (dfb);

  return 23;
}
```

5.2　GIF 画像を表示するプログラム

gedit　k03.c

```
#include <stdio.h>
#include <unistd.h>
#include <directfb.h>
#include "mydfb.h"

// DirectFB の機能を実現するためのスーパサーフェイス（最上位のインターフェース）
 static IDirectFB *dfb = NULL;

// プライマリ・サーフェイス（フル画面のプライマリレイヤ）
 static IDirectFBSurface *primary = NULL;

// 画面の解像度
static int screen_width  = 0;
static int screen_height = 0;

int main (int argc, char **argv)
{
 //サーフェイス
  DFBSurfaceDescription dsc;
  DFBRectangle r;

  //画像読み込み用イメージプロバイダ.
  IDirectFBImageProvider *provider;

 // DirectFB の初期化
  DFBCHECK (DirectFBInit (&argc, &argv));

 //スーパインターフェイスの生成
  DFBCHECK (DirectFBCreate (&dfb));

  //スーパインターフェイスのフルスクリーン指定
  DFBCHECK (dfb->SetCooperativeLevel (dfb, DFSCL_FULLSCREEN));
```

```
    //サーフェイスの設定（プライマリサーフェイス）
    dsc.flags = DSDESC_CAPS;
    dsc.caps  = DSCAPS_PRIMARY | DSCAPS_FLIPPING;

    //プライマリサーフェイス生成
    DFBCHECK (dfb->CreateSurface( dfb, &dsc, &primary ));

    //解像度の取得
    DFBCHECK (primary->GetSize (primary, &screen_width, &screen_height));

    //画像プロバイダを生成
    DFBCHECK (dfb->CreateImageProvider (dfb, "london.gif", &provider));

    //画像プロバイダの設定値取得（画像ファイル情報、解像度、ピクセル当たりのビット数等）
    DFBCHECK (provider->GetSurfaceDescription (provider, &dsc));
    printf("dsc.width = %d¥n", dsc.width);
    printf("dsc.height = %d¥n", dsc.height);

    r.x = 80;
    r.y = 8;
    r.w = dsc.width/4;
    r.h = dsc.height/4;
    DFBCHECK (provider->RenderTo (provider, primary, &r));

    DFBCHECK (primary->Flip (primary, NULL, DSFLIP_WAITFORSYNC));

    sleep(10);

    //リソースを解放
    provider->Release (provider);
    primary->Release (primary);
    dfb->Release (dfb);

    return 23;
}
```

2012/9/25

LCD プログラミング(その４)

― 文字列表示 ―

DirectFB を用いて、LCD に文字を表示する。文字の表示は、フォントデータを CreateFont 関数で生成、フォントを SetFont 関数でセットし、DrawString 関数で文字を表示する。CreateFont 関数では、文字の大きさなどが指定できる。文字列の表示位置は、画面で指定した座標（x,,y）に基準位置が合うように表示する。

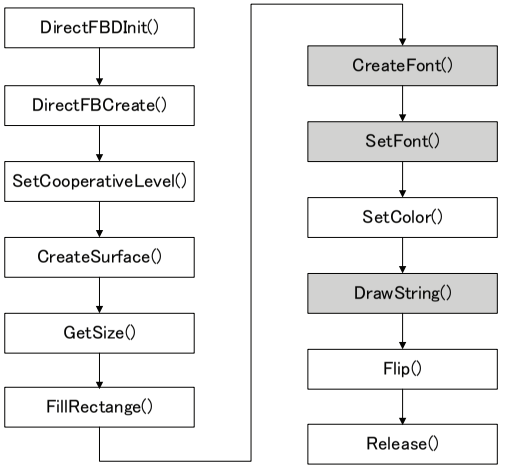


図１　文字表示の流れ

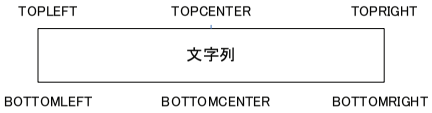


図２　文字列の基準位置

1　準備

1.1　ゲスト OS に「guest」でログイン

1.2　ディレクトリ「lcd」に移る

1.3　フォントファイル「decker.tff」を得る

2　画像を表示

2.1　画像を表示するプログラムを作成

$ gedit　text.c

```
#include <stdio.h>
#include <unistd.h>
#include <directfb.h>
#include "mydfb.h"

static IDirectFB *dfb=NULL;
static IDirectFBSurface *primary=NULL;
static IDirectFBFont *font=NULL;

static int screen_width=0;
static int screen_height=0;

int main(int argc,char** argv)
{
    DFBSurfaceDescription dsc;
    DFBFontDescription font_dsc;

    /*DirectFB の初期化*/
    DFBCHECK(DirectFBInit(&argc,&argv));

    /*スーパインターフェースの取得*/
    DFBCHECK(DirectFBCreate(&dfb));

    //スーパインターフェイスのフルスクリーン指定
    DFBCHECK(dfb->SetCooperativeLevel(dfb,DFSCL_FULLSCREEN));

    /*画面設定*/
    dsc.flags=DSDESC_CAPS;
    dsc.caps=DSCAPS_PRIMARY;

    /*描画画面の取得*/
    DFBCHECK(dfb->CreateSurface(dfb,&dsc,&primary));

    /*画面サイズの取得*/
    DFBCHECK(primary->GetSize(primary,&screen_width,&screen_height));

    /*バックグランド色での塗潰し*/
    DFBCHECK(primary->FillRectangle(primary,0,0,screen_width,screen_height));

    /*フォントサイズの指定*/
    font_dsc.flags=DFDESC_HEIGHT;
    font_dsc.height=48;

    /*フォントデータの作成*/
    DFBCHECK(dfb->CreateFont(dfb,"./decker.ttf",&font_dsc,&font));

    /*フォント設定*/
    DFBCHECK(primary->SetFont(primary,font));

    /*フォント色の指定：赤*/
    DFBCHECK(primary->SetColor(primary,255,0,0,0xFF));

    /*文字描画*/
    DFBCHECK(primary->DrawString(primary,"DirectFB",-1,240,136,DSTF_BOTTOMCENTER));

    /*画面表示（ちらつき防止）の実行*/
    DFBCHECK(primary->Flip(primary,NULL,0));

    getchar();

    /*フォントの破棄*/
    DFBCHECK(font->Release(font));

    /*描画画面の破棄*/
    DFBCHECK(primary->Release(primary));

    /*メインインターフェースの破棄*/
    DFBCHECK(dfb->Release(dfb));

    return 0;
}
```

2.2　Make ファイルを変更

$ gedit　Makefile

```
        | | |

objs = text              <-  拡張子のないファイル名

        | | |
```

2.3　プログラムをコンパイル

$ **make**

2.4　実行ファイルをコピー

$ **cp　-a　text　/nfs**

3　プログラムを実行

3.1　ターゲットボードにログイン

3.2　ゲスト OS の「/nfs」をターゲットボードの「/mnt/nfs」にマウントしていることを確認

3.3　実行

 # ./text

3.4　確認

LCD に文字列「DirectFB」を表示したことを確認

4　課題

文字列“Name”を TOPLEFT=(200,100)の位置に表示し、キーボードから自分の名前“myname“を入力すると TOPLEFT=(200,150)の位置に名前を表示するプログラムを作成しなさい。フォントの大きさは高さ 32 ピクセル、緑色とする。

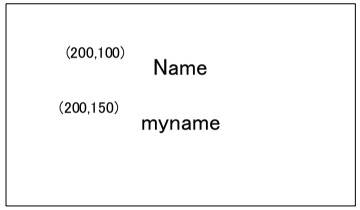


図３　入力文字の表示

５．プログラム例

5.1　キーボードからの文字を表示列するプログラム

$ gedit k04.c

```
#include <stdio.h>
#include <unistd.h>
#include <directfb.h>
#include "mydfb.h"

static IDirectFB *dfb=NULL;
static IDirectFBSurface *primary=NULL;
static IDirectFBFont *font=NULL;

static int screen_width=0;
static int screen_height=0;

int main(int argc,char** argv)
{
    DFBSurfaceDescription dsc;
    DFBFontDescription font_dsc;
    char buf[128];
    int i;
    char c;

    /*DirectFB の初期化*/
    DFBCHECK(DirectFBInit(&argc,&argv));

    /*スーパインターフェースの取得*/
    DFBCHECK(DirectFBCreate(&dfb));

    //スーパインターフェイスのフルスクリーン指定
    DFBCHECK(dfb->SetCooperativeLevel(dfb,DFSCL_FULLSCREEN));

    /*画面設定*/
    dsc.flags=DSDESC_CAPS;
    dsc.caps=DSCAPS_PRIMARY;

    /*描画画面の取得*/
    DFBCHECK(dfb->CreateSurface(dfb,&dsc,&primary));

    /*画面サイズの取得*/
    DFBCHECK(primary->GetSize(primary,&screen_width,&screen_height));

    /*バックグランド色での塗潰し*/
    DFBCHECK(primary->FillRectangle(primary,0,0,screen_width,screen_height));

    /*フォントサイズの指定*/
    font_dsc.flags=DFDESC_HEIGHT;
    font_dsc.height=32;

    /*フォントデータの作成*/
    DFBCHECK(dfb->CreateFont(dfb,"./decker.ttf",&font_dsc,&font));

    /*フォント設定*/
    DFBCHECK(primary->SetFont(primary,font));

    /*フォント色の指定：青*/
    DFBCHECK(primary->SetColor(primary,0,0,255,0xFF));

    /*文字描画*/
```

```
    DFBCHECK(primary->DrawString(primary,"Name",-1,200,100,DSTF_TOPLEFT));

    /*画面表示（ちらつき防止）の実行*/
//    DFBCHECK(primary->Flip(primary,NULL,0));

     c = getchar();
     for (i = 0; c != '¥n' ; i++) {
        buf[i] = c;
        c = getchar();
     };
     buf[i] = '¥0';

     DFBCHECK(primary->DrawString(primary, buf,-1,200,150,DSTF_TOPLEFT));

     getchar();

     /*フォントの破棄*/
     DFBCHECK(font->Release(font));

     /*描画画面の破棄*/
     DFBCHECK(primary->Release(primary));

     /*メインインターフェースの破棄*/
     DFBCHECK(dfb->Release(dfb));

     return 23;
}
```

注意）
dsc.caps=DSCAPS_PRIMARY**|DSCAPS_FLIPPING;**
を指定すると、一瞬しか表示しない。

2012/9/25

LCD プログラミング(その５)

― 重ね表示 ―

DirectFB においてサーフェイスを２枚用いて、画像を重ねて表示する。サーフェイスはプライマリサーフェイス（Primary Surface）、セカンダリサーフェイス（Secondary Surface）の２枚を生成する。プライマリサーフェイスにはペンギンの画像を表示、セカンダリサーフェイスには文字列“Penguin”を表示する。描画の順序は最初にプライマリサーフェイスの画像を RendTo 関数で、次にセカンダリサーフェイスの文字列を DrawString 関数で描く。この順序で描画すると、ペンギンの画像の上に文字列“Penguin”を表示する。もし、逆の順序で描画すると文字列はペンギンの画像の下となり、ペンギンの画像のみ表示する。

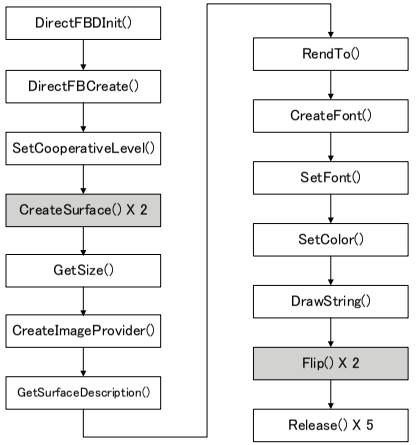


図１　重ね表示の流れ

１　準備
1.1　ゲスト OS に「guest」でログイン
1.2　ディレクトリ「lcd」に移る
1.3　画像「penguin.jpg」を「/nfs」にコピー（その３で実施済み）

２　画像を表示
2.1　画像を表示するプログラムを作成

```
$ gedit  surface.c
```

```
#include <stdio.h>
#include <unistd.h>
#include <directfb.h>
#include "mydfb.h"
```

```
static IDirectFB *dfb=NULL;
static IDirectFBSurface *primary=NULL;
static IDirectFBSurface *secondary=NULL;
static IDirectFBFont *font=NULL;

static int screen_width=0;
static int screen_height=0;

int main(int argc,char** argv)
{
    DFBSurfaceDescription dsc;
    DFBFontDescription font_dsc;
    DFBRectangle r;

    //画像読み込み用イメージプロバイダ.
    IDirectFBImageProvider *provider;

    //DirectFB の初期化
    DFBCHECK(DirectFBInit(&argc,&argv));

    //スーパインターフェースの取得
    DFBCHECK(DirectFBCreate(&dfb));

    //スーパインターフェイスのフルスクリーン指定
    DFBCHECK(dfb->SetCooperativeLevel(dfb,DFSCL_FULLSCREEN));

    //画面設定
    dsc.flags=DSDESC_CAPS;
    dsc.caps  = DSCAPS_PRIMARY;

    //描画画面の取得
    DFBCHECK(dfb->CreateSurface(dfb,&dsc,&primary));
    DFBCHECK(dfb->CreateSurface(dfb,&dsc,&secondary));

    //画面サイズの取得
    DFBCHECK(primary->GetSize(primary,&screen_width,&screen_height));

    //画像プロバイダを生成
    DFBCHECK (dfb->CreateImageProvider (dfb, "penguin.jpg", &provider));

    //画像プロバイダの設定値取得（画像ファイル情報、解像度、ピクセル当たりのビット数等）
    DFBCHECK (provider->GetSurfaceDescription (provider, &dsc));
    printf("dsc.width = %d¥n", dsc.width);
    printf("dsc.height = %d¥n", dsc.height);

    //画像の表示位置指定
    r.x = 0;
    r.y = 0;
    r.w = screen_width;
    r.h = screen_height;
    DFBCHECK (provider->RenderTo (provider, primary, &r));

    //フォントサイズの指定
    font_dsc.flags=DFDESC_HEIGHT;
    font_dsc.height=48;

    //フォントデータの作成
    DFBCHECK(dfb->CreateFont(dfb,"./decker.ttf",&font_dsc,&font));

    //フォント設定
    DFBCHECK(secondary->SetFont(secondary,font));

    //フォント色の指定：赤
    DFBCHECK(secondary->SetColor(secondary,255,0,0,0xff));

    //文字描画
    DFBCHECK(secondary->DrawString(secondary,"Penguin",-1,150,100,DSTF_TOPLEFT));

    //画面表示（ちらつき防止）の実行
    DFBCHECK(primary->Flip(primary,NULL,0));
    DFBCHECK(secondary->Flip(secondary,NULL,0));

    getchar();

    //破棄
    DFBCHECK(font->Release(font));
    DFBCHECK(provider->Release(provider));
    DFBCHECK(primary->Release(primary));
    DFBCHECK(secondary->Release(secondary));
    DFBCHECK(dfb->Release(dfb));

    return 23;
}
```

2.2 Make ファイルを変更

```
$ gedit  Makefile
```

```
        | | |

objs = surface            <-  拡張子のないファイル名

        | | |
```

2.3 プログラムをコンパイル

```
$ make
```

2.4 実行ファイルをコピー

```
$ cp  -a  surface  /nfs
```

3 プログラムを実行

3.1 ターゲットボードにログイン

3.2 ゲスト OS の「/nfs」をターゲットボードの「/mnt/nfs」にマウントしていることを確認

3.3 実行

```
# ./surface
```

3.4 確認

LCD に画像と文字列が重ねて表示したことを確認

2012/9/25

LCD プログラミング(その６)

― 透過的な重ね表示 ―

DirectFB においてアルファブレンディング（Alpha Blending）を用い、透過的に画像を重ねて表示する。透過度はアルファ値 a で表し、0≦a≦1 の値をもつ。もとのピクセルの RGB が(r, g, b)である場合、a を適用したピクセル値は(r*a, g*a, b*a)である。a を 8 ビットの整数で表した場合、a=0 が a=0x00、a＝１が a=0xff である。図１のように２個の矩形を透過的に表示すると、矩形が重なった部分は透過的に合成される。画像を透過的に合成するためには、サーフェイスを生成する CreateSurface 関数で DSCAPS_PREMULTIPLIED、描画の合成を SetDrawingFlags 関数で DSDRAW_BLEND を指定すればよい。

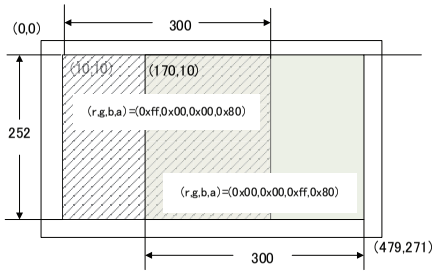


図１　透過的な重ね表示

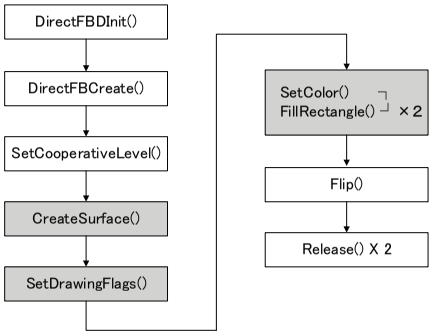


図２　透過的な重ね表示の流れ

１　準備

1.1　ゲスト OS に「guest」でログイン

1.2　ディレクトリ「lcd」に移る

２　画像を表示

2.1　画像を表示するプログラムを作成

$ gedit blend.c

```
#include <stdio.h>
#include <unistd.h>
#include <directfb.h>
#include "mydfb.h"

// DirectFB の機能を実現するためのスーパサーフェイス（最上位のインターフェース）
static IDirectFB *dfb = NULL;

// プライマリ・サーフェイス（フル画面のプライマリレイヤ）
static IDirectFBSurface *primary = NULL;

int main (int argc, char **argv)
{
    //サーフェイス
    DFBSurfaceDescription dsc;

    // DirectFB の初期化
    DFBCHECK (DirectFBInit (&argc, &argv));

    //スーパインターフェースの生成
    DFBCHECK (DirectFBCreate (&dfb));

    //スーパインターフェイスのフルスクリーン指定
    DFBCHECK (dfb->SetCooperativeLevel (dfb, DFSCL_FULLSCREEN));

    //サーフェイスの設定（プライマリサーフェイス）
    dsc.flags = DSDESC_CAPS;
    dsc.caps=DSCAPS_PRIMARY | DSCAPS_PREMULTIPLIED;

    //プライマリサーフェイス生成
    DFBCHECK (dfb->CreateSurface( dfb, &dsc, &primary ));

    //描画の合成を指示（alpha 値を使用）
    DFBCHECK(primary->SetDrawingFlags(primary,DSDRAW_BLEND));

    //赤の矩形を表示
    DFBCHECK (primary->SetColor (primary, 0xff, 0x00, 0x00, 0x80));
    DFBCHECK (primary->FillRectangle (primary, 10, 10, 300, 252));

    //青に塗りつぶした矩形を表示
    DFBCHECK (primary->SetColor (primary, 0x00, 0x00, 0xff, 0x80));
    DFBCHECK (primary->FillRectangle (primary, 170, 10, 300, 252));

    //描画を反映（ちらつき防止）
    DFBCHECK (primary->Flip (primary, NULL, 0));

    sleep (10);

    //リソースを解放
    primary->Release( primary );
    dfb->Release( dfb );

    return 23;
}
```

2.2　Make ファイルを変更

$ gedit Makefile

```
        | | |

objs = blend              <-  拡張子のないファイル名

        | | |
```

2.3　プログラムをコンパイル

$ make

2.4　実行ファイルをコピー

$ cp -a blend /nfs

3　プログラムを実行

3.1　ターゲットボードにログイン

3.2　ゲスト OS の「/nfs」をターゲットボードの「/mnt/nfs」にマウントしていることを確認

3.3　実行

 # ./blend

3.4　確認

LCD に矩形が重なった部分が合成されていることを確認

4　課題

サーフェイスを２枚(primay, secondary)使用して、ペンギンの画像を primary サーフェイスに、文字列 “Penguin” と塗りつぶしの矩形を secondary サーフェイスに表示しなさい。ただし、文字列は LEFTTOP を（150, 50)、フォントの高さを 48、(r, g, b, a) ＝（0xff, 0x00, 0x00, 0x80)、矩形は画面の下半分を（r, g, b, a) ＝（0x00, 0xff, 0x00, 0x40）で描画しなさい。

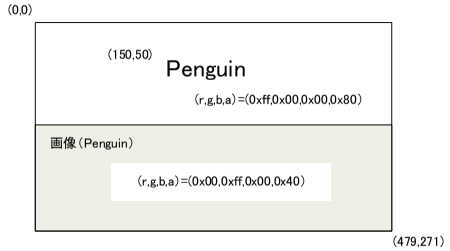


図３　画像の合成

4.1　サンプルプログラム

$ gedit blend_2.c

```
// DirectFB の機能を実現するためのスーパサーフェイス（最上位のインターフェース）
static IDirectFB *dfb = NULL;

// プライマリ・サーフェイス（フル画面のプライマリレイヤ）
static IDirectFBSurface *primary = NULL;
static IDirectFBSurface *secondary = NULL;
static IDirectFBFont *font=NULL;

// 画面の解像度
static int screen_width  = 0;
static int screen_height = 0;

int main (int argc, char **argv)
{
    DFBSurfaceDescription dsc;
    DFBFontDescription font_dsc;
    DFBRectangle r;
    IDirectFBImageProvider *provider;

    // DirectFB の初期化
    DFBCHECK (DirectFBInit (&argc, &argv));

    //スーパインターフェースの生成
    DFBCHECK (DirectFBCreate (&dfb));

    //スーパインターフェイスのフルスクリーン指定
    DFBCHECK (dfb->SetCooperativeLevel (dfb, DFSCL_FULLSCREEN));

    //サーフェイスの設定（プライマリサーフェイス）
    dsc.flags = DSDESC_CAPS;
    dsc.caps=DSCAPS_PRIMARY | DSCAPS_PREMULTIPLIED;

    //サーフェイス生成
    DFBCHECK (dfb->CreateSurface( dfb, &dsc, &primary ));
    DFBCHECK (dfb->CreateSurface( dfb, &dsc, &secondary ));

    DFBCHECK (primary->GetSize (primary, &screen_width, &screen_height));

    DFBCHECK(primary->SetDrawingFlags(primary,DSDRAW_BLEND));
    DFBCHECK(primary->SetDrawingFlags(secondary,DSDRAW_BLEND));

    DFBCHECK (dfb->CreateImageProvider (dfb, "penguin.jpg", &provider));

    //画像の表示位置指定
    r.x = 0;
    r.y = 0;
    r.w = screen_width;
    r.h = screen_height;
    DFBCHECK (provider->RenderTo (provider, primary, &r));

    //フォントサイズの指定
    font_dsc.flags=DFDESC_HEIGHT;
    font_dsc.height=48;

    //フォントデータの作成
    DFBCHECK(dfb->CreateFont(dfb,"./decker.ttf",&font_dsc,&font));

    //フォント設定
```

```
    DFBCHECK(secondary->SetFont(secondary,font));

    //フォント色の指定：赤
    DFBCHECK(secondary->SetColor(secondary,255,0,0,0x80));

    //文字描画
    DFBCHECK(secondary->DrawString(secondary,"Penguin",-1,150,50,DSTF_TOPLEFT));

    //緑に塗りつぶした矩形を表示
    DFBCHECK (primary->SetColor (secondary, 0x00, 0xff, 0x00, 0x40));
    DFBCHECK (primary->FillRectangle (secondary, 0, screen_height/2,
                                screen_width, screen_height/2))

    //描画を反映（ちらつき防止）
    DFBCHECK (primary->Flip (primary, NULL, 0));
    DFBCHECK (primary->Flip (secondary, NULL, 0));

    sleep (10);

    //リソースを解放
    primary->Release( primary );
    dfb->Release( dfb );

    return 23;
}
```

2012/10/2

JPEG(その１)

― RGB を JPEG に変換 ―

JPEG（Joint Photographic Experts Group）は、静止画像をデータ圧縮する方法、または組織の略称である。JPEG は高い圧縮率を得られるが、非可逆圧縮なので圧縮したファイルを伸張したときに完全に元の画像にはならない。

ライブラリ libjpeg を用いて、RGB データを JPEG ファイルに変換する。libjpeg の内部で用いる画像のデータ構造は、図１のとおりである。1 ピクセル（pixel）は 1 バイトごとの R、G、B で構成され３バイトである。画面全体のデータサイズは画面の幅（width）×高さ（height）×3 バイトである。このデータ構造を扱うために JSAMPLE、JSAMPROW、JSAMPARRAY の３つのデータ型が定義されている。JSAMPLE は unsigned char であり、１ピクセルの各 RGB 要素を表すデータ型である。JSAMPROW は画面の一行を表す配列へのポインタである。JSAMPARRAY は各行へのポインタ配列を指すポインタである。

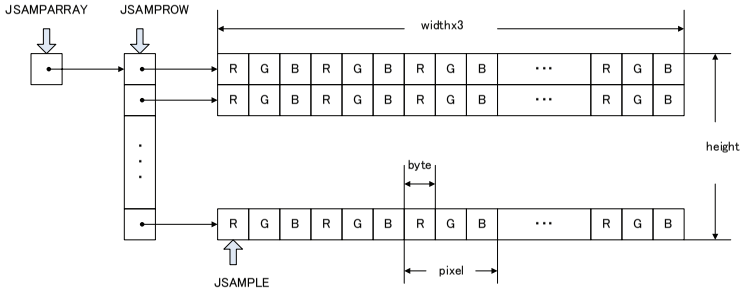


図１　画像のデータ構造

libjpeg を用いた圧縮操作の流れは以下のとおりである。

(1)　JPEG のオブジェクトの領域確保と初期化（jpeg_std_error, jpeg_create_compress）
(2)　圧縮されたデータの出力先を指定（jpeg_stdio_dest）
(3)　圧縮パラメータの設定（jpeg_set_defaults, jpeg_set _quality）
(4)　圧縮の開始（jpeg_start_compress）
(5)　圧縮データの書き込み（jpeg_write_scanlines）
(6)　圧縮の終了（jpeg_finish_compress）
(7)　オブジェクトの解放（jpeg_destroy_compress）

1　サンプル画像を作成

1.1　直線を表示するプログラムを作成

```
$ gedit  jpeg_samp.c
```

```
#include <stdio.h>
#include <stdlib.h>
#include <jpeglib.h>

int main ()
{
    // JPEG オブジェクト，エラーハンドラを確保
    struct jpeg_compress_struct cinfo;
    struct jpeg_error_mgr jerr;
    // 変数の宣言
    int  i, j;
    // 出力ファイルのハンドラを確保、出力ファイル名を指定
    FILE *fp;
    char *filename = "output.jpg";
    // 画像のサイズ
    int width = 256;
    int height = 256;

    // エラーハンドラにデフォルト値を設定
    cinfo.err = jpeg_std_error(&jerr);

    // JPEG オブジェクトを初期化
    jpeg_create_compress(&cinfo);

    // 出力ファイルをオープン
    if ((fp = fopen(filename, "wb"))== NULL) {
        fprintf(stderr, "cannot open %s¥n", filename);
        exit(EXIT_FAILURE);
    }
    // 出力ファイルを指定
    jpeg_stdio_dest(&cinfo, fp);

    // 画像のパラメータを設定
    cinfo.image_width = width;              // 画像の幅
    cinfo.image_height = height;            // 画像の高さ
    cinfo.input_components = 3;             // 1pixel あたりのカラー要素数
    cinfo.in_color_space = JCS_RGB;         // 入力画像の形式
    jpeg_set_defaults(&cinfo);
    jpeg_set_quality(&cinfo, 75, TRUE);     // 圧縮効率（0～100）

    // 圧縮を開始
    jpeg_start_compress(&cinfo, TRUE);

    // データ（RGB 値）を生成
    JSAMPARRAY img = (JSAMPARRAY) malloc(sizeof(JSAMPROW) * height);
    for (i = 0; i < height; i++) {
        img[i] = (JSAMPROW) malloc(sizeof(JSAMPLE) * 3 * width);
        for (j = 0; j < width; j++) {
            img[i][j*3 + 0] = i;
            img[i][j*3 + 1] = j;
            img[i][j*3 + 2] = (i+j)/2;
        }
    }
    // １行でなく画像全体を出力
    jpeg_write_scanlines(&cinfo, img, height);
```

```
    // 圧縮を終了
    jpeg_finish_compress(&cinfo);

    // JPEGオブジェクトを破棄
    jpeg_destroy_compress(&cinfo);

    for (i = 0; i < height; i++) {
        free(img[i]);
    }
    free(img);
    fclose(fp);
}
```

1.2　プログラムをコンパイル

```
$ gcc  jpeg_samp.c  -ljpeg
```

（l はエル）

1.3　JPEG 画像を確認

2　課題１

libjpeg ライブラリを用いて、PPM ファイル「foo.ppm」を JPEG に変換しなさい。PPM ファイルのヘッダには、形式、サイズが記述されているが、P6、320x240 を仮定し、ヘッダは読み飛ばしてよい。

3　課題２

課題１のプログラムを汎用的に使用できるように書き換えなさい。コマンドラインから入力ファイル名「foo.ppm」、出力ファイル名「foo.jpg」を受け取り、入力ファイルのヘッダから形式、サイズを読み PPM から JPEG に変換しなさい。但し、実装は形式 P6 だけでよい。

```
$ ppm2jpeg  foo.ppm  foo.jpg
```

4　プログラム例

```
$ gedit  jpeg_samp_2.c
```

```
#include <stdio.h>
#include <stdlib.h>
#include <jpeglib.h>

int main () {
    // JPEGオブジェクト，エラーハンドラを確保
    struct jpeg_compress_struct cinfo;
    struct jpeg_error_mgr jerr;
    FILE  *fp_in, *fp_out;
    int   i,j;
    char  *filename_in = "camera320x240.ppm";
    char  *filename_out = "camera320x240.jpg";
    // 画像サイズ
    int width = 320;
    int height = 240;

    // エラーハンドラにデフォルト値を設定
    cinfo.err = jpeg_std_error(&jerr);

    // JPEGオブジェクトの初期化
    jpeg_create_compress(&cinfo);

    // 入力ファイルをオープン
    if ((fp_in = fopen(filename_in, "rb")) == NULL) {
        fprintf(stderr, "cannot open %s¥n", filename_in);
        exit(EXIT_FAILURE);
    }
    // 出力ファイルをオープン
    if ((fp_out = fopen(filename_out, "wb")) == NULL) {
        fprintf(stderr, "cannot open %s¥n", filename_out);
        exit(EXIT_FAILURE);
    }
    // 出力ファイルを設定
    jpeg_stdio_dest(&cinfo, fp_out);

    // 画像をパラメータの設定
    cinfo.image_width = width;
    cinfo.image_height = height;
    cinfo.input_components = 3;
    cinfo.in_color_space = JCS_RGB;
    jpeg_set_defaults(&cinfo);
    jpeg_set_quality(&cinfo, 75, TRUE);

    // 圧縮を開始
    jpeg_start_compress(&cinfo, TRUE);

    // ヘッダをスキップ
    fseek(fp_in, 15L, SEEK_SET);

    // データを設定
    JSAMPARRAY img = (JSAMPARRAY) malloc(sizeof(JSAMPROW) * height);
    for (i = 0; i < height; i++) {
        img[i] = (JSAMPROW) malloc(sizeof(JSAMPLE) * 3 * width);
        for (j = 0; j < width * 3; j++) {
            img[i][j] = fgetc(fp_in);
        }
    }
```

```
    // ファイルに書き込む
    jpeg_write_scanlines(&cinfo, img, height);

    // 圧縮を終了
    jpeg_finish_compress(&cinfo);

    // JPEGオブジェクトを破棄
    jpeg_destroy_compress(&cinfo);

    for (i = 0; i < height; i++) {
        free(img[i]);
    }
    free(img);
    fclose(fp_in);
    fclose(fp_out);
}
```

5　プログラム例

```
$ gedit  jpeg_samp_3.c
```

6　参考

Cjpegwo 用いて JPEG に変換

```
$ cjpeg  -quality  60  foo.ppm  >  foo.jpg
```


JPEG(その２)

― YUV の濃淡画像を JPEG に変換 ―

Web カメラで画像を取得する場合、YUV が使用できる。YUV は輝度（luminance）を表す Y と、色差（chrominance）を表す U、V で構成される。YUV が用いられるきっかけは、アナログの TV 映像が白黒画像からカラー画像に進歩したときである。カラー画像は RGB で表現するので、白黒画像に対して 3 倍の帯域が必要であり、帯域を減らす必要があった。また、社会には白黒 TV とカラーTV が混在するので、2 種類の TV に対応しなければならなかった。人間の目は輝度の変化に比べ、色度の変化に気づかないという性質がある。よって、カラーの要素である RGB を YUV に変換することにより、白黒 TV では Y を、カラーTV では YUV を用いて映像を再生する。また、色度のデータを間引くことにより帯域を減らすことができる。

Y、U、Vは基準の周波数でサンプリングすることにより得られる。サンプリング値をY、C_b、C_rで表すと、Y、U、VはY、C_b、C_rを正規化した値である。RGBとY、C_b、C_rの関係式は以下のとおりである。

$$
\begin{aligned}
Y &= 0.299R + 0.587G + 0.114B \\
C_b &= -0.169R - 0.331G + 0.500B \\
C_r &= 0.500R - 0.419G - 0.081B
\end{aligned}
$$

Y、C_b、C_r についてサンプリング値をそれぞれY_s,C_{bs}, C_{rs}とすると、YUVとYC_bC_rの関係式は次のとおりである。Y_sの値域は 0 ≦Y_s≦1、C_{bs}、C_{rs}の値域は-0.5≦C_{bs}≦0.5、-0.5≦C_{rs}≦0.5 である。Y、U、Vの値は 8 ビットの整数なのでY_s、C_{bs}、C_{rs}の値を正規化する。Yの値は 16 ≦Y≦235、U、Vの値は 16 ≦U≦240、16 ≦V≦240 である。

$$
\begin{aligned}
Y &= 219Y_s + 16 \\
U &= 224C_{bs} + 128 \\
V &= 224C_{rs} + 128
\end{aligned}
$$

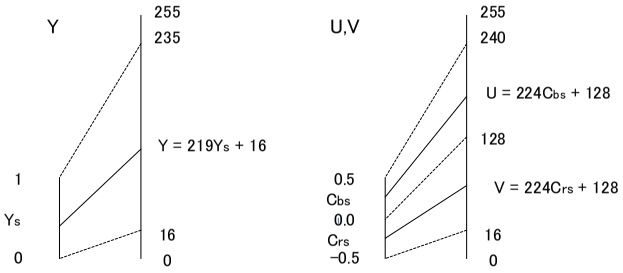


図１　YC_bC_rからYUVへの変換

図２は Web カメラで取得する画像フォーマット YUV422 である。１ピクセルは Y、U、V の３成分で構成される。YUV422 の 422 は、Y、U、V がそれぞれ基準サンプリング周波数 3.725MHz(1 秒間に 372.5 万回のサンプリング)の４倍、２倍、２倍でサンプリングすることを表している。すなわち、色度を表す U、V のデータ量は、輝度を表す Y のデータ量の半分である。１番目の画素は、Y1、U1、V1、２番目の画素は Y2、U1、V1 で構成され、U1、V1 のデータは１番目の画素と２番目の画素で共用される。YUV422 でのデータの並びは、Y1,U1,Y2,V1,…の順である。

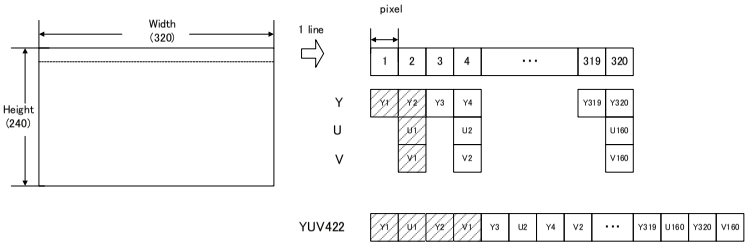


図２　YUV422

１　課題１

YUV データの Y 成分のみを使用して、グレイ（濃淡）での画像 PGM（Portable GrayMap format）に変換するプログラムを作成しなさい。

```
$ ./yuv2pgm  room.yuv  room.pgm
```

２　課題２

課題１で作成した PGM ファイルを JPEG ファイルに変換しなさい。入力データがグレイスケールであることは、カラースペース「cinfo.in_color_space」に「JCS_GRAYSCALE」を指定すればよい。

```
$ ./pgm2jpeg  room.pgm  room_gray.jpeg
```

３　プログラム例

```
$ gedit  yuv2pgm.c
```

```
/*
 yuv422 の画像から Y 成分（濃淡）のみを取り出し
    グレイ画像を生成する
*/

#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]) {
    FILE  *fp_yuv, *fp_pgm;
    int   i,size;
    int   Y, C;
    int width = 320;
    int height = 240;

    if ((fp_yuv = fopen(argv[1], "rb")) == NULL) {
        fprintf(stderr, "cannot open %s¥n", argv[1]);
        exit(EXIT_FAILURE);
    }
    if ((fp_pgm = fopen(argv[2], "wb")) == NULL) {
        fprintf(stderr, "cannot open %s¥n", argv[2]);
        exit(EXIT_FAILURE);
    }

    fprintf(fp_pgm, "P5¥n320 240¥n255¥n");
    size = width * height;
    for (i = 0 ; i < size ; i++) {
        fread(&Y, 1, 1, fp_yuv);
        fread(&C, 1, 1, fp_yuv);   // 読み飛ばす
        fwrite(&Y, 1, 1, fp_pgm);
    }
    fclose(fp_yuv);
    fclose(fp_pgm);
}
```

４　プログラム例

```
$ gedit  pgm2jpeg.c
```

```
/*
     グレイ画像を JPEG 画像に変換
*/
#include <stdio.h>
#include <stdlib.h>
#include <jpeglib.h>

int main (int argc, char *argv[]) {
    struct jpeg_compress_struct cinfo;
    struct jpeg_error_mgr jerr;
    FILE *fp_gray, *fp_jpeg;
    int   i,j;
    int width = 0;
    int height = 0;
    char format[3] = "  ";

    if ((fp_gray = fopen(argv[1], "rb")) == NULL) {
        fprintf(stderr, "cannot open %s¥n", argv[1]);
```

```
        exit(EXIT_FAILURE);
    }
    if ((fp_jpeg = fopen(argv[2], "wb")) == NULL) {
        fprintf(stderr, "cannot open %s¥n", argv[2]);
        exit(EXIT_FAILURE);
    }
    // エラーハンドラにデフォルト値を設定
    cinfo.err = jpeg_std_error(&jerr);

    // JPEGオブジェクトの初期化
    jpeg_create_compress(&cinfo);

    // 出力ファイルを設定
    jpeg_stdio_dest(&cinfo, fp_jpeg);

    // 画像パラメータを取得
    fscanf(fp_gray, "%s%d%d", format, &width, &height);
    printf("format = %s¥n", format);
    printf("width = %d  height = %d¥n", width, height);

    // 画像パラメータを設定
    cinfo.image_width = width;
    cinfo.image_height = height;
    cinfo.input_components = 1;
    cinfo.in_color_space = JCS_GRAYSCALE;
    jpeg_set_defaults(&cinfo);
    jpeg_set_quality(&cinfo, 75, TRUE);

    // 圧縮を開始
    jpeg_start_compress(&cinfo, TRUE);

    // ヘッダをスキップ
    fseek(fp_gray, 15L, SEEK_SET);

    // データを設定
    JSAMPARRAY img = (JSAMPARRAY) malloc(sizeof(JSAMPROW) * height);
    for (i = 0; i < height; i++) {
        img[i] = (JSAMPROW) malloc(sizeof(JSAMPLE) * width);
        for (j = 0; j < width ; j++) {
            img[i][j] = fgetc(fp_gray);
        }
    }

    // ファイルに書き込む
    jpeg_write_scanlines(&cinfo, img, height);

    // 圧縮を終了
    jpeg_finish_compress(&cinfo);

    // JPEGオブジェクトを破棄
    jpeg_destroy_compress(&cinfo);

    for (i = 0; i < height; i++) {
       free(img[i]);
    }
    free(img);
    fclose(fp_gray);
    fclose(fp_jpeg);
}
```


JPEG(その３)

―　YUV のカラー画像を JPEG に変換　―

YUVのカラー画像をJPEGに変換する。変換はYUVからPPM（Portable PixMap format）に、PPMからJPEGに２段階で実行する。YC_bC_rからRGBへの変換は、以下の関係式で求めることができる。

$$\begin{aligned} R &= 1.0Y + 0.0C_b + 1.402C_r \\ G &= 1.0Y - 0.344C_b - 0.714C_r \\ B &= 1.0Y + 1.772C_b + 0.0C_r \end{aligned} \quad \cdots(1)$$

1　課題１

以下のプログラムを参考に YUV データからカラー画像 PPM（Portable PixMap format）に変換するプログラムを作成しなさい。

```
$ ./yuv2ppm  room.yuv  room.ppm
```

このプログラムは、YUV を RGB に変換するプログラムの一部である。

```
int       Y, U, V;
int       R, G, B;
double    r, g, b;
double    y, Cb, Cr;

y  = (255 / 219.0) * (Y - 16);
Cb = (255 / 224.0) * (U - 128);
Cr = (255 / 224.0) * (V - 128);

r = 1.0 * y + 0.0   * Cb + 1.402 * Cr;
g = 1.0 * y - 0.344 * Cb - 0.714 * Cr;
b = 1.0 * y + 1.772 * Cb + 0.0   * Cr;

R = clamp(r);
G = clamp(g);
B = clamp(b);
```

YUV から PPM に変換する場合、浮動小数での計算結果を整数に丸める必要があるので clamp 関数を定義する。

```
int       clamp(double x)
{
 int      r = x;

 if (r < 0)
          return 0;
 else if (r > 255)
          return 255;
 else
          return r;
}
```

2　課題２

「JPEG（その１）」課題２で作成したプログラムを用い、「JPEG（その３）」課題１の PPM ファイル room.ppm を JPEG に変換しなさい。

```
$ ./ppm2pgm  room.ppm  room.jpeg
```

3　課題３

「JPEG（その２）」でのRGBからYC_bC_rへの変換式を行列で表現すると式(2)となる.

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \qquad \cdots (2)$$

行列Mを式(3)で定義する。Mの逆行列M^{-1}でYC_bC_rからRGBへの変換式を表すと式(4)となる。M^{-1}が式(5)となることを確認しなさい。

$$M = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{pmatrix} \qquad \cdots (3)$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.0 & 0.0 & 1.402 \\ 1.0 & -0.344 & -0.714 \\ 1.0 & 1.772 & 0.0 \end{pmatrix} \begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} \qquad \cdots (4)$$

$$M^{-1} = \begin{pmatrix} 1.0 & 0.0 & 1.402 \\ 1.0 & -0.344 & -0.714 \\ 1.0 & 1.772 & 0.0 \end{pmatrix} \qquad \cdots (5)$$

4　参考

4.1　2X2 行列の逆行列

2X2 の正方行列Aが$|A| \neq 0$のときAの逆行列A^{-1}が計算できる。ただし、$|A|$はAの行列式である。

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

$$A^{-1} = \frac{1}{|A|}\begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix} \quad , \quad \det A = |A| = a_{11}a_{22} - a_{12}a_{21}$$

4.2　例１

$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$の逆行列は$A^{-1} = \begin{pmatrix} -2 & 1 \\ 3/2 & -1/2 \end{pmatrix}$である。

$$A^{-1} = \frac{1}{|A|}\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \frac{1}{1\cdot 4 - 2\cdot 3}\begin{pmatrix} 4 & -2 \\ -3 & 1 \end{pmatrix} = \begin{pmatrix} -2 & 1 \\ 3/2 & -1/2 \end{pmatrix}$$

4.3　3X3 行列の逆行列

3X3 の正方行列Aが$|A| \neq 0$のときAの逆行列A^{-1}が計算できる。ただし、$|A|$はAの行列式、A_{ij}はAの余因数である。

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$A^{-1} = \frac{1}{|A|}\begin{pmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{13} & A_{23} & A_{33} \end{pmatrix}$$

$$\det A = |A| = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}$$

$$A_{ij} = (-1)^{i+j} D_{ij}$$

D_{ij}はAの第i行、第j列を除いた残りの行列の行列式である。たとえば、D_{12}はAの第１行、第２列を除いた残りの行列$\begin{pmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{pmatrix}$の行列式$\det\begin{pmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{pmatrix} = a_{21}a_{33} - a_{23}a_{31}$である。

4.4　例２

$A=\begin{pmatrix}1 & 0 & 2\\ 4 & 1 & 3\\ 2 & -1 & 0\end{pmatrix}$ の逆行列は $A^{-1}=\frac{1}{9}\begin{pmatrix}-3 & 2 & 2\\ -6 & 4 & -5\\ 6 & -1 & -1\end{pmatrix}$ である。

A の行列式を計算する。

$$\det A=|A|=a_{11}a_{22}a_{33}+a_{12}a_{23}a_{31}+a_{13}a_{21}a_{32}-a_{13}a_{22}a_{31}-a_{12}a_{21}a_{33}-a_{11}a_{23}a_{32}$$
$$=1\cdot1\cdot0+0\cdot3\cdot2+2\cdot4\cdot(-1)-2\cdot1\cdot2-0\cdot4\cdot0-1\cdot3\cdot(-1)=-9$$

第１行、第２列の余因数 A_{12} を計算すると $A_{12}=6$ が得られる。

$$D_{12}=\det\begin{pmatrix}4 & 3\\ 2 & 0\end{pmatrix}=4\cdot0-3\cdot2=-6$$

$$A_{12}=(-1)^{1+2}D_{12}=6$$

残りの A_{ij} を計算し、A^{-1} を求める。

$$A^{-1}=\frac{1}{9}\begin{pmatrix}-3 & 2 & 2\\ -6 & 4 & -5\\ 6 & -1 & -1\end{pmatrix}$$

5　プログラム例

$ **gedit yuv2ppm.c**

```
/*
   YUV422 から PPM への変換
 */

#include <stdio.h>
#include <stdlib.h>

int       clamp(double x)
{
    int   r = x;

    if (r < 0)
        return 0;
    else if (r > 255)
        return 255;
    else
        return r;
}

int main (int argc, char *argv[]) {
    FILE    *fp_yuv, *fp_ppm;
    int     i,size;
    int     Y1, Y2, U, V;
    int     R1, G1, B1, R2, G2, B2;
    double  r1, g1, b1, r2, g2, b2;
    double  y1, y2, Cb, Cr;

    int width = 320;
    int height = 240;

    if ((fp_yuv = fopen(argv[1], "rb")) == NULL) {
        fprintf(stderr, "cannot open %s¥n", argv[1]);
        exit(EXIT_FAILURE);
    }
    if ((fp_ppm = fopen(argv[2], "wb")) == NULL) {
        fprintf(stderr, "cannot open %s¥n", argv[2]);
        exit(EXIT_FAILURE);
    }
    fprintf(fp_ppm, "P6¥n320 240¥n255¥n");
    size = width * height / 2;

    for (i = 0 ; i < size ; i++) {
        fread(&Y1, 1, 1, fp_yuv);
        fread(&U,  1, 1, fp_yuv);
        fread(&Y2, 1, 1, fp_yuv);
        fread(&V,  1, 1, fp_yuv);

        y1 = (255 / 219.0) * (Y1 - 16);
        y2 = (255 / 219.0) * (Y2 - 16);
        Cb = (255 / 224.0) * (U - 128);
        Cr = (255 / 224.0) * (V - 128);

        r1 = 1.0 * y1 + 0     * Cb + 1.402 * Cr;
        g1 = 1.0 * y1 - 0.344 * Cb - 0.714 * Cr;
        b1 = 1.0 * y1 + 1.772 * Cb + 0     * Cr;
        r2 = 1.0 * y2 + 0     * Cb + 1.402 * Cr;
        g2 = 1.0 * y2 - 0.344 * Cb - 0.714 * Cr;
        b2 = 1.0 * y2 + 1.772 * Cb + 0     * Cr;
```

```
            R1 = clamp (r1);
            G1 = clamp (g1);
            B1 = clamp (b1);
            R2 = clamp (r2);
            G2 = clamp (g2);
            B2 = clamp (b2);

            fwrite(&R1, 1, 1, fp_ppm);
            fwrite(&G1, 1, 1, fp_ppm);
            fwrite(&B1, 1, 1, fp_ppm);
            fwrite(&R2, 1, 1, fp_ppm);
            fwrite(&G2, 1, 1, fp_ppm);
            fwrite(&B2, 1, 1, fp_ppm);
        }
        fclose(fp_yuv);
        fclose(fp_ppm);

}
```

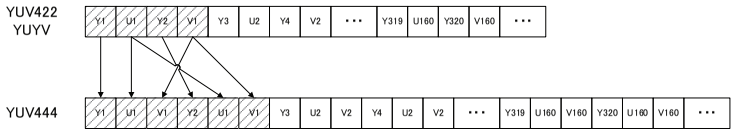
2012/10/17

JPEG(その４)

― YUV のカラー画像を JPEG に変換 ―

1 課題１

YUV422 形式の画像を PPM に変換せずに JPEG に変換しなさい。YUV422 を YUV444 に変換すると YUV から直接 JPEG に変換できる。画像データの並びは、Y1,U1,Y2,V1,... から Y1,U1,V1,Y2,U1,V1,... に変換すればよい。



画像パラメータは、1 ピクセル当たりのカラー要素数が 3 なので cinfo.input_components に 3 を、カラースペースには JCS_YCbCr を指定しなさい。

```
cinfo.input_components = 3;
cinfo.in_color_space = JCS_YCbCr;
```

3　プログラム例

$ **gedit yuv2jpeg.c**

```
/*
 YUV422 から JPEG への変換
*/

#include <stdio.h>
#include <stdlib.h>
#include <jpeglib.h>

int main (int argc, char *argv[]) {
    // JPEG オブジェクト, エラーハンドラを確保
    struct jpeg_compress_struct cinfo;
    struct jpeg_error_mgr jerr;
    FILE *fp_yuv, *fp_jpeg;
    int   i, j, width_length;
    JSAMPLE      Y1, Y2, U, V;
   // 画像サイズ
   int width = 320;
   int height = 240;

    // 入力ファイルをオープン
    if ((fp_yuv = fopen(argv[1], "rb")) == NULL) {
        fprintf(stderr, "cannot open %s¥n", argv[1]);
        exit(EXIT_FAILURE);
    }
    // 出力ファイルをオープン
   if ((fp_jpeg = fopen(argv[2], "wb")) == NULL) {
        fprintf(stderr, "cannot open %s¥n", argv[2]);
        exit(EXIT_FAILURE);
    }
    // エラーハンドラにデフォルト値を設定
    cinfo.err = jpeg_std_error(&jerr);

    // JPEG オブジェクトの初期化
    jpeg_create_compress(&cinfo);

    // 出力ファイルを設定
    jpeg_stdio_dest(&cinfo, fp_jpeg);

    // 画像パラメータを設定
    cinfo.image_width = width;
    cinfo.image_height = height;
    cinfo.input_components = 3;
    cinfo.in_color_space = JCS_YCbCr;
    jpeg_set_defaults(&cinfo);
    jpeg_set_quality(&cinfo, 75, TRUE);

// YUV での設定？
//  cinfo.dct_method = JDCT_FLOAT;
//  jpeg_suppress_tables(&cinfo, TRUE);

    // 圧縮を開始
    jpeg_start_compress(&cinfo, TRUE);

    // データを設定
    width_length = width * 3;
    JSAMPARRAY img = (JSAMPARRAY) malloc(sizeof(JSAMPROW) * height);
    for (i = 0; i < height; i++) {
        img[i] = (JSAMPROW) malloc(sizeof(JSAMPLE) * width_length);
        for (j = 0; j < width_length; j+=6) {
            Y1 = fgetc(fp_yuv);
             U = fgetc(fp_yuv);
            Y2 = fgetc(fp_yuv);
             V = fgetc(fp_yuv);
            img[i][j + 0] = Y1;
            img[i][j + 1] = U;
            img[i][j + 2] = V;
            img[i][j + 3] = Y2;
            img[i][j + 4] = U;
            img[i][j + 5] = V;
        }
    }

    // ファイルに書き込む
    jpeg_write_scanlines(&cinfo, img, height);

    // 圧縮を終了
    jpeg_finish_compress(&cinfo);

    // JPEG オブジェクトを破棄
    jpeg_destroy_compress(&cinfo);

    for (i = 0; i < height; i++) {
        free(img[i]);
    }
    free(img);
    fclose(fp_yuv);
    fclose(fp_jpeg);
}
```

2012 年 12 月 12 日

生産電子情報システム技術科

システム計画書

生産電子情報システム技術科における標準課題「組込みシステム構築課題実習」では、実習課題として遠隔監視システムを作成する。遠隔監視システムは、複数の遠隔監視装置で構成され、複数の監視地点からの監視データ（画像）を中央の監視室に収集して一括管理する。遠隔監視装置は、遠隔地の監視地点にカメラ装置を設置し、中央の監視室に表示装置を設置する。遠隔地のカメラ装置と中央の表示装置間のデータ通信は、インターネットを用いる。監視データを集中管理することで監視員を削減し、インターネットを用いることでシステムを安価に実現する。

学生をグループに分け、各グループは遠隔監視装置を１台作成する。以下に遠隔監視装置の要求仕様を示す。

1　システム構成

図１に遠隔監視装置のシステム構成、表１に構成部品を示す。遠隔監視装置は、カメラ装置、表示装置、ネットワークで構成する。カメラ装置はカメラ、ターゲットボードで構成する。表示装置はLCD、ターゲットボードで構成する。ネットワークは、インターネットを模擬した装置で構成する。

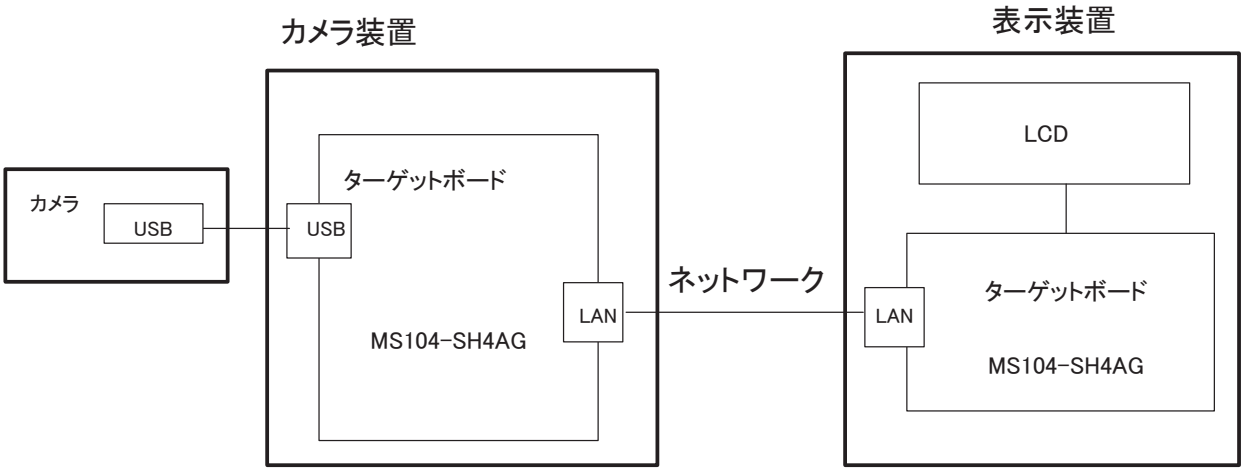


図１　システム構成

表１　構成部品

装置	名称	構成部品	個数
カメラ装置	カメラ	Webcam C250（予定）	1
〃	ターゲットボード	MS104-SH4AG	1
表示装置	LCD	MS104-LCD／AUDIO	1
〃	ターゲットボード	MS104-SH4AG	1
ネットワーク	HUB		1
付属物	LAN ケーブル		2

2　遠隔監視装置仕様

遠隔監視装置は、カメラから受信した画像を表示装置の LCD に表示する。遠隔監視装置は画像を n 回取得・表示後、正常終了する。

注意）画像を取得・表示する回数 n については、後日決定する。

2.1　画像表示

(1)　カメラ装置と表示装置の電源を ON にすると、システムが自動的に起動し、画像を表示する。
(2)　カメラ装置と表示装置の電源は、任意の順に ON にしてよい。
(3)　カメラは画像を常時連続して受信し、リアルタイムに LCD に表示する。
(4)　画像は 1 秒間に 1 枚以上表示する。
(5)　画像の大きさは、幅 320 ピクセル、高さ 240 ピクセルである。
(6)　画像を n 回取得・表示すると、カメラ装置と表示装置はシステムを正常終了する。
(7)　カメラ装置の電源を OFF にすると、n 回表示する途中でシステムを終了する。
(8)　表示装置の電源を OFF にすると、n 回表示する途中でシステムを終了する。
(9)　画像は、データの取得、通信の状態により一時的に乱れてもよい。

2.2　カメラ装置仕様

(1)　ターゲットボードのオペレーティングシステムは Linux である。
(2)　カメラとターゲットボードは、USB で接続する。
(3)　カメラから取り込む画像は、YUV 形式の画像である。
(4)　YUV 形式の画像から JPEG 形式の画像に変換する。
(5)　JPEG 形式の画像はネットワークを介して表示装置に送信される。

2.3　表示装置仕様

(1)　ターゲットボードのオペレーティングシステムは Linux である。
(2)　表示装置はカメラ装置から JPEG 形式の画像を受信する。
(3)　表示装置は JPEG 形式の画像を LCD に表示する。

2.4　ネットワーク仕様

(1)　インターネットを模擬したネットワークとして、HUB を用いて LAN を構築する。

(2)　プロトコルは、TCP/IP の Version4 である。

(3)　通信速度は 100Mbps である。

３　提出物

(1)　カメラ装置（組上げた状態であること）

(2)　表示装置（組上げた状態であること）

(3)　カメラ装置、表示装置の運用マニュアル

(4)　システム要求仕様書

(5)　システム概要設計書

(6) システム詳細設計書

2013 年 3 月 1 日
生産電子情報システム技術科
プロジェクト実施チーム

遠隔監視装置の設計

1　組込みシステムの設計

オペレーティングシステムを用いた組込みシステムは、動作、時間、機能、構造の４つの視点から設計する必要がある。従来の組込みシステムは、機器にマイコンを搭載し、マイコンがシステム全体を制御した。マイコンを用いた組込みシステムでは、システムの動作と時間を重視した設計手法が採用されてきた。近年、コンピュータの小型化により、マイコンでなくオペレーティングシステムを搭載した組込みシステムが実現可能となった。組込みシステムにオペレーティングシステムを用いると、システムの多機能化、処理の高度化が図れる。しかし、オペレーティングシステムを用いた組込みシステムは、機能が多く、構造も複雑であり、動作と時間の視点による設計では不十分である。オペレーティングシステムを用いた組込みシステムの設計は、ソフトウェア設計で用いられてきた機能と構造の視点を取り入れる必要である。

組込みシステムの設計手法には、構造化分析手法とオブジェクト指向設計手法がある。構造化分析手法はソフトウェアの設計手法として、1960 年代後半に提案され、リアルタイムシステムの設計に適用できるまで拡張されている。オブジェクト指向設計手法は 1970 年代から始まり、ソフトウェア開発の大規模化、短納期化に対応できる手法として考案された。

構造化分析手法はオブジェクト分析手法など最新の技法の基礎となっており、設計の手順が明確で、初めてソフトウェアの設計手法を学ぶ場合、設計の本質を理解し易い。生産電子情報システム技術科の標準課題「組込みシステム構築課題実習」では、組込みソフトウェア管理者・技術者育成研究会（SESSAME: Society of Embedded Software Skill Acquisition for Managers and Engineers）で提唱された構造化分析手法[1]に基づいてシステムを設計する。

２　構造化モデリング

構造化モデリングは要求モデリング、分析モデリング、設計モデリングの順に設計を進め、機能の実装方法を後の工程で決める。組込みシステムはハードウェアとソフトウェアのいずれでも実装することが可能な機能がある。システム設計の抽象化の考え方から、機能の実装においてハードウェアまたはソフトウェアの選択は、できる限り後の工程で決める。SESSAME が提案した構造化分析手法では、設計モデリングにおいてタスク構造の設計、パッケージの設計、メモリ構造の設計が記述されていない。この３つの設計は構造化分析手法においても必要なので、同じく SESSAME が提案したオブジェクト指向モデリングを参考に構造化分析手法に置き換えて設計を行う。設計モデリング１は文献１で述べられた設計モデリングである。設計モデリング２は文献２の設計モデリングでのタスク構造の設計、メモリ構造の設計を構造化分析手法に置き換えた方法である。

2.1　要求モデリング

要求モデリングでは、「システムに何をしてほしいか」、その結果「何が起こるのか」を考え、システムに必要な機能要件を整理し、仕様書にまとめる。また、システムの機能とは別に時間的制約、品質など非機能要件を整理する。要求モデリングは以下の手順で行う（図 2.1）。

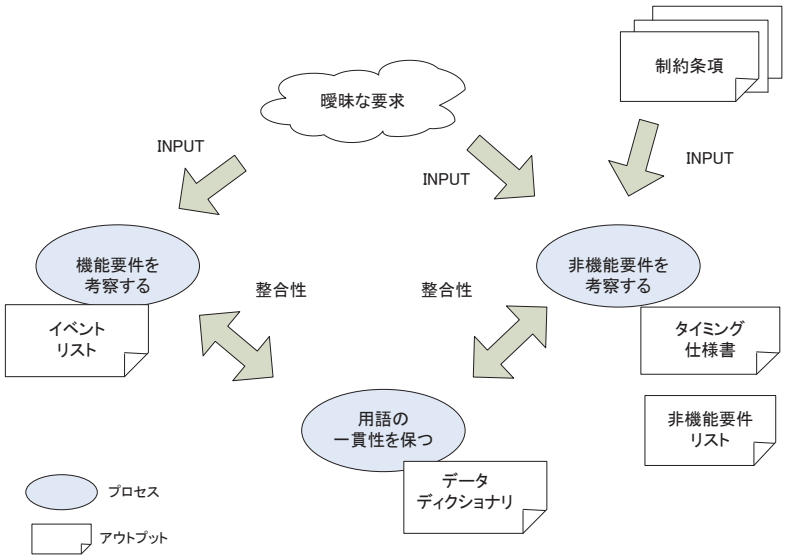


図 2.1　要求モデリング

(1)　イベントリストの作成

システムに対する曖昧な要求から、システムの外部で発生するイベント（事象）に対して、システムの行動と応答、システム外への影響をまとめイベントリストを作成する。イベントリストにより、システムと外部との入出力が明確になる。

(2)　タイミング仕様書の作成

システムに対する曖昧な要求と制約条項から、システムのリアルタイム性に要求される時間的制約を整理し、タイミング仕様書を作成する。

(3) 非機能要件リストの作成

タイミング仕様書に盛り込まない安全性、信頼性、拡張性、保守性、ハードウェア制約などを非機能要件リストをしてまとめる。

(4) データディクショナリの作成

設計の各工程で用いられる用語を統一するため、データディクショナリを作成する。データディクショナリの用語は以下の書式を用いる。

【記号の定義】

A ＝ B	A を B で定義する
A ＝ “text”	A はリテラル”text”である
A ＝ 型、値域、単位系	A は値域の範囲の値をもつ
A ＝ B ＋ C	A は B と C をもつ
［ A ｜ B ］	A または B
{ A }	A の繰返し
(A)	A はオプション

2.2 分析モデリング

分析モデリングでは、要求モデリングで「システムに何をしてほしいか」という視点で整理した結果をもとに、「システムが何を実現するか」という視点で分析する。分析モデリングは以下の手順で行う（図 2.2）。

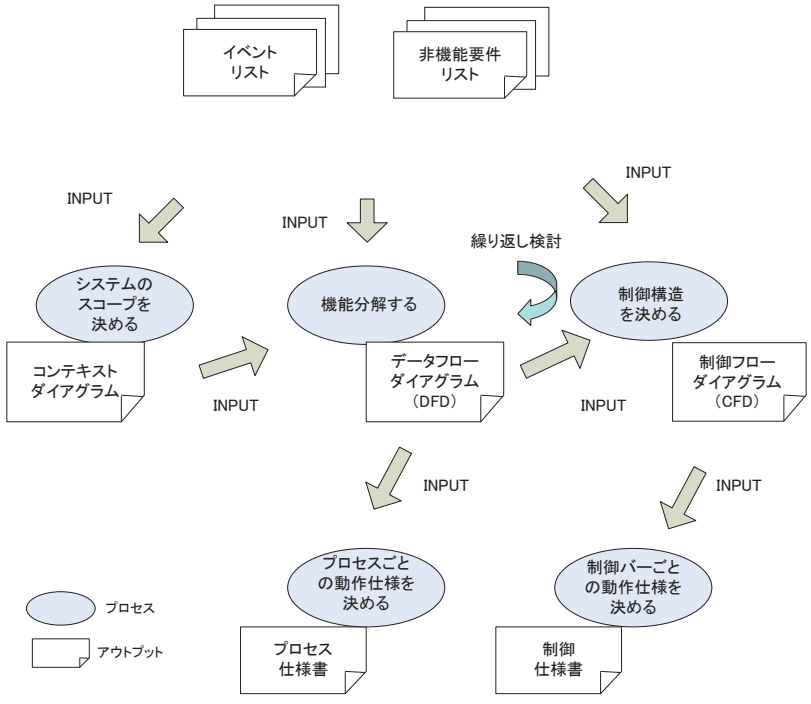


図 2.2 分析モデリング

(1)　コンテキストダイアグラムの作成

要求モデル（イベントリスト、非機能要件リストのハードウェア制約）をもとにシステムの開発対象範囲を決め、コンテキストダイアグラムを作成する。

コンテキストダイアグラムは、ターミネータ、プロセス、データ／制御フローで構成する。ターミネータは外部エンティティであり長方形で表す。プロセスを○で表し、ターミネータからプロセスへのデータまたは制御の入出力フローを矢印で示す。

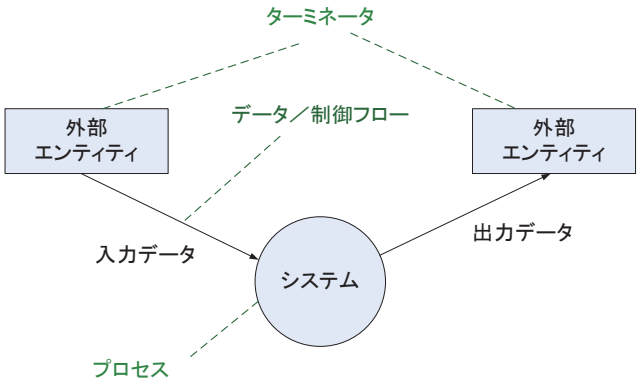


図 2.3　コンテキストダイアグラムの表記法

(2)　データフローダイアグラムの作成

要求モデルとコンテキストダイアグラムをもとに、システムの機能に着目し、抽象的なコンテキストダイアグラムから、より具体的なプロセスへ段階的に機能を分割する。このプロセスの分割は具体的な動作仕様が見えるまで行う。プロセスとプロセス間のデータの流れをデータフローダイアグラム（DFD: Data Flow Diagram）にまとめる。

データフローダイアグラムは、プロセス、データフロー、データストアで構成する。プロセスは、プログラムであり○で表す。データフローは、データ流れであり矢印で表す。データストアは、データを一時的に蓄積するものであり 2 本線で表す。

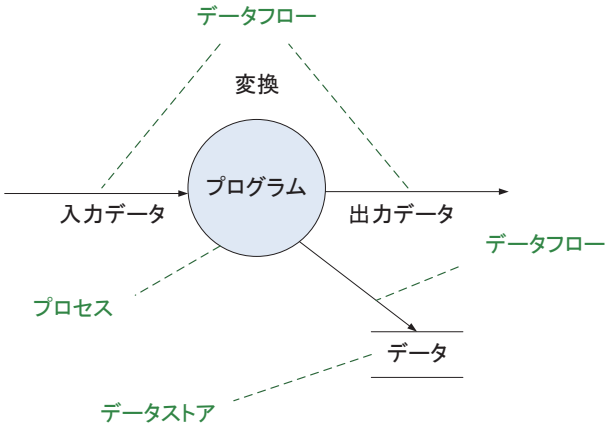


図 2.4　DFD の表記法

(3)　制御フローダイアグラムの作成

要求モデルとデータフローダイアグラムをもとに、システムの機能に制御関係がある場合、その制御の流れを制御フローダイアグラム（CFD: Control Flow Diagram）としてまとめる。DFD と CFD は相互に検討を繰り返すことでより、洗練された仕様にする。制御フローダイアグラムは、プロセス、制御バー、制御フローで構成する。制御バーは、制御の集約であり短い直線の棒で表す。制御フローは、制御情報の流れであり点線の矢印で表す。

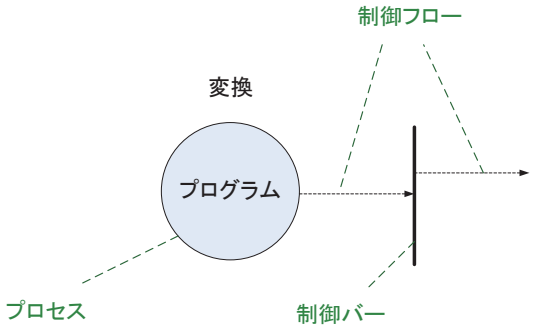


図 2.5　CFD の表記法

(4)　制御仕様書の作成

制御フローダイアグラムをもとに、複数の制御と束ねた制御バーごとに動作仕様を定義し、制御仕様書にまとめる。制御仕様書は状態遷移図（STD：State Transition Diagram）、状態遷移表（STT：State Transition Table）、デシジョンテーブル（DT：Decision Table）、プロセス起動テーブル（PAT：Process Activate Table）で記述する。

状態遷移図は、開始、状態、イベント／アクションにより構成する。状態遷移の開始を●、状態を長方形、状態間の遷移を矢印で示す。矢印には遷移の要因であるイベントと状態遷移とともに実行するアクションを記述する。

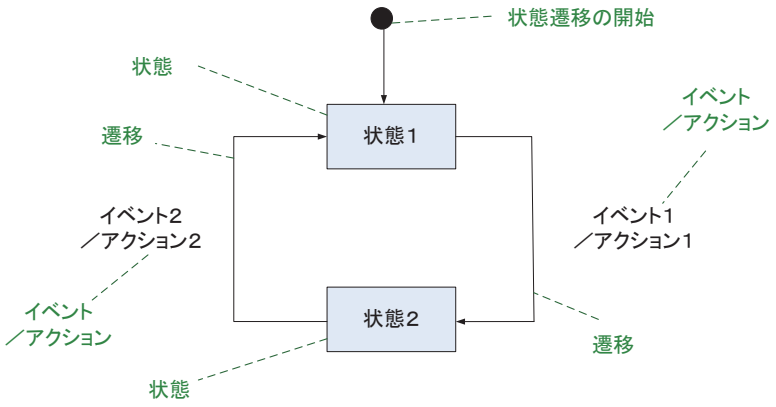


図 2.6　状態遷移図の表記法

状態遷移表は、ある状態でイベントが発生した場合、次にどの状態に遷移するか記述する。

表 2.1　状態遷移表の表記法

イベント / 状態	イベント１	・・・	イベント m
状態１	次の状態	・・・	次の状態
・・・	・・・	・・・	・・・
状態ｎ	次の状態	・・・	次の状態

×　＝　“起りえない（Can’t Happen）”

／　＝　“イベント無視（Don’t Care）”

次の状態　＝　［　状態１　｜　・・・　｜　状態ｎ　｜　×　｜　／　］

デシジョンテーブルは、入力値としてn個の条件があり、そのすべての組み合わせ 2^nに対して、出力であるm個の動作が実行されるか否かの組み合わせを記述する。

表 2.2　デシジョンテーブルの表記法

条件１	Y	Y	・・・	Y	N
条件２	Y	Y	・・・	N	N
・・・	・・・	・・・	・・・	・・・	・・・
条件 n-1	Y	Y	・・・	N	N
条件ｎ	Y	N	・・・	N	N
動作１	出力値	出力値	・・・	出力値	出力値
・・・	・・・	・・・	・・・	・・・	・・・
動作 m	出力値	出力値	・・・	出力値	出力値

Y　＝　“条件を満たす”、　N　＝　“条件を満たさない”

出力値　＝　［　○｜　―　］、　○　＝　“動作を実行”、　―　＝　“動作を実行しない”

プロセス起動デーブルは、入力イベントとプロセスの起動関係を記述する。

表 2.3　プロセス起動テーブルの表記法

入力	プロセス		
	プロセス番号１	・・・	プロセス番号m
イベント１	０または１	・・・	０または１
・・・	・・・	・・・	・・・
イベントｎ	０または１	・・・	０または１

０　＝　“プロセス停止”、　１　＝　“プロセス起動”

(5) プロセス仕様書の作成

データフローダイアグラムの分割された最後のプロセスごとにプロセスの動作仕様を定義し、プロセス仕様書にまとめる。

(6) データディクショナリの更新

分析モデリングにおいて新たに発生した用語をデータディクショナリに追加する。データディクショナリに存在するが、より詳細な定義ができる用語は修正する。

2.3 設計モデリング

設計モデリングでは、分析モデリングで「システムが何を実現するか」という視点で整理した結果をもとに、「どのようにしてシステムを実現するか」という視点で検討する。設計モデリング1は以下の手順で行う（図 2.7）。

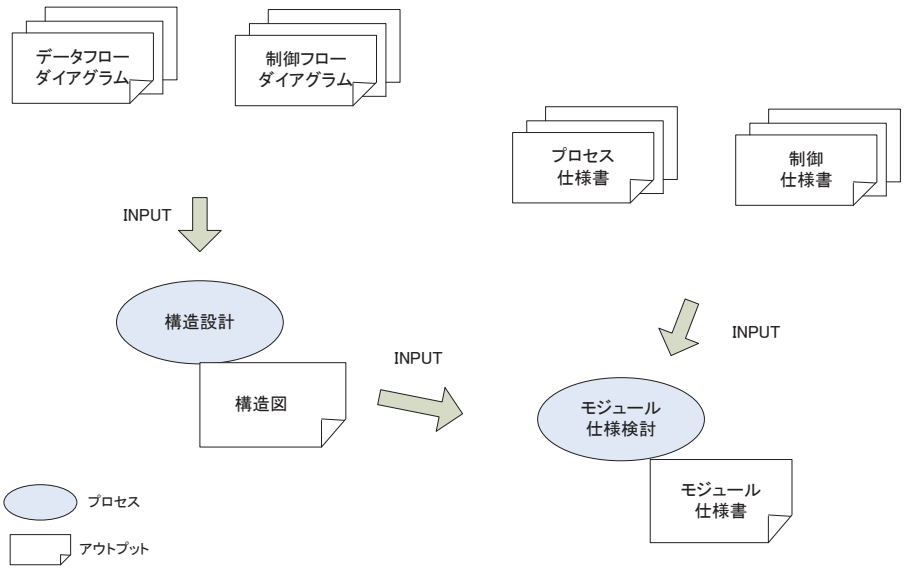


図 2.7 設計モデリング

(1) 構造図の作成

データフローダイアグラムと制御フローダイアグラムをもとシステムの機能を分割し、構造図にまとめる。構造図は機能をまとめて制御する BOSS モジュールを定め、BOSS モジュールを中心に組み合わせて一つの構造図を作成する。

構造図は、モジュール、呼び出し、情報で構成する。モジュールには処理または機能の単位であるプレーンモジュールと、データストアであるデータモジュールがある。プレーンモジュールを長方形で、データモジュールを六角形で表す。モジュール間の呼び出しを矢印で表す。呼び出す側のモジュールを親モジュール、呼び出される側を子モジュールと呼ぶ。呼び出しには、同期と非同期がある。同期の呼び出しは親モジュールが子モジュールを呼び出すと直ちに動作を開始する。非同期の呼び出しは親モジュールが子モジュールを呼び出しても、直ちに実行をされずコンテキストが実行可能になると動作を開始する。同期の呼び出しは実線の矢印で、非同期の呼び出しは点線の矢印で示す。モジュールの呼び出しにおいて、

条件に合致した場合に行われる処理をコンディションとよび、菱形で示す。モジュール間を流れる情報において、○の矢印はデータを、●の矢印は制御を示す。

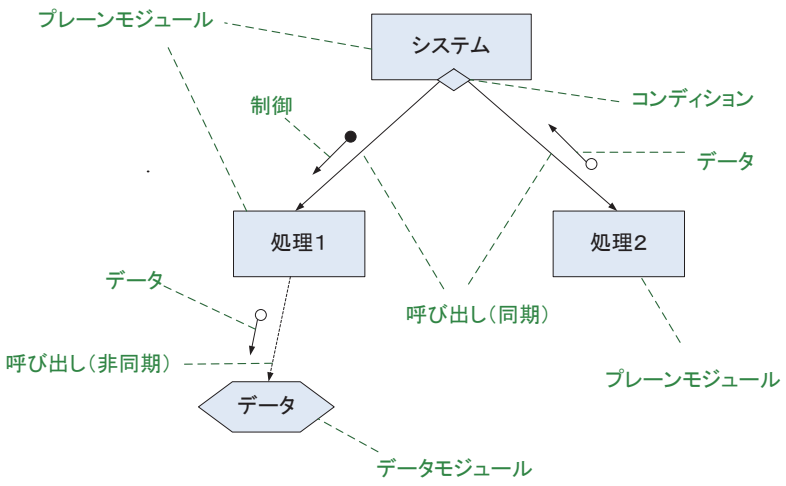


図 2.8　構造図の表記法

(2)　モジュール仕様書の作成

プロセス仕様書、制御仕様書、構造図をもとに単一の機能を実現するモジュールに分割し、インタフェースの仕様、機能仕様をまとめる。

(3)　データディクショナリの更新

設計モデリングにおいて新たに発生した用語をデータディクショナリに追加する。データディクショナリに存在するが、より詳細な定義ができる用語は修正する。

3　要求モデリング

3.1　イベントリスト

イベントリストには、対象とするシステムの外部で発生したイベント（事象）のうち、いつ発生するか分らず、システムが発生を制御できないイベントに対して提供する価値をまとめる。

遠隔監視システムは独立に画像を監視する複数の遠隔監視装置で構成する。遠隔監視装置の利用者は、システムを設置・保守するシステム管理者と、画像を常に監視し不測の事態に対応する監視者である。遠隔監視装置は利用者の意思により、任意の時間に画像を表示または非表示できる。

遠隔監視装置が正常に動作している場合、表示装置の LCD には監視者が認識できる物体や背景を表示する。しかし、何らかの事故、故障が発生した場合、利用者は LCD に表示している画像の乱れや、画像の非表示により異常を認識できる。

表 3.1　イベントリスト

イベント	スティミュラス	アクション	レスポンス	エフェクト
画像表示	監視開始	カメラから画像を取り込み、表示装置に画像を表示する	画像	監視する
画像非表示	監視停止	カメラから画像取り込みを終了し、表示装置を非表示とする	画像	監視終了
画像乱れ	（なし）	（なし）	画像	システムの異常が分る

【イベントリストの項目】

①イベント（事象）：　対象システムの外部で生じ、その発生をシステムが制御できない事象。

②スティミュラス（刺激）：　イベントが発生したことを伝えるデータ。

③アクション（行動）：　イベントが発生したときのシステムの果たすべき機能。

④レスポンス（応答）：　イベントが発生したときのシステムの外部への反応。

⑤エフェクト（影響）：　システムの反応による外部への影響。

3.2　タイミング仕様書

タイミング仕様書は、システムの内部からでなく外部から見て、入力のイベントに対し、出力のイベントが発生するまでの対応時間を記述する。

遠隔監視装置は表示装置の LCD に静止画像を 1 秒間に 1 枚以上表示することにより動画化する。よって、動画化についての応答時間は最大 60 秒である。

表 3.2　タイミング仕様書

入力	イベント	出力	イベント	対応時間
画像	動画化	画像	動画化	最大 60 秒

3.3　非機能要件リスト

非機能要件リストにはシステムの機能以外の制約、品質に関するものをまとめる。

システム計画書では、遠隔監視装置が使用するターゲットボード、オペレーティングシステム、ネットワークプロトコルが決められている。また、システムの不具合による修正プログラムやバージョンアップの容易さも望まれる。

表 3.3　非機能要件リスト

No	分類	内容	優先度	関係部署
1	環境	消費電力を x ワット以下とする	低	製品企画
2	保守性	ソフトウェアの修正プログラムを容易にインストールできる	中	製品企画
3	信頼性	5 年間連続使用を保障する	中	品質管理
4	拡張性	ソフトウェアのバーションアップが容易に行える	中	製品企画
5	ハードウェア	システム計画書で指示されたターゲットボードを使用する	高	開発部門
6	ソフトウェア	システム計画書で指示されたオペレーティングシステムを使用する	高	開発部門
7	ネットワーク	システム計画書で指示されたプロトコルを使用する	高	開発部門

3.4　データディクショナリ

データディクショナリは設計の各工程で使用する用語に一貫性をもたすため、誰でもわかる言葉で用語を定義する。システム計画書で定義された用語を管理するデータディクショナリに、要求モデリングで発生した用語を追加する。

3.4.1　システム計画書で定義された用語

遠隔監視システム　＝　｛　遠隔監視装置　｝
遠隔監視装置　＝　カメラ装置　＋　表示装置　＋　ネットワーク
カメラ装置　＝　カメラ　＋　ターゲットボード
表示装置　＝　LCD　＋　ターゲットボード
カメラ　＝　“遠隔監視装置で用いる USB カメラ”
LCD　＝　“遠隔監視装置で用いる表示デバイス”
ターゲットボード　＝　“組込み Linux を搭載したマイコンボード”
ネットワーク　＝　“インターネットを模擬した装置”
画像　＝　［　YUV 画像　｜　JPEG 画像　｜　非表示画像　］
YUV 画像　＝　“YUV 形式の圧縮されていない画像”
JPEG 画像　＝　“JPEG 方式で圧縮された画像”
非表示画像　＝　“背景色の画像”
動画化　＝　“1 秒間に 1 枚以上の静止画像を表示する”

3.4.2　要求モデリングで定義された用語

利用者　＝　［　システム管理者　｜　監視者　］
システム管理者　＝　“遠隔監視システムを設置、保守する者”
監視者　＝　“画像を監視する者”
監視開始　＝　“遠隔監視を開始する指示”
監視停止　＝　“遠隔監視を停止する指示”
画像表示　＝　“LCD に画像を表示する”
画像の乱れ　＝　“LCD に表示した背景や物体を利用者が正しく認識できない状態”
画像非表示　＝　“LCD に背景色を表示する”

４　分析モデリング

4.1　コンテキストダイアグラム

4.1.1　ターミネータの抽出

イベントリストからイベントの発生源、検出手段を整理し、コンテキストダイアグラムのターミネータを抽出する。発生源、検出手段の項目がターミネータの候補である。

イベントリストの「画像表示」「画像非表示」は、利用者の意思で発生するイベントである。「画像乱れ」は利用者でなく、遠隔監視装置の異常により発生するイベントである。これら３つのイベントは、すべてLCDに表示した画像を見ることで検出ができる。「遠隔監視装置」は開発対象システムなのでターミネータとせずに、「利用者」「LCD」をターミネータとする。

表4.1　イベントリストからターミネータの抽出

イベント	発生源	検出手段
画像表示	利用者	LCD
画像非表示	利用者	LCD
画像乱れ	遠隔監視装置	LCD

4.1.2　初版コンテキストダイアグラム

開発対象システムを中心に、4.1.1節で抽出したターミネータを周りに配置して初版コンテキストダイアグラムを作成する。

開発対象システムを「遠隔監視装置」、ターミネータを「利用者」「LCD」として初版コンテキストダイアグラムを作成する。イベントリストのスティミュラス「監視開始」「開始停止」を「ユーザ要求」としてまとめ、「利用者」から「遠隔監視装置」への入力フローとする。イベントリストのレスポンス「画像」を「遠隔監視装置」から「LCD」への出力フローとする。

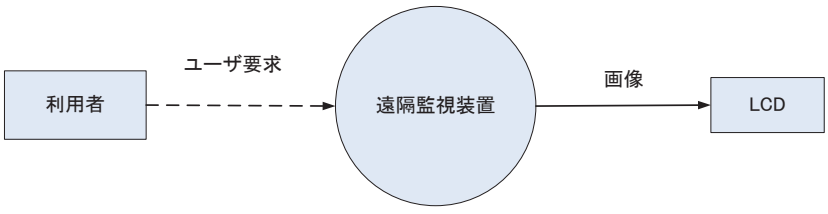


図4.1　初版コンテキストダイアグラム

4.1.3　初版 DFD0

初版コンテキストダイアグラムをもとに、主要プロセスを抽出、ターミネータを追加して初版 DFD0/CFD0 を作成する。抽出した主要プロセスの入力フローと出力フローを検討し、入力元と出力先をターミネータとする。イベントリストのスティミュラスを発するエンティティ、レスポンスを受け取るエンティティがターミネータの候補である。

遠隔監視装置には、画像を表示または非表示する機能がある。カメラ装置で画像を取得し、表示装置で画像を表示するので、2 つのプロセスに分割する。イベントリストのレスポンスから、画像を表示するためには入力元として「カメラ」が必要である。よって、「カメラ」をターミネータとして、初版 DFD0/CFD0 に追加する。カメラ装置と表示装置を結ぶネットワークは、データを一時的に保持するデータストアとして考える。

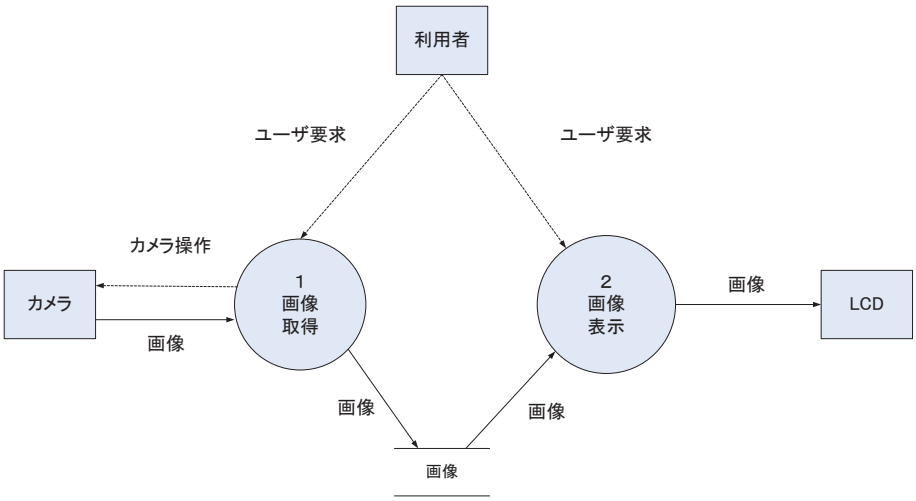


図 4.2　　初版 DFD0/CFD0

4.1.4　コンテキストダイアグラム

初版 DFD0/CFD0 を考慮して初版コンテキストダイアグラムを修正し、コンテキストダイアグラムを作成する。

初版 DFD0/CFD0 よりターミネータ「カメラ」を、初版コンテキストダイアグラムに追加する。

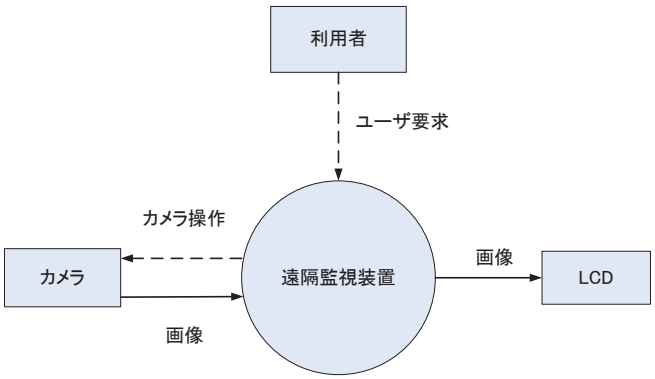


図 4.3　コンテキストダイアグラム

4.1.5　DFD0/CFD0

遠隔監視装置の主要機能は、画像の表示、非表示機能であり、この２つの機能以外に追加する機能がないので、初版 DFD0/CFD0 を変更せずに DFD0/CFD0 とする。

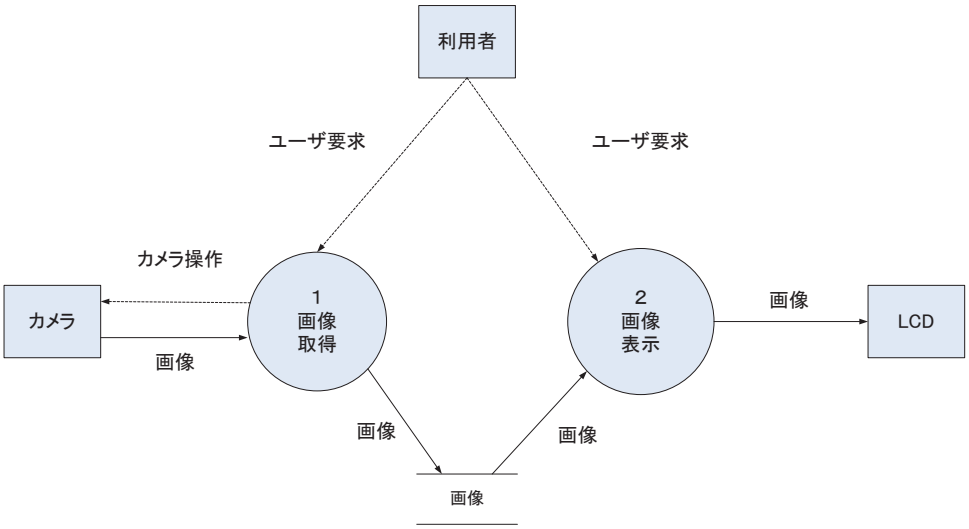


図 4.4　DFD0/CFD0

4.2　DFD/CFD の詳細化

DFD0/CFD0 の各プロセスを、より詳細なプロセスに分割する。プロセスの分割は具体的な動作仕様が見えるレベルまで行う。

DFD0/CFD0 のプロセス「1 画像取得」を詳細な機能に分割する。「利用者」からの「ユーザ要求」は、「監視開始」と「監視停止」である。「監視開始」を受け取ると、カメラ装置はカメラとネットワークを初期設定し、カメラから画像を取得して画像を表示装置に送信する。よって、カメラとネットワークを初期設定する「1.1　カメラ装置前処理」、画像の取り込みを開始する「1.2　キャプチャ開始」、画像取得と送信を繰り返す「1.3　画像処理」に分割する。

カメラから画像を取得する場合、複数のバッファを用いる。複数のバッファはバッファ識別子により管理される。カメラが画像をキャプチャする場合、バッファ識別子を入力キューに挿入し、カメラは入力キューからバッファ識別子を取り出し、キャプチャした画像をバッファ識別子に対応したバッファに格納する。画像を格納したバッファの識別子は出力キューに挿入される。アプリケーションは出力キューからバッファ識別子を取り出し、バッファに必要な処理を実行する。

「1.1 カメラ装置前処理」では、複数のバッファを確保する。「1.2　キャプチャ開始」では入力キューに複数のバッファ識別子を挿入する。「1.3　画像処理」では出力キューから画像を格納したバッファ識別子を取り出し、画像のフォーマットを変更し、表示装置に送信する。

「監視停止」を受け取ると、画像の取り込みを停止し、デバイスとネットワークを初期化前に戻す。「1.4　キャプチャ停止」は、画像の取り込み停止、「1.5　カメラ装置後処理」はデバイスをクローズ、ネットワークの後処理を実行する。

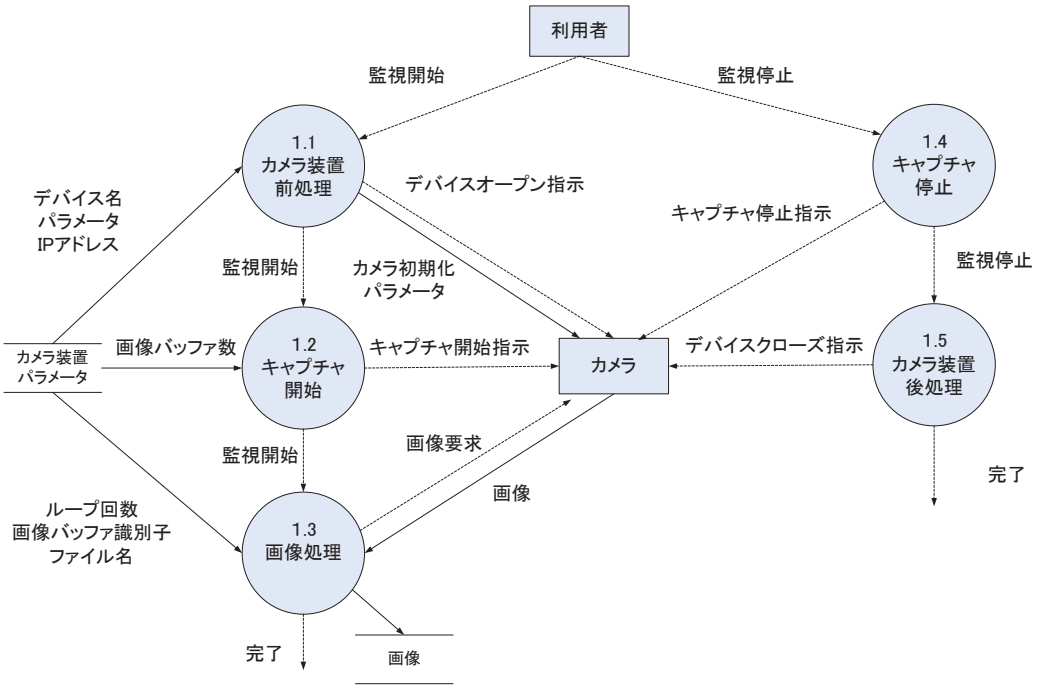


図 4.5　DFD1/CFD1

カメラ装置の初期化にはデバイスの初期化とネットワークの初期化があるので、DFD1/CFD1 の「1.1 カメラ装置前処理」をより詳細なプロセスに分割する。「1.1.1　デバイスオープン」はカメラのデバイスをオープン、「1.1.2　デバイス初期化」はデバイスを初期化し、画像を取り込む準備をする。「1.1.2　デバイス初期化」では、画像の取り込みに必要な複数のバッファを確保する。「1.1.3　カメラ装置ネットワーク初期化」はネットワークを初期設定し、画像を送信する準備をする。

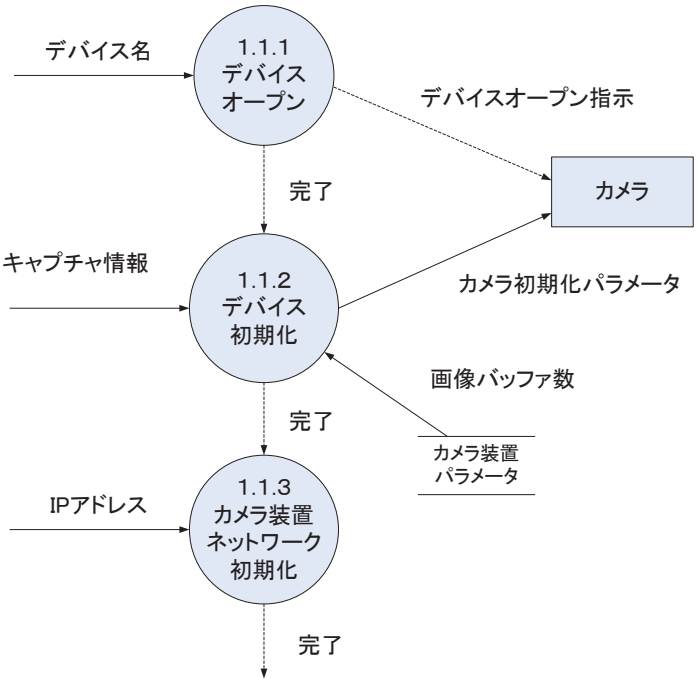


図 4.6　DFD1.1/CFD1.1

デバイスの初期化にはキャプチャに必要な情報を設定し、デバイスメモリとアプリケーションメモリのメモリマッピングが必要なので、DFD1.1/CFD1.1 の DFD1.1.2/CFD1.1.2 をさらに分割する。「1.1.2.1 画像初期化」は、画像の幅、画像の高さ、ピクセルフォーマットなどビデオキャプチャに必要な情報を設定する。「1.1.2.2 メモリマッピング初期化」はデバイスメモリとアプリケーションメモリをマッピングする。メモリマッピングを用いることにより、アプリケーションメモリから画像データを読むことができる。

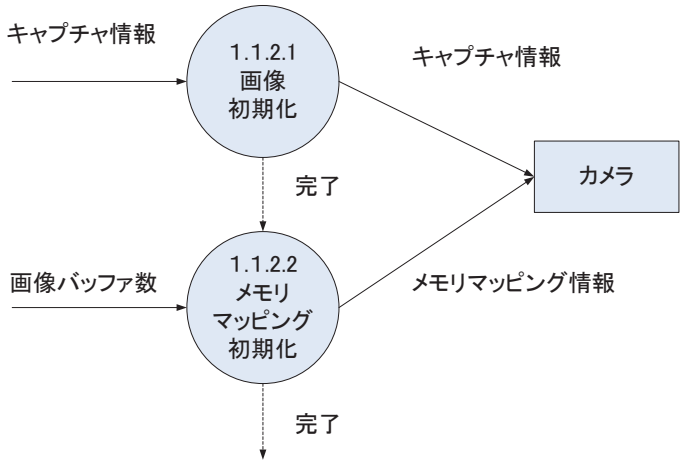


図 4.7　DFD1.1.2/CFD1.1.2

DFD1 の「1.3　画像処理」を分割する。カメラはキャプチャすると、YUV 形式の画像をバッファに格納する。YUV 形式の画像はサイズが大きいので、ネットワークでの伝送に適切でない。そこで、YUV 形式の画像を圧縮した JPEG 形式の画像に変換する。「1.3.1　画像入力」は、カメラから YUV 画像を取り込み、「1.3.2　画像変換」は YUV 画像を JPEG 画像に変換、「1.3.3　画像送信」は JPEG 画像を表示装置に送信する。「1.3.4　画像再処理準備」は、使用済みのバッファを再処理する。

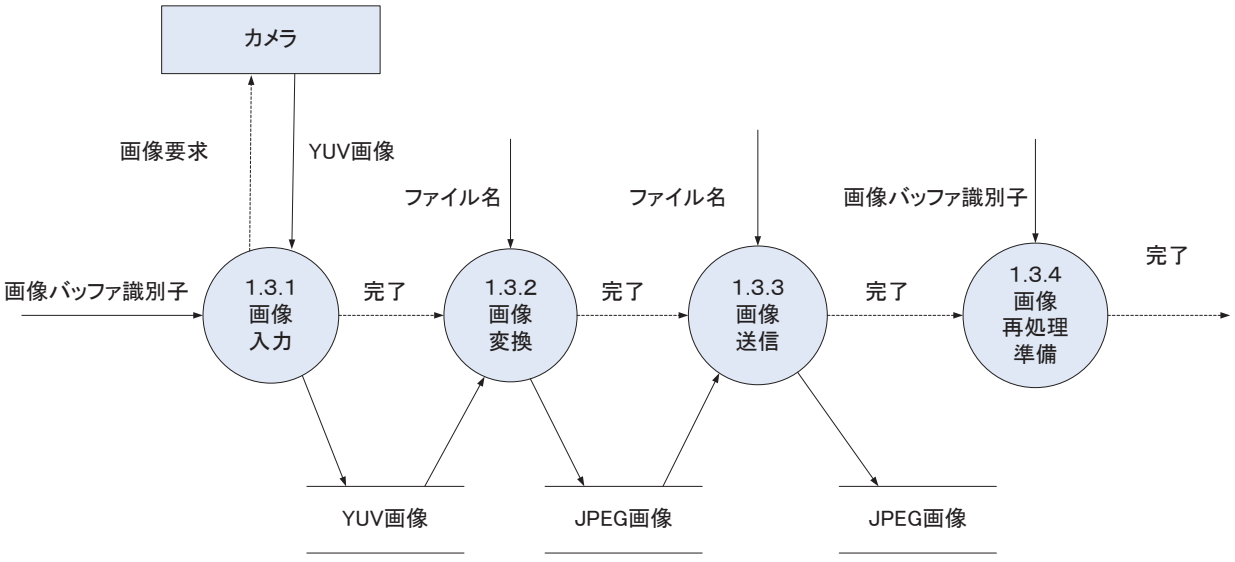


図 4.8　DFD1.3/CFD1.3

DFD1/CFD1 の「1.5　カメラ装置後処理」を分割する。監視を停止する場合、カメラ装置を初期化前の状態に戻す。「1.5.1　カメラ装置ネットワーク後処理」はネットワークを初期化前に戻し、「1.5.2 デバイス後処理」はメモリを解放、「1.5.3　デバイスクローズ」はカメラのデバイスをクローズする。

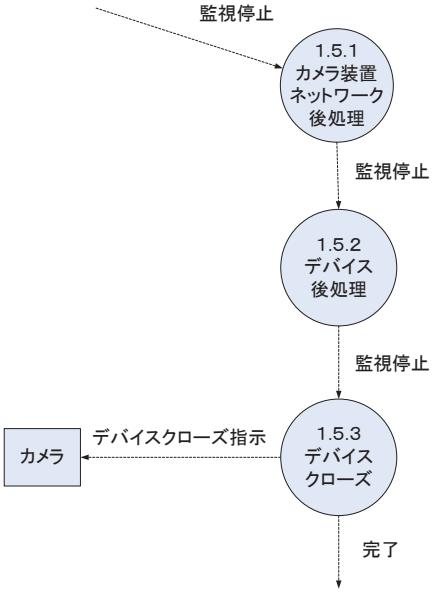


図 4.9　DFD1.5/CFD1.5

次に DFD0/CFD0 の「2 画像表示」プロセスを詳細な機能に分割する。「利用者」からの「ユーザ要求」は、「監視開始」と「監視停止」である。「監視開始」を受け取ると、表示装置は LCD とネットワークを初期設定し、ネットワークから受信した画像を LCD に表示する。よって、LCD とネットワークを初期化する「2.1　表示装置前処理」と、受信した画像を LCD に表示する「2.2　表示処理」に分割する。

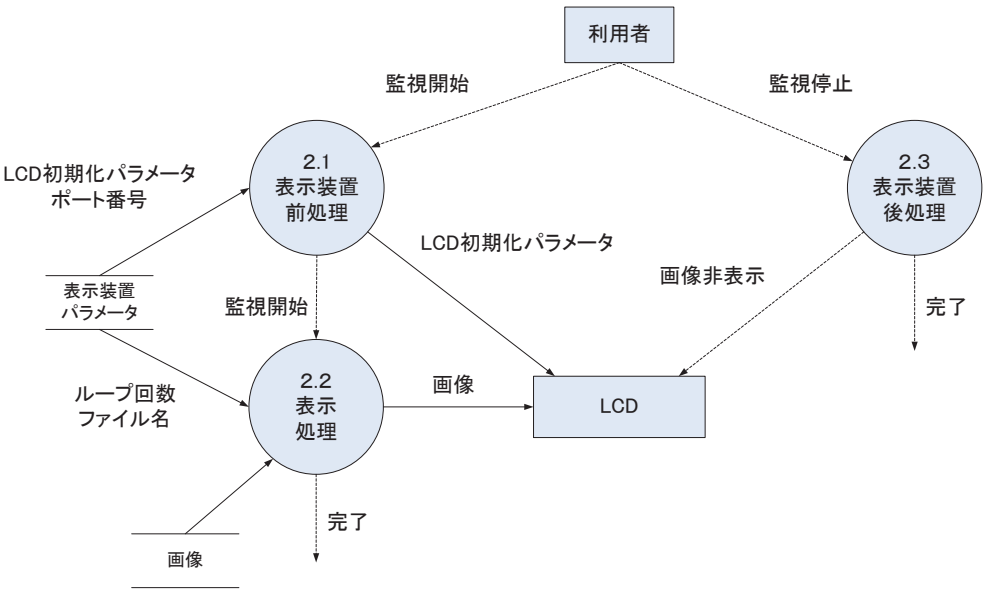


図 4.10　DFD2/CFD2

さらに「2.1　表示装置前処理」はLCDとネットワークの初期化があるので、「2.1.1　LCD初期化」と「2.1.2　表示装置ネットワーク初期化」に分割する。

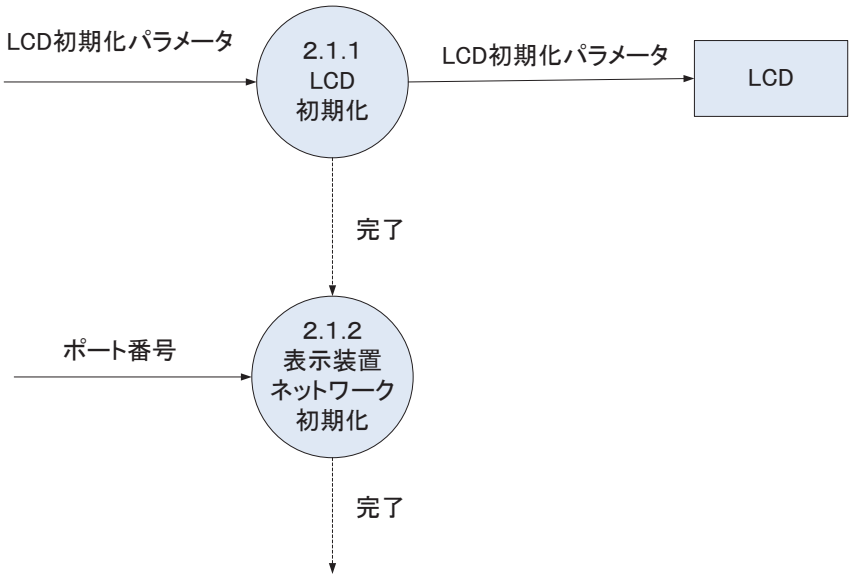


図 4.11　DFD2.1/CFD2.1

DFD2/CFD2の「2.2　表示処理」を分解する。ネットワークを介して受信する画像は、JPEG形式の画像であり、直接LCDに表示できる。「2.2.1　画像受信」はネットワークからJPEG画像を受信する。「2.2.2　画像出力」は、LCDにJPEG画像を表示する。

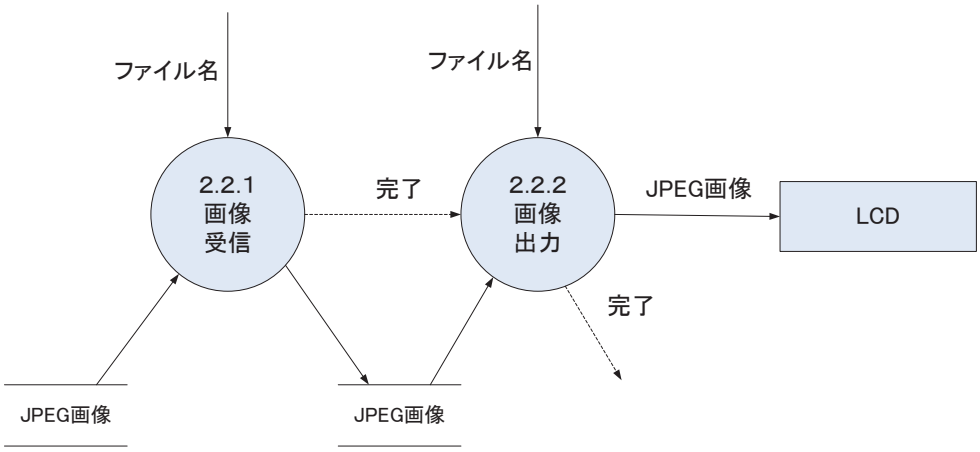


図 4.12　DFD2.2/CFD2.2

DFD2/CFD2 の「2.3　表示装置後処理」を分割する。監視を停止する場合、表示装置を初期化前の状態に戻す。よって、「2.3.1　表示装置ネットワーク後処理」は、ネットワークを初期化前に戻し、「2.3.2　LCD 後処理」は使用したオブジェクトを解放する。

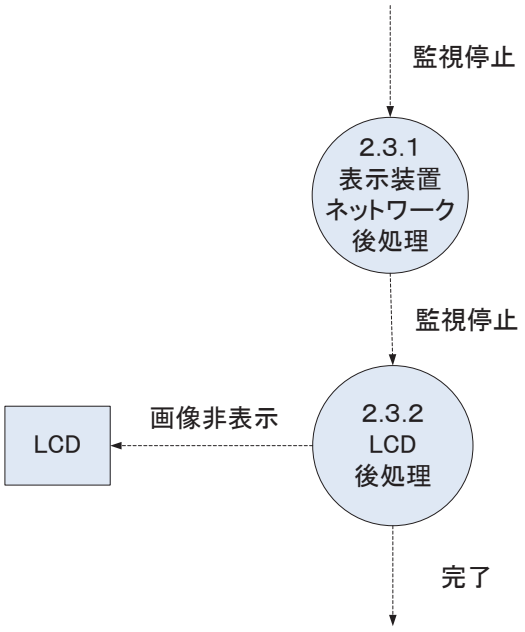


図 4.13　DFD2.3/CFD2.3

4.4　プロセス仕様書

プロセス仕様書は、DFD/CFD の末端であるリーフのプロセスに対して、入力から出力までどのように処理されるか記述する。

プロセス番号	1.1.1
プロセス名	デバイスオープン
入力	デバイス名
出力	デバイスオープン指示、完了
動作	カメラのデバイスをオープン

プロセス番号	1.1.2.1
プロセス名	画像初期化
入力	キャプチャ情報
出力	キャプチャ情報、完了
動作	ビデオフォーマットを設定

プロセス番号	1.1.2.2
プロセス名	メモリマッピング初期化
入力	画像バッファ数
出力	メモリマッピング情報、完了
動作	メモリマッピングを初期化

プロセス番号	1.1.3
プロセス名	カメラ装置ネットワーク初期化
入力	IP アドレス
出力	完了
動作	ソケットをオープン 送信バッファの領域を確保

プロセス番号	1.2
プロセス名	キャプチャ開始
入力	監視開始、画像バッファ数
出力	キャプチャ開始指示、監視開始
動作	バッファを入力キューに挿入 ストリーミングを開始

プロセス番号	1.3.1
プロセス名	画像入力
入力	YUV 画像、画像バッファ識別子
出力	画像要求、YUV 画像、完了
動作	出力キューから YUV 画像を取得

プロセス番号	1.3.2
プロセス名	画像変換
入力	YUV 画像、ファイル名
出力	JPEG 画像、完了
動作	YUV 画像を JPEG 画像に変換

プロセス番号	1.3.3
プロセス名	画像送信
入力	JPEG 画像、ファイル名
出力	JPEG 画像、完了
動作	JPEG 画像を送信

プロセス番号	1.3.4
プロセス名	画像再処理準備
入力	画像バッファ識別子
出力	完了
動作	入力キューにバッファを挿入

プロセス番号	1.4
プロセス名	キャプチャ停止
入力	監視停止
出力	キャプチャ停止指示、監視停止
動作	ストリーミングを停止

プロセス番号	1.5.1
プロセス名	カメラ装置ネットワーク後処理
入力	監視停止
出力	監視停止
動作	デバイスを初期化前に戻す

プロセス番号	1.5.2
プロセス名	デバイス後処理
入力	監視停止
出力	監視停止
動作	メモリのアンマップ

プロセス番号	1.5.3
プロセス名	デバイスクローズ
入力	監視停止
出力	デバイスクローズ指示、完了
動作	デバイスをクローズ

プロセス番号	2.1.1
プロセス名	LCD 初期化
入力	LCD 初期化パラメータ
出力	LCD 初期化パラメータ、完了
動作	LCD を初期化

プロセス番号	2.1.2
プロセス名	表示装置ネットワーク初期化
入力	ポート番号
出力	完了
動作	ソケットを生成 パート番号の結び付け 受信バッファの領域確保

プロセス番号	2.2.1
プロセス名	画像受信
入力	JPEG 画像、ファイル名
出力	JPEG 画像、完了
動作	JPEG 画像を受信 JEPG 画像をファイルに保存

プロセス番号	2.2.2
プロセス名	画像出力
入力	JPEG 画像、ファイル名
出力	JPEG 画像、完了
動作	LCD に画像を表示

プロセス番号	2.3.1
プロセス名	表示装置ネットワーク後処理
入力	監視停止
出力	監視停止
動作	受信バッファを解放

プロセス番号	2.3.2
プロセス名	LCD 後処理
入力	監視停止
出力	画像非表示、完了
動作	LCD に表示している画像を消去 オブジェクトを解放

4.5　データディクショナリ

分析モデリングで定義した用語

ユーザ要求　＝　［　監視開始　｜　監視停止　］
カメラ操作　＝　［　デバイスオープン指示　｜　デバイスクローズ指示　｜
　　キャプチャ開始指示　｜　キャプチャ停止指示　｜
　　カメラ初期化パラメータ　｜　画像要求　］
デバイスオープン指示　＝　“カメラデバイスをオープンする指示”
デバイスクローズ指示　＝　“カメラデバイスをクローズする指示”
キャプチャ開始指示　＝　“カメラにキャプチャを開始させる指示”
キャプチャ停止指示　＝　“カメラにキャプチャを停止させる指示”
カメラ初期化パラメータ　＝　［　キャプチャ情報　｜　メモリマッピング情報　］
キャプチャ情報　＝　“画像の幅、高さ、ピクセルフォーマットなどキャプチャに必要な情報”
メモリマッピング情報　＝　“要求する画像バッファ数、画像バッファのタイプなどメモリ
　　マッピングに必要な情報”
画像要求　＝　“出力キューから画像を格納したバッファを取り出す指示”
カメラ装置パラメータ　＝　［　デバイス名　｜　IP アドレス　｜　画像バッファ数　｜
　　画像バッファ識別子　｜　ループ回数　｜　ファイル名　｜
　　カメラ初期化パラメータ　］
デバイス名　＝　“カメラデバイスの名前「/dev/video0」”
IP アドレス　＝　“表示装置の IP アドレス”
画像バッファ数　＝　“キャプチャで用いるバッファの個数”
画像バッファ識別子　＝　整数：0～バッファ数-1
ループ回数　＝　“画像をキャプチャ・表示する回数”
ファイル名　＝　“JPEG 画像を一時的に保存するファイルの名前”
表示装置パラメータ　＝　［　ポート番号　｜　ループ回数　｜　ファイル名　｜
　　LCD 初期化パラメータ　］
ポート番号　＝　“TCP/IP でアプリケーションを識別するポートの番号”
LCD 初期化パラメータ　＝　［　画像の幅　｜画像の高さ　｜　通信バッファサイズ　］
画像の幅　＝　“LCD での表示画面の幅、単位はピクセル”
画像の高さ　＝　“LCD での表示画面の高さ、単位はピクセル”
通信バッファサイズ　＝　“通信で用いる送信バッファと受信バッファの大きさ”
完了　＝　“処理の正常終了を示す”

5　設計モデリング

5.1　BOSS モジュール

データフローダイアグラムをもとに、BOSS モジュールを見つける。BOSS モジュールとは、機能をまとめて制御する中心となるプロセスである。BOSS モジュールを見つける方法には、変換分析とトランザクション分析がある。変換分析はデータの源泉、変換、吸収の流れのなかで、データの変換を行うプロセスを BOSS モジュールとする方法である。トランザクション分析は、データが分岐するプロセスまたは制御バーを、BOSS モジュールとする方法である。

データフローダイアグラムから構造図を作成する場合、上位プロセスに上位モジュール、下位プロセスに下位モジュールを 1 対 1 で対応付ける。すなわち、上位プロセスが BOSS モジュールである。

データフローダイアグラムにおいて、リーフプロセス以外のプロセスは、下位プロセスをもつので BOSS モジュールである。よって、「1　画像取得」「1.1　カメラ装置前処理」「1.1.2　デバイス初期化」「1.3　画像処理」「1.5　カメラ装置後処理」「2　画像表示」「2.1　表示装置前処理」「2.2　表示処理」「2.3　表示装置後処理」が BOSS モジュールである。特に「3　画像処理」はカメラから YUV 画像を取り込み、JPEG 画像に変換し、JPEG 画像を送信しており、変換分析の観点で典型的である。

5.2　構造図

5.1 節で見つけた BOSS モジュールを中心に構造図を作成する。その際、適当なデータフローで結合できない場合は、新たに全体の動きを管理するためのモジュールを設ける。

システム全体を管理するモジュールを「遠隔監視装置」とする。「遠隔監視装置」の下位モジュールとして「1　画像取得」「2　画像表示」を配置する。同様に 5.1 節で見つけた BOSS モジュールを上位モジュールとしてすべてのプロセスを木構造で配置する。

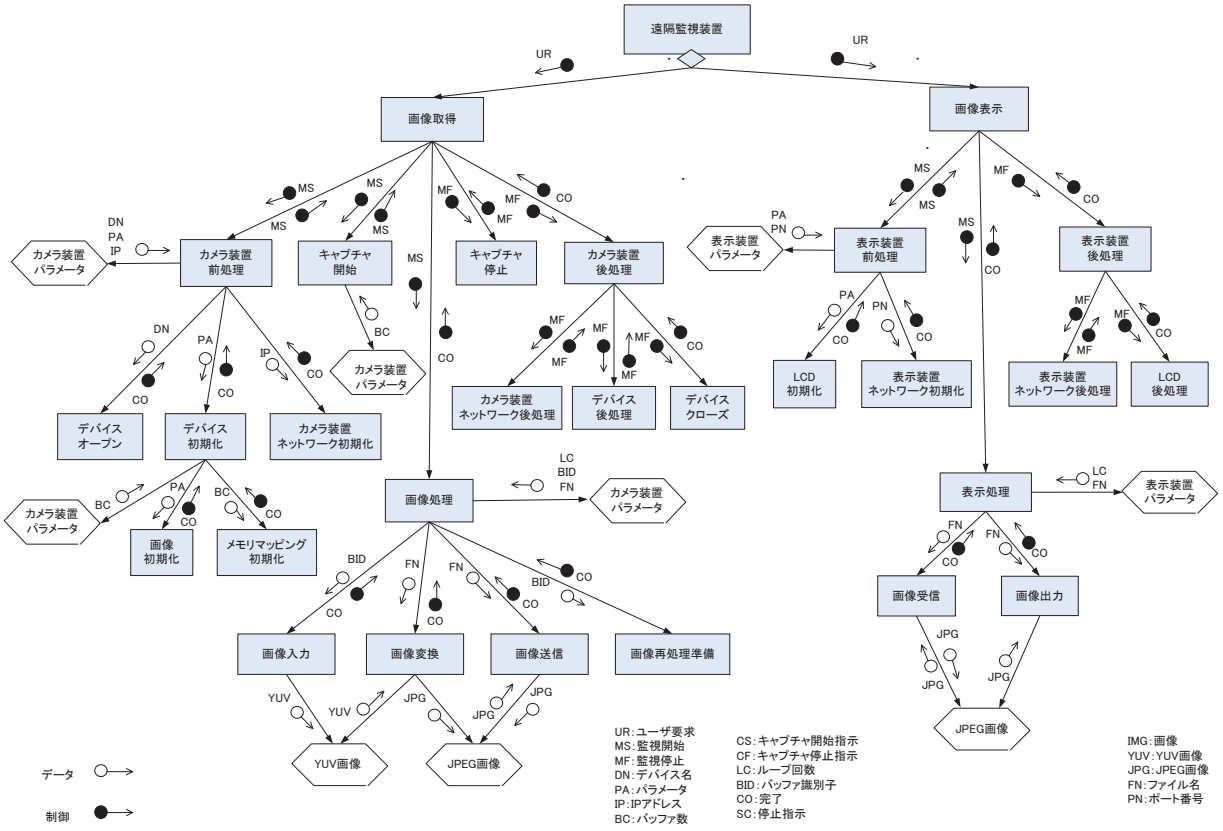


図 5.1　構造図

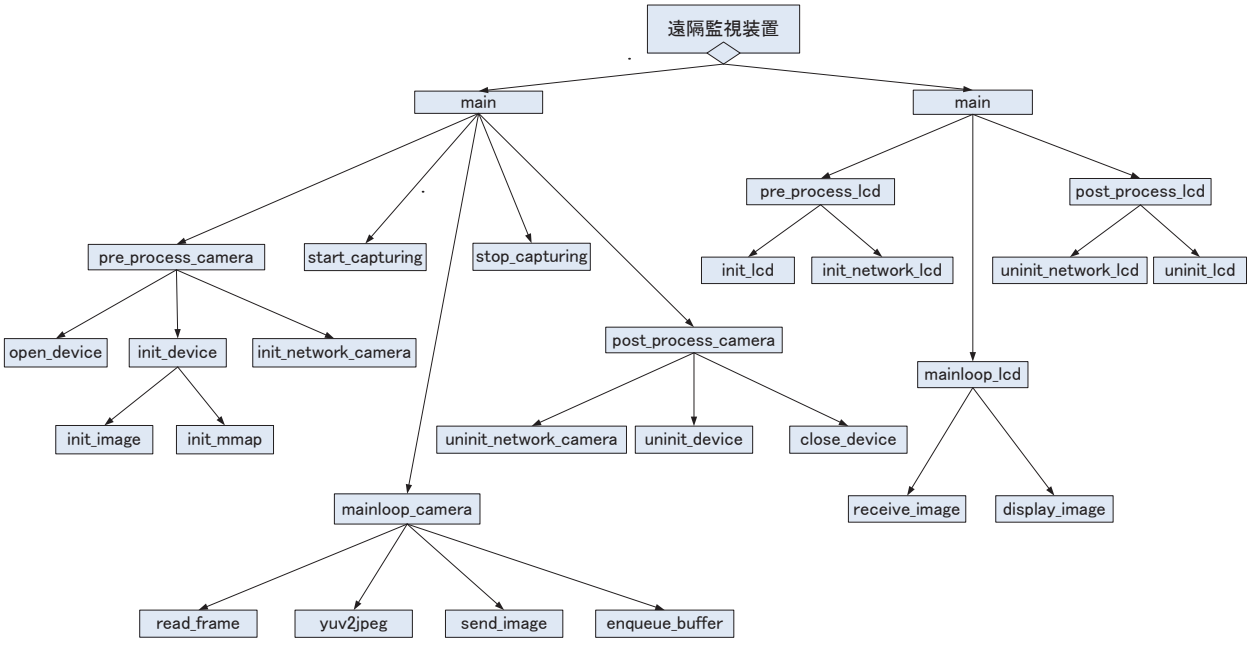


図 5.2　モジュール構造図

5.3　タスク構造の分割

図 5.1 の構造図をもとにタスクを分割する。「画像取得」と「画像表示」は、それぞれカメラ装置と表示装置で動作する。物理的に異なる装置で実行するので、「画像取得」と「画像表示」は、異なるタスクでなければならない。

5.4　モジュール仕様一覧

構造図をもとに BOSS モジュールを含めたモジュール仕様を決める。

表 3.1　モジュール一覧

ファイル	モジュール名称	目的
camera.c	main	画像取得
	pre_process_camera	カメラ装置前処理
	open_device	デバイスオープン
	init_device	デバイス初期化
	init_image	画像設定
	init_mmap	メモリマッピング初期化
	init_network_camera	カメラ装置ネットワーク初期化
	start_capturing	キャプチャ開始
	mainloop_camera	画像処理
	read_frame	画像入力
	yuv2jpeg	画像変換
	send_image	画像送信
	enqueue_buffer	画像再処理準備
	stop_capturing	キャプチャ停止
	post_process_camera	カメラ装置後処理
	uninit_network_camera	カメラ装置ネットワーク後処理
	uninit_device	デバイス後処理
	close_device	デバイスクローズ
lcd.c	main	画像表示
	pre_process_lcd	表示装置前処理
	init_lcd	LCD 初期化
	init_network_lcd	表示装置ネットワーク初期化
	mainloop_lcd	表示処理
	receive_image	画像受信
	display_image	画像出力
	post_process_lcd	表示装置後処理
	uninit_network_lcd	表示装置ネットワーク後処理
	uninit_lcd	LCD 後処理

5.5　モジュール仕様書

モジュール名称	main	
役割	画像取得	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	**名称**	**意味**
int	argc	引数の個数
char **	argv	引数の文字列へのポインタ
機能仕様		
カメラ装置の前処理（pre_process_camera） キャプチャを開始（start_capturing） 画像処理（mainloop_camera） キャプチャを停止（stop_capturing） カメラ装置の後処理（post_process_camera）		

モジュール名称	pre_process_camera	
役割	カメラ装置前処理	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	**名称**	**意味**
char *	dev_name	USB カメラのデバイス名
char *	ipaddr	表示装置の IP アドレス
unsigned int	req_buffers	バッファの要求数
機能仕様		
USB カメラのデバイスをオープン（open_device） デバイスを初期化（init_device） ネットワークを初期化（init_network_camera）		

モジュール名称	open_device	
役割	デバイスオープン	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
char *	dev_name	デバイス名
機能仕様		
USB カメラのデバイスをオープン（open）		

モジュール名称	init_device	
役割	デバイス初期化	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
unsigned int	req_buffers	入力バッファ（incoming queue）に入れるバッファ数
機能仕様		
ビデオフォーマットを設定 （init_image） メモリマッピングを初期化（init_mmap）		

モジュール名称	init_image	
役割	画像設定	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
Void		
機能仕様		
ビデオフォーマットを設定		

モジュール名称	init_mmap	
役割	メモリマッピング初期化	
返り値の型	int	
返り値の意味	割り当てできたバッファ数	
引数		
型	名称	意味
unsigned int	req_buffers	入力バッファ（incoming queue）に入れるバッファ数
機能仕様		
メモリマッピングで用いるメモリ領域を確保（calloc） 要求されたバッファ数だけバッファを確保 デバイスメモリとのマッピング（mmap）		

モジュール名称	init_network_camara	
役割	カメラ装置ネットワーク初期化	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
char *	ipaddr	表示装置の IP アドレス
機能仕様		
ソケットをオープン（socket） 送信バッファの領域を確保（malloc）		

モジュール名称	start_capturing	
役割	キャプチャ開始	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
void		
機能仕様		
バッファを入力キュー（incoming queue）に挿入 ストリーミングを開始		

モジュール名称	mainloop_camera	
役割	画像処理	
返り値の型	int	
返り値の意味	繰り返し回数	
引数		
型	名称	意味
unsigned int	loop_count	画像処理の繰り返し回数
機能仕様		

while（画像処理の繰り返し回数）{
　タイムアウト処理
　YUV 画像を 1 フレーム読む（read_frame）
　YUV 画像を JPEG 画像に変換（yuv2jpeg）
　JPEG 画像を送信（send_image）
　スリープ（sleep）
　入力キューにバッファを挿入（enqueue_buffer）
}		

モジュール名称	read_frame	
役割	画像入力	
返り値の型	int	
返り値の意味	バッファの識別子	
引数		
型	名称	意味
unsigned int	req_buffers	incoming queue に入れるバッファ数
機能仕様		
出力キュー（outgoing queue）から YUV 画像 1 フレーム分のバッファ取り出す バッファ識別子をチェック（assert）		

モジュール名称	yuv2jpeg	
役割	画像変換	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
char *	yuv_image	YUV 画像を指すポインタ
char *	jpegfile	JPEG 画像のファイル名
機能仕様		
JPEG 画像のファイルをオープン（fopen） JPEG オブジェクトを生成（jpeg_create_compress） エラーハンドラにデフォルト値を設定（jpeg_std_error） 出力ファイルを指定（jpeg_stdio_dest） 画像パラメータを設定（jpeg_set_quality） 圧縮開始（jpeg_start_compress） 作業用のメモリ領域を確保（malloc） YUV422 形式を YUV444 形式に変換 ファイルに出力（jpeg_write_scanlines） 圧縮終了（jpeg_finish_compress） JPEG オブジェクトを破棄（jpeg_destroy_compress） メモリ領域を解放（free） ファイルをクローズ（fclose）		

モジュール名称	send_image	
役割	画像送信	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
char *	filename	JPEG 画像のファイル名
機能仕様		
JPEG 画像ファイルをオープン（fopen） ファイル属性を得る（stat） ファイルを読む（fread） JPEG 画像ファイルを送信（sendto）		

モジュール名称	enqueue_buffer	
役割	画像再処理準備	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
int	buf_index	バッファの識別子
機能仕様		
入力キューにバッファを挿入		

モジュール名称	stop_capturing	
役割	キャプチャ停止	
返り値の型	void	
返り値の意味		
引数		
型	名称	意味
void		
機能仕様		
ストリーミングを停止		

モジュール名称	post_process_camera	
役割	カメラ装置後処理	
返り値の型	void	
返り値の意味		
引数		
型	名称	意味
void		
機能仕様		
ネットワークを初期化前に戻す（uninit_network_camera） デバイスを初期化前の状態に戻す（uninit_device） デバイスをクローズ（close_device）		

モジュール名称	uninit_network_camera	
役割	カメラ装置ネットワーク後処理	
返り値の型	void	
返り値の意味		
引数		
型	名称	意味
void		
機能仕様		
バッファ領域を解放（free） ソケットをクローズ（close）		

モジュール名称	uninit_device	
役割	デバイス後処理	
返り値の型	void	
返り値の意味		
引数		
型	名称	意味
void		
機能仕様		
メモリのアンマップ（munmap） バッファ領域を解放（free）		

モジュール名称	close_device	
役割	デバイスクローズ	
返り値の型	void	
返り値の意味		
引数		
型	名称	意味
void		
機能仕様		
USB カメラデバイスをクローズ（close）		

モジュール名称	main	
役割	画像表示	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
int	argc	引数の個数
char **	argv	引数の文字列へのポインタ
機能仕様		
表示装置の前処理（pre_process_lcd） 表示処理（mainloop_lcd） 表示装置の後処理（post_process_lcd）		

モジュール名称	pre_process_lcd	
役割	表示装置前処理	
返り値の型	int	
返り値の意味	成功	
引数		
型	名称	意味
int	ac	引数の個数（argc）
char **	av	引数の文字列へのポインタ（argv）
機能仕様		
LCD を初期化（init_lcd） ネットワークを初期化（init_network_lcd）		

モジュール名称	init_lcd	
役割	LCD 初期化	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
int	ac	引数の個数（argc）
char **	av	引数の文字列へのポインタ（argv）
機能仕様		
DirectFB を初期化（DirectFBInit） スーパインタフェースを生成（DirectFBCreate） フルスクリーンを設定（SetCooperativeLevel） プライマリサーフェイスを生成（CreateSurface） 解像度を取得（GetSize） バックカラーを設定（SetColor, FillRectangle）		

モジュール名称	init_network_lcd	
役割	表示装置ネットワーク初期化	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
int	port_number	表示装置のポート番号
機能仕様		
ソケットを生成（socket） ポート番号の結び付け（bind） 受信バッファの領域確保（malloc）		

モジュール名称	mainloop_lcd	
役割	表示処理	
返り値の型	int	
返り値の意味	繰り返し回数	
引数		
型	名称	意味
unsigned int	loop_count	表示処理の繰り返し回数
機能仕様		

while（画像処理の繰り返し回数）{
　画像を受信（receive_image）
　画像を表示（display_jpeg）
}		

モジュール名称	receive_image	
役割	画像受信	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
char *	filename	ファイル名
機能仕様		
JPEG 画像を受信（recvfrom） ファイルをオープン（fopen） JPEG 画像をファイルに書く（fwrite） ファイルをクローズ（fclose）		

モジュール名称	display_image	
役割	画像出力	
返り値の型	int	
返り値の意味	0：成功	
引数		
型	名称	意味
char *	filename	ファイル名
機能仕様		
画像プロバイダを生成（CreateImageProvider） 画像ファイルを変換（RenderTo） ちらつき防止（Flip）		

モジュール名称	post_process_lcd	
役割	表示装置後処理	
返り値の型	void	
返り値の意味		
引数		
型	名称	意味
void		
機能仕様		
ネットワークを初期化前の状態に戻す（uninit_network_lcd） LCD を初期化前に戻す（uninit_lcd）		

モジュール名称	uninit_network_lcd	
役割	表示装置ネットワーク後処理	
返り値の型	void	
返り値の意味		
引数		
型	名称	意味
void		
機能仕様		
受信バッファ領域を解放（free） ソケットをクローズ（close）		

モジュール名称	uninit_lcd	
役割	LCD 後処理	
返り値の型	void	
返り値の意味		
引数		
型	名称	意味
void		
機能仕様		
プロバイダを解放（Release） プライマリサーフェイスを解放（Release） DirectFB を解放（Release）		

6　プログラム

モジュール仕様書をもとにカメラ装置プログラムと表示装置プログラムを作成する。エラー処理が頻繁に必要なので 2 個の関数と 1 個のマクロを追加する。カメラ装置プログラムでは、デバイストライバを制御するシステムコール「ioctl」にエラー処理を付加した関数「xioctl」と、関数「exit」にエラーメッセージの出力を付加した関数「errno_exit」を作成する。表示装置プログラムでは、DirectFB からのエラーメッセージを表示するマクロ「DFBCHECK」を作成する。

6.1　カメラ装置のプログラム

表 6.1　カメラ装置プログラム

プログラム	ファイル名
メイクファイル	Makefile
共通ヘッダファイル	remote-monitor.h
カメラ装置プログラムヘッダファイル	camera.h
カメラ装置プログラム	camera.c

6.1.1　メイクファイル (Makefile)

```
------------------------------------------------------------------------------
CC = sh4-linux-gcc

export SYSROOT=$(shell readlink -f `$(CC) -print-prog-name=gcc` | sed -e
s!/usr/sh4-linux-uclibc/.*!!)

CFLAGS = -Wall `sh4-linux-directfb-config --cflags`
LDFLAGS = `sh4-linux-directfb-config --libs` -lasound -ljpeg

OBJS = camera.o
PROGRAM = camera


all: $(PROGRAM)
    cp $(PROGRAM) /nfs

$(PROGRAM): $(OBJS)

taa:
    @echo $(SYSROOT)

clean:
    $(RM) $(OBJS) $(PROGRAM)
------------------------------------------------------------------------------
```

6.1.2　共通ヘッダファイル（remote-monitor.h）

```
/*
 *     遠隔監視装置　共通ヘッダ（remote-monitor.h）
 *
 *     2012/12/7
 */

#define IMG_WIDTH    320
#define IMG_HEIGHT   240

#define LCD_PORT     9877
#define SR_BUF_SIZE 100000
#define OK           1
```

6.1.3　カメラ装置プログラムヘッダファイル（camera.h）

```
/*
 *     遠隔監視装置　カメラ装置ヘッダファイル（camera.h）
 *
 *     2012/12/7
 */

#define CLEAR(x) memset(&(x), 0, sizeof (x))

struct buffer {
    void    *start;
    size_t   length;
};

static int pre_process_camera(char *dev_name, char *ipaddr, unsigned int req_buffers);
static int open_device(char *dev_name);
static int init_device(unsigned int   req_buffers);
static int init_mmap(unsigned int req_buffers);
static int init_network_camera(char *ipaddr);
static int start_capturing(void);
static void mainloop_camera(unsigned int loop_count);
static int read_frame(void);
static int yuv2jpeg(char *yuv_image, char *jpegfile);
static int send_image(char *filename);
static int enqueue_buffer(int buf_index);
static void stop_capturing(void);
static void post_process_camera(void);
static void uninit_network_camera(void);
static void uninit_device(void);
static void close_device(void);
static void errno_exit(const char *s);
static int xioctl(int fd, int request, void * arg);
```

6.1.4　カメラ装置プログラム（camera.c）

```
/*
 *     遠隔監視装置　カメラ装置プログラム（camera.c）
 *
 *     2012/12/7
 */

#include    <stdio.h>
#include    <stdlib.h>
#include    <string.h>
#include    <assert.h>
#include    <getopt.h>              /* getopt_long() */
#include    <fcntl.h>               /* low-level i/o */
#include    <unistd.h>
#include    <errno.h>
#include    <malloc.h>
#include    <time.h>
#include    <sys/stat.h>
#include    <sys/types.h>
#include    <sys/time.h>
#include    <sys/mman.h>
#include    <sys/ioctl.h>
#include    <asm/types.h>           /* for videodev2.h */
#include    <linux/videodev2.h>
#include    <sys/socket.h>
#include    <netinet/in.h>
#include    <arpa/inet.h>
#include    <unistd.h>
#include    <jpeglib.h>
#include    "remote-monitor.h"
#include    "camera.h"

int fd                  = -1;       // デバイスのハンドラ
int sockfd              = 0;        // ソケットハンドラ
unsigned int n_buffers  = 0;        // 割り当てできたバッファ数
struct buffer *buffers  = NULL;     // バッファへの先頭アドレス
unsigned char *send_buf;            // 送信バッファへのポインタ
struct sockaddr_in   servaddr;

int main(int argc, char **argv)
{
    char    dev_name[] = "/dev/video0";  // デバイス名
    charipaddr[] = "xx.xx.xx.xx";        // 表示装置の ip アドレス
    unsigned int req_buffers  = 4;       // 要求バッファ数
    unsigned int loop_count   = 100;     // 画像の取得回数

    pre_process_camera(dev_name, ipaddr, req_buffers);// カメラ装置の前処理
    start_capturing();                   //キャプチャを開始
    mainloop_camera(loop_count);         //  画像処理
    stop_capturing();                    //  キャプチャを停止
    post_process_camera();               //  カメラ装置の後処理

    return 0;
}
static int   pre_process_camera(char *dev_name, char *ipaddr, unsigned int req_buffers)
{
    open_device(dev_name);           //  USB カメラのデバイスをオープン
    init_device(req_buffers);        //  デバイスを初期化
```

```
    init_network_camera(ipaddr);      //  ネットワークを初期化

    return OK;
}

static int open_device(char *dev_name)
{
    if ((fd = open(dev_name, O_RDWR | O_NONBLOCK, 0)) == -1) { // USB カメラをオープン
        fprintf(stderr, "Cannot open %s¥n", dev_name);
        exit(EXIT_FAILURE);
    }
    return OK;
}

static int init_device(unsigned int   req_buffers)
{
    struct v4l2_format fmt;

    CLEAR(fmt);
    fmt.type                = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    fmt.fmt.pix.width       = IMG_WIDTH;     // キャプチャ画像の解像度：幅
    fmt.fmt.pix.height      = IMG_HEIGHT;    // キャプチャ画像の解像度：高さ
    fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_YUYV; // ピクセルフォーマットの設定：YUYV
    fmt.fmt.pix.field       = V4L2_FIELD_INTERLACED;  // 受信するビデオフォーマットの設定
    xioctl(fd, VIDIOC_S_FMT, &fmt);           // ビデオフォーマットを設定

    n_buffers = init_mmap(req_buffers);       // メモリマッピングを初期化
    if (n_buffers < req_buffers) {
        fprintf(stderr, "less buffers¥n");
        exit(EXIT_FAILURE);
    }
    return OK;
}

static int init_mmap(unsigned int req_buffers)
{
    struct v4l2_requestbuffers req;

    CLEAR (req);
    req.count    = req_buffers;                    // 要求バッファ数
    req.type= V4L2_BUF_TYPE_VIDEO_CAPTURE;         // バッファタイプ
    req.memory   = V4L2_MEMORY_MMAP;               // MMAP を使用
    xioctl(fd, VIDIOC_REQBUFS, &req);              // メモリマッピングの設定

            // メモリマッピングで用いるメモリ領域を確保
    buffers = calloc(req.count, sizeof(*buffers));
    if (buffers == NULL) {
        fprintf(stderr, "memory allocation error¥n");
        exit(EXIT_FAILURE);
    }
                            // 要求されたバッファ数だけバッファを確保
    for (n_buffers = 0; n_buffers < req.count; ++n_buffers) {
        struct v4l2_buffer buf;

        CLEAR(buf);
        buf.type         = V4L2_BUF_TYPE_VIDEO_CAPTURE;
        buf.memory       = V4L2_MEMORY_MMAP;
        buf.index        = n_buffers;
        xioctl(fd, VIDIOC_QUERYBUF, &buf);
```

```
            buffers[n_buffers].length = buf.length;
                                //  デバイスメモリとのマッピング
            buffers[n_buffers].start = mmap(NULL, // start anywhere
                buf.length,                       // バッファの長さ
                PROT_READ | PROT_WRITE,           // ポートの read/write
                MAP_SHARED,                       // 共有を指定
                fd, buf.m.offset);
        }
        return n_buffers;
    }

    static int   init_network_camera(char *ipaddr)
    {
        memset(&servaddr, 0x00, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(LCD_PORT);
        inet_pton(AF_INET, ipaddr, &servaddr.sin_addr);

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);          // ソケットをオープン
        if (sockfd == -1) {
            fprintf(stderr, "can't socket open ¥n");
            exit(EXIT_FAILURE);
        }
        send_buf = (unsigned char *)malloc(SR_BUF_SIZE);  // 送信バッファ領域を確保
        if (send_buf == NULL) {
            fprintf(stderr, "memory allocation error¥n");
            exit(EXIT_FAILURE);
        }
        return OK;
    }

    static int start_capturing(void)
    {
        unsigned int i;
        enum v4l2_buf_type type;

        for (i = 0; i < n_buffers; ++i) {
            struct v4l2_buffer buf;

            CLEAR(buf);
            buf.type        = V4L2_BUF_TYPE_VIDEO_CAPTURE;
            buf.memory      = V4L2_MEMORY_MMAP;
            buf.index       = i;          // バッファの識別子
            xioctl(fd, VIDIOC_QBUF, &buf);// バッファを入力キュー (incoming queue) に挿入
        }
        type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
        xioctl(fd, VIDIOC_STREAMON, &type);   //  ストリーミングを開始
        return OK;
    }

    static void mainloop_camera(unsigned int loop_count)
    {
        char *filename = "temp.jpeg";
    //  time_t t;
        int buf_index;

    //  printf("start time : %d¥n", time(&t));
        while (loop_count-- > 0) {
            for (;;) {
                fd_set fds;
```

```
            struct timeval tv;
            int r;

            FD_ZERO(&fds);
            FD_SET(fd, &fds);
            tv.tv_sec = 5;                              // タイムアウト時間を設定
            tv.tv_usec = 0;
            r = select(fd+1, &fds, NULL, NULL, &tv);  // USB カメラからのイベント待ち

            if (r == -1) {                              // エラー処理
                if (EINTR == errno)
                    continue;
                errno_exit("select");
            }
            if (r == 0) {                               // タイムアウト処理
                fprintf (stderr, "select timeout¥n");
                exit (EXIT_FAILURE);
            }
            if ((buf_index = read_frame()) >= 0){     // YUV 画像を 1 フレーム読む
                printf(".¥n");
                break;
            } else {
                printf("EAGAIN¥n");
            }
            // EAGAIN : outgoing queue が空のとき繰り返す
        }
//      printf("buf_index = %d¥n", buf_index);
        yuv2jpeg((char *)buffers[buf_index].start, filename);  // YUV 画像を JPEG 画像に変換
        send_image(filename);                                  // JPEG 画像を送信
//      usleep(600000);
        sleep(1);                                              // 待ち
        enqueue_buffer(buf_index);                             // 入力キューにバッファを挿入
    }
//  printf("finish time : %d¥n", time(&t));
}

static int read_frame(void)
{
    struct v4l2_buffer buf;

    CLEAR (buf);
    buf.type      = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    buf.memory = V4L2_MEMORY_MMAP;

            // 出力キュー (outgoing queue) から YUV 画像 1 フレーム分のバッファを取り出す
    if (xioctl(fd, VIDIOC_DQBUF, &buf) == -1) {
        switch (errno) {
        case EAGAIN:                            // 出力キューが空のとき
            return -1;
        default:
            errno_exit("VIDIOC_DQBUF");
        }
    }
    assert(buf.index < n_buffers);              // バッファ識別子をチェック、false ならば停止
    return buf.index;
}

static int yuv2jpeg(char *yuv_image, char *jpegfile)
{
                // JPEG オブジェクト, エラーハンドラを確保
```

```
    struct jpeg_compress_struct cinfo;
    struct jpeg_error_mgr jerr;
    FILE    *fp_jpeg;
    int i, j, width_length;
    JSAMPLE Y1, Y2, U, V;

    fp_jpeg = fopen(jpegfile, "wb");      // JPEG 画像のファイルをオープン
    if (fp_jpeg == NULL) {
        fprintf(stderr, "cannot open %s¥n", jpegfile);
        exit(EXIT_FAILURE);
    }

   cinfo.err = jpeg_std_error(&jerr);     // エラーハンドラにデフォルト値を設定
   jpeg_create_compress(&cinfo);          // JPEG オブジェクトを生成

   jpeg_stdio_dest(&cinfo, fp_jpeg);      // 出力ファイルを設定

   cinfo.image_width = IMG_WIDTH;         // 画像パラメータを設定
   cinfo.image_height = IMG_HEIGHT;
   cinfo.input_components = 3;
   cinfo.in_color_space = JCS_YCbCr;
   jpeg_set_defaults(&cinfo);
   jpeg_set_quality(&cinfo, 100, TRUE);

   jpeg_start_compress(&cinfo, TRUE);     // 圧縮を開始

                                          // YUV422 (YUYV) を YUV444 に変換
    width_length = IMG_WIDTH * 3;

                                          // 作業用のメモリ領域を確保
    JSAMPARRAY img = (JSAMPARRAY)malloc(sizeof(JSAMPROW) * IMG_HEIGHT);
    for (i = 0; i < IMG_HEIGHT; i++) {
       img[i] = (JSAMPROW) malloc(sizeof(JSAMPLE) * width_length);// 作業用のメモリ領域を確保
       for (j = 0; j < width_length; j+=6) {
            Y1 = *yuv_image++;
            U  = *yuv_image++;
            Y2 = *yuv_image++;
            V  = *yuv_image++;
          img[i][j + 0] = Y1;
          img[i][j + 1] = U;
          img[i][j + 2] = V;
          img[i][j + 3] = Y2;
          img[i][j + 4] = U;
          img[i][j + 5] = V;
        }
    }

    jpeg_write_scanlines(&cinfo, img, IMG_HEIGHT);    // ファイルに出力

    jpeg_finish_compress(&cinfo);                     // 圧縮終了

    jpeg_destroy_compress(&cinfo);                    // JPEG オブジェクトを破棄

    for (i = 0; i < IMG_HEIGHT; i++) {                // メモリ領域を開放
       free(img[i]);
    }
    free(img);

    fclose(fp_jpeg);                                  // ファイルをクローズ
```

```
    return OK;
}

static int   send_image(char *filename)
{
    FILE    *fp;
    struct stat st;

    fp = (FILE *)fopen(filename, "rb");                // JPEG ファイルをオープン
    if (fp == NULL) {
    fprintf(stderr, "cannot open %s¥n", filename);
        exit(EXIT_FAILURE);
    }
    stat(filename, &st);                               // ファイル属性を得る
    fread(send_buf, 1, st.st_size, fp);                // ファイルを読む
    sendto(sockfd, send_buf, (int)st.st_size, 0,
            (struct sockaddr *)&servaddr,
            sizeof(servaddr));                         // JPEG ファイルを送信
    printf("send image = %d¥n", (int)st.st_size);
    return OK;
}

static int enqueue_buffer(int buf_index)
{
    struct v4l2_buffer buf;

    CLEAR (buf);
    buf.type   = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    buf.memory = V4L2_MEMORY_MMAP;
    buf.index  = buf_index;              // バッファの識別子
    xioctl(fd, VIDIOC_QBUF, &buf);       // 入力キュー (incoming queue) にバッファを挿入
    return OK;
}

static void stop_capturing(void)
{
    enum v4l2_buf_type type;

    type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    xioctl(fd, VIDIOC_STREAMOFF, &type); // ストリーミングを停止
}

static void  post_process_camera(void)
{
    uninit_device();                     // デバイスを初期化前の状態に戻す
    close_device();                      // デバイスをクローズ
}

static void uninit_network_camera(void)
{
    free(send_buf);                      // バッファ領域を開放
    close(sockfd);                       // ソケットをクローズ
}

static void uninit_device(void)
{
    unsigned int i;

    for (i = 0; i < n_buffers; ++i)   {   // メモリのアンマップ
        munmap(buffers[i].start, buffers[i].length);
```

```
        }
        free(buffers);                          // バッファ領域を開放
    }

    static void close_device(void)
    {
        close(fd);                              // USB カメラデバイスをクローズ
    }

    static void errno_exit(const char *s)
    {
        fprintf(stderr,"%s error %d, %s¥n", s, errno, strerror (errno));
        exit (EXIT_FAILURE);
    }

    static int xioctl(int fd, int request, void *arg)
    {
        int r;

        do r = ioctl(fd, request, arg);
        while (-1 == r && EINTR == errno);

        return r;
    }
```

6.2　表示装置のプログラム

表 6.2　表示装置プログラム

プログラム	ファイル名
メイクファイル	Makefile
共通ヘッダファイル	remote-monitor.h
表示装置プログラムヘッダファイル	lcd.h
表示装置プログラム	lcd.c

6.2.1　メイクファイル（Makefile）

```
-------------------------------------------------------------------------------
CC = sh4-linux-gcc

export SYSROOT=$(shell readlink -f `$(CC) -print-prog-name=gcc` | sed -e
s!/usr/sh4-linux-uclibc/.*!!)

CFLAGS = -Wall `sh4-linux-directfb-config --cflags`
LDFLAGS = `sh4-linux-directfb-config --libs` -lasound -ljpeg

OBJS = lcd.o
PROGRAM = lcd


all: $(PROGRAM)
    cp $(PROGRAM) /nfs

$(PROGRAM): $(OBJS)

taa:
    @echo $(SYSROOT)

clean:
    $(RM) $(OBJS) $(PROGRAM)
-------------------------------------------------------------------------------
```

6.2.2　共通ヘッダファイル（remote-monitor.h）

```
-------------------------------------------------------------------------------
/*
 *     遠隔監視装置　共通ヘッダ（remote-monitor.h）
 *
 *     2012/12/7
 */

#define IMG_WIDTH    320
#define IMG_HEIGHT   240

#define LCD_PORT     9877
#define SR_BUF_SIZE 100000
#define OK           1
-------------------------------------------------------------------------------
```

6.2.3　表示装置プログラムヘッダファイル（lcd.h）

```
/*
 *     遠隔監視装置　表示装置ヘッダファイル（lcd.h）
 *
 *     2012/12/7
 */

    // DirectFB から呼び出されるエラーチェック用マクロ
#define DFBCHECK(x...)                                                  ¥
{                                                                       ¥
    DFBResult err = x;                                                  ¥
                                                                        ¥
    f (err != DFB_OK)                                                   ¥
    {                                                                   ¥
        fprintf( stderr, "%s <%d>:¥n¥t", __FILE__, __LINE__ );          ¥
        directFBErrorFatal( #x, err );                                  ¥
    }                                                                   ¥
}                                                                       ¥

static int   pre_process_lcd(int ac, char **av);
static int   init_lcd(int ac, char **av);
static int   init_network_lcd(int port_number);
static int   mainloop_lcd(unsigned int loop_count);
static int   receive_image(char *filename);
static int   display_image(char *filename);
static void  post_process_lcd(void);
static void  uninit_network_lcd(void);
static void  uninit_lcd(void);
```

6.2.4　表示装置プログラム（lcd.c）

```
/*
 *     遠隔監視装置　表示装置プログラム（lcd.c）
 *
 *     2012/12/7
 */

#include    <stdio.h>
#include    <stdlib.h>
#include    <string.h>
#include    <sys/socket.h>
#include    <netinet/in.h>
#include    <arpa/inet.h>
#include    <unistd.h>
#include    <directfb.h>
#include    "remote-monitor.h"
#include    "lcd.h"

static IDirectFB         *dfb = NULL;     // スーパサーフェイス
static IDirectFBSurface  *primary = NULL; // プライマリ・サーフェイス
DFBSurfaceDescription    dsc;             // サーフェイス
IDirectFBImageProvider   *provider;       // 画像読み込み用イメージプロバイダ.
static int screen_width  = 0;             // 画面の幅
static int screen_height = 0;             // 画面の高さ

static int   sockfd = 0;                  // ソケットハンドラ
char    *receive_buf;                     // 受信バッファへのポインタ

int main(int argc, char **argv)
{
    unsigned int loop_count = 100;

    pre_process_lcd(argc, argv);          // 表示装置の前処理
    mainloop_lcd(loop_count);             // 表示処理
    post_process_lcd();                   // 表示装置の後処理

    return OK;
}

static int   pre_process_lcd(int ac, char **av)
{
    init_lcd(ac, av);                         // LCD を初期化
    init_network_lcd(LCD_PORT);               // ネットワークを初期化
    return OK;
}

static int   init_lcd(int ac, char **av)
{
    DFBCHECK(DirectFBInit(&ac, &av));             // DirectFB を初期化
    DFBCHECK(DirectFBCreate(&dfb));               // スーパインターフェイスを生成

    DFBCHECK (dfb->SetCooperativeLevel(dfb, DFSCL_FULLSCREEN));    // フルスクリーンを指定

    dsc.flags = DSDESC_CAPS;
    dsc.caps  = DSCAPS_PRIMARY | DSCAPS_FLIPPING;
    DFBCHECK(dfb->CreateSurface(dfb, &dsc, &primary ));    // プライマリサーフェイスを生成

    DFBCHECK (primary->GetSize (primary, &screen_width, &screen_height));  // 解像度を取得
```

```
                                                        // バックカラーを設定
    DFBCHECK (primary->SetColor (primary, 0xff, 0xff, 0xff, 0xff));
    DFBCHECK (primary->FillRectangle (primary, 0, 0, screen_width, screen_height));

    return OK;
}

static int  init_network_lcd(int port_number)
{
    struct  sockaddr_in  servaddr;

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);              // ソケットを生成
    if (sockfd == -1) {
        fprintf(stderr, "can't socket open ¥n");
        exit(EXIT_FAILURE);
    }
    memset(&servaddr, 0x00, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(port_number);

    bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));  // ポート番号の結び付け

    receive_buf = (char *)malloc(SR_BUF_SIZE );                    // 受信バッファの領域確保
    if (receive_buf == NULL) {
        fprintf(stderr, "memory allocation error¥n");
        exit(EXIT_FAILURE);
    }
    return OK;
}

static int mainloop_lcd(unsigned int loop_count)
{
    char *filename = "temp.jpeg";

//  while (loop_ount-- > 0) {
    for (;;) {
        receive_image(filename);              // 画像を受信
        display_image(filename);              // 画像を表示
    }
    return OK;
}

static int  receive_image(char *filename)
{
    FILE*fp;
    int     n;
    struct  sockaddr_in  cliaddr;
    socklen_t len;

    len = sizeof(cliaddr);
                                              // JPEG ファイルを受信
    n = recvfrom(sockfd, receive_buf, SR_BUF_SIZE , 0, (struct sockaddr *)&cliaddr, &len);
    printf("receive image = %d¥n", n);
    fp = (FILE *)fopen(filename, "wb");       // ファイルをオープン
    if (fp == NULL) {
        fprintf(stderr, "can't open ¥n");
        exit(EXIT_FAILURE);
    }
```

```
    fwrite(receive_buf, 1, n, fp);            // JPEG 画像をファイルに書く
    fclose(fp);                               // ファイルをクローズ
    return OK;
}

static int   display_image(char *filename)
{
    DFBRectangle r;

  //画像の表示位置指定
/*
  r.x = 100;
  r.y = 50;
  r.w = screen_width/2;
  r.h = screen_height/2;
*/
    r.x = 0;
    r.y = 0;
    r.w = screen_width;
    r.h = screen_height;

    DFBCHECK (dfb->CreateImageProvider(dfb, filename, &provider));  // 画像プロバイダを生成

//画像プロバイダの設定値取得（画像ファイル情報、解像度、ピクセル当たりのビット数等）
//  DFBCHECK (provider->GetSurfaceDescription (provider, &dsc));
//  printf("dsc.width = %d¥n", dsc.width);
//  printf("dsc.height = %d¥n", dsc.height);

    DFBCHECK (provider->RenderTo(provider, primary, &r));          // 画像ファイルを変換
    DFBCHECK (primary->Flip(primary, NULL, DSFLIP_WAITFORSYNC));   // ちらつき防止
    return OK;
}

static void  post_process_lcd(void)
{
    uninit_network_lcd();             // ネットワークを初期化前に戻す
    uninit_lcd();                     // LCD を初期化前に戻す
}

static void  uninit_network_lcd(void)
{
    free(receive_buf);                // 受信バッファを解放
    close(sockfd);                    // ソケットをクローズ
}

static void  uninit_lcd(void)
{
  provider->Release (provider);       // プロバイダを解放
  primary->Release (primary);         // プライマリサーフェイスを解放
  dfb->Release (dfb);                 // DirectFB を解放
}
```

参考文献

1 SESSAME WG 2、組込みソフトウェア開発のための構造化モデリング、翔泳社、2006
2 Michael H Schimek, et.al、Video for Linux API Specification Revision 0.24
3 Thomas G. Lane, IJG JPEG Library
4 W. Richard Stevens, UNIX Network Programming 2ed, Prentice Hall, 1998

資料 4

生産電子システム技術科

技能照査試験

平成２４年度

応用課程　生産電子情報システム技術科

技能照査　学科試験問題

職業能力開発総合大学校

下記の問題１から問題５０について解答しなさい。解答はア、イ、ウの中から１つ選択し、解答用紙に記入しなさい。

問題１　労働安全衛生法の目的について誤っている文章を選びなさい。

ア　この法律は、職場および家庭における労働者の安全を補償することが目的である。
イ　この法律は、労働災害の防止のための危害防止基準の確立を目指したものである。
ウ　この法律は、労働災害の防止のための責任体制の明確化及び自主的活動の促進を目的としている。

問題２　ＶＤＴ作業が健康に及ぼす影響について誤っている文章を選びなさい。

ア　疲労とは、身体的または精神的作業の結果、作業能力が低下する状態を示す。
イ　ＶＤＴ作業における疲労の現れ方には、眼の疲労による場合、外的環境に起因する場合、内的環境に起因する場合、心的規制に起因する場合がある。
ウ　疲労の原因には、騒音と、不安の２種類だけである。

問題３　図1はある仕事の日程計画を表している。丸印は仕事の着手または完了ポイント、矢印は作業である。開始日①からA～Gまでのすべての作業を行い、完了日⑥に至るまでに必要な標準日数を求めなさい。

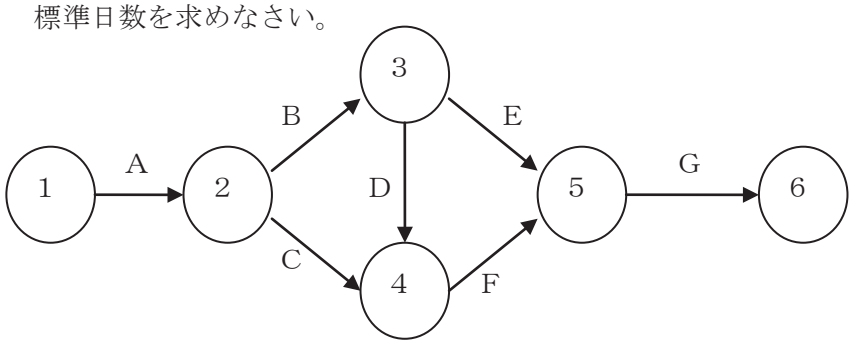


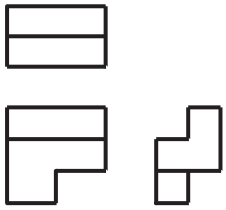
図１　日程計画

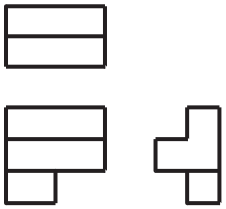
作業	標準日数（日）
A	3
B	6
C	5
D	3
E	4
F	5
G	3

ア　16
イ　20
ウ　29

問題４　図２の立体図に示す品物Ａを表わす投影図を選択しなさい。

品物Ａ 

ア 

イ 

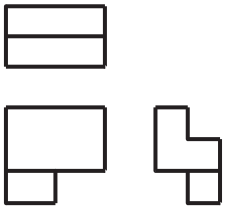
ウ 

図２　品物Ａの投影図

問題５　次の英文和訳の概略訳で、正しいものはどれか。

Unlike a general diode, if the constant current diode is installed with the different polarity, it will conduct heavily and damage both itself and the meter

ア　定電流ダイオードの極性を間違えて取り付けるとダイオードとメータを損傷する。
イ　整流ダイオードの極性を間違えて取り付けると定電流が流れ、ダイオードとメータが損傷する。
ウ　定電圧ダイオードの極性を間違えて取り付けると定電圧により、ダイオードとメータが損傷する。

問題６　次の英文による説明として誤っているものを選びなさい。

The decibel is a logarithmic function that allows large variations to be shown in very small increments. Increases or decreases of 3dB will result in a doubling or halving of the power. Increase or decreases of 6 dB will cause a doubling or halving of the voltage.

ア　3dB の増加によりパワーは２倍となる。
イ　6dB の減少により電圧は半分になる。
ウ　3dB の減少により電圧は半分になる。

問題７　次の英文は、何について説明したものか。

This is a discontinuous signal that changes from one state to another in a number of distinct steps within fixed bit framing time slots. In its basic form, there are two states or signal levels – on and off where the “on “ translates to　a one level corresponding to the digit one, and “off “ translates to a zero level corresponding to the digit zero.

ア　Frequency
イ　Digital　Signal
ウ　Digital　Transmission

問題８　優先順位の高いタスク A が実行可能状態になったとき、実行中のタスク B を中断してタスク A を実行することを何というか。最も適切なものを選びなさい。

ア　リアルタイム
イ　フェッチサイクル
ウ　プリエンプション

問題９　以下の記述のうち、最も適切なものはどれか。

ア　セマフォはシングルタスク環境では無意味である。
イ　セマフォを使用すればデッドロックを防ぐことができる。
ウ　セマフォにはプロセス間で同期をとる機能はない。

問題１０　以下の記述のうち、最も適切なものはどれか。

ア　メッセージキューに書き込まれたデータは、読み出されると消滅する。
イ　メッセージキューは特権ユーザでないと使用できない。
ウ　メッセージキューには通常、同期機能はない。

問題１１　多重割り込みをサポートするシステムで、割り込みが図3のタイミングで発生した場合に、割り込みCによって起動されるタスクCが終了するのは、割り込みCが発生してから何ミリ秒後か。最も適切なものを選びなさい。

ここで、割り込みA、B、Cによって起動されるタスクA、B、Cの処理時間はそれぞれ30ミリ秒、10ミリ秒、40ミリ秒とし、割り込み優先順位はA>B>Cとする。

ア　40ミリ秒後
イ　50ミリ秒後
ウ　60ミリ秒後

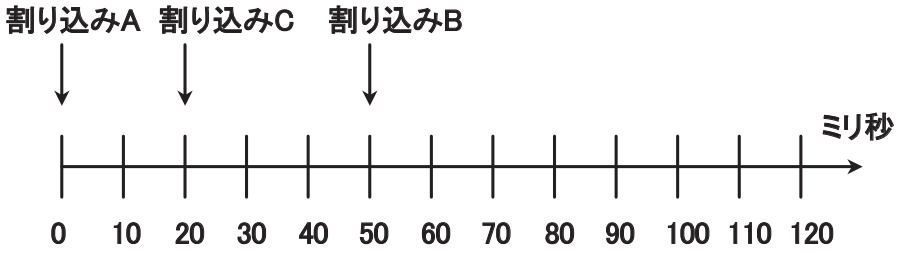


図3　多重割り込み

問題１２　以下のリアルタイムOSのイベントドリブンプリエンプション方式に関する記述について、空欄に入る語句の最も適切な組み合わせはどれか。

(a)　他に実行されているタスクがないときにタスクが起動されると（　(1)　）状態になる。
(b)　優先度の高いタスクが実行されているために実行できないタスクの状態を（　(2)　）状態と呼ぶ。
(c)　割り込みハンドラなどで、より優先度の高いタスクが起動されると、それまで実行状態だったタスクは（　(3)　）状態になる。

	(1)	(2)	(3)
ア	実行	待ち	実行可能
イ	実行	実行可能	実行可能
ウ	実行可能	実行可能	待ち

問題１３　以下の記述のうち、最も適切なものはどれか。

ア　組込み機器は、組込み機器に使用する CPU と同じ CPU を持ったパソコンを使用して開発されるのが一般的である。
イ　複数のオブジェクトファイルはライブラリを結合してひとつの実行プログラムを生成するプログラムをリンカ（リンケージエディタ）という。
ウ　異なったアーキテクチャ用の実行プログラムを生成するコンパイラを逆コンパイラと呼ぶ。

問題１４　以下の記述のうち、最も適切なものはどれか。

ア　共有メモリには同期機能がないので、共有メモリの機能だけではメモリに書いたということをほかのプロセスに伝えることができない。
イ　共有メモリは、別のマシンで実行しているプロセスからでもネットワーク経由で使用できる。
ウ　共有メモリは、複数のプロセスからアクセスされるが、排他制御機能があるので競合状態を回避することができる。

問題１５　H8 マイコンに関する次の記述に関して、間違っているものはどれか。

ア　H8 マイコンは、ROM、RAM や I/O ポートなど、システムを構成するうえで必要となる基本的な機能が１つのチップにまとめられている。
イ　ワンチップマイコンの一種ではない。
ウ　開発環境としては純正の C/C++/アセンブラパッケージや統合開発環境 HEW などがある。

問題１６　図４はH8マイコンのプログラム開発手順である。(A)～(D)に当てはまるファイル名を答えよ。

	(A)	(B)	(C)	(D)
ア	3	2	4	1
イ	4	1	3	2
ウ	2	4	1	3

１　バッチファイル
２　ロードモジュール
３　オブジェクトファイル
４　実行ファイル

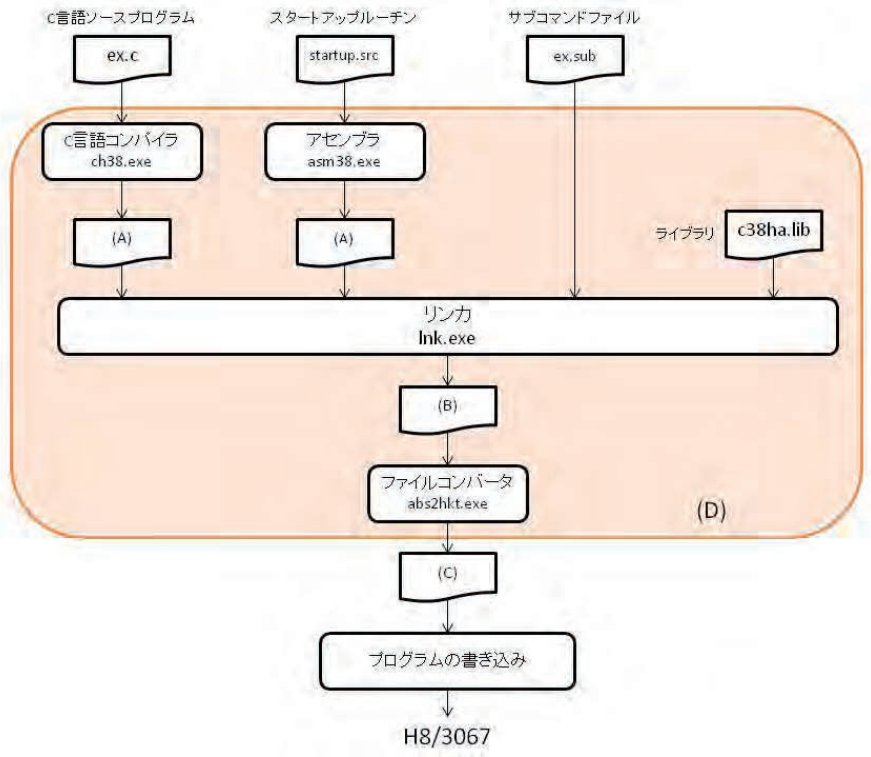


図４　H8/3067プログラム開発手順

問題１７　図５は、H8/3067 の I/O ポートのレジスタである。(A) ～ (C) にあてはまるレジスタ名を答えよ。

	(A)	(B)	(C)
ア	1	3	2
イ	3	2	1
ウ	2	1	3

１　PCR

２　DDR

３　DR

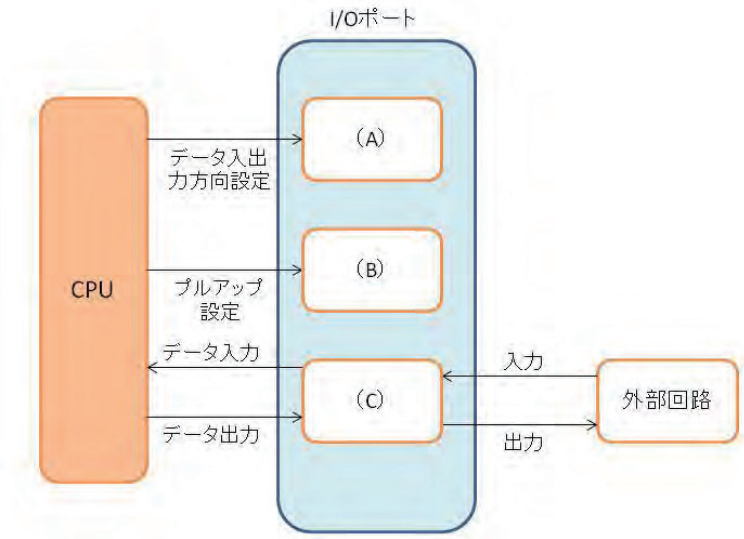


図５　I/O ポートのレジスタ

問題１８　PIC マイコンに関する次の記述に関して、間違っているものはどれか。

ア　PIC とは、Peripheral Interface Controller のことである。

イ　使用目的が比較的明確な範囲に限られているマイクロコンピュータである。

ウ　PIC のハードウェアアーキテクチャでは、データ用のメモリとプログラム用のメモリが同じものである。

問題１９　次のC言語によるプログラムは、2個のIPアドレスを比較して両者が同じである場合に0、異なる場合に1を返す関数である。(A)に入るコードで正しいものはどれか。

ア　IP_a[i] != IP_b[i]

イ　IP_a[i] == IP_b[i]

ウ　IP_a[i] &= IP_b[i]

```
unsigned char IP_compare(unsigned char *IP_a, unsigned char *IP_b)
{
   unsigned short i;

   for (i = 0; i < 4; i++)
   {
      if (   (A)   )
      {
         return 1;
      }
   }
   return 0;
}
```

問題２０　ルータに関する記述として、適切なものはどれか。

ア　データリンク層での接続を行い、トラフィック分離機能をもつ。

イ　トランスポート層以上のプロトコルを含めてプロトコル変換を行い、異なるネットワークアーキテクチャをもつネットワークを相互に接続する。

ウ　ネットワーク層での接続を行い、広域網を介してLAN間を接続する場合などに使われる。

問題２１　DMZ型のファイウォールについて、適切でないものはどれか。

ア　DMZ型のファイアウォールは、サーバを外部ネットワークに、クライアントをDMZに配置する。

イ　プライベートアドレスとグローバルアドレスの変換は、NAPTにより実現できる。

ウ　アクセス制限にはパケットフィルタリングを用いる。

問題２２　インターネットサーバについて、適切でないものはどれか。

ア　DNS サーバはホスト名を IP アドレスに変換する。
イ　メールの配送のプロトコルは SMTP である。
ウ　HTTP のポート番号は 8080 である。

問題２３　TCP/IP について、適切でないものはどれか。

ア　アプリケーションはポート番号で識別される。
イ　TTL の値はパケットの生存時間を秒単位であらわしたものである。
ウ　MTU より大きいパケットは、ルータなどの中継器で分割される。

問題２４　公開鍵暗号の記述について、適切でないものはどれか。

ア　公開鍵暗号はメッセージを公開鍵で暗号化し、秘密鍵で復号する。
イ　RSA 暗号は素因数分解問題が基礎となっている。
ウ　公開鍵暗号を用いて電子署名を実現する場合、公開鍵で署名し秘密鍵で検証する。

問題２５　他の機器に悪影響を与える電磁波ノイズを出さず、また電磁波ノイズを受けても誤動作しないことを両立させることを表す用語はどれか。

ア　ＥＭＣ
イ　ＥＭＩ
ウ　ＥＭＳ

問題２６　アナログ、ディジタル混在回路のグランドの取り方で正しい記述はどれか。

ア　アナログ回路とディジタル回路のグランドは常に共通にしておく。
イ　アナログ回路とディジタル回路のグランドは別々にとり、それぞれのグランドからできるだけ離れた所で１点で共通にする。
ウ　アナログ回路とディジタル回路のグランドは別々にとり、それぞれのグランドからできるだけ近い所で１点で共通にする。

問題２７　図６の回路は何と呼ばれているフィルター回路か。

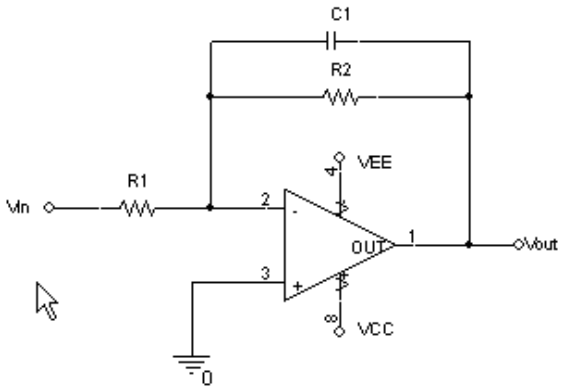


図６　フィルター回路

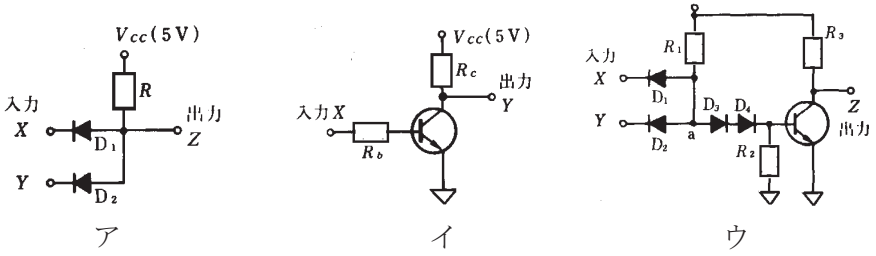
ア　ローパスフィルター

イ　ハイパスフィルター

ウ　バンドパスフィルター

問題２８　図７に示すＮＡＮＤ回路に相当する等価回路で、最も適切なものを選びなさい。

ＮＡＮＤ回路



ア　　イ　　ウ

図７　ＮＡＮＤの等価回路

問題２９　図８に示す各シンボルに対応した負論理シンボルで適切なものを選びなさい。

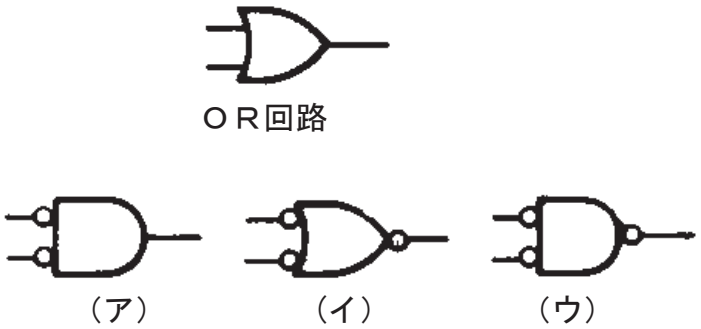


図８　負論理シンボル

問題３０　図９に示す論理回路の出力Ｗを入力Ｘ、Ｙ、Ｚを用いて論理式で表したものを選びなさい。

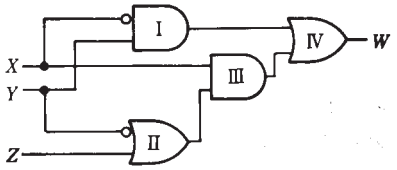


図９　論理回路

ア　$\bar{Y}+Z$

イ　$X\cdot(\bar{Y}+Z)+\bar{X}\cdot Y$

ウ　$X\cdot(\bar{Y}+Z)$

問題３１　図10に示す回路において入力Ｓ ＝“H”，入力Ｒ ＝“Ｌ”のとき、 出力Ｑ の動作状態に最も適切なものを選びなさい。

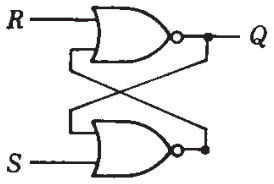


図 10　論理回路

ア　リセット状態
イ　変わらない
ウ　セット状態

問題３２　次の A/D・D/A 変換に関する記述で、正しいものはどれか。

ア　逐次比較型 A/D 変換は、２重積分型に比べ変換速度が遅い。
イ　A/D 変換の精度は、サンプリング周期が短いほど精度が高い。
ウ　１ビットの A/D 変換器、D/A 変換器は音楽用には使用できない。

問題３３　下記に示すシャノンの定理で、（　　）内に入る適切な式を選びなさい。

T（標本化間隔時間）＜＝（　　）伝送する情報の最高周波数

ア　1/2f
イ　1/3f
ウ　1/4f

問題３４　１ＫＨｚのパルス波をスペクトラムアナライザで観測したら図 11 となり、シミュレーションのＦＦＴ解析結果と同じである。レベルが最も高い周波数は１ＫＨｚである。それでは、レベルが 2 番目に高い周波数はどれくらいか。

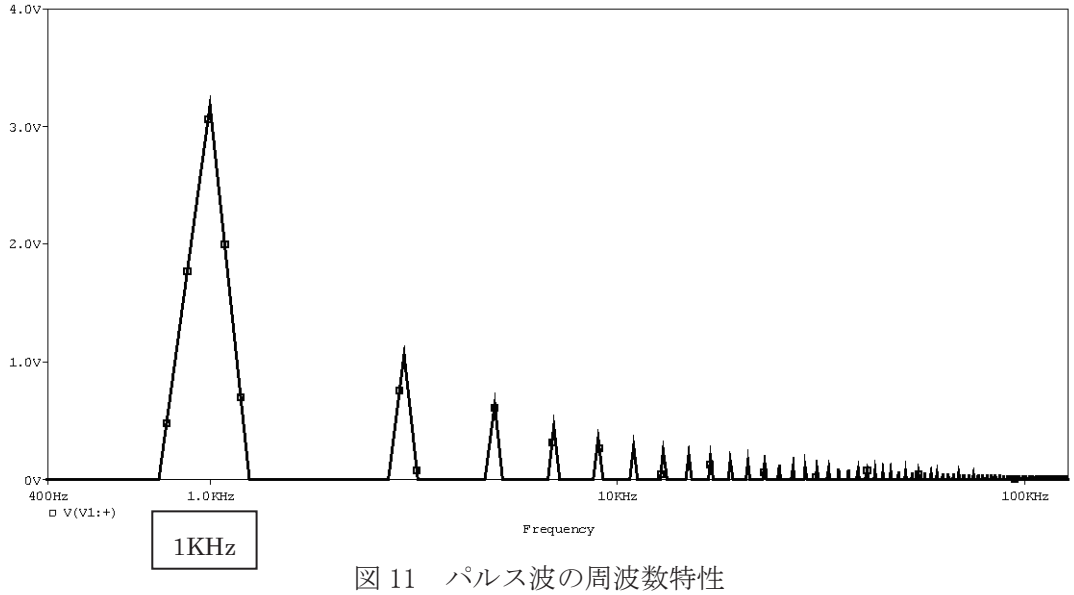


図 11　パルス波の周波数特性

ア　２ＫＨｚ

イ　３ＫＨｚ

ウ　５ＫＨｚ

問題３５　図 12 のＡＭ変調回路の変調度を波形から求めると、凡そどれくらいか。

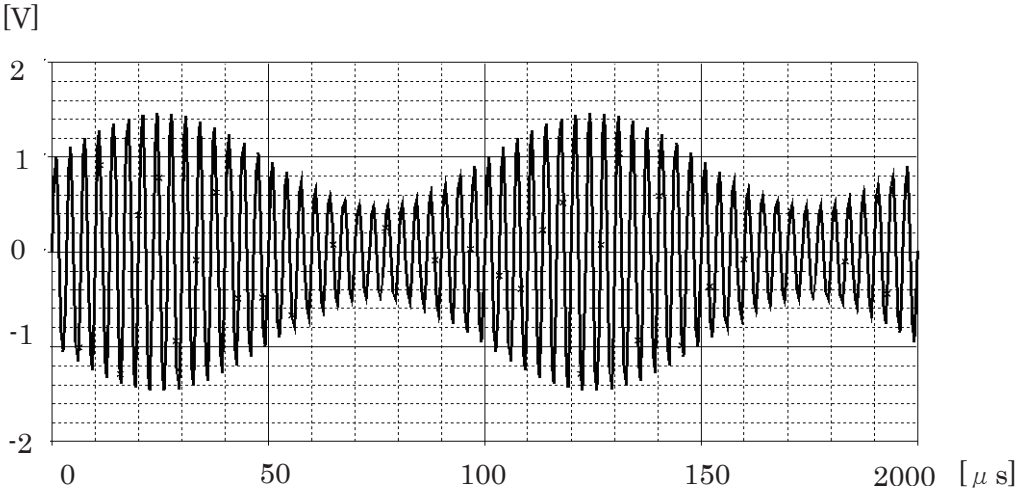


図 12　AM 変調回路の波形

ア　０．５

イ　１．０

ウ　１．５

問題３６　アナログテスタによるトランジスタの良否検査測定で不良個所と考えられる箇所はどれか。

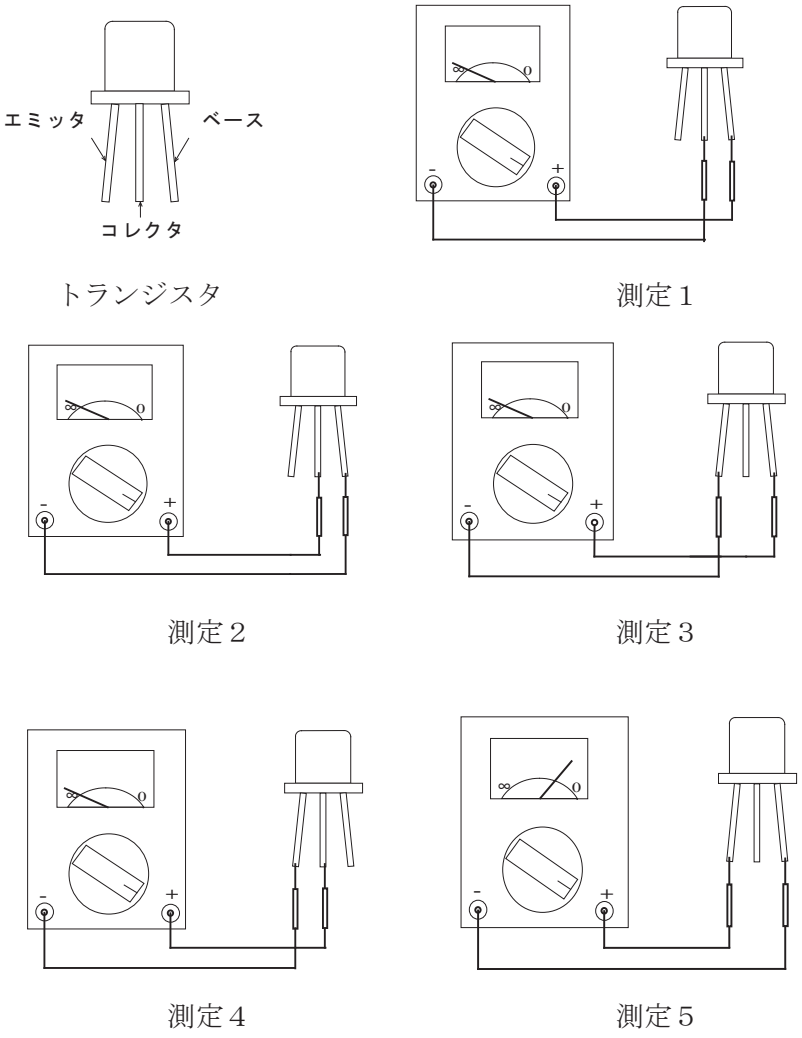


図 13　トランジスタの良否判定

ア　ベース・エミッタ間短絡
イ　ベース・エミッタ間開放
ウ　ベース・コレクタ間開放

問題３７　次の記述で、（　）内にあてはまる適切なものはどれか。

半導体において、自由電子が電気伝導に寄与しているものをN形半導体といい、ヒ素（Ａｓ）のように電子を供給する不純物のことを（　　）という。

ア　ソース
イ　アクセプター
ウ　ドナー

問題３８　次の記述で、（　）内にあてはまる適切なものはどれか。

サーミスタの用途は、電子回路の温度補償、センサなどの温度検出に使用されており、温度変化に対してその抵抗値が大きく変化する半導体素子である。温度―抵抗値特性としては、一般に（　　）のものがよく使用される。

ア　負の温度係数
イ　正の温度係数
ウ　直熱形

問題３９　VerilogＨＤＬ記述で、always 文中で代入される信号のデータの型は何か。

ア　レジスタ型
イ　ネット型
ウ　入出力型

問題４０　組み合わせ回路で代入するときに使用する [　　] 内の文は何か。

[　　] OUT = IN1 & IN2;

ア　ａｓｓｉｇｎ
イ　ｉｎｐｕｔ
ウ　ｉｎｏｕｔ

問題４１　次の回路は何を表しているか。

```
function   [1:0]   adder ;
  input    [1:0] a,  b ;
  input          sel ;
    case(sel)
      0:    adder = a ;
      1:    adder = b ;
    endcase
endfunction

assign   out = adder ( aa, bb, sel ) ;
```

ア　加算器
イ　カウンタ
ウ　セレクター

問題４２　図 14 は自動化センサシステム構成を表したものである。センサシステム構成に関する記述の（　①　）～（　③　）内に当てはまるものの組合せで、最も適切なものはどれか。

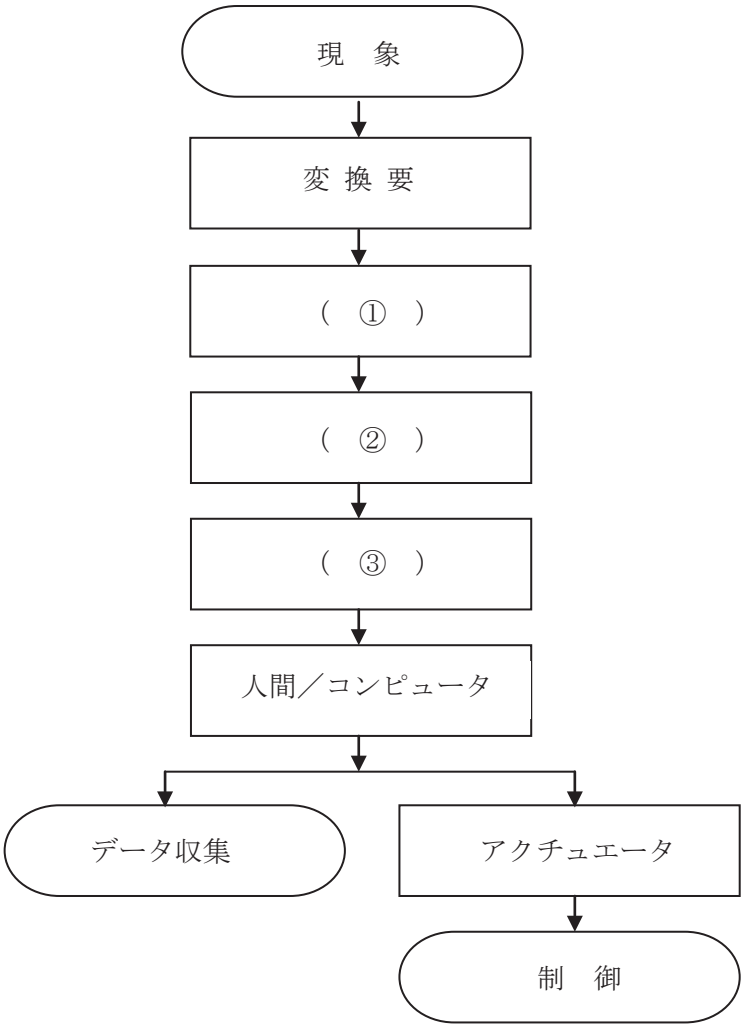


図 14　自動化センサシステム構成

	①	②	③
ア	信号受信・処理要素	情報出力要素	信号伝送要素
イ	信号伝送要素	信号受信・処理要素	情報出力要素
ウ	信号伝送要素	情報出力要素	信号受信・処理要素

問題４３　次の計測技術に関する記述で、誤っているものはどれか。

ア　計測技術における２値形センサには、タッチセンサ、シグナルセンサがある。
イ　計測技術におけるアナログ形センサには、エアマイクロメータ、ロードセルがある。
ウ　計測技術におけるデジタル形センサには、電気マイクロメータがある。

問題４４　次のセンサに関する記述で、誤っているものはどれか。

ア　誘導型近接センサは、導電性プラスチックを検出できる。
イ　導電性プラスチックは、圧力センサとして広く利用されている。
ウ　静電容量型近接センサは、導電性プラスチックを検出できる。

問題４５　図15は、コンピュータを使用した自動計測手法のGP-IB計測システムである。この計測システムでは各インタフェースに、コントローラ、トーカ、リスナの機能を持たせている。その組合せで、誤っているものはどれか。

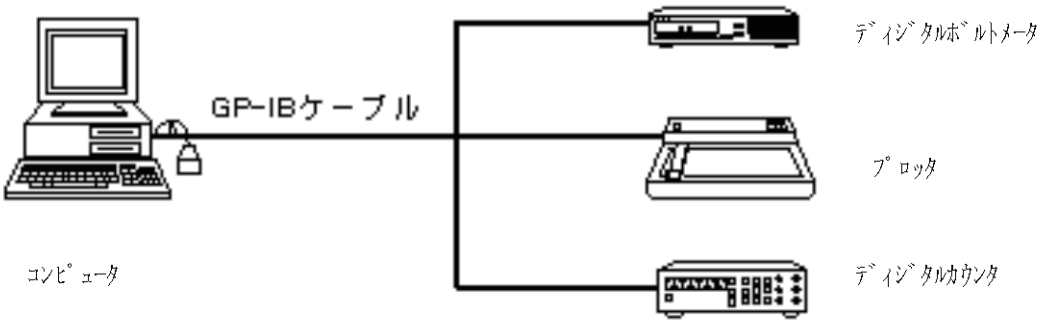


図15　自動計測（GP-IB計測）システム

ア　コンピューター→コントローラ
イ　デジタルボルトメーター→リスナ
ウ　プロッター→トーカ

問題４６　次の現象に関する効果の記述で（　①　）～（　③　）内に当てはまるものの組合せで、最も適切なものはどれか。

異種金属を結び、２つの接点間に温度差を与えると一定の方向に電流が流れる。この効果を（　①　）効果と呼ぶ。逆に電圧を加えると温度差ができる効果を（　②　）効果と呼ぶ。また、１つの金属上で温度差がある 2 点間に電流を流すと熱を吸収したり発生したりする効果を（　③　）効果と呼ぶ。

	①	②	③
ア	ペルチェ	トムソン	ゼーベック
イ	ゼーベック	ペルチェ	トムソン
ウ	ペルチェ	ゼーベック	トムソン

問題４７　次のプリント配線板設計及び実装に関する記述で、誤っているものはどれか。

ア　図 16 に示すプリント基板への部品取り付けは、プリント基板の基準方向（X―Y 方向）に従い、表示が上から下へ、左から右への方向に読み取れるように垂直、水平に整然と実装する。

イ　フラックスの作用には、洗浄化作用、表面張力を低下させる作用及び酸化防止作用がある。

ウ　めっきした金属にはんだ付けをするとき、そのめっきによるはんだ付けのしやすさは、はんだめっき、すずめっき、ニッケルめっきの順である。

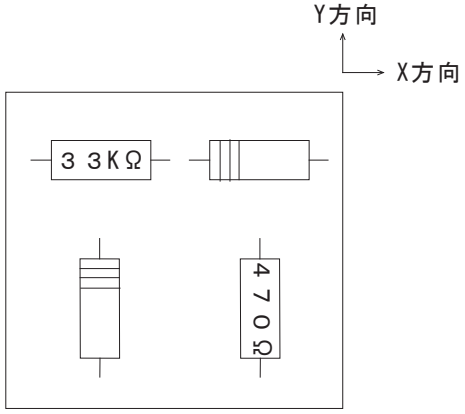


図 16　部品の取り付け方向と表示

問題４８　プリント基板製作作業に関する記述の（　①　）～（　④　）内に当てはまるものの組合せで、最も適切なものはどれか。

電子回路・作成
↓
（　①　）
↓
感光基板の準備
↓
（　②　）
↓
現像処理
↓
（　③　）
↓
穴あけ
↓
（　④　）
↓
プリント基板完成

ア	① エッチング作業 ② パターン原図作成作業 ③ パターン原図のネガフィルム露光作業 ④ はんだ付け作業
イ	① パターン原図作成作業 ② パターン原図のネガフィルム露光作業 ③ エッチング作業 ④ はんだ付け作業
ウ	① パターン原図作成作業 ② パターン原図のネガフィルム露光作業 ③ はんだ付け作業 ④ エッチング作業

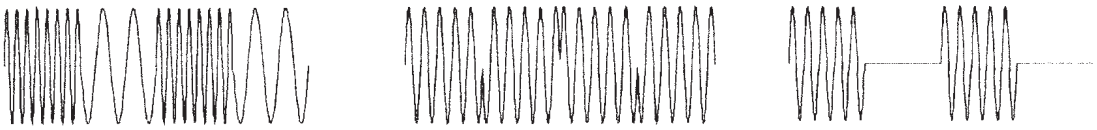
問題４９　マンチェスター伝送方式について、下記の伝送符号から正しいものを選びなさい。

ア　ビット０が電位０に、ビット１が正電位もしくは負電位に対応する。また、ビット間は電位０に戻る。

イ　ビット０は正電位から負電位に変換し、ビット１は負電位から正電位に変換する。

ウ　ビット０が負電位に、ビット１が正電位に対応する。ビット間は電位０に戻らない。

問題５０　デジタル信号送受信の際に使用する変調方式の１つに、振幅シフトキーイング（ASK：Amplitude-Shift Keying）がある。この方式の変調波形を図 17 から選びなさい。



ア　　イ　　ウ

図 17　ASK の変調波形

正解

応用課程　技能照査学科試験　解答用紙

生産電子情報システム技術科

学籍番号＿＿＿＿＿＿　　氏名＿＿＿＿＿＿＿＿

番号	解答欄	番号	解答欄
1	ア	26	ウ
2	ウ	27	ア
3	イ	28	ウ
4	イ	29	ウ
5	ア	30	イ
6	ウ	31	ウ
7	イ	32	イ
8	ウ	33	ア
9	ア	34	イ
10	ア	35	ア
11	ウ	36	ウ
12	イ	37	ウ
13	イ	38	ア
14	ア	39	ア
15	イ	40	ア
16	ア	41	ウ
17	ウ	42	イ
18	ウ	43	ウ
19	ア	44	イ
20	ウ	45	ウ
21	ア	46	イ
22	ウ	47	ア
23	イ	48	イ
24	ウ	49	イ
25	ア	50	ウ

アンケート調査へのご協力のお願い　　調査研究資料No.135

今後、基盤整備センターがより良い調査・研究を行うために、本書のご活用事例のアンケート調査へのご協力をお願い申し上げます。

以下のフォームに直接ご記入いただくか、ホームページ（http://www.tetras.uitec.jeed.or.jp/）からダウンロードしていただき、FAXまたはメールで下記までお送りください。

1）活用した内容 （いつ、何のために、活用したページ、どのように、複製の有無）	
2）本書に対するご意見、ご要望、今後期待するテーマ	
3）連絡先 （施設名、役職、電話番号）	

宛先　　基盤整備センター普及促進室

FAX	0422-38-5228
メール	fukyu@uitec.ac.jp

その他、お問い合わせは基盤整備センター普及促進室（TEL 0422-38-5225）にご連絡下さい。

本報告書等は、基盤整備センターホームページ「職業能力開発ステーションサポートシステム（TETRAS）」の「基盤整備センター刊行物検索」から閲覧、ダウンロードができます。

URL：　http://www.tetras.uitec.jeed.or.jp/

調査研究資料　No.135
「新訓練科（高度職業訓練専門課程及び応用課程）の試行検証に関する調査研究」

発行　2013年3月
発行者　独立行政法人高齢・障害・求職者雇用支援機構
職業能力開発総合大学校　基盤整備センター
所長　長谷川　健治
〒180-0006　東京都武蔵野市中町１－１９－１８　武蔵野センタービル４F
電話　0422-38-5225（普及促進室）
印刷　株式会社旭クリエイト
〒220-0023　神奈川県横浜市西区平沼１－３－１７　宮方ビル４F
電話　045-620-8890

ISSN 1340-2404

調査研究資料 No.135
2013

THE INSTITUTE OF RESEARCH AND DEVELOPMENT
POLYTECHNIC UNIVERSITY