平成 14 年度

学士学位論文

高知工科大学情報システム工学科

DSP ボードを用いたオンライン学習・評価システムの研究

Research on Online-learning and Evaluation System with DSP Board

1030115　佐藤　公信

指導教員　情報システム工学科　竹田史章

2003 年 2 月 12 日

高知工科大学　情報システム工学科

要旨

高知工科大学情報システム工学科

DSP ボードを用いたオンライン学習・評価システムの研究

佐藤公信

近年，携帯電話などの情報端末が普及しており，このため情報機器の小型化，省電力化が求められている．それらに対応するため小型で，低消費電力で駆動，さらに高速な乗算器を持つ DSP を情報端末の演算部として用いることを前提とし，ニューラルネットワーク（NN）システムを搭載した DSP ボードを用いたオンラインチューニングを実現するための基礎実験を行う．オンラインチューニングとはある程度の認識率を有する NN ウエイトに対して更に使用システムに適した NN ウエイトへと最適化を行うことを意味する．これにより使用対象システムに特化した認識能力を有する NN システムの実現を可能とする．

本論文では，オンラインチューニングの基幹要素である継続学習の基本性能確認実験を実施し，さらに筋活動電位（EMG）パターン識別において PC 上で動作する NN システムと DSP ボード上で動作する NN システムの互換性を示すための性能比較実験を併せて実施し，データの互換性を検証する．

キーワード：DSP，紙幣識別，筋活動電位，オンラインチューニング

Abstract

Research on Online-learning and Evaluation System with DSP Board

Hironobu SATOH

Recently, information terminals such as cellular phone are widely used. As a result, this kind of information tool is required to be minimized for electric power conservation. Therefore, the DSP board, which for a high-speed multiplier and consumes little electric power, has been applied as the operation part of the information terminal. Furthermore, the neural network has been implemented to this DSP board for online tuning. Using the online tuning, the suitable NN weights are yielded to optimize the recognition ability of the system.

In this research, the online tuning is relised by the continuous learning and the recognition system is performed to improve its ability. Furthermore, this system is tested with the electromyogram and Thai banknote recognition for the tuning ability. Still more, we show the compatibility between the system ability of the PC and the one of the DSP board.

Keyword: DSP, Banknote recognition, Electromyogram, Online-tuning

目次

図目次

表目次

第1章　はじめに

近年，携帯電話をはじめとする私たちの生活を支えるために必需品である情報処理電化製品は，簡要に持ち運べ，外出先でも長時間使用する必要がある．そのために省電力化に向けて小型，軽量，省電力化の一途をたどっている．音声処理や識別の分野などで必要とされる能力はリアルタイムでの処理である．しかしながら，情報を処理するため不可欠な PC（パーソナルコンピュータ）の重要な構成要素である CPU は現在の技術を使用する場合，情報を高速に処理するため，マシンクロックの高速化の一途を辿っている．マシンクロックの高速化は CPU に供給する電圧の上昇という問題を発生させている．したがって，現在では CPU の小型化，省電力，低発熱量という言葉は情報端末の進化において不可欠なキーワードとなっている．そのような問題を解決する方法として，DSP を情報機器の演算装置として用いることを提案する．DSP を用いた研究は数多く報告されているが，NN（ニューラルネットワーク）を用いた研究は少数であり数多く報告されていない[(1), (2)]．

本研究では特に，NN システムを搭載した DSP ボードを用いてオンラインチューニングを実現するための基礎実験を行う．オンラインチューニングでは，まず PC を用いあらかじめ種々のばらつきを有する学習データから様々なデータに対し一定の汎化能力を有する NN ウエイトを作成する．次に，作成した NN ウエイトを DSP ボードにダウンロードし，データの追加と継続学習を行うことにより NN ウエイトをチューニングする．これにより使用対象システムに特化した認識能力を有する NN システムの構築を可能であることが考えられる．また PC 上で動作する NN システムと DSP 上で動作する NN システムとの間に互換性を持たせることによりシステムの汎用化がはかれると推測される．そのため，PC の NN システムと DSP の NN システムが同等の性能を有するならば NN ウエイトの共有を行うことが可能であると考えられる．

本研究では，オンラインチューニングの基幹要素である継続学習の基本性能確認を実施

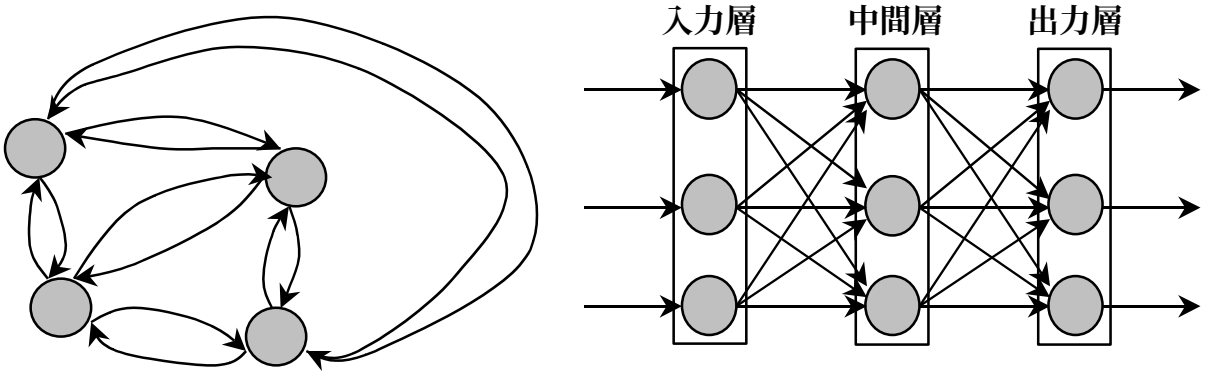
し，さらにEMGパターン識別においてPCのNNシステムとDSPボードのNNシステムの互換性を検証するための性能比較実験を実施する．継続学習の評価対象としては，筋活動電位（EMG：Electoromyogram）とタイ紙幣を用いる．また，性能評価実験の評価システムとしてEMGパターン識別を使用する．

第2章　ニューラルネットワークシステム

NN 情報処理の特徴は，高度並列分散処理型の情報処理と表現できる．NN は脳神経細胞の繋がりを模倣したもので，ある程度の柔軟性を有した識別や認識を行うことができる．したがって，NN は柔軟な識別を要求される紙幣識別分野などで用いられている[3]．

NN は相互結合ネットワークと階層型ネットワークなどに大別され，階層型 NN はパターンの識別に特化しており，ユニットが複数の階層をなすようにならび，下の層から上の層へ向かう結合のみがあるネットワークである．また，相互結合ネットワークは自己想起的な連想記憶に用いることが多く，記憶させたいネットワークの状態が安定的な平衡状態になるように，ネットワークの結合を設定する必要がある．このような特徴を考慮し，本論文で用いる NN は階層型 NN である[3]．

図 2.1 は相互結合ネットワークと階層型ネットワークについて示し，表 2.1 に階層型ネットワークと相互結合型ネットワークを比較した表を示す．



（a）相互結合型 NN　　（b）階層型 NN

図　2.1　NN の分類

表 2.1　分類別における NN の特徴[4]

ネットワーク構造	階層型 NN	相互結合型 NN
学習アルゴリズム	教師あり学習	教師なし学習
入力信号の流れ	フィードフォワード	フィードバック
時間の取り扱い	連続（アナログ的）	不連続（ディジタル的）
応用分野	パターン認識	自己想起

階層型 NN は多様なニューロンの組み合わせにより形成され，複雑な処理を行うことが可能である．しかしながら，階層型 NN を構成するニューロンは単体では図 2.2 に示される単純な処理しか行うことができない多入力一出力の素子である．つまり，ニューロンへの入力 x は式 2.1 に示される前層ニューロンの出力値の総和である．ここで，W はネットワークの結合荷重，S は前層のニューロンの出力を示す．

$$x = \sum_{k=1}^{K} w_k s_k \tag{2.1}$$

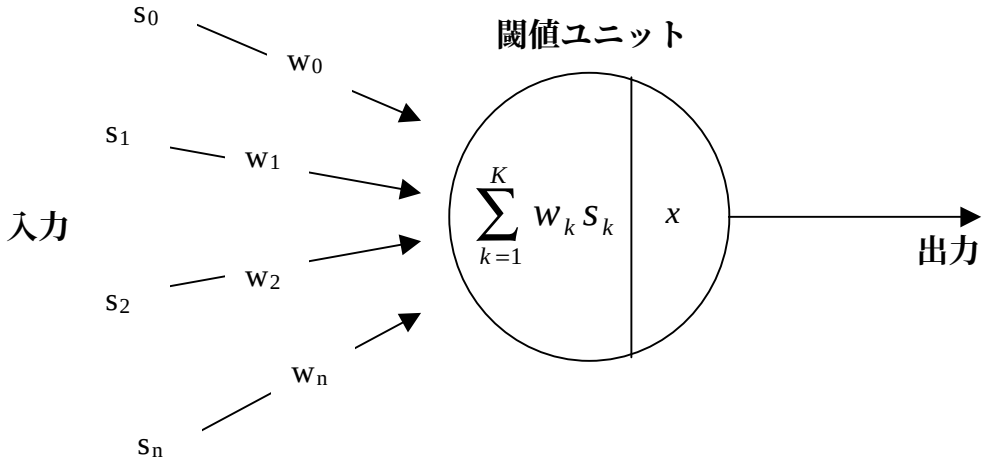
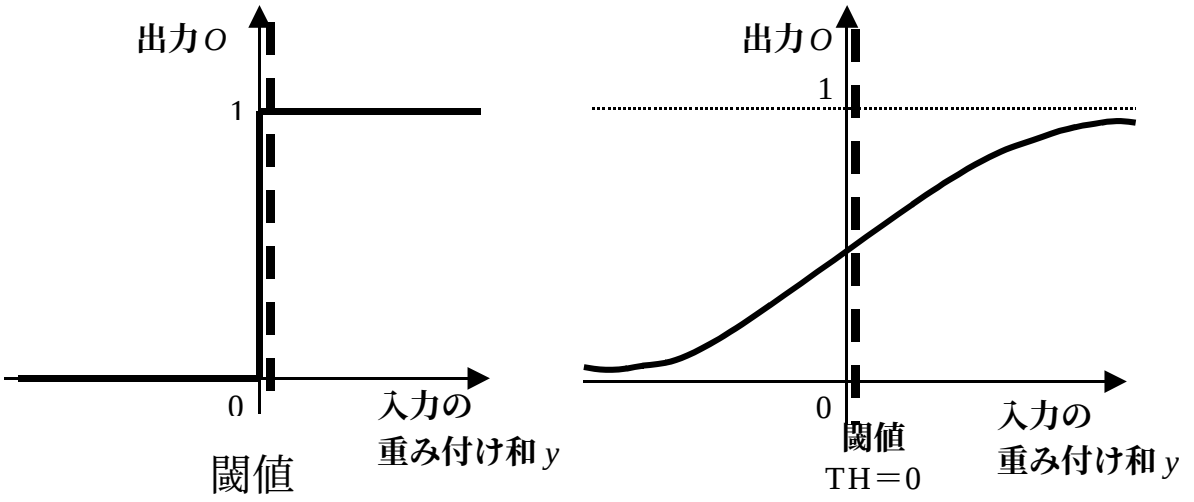


図 2.2 基本ニューロモデル

ニューロンの出力は ON または OFF の 2 出力である．つまり，閾値（TH：Threshold）を 0 とするとニューロンの出力関数は図 2.3 (a) に示されるステップ関数であると考えられる．しかしながら，ステップ関数は重みの適切な修正量が出力されないという問題が考えられるため，ステップ関数の ON-OFF 特性を保存しながら，適切な重みの変化量を出力できるシグモイド関数を採用する．図 2.3 (b) にシグモイド関数を示す．閾値と入力の重み付き総和が近似した値の場合，ニューロンの出力は中間値付近の出力値となる．これはステップ関数の特徴を保持しながら，ニューロンの出力が学習を行う際に有益な形で入力と関連付けられることを示す．また，このシグモイド関数は単調増加で微分可能であるため，後述する逆誤差伝播法（Error Back Propagation Method）に適用できる．



（a）ステップ関数　　（b）シグモイド関数

図 2.3 出力関数

2.1 学習

学習とは，ネットワークのあるパターンを分類させてみて間違った場合には重みを修正する，ということを繰り返すことによって最終的に全てのパターンを正しく識別することである．

更新された新しい重み w_{new} は式 (2.2) に示すように現在の重み w_{old} と重みの修正量 Δw の和で表される. まず, 出力層から入力層に向けての層ごとの重み更新法を示し, Δw の求め方について述べる.

$$w_{new} = w_{old} + \Delta w \tag{2.2}$$

まず, 出力層－中間層間の重み修正方法について述べる. ある入力パターン与えた場合, 出力層の k 番目のニューロンの期待出力 d_k と k 番目のニューロンの実際の出力 o_k の誤差 E_p は

$$E_p = \frac{1}{2}(d_k - o_k)^2 \tag{2.3}$$

と表される. したがって, 1 つの学習パターン q における出力層の誤差 E_q は

$$E_q = \frac{1}{2}\sum_{k=1}^{K}(d_k - o_k)^2 \tag{2.4}$$

と示される. すべての学習パターンの誤差和を E とすると, 誤差関数は

$$E = \frac{1}{2}\sum E_q \tag{2.5}$$

と表される. また, ニューロンへの入力を式 (2.6) に出力を式 (2.7) に示す. ここで, w_{kj} はニューロン k からニューロン j への重みである.

$$x_k = \sum_{j=1}^{J} w_{kj} y_j \tag{2.6}$$

$$O_k = f(x_k) \tag{2.7}$$

本論文では出力関数として，式 (2.11) に示すシグモイド関数を示す．ここで *TH* はニューロンの閾値を示す（TH=0）．

$$f(x) = \frac{1}{1+\exp\left(\frac{-x+TH}{T}\right)} \tag{2.8}$$

出力ニューロンの最大発火値を利用することから *TH*＝0 と設定する．図 2.4 に示すようにネットワークの温度 *T* は誤差に比例して，1.3 から 0.7 まで変化させ，アニーリングを行なう．この操作は学習プログラムが自動的に行なう．

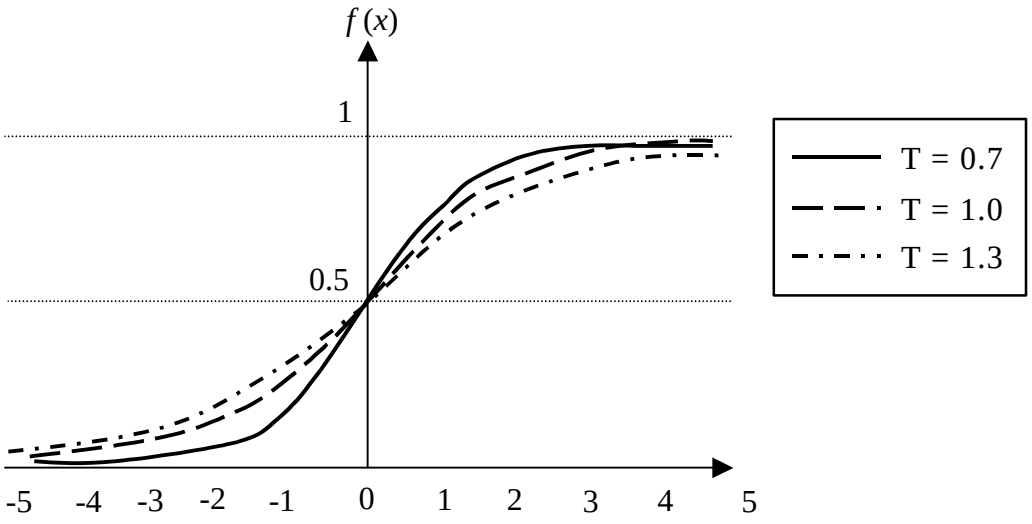


図 2.4 シグモイド関数

式（2.5）と式（2.6）より最急降下法と連鎖則を用い

$$\Delta w_{kj} = -h \frac{\partial E}{\partial w_{kj}}$$

$$= -h \frac{\partial E}{\partial x_k} \frac{\partial x_k}{\partial w_{kj}} \qquad (2.9)$$

誤差の変化をニューロンへの総入力の変化の関数として

$$-\frac{\partial E}{\partial x_k} = d_k \qquad (2.10)$$

と表すことができる．ここで，h を学習定数とする．したがって

$$\Delta w_{kj} = h d_k y_j \qquad (2.11)$$

となる．つまり，重みを$d_j \cdot o_j$ に比例して更新することにより E を減少させることができる.

$$d_k = -\frac{\partial E}{\partial x_k}$$

$$= -\frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial x_k}$$

$$= -\{-(d_k - o_k)\} f'(x_k)$$

$$= (d_k - o_k) f'(x_k) \qquad (2.12)$$

出力層以外の重み変更方法は期待出力が存在しないため，同様の処理が不可能である.

そのため，出力層以外の場合は連鎖則を再び用い，

$$\Delta v_{ji} = -h\frac{\partial E}{\partial w_{ji}}$$

$$= -h\frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial x_j}\frac{\partial x_j}{\partial w_{ji}}$$

$$= -h\frac{\partial E}{\partial o_j} f'(x_j) y_j$$

$$= -h\sum_k \frac{\partial E}{\partial x_i}\frac{\partial x_i}{\partial o_j} f'(x_j) y_j$$

$$= -h\sum_i \frac{\partial E}{\partial x_i} w_{ji} f'(x_j) y_j \qquad (2.13)$$

ここで，式（2.14）より

$$= h\sum_i d_i w_{ji} f'(x_j) y_j \qquad (2.14)$$

よって，式（2.15）に対応させて記述すると

$$d_k = f'(x_j)\sum_i d_i w_{ji} \qquad (2.15)$$

となる．この重みの修正方法を BP 法と呼ぶ．さらに最急降下法の収束を早めるために上記に BP 法に現在の重みの変化を考慮し，慣性項と振動項を付加する．本論文では式（2.16）を NN の学習アルゴリズムとして採用する．以降，改良型 BP 法と記述する．

$$\Delta W_{kj}(t) = -hd_k O_k + a\Delta W_{kj}(t-1) + b\Delta W_{kj}(t-2) \qquad (2.16)$$

式 2.16 で t は学習回数，αは慣性項の比例定数，βは振動項の比例定数を示す．慣性項を付加することによりより学習初期には誤差が局所的に増加するような方向へも変化が起こり，その後全体的な最小値へ向かうことが期待できるからである．すなわち，学習初期でネットワークがローカルミニマムに与えられる機会を減少させることができる．また，振動項は極小値から脱出させる働きをする．これは Asakawa らがバネ－ダッシュポット系として，振動項を導出し，図2.5 に示す慣性項・振動項の境界条件を設定した[5]．したがって，慣性項・振動項について，その領域内で線形計画法と同じような形で任意の組み合わせをその範囲内から試行錯誤的に選択すればよい．この手法を用いて，竹田らは紙幣識別において，最も早く安定した状態で学習収束状態に到達した組み合わせ（α＝0.95，β＝－0.1）を算出した[6]．本論文ではこの組み合わせを採用し，学習を行なう．

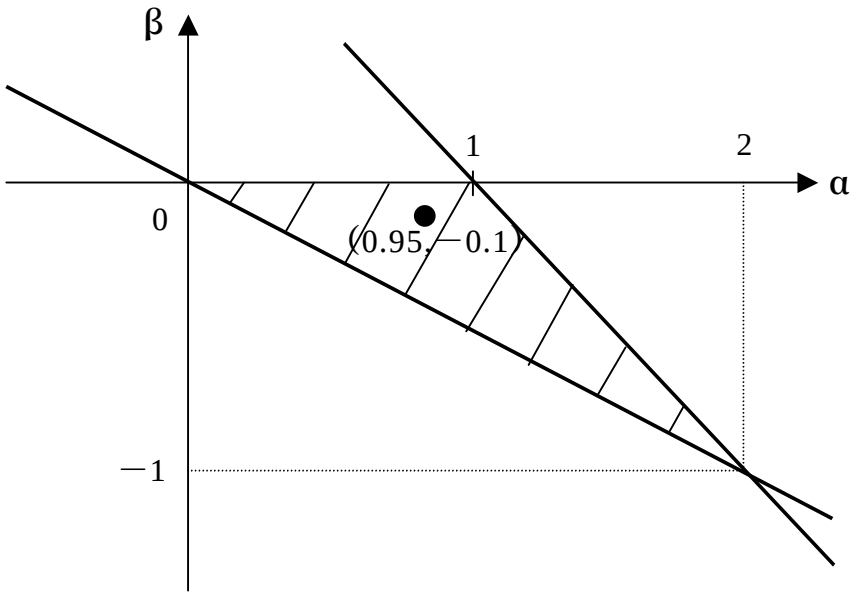


図 2.5 慣性定数と振動定数の範囲

2.2 評価

階層型 NN は階層的なニューロンの組み合わせにより形成されている．また，個々のニューロ細胞への入力は図 2.2 に表すようになっている．学習により NN ウエイトの重みが調節されている．評価とは，評価対象をスラブ値として入力し出力層の発火するニューロ細胞が NN の出力となる．この際に，学習により修正された NN ウエイトを用いることにより正しい出力を得ることが可能である．

第3章 DSP ボード

本稿では，これまでに開発した紙幣識別用 DSP ボードを使用する．また，紙幣識別用 DSP ボードはプログラム可変型ボードである．本章では，紙幣識別用 DSP ボードのハードウェアとソフトウェアについての説明を行う．

3.1　シミュレーションシステム

シミュレーションシステムは学習を行うための NN システムを PC 上で実行するためのプログラムである．NN の学習のための種々の設定ファイルを有し，学習を行う対象に応じて環境を設定することが可能である．以下に設定ファイルの説明を行う．

①環境設定ファイル

環境設定ファイル“NEURO.CFG”（テキスト形式）である．NN の構成，学習パターン数などを項目記述する．設定する項目と設定値の範囲は表 3.1 に示す

ファイルはテキスト形式であり，1 行に 1 項目とし，数値の右側にスペースまたはタブで区切ってコメントを書く，コメントは半角 32 文字（全角 16 文字）までとし，省略はできない．設定の具体例を図 3.1 に示す．

表 3.1 NEURO.CFG の設定

行	項目	設定値（x）の範囲
1	入力層の細胞数	2≦x≦50（整数）
2	中間層の数	x＝1（整数）
3	中間層の細胞数	2≦x≦50（整数）
4	出力層の細胞数	2≦x≦50（整数）
5	学習定数の初期値	x＝0．05
6	慣性定数	x＝0．95
7	振動定数	x＝－0．1
8	学習パターン数	2≦x≦50（整数）
9	1パターンの学習枚数	2≦x≦65534（整数）
10	最終誤差判定値	0．0＜x
11	最大学習回数	2≦x≦65534（整数）

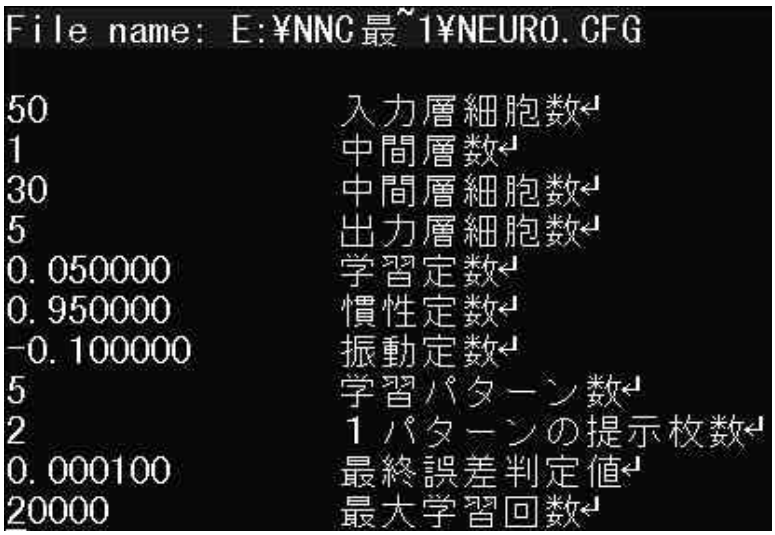


図 3.1 NN 環境設定ファイル NEURO.CFG

②学習データベース設定ファイル

学習データベース設定ファイルは“DATABASE.CFG”（テキスト形式）である．データベース

ファイルのパス名をパターン番号順に 1 行に 1 ファイル記述する. 設定の具体例を図 3.2 に示す.

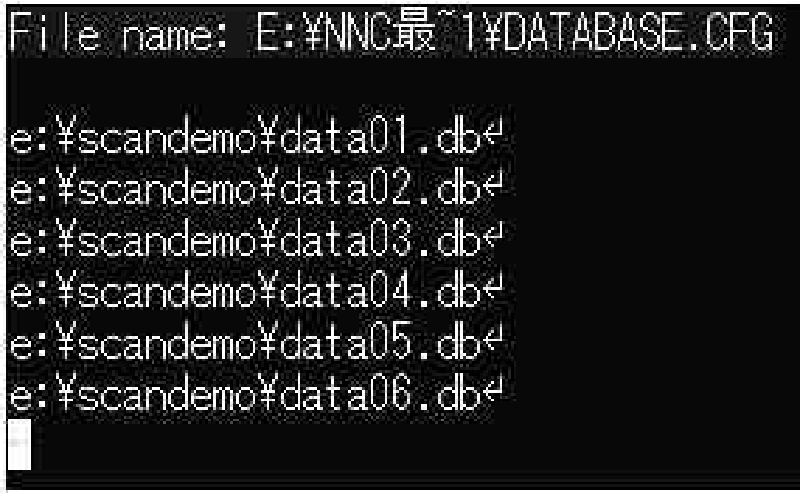


図 3.2 学習データベース設定ファイル DATABASE.CFG

学習データはこの学習データベース設定ファイルに基づいて抽出されるので, 記述する順番は重要である. 学習データは, ニューロフォーマットデータとして保存される. 表 3.2 は NN データのフォーマットを示している.

表 3.3 は 1 枚分のフォーマットで, ヘッダ部分 8 バイトとスラブ値データで構成される. 各々のデータはすべてバイナリ形式である. ここでスラブ値とは NN への入力値のことで使用する識別対象により適した形に前処理を行ったデータをさ示す.

表 3.2 NN データの 1 サンプルフォーマット

ヘッダ情報	64 byte	
ダミーデータ	400 byte	
SLAB値データ	入力層細胞数 50	200 byte
	入力層細胞数 80	320 byte

表 3.3　ヘッダ情報のフォーマット

予約領域 1	（4byte）
データ 1枚分の総サイズ	（4byte）
パターン番号	（1byte）
金種	（1byte）
方向	（1byte）
国情報	（1byte）
通し番号	（2byte）
機種	（1byte）
号機	（1byte）
種別	（1byte）
センサ番号	（1byte）
画像のXサイズ（画素数）	（2byte）
画像のYサイズ（画素数）	（2byte）
センサ有効域チャネル番号（L）	（2byte）
センサ有効域チャネル番号（H）	（2byte）
予約領域 2	（2byte）
センサ分解能（X方向）	（1byte）
センサ分解能（Y方向）	（1byte）
予約領域 3	（6byte）
紙幣の中心画素のX座標	（2byte）
紙幣の中心画素のY座標	（2byte）
画像データのオフセット	（4byte）
予約領域 4	（4byte）
SLAB値の種類数	（1byte）
1種類目のSLAB値のマスクID	（1byte）
1種類目のSLAB値の個数	（1byte）
予約領域 5	（5byte）
SLAB値のオフセット	（4byte）
予約領域 6	（4byte）

・ヘッダ部分の各項目はデータベースファイルの内容と同じである．

・スラブ値は IEEE 浮動小数点フォーマットである．

・バイナリーデータへ変換する際の変換方法は識別対象によって異なる．

3.2　ハードウェア

本論文で使用する紙幣識別用 DSP ボードに演算装置として組み込まれている DSP はテキサス・インスツルメンツ（TI）社製 TMS320CPQL である．本節では，TI 社製 DSP（TMS320CPQL）と，その開発環境に視点を置き DSP の NN への応用性の高さについて説明を行う．また，実験で用いる紙幣識別用 DSP ボードについても説明を行う．

3.2.1　DSP

DSP はデジタル信号処理（リアルタイムでの音声処理を目的としている）のために生み出されたプロセッサであり，その開発には汎用プロセッサで培われた技術が投入されている．第一世代の DSP が登場したのは 1970 年代後半から 1980 年代初頭である．これにより，汎用大型計算機や専用のハードウェアでしか実現されていなかったリアルタイムでのデジタル信号処理などが，民間レベルでも簡単に行えるようになった．処理速度の向上としては，1999 年には第一世代の約 320 倍と処理速度が上昇した．このことにより，開発当初考えられてきたリアルタイムでの音声処理の分野以外に応用がされている．

本論文で用いられている DSP は TI 社製 TMS320CPQL である．その特徴を以下に列挙する．

性能[7]

- 60ns（1 マシンサイクル）の命令実行時間
- 33.3MFLOPS
- 16.7MIPS
- 4K×32 ビットの 1 サイクルに 2 回アクセス可能なオン・チップの ROM
- 1K×32 ビットの 1 サイクルに 2 回アクセス可能な RAM ブロックが 2 個
- 64×32 ビットの命令キャッシュ
- 32 ビットの命令データ・ワード
- 40/32 ビットの浮動小数点・整数の乗算器と ALU
- 32 ビット・パラレル・シフタ
- 8 個の拡張精度レジスタ
- 8 この補助レジスタを持つ 2 個のアドレス生成ユニットで，2 個のアドレスを発生
- 整数演算，浮動小数点演算，および，論理演算
- 単一サイクルで ALU と乗算器で並列命令

この性能は，50/60ns とういう命令実行サイクルと共に，命令用キャッシュ・メモリの内蔵，パイプライン処理と命令の同時実行という 2 種類の並列処理によって達成されている．さらに，メモリ空間は 16M ワードとなっている．外部バスも 2 系統独立で設けられており，外部メモリなどに対しても並列的なアクセスが可能である．

また，システムを低コストで実現するため，周辺デバイスの制御のための DMA コントローラ，シリアル・ポート，タイマが内蔵されている．

浮動小数点の基本的なデータ形式は，指数部 87 ビット，仮数部 24 ビットとなっている．この精度は IEEE の単精度データの規格とデノーマル表現を除くと同じになっている．また，DSP の内部の ALU や演算用のレジスタは，仮数部が 32 ビットに拡張され，乗算の切り捨て誤差の発生を減らすように工夫されている．

これらの特長を利用することにより，行列の前向き計算で表すことが可能である NN の処

理を高速に行うことが可能である．また，DSP は内部データをすべて 4 ビット長で表現し，さらに浮動小数型の変数を扱うことが可能である．

次に，TI が開発している DSP の開発環境について，その汎用性と実用性の説明を行う．図 3.3 は DSP プログラムのアセンブラレベルまでの開発の流れを図示している．本論文では C ソースファイルからの開発を行っている[8],[9],[10]．本項では，その開発環境について論議を行う．

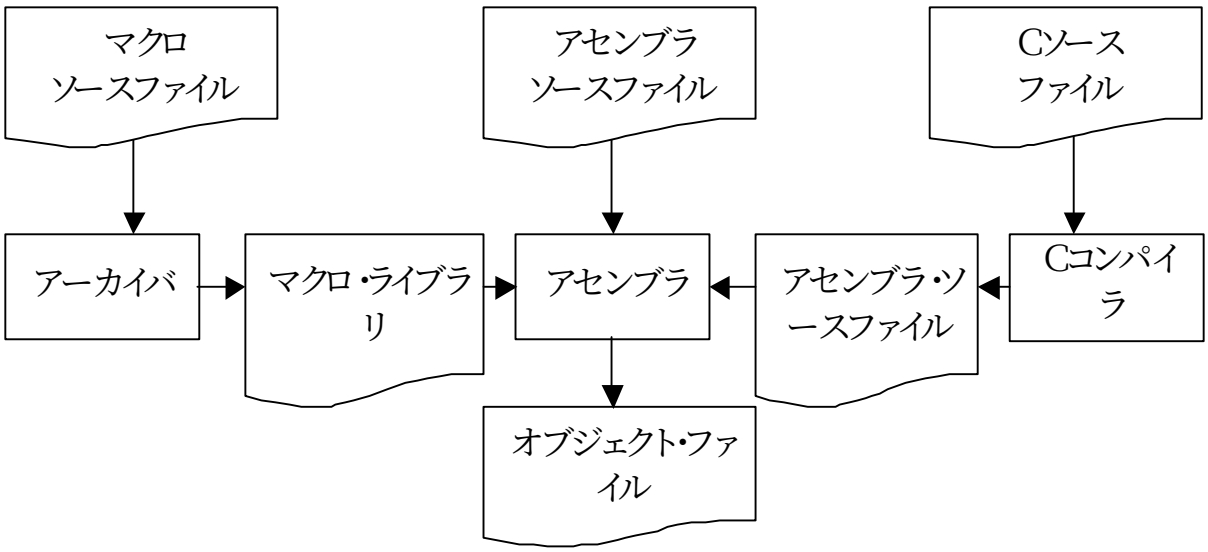


図 3.3 オブジェクトファイルまでの開発手順

オブジェクトファイルから実行可能なオブジェクトファイルまでの開発の流れを，図3.4に図示する．

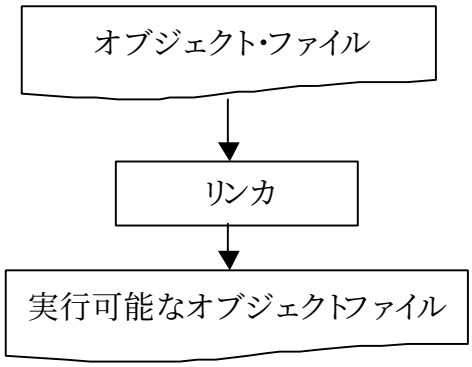


図 3.4 実行可能なオブジェクトファイルまでの開発手順

ここからは，TI が提供するさまざまな開発ツール列挙し開発の簡要さについて議論する．また，DSP プログラムの開発環境の流れについても記述する．

①C コンパイラ

DSP 用 C コンパイラは，C プログラムを DSP 用のアセンブラプログラム（ASM）へと変換を行う．

TI が提供する DSP の C コンパイラは ANIS－C 仕様の完全準拠（Kernighan&RiChie にも準拠可能）である．また，Cソースからアセンブラを生成出力するプリコンパイラで，Cソースから実行可能コードへ 1 ステップで返還できるシェル･プログラムである．機能としては，シンボリック･デバッグや，種々のリスト･ファイルを作成することなどが可能である．DSP ではASM と実行サイクルが完全に対応しており，コンパイラにより変換された ASM の実行サイクル数がプログラム実行速度となる．高速なプログラム実行を満たすためには，コンパイラのプログラム変換過程におけるオプティマイザが重要となる．DSP メーカーは，最適化 C コンパイラの改善を，新しい Version により対応している．これにより固有のハードウェア（繰りかえし演

算，データ転送，並列動作）を引用した最適な変換が行われるようになってきているが，C プログラムのアルゴリズム等の諸条件に左右される場合が多い．現時点では，コンパイラから最速のプログラム実行速度を得ることは困難である．

②リンカによるメモリマッピング

PC の場合，OS によりメモリ配列が制御されるが，DSP の多くの場合組み込みシステムの中でデジタル信号処理を実現している．したがって，組み込みシステムでは実現しようとする機能に特化したハードウェア構成設計がなされている．そのため，DSP にプログラミングにおいてはハードウェアを動かすソフトウェアもハードウェアに強く依存している．PC とは異なり組み込みシステムは，OS によりメモリマッピングを制御されていない．そのことから，DSP プログラムミングにおいては，メモリマッピングは重要なウエイトを占める．C 言語によるプログラミングを行う場合に組み込みシステムの特徴を反映したプログラミングを行えば，組み込みシステムの性能を十分に引き出すことが可能である．

③シミュレータ

シミュレータは PC/MS-DOS と SUN ワークステーションをホスト・コンピュータとして TMS320C3x の動作環境を模倣するソフトウェア・プログラムである．HLL（高級言語）デバッガによるアセンブラをＣソースコードレベルで行うデバッグ，主な周辺機能（DAM，タイマ，シリアル・ポート）のシミュレーション，ホスト・コンピュータ・ファイルからファイルへのバス，シリアル・ポート，割り込み入出力のシミュレート，キャッシュ使用のシミュレートなどが可能である．また，シミュレートの際には，シングルステップ実行，ソフトウェア・ブレイクポイントの設定，実行サイクル計算が可能である．その他に，プロファイラ機能を備える事によりアセンブラ，C 言語プログラム両方の行，関数，範囲を指定し，その部分の実行統計を表示することにより，プログラムの開発の際にどの部分にどのくらい CPU 負荷がかかっているかを測定することができる．これにより，効率アップのための再プログラミングをすることが可能である．これらの

機能を備えたシミュレータを使用することにより, DSP 上でプログラムを実行することなく, PC 上で開発中の DSP プログラムのデバッグ, および効率アップのための再プログラミングが可能となっている.

④エミュレータ

エミュレータ (XDS-510XL/WA) は TMS320C3x を最高速でイン・サーキット・エミュレーション (ICE) を行うために必要な機能をすべて備えた開発システムである. ホスト・コンピュータは, 386 以降の IBM PC と SUN4 または SPARC ステーションである. IBM PC の場合, OS に Windows が必要であり, 使用可能なメモリおよびハードディスク環境が必要である. また, ISA/EISA バスの 1 スロットに DSP と通信を行うためのインターフェイス用 PC カードを挿入し使用する. SUN4 または SPARC ステーションにおいて, OS として SUNOS ver. 4 1 以降, OpenWindows ver.3.0 以降が必要であり, DSP との通信ケーブルを SCSI コントローラに接続し使用する. スキャンパス・インターフェイスによるターゲット・システム上の TMS320C3x を最高速で動作させ, モニタリングを行うことが可能である. また, シミュレータ同様 HLL デバッガによるアセンブラを C ソースコード・レベルでデバッガを行うことが可能である.

またこの際に, 使用できる機能として以下に示すさまざまな機能が使用可能である.

- 30 個のソフトウェア・ブレイクポイントの設定,
- シングルステップ実行,
- ターゲット・システム上の TMS320C3x のすべてのメモリとレジスタをロード, 検査, 修正,
- ターゲット・システム上へプログラムデータおよびメモリ・データをアップロード, ダウンロード
- リアルタイムでの実行時間クロック・サイクル測定
- プロファイル機能の使用

エミュレータは, シミュレータと同様にプロファイラ機能を備えることによりアセンブラ, C 言語プログラム両方の行, 関数, 範囲を指定する. その部分の実行統計を表示することにより,

プログラムの開発の際にどの部分にどのくらいCPU負担がかかっているかを測定することができる．これにより，効率アップのための再プログラミングをすることが可能となる．

これらの機能を備えたエミュレータを使用することにより，容易にソフトウェアとハードウェアの両方を開発するが可能となり，開発中であるDSPプログラムが効率を上げるために再プログラミングすることが可能となる．

⑤Code CompOSer Studio

Code CompOSer StudioはDSPアプリケーション・ソフトウェア開発の全ステージをカバーする統合開発環境である．このソフトウェア・パッケージにはコンパイラ，アセンブラ，リンカ，シミュレータ，エミュレータ用デバッガが含まれ，一つの統合環境上でソースのエディット，コードの生成，デバッグ等を行うことができる．また，ターゲットDSP上で動作する極小のOS"DSP/BIOS™"も付属しており，これによりターゲット・アプリケーションのリアルタイム解析，およびタスクのスケジューリングを容易に行うことができる．これらの機能を併せもつCode CompOSer Studioを使用することにより開発の効率化を実現している．

⑥サードパーティー・プログラム

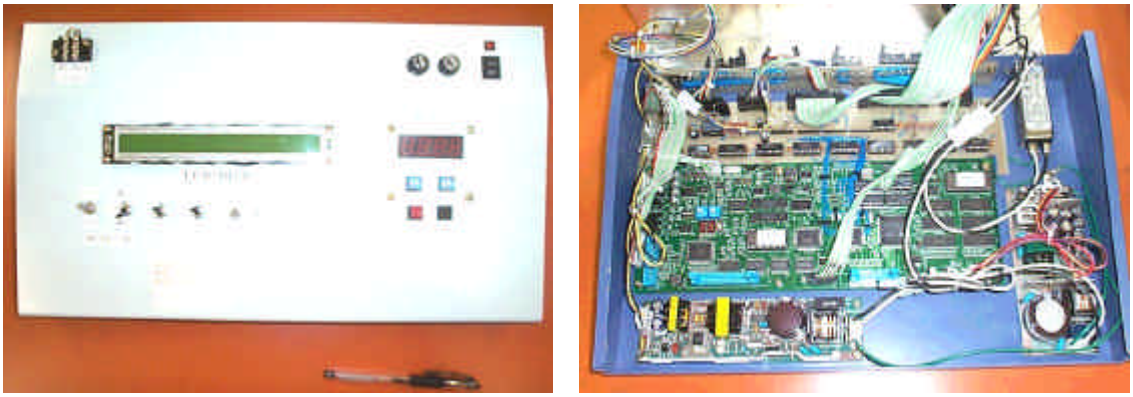
TIはTDSPに関する開発サポートを業界最大の規模で提供している．また，現在600社以上のサードパーティーが，開発ツール，ソフトウェア・アルゴリズム，ライブラリー，ボード製品，コンサルティング，依託開発といったサービスを全世界で展開している．日本においても約80社がTIのサードパーティー・ネットワークに参加している．1000を超えるサードパーティー製品によるサポートで，用途にあわせた開発の効率化と，製品の早期市場投入を実現している．

DSPは，以上に示したようなアーキテクチャおよび開発環境を提供されており，CPUと比較して，NNシステムや音声処理などのリアルタイムでの信号処理を実現する組み込みシステムを構築する際には，非常に有効な演算部となる．

3.2.2　プログラム可変型 DSP ボード

紙幣識別用 DSP ボードはプログラム可変型 DSP ボードであり，ユーザの仕様にあわせてフラッシュメモリに保存されているユーザプログラムを書き換えることが可能である. (11),(12),(13),(14),(15)

図3.5 にプログラム可変型 DSP ボードの外観を，図3.6 プログラム可変型ボードの内部を，図 3.7 にプログラム可変型ボードのブロック図を示す.



ⓐ DSP ボード外観　　ⓑ DSP ボードの内部

図　3.5 DSP ボード

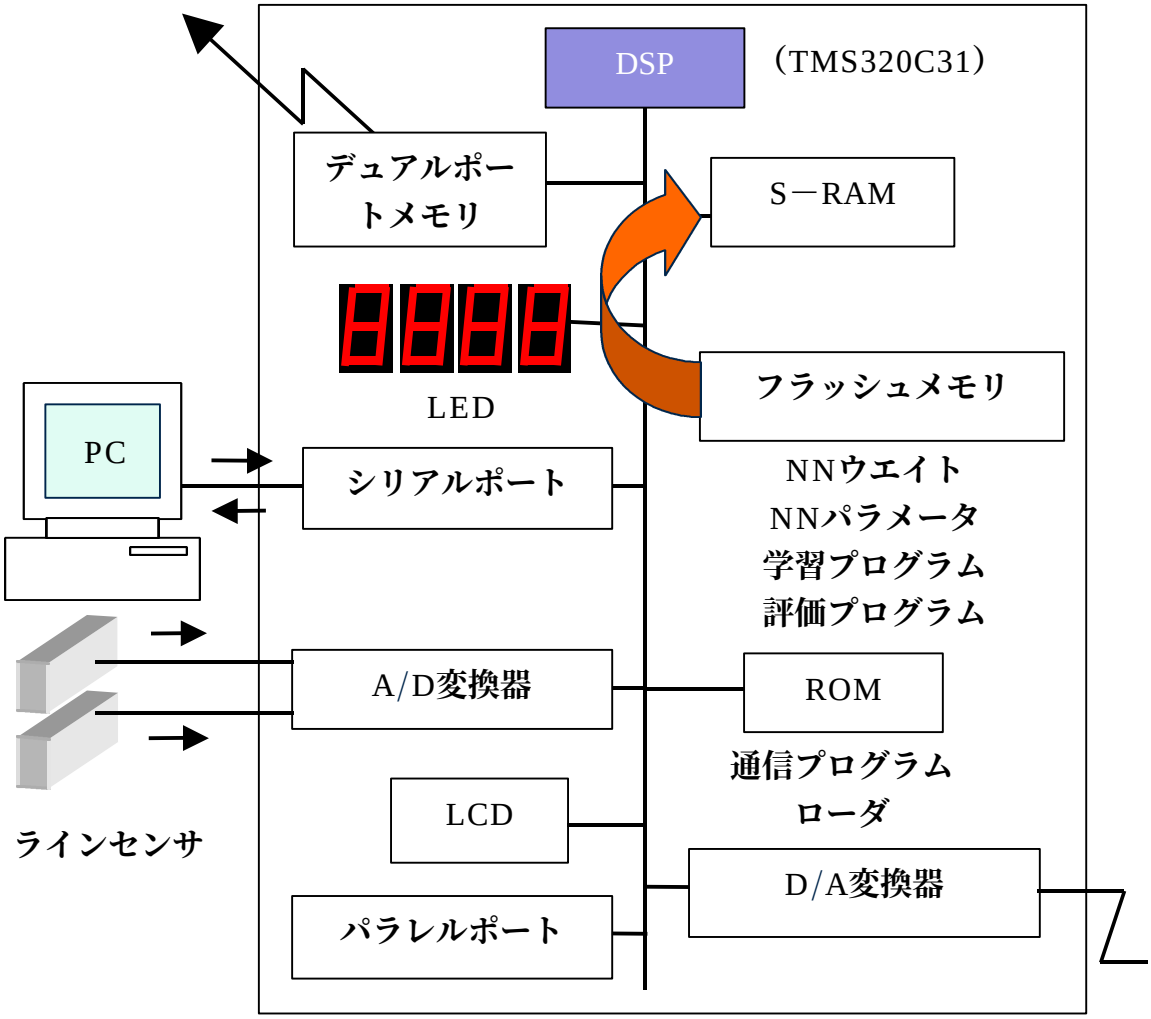


図　3.6　DSP ボードブロック図

プログラム可変型 DSP ボードは，識別および制御などの各種使用に対応するためにモード設定スイッチ，LED，液晶表示板（LCD），D/A 変換器，A/D 変換器，パラレル通信ポート，デュアルポートメモリ（DPM）への延長ポートを実装する．LED および LCD は NN の学習状況，動作モード，さらに，エラーなどが表示される．D／A変換器はボードから得られる制御信号で直接モータなどのアクチュエータ類を起動する場合に使用される．DPM は本来，上位のメカニズム制御用 CPU との通信に使用される．さらに，DPM への延長ポートは PC の内

部バスと専用ケーブルで接続することにより，イメージスキャナやデジタルカメラなどで得られるデータについて PC を媒介してボード上でアクセス可能としている．

3.2.3　フラッシュメモリ上のメモリマッピング

フラッシュメモリはボード上に 4 ページあり，1 ページは 128 ブロックで構成されている．4 ページ目のメモリマッピングは，その上位 64 ブロックがプログラムパラメータ（センサ補正情報，特徴抽出情報など）領域である．一方下位 64 ブロックには NN ウエイトを格納している．このメモリマッピングにより，外部 PC による NN プログラム，およびプログラムパラメータ，NN ウエイトのダウンロードが可能である．また，プログラムパラメータ，NN ウエイトにおいては，アップロードが可能である．

NN ウエイトは，学習途中に 4 ページ目の下位 64 ブロックをすべて使用する個数を保存する．保存された NN ウエイトは飽和状態になると 4 ページ目の下位 64 ブロックを初期化し再度この作業を実行する．また，学習が収束すると 4 ページ目の下位 64 ブロックは初期化され，学習収束ＮＮウエイトとして保存する．図 3.7 にメモリマッピングを示す．

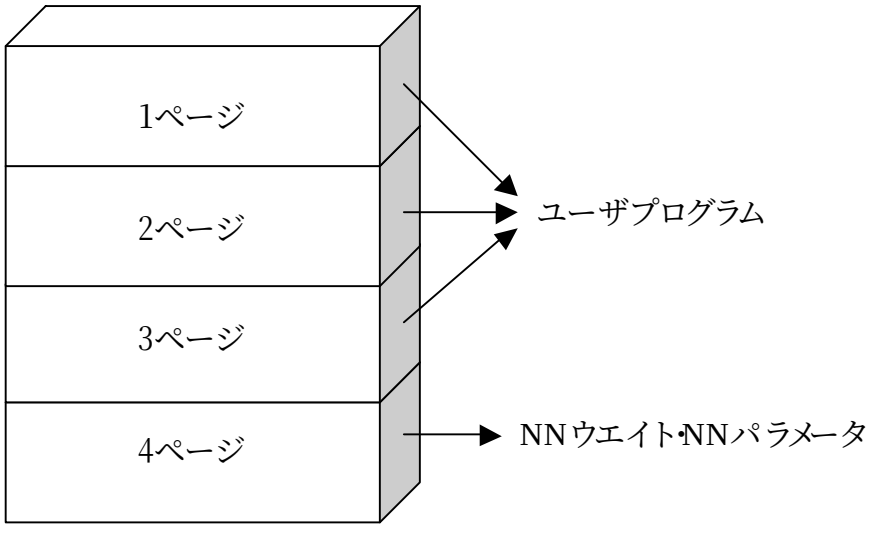


図　3.7　メモリマッピング

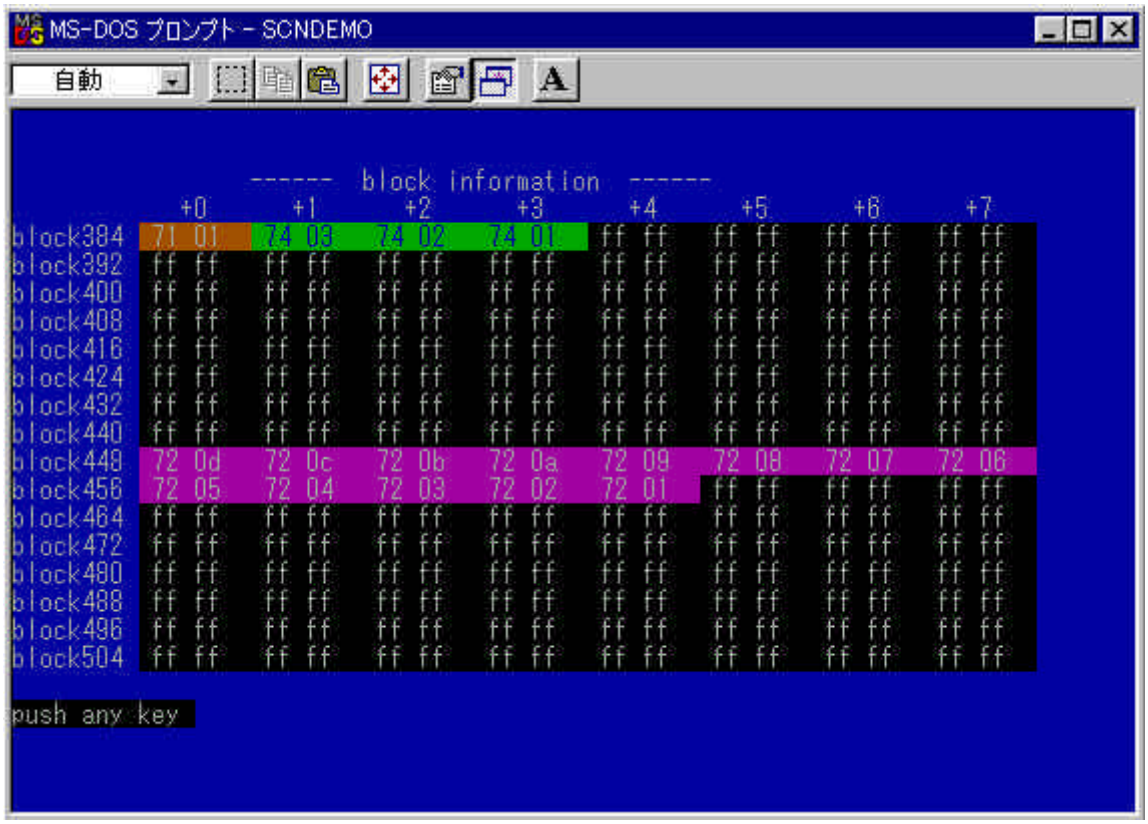


図 3.8 通信プログラムによるフラッシュメモリ参照画面

図 3.8 は，フラッシュメモリ 4 ページ目に保存されているデータを，通信プログラムを使用して参照した．71 で始まるデータはセンサ補正情報であり，74 で始まるデータは特徴抽出情報，72 で始まるデータは，NN ウエイトを示している．

3.2.4 DSP ボードのブートシークエンス

モード設定スイッチにより EPROM，またはフラッシュメモリから DSP ボードをブートするか選択することが可能である．

これにより，フラッシュメモリを初期化した状態において，EPROM からの起動により DSP ボード単体での学習，シリアル通信を利用しての NN プログラムのダウンロードが可能である．

3.3　ソフトウェア

本節では，プログラム可変型 DSP ボードのソフトウェアにおける NN システムへの適応性について検証する．

3.3.1　通信プログラム

通信プログラムは C 言語で記述されており，プログラム可変型 DSP ボードと PC 間をシリアルケーブルで接続し，PC からコマンドパケットを送信するプログラムと，DSP からの処理処理結果をレスポンスとして PC に返す機能を有している．また通信プログラム単体で NN ウエイトなどの作成を行えるように，DSP とのオフラインでの起動が可能である．

動作モードは DSP ボード本体でモード設定スイッチにより設定を行う，

以下に各種の動作モードについて説明を行う．

①通信モード

PC と DSP ボード間のシリアル通信を行う．これにより，NN ウエイトおよびそのほかのパラメータの PC とのダウンロード，およびアップロードが可能となる．

②学習モード

PC 側から DSP ボードに学習対象となるデータを送信し，DSP により学習を行う．また，学習の程度などを PC のディスプレイに表示することが可能である．

③識別モード

PC から評価対象のデータを DSP ボードに送信し，NN のウエイトによる前向き計算を行い，評価結果を PC のディスプレイに表示する．

また，以下に学習モードの違いについて説明を行う．

①オンライン学習モード

識別機などの制御対象を動作させながら学習を実現する．学習途上のウエイトで制御対象に入力を与え，制御対象から動作情報を得る．この情報を新たな NN の学習データとして

用いる学習方法である．

②オフライン学習モード

オフライン学習モードは，本論文で使用する学習方法である．PC 上でシミュレーションシステムとして行ってきた学習動作をボード上で行う．固定した学習データで学習が行われる．

特にオンライン学習とオフライン学習の違いは，オンライン学習の場合学習途中で学習データが更新されるのに対して，オフライン学習は学習が完了するまで学習データは更新されないという点である．モード決定スイッチと起動モードの関係を図 3.9 に示す．

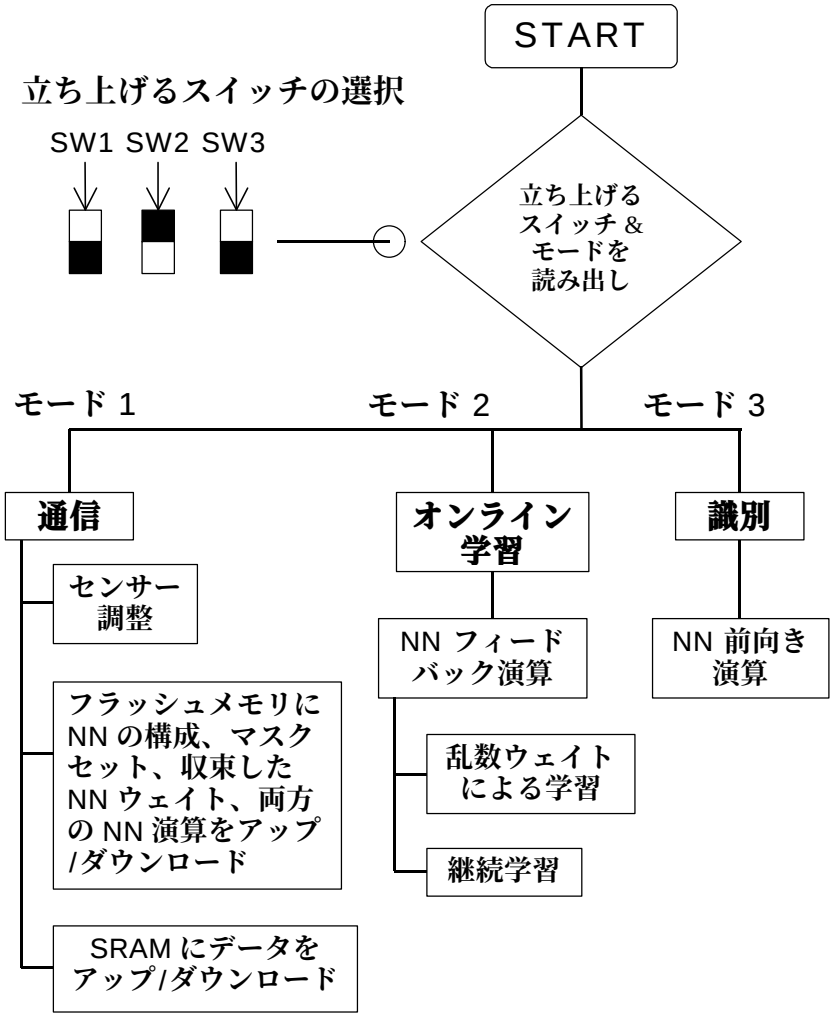


図 3.9　モード設定スイッチと起動モード

3.3.2　NN プログラム

本論文で使用する NN プログラムは学習部分については主に C 言語で記述されており，プログラム可変型 DSP ボード上で動作するユーザプログラムである．また，リアルタイム処理が必要とされる識別処理についてはアセンブラレベルで最適化処理を施している．　NN プログラムの主な機能は 2 種類ある．シリアル通信を使用して PC と DSP ボード間の通信を行うための機能と DSP ボード上で NN を使用する学習機能である．

前者の機能は，ときにシリアル通信を使用して PC とのデータの受授するために，通信プログラムが通信モード，学習モード，評価モードに設定されている．これにより，DSP ボードと PC 間では NN ウエイトや，その他の仕様書情報，NN プログラムなどをダウンロード，アップロードすることが可能である．

後者は，著者らがこれまでにシミュレーションシステムで開発してきた NN プログラムの学習や評価モードをアセンブラレベルで DSP に適するよう最適化を行い，プログラム可変型 DSP ボードに移植したものである．

3.4　シミュレーションシステムの DSP ボードへの移植

本節では，ボードへの学習アルゴリズムの NN の構造およびパラメータ決定，学習の収束判断について述べる．また，シミュレーションシステムにおける DSP への移植について説明を行う．

DSP の NN プログラムは，シミュレーションシステムを元に C 言語により開発を行った．開発の流れとしては，C 言語により記述されたソースファイルを TI 社の提供する C コンパイラによりコンパイルを行い，オブジェクトファイルを作成する．また，リアルタイム処理が要求される識別を行うプログラムはアセンブラ言語開発を行う．次に，リンク作業を行い実行可能であるオブジェクトファイルにする．その後，DSP の ROM へと格納可能なインテルヘキサ形式に変

換を行う．インテルヘキサ形式に変換されたプログラムはフラッシュメモリに格納される．

この際，シミュレーションシステムで用いた設定ファイルは通信プログラムが読み込みを行い，DSP 学習環境ファイルの設定を DSP ボードに反映する．また，環境設定ファイルは DATABASE/CFG と SLABE.CFG という 2 つのファイルからなり，DATABASE.CFG をニューロフォーマットデータ・データベースファイルのパス名を記述するものとし，SLABBASE.CFG はスラブ化されたデータベースファイルのパス名を記述するものとする．

プログラム可変型 DSP ボード上での NN の構造決定は，NN の構造パラメータを PC 上の通信プログラムを使用してメニュー形式で作成する．初期値は通信プログラムが読み込む NN 設定パラメータファイル（PC シミュレーションシステムでは NEURO.CFG に相当する）内に記述され，通信プログラム起動後メニューを選択することによりディスプレイに初期値が表示される．次に，会話形式でパラメータの変更をメニュー上で実施する．基本的な入力パラメータの確認を実施後に，その内容はウエイトファイルに格納され，通信プログラムによりシリアル通信を介して DSP ボードのフラッシュメモリの 4 ページ下位 64 ブロックにダウンロードされる．DSP は起動した時点でフラッシュメモリの内容を読みとり，ウエイトファイルパラメータ領域に書き込まれている更新時間情報を読み込む．多数あるウエイトファイルの中でも，更新時間の新しいウエイトファイルにおける NN 構造は SRAM に展開されることとなる．

NN の入力変数の設定においては以下の手続きで実施するものとしている．まず，NN の入力変数としては学習定数，可変温度勾配 T，学習回数，収束判定値，学習データ数などがある．これは NN の構造情報としてウエイトファイルに格納され，NN の構造と同様に PC からフラッシュメモリにダウンロードされる．

PC と DSP 間の通信は C 言語で記述された SCNDEMO で行う．これにより，NN ウエイトファイルなどのパラメータや NN プログラムなどのユーザプログラムのダウンロード，およびアップロードを実現している．さらに，SCNDEMO はシミュレーションシステムと同様の学習，識別画面を有している．

DSP ボードの NN ウエイトは通信プログラムを用いてアップロード・ダウンロードが可能であ

る. NN ウエイトは NN ウエイトファイルヘッダを修正することにより, PC と DSP ボードの両 NN システムにおいて使用が可能である. これは移植に際して PC と DSP ボードの NN システムに互換性を持たせる意味を持つ.

第4章　DSP ボードを用いたオンラインチューニング実験

本論文で提案するオンラインチューニングとは，まず PC を用いあらかじめ様々なばらつきを有する学習データから様々なデータに対し，一定の汎化能力を有する NN ウエイトを作成する．次に，作成した NN ウエイトを DSP ボードにダウンロードし，データの追加と継続学習を行うことにより NN ウエイトをチューニングすることである．これにより使用対象システムに特化した認識能力を有する NN システムの構築を可能となると考えられる．

本章では，評価対象として EMG データとタイ紙幣[16],[17],[18]を用いてオンラインチューニングの基幹要素である継続学習の基本性能確認実験，学習データ増加にともなう認識率確認実験を行う．また，EMG パターン識別[19],[20],[21],[22]については PC の NN システムと DSP ボードの NN システムの互換性を示すため性能比較実験を行う．

4.1　タイ紙幣識別システムを用いた実験

本項では，タイ紙幣識別システムを実験対象システムとして次のオンラインチューニング基幹要素確認実験を行う．PC の NN システムを用いて継続学習の基本性能確認実験を行う．タイ紙幣識別システムとはタイ紙幣に特化した識別が行えるよう開発を行われた NN システムである．

4.1.1　概要

はじめに，5 金種のタイ紙幣の表面，裏面をビットマップ形式画像データとして，スキャナーで PC に取り込む．次に，PC を用いて取り込まれたビットマップデータを 256 階調のモノクロビットマップデータに変換する．次に NN システムでビットマップデータを扱えるように NN データにデータ変換を行う．さらに，NN データにニューロフォーマットの仕様情報を記述したヘッダを付加し，NN システムで扱える状態（ニューロフォーマット）にする．次に PC の NN システムを使用し，10 枚の学習データを用い初期学習を行う．その後，初期学習で作成さ

れたNNウエイトを用い，さらに未学習の10枚の学習データを用いて継続学習を行う．最後に，初期学習と継続学習で作成されたNNウエイトの認識率を未学習評価データ（1パターンにつき80サンプル）を使用し確認する．

ここでニューロフォーマットにデータを変換する際に使用するスラブ値は，紙幣画像をマスクで部分的に被覆し，被覆されない領域の画素値の合計を意味する．紙幣パターンについては紙幣 1 金種につき，正面（正立，倒立）画像，裏面（正立，倒立）画像が存在することから識別パターンとして4パターンが必要である．ここで，紙幣の正立画像と倒立画像についてマスクを同スラブ値となるように設定したため1金種につき識別パターン数は2となる．

4.1.2　実験条件

表4.1に示す条件設定を行う．

表　4.1　実験条件

紙幣種類	5
紙幣パターン	10
学習アルゴリズム	改良型BP法　式（2.17）
NN構造	50×30×10
学習定数η	0.05
慣性定数α	0.95
振動定数β	－0.1
1パターンあたりの評価サンプル数	80
最大学習回数	20000
最終誤差判定値	1.0E－4

式（4.1）に示すとおり目的データを NN に提示した際，目的対象パターンに出力ユニット

が最大発火したパターンを正解とする.

$$\text{認識率} = \frac{\text{目的対象動作が最大発火した個数}}{\text{未学習データ数}} \times 100 \quad (4.1)$$

4.1.3 実験結果

表 4.2 に実験結果を示す.

表 4.2 評価結果

紙幣タイプ	認識率 (%)	
	初期学習	継続学習
20 HB	100.00	100.00
20 TB	100.00	100.00
50 HB	95.00	95.56
50 TB	98.89	100.00
100 HB	98.75	98.89
100 TB	100.00	100.00
500 HB	100.00	100.00
500 TB	100.00	100.00
1000 HB	100.00	100.00
1000 TB	100.00	100.00
平均	99.26	99.45

HB:バーツ表

TB:バーツ裏

4.1.4 考察

継続学習を行ったところ, 平均での認識率が上昇した. これは, NN システムの NN ウエイ

トがよりタイ紙幣の認識に適した形で，学習が収束したためと推測される．認識率の上昇により，継続学習の有用性を確認することができた．

4.2 EMG パターン識別システムを用いた実験

本節では，EMG パターン識別システムを実験対象システムとして，オンラインチューニングのための基礎実験を行う．これにより学習データ増加にともなう認識率変動，また PC の NN システムと DSP の NN システムとの互換性を示す．

EMG パターン識別システムは EMG 波形に特化した識別が行えるよう開発を行われた NN システムである．オンラインチューニングを EMG パターン識別において用いることにより，PC の NN システムでは認識率が低かった問題を解消できるものと考えられる．

4.2.1 概要

はじめに，手首に装着した 4 つの電極より EMG 信号を差動アンプで信号増幅処理を行い PC に取り込む形により採取する．次に，PC を用いて FFT 変換を行い NN システムに適したスラブ値に変換する．ここでニューロフォーマットにデータを変換する際に使用するスラブ値は EMG 信号を FFT 処理したパワースペクトルの平均値である．図 4.1 にシステムの構成図を示す．

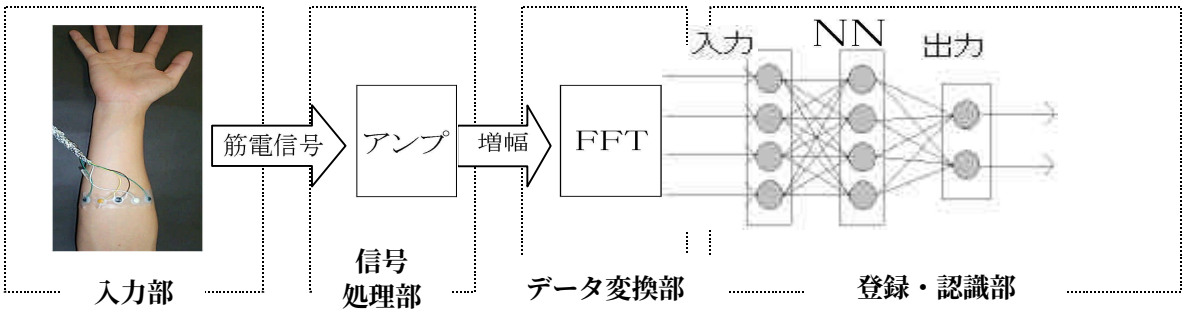


図 4.1 システム構成図

学習データ増加に伴う認識率確認実験では6動作（背屈，掌屈，回内，回外，開き，握り）それぞれつき5サンプルの学習と10サンプルの学習を行い，認識率をそれぞれ同じ未学習評価データ（1パターンにつき20サンプル）を使用し確認する．PCのNNシステムとDSPのNNシステムの性能比較実験を行う

4.2.2　実験条件

学習データ増加に伴う認識率確認実験の際，DSPボードのNNシステムの設定条件は表4.3に示す．また，学習データの作成について，10サンプルの学習データは，5サンプルの学習データに新たに5サンプルを追加したものである．

表 4.3　DSPとPCを用いた性能確認実験の実験条件

電極数	4
電極種類	乾式電極
学習アルゴリズム	改良型BP法　式（2.17）
NN構造	50×25×6
学習定数η	0.05
慣性定数α	0.95
振動定数β	−0.1
1パターンあたりの評価サンプル数	20
最大学習回数	20000
最終誤差判定値	1.0E−4

それぞれの実験において，式（4.1）に示すとおり目的データをNNに提示した際，目的対象パターンに出力ユニットが最大発火したパターンを正解とする．

4.2.3　実験結果

DSP ボードを用いた実験を表 4.4 に示す.

表 4.4　DSP ボードを用いた実験結果

認識率（%）	学習サンプル数	
	5 サンプル	10 サンプル
背屈	35.0%	65.0%
掌屈	80.0%	70.0%
回内	40.0%	55.0%
回外	100.0%	95.0%
開き	100.0%	100.0%
握り	60.0%	90.0%
平均	69.2%	79.2%

また，PC を用いた実験結果を表 4.5 に示す.

表 4.5　PC を用いた実験結果

認識率（%）	学習サンプル数	
	5 サンプル	10 サンプル
背屈	20.0%	70.0%
掌屈	75.0%	60.0%
回内	65.0%	50.0%
回外	100.0%	100.0%
開き	100.0%	100.0%
握り	50.0%	75.0%
平均	68.3%	75.8%

登録者を限定した場合，登録データを 50 サンプルまで随時増加させると同一使用者の未学習データに対する汎化能力は徐々に向上することが図 4.2 より確認できる（55 サンプル以上は理想波形）．したがって，現在のアンプ特性ならびに，データ前処理法であっても，登録者だけの手首挙動を認識するのであれば，登録データを増加させることにより実用域（90%程度）に本システムの性能を展開することが可能であると考えられる．

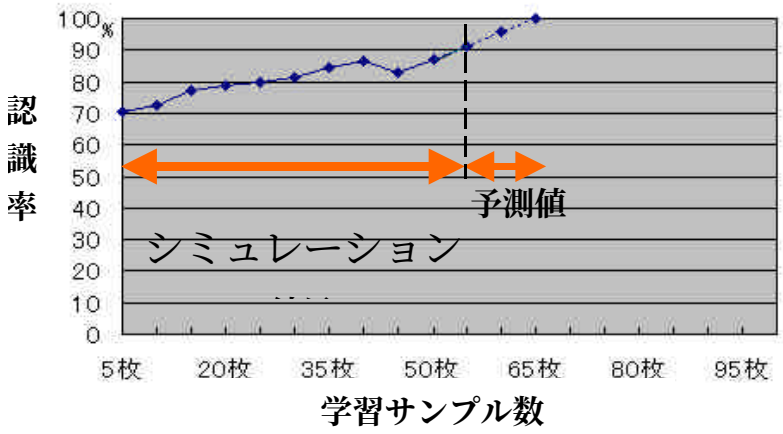


図 4.2 学習サンプル数と認識率との関係

4.2.4 考察

表 4.4 と表 4.5 より認識率はほぼ同等の結果が得られた．この際に，認識率が PC と DSP ボードの NN システム間で異なった要因として，初期 NN ウエイトを作成する際に用いる乱数が PC の NN システムと DSP ボードの NN システム双方の間で異なっていたためであると考えられる．学習データ増加に伴う性能確認実験では，5 サンプルでの学習と 10 サンプルでの学習の際を比較した，10 サンプルでの学習を行った場合の認識率が飛躍的に上昇したことが確認できた．また，掌屈，回内動作における認識率低下が見られるが，これは，新たに追加した 5 サンプルの中に，ノイズ混入などの認識率に悪影響を及ぼすサンプルが入っていたためと推測される．

この実験により，PCのNNシステムとDSPのNNシステムの性能は同等であると判明した．これにより，PCとDSPボードのNNシステムの互換性を確認することができた．また，学習データを追加していくと認識率が上昇することが確認できた．

今回の実験では初期学習の際には学習時間に多くの時間を消費することが確認できた．継続学習時には初期学習で費やしたほどの学習時間は消費ないものと推測される．

第5章　まとめ

本稿では，種々の認識対象について，オンラインチューニングの基礎実験を行った．実験を行うことにより学習データの追加および継続学習による認識率の向上を確認した．これによりタイ紙幣識別および EMG パターン識別についてオンラインチューニングの基幹要素の有用性を確認することができた．

また，PC の NN システムと DSP の NN システムの性能比較実験を行った結果，両者がほぼ同等の認識性能であることを確認することができた．これにより PC と DSP ボードの互換性を示した．この結果より NN ウエイトを両システムにおいて共有できるものと推察される．これらの実験結果より，オンラインチューニングを用いたポータブル DSP ボードの開発が可能であることが予想される．オンラインチューニングを開発する際に残された問題としては，システムとしての信頼性，DSP ボードの処理速度などの問題がある．今後は，これらの問題の解決を図るとともに，DSP ボードを用いてさらに使用対象システムに特化した認識が可能なオンラインチューニングシステムの開発，リアルタイムオンラインチューニング学習システムの開発，またポータブルシステムの開発を行う予定である．

謝辞

本研究の遂行ならびに本論文の執筆にあたり的確なるご指導とご校閲を承りました高知工科大学情報システム工学科竹田史章教授に深甚なる敬意を表します．また，本研究の進行する上でご指導頂きましたグローリー工業（株）西蔭紀洋氏心から厚く御礼申し上げます．また，本研究の実験をお手伝い頂いた中原昌樹氏，Miss.Sakoobunthu　Lalita，津末慎太郎氏，心より感謝申し上げます．

参考文献

(1) 瀬谷啓介, “ DSP　C プログラミング入門 ”, 技術評論社, pp.5-40, 1998

(2) 三上直樹, “ デジタル信号処理プログラミング入門 ”, CQ 出版社, pp.10-26, 1993

(3) 麻生秀樹, “ ニューラルネットワーク情報処理 ”, 産業図書, pp.13-46, 1994

(4) 萩原将文, “ ニューロ・ファジィ・遺伝的アルゴリズム ”, 産業図書, pp.21-39, 2000

(5) S.Nagata, M.Sekiguchi and K.Asakawa, “ Mobil Robot Control by Structured Hierarchical Neural Network ”, IEEE Control System Magazine, April, pp.69-76, 1990

(6) 竹田史章, 大松繁, 井上卓, 尾波宰三, ”フーリエ変換を前処理とするニューラルネットワークによる紙幣識別”, システム制御情報学会論文誌, Vol.5, No.7, pp.265-273, 1992

(7) “TMS320　DSP ファミリディジタル・シグナル・プロセッサ　プロダクトセミナーシート” 日本テキサスインスツルメント株式会社,

(8) “TMS320C3x/C4x ディジタル・シグナル・プロセッサ　ユーザーズ・マニュアル”, 日本テキサスインスツルメンツ株式会社, pp.12-24, 1997

(9) “TMS320C3x/C4x オプティマイジング (最適化) C コンパイラ　ユーザーズ・マニュアル”, 日本テキサスインスツルメンツ株式会社, pp.2-6, 1998

(10)　“TMS320C3x/C4x アセンブリ言語開発ツール”, 日本テキサスインスツルメンツ株式会社, pp.180-230, 1998

(11)　佐藤公信, 竹田史章, 中原昌樹, 津末慎太郎, ”DSP ボードによるニューロオンライン学佐藤習と筋活動電位によるパターン認識への応用”, 日本シミュレーション＆ゲーミング学会秋季全国大会発表論文集, pp.92-97, 2002

(12)　佐藤公信, 竹田史章, ”プログラム可変型 DSP ボードの開発とそのニューラルネットワークへの応用 ” 計測自動制御学会 SI 部門学術講演会, pp489-pp490, 2001

(13)　中原昌樹, 竹田史章, “筋活動電位によるモバイル端末用インターフェースの開発

“，FAN‘02，pp.245-250，2002

(14)　竹田史章，大松繁，“ニューロ紙幣識別ボードの開発”，電気学会論文誌 C，116，No.3，pp336-340 1996

(15)　Lalita Sakoobunthu，竹田史章，佐藤公信，“Online Continuous Learning by DSP Unit”(社) 計測自動制御学会システムインテグレーション部門講演会講演論文集 (III)，pp.321-322

(16)　竹田史章，大松繁，井上卓，尾波宰三，小西健一，“ニューラルネットワークを用いた高速搬送紙幣の識別”，電気学会論文誌，Vol.112-C，No.4，pp.249-258，1992

(17)竹田史章，大松繁，寺田兼吾，“遺伝的アルゴリズムと DSP による紙幣識別用ニューロシステム技術とその応用”，機械電気誌(C)，63，No.615，pp.3933-3940，1997

(18)竹田史章，西蔭紀洋，藤田靖，”自己学習型ニューロ紙幣識別ボードの開発とその汎用展開”，電気学会論文誌 C，Vol.121，No.2，pp.357-365，2001/2

(19)竹田史章，中原昌樹，中浦一浩，山本祥弘，“EMG による手首挙動パターン識別システムの開発とオンラインチューニングの検討”，電気学会論文誌投稿中

(20)中原昌樹，竹田史章，”ニューラルネットワークを用いた筋活動電位のパターン認識システムの構築”，高速信号処理応用技術学会春季研究会講演論文集，pp.46-51，2001

(21)中原昌樹，竹田史章，”ニューラルネットワークを用いた筋活動電位のパターン認識システムの開発”，システム制御情報学会講演論文集，pp.109-110,2001

(22)竹田史章，西蔭紀洋，内田久也，中原昌樹，“ニューロテンプレートによるパターンマッチング識別手法の開発とその紙幣への適用”，システム制御情報学会講演論文集，pp.415-416，2000

付録

NN ウエイトデータのアップデートを行うプログラムを示す.

```
/*****************************************************
名称:       upload_data
機能:       フラッシュメモリのメモリアップロード
引数:       sw  データの種類
引数:       blk  未使用ブロックの先頭
戻値:       なし
@*****************************************************/
void upload_data( int sw,int blk )
{
        FILE        *fwr;
        UCHAR       *bip,*rp;
        int         i,j; /* loop counter */
        int         block,block_st,status[2];  /*  探索対象,  探索開始ブロック,  ブロックステータス(0:データの先頭ブロック  1:データが使用するブロック数)*/
//      int         total,cap;
        long        total,cap;
        int         nblk;
        char        nn_name[33],bv_serial[9],ymd[9];
        char        fname[14],buf[4],swp[5];
        char        *msg;
        int         tabs[6] = {17,2,17,25,8,8}; /*  表示時の位置合わせ変数  */
        int         lflag;                                          /*  local flag */
        short       start[64];
        char        *skind[3];

        skind[0] = "GREEN";
        skind[1] = "    RED";
        skind[2] = "    RED";

        cls();
        block_st = (blk/SECTOR_SIZE)*SECTOR_SIZE; /*  該当セクタの先頭ブロック  */
        lflag=1;
        nblk = 0;
        while(lflag){
/*          bip = work+4096; */
            bip = work+6144;

            /* 021008mod  以下 3 行のコメントアウト  */
//          if (sw ==0x32)          /*weight*/
//                  block =block_st + 384;
//          else
            /* 021022mod block はページ先頭位置を 0 としているので*/
            /*  セクタの先頭は 0 あるいは SECTOR_SIZE となる  */
            /*  また,  bolock_st は SECTOR_SIZE の倍数となる  */
//          block=block_st;
            block=block_st%(SECTOR_SIZE*2);

            bip += (block)*2; /*  該当セクタの先頭へポインタ移動  */
            status[0] = *(bip);
            status[1] = *(bip+1);
            cls();

            switch(sw){
              case 0x31:
                msg = "Sensor Correction";
                printf("¥x1b[1;%dH      *** Sensor Correction Data in FLASH MEMORY ***",tabs[0]);
                printf("¥x1b[2;%dH      BLOCK No.   Kind   No.  Serial No.   Making Date",tabs[0]);
                printf("¥x1b[3;%dH/*          ２波長センサ対応  */
/*                      1998.01.29                niskage
/*****************************************/
----------+------+-----+-----------+-------------",tabs[0]);
                break;
              case 0x32:
                msg = "Weight";
                printf("¥x1b[1;%dH                                  *** Weight Data in FLASH MEMORY ***",tabs[1]);
                printf("¥x1b[2;%dH      BLOCK No.  NN's ID   Mask ID  num. of ptrns  NN's Name",tabs[1]);
                printf("¥x1b[3;%dH
----------+--------+-------+-------------+--------------------------------",tabs[1]);
                break;
              case 0x33:
                msg = "Trap";
                printf("¥x1b[1;%dH                      ***  Trap  Data  in  FLASH  MEMORY ***",tabs[2]);
                printf("¥x1b[2;%dH      BLOCK No.  NN's ID   input_cell  output_cell  kind",tabs[2]);
                printf("¥x1b[3;%dH      ----------+--------+----------+----------+-----",tabs[2]);
                break;
              case 0x34:
```

```
                    msg = "Slab-mask";/*          ２波長センサ対応 */
/*                          1998.01.29              niskage
/*****************************************/
                    printf("¥x1b[1;%dH          *** Slab-mask Data in FLASH MEMORY ***",tabs[3]-4);
                    printf("¥x1b[2;%dH        BLOCK No. Mask ID   mask ptrn    kind",tabs[3]);
                    printf("¥x1b[3;%dH        ----------+-------+-----------+------",tabs[3]);
                    break;
                case 0x35:
                    msg = "Truth Trap";
                    printf("¥x1b[1;%dH                          *** Truth Trap Data in FLASH MEMORY ***",tabs[4]);
                    printf("¥x1b[2;%dH        BLOCK No.   NN's ID   Num of ptrn    Kind of trap     Num of BLOCK",tabs[4]);
                    printf("¥x1b[3;%dH
----------+--------+------------+--------------+---------------",tabs[4]);
                    break;
                case 0x36:
                    msg = "Size Trap";
                    printf("¥x1b[1;%dH                           *** Size Trap Data in FLASH MEMORY ***",tabs[5]);
                    printf("¥x1b[2;%dH        BLOCK No.   NN's ID   Num of ptrn    Kind of trap     Num of BLOCK",tabs[5]);
                    printf("¥x1b[3;%dH
----------+--------+------------+--------------+---------------",tabs[5]);
                    break;
                default:
                    return;
            }
//          while((block+status[1]<=blk)&&(block                                    != ((blk+SECTOR_SIZE)/SECTOR_SIZE)*SECTOR_SIZE)){
//          while((block+status[1]<=blk)&&(block != blk)){
          // 021022mod
            // block_st:該当セクタの先頭位置
            // block :  該当ページの先頭が 0 ->   block%SECTOR_SIZE :  該当セクタの先頭が 0
            while((block_st+(block%SECTOR_SIZE)+status[1]                                <= blk)&&( block_st+(block%SECTOR_SIZE) != blk)){
                if(sw+0x40 == status[0]){
                    rp = work;
//                  if(0 != cmd_read_blk( block, FLASH_BLOCK_DATA, rp )){
                // 021022mod/*          ２波長センサ対応 */
/*                          1998.01.29              niskage
/*****************************************/
                    if(0        !=        cmd_read_blk(        block_st+(block%SECTOR_SIZE), FLASH_BLOCK_DATA, rp )){
                        printf("¥n*** flash memory data read error!! ***");
                        printf("¥npush any key to return to previous menu ");
                        getch();
                        return;
                    }
                    nblk++;
                    start[nblk]=block;
                    printf("¥n¥x1b[%dC",tabs[(status[0]&0x0f)-1] );
//                  printf("%2d : %3d - %3d ",nblk,block,block+status[1]-1 );
                // 021022mod
                    printf("%2d    :    %3d    -    %3d    ",nblk,block_st+block%SECTOR_SIZE, block_st+block%SECTOR_SIZE+status[1]-1 );
                    switch(status[0]){
                      case 0x71:/*          ２波長センサ対応 */
/*                          1998.01.29              niskage
/*****************************************/
                        for ( i=0; i<8; i++ ){
                          bv_serial[i]              = *(rp+16+i);
            /* シリアル番号                                                 */
                          ymd[i]                    = *(rp+24+i);
            /* 日付                                                         */
                        }

            printf("%7s  %4d       %.8s         %.8s",skind[c2toui(rp+4)/256],(c2toui(rp+4))%256,bv_serial,ymd);
                        break;
                    case 0x72:
                        for ( i=0; i<32; i++ ){
                          nn_name[i]                = *(rp+20+i);
                        }
                        rp = work+12;
                        printf(" %02x%02x%02x%02x",*rp,*(rp+1),*(rp+2),*(rp+3));
                        printf("%7ld%11ld      %-32s",c4toul(rp+156),c4toul(rp+116),nn_name);
                        break;
                    case 0x73:
                        rp = work + 12;
                        printf(" %02x%02x%02x%02x",*rp,*(rp+1),*(rp+2),*(rp+3));
                        printf("%10ld%11ld",c4toul(rp+40),c4toul(rp+44));
                        printf("%7s",skind[*(rp+3)]);
                        break;
                    case 0x74:
                        printf("%7d%11d",*(rp+4),c2toui(rp+20));
                        printf("%7s",skind[*(rp+4)/127]);
                        break;/*          ２波長センサ対応 */
/*                          1998.01.29              niskage
/*****************************************/
                    case 0x75:
                        rp = work+12;
                        printf(" %02x%02x%02x%02x ",*rp,*(rp+1),*(rp+2),*(rp+3));
                        printf("%11ld%14ld%14ld",c4toul(rp+40),c4toul(rp+44),c4toul(rp+48));
                        break;
                    case 0x76:
                        rp = work+12;
                        printf(" %02x%02x%02x%02x ",*rp,*(rp+1),*(rp+2),*(rp+3));
```

```
                        printf("%11ld%14ld%14ld",c4toul(rp+40),c4toul(/*          2波長センサ
対応 */
/*                      1998.01.29                  niskage
/*****************************************/rp+44),c4toul(rp+48));
                        break;
                    default:
                        printf("PUSH ANY KEY [data format error]");
                        getch();
                        return;
                    }
                }
                block += (int)status[1];
//              status[0] = *(bip+(block-block_st)*2    );
//              status[1] = *(bip+(block-block_st)*2+1);
                // 0201105mod
                status[0] = *(bip+(block%SECTOR_SIZE)*2    );
                status[1] = *(bip+(block%SECTOR_SIZE)*2+1);
//              printf("block=%d status=%2x/%d",block,status[0],status[1]);getch();
            }
            if(nblk!=0){
                printf("¥n¥n¥t¥t%d pattern %s data in flash memory¥n¥n",nblk,msg);
                printf("¥t¥t¥tSPACE key : UPLOAD %s DATA¥n",msg);

            printf("¥t¥t¥tother key : return to previous menu¥n");

                i = getch();
                if(i != 0x20){
                    lflag=0;
                    return;
                }
            }
            else{
                printf("¥n¥n¥t¥tNO %s data in flash memory¥n¥n",msg);
                printf("¥t¥t¥t PUSH ANY KEY to return to previous menu¥n¥n");
                getch();
                lflag=0;
                return;
            }
            /* アップロードするブロックの番号を入力 */
            lflag=0;
            while(!lflag){
                printf("¥t¥t¥tINPUT No. to UPLOAD [ 1 - %d ]> ",nblk);
                scanf("%s",buf);
                i = atoi(buf);
                if((i>=1)&&(i<=nblk))
                    lflag = 1;
            }
            cls();
//          bip = work+4096;
            bip = work+6144;
//          status[0] = *(bip);
//          status[1] = *(bip+1);
//          block = 0;
            block_st = (blk/SECTOR_SIZE)*SECTOR_SIZE; /* 該当セクタの先頭ブロック */
/*
            while( (block < start[i]) && (block < cfg.fl_block) ){
                block += (int)status[1];
                status[0] = *(bip+block*2    );
                status[1] = *(bip+block*2+1);
            }
*/
            status[0] = *(bip+start[i]*2);
            status[1] = *(bip+start[i]*2+1);
//          block=start[i];
            // 021022mod
            block= start[i]%SECTOR_SIZE + block_st;

            /* データ先頭ブロックのリードコマンド発行*/
            rp = work;
            /* ヘッダ情報取得の為，"block"ブロックのデータを */
            /* FLASH_BLOCK_DATA バイト(１ブロック分)読み込み */
            /* rp に格納する */
            if(0 != cmd_read_blk( block, FLASH_BLOCK_DATA, rp )){
                printf("¥n*** flash memory data read error!! ***");
                printf("¥npush any key to return to previous menu ");
                getch();
                return;
            }
            /* データの種類（correction,weight,trap,mask,truth）を調べ，コール */
            switch (status[0]){
                case 0x71 :         /* センサ補正データ                    */
                  /* ヘッダ情報の取得，表示（形式は無視）*/
                    for ( i=0; i<8; i++ ){
                        bv_serial[i]  = *(rp+16+i);                         /* シリ
アル番号                                */
                        ymd[i]                  = *(rp+24+i);
/* 日付                                                                    */
                    }
                    total  = sizeof(struct sns_crct_header) +c2toui(rp+6) * 2 * 2;          /* ファ
イルのサイズ                            */
                    printf("¥nSENSOR CORRECTION HEADER-DATA [%d]¥n",block);
                    printf("¥n      sensor          = %d (%s)"
,c2toui(rp+4)%256,skind[c2toui(rp+4)/256]);
                        printf("¥n      pixel/line      = %d"
,c2toui(rp+6));
                    printf("¥n      max value       = %04xhex",c2toui(rp+8));
                    printf("¥n      max channel     = %d"               ,c2toui(rp+10));
                    printf("¥n      serial No.      = %.8s"     ,bv_serial    );
                    printf("¥n      date            = %.8s"     ,ymd                );
                    break;
```

```
                case 0x72 :             /*  ウエイトデータ                          */
                  for ( i=0; i<32; i++ )
                        nn_name[i]  = *(rp+20+i);
              /*  ＮＮの名前                                          */
                  total = c4toul(rp+4);                             /*  ファイルのサイズ
              */
                  printf("¥nWEIGHT HEADER-DATA¥n");
                  printf("¥n        N.N. name                 = %-32s",nn_name);
                  printf("¥n        N.N. ID                   = %02x%02x%02x%02x"
                              ,*(rp+12),*(rp+13),*(rp+14),*(rp+15));
                  printf("¥n        Number of Patterns      = %ld",c4toul(rp+128));
                  printf("¥n        Mask ID                   = %ld",c4toul(rp+168));
                  printf("¥n¥n    This N.N. has %2ld layers",c4toul(rp+16));
                  printf("¥n        layer 1    has    %3ld cells",c4toul(rp+188));
                  printf("¥n        layer/*           ２波長センサ対応  */
/*                        1998.01.29                  niskage
/*******************************************/ 2    has    %3ld cells",c4toul(rp+192));
                  printf("¥n        layer 3    has    %3ld cells",c4toul(rp+196));
                  printf("¥n        eps      %f",c4tof(rp+148)     );
                  printf("¥n        alpha %f",c4tof(rp+152)      );
                  printf("¥n        beta    %f",c4tof(rp+156)     );
                  break;
                case 0x73 :             /*  濃淡トラップデータ */
                  total  = c4toul(rp+4);
              /*  ファイルのサイズ                    */
                  printf("¥nTRAP HEADER-DATA¥n");
                  printf("¥n        N.N. ID                   = %02x%02x%02x%02x"
                              ,*(rp+12),*(rp+13),*(rp+14),*(rp+15));
                  printf(" (%s)",skind[*(rp+15)%2]);
                  printf("¥n        num. /*           ２波長センサ対応  */
/*                        1998.01.29                  niskage
/*******************************************/of input             = %ld",c4toul(rp+52));
                  printf("¥n        num. of output            = %ld",c4toul(rp+56));
                  printf("¥n        threshold1                = %6.4f",c4tof(rp+60));
                  printf("¥n        thres/*           ２波長センサ対応  */
/*                        1998.01.29                  niskage
/*******************************************/hold2                 = %6.4f",c4tof(rp+64));
                  printf("¥n        limits of average       = %6.4f",c4tof(rp+68));
                  printf("¥n        limits of deflection = %6.4f",c4tof(rp+72));
                  break;
                case 0x74 :             /*  スラブマスクデータ */
                  total  = (c4toul(rp+24) +    c2toui(rp+16) * c2toui(rp+18) * c2toui(rp+20));
                  printf("¥nSLAB MASK HEADER-DATA¥n");
                  printf("¥n                mask    ID                                        =    %3d
(%s)",*(rp+4),skind[*(rp+4)/127]);
                  printf("¥n        offset X from center = %3d",c2toui(rp+8));
                  printf("¥n        offset Y from center = %3d",c2toui(rp+10));
                  printf("¥n        X size of unit            = %3d",c2toui(rp+12));
                  printf("¥n        Y size of unit            = %3d",c2toui(rp+14));
                  printf("¥n        units / line              = %3d",c2toui(rp+16));
                  printf("¥n        units / column            = %3d",c2toui(rp+18));
                  printf("¥n        patterns                  = %3d",c2toui(rp+20));
                  printf("¥n        th. for slant correct= %3d/255",*(rp+28)  );
                  break;
                case 0x75 :             /*  真偽トラップデータ */
                  total  = c4toul(rp+4);
              /*  ファイルのサイズ                    */
                  printf("¥nTRUTH TRAP HEADER-DATA¥n");
                  printf("¥n        N.N. ID                   = %02x%02x%02x%02x"
                              ,*(rp+12),*(rp+13),*(rp+14),*(rp+15));
                  printf("¥n        number of patterns      = %3ld",c4toul(rp+52));
                  printf("¥n        number of sensors       = %3ld",c4toul(rp+56));
                  printf("¥n        number of blocks        = %3ld",c4toul(rp+60));
                  printf("¥n        method No.                = %3ld",c4toul(rp+68));
                  printf("¥n                  val1            = %f",c4tof(rp+72));
                  printf("¥n                  val2            = %f",c4tof(rp+76));
                  break;
                case 0x76 :             /* サイズトラップデータ*/
                  total  = c4toul(rp+4);
              /*  ファイルのサイズ                    */
                  printf("¥nSIZE TRAP HEADER-DATA¥n");
                  printf("¥n        N.N. ID                   = %02x%02x%02x%02x"
                              ,*(rp+12),*(rp+13),*(rp+14),*(rp+15));
                  printf("¥n        number of patterns      = %3ld",c4toul(rp+52));
                  printf("¥n        kind      of size        = %3ld",c4toul(rp+56));
                  printf("¥n        number of blocks        = %3ld",c4toul(rp+60));
                  printf("¥n        Width    method No.     = %3ld",c4toul(rp+68));
                  printf("¥n        Height method No.       = %3ld",c4toul(rp+72));
                  //printf("¥n                  val1            = %f",c4tof(rp+68));
                  //printf("¥n                  val2            = %f",c4tof(rp+72));
                  break;
                default :               /*  仕様外  */
                  printf("¥nThis block[%3d] is not 'data start block' !![%2x]",block,status[0]);
                  printf("¥npush any key to return ");
                  getch();
                  return;
              }
              printf("¥n¥nThis data uses %d-%d blocks",block,block+status[1]-1          );
              printf("¥n¥n    make file size is %ldbyte",total                          );
              /*  書き込み用ファイルのファイル名の取得  */
              printf("¥n¥ninput save filename --> ");
              scanf("%s",fname);
              printf("¥n SPACE      key : UPLOAD %s DATA to file [%s]",msg,fname);
              printf("¥n other key : previous menu");
              i = getch();
              if ( i != 0x20 )
                return;
              /*  データ書き込み用ファイルのオープン  */
              if (( fwr = fopen( fname, "wb")) == NULL ){
                printf("¥n*** file open error!! [%s] ***",fname);
```

```
    printf("¥npush any key to return to previous menu ");
    getch();
    return;
  }
  /*  フラッシュから読み込み＆ファイルへの書き込み        */
  /* i: 処理が終了したブロック，page: 処理が必要なブロック block: 対象ブロック */
    for ( i=0; i<status[1]; i++,block++,total-=cap ){
      /* 必要バイト数を決定 １ブロック処理する度に total は cap だけ減っていく */
       cap = (FLASH_BLOCK_DATA<total) ? FLASH_BLOCK_DATA : total;
       printf("¥nread      %d block (%3d/%3d)",block,i+1,status[1]);
       rp = work;
       /*  フラッシュメモリからデータを読み出す  */
       if(0 != cmd_read_blk( block, cap, rp )){
            printf("¥n*** flash memory data read error!! ***");
            printf("¥npush any key to return to previous menu ");
            fclose(fwr);
            getch();
            return;
       }
       /*  データの並びを変換  */
                    rp = work;
                    switch(status[0]){
                    case 0x71:
                              for(j=0;j<cap;j+=2,rp +=2){

                                        if          (          !((i==0)          &&
((j==0)||(j==2)||((16<=j)&&(j<32)))) ){
                                                  swp[0]      =      *rp;
                         swp[1]    = *(rp+1);
                                                  *rp          =   swp[1];
               *(rp+1) = swp[0];
                                                  }
                                        }
                              break;
                    case 0x72:
                              for ( j=0; j<cap; j+=4, rp+=4 ){
                                        if          (          !((i==0)          &&
((j==0)||(j==12)||((20<=j)&&(j<116)))) ){
                                                  swp[0]      =      *rp;
                         swp[1] = *(rp+1);
                                                  swp[2]    =    *(rp+2);
               swp[3] = *(rp+3);
                                                            *rp
= swp[3];     *(rp+1) = swp[2];
                                                            *(rp+2)
= swp[1];     *(rp+3) = swp[0];
                                                  }
                              }
                              break;
                    case 0x73:
                              for ( j=0; j <cap ; j+=4, rp+=4 ){
                                        if          (          !((i==0)          &&
((j==0)||(j==12)||((16<=j)&&(j<48)))) ){
                                                  swp[0]      =      *rp;
                                        swp[1] = *(rp+1);
                                                                    swp[2]
= *(rp+2);    swp[3] = *(rp+3);
                                                                    *rp
= swp[3];     *(rp+1) = swp[2];
                                                                    *(rp+2)
= swp[1];     *(rp+3) = swp[0];
                                                  }
                                        }
                              break;
                    case 0x74:
                              for ( j=0; j<cap ; j+=2, rp+=2 ){
                                        if          (          (i==0)          &&
((8<=j)&&(j<22)) ){
                                                  swp[0]      =      *rp;
                                        swp[1]    = *(rp+1);
                                                  *rp          =   swp[1];
                    *(rp+1) = swp[0];
                                                  }
                                        if    (    (i==0)    &&
(j==24) ){
                                                  swp[0]      =      *rp;
                                        swp[1] = *(rp+1);
                                                        swp[2]          =
*(rp+2);      swp[3] = *(rp+3);
                                                  *rp          =   swp[3];
                                        *(rp+1) = swp[2];
                                                  *(rp+2)    =    swp[1];
                                        *(rp+3) = swp[0];
                                        }
                              }
                              break;
                    case 0x75:
                              for ( j=0; j<cap; j+=4, rp+=4 ){
                                        if          (          !((i==0)          &&
((j==0)||(j==12)||((16<=j)&&(j<48)))) ){
                                                  swp[0]      =      *rp;
                                        swp[1] = *(rp+1);
                                                                    swp[2]
= *(rp+2);    swp[3] = *(rp+3);
                                                                    *rp
= swp[3];     *(rp+1) = swp[2];
                                                  *(rp+2)    =    swp[1];
                                        *(rp+3) = swp[0];
                                        }
                              }
```

```
                                        break;
                                case 0x76:
                                        for ( j=0; j<cap; j+=4, rp+=4 ){
                                                if      (       !((i==0)        &&
((j==0)||(j==12)||((16<=j)&&(j<48))) ){
                                                                        swp[0]      =       *rp;
                        swp[1] = *(rp+1);
                                                                                swp[2]
= *(rp+2);      swp[3] = *(rp+3);
                                                                                *rp
= swp[3];       *(rp+1) = swp[2];
                                                                        *(rp+2)     =    swp[1];
                *(rp+3) = swp[0];
                                                }
                                        }
                                        break;
                                default    :    return;
                        }
                        /*  ファイルにデータを書き込む  */
                              rp = work;
                        if( fwrite( rp,1,(size_t)cap,fwr) < (size_t)cap ){
                                printf("¥n*** file write error!! [%s] ***",fname);
                                printf("¥npush any key to return to previous menu ");
                                      fclose(fwr);
                                getch();
                                return;
                        }
                }
                      fclose(fwr);
                printf("¥n¥ncomplete!!");
                      printf("¥n%s data was saved to '%s' ",msg,fname);
                              printf("¥nSPACE key : continue UPLOAD %s data",msg);
                printf("¥nother key : return to previous menu");
                              i=getch();
                      if(i != 0x20 )
                            return;
        }
}
```