〔非公開〕

ＴＲ－Ｃ－０１３９

周波数領域変換を用いた実時間表情検出

海老原　一之　Kazuyuki EBIHARA
大谷　淳　Jun OHYA
岸野　文郎　Fumio KISHINO

１９９５　３．１５

ＡＴＲ通信システム研究所

目次

1．まえがき

異なる場所にいる人々があたかも一堂に会しているかのような感覚を持ちつつ，会議や協調作業を行える環境を提供することを目的とした通信を筆者らは臨場感通信会議とよび，要素技術の検討を進めている [1] [2]．従来のテレビ会議システムでは，人間同士の自然なコミュニケーションのために重要な視線一致や運動視差の再現が困難であったが，臨場感通信会議ではこれらの問題を解決するために，3次元コンピュータグラフィックス技術を用いて生成された仮想的な共有空間に，会議参加者の3次元人物像や物体を配置し，立体ディスプレイに表示する [1]．ここで，会議参加者については，送信側で表情や動きを検出し，受信側の3次元人物モデルにおいて再現する（図1）．したがって，参加者の表情検出を実時間で実現し，3次元顔モデル（3Dimension Wire Frame Model により構築．以後，3D WFM）で忠実に再現できれば，参加者同士の円滑なコミュニケーションが可能となり，臨場感通信会議の実現に向けて大きな前進となる．仮想空間を共有して協調作業や会議を実現する臨場感通信会議におれる実時間表情検出・再現は，参加者の表情を実時間で検出し忠実に再現する必要はあるが，参加者の感情は参加者が直接判断するため，表情検出は物理的な変化のみを検出すればよい．

会議参加者の顔の実時間表情検出を行う場合，顔画像中から非接触な形式を用いて実現することが望ましく，従来から顔画像のみを用いて実時間で顔の表情検出を行う研究は数多く見られる．例えば，顔全体を均等間隔のライン分割を行いラインの輝度変化により表情を検出する研究 [3] や，顔画像のエッジ検出を行い形状変化を検出して表情変化を捉える研究 [4] がある．しかし，実時間性の問題や，処理の安定性に問題が見らた．また，3次元顔モデルにおける表情生成の研究も数多く見られるが（例えば，文献 [5]，[6]），表情検出系とリンクして実時間で表情を再現した研究としては，音声解析を用いて口の形状を実時間で再現しようとしたシステム [7]，[8] が見られる程度である．

筆者らは，表情検出のためのカメラを顔に対して相対的に常に同一の位置に保つために，小型ＣＣＤカメラを固定したヘルメットを会議参加者が被るようにし，表情検出処理の簡易化のため参加者の顔に小球状のマーカを貼付して，前述のカメラからの顔画像中で追跡するシステムを構築した（図2）．しかし，この従来システムでは，会議参加者は，顔にマーカを貼付する必要があり，マンマシンインターフェースの観点からは望ましいとは言えず，システムの実用性に課題を残していた．

本稿では，前述のヘルメットに固定したカメラを用いて，マーカを貼付していない状態の顔を観測することにより獲得される顔画像から，画像処理のみを用いて実時間表情検出を行う手法を検討する．本方式では，小型ＣＣＤカメラから得られる顔画像中において顔の位置が常に同一になるように，毎回ヘルメットを被ることは困難であるほか，顔画像は周囲の明るさの変化や映像信号のノイズ等の影響を受けるために，実時間で安定に表情検出を行うことは非常に困難であった．

そこで本稿では，ヘルメットに固定した小型ＣＣＤカメラから得られた顔画像を周波数領域に変換し，周波数成分の変化を捉えることにより，周波数成分の増減から会議参加者の表情の変化を検出する手法を提案する．ここで，周波数成分の変化は，無表情時からの相対変

化量なので，3次元顔モデルを変形することにより，表情の変化を再現する情報に変換する必要がある．本稿では，周波数成分の変化量と顔の表情筋の移動量を，GA（Genetic Algorithm）[9] を用た学習により関連付け，3次元人物モデルの変形情報として用いている．

以下，2章では，臨場感通信会議における実時間表情検出・再現について説明する．3章では，本稿で提案する実時間表情検出法の処理の詳細について述べる．4章では，実際の適用例と実験結果について述べ，5章で本稿をまとめる．

図1　臨場感通信会議システム

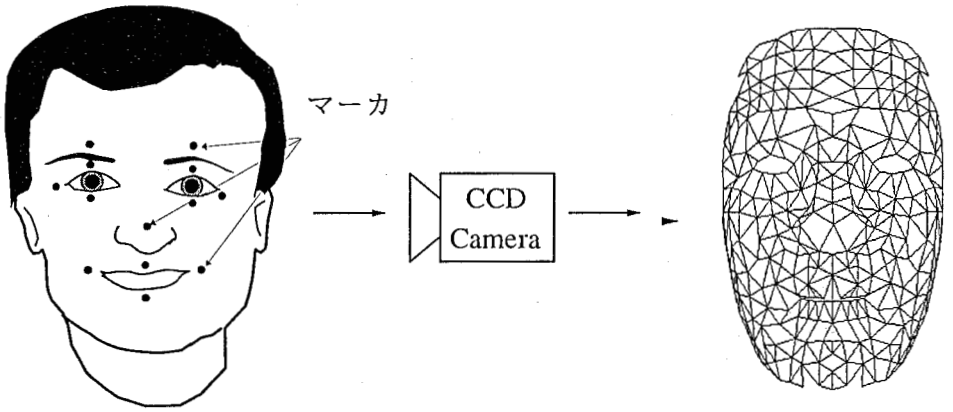


図2　現在の表情検出・再現システム

2．実時間表情検出・再現

臨場感通信会議システムにおいて人物の表情を再現するためには，1章で述べたように，各会議参加者の表情を送信側で検出し，受信側での3次元顔モデルで実時間再現を実現する必要がある．臨場感通信会議における表情再現のための処理は，図3の中の（1）～（3）で示される3つのモジュールから構成される [1] [2].

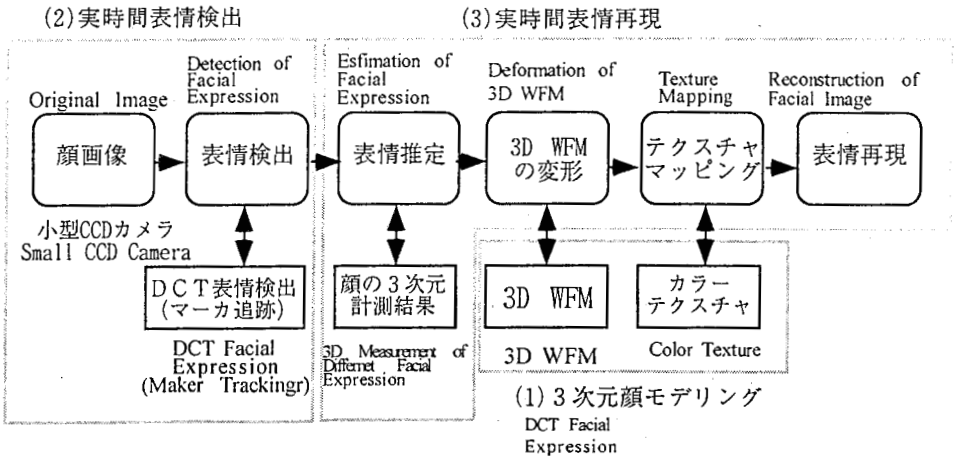


図3　顔画像処理ブロック図

(1) 顔の3次元モデリング

人物の顔の3次元モデリングは，例えば3Dスキャナ（Cyberware社製 Color 3D Digitizer）[10] を用いて行うことができる（図4）．3Dスキャナを用いると，計測対象の周囲を回転しながらLaser Stripeを照射し，各視線方向の距離情報と計測対象の色彩情報であるカラーテクスチャ情報を併せて入力することが可能である．獲得された距離情報は，顔の3次元形状を近似する三角パッチの集合，即ち3D WFMに変換される．カラーテクスチャ情報は，対応する三角パッチにマッピングされる．

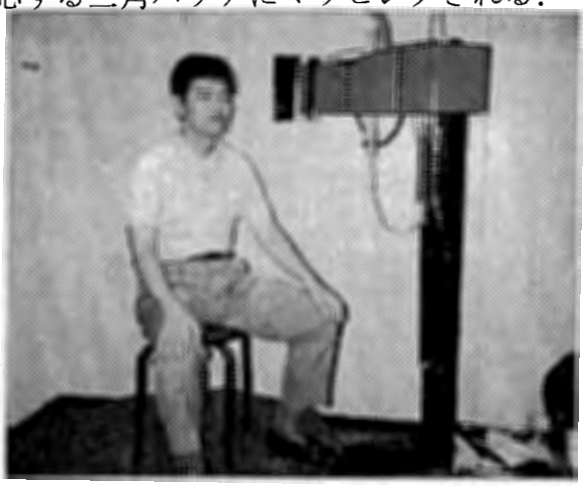


図4　3次元イメージスキャナ

(2) 実時間表情検出

1章で述べたように，表情検出部では頭部追跡処理を不要とするために，小型ＣＣＤカメラを固定したヘルメットを会議参加者が被り（図5），顔を正面から撮影した画像から表情を検出する．従来は，参加者の顔に，皮膚の色と異なる色のマーカを貼付しておき，クロマキー等により顔画像のマーカの色のみを抽出し2値化することにより，顔画像中からマーカの色に対応する画素のみを識別可能にし，各マーカの移動量を追跡することにより実時間表情再現を実現していた [2]．しかし，会議参加者は，会議毎に顔画像中の同じ位置にマーカを貼付した後，小型ＣＣＤカメラが固定されたヘルメットを正確に被る必要があり，また，異物を顔に貼付しなければならないというヒューマンインタフェース上の問題もあった．本稿では3章で述べるように，従来のように顔にマーカを貼付せず，素顔の状態で安定に実時

間表情検出が可能な手法を提案する.

図5　ヘルメットに固定したＣＣＤカメラ

(3) 実時間表情再現

表情再現部では，(2) におけるマーカの追跡結果を用いて，適宜3D WFMを変形し，これにテクスチャマッピングを施すことにより，表情を再現する．なお，後述のように，本稿で提案する表情検出法も，マーカが貼付されていた点の動きを検出するものなので，以後の説明は”マーカ”を用いて行う．筆者らは既に，３次元計測結果に基づいた実時間表情再現を提案している［11］［12］．この手法ではまず，表情再現の前処理として顔に多数のドットを描き，種々の表情表出時の顔表面形状（各ドットの無表情状態からの位置変化）の３次元計測をあらかじめ行っておく．なお，これらのドットは，表情検出用のマーカを貼付する場所の候補であり，表情再現時には，選択されたドットの位置にマーカを貼付し，顔画像中におけるマーカの追跡結果を，前述の３次元計測結果を用いて，３次元顔モデルの変形情報に変換する．即ち，前述の３次元計測においては，図６に示すような，顔に存在する主要な表情筋の動作を網羅する種々の表情の表出時において，各ドットの無表情状態における位置からの３次元移動ベクトルを求めておく．このようにして得られた各ドットの各表情に対応する３次元移動ベクトルを，表情検出系におけるカメラから入力される顔画像に基準ベクトルとして投影しておく．表情再現時にはまず，顔画像中で検出されたマーカの移動ベクトルを挟む２つの基準ベクトルを見出し，マーカの移動ベクトルをこれらのベクトル和で表現する．図７で示すように，各マーカのベクトル和の表現に基づき，3D WFMの各頂点を駆動する３次元移動ベクトルを求めることにより表情再現を行う．例えば，図７に示すように，WFMの頂点hにおいて，２Ｄベクトルを$\overrightarrow{m_i}$とすると，基本表情aとbとの中間表情の２Ｄ移動ベクトル$\overrightarrow{x_h}$は，

$$\overrightarrow{x_h} = h_h \overrightarrow{m_{ha}} + h_h \overrightarrow{m_{hb}} \qquad (1)$$

で表される．従って，$\overrightarrow{m_i}$の３Ｄスキャナによる３Ｄベクトルを$\overrightarrow{M_h}$とすると，hの３Ｄ移動

ベクトルは次式のように得られる.

$$\overrightarrow{X_h} = k_h \overrightarrow{M_{ha}} + h_h \overrightarrow{M_{hb}} \qquad (2)$$

このようにして，3D WFMを変形することにより任意の表情を高速かつ高品質な再現を実現して図6．

図6．３次元計測を行った顔画像

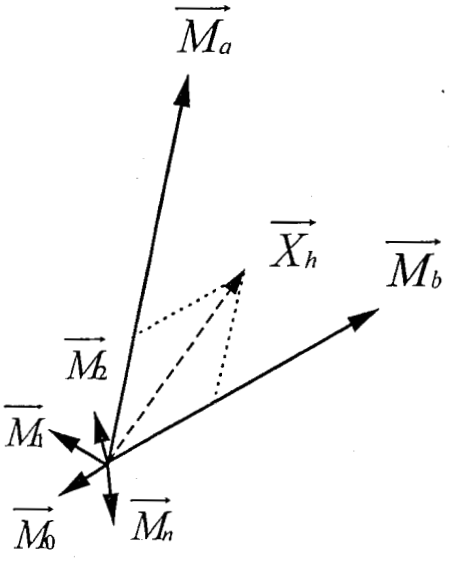


図７　３次元計測データを用いた
２次元移動ベクトルの表現例

３．周波数領域変換を用いた実時間表情検出

3.1 基本的原理

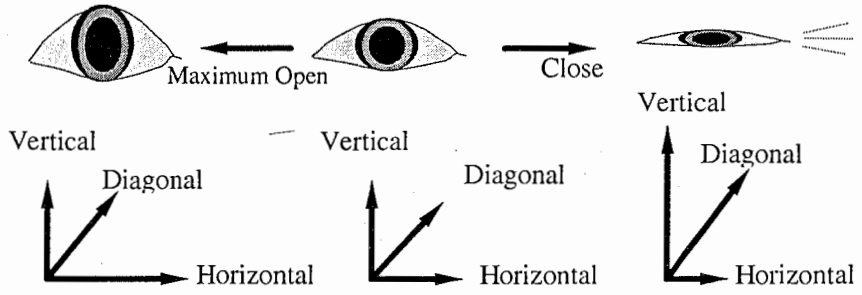
本稿で使用するヘルメットに固定した小型ＣＣＤカメラから得られる顔画像は，会議参加者の被り方により装着位置が毎回微妙に異なるうえ，会議参加者の臨場感通信会議中の協調作業等の動作により微妙にズレを生じ検出の障害となっていた．本稿では，表情検出を実時間で安定して実現出来る手法をとして周波数領域変換を利用する方法を提案する［13］［14］．従来の周波数領域変換を顔画像処理に適用した例として，フーリエ展開によりスペクトルをＫＬ展開する手法［15］や，Waveletを用いて表情認識を行う手法［16］などの検討が行われていたが，表情検出への適用は見られなかった．本手法では顔の表情変化に伴い，顔構成要素が変形することにより，周波数成分が変化することを利用して表情検出を行っている．例えば，目を細める表情を表出すると，目が細くなることにより垂直方向の変化の間隔が短くなることから垂直周波数成分が増加し，また，１本の水平線に近くなるために水平周波数成分が減少する（図８ａ）．逆に目を見開く動作により，より円形に近づくことにより，垂直方向の周波数成分が減少し，水平方向の周波数成分が増加する．額の場合には，水平の

シワが発生することにより，垂直周波数成分のみが増加する（図８ｂ）．これらの周波数成分の変化は，無表情状態からの相対変化であり，これを用いて，表情筋の動作を行う表情再現データ（顔の３次元モデルの変形情報）に変換する必要がある．そこで，ヘルメットに装着した小型ＣＣＤカメラから得られる顔画像中から図９で示すように，人間の表情変化が最も良く表れる，両目，口，額の４カ所の顔構成要素を選択し検出の領域とする．これらは，それぞれ，

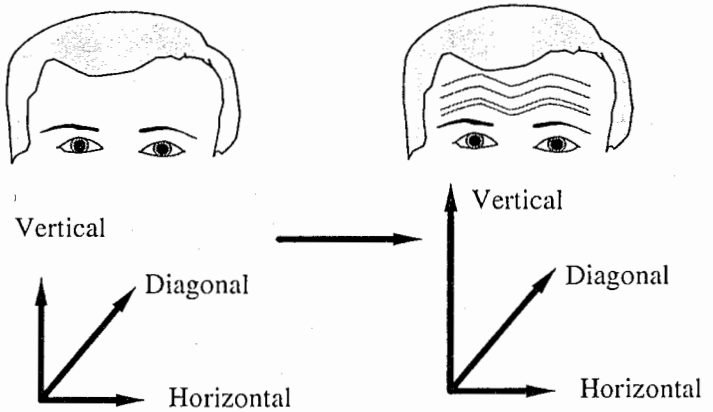
(1) 両目の開閉度
(2) 額のシワの発生度
(3) 口の開閉度

を検出し，表情筋の動き，即ち，筆者らの従来システムでマーカが貼付されていた各点の移動量に変換して筆者らの提案する表情再現手法を用いて表情再現を行う．

本稿で提案した手法は，検出領域の画像情報を周波数領域に変換しているため，領域の周波数成分の電力量と位置情報に相当する位相情報に分解される．本手法では後述のように，電力量のみを用いているため，検出すべき顔構成要素が領域内に存在すれば，内部での位置は重要ではない．従って，４つの検出領域は顔画像で固定であるが，顔構成要素の位置の個人差は，十分に吸収できるほか，ヘルメット装着時の微妙な位置のズレや，会議中の協調作業による微妙な移動は無視することが可能である．以下，本手法の具体的な内容について詳

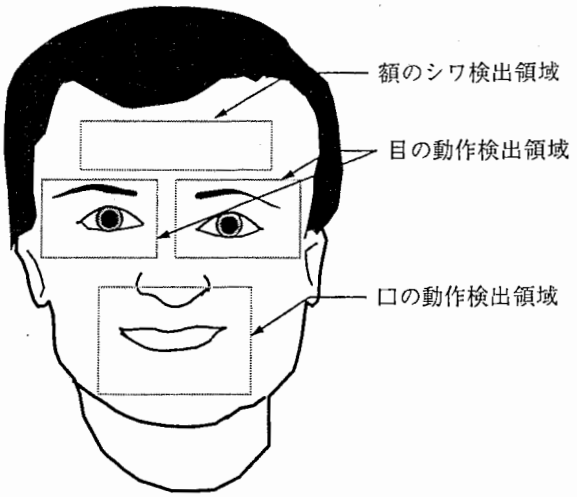


(a) 目の開閉動作の検出



(b) 額のシワの検出

図８　表情検出例



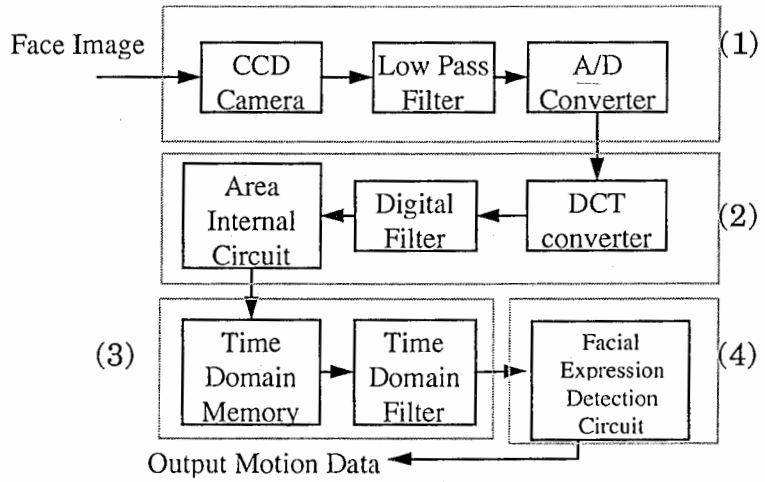
述する.

図９　表情検出エリア

3.2 アルゴリズム

図１０に本稿で提案する処理のブロック図を示す．同図に示されるように，

(1) 入力信号の帯域制限を行う前処理部
(2) 周波数領域変換部
(3) 時間軸上のフィルタリング処理部
(4) 表情検出部



から構成されている.

図１０　表情検出全体ブロック図

3.2.1 前処理

小型ＣＣＤカメラから入力される映像信号は，NTSC方式のコンポジット信号を使用しているため，約4.2MHzの映像帯域を有し，映像信号には，色情報を含む高域成分が含まれている．ところが，表情を検出しようとする目，口等の顔構成要素が低域成分に集中しているのに対し，毛穴や産毛などの顔全体を占める部分は高域成分に集中している．そのため，表情変化に伴い，顔画像中全面で高域成分の変化が発生し，低域成分の変化の検出を困難にする．そこで，表情検出は輝度信号のみを用い，高域成分によるノイズの影響をさけるため

に，約2.8MHzで帯域制限を行い，高域成分の影響を防止している．

本システムで用いる小型ＣＣＤカメラは、単板ＣＣＤを用いているため、実際の解像度は4.2MHzの帯域幅程は無い。また、映像信号をパソコンに入力するビデオキャプチャーボードは、モトローラ社の１チップY/C分離回路を用いているために精度はあまり高くなく、解像度も低い。しかも、パソコン内部では14.3MHzの発信器を通信制御や実時間カレンダー用に使用する場合が多く、これらのクロックノイズが映像信号に回り込む現象が見られた。そこで、自作のLPF（LowPassFilter）をキャプチャボードの前段に追加し、これらの悪影響を避けている。これらの問題を解決するためには、キャプチャボードは別ケースで作成し、パソコンとの間は高速ディジタルインターフェースで接続し、アナログ回路と完全に分離することが望ましいと考えられる。

3.2.2 周波数領域変換

周波数領域への変換方法は様々な手法が提案されている．本稿では，周波数成分の電力量が得られれば，位相情報は不要である点に注目し，実時間性に優れ，既に，MPEGやJPEGで採用され，実時間用処理用の集積回路の開発が終了している，２次元ＤＣＴ（2 Dimension Discreat Cosine Transform）[17] を用いてこれを実現している．検出する領域（図９）を２次元ＤＣＴを用いて周波数領域に変換する．以下に２次元ＤＣＴの式を示す．

$$F(u,v)=\frac{1}{4}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}C(u)\cdot C(v)\cdot f(x,y)\cdot \cos\frac{(2x+1)u\pi}{2\cdot M}\cdot\cos\frac{(2y+1)v\pi}{2\cdot N} \qquad (3)$$

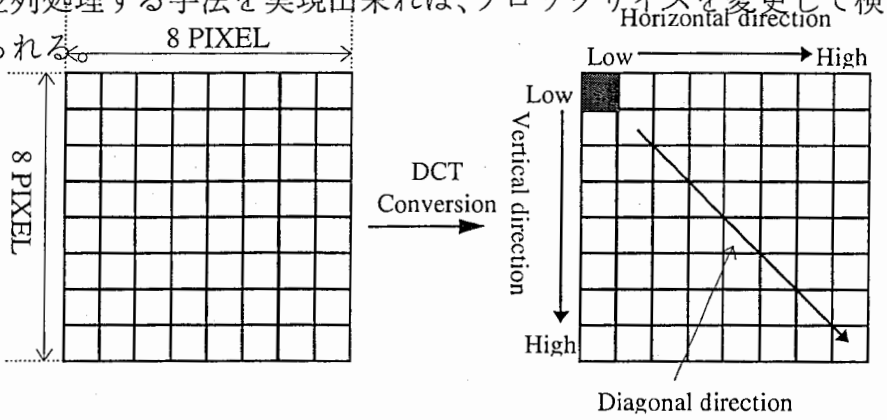
$$\text{if } u=0,\ \text{then}\ \ C(u)=1/\sqrt{2},\ \text{otherwise } C(u)=1;$$
$$\text{if } v=0,\ \text{then}\ \ C(v)=1/\sqrt{2},\ \text{otherwise } C(v)=1.$$

x，yは，実際の顔画像領域，u,vは周波数領域，$f(x,y)$の顔画像を２次元ＤＣＴを用いて周波数領域に変換した結果が$F(u,v)$である．本稿では$M=N=8$とし，ブロックサイズの大型化に伴う実時間性の低下や回路の構成規模の増大を避けるために，JPEG，MPEGで用いられる一般的なブロックサイズを用いている．

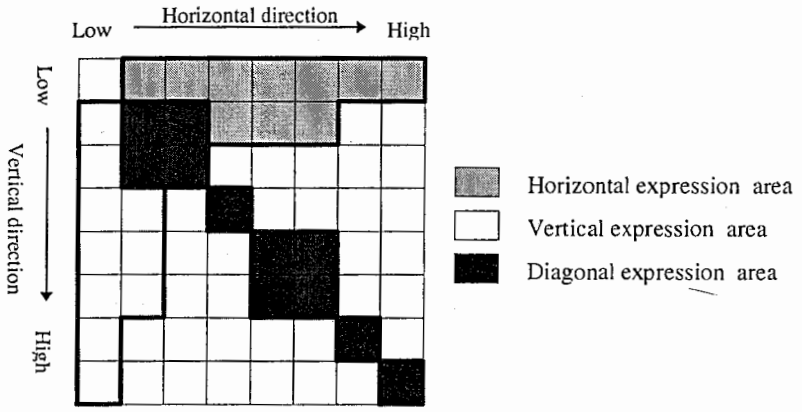
２次元ＤＣＴ後のデータは図１１ａで示すように，直流成分，交流成分に分解される．交流成分は，水平方向，垂直方向の各周波数成分電力量を表している．周波数成分の変化を測定したところ，ブロック内の水平方向，垂直方向，斜め方向３つ用を用いれば，表情変化を検出するために十分である事が予備実験で判明したため，図１１ｂで示すように，各方向の周波数成分毎に領域分割を行い，領域内の交流電力量和を求めている．得られた各方向毎の電力量和の定常状態（無表情状態）からの増減を特徴量とする．

周波数領域変換は、ＤＣＴの他に２次元ＦＦＴ等が考えられるが、高速変換においては、ＤＣＴに勝るものは無い。また、集積回路の規模を考えるとＤＣＴのブロックサイズを大き

くする事はあまり現実的ではないので、今回は精度に多少の問題を残しながらも、8ピクセルのブロックを用いている。ＤＳＰ（DigiralSignalProcesser）を用いて、高速処理をおこない、これらの問題を解決する手段も提案されているが、実際には、パソコンとの通信のオーバーヘッドが大きくパフォーマンスはそれ程上がらない。高速密結合が可能なプロセッサを用いて並列処理する手法を実現出来れば、ブロックサイズを変更して検出精度を上げる方法も考えられる。



(a) ＤＣＴによる周波数領域変換

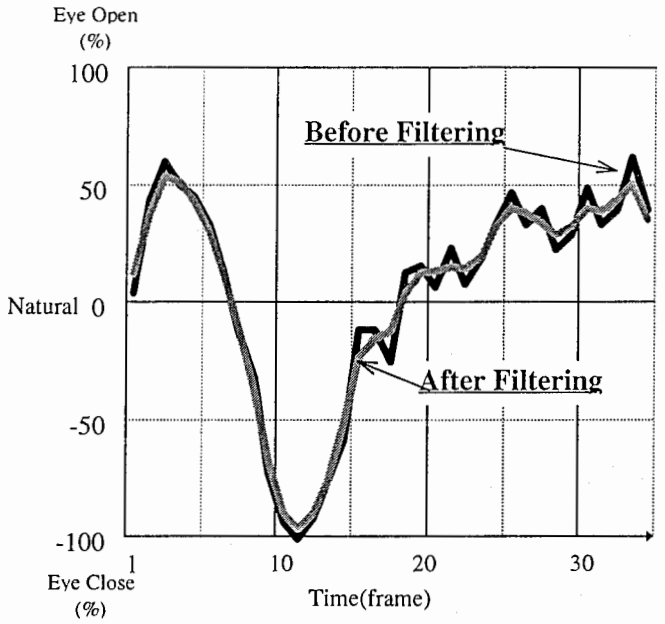


(b) ＤＣＴによるベクトル化用ディジタルフィルタ

図１１　ＤＣＴによる周波数成分のベクトル化

3.2.3 時間軸処理

検出された周波数成分の変化は，各フレーム毎の瞬間の値であるため，これを時間方向に並べた場合に，入力用の小型ＣＣＤカメラのノイズや，回路のノイズ成分を有している．そこで，１次元の時間軸フィルタを施し，これらの影響を除去している．このフィルタ係数は，「瞬き」の検出は可能であるが，それ以上早い動作はノイズとみなし除去出来る範囲に設定する．図１２は，フィルタ処理前と処理後の比較のグラフであるが，「瞬き」は検出し，それ以外のノイズのみを除去している．本稿では前後の３フレームを利用した一般的なローパスフィルタでこれを実現しているため，入力された顔画像に対し約90msの遅れで表情検



出結果を表情再現している．　図１２　ノイズ除去例（驚き）

3.2.4 表情検出

3.1で述べたように，筆者らのシステムではマーカの貼付されている各点の動きの検出結果に基づき表情筋の動きを３次元顔モデルにおいて忠実に再現することにより，高品質な表情再現を実現する．そこで，本稿では従来マーカが貼付されていた各点を仮想マーカと呼び，3.2.2で述べた３方向の周波数成分の特徴量をマーカの移動量に変換する手法を提案する．図１３に，口のマーカと検出領域例を示すように，本稿では，周波数成分の変化と仮想マーカの移動量を，ＧＡを用いた学習により，関連付ける式を求めこれを利用する．つまり，会議参加者はマーカを貼付して目や口の開閉動作等の表情筋の最小から最大までの連続動作を行い，マーカの移動量と周波数成分の変化を同時に計測する．計測した両者のデータの関連を学習し，周波数成分の変化からマーカの位置を計算出来るパラメータを求めることにより表情筋の動作に変換出来る．筆者らはこの周波数成分の変化から得られるマーカ位置を仮想マーカとして表情再現に用いている．

口の開閉動作（図１３）を，ＧＡを用いて学習する場合には，フレーム数iの口の３つの領域の水平周波数成分をH_i，垂直周波数成分をV_i，斜方向周波数成分D_i（i=0～n：フレーム数）とすると，無表情状態の時からの各周波数成分の変化量H_{dn}，V_{dn}，D_{dn}は，以下のようにして求められる．

$$\begin{aligned} H_{dn} &= H_n - H_0 \\ V_{dn} &= V_n - V_0 \\ D_{dn} &= D_n - D_0 \end{aligned} \qquad (4)$$

同様に，口のマーカの２次元座標の位置ベクトルを$\overrightarrow{x_i}$，$\overrightarrow{y_i}$とすると，マーカの移動ベク

トルは

$$\overrightarrow{X_i} = \overrightarrow{x_n} - \overrightarrow{x_0}$$
$$\overrightarrow{Y_i} = \overrightarrow{y_n} - \overrightarrow{y_0} \qquad (5)$$

で表されれる．従って，ＤＣＴにより求めた各周波数成分の変化量パラメータとする

$$\overrightarrow{X_i} = A_h H_{di} + A_v V_{di} + A_d D_{di}$$
$$\overrightarrow{Y_i} = B_h H_{di} + B_v V_{di} + B_d D_{di} \qquad (6)$$

を定義し，ＧＡの学習データからパラメータ列A_m，B_mを求めれば，周波数成分の変化のみで仮想マーカの移動量を求めることが出来る．これらの処理は，ニューラルネットワークの分野で行われる逐次学習では無いため，ＧＡによる学習は一度だけ行えばよい．

図１３の例では，口の周辺に４つのマーカを配置し表情筋の動きを検出している．ここで，実際に貼付したマーカと仮想マーカのと実際に貼付した下唇の直下にあるマーカｃの移動量を，垂直方向の動きは定常状態のとの差であるで表され，水平方向の動きはととの差であるで表される．図１４は，下唇の下（図１３中のａのマーカ）の動きを，式（６）のＧＡで求めたパラメータを用いて算出した仮想マーカの移動量と，実際のマーカの移動量とを１つのグラフにまとめたものである．口の中では最も動きの激しい部分で比較したが，両者の値は極めて近い値を示しており，周波数成分の変化量が表情筋の動きに極めて高い精度で変換されていることを示している．

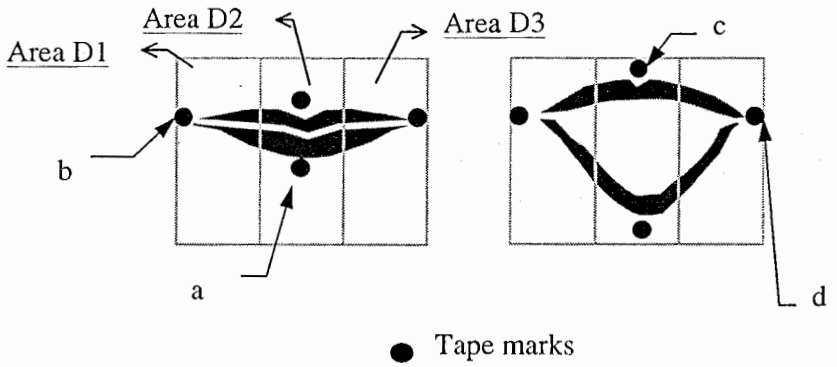


図１３　口の検出領域

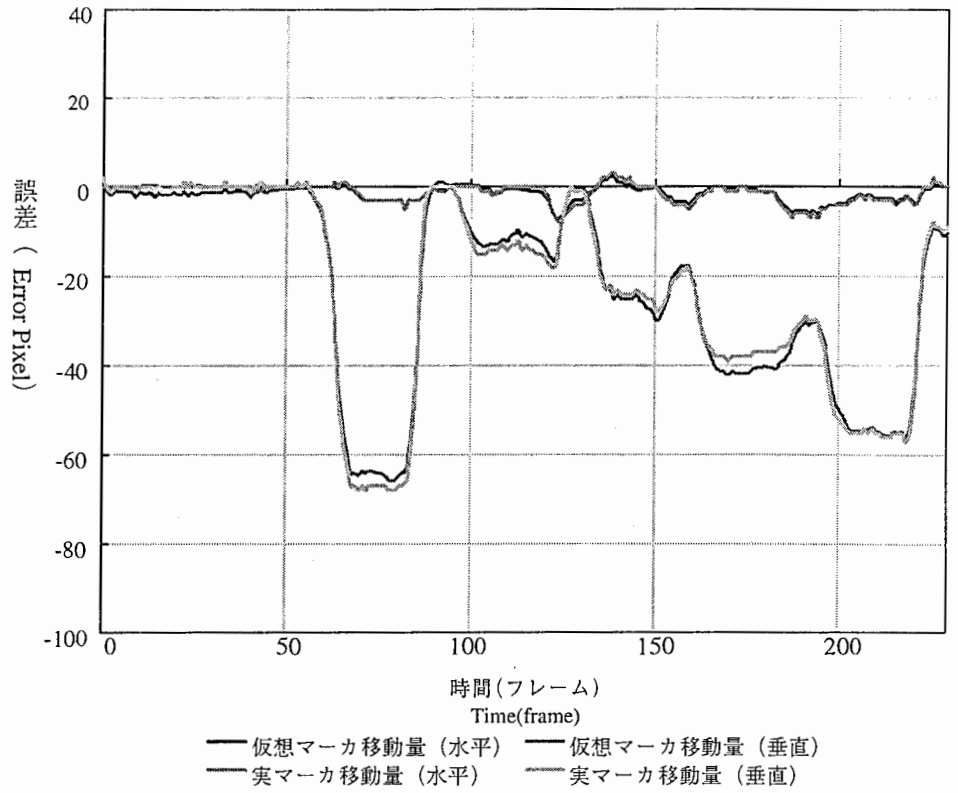


図１４　仮想マーカと実マーカの移動比較

４．実験結果

本稿で提案する実時間表情検出法の有効性を検証するために,パソコンをベースとするシステムを構築するとともに，２章で概説した筆者らの表情再現系と接続し，実験を行った.

まず,顔画像中に設定した表情検出領域のロバスト性を確認するため,図設定した図１５で示すように右目の検出領域を顔画像中で移動し無表情から笑顔までの表情変化をシーケンスで検出したデータと,正規位置で検出した周波数成分との誤差をフレーム毎の平均値に変換して両者の比較検討を行った.図１６に各方向に移動した場合の誤差のグラフを示すが,検出領域内に目と眉が存在すれば,得られる周波数成分の変化量に大きな差が無いが,下方に検出領域が移動し眉が領域外に移動すると急激に精度が悪化する事が判明した.しかし,実際にヘルメットに固定した小型ＣＣＤカメラは微調整が可能であり正常検出範囲が２０画素以上あり,貼付したマーカが数画素のサイズで微妙な調整が要求される場合に比べ,大幅な自由度が確保されていることが確認できた.つぎに,ヘルメットを斜め方向に装着した場合を想定し,顔画像を回転させて同様の比較を行った.図１７にしめすように時計回りに５度ずつ顔画像を回転させて前者と同様の実験を行った.結果を図１８に示すが,２０度までの周波数成分の変化に大きな差は見られないが,２０度を超える回転で急激に誤差が増大する．本手法では水平，垂直，斜め方向の周波数成分のみを用い，さらに，図１１ｂで示すように各領域の設定しているため,画像の回転に対しては各方法の周波数成分が他の周波数成分の領域に混入するまで影響が出にくい特徴を有している.このように,表情検出領域が固定であっても,ロバスト性に優れているためにヘルメット装着時の位置精度が大幅に緩和され，同様に，同一のヘルメットを装着可能な頭部を持っている人であれば，ほぼ，同様の位置に顔構成要素が存在するため，一般的な個人差も吸収できることを確認した.

最後に実時間性を確認するために，図１９の構成でシステムを構築して動作確認を行った.パソコンの内部で用いているビデオ入力ボードは,高速バスを介して中央演算装置に転送され周波数領域に変換され,前述のＧＡで求められたパラメータを用いて,仮想マーカの移動量を算出し,表情再現系にネットワークを通じて通信される.パソコンはシングルタスクのオペレーティングシステムを用いているために,表情処理以外のオーバーヘッドが小さく，処理が条件分岐などの例外処理を必要としないため，表情再現品質は従来方式の顔にマーカを貼付する方式と同等の精度で表情検出が行われ，検出速度は２４（フレーム／秒）の速度を実現し，表情再現は２０（フレーム／秒）の速度を実現している．同一フレームの表情再現例を図２０に示すが，前述のように時間軸処理のために90msの時間遅れが存在するために,実画像に比べ再現した顔画像では口が完全に開ききってはいない．しかし,伝送の時間遅れは，従来の研究で500msまで許容されるとされているため [18]，十分な処理速度で検出から再現までを実現していると言える．

また,従来の研究によりテレビ電話などの双方向コミュニケーションに用いられる再生フレーム数は，従来の研究より１５（フレーム/秒）の速度で実現すれば画質的に満足する結果が得られるとされている [19]．現状のシステムは，ソフトウェアのみで実現しても，処理速度はこれを上回ってビデオレートに近い処理速度を実現しており,極めて有望な結果と言える．

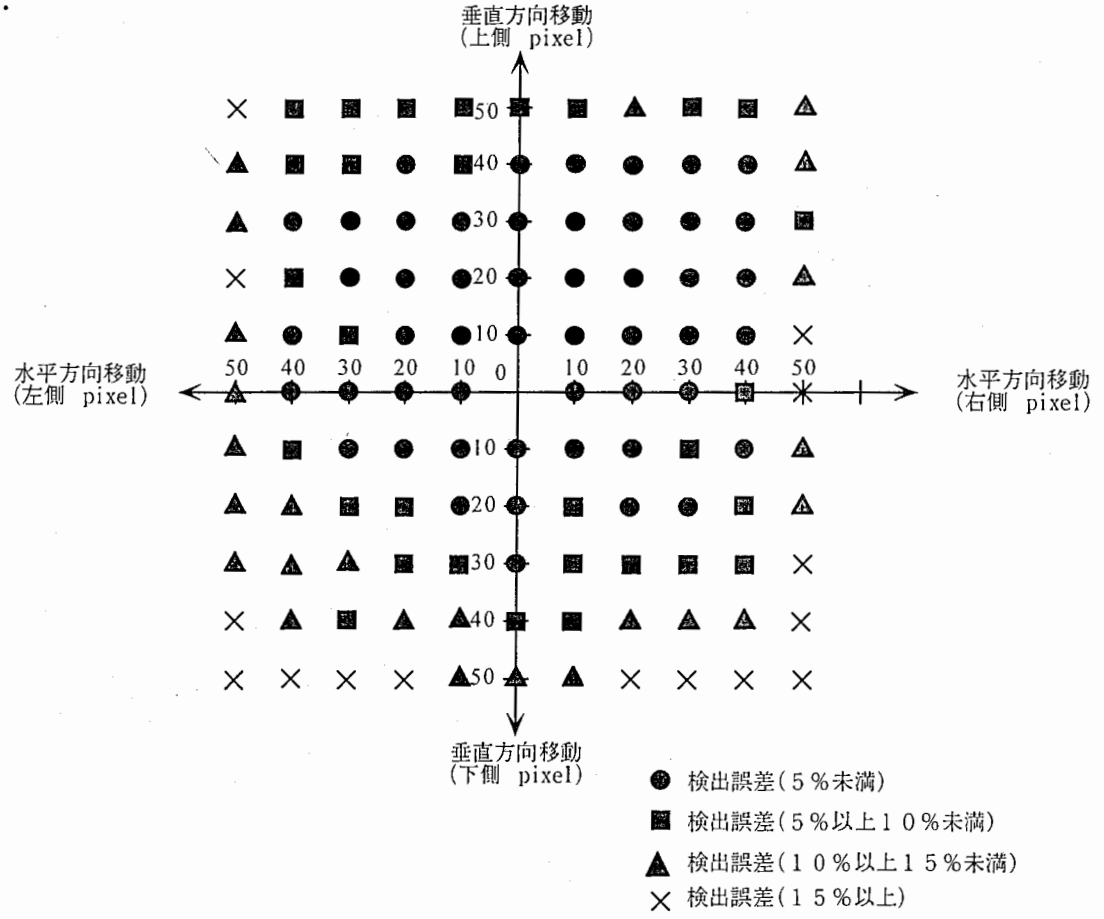


図１６　検出エリア移動による誤差

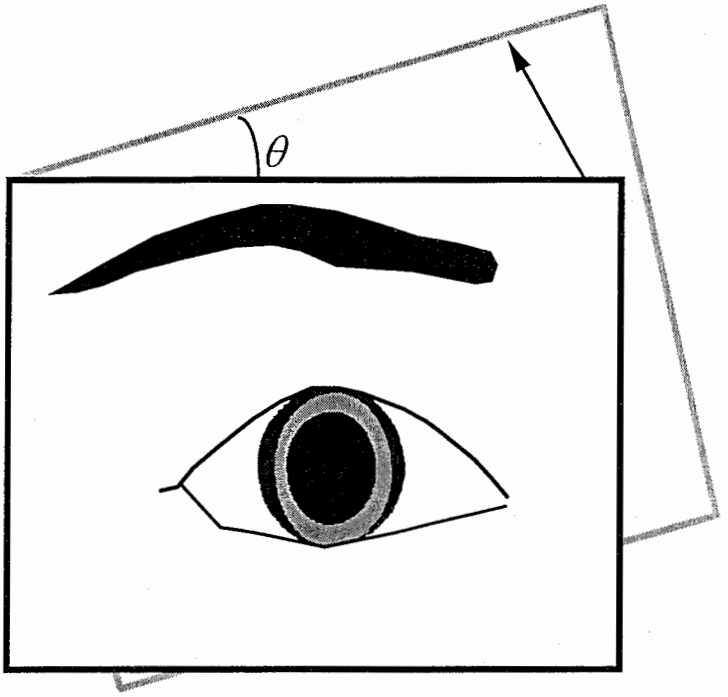


図１７　検出領域の回転

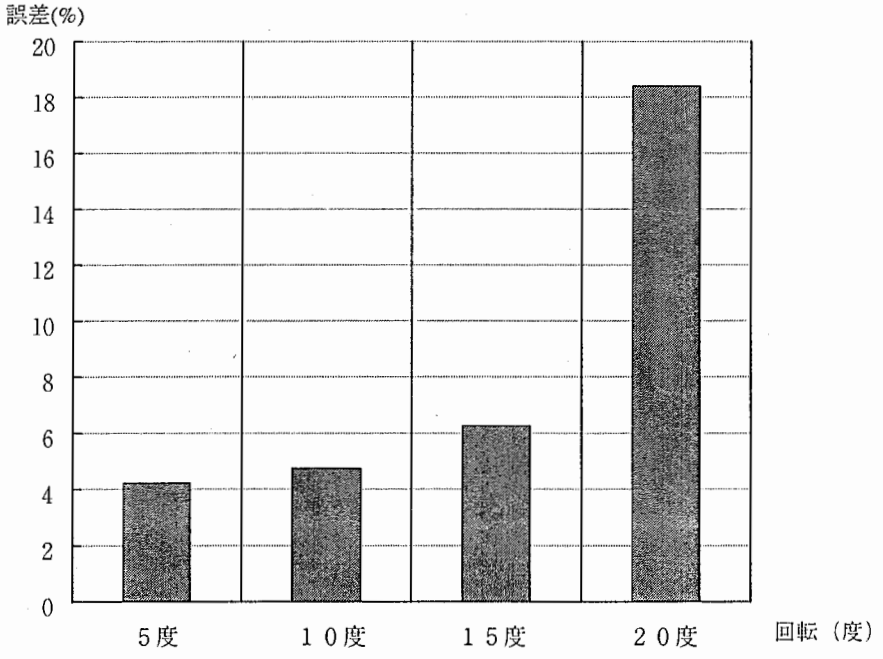


図１８　検出エリア回転による誤差

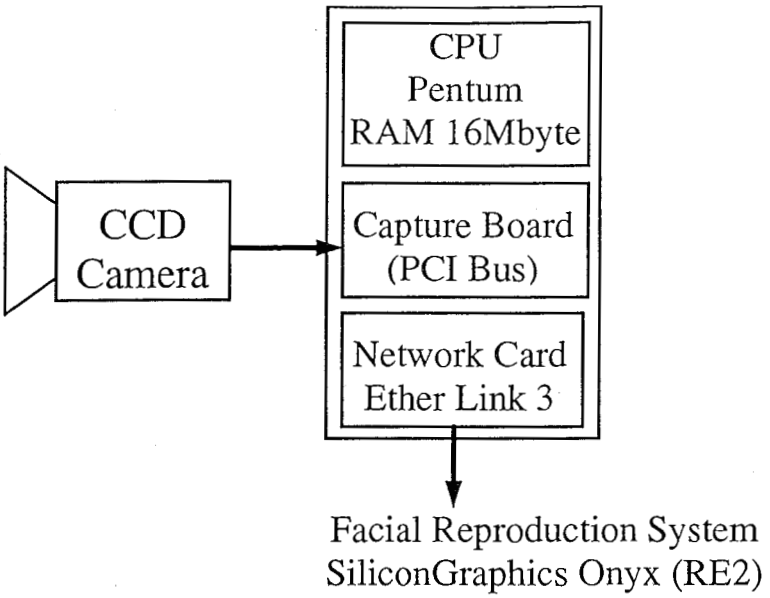


図１９　実時間表情検出システム

図２０　表情再現例

５．まとめ

臨場感通信会議システムにおいて，従来，顔に貼付していた実時間表情検出用のマーカを不要とし，素顔画像からの実時間表情検出を可能とする手法として，周波数領域変を利用する新しい手法を提案した．本稿の手法では，まずあらかじめ顔画像中で目，口，額の表情を検出する領域を設定し，水平，垂直，斜め方向の周波数のエネルギー成分を特徴量として検出する．この特徴量を，予め学習データに対してＧＡによる学習により獲得しておいた変換式を用いて，顔にマーカが貼付してあった場合に移動するべき仮想的マーカの位置に変換する．仮想マーカの移動量を，筆者らの表情再現系に入力することより表情再現される．

本稿では，以下のような実験的検討を行った．前述の周波数成分の変化と顔に貼付したマーカの移動量の関係をＧＡを用いて学習により関連付けて，検出された周波数成分の変化を仮想マーカの移動量に変換することで，高精度な表情検出が実現出来ることを確認した．また，表情検出のアルゴリズムは，ロバスト性に優れており，マーカ貼付時の厳密な位置精

度を要求せずに実時間表情検出が実現出来る見通しを得た.

今後は,不特定多数の人物の表情検出を行うえるようなアルゴリズムの確立を目指す.また,現状では会議参加者にはヘルメットの装着が必要であるが,これを不要とする方式を検討していく.

謝辞

本研究を進めるにあたり，ご指導いただきました（株）ＡＴＲ通信システム研究所 葉原耕平会長および寺島信義社長に感謝いたします.
また，有益な御討論御助言をしていただいた知能処理研究室の皆様に感謝いたします.

参考文献

[1] 岸野文郎："臨場感通信"，テレビ誌， Vol. 46, No. 6 , pp. 698-702 (Jun. '92).

[2]J. Ohya Y. Kitamura H.Takemura, H.Ishii, F. Kishino and N.Terashima:"Virtual Space Teleconferencing: Real-Time Reproduction of 3-D Human Image",CVCIR Vol. 6 No. 1 pp.1 - 25 (Mar. '95)

[3] 小林 宏，原 文雄："ニューラルネットによる人の顔の基本表情認識"，計測自動制御学会論文集，Vol.29 No.1 pp.112-118,1993

[4] Y.Kitamura,Y.Nagashima,J.Ohya and F.Kishino:"Facial image synthesis by hierarchical wire frame model",SPIE'92,Visual Comunication Image Processing '92.

[5] 森島繁生，岡田信一，原島博."知的インタフェースのための顔の表情合成法の一検討"，信学論 D-II, Vol. J73-D-II,No. 3, pp. 351-359 (Mar. '90)

[6] D. Terzopoulos and K. Waters:"Physically-based facial modeling, analysis ans animation",Journal of Visualization and Computer Animation,Vol. 1, pp. 73-80 ('90)

[7] W. J. Welsh,A. D. Simons,R. A. Hutchinson and S. Searby,"Synthetic face generation for enhancing a user interface",Proc. Image'Com '90,pp. 177-182 ('90)

[8] 森島繁生，小林誠司，原島博."マルチプロセッサ構成による知的画像符号化のためのリアルタイム表情合成の試み"，信学論 D-II, J73-D-II, No. 10, pp. 1647-1654 (Oct. '90)

[9] D.E.Golaberg,"Genetic Algorithms in Search,Optimization, and Machine Learning",Addison-Wesley Publishing Company Inc,1989

[10] 大谷淳，北村泰一，竹村治雄，岸野文郎："臨場感通信会議システムにおける３次元顔画像の実時間表示"，信学技報 HC92-61.(Jan. '93)

[11] 海老原一之，大谷淳，鈴木紀子，岸野文郎："３次元計測に基づいた顔画像の実時間表情再現"，電子情報通信学会誌Ａ ,Vol.J9-A,No2,pp.527-536,1996

[12] 海老原一之他，"臨場感通信会議システムにおける３次元顔画像処理"，信学技報 HC93-74,IE95-60(Sep.'95)

[13] K.Ebihara,J.Ohya,F.kishino:"A Study of Real Time Facial Expression Detection for Virtual Space Teleconferencing",Roman'95,pp.247-252

[14] 海老原一之，大谷淳，岸野文郎："周波数領域変換を用いた実時間表情検出"，電子情報通信学会総合大会，SD-8-1,(Mar.'95)
[15] 赤松，佐々木，深町，末永："濃淡画像マッチングによるロバストな正面顔の識別法－フーリエスペクトルのＫＬ展開の応用"，電子情報通信学会誌Ｄ－ＩＩ，Vol.76-D-II,No7,pp.1363-1373,1993
[16] 坂口竜己，大谷淳，岸野文郎："隠れマルコフモデルによる顔動画像からの表情認識"，テレビ誌 Vol.49,No.8,pp.1060-1067,(Aug.'95)
[17] "Digital compression and coding of continuous-tone still images: requirements and guidelines", ISO/IEC 10918-1.
[18] 鎧澤勇，滝川啓，大久保栄，渡辺義郎："衛星を用いた画像会議にいけるエコー及び伝搬遅延の影響"，電子情報通信学会誌Ｂ，Vol.J64-B,No11,pp.1281-1288,1981
[19] 鎧澤勇："特集： テレビ電話・テレコンファレンス，5. 共通要素技術 5.1 ヒューマンファクタ"，テレビ誌 Vol.42,No.11,pp.1193--1198,(Nov. '88)

付録Ａ　インプリメント上の注意

１．ビデオキャプチャーボード

顔画像を取り込むビデオキャプチャボードは、ＰＣＩバス対応の物が望ましい。特に顔画像全サイズを取り込む事を考えると、キャプチャしたデータをＣＰＵに転送するスピードは可能な限り早い方が良い。今回は、カナダのMatrox社製のCometを採用したが、このボードは、ソフト開発キットが提供されると共に、アナログ回路の設計が非常に良く出来ている。現在は、この後継機種が発売されているが、在庫品の入手が可能な限りはこのボードを採用する事が望ましい。(特性データがそのまま使用可能である)

２．マザーボード

本システムで使用した、IBM　PC（100%compatible）は、GATEWAY社の物を使用した。動作クロックは120MHzで、バス転送速度は30MHzで規定されている。従って、１で示したキャプチャーボードとＣＰＵ間のタイミングは、約26MHzで送信されている。

ＰＣＩバスは、いくつかの規定があり、本システムで使用した規定は2.0と呼ばれるものである。このバージョンが異なると、転送時に問題が発生する可能性があるが、ソフト以外にも、バス転送速度で問題が発生する危険性がある。タイミングの問題を解決するためには、120MHz以外を使用する場合には、90MHz、150MHzを使用し、133MHz、166MHzの使用を避けるようにする。

３．小型ＣＣＤカメラ

本システムで、松下電器製のWV-KS102を使用している。現在はこの後継機種が出ているが、水平解像度が２８０本（約2.8MHzの帯域）以上であれば、あまり機種依存は考えられない。しかし、装着するレンズは、広角レンズのなかで収差の少ないものを選べる機種の方がマーカ移動量の誤差を最少に押さえる事が出来る。

あまり、望遠側のレンズを使用すると、顔とカメラの位置が離れすぎると、人間の動作による振動が映像信号に悪影響を与えるため、可能な限り広角レンズを使用する事を推賞する。

４．マーカ抽出装置

本システムでは、学習用に１度だけ顔にマーカを貼付する必要が、このマーカ抽出装置については、別のテクニカルレポートに回路図付きで発行されている。そちらを併せて参照すれば、この装置を作成できる。

５．ＤＣＴライブラリ

本テクニカルレポートに添付したプログラムソースでは、一般的なＤＣＴアルゴリズムを用いている。これは、「チェンのアルゴリズム」と呼ばれている最高速のアルゴリズムをインプリメントしたものである。しかし、プログラムはＣ言語で記述され、必ずしも最高速と

は言えない。

そこで、実際に用いているライブラリは、アセンブラ言語で書かれたライブラリである。この高速ライブラリは、カノープス（株）の提供によるが、ＡＴＲでライセンスを受けているため、販売以外の目的で使用する場合は、ＡＴＲ側から提供出来る。しかし、販売する場合には、個別にカノープス（株）とライセンスを締結する必要がある。

```
1:// area.h : header file
2://
3:
4://///////////////////////////////////////////////////////////////////////////
5:// CArea dialog
6:
7:class CArea : public CDialog
8:{
9:// Construction
10:public:
11:    CArea(CWnd* pParent = NULL);    // standard constructor
12:    CGrabDlg(CClientDC* pDC);
13:    BOOL Create();
14:
15:// Member variables
16:public:
17:    CWnd *m_hParent;
18://  CTraceView* m_pView;
19:
20:// Dialog Data
21:    //{{AFX_DATA(CArea)
22:    enum { IDD = IDD_DIALOG4 };
23:    CButton m_Ok;
24:    CButton m_Cancel;
25:    CString m_AreaFile;
26:    //}}AFX_DATA
27:
28:
29:// Overrides
30:public:
31:
32:protected:
33:    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
34:
35:// Implementation
36:protected:
37:
38:    // Generated message map functions
39:    //{{AFX_MSG(CArea)
40:        // NOTE: the ClassWizard will add member functions here
41:    //}}AFX_MSG
42:    DECLARE_MESSAGE_MAP()
43:};
44:
```

```
 1:// capboard.h : for matlox Comet
 2://
 3://////////////////////////////////////////////////////////////////////////////
 4:
 5:// Class definition
 6:
 7:class CCapBoard : public CObject
 8:{
 9:public:
10:    CCapBoard();
11:    DECLARE_DYNCREATE( CCapBoard )
12:
13:// Member variables
14:
```

```
1:////////////////////////////////////////////////////////////////////////////
2:// Class CCapture : CObject
3:
4:
5:// Class definition.
6:
7:class CCapture : public CObject
8:{
9:public:
10:    CCapture();
11:    ~CCapture();
12:    DECLARE_DYNCREATE( CCapture )
13:    CCapture(CView*);
14:
15:// Member variables
16:private:
17:    CView* m_pView;
18:    static int initFlag;
19:public:
20:    short *m_Buffer;
21:    int m_aNum;
22:    short   m_Mono[64];
23:    int     m_Ix[20];
24:    int     m_Iy[20];
25:    int     m_wx[20];
26:    int     m_wy[20];
27:    long int    m_GH[20];
28:    long int    m_GV[20];
29:    long int    m_GHV[20];
30:    long int    m_S[20];
31:    unsigned short  m_EY[2][3][16];
32:    short   m_HCut[8][8];
33:    short   m_VCut[8][8];
34:    short   m_HVCut[8][8];
35:    int     *m_Imax;
36:    int     *m_kmax;
37:    unsigned long m_Counter;
38:    int     m_EyeMid[2];
39:    CString m_AreaFile;
40:// Member functions
41:
42:    long  Addr(long, long);
43:    short Bufpix(int, int);
44:    void  Putpix(int, int, short);
45:
46:    int  CaptureInit();
47:    void CaptureQuit();
48:
49:    void CaptureStop();
50:    void CaptureStart();
51:    void FreeMalloc();
52:
53:    int  DctInit();
54:    void Dct();
55:    void dct_cul(int, int, int);
56:    void RGBtoMONO(int, int, int, int);
57:    void SigmaDCTBlock2(int);
58:};
59:
```

```
1:// GRABDLG.H : header file
2://
3:
4://///////////////////////////////////////////////////////////////////////////
5:// CGrabDlg dialog
6:class CCapture;
7:class CTraceView;
8:class CGrabDlg : public CDialog
9:{
10:// Construction
11:public:
12:    CGrabDlg(CWnd* pParent = NULL); // standard constructor
13:    CGrabDlg(CClientDC* pDC);
14:    CGrabDlg(CCapture* pCapture);
15:    BOOL Create();
16:
17:// Member variables
18:public:
19:    CWnd *m_hParent;
20:    CTraceView* m_pView;
21:    CCapture* m_pCapture;
22:
23:// Dialog Data
24:    //{{AFX_DATA(CGrabDlg)
25:    enum { IDD = IDD_DIALOG3 };
26:        // NOTE: the ClassWizard will add data members here
27:    //}}AFX_DATA
28:
29:// Member functions
30:public:
31:    void SetView(CTraceView* pView);
32:
33:// Overrides
34:public:
35:
36:protected:
37:    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
38:
39:// Implementation
40:protected:
41:
42:    // Generated message map functions
43:    //{{AFX_MSG(CGrabDlg)
44:    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
45:    //}}AFX_MSG
46:    DECLARE_MESSAGE_MAP()
47:};
48:
49:#define WM_DLGMSG WM_USER+301
50:
```

```
 1:/*
 2: * jconfig.h
 3: *
 4: * Copyright (C) 1991, 1992, 1993, Thomas G. Lane.
 5: * This file is part of the Independent JPEG Group's software.
 6: * For conditions of distribution and use, see the accompanying README file.
 7: *
 8: * This file contains preprocessor declarations that help customize
 9: * the JPEG software for a particular application, machine, or compiler.
10: * Edit these declarations as needed (or add -D flags to the Makefile).
11: */
12:
13:
14:/*
15: * These symbols indicate the properties of your machine or compiler.
16: * The conditional definitions given may do the right thing already,
17: * but you'd best look them over closely, especially if your compiler
18: * does not handle full ANSI C.  An ANSI-compliant C compiler should
19: * provide all the necessary features; __STDC__ is supposed to be
20: * predefined by such compilers.
21: */
22:
23:/*
24: * HAVE_STDC is tested below to see whether ANSI features are available.
25: * We avoid testing __STDC__ directly for arcane reasons of portability.
26: * (On some compilers, __STDC__ is only defined if a switch is given,
27: * but the switch also disables machine-specific features we need to get at.
28: * In that case, -DHAVE_STDC in the Makefile is a convenient solution.)
29: */
30:
31:#ifdef __STDC__         /* if compiler claims to be ANSI, believe it */
32:#define HAVE_STDC
33:#endif
34:
35:
36:/* Does your compiler support function prototypes? */
37:/* (If not, you also need to use ansi2knr, see SETUP) */
38:
39:#ifdef HAVE_STDC        /* ANSI C compilers always have prototypes */
40:#define PROTO
41:#else
42:#ifdef __cplusplus      /* So do C++ compilers */
43:#define PROTO
44:#endif
45:#endif
46:
47:/* Does your compiler support the declaration "unsigned char" ? */
48:/* How about "unsigned short" ? */
49:
50:#ifdef HAVE_STDC        /* ANSI C compilers must support both */
51:#define HAVE_UNSIGNED_CHAR
52:#define HAVE_UNSIGNED_SHORT
53:#endif
54:
55:/* Define this if an ordinary "char" type is unsigned.
56: * If you're not sure, leaving it undefined will work at some cost in speed.
57: * If you defined HAVE_UNSIGNED_CHAR then it doesn't matter very much.
58: */
59:
60:/* #define CHAR_IS_UNSIGNED */
61:
62:/* Define this if your compiler implements ">>" on signed values as a logical
63: * (unsigned) shift; leave it undefined if ">>" is a signed (arithmetic) shift,
64: * which is the normal and rational definition.
65: */
66:
67:/* #define RIGHT_SHIFT_IS_UNSIGNED */
68:
```

```
 69:/* Define "void" as "char" if your compiler doesn't know about type void.
 70: * NOTE: be sure to define void such that "void *" represents the most general
 71: * pointer type, e.g., that returned by malloc().
 72: */
 73:
 74:/* #define void char */
 75:
 76:/* Define const as empty if your compiler doesn't know the "const" keyword. */
 77:/* (Even if it does, defining const as empty won't break anything.) */
 78:
 79:#ifndef HAVE_STDC       /* ANSI C and C++ compilers should know it. */
 80:#ifndef __cplusplus
 81:#define const
 82:#endif
 83:#endif
 84:
 85:/* For 80x86 machines, you need to define NEED_FAR_POINTERS,
 86: * unless you are using a large-data memory model or 80386 flat-memory mode.
 87: * On less brain-damaged CPUs this symbol must not be defined.
 88: * (Defining this symbol causes large data structures to be referenced through
 89: * "far" pointers and to be allocated with a special version of malloc.)
 90: */
 91:
 92:#ifdef MSDOS
 93:#define NEED_FAR_POINTERS
 94:#endif
 95:
 96:
 97:/* The next three symbols only affect the system-dependent user interface
 98: * modules (jcmain.c, jdmain.c).  You can ignore these if you are supplying
 99: * your own user interface code.
100: */
101:
102:/* Define this if you want to name both input and output files on the command
103: * line, rather than using stdout and optionally stdin.  You MUST do this if
104: * your system can't cope with binary I/O to stdin/stdout.  See comments at
105: * head of jcmain.c or jdmain.c.
106: */
107:
108:#ifdef MSDOS            /* two-file style is needed for PCs */
109:#ifndef USE_FDOPEN      /* unless you have fdopen() or setmode() */
110:#ifndef USE_SETMODE
111:#define TWO_FILE_COMMANDLINE
112:#endif
113:#endif
114:#endif
115:#ifdef THINK_C          /* it's needed for Macintosh too */
116:#define TWO_FILE_COMMANDLINE
117:#endif
118:
119:/* Define this if your system needs explicit cleanup of temporary files.
120: * This is crucial under MS-DOS, where the temporary "files" may be areas
121: * of extended memory; on most other systems it's not as important.
122: */
123:
124:#ifdef MSDOS
125:#define NEED_SIGNAL_CATCHER
126:#endif
127:
128:/* By default, we open image files with fopen(...,"rb") or fopen(...,"wb").
129: * This is necessary on systems that distinguish text files from binary files,
130: * and is harmless on most systems that don't.  If you have one of the rare
131: * systems that complains about the "b" spec, define this symbol.
132: */
133:
134:/* #define DONT_USE_B_MODE */
135:
136:
```

```
/* If you're getting bored, that's the end of the symbols you HAVE to
 * worry about.  Go fix the makefile and compile.
 */


/* If your compiler supports inline functions, define INLINE
 * as the inline keyword; otherwise define it as empty.
 */

#ifdef __GNUC__         /* for instance, GNU C knows about inline */
#define INLINE __inline__
#endif
#ifndef INLINE          /* default is to define it as empty */
#define INLINE
#endif

/* On a few systems, type boolean and/or macros FALSE, TRUE may appear
 * in standard header files.  Or you may have conflicts with application-
 * specific header files that you want to include together with these files.
 * In that case you need only comment out these definitions.
 */

typedef int boolean;
#undef FALSE            /* in case these macros already exist */
#undef TRUE
#define FALSE   0       /* values of boolean */
#define TRUE    1

/* This defines the size of the I/O buffers for entropy compression
 * and decompression; you could reduce it if memory is tight.
 */

#define JPEG_BUF_SIZE   4096 /* bytes */


/* These symbols determine the JPEG functionality supported. */

/*
 * These defines indicate whether to include various optional functions.
 * Undefining some of these symbols will produce a smaller but less capable
 * program file.  Note that you can leave certain source files out of the
 * compilation/linking process if you've #undef'd the corresponding symbols.
 * (You may HAVE to do that if your compiler doesn't like null source files.)
 */

/* Arithmetic coding is unsupported for legal reasons.  Complaints to IBM. */

/* Encoder capability options: */
#undef  C_ARITH_CODING_SUPPORTED    /* Arithmetic coding back end? */
#undef  C_MULTISCAN_FILES_SUPPORTED /* Multiple-scan JPEG files?  (NYI) */
#define ENTROPY_OPT_SUPPORTED       /* Optimization of entropy coding parms? */
#define INPUT_SMOOTHING_SUPPORTED   /* Input image smoothing option? */
/* Decoder capability options: */
#undef  D_ARITH_CODING_SUPPORTED    /* Arithmetic coding back end? */
#define D_MULTISCAN_FILES_SUPPORTED /* Multiple-scan JPEG files? */
#define BLOCK_SMOOTHING_SUPPORTED   /* Block smoothing during decoding? */
#define QUANT_1PASS_SUPPORTED   /* 1-pass color quantization? */
#define QUANT_2PASS_SUPPORTED   /* 2-pass color quantization? */
/* these defines indicate which JPEG file formats are allowed */
#define JFIF_SUPPORTED      /* JFIF or "raw JPEG" files */
#undef  JTIFF_SUPPORTED     /* JPEG-in-TIFF (not yet implemented) */
/* these defines indicate which image (non-JPEG) file formats are allowed */
#define GIF_SUPPORTED       /* GIF image file format */
/* #define RLE_SUPPORTED */ /* RLE image file format (by default, no) */
#define PPM_SUPPORTED       /* PPM/PGM image file format */
#define TARGA_SUPPORTED     /* Targa image file format */
#undef  TIFF_SUPPORTED      /* TIFF image file format (not yet impl.) */
```

```
205:
206:/* more capability options later, no doubt */
207:
208:
209:/*
210: * Define exactly one of these three symbols to indicate whether you want
211: * 8-bit, 12-bit, or 16-bit sample (pixel component) values.  8-bit is the
212: * default and is nearly always the right thing to use.  You can use 12-bit if
213: * you need to support image formats with more than 8 bits of resolution in a
214: * color value.  16-bit should only be used for the lossless JPEG mode (not
215: * currently supported).  Note that 12- and 16-bit values take up twice as
216: * much memory as 8-bit!
217: * Note: if you select 12- or 16-bit precision, it is dangerous to turn off
218: * ENTROPY_OPT_SUPPORTED.  The standard Huffman tables are only good for 8-bit
219: * precision, so jchuff.c normally uses entropy optimization to compute
220: * usable tables for higher precision.  If you don't want to do optimization,
221: * you'll have to supply different default Huffman tables.
222: */
223:
224:#define EIGHT_BIT_SAMPLES
225:#undef  TWELVE_BIT_SAMPLES
226:#undef  SIXTEEN_BIT_SAMPLES
227:
228:
229:
230:/*
231: * The remaining definitions don't need to be hand-edited in most cases.
232: * You may need to change these if you have a machine with unusual data
233: * types; for example, "char" not 8 bits, "short" not 16 bits,
234: * or "long" not 32 bits.  We don't care whether "int" is 16 or 32 bits,
235: * but it had better be at least 16.
236: */
237:
238:/* First define the representation of a single pixel element value. */
239:
240:#ifdef EIGHT_BIT_SAMPLES
241:/* JSAMPLE should be the smallest type that will hold the values 0..255.
242: * You can use a signed char by having GETJSAMPLE mask it with 0xFF.
243: * If you have only signed chars, and you are more worried about speed than
244: * memory usage, it might be a win to make JSAMPLE be short.
245: */
246:
247:#ifdef HAVE_UNSIGNED_CHAR
248:
249:typedef unsigned char JSAMPLE;
250:#define GETJSAMPLE(value)  (value)
251:
252:#else /* not HAVE_UNSIGNED_CHAR */
253:#ifdef CHAR_IS_UNSIGNED
254:
255:typedef char JSAMPLE;
256:#define GETJSAMPLE(value)  (value)
257:
258:#else /* not CHAR_IS_UNSIGNED */
259:
260:typedef char JSAMPLE;
261:#define GETJSAMPLE(value)  ((value) & 0xFF)
262:
263:#endif /* CHAR_IS_UNSIGNED */
264:#endif /* HAVE_UNSIGNED_CHAR */
265:
266:#define BITS_IN_JSAMPLE   8
267:#define MAXJSAMPLE  255
268:#define CENTERJSAMPLE   128
269:
270:#endif /* EIGHT_BIT_SAMPLES */
271:
272:
```

```
273:#ifdef TWELVE_BIT_SAMPLES
274:/* JSAMPLE should be the smallest type that will hold the values 0..4095. */
275:/* On nearly all machines "short" will do nicely. */
276:
277:typedef short JSAMPLE;
278:#define GETJSAMPLE(value)  (value)
279:
280:#define BITS_IN_JSAMPLE   12
281:#define MAXJSAMPLE  4095
282:#define CENTERJSAMPLE   2048
283:
284:#endif /* TWELVE_BIT_SAMPLES */
285:
286:
287:#ifdef SIXTEEN_BIT_SAMPLES
288:/* JSAMPLE should be the smallest type that will hold the values 0..65535. */
289:
290:#ifdef HAVE_UNSIGNED_SHORT
291:
292:typedef unsigned short JSAMPLE;
293:#define GETJSAMPLE(value)  (value)
294:
295:#else /* not HAVE_UNSIGNED_SHORT */
296:
297:/* If int is 32 bits this'll be horrendously inefficient storage-wise.
298: * But since we don't actually support 16-bit samples (ie lossless coding) yet,
299: * I'm not going to worry about making a smarter definition ...
300: */
301:typedef unsigned int JSAMPLE;
302:#define GETJSAMPLE(value)  (value)
303:
304:#endif /* HAVE_UNSIGNED_SHORT */
305:
306:#define BITS_IN_JSAMPLE    16
307:#define MAXJSAMPLE  65535
308:#define CENTERJSAMPLE   32768
309:
310:#endif /* SIXTEEN_BIT_SAMPLES */
311:
312:
313:/* Here we define the representation of a DCT frequency coefficient.
314: * This should be a signed 16-bit value; "short" is usually right.
315: * It's important that this be exactly 16 bits, no more and no less;
316: * more will cost you a BIG hit of memory, less will give wrong answers.
317: */
318:
319:typedef short JCOEF;
320:
321:
322:/* The remaining typedefs are used for various table entries and so forth.
323: * They must be at least as wide as specified; but making them too big
324: * won't cost a huge amount of memory, so we don't provide special
325: * extraction code like we did for JSAMPLE.  (In other words, these
326: * typedefs live at a different point on the speed/space tradeoff curve.)
327: */
328:
329:/* UINT8 must hold at least the values 0..255. */
330:
331:#ifdef HAVE_UNSIGNED_CHAR
332:typedef unsigned char UINT8;
333:#else /* not HAVE_UNSIGNED_CHAR */
334:#ifdef CHAR_IS_UNSIGNED
335:typedef char UINT8;
336:#else /* not CHAR_IS_UNSIGNED */
337:typedef short UINT8;
338:#endif /* CHAR_IS_UNSIGNED */
339:#endif /* HAVE_UNSIGNED_CHAR */
340:
```

```
341:/* UINT16 must hold at least the values 0..65535. */
342:
343:#ifdef HAVE_UNSIGNED_SHORT
344:typedef unsigned short UINT16;
345:#else /* not HAVE_UNSIGNED_SHORT */
346:typedef unsigned int UINT16;
347:#endif /* HAVE_UNSIGNED_SHORT */
348:
349:/* INT16 must hold at least the values -32768..32767. */
350:
351:#ifndef XMD_H           /* X11/xmd.h correctly defines INT16 */
352:typedef short INT16;
353:#endif
354:
355:/* INT32 must hold signed 32-bit values; if your machine happens */
356:/* to have 64-bit longs, you might want to change this. */
357:
358:#ifndef XMD_H           /* X11/xmd.h correctly defines INT32 */
359:typedef long INT32;
360:#endif
361:
```

```
 1:/*
 2: * jinclude.h
 3: *
 4: * Copyright (C) 1991, 1992, 1993, Thomas G. Lane.
 5: * This file is part of the Independent JPEG Group's software.
 6: * For conditions of distribution and use, see the accompanying README file.
 7: *
 8: * This is the central file that's #include'd by all the JPEG .c files.
 9: * Its purpose is to provide a single place to fix any problems with
10: * including the wrong system include files.
11: * You can edit these declarations if you use a system with nonstandard
12: * system include files.
13: */
14:
15:
16:/*
17: * Normally the __STDC__ macro can be taken as indicating that the system
18: * include files conform to the ANSI C standard.  However, if you are running
19: * GCC on a machine with non-ANSI system include files, that is not the case.
20: * In that case change the following, or add -DNONANSI_INCLUDES to your CFLAGS.
21: */
22:
23:#ifdef __STDC__
24:#ifndef NONANSI_INCLUDES
25:#define INCLUDES_ARE_ANSI   /* this is what's tested before including */
26:#endif
27:#endif
28:
29:/*
30: * <stdio.h> is included to get the FILE typedef and NULL macro.
31: * Note that the core portable-JPEG files do not actually do any I/O
32: * using the stdio library; only the user interface, error handler,
33: * and file reading/writing modules invoke any stdio functions.
34: * (Well, we did cheat a bit in jmemmgr.c, but only if MEM_STATS is defined.)
35: */
36:
37:#include <stdio.h>
38:
39:/*
40: * We need the size_t typedef, which defines the parameter type of malloc().
41: * In an ANSI-conforming implementation this is provided by <stdio.h>,
42: * but on non-ANSI systems it's more likely to be in <sys/types.h>.
43: * On some not-quite-ANSI systems you may find it in <stddef.h>.
44: */
45:
46:#ifndef INCLUDES_ARE_ANSI   /* shouldn't need this if ANSI C */
47:#include <sys/types.h>
48:#endif
49:#ifdef __SASC           /* Amiga SAS C provides it in stddef.h. */
50:#include <stddef.h>
51:#endif
52:
53:/*
54: * In ANSI C, and indeed any rational implementation, size_t is also the
55: * type returned by sizeof().  However, it seems there are some irrational
56: * implementations out there, in which sizeof() returns an int even though
57: * size_t is defined as long or unsigned long.  To ensure consistent results
58: * we always use this SIZEOF() macro in place of using sizeof() directly.
59: */
60:
61:#undef SIZEOF           /* in case you included X11/xmd.h */
62:#define SIZEOF(object)  ((size_t) sizeof(object))
63:
64:/*
65: * fread() and fwrite() are always invoked through these macros.
66: * On some systems you may need to twiddle the argument casts.
67: * CAUTION: argument order is different from underlying functions!
68: */
```

```
 69:
 70:#define JFREAD(file,buf,sizeofbuf)  ¥
 71:  ((size_t) fread((void *) (buf), (size_t) 1, (size_t) (sizeofbuf), (file)))
 72:#define JFWRITE(file,buf,sizeofbuf)  ¥
 73:  ((size_t) fwrite((const void *) (buf), (size_t) 1, (size_t) (sizeofbuf), (file)))
 74:
 75:/*
 76: * We need the memcpy() and strcmp() functions, plus memory zeroing.
 77: * ANSI and System V implementations declare these in <string.h>.
 78: * BSD doesn't have the mem() functions, but it does have bcopy()/bzero().
 79: * Some systems may declare memset and memcpy in <memory.h>.
 80: *
 81: * NOTE: we assume the size parameters to these functions are of type size_t.
 82: * Change the casts in these macros if not!
 83: */
 84:
 85:#ifdef INCLUDES_ARE_ANSI
 86:#include <string.h>
 87:#define MEMZERO(target,size)    memset((void *)(target), 0, (size_t)(size))
 88:#define MEMCOPY(dest,src,size)  memcpy((void *)(dest), (const void *)(src), (size_t)(size))
 89:#else /* not ANSI */
 90:#ifdef BSD
 91:#include <strings.h>
 92:#define MEMZERO(target,size)    bzero((void *)(target), (size_t)(size))
 93:#define MEMCOPY(dest,src,size)  bcopy((const void *)(src), (void *)(dest), (size_t)(size))
 94:#else /* not BSD, assume Sys V or compatible */
 95:#include <string.h>
 96:#define MEMZERO(target,size)    memset((void *)(target), 0, (size_t)(size))
 97:#define MEMCOPY(dest,src,size)  memcpy((void *)(dest), (const void *)(src), (size_t)(size))
 98:#endif /* BSD */
 99:#endif /* ANSI */
100:
101:
102:/* Now include the portable JPEG definition files. */
103:
104:#include "jconfig.h"
105:
106:#include "jpegdata.h"
107:
```

```
/*
 * jpegdata.h
 *
 * Copyright (C) 1991, 1992, 1993, Thomas G. Lane.
 * This file is part of the Independent JPEG Group's software.
 * For conditions of distribution and use, see the accompanying README file.
 *
 * This file defines shared data structures for the various JPEG modules.
 */


/*
 * You might need to change some of the following declarations if you are
 * using the JPEG software within a surrounding application program
 * or porting it to an unusual system.
 */


/* If the source or destination of image data is not to be stdio streams,
 * these types may need work.  You can replace them with some kind of
 * pointer or indicator that is useful to you, or just ignore 'em.
 * Note that the user interface and the various jrdxxx/jwrxxx modules
 * will also need work for non-stdio input/output.
 */

typedef FILE * JFILEREF;    /* source or dest of JPEG-compressed data */

typedef FILE * IFILEREF;    /* source or dest of non-JPEG image data */


/* These defines are used in all function definitions and extern declarations.
 * You could modify them if you need to change function linkage conventions,
 * as is shown below for use with C++.  Another application would be to make
 * all functions global for use with code profilers that require it.
 * NOTE: the C++ test does the right thing if you are reading this include
 * file in a C++ application to link to JPEG code that's been compiled with a
 * regular C compiler.  I'm not sure it works if you try to compile the JPEG
 * code with C++.
 */

#define METHODDEF static    /* a function called through method pointers */
#define LOCAL     static    /* a function used only in its module */
#define GLOBAL          /* a function referenced thru EXTERNs */
#ifdef __cplusplus
#define EXTERN    extern "C"    /* a reference to a GLOBAL function */
#else
#define EXTERN    extern    /* a reference to a GLOBAL function */
#endif


/* Here is the pseudo-keyword for declaring pointers that must be "far"
 * on 80x86 machines.  Most of the specialized coding for 80x86 is handled
 * by just saying "FAR *" where such a pointer is needed.  In a few places
 * explicit coding is needed; see uses of the NEED_FAR_POINTERS symbol.
 */

#ifdef NEED_FAR_POINTERS
#define FAR  far
#else
#define FAR
#endif


/* The remaining declarations are not system-dependent, we hope. */


/*
```

```
 69: * NOTE: if you have an ancient, strict-K&R C compiler, it may choke on the
 70: * similarly-named fields in Compress_info_struct and Decompress_info_struct.
 71: * If this happens, you can get around it by rearranging the two structs so
 72: * that the similarly-named fields appear first and in the same order in
 73: * each struct.  Since such compilers are now pretty rare, we haven't done
 74: * this in the portable code, preferring to maintain a logical ordering.
 75: */
 76:
 77:
 78:
 79:/* This macro is used to declare a "method", that is, a function pointer. */
 80:/* We want to supply prototype parameters if the compiler can cope. */
 81:/* Note that the arglist parameter must be parenthesized! */
 82:
 83:#ifdef PROTO
 84:#define METHOD(type,methodname,arglist)  type (*methodname) arglist
 85:#else
 86:#define METHOD(type,methodname,arglist)  type (*methodname) ()
 87:#endif
 88:
 89:/* Forward references to lists of method pointers */
 90:typedef struct External_methods_struct * external_methods_ptr;
 91:typedef struct Compress_methods_struct * compress_methods_ptr;
 92:typedef struct Decompress_methods_struct * decompress_methods_ptr;
 93:
 94:
 95:/* Data structures for images containing either samples or coefficients. */
 96:/* Note that the topmost (leftmost) index is always color component. */
 97:/* On 80x86 machines, the image arrays are too big for near pointers, */
 98:/* but the pointer arrays can fit in near memory. */
 99:
100:typedef JSAMPLE FAR *JSAMPROW;  /* ptr to one image row of pixel samples. */
101:typedef JSAMPROW *JSAMPARRAY;   /* ptr to some rows (a 2-D sample array) */
102:typedef JSAMPARRAY *JSAMPIMAGE; /* a 3-D sample array: top index is color */
103:
104:
105:#define DCTSIZE     8   /* The basic DCT block is 8x8 samples */
106:#define DCTSIZE2    64  /* DCTSIZE squared; # of elements in a block */
107:
108:typedef JCOEF JBLOCK[DCTSIZE2]; /* one block of coefficients */
109:typedef JBLOCK FAR *JBLOCKROW;  /* pointer to one row of coefficient blocks */
110:typedef JBLOCKROW *JBLOCKARRAY;     /* a 2-D array of coefficient blocks */
111:typedef JBLOCKARRAY *JBLOCKIMAGE;   /* a 3-D array of coefficient blocks */
112:
113:typedef JCOEF FAR *JCOEFPTR;    /* useful in a couple of places */
114:
115:
116:/* The input and output data of the DCT transform subroutines are of
117: * the following type, which need not be the same as JCOEF.
118: * For example, on a machine with fast floating point, it might make sense
119: * to recode the DCT routines to use floating point; then DCTELEM would be
120: * 'float' or 'double'.
121: */
122:
123:typedef JCOEF DCTELEM;
124:typedef DCTELEM DCTBLOCK[DCTSIZE2];
125:
126:
127:/* Types for JPEG compression parameters and working tables. */
128:
129:
130:typedef enum {          /* defines known color spaces */
131:    CS_UNKNOWN,     /* error/unspecified */
132:    CS_GRAYSCALE,       /* monochrome (only 1 component) */
133:    CS_RGB,         /* red/green/blue */
134:    CS_YCbCr,       /* Y/Cb/Cr (also known as YUV) */
135:    CS_YIQ,         /* Y/I/Q */
136:    CS_CMYK         /* C/M/Y/K */
```

```
137:} COLOR_SPACE;
138:
139:
140:typedef struct {        /* Basic info about one component */
141:  /* These values are fixed over the whole image */
142:  /* For compression, they must be supplied by the user interface; */
143:  /* for decompression, they are read from the SOF marker. */
144:    short component_id; /* identifier for this component (0..255) */
145:    short component_index;  /* its index in SOF or cinfo->comp_info[] */
146:    short h_samp_factor;    /* horizontal sampling factor (1..4) */
147:    short v_samp_factor;    /* vertical sampling factor (1..4) */
148:    short quant_tbl_no; /* quantization table selector (0..3) */
149:  /* These values may vary between scans */
150:  /* For compression, they must be supplied by the user interface; */
151:  /* for decompression, they are read from the SOS marker. */
152:    short dc_tbl_no;    /* DC entropy table selector (0..3) */
153:    short ac_tbl_no;    /* AC entropy table selector (0..3) */
154:  /* These values are computed during compression or decompression startup */
155:    long true_comp_width;   /* component's image width in samples */
156:    long true_comp_height;  /* component's image height in samples */
157:    /* the above are the logical dimensions of the downsampled image */
158:  /* These values are computed before starting a scan of the component */
159:    short MCU_width;    /* number of blocks per MCU, horizontally */
160:    short MCU_height;   /* number of blocks per MCU, vertically */
161:    short MCU_blocks;   /* MCU_width * MCU_height */
162:    long downsampled_width; /* image width in samples, after expansion */
163:    long downsampled_height; /* image height in samples, after expansion */
164:    /* the above are the true_comp_xxx values rounded up to multiples of */
165:    /* the MCU dimensions; these are the working dimensions of the array */
166:    /* as it is passed through the DCT or IDCT step.  NOTE: these values */
167:    /* differ depending on whether the component is interleaved or not!! */
168:  /* This flag is used only for decompression.  In cases where some of the */
169:  /* components will be ignored (eg grayscale output from YCbCr image), */
170:  /* we can skip IDCT etc. computations for the unused components. */
171:    boolean component_needed; /* do we need the value of this component? */
172:} jpeg_component_info;
173:
174:
175:/* DCT coefficient quantization tables.
176: * For 8-bit precision, 'INT16' should be good enough for quantization values;
177: * for more precision, we go for the full 16 bits.  'INT16' provides a useful
178: * speedup on many machines (multiplication & division of JCOEFs by
179: * quantization values is a significant chunk of the runtime).
180: * Note: the values in a QUANT_TBL are always given in zigzag order.
181: */
182:#ifdef EIGHT_BIT_SAMPLES
183:typedef INT16 QUANT_VAL;    /* element of a quantization table */
184:#else
185:typedef UINT16 QUANT_VAL;   /* element of a quantization table */
186:#endif
187:typedef QUANT_VAL QUANT_TBL[DCTSIZE2];  /* A quantization table */
188:typedef QUANT_VAL * QUANT_TBL_PTR;  /* pointer to same */
189:
190:
191:/* Huffman coding tables.
192: */
193:
194:#define HUFF_LOOKAHEAD  8   /* # of bits of lookahead */
195:
196:typedef struct {
197:  /* These two fields directly represent the contents of a JPEG DHT marker */
198:    UINT8 bits[17];     /* bits[k] = # of symbols with codes of */
199:                /* length k bits; bits[0] is unused */
200:    UINT8 huffval[256]; /* The symbols, in order of incr code length */
201:  /* This field is used only during compression.  It's initialized FALSE when
202:   * the table is created, and set TRUE when it's been output to the file.
203:   */
204:    boolean sent_table; /* TRUE when table has been output */
```

```
205:  /* The remaining fields are computed from the above to allow more efficient
206:   * coding and decoding.  These fields should be considered private to the
207:   * Huffman compression & decompression modules.  We use a union since only
208:   * one set of fields is needed at a time.
209:   */
210:    union {
211:      struct {      /* encoding tables: */
212:        UINT16 ehufco[256]; /* code for each symbol */
213:        char ehufsi[256];   /* length of code for each symbol */
214:      } enc;
215:      struct {      /* decoding tables: */
216:        /* Basic tables: (element [0] of each array is unused) */
217:        INT32 mincode[17];  /* smallest code of length k */
218:        INT32 maxcode[18];  /* largest code of length k (-1 if none) */
219:        /* (maxcode[17] is a sentinel to ensure huff_DECODE terminates) */
220:        int valptr[17]; /* huffval[] index of 1st symbol of length k */
221:        /* Lookahead tables: indexed by the next HUFF_LOOKAHEAD bits of
222:         * the input data stream.  If the next Huffman code is no more
223:         * than HUFF_LOOKAHEAD bits long, we can obtain its length and
224:         * the corresponding symbol directly from these tables.
225:         */
226:        int look_nbits[1<<HUFF_LOOKAHEAD]; /* # bits, or 0 if too long */
227:        UINT8 look_sym[1<<HUFF_LOOKAHEAD]; /* symbol, or unused */
228:      } dec;
229:    } priv;
230:} HUFF_TBL;
231:
232:
233:#define NUM_QUANT_TBLS      4   /* quantization tables are numbered 0..3 */
234:#define NUM_HUFF_TBLS       4   /* Huffman tables are numbered 0..3 */
235:#define NUM_ARITH_TBLS      16  /* arith-coding tables are numbered 0..15 */
236:#define MAX_COMPS_IN_SCAN   4   /* JPEG limit on # of components in one scan */
237:#define MAX_SAMP_FACTOR     4   /* JPEG limit on sampling factors */
238:#define MAX_BLOCKS_IN_MCU   10  /* JPEG limit on # of blocks in an MCU */
239:
240:
241:/* Working data for compression */
242:
243:struct Compress_info_struct {
244:/*
245: * All of these fields shall be established by the user interface before
246: * calling jpeg_compress, or by the input_init or c_ui_method_selection
247: * methods.
248: * Most parameters can be set to reasonable defaults by j_c_defaults.
249: * Note that the UI must supply the storage for the main methods struct,
250: * though it sets only a few of the methods there.
251: */
252:    compress_methods_ptr methods; /* Points to list of methods to use */
253:
254:    external_methods_ptr emethods; /* Points to list of methods to use */
255:
256:    IFILEREF input_file;    /* tells input routines where to read image */
257:    JFILEREF output_file;   /* tells output routines where to write JPEG */
258:
259:    long image_width;   /* input image width */
260:    long image_height;  /* input image height */
261:    short input_components; /* # of color components in input image */
262:
263:    short data_precision;   /* bits of precision in image data */
264:
265:    COLOR_SPACE in_color_space; /* colorspace of input file */
266:    COLOR_SPACE jpeg_color_space; /* colorspace of JPEG file */
267:
268:    double input_gamma; /* image gamma of input file */
269:
270:    boolean write_JFIF_header; /* should a JFIF marker be written? */
271:    /* These three values are not used by the JPEG code, only copied */
272:    /* into the JFIF APP0 marker.  density_unit can be 0 for unknown, */
```

```
273:    /* 1 for dots/inch, or 2 for dots/cm.  Note that the pixel aspect */
274:    /* ratio is defined by X_density/Y_density even when density_unit=0. */
275:    UINT8 density_unit; /* JFIF code for pixel size units */
276:    UINT16 X_density;   /* Horizontal pixel density */
277:    UINT16 Y_density;   /* Vertical pixel density */
278:
279:    char * comment_text;    /* Text for COM block, or NULL for no COM */
280:    /* note: JPEG library will not free() the comment string, */
281:    /* unless you allocate it via alloc_small(). */
282:
283:    short num_components;   /* # of color components in JPEG image */
284:    jpeg_component_info * comp_info;
285:    /* comp_info[i] describes component that appears i'th in SOF */
286:
287:    QUANT_TBL_PTR quant_tbl_ptrs[NUM_QUANT_TBLS];
288:    /* ptrs to coefficient quantization tables, or NULL if not defined */
289:
290:    HUFF_TBL * dc_huff_tbl_ptrs[NUM_HUFF_TBLS];
291:    HUFF_TBL * ac_huff_tbl_ptrs[NUM_HUFF_TBLS];
292:    /* ptrs to Huffman coding tables, or NULL if not defined */
293:
294:    UINT8 arith_dc_L[NUM_ARITH_TBLS]; /* L values for DC arithmetic-coding tables */
295:    UINT8 arith_dc_U[NUM_ARITH_TBLS]; /* U values for DC arithmetic-coding tables */
296:    UINT8 arith_ac_K[NUM_ARITH_TBLS]; /* Kx values for AC arithmetic-coding tables */
297:
298:    boolean arith_code; /* TRUE=arithmetic coding, FALSE=Huffman */
299:    boolean interleave; /* TRUE=interleaved output, FALSE=not */
300:    boolean optimize_coding; /* TRUE=optimize entropy encoding parms */
301:    boolean CCIR601_sampling; /* TRUE=first samples are cosited */
302:    int smoothing_factor;   /* 1..100, or 0 for no input smoothing */
303:
304:    /* The restart interval can be specified in absolute MCUs by setting
305:     * restart_interval, or in MCU rows by setting restart_in_rows
306:     * (in which case the correct restart_interval will be figured
307:     * for each scan).
308:     */
309:    UINT16 restart_interval;/* MCUs per restart interval, or 0 for no restart */
310:    int restart_in_rows;    /* if > 0, MCU rows per restart interval */
311:
312:/*
313: * These fields are computed during jpeg_compress startup
314: */
315:    short max_h_samp_factor; /* largest h_samp_factor */
316:    short max_v_samp_factor; /* largest v_samp_factor */
317:
318:/*
319: * These fields may be useful for progress monitoring
320: */
321:
322:    int total_passes;   /* number of passes expected */
323:    int completed_passes;   /* number of passes completed so far */
324:
325:/*
326: * These fields are valid during any one scan
327: */
328:    short comps_in_scan;    /* # of JPEG components output this time */
329:    jpeg_component_info * cur_comp_info[MAX_COMPS_IN_SCAN];
330:    /* *cur_comp_info[i] describes component that appears i'th in SOS */
331:
332:    long MCUs_per_row;  /* # of MCUs across the image */
333:    long MCU_rows_in_scan;  /* # of MCU rows in the image */
334:
335:    short blocks_in_MCU;    /* # of DCT blocks per MCU */
336:    short MCU_membership[MAX_BLOCKS_IN_MCU];
337:    /* MCU_membership[i] is index in cur_comp_info of component owning */
338:    /* i'th block in an MCU */
339:
340:    /* these fields are private data for the entropy encoder */
```

```
341:    JCOEF last_dc_val[MAX_COMPS_IN_SCAN]; /* last DC coef for each comp */
342:    JCOEF last_dc_diff[MAX_COMPS_IN_SCAN]; /* last DC diff for each comp */
343:    UINT16 restarts_to_go;  /* MCUs left in this restart interval */
344:    short next_restart_num; /* # of next RSTn marker (0..7) */
345:};
346:
347:typedef struct Compress_info_struct * compress_info_ptr;
348:
349:
350:/* Working data for decompression */
351:
352:struct Decompress_info_struct {
353:/*
354: * These fields shall be established by the user interface before
355: * calling jpeg_decompress.
356: * Most parameters can be set to reasonable defaults by j_d_defaults.
357: * Note that the UI must supply the storage for the main methods struct,
358: * though it sets only a few of the methods there.
359: */
360:    decompress_methods_ptr methods; /* Points to list of methods to use */
361:
362:    external_methods_ptr emethods; /* Points to list of methods to use */
363:
364:    JFILEREF input_file;    /* tells input routines where to read JPEG */
365:    IFILEREF output_file;   /* tells output routines where to write image */
366:
367:    /* these can be set at d_ui_method_selection time: */
368:
369:    COLOR_SPACE out_color_space; /* colorspace of output */
370:
371:    double output_gamma;    /* image gamma wanted in output */
372:
373:    boolean quantize_colors; /* T if output is a colormapped format */
374:    /* the following are ignored if not quantize_colors: */
375:    boolean two_pass_quantize;  /* use two-pass color quantization? */
376:    boolean use_dithering;      /* want color dithering? */
377:    int desired_number_of_colors;   /* max number of colors to use */
378:
379:    boolean do_block_smoothing; /* T = apply cross-block smoothing */
380:    boolean do_pixel_smoothing; /* T = apply post-upsampling smoothing */
381:
382:/*
383: * These fields are used for efficient buffering of data between read_jpeg_data
384: * and the entropy decoding object.  By using a shared buffer, we avoid copying
385: * data and eliminate the need for an "unget" operation at the end of a scan.
386: * The actual source of the data is known only to read_jpeg_data; see the
387: * JGETC macro, below.
388: * Note: the user interface is expected to allocate the input_buffer and
389: * initialize bytes_in_buffer to 0.  Also, for JFIF/raw-JPEG input, the UI
390: * actually supplies the read_jpeg_data method.  This is all handled by
391: * j_d_defaults in a typical implementation.
392: */
393:    char * input_buffer;    /* start of buffer (private to input code) */
394:    char * next_input_byte; /* => next byte to read from buffer */
395:    int bytes_in_buffer;    /* # of bytes remaining in buffer */
396:
397:/*
398: * These fields are set by read_file_header or read_scan_header
399: */
400:    long image_width;   /* overall image width */
401:    long image_height;  /* overall image height */
402:
403:    short data_precision;   /* bits of precision in image data */
404:
405:    COLOR_SPACE jpeg_color_space; /* colorspace of JPEG file */
406:
407:        /* These three values are not used by the JPEG code, merely copied */
408:    /* from the JFIF APP0 marker (if any). */
```

```
409:    UINT8 density_unit; /* JFIF code for pixel size units */
410:    UINT16 X_density;   /* Horizontal pixel density */
411:    UINT16 Y_density;   /* Vertical pixel density */
412:
413:    short num_components;   /* # of color components in JPEG image */
414:    jpeg_component_info * comp_info;
415:    /* comp_info[i] describes component that appears i'th in SOF */
416:
417:    QUANT_TBL_PTR quant_tbl_ptrs[NUM_QUANT_TBLS];
418:    /* ptrs to coefficient quantization tables, or NULL if not defined */
419:
420:    HUFF_TBL * dc_huff_tbl_ptrs[NUM_HUFF_TBLS];
421:    HUFF_TBL * ac_huff_tbl_ptrs[NUM_HUFF_TBLS];
422:    /* ptrs to Huffman coding tables, or NULL if not defined */
423:
424:    UINT8 arith_dc_L[NUM_ARITH_TBLS]; /* L values for DC arith-coding tables */
425:    UINT8 arith_dc_U[NUM_ARITH_TBLS]; /* U values for DC arith-coding tables */
426:    UINT8 arith_ac_K[NUM_ARITH_TBLS]; /* Kx values for AC arith-coding tables */
427:
428:    boolean arith_code; /* TRUE=arithmetic coding, FALSE=Huffman */
429:    boolean CCIR601_sampling; /* TRUE=first samples are cosited */
430:
431:    UINT16 restart_interval;/* MCUs per restart interval, or 0 for no restart */
432:
433:/*
434: * These fields are computed during jpeg_decompress startup
435: */
436:    short max_h_samp_factor; /* largest h_samp_factor */
437:    short max_v_samp_factor; /* largest v_samp_factor */
438:
439:    short color_out_comps;  /* # of color components output by color_convert */
440:                /* (need not match num_components) */
441:    short final_out_comps;  /* # of color components sent to put_pixel_rows */
442:    /* (1 when quantizing colors, else same as color_out_comps) */
443:
444:    JSAMPLE * sample_range_limit; /* table for fast range-limiting */
445:
446:/*
447: * When quantizing colors, the color quantizer leaves a pointer to the output
448: * colormap in these fields.  The colormap is valid from the time put_color_map
449: * is called (must be before any put_pixel_rows calls) until shutdown (more
450: * specifically, until free_all is called to release memory).
451: */
452:    int actual_number_of_colors; /* actual number of entries */
453:    JSAMPARRAY colormap;    /* NULL if not valid */
454:    /* map has color_out_comps rows * actual_number_of_colors columns */
455:
456:/*
457: * These fields may be useful for progress monitoring
458: */
459:
460:    int total_passes;   /* number of passes expected */
461:    int completed_passes;   /* number of passes completed so far */
462:
463:/*
464: * These fields are valid during any one scan
465: */
466:    short comps_in_scan;    /* # of JPEG components input this time */
467:    jpeg_component_info * cur_comp_info[MAX_COMPS_IN_SCAN];
468:    /* *cur_comp_info[i] describes component that appears i'th in SOS */
469:
470:    long MCUs_per_row;  /* # of MCUs across the image */
471:    long MCU_rows_in_scan;  /* # of MCU rows in the image */
472:
473:    short blocks_in_MCU;    /* # of DCT blocks per MCU */
474:    short MCU_membership[MAX_BLOCKS_IN_MCU];
475:    /* MCU_membership[i] is index in cur_comp_info of component owning */
476:    /* i'th block in an MCU */
```

```
477:
478:    /* these fields are private data for the entropy encoder */
479:    JCOEF last_dc_val[MAX_COMPS_IN_SCAN]; /* last DC coef for each comp */
480:    JCOEF last_dc_diff[MAX_COMPS_IN_SCAN]; /* last DC diff for each comp */
481:    UINT16 restarts_to_go;  /* MCUs left in this restart interval */
482:    short next_restart_num; /* # of next RSTn marker (0..7) */
483:};
484:
485:typedef struct Decompress_info_struct * decompress_info_ptr;
486:
487:
488:/* Macros for reading data from the decompression input buffer */
489:
490:#ifdef CHAR_IS_UNSIGNED
491:#define JGETC(cinfo)    ( --(cinfo)->bytes_in_buffer < 0 ? ¥
492:             (*(cinfo)->methods->read_jpeg_data) (cinfo) : ¥
493:             (int) (*(cinfo)->next_input_byte++) )
494:#else
495:#define JGETC(cinfo)    ( --(cinfo)->bytes_in_buffer < 0 ? ¥
496:             (*(cinfo)->methods->read_jpeg_data) (cinfo) : ¥
497:             (int) (*(cinfo)->next_input_byte++) & 0xFF )
498:#endif
499:
500:#define JUNGETC(ch,cinfo)  ((cinfo)->bytes_in_buffer++, ¥
501:                *(--((cinfo)->next_input_byte)) = (char) (ch))
502:
503:#define MIN_UNGET   4   /* may always do at least 4 JUNGETCs */
504:
505:
506:/* A virtual image has a control block whose contents are private to the
507: * memory manager module (and may differ between managers).  The rest of the
508: * code only refers to virtual images by these pointer types, and never
509: * dereferences the pointer.
510: */
511:
512:typedef struct big_sarray_control * big_sarray_ptr;
513:typedef struct big_barray_control * big_barray_ptr;
514:
515:/* Although a real ANSI C compiler can deal perfectly well with pointers to
516: * unspecified structures (see "incomplete types" in the spec), a few pre-ANSI
517: * and pseudo-ANSI compilers get confused.  To keep one of these bozos happy,
518: * add -DINCOMPLETE_TYPES_BROKEN to CFLAGS in your Makefile.  Then we will
519: * pseudo-define the structs as containing a single "dummy" field.
520: * The memory managers #define AM_MEMORY_MANAGER before including this file,
521: * so that they can make their own definitions of the structs.
522: */
523:
524:#ifdef INCOMPLETE_TYPES_BROKEN
525:#ifndef AM_MEMORY_MANAGER
526:struct big_sarray_control { long dummy; };
527:struct big_barray_control { long dummy; };
528:#endif
529:#endif
530:
531:
532:/* Method types that need typedefs */
533:
534:typedef METHOD(void, MCU_output_method_ptr, (compress_info_ptr cinfo,
535:                        JBLOCK *MCU_data));
536:typedef METHOD(void, MCU_output_caller_ptr, (compress_info_ptr cinfo,
537:                        MCU_output_method_ptr output_method));
538:typedef METHOD(void, downsample_ptr, (compress_info_ptr cinfo,
539:                      int which_component,
540:                      long input_cols, int input_rows,
541:                      long output_cols, int output_rows,
542:                      JSAMPARRAY above,
543:                      JSAMPARRAY input_data,
544:                      JSAMPARRAY below,
```

```
545:                     JSAMPARRAY output_data));
546:typedef METHOD(void, upsample_ptr, (decompress_info_ptr cinfo,
547:                    int which_component,
548:                    long input_cols, int input_rows,
549:                    long output_cols, int output_rows,
550:                    JSAMPARRAY above,
551:                    JSAMPARRAY input_data,
552:                    JSAMPARRAY below,
553:                    JSAMPARRAY output_data));
554:typedef METHOD(void, quantize_method_ptr, (decompress_info_ptr cinfo,
555:                      int num_rows,
556:                      JSAMPIMAGE input_data,
557:                      JSAMPARRAY output_workspace));
558:typedef METHOD(void, quantize_caller_ptr, (decompress_info_ptr cinfo,
559:                      quantize_method_ptr quantize_method));
560:
561:
562:/* These structs contain function pointers for the various JPEG methods. */
563:
564:/* Routines to be provided by the surrounding application, rather than the
565: * portable JPEG code proper.  These are the same for compression and
566: * decompression.
567: */
568:
569:struct External_methods_struct {
570:    /* User interface: error exit and trace message routines */
571:    /* NOTE: the string msgtext parameters will eventually be replaced
572:     * by an enumerated-type code so that non-English error messages
573:     * can be substituted easily.  This will not be done until all the
574:     * code is in place, so that we know what messages are needed.
575:     */
576:    METHOD(void, error_exit, (const char *msgtext));
577:    METHOD(void, trace_message, (const char *msgtext));
578:
579:    /* Working data for error/trace facility */
580:    /* See macros below for the usage of these variables */
581:    int trace_level;    /* level of detail of tracing messages */
582:    /* Use level 0 for important warning messages (nonfatal errors) */
583:    /* Use levels 1, 2, 3 for successively more detailed trace options */
584:
585:    /* For recoverable corrupt-data errors, we emit a warning message and
586:     * keep going.  A surrounding application can check for bad data by
587:     * seeing if num_warnings is nonzero at the end of processing.
588:     */
589:    long num_warnings;  /* number of corrupt-data warnings */
590:    int first_warning_level; /* trace level for first warning */
591:    int more_warning_level; /* trace level for subsequent warnings */
592:
593:    int message_parm[8];    /* store numeric parms for messages here */
594:
595:    /* Memory management */
596:    /* NB: alloc routines never return NULL. They exit to */
597:    /* error_exit if not successful. */
598:    METHOD(void *, alloc_small, (size_t sizeofobject));
599:    METHOD(void, free_small, (void *ptr));
600:    METHOD(void FAR *, alloc_medium, (size_t sizeofobject));
601:    METHOD(void, free_medium, (void FAR *ptr));
602:    METHOD(JSAMPARRAY, alloc_small_sarray, (long samplesperrow,
603:                         long numrows));
604:    METHOD(void, free_small_sarray, (JSAMPARRAY ptr));
605:    METHOD(JBLOCKARRAY, alloc_small_barray, (long blocksperrow,
606:                          long numrows));
607:    METHOD(void, free_small_barray, (JBLOCKARRAY ptr));
608:    METHOD(big_sarray_ptr, request_big_sarray, (long samplesperrow,
609:                              long numrows,
610:                              long unitheight));
611:    METHOD(big_barray_ptr, request_big_barray, (long blocksperrow,
612:                              long numrows,
```

```
613:                            long unitheight));
614:    METHOD(void, alloc_big_arrays, (long extra_small_samples,
615:                    long extra_small_blocks,
616:                    long extra_medium_space));
617:    METHOD(JSAMPARRAY, access_big_sarray, (big_sarray_ptr ptr,
618:                            long start_row,
619:                            boolean writable));
620:    METHOD(JBLOCKARRAY, access_big_barray, (big_barray_ptr ptr,
621:                            long start_row,
622:                            boolean writable));
623:    METHOD(void, free_big_sarray, (big_sarray_ptr ptr));
624:    METHOD(void, free_big_barray, (big_barray_ptr ptr));
625:    METHOD(void, free_all, (void));
626:
627:    long max_memory_to_use; /* maximum amount of memory to use */
628:};
629:
630:/* Macros to simplify using the error and trace message stuff */
631:/* The first parameter is generally cinfo->emethods */
632:
633:/* Fatal errors (print message and exit) */
634:#define ERREXIT(emeth,msg)      ((*(emeth)->error_exit) (msg))
635:#define ERREXIT1(emeth,msg,p1)      ((emeth)->message_parm[0] = (p1), ¥
636:                        (*(emeth)->error_exit) (msg))
637:#define ERREXIT2(emeth,msg,p1,p2)   ((emeth)->message_parm[0] = (p1), ¥
638:                        (emeth)->message_parm[1] = (p2), ¥
639:                        (*(emeth)->error_exit) (msg))
640:#define ERREXIT3(emeth,msg,p1,p2,p3)    ((emeth)->message_parm[0] = (p1), ¥
641:                        (emeth)->message_parm[1] = (p2), ¥
642:                        (emeth)->message_parm[2] = (p3), ¥
643:                        (*(emeth)->error_exit) (msg))
644:#define ERREXIT4(emeth,msg,p1,p2,p3,p4) ((emeth)->message_parm[0] = (p1), ¥
645:                        (emeth)->message_parm[1] = (p2), ¥
646:                        (emeth)->message_parm[2] = (p3), ¥
647:                        (emeth)->message_parm[3] = (p4), ¥
648:                        (*(emeth)->error_exit) (msg))
649:
650:#define MAKESTMT(stuff)     do { stuff } while (0)
651:
652:/* Nonfatal errors (we'll keep going, but the data is probably corrupt) */
653:/* Note that warning count is incremented as a side-effect! */
654:#define WARNMS(emeth,msg)    ¥
655:  MAKESTMT( if ((emeth)->trace_level >= ((emeth)->num_warnings++ ? ¥
656:        (emeth)->more_warning_level : (emeth)->first_warning_level)) { ¥
657:        (*(emeth)->trace_message) (msg); } )
658:#define WARNMS1(emeth,msg,p1)    ¥
659:  MAKESTMT( if ((emeth)->trace_level >= ((emeth)->num_warnings++ ? ¥
660:        (emeth)->more_warning_level : (emeth)->first_warning_level)) { ¥
661:        (emeth)->message_parm[0] = (p1); ¥
662:        (*(emeth)->trace_message) (msg); } )
663:#define WARNMS2(emeth,msg,p1,p2)    ¥
664:  MAKESTMT( if ((emeth)->trace_level >= ((emeth)->num_warnings++ ? ¥
665:        (emeth)->more_warning_level : (emeth)->first_warning_level)) { ¥
666:        (emeth)->message_parm[0] = (p1); ¥
667:        (emeth)->message_parm[1] = (p2); ¥
668:        (*(emeth)->trace_message) (msg); } )
669:
670:/* Informational/debugging messages */
671:#define TRACEMS(emeth,lvl,msg)    ¥
672:  MAKESTMT( if ((emeth)->trace_level >= (lvl)) { ¥
673:        (*(emeth)->trace_message) (msg); } )
674:#define TRACEMS1(emeth,lvl,msg,p1)    ¥
675:  MAKESTMT( if ((emeth)->trace_level >= (lvl)) { ¥
676:        (emeth)->message_parm[0] = (p1); ¥
677:        (*(emeth)->trace_message) (msg); } )
678:#define TRACEMS2(emeth,lvl,msg,p1,p2)    ¥
679:  MAKESTMT( if ((emeth)->trace_level >= (lvl)) { ¥
680:        (emeth)->message_parm[0] = (p1); ¥
```

```
681:        (emeth)->message_parm[1] = (p2); ¥
682:        (*(emeth)->trace_message) (msg); } )
683:#define TRACEMS3(emeth,lvl,msg,p1,p2,p3)    ¥
684:  MAKESTMT( if ((emeth)->trace_level >= (lvl)) { ¥
685:        int * _mp = (emeth)->message_parm; ¥
686:        *_mp++ = (p1); *_mp++ = (p2); *_mp = (p3); ¥
687:        (*(emeth)->trace_message) (msg); } )
688:#define TRACEMS4(emeth,lvl,msg,p1,p2,p3,p4)    ¥
689:  MAKESTMT( if ((emeth)->trace_level >= (lvl)) { ¥
690:        int * _mp = (emeth)->message_parm; ¥
691:        *_mp++ = (p1); *_mp++ = (p2); *_mp++ = (p3); *_mp = (p4); ¥
692:        (*(emeth)->trace_message) (msg); } )
693:#define TRACEMS8(emeth,lvl,msg,p1,p2,p3,p4,p5,p6,p7,p8)    ¥
694:  MAKESTMT( if ((emeth)->trace_level >= (lvl)) { ¥
695:        int * _mp = (emeth)->message_parm; ¥
696:        *_mp++ = (p1); *_mp++ = (p2); *_mp++ = (p3); *_mp++ = (p4); ¥
697:        *_mp++ = (p5); *_mp++ = (p6); *_mp++ = (p7); *_mp = (p8); ¥
698:        (*(emeth)->trace_message) (msg); } )
699:
700:
701:/* Methods used during JPEG compression. */
702:
703:struct Compress_methods_struct {
704:    /* Hook for user interface to get control after input_init */
705:    METHOD(void, c_ui_method_selection, (compress_info_ptr cinfo));
706:    /* Hook for user interface to do progress monitoring */
707:    METHOD(void, progress_monitor, (compress_info_ptr cinfo,
708:                    long loopcounter, long looplimit));
709:    /* Input image reading & conversion to standard form */
710:    METHOD(void, input_init, (compress_info_ptr cinfo));
711:    METHOD(void, get_input_row, (compress_info_ptr cinfo,
712:                     JSAMPARRAY pixel_row));
713:    METHOD(void, input_term, (compress_info_ptr cinfo));
714:    /* Color space and gamma conversion */
715:    METHOD(void, colorin_init, (compress_info_ptr cinfo));
716:    METHOD(void, get_sample_rows, (compress_info_ptr cinfo,
717:                       int rows_to_read,
718:                       JSAMPIMAGE image_data));
719:    METHOD(void, colorin_term, (compress_info_ptr cinfo));
720:    /* Expand picture data at edges */
721:    METHOD(void, edge_expand, (compress_info_ptr cinfo,
722:                   long input_cols, int input_rows,
723:                   long output_cols, int output_rows,
724:                   JSAMPIMAGE image_data));
725:    /* Downsample pixel values of a single component */
726:    /* There can be a different downsample method for each component */
727:    METHOD(void, downsample_init, (compress_info_ptr cinfo));
728:    downsample_ptr downsample[MAX_COMPS_IN_SCAN];
729:    METHOD(void, downsample_term, (compress_info_ptr cinfo));
730:    /* Extract samples in MCU order, process & hand off to output_method */
731:    /* The input is always exactly N MCU rows worth of data */
732:    METHOD(void, extract_init, (compress_info_ptr cinfo));
733:    METHOD(void, extract_MCUs, (compress_info_ptr cinfo,
734:                    JSAMPIMAGE image_data,
735:                    int num_mcu_rows,
736:                    MCU_output_method_ptr output_method));
737:    METHOD(void, extract_term, (compress_info_ptr cinfo));
738:    /* Entropy encoding parameter optimization */
739:    METHOD(void, entropy_optimize, (compress_info_ptr cinfo,
740:                    MCU_output_caller_ptr source_method));
741:    /* Entropy encoding */
742:    METHOD(void, entropy_encode_init, (compress_info_ptr cinfo));
743:    METHOD(void, entropy_encode, (compress_info_ptr cinfo,
744:                      JBLOCK *MCU_data));
745:    METHOD(void, entropy_encode_term, (compress_info_ptr cinfo));
746:    /* JPEG file header construction */
747:    METHOD(void, write_file_header, (compress_info_ptr cinfo));
748:    METHOD(void, write_scan_header, (compress_info_ptr cinfo));
```

```
749:    METHOD(void, write_jpeg_data, (compress_info_ptr cinfo,
750:                       char *dataptr,
751:                       int datacount));
752:    METHOD(void, write_scan_trailer, (compress_info_ptr cinfo));
753:    METHOD(void, write_file_trailer, (compress_info_ptr cinfo));
754:    /* Pipeline control */
755:    METHOD(void, c_pipeline_controller, (compress_info_ptr cinfo));
756:    METHOD(void, entropy_output, (compress_info_ptr cinfo,
757:                      char *dataptr,
758:                      int datacount));
759:    /* Overall control */
760:    METHOD(void, c_per_scan_method_selection, (compress_info_ptr cinfo));
761:};
762:
763:/* Methods used during JPEG decompression. */
764:
765:struct Decompress_methods_struct {
766:    /* Hook for user interface to get control after reading file header */
767:    METHOD(void, d_ui_method_selection, (decompress_info_ptr cinfo));
768:    /* Hook for user interface to process comment blocks */
769:    METHOD(void, process_comment, (decompress_info_ptr cinfo,
770:                       long comment_length));
771:    /* Hook for user interface to do progress monitoring */
772:    METHOD(void, progress_monitor, (decompress_info_ptr cinfo,
773:                     long loopcounter, long looplimit));
774:    /* JPEG file scanning */
775:    METHOD(void, read_file_header, (decompress_info_ptr cinfo));
776:    METHOD(boolean, read_scan_header, (decompress_info_ptr cinfo));
777:    METHOD(int, read_jpeg_data, (decompress_info_ptr cinfo));
778:    METHOD(void, resync_to_restart, (decompress_info_ptr cinfo,
779:                      int marker));
780:    METHOD(void, read_scan_trailer, (decompress_info_ptr cinfo));
781:    METHOD(void, read_file_trailer, (decompress_info_ptr cinfo));
782:    /* Entropy decoding */
783:    METHOD(void, entropy_decode_init, (decompress_info_ptr cinfo));
784:    METHOD(void, entropy_decode, (decompress_info_ptr cinfo,
785:                      JBLOCKROW *MCU_data));
786:    METHOD(void, entropy_decode_term, (decompress_info_ptr cinfo));
787:    /* MCU disassembly: fetch MCUs from entropy_decode, build coef array */
788:    /* The reverse_DCT step is in the same module for symmetry reasons */
789:    METHOD(void, disassemble_init, (decompress_info_ptr cinfo));
790:    METHOD(void, disassemble_MCU, (decompress_info_ptr cinfo,
791:                      JBLOCKIMAGE image_data));
792:    METHOD(void, reverse_DCT, (decompress_info_ptr cinfo,
793:                   JBLOCKIMAGE coeff_data,
794:                   JSAMPIMAGE output_data, int start_row));
795:    METHOD(void, disassemble_term, (decompress_info_ptr cinfo));
796:    /* Cross-block smoothing */
797:    METHOD(void, smooth_coefficients, (decompress_info_ptr cinfo,
798:                       jpeg_component_info *compptr,
799:                       JBLOCKROW above,
800:                       JBLOCKROW currow,
801:                       JBLOCKROW below,
802:                       JBLOCKROW output));
803:    /* Upsample pixel values of a single component */
804:    /* There can be a different upsample method for each component */
805:    METHOD(void, upsample_init, (decompress_info_ptr cinfo));
806:    upsample_ptr upsample[MAX_COMPS_IN_SCAN];
807:    METHOD(void, upsample_term, (decompress_info_ptr cinfo));
808:    /* Color space and gamma conversion */
809:    METHOD(void, colorout_init, (decompress_info_ptr cinfo));
810:    METHOD(void, color_convert, (decompress_info_ptr cinfo,
811:                      int num_rows, long num_cols,
812:                      JSAMPIMAGE input_data,
813:                      JSAMPIMAGE output_data));
814:    METHOD(void, colorout_term, (decompress_info_ptr cinfo));
815:    /* Color quantization */
816:    METHOD(void, color_quant_init, (decompress_info_ptr cinfo));
```

```
817:    METHOD(void, color_quantize, (decompress_info_ptr cinfo,
818:                        int num_rows,
819:                        JSAMPIMAGE input_data,
820:                        JSAMPARRAY output_data));
821:    METHOD(void, color_quant_prescan, (decompress_info_ptr cinfo,
822:                         int num_rows,
823:                         JSAMPIMAGE image_data,
824:                         JSAMPARRAY workspace));
825:    METHOD(void, color_quant_doit, (decompress_info_ptr cinfo,
826:                       quantize_caller_ptr source_method));
827:    METHOD(void, color_quant_term, (decompress_info_ptr cinfo));
828:    /* Output image writing */
829:    METHOD(void, output_init, (decompress_info_ptr cinfo));
830:    METHOD(void, put_color_map, (decompress_info_ptr cinfo,
831:                        int num_colors, JSAMPARRAY colormap));
832:    METHOD(void, put_pixel_rows, (decompress_info_ptr cinfo,
833:                        int num_rows,
834:                        JSAMPIMAGE pixel_data));
835:    METHOD(void, output_term, (decompress_info_ptr cinfo));
836:    /* Pipeline control */
837:    METHOD(void, d_pipeline_controller, (decompress_info_ptr cinfo));
838:    /* Overall control */
839:    METHOD(void, d_per_scan_method_selection, (decompress_info_ptr cinfo));
840:};
841:
842:
843:/* External declarations for routines that aren't called via method ptrs. */
844:/* Note: use "j" as first char of names to minimize namespace pollution. */
845:/* The PP macro hides prototype parameters from compilers that can't cope. */
846:
847:#ifdef PROTO
848:#define PP(arglist) arglist
849:#else
850:#define PP(arglist) ()
851:#endif
852:
853:
854:/* main entry for compression */
855:EXTERN void jpeg_compress PP((compress_info_ptr cinfo));
856:
857:/* default parameter setup for compression */
858:EXTERN void j_c_defaults PP((compress_info_ptr cinfo, int quality,
859:                 boolean force_baseline));
860:EXTERN void j_monochrome_default PP((compress_info_ptr cinfo));
861:EXTERN void j_set_quality PP((compress_info_ptr cinfo, int quality,
862:                  boolean force_baseline));
863:/* advanced compression parameter setup aids */
864:EXTERN void j_add_quant_table PP((compress_info_ptr cinfo, int which_tbl,
865:                  const QUANT_VAL *basic_table,
866:                  int scale_factor, boolean force_baseline));
867:EXTERN int j_quality_scaling PP((int quality));
868:
869:/* main entry for decompression */
870:EXTERN void jpeg_decompress PP((decompress_info_ptr cinfo));
871:
872:/* default parameter setup for decompression */
873:EXTERN void j_d_defaults PP((decompress_info_ptr cinfo,
874:                 boolean standard_buffering));
875:
876:/* forward DCT */
877:EXTERN void j_fwd_dct PP((DCTBLOCK data));
878:/* inverse DCT */
879:EXTERN void j_rev_dct PP((DCTBLOCK data));
880:
881:/* utility routines in jutils.c */
882:EXTERN long jround_up PP((long a, long b));
883:EXTERN void jcopy_sample_rows PP((JSAMPARRAY input_array, int source_row,
884:                  JSAMPARRAY output_array, int dest_row,
```

```
885:                    int num_rows, long num_cols));
886:EXTERN void jcopy_block_row PP((JBLOCKROW input_row, JBLOCKROW output_row,
887:                   long num_blocks));
888:EXTERN void jzero_far PP((void FAR * target, size_t bytestozero));
889:
890:/* method selection routines for compression modules */
891:EXTERN void jselcpipeline PP((compress_info_ptr cinfo)); /* jcpipe.c */
892:EXTERN void jselchuffman PP((compress_info_ptr cinfo)); /* jchuff.c */
893:EXTERN void jselcarithmetic PP((compress_info_ptr cinfo)); /* jcarith.c */
894:EXTERN void jselexpand PP((compress_info_ptr cinfo)); /* jcexpand.c */
895:EXTERN void jseldownsample PP((compress_info_ptr cinfo)); /* jcsample.c */
896:EXTERN void jselcmcu PP((compress_info_ptr cinfo)); /* jcmcu.c */
897:EXTERN void jselccolor PP((compress_info_ptr cinfo));   /* jccolor.c */
898:/* The user interface should call one of these to select input format: */
899:EXTERN void jselrgif PP((compress_info_ptr cinfo)); /* jrdgif.c */
900:EXTERN void jselrppm PP((compress_info_ptr cinfo)); /* jrdppm.c */
901:EXTERN void jselrrle PP((compress_info_ptr cinfo)); /* jrdrle.c */
902:EXTERN void jselrtarga PP((compress_info_ptr cinfo)); /* jrdtarga.c */
903:/* and one of these to select output header format: */
904:EXTERN void jselwjfif PP((compress_info_ptr cinfo)); /* jwrjfif.c */
905:
906:/* method selection routines for decompression modules */
907:EXTERN void jseldpipeline PP((decompress_info_ptr cinfo)); /* jdpipe.c */
908:EXTERN void jseldhuffman PP((decompress_info_ptr cinfo)); /* jdhuff.c */
909:EXTERN void jseldarithmetic PP((decompress_info_ptr cinfo)); /* jdarith.c */
910:EXTERN void jseldmcu PP((decompress_info_ptr cinfo)); /* jdmcu.c */
911:EXTERN void jselbsmooth PP((decompress_info_ptr cinfo)); /* jbsmooth.c */
912:EXTERN void jselupsample PP((decompress_info_ptr cinfo)); /* jdsample.c */
913:EXTERN void jseldcolor PP((decompress_info_ptr cinfo)); /* jdcolor.c */
914:EXTERN void jsel1quantize PP((decompress_info_ptr cinfo)); /* jquant1.c */
915:EXTERN void jsel2quantize PP((decompress_info_ptr cinfo)); /* jquant2.c */
916:/* The user interface should call one of these to select input format: */
917:EXTERN void jselrjfif PP((decompress_info_ptr cinfo)); /* jrdjfif.c */
918:/* and one of these to select output image format: */
919:EXTERN void jselwgif PP((decompress_info_ptr cinfo)); /* jwrgif.c */
920:EXTERN void jselwppm PP((decompress_info_ptr cinfo)); /* jwrppm.c */
921:EXTERN void jselwrle PP((decompress_info_ptr cinfo)); /* jwrrle.c */
922:EXTERN void jselwtarga PP((decompress_info_ptr cinfo)); /* jwrtarga.c */
923:
924:/* method selection routines for system-dependent modules */
925:EXTERN void jselerror PP((external_methods_ptr emethods)); /* jerror.c */
926:EXTERN void jselmemmgr PP((external_methods_ptr emethods)); /* jmemmgr.c */
927:
928:
929:/* We assume that right shift corresponds to signed division by 2 with
930: * rounding towards minus infinity.  This is correct for typical "arithmetic
931: * shift" instructions that shift in copies of the sign bit.  But some
932: * C compilers implement >> with an unsigned shift.  For these machines you
933: * must define RIGHT_SHIFT_IS_UNSIGNED.
934: * RIGHT_SHIFT provides a proper signed right shift of an INT32 quantity.
935: * It is only applied with constant shift counts.  SHIFT_TEMPS must be
936: * included in the variables of any routine using RIGHT_SHIFT.
937: */
938:
939:#ifdef RIGHT_SHIFT_IS_UNSIGNED
940:#define SHIFT_TEMPS INT32 shift_temp;
941:#define RIGHT_SHIFT(x,shft)  ¥
942:    ((shift_temp = (x)) < 0 ? ¥
943:     (shift_temp >> (shft)) | ((~((INT32) 0)) << (32-(shft))) : ¥
944:     (shift_temp >> (shft)))
945:#else
946:#define SHIFT_TEMPS
947:#define RIGHT_SHIFT(x,shft) ((x) >> (shft))
948:#endif
949:
950:
951:/* Miscellaneous useful macros */
952:
```

```
953:#undef MAX
954:#define MAX(a,b)    ((a) > (b) ? (a) : (b))
955:#undef MIN
956:#define MIN(a,b)    ((a) < (b) ? (a) : (b))
957:
958:
959:#define RST0    0xD0        /* RST0 marker code */
960:
```

```
// mainfrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
    CMainFrame();
    DECLARE_DYNCREATE(CMainFrame)

// Attributes
public:

// Operations
public:

// Implementation
public:
    virtual ~CMainFrame();
// Precreate
    virtual BOOL PreCreateWindow(CREATESTRUCT &cs);

#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
 1:#ifdef __cplusplus
 2:extern "C" {
 3:#endif
 4:
 5:int compMTempLength(const void *v1, const void *v2);
 6:int compMTempY(const void *v1, const void *v2);
 7:int compMTempX(const void *v1, const void *v2);
 8:void lineUPMarker(MarkConf *conf, Mark grn[], Mark red[], Mark marker[], int *mn);
 9:
10:#ifdef __cplusplus
11:}
12:#endif
13:
```

```
 1:// ﾏｰｶ追跡に関する定義ﾍｯﾀﾞﾌｧｲﾙ
 2:
 3:#define RLevel(x) ((x & 0x001f) >>  0)
 4:#define GLevel(x) ((x & 0x02e0) >>  5)
 5:#define BLevel(x) ((x & 0x7c00) >> 10)
 6:
 7:// config. file
 8:#define GRAB_WIN_X  320
 9:#define GRAB_WIN_Y  240
10:
11:// ﾏｰｶ検索範囲のﾏｰｼﾞﾝ
12:#define LEFT_MARGIN          30
13:#define RIGHT_MARGIN         30
14:#define TOP_MARGIN           30
15:#define BOTTOM_MARGIN        30
16:
17:// ﾏｰｶと認識する下限
18:#define MARK_LEVEL           0x7f00
19:#define RED_MARK_LEVEL       0
20:
21:// ﾏｰｶ検索範囲
22:#define MARK_SCAN_WIDTH      40
23:#define MARK_SCAN_HEIGHT     40
24:
25:#define RED_MARK_SCAN_WIDTH      60
26:#define RED_MARK_SCAN_HEIGHT     200
27:
28:// 眼球追跡に関する定義
29:#define EYE_CLOSE_RATE       40
30:#define EYE_AREA_MOVE_X      100
31:#define EYE_AREA_MOVE_Y      30
32:
33:// その他
34:#define REGION_RATIO         0.035
35:#define PORT_NUM             7000
36:#define PKT_SIZE             128
37:
38:// Type definition.
39:
40:typedef unsigned char uchar;
41:typedef struct {
42:    int x, y;
43:    long area;
44:} Mark;
45:
46:typedef struct {
47:    int xl, xr;
48:    int yb, yt;
49:    int attr;
50:} Area;
51:
52:typedef struct {
53:    int id;
54:    Area area;
55:} MarkID;
56:
57:typedef struct {
58:    int width;
59:    int height;
60:} Size;
61:
62:typedef struct {
63:    int x, y;
64:} VideoSize;
65:
66:typedef struct {
67:    int nOfGreen;
68:    int nOfRed;
```

```
69:     int nOfCenter;
70:     MarkID *center;
71:     int nOfUpper;
72:     MarkID *upper;
73:     int nOfBottom;
74:     MarkID *bottom;
75:     MarkID eye;
76:     int eyeOpenID;
77:     Size eyeMove;
78:     Size grnScan;
79:     Size redScan;
80:} MarkConf;
81:
```

```
1://{{NO_DEPENDENCIES}}
2:// App Studio generated include file.
3:// Used by TRACE.RC
4://
5:#define IDR_MAINFRAME                   2
6:#define IDD_ABOUTBOX                    100
7:#define IDD_DIALOG1                     102
8:#define IDD_DIALOG2                     104
9:#define IDD_DIALOG3                     105
10:#define IDB_BITMAP1                     106
11:#define IDD_DIALOG4                     107
12:#define IDC_SERVER                      1000
13:#define IDC_RADIO2                      1003
14:#define IDC_RADIO3                      1004
15:#define IDC_RADIO4                      1005
16:#define IDC_RADIO5                      1006
17:#define IDC_RADIO6                      1007
18:#define IDC_EDIT1                       1008
19:#define IDC_EDIT2                       1009
20:#define IDC_EDIT3                       1010
21:#define IDC_COMBO2                      1010
22:#define IDC_EDIT4                       1011
23:#define IDC_EDIT5                       1012
24:#define IDC_EDIT6                       1013
25:#define ID_END                          32771
26:#define ID_MENUITEM32772                32772
27:#define ID_MENUSTART                    32773
28:#define ID_MENUITEM32774                32774
29:#define ID_RETRY                        32778
30:#define ID_QUIT                         32782
31:#define ID_LD                           32784
32:#define ID_MENUITEM32786                32786
33:
34:// Next default values for new objects
35://
36:#ifdef APSTUDIO_INVOKED
37:#ifndef APSTUDIO_READONLY_SYMBOLS
38:
39:#define _APS_NEXT_RESOURCE_VALUE        109
40:#define _APS_NEXT_COMMAND_VALUE         32787
41:#define _APS_NEXT_CONTROL_VALUE         1011
42:#define _APS_NEXT_SYMED_VALUE           101
43:#endif
44:#endif
45:
```

```
 1://////////////////////////////////////////////////////////////////////////////
 2:// Class CSerial : public CObject
 3:
 4:// Class definition
 5:
 6:class CSerial : public CObject
 7:{
 8:public:
 9:    CSerial();
10:    CSerial(HWND);
11:    DECLARE_DYNCREATE( CSerial )
12:
13:// Member variables
14:private:
15:    HWND m_hSWnd;
16:
17:    DCB m_dcb;
18:    int m_idComDev;
19:    int m_Port;
20:    int m_Baud;
21:    int m_Parity;
22:    int m_Length;
23:    int m_Stop;
24:    int m_Qr;
25:    int m_Qs;
26:
27:// Member functions
28:public:
29:    int InitSerial(int, int, int, int, int, int, int);
30:    int ChangeStatus(int, int, int, int);
31:    int SetBreak();
32:    int ClearBreak();
33:    int CloseSerial();
34:    int FlushQueue(int);
35:    int EnableNotifyMsg(int, int);
36:    int GetLastError();
37:    int GetSerialStatus(int);
38:    int Read(unsigned char*, int);
39:    int Write(unsigned char*, int);
40:};
41:
42:#define CS_PORT_COM1    0
43:#define CS_PORT_COM2    1
44:#define CS_PORT_LPT1    2
45:#define CS_PORT_LPT2    3
46:
47:#define CS_BOUD_1200    0
48:#define CS_BOUD_2400    1
49:#define CS_BOUD_4800    2
50:#define CS_BOUD_9600    3
51:#define CS_BOUD_19200   4
52:
53:#define CS_PARITY_NONE  0
54:#define CS_PARITY_ODD   1
55:#define CS_PARITY_EVEN  2
56:
57:#define CS_LENGTH_4     0
58:#define CS_LENGTH_5     1
59:#define CS_LENGTH_6     2
60:#define CS_LENGTH_7     3
61:#define CS_LENGTH_8     4
62:
63:#define CS_STOP_1       0
64:#define CS_STOP_1H      1
65:#define CS_STOP_2       2
66:
67:#define CS_NOW          255
68:
```

```
69:#define CS_QUE_SEND     0
70:#define CS_QUE_RECEIVE  1
71:#define CS_QUE_BOTH     2
72:
```

```
 1:// SOCKDLG.H : header file
 2://
 3:
 4://///////////////////////////////////////////////////////////////////////////
 5:// CSockDlg dialog
 6:
 7:class CSockDlg : public CDialog
 8:{
 9:// Construction
10:public:
11:    CSockDlg(CWnd* pParent = NULL); // standard constructor
12:
13:// Dialog Data
14:    //{{AFX_DATA(CSockDlg)
15:    enum { IDD = IDD_DIALOG1 };
16:    CButton m_Ok;
17:    CButton m_Cancel;
18:    CString m_Name;
19:    //}}AFX_DATA
20:
21:
22:// Overrides
23:public:
24:
25:protected:
26:    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
27:
28:// Implementation
29:protected:
30:
31:    // Generated message map functions
32:    //{{AFX_MSG(CSockDlg)
33:    //}}AFX_MSG
34:    DECLARE_MESSAGE_MAP()
35:};
36:
```

```
1:////////////////////////////////////////////////////////////////////////////
2:// Class CSock : CObject
3:
4:class CSocket : public CObject
5:{
6:public:
7:    CSocket();
8:    ~CSocket();
9:    DECLARE_DYNCREATE( CSocket )
10:
11:// ﾒﾝﾊﾞ変数
12:private:
13:    int     m_Socket;                   // ｿｹｯﾄﾃﾞｨｽｸﾘﾌﾟﾀ
14:    CString m_SrvName;                  // ｻｰﾊﾞ名
15:
16:    int     InitSocket(void);
17:// ﾒﾝﾊﾞ関数
18:public:
19:    int     Connect();                  // ｻｰﾊﾞとの接続
20:    void    SShutdown();                // ｼｬｯﾄﾀﾞｳﾝ
21:    void    SetSrvName(CString buf);    // ｻｰﾊﾞ名の変更
22:    CString GetSrvName();               // 現在のｻｰﾊﾞ名の取得
23:    void    SetSocket(int s);           // ｿｹｯﾄﾃﾞｨｽｸﾘﾌﾟﾀの変更
24:    int     GetSocket();                // 現在のｿｹｯﾄﾃﾞｨｽｸﾘﾌﾟﾀを取得
25:    void    Send(short Pkt_Tbl[]);      // 送信
26:    short   Receive(int* len);          // 受信
27:
28:    int     StartInterrupt(HWND hWnd);  // ｳｨﾝﾄﾞｳｽﾞﾒｯｾｰｼﾞの発生
29:    int     StopInterrupt(HWND hWnd);   // ｳｨﾝﾄﾞｳｽﾞﾒｯｾｰｼﾞの発生
30:};
31:
32:#define WM_WINSOCK  WM_USER + 50        // ｻｰﾊﾞの送信要求時に発生するｳｨﾝﾄﾞｳｽﾞﾒｯｾｰｼﾞ
33:
```

```
1:// stdafx.h : include file for standard system include files,
2://  or project specific include files that are used frequently, but
3://      are changed infrequently
4://
5:
6:#include <afxwin.h>         // MFC core and standard components
7:#include <afxext.h>         // MFC extensions (including VB)
8:
```

```
// trace.h : main header file for the TRACE application
//

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CTraceApp:
// See trace.cpp for the implementation of this class
//

class CTraceApp : public CWinApp
{
public:
    CTraceApp();

// Overrides
    virtual BOOL InitInstance();

// Implementation

    //{{AFX_MSG(CTraceApp)
    afx_msg void OnAppAbout();
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

```

```
// tracedoc.h : interface of the CTraceDoc class
//
/////////////////////////////////////////////////////////////////////////////

class CTraceDoc : public CDocument
{
protected: // create from serialization only
    CTraceDoc();
    DECLARE_DYNCREATE(CTraceDoc)

// Attributes
public:
// Operations
public:

// Implementation
public:
    virtual ~CTraceDoc();
    virtual void Serialize(CArchive& ar);   // overridden for document i/o
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:
    virtual BOOL OnNewDocument();

// Generated message map functions
protected:
    //{{AFX_MSG(CTraceDoc)
        // NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

```

```
// tracevw.h : interface of the CTraceView class
//
/////////////////////////////////////////////////////////////////////////////

class CSocket;
class CCapture;
class CGrabDlg;
class CTraceView : public CView
{
protected: // create from serialization only
    CTraceView();
    DECLARE_DYNCREATE(CTraceView)

// Attributes
public:
    CTraceDoc* GetDocument();

// Operations
public:

// Member variables
private:
    BOOL        m_FFlag, m_CFlag, m_QFlag;
    CClientDC   *m_pDC;
    int  m_tid;
    CBitmap *phBitmap;
    CBrush  *phBrush;
    CPen    *phPen;

// for Socket
    int         m_Socket;
    CSocket*    m_pSocket;
    CString     m_SrvName;
    short       m_InitBuf[256];
    short       m_SockBuf[256];

// for Capture
    int         m_Counter;
    CGrabDlg* m_pDlg;
    CCapture* m_pCapture;
    CDC * phdc;

// Member functions
public:
    // Create error message box
    void PutErrorMsg(int err, int Param1, CString Param2);
    // Abnormal termination
    void Terminate(int x);
    void GetInitValue(void);

// Implementation
public:
    virtual ~CTraceView();
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CTraceView)
    afx_msg void OnMenustart();
    afx_msg void OnEnd();
    afx_msg void OnTimer(UINT nIDEvent);
    afx_msg void OnQuit();
```

```
    afx_msg LONG OnWinsock(UINT, LONG);
    afx_msg void OnRetry();
    afx_msg void OnSetSock();
    afx_msg void OnMenuArea();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in tracevw.cpp
inline CTraceDoc* CTraceView::GetDocument()
   { return (CTraceDoc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////

// Definition of error codes
#define ERR_CONFIG  0
#define ERR_SOCKET  1
#define ERR_MARKER  2
#define ERR_MEMORY  3


```

```
// VINDLG.H : header file
//

/////////////////////////////////////////////////////////////////////////////
// CVinDlg dialog

class CVinDlg : public CDialog
{
// Construction
public:
    CVinDlg(CWnd* pParent = NULL);  // standard constructor

// Dialog Data
    //{{AFX_DATA(CVinDlg)
    enum { IDD = IDD_DIALOG2 };
        // NOTE: the ClassWizard will add data members here
    //}}AFX_DATA


// Overrides
public:

protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CVinDlg)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

```
1:#include "cmtlib.h"
2://#include "intrn¥cmtlib.h"
3:
4:int GrabFlag;
5:
6:void interrupt GrabInt(void)
7:{
8:    GrabFlag = TRUE;
9:}
```

```
/*
 * jfwddct.c
 *
 * Copyright (C) 1991, 1992, Thomas G. Lane.
 * This file is part of the Independent JPEG Group's software.
 * For conditions of distribution and use, see the accompanying README file.
 *
 * This file contains the basic DCT (Discrete Cosine Transform)
 * transformation subroutine.
 *
 * This implementation is based on an algorithm described in
 *   C. Loeffler, A. Ligtenberg and G. Moschytz, "Practical Fast 1-D DCT
 *   Algorithms with 11 Multiplications", Proc. Int'l. Conf. on Acoustics,
 *   Speech, and Signal Processing 1989 (ICASSP '89), pp. 988-991.
 * The primary algorithm described there uses 11 multiplies and 29 adds.
 * We use their alternate method with 12 multiplies and 32 adds.
 * The advantage of this method is that no data path contains more than one
 * multiplication; this allows a very simple and accurate implementation in
 * scaled fixed-point arithmetic, with a minimal number of shifts.
 */

#include "jinclude.h"

/*
 * This routine is specialized to the case DCTSIZE = 8.
 */

#if DCTSIZE != 8
  Sorry, this code only copes with 8x8 DCTs. /* deliberate syntax err */
#endif


/*
 * A 2-D DCT can be done by 1-D DCT on each row followed by 1-D DCT
 * on each column.  Direct algorithms are also available, but they are
 * much more complex and seem not to be any faster when reduced to code.
 *
 * The poop on this scaling stuff is as follows:
 *
 * Each 1-D DCT step produces outputs which are a factor of sqrt(N)
 * larger than the true DCT outputs.  The final outputs are therefore
 * a factor of N larger than desired; since N=8 this can be cured by
 * a simple right shift at the end of the algorithm.  The advantage of
 * this arrangement is that we save two multiplications per 1-D DCT,
 * because the y0 and y4 outputs need not be divided by sqrt(N).
 *
 * We have to do addition and subtraction of the integer inputs, which
 * is no problem, and multiplication by fractional constants, which is
 * a problem to do in integer arithmetic.  We multiply all the constants
 * by CONST_SCALE and convert them to integer constants (thus retaining
 * CONST_BITS bits of precision in the constants).  After doing a
 * multiplication we have to divide the product by CONST_SCALE, with proper
 * rounding, to produce the correct output.  This division can be done
 * cheaply as a right shift of CONST_BITS bits.  We postpone shifting
 * as long as possible so that partial sums can be added together with
 * full fractional precision.
 *
 * The outputs of the first pass are scaled up by PASS1_BITS bits so that
 * they are represented to better-than-integral precision.  These outputs
 * require BITS_IN_JSAMPLE + PASS1_BITS + 3 bits; this fits in a 16-bit word
 * with the recommended scaling.  (To scale up 12-bit sample data, an
 * intermediate INT32 array would be needed.)
 *
 * To avoid overflow of the 32-bit intermediate results in pass 2, we must
 * have BITS_IN_JSAMPLE + CONST_BITS + PASS1_BITS <= 25.  Error analysis
 * shows that the values given below are the most effective.
 */

```

```
 69:#ifdef EIGHT_BIT_SAMPLES
 70:#define CONST_BITS  13
 71:#define PASS1_BITS  2
 72:#else
 73:#define CONST_BITS  13
 74:#define PASS1_BITS  0       /* lose a little precision to avoid overflow */
 75:#endif
 76:
 77:#define ONE ((INT32) 1)
 78:
 79:#define CONST_SCALE (ONE << CONST_BITS)
 80:
 81:/* Convert a positive real constant to an integer scaled by CONST_SCALE. */
 82:
 83:#define FIX(x)  ((INT32) ((x) * CONST_SCALE + 0.5))
 84:
 85:/* Some C compilers fail to reduce "FIX(constant)" at compile time, thus
 86: * causing a lot of useless floating-point operations at run time.
 87: * To get around this we use the following pre-calculated constants.
 88: * If you change CONST_BITS you may want to add appropriate values.
 89: * (With a reasonable C compiler, you can just rely on the FIX() macro...)
 90: */
 91:
 92:#if CONST_BITS == 13
 93:#define FIX_0_298631336  ((INT32)  2446)    /* FIX(0.298631336) */
 94:#define FIX_0_390180644  ((INT32)  3196)    /* FIX(0.390180644) */
 95:#define FIX_0_541196100  ((INT32)  4433)    /* FIX(0.541196100) */
 96:#define FIX_0_765366865  ((INT32)  6270)    /* FIX(0.765366865) */
 97:#define FIX_0_899976223  ((INT32)  7373)    /* FIX(0.899976223) */
 98:#define FIX_1_175875602  ((INT32)  9633)    /* FIX(1.175875602) */
 99:#define FIX_1_501321110  ((INT32)  12299)   /* FIX(1.501321110) */
100:#define FIX_1_847759065  ((INT32)  15137)   /* FIX(1.847759065) */
101:#define FIX_1_961570560  ((INT32)  16069)   /* FIX(1.961570560) */
102:#define FIX_2_053119869  ((INT32)  16819)   /* FIX(2.053119869) */
103:#define FIX_2_562915447  ((INT32)  20995)   /* FIX(2.562915447) */
104:#define FIX_3_072711026  ((INT32)  25172)   /* FIX(3.072711026) */
105:#else
106:#define FIX_0_298631336  FIX(0.298631336)
107:#define FIX_0_390180644  FIX(0.390180644)
108:#define FIX_0_541196100  FIX(0.541196100)
109:#define FIX_0_765366865  FIX(0.765366865)
110:#define FIX_0_899976223  FIX(0.899976223)
111:#define FIX_1_175875602  FIX(1.175875602)
112:#define FIX_1_501321110  FIX(1.501321110)
113:#define FIX_1_847759065  FIX(1.847759065)
114:#define FIX_1_961570560  FIX(1.961570560)
115:#define FIX_2_053119869  FIX(2.053119869)
116:#define FIX_2_562915447  FIX(2.562915447)
117:#define FIX_3_072711026  FIX(3.072711026)
118:#endif
119:
120:
121:/* Descale and correctly round an INT32 value that's scaled by N bits.
122: * We assume RIGHT_SHIFT rounds towards minus infinity, so adding
123: * the fudge factor is correct for either sign of X.
124: */
125:
126:#define DESCALE(x,n)  RIGHT_SHIFT((x) + (ONE << ((n)-1)), n)
127:
128:/* Multiply an INT32 variable by an INT32 constant to yield an INT32 result.
129: * For 8-bit samples with the recommended scaling, all the variable
130: * and constant values involved are no more than 16 bits wide, so a
131: * 16x16->32 bit multiply can be used instead of a full 32x32 multiply;
132: * this provides a useful speedup on many machines.
133: * There is no way to specify a 16x16->32 multiply in portable C, but
134: * some C compilers will do the right thing if you provide the correct
135: * combination of casts.
136: * NB: for 12-bit samples, a full 32-bit multiplication will be needed.
```

```
137: */
138:
139:#ifdef EIGHT_BIT_SAMPLES
140:#ifdef SHORTxSHORT_32       /* may work if 'int' is 32 bits */
141:#define MULTIPLY(var,const)  (((INT16) (var)) * ((INT16) (const)))
142:#endif
143:#ifdef SHORTxLCONST_32      /* known to work with Microsoft C 6.0 */
144:#define MULTIPLY(var,const)  (((INT16) (var)) * ((INT32) (const)))
145:#endif
146:#endif
147:
148:#ifndef MULTIPLY        /* default definition */
149:#define MULTIPLY(var,const)  ((var) * (const))
150:#endif
151:
152:
153:/*
154: * Perform the forward DCT on one block of samples.
155: */
156:
157:void j_fwd_dct (data)
158:DCTELEM   data[64] ;
159:{
160:  INT32 tmp0, tmp1, tmp2, tmp3, tmp4, tmp5, tmp6, tmp7;
161:  INT32 tmp10, tmp11, tmp12, tmp13;
162:  INT32 z1, z2, z3, z4, z5;
163:  register DCTELEM *dataptr;
164:  int rowctr;
165:  SHIFT_TEMPS
166:
167:  /* Pass 1: process rows. */
168:  /* Note results are scaled up by sqrt(8) compared to a true DCT; */
169:  /* furthermore, we scale the results by 2**PASS1_BITS. */
170:
171:  dataptr = data;
172:  for (rowctr = DCTSIZE-1; rowctr >= 0; rowctr--) {
173:    tmp0 = dataptr[0] + dataptr[7];
174:    tmp7 = dataptr[0] - dataptr[7];
175:    tmp1 = dataptr[1] + dataptr[6];
176:    tmp6 = dataptr[1] - dataptr[6];
177:    tmp2 = dataptr[2] + dataptr[5];
178:    tmp5 = dataptr[2] - dataptr[5];
179:    tmp3 = dataptr[3] + dataptr[4];
180:    tmp4 = dataptr[3] - dataptr[4];
181:
182:    /* Even part per LL&M figure 1 --- note that published figure is faulty;
183:     * rotator "sqrt(2)*c1" should be "sqrt(2)*c6".
184:     */
185:
186:    tmp10 = tmp0 + tmp3;
187:    tmp13 = tmp0 - tmp3;
188:    tmp11 = tmp1 + tmp2;
189:    tmp12 = tmp1 - tmp2;
190:
191:    dataptr[0] = (DCTELEM) ((tmp10 + tmp11) << PASS1_BITS);
192:    dataptr[4] = (DCTELEM) ((tmp10 - tmp11) << PASS1_BITS);
193:
194:    z1 = MULTIPLY(tmp12 + tmp13, FIX_0_541196100);
195:    dataptr[2] = (DCTELEM) DESCALE(z1 + MULTIPLY(tmp13, FIX_0_765366865),
196:                   CONST_BITS-PASS1_BITS);
197:    dataptr[6] = (DCTELEM) DESCALE(z1 + MULTIPLY(tmp12, - FIX_1_847759065),
198:                   CONST_BITS-PASS1_BITS);
199:
200:    /* Odd part per figure 8 --- note paper omits factor of sqrt(2).
201:     * cK represents cos(K*pi/16).
202:     * i0..i3 in the paper are tmp4..tmp7 here.
203:     */
204:
```

```
205:     z1 = tmp4 + tmp7;
206:     z2 = tmp5 + tmp6;
207:     z3 = tmp4 + tmp6;
208:     z4 = tmp5 + tmp7;
209:     z5 = MULTIPLY(z3 + z4, FIX_1_175875602); /* sqrt(2) * c3 */
210:
211:     tmp4 = MULTIPLY(tmp4, FIX_0_298631336); /* sqrt(2) * (-c1+c3+c5-c7) */
212:     tmp5 = MULTIPLY(tmp5, FIX_2_053119869); /* sqrt(2) * ( c1+c3-c5+c7) */
213:     tmp6 = MULTIPLY(tmp6, FIX_3_072711026); /* sqrt(2) * ( c1+c3+c5-c7) */
214:     tmp7 = MULTIPLY(tmp7, FIX_1_501321110); /* sqrt(2) * ( c1+c3-c5-c7) */
215:     z1 = MULTIPLY(z1, - FIX_0_899976223); /* sqrt(2) * (c7-c3) */
216:     z2 = MULTIPLY(z2, - FIX_2_562915447); /* sqrt(2) * (-c1-c3) */
217:     z3 = MULTIPLY(z3, - FIX_1_961570560); /* sqrt(2) * (-c3-c5) */
218:     z4 = MULTIPLY(z4, - FIX_0_390180644); /* sqrt(2) * (c5-c3) */
219:
220:     z3 += z5;
221:     z4 += z5;
222:
223:     dataptr[7] = (DCTELEM) DESCALE(tmp4 + z1 + z3, CONST_BITS-PASS1_BITS);
224:     dataptr[5] = (DCTELEM) DESCALE(tmp5 + z2 + z4, CONST_BITS-PASS1_BITS);
225:     dataptr[3] = (DCTELEM) DESCALE(tmp6 + z2 + z3, CONST_BITS-PASS1_BITS);
226:     dataptr[1] = (DCTELEM) DESCALE(tmp7 + z1 + z4, CONST_BITS-PASS1_BITS);
227:
228:     dataptr += DCTSIZE;     /* advance pointer to next row */
229:   }
230:
231:   /* Pass 2: process columns. */
232:   /* Note that we must descale the results by a factor of 8 == 2**3, */
233:   /* and also undo the PASS1_BITS scaling. */
234:
235:   dataptr = data;
236:   for (rowctr = DCTSIZE-1; rowctr >= 0; rowctr--) {
237:     tmp0 = dataptr[DCTSIZE*0] + dataptr[DCTSIZE*7];
238:     tmp7 = dataptr[DCTSIZE*0] - dataptr[DCTSIZE*7];
239:     tmp1 = dataptr[DCTSIZE*1] + dataptr[DCTSIZE*6];
240:     tmp6 = dataptr[DCTSIZE*1] - dataptr[DCTSIZE*6];
241:     tmp2 = dataptr[DCTSIZE*2] + dataptr[DCTSIZE*5];
242:     tmp5 = dataptr[DCTSIZE*2] - dataptr[DCTSIZE*5];
243:     tmp3 = dataptr[DCTSIZE*3] + dataptr[DCTSIZE*4];
244:     tmp4 = dataptr[DCTSIZE*3] - dataptr[DCTSIZE*4];
245:
246:     /* Even part per LL&M figure 1 --- note that published figure is faulty;
247:      * rotator "sqrt(2)*c1" should be "sqrt(2)*c6".
248:      */
249:
250:     tmp10 = tmp0 + tmp3;
251:     tmp13 = tmp0 - tmp3;
252:     tmp11 = tmp1 + tmp2;
253:     tmp12 = tmp1 - tmp2;
254:
255:     dataptr[DCTSIZE*0] = (DCTELEM) DESCALE(tmp10 + tmp11, PASS1_BITS+3);
256:     dataptr[DCTSIZE*4] = (DCTELEM) DESCALE(tmp10 - tmp11, PASS1_BITS+3);
257:
258:     z1 = MULTIPLY(tmp12 + tmp13, FIX_0_541196100);
259:     dataptr[DCTSIZE*2] = (DCTELEM) DESCALE(z1 + MULTIPLY(tmp13, FIX_0_765366865),
260:                                            CONST_BITS+PASS1_BITS+3);
261:     dataptr[DCTSIZE*6] = (DCTELEM) DESCALE(z1 + MULTIPLY(tmp12, - FIX_1_847759065),
262:                                            CONST_BITS+PASS1_BITS+3);
263:
264:     /* Odd part per figure 8 --- note paper omits factor of sqrt(2).
265:      * cK represents cos(K*pi/16).
266:      * i0..i3 in the paper are tmp4..tmp7 here.
267:      */
268:
269:     z1 = tmp4 + tmp7;
270:     z2 = tmp5 + tmp6;
271:     z3 = tmp4 + tmp6;
272:     z4 = tmp5 + tmp7;
```

```
273:     z5 = MULTIPLY(z3 + z4, FIX_1_175875602); /* sqrt(2) * c3 */
274:
275:     tmp4 = MULTIPLY(tmp4, FIX_0_298631336); /* sqrt(2) * (-c1+c3+c5-c7) */
276:     tmp5 = MULTIPLY(tmp5, FIX_2_053119869); /* sqrt(2) * ( c1+c3-c5+c7) */
277:     tmp6 = MULTIPLY(tmp6, FIX_3_072711026); /* sqrt(2) * ( c1+c3+c5-c7) */
278:     tmp7 = MULTIPLY(tmp7, FIX_1_501321110); /* sqrt(2) * ( c1+c3-c5-c7) */
279:     z1 = MULTIPLY(z1, - FIX_0_899976223); /* sqrt(2) * (c7-c3) */
280:     z2 = MULTIPLY(z2, - FIX_2_562915447); /* sqrt(2) * (-c1-c3) */
281:     z3 = MULTIPLY(z3, - FIX_1_961570560); /* sqrt(2) * (-c3-c5) */
282:     z4 = MULTIPLY(z4, - FIX_0_390180644); /* sqrt(2) * (c5-c3) */
283:
284:     z3 += z5;
285:     z4 += z5;
286:
287:     dataptr[DCTSIZE*7] = (DCTELEM) DESCALE(tmp4 + z1 + z3,
288:                          CONST_BITS+PASS1_BITS+3);
289:     dataptr[DCTSIZE*5] = (DCTELEM) DESCALE(tmp5 + z2 + z4,
290:                          CONST_BITS+PASS1_BITS+3);
291:     dataptr[DCTSIZE*3] = (DCTELEM) DESCALE(tmp6 + z2 + z3,
292:                          CONST_BITS+PASS1_BITS+3);
293:     dataptr[DCTSIZE*1] = (DCTELEM) DESCALE(tmp7 + z1 + z4,
294:                          CONST_BITS+PASS1_BITS+3);
295:
296:     dataptr++;          /* advance pointer to next column */
297:   }
298:}
299:
```

```
 1:#include <stdlib.h>
 2:#include <malloc.h>
 3:
 4:#include "mtrace.h"
 5:
 6:typedef struct {
 7:    int id;
 8:    int length;
 9:    int x;
10:    int y;
11:} MTemp;
12:
13:int compMTempLength(const void *v1, const void *v2)
14:{
15:    MTemp *t1, *t2;
16:
17:    t1 = (MTemp *)v1;
18:    t2 = (MTemp *)v2;
19:
20:    return (t1->length - t2->length);
21:}
22:
23:int compMTempY(const void *v1, const void *v2)
24:{
25:    MTemp *t1, *t2;
26:
27:    t1 = (MTemp *)v1;
28:    t2 = (MTemp *)v2;
29:
30:    return (t1->y - t2->y);
31:}
32:
33:int compMTempX(const void *v1, const void *v2)
34:{
35:    MTemp *t1, *t2;
36:
37:    t1 = (MTemp *)v1;
38:    t2 = (MTemp *)v2;
39:
40:    return (t2->x - t1->x);
41:}
42:
43:void lineUPMarker(MarkConf *conf, Mark grn[], Mark red[], Mark marker[], int *mn)
44:{
45:    int i, n;
46:    MTemp *index;
47://  MTemp dind[20];
48:
49:    index = calloc(conf->nOfGreen, sizeof(MTemp));
50:    n = conf->nOfGreen;
51:    // 顔中央の線上のﾏｰｶを抽出
52:    for (i = 0; i < conf->nOfGreen; i++) {
53:        index[i].id = i;
54:        index[i].length = abs(grn[i].y - GRAB_WIN_Y/2);
55:        index[i].x = grn[i].x;
56:        index[i].y = grn[i].y;
57:    }
58:    // y座標が中央値に近いﾏｰｶをconf->nOfCenter個抽出
59:    qsort(index, n, sizeof(MTemp), compMTempLength);
60:    // x座標でｿｰﾄして右から順番に整列
61:    qsort(index, conf->nOfCenter, sizeof(MTemp), compMTempX);
62:    for (i=0; i<conf->nOfCenter; i++) {
63:        marker[ conf->center[i].id - 1] = grn[index[i].id];
64:    }
65:    index += conf->nOfCenter;
66:    n -= conf->nOfCenter;
67:    // y座標が小さいﾏｰｶをconf->nOfBottom個抽出
68:    qsort(index, n, sizeof(MTemp), compMTempY);
```

```
69:     // x座標でソートして右から順番に整列
70:     qsort(index, conf->nOfBottom, sizeof(MTemp), compMTempX);
71:     for (i = 0; i < conf->nOfBottom; i++) {
72:         marker[conf->bottom[i].id - 1] = grn[index[i].id];
73:     }
74:     index += conf->nOfBottom;
75:     n -= conf->nOfBottom;
76:     // x座標でソートして右から順番に整列
77:     qsort(index, conf->nOfBottom, sizeof(MTemp), compMTempX);
78:     for (i = 0; i < conf->nOfUpper; i++) {
79:         marker[conf->upper[i].id - 1] = grn[index[i].id];
80:     }
81:     *mn = conf->nOfGreen;
82://   free(index);
83:|
```

```
 1:// area.cpp : implementation file
 2://
 3:
 4:#include "stdafx.h"
 5:#include "trace.h"
 6:#include "area.h"
 7:
 8:#ifdef _DEBUG
 9:#undef THIS_FILE
10:static char BASED_CODE THIS_FILE[] = __FILE__;
11:#endif
12:
13://///////////////////////////////////////////////////////////////////////////
14:// CArea dialog
15:
16:
17:CArea::CArea(CWnd* pParent /*=NULL*/)
18:        : CDialog(CArea::IDD, pParent)
19:{
20:        //{{AFX_DATA_INIT(CArea)
21:        m_hParent = pParent;
22:        m_AreaFile = _T("hirose.ara");
23:        //}}AFX_DATA_INIT
24:}
25:
26:BOOL CArea::Create()
27:{
28:        return CDialog::Create(CArea::IDD);
29:}
30:
31:void CArea::DoDataExchange(CDataExchange* pDX)
32:{
33:        CDialog::DoDataExchange(pDX);
34:        //{{AFX_DATA_MAP(CArea)
35:        DDX_Control(pDX, IDOK, m_Ok);
36:        DDX_Control(pDX, IDCANCEL, m_Cancel);
37:        DDX_CBString(pDX, IDC_COMBO2, m_AreaFile);
38:        //}}AFX_DATA_MAP
39:}
40:
41:BEGIN_MESSAGE_MAP(CArea, CDialog)
42:        //{{AFX_MSG_MAP(CArea)
43:                // NOTE: the ClassWizard will add message map macros here
44:        //}}AFX_MSG_MAP
45:END_MESSAGE_MAP()
46:
47:
48://///////////////////////////////////////////////////////////////////////////
49:// CArea message handlers
50:
```

```
 1:// capture.cpp : implementation of the CCapture class
 2://
 3:#include "stdafx.h"
 4:#include "mtrace.h"
 5:#include "capture.h"
 6:
 7:#include "cmtlib.h"
 8:#include "grabint.h"
 9:
10:#include "math.h"
11:
12:#define MAX_X 640
13:#define DCTSIZE 8
14:#define RShift(x) (x >> 5)
15:#define LShift(x) (x << 5)
16:#define rd(ql, qk, qx, qy) m_Mono[(((qk+ql*x_size)*DCTSIZE)+(qx)+((qy)*x_size))]
17://#define Addr(x, y) ((x + y*800)*2)
18:extern "C" {
19:void j_fwd_dct(short*);
20:}
21:
22:int CCapture::initFlag = 0;
23://///////////////////////////////////////////////////////////////////////////
24:// CCapture
25:
26:IMPLEMENT_DYNCREATE(CCapture, CObject)
27:
28://///////////////////////////////////////////////////////////////////////////
29:// CCapture construction
30:
31:CCapture::CCapture()
32:{
33:        m_pView = NULL;
34:        m_Counter = 0;
35:        m_aNum    = -1;
36:        m_Buffer = NULL;
37:        m_lmax = m_kmax = NULL;
38:}
39:
40:CCapture::CCapture(CView* pView)
41:{
42:        m_pView = pView;
43:        m_Counter = 0;
44:        m_aNum    = -1;
45:        m_Buffer = NULL;
46:        m_lmax = m_kmax = NULL;
47:}
48:
49:CCapture::~CCapture()
50:{
51:        CaptureQuit();
52:        FreeMalloc();
53:}
54://///////////////////////////////////////////////////////////////////////////
55:// CCapture member functions
56:
57:int CCapture::CaptureInit()
58:{
59:        VG_WORD lwError;
60:
61:        lwError = CmtGenLibInit();
62:        if (lwError < 0) return FALSE;
63:        CmtGenBoardReset();
64:        CmtVinInSelect(CMT_VIN_CHAN_A);
65:        CmtVinInSetStandard(CMT_VIN_NTSC);
66:        CmtVipGrabSetMode(CMT_VIP_GRAB_555);
67:        CmtVipGrabSetSourcePos( 0, 0, 640, 480);
68:        CmtVipGrabSetDestPos(0, 0, GRAB_WIN_X, GRAB_WIN_Y);
```

```
 69:        CmtVipGrabSetRate(CMT_VIP_GRAB_CONTINUOUS);
 70:        CmtVipGrabRegisterIntFunction(&GrabInt);
 71:        CmtVipGrabStartInterrupts();
 72:        initFlag = 1;
 73:        m_AreaFile = "hirose.ara";
 74:        return TRUE;
 75:}
 76:
 77:void CCapture::CaptureQuit()
 78:{
 79:        if ( initFlag != 0 ) {
 80:                CmtVipGrabSetRate(CMT_VIP_GRAB_OFF);
 81:                CmtVipGrabRegisterIntFunction(NULL);
 82:                CmtVipDisplaySetSt(0);
 83:                CmtVipGrabStopInterrupts();
 84:                CmtGenLibQuit();
 85:        }
 86:        initFlag = 0;
 87:}
 88:
 89:long CCapture::Addr(long x, long y)
 90:{
 91:        return ((x + y * 800)*2);
 92:}
 93:
 94:void CCapture::FreeMalloc()
 95:{
 96://
 97:        if ( m_Buffer != NULL ) delete m_Buffer;
 98:        if ( m_Imax   != NULL ) delete m_Imax;
 99:        if ( m_kmax   != NULL ) delete m_kmax;
100:        m_Buffer = NULL;
101:        m_Imax   = NULL;
102:        m_kmax   = NULL;
103:}
104:
105:short CCapture::Bufpix(int x, int y)
106:{
107:        short buf[2];
108:        CmtGenBlockRead(Addr(x, y), 1, 1, (unsigned char*)buf, sizeof(short)*2);
109://      buf[0] = Frame[x + y * GRAB_WIN_Y];
110:        buf[0] &= 0x7fff;
111:        return buf[0];
112:}
113:
114:void CCapture::Putpix(int x, int y, short src)
115:{
116:        short buf[2];
117:        buf[0] = src;
118:        CmtGenBlockWrite(Addr(x, y), 1, 1, (unsigned char*)buf, 2);
119:}
120:
121:void CCapture::CaptureStop()
122:{
123:        CmtVipGrabSetRate(CMT_VIP_GRAB_OFF);
124:}
125:void CCapture::CaptureStart()
126:{
127:        CmtVipGrabSetRate(CMT_VIP_GRAB_CONTINUOUS);
128:}
129:
130:int CCapture::DctInit()
131:{
132:        int i,j;
133:        FILE *fp;
134:        static char dmy[80], buf[80];
135:        int tmp1;
136:        static short HCut[8][8] = {
```

```
137:                0, 1, 1, 1, 1, 1, 1, 1,
138:                0, 0, 0, 1, 1, 1, 1, 1,
139:                0, 0, 0, 0, 0, 0, 0, 0,
140:                0, 0, 0, 0, 0, 0, 0, 0,
141:                0, 0, 0, 0, 0, 0, 0, 0,
142:                0, 0, 0, 0, 0, 0, 0, 0,
143:                0, 0, 0, 0, 0, 0, 0, 0,
144:                0, 0, 0, 0, 0, 0, 0, 0
145:        };
146:
147:        static short VCut[8][8] = {
148:                0, 0, 0, 0, 0, 0, 0, 0,
149:                1, 0, 0, 0, 0, 0, 0, 0,
150:                1, 0, 0, 0, 0, 0, 0, 0,
151:                1, 1, 0, 0, 0, 0, 0, 0,
152:                1, 1, 0, 0, 0, 0, 0, 0,
153:                1, 1, 0, 0, 0, 0, 0, 0,
154:                1, 1, 0, 0, 0, 0, 0, 0,
155:                1, 1, 0, 0, 0, 0, 0, 0
156:        };
157:
158:        static short HVCut[8][8] = {
159:                0, 0, 0, 0, 0, 0, 0, 0,
160:                0, 1, 0, 0, 0, 0, 0, 1,
161:                0, 0, 1, 1, 0, 0, 0, 0,
162:                0, 0, 1, 1, 1, 0, 0, 0,
163:                0, 0, 0, 1, 1, 1, 0, 0,
164:                0, 0, 0, 0, 1, 1, 1, 0,
165:                0, 0, 0, 0, 0, 1, 1, 1,
166:                0, 0, 0, 0, 0, 0, 1, 1
167:        };
168:
169:
170:        // 設定ファイルの読み込み
171:        if ((fp = fopen(m_AreaFile, "rt")) == NULL) return 255;
172:
173:        fgets(buf, 80, fp);
174:        sscanf(buf, "%s %d %d", dmy, &m_EyeMid[0], &m_EyeMid[1]);
175:        m_EyeMid[0] -= 2;
176:        m_EyeMid[1] -= 2;
177:        fgets(buf, 80, fp);
178:        sscanf(buf, "%s %d", dmy, &tmp1);
179:        m_aNum = tmp1;
180:        for (i=0;i<m_aNum;i++) {
181:                fgets(buf, 80, fp);
182:                sscanf(buf, "%s %d %d %d %d", dmy, &m_lx[i], &m_ly[i], &m_wx[i], &m_wy[i]);
183:        }
184:/*     while (fgets(buf, 80, fp) != NULL) {
185:                sscanf(buf, "%s", dmy);
186:                if (strcmp(dmy, "EYE") == 0) {
187:                        sscanf(buf, "%s %d %d", dmy, &m_EyeMid[0], &m_EyeMid[1]);
188:                m_EyeMid[0] -= 2;
189:                m_EyeMid[1] -= 2;
190:        }*/
191:        fclose(fp);
192:
193:        // ｶｯﾄﾃｰﾌﾞﾙの読み込み
194:        for (i=0;i<8;i++) {
195:                for (j=0;j<8;j++) {
196:                        m_HCut[j][i]  = HCut[j][i];
197:                        m_VCut[j][i]  = VCut[j][i];
198:                        m_HVCut[j][i] = HVCut[j][i];
199:                }
200:        }
201:
202:        m_lmax   = new int[m_aNum];
203:        m_kmax   = new int[m_aNum];
204:        m_Buffer = new short[30000];
```

```
205:
206:        for (i=0;i<m_aNum;i++) {
207:                m_S[i] = m_wx[i] * m_wy[i];
208:                m_lmax[i] =     m_wy[i] / DCTSIZE;
209:                m_kmax[i] = m_wx[i] / DCTSIZE;
210:        }
211:
212:        return 0;
213:}
214:
215:void CCapture::Dct()
216:{
217:        int i, l, k, j, tmp;
218://      long t;
219:        double rt, ry;
220:
221:        for (i=0; i<m_aNum; i++) {
222:                m_GH[i] = m_GV[i] = m_GHV[i] = 0;
223:                CmtGenBlockRead(Addr(m_lx[i],m_ly[i]), m_wx[i], m_wy[i], (unsigned char*)m_Buffer, sizeof(short)*m_S[i]);
224://              CmtGenBlockWrite(Addr(m_lx[i]+100, 0), m_wx[i], m_wy[i], (unsigned char*)m_Buffer, sizeof(short)*1000);
225:                if ( i < 2 ) {
226:                        for ( j = 0; j < 3; j++ ) {
227:                                rt = (m_wx[i] / 2) / 6;
228:                                ry = m_wy[i] / 16;
229:                                for ( k = 0; k < 16; k++ ) {
230://                                      m_EY[i][j][k] = m_Buffer[int(ry*k) * m_wx[i]+int(rt*(j+2))+5];
231:                                        m_EY[i][j][k] = m_Buffer[int(ry*k) * m_wx[i]+m_EyeMid[i]+j*2];

232:                                }
233:                        }
234:                }
235:                for (l=0; l<m_lmax[i]; l++) {
236:                        for (k=0; k<m_kmax[i]; k++) {
237:                                RGBtoMONO(8, 8, ((k+l*m_wx[i])*DCTSIZE), i);    // ﾓﾉｸﾛ画像に変換
238:                                j_fwd_dct(m_Mono);
239:                                for ( j=0; j < 8*8; j++ ) {
240:                                        if ( (tmp = abs(m_Mono[j])) > 255 ) {
241:                                                tmp = 255;
242:                                        }
243:                                        m_Mono[j] = tmp;
244:                                }
245:                                SigmaDCTBlock2(i);                                                      // H､V､HVの
    計算
246:                        }
247:                }
248:                m_GH[i]  /= m_lmax[i] * m_kmax[i];
249:                m_GV[i]  /= m_lmax[i] * m_kmax[i];
250:                m_GHV[i] /= m_lmax[i] * m_kmax[i];
251:        }
252:// ========================================                     j_fwd_dct(m_Mono);
                        // DCTを行う
253:}
254:
255:void CCapture::RGBtoMONO(int x_size, int y_size, int wl, int i)
256:{
257:        int x, y, my, ww;
258:
259:        my = 0;         ww = 0;
260:        for (y=0; y<y_size; y++) {
261:                my = x_size * y;
262:                ww = m_wx[i] * y;
263:                for (x=0; x<x_size; x++) {
264:                        m_Mono[x+my] = (short)((unsigned short)(m_Buffer[wl+x+ww] & 0x03e0) >> 2);
265:                }
266:        }
267:}
268:
269:void CCapture::SigmaDCTBlock2(int wn)
```

```
270:{
271:        int x, y, my;
272:
273:        my = 0;
274:        for(y=0; y<8; y++) {
275:                for (x=0; x<8; x++)      {
276:                        m_GH[wn]  += m_Mono[x+my] * m_HCut[y][x];
277:                        m_GV[wn]  += m_Mono[x+my] * m_VCut[y][x];
278:                        m_GHV[wn] += m_Mono[x+my] * m_HVCut[y][x];
279:                }
280:                my += 8;
281:        }
282:}
```

```
 1:// capture.cpp : implementation of the CCapture class
 2://
 3:#include "stdafx.h"
 4:#include "mtrace.h"
 5:#include "capture.h"
 6:
 7:#include "cmtlib.h"
 8:#include "grabint.h"
 9:
10:#include "math.h"
11:
12://///////////////////////////////////////////////////////////////////////////
13:// CCapture
14:
15:IMPLEMENT_DYNCREATE(CCapture, CObject)
16:
17://///////////////////////////////////////////////////////////////////////////
18:// CCapture construction
19:
20:CCapture::CCapture()
21:{
22:        m_pView = NULL;
23:        m_Standard = CMT_VIN_PAL;
24:        m_Counter = 0;
25:}
26:
27:CCapture::CCapture(CView* pView)
28:{
29:        m_pView = pView;
30:}
31:
32://///////////////////////////////////////////////////////////////////////////
33:// CCapture member functions
34:
35:void CCapture::SetStandard(uchar st)
36:{
37:        if (st > 2) st = 2;
38:        if (st < 0) st = 0;
39:        m_Standard = st;
40:        CmtVinInSetStandard(st);
41:}
42:
43:uchar CCapture::GetStandard()
44:{
45:        return m_Standard;
46:}
47:
48:void CCapture::SetType(uchar ty)
49:{
50:        if (ty > 1) ty = 1;
51:        if (ty < 0) ty = 0;
52:        m_Type = ty;
53:        CmtVinInSetType(ty);
54:}
55:
56:uchar CCapture::GetType()
57:{
58:        return m_Type;
59:}
60:
61:void CCapture::SetBrightness(uchar br)
62:{
63:        if (br > 255) br = 255;
64:        if (br <   0) br = 0;
65:        m_Brightness = br;
66:        CmtVinLumSetBrightness(br);
67:}
68:
```

```
69:uchar CCapture::GetBrightness()
70:{
71:        return m_Brightness;
72:}
73:
74:int CCapture::CaptureInit()
75:{
76:        VG_WORD lwError;
77:
78:        lwError = CmtGenLibInit();
79:        if (lwError < 0) return FALSE;
80:        CmtGenBoardReset();
81:        CmtVinInSelect(CMT_VIN_CHAN_A);
82:        CmtVinInSetStandard(CMT_VIN_NTSC);
83:        CmtVipGrabSetMode(CMT_VIP_GRAB_555);
84:        CmtVipGrabSetSourcePos( 0, 0, 640 ,480);
85:        CmtVipGrabSetDestPos(0, 0, GRAB_WIN_X, GRAB_WIN_Y);
86:        CmtVipGrabSetRate(CMT_VIP_GRAB_CONTINUOUS);
87:
88:        CmtVipGrabRegisterIntFunction(&GrabInt);
89:        CmtVipGrabStartInterrupts();
90:        return TRUE;
91:}
92:
93:void CCapture::CaptureQuit()
94:{
95:        CmtVipGrabStopInterrupts();
96:        CmtVipGrabRegisterIntFunction(NULL);
97:        CmtVipGrabSetRate(CMT_VIP_GRAB_OFF);
98:        CmtVipDisplaySetSt(0);
99:        CmtGenLibQuit();
100:       FreeMalloc(2);
101:}
102:
103:long CCapture::Addr(long x, long y)
104:{
105:       return ((x + y * 800)*2);
106:}
107:
108:int CCapture::ReadConfig(char *name)
109:{
110:       FILE *fp;
111:       char buff[128];
112:       char key[128];
113://     int sts;
114:       int cnt;
115:       int lines = 0;
116:
117:       m_Conf.grnScan.width = MARK_SCAN_WIDTH;
118:       m_Conf.grnScan.height = MARK_SCAN_HEIGHT;
119:       m_Conf.redScan.width = RED_MARK_SCAN_WIDTH;
120:       m_Conf.redScan.height = RED_MARK_SCAN_HEIGHT;
121:       m_Conf.nOfGreen = 0;
122:       m_Conf.nOfCenter = 0;
123:       m_Conf.nOfUpper = 0;
124:       m_Conf.nOfBottom = 0;
125:       m_Conf.eye.id = -1;
126:       m_Conf.eyeMove.width = EYE_AREA_MOVE_X;
127:       m_Conf.eyeMove.height = EYE_AREA_MOVE_Y;
128:
129:       if ((fp = fopen(name, "r")) == NULL) {
130:               return -1;
131:       }
132:       while (fgets(buff, 128, fp) != NULL) {
133:               lines++;
134:               if (*buff == '#') continue;
135:
136:               sscanf(buff, "%s", key);
```

```
137:                if (strcmp(key, "number") == 0) {
138:                        if (sscanf(buff, "%s %d %d", key, &m_Conf.nOfGreen, &m_Conf.nOfRed) != 3) {
139:                                fclose(fp);
140:                                return lines;
141:                        }
142:                } else
143:                if (strcmp(key, "scansize") == 0) {
144:                        if (sscanf(buff, "%s %d %d %d %d",
145:                                key,
146:                                &m_Conf.grnScan.width,
147:                                &m_Conf.grnScan.height,
148:                                &m_Conf.redScan.width,
149:                                &m_Conf.redScan.height ) != 5) { fclose(fp); return lines; }
150:                } else
151:                if (strcmp(key, "eye") == 0) {
152:                        if (sscanf(buff, "%s %d %d %d %d %d %d",
153:                                key,
154:                                &m_Conf.eye.id,
155:                                &m_Conf.eyeOpenID,
156:                                &m_Conf.eye.area.xl,
157:                                &m_Conf.eye.area.xr,
158:                                &m_Conf.eye.area.yb,
159:                                &m_Conf.eye.area.yt ) != 7) {fclose(fp); return lines;}
160:                        m_Conf.eye.area.attr = -1;
161:                        m_Conf.nOfRed = 1;
162:                } else
163:                if (strcmp(key, "eyeMove") == 0)  {
164:                        if (sscanf(buff, "%s %d %d",
165:                                key,
166:                                &m_Conf.eyeMove.width,
167:                                &m_Conf.eyeMove.height) != 3) {fclose(fp); return lines;}
168:                } else
169:                if (strcmp(key, "center") == 0) {
170:                        if (sscanf(buff, "%s %d", key, &cnt) != 2) {fclose(fp); return lines;}
171:                        if (readCenterLineMarkerConf(fp, cnt, &lines) != 0) {
172:                                FreeMalloc(0);
173:                                fclose(fp);
174:                                return lines;
175:                        }
176:                } else
177:                if (strcmp(key, "upper") == 0) {
178:                        if (sscanf(buff, "%s %d", key, &cnt) != 2) {
179:                                FreeMalloc(0);
180:                                fclose(fp);
181:                                return lines;
182:                        }
183:                        if (readBottomLineMarkerConf(fp, cnt, &lines) != 0) {
184:                                FreeMalloc(1);
185:                                fclose(fp);
186:                                return lines;
187:                        }
188:                } else
189:                if (strcmp(key, "bottom") == 0) {
190:                        if (sscanf(buff, "%s %d", key, &cnt) != 2) {
191:                                FreeMalloc(1);
192:                                fclose(fp);
193:                                return lines;
194:                        }
195:                        if (readUpperLineMarkerConf(fp, cnt, &lines)) {
196:                                FreeMalloc(2);
197:                                fclose(fp);
198:                                return lines;
199:                        }
200:                } else {fclose(fp); return lines;}
201:        }
202:        fclose(fp);
203:        return 0;
204:}
```

```
205:
206:int CCapture::readCenterLineMarkerConf(FILE* fp, int numb, int *lines)
207:{
208:        char buff[128];
209:        int result = -1;
210:        int n;
211:
212:        if (m_Conf.nOfCenter != 0 || numb == 0) return result;
213:        m_Conf.nOfCenter = numb;
214://      m_Conf.center = (MarkID *) malloc(sizeof(MarkID) * numb);
215:        m_Conf.center = new MarkID[numb];
216:        n = 0;
217:        while (fgets(buff, 128, fp) != NULL && result != 0) {
218:                (*lines)++;
219:                if (*buff == '#') continue;
220:                if (sscanf(buff, "%d %d %d %d %d %d",
221:                        &m_Conf.center[n].id,
222:                        &m_Conf.center[n].area.xl,
223:                        &m_Conf.center[n].area.xr,
224:                        &m_Conf.center[n].area.yb,
225:                        &m_Conf.center[n].area.yt,
226:                        &m_Conf.center[n].area.attr) != 6) return result;
227:                n++;
228:                if (n >= numb) {
229:                        result = 0;
230:                }
231:        }
232:        return result;
233:}
234:
235:int CCapture::readUpperLineMarkerConf(FILE *fp, int numb, int *lines)
236:{
237:        char buff[128];
238:        int result = -1;
239:        int n;
240:
241:        if (m_Conf.nOfUpper != 0 || numb == 0) return result;
242:        m_Conf.nOfUpper = numb;
243://      m_Conf.upper = (MarkID *)malloc(sizeof(MarkID) * numb);
244:        m_Conf.upper = new MarkID[numb];
245:        n = 0;
246:        while (fgets(buff, 128, fp) != NULL && result != 0) {
247:                (*lines)++;
248:                if (*buff == '#') continue;
249:                if (sscanf(buff, "%d %d %d %d %d %d",
250:                        &m_Conf.upper[n].id,
251:                        &m_Conf.upper[n].area.xl,
252:                        &m_Conf.upper[n].area.xr,
253:                        &m_Conf.upper[n].area.yb,
254:                        &m_Conf.upper[n].area.yt,
255:                        &m_Conf.upper[n].area.attr) != 6) return result;
256:                n++;
257:                if (n >= numb) {
258:                        result = 0;
259:                }
260:        }
261:        return result;
262:}
263:
264:int CCapture::readBottomLineMarkerConf(FILE *fp, int numb, int *lines)
265:{
266:        char buff[128];
267:        int result = -1;
268:        int n;
269:
270:        if (m_Conf.nOfBottom != 0 || numb == 0) return result;
271:        m_Conf.nOfBottom = numb;
272://      m_Conf.bottom = (MarkID *) malloc(sizeof(MarkID) * numb);
```

```
273:        m_Conf.bottom = new MarkID[numb];
274:        n = 0;
275:        while (fgets(buff, 128, fp) != NULL && result != 0) {
276:                (*lines)++;
277:                if (*buff == '#') continue;
278:                if (sscanf(buff, "%d %d %d %d %d %d",
279:                        &m_Conf.bottom[n].id,
280:                        &m_Conf.bottom[n].area.xl,
281:                        &m_Conf.bottom[n].area.xr,
282:                        &m_Conf.bottom[n].area.yb,
283:                        &m_Conf.bottom[n].area.yt,
284:                        &m_Conf.bottom[n].area.attr) != 6) return result;
285:                n++;
286:                if (n >= numb) {
287:                        result = 0;
288:                }
289:        }
290:        return result;
291:}
292:
293:void CCapture::FreeMalloc(int x)
294:{
295:        switch(x) {
296:                case 2:
297://                      free(m_Conf.bottom);
298:                        delete m_Conf.bottom;
299:                case 1:
300://                     free(m_Conf.upper);
301:                        delete m_Conf.upper;
302:                case 0:
303://                     free(m_Conf.center);
304:                        delete m_Conf.center;
305:                default:
306:                break;
307:        }
308:}
309:
310:// 探索範囲の緑ﾌﾟﾚｰﾝがONのﾄﾞｯﾄ群をﾗﾍﾞﾙとして、その重心と面積を返す
311:
312:void CCapture::ScanMarker(int sx, int sy, int *rx, int *ry, long *rsize, int sW, int sH)
313:{
314:        long m00, m01, m10;
315:        int x, y;
316:        int xt, yt;
317:
318:        m00 = m01 = m10 = 0;
319:        for (x = 0; x < sW; x++) {
320:                xt = sx - x;
321:                if (xt > GRAB_WIN_X) break;
322:                for (y = -sH/2; y <= sH/2; y++) {
323:                        yt = sy + y;
324:                        if (yt >= 0 && yt < GRAB_WIN_Y) {
325:                                if (Bufpix(xt, yt) > MARK_LEVEL) {
326:                                        m01 -= x;
327:                                        m10 += y;
328:                                        m00++;
329:                                }
330:                                Putpix(xt, yt, 0x0000);
331:                        }
332:                }
333:        }
334:        *rx = (int) (m01/m00 + sx);
335:        *ry = (int) (m10/m00 + sy);
336:        *rsize = m00;
337:}
338:
339:// 画面上のﾏｰｶの座標を返す (順不同)
340:// redMarksは視線追跡拡張時のためのﾀﾞﾐｰﾊﾟﾗﾒｰﾀ
```

```
341:
342:int CCapture::ScanMarkerPosition(Mark marks[], Mark redMarks[])
343:{
344:        int x, y;
345:        int gx, gy;
346:        long gsize;
347:        int result = -1;
348:        int cnt, redCnt;
349:
350:        CmtVipGrabSetRate(CMT_VIP_GRAB_OFF);
351://      GetRect();
352://      CmtVipGrabSingle();
353:        cnt = 0;
354:        redCnt = 0;
355:
356:        for (x = GRAB_WIN_X - RIGHT_MARGIN; x >= LEFT_MARGIN; x--) {
357:                for (y = TOP_MARGIN; y < GRAB_WIN_Y - BOTTOM_MARGIN; y++) {
358:                        if (Bufpix(x, y) > MARK_LEVEL && cnt < m_Conf.nOfGreen) {
359:                                //Putpix(x, y, 0x7c00);
360:                                ScanMarker(x, y, &gx, &gy, &gsize, m_Conf.grnScan.width, m_Conf.grnScan.height);
361:                                if ((gsize >= REGION_RATIO * m_Conf.grnScan.width * m_Conf.grnScan.height) &&
362:                                        (gsize <= 0.80 * m_Conf.grnScan.width * m_Conf.grnScan.height)) {
363:                                        marks[cnt].x = gx;
364:                                        marks[cnt].y = gy;
365:                                        marks[cnt].area = gsize;
366:                                        cnt++;
367:                                }
368:                        }
369:                }
370:        }
371://      SetRect();
372://      if (cnt == m_Conf.nOfGreen) result = 0;
373://      return result;
374:        CmtVipGrabSetRate(CMT_VIP_GRAB_CONTINUOUS);
375:        return cnt;
376:}
377:
378:// 指定のﾏｰｶの移動先を調べる
379:
380:void CCapture::ScanMarkMove(Area area, Mark orgPos, Mark * pos)
381:{
382:        int xs, ys;
383:        int x, y;
384:        long m00, m01, m10;
385:        int orgx, orgy;
386:        short tmp;
387:        int tmp_x, tmp_y;
388:
389:        xs = area.xr - area.xl + 1;
390:        ys = area.yt - area.yb + 1;
391:
392:        tmp_x = pos->x;
393:        tmp_y = pos->y;
394:        if (area.attr != 1) {
395:                orgx = pos->x;
396:                orgy = pos->y;
397:        } else {
398:                orgx = orgPos.x;
399:                orgy = orgPos.y;
400:        }
401:        CmtGenBlockRead(Addr(orgx+area.xl, orgy+area.yb), xs, ys, (unsigned char*)Buffer, sizeof(short)*xs*ys);
402:
403:        m00 = m01 = m10 = 0;
404:        if (area.attr >= 0) {
405:                for (y = 0; y < ys; y++) {
406:                        for (x = 0; x < xs; x++) {
407:                                tmp = Buffer[x + y * xs] & 0x7fff;
408:                                if (tmp > MARK_LEVEL) {
```

```
409:                                    m01 += x;
410:                                    m10 += y;
411:                                    m00++;
412:                                }
413:                            }
414:                    }
415:            }
416:            pos->area = m00;
417:            if (m00 > 0) {
418:                    pos->x = (int)((m01/m00) + (orgx + area.xl));
419://                  if (abs(pos->x - tmp_x) < 3) pos->x = tmp_x;
420:                    pos->y = (int)((m10/m00) + (orgy + area.yb));
421://                  if (abs(pos->y - tmp_y) < 3) pos->y = tmp_y;
422:            }
423:}
424:
425:int CCapture::SearchNearlyMarker(int px, int py, Mark *mark, int mn)
426:{
427:            int i, t = -1;
428:            double xl, yl, l, len = HUGE;
429:
430:            for (i=0;i < mn; i++) {
431:                    xl = px - mark[i].x;
432:                    yl = py - mark[i].y;
433:                    l = sqrt(xl*xl + yl*yl);
434:                    if (len>l) {
435:                            len = l;
436:                            t = i;
437:                    }
438:            }
439:
440:            return t;
441:}
442:
443:short CCapture::Bufpix(int x, int y)
444:{
445:            short buf[2];
446:            CmtGenBlockRead(Addr(x, y), 1, 1, (unsigned char*)buf, sizeof(short)*2);
447://          buf[0] = Frame[x + y * GRAB_WIN_Y];
448:            buf[0] &= 0x7fff;
449:            return buf[0];
450:}
451:
452:void CCapture::Putpix(int x, int y, short src)
453:{
454:            short buf[2];
455:            buf[0] = src;
456:            CmtGenBlockWrite(Addr(x, y), 1, 1, (unsigned char*)buf, 2);
457:}
458:
459:void CCapture::CaptureStop()
460:{
461:            CmtVipGrabSetRate(CMT_VIP_GRAB_OFF);
462:}
463:
464:void CCapture::CaptureStart()
465:{
466:            CmtVipGrabSetRate(CMT_VIP_GRAB_CONTINUOUS);
467:}
```

```
// GRABDLG.CPP : implementation file
//

#include "stdafx.h"
#include "mtrace.h"
#include "trace.h"
#include "GRABDLG.H"

#include "capture.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

#define WM_DLGMSG WM_USER+301

/////////////////////////////////////////////////////////////////////////////
// CGrabDlg dialog


CGrabDlg::CGrabDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CGrabDlg::IDD, pParent)
{
	m_hParent = pParent;
	//{{AFX_DATA_INIT(CGrabDlg)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
}


CGrabDlg::CGrabDlg(CCapture* pCapture) : CDialog()
{
	m_pCapture = pCapture;
}


BOOL CGrabDlg::Create()
{
	return CDialog::Create(CGrabDlg::IDD);
}

void CGrabDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CGrabDlg)
		// NOTE: the ClassWizard will add DDX and DDV calls here
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CGrabDlg, CDialog)
	//{{AFX_MSG_MAP(CGrabDlg)
//	ON_WM_LBUTTONDOWN()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()


//
void CGrabDlg::SetView(CTraceView* pView)
{
	m_pView = pView;
}
/////////////////////////////////////////////////////////////////////////////
// CGrabDlg message handlers

void CGrabDlg::OnLButtonDown(UINT nFlags, CPoint point)
{
	if (point.x < 320 && point.y < 240)
		::SendMessage(m_hParent->m_hWnd, WM_DLGMSG, (UINT)point.x, (LONG)point.y);
}

```

```
 1:// mainfrm.cpp : implementation of the CMainFrame class
 2://
 3:
 4:#include "stdafx.h"
 5:#include "trace.h"
 6:
 7:#include "mainfrm.h"
 8:
 9:#ifdef _DEBUG
10:#undef THIS_FILE
11:static char BASED_CODE THIS_FILE[] = __FILE__;
12:#endif
13:
14://///////////////////////////////////////////////////////////////////////////
15:// CMainFrame
16:
17:IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)
18:
19:BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
20:        //{{AFX_MSG_MAP(CMainFrame)
21:        //}}AFX_MSG_MAP
22:END_MESSAGE_MAP()
23:
24://///////////////////////////////////////////////////////////////////////////
25:// CMainFrame construction/destruction
26:
27:CMainFrame::CMainFrame()
28:{
29:        // TODO: add member initialization code here
30:}
31:
32:CMainFrame::~CMainFrame()
33:{
34:}
35:
36://///////////////////////////////////////////////////////////////////////////
37:// CMainFrame diagnostics
38:
39:#ifdef _DEBUG
40:void CMainFrame::AssertValid() const
41:{
42:        CFrameWnd::AssertValid();
43:}
44:
45:void CMainFrame::Dump(CDumpContext& dc) const
46:{
47:        CFrameWnd::Dump(dc);
48:}
49:
50:#endif //_DEBUG
51://///////////////////////////////////////////////////////////////////////////
52:// CMainFrame member functions
53:
54:BOOL CMainFrame::PreCreateWindow(CREATESTRUCT &cs)
55:{
56:        CFrameWnd::PreCreateWindow(cs);
57:
58:        TEXTMETRIC tm;
59:        CDC* pDC = GetDC();
60:        pDC->GetTextMetrics(&tm);
61:        ReleaseDC(pDC);
62:
63:        cs.lpszName = "表情解析";
64:        cs.style = WS_OVERLAPPED | WS_SYSMENU;
65:        cs.cx = 800;
66:        cs.cy = 120;
67:        cs.x = 0;
68:        cs.y = 480;
```

```
69:         return TRUE;
70:}
71:
72://////////////////////////////////////////////////////////////////////////////
73:// CMainFrame message handlers
74:
```

```
// serial.cpp : implementation of the CSerial class

#include "stdafx.h"
#include "serial.h"

/////////////////////////////////////////////////////////////////////////////
// CSerial

IMPLEMENT_DYNCREATE(CSerial, CObject)

/////////////////////////////////////////////////////////////////////////////
// CCerial construction

CSerial::CSerial()
{
}

CSerial::CSerial(HWND hwnd)
{
        m_hSWnd = hwnd;
}

/////////////////////////////////////////////////////////////////////////////
// CSerial member functions

// 通信ﾃﾞﾊﾞｲｽの初期化

int CSerial::InitSerial(int port, int baud, int parity, int length, int stop, int qs, int qr)
{
        char *npt[] = {"COM1", "COM2", "LPT1", "LPT2"};
        char *bud[] = {"1200", "2400", "4800", "9600", "19200"};
        char *lpt[] = {"n", "o", "e"};
        char *len[] = {"4", "5", "6", "7", "8"};
        char *stb[] = {"1", "1.5", "2"};
        char buf[128];

        if (port   < 0 || port   > 3) port   = 0;
        if (baud   < 0 || baud   > 4) baud   = 3;
        if (parity < 0 || parity > 2) parity = 0;
        if (length < 0 || length > 4) length = 4;
        if (stop   < 0 || stop   > 2) stop   = 0;

        if (qr < 128) qr = 128;
        if (qs < 128) qs = 128;

        // 通信ﾃﾞﾊﾞｲｽｵｰﾌﾟﾝ
        if ((m_idComDev = OpenComm(npt[port], qr, qs)) < 0) return FALSE;

        // 文字列をｼﾘｱﾙDCBに変換
        sprintf(buf, "%s:%s, %s, %s, %s", npt[port], bud[baud], lpt[parity], len[length], stb[stop]);
        if (BuildCommDCB(buf, &m_dcb) < 0) return FALSE;

        // DCBの状態に通信ﾃﾞﾊﾞｲｽを設定
        if (SetCommState(&m_dcb) < 0) return FALSE;

        // 初期値を保存する
        m_Port = port;
        m_Baud = baud;
        m_Parity = parity;
        m_Length = length;
        m_Stop = stop;
        m_Qr = qr;
        m_Qs = qs;

        return TRUE;
}

// 通信ﾃﾞﾊﾞｲｽの設定を変更する
```

```
 69:
 70:int CSerial::ChangeStatus(int baud, int parity, int length, int stop)
 71:{
 72:        char *npt[] = {"COM1", "COM2", "LPT1", "LPT2"};
 73:        char *bud[] = {"1200", "2400", "4800", "9600", "19200"};
 74:        char *lpt[] = {"n", "o", "e"};
 75:        char *len[] = {"4", "5", "6", "7", "8"};
 76:        char *stb[] = {"1", "1.5", "2"};
 77:        char buf[128];
 78:
 79:        if (baud   == CS_NOW) baud   = m_Baud;
 80:        if (parity == CS_NOW) parity = m_Parity;
 81:        if (length == CS_NOW) length = m_Length;
 82:        if (stop   == CS_NOW) stop   = m_Stop;
 83:
 84:        if (baud   < 0 || baud   > 4) baud   = 3;
 85:        if (parity < 0 || parity > 2) parity = 0;
 86:        if (length < 0 || length > 4) length = 4;
 87:        if (stop   < 0 || stop   > 2) stop   = 0;
 88:
 89:        // 文字列をｼﾘｱﾙDCBに変換
 90:        sprintf(buf, "%s:%s, %s, %s, %s", npt[m_Port], bud[baud], lpt[parity], len[length], stb[stop]);
 91:        if (BuildCommDCB(buf, &m_dcb) < 0) return FALSE;
 92:
 93:        // DCBの状態に通信ﾃﾞﾊﾞｲｽを設定
 94:        if (SetCommState(&m_dcb) < 0) return FALSE;
 95:
 96:        // 変更を保存する
 97:        m_Baud = baud;
 98:        m_Parity = parity;
 99:        m_Length = length;
100:        m_Stop = stop;
101:
102:        return TRUE;
103:}
104:
105:// 通信ﾃﾞﾊﾞｲｽをﾌﾞﾚｰｸ状態にする
106:
107:int CSerial::SetBreak()
108:{
109:        if (SetCommBreak(m_idComDev) == 0) {
110:                return TRUE;
111:        } else {
112:                return FALSE;
113:        }
114:}
115:
116:// 通信ﾃﾞﾊﾞｲｽのﾌﾞﾚｰｸ状態を解除する
117:
118:int CSerial::ClearBreak()
119:{
120:        if (ClearCommBreak(m_idComDev) == 0) {
121:                return TRUE;
122:        } else {
123:                return FALSE;
124:        }
125:}
126:
127:// 通信ﾃﾞﾊﾞｲｽをｸﾛｰｽﾞする
128:
129:int CSerial::CloseSerial()
130:{
131:        if (CloseComm(m_idComDev) == 0) {
132:                return TRUE;
133:        } else {
134:                return FALSE;
135:        }
136:}
```

```
137:
138:// 通信ﾃﾞﾊﾞｲｽの送信/受信ｷｭｰをﾌﾗｯｼｭする
139:
140:int CSerial::FlushQueue(int queue)
141:{
142:        if (queue < 0 || queue > 2) queue = 2;
143:
144:        // queue が2のときは送受信ともにﾌﾗｯｼｭする
145:        if (queue == 0 || queue == 2) {
146:                if (FlushComm(m_idComDev, 0) != 0) return FALSE;
147:        }
148:        if (queue == 1 || queue == 2) {
149:                if (FlushComm(m_idComDev, 1) != 0) return FALSE;
150:        }
151:        return TRUE;
152:}
153:
154:// WM_COMMNOTIFYﾒｯｾｰｼﾞのﾎﾟｽﾄの許可、不許可
155:
156:int CSerial::EnableNotifyMsg(int wNot, int oNot)
157:{
158:        if (!(EnableCommNotification(m_idComDev, m_hSWnd, wNot, oNot))) {
159:                return TRUE;
160:        } else {
161:                return FALSE;
162:        }
163:}
164:
165:// 通信ﾃﾞﾊﾞｲｽの最近のｴﾗｰ値を取得する
166:
167:int CSerial::GetLastError()
168:{
169:        return GetCommError(m_idComDev, NULL);
170:}
171:
172:// 通信ﾃﾞﾊﾞｲｽの状態を調べる
173:
174:int CSerial::GetSerialStatus(int queue)
175:{
176:        COMSTAT stat;
177:
178:        GetCommError(m_idComDev, &stat);
179:        switch (queue) {
180:                case CS_QUE_BOTH:
181:                        return (int) stat.status;
182:                case CS_QUE_RECEIVE: // 受信ｷｭｰの文字数を調べる
183:                        return (int) stat.cbInQue;
184:                case CS_QUE_SEND: // 送信ｷｭｰの文字数を調べる
185:                        return (int) stat.cbOutQue;
186:                default:
187:                        return -1;
188:                        break;
189:        }
190:}
191:
192:// ﾃﾞｰﾀ読み取り
193:
194:int CSerial::Read(unsigned char *buf, int num)
195:{
196:        if (num < 1) num = 1;
197:        return ReadComm(m_idComDev,(unsigned char*) buf, num);
198:}
199:
200:// ﾃﾞｰﾀ書き込み
201:
202:int CSerial::Write(unsigned char *buf, int num)
203:{
204:        int mem;
```

```
205:
206:        if (num < 1) num = 1;
207:        mem = GetSerialStatus(CS_QUE_SEND);     // 送信ｷｭｰの文字数を得る
208:        if ((m_Qs - mem) < num) return -1; // 送信ｷｭｰに空き容量はあるか
209:        return WriteComm(m_idComDev, buf, num);
210:}
211:
```

```
// SOCKDLG.CPP : implementation file
//

#include "stdafx.h"
#include "trace.h"
#include "SOCKDLG.H"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSockDlg dialog


CSockDlg::CSockDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CSockDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CSockDlg)
        m_Name = _T("ciris7");
        //}}AFX_DATA_INIT
}


void CSockDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CSockDlg)
        DDX_Control(pDX, IDOK, m_Ok);
        DDX_Control(pDX, IDCANCEL, m_Cancel);
        DDX_CBString(pDX, IDC_SERVER, m_Name);
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CSockDlg, CDialog)
        //{{AFX_MSG_MAP(CSockDlg)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CSockDlg message handlers


```

```
 1:// socket.cpp : implementation of the CSocket class
 2://
 3:
 4:#include <stdio.h>
 5:#include "stdafx.h"
 6:
 7:#include "socket.h"
 8:#include "c:¥winsock¥winsock.h"
 9:
10:#define SEND_SIZE 128 // 送受信ﾃｰﾌﾞﾙの大きさ
11:
12:
13://///////////////////////////////////////////////////////////////////////////
14:// CSocket
15:
16:IMPLEMENT_DYNCREATE(CSocket, CObject)
17:
18://///////////////////////////////////////////////////////////////////////////
19:// CSocket construction
20:
21:CSocket::CSocket()
22:{
23:        m_Socket = 0;
24:        m_SrvName = "";
25:}
26:
27:CSocket::~CSocket()
28:{
29:        SShutdown();
30:        m_Socket = 0;
31:        m_SrvName = "";
32:}
33://///////////////////////////////////////////////////////////////////////////
34:// CSocket member functions
35:
36:int CSocket::InitSocket(void)
37:{
38:        int             result = 0;
39:        WORD    wVersionRequested;
40:        WSADATA wsaData;
41:
42:        wVersionRequested = (unsigned short)MAKEWORD(1, 1);
43:        if (WSAStartup(wVersionRequested, &wsaData) != 0) {
44:                result = -1;
45:        }
46:        return result;
47:}
48:
49:// ｻｰﾊﾞと接続する
50:
51:int CSocket::Connect()
52:{
53:        struct protoent FAR *pp;
54:        struct hostent FAR *host;
55:        struct sockaddr_in sin;
56:        int err;
57:
58:        if ( (err = InitSocket()) < 0 )         return err;
59:        // ｿｹｯﾄを生成する
60:        pp = getprotobyname("tcp");
61:        if ((m_Socket = socket(AF_INET, SOCK_STREAM, pp->p_proto)) < 0) {
62:                return -1;
63:        }
64:
65:        if ((host = gethostbyname(m_SrvName)) == NULL) {
66:                closesocket(m_Socket);  m_Socket = 0;
67:                WSACleanup();
68:                return -1;
```

```
69:         }
70:         _fmemset((char *)&sin, 0, sizeof(sin));
71:         sin.sin_family = AF_INET;
72:         sin.sin_port = htons(7000);
73:         _fmemcpy(&sin.sin_addr, host->h_addr, host->h_length);
74:
75:         // ｻｰﾊﾞと接続する
76:         if ((err = connect(m_Socket, (SOCKADDR *)&sin, sizeof(sin))) != 0) {
77:                 shutdown(m_Socket, 2);
78:                 closesocket(m_Socket);    m_Socket = 0;
79:                 WSACleanup();
80:                 return -1;
81:         }
82:         return m_Socket;
83:}
84:
85:// ｼｬｯﾄﾀﾞｳﾝを行う
86:
87:void CSocket::SShutdown()
88:{
89:         if ( m_Socket != 0 ) {
90:                 shutdown(m_Socket, 2); // ｻｰﾊﾞとの接続を切る
91:                 closesocket(m_Socket); // ｿｹｯﾄを閉じる
92:                 WSACleanup();
93:         }
94:         m_Socket = 0;
95:}
96:
97:// ｻｰﾊﾞ名を変更する
98:
99:void CSocket::SetSrvName(CString buf)
100:{
101:         if (strcmp(buf, "") != 0)
102:                 m_SrvName = buf;
103:}
104:
105:// 現在のｻｰﾊﾞ名を得る
106:
107:CString CSocket::GetSrvName()
108:{
109:         return m_SrvName;
110:}
111:
112:// ｿｹｯﾄﾃﾞｨｽｸﾘﾌﾟﾀの変更
113:
114:void CSocket::SetSocket(int s)
115:{
116:         if (s > 0)
117:                 m_Socket = s;
118:}
119:
120:// 現在のｿｹｯﾄﾃﾞｨｽｸﾘﾌﾟﾀを得る
121:
122:int CSocket::GetSocket()
123:{
124:         return m_Socket;
125:}
126:
127:// ﾊﾞｯﾌｧの内容を送信する
128:
129:void CSocket::Send(short Pkt_Tbl[])
130:{
131:         static unsigned short pk[SEND_SIZE];
132:
133:         int     psize = sizeof(short)*SEND_SIZE;
134:         int     len,i;
135:         if ( Pkt_Tbl == NULL )  return;
136:         for (i =0;i<SEND_SIZE;i++) pk[i] = htons((unsigned short)Pkt_Tbl[i]);
```

```
137:
138:        len = send(m_Socket, (char *)pk, psize, 0);
139:}
140:
141:// 受信ﾃﾞｰﾀの先頭を返す
142:
143:short CSocket::Receive(int *len)
144:{
145:        static unsigned short tmp[SEND_SIZE];
146:
147:        *len = recv(m_Socket, (char *)tmp, sizeof(short)*SEND_SIZE, 0);
148:        return ntohs(tmp[0]);
149:}
150:
151:// ｻｰﾊﾞからの送信要求時にｳｨﾝﾄﾞｳｽﾞﾒｯｾｰｼﾞを発生させる
152:
153:int CSocket::StartInterrupt(HWND hWnd)
154:{
155:        if ( m_Socket > 0 ) {
156:                if (WSAAsyncSelect(m_Socket, hWnd, WM_WINSOCK, FD_READ | FD_CLOSE) != 0) {
157:                        return WSAGetLastError();
158:                }
159:        }
160:        return 0;
161:}
162:
163:int CSocket::StopInterrupt(HWND hWnd)
164:{
165:        if (m_Socket > 0 ) {
166:                if (WSAAsyncSelect(m_Socket, hWnd, 0, 0) != 0 ) {
167:                        return WSAGetLastError();
168:                }
169:        }
170:        return 0;
171:}
172:
```

```
1:// stdafx.cpp : source file that includes just the standard includes
2://  stdafx.pch will be the pre-compiled header
3://  stdafx.obj will contain the pre-compiled type information
4:
5:#include "stdafx.h"
6:
```

```
// trace.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "mtrace.h"
#include "trace.h"

#include "mainfrm.h"
#include "tracedoc.h"
#include "tracevw.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CTraceApp

BEGIN_MESSAGE_MAP(CTraceApp, CWinApp)
        //{{AFX_MSG_MAP(CTraceApp)
        ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG_MAP
        // Standard file based document commands
//      ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
//      ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CTraceApp construction

CTraceApp::CTraceApp()
{
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CTraceApp object

CTraceApp NEAR theApp;

/////////////////////////////////////////////////////////////////////////////
// CTraceApp initialization

BOOL CTraceApp::InitInstance()
{
        // Standard initialization
        // If you are not using these features and wish to reduce the size
        //  of your final executable, you should remove from the following
        //  the specific initialization routines you do not need.

        SetDialogBkColor();        // Set dialog background color to gray
        LoadStdProfileSettings();  // Load standard INI file options (including MRU)

        // Register the application's document templates.  Document templates
        //  serve as the connection between documents, frame windows and views.

        CSingleDocTemplate* pDocTemplate;
        pDocTemplate = new CSingleDocTemplate(
                IDR_MAINFRAME,
                RUNTIME_CLASS(CTraceDoc),
                RUNTIME_CLASS(CMainFrame),      // main SDI frame window
                RUNTIME_CLASS(CTraceView));
        AddDocTemplate(pDocTemplate);

```

```
69:         // create a new (empty) document
70:         OnFileNew();
71:
72:         if (m_lpCmdLine[0] != '¥0')
73:         {
74:                 // TODO: add command line processing here
75:         }
76:
77:
78:         return TRUE;
79:}
80:
81://///////////////////////////////////////////////////////////////////////////
82:// CAboutDlg dialog used for App About
83:
84:class CAboutDlg : public CDialog
85:{
86:public:
87:         CAboutDlg();
88:
89:// Dialog Data
90:         //{{AFX_DATA(CAboutDlg)
91:         enum { IDD = IDD_ABOUTBOX };
92:         //}}AFX_DATA
93:
94:// Implementation
95:protected:
96:         virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
97:         //{{AFX_MSG(CAboutDlg)
98:                 // No message handlers
99:         //}}AFX_MSG
100:        DECLARE_MESSAGE_MAP()
101:};
102:
103:CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
104:{
105:        //{{AFX_DATA_INIT(CAboutDlg)
106:        //}}AFX_DATA_INIT
107:}
108:
109:void CAboutDlg::DoDataExchange(CDataExchange* pDX)
110:{
111:        CDialog::DoDataExchange(pDX);
112:        //{{AFX_DATA_MAP(CAboutDlg)
113:        //}}AFX_DATA_MAP
114:}
115:
116:BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
117:        //{{AFX_MSG_MAP(CAboutDlg)
118:                // No message handlers
119:        //}}AFX_MSG_MAP
120:END_MESSAGE_MAP()
121:
122:// App command to run the dialog
123:void CTraceApp::OnAppAbout()
124:{
125:        CAboutDlg aboutDlg;
126:        aboutDlg.DoModal();
127:}
128:
129://///////////////////////////////////////////////////////////////////////////
130:// CTraceApp commands
131:
```

```
// tracedoc.cpp : implementation of the CTraceDoc class
//

#include "stdafx.h"
#include "trace.h"

#include "tracedoc.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CTraceDoc

IMPLEMENT_DYNCREATE(CTraceDoc, CDocument)

BEGIN_MESSAGE_MAP(CTraceDoc, CDocument)
	//{{AFX_MSG_MAP(CTraceDoc)
		// NOTE - the ClassWizard will add and remove mapping macros here.
		//    DO NOT EDIT what you see in these blocks of generated code!
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CTraceDoc construction/destruction

CTraceDoc::CTraceDoc()
{
	// TODO: add one-time construction code here

}

CTraceDoc::~CTraceDoc()
{
}

BOOL CTraceDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;

	// TODO: add reinitialization code here
	// (SDI documents will reuse this document)

	return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CTraceDoc serialization

void CTraceDoc::Serialize(CArchive& ar)
{
	if (ar.IsStoring())
	{
		// TODO: add storing code here
	}
	else
	{
		// TODO: add loading code here
	}
}

/////////////////////////////////////////////////////////////////////////////
// CTraceDoc diagnostics

#ifdef _DEBUG
void CTraceDoc::AssertValid() const
```

```
69:{
70:         CDocument::AssertValid();
71:}
72:
73:void CTraceDoc::Dump(CDumpContext& dc) const
74:{
75:         CDocument::Dump(dc);
76:}
77:#endif //_DEBUG
78:
79://///////////////////////////////////////////////////////////////////////////
80:// CTraceDoc commands
81:
```

```
// tracevw.cpp : implementation of the CTraceView class
//

#include "stdafx.h"
#include "mtrace.h"
#include "trace.h"

#include "tracedoc.h"
#include "tracevw.h"

#include "socket.h"                                 // ｿｹｯﾄ通信用ｸﾗｽ
#include "c:¥winsock¥winsock.h" // Winsock ﾗｲﾌﾞﾗﾘ
#include "capture.h"                        // ﾏｰｶ追跡に関するｸﾗｽ
#include "grabdlg.h"                        // 入力画像表示用ﾀﾞｲｱﾛｸﾞ
#include "sockdlg.h"                        // ｿｹｯﾄに関する設定用ﾀﾞｲｱﾛｸﾞ
#include "area.h"                                   // DCTｴﾘｱ定義ﾌｧｲﾙ選択用ﾀﾞｲｱﾛｸﾞ

#include "grabint.h"                        // ｸﾞﾗﾌﾞ割込み関数
#include "marksort.h"                       // ﾏｰｶｿｰﾄ用関数

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

#define WM_DLGMSG WM_USER+301

extern CTraceApp near theApp;
/////////////////////////////////////////////////////////////////////////////
// CTraceView

IMPLEMENT_DYNCREATE(CTraceView, CView)

BEGIN_MESSAGE_MAP(CTraceView, CView)
        //{{AFX_MSG_MAP(CTraceView)
        ON_COMMAND(ID_MENUSTART, OnMenustart)
        ON_COMMAND(ID_END, OnEnd)
        ON_WM_TIMER()
        ON_COMMAND(ID_QUIT, OnQuit)
        ON_MESSAGE(WM_WINSOCK, OnWinsock)
        ON_COMMAND(ID_RETRY, OnRetry)
        ON_COMMAND(ID_MENUITEM32774, OnSetSock)
        ON_WM_DESTROY()
        ON_COMMAND(ID_MENUITEM32786, OnMenuArea)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CTraceView construction/destruction

CTraceView::CTraceView()
{
        phBitmap = new CBitmap();
        phBrush  = new CBrush();
        phPen    = new CPen(PS_SOLID, 1, RGB(255, 0, 0));
        m_SrvName = "ciris1";
        m_FFlag = TRUE;
        m_QFlag = FALSE;
        m_pDC  = NULL;
        m_tid  = -1;
        m_pDlg = new CGrabDlg(this);
        m_pCapture = new CCapture();
        // ｷｬﾌﾟﾁｬ画像表示用ﾓｰﾄﾞﾚｽﾀﾞｲｱﾛｸﾞ(移動不可)
        if (m_pDlg->GetSafeHwnd() == 0) {
                m_pDlg->Create();
        }
        m_pDlg->MoveWindow(0, 0, GRAB_WIN_X, GRAB_WIN_Y, TRUE);
        phBrush->CreateSolidBrush(RGB(255, 0, 0)); // 赤のﾌﾞﾗｼ
```

```
 69:
 70:       m_pCapture->CaptureInit();
 71:}
 72:
 73:CTraceView::~CTraceView()
 74:{
 75:       if ( m_tid > 0 ) {
 76:             KillTimer(m_tid);       m_tid = -1;
 77:       }
 78:       delete m_pCapture;
 79:       m_pDlg->DestroyWindow();
 80:       RedrawWindow(NULL, NULL, RDW_ERASE | RDW_INVALIDATE);
 81:       delete m_pDlg;
 82:}
 83:
 84://///////////////////////////////////////////////////////////////////////////
 85:// CTraceView drawing
 86:
 87:void CTraceView::OnDraw(CDC* pDC)
 88:{
 89:       CTraceDoc* pDoc = GetDocument();
 90:       ASSERT_VALID(pDoc);
 91:}
 92:
 93://///////////////////////////////////////////////////////////////////////////
 94:// CTraceView diagnostics
 95:
 96:#ifdef _DEBUG
 97:void CTraceView::AssertValid() const
 98:{
 99:       CView::AssertValid();
100:}
101:
102:void CTraceView::Dump(CDumpContext& dc) const
103:{
104:       CView::Dump(dc);
105:}
106:
107:CTraceDoc* CTraceView::GetDocument() // non-debug version is inline
108:{
109:       ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CTraceDoc)));
110:       return (CTraceDoc*)m_pDocument;
111:}
112:#endif //_DEBUG
113:
114://///////////////////////////////////////////////////////////////////////////
115:// Member functions
116:// ｴﾗｰ表示用ﾒｯｾｰｼﾞﾎﾞｯｸｽ
117:
118:void CTraceView::PutErrorMsg(int err, int Param1, CString Param2)
119:{
120:       char buf[80];
121:
122:       switch (err) {
123:             case ERR_CONFIG: // configﾌｧｲﾙｴﾗｰ
124:                   if (Param1 == -1)
125:                         sprintf(buf, "Cannot open config file '%s'", Param2);
126:                   else
127:                         sprintf(buf, "Config file error : line %d", Param1);
128:                   break;
129:             case ERR_SOCKET: // ｿｹｯﾄｺﾈｸﾄｴﾗｰ
130:                   sprintf(buf, "Cannot connect with the server '%s'", Param2);
131:                   break;
132:             case ERR_MARKER: // ﾏｰｶ検出ｴﾗｰ
133:                   sprintf(buf, "Found only %d marker(s)", Param1);
134:                   break;
135:             case ERR_MEMORY: // ﾒﾓﾘ確保ｴﾗｰ
136:                   sprintf(buf, "Not enough memory");
```

```
137:                        break;
138:                default:
139:                        return;
140:        }
141:        MessageBox(buf, NULL, MB_OK);
142:}
143://
144://
145://
146:#include <iostream.h>
147:#include <fstream.h>
148:
149:void CTraceView::GetInitValue(void)
150:{
151:        MSG lpmsg;                              // Windows ﾒｯｾｰｼﾞ処理用
152:        int i, j, cnt;
153:        const int INIT_FRAME_COUNT = 15;
154:
155:// ofstream file;
156:
157:        for ( i = 0; i < m_pCapture->m_aNum; i++ ) {
158:                for ( j = 0; j < 3; j++ ) m_InitBuf[i*3+j] = 0;
159:        }
160:        cnt = 0;
161:        while ((lpmsg.message != WM_DESTROY) && (cnt < INIT_FRAME_COUNT) && (m_FFlag == FALSE)) {
162:                if (PeekMessage(&lpmsg, NULL, 0, 0, PM_REMOVE) != 0) { // ﾒｯｾｰｼﾞｷｭｰをpeek
163:                        TranslateMessage(&lpmsg);
164:                        DispatchMessage(&lpmsg);
165:                } else {
166:                        if (GrabFlag == TRUE) { // grab割込みが発生したかどうか
167:                                GrabFlag = FALSE;
168:                                m_CFlag  = FALSE;
169:                                m_pCapture->Dct();
170:                                for (i = 0; i < m_pCapture->m_aNum; i++) {
171:                                        m_InitBuf[i*3+0] += (short)m_pCapture->m_GH[i];
172:                                        m_InitBuf[i*3+1] += (short)m_pCapture->m_GV[i];
173:                                        m_InitBuf[i*3+2] += (short)m_pCapture->m_GHV[i];
174:                                }
175:                                cnt++;
176:                        }
177:                }
178:        }
179:
180:        if ( m_FFlag == FALSE && lpmsg.message != WM_DESTROY && cnt > 0) {
181:// file.open("init.log",ios::out);
182:
183:                for (i = 0; i < m_pCapture->m_aNum; i++) {
184:                        m_InitBuf[i*3+0] /= cnt;
185:                        m_InitBuf[i*3+1] /= cnt;
186:                        m_InitBuf[i*3+2] /= cnt;
187:// file << m_InitBuf[i*3+0] << " " << m_InitBuf[i*3+1] << " " << m_InitBuf[i*3+2] << endl;
188:                }
189:// file.close();
190:        }
191:}
192:
193://////////////////////////////////////////////////////////////////////////////
194:// CTraceView message handler
195:void CTraceView::OnMenustart()
196:{
197:        static char str[128];                   // 文字列処理用
198://      CClientDC nDC(this);    // ﾒｲﾝﾌﾚｰﾑﾋﾞｭｰへのﾃﾞﾊﾞｲｽｺﾝﾃｷｽﾄ
199:        MSG lpmsg;                              // Windows ﾒｯｾｰｼﾞ処理用
200:        int i, j, k;
201:        long t;
202://      CDC *phdc;
203:        RECT rect1;
204://      int      sts;
```

```
205:
206:        if (m_FFlag == FALSE) return;
207:        m_FFlag = FALSE;
208:
209:        RedrawWindow(NULL, NULL, RDW_ERASE | RDW_INVALIDATE);
210:
211:// 画面描画用
212://      if ( m_pDC == NULL )    m_pDC  = new CClientDC(this);
213:        m_pDC = new CClientDC(this);
214:
215:        if (m_pCapture->DctInit() == 255) {
216:                sprintf(str, "DCTｴﾘｱ定義ﾌｧｲﾙ '%s' がｵｰﾌﾟﾝできません", m_pCapture->m_AreaFile);
217:                m_pDC->TextOut(0, 0, str);
218:                m_FFlag = TRUE;
219:                return;
220:        }
221:
222:
223:// Winsock 関連
224:        m_pSocket = new CSocket();                                  // ｿｹｯﾄｵﾌﾞｼﾞｪｸﾄ
225:        for (i = 0; i< 256; i++) {
226:                m_SockBuf[i] = 0;
227:        }
228:
229:        phdc = m_pDlg->GetDC(); // ﾋﾞﾃﾞｵ入力表示用ﾀﾞｲｱﾛｸﾞのﾃﾞﾊﾞｲｽｺﾝﾃｷｽﾄを取得
230:        if (phdc == NULL) {
231:                m_pDC->TextOut(0, 0, "ﾃﾞﾊﾞｲｽｺﾝﾃｷｽﾄが取得できません");
232:                m_FFlag = TRUE;
233:                return;
234:        }
235:        phdc->SetTextColor(RGB(255, 0, 0));     // ﾃｷｽﾄの色を設定
236:        phdc->SetBkMode(TRANSPARENT);           // ﾃｷｽﾄのﾊﾞｯｸｸﾞﾗｳﾝﾄﾞを描画しない
237:
238:        ////////// ｿｹｯﾄ通信 //////////
239:        m_pSocket->SetSrvName(m_SrvName);
240:
241:        if ((m_Socket = m_pSocket->Connect()) != -1) { // ｻｰﾊﾞにｺﾈｸﾄする
242:                sprintf(str, "Connect to %s", m_SrvName);
243:                m_pDC->TextOut(0, 0, str);
244:        } else {
245://              PutErrorMsg(ERR_SOCKET, 0, m_SrvName);
246:                m_FFlag = TRUE;
247:                m_pDC->TextOut(0, 0, "Cannot connect.");
248:                goto exit;
249:        }
250:        GetInitValue();                                             // DCTの初期値を得る
251:
252:        m_pSocket->Send(m_SockBuf);             // ｻｰﾊﾞに初期位置を送信
253:        m_pSocket->StartInterrupt(m_hWnd);  // 送信要求割込み開始
254:
255:        m_tid = 1;
256:        while ( SetTimer(m_tid, 10000, NULL) == 0 ) m_tid++; // ｲﾝﾀｰﾊﾞﾙﾀｲﾏの設定
257:        m_Counter = 0;
258:
259:        // ﾒｲﾝﾙｰﾌﾟ (ｱﾌﾟﾘｹｰｼｮﾝの終了ﾒｯｾｰｼﾞが来るか「終了」が選択されるまでﾙｰﾌﾟ)
260:        while ((lpmsg.message != WM_DESTROY) && (m_FFlag == FALSE)) {
261:                if (PeekMessage(&lpmsg, NULL, 0, 0, PM_REMOVE) != 0) { // ﾒｯｾｰｼﾞｷｭｰをpeek
262:                        TranslateMessage(&lpmsg);
263:                        DispatchMessage(&lpmsg);
264:                } else {
265:                        if (GrabFlag == TRUE) { // grab割込みが発生したかどうか
266:                                GrabFlag = FALSE;
267:                                m_CFlag  = FALSE;
268:                                m_pCapture->Dct();
269:                                m_Counter++; // ﾊﾟﾌｫｰﾏﾝｽ測定用
270:
271:                                for (i = 0; i < m_pCapture->m_aNum; i++) {
272:                                        m_SockBuf[i*3+0] = (short)(m_pCapture->m_GH[i] - m_InitBuf[i*3+0]);
```

```
273:                                    m_SockBuf[i*3+1] = (short)(m_pCapture->m_GV[i] - m_InitBuf[i*3+1]);
274:                                    m_SockBuf[i*3+2] = (short)(m_pCapture->m_GHV[i]- m_InitBuf[i*3+2]);
275:
276:                                    rect1.left   = m_pCapture->m_lx[i]-1;
277:                                    rect1.top    = m_pCapture->m_ly[i]-1;
278:                                    rect1.right  = rect1.left + m_pCapture->m_wx[i] + 2;
279:                                    rect1.bottom = rect1.top  + m_pCapture->m_wy[i] + 2;
280:                                    phdc->FrameRect(&rect1, phBrush);
281:                                    if ( i < 2 ) {
282:                                            rect1.left   = m_pCapture->m_lx[i]-1;
283:                                            rect1.top    = m_pCapture->m_ly[i]-1;
284:                                            rect1.right  = rect1.left + m_pCapture->m_EyeMid[i]+3;
285:                                            rect1.bottom = rect1.top  + m_pCapture->m_wy[i] + 2;
286:                                            phdc->FrameRect(&rect1, phBrush);
287:                                    }
288:                            }
289:                            t = m_pCapture->m_aNum * 3;
290:                            for ( i = 0; i < 2; i++ ) {
291:                                    for ( j = 0; j < 3; j++ ) {
292:                                            for ( k = 0; k < 16; k++ ) {
293:                                                    m_SockBuf[t+(i*3+j)*16+k] = m_pCapture->m_EY[i][j][k];
294:                                            }
295:                                    }
296:                            }
297:/*
298:                            rect1.left   = m_pCapture->m_lx[0]+24;
299:                            rect1.top    = m_pCapture->m_ly[0]-1;
300:                            rect1.right  = rect1.left;
301:                            rect1.bottom = rect1.top  + m_pCapture->m_wy[0] + 2;
302:                            phdc->FrameRect(&rect1, phBrush);
303:                            t += m_pCapture->m_wy[0] / 2;
304:*/
305:/*
306:                            rect1.left   = m_pCapture->m_lx[1]+24;
307:                            rect1.top    = m_pCapture->m_ly[1]-1;
308:                            rect1.right  = rect1.left;
309:                            rect1.bottom = rect1.top  + m_pCapture->m_wy[1] + 2;
310:                            phdc->FrameRect(&rect1, phBrush);
311:*/
312:                    }
313:            }
314:    }
315:    if ( KillTimer(m_tid) == 0 ) {
316:            m_pDC->TextOut(0, 0, "KillTimer に失敗しました(;_;)");
317:    }
318:    m_pSocket->StopInterrupt(m_hWnd);
319:    m_tid = -1;
320:    m_Counter = 0;
321:    m_pSocket->SShutdown();
322:    sprintf(str,"Socket をshutdown しました");
323:    m_pDC->TextOut(0, 0, str);
324:exit:
325:    m_FFlag = TRUE;
326:    ReleaseDC(phdc);
327:    delete m_pSocket;       m_pSocket = NULL;
328:    delete m_pDC;   m_pDC = NULL;
329:    if ( m_QFlag ) {
330:            m_pCapture->CaptureQuit();
331:            ::PostQuitMessage(0);
332:    }
333:}
334:
335:// 「終了」ﾒﾆｭｰが選択されたとき
336:
337:void CTraceView::OnEnd()
338:{
339:    if (m_FFlag == FALSE) { // 「開始」ﾒﾆｭｰが選択されているか
340://            m_pDC->TextOut(700, 0, "          ");
```

```
341:                m_FFlag = TRUE;
342:        }
343:}
344:
345:void CTraceView::OnQuit()
346:{
347:        CClientDC *nDC = new CClientDC(this);
348:
349:        m_QFlag = TRUE;
350:        if ( m_FFlag == FALSE ) {
351:                m_FFlag = TRUE;
352:        } else {
353:                if ( m_pCapture != NULL ) {
354:                        m_pCapture->CaptureQuit();
355:                }
356:/*              if ( m_pSocket != NULL ) {
357:                        m_pSocket->SShutdown();
358:                }*/
359:                m_pDlg->DestroyWindow();
360:                delete m_pDlg;
361:                delete m_pCapture;
362://              delete m_pSocket;
363:                delete m_pDC;
364:                ::PostQuitMessage(0);
365:        }
366:}
367:
368:// ｲﾝﾀﾊﾞﾙﾀｲﾏ
369:void CTraceView::OnTimer(UINT nIDEvent)
370:{
371:        char buf[80];
372:        int a;
373:
374:        if ( m_pDC != NULL ) {
375:                m_pDC->TextOut(0, 20, "                                ");
376:                m_pDC->TextOut(0, 40, "                                ");
377:                a = (int)(m_Counter / 10); // ﾊﾟﾌｫｰﾏﾝｽの算出
378:                sprintf(buf, "1秒間に %d 回のﾏｰｶ検出", a);
379:                m_pDC->TextOut(0, 0, buf);
380:        }
381:        m_Counter = 0; // ｶｳﾝﾀｸﾘｱ
382:        CView::OnTimer(nIDEvent);
383:}
384:
385:void CTraceView::OnRetry()
386:{
387:        if ( m_FFlag == FALSE ) {
388:                GetInitValue();
389:        }
390:}
391:
392:// ｻｰﾊﾞからの送信要求ﾒｯｾｰｼﾞ処理
393:
394:LONG CTraceView::OnWinsock(UINT Param1, LONG Param2)
395:{
396:        int Socket;
397:        int len;
398:        short tmp;
399:        fd_set r_mask, e_mask;
400:
401:        if ( m_pSocket != NULL ) {
402:                Socket = m_pSocket->GetSocket();
403:                if ( Socket > 0 ) {
404:                        FD_ZERO(&r_mask);
405:                        FD_ZERO(&e_mask);
406:                        FD_SET(Socket, &r_mask);
407:                        FD_SET(Socket, &e_mask);
408:                        select( 0, &r_mask, NULL, &e_mask, NULL);
```

```
409:                        if (FD_ISSET(Socket, &r_mask)) {
410:                                tmp = m_pSocket->Receive(&len); // ｻｰﾊﾞからのﾒｯｾｰｼﾞを受信
411:                                if (tmp == 1) {
412:                                        m_pSocket->Send(m_SockBuf);
413:                                }
414:                                if (len <= 0) {
415:                                        OnEnd();
416:                                }
417:                        }
418:                        if (FD_ISSET(Socket, &e_mask)) {
419:                                OnEnd();
420:                        }
421:                }
422:        } else {
423:        }
424:        return 0;
425:}
426:
427://「通信に関する設定」ﾒﾆｭｰが選択されたとき
428:
429:void CTraceView::OnSetSock()
430:{
431:        if (m_FFlag == TRUE) {
432:                CSockDlg sDlg; // 通信関係設定用ﾀﾞｲｱﾛｸﾞｵﾌﾞｼﾞｪｸﾄ
433:
434:                sDlg.m_Name = m_SrvName;
435:                if (sDlg.DoModal() == IDOK) { // ﾓｰﾀﾞﾙﾀﾞｲｱﾛｸﾞとして表示
436:                        m_SrvName = sDlg.m_Name;
437:                }
438:        }
439:}
440:
441:void CTraceView::OnMenuArea()
442:{
443:        if (m_FFlag == TRUE) {
444:                CArea *nDlg = new CArea(this);
445:                if (nDlg->DoModal() == IDOK) {
446:                        m_pCapture->m_AreaFile = nDlg->m_AreaFile;
447:                }
448:        }
449:}
450:
451:////////////////////////////////////////////////////////////////////////////
452:// Winsock Implementation
453:
454:int PASCAL FAR __WSAFDIsSet(SOCKET fd, fd_set FAR *set)
455:{
456:        int i = set->fd_count;
457:
458:        while(i--)
459:                if (set->fd_array[i] == fd)
460:                        return 1;
461:
462:        return 0;
463:}
464:
```

```
// VINDLG.CPP : implementation file
//

#include "stdafx.h"
#include "trace.h"
#include "VINDLG.H"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CVinDlg dialog


CVinDlg::CVinDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CVinDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CVinDlg)
                // NOTE: the ClassWizard will add member initialization here
        //}}AFX_DATA_INIT
}


void CVinDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CVinDlg)
                // NOTE: the ClassWizard will add DDX and DDV calls here
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CVinDlg, CDialog)
        //{{AFX_MSG_MAP(CVinDlg)
                // NOTE: the ClassWizard will add message map macros here
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CVinDlg message handlers
```

```
1:; trace.def : Declares the module parameters for the application.
2:
3:NAME          TRACE
4:DESCRIPTION   'TRACE'
5:EXETYPE       WINDOWS
6:
7:CODE          PRELOAD MOVEABLE DISCARDABLE
8:DATA          PRELOAD MOVEABLE MULTIPLE
9:
10:HEAPSIZE     1024    ; initial heap size
11:; Stack size is passed as argument to linker's /STACK option
12:IMPORTS
13:;Error function
14:    CMTERR.CMTErrGetErrorString
15:
16:;Gen functions
17:    CMTGEN.CMTGenLibInit
18:    CMTGEN.CMTGenLibQuit
19:    CMTGEN.CMTGenBoardReset
20:    CMTGEN.CMTGenBoardSelect
21:    CMTGEN.CMTGenContextRestore
22:    CMTGEN.CMTGenContextSave
23:    CMTGEN.CMTGenLibGetVersion
24:    CMTGEN.CMTGenEbiBusEnable
25:    CMTGEN.CMTGenEbiSetBusMode
26:    CMTGEN.CMTGenAcsSetVGABuffer
27:    CMTGEN.CMTGenAcsGetVGABuffer
28:    CMTGEN.CMTGenXpressGetDelay
29:    CMTGEN.CMTGenEbiCtrlEnable
30:    CMTGEN.CMTGenParamGetDefault
31:
32:;Gen variables
33:    CMTGEN._gbyCMTGenNumberOfBoards
34:    CMTGEN._gbyCMTGenCurBoardNu
35:    CMTGEN.__gCMTGenBoardType
36:    CMTGEN.__gwCMTGenMajorBoardVersion
37:    CMTGEN.__gwCMTGenMinorBoardVersion
38:
39:;Rgb functions
40:    CMTRGB.CMTRgbInSetReference
41:    CMTRGB.CMTRgbInSetLevel
42:    CMTRGB.CMTRgbInSetSync
43:    CMTRGB.CMTRgbOutSetMode
44:    CMTRGB.CMTRgbOutSetLut
45:    CMTRGB.CMTRgbOutSelectLut
46:    CMTRGB.CMTRgbInSetStandard
47:    CMTRGB.CMTRgbParamGetDefault
48:
49:;Vin functions
50:    CMTVIN.CMTVinChromaSetHue
51:    CMTVIN.CMTVinChromaSetSat
52:    CMTVIN.CMTVinInSelect
53:    CMTVIN.CMTVinInSetStandard
54:    CMTVIN.CMTVinInSetType
55:    CMTVIN.CMTVinInSetGenLockSrc
56:    CMTVIN.CMTVinLumSetAGC
57:    CMTVIN.CMTVinLumSetBrightness
58:    CMTVIN.CMTVinLumSetClamp
59:    CMTVIN.CMTVinLumSetContrast
60:    CMTVIN.CMTVinLumSetCoring
61:    CMTVIN.CMTVinLumSetGain
62:    CMTVIN.CMTVinLumSetSharp
63:    CMTVIN.CMTVinParamGetDefault
64:
65:;Vip functions
66:    CMTVIP.CMTVipGrabSetDestPos
67:    CMTVIP.CMTVipGrabSetSourcePos
68:    CMTVIP.CMTVipGrabSetRate
```

```
 69:    CMTVIP.CMTVipGrabSetMode
 70:    CMTVIP.CMTVipGrabGetDestPos
 71:    CMTVIP.CMTVipGrabSingle
 72:    CMTVIP.CMTVipGrabGetOffScreenSize
 73:    CMTVIP.CMTVipGrabGetOffScreenAddress
 74:    CMTVIP.CMTVipGrabStartInterrupts
 75:    CMTVIP.CMTVipGrabStopInterrupts
 76:    CMTVIP.CMTVipGrabRegisterIntFunction
 77:    CMTVIP.CMTVipKeyFillRect
 78:    CMTVIP.CMTVipKeyFillRow
 79:    CMTVIP.CMTVipKeyFillColumn
 80:    CMTVIP.CMTVipKeySetMode
 81:    CMTVIP.CMTVipOverlayVideo
 82:    CMTVIP.CMTVipPictureLoad
 83:    CMTVIP.CMTVipInSelect
 84:    CMTVIP.CMTVipDisplaySetSt
 85:    CMTVIP.CMTVipGrabSetOffScreenPos
 86:    CMTVIP.CMTVipDisplayFromOffScreen
 87:    CMTVIP.CMTVipParamGetDefault
 88:    CMTVIP.CMTVipOverlayVideoWin
 89:
 90:;Vpe functions
 91:    CMTVPE.CMTVpeSetEnableSt
 92:    CMTVPE.CMTVpeSetAntiFlickerSt
 93:    CMTVPE.CMTVpeSetStandard
 94:    CMTVPE.CMTVpeSetUnderscan
 95:    CMTVPE.CMTVpeDencSetChromBand
 96:    CMTVPE.CMTVpeDencSetStd
 97:    CMTVPE.CMTVpeDencSetGenLockSrc
 98:    CMTVPE.CMTVpeDencLutLoad
 99:    CMTVPE.CMTVpeDencLutSetByPass
100:    CMTVPE.CMTVpeDencSetScPhase
101:    CMTVPE.CMTVpeDencSetHsPhase
102:    CMTVPE.CMTVpeParamGetDefault
103:
104:    CmtVip._Mrv2VipWaitForVBlank
105:    CmtGen.CmtGenBlockRead
106:    CmtGen.CmtGenBlockWrite
107:
108:
109:
```

```
1:# Microsoft Visual C++ generated build script - Do not modify
2:
3:PROJ = TRACE
4:DEBUG = 0
5:PROGTYPE = 0
6:CALLER =
7:ARGS =
8:DLLS =
9:D_RCDEFINES = /d_DEBUG
10:R_RCDEFINES = /dNDEBUG
11:ORIGIN = MSVC
12:ORIGIN_VER = 1.00
13:PROJPATH = C:¥SOURCE¥DCT004¥
14:USEMFC = 1
15:CC = cl
16:CPP = cl
17:CXX = cl
18:CCREATEPCHFLAG =
19:CPPCREATEPCHFLAG = /YcSTDAFX.H
20:CUSEPCHFLAG =
21:CPPUSEPCHFLAG = /YuSTDAFX.H
22:FIRSTC = GRABINT.C
23:FIRSTCPP = STDAFX.CPP
24:RC = rc
25:CFLAGS_D_WEXE = /nologo /G2 /W3 /Zi /AL /Od /D "_DEBUG" /D "WIN30" /D "CMT" /FR /GA /Fd"TRACE.PDB"
26:CFLAGS_R_WEXE = /nologo /Gs /G3 /FPc /W3 /ALw /Gx- /O2 /D "NDEBUG" /D "WIN30" /D "CMT" /FR /GA
27:LFLAGS_D_WEXE = /NOLOGO /NOD /PACKC:61440 /STACK:10240 /ALIGN:16 /ONERROR:NOEXE /CO
28:LFLAGS_R_WEXE = /NOLOGO /NOD /PACKC:61440 /STACK:10240 /NOPACKF /ALIGN:16 /ONERROR:NOEXE /MAP /LINE
29:LIBS_D_WEXE = lafxcwd oldnames libw llibcew commdlg.lib shell.lib
30:LIBS_R_WEXE = lafxcw oldnames libw llibcew commdlg.lib shell.lib
31:RCFLAGS = /nologo /z
32:RESFLAGS = /nologo /t
33:RUNFLAGS =
34:DEFFILE = TRACE.DEF
35:OBJS_EXT = MARKSORT.OBJ
36:LIBS_EXT = ..¥..¥WINSOCK¥WINSOCK.LIB
37:!if "$(DEBUG)" == "1"
38:CFLAGS = $(CFLAGS_D_WEXE)
39:LFLAGS = $(LFLAGS_D_WEXE)
40:LIBS = $(LIBS_D_WEXE)
41:MAPFILE = nul
42:RCDEFINES = $(D_RCDEFINES)
43:!else
44:CFLAGS = $(CFLAGS_R_WEXE)
45:LFLAGS = $(LFLAGS_R_WEXE)
46:LIBS = $(LIBS_R_WEXE)
47:MAPFILE = nul
48:RCDEFINES = $(R_RCDEFINES)
49:!endif
50:!if [if exist MSVC.BND del MSVC.BND]
51:!endif
52:SBRS = STDAFX.SBR ¥
53:              TRACE.SBR ¥
54:              MAINFRM.SBR ¥
55:              TRACEDOC.SBR ¥
56:              TRACEVW.SBR ¥
57:              SOCKET.SBR ¥
58:              GRABDLG.SBR ¥
59:              VINDLG.SBR ¥
60:              SOCKDLG.SBR ¥
61:              CAPTURE.SBR ¥
62:              GRABINT.SBR ¥
63:              JFWDDCT.SBR ¥
64:              AREA.SBR
65:
66:
67:MARKSORT_DEP =
68:
```

```
 69:WINSOCK_DEP =
 70:
 71:TRACE_RCDEP = c:¥source¥dct004¥res¥trace.ico ¥
 72:         c:¥source¥dct004¥res¥bitmap1.bmp ¥
 73:         c:¥source¥dct004¥res¥trace.rc2
 74:
 75:
 76:STDAFX_DEP = c:¥source¥dct004¥stdafx.h
 77:
 78:
 79:TRACE_DEP = c:¥source¥dct004¥stdafx.h ¥
 80:         c:¥source¥dct004¥mtrace.h ¥
 81:         c:¥source¥dct004¥trace.h ¥
 82:         c:¥source¥dct004¥mainfrm.h ¥
 83:         c:¥source¥dct004¥tracedoc.h ¥
 84:         c:¥source¥dct004¥tracevw.h
 85:
 86:
 87:MAINFRM_DEP = c:¥source¥dct004¥stdafx.h ¥
 88:         c:¥source¥dct004¥trace.h ¥
 89:         c:¥source¥dct004¥mainfrm.h
 90:
 91:
 92:TRACEDOC_DEP = c:¥source¥dct004¥stdafx.h ¥
 93:         c:¥source¥dct004¥trace.h ¥
 94:         c:¥source¥dct004¥tracedoc.h
 95:
 96:
 97:TRACEVW_DEP = c:¥source¥dct004¥stdafx.h ¥
 98:         c:¥source¥dct004¥mtrace.h ¥
 99:         c:¥source¥dct004¥trace.h ¥
100:         c:¥source¥dct004¥tracedoc.h ¥
101:         c:¥source¥dct004¥tracevw.h ¥
102:         c:¥source¥dct004¥socket.h ¥
103:         c:¥winsock¥winsock.h ¥
104:         c:¥source¥dct004¥capture.h ¥
105:         c:¥source¥dct004¥grabdlg.h ¥
106:         c:¥source¥dct004¥sockdlg.h ¥
107:         c:¥source¥dct004¥grabint.h ¥
108:         c:¥source¥dct004¥marksort.h
109:
110:
111:SOCKET_DEP = c:¥source¥dct004¥stdafx.h ¥
112:         c:¥source¥dct004¥socket.h ¥
113:         c:¥winsock¥winsock.h
114:
115:
116:GRABDLG_DEP = c:¥source¥dct004¥stdafx.h ¥
117:         c:¥source¥dct004¥mtrace.h ¥
118:         c:¥source¥dct004¥trace.h ¥
119:         c:¥source¥dct004¥grabdlg.h ¥
120:         c:¥source¥dct004¥capture.h
121:
122:
123:VINDLG_DEP = c:¥source¥dct004¥stdafx.h ¥
124:         c:¥source¥dct004¥trace.h ¥
125:         c:¥source¥dct004¥vindlg.h
126:
127:
128:SOCKDLG_DEP = c:¥source¥dct004¥stdafx.h ¥
129:         c:¥source¥dct004¥trace.h ¥
130:         c:¥source¥dct004¥sockdlg.h
131:
132:
133:CAPTURE_DEP = c:¥source¥dct004¥stdafx.h ¥
134:         c:¥source¥dct004¥mtrace.h ¥
135:         c:¥source¥dct004¥capture.h ¥
136:         c:¥cmtlib¥include¥cmtlib.h ¥
```

```
137:        c:¥cmtlib¥include¥vmtxdef.h ¥
138:        c:¥cmtlib¥include¥cmtdef.h ¥
139:        c:¥cmtlib¥include¥cmttype.h ¥
140:        c:¥cmtlib¥include¥cmtfnct.h ¥
141:        c:¥cmtlib¥include¥cmtvar.h ¥
142:        c:¥source¥dct004¥grabint.h
143:
144:
145:GRABINT_DEP = c:¥cmtlib¥include¥cmtlib.h ¥
146:        c:¥cmtlib¥include¥vmtxdef.h ¥
147:        c:¥cmtlib¥include¥cmtdef.h ¥
148:        c:¥cmtlib¥include¥cmttype.h ¥
149:        c:¥cmtlib¥include¥cmtfnct.h ¥
150:        c:¥cmtlib¥include¥cmtvar.h
151:
152:
153:JFWDDCT_DEP = c:¥source¥dct004¥jinclude.h ¥
154:        c:¥source¥dct004¥jconfig.h ¥
155:        c:¥source¥dct004¥jpegdata.h
156:
157:
158:all:    $(PROJ).EXE $(PROJ).BSC
159:
160:TRACE.RES:      TRACE.RC $(TRACE_RCDEP)
161:        $(RC) $(RCFLAGS) $(RCDEFINES) -r TRACE.RC
162:
163:STDAFX.OBJ:     STDAFX.CPP $(STDAFX_DEP)
164:        $(CPP) $(CFLAGS) $(CPPCREATEPCHFLAG) /c STDAFX.CPP
165:
166:TRACE.OBJ:      TRACE.CPP $(TRACE_DEP)
167:        $(CPP) $(CFLAGS) $(CPPUSEPCHFLAG) /c TRACE.CPP
168:
169:MAINFRM.OBJ:    MAINFRM.CPP $(MAINFRM_DEP)
170:        $(CPP) $(CFLAGS) $(CPPUSEPCHFLAG) /c MAINFRM.CPP
171:
172:TRACEDOC.OBJ:   TRACEDOC.CPP $(TRACEDOC_DEP)
173:        $(CPP) $(CFLAGS) $(CPPUSEPCHFLAG) /c TRACEDOC.CPP
174:
175:TRACEVW.OBJ:    TRACEVW.CPP $(TRACEVW_DEP)
176:        $(CPP) $(CFLAGS) $(CPPUSEPCHFLAG) /c TRACEVW.CPP
177:
178:SOCKET.OBJ:     SOCKET.CPP $(SOCKET_DEP)
179:        $(CPP) $(CFLAGS) $(CPPUSEPCHFLAG) /c SOCKET.CPP
180:
181:GRABDLG.OBJ:    GRABDLG.CPP $(GRABDLG_DEP)
182:        $(CPP) $(CFLAGS) $(CPPUSEPCHFLAG) /c GRABDLG.CPP
183:
184:VINDLG.OBJ:     VINDLG.CPP $(VINDLG_DEP)
185:        $(CPP) $(CFLAGS) $(CPPUSEPCHFLAG) /c VINDLG.CPP
186:
187:SOCKDLG.OBJ:    SOCKDLG.CPP $(SOCKDLG_DEP)
188:        $(CPP) $(CFLAGS) $(CPPUSEPCHFLAG) /c SOCKDLG.CPP
189:
190:CAPTURE.OBJ:    CAPTURE.CPP $(CAPTURE_DEP)
191:        $(CPP) $(CFLAGS) $(CPPUSEPCHFLAG) /c CAPTURE.CPP
192:
193:GRABINT.OBJ:    GRABINT.C $(GRABINT_DEP)
194:        $(CC) $(CFLAGS) $(CCREATEPCHFLAG) /c GRABINT.C
195:
196:JFWDDCT.OBJ:    JFWDDCT.C $(JFWDDCT_DEP)
197:        $(CC) $(CFLAGS) $(CUSEPCHFLAG) /c JFWDDCT.C
198:
199:AREA.OBJ:       AREA.CPP $(AREA_DEP)
200:        $(CPP) $(CFLAGS) $(CPPUSEPCHFLAG) /c AREA.CPP
201:
202:
203:$(PROJ).EXE::   TRACE.RES
204:
```

```
205:$(PROJ).EXE::   STDAFX.OBJ TRACE.OBJ MAINFRM.OBJ TRACEDOC.OBJ TRACEVW.OBJ SOCKET.OBJ ¥
206:        GRABDLG.OBJ VINDLG.OBJ SOCKDLG.OBJ CAPTURE.OBJ GRABINT.OBJ JFWDDCT.OBJ AREA.OBJ $(OBJS_EXT) $(DEFFILE)
207:        echo >NUL @<<$(PROJ).CRF
208:STDAFX.OBJ +
209:TRACE.OBJ +
210:MAINFRM.OBJ +
211:TRACEDOC.OBJ +
212:TRACEVW.OBJ +
213:SOCKET.OBJ +
214:GRABDLG.OBJ +
215:VINDLG.OBJ +
216:SOCKDLG.OBJ +
217:CAPTURE.OBJ +
218:GRABINT.OBJ +
219:JFWDDCT.OBJ +
220:AREA.OBJ +
221:$(OBJS_EXT)
222:$(PROJ).EXE
223:$(MAPFILE)
224:C:¥MSVC¥CDK16¥LIB¥+
225:c:¥msvc¥lib¥+
226:c:¥msvc¥mfc¥lib¥+
227:c:¥windows¥+
228:..¥..¥WINSOCK¥WINSOCK.LIB+
229:$(LIBS)
230:$(DEFFILE);
231:<<
232:        link $(LFLAGS) @$(PROJ).CRF
233:        $(RC) $(RESFLAGS) TRACE.RES $@
234:        @copy $(PROJ).CRF MSVC.BND
235:
236:$(PROJ).EXE::   TRACE.RES
237:        if not exist MSVC.BND   $(RC) $(RESFLAGS) TRACE.RES $@
238:
239:run: $(PROJ).EXE
240:        $(PROJ) $(RUNFLAGS)
241:
242:
243:$(PROJ).BSC: $(SBRS)
244:        bscmake @<<
245:/o$@ $(SBRS)
246:<<
247:
```