(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3993896号
(P3993896)
(45)発行日　平成19年10月17日(2007.10.17)　(24)登録日　平成19年8月3日(2007.8.3)

(51)Int.Cl.　F I
G06T 1/00 (2006.01)　G06T 1/00 400G
A61B 5/117 (2006.01)　A61B 5/10 322

請求項の数 22　(全 70 頁)

(21)出願番号　特願平9-539259
(86)(22)出願日　平成9年4月30日(1997.4.30)
(65)公表番号　特表2002-515145(P2002-515145A)
(43)公表日　平成14年5月21日(2002.5.21)
(86)国際出願番号　PCT/US1997/007427
(87)国際公開番号　WO1997/041528
(87)国際公開日　平成9年11月6日(1997.11.6)
審査請求日　平成16年3月9日(2004.3.9)
(31)優先権主張番号　08/640,006
(32)優先日　平成8年4月30日(1996.4.30)
(33)優先権主張国　米国(US)

(73)特許権者
アイデンティクス・インコーポレーテッド
アメリカ合衆国、カリフォルニア州　９４０８６、サニーベイル、ノース・パストリア・アベニュー　５１０
(74)代理人
弁理士　山崎　行造
(74)代理人
弁理士　岡田　希子
(72)発明者　マース、ダニエル・フレドリク
アメリカ合衆国、カリフォルニア州　９５００８、キャンプベル、スウィートブライアー・ドライブ　８５４

最終頁に続く

(54)【発明の名称】ころがされた指紋イメージのよごれを減少させる方法及び装置

(57)【特許請求の範囲】
【請求項１】
ころがされたイメージアレーによって表されたころがされた指紋イメージ内のよごれを減少させる方法であって、
光イメージの対応する位置の光強度を表すデータ値を含む光イメージ信号のフレームを連続的に発生する工程であって、前記光イメージが表面上を転がる指の指紋イメージを含む工程と、
前記光イメージ信号の各フレームごとに、指紋イメージの先頭端部と後端部との間に配置され、かつ転がる指の転がり方向を横切るように指向されたラインを表す固定コラムを決定する工程と、
光イメージ信号のフレームの蓄積で、ころがされた指紋の一時的なイメージを表す一時的なアレーを連続的に更新する工程であって、前記光イメージ信号の最新のフレームの対応するデータ値が、前記一時的なアレーのピクセル値より小さい場合に、前記光イメージ信号の前記最新のフレームからの対応するデータ値と前記一時的なアレーのピクセル値との間の差の一部によって、前記一時的なアレーのピクセル値を減少することによって、前記一時的なアレーの最新の更新が形成される工程と、
一時的なアレーの一部をころがされたイメージアレーに転送することによってそのころがされたイメージアレーを発生する工程であって、前記一時的なアレーの最新の更新の転送された部分は、前記光イメージ信号の前記最新のフレームより先行した光イメージ信号の前のフレームから決定された固定コラムから、指の転がりの方向に向かって前方に延在す

る工程とを備える方法。
【請求項２】
請求項１の方法において、前記一時的な最新の更新の転送された部分は、前記一時的なイメージのころがされた指紋の先頭端部の近くまでの一時的イメージの部分を表すデータを含む方法。
【請求項３】
請求項１の方法において、前記一時的な最新の更新の転送された部分は、前記光イメージ信号の最新のフレームから決定された固定コラムの近くまで延在する方法。
【請求項４】
請求項１の方法において、前記光イメージ信号の各フレームから決定された固定コラムによって表された固定ラインは、指紋イメージの先頭端部と後端部との間の転がり方向への距離の少なくともほぼ半分に配置されている方法。
【請求項５】
請求項１の方法において、前記光イメージ信号の各フレームから決定された固定コラムによって表された固定ラインは、指紋イメージの先頭端部と後端部との間の転がり方向への距離の半分より多くに配置されている方法。
【請求項６】
請求項１の方法において、前記一時的なアレーの第１の転送された部分は、前記光イメージ信号の第１のフレームから決定された固定コラムの近くから指の転がる方向の後方に延在する方法。
【請求項７】
請求項６の方法において、前記一時的なアレーの前記第１の転送された部分は、前記第１の一時的なイメージの後端部のあたりから転がる方向の前方の一時的なイメージの一部を表す方法。
【請求項８】
請求項１の方法において、さらに、最大値によって前記一時的なアレーのピクセルを初期化する工程を含む方法。
【請求項９】
請求項１の方法において、さらに、前記光イメージ信号のフレームのデータ値を用いて前記一時的なアレーを初期化する工程を含む方法。
【請求項１０】
請求項１の方法において、さらに、最後の一時的なアレーの一部をころがされたイメージのアレーに転送した後に、そのころがされたイメージのアレーを保存する工程を含む方法。
【請求項１１】
請求項１の方法において、さらに、前記ころがされたイメージアレーによって表されたころがされた指紋のイメージを、それが発生されたときに、表示装置に表示させる工程を含む方法。
【請求項１２】
請求項１１の方法において、さらに、前記ころがされたイメージアレーが前記一時的なアレーより少ないピクセルを持つように、前記転送された部分の１０分の１を除く工程を含む方法。
【請求項１３】
請求項１の方法において、前記一時的なアレーを連続的に更新する工程は、光イメージ信号のフレームが生成されたときに、前記一時的なアレーを実時間で更新する工程を含み、前記ころがされたイメージアレーを発生する工程が、一時的なアレーが更新されたときに、該一時的なアレーの一部をころがされたイメージアレーに転送する工程を含む方法。
【請求項１４】
請求項１の方法において、前記一時的なアレーの搬送された部分は隣接しかつ重複していない方法。

【請求項１５】
ころがされた指紋イメージを表すころがされた指紋イメージアレーを生成する方法であって、
表面上を連続する時間で転がる指の光イメージを表す光イメージ信号の連続するフレームを発生する手段であって、前記フレームがデータを含み、各データの値が転がる指の光イメージの対応する位置の光強度を表す手段と、
各フレームから固定コラムを決定する手段であって、各固定コラムが、前記転がる指の対応する光イメージの先頭端部と後端部との間の位置を表す手段と、
フレームが発生されたときに、該フレームを用いてイメージメモリの一時的なアレーを連続的に更新し、前記一時的なアレーが先頭端部と後端部とを持つころがされた指紋の一時的なイメージを表す手段であって、光イメージ信号の第１の一つをイメージメモリに転送することによって前記一時的なアレーを最初に更新し、次に、最新のフレームの対応するデータ値が、前記一時的なアレーのピクセル値より暗いイメージを示す場合に、前記最新のフレームの対応するデータ値と前記一時的なアレーのピクセル値との間の差の一部を用いて、前記一時的なアレーのピクセル値を減少することによって、前記一時的なアレーを更に更新する手段と、
前記フレームから決定される固定コラムを前記一時的なアレーの対応する更新に関連させる手段と、
一時的なアレーの各更新の一部を出力メモリに連続的に転送することによって、一時的なアレーの更新を有する出力メモリ内のころがされた指紋イメージアレーを連続的に更新する手段であって、前記第１の更新された一時的なアレーに関連する固定コラムの近くから後方へ指の転がりの方向に延在する一時的なアレーの第１の更新の一部を転送し、次に、一時的なアレーの前の更新と関連する固定コラムの近くから前方へ指の転がりの方向に延在する一時的なアレーの連続的な更新の一部を転送する手段とを備える方法。
【請求項１６】
請求項１５の装置において、一時的なアレーの連続的な更新のために転送された部分は、連続的に更新された一時的なアレーの固定コラムのみに向かって前方へ延在する方法。
【請求項１７】
請求項１５の装置において、前記一時的なアレーの連続する更新の転送された部分は隣接していて重複していない方法。
【請求項１８】
請求項１５の装置において、前記一時的なアレーの連続的な更新の転送された部分は、隣接するとともに、前の光イメージ信号から決定される固定コラムから指の転がりの方向の後方へ重複していない方法。
【請求項１９】
ころがされたイメージアレーによって表されたころがされた指紋イメージ内のよごれを減少する装置であって、
表面上を転がる指の指紋のイメージを含む光イメージを表す連続する電子信号のフレームを連続的に発生するイメージ化装置と、
前記電子信号に応答して光イメージ信号のフレームを連続的に発生するする手段であって、各光イメージ信号がデータを含み、各データの値が、光イメージの対応する位置の光強度を表す手段と、
前記光イメージ信号に応答して先頭端部と後端部とを持つころがされた指紋の一時的なイメージを表す一時的なアレーを連続的に更新するイメージ捕捉装置であって、一時的なアレーの最新の更新は、最新のフレームの対応するデータ値が、前記一時的なアレーの前の更新のピクセル値より暗いイメージを示す場合に、前記最新のフレームの対応するデータ値と前記一時的なアレーの前の更新のピクセル値との間の差の一部を用いて、前記一時的なアレーの前の更新のピクセル値を減少することによって、前記一時的なアレーの前の更新と光りイメージ信号の最新のフレームとから形成されるイメージ捕捉装置と、
光イメージ信号の各フレームごとに、指紋イメージの先頭端部及び後端部の間に配置され

てころがされた指の転がり方向を横切るように指向されたラインを表す固定コラムを決定する手段と、
一時的なアレーの一部をころがされたイメージアレーに転送することによってそのころがされたイメージアレーを発生する手段であって、前記最新の一時的なアレーの転送された部分は、前記光イメージ信号の前記最新のフレームより先行した光イメージ信号の前のフレームから決定された固定コラムから、指の転がりの方向に向かって前方に延在する手段とを備える装置。
【請求項２０】
請求項１の方法において、前記一時的なアレーF^nの最新の更新が以下の関係から形成される方法：
$F^n = F^{n-1} - K * (F^{n-1} - \underline{I^n})$
ここで、F^nは、一時的なアレーの最新の更新のピクセル値であり、F^{n-1}は、一時的なアレーのピクセル値であり、I^nは、光イメージ信号の対応するデータ値であり、Ｋは１に等しいかまたはそれより小さい係数である。
【請求項２１】
請求項２０の方法において、Ｋは０．２５乃至０．５の範囲にある方法。
【請求項２２】
請求項２０の方法において、Ｋは約０．３３である方法。
【発明の詳細な説明】
発明の背景
本願発明は、電子指紋イメージ捕捉装置に関し、特に、捕捉されたころがされた指紋イメージのよごれを減少させる方法に関する。
指紋のイメージを得る従来の方法は、最初にインクを対象の指に塗布し、次に、指を紙に押し付けることによって、隆起線及び谷の指紋パターンを紙に転写している。隆起線の指紋のパターンは紙に転写されるが、谷は写らない。転がる指紋イメージを得るために、インクが付けられた指の側面を紙の所望の領域に置き、次に、その指をその指の他の面に向けて紙上を転がす。
光電子装置はインクを使うことなくころがされた指紋のイメージを捕捉することができる。一般には、イメージ化の表面上を転がる指の連続する光イメージは、そのイメージ化装置から伝達されてデジタルデータに変換される。さまざまな方法を用いて、連続するイメージを表すデジタルデータからころがされた指紋のイメージを発生することができる。１つの方法は、米国特許第4,933,976号に開示されている。その方法によると、伝達されたイメージはイメージデータのデジタルアレーの形式で連続的に記憶される。指紋の特徴を表すアレーの能動領域は、記憶されたイメージデータの数値的関数として特定されている。隣接する二次元の能動領域が十分な重複部分を持つ場合には、それらはデータの数値的関数に従って重複領域に合併されて、ころがされた指紋イメージを表す複合アレーを形成する。ステップを発生するその複合アレーの数値関数は、隣接する能動領域内の重複するデータの平均、比較又は平均及び比較である。
他の方法が、カリフォルニア州サニーバレーのIdentix Inc.によって製造されたモデルTP-600装置で用いられている。そのTP-600は、光プラテンの全体のイメージ化表面を収容する大きな電荷結合装置（ＣＣＤ）のイメージ化装置を持つ光光学系を備える。そのＣＣＤ出力は、イメージ化表面上の明暗のパターンを示すアナログ信号である。指がプラテン上に置かれると、そのアナログ信号は、指紋の隆起線情報のために低い値（暗い）を持ち、指紋の谷の情報のために高い値（明るい）を持り、それは、インクが指紋の転写のために用いられたときに生じるのと同様である。そのアナログ信号は、アナログ・デジタル（Ａ／Ｄ）変換器に供給され、その出力は、最小関数によってイメージメモリ内のアレーの内容を更新するために用いられるデジタルイメージデータである。アレー内の各要素は、最初に、プラテン上の対応する位置でイメージ化された光強度を表す値を持つ。指がプラテンのイメージ化表面を横切って転がされるにつれて、イメージメモリ内にデータが作られて更新される。

最小関数は、イメージメモリ内に、入力イメージデータの対応する値より低いピクセル値を保存することによって作動する。最新のイメージデータの値が、イメージメモリ内の対応するピクセル値より低い場合には、その低いイメージデータは、アレー内で大きな値と置き換わる。従って、指の隆起線がイメージ化表面に接触した位置ごとに、低いピクセル値（暗い）がイメージメモリに保存される。イメージメモリの内容は、周辺装置に出力されて、補足されたころがされた指紋イメージを記憶し、増加するころがされた指紋イメージの実時間表示を行う。

最も暗い強度値を保存することによって、ころがされたプリントを捕捉する最小関数方法を用いることは、高品質のプリントで認識できるアーティファクトが不要であり、さらに、指がころがされる速度に対する感度が鈍いが、プリントのいくつかの領域は汚された特徴を持ち、それにより、隆起線と谷との間の差が減少する。この影響は、指が接触しているがイメージ化表面上を滑る場合に生じる。そのよごれは、指の先端及び接触領域の端部で発生することが多い。接触面に粘性の被覆を用いることは滑りを全体的に減少させるが、指の周囲の形状による先端のよごれによって依然として問題が残る。よごれは、インクが付けられたプリントと光電子装置によって得られたものとに見られるが、光電子装置が滑りが生じた領域内のイメージをきれいにするので利点がある。

発明の概要

本願発明は、ころがされたイメージアレーによって表されたころがされた指紋イメージ内のよごれを減少させる方法を提供する。その方法は、光イメージ信号のフレームを連続的に発生する工程であって、光イメージ信号が光イメージの対応する位置の光強度を表すデータを含み、さらに、光イメージが表面上を転がる指の指紋イメージを含む工程を含む。その方法は、光イメージ信号の各フレームごとに、指紋イメージの先頭端部と後端部との間に配置され、かつ転がる指の転がり方向を横切るように指向されたラインを表す固定コラムを決定する工程を含む。その方法は、さらに、光イメージ信号のフレームの蓄積で、ころがされた指紋の一時的なイメージを表す一時的なアレーを連続的に更新する工程を含む。光イメージ信号の最新のフレームの対応するデータ値が、一時的なアレーのピクセル値より小さい場合（暗い特徴を表す）に、光イメージ信号の最新のフレームからの対応するデータ値と一時的なアレーのピクセル値との間の差の一部によって、一時的なアレーのピクセル値を減少することによって、一時的なアレーの最新の更新が形成される。そのころがされたイメージアレーは、一時的なアレーの一部をころがされたイメージアレーに転送することによって生成される。

各更新サイクルの間に、新たな固定コラムが、指の接触領域の中央部の近くの位置に決定され、それは次に、光イメージ信号の最新のフレームと関連する指紋イメージの先頭端部及び後端部から決定される。そのころがされたイメージアレーは、最新の一時的なアレーの後端部によって初期化され、その後端部は指の転がり方向における最新の固定コラムの後の一時的なアレーである。各時間で、新たな固定コラムは、装置内のプロセッサによって決定され、最新の固定コラムと前の固定コラムとの間の最新の一時的なアレーはころがされた指紋イメージアレーに転送される。別の例では、前の固定コラムと一時的なイメージ内のころがされた指紋の先頭端部を表すデータとの間の最新の一時的なアレーは、新たな固定コラムが決定される毎に、ころがされた指紋イメージアレーに転送される。両方の場合ともに、前の固定コラムの後ろの一時的なアレーの後端部は、さらにころがされたイメージアレーを更新するためには用いられない。従って、そのころがされたイメージアレー内のデータは、指の転がり方向に移動する固定コラムの後には固定され、そのコラムの後の指の移動のためにそのころがされた指紋イメージ内の生じたよごれは取り除かれる。固定コラムの後のころがされた指紋イメージ内のよごれを取り除くことに加えて、本願発明は、イメージデータ信号の連続するフレームを結合する際に最小関数によって提供される利点は保持する。

【図面の簡単な説明】

明細書に組み込まれてその一部を構成する添付図面は、概略、本願発明の一実施例を示しており、さらに、上記の一般的な説明及び以下に説明する実施例の詳細な説明とともに、

本願発明の主要な点を説明するのに役に立つ。
図１は、イメージ捕捉装置の斜視図である。
図２は、図１に示されたイメージ捕捉装置の従来のころがされた指紋の光イメージ装置の概略図である。
図３は図１のイメージ捕捉装置の機能ブロック図である。
図４は図１のイメージ捕捉装置の一部の機能ブロヅク図である。
図５Ａ－図５Ｅは、パターン上を回転するの指の連続する光イメージを示す。
図６Ａ－図６Ｅは、一時的なデータメモリアレーによって図３に示されたイメージメモリに提供された連続するイメージを示す。それらのイメージは一時的にそれぞれ図５Ａ－図５Ｅに示されたイメージに対応する。
図７Ａ－図７Ｅは、ころがされた指紋のイメージアレーによって出力ＤＲＡＭに提供された連続するイメージを示す。それらのイメージは一時的にそれぞれ図６Ａ－図６Ｅに示されたイメージに対応する。
図８Ａ－図８Ｅは、ころがされた指紋のイメージアレーによって出力ＤＲＡＭに提供された連続するイメージを示す。それらのイメージは一時的にそれぞれ図６Ａ－図６Ｅに示されたイメージに対応する。

本願発明の詳細な説明

ころがされた指紋イメージのよごれの影響を少なくするよごれ減少方法が提供されている。図１を参照すると、そのよごれ減少方法は、本願発明の譲受人であるIdentix Inc.によって製造されたモデルTP-600指紋捕捉装置１０の動作に組み込むことができる。
そのTP-600は、ころがされた指紋イメージを得、さらに、平坦なイメージまたはスラップイメージを得るために、別のイメージ装置を備える。その平坦な指紋イメージ装置１２は、平坦なプリントプラテン１４に押し付けられた１又は２以上の指のイメージを表すアナログ信号を生成する。ころがされた指紋イメージ装置１６は、ロールプリントプラテン２０を横切ってころがされた指１８のイメージを表すアナログ信号を生成する。図２も参照すると、各イメージ装置は照射源２２と、光学機器２４と、プラテン表面からの全体のイメージを収容する大きなＣＣＤイメージ装置２６とを備える。この説明した例においては、ころがされた指紋イメージ用のＣＣＤイメージ装置２６は、テキサス州ダラスのTexas Instruments Inc.から入手できるモデルTC217のＣＣＤイメージ化アレーである。図２には１つのみのミラーを示すが、光学機器２４は実際にはプリズム、ミラー及びレンズを備えており、それらはプラテン表面からＣＣＤイメージ装置２６までイメージを導くように選択されかつ配置されている。平坦なプリントプラテン１４は、ロールプリントプラテン２０よりも幅広く、それにより、その表面上に１つのみならず４つの指を収容するもので、その光学機器２４はその大きなイメージ化表面を収容するように配置されている。各装置の目的は、指がプラテンのイメージ化表面に接触する時に、ＣＣＤイメージ装置の表面に指紋イメージを与える点にある。
図３を参照すると、ＣＣＤイメージ装置２６の出力はアナログ信号３２であり、それはイメージ捕捉装置２８に供給される。照明及びイメージ化、使用されたＣＣＤ出力により、隆起線情報に関する低い値（暗い）と谷情報に関する高い値（明るい）とを持つイメージ信号が提供される。
プロセッサ３０は、その装置の各機能的要素の間での及びそれを介してのデータの移動を管理し、さらに、イメージ表示及び出力装置３４のイメージ表示モニタ６０にテキストを書いて送り、スイッチの閉鎖を遮り、さらに、システムの起動及び停止操作を実行するような他の「ハウスキーピング」機能を実行するために用いられる。以下に詳細に説明されているように、そのプロセッサ３０は、また、ころがされた指紋イメージを形成する際に、指がプラテン表面上を転がされたときのイメージデータの処理も能動的に管理する。望ましい実施例のために、Texas Instruments, Inc. によって製造されたグラフィックプロセッサ、部品番号TMS34020、が用いられる。この特別なプロセッサは、メモリの二次元アレーを処理するための特別の機能を有する。TP-600を作動するためのＣ言語のソースコードの写しがマイクロフィッシュの付録に含まれている。

ここで図４を参照すると、イメージ装置１６のＣＣＤイメージ装置２６からのアナログ信号３２が、イメージ捕捉装置２８の一部であるアナログ・デジタル（Ａ／Ｄ）変換器３６に供給される。スキャナの指紋の照射は均一ではないので、Ａ／Ｄ出力デジタルデータ３５のデータ値は、等化メモリ（EQU DRAM)４０に記憶されている等化ルックアップテーブル（EQU LUT）３８によって個別に定められる。記憶されている基準値は、ブランクのプラテン２０の表面のイメージに対応し、それはノイズ及び表面のよごれを取り除くようにスムース化されている。

Equ LUT３８からの出力は、複数のフレームにまとめることができるデジタルデータの流れの形式の光イメージ信号４２である。そのデータはプラテン２０のイメージ化表面の対応する位置の光強度を表す値を持つ。各フレームは異なる時間でのプラテンのイメージに対応する。そのデータ値は１秒ごとに約１５回更新される。従って、指１８がプラテン２０の表面を横切って転がる間の時間に、光イメージ信号４２の約２５－３５フレームが生成される。

光イメージ信号４２は、「最少機能」ルックアップテーブル（Min LUT)４８というように特定される機能的要素によって、968×968ピクセルの仮データアレーを保持するイメージメモリ４４の内容を更新するように用いられる。そのサイズのアレーは、１インチ（2.54cm）あたり６００ドットの解像度のイメージを生成するのに十分である。Min LUT４８への入力は、Equ LUT３８によって変えられるＡ／Ｄ変換器の出力３５である最新の光イメージ信号４２と、更新が予定されている、対応する古い一時的なデータアレーピクセルの値４６とである。「最新の値」は、光イメージ信号４２の最新のフレームからの入力であり、「古い値」は、光イメージ信号４２の前のフレームによって最新に更新されたときの、最新の一時的なデータアレーからの入力である。

最も簡単な実行の際には、Min LUT４８は、行ｉ及び列ｊの一時的なデータアレーの新たなピクセル値４９である$F^{n}_{i,j}$を、最小値、$F^{n}_{i,j}=\min(I^{n}_{i,j}, F^{n-1}_{i,j})$として計算処理を行う。ここで、$I^{n}$はEqu LUT３８からのｎ番目のフレームの入力データ値であり、$F^{n-1}_{i,j}$は前のフレームからのイメージメモリ４４からのフィードバック４６である。一時的なデータアレーの各ピクセルごとのMin LUT４８の出力信号４９は、その名前で示されているように、その２つの入力の低いほうの値である。Ａ／Ｄ変換器３６によって出力された各データ出力ごとに（Equ LUT３８によって変更されて光イメージ信号４２を形成するときに）、イメージメモリ４４の一時的なデータアレーの対応するピクセルが更新される。

Min LUT４８によって処理される前に、アレーとしてのＡ／Ｄ変換器３６から出力されたデジタルデータ３５とEqu LUT３８からの光イメージ信号４２とを保存する必要はない。一時的なデータアレーを更新するために用いられるMin LUT４８からの出力データ４９の値は、光イメージ信号４２の対応するデータ値と一時的なデータアレーの古い対応ピクセル値４６とにのみ依存する。指の隆起線がプラテン２０のイメージ化表面に接触する各位置ごとに、低いピクセル値（暗い）が保存される。この方法の結果、指１８がプラテン２０のイメージ化表面を横切って転がされるにつれて、一時的なころがされた指紋のイメージがイメージメモリ４４に構成される。このプロセスは、真の指紋の一部ではない指紋の特徴または不連続部のようなアーティファクトを排除することがわかっている。

ころがされた指紋イメージの捕捉を実行するときには、フィードバックを必要とするので、イメージメモリ４４内の一時的なデータアレーは初期化しなければならない。イメージメモリ４４を初期化する一つの方法としては、すべてのピクセル値を最大値に設定する方法がある。その後、イメージメモリ４４の一時的なデータアレーは、直ちに、次のフレームに入力されるいずれのデータをも反映する。他の例では、一時的なデータアレーは、ストレートスルー機能のようにMin LUT４８をセットアップすることによって初期化でき、それにより、その出力はEqu LUT３８から入力された光イメージ信号４２と同じになる。光イメージ信号４２の第１のフレームは、既に記憶されたものとは無関係にそのイメージメモリ４４を更新することができる。

指１８が転がされるにつれて、プラテン上にあるその指の接触領域の端が、フレーム更新

速度に対しかなり瞬間的に動くことがある。これにより、いくつかの不連続部分がビデオの飛越しされたフィールドの間に発生することがある。同様に、指の先端がプラテンに接触するときに滑り、それにより不連続部が生じることがある。この問題を取り除くために、Min LUT４８にロードされる機能は、厳密な最小値から変更することができ、それにより、入力データ値I^nが先の一時的なアレー値F^{n-1}より小さい場合には、その一時的なアレー値はその差の一部によって減少される。つまり、$F^n = F^{n-1} - K * (F^{n-1} - I^n)$である。但し、Ｋはピクセルの値がどのように速く変わるのかを設定する係数と等しいかそれより小さい係数である。０．２５乃至０．５の範囲内のＫによって画質を顕著に改善することができる。上記の実施例の場合には、Ｋは約０．３３に設定されている。この機能により、対象の状態がぎざぎざの不連続部ではなくてグレーのよごれのように現れるが、それは、その状態は多くの場合わずかな数のフレームに対してのみ存在するからである。Min LUT４８は６４Kb×８のSRAM及びレジスタを持ち、それにより、入力及び出力のパイプライン制御を行う。６４Kbのアドレス空間は１６のアドレスラインを必要とする。Min LUT４８への２つの８ビット入力は、それぞれ８アドレスラインに接続される。従って、２つの入力値の各組に対し、出力予定の所望の値を含むそのSRAMに、１つの対応する位置が存在する。どのような機能も表形式で実行できるので、この実行は非常に非制限的である。Min LUT４８で用いられる別の関数は、一般に、予め演算されてMain DRAM５４に記憶され、そして、必要なときに、SRAMにロードされる。

イメージメモリ４４は、２つの重複するメモリ、Min DRAM５０及びCatch VRAM５２を備える。それらは独立してかつ同時に同一の一時的なデータアレーを保持してイメージ表示及び出力装置３４に転送を行う。そのイメージ表示及び出力装置３４は、メインの出力メモリ５４（Main DRAMに配置されているもの）及び表示メモリ５６（ディスプレイVRAMに配置されているもの）を備えており、それらは、それぞれ、プロセッサ３０によってMin DRAM５０及びCatch VRAM５２から転送されたデータを受取る。表示メモリ５６は、情報を実時間でオペレータに提供するために用いられる。表示メモリ５６は、指紋配置カーソルを伴うイメージ情報と指示情報をオペレータに提供するためのテキスト情報とを受取る。表示メモリ５６は、データ効率、表示ラスター寸法及び他の表示制限事項のために、一般に、イメージメモリ４４または出力メモリ５４に含まれたものより少ない情報を含む。出力メモリ５４はテキスト情報及び指配置カーソル情報を含まない。そのメモリは高品質のイメージデータのすべてを含む。

イメージメモリ４４内の一時的なデータアレーは、暫定的なころがされた指紋イメージを表わしており、また、その一時的なデータアレーは、各フレーとともに全体として出力メモリ５４または表示メモリ５６に転送されて、ころがされた指紋イメージアレーを形成する。それは従来のTP-600のあの方法である。しかし、指１８が、一時的なデータアレーが形成されているときに、プラテンのイメージ化表面の上を滑ると、そのころがされた指紋のイメージは汚れたようになり、それは、ころがされた指紋のイメージを得るインク及び紙を用いたときに生じるのと同じようなものである。本願発明のそのよごれを減少させる方法は、出力メモリ５４及び表示メモリ５６に、各ビデオフレームの一時的なデータアレー４６の選択した部分のみを転送することによって、ころがされた指紋のイメージによごれが生じることを減少させる。

一般には、オペレータは、ころがされた指紋イメージを得るための捕捉モードに入る前にその指のイメージを観察する。指２０をプラテン１８の上に適切に配置するために、表示モニタ６０で指のイメージを観察しながらその指を中央に配置できることは有用である。オペレータはMin LUT４８をストレートスルー機能に設定し、そして、その指を一方の側に転がしてころがされたイメージの捕捉の準備を行う。表示されたイメージはプラテン２０上の指の転がされたイメージではなく、その指の直接のイメージである。走査ボタンが押されると捕捉モードに入るので、プレビューモードの終わりのイメージメモリ４４内のデータは、捕捉のためにそのメモリの初期化を行うように働く。

ここで、図５Ａを参照すると、捕捉モードに入った後の光イメージ信号４２の第１のフレームは、プラテン２０の表面の光イメージ６２ａを表しており、それは、プラテン２０上

の指１８の接触領域６４ａのイメージを含む。（その図面内のハッチングは、指紋の特徴を表している。）図５Ｂにおいては、接触領域６４ｂが接触領域６４ａの位置の右の方にあり、それは、指１８が右に向かってころがされたことを示す。その接触領域６４は、図５Ｃ及び５Ｄにおいて、右側に向かって徐々に移動しつづけている。図５Ｅにおいては、指１８はプラテン２０から持ち上げられているので、接触領域６４ｅが、前の接触領域６４ｄよりサイズが小さくなっている。

捕捉モードに入るときに、Min LUT４８は、上記のように、変更された最少関数に再登録される。ここで、また図６Ａも参照すると、Min LUT４８を通過した光イメージ信号４２の最新のフレームが、一時的なデータアレーの最初のフレームとなり、それは、一時的なころがされた指紋のイメージ６８ａを含む一時的なイメージ６６ａの特性を示す。一時的なころがされた指紋のイメージ６８ａは、この実施例では、図５Ａに示された対応する接触領域６４ａと同一である。別の例では、一時的なデータアレーは、ブランクの照射されたプラテン（図示せず）を示すようにすべてを高ピクセル値にすることによって初期化することができる。Min LUT４８は、次に、変更された最少関数を用いて一時的なデータアレーを更新することができ、その際には、１つの入力として光イメージ６２ａによって表された光イメージ信号の第１のフレームと、他の入力としての「ブランク」の一時的なデータアレーの対応するピクセル値も用いる。その結果としての一時的なデータアレーはいずれの場合も本質的に同一である。

その一時的なデータアレーは、次に、図５Ｂの光イメージ６２ｂによって表された光イメージ信号４２の第２のフレームが、前の一時的なイメージ６６ａによって表された一時的なデータアレーの対応するピクセル値４９によって、Min LUT４８を通過する際に処理されるときに更新される。その更新された一時的なデータアレーは、ここでは、図６Ｂに示すように、一時的なころがされた指紋イメージを含む一時的なイメージ６６ｂを表す。同様に、図６Ｃは、一時的なイメージ６６Ｃと一時的なころがされた指紋イメージ６８Ｃとを示しており、それらは、図５Ｃに示す光イメージ６２Ｃによって表されたイメージ信号４２の次のフレームを用いて更新された後に、イメージメモリ４４の一時的なデータアレーによって表される。図６Ｄ及び図６Ｅは、個々の一時的なイメージ６６ｄ、６６ｅと一時的なころがされた指紋イメージ６８ｄ、６８ｅとを示しており、それらは、一時的なデータアレーへの次の更新によって表されている。

接触領域の検出はさまざまな方法で行うことができる。一つの方法は、光イメージ信号を細かく区分けし、次に、その各区分のデータ値の変化を演算する方法である。小さな変化の区分は接触がないと判断する。他の方法は、各データ値をしきい値と比べ、値が一定レベルより下に落ちたときには、接触があったと判断する。これは、イメージの背景がEqu LUT３８によって等しくされたときには許容できるが、それは、一定レベルが、完全なイメージを横切る接触の一貫した程度に相当するからである。

イメージメモリ４４内の一時的なデータアレーの最下ビットまたはタグビット（ビット０）は、接触を表す関数に割り当てられる。Min LUT４８は、それに何がプログラムされるかということに関しては完全に弾力的なので、タグビット０に関する関数はビット１－７とは別に処理することができる。そのタグビットは、光イメージ信号４２からMin LUT４８への入力値が、しきい値Ｔよりも小さいときには、１に設定される。従って、接触領域６４を反映する情報が、二値イメージ７０ａ－７０ｅとして、Min DRAM５０及びCatch VRAM５２のタグビット０において得られ、それらのイメージはそれぞれ破線によって表された図６Ａ－６Ｅに示す輪郭線を有する。蓄積されたグレースケールの一時的なイメージ６６ａ－６６ｅは、イメージメモリ４４のCatch VRAM５２及びMin DRAM５０の高ビット１－７において得られる。たとえ、Min LUT４８がプレビューモードにあり、そこで光イメージ信号がイメージメモリ４４の送られたときでも、プロセッサ３０は、その二値イメージ７０ａ－７０ｅを決定することができることは理解すべきである。

光イメージ信号４２の各フレームによって表された指紋の接触領域６４は、左側のエッジを持つ接触ストリップ７２によって非常に簡単にモデル化することができる。その接触領域６４は通常凸面の周辺部を有するが、接触ストリップ７２を矩形状にし、それが、接触

領域６４の最も左端にあるコラムのような左端部７４と、その接触領域６４の最も右端にあるコラムのような右端部７６とを持つように考えられることは受け入れられることであることはわかっている。プロセッサ３０は、Catch VRAM５２にある、参照番号７０で概略が示されている二値の接触イメージから接触ストリップ７２の右端部７６及び左端部７４を決定する。それは、転がる指から遅れないために、フレームの更新速度と同等な時間のフレームで行われる。

接触ストリップ７２を決定する１つの方法は、二値の接触イメージ７０の中央を横切る１つのタグビットの行を調べる方法である。最も左側のタグ付けされたピクセルは、一時的なデータアレーの左端からその行内の第１のタグ付けされたビットを調査することによって発見され、最も右側のタグ付けされたピクセルは、その一時的なデータアレーの右端からその行内の第１のタグ付けされたビットを調査することによって発見される。接触ストリップ７２の左端部７４及び右端部７６は、次に、そのもっとも左のタグ付けされたピクセルともっとも右のタグ付けされたピクセルとによって識別される。

プラテン２０上に存在するほこりやよごれにより、接触領域６４の外で孤立したピクセルが、二値の接触イメージ７０を形成する際にタグ付けされることがある。指紋は隆起線からなり、それは、チェックされる線と整列し、それにより、谷が端部の位置を不明瞭にすることがある。それらの問題は、二値接触イメージ７０の中央の近くに多数の水平ラインを含む垂直バンド７８を調査することによって減少させることができる。例えば、二値の接触イメージ７０の中央部を横切り互いに４ライン離れた１０ラインの垂直バンド７８を用いることができる。

接触ストリップ７２の左端部７４としてもっとも左側のタグ付けされたピクセルのみを用いると、二値の接触イメージの中央部近くに２以上のラインを用いたとしても、よごれの存在に非常に敏感となり、接触領域６４の端部の指のイメージの接触を誤って示したり、そのイメージを歪ませたりすることがある。この問題を緩和するために、一つの実施例では、接触ストリップ７２の左端部７４のようなコラムを確立する前に、多数のタグ付けされたピクセルが、二値の接触イメージ７０の左側から数えられる。プロセッサ３０は、同様な方法で、接触ストリップ７２の右端部７６を決定する。接触ストリップ７２の左及び右端部７４、７６は、外側の端部から１０番目のタグ付けピクセルとして確立される。

プロセッサ３０は、処理操作が二次元ブロックの移動の間に実行できるようにする特別のモデルを有する。操作の１つは、論理ＯＲである。したがって、論理ＯＲを実行する間に、多数の行が１つの最後の行まで移動できる。その到着行は、単一のラインの代わりのバンド７８を越える接触ストリップの指示を提供する。

接触ストリップ７２を確認することによって、プロセッサ３０は、他のいくつもの新たな機能を実行することもできる。それは、接触ストリップ７２の左端部７４及び右端部７６の跡を追跡し、いつ指１８がブランクプラテンの上に配置され、いつ指１８が転がされ、さらに、いつ指１８がプラテン２０から持ち上げられるかを決定する。

指１８がブランクプラテン２０に配置されると、プラテンに接触する指の左端部が、右から左の方向に右端部を通り過ぎるであろう。指１８が置かれると、接触ストリップ７２は、右端部７６と左端部７４との間の正の幅を持つ。接触ストリップ７２の左端部７４が左に動きつづけ、接触ストリップ７２の右端部７６が右に移動しつづける限り、指１８はまだプラテン２０上に配置されている段階にあり、接触ストリップ７２は増大していると考えることができる。指１８が既にプラテン２０上に配置されていると、接触ストリップ７２は正の値からスタートする。プレビューモードが、指１８を配置し、さらにそれをスタート位置に戻すように用いられるときは、それは最も一般的な状況である。指１８は通常は捕捉モードが開始される前は再び上げられることはない。

プロセッサ３０は、接触ストリップ７２の一端が外側ではなく内側に動くようにスタートする時には、指１８の転がりは始まっていると決定する。例えば、図６Ａ－図６Ｄに示すように、左端部７４が右に移動するようにスタートするときは、プロセッサ３０は指１８が右に転がっていると決定する。この場合には、転がっている指の右端部７６が先頭端部で、左端部７４が後端部である。それより、右端部７６が左に移動している場合には、プ

ロセッサ３０は、指１８は左に転がっていると決定する。この場合には、転がっている指の左端部７４が先頭端部で、右端部７６が後端部である。接触ストリップ７２の左端部７４（又は右端部７６）が極限の位置から少しの数のピクセル、約５まで戻るように動くと、転がりが始まると決定することによって、ジターに対し小さな許容差を見込むことができる。所定数のフレームごとに、左端部７４が右に移動し、右端部７６が左に移動すると、プロセッサ３０は指１８がプラテン２０から持ち上げられていると決定する。

また、プロセッサは、イメージ信号４２の各フレームから固定コラム８０を決定する。それは、接触領域６４又は二値イメージ７０の位置にあって、各フレームごとに接触ストリップ７２の左端部７４と右端部７６との間にある位置に対応する。一つの実施例では、固定コラム８０は、後端部から先頭端部までの距離の約半分にある位置に対応する。他の実施例では、その固定コラムは、後端部から先頭端部までの距離の半分より長いところにある位置に対応する。

TP-600の従来の装置で行われていたように、指１８がイメージプラテン２０を横切って転がるのを終わらせた時のみに、一時的なデータアレーの全体を出力メモリ５４に転送する代わりに、プロセッサ３０は、指が転がるにつれて各新たなフレームを用いて一時的なデータアレーの一部を出力メモリに転送する。プロセッサ３０は、光イメージ信号４２の前のフレームから決定された固定コラムの後は、出力メモリ５４内のころがされた指紋のイメージアレーの一部の更新を止める。固定コラム８０は、接触ストリップ７２の右及び左端部７６、７４とともに、フレームごとに、増加するように移動する。固定コラム８０の後の発達するころがされた指紋のイメージ内のデータは更新されないので、一時的なデータアレーによって表された一時的なイメージ６２の後端部内で発達するイメージのどのようなよごれも、ころがされた指紋イメージアレー内には現れない。

図７Ａを参照すると、出力メモリ５２に出力された、ころがされた指紋のイメージが、ブランクの背景を示す高ピクセル値によって初期化される。図７Ｂを参照すると、指１８が転がり始めると、プロセッサ３０が、イメージメモリ４４から一時的なデータアレーの後端部を転送することによって、出力メモリ５４に出力されたころがされた指紋のイメージアレーを更新する。その後端部は、例えば、一時的なイメージ６６ｂを表す一時的なデータアレーの後端部となることがあり、それは、接触ストリップ７２の後端部７４に相当するコラムからスタートし、最新の光イメージ信号から決定される固定コラム８０を含む位置まで達する。その点における出力された、ころがされた指紋は、ころがされた指紋イメージ８２ｂを表しており、それは、転送された部分のイメージ８４ｂを含む。指１８が転がるにつれて、プロセッサ３０は、移動する接触ストリップ７２の中央近くの位置に送れないように出力メモリ５４の更新を行う。

出力メモリ内のころがされた指紋イメージアレーに対する後続の更新の各々は、イメージメモリ４４からブロック転送された一時的なデータアレーの一部である。一つの実施例では（図８Ａ－図８Ｅ及び下記の関連説明を参照）、最新の一時的なデータアレーの転送された部分は、先頭端部近くまで、つまり、最新の接触ストリップ７２の先頭端部に相当する一時的なデータアレーのコラムに達するまでの一時的なころがされた指紋イメージを表す。図７Ｃ－図７Ｄに示すような他の実施例では、イメージメモリ４４からの最新の一時的なデータアレーの転送された部分は、狭く、また、最新の光イメージ信号の固定コラム８０の近くまで延在する。図７Ｃは出力されたころがされた指紋イメージ８２Ｃを示しており、そこでは、一時的なデータアレーの最新の更新（図６Ｃ参照）の対応する部分が、出力されたころがされた指紋イメージアレーに転送される。その例での一時的なデータアレーの転送された部分は、固定コラム８０ｂの右側に、つまり、指が転がる方向に、コラム８０Ｃを含むところまで至るすべてのデータを含む。同様に、図７Ｄに示す出力されたころがされた指紋のイメージ８２ｄは、後続の更新及び一時的なデータアレー（図６Ｄ参照）の一部に続く出力されたころがされた指紋のイメージによって表される。その転送された部分は、固定コラム８０Ｃの右に向かい固定コラム８０ｄを含むまで達するすべてのデータを含む。

出力メモリ５４内のころがされた指紋イメージアレーに対する最後の更新のために、その

転送された部分は、先の光イメージ信号４２から決定された固定コラムに対応する一時的なデータアレーのコラムから接触ストリップ７２の先頭端部の最も極限の位置に対応する一時的なデータアレーの少なくともコラムまで延在する。すべての場合、最新の一時的なデータアレーの転送された部分は、前の光イメージ信号４２から抽出された固定コラム８０によって表されたラインから前方に向かう一時的なころがされた指紋イメージ６８を表す。例えば、図７Ｅに示すように、転送された部分のイメージ８４ｅを表す最後の転送された部分は、データの一時的なデータアレーであり、それは、前の固定コラム８４ｄから指の転がる方向に前方に向かう一時的なイメージ６６ｅの一部を表す。

したがって、プロセッサ３０は、移動する接触ストリップ７２のほぼ中央に対応する垂直ラインを表す、移動する固定コラム８０の後ろの出力メモリ５４を停止する。出力メモリ５４の更新は、その後ろでは起こらない。それは先端のよごれは除かないが、それはそれを約５０－６０％まで減少する。指の後端部の移動による指紋の主な部分のよごれは取り除かれる。よごれは、先端部と固定コラム８０との間でも発生することがあるので、後端部と先頭端部との間の距離の８分の５（５／８）、先頭端部に近い位置に固定コラム８０をセッティングすることによって、方法を改善することができる。

プロセス３０は、図５－７に示す実施例において右端部７６となる接触ストリップ７２の先頭端部の後発達を追跡する。指１８がプラテン２０から持ち上げられるにつれて、所定数のコラムによって先頭端部の最大限の位置から後退すると、プロセッサ３０は、捕捉は完了したと決定し、出力メモリ５４に最後の更新を実行し、そして、出力メモリ５４内のころがされた指紋イメージアレーのそれ以上の更新を停止する。これは、指１８がプラテン２０から持ち上げられる時のよごれをを防ぐ。したがって、指１８がプラテン２０から持ち上がり、または、元の方向に転がると、プロセッサは、一時的なデータアレーの前方部分いよって出力メモリ５４を更新し、それ以降の更新を停止する。

プロセッサ３０は、イメージメモリ４４の内容の一部を通過するだけでいつでも出力メモリ５４まで到達するということに気がつくのは重要である。その部分は、一時的なデータアレーの狭いストリップに対応しており、それは、前に通過したデータによって規定される固定コラムの近くに、しかし、それとは重複することなく配置されている。出力メモリ５４を更新するために用いられたデータは生の指紋イメージを表していないという点に気づくことも価値のあることである。それより、転送されたデータは、その狭いストリップ８４内にMin LUT４８によって生成された一時的なころがされた指紋イメージ６８を表すが、それは、指１８は転がり始めたからである。

ころがされた指紋のイメージは、それが捕捉されるにつれて表示される。プロセッサは、データをイメージメモリ４４内のCatch VRAM５２から表示メモリ（VRAM）５６まで移動する。そのデータは、次に、デジタル・アナログ変換器（D/A）５８を通じてビデオフォーマットに変換され、そして表示モニタ６０に出力する。図４に示す実施例においては、表示モニタは７２０×７２０のピクセルアレーによって形成された表示領域を持つ。プロセッサ３０は、イメージが表示フォーマットに適合するようにCatch VRAM５２から表示VRAM５６までの転送の間に、４つの中からの１つのピクセル行及び４つの中からの１つの行によってイメージの１０分の１を除く。

ここで、図８Ａ－図８Ｅを参照すると、プロセッサ３０は、イメージメモリ４４から表示メモリ５６（上記のように減少したフォーマットで）を更新するが、その間、指１８はプラテン２０上を転がって、表示のころがされた指紋イメージ８６を表す表示のころがされた指紋イメージアレーを生成する。指１８が下に置かれるにつれて、つまり、プレビューモードにある間、一時的なイメージ６６ａ（光イメージ６２ａと同じである）を表す一時的なデータアレーの全体が、表示VRAM５６に転送されて、「ライブ」と表示される。図８Ａにおいて、ライブイメージはイメージ８６ａである。上記のとおり、捕捉モータに入ると、二値の接触イメージ７０の左及び右端部７４、７６の間の接触ストリップ７２を表す一時的なデータアレーのデータ更新される。指が転がり始めると、指の転がりの方向に関して最新の固定コラム８０ｂの後ろのストリップを表す、Catch VRAM５２からの一時的なデータアレー内のデータの一部が転送されて、ころがされた指紋イメージアレーを表示す

る。図８Ｂにストリップイメージ８８ｂとして示したそのストリップは、一時的なイメージ６６ｂの遠くの左端部まで延在し、その結果、指２０が接触しないプラテンの表面の領域を表している。したがって、前の固定コラム８０ｂ、８０ｃのそれぞれと重ならずにかなり接近している一時的なイメージ６６のストリップを表す一時的なデータアレーから接触ストリップ７２の先頭端部７４までのデータの部分は、Catch VRAM５２から表示VRAM５６まで転送される。それらのストリップは図７Ｃ及び図７Ｄにそれぞれストリップイメージ８８ｃ及び８８ｄとして示されている。それは、表示VRAM５６内の表示のころがされた指紋イメージアレー内の転がる指紋イメージの重なりの完全なイメージ８６ｃ、８６ｄのそれぞれを保持する。図８Ｅを参照すると、捕捉が完了したとみなされたときに、前の固定コラム８０ｄから一時的なデータアレーの前方端部まで一時的なイメージ６６ｅの前方へ進んだストリップ８８ｅを表す一時的なデータアレーの一部が、表示VRAM５６に転送される。この最後の更新は、指１８が接触しないプラテン２０の遠方の右端部までの領域を表すデータも含むことがある。各更新は最少関数４８によって処理された一時的なデータアレーの一部である。

表示の品質の違いに対応して、多くの様々な表示方法を実行することができる。例えば、接触領域の右及び左までブランク領域を更新するためか否か、又は、出力イメージメモリ又はキャッチメモリから更新するためか否かは、処理効率に対するトレードオフとすることができるオプションである。他の例では、例えば、処理時間が必要な場合には、最新の固定コラム８０のどのような前進方向への進みも表示することはない。

捕捉が完了した後は、オペレータは、プリントを拒否するか、又はそのプリントを保存するためにボタンを押す。そのプリントが保存されたときには、背景を白くしてクリアな画像を提供することができる。それは、出力イメージとプラテンに残っているイメージとを比較することによって達成される。オペレータは、保存ボタンを押す前に指を持ち上げたと仮定する。出力イメージのピクセル値が、残留イメージ（つまり、指がない状態でのプラテンのイメージ）の対応す値より選択された割合（例えば、約５％）まで下方にある場合には、そのピクセルは接触されたものだと考えることができ、したがってそれはタグ付けされている。タグ付けされていないすべてのピクセルは、一致する背景レベルまで白色化されていない。それは、その背景又はプリント内の隙間に存在することがあるどのような潜在的なイメージも取り除く。

イメージメモリ４４、出力メモリ５４及び表示メモリ５６の各々が、最初のイメージを表すデータによって初期化された後には、光イメージ信号４２又はその光イメージ信号が実際に記憶されている場所は装置内には存在しない点に注意すべきである。イメージ装置１６からの光指紋イメージを表す光イメージ信号４２の後続のフレームのすべては、最少的機能４８を通じて処理される。それは、出力及び表示メモリ５４、５６を更新するために用いられているイメージメモリ４４内の各更新済みの一時的なデータアレー内のデータの単なる一部である。

上記の実施例では、Min LUT４８は、一時的なデータアレーの現存のピクセル値からのイメージメモリ４４の一時的なデータと、Ａ／Ｄ変換器１８及びEqu LUT３８を経由して入力された光イメージ信号４２の新たなフレームからの対応するデータとを更新する。指２０が回転するにつれて、プロセッサは、イメージメモリ４４内の接触ストリップ７２の前方の部分を表す一時的なデータアレーの一部を用いて、出力及び表示メモリ５４、５６を更新する。したがって、出力及び表示のころがされた指紋イメージアレーは、それぞれ、出力及び表示メモリ５４、５６内で形成される。そのころがされた指紋のイメージアレーは、ころがされた指紋のイメージを表す。

他の実施例においては、その一時的なデータアレーは、光イメージ信号４２及び従ってその一時的なデータアレーも固定コラム８０の後ろで止められるように、イメージメモリ４４内に作られる。それは、固定コラム８０を特定する上方を記憶するハードウエアレジスタ（図示せず）と、指１８の転がる方法に応じて、ラインの右または左のいずれかの記憶を禁止する制御とのようなハードウエア（図示せず）によって実行される。プロセッサ３０は、その固定コラム８０の更新のみが必要であり、捕捉が終了したときに一時的なデー

タアレーを出力メモリ５４に転送する。
イメージメモリ４４内の一時的なイメージアレーの更新を固定する機能は、Min LUT４８の動作の一部として実行することもできる。例えば、そのMin LUT４８へのアドレス入力の一つは、一時的なデータアレーに固定されるようにまたはMin LUT４８によって更新されるようにデータを選択するように割り当てることができる。そのビットは、そのイメージコラムと、プロセッサ３０によって更新されたレジスタに記憶された固定コラム８０と比較することによって制御することができる。
よごれの減少は、接触ストリップを特定するようにする多くの一般的な方法、例えば、行ベースで行上の接触ストリップを決定する方法を用いることによって改善することができるが、それは非常に多くの処理を必要とする。Min LUT４８は、イメージメモリ４４内の一時的なデータアレーを更新するためにも用いることができる。固定位置は、ライン数によってアドレス付けられたメモリ（図示せず）に各ライン毎に固定コラム８０を記憶することによって、ラインベースでライン上で制御することができる。そのメモリは、ビデオのブランキング期間内に、又は二重ポート技術を用いることによって、各フィールドごとに、プロセッサ３０によって更新することができる。各ラインごとの固定位置は、イメージを通るいくつかのラインから能動的レンジを検出し、次に、スムース化されまたは補間された位置を、中間に入るラインの固定位置に提供することから作られる。
最近の機器は、飛越しビデオ入力を用いているが、本願発明は、順次走査、つまり、ラインの飛越しを行うことなく、１つのフレームのみを出力する走査を提供するカメラを用いても実施することができる。これは、最少関数の多くの変形の必要性をなくすことになる。
転がりの進行はプロセッサ３０によって追跡されるので、オペレータがいくつかのボタンを押すことを排除することができる。これは、イメージメモリ４４を消去し、さらに、システムの操作の望ましいモードに従って所定の状態下で自動的に捕捉を再開することによって理論的に達成することができる。
その方法の例としては、イメージ捕捉が上記の方法の場合のように完了したと考えられた後に、オペレータは、保存ボタン（図示せず）又は保存フットスイッチ（図示せず）を押すことによってイメージを保存したいということを知らせる。プリントを拒否するために、指は下に戻されて再度転がされる。プロセッサが、指が再度接触状態にあることを検出すると、イメージメモリは消去されて、捕捉が再開される。それによって、プレビューモードを捕捉モードと一体化することができる点に注目すべきである。ただし、その際には、オペレータが捕捉の前にその指を持ち上げて再配置したとする。
プロセッサ３０は、指１８が最初のときに方向を変えたと決定したときには、捕捉を再開するように構成することができる。これは、指を中央に置くように配置し、指を一方の側に転がして戻し、次に、指紋ををを転がすという通常の操作のよく対応する。
オペレータは、すべての指を順番に転がすと仮定する。そのオペレータが指紋を拒否するときには、そのオペレータはボタンを押す。オペレータは、指が中央に配置されるまで、転がすことなく、それを配置しつづけ、次に、捕捉のためにその指を転がす。この操作は、イメージを固定するために確認された状態、つまり、指を下に配置し、左又は右に転がし、その指を持ち上げることから決定することができる。指を配置することと完全な捕捉を実行することとを識別するために、追加の基準を転がり量に設けてもよい。
イメージ化装置１６は、高い値によって指紋の隆起線を示し、低い値によって指紋の谷の特徴を示すような信号をＡ／Ｄ変換器３６に提供するように等価な実施例に設計することができる。上述の方法及び装置は、その変化に対応するためにはわずかな変更のみを必要とする。
光イメージ信号を上ではデータ流れとして説明したが、その光イメージ信号はピクセルのアレーとしてフォーマットすることも可能である。
次の添付書面はIdentix TP-600の作動のためのソフトウエアに関するＣ言語ソースを含める。本願明細書の開示の一部は著作権の保護を受けた資料を含む。その著作権の所有者は、特許書面または特許開示のいずれのものによるファクシミリ再現に対しては異議はない

が、それ以外の場合には、ともかくもすべての著作権を保有する。

C-DOC

```
Defined Functions, SUMMARY Graphic TREEs (of CALLER/CALLED flow Structure)
*************************************************************************

 812 FNGRCAPT. C        1  CaptSaveAct
 181 GSP_INIT.C         2  |-- load_LUTTBL_to_MinSram
  79 GRABBER.C          3  |-- turn_grab-on
  46 GRABBER.C          4  |-- turn_grab_off
 624 FNGRCAPT.C         5  |-- display_fingerprint
1014 FNGRCAPT.C         6  |   |-- display_4_3v
  61 MIMIO.C            7  |   |   --mim_size
  53 MIMIO.C            8  |   |   --mim_adr
  70 MIMIO.C            9  |   |   --mim_inc
  15 MIMIO.C           10  |   |   --mim_new
  38 MIMIO.C           11  |   |   --mim_move
                       12  |   |     |--..bytebit
                       13  |   |..poke_breg asm
1110 FNGRCAPT.C        14  |-- fill_even_pixels
  61 MIMIO.C           15  |   |-- mim_size ( 7 )
  53 MIMIO.C           16  |   |-- mim_adr ( 8 )
 282 GSP_INIT.C        17  |-- copy_image_to_Dram
 111 GRABBER.C         18  |-- turn_grab_on_normal
  38 MIMIO.C           19  |-- mim_move ( 11 )
  61 MIMIO.C           20  |-- mim_size ( 7 )
                       21  |..display_columns_4_3
 798 FNGRCAPT.C        22  CaptScanAct
  46 GRABBER.C         23  |-- turn_grab_off ( 4 )
 758 FNGRCAPT.C        24  PrevSaveAct
  46 GRABBER.C         25  |-- turn_grab_off ( 4 )
  79 GRABBER.C         26  |-- turn_grab_on ( 3 )
                       27  |-- OSQPend_key
 732 FNGRCAPT.C        28  PrevScanAct
  46 GRABBER.C         29  |-- turn_grab_off ( 4 )
 181 GSP_INIT.C        30  |-- load_LUTTBL_to_MinSram ( 2 )
 111 USERINTF.C        31  PrintTitle
                       32  |..set_colors select_font text_width text_out
  88 FNGRCAPT.C        33  VOID
                       34  |..prevscr_key_scr
 128 FNGRCAPT.C        35  captscr_key_act
 109 FNGRCAPT.C        36  captscr_key_scr
 118 FNGRCAPT.C        37  capture_t
                       38  |..prvscr_key_act
```

C-DOC

```
 659 FNGRCAPT.C        39  display_camera
 514 GSP_INIT.C        40  |-- load_straight_thruLUT_to_EquSram
 181 GSP_INIT.C        41  |-- load_LUTTBL_to_MinSram ( 2 )
  79 GRABBER.C         42  |-- turn_grab_on ( 3 )
 624 FNGRCAPT.C        43  |-- display_fingerprint ( 5 )
  46 GRABBER.C         44  |-- turn_grab_off ( 4 )
                       45  |..ramdac_display_cross_symbol OSQPend_key
                               ramdac_clear_cross_symbol
  74 USERINTF.C        46  display_text
                       47  |..text_out
1036 FNGRCAPT.C        48  fill_col_4_3
  61 MIMIO.C           49  |-- mim_size ( 7 )
  53 MIMIO.C           50  |-- mim_adr ( 8 )
  70 MIMIO.C           51  |-- mim_inc ( 9 )
 192 FNGRCAPT.C        52  image_capture
 931 FNGRCAPT.C        53  |-- finger_type_check
 159 FNGRCAPT.C        54  |-- my_mim_init
  15 MIMIO.C           55  |   |-- mim_new ( 10 )
  27 MIMIO.C           56  |   |-- mim_subset
  70 MIMIO.C           57  |   |-- mim_inc ( 9 )
  53 MIMIO.C           58  |   |-- mim_adr ( 8 )
  61 MIMIO.C           59  |-- mim_size ( 7 )
 323 GSP_INIT.C        60  |-- load_equref_to_EquDram
 478 GSP_INIT.C        61  |   |-- load_equLUT_to_EQUSram
 551 GSP_INIT.C        62  |   |       --floor0
 514 GSP_INIT.C        63  |   |-- load_straight_thruLUT_to_EquSram ( 40 )
 705 FNGRCAPT.C        64  |-- PrevInitAct
 181 GSP_INIT.C        65  |   |-- load_LUTTBL_to_MinSram ( 2 )
  79 GRABBER.C         66  |   |-- turn_grab_on ( 3 )
 624 FNGRCAPT.C        67  |-- display_fingerprint ( 5
 111 GRABBER.C         68  |-- turn_grab_on_normal ( 18
 857 FNGRCAPT.C        69  |-- fz_detect_edges
  77 MIMIO.C           70  |   |-- mim_fill
                       71  |   |       ..bytefil
  45 MIMIO.C           72  |   |-- mim_move_c
                       73  |   |       ..byteblt
  61 MIMIO.C           74  |   |--mim_size ( 7 )
  53 MIMIO.C           75  |   |--mim_adr ( 8 )
1062 FNGRCAPT.C        76  |-- smooth_scan_line
 913 FNGRCAPT.C        77  |-- save_columns
  61 MIMIO.C           78  |   |-- mim_size ( 7 )
  27 MIMIO.C           79  |   |-- mim_subset ( 56 )
  38 MIMIO.C           80  |   |-- mim_move ( 11 )
  38 MIMIO.C           81  |-- mim_move ( 11 )
  46 GRABBER.C         82  |-- turn_grab_off ( 4 )
  79 GRABBER.C         83  |-- turn_grab_on ( 3 )
                       84  |--CaptureScreen OSQPend_key display_columns_4_3
 113 GSP_INIT.C        85  initialize_34020
 563 GSP_INIT.C        86  |-- setup_graphic
                       87  |   |--set_config clear_whole_screen intall_font
                               select_font get_fontinfo
  23 GRABBER.C         88  |--init_gsp_grabber
  46 GRABBER.C         89  |   |--turn_grab_off ( 4 )
 201 GSP_INIT.C        90  |--calculate_minfuncLUT_to_Dram
 551 GSP_INIT.C        91  |   |--floor0 ( 62)
 232 GSP_INIT.C        92  |--calculate_straight_thruLUT_to_Dr
 252 GSP_INIT.C        93  |--calculate taggingLUT_to_Dram
                       94  |..asm init_ramdac turn_illuminator_on
  85 MIMIO.C           95  mim_fill_c
                       96  |..bytefil
  98 FNGRCAPT.C        97  misfngscr_key_scr
  87 USERINTF.C        98  text_length
                       99  |..text_width
 143 GRABBER.C         100 wait_for_grab_VBLANK_on_off
```

```
/*
** Filename: fngrcapt.c
** Purpose:  Routines to handle the finger image capture and user interface.
** Date:     08/03/93
** Author:   Joyce Young
** Revised:
**           12/10/93 - Ellen Yu, rewrote this routine according new spec.
** History:
**  02/25/94 Joyce Young, Added the fng_num in IMAGE_ATTR_T
**  05/10/94 Joyce Young, Moved display_camera() & display_fingerprint() here
**  06/03/94 Joyce Young, adjust the offset of ROLL/PLAIN active image in
**              GRBVRAM for new board
**  06/09/94 Ellen Yu, add #define PROTOTYPE difference the hardware
**              prototype board and other new revisions.
**  11/18/94 JY, delete CAPT_ABORT case
**   4/15/95 TS, fixed up for tip desmearing
*/

/* ------------------------------ Includes ---------------------------- */
#include <gspreg.h>
#include <gsptypes.h>
#include <gspglobs.h>
#include <stdlib.h>

#include "stdtypes.h"
#include "coorddef.h"
#include "gsp_defs.h"
#include "gsp_imgs.h"
#include "imgglobs.h"
#include "mem_addr.h"
#include "mem_allc.h"
#include "mimio.h"
#include "gsp_func.h"


/* ------------------------- External Functions  -------------------------*/

extern VOID load_equref_to_EquDram (image_type_t camera);
extern VOID load_LUTTBL_to_MinSram (UBYTE *src_ptr);
extern VOID load_straight_thruLUT_to_EquSram (VOID);
extern keytype_t OSQPend_key(VOID);
extern VOID setup_display_screen (char *namep);

extern VOID copy_image_to_Dram (image_type_t camera, ULONG img_store_addr,
          int ht_bytes, int wd_words);

extern VOID CaptureScreen(IMAGE_ATTR_T, scan_type_t, char *);
extern VOID Capt_CaptScr(IMAGE_ATTR_T, short);
extern VOID Capt_PrevScr(IMAGE_ATTR_T, short);
extern VOID Misfng_MisfngScr(IMAGE_ATTR_T, short);
extern VOID Misfng_PrevScr(IMAGE_ATTR_T, short);
extern VOID Prev_CaptScr(IMAGE_ATTR_T, short);
extern VOID Prev_MisfngScr(IMAGE_ATTR_T, short);
extern VOID PrevInitScr(IMAGE_ATTR_T, short);

/* ------------------------ External Variables  ------------------------ */
extern image_type_t equref_camera;


/* ------------------------ Function Prototype  -------------------------*/
```

```
VOID my_mim_init(UBYTE *store_addr);
capture_t image_capture (image_type_t camera, scan_type_t fng_num,
        char *namep, ULONG img_store_addr);
VOID display_camera (image_type_t camera);
VOID display_columns_4_3(MIM *source,MIM *disp,int x,int w);
VOID display_4_3v(MIM *source,MIM *dest);
VOID fill_col_4_3(MIM *dest, int col, int value);
VOID save_columns(MIM *source,MIM *dest,int start, int stop);
VOID display_fingerprint (image_type_t camera, UBYTE *src_start_ptr,
        UBYTE *dst_start_ptr);
VOID fill_even_pixels(MIM *dest,int value);
capture_t CaptScanAct( IMAGE_ATTR_T image );
capture_t CaptSaveAct( IMAGE_ATTR_T image );
capture_t PrevSaveAct( IMAGE_ATTR_T image );
capture_t PrevScanAct( IMAGE_ATTR_T image );
capture_t PrevInitAct( IMAGE_ATTR_T image );

typedef enum
{
        NO_DIR,
        RIGHT_DIR,
        LEFT_DIR
} ROLL_DIRECTION;

int finger_type_check(scan_type_t fngnum,image_type_t *camera,
        HAND_TYPE *hand, ROLL_DIRECTION *desmear);

VOID (*prevscr_key_scr[])(IMAGE_ATTR_T, short) =
{
        NULL,
        Prev_CaptScr,
        NULL,
        Prev_MisfngScr,
        NULL,
        NULL
};

VOID (*misfngscr_key_scr[])(IMAGE_ATTR_T, short) =
{
        NULL,
        Misfng_PrevScr,
        NULL,
        Misfng_MisfngScr,
        NULL,
        NULL
};

VOID (*captscr_key_scr[])(IMAGE_ATTR_T, short) =
{
        NULL,
        Capt_PrevScr,
        NULL,
        Capt_CaptScr,
        NULL,
        NULL
};

capture_t (*prevscr_key_act[])(IMAGE_ATTR_T) =
{
        NULL,
```

```
        PrevScanAct,
        NULL,
        PrevSaveAct,
        NULL,
        NULL
};

capture_t (*captscr_key_act[])(IMAGE_ATTR_T) =
{
        NULL,
        CaptScanAct,
        NULL,
        CaptSaveAct,
        NULL,
        NULL
};

/* structures for defined arrays in memory */

MIM gvram;
MIM gvram_reduced;
MIM gminram;
MIM disp_window;
MIM plain_disp_window;
MIM data_store;
MIM central_rows;
MIM detect_row;
MIM check_row;

UBYTE detect_buf[1024];

/* ------------------------------------------------------------------------*/
/*
** Func name: VOID my_mim_init()
** Purpose:   initialize the memory areas for the grabber vram, the display
**            and the roll storage area.
** Returns:   None
*/
VOID my_mim_init(UBYTE *store_addr)
{
        mim_new(&gvram,((UBYTE *)GRB_VRAM_END)-GRBVRAM_WD_BYTES*26+100,
                -GRBVRAM_WD_BYTES,ROLL_WD_PIXELS,ROLL_HT_PIXELS);

        mim_new(&gvram_reduced,((UBYTE *)GRB_VRAM_END)-GRBVRAM_WD_BYTES*27+76,
                -GRBVRAM_WD_BYTES,ROLL_WD_PIXELS*3/4,ROLL_HT_PIXELS);

        mim_new(&gminram,((UBYTE *)MIN_DRAM_END)-MINDRAM_WD_BYTES*26+100,
                -MINDRAM_WD_BYTES,ROLL_WD_PIXELS,ROLL_HT_PIXELS);

        mim_new(&disp_window,((UBYTE *)DPY_VRAM_BASE)+DPYVRAM_WD_BYTES*4+12,
                DPYVRAM_WD_BYTES,720,720);

        mim_subset(&disp_window,&plain_disp_window,0,732-IMG_PLAIN_H,
                IMG_PLAIN_W-12,IMG_PLAIN_H);

        mim_new(&data_store,store_addr,ROLL_WD_PIXELS,
                ROLL_WD_PIXELS,ROLL_HT_PIXELS);

        mim_new(&central_rows,mim_adr(&gvram,0,400),5*mim_inc(&gvram),
                ROLL_WD_PIXELS,25);
```

```
        mim_new(&detect_row,detect_buf,0,ROLL_WD_PIXELS,1);
}

/* ------------------------------------------------------------------------*/
/*
** Func name: VOID my_mim_init()
** Purpose:   Image capture process from the given camera
**            (Roll = 0, Plain = 1).
** Returns:   type in the typedef capture_t
*/
capture_t image_capture (image_type_t camera, scan_type_t fng_num,
        char *namep, ULONG img_store_addr)
{
        IMAGE_ATTR_T  image;
        keytype_t  key_pressed;
        capture_t  ret;
        UBYTE *src_start_ptr, *dst_start_ptr;

        int far_r_edge,far_l_edge;
        int r_edge,l_edge;
        int s_w,s_h;
        int new_scan_line,scan_line;

        image_type_t fng_camera;
        HAND_TYPE hand;
        ROLL_DIRECTION capt_dir;

        /*
        ** parameters controlling the states of the rolling
        */
        int edge_tol = 20;
        int active_threshold = 150;
        int enough = 10;
        int extra = 25;
        int max_columns_in_step = 50;
        enum
        {
              BLANK,
              PRESS,
              ROLL_RIGHT,
              ROLL_LEFT,
              LIFT,
              GONE
        } capt_state;

        /*
        ** check what camera to use      and required direction
        */

        if (finger_type_check(fng_num, &fng_camera, &hand, &capt_dir)
           || fng_camera != camera)
        return(CAPT_WRONG_TYPE);

        my_mim_init((UBYTE *)img_store_addr);

        mim_size(&gvram,&s_w,&s_h);

        /*
        ** image offset should be matched in image_capture() and display_camera
```

```
** these pointers are only used for the plain
*/
src_start_ptr = (UBYTE *)GRB_VRAM_END - SCR_W * 24 + 79;
dst_start_ptr = (UBYTE *)DPY_VRAM_BASE + SCR_W * 4 + 12;

if ( camera == PLAIN )
{
      src_start_ptr += 20;
      dst_start_ptr += SCR_W * (732 - IMG_PLAIN_H);
}

/*
** Get attributes of image and put into structure( IMAGE_ATTR_T )
*/
image.camera = camera;
image.hand = hand;
image.store_addr = img_store_addr;
image.src_addr = src_start_ptr;
image.dst_addr = dst_start_ptr;
image.fng_num = fng_num;

CaptureScreen( image, fng_num, namep );

/*
** Load the Equ data that were created during calibration for ROLL
** or PLAIN from the Dram to EquDram, when the current calling camera
** is different from the previous camera,
*/
if ( equref_camera != camera )
      load_equref_to_EquDram (camera);

/*
** Wait for key pressed
*/
key_pressed = TYPE_NO_RESPONSE;
while( key_pressed != TYPE_SAVE_KEY )
{
      /*
      ** 1. Copy the straight thru function from Dram to MinSram
      ** 2. Turn grabber on
      */
      PrevInitAct( image );

      /*
      ** Preview operation
      */
      key_pressed = TYPE_NO_RESPONSE;
      while( key_pressed != TYPE_SCAN_KEY )
      {
            /*
            **  Display image by copy fingerprint from grabber VRAM to
            **  display VRAM.
            */
            display_fingerprint (camera, src_start_ptr, dst_start_ptr);

            key_pressed = OSQPend_key();

            if( (*prevscr_key_scr[key_pressed]) != NULL )
                  (* prevscr_key_scr[key_pressed])( image, DRAW );
```

```
        if( (*prevscr_key_act[key_pressed]) != NULL )
        {
                ret = (*prevscr_key_act[key_pressed])( image );
                if( ret == CAPT_MISSFNG ){
                        return( ret );
                }
        }
}   /* end of while */

/*
** Capture operation
*/
if (capt_dir == RIGHT_DIR || capt_dir == LEFT_DIR)
{
        turn_grab_on_normal (image.camera, 1);

        capt_state=BLANK;
        key_pressed = TYPE_NO_RESPONSE;
        while( key_pressed != TYPE_SCAN_KEY && key_pressed != TYPE_SAVE_KEY)
        {
                switch (capt_state)
                {
                case BLANK:
                        /*
                        **     wait for the finger to come down
                        **     display the active area   when it is sensed
                        */
                        if (fz_detect_edges(&central_rows,edge_tol,active_threshold,
                                        &r_edge,&l_edge))
                        {
                                        capt_state=PRESS;
                                        far_r_edge=r_edge;
                                        far_l_edge=l_edge;
                        }
                        break;

                case PRESS:
                        /*
                        **     display the active area
                        **  until both edges begin to roll in the correct direction
                        **  then save the back side of the active area
                        **  Allow the roll to go in either direction
                        */
                        if (fz_detect_edges(&central_rows,edge_tol,active_threshold,
                                        &r_edge,&l_edge))
                        {
                                if (l_edge > far_l_edge-enough)
                                {
                                scan_line =
                                    smooth_scan_line(-1,r_edge,l_edge,capt_dir,
                                    max_columns_in_step);
                                /*
                                **  Save the left side of the active area
                                **  Don't update the display
                                */
                                save_columns(&gvram,&data_store,0,scan_line);
                                capt_state=ROLL_RIGHT;
                                break;
                        }
```

```
            if (r_edge < far_r_edge-enough)
            {
                  scan_line =
                        smooth_scan_line(-1,r_edge,l_edge,capt_dir,
                        max_columns_in_step);
                  /*
                  **  Save the left side of the active area
                  **  Don't update the display
                  */
                  save_columns(&gvram,&data_store,scan_line,s_w);
                  capt_state=ROLL_LEFT;
                  break;
            }
            /*
            **  While not rolling just display the active columns
            **
            */
            if (l_edge-extra >= 0)
            {
                  if (r_edge+extra < s_w)
                        display_columns_4_3(&gvram,
                        &disp_window,l_edge-extra,
                        r_edge-l_edge+2*extra);
                  else
                        display_columns_4_3(&gvram,
                        &disp_window,l_edge-extra,
                        s_w-l_edge+extra);
            }
            else if (r_edge+extra < s_w)
                  display_columns_4_3(&gvram,
                  &disp_window,0,r_edge+extra);
            else
                  display_columns_4_3(&gvram,&disp_window,0,s_w);


            if (l_edge < far_l_edge)
                  far_l_edge = l_edge;
            if (r_edge > far_r_edge)
                  far_r_edge=r_edge;
      }
      else
      {
            mim_move(&gvram,&data_store);
            display_columns_4_3(&data_store,&disp_window,0,s_w);
            capt_state=LIFT;
      }
      break;
case ROLL_RIGHT:
      /*
      **  save the area incremented by the scan line
      **  display right side of the active area
      **     until the right edge stops
      **  then save the right side of the active area
      */
      if (fz_detect_edges(&central_rows,edge_tol,active_threshold,
                        &r_edge,&l_edge))
      {
            new_scan_line = smooth_scan_line(scan_line,
                  r_edge,l_edge,RIGHT_DIR,30);
                  if (new_scan_line > scan_line)
                  {
                  save_columns(&gvram,&data_store,
                  scan_line,new_scan_line);
                  scan_line=new_scan_line;
                  if (r_edge+extra < s_w)
```

```
                        display_columns_4_3(&gvram,&disp_window,scan_line,
                                r_edge+extra-scan_line);
                else
                        display_columns_4_3(&gvram,&disp_window,scan_line,
                                s_w-scan_line);
        /* display the scan line for debug
                fill_col_4_3(&disp_window,scan_line,0);
        */
        }
        if (r_edge < far_r_edge - enough)
        {
                save_columns(&gvram,&data_store,scan_line,s_w);
                display_columns_4_3(&data_store,&disp_window,scan_line,
                                s_w-scan_line);
                capt_state=LIFT;
        }
        if (l_edge < far_l_edge)
                far_l_edge = l_edge;
        if (r_edge > far_r_edge)
                far_r_edge=r_edge;

    }
    else
    {
        save_columns(&gvram,&data_store,
                scan_line,s_w);
        display_columns_4_3(&data_store,&disp_window,
                scan_line,s_w-scan_line);
        capt_state=LIFT;
    }

    break;

case ROLL_LEFT:
    /*
    **  save the area incremented by the scan line
    **  display left side of the active area
    **     until the left edge stops
    **  then save the left side of the active area
    */
    if (fz_detect_edges(&central_rows,edge_tol,active_threshold,
        &r_edge,&l_edge))
    {
        new_scan_line = smooth_scan_line(scan_line,r_edge,
                l_edge,LEFT_DIR,30);
        if (new_scan_line < scan_line)
        {
                save_columns(&gvram,&data_store,new_scan_line,scan_line);
                scan_line=new_scan_line;
                if (l_edge-extra >= 0)
                        display_columns_4_3(&gvram,&disp_window,l_edge-extra,
                                        scan_line-l_edge+extra);
                else
                        display_columns_4_3(&gvram,&disp_window,0,scan_line-
        /* display the scan line for debug
                fill_col_4_3(&disp_window,scan_line,0);
        */
        }
        if (l_edge > far_l_edge + enough)
        {
                save_columns(&gvram,&data_store,0,scan_line);
                display_columns_4_3(&data_store,&disp_window,0,scan_line-
```

```
                        capt_state=LIFT;
                }

                if (l_edge < far_l_edge)
                        far_l_edge = l_edge;
                if (r_edge > far_r_edge)
                        far_r_edge=r_edge;

        }
        else
        {
                save_columns(&gvram,&data_store,0,scan_line);
                display_columns_4_3(&data_store,&disp_window,
                        0,scan_line);
                capt_state=LIFT;
        }

        break;

case LIFT:
        /*
        **  wait for the active area to go away
        **  when it does, detect the changed areas and
        **  tag the final image
        */
        if (fz_detect_edges(&central_rows,edge_tol,active_threshold,
                                &r_edge,&l_edge))
        {
        }
        else
        {
                turn_grab_off(image.camera );
                capt_state=GONE;
        }
        break;

case GONE:
        /*
        **  don't do anything
        */
        break;
default:
        break;
}
key_pressed = OSQPend_key();
/*
**      fix up the data store in case the save key is pressed
**  before the end of the roll
**  Then turn the grabber off and move the data_store
**  back into the grabber min dram.
*/
if(key_pressed == TYPE_SAVE_KEY)
{
        switch (capt_state)
        {
                case BLANK:
                case PRESS:
```

```
                        mim_move(&gvram,&data_store);
                        break;
                case ROLL_RIGHT:
                        save_columns(&gvram,&data_store,
                                scan_line,s_w);
                        break;

                case ROLL_LEFT:
                        save_columns(&gvram,&data_store,0,scan_line);
                        break;
        }
        turn_grab_off(camera);
        mim_move(&data_store,&gminram);

}

        if( (*captscr_key_scr[key_pressed]) != NULL )
                (*captscr_key_scr[key_pressed])( image, DRAW );

        if( (*captscr_key_act[key_pressed]) != NULL )
        {
                ret = (*captscr_key_act[key_pressed])( image );
                if( ret == CAPT_SAVEIMG )
                {
                        return( ret );
                }
        }
        }   /* end of while */
}

else if ((camera == ROLL && capt_dir == NO_DIR) || camera == PLAIN)
{

        turn_grab_on (image.camera, 1);

        key_pressed = TYPE_NC_RESPONSE;
        while( key_pressed != TYPE_SCAN_KEY && key_pressed != TYPE_SAVE_KEY)
        {
                /*
                **  Display image by copy fingerprint from grabber VRAM to
                **  display VRAM.
                */
display_fingerprint (camera, src_start_ptr, dst_start_ptr);

                key_pressed = OSQPend_key();

                /*
                **      turn off the grabber when the save key is pressed
                */
                if(key_pressed == TYPE_SAVE_KEY)
                {
                turn_grab_off(camera);
                }

                if( (*captscr_key_scr[key_pressed]) != NULL )
                                (*captscr_key_scr[key_pressed])( image, DRAW );

                if( (*captscr_key_act[key_pressed]) != NULL )
                {
                        ret = (*captscr_key_act[key_pressed])( image );
```

```
                        if( ret == CAPT_SAVEIMG )
                        {
                                return( ret );
                        }
                }
        }   /* end of while */
    }       /* end of if */

}   /* end of while */
} /* image_capture */

/* ------------------------------------------------------------------------*/
/*
** Func name: VOID display_fingerprint()
** Purpose:   Draw the fingerprint in reduced form in display window.
** The fingerprint thru the TS's camera is stored upside-down in the GVRAM.
** the horizontal reduction is done in hardware
** the vertical reduction is done by dropping 1 of 4 lines in the roll
** and 1 of 2 lines in the plain.
*/
VOID display_fingerprint (image_type_t camera, UBYTE *src_start_ptr,
        UBYTE *dst_start_ptr)
{
        UBYTE *src_ptr, *dst_ptr;
        int ii;
        if ( camera == ROLL )
        {

                /* set up 4/3 reduction */
                display_4_3v(&gvram_reduced,&disp_window);

        }
        else
        {
                /*
** Only display every other lines;
** so increment the source array 2 lines, dest. array 1 line
** The first display line is in the middle of the screen.
*/
poke_breg (SADDR, src_start_ptr);  /* B0: starting addr of source array */
poke_breg (DADDR, dst_start_ptr);     /* B2: starting addr of dest.  array */
poke_breg (SPTCH, PLAIN_GRBVRAM_PITCH); /* B1: the pitch of source array */
poke_breg (DPTCH, PLAIN_DPYVRAM_PITCH); /* B3: the pitch of dest. array */
poke_breg (DYDX,  PLAIN_DYDX_VAL);      /* B7: DX: array width, DY: array height */
asm ("        PIXBLT L, L  ");
        }
} /* display_fingerprint */

/* ------------------------------------------------------------------------*/
/*
** Func name: VOID display_camera()
** Purpose:   Display the Roll or Plain image from the given camera
**            (Roll or Plain) by using straight_thru function
** Returns:   None
*/
VOID display_camera (image_type_t camera)
{
        keytype_t  key_type;
        UBYTE *src_start_ptr, *dst_start_ptr;
```

```
if( camera == ROLL )
        ramdac_display_cross_symbol();

src_start_ptr = (UBYTE *)GRB_VRAM_END;
src_start_ptr -= SCR_W * 30 - 79;
dst_start_ptr = (UBYTE *)DPY_VRAM_BASE;
dst_start_ptr += SCR_W * 4 + 12;

if ( camera == PLAIN )
{
        src_start_ptr += 20;
        dst_start_ptr += SCR_W * (732 - IMG_PLAIN_H);
}

load_straight_thruLUT_to_EquSram ();
load_LUTTBL_to_MinSram ((UBYTE *)STRAIGHT_THRU_LUTTBL);
turn_grab_on (camera, 1);

for (;;)
{
        display_fingerprint (camera, src_start_ptr, dst_start_ptr);

        switch ( key_type = OSQPend_key () )
        {
                case TYPE_SCAN_KEY:
                case TYPE_SAVE_KEY:
                case TYPE_ABORT:
                        turn_grab_off (camera);
                        if( camera == ROLL )
                                ramdac_clear_cross_symbol();
                        return;
        }
        }
} /* display_camera */

/* ------------------------------------------------------------------------*/
/*
** Func name: PrevInitAct()
** Purpose:   Actions when enter a preview screen.
** Returns:   CAPT_NO_RESPONSE, no captured images
*/
capture_t PrevInitAct( IMAGE_ATTR_T image )
{
        /*
        ** load straight thru LUT from Dram to MinSram
        */
#if IMG_DEBUG
        load_straight_thruLUT_MEM_to_EquSram ();
#else
        load_LUTTBL_to_MinSram ((UBYTE *)STRAIGHT_THRU_LUTTBL);
#endif

        /*
        ** Turn grabber on 2 frame, otherwise, the second finger image will
        ** display garbage lines if using multiple finger buffers
        */
        turn_grab_on (image.camera, 2);

   return( CAPT_NO_RESPONSE );
}  /* PrevInitAct */
```

```
/* ------------------------------------------------------------------------*/
/*
** Func name: PrevScanAct()
** Purpose:   Actions when SCAN key pressed in preview screen.
** Returns:   CAPT_NO_RESPONSE/CAPT_WRONG_TYPE, no captured images or
**            wrong camera type.
*/
capture_t PrevScanAct( IMAGE_ATTR_T image )
{
        /*--- Turn grabber off ---*/
        turn_grab_off( image.camera );

        /*
        ** Load minfunc from Dram to MinSram
        */
#if IMG_DEBUG
        load_straight_thruLUT_to_EquSram ();
        load_LUTTBL_to_MinSram ((UBYTE *)STRAIGHT_THRU_LUTTBL);
#else
        load_LUTTBL_to_MinSram ((UBYTE *)MINFUNC_LUTTBL);
#endif

   return( CAPT_NO_RESPONSE );
}  /* PrevScanAct */

/* ------------------------------------------------------------------------*/
/*
** Func name: PrevSaveAct()
** Purpose:   Actions when SAVE key pressed in preview screen.
** Returns:   CAPT_NO_RESPONSE/CAPT_MISSFNG, no captured images or missing
**            finger
** History:   01-31-94 Joyce Young: Turn on the grabber if SCAN key pressed
*/
capture_t PrevSaveAct( IMAGE_ATTR_T image )
{
        keytype_t key_pressed;
        capture_t response_misfng[] =
        {
              CAPT_NO_RESPONSE,
              CAPT_NO_RESPONSE,
              CAPT_NO_RESPONSE,
              CAPT_MISSFNG,
              CAPT_MISSFNG,
        };

        /*--- Turn grabber off ---*/
        turn_grab_off( image.camera );

        key_pressed = TYPE_NO_RESPONSE;
        while( key_pressed != TYPE_SCAN_KEY && key_pressed != TYPE_SAVE_KEY )
              key_pressed = OSQPend_key();

        if( (*misfngscr_key_scr[key_pressed]) != NULL )
              (* misfngscr_key_scr[key_pressed])( image, DRAW );

        /*
        ** SCAN key will go to preview screen, as we turn off the grabber above,
        ** so the grabber needs to turn on here, otherwise, the finger image will
        ** not show on the preview screen
```

```
        */
        if( key_pressed == TYPE_SCAN_KEY )
              turn_grab_on (image.camera, 1);

        return( response_misfng[key_pressed] );
}  /* PrevSaveAct */

/* ------------------------------------------------------------------------*/
/*
** Func name: CaptScanAct()
** Purpose:   Actions when SCAN key pressed in capture screen.
** Returns:   CAPT_NO_RESPONSE
** History:   01-31-94 Joyce Young: Add return value
*/
capture_t CaptScanAct( IMAGE_ATTR_T image )
{
        /*--- Turn grabber off ---*/
        turn_grab_off (image.camera);
        return (CAPT_NO_RESPONSE);
}

/* ------------------------------------------------------------------------*/
/*
** Func name: CaptSaveAct()
** Purpose:   Actions when SAVE key pressed in capture screen.
** Returns:   CAPT_SAVEIMG/CAPT_WRONG_TYPE
** History:   02-18-94 Joyce Young: Pass camera in copy_image_to_Dram ()
*/
capture_t CaptSaveAct( IMAGE_ATTR_T image )
{
        int  wd_words, ht_bytes, w, h;

        /*
        **     grabber should already be off
        */

        if (image.camera == PLAIN)
        {
        /*--- Tag the images for changes from the remainder image ---*/
        load_LUTTBL_to_MinSram ((UBYTE *)FNGR_TAGGING_LUTTBL);
        turn_grab_on (image.camera, 1);
        turn_grab_off (image.camera);
        display_fingerprint (image.camera, image.src_addr, image.dst_addr);
              fill_even_pixels(&plain_disp_window,254);
              /*--- Download image to the GSP DRAM ---*/
              wd_words = PLAIN_WD_WORDS;
              ht_bytes = PLAIN_HT_BYTES;
        copy_image_to_Dram (image.camera, image.store_addr, ht_bytes, wd_words
        }
        else if (image.camera == ROLL)
        {
              load_LUTTBL_to_MinSram ((UBYTE *)FNGR_TAGGING_LUTTBL);
              turn_grab_on_normal(image.camera,1);
              turn_grab_off(image.camera);
              mim_move(&gminram,&data_store);
              mim_size(&data_store,&w,&h);
              display_columns_4_3(&data_store,&disp_window,0,w);
              fill_even_pixels(&disp_window,254);
        }
        else
```

```
        return( CAPT_WRONG_TYPE );

        return( CAPT_SAVEIMG );
}  /* CaptSaveAct */

/* ------------------------------------------------------------------------*/
/*
** Func name: int fz_detect_edges()
** Purpose:   Determine the right and left edges of the contact strip of
**            the fingerprint
** Returns:   0 - if there is no contact
**            1 - otherwise
*/
int fz_detect_edges(MIM *row, int edge_count, int threshold,
        int *right_edge, int *left_edge)
{
        int width,height,ii,sum;
        unsigned char *lp,*rp;

        /*
        **     Get the center rows from the image
        **  Or together all the tag bits
        */
        mim_fill(&detect_row, 0);
        mim_move_c(row,&detect_row,0x200c);
        mim_size(row, &width, &height);

        lp = mim_adr(&detect_row,0,0);
        rp = lp+width-1;

        /* find the left edge by adding up and the right edge by adding down *
        sum = 0;
        for (ii=0; ii<width; ii++)
        {
              if (*lp++ & 1) {
                    sum++;
                    if (sum >= edge_count)
                          break;
              }
        }
        *left_edge=ii;

        sum = 0;
        for (ii=width-1; ii>=0; ii--)
        {
              if (*rp-- & 1) {
                    sum++;
                    if (sum >= edge_count)
                          break;
              }
        }
        *right_edge=ii;

        /*
        **     there is contact if the right edge is farther right than the left
        **     edge
        */
        if (*right_edge > *left_edge)
              return(1);
        else
```

```
                return(0);
} /* fz_detect_edges */

/* ------------------------------------------------------------------------*/
/*
** Func name: save_columns()
** Purpose:   block transfer selected columns of the source to the image
** Return:    None
*/
VOID save_columns(MIM *src,MIM *dest,int start, int stop)
{
        int s_w,s_h;
        MIM src_cols,dest_cols;

        mim_size(src,&s_w,&s_h);
        mim_subset(src,&src_cols,start,0,stop-start,s_h);
        mim_subset(dest,&dest_cols,start,0,0,0);
        mim_move(&src_cols,&dest_cols);
} /* save_columns */

/* ------------------------------------------------------------------------*/
/*
** Func name: finger_type_check()
** Purpose:   decode the finger type, camera, whether to desmear and validity
** Returns:   0 - if no error
**            1 - otherwise
*/
int finger_type_check(scan_type_t fng_num,image_type_t *camera,
        HAND_TYPE *hand, ROLL_DIRECTION *desmear)
{
        /*
        ** decode the finger type, camera, whether to desmear and validity
        ** return (0) if no error
        */
        switch ( fng_num )
        {
              case STYPE_ROLL_RIGHT_THUMB:
                    *camera = ROLL;
                    *hand = RIGHT;
                    *desmear = LEFT_DIR;
                    break;

              case STYPE_ROLL_RIGHT_INDEX:
              case STYPE_ROLL_RIGHT_MIDDLE:
              case STYPE_ROLL_RIGHT_RING:
              case STYPE_ROLL_RIGHT_LITTLE:
                    *camera = ROLL;
                    *hand = RIGHT;
                    *desmear = RIGHT_DIR;
                    break;

              case STYPE_ROLL_LEFT_THUMB:
                    *camera = ROLL;
                    *hand = LEFT;
                    *desmear = RIGHT_DIR;
                    break;

              case STYPE_ROLL_LEFT_INDEX:
              case STYPE_ROLL_LEFT_MIDDLE:
              case STYPE_ROLL_LEFT_RING:
```

```
                case STYPE_ROLL_LEFT_LITTLE:
                        *camera = ROLL;
                        *hand = LEFT;
                        *desmear = LEFT_DIR;
                        break;

                case STYPE_PLAIN_TWO_THUMBS:
                        *camera = ROLL;
                        *hand = RIGHT;
                        *desmear = NO_DIR;
                        return(1);
                        break;

                case STYPE_PLAIN_RIGHT4:
                        *camera = PLAIN;
                        *hand = RIGHT;
                        *desmear = NO_DIR;
                        break;

                case STYPE_PLAIN_LEFT4:
                        *camera = PLAIN;
                        *hand = LEFT;
                        *desmear = NO_DIR;
                        break;

                case STYPE_PLAIN_RIGHT_THUMB:
                        *camera = ROLL;
                        *hand = RIGHT;
                        *desmear = NO_DIR;
                        break;

                case STYPE_PLAIN_LEFT_THUMB:
                        *camera = ROLL;
                        *hand = LEFT;
                        *desmear = NO_DIR;
                        break;

                default:
                        return( 1 );
        }
        return (0);
} /* finger_type_check */

/* ------------------------------------------------------------------------*/
/*
** Func name: VOID display_4_3v()
** Purpose:   display the whole fingerprint with a 4:3 reduction in the
**            vertical direction, drop every fourth line
** Returns:   None
*/
VOID display_4_3v(MIM *source,MIM *dest)
{
        MIM src_rows,dst_rows;
        int w,h,s_i,ii;

        mim_size(source,&w,&h);


        for (ii=0; ii<3; ii++)
        {
```

```
                mim_new(&src_rows,mim_adr(source,0,ii),4*mim_inc(source),w,h/4);
                mim_new(&dst_rows,mim_adr(dest,0,ii),3*mim_inc(dest),0,0);
                mim_move(&src_rows,&dst_rows);
        }
} /* display_4_3v */

/* ------------------------------------------------------------------------*/
/*
** Func name: VOID fill_col_4_3()
** Purpose:   fill a column in a range reduced by 4/3
** Returns:   None
*/
VOID fill_col_4_3(MIM *dest, int col, int value)
{
        register int ii,k,d_i;
        register unsigned char *d;
        int w,h;

        k = value;
        mim_size(dest,&w,&h);
        d = mim_adr(dest,col*3/4,0);
        d_i = mim_inc(dest);

        ii=h;
        do
        {
        *d=k;
        d+=d_i;
        } while (--ii);

} /* fill_col_4_3 */

/* ------------------------------------------------------------------------*/
/*
** Func name: int smooth_scan_line()
** Purpose:   even out the scan line steps and limit them to a maximum size.
** Returns:   new scan line
*/
int smooth_scan_line(int old_scan_line, int right, int left,
    ROLL_DIRECTION roll_dir, int max_columns)
{
        int new_median, new_scan_line;
        int scan_split = 5;
        int factor = 4;

        if (old_scan_line < 0 )
        {
                new_median = (right + left) / 2;
                return (new_median);
        }

        if (roll_dir == RIGHT_DIR)
        {
                new_median = (scan_split*right + (8-scan_split)*left) /8;
                if (new_median > old_scan_line)
                {
                        new_scan_line = (factor*new_median +
                                (8-factor)*old_scan_line + 4)/8;
                        if (new_scan_line > old_scan_line + max_columns)
                                new_scan_line = old_scan_line + max_columns;
                }
```

```
                else
                        new_scan_line = old_scan_line;
        }
        else if (roll_dir == LEFT_DIR)
        {
                new_median = (scan_split*left + (8-scan_split)*right) /8;
                if (new_median < old_scan_line)
                {
                        new_scan_line = (factor*new_median +
                                (8-factor)*old_scan_line + 4)/8;
                        if (new_scan_line < old_scan_line - max_columns)
                                new_scan_line = old_scan_line - max_columns;
                }
                else new_scan_line = old_scan_line;
        }
        else new_scan_line = new_median;

        return(new_scan_line);

} /* int smooth_scan_line */

/*********************************************************************
**
** replace any untagged pixels (even values) in the destination area
** with value
**
*/
VOID fill_even_pixels(MIM *dest,int value)
{
        register UBYTE *pp;
        register int vv,jj;

        int w,h,ii;

        vv = value;
        mim_size(dest,&w,&h);

        for (ii=0; ii<h; ii++)
        {
                pp = mim_adr(dest,0,ii);
                jj =w;
                do
                {
                        if (!(*pp & 1)) *pp = vv;
                        pp++;
                }
                while (--jj);
        }
}
/* ------------------------------------------------------------------*/
/* end of fngrcapt.c */
```

```
/*
** Filename: gsp_init.c
** Purpose:  GSP Initialization.
** Date:     08/10/93
** Author:   Joyce Young
** History:
**  02/18/94 Joyce Young, modify the offset of PLAIN active image
**  05/05/94 Joyce Young, if image parameter is 255, set default value
**  06/03/94 Joyce Young, adjust the offset of ROLL/PLAIN active image in
**              MINDRAM for new board
**  06/28/94 Joyce Young, turn_illuminator_on in initialize_34020, otherwise,
**              turning illuminator off during calibration and then receiving
**              ABORT command will cause the illuminator off after ABORT done
**  07/14/94 Joyce Young, change default roll_peak_illum_level = 10 &
**              plain_peak_illum_level = 10
**  08/11/94 TS, fix up offsets so that they add to the offsets calculated
**           from the dark level histogram instead of replacing it.
**  09/05/94 TS, change tagging so that a 0 change_threshold value causes
**              everything to be tagged.
**  09/22/94 JY, move setup of talon_font[CORPUS29] & talon_font[CORPUS49]
**              from PrintTitle() to setup_graphic(), otherwise, if calling
**              PrintTitle() several times, will mess up the talon_font table
**  09/29/94 TS, fixed error in Equ Sram initialization - needed to subtract
**              out offset after scaling. Cause severe clipping at white.
**  01/18/95 JY, read existing of palm.equ from PC_PARAMETERS
**  01/20/95 JY, delete unnecessary include files
**  01/23/95 JY, load palm equ to EQU_DRAM if file exists
*/

/* ------------------------------ Includes ---------------------------- */
#include <gspreg.h>
#include <gsptypes.h>
#include <gspglobs.h>

#include "stdtypes.h"
#include "gsp_font.h"
#include "img_defs.h"
#include "mem_addr.h"
#include "mem_allc.h"
#ifndef PALM_SCANNER
#include "gsp_imgs.h"
#endif

/* ------------------- Extern Variable definition --------------------*/

/*
** variables for the graphic
*/
extern FONT corpus29, corpus49;

/* ------------------- Global Variable definition --------------------*/

FONTINFO fontinfo;
short smlfont_id, bigfont_id;
short smlfont_charhigh, bigfont_charhigh;
short smlfont_charwide, bigfont_charwide;

/*
** variables for the image processing
*/
```

```
image_type_t equref_camera = NON_CAMERA;   /* which camera equ in EquDram */
int rollequ_exists = 0;          /* 1=ROLL equ exists */
int plainequ_exists = 0;         /* 1=PLAIN equ exists */
int palmequ_exists = 0;          /* 1=PALM equ exists */
int palmwrp_exists = 0;          /* 1=PALM dewarping exists */

/* ------------------- Local Variable definition ---------------------*/
/* this sets up the defaults if no parameter file exists */

static int para_exists = 0;              /* 1=image para file exists */
static ULONG recursive_factor = 66;/* sensitivity of capture to field rate */
static ULONG fngr_desired_equ_value = 252;/*value of blank platen equalized*/
static ULONG desired_equ_value = 252;  /* value of blank platen equalized */
static ULONG changing_threshold = 13;  /* percentage change of final to post
                  scan image, if not exceeded will white/tag out the data */
static ULONG too_dark_value = 5;         /* to remove A/D offset */
static ULONG roll_offset = 10;   /* ROLL peak histogram value */
static ULONG plain_offset = 10;  /* PLAIN peak histogram value */
static ULONG palm_offset = 10;   /* PALM peak histogram value */
static ULONG palm_desired_equ_value = 252;/*value of blank platen equalized*/
static ULONG detect_level = 150;  /* detection level for active area */

/* ------------------------- External Functions  ---------------------*/
extern VOID init_ramdac (VOID);
extern VOID init_gsp_grabber (VOID);
extern VOID send_command_to_BIC (UBYTE command_to_send);
extern VOID turn_illuminator_on (VOID);

/* ------------------------- Function Prototypes ---------------------*/

VOID copy_image_to_Dram (image_type_t camera, ULONG img_store_addr,
    int ht_bytes, int wd_words);
VOID initialize_34020 (VOID);
VOID load_equref_to_EquDram (image_type_t camera);
VOID load_LUTTBL_to_MinSram (UBYTE *src_ptr);
VOID load_straight_thruLUT_to_EquSram (VOID);
VOID setup_graphic (VOID);

static VOID calculate_minfuncLUT_to_Dram (VOID);
static VOID calculate_straight_thruLUT_to_Dram (VOID);
static VOID calculate_taggingLUT_to_Dram (VOID);
static ULONG flcor0 (ULONG a, ULONG b);
static VOID load_equLUT_to_EQUSram (ULONG offset, ULONG dark_offset, ULONG
desired);
static VOID load_straight_thruLUT_to_MinSram (VOID);

/* ---------------------------------------------------------------------*/
/*
** Initialize the Ramdac, the GSP registers, setup the graphic functions
** and create all LUTs.
*/
VOID initialize_34020 (VOID)
{
    volatile ULONG *ptr;

    *((USHORT *)DPYCTL) = 0;
    asm ("        MWAIT        ");
    init_ramdac ();
    setup_graphic ();
```

```
    turn_illuminator_on ();
    ptr = (ULONG *)PC_PARAMETERS;

    rollequ_exists = *ptr++;
    plainequ_exists = *ptr++;
    /* this is where version 1.3 loader puts it */
          palmequ_exists = *ptr++;
          palmwrp_exists = *ptr++;

    para_exists = *ptr++;

    if ( para_exists )
        {
         if (*ptr != 255) recursive_factor = *ptr;
         ptr++;
         if (*ptr != 255) fngr_desired_equ_value = *ptr;
         ptr++;
         if (*ptr != 255) changing_threshold = *ptr;
         ptr++;
         if (*ptr != 255) too_dark_value = *ptr;
         ptr++;
         if (*ptr != 255) roll_offset = *ptr;
         ptr++;
         if (*ptr != 255) plain_offset = *ptr;
         ptr++;
         if (*ptr != 255) palm_offset = *ptr;
         ptr++;
         if (*ptr != 255) palm_desired_equ_value = *ptr;
         ptr++;
         }

         /* this is where version 1.4 and beyond loader keeps it
    palmequ_exists = *ptr++;
          palmwrp_exists = *ptr++;
          */

#ifndef PALM_SCANNER
    init_gsp_grabber ();

    calculate_minfuncLUT_to_Dram ();
    calculate_straight_thruLUT_to_Dram ();
    calculate_taggingLUT_to_Dram ();
#else
    /*
    ** Turn off grabber
    */
    *(UBYTE *)GRABBER_CTRL0_BASE = 0x0;
    *(UBYTE *)GRABBER_CTRL0_BASE = 0x0;

    load_straight_thruLUT_to_MinSram();
#endif
} /* initialize_34020 */

#ifndef PALM_SCANNER

/* ------------------------------------------------------------------------*/
/*
** copy the LUTTBL from the given source address (in Dram by BYTE pointer)
** to MinSram (by LONG pointer)
*/
```

```
VOID load_LUTTBL_to_MinSram (UBYTE *src_ptr)
{
   register ULONG *d;
          register UBYTE *s;
          register LONG ii;

   s = src_ptr;
   d = (ULONG *)MIN_SRAM_BASE;
          ii = 65536;
          do
       *d++ = *s++;
          while (--ii);

} /* load_LUTTBL_to_MinSram */

/* ------------------------------------------------------------------------*/
/*
** Calculate the Min Function and store to the Dram,
** later will copy to MinSram
*/
VOID calculate_minfuncLUT_to_Dram (VOID)
{
   UBYTE a_d, mem, *ptr, out;
   ULONG cond;

   ptr  = (UBYTE *)MINFUNC_LUTTBL;

   for ( mem = 0; mem < 256; mem++ )
       for ( a_d = 0; a_d < 256; a_d++ )
       {
           cond = (UBYTE)floor0 (mem, a_d) * recursive_factor;

           if ( cond >= 50 )
               out = mem - (cond + 50) / 100;
           else if ( cond > 0 )
               out = (UBYTE)floor0 (mem, 1);
           else
               out = mem;

           if (a_d < detect_level)
                          *ptr++ = out | 1;
                  else
                         *ptr++ = out & 0xfe;
       }
} /* calculate_minfuncLUT_to_Dram */

/* ------------------------------------------------------------------------*/
/*
** Calculate the Straight Thru and store to the Dram,
** later will copy to MinSram
*/
VOID calculate_straight_thruLUT_to_Dram (VOID)
{
    UBYTE a_d, mem, *ptr;

    ptr = (UBYTE *)STRAIGHT_THRU_LUTTBL;

    for ( mem = 0; mem < 256; mem++ )
         for ( a_d = 0; a_d < 256; a_d++ )
                  if (a_d < 150)
```

```
                        *ptr++ = a_d | 1;
                      else
                        *ptr++ = a_d & 0xfe;

} /* calculate_straight_thruLUT_to_Dram */

/* ------------------------------------------------------------------------*/
/*
** Calculate the Tagging Function and store to the Dram,
** later will copy to MinSram
*/
VOID calculate_taggingLUT_to_Dram (VOID)
{
    UBYTE a_d, mem, *ptr;
    UBYTE vv;

    ptr = (UBYTE *)FNGR_TAGGING_LUTTBL;

    for ( mem = 0; mem < 256; mem++ )
         for ( a_d = 0; a_d < 256; a_d++ )
         {
              if (changing_threshold > 0)
              {
                vv = ((ULONG)a_d * (ULONG)(100 - changing_threshold) / (ULONG)100 ;

                if ( (vv < mem) || (a_d < too_dark_value) )
                   *ptr++ = mem & 0xFFFFFFFE;     /* untag (white) */
                else
                   *ptr++ = mem | 1;              /* tag  dark) */
              }
              else
                *ptr++ = mem | 1;
         }
} /* calculate_taggingLUT_to_Dram */

/* ------------------------------------------------------------------------*/
/*
** Copy image from MinDram (2048 * 1024) to Dram (ROLL:976, 976... or
** PLAIN:1600, 1600... continuously).
** The image is upside down in MinDram and is stored to Dram rightside up
*/
VOID copy_image_to_Dram (image_type_t camera, ULONG img_store_addr,
    int ht_bytes, int wd_words)
{
    int ii, jj;
    ULONG *src_ptr, *dst_ptr;

    /*
    ** calculate the offset of the active image
    ** NOTE: image offset in copy_image_to_Dram(), load_equref_to_EquDram(),
    **       process_calibration() should be the same.
    */
    dst_ptr = (ULONG *)img_store_addr;
    src_ptr = (ULONG *)MIN_DRAM_BASE;

    if ( camera == ROLL )
         {
         src_ptr += MINDRAM_WD_WORDS * (ht_bytes + MINDRAM_ROLL_ROW_B_OFFSET)
              + MINDRAM_ROLL_COL_W_OFFSET;     /* 512 *  960 + 38) + 24 */
         }
```

```
    else
        {
        src_ptr += MINDRAM_WD_WORDS * (ht_bytes + MINDRAM_PLAIN_ROW_B_OFFSET)
             + MINDRAM_PLAIN_COL_W_OFFSET;   /* 512 * (976 + 31) + 39 */
        }

    for ( ii = 0; ii < ht_bytes; ii++ )
    {
        for ( jj = 0; jj < wd_words; jj++ )
            *dst_ptr++ = *src_ptr++;

        /*
        ** go to the beginning of the previous row
        */
        src_ptr -= MINDRAM_WD_WORDS + wd_words;
    }
} /* copy_image_to_Dram */

/* ------------------------------------------------------------------------*/
/*
** copy the equalize reference of the given camera from the Dram to EquDram
*/
VOID load_equref_to_EquDram (image_type_t camera)
{
   ULONG *src_ptr, *dst_ptr;
   ULONG offset_level;
   int wd_words, ht_bytes;
   int ii, jj, equ_exists;
   ULONG dark_offset = 0;
          register ULONG *s,*d;

   if ( camera == ROLL )
       {
       ht_bytes = ROLL_HT_BYTES;
       wd_words = ROLL_WD_WORDS;
       src_ptr = (ULONG *)ROLL_EQU_REFTBL;
       equ_exists = rollequ_exists;
       offset_level = roll_offset;
       }
   else
       {
       ht_bytes = PLAIN_HT_BYTES;
       wd_words = PLAIN_WD_WORDS;
       src_ptr = (ULONG *)PLAIN_EQU_REFTBL;
       equ_exists = plainequ_exists;
       offset_level = plain_offset;
       }

   if ( equ_exists )
   {
       /*
       ** if the equalization reference exists we get the value of the
       ** dark offset calculated from the peak level of
       ** the dark image histogram.
       */
       dark_offset = *(UBYTE *)src_ptr;

       /*
       ** The image at the EquDram should be exactly as MinDram. So use all
       ** MinDram defines.
```

```
        ** calculate the offset of the active image
        ** NOTE: image offset in copy_image_to_Dram(),
        **       load_equref_to_EquDram(), process_calibration() should be
        **       the same.
        */
        dst_ptr = (ULONG *)EQU_DRAM_BASE;

        if ( camera == ROLL )
            {
            dst_ptr += MINDRAM_WD_WORDS * (ht_bytes + MINDRAM_ROLL_ROW_B_OFFSET)
                + MINDRAM_ROLL_COL_W_OFFSET;     /* 512 * (960 + 38) + 24 */
            }
        else
            {
            dst_ptr += MINDRAM_WD_WORDS * (ht_bytes + MINDRAM_PLAIN_ROW_B_OFFSET)
                + MINDRAM_PLAIN_COL_W_OFFSET;    /* 512 * (976 + 31) + 39 */
            }

                /*
        ** transfer the bytes to the words
        */
                s = src_ptr;
                ii=ht_bytes;
            do
            {
                    d = dst_ptr;
                    jj=wd_words;
                    do
                    {
                            *d++ = *s++;
                    } while (--jj);
            dst_ptr += MINDRAM_WD_WORDS;

            } while (--ii);

        #if IMG_DEBUG
        load_straight_thruLUT_MEM_to_EquSram ();
        #else
        load_equLUT_to_EQUSram (offset_level,dark_offset,desired_equ_value);
        #endif
    }
    else
    {
        load_straight_thruLUT_to_EquSram ();
    }

    equref_camera = camera;
} /* load_equref_to_EquDram */

#else /* PALM_SCANNER */

/* ---------------------------------------------------------------------*/
/*
** Calculate the Straight Thru and store to the MinSram for PALM scanner
*/
VOID load_straight_thruLUT_to_MinSram (VOID)
{
    ULONG a_d, mem, *ptr;

    ptr = (ULONG *)MIN_SRAM_BASE;
```

```
    for ( mem = 0; mem < 256; mem++ )
         for ( a_d = 0; a_d < 256; a_d++ )
              {
              *ptr++ = a_d | 1;
              }
} /* load_straight_thruLUT_to_MinSram */

/* ------------------------------------------------------------------------*/
/*
** copy the equalize reference of PALM from the Dram to EquDram
*/
VOID load_palm_equref_to_EquDram (VOID)
{
   UBYTE *src_ptr;
   ULONG *dst_ptr;
   int ii;
   ULONG dark_offset;

   if ( palmequ_exists )
        {
        src_ptr = (UBYTE *)PALM_EQU_REFTBL;
        dst_ptr = (ULONG *)EQU_DRAM_BASE;

        /*
        ** we get the value of the dark offset calculated from the peak
        ** level of the dark image histogram.
        */
        dark_offset = *(ULONG *)src_ptr;

        src_ptr += 7;

        for ( ii = 0; ii < PALM_WD_PIXELS; ii++ )
            *dst_ptr++ = *src_ptr++;

 #if IMG_DEBUG
        load_straight_thruLUT_MEM_to_EquSram ();
 #else
        load_equLUT_to_EQUSram (palm_offset,dark_offset,desired_equ_value);
 #endif
        }
   else
        load_straight_thruLUT_to_EquSram ();

    equref_camera = PALM_SCAN;
} /* load_equref_to_EquDram */

#endif   /* PALM_SCANNER */

/*
** below functions are common for finger scanner and palm scanner
*/

/* ------------------------------------------------------------------------*/
/*
** Calculate the equalize LUT TABLE and store to the EQU_SRAM
*/
VOID load_equLUT_to_EQUSram (ULONG offset, ULONG dark_offset, ULONG desired)
{
    ULONG a_d, mem, *ptr, out, vv, ee;
```

```
    ptr = (ULONG *)EQU_SRAM_BASE;

    for ( a_d = 0; a_d < 256; a_d++ )
         for ( mem = 0; mem < 256; mem++ )
         {
              ee = floor0 (mem, dark_offset);

              if ( ee != 0 )
              {
                   out = (ULONG)(desired+offset) *
                       floor0(a_d,dark_offset) / ee;
                   if (out > offset)
                       out = out - offset;
                   else out = 0;
              }
                else
                   out = 254;

              if ( out < 0 )
                    out = 0;

              if ( out > 254 )
                   out = 254;

              *ptr++ = out | 1;
         }
} /* load_equLUT_to_EQUSram */

/* ------------------------------------------------------------------------*/
/*
** Calculate the Straight Thru of new image and store to the Equ Sram
*/
VOID load_straight_thruLUT_to_EquSram (VOID)
{
    ULONG a_d, mem, *ptr;

    ptr = (ULONG *)EQU_SRAM_BASE;

    for ( a_d = 0; a_d < 256; a_d++ )
         for ( mem = 0; mem < 256; mem++ )
              {
              *ptr++ = a_d | 1;
              }
} /* load_straight_thruLUT_to_EquSram */

#if IMG_DEBUG
/* ------------------------------------------------------------------------*/
/*
** Calculate the Straight Thru of memory's image and store to the Equ Sram
** NOTE: This routine is for debugging purpose
*/
VOID load_straight_thruLUT_MEM_to_EquSram (VOID)
{
    ULONG a_d, mem, *ptr;

    ptr = (ULONG *)EQU_SRAM_BASE;

    for ( a_d = 0; a_d < 256; a_d++ )
         for ( mem = 0; mem < 256; mem++ )
```

```
            {
            *ptr++ = mem | 1;
            }
} /* load_straight_thruLUT_MEM_to_EquSram */
#endif

/* ------------------------------------------------------------------------*/
/*
** Calculate the division to get rid of the overflow
*/
ULONG floor0 (ULONG a, ULONG b)
{
    if ( a >= b )
         return (a - b);
    else
         return (0);
} /* floor0 */

/* ------------------------------------------------------------------------*/
/*
** Setup the basic graphic function
*/
void setup_graphic (void)
{
    set_config (0, !0);
    clear_whole_screen();

    smlfont_id = install_font (&corpus29);
    bigfont_id = install_font (&corpus49);

    talon_font[CORPUS29] = install_font (&corpus29);
    talon_font[CORPUS49] = install_font (&corpus49);

    select_font (smlfont_id);
    get_fontinfo (smlfont_id, &fontinfo);
    smlfont_charhigh = fontinfo.charhigh;
    smlfont_charwide = fontinfo.charwide;

    select_font (bigfont_.d);
    get_fontinfo (bigfont_id, &fontinfo);
    bigfont_charhigh = fontinfo.charhigh;
    bigfont_charwide = fontinfo.charwide;
} /* setup_graphic */


/* end of gsp_init.c */
```

```
/* ---------------------------------------------------------------------
 *
 *  File name: userintf.c
 *
 *  Purpose:   User interface on scanner for talon 1000.
 *
 *  Date:      Jan-05-94
 *
 *  Author:    Ellen Yu
 *
 *  History:
 *   05-12-94 Joyce Young, fill MinDram with 0xFFFFFFFF is not needed.
 *   06-13-94 Joyce Young, add #ifndef NPA for NPA
 *   06-30-94 Ellen Yu,    turn grabber on 2 frame in PrevInitAct
 *   09-12-94 Joyce Young, add display_text() to display general message to
 *            call text_out() & kanji_out() separately
 *   09/22/94 JY, move setup of talon_font[CORPUS29] & talon_font[CORPUS49]
 *            from PrintTitle() to setup_graphic(), otherwise, if calling
 *            PrintTitle() several times, will mess up the talon_font table
 *   11-15-94 JY, delete #ifndef NPA, so keep displaying MINfunc image
 *   11-17-94 JY, add text_length() to get text length to call text_width()
 *            & kanji_length() separately
 *   11/18/94 JY, delete CAPT_ABORT case
 *   12-21-94 EY, fixed displaying MINfunc image
 *   01/21/95 JY, delete unnecessary include files
 *   02/17/95 TS, moved out button action routines to fngrcapt
 * --------------------------------------------------------------------- */

/* ------------------------------ Includes ------------------------------ */
#include "stdtypes.h"
#include "colordef.h"
#include "coorddef.h"
#include "gsp_defs.h"
#include "gsp_font.h"
#include "gsp_text.h"
#include "imgglobs.h"
#include "mem_allc.h"

/* ------------------------- External Functions  ------------------------- */

/* ------------------------------ Prototype ----------------------------- */
VOID PrintTitle (VOID);
VOID display_text (SHORT xx, SHORT yy, VOID *string, SHORT knj_font);
SHORT text_length (VOID *string, SHORT knj_font);

/* ---------------------------- Global Vars ---------------------------- */
SCRTEXT_T titletext[] =
{
   { CORPUS49, GREY6, WHITE, TITLE_TXT_X,      TITLE_TXT_Y,      TITLE_TXT_X,
     TITLE_FONT_H,      0, TITLE_TXT },
   { CORPUS49, GREY4, WHITE, COPYRIGHT_TXT_X, COPYRIGHT_TXT_Y, COPYRIGHT_TXT_X,
     COPYRIGHT_FONT_H,  0, COPYRIGHT_TXT },
#ifndef PALM_SCANNER
   { CORPUS29, GREY6, WHITE, VERSION_TXT_X,    VERSION_TXT_Y,    VERSION_TXT_X,
     VERSION_FONT_H,    0, FNGR_VER_TXT }
#else
   { CORPUS29, GREY6, WHITE, VERSION_TXT_X,    VERSION_TXT_Y,    VERSION_TXT_X,
```

```
    VERSION_FONT_H,    0, PALM_VER_TXT }
#endif
};

/* ------------------------------------------------------------------------ */

/* ------------------------------------------------------------------------ */
/*
** Display the given string on the (xx, yy) position for either English or
** Kanji
*/
VOID display_text (SHORT xx, SHORT yy, VOID *string, SHORT knj_font)
{
#ifndef KANJI
   text_out (xx, yy, (char *)string);
#else
   kanji_out (xx, yy, (SHORT *)string, knj_font);
#endif
} /* display_text */

/* ------------------------------------------------------------------------ */
/*
** Get the length of the given string either for English or for Kanji
*/
SHORT text_length (VOID *string, SHORT knj_font)
{
   short txt_w;

#ifndef KANJI
   txt_w = text_width ((char *)string);
#else
   txt_w = kanji_length ((SHORT *)string, knj_font);
#endif

   return (txt_w);
} /* text_length */

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *
 *  Func name: PrintTitle()
 *
 *  Purpose:   Display the TALON 1000 title and copyright
 *
 *  Date:      Jan-05-94
 *
 *  Author:    Ellen Yu
 *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
VOID PrintTitle (VOID)
{
    SCRTEXT_T *txtptr;
    short txt_w;

    /*
    **  Company title
    */
    txtptr = &titletext[0];
    set_colors( txtptr->fcolor, txtptr->bcolor );
```

```
    select_font( talon_font[txtptr->font] );

    txt_w = text_width ( txtptr->strptr );
    txtptr->xposn = (SCR_W - txt_w) / 2;
    text_out( txtptr->xposn, txtptr->yposn, txtptr->strptr );

    /*
    **  Copyright
    */
    txtptr = &titletext[1];
    set_colors( txtptr->fcolor, txtptr->bcolor );
    select_font( talon_font[txtptr->font] );

    txt_w = text_width ( txtptr->strptr );
    txtptr->xposn = (SCR_W - txt_w) / 2;
    text_out( txtptr->xposn, txtptr->yposn, txtptr->strptr );

    /*
    **  Version and type of scanner
    */
    txtptr = &titletext[2];
    set_colors( txtptr->fcolor, txtptr->bcolor );
    select_font( talon_font[txtptr->font] );

    txt_w = text_width ( txtptr->strptr );
    txtptr->xposn = (SCR_W - txt_w) / 2;
    text_out( txtptr->xposn, txtptr->yposn, txtptr->strptr );

}  /* PrintTitle */

/* end of userintf.c */
```

```
/* ---------------------------------------------------------------------
 *  File name: mimio.c
 *  Purpose:   Routines for handling memory images in the 34010.
 *  Date:      Oct-05-93
 *  Author:    Tom Sartor
 *  History:
 * --------------------------------------------------------------------- */

/* ------------------------------ Includes ------------------------------ */
#include "mimio.h"

void byteblt(),bytefil();

/* --------------------------------------------------------------------- */
void mim_new(d,a,i,w,h)
MIM *d;
unsigned char *a;
int i,w,h;
{
    d->a = a;
    d->i = i;
    d->w = w;
    d->h = h;
}

/* --------------------------------------------------------------------- */
void mim_subset(s,d,x,y,w,h)
MIM *s,*d;
int x,y,w,h;
{
    d->i = s->i;
    d->a = (s->a)+x+y*(d->i);
    d->w = w;
    d->h = h;
}

/* --------------------------------------------------------------------- */
void mim_move(s,d)
MIM *s,*d;
{
    byteblt(s->a,(s->i)<<3,d->a,(d->i)<<3,s->h,s->w,0xCCCC);
}

/* --------------------------------------------------------------------- */
void mim_move_c(s,d,c)
MIM *s,*d;
int c;
{
    byteblt(s->a,(s->i)<<3,d->a,(d->i)<<3,s->h,s->w,c);
}

/* --------------------------------------------------------------------- */
unsigned char * mim_adr(s,x,y)
MIM *s;
int x,y;
{
    return((s->a)+x+y*(s->i));
}

/* --------------------------------------------------------------------- */
```

```
void mim_size(s, w, h)
MIM *s;
int *w,*h;
{
    *w = s->w;
    *h = s->h;
}

/* ---------------------------------------------------------------------- */
int mim_inc(s)
MIM *s;
{
        return(s->i);
}

/* ---------------------------------------------------------------------- */
void mim_fill(d, v)
MIM *d;
int v;
{
    bytefil(v&0xff,d->a,(d->i)<<3,d->h,d->w,0x000c);
}

/* ---------------------------------------------------------------------- */
void mim_fill_c(d, v, c)
MIM *d;
int v,c;
{
    bytefil(v&0xff,d->a,(d->i)<<3,d->h,d->w,c);
}

/* ---------------------------------------------------------------------- */
/* end of mimio.c */
```

```
/*
** Filename: grabber.c
** Purpose:  Routines to handle the Grabber.
** Date:     11/16/93
** Author:   Joyce Young
** History:
**  05/05/94 Joyce Young, Add mask register for display (GRABBER_CTRL1_BASE)
**  01/04/95 Ellen Yu, Change identatin style.
**  04/13/95 ts - added different turn_grab_on_normal, modified turn_off,
**               and moved prototypes to gsp_func.h
**  04/19/95 EY - Added comments.
*/

#include "stdtypes.h"
#include "img_defs.h"
#include "mem_addr.h"
#include "gsp_func.h"

/* ------------------------------------------------------------------------*/
/*
** Initialize the GSP Grabber: setup the Video Control and Grabber Control.
*/
VOID init_gsp_grabber (VOID)
{
/*
** Setup the Video Control:
**  read the System Resources (address is the same as Video Control),
**  if the value of the SVGA Sync input detected (bit 6) is 0 (normal is 1),
**  then set Video Source as External Sync Source (bit1-0 = 01), otherwise,
**  set as Internal Sync Source (bit1-0 = 00).
**
*/
   *(ULONG *)VIDEO_CONTROL_BASE = 0x00000000;
   *(ULONG *)GRABBER_CTRL1_BASE = 0xFFFFFFFF;

   /*
   ** Turn the Grabber OFF
   */
   turn_grab_off (ROLL);
} /* init_gsp_grabber */

/* ------------------------------------------------------------------------*/
/*
** Turn Grab off, polling Grabber status till off (bit0 = 0).
*/
VOID turn_grab_off (image_type_t camera)
{
#ifndef DISABLE_GRAB
   /*--- Be sure bit 7 be cleared, no image reduction ---*/
   if ( camera == ROLL )
       *(UBYTE *)GRABBER_CTRL0_BASE = 0x70;
   else
       *(UBYTE *)GRABBER_CTRL0_BASE = 0xE0;

   /*
   ** wait for Grab is disabled (BIT0 becomes 1)
   */
   while ( !((*(UBYTE *)GRABBER_CTRL0_BASE) & 1) )
```

```
    /*
    ** wait for the VBLANK start
    ** NOTE: need to wait VBLANK off/on, otherwise, Grabber sometimes does not
    **  really turn off
    */
    while ( !((*(UBYTE *)GRABBER_CTRL0_BASE) & 2) )
        ;

    while ( ((*(UBYTE *)GRABBER_CTRL0_BASE) & 2) )
        ;
#endif
} /* turn_grab_off */

/* ------------------------------------------------------------------------*/
/*
** Turn Grabber on with reduced display
** polling Grabber status till on (bit0 = 1).
*/
VOID turn_grab_on (image_type_t camera, int frame_num)
{
#ifndef DISABLE_GRAB
    int ii;

    if ( camera == ROLL )
        *(UBYTE *)GRABBER_CTRL0_BASE = 0xFF;
    else
        *(UBYTE *)GRABBER_CTRL0_BASE = 0xEF;

    /*
    ** wait for Grab is enabled (BIT0 becomes 0)
    */
    while ( ((*(UBYTE *)GRABBER_CTRL0_BASE) & 1) )
        ;

    for ( ii = 0; ii < frame_num; ii++ )
        {
        while ( !((*(UBYTE *)GRABBER_CTRL0_BASE) & 2) )
            ;

        while ( ((*(UBYTE *)GRABBER_CTRL0_BASE) & 2) )
            ;
        }
#endif
} /* turn_grab_on */

/* ------------------------------------------------------------------------*/
/*
** Turn Grabber on with normal display
** polling Grabber status till on (bit0 = 1).
*/
VOID turn_grab_on_normal (image_type_t camera, int frame_num)
{
#ifndef DISABLE_GRAB
    int ii;

    if ( camera == ROLL )
        *(UBYTE *)GRABBER_CTRL0_BASE = 0x7F;
    else
        *(UBYTE *)GRABBER_CTRL0_BASE = 0xEF;
```

```
    /*
    ** wait for Grab is enabled (BIT0 becomes 0)
    */
    while ( ((*(UBYTE *)GRABBER_CTRL0_BASE) & 1) )
        ;

    for ( ii = 0; ii < frame_num; ii++ )
        {
        while ( !((*(UBYTE *)GRABBER_CTRL0_BASE) & 2) )
            ;

        while ( ((*(UBYTE *)GRABBER_CTRL0_BASE) & 2) )
            ;
        }
#endif
} /* turn_grab_on_normal */


/* ------------------------------------------------------------------------*/
/*
** wait for the VBLANK start on and off
*/
VOID wait_for_grab_VBLANK_on_off (VOID)
{
#ifndef DISABLE_GRAB
    while ( !((*(UBYTE *)GRABBER_CTRL0_BASE) & 2) )
        ;

    while ( ((*(UBYTE *)GRABBER_CTRL0_BASE) & 2) )
        ;
#endif
} /* wait_for_grab_VBLANK_on_off */


/* end of grabber.c */
```

```
/*
** Filename: img_defs.h
** Purpose:  Definitions to handle the images display or capture for Host,
**           DSP and GSP.
** Date:     10/11/93
** Author:   Joyce Young
** Revised:
**  11/02/93 - Ellen Yu
**  01/28/94 - Joyce Young, Add ROLL_RESOLUTION & PLAIN_RESOLUTION
**  02/02/94 - Ellen Yu, Seperate into img_defs.h and imgglobs.h.
**  02/02/94 - Joyce Young, Add the definitions for the PALM camera
**  11/21/94 - JY, Add #ifdef NEW_PALM to support new PALM
**  01/13/95 - JY, change palm pixels for new NPA palm
*/

#ifndef IMG_DEFS_H
#define IMG_DEFS_H

#define  ROLL_WD_BYTES          960
#define  ROLL_WD_WORDS          (ROLL_WD_BYTES / 4)
#define  ROLL_HT_BYTES          960
#define  ROLL_WD_PIXELS         960
#define  ROLL_HT_PIXELS         960
#define  ROLL_HT_RESOLUTION     600
#define  ROLL_VT_RESOLUTION     600
#define  ROLL_BIT_PERPIXEL      8

#define  PLAIN_WD_BYTES         1600
#define  PLAIN_WD_WORDS         (PLAIN_WD_BYTES / 4)
#define  PLAIN_HT_BYTES         976
#define  PLAIN_WD_PIXELS        1600
#define  PLAIN_HT_PIXELS        976
#define  PLAIN_HT_RESOLUTION    500
#define  PLAIN_VT_RESOLUTION    500
#define  PLAIN_BIT_PERPIXEL     8

#ifdef NEW_PALM
#define  PALM_WD_PIXELS         2400                /* pixels per row */
#define  PALM_HT_PIXELS         2040                /* rows */
#define  PALM_HT_RESOLUTION     508
#define  PALM_VT_RESOLUTION     508
#define  PALM_BIT_PERPIXEL      8
#define  HEEL_WD_PIXELS         2400                /* pixels per row */
#define  HEEL_HT_PIXELS         1040                /* rows */
#define  HEEL_HT_RESOLUTION     PALM_HT_RESOLUTION
#define  HEEL_VT_RESOLUTION     PALM_VT_RESOLUTION
#define  HEEL_BIT_PERPIXEL      8
#else
#define  PALM_WD_BYTES          2748
#define  PALM_WD_WORDS          (PALM_WD_BYTES / 4)
#define  PALM_HT_BYTES          2748
#define  PALM_WD_PIXELS         PALM_WD_BYTES
#define  PALM_HT_PIXELS         PALM_HT_BYTES
#define  PALM_HT_RESOLUTION     500
#define  PALM_VT_RESOLUTION     500
#define  PALM_BIT_PERPIXEL      8
#endif

 /*
 ** ROLL: must be 0
*/
typedef enum image_type_t
{
   ROLL,
   PLAIN,
   PALM_SCAN,
   NON_CAMERA
} image_type_t;


#endif   /* IMG_DEFS_H */
```

```
/*
** Filename: mem_allc.h
** Purpose:  Definition of memory allocation of software.
**           If the program wants to run GSP/RAMDAC/8051 functions under
**           the DSP map, add the compiler option -dDSP_MAP; otherwise,
**           those functions will run under GSP map.
** Date:     05/27/94
** Author:   Joyce Young
** History:
**   02/04/94 Joyce Young, Add the definitions of the GSP DRAM BANKs
**   03/15/94 Joyce Young, Rearrange the DramBanks
**   04/07/94 Joyce Young, Rearrange the DramBanks to support Kanji
**   05/20/94 Ellen Yu,    Removed unnecessary define's. Changed the
**                  FNGR_FINAL_IMAGE_ADDR from 0xE00000 to 0xF00000
**   06/07/94 Joyce Young, add COMMAND_RETURN_RESULTS to support platen check.
**   06/13/94 Joyce Young, modify the comments
**   11/22/94 JY, add #ifndef NEW_PALM to support new PALM
**   01/18/95 JY, modify for palm tables
*/

#ifndef MEM_ALLC_H
#define MEM_ALLC_H

/*
** Below memory maps are different from DSP and GSP.
** We use the same name on the both GSP and DSP codes, but
** compile with -dDSP_MAP option if the program is emulated on DSP or
** compile without -dDSP_MAP option if the program is emulated on GSP.
*/

#ifdef DSP_MAP

/*
**  NOTE: finger scanner will use Bank0 & Bank3,
**        palm scanner will use all banks.
**        If any address is changed in mem_addr.h, make sure the address here
**        will be matched.
**                              +------------------+
**  DramBank0:    0xC00100      | 1M words valid   |
**                              +------------------+
**  DramBank1:    0xD00000      | 1M words valid   |
**                              +------------------+
**  DramBank2:    0xE00000      | 1M words valid   |
**                              +------------------+
**  DramBank3:    0xF00000      | 1M words valid   |
**                              +------------------+
*/

/*
** Dram area to store the image process by WORD length:
** The first 2 items, DSPGSP_CMMD_BUF, PC_PARAMETERS will be the same area
** to the finger scanner and the palm scanner.
** GSP_COFF_PROGRAM will be stored after all tables, then followed by
** FINAL_IMAGE_ADDR.
*/
#define  DSPGSP_CMMD_BUF          0xC00100       /* 0x100W  */

/*
** The order of the PC_PARAMETERS (ULONG):
**  For Finger scanner
```

```
**        1. flag of the ROLL  equalization file exists (1=exists)
**        2. flag of the PLAIN equalization file exists (1=exists)
**        3. flag of the parameter file exists (1=exists)
**        4.... the numbers of the values in the para.dat file if exists
**
**  For Palm scanner
**        1. flag of the PALM equalization file (palm.equ) exists (1=exists)
**        2. flag of the PALM dewarping file    (palm.wrp) exists (1=exists)
*/
#define   PC_PARAMETERS             0xC00200        /* 0x40    */

/*
** The results of the platen check for the finger scanner, defined in the
** structure of platen_check_result_t in scancmmd.h, will be stored after
** all image parameters in PC_PARAMETERS.
** If total numbers of the image parameters exceed to 0x40 in the future,
** this address needs to be changed.
*/
#define   COMMAND_RETURN_RESULTS    0xC00230        /* reserved 10W spaces */

/*
** below for Finger scanner
*/
#define   STRAIGHT_THRU_LUTTBL      0xC00240        /* 0x4000   256*256/4 */
#define   MINFUNC_LUTTBL            0xC04240        /* 0x4000   256*256/4 */
#define   FNGR_TAGGING_LUTTBL       0xC08240        /* 0x4000   256*256/4 */
#define   ROLL_EQU_REFTBL           0xC0C240        /* 0x38400  960*960/4 */
#define   PLAIN_EQU_REFTBL          0xC44640        /* 0x5F500 1600*976/4 */

/*
** GDRAM address in fngrgspe.cmd & fngrgspj.cmd need to be matched with
** FNGR_GSP_COFF_PROGRAM. If any one been changed, change the other, too
*/
#define   FNGR_GSP_COFF_PROGRAM     0xCA3B40        /* 0x7000  */
#define   FNGR_FINAL_IMAGE_ADDR     0xF00000

/*
** below for PALM scanner
*/
/*
** GDRAM address in palmgspe.cmd & palmgspj.cmd need to be matched with
** PALM_GSP_COFF_PROGRAM. If any one been changed, change the other, too
*/
#define   PALM_EQU_REFTBL           0xC00240        /* 2400/4 */
#define   PALM_GSP_COFF_PROGRAM     0xC00640        /* 0x7000 */

#ifndef NEW_PALM
#define   PALM_FINAL_IMAGE_ADDR     0xC07640        /* 2748*2748/4=0x1CCE84 */
#else
#define   PALM_DEWARP_TBL           0xC07640        /* 0x300 (3072/4) */
#define   HEEL_FINAL_IMAGE_ADDR     0xC07940        /* 2400*1040/4=0x98580 */
#define   PALM_FINAL_IMAGE_ADDR     0xC9FEC0        /* 2400*2040/4=0x12ad40 */
#endif

#else    /* GSP_MAP */

/*
** Dram area to store the image process by WORD length:
** The first 2 items, DSPGSP_CMMD_BUF, PC_PARAMETERS will be the same area
** to the finger scanner and the palm scanner.
```

```
** GSP_COFF_PROGRAM will be stored after all tables, then followed by
** FINAL_IMAGE_ADDR.
*/
#define  DSPGSP_CMMD_BUF          0xC0002000       /* 0x100W  */

/*
** The order of the PC_PARAMETERS (ULONG):
**  For Finger scanner
**       1. flag of the ROLL  equalization file exists (1=exists)
**       2. flag of the PLAIN equalization file exists (1=exists)
**       3. flag of the parameter file exists (1=exists)
**       4.... the numbers of the values in the para.dat file if exists
**
**  For Palm scanner
**       1. flag of the PALM equalization file (palm.equ) exists (1=exists)
**       2. flag of the PALM dewarping file    (palm.wrp) exists (1=exists)
*/
#define  PC_PARAMETERS            0xC0004000       /* 0x40    */

/*
** The general return results of GSP command processing, defined in the
** scancmmd.h, will be stored after all image parameters in PC_PARAMETERS.
** If total numbers of the image parameters exceed to 0x40 in the future,
** this address needs to be changed.
*/
#define  COMMAND_RETURN_RESULTS   0xC0004600       /* reserved 10W spaces */

/*
** below for Finger scanner
*/
#define  STRAIGHT_THRU_LUTTBL     0xC0004800       /* 0x400C   256*256/4 */
#define  MINFUNC_LUTTBL           0xC0084800       /* 0x4000   256*256/4 */
#define  FNGR_TAGGING_LUTTBL      0xC0104800       /* 0x4000   256*256/4 */
#define  ROLL_EQU_REFTBL          0xC0184800       /* 0x38400  960*960/4 */
#define  PLAIN_EQU_REFTBL         0xC088C800       /* 0x5F500 1600*976/4 */

/*
** GDRAM address in fngrgspe.cmd & fngrgspj.cmd need to be matched with
** FNGR_GSP_COFF_PROGRAM. If any one been changed, change the other, too
*/
#define  FNGR_GSP_COFF_PROGRAM    0xC1476800       /* 0x7000  */
#define  FNGR_FINAL_IMAGE_ADDR    0xC6000000

/*
** below for PALM scanner
*/
/*
** GDRAM address in palmgspe.cmd & palmgspj.cmd need to be matched with
** PALM_GSP_COFF_PROGRAM. If any one been changed, change the other, too
*/
#define  PALM_EQU_REFTBL          0xC0004800       /* 2400/4 */
#define  PALM_GSP_COFF_PROGRAM    0xC000C800       /* 0x7000  */

#ifndef NEW_PALM
#define  PALM_FINAL_IMAGE_ADDR    0xC00EC800       /* 2748*2748/4=0x1CCE84 */
#else
#define  PALM_DEWARP_TBL          0xC00EC800       /* 0x300 (3072/4) */
#define  HEEL_FINAL_IMAGE_ADDR    0xC00F2800       /* 2400*1040/4=0x99580 */
#define  PALM_FINAL_IMAGE_ADDR    0xC13FD800       /* 2400*2040/4=0x12ad40 */
#endif
#endif   /* DSP_MAP */

#endif   /* MEM_ALLC_H */
```

```
/*
** Filename: imgglobs.h
** Purpose:  Definition & variables to handle the images display or capture
**           on DSP/GSP.
** Date:     10/11/93
** Author:   Joyce Young
** Revised:
**  11/02/93 - Ellen Yu
**  06/07/94 - Ellen Yu, Support platen check function, add dirty_threshold
**             and damage_threshold to the image_object_t structure.
**  11/18/94 - JY, delete CAPT_ABORT in capture_t;
**             expand length of ic_name in image_object_t
**  12/21/94 - JY, add CAPT_NULL & CAPT_SAVE_PALM in capture_t
*/

#include "stdtypes.h"

#ifndef IMGGLOBS_H
#define IMGGLOBS_H

typedef enum capture_t
{
    CAPT_NO_RESPONSE,
    CAPT_SAVEIMG,       /* ROLL or PLAIN for finger, PALM & HEEL for palm */
    CAPT_MISSFNG,       /* missing finger/hand */
    CAPT_WRONG_TYPE,    /* imgage type and finger mismatched */
    CAPT_NULL,          /* 4 reserved for ADMIT_ABORT */
    CAPT_SAVE_PALM      /* save PALM only; only used by palm scanner */
} capture_t;

/*
** if MAX_NAME_LEN in scancmmd.h expands, ic_name needs to expand too.
*/
typedef struct image_object_t
{
   ULONG ic_cmd;            /* ROLL, PLAIN */
   ULONG ic_size;              /* sizeof (image_object_t)/sizeof (ULONG) */
   ULONG ic_scantype;       /* STYPE_... */
   ULONG ic_imgtype;        /* ROLL, PLAIN or PALM */
   ULONG ic_set;            /* OR of shifted image IDs */
   ULONG ic_addr;           /* the address of image */
   ULONG ic_name[11];/* convict name (4 chars/ULONG) */
   ULONG ic_reserve1;
         ULONG ic_dirty_threshold;
         ULONG ic_damage_threshold;
   ULONG ic_reserve2;
   ULONG ic_flag;           /* the flag indicate all words written */
} image_object_t;


#endif   /* IMGGLOBS_H */

/*
** Filename: gsp_func.h
** Purpose:  function prototypes
** Date:     04-19-95
** Author:   Ellen Yu
** History:
*/

/* ------------------------------ grabber.c ------------------------------*/
VOID init_gsp_grabber (VOID);
VOID turn_grab_off (image_type_t camera);
VOID turn_grab_on (image_type_t camera, int frame_num);
VOID turn_grab_on_normal (image_type_t camera, int frame_num);
VOID wait_for_grab_VBLANK_on_off (VOID);
```

```
/*
** Filename: gsp_imgs.h
** Purpose:  Definition & variables to handle the images display or capture.
** Date:     18/01/94
** Author:   Ellen Yu
** History:
**  02/20/94 Joyce Young, modify the MINDRAM offset
**  06/03/94 Joyce Young, adjust the offset of ROLL/PLAIN active image in
**              MINDRAM for new board
**  06/09/94 Ellen Yu, add #define PROTOTYPE difference the hardware
**              prototype board and other new revisions.
*/

#ifndef GSP_IMGS_H
#define GSP_IMGS_H

#define  MINDRAM_WD_BYTES       2048
#define  MINDRAM_WD_WORDS       (MINDRAM_WD_BYTES / 4)
#define  MINDRAM_HT_BYTES       1024
#define  MINDRAM_PITCH_BYTES    2048
#define  MINDRAM_PITCH_BITS     (MINDRAM_PITCH_BYTES * 8)      /* 0x4000 */
#ifndef PROTOTYPE
#define  MINDRAM_ROLL_ROW_B_OFFSET   38
#define  MINDRAM_ROLL_COL_W_OFFSET   24
#define  MINDRAM_ROLL_COL_B_OFFSET   (MINDRAM_ROLL_COL_W_OFFSET * 4)
#define  MINDRAM_PLAIN_ROW_B_OFFSET  31
#define  MINDRAM_PLAIN_COL_W_OFFSET  38
#define  MINDRAM_PLAIN_COL_B_OFFSET  (MINDRAM_PLAIN_COL_W_OFFSET * 4)
#else
#define  MINDRAM_ROW_B_OFFSET  32
#define  MINDRAM_COL_W_OFFSET  24
#define  MINDRAM_COL_B_OFFSET  (MINDRAM_COL_W_OFFSET * 4)
#define  MINDRAM_PLAIN_COL_W_OFFSET  13
#define  MINDRAM_PLAIN_COL_B_OFFSET  (MINDRAM_PLAIN_COL_W_OFFSET * 4)
#endif

#define  GRBVRAM_WD_BYTES       1024
#define  GRBVRAM_HT_BYTES       1024
#define  GRBVRAM_PITCH_BYTES    1024
#define  GRBVRAM_PITCH_BITS     (GRBVRAM_PITCH_BYTES * 8)      /* 0x2000 */

#define  DPYVRAM_WD_BYTES       1024
#define  DPYVRAM_HT_BYTES       1024
#define  DPYVRAM_PITCH_BYTES    1024
#define  DPYVRAM_PITCH_BITS     (1024 * 8)      /* 0x2000 */

#define  ROLL_PITCH_BYTES       1024
#define  ROLL_PITCH_BITS        (ROLL_PITCH_BYTES * 8)
#define  ROLL_GRBVRAM_PITCH     (-(GRBVRAM_PITCH_BITS * 4))
#define  ROLL_DPYVRAM_PITCH     (DPYVRAM_PITCH_BITS * 3)
#define  ROLL_DYDX_VAL      ((((IMG_ROLL_H - 12) / 3) << 16) | (IMG_ROLL_W - 16))

#define  PLAIN_PITCH_BYTES      2048
#define  PLAIN_PITCH_BITS       (PLAIN_PITCH_BYTES * 8)
#define  PLAIN_GRBVRAM_PITCH    (-(GRBVRAM_PITCH_BITS * 2))
#define  PLAIN_DPYVRAM_PITCH    (DPYVRAM_PITCH_BITS)
#define  PLAIN_DYDX_VAL         ((IMG_PLAIN_H - 12) << 16) | (IMG_PLAIN_W - 12)


#endif   /* GSP_IMGS_H */
```

```
/* $Header:   G:/t1000/gsp/mimio.h_v   1.0   08 Apr 1995 11:06:02   TOM  $ */
/*
 * $Log:   G:/t1000/gsp/mimio.h_v  $
 *
 *    Rev 1.0   08 Apr 1995 11:06:02   TOM
 * Initial revision.
 *
 *    Rev 1.2   26 Oct 1992 17:44:50   TOM
 * added mim_fill declarations
 *
 *    Rev 1.1   16 Oct 1992 01:52:10   TOM
 *
 *    Rev 1.0   16 Sep 1992 18:41:04   TOM
 * Initial revision.
 */
/*****************************************
file: mimio.h

definitions for memory image segments
only 9-bit pixels are allowed

*/

typedef struct mimage {
    unsigned char *a;       /* address of first element of first row */
    int   i;                /* increment in bytes  from row to row  */
    int   w;                /* image width */
    int   h;                /* image height */
} MIM;

void mim_new(MIM *d,unsigned char *a,int i,int w,int h );
void mim_subset( MIM *s,MIM *d,int x,int y,int w,int h );
void mim_move( MIM *s,MIM *d );
void mim_move_c( MIM *s,MIM *d,int c);
void mim_op_move( MIM *s,MIM *d );
unsigned char *mim_adr( MIM *s,int x,int y );
void mim_size( MIM *s, int *w, int *h );
int mim_inc( MIM *s);
void mim_fill( MIM *d, int v );
void mim_fill_c( MIM *d, int v, int c );
```

```
/*
** Filename: gsp_defs.h
** Purpose:  #define in GSP application.
** Date:     10-05-93
** Author:   Ellen Yu
** History:
**  01-31-94 Joyce Young, Changed the src_addr & dst_addr in IMAGE_ATTR_T as
**                        pointer
**  02-25-94 Joyce Young, Added the fng_num in IMAGE_ATTR_T
**  04-01-94 Joyce Young, support KANJI in MODE_TYPE
**  12-19-94 Ellen Yu, remove some unused define
*/

#ifndef GSP_DEFS_H
#define GSP_DEFS_H

#include "img_defs.h"
#include "scancmmd.h"

#define NULL   0

#define FALSE  0
#define TRUE   1

/*
**
*/
#define X1E    0x0002
#define HIE    0x0200
#define INTIN  0x0008
#define INTOUT 0x0080

#define BIT0   0x01
#define BIT1   0x02
#define BIT2   0x04
#define BIT3   0x08

/*
**
*/
typedef enum
{
    MODE_ROLL,
    MODE_PLAIN,
    MODE_PREV,
    MODE_CENTR,
    MODE_RARROW,
    MODE_LARROW,
    MODE_PLACEHOLD,
    MODE_PLACEHOLD1,
    MODE_ROLLHOLD,
    MODE_ROLLHOLD1,
    MODE_LIFTHAND
} MODE_TYPE;

typedef enum
{
   FINGER_LEGEND,
   MODE_LEGEND
} MGS_LEGEND_TYPE;
```

```
typedef enum
{
    PREV_ROLL,
    PREV_PLAIN,
    PREV_MISSING,
    PREV_YES,
    PREV_NO,
    PREV_MISFNG,
    CAPT_REPEAT,
    CAPT_NEXT,
    CAPT_FINGER,
    PLACE_START,
    PLACE_ABORT,
    ROLL_CONTINUE,
    ROLL_YES
} KEYACT_TYPE;

typedef enum
{
          RIGHT,
          LEFT
} HAND_TYPE;

typedef enum
          {
          ROLL_RIGHT_THUMB,              /* 0  */
          ROLL_RIGHT_INDEX,              /* 1  */
          ROLL_RIGHT_MIDDLE,             /* 2  */
          ROLL_RIGHT_RING,               /* 3  */
          ROLL_RIGHT_LITTLE,             /* 4  */
          ROLL_LEFT_THUMB,               /* 5  */
          ROLL_LEFT_INDEX,               /* 6  */
          ROLL_LEFT_MIDDLE,              /* 7  */
          ROLL_LEFT_RING,                /* 8  */
          ROLL_LEFT_LITTLE,              /* 9  */
          PLAIN_TWO_THUMBS,              /* 10 (unsupported) */
          PLAIN_RIGHT4,                  /* 11 */
          PLAIN_LEFT4,                   /* 12 */
          RIGHT_PALM,                    /* 13 */
          LEFT_PALM,                     /* 14 */
          PLAIN_RIGHT_THUMB,             /* 15  */
          PLAIN_LEFT_THUMB               /* 16  */
} FINGER_TYPE;

/*
** user-selected type of the key pressed or command for image caputre
*/
typedef enum keytype_t
{
    TYPE_NO_RESPONSE,
    TYPE_SCAN_KEY,
    RELEASE_SCAN_KEY,
    TYPE_SAVE_KEY,
    RELEASE_SAVE_KEY,
    TYPE_ABORT
} keytype_t;

typedef enum
{
```

```
    ERASE,
    DRAW
} TEXT_OPER_T;

typedef enum
{
    SCAN_KEY,
    SAVE_KEY
} KEY_TYPE;

typedef enum
{
    RET_ERR_ADDR,
    READ_FOREVER,
    WRITE_FOREVER
} MEM_TEST_RESULT_TYPE;

typedef enum
{
    UNPRESSED_BUTTON,
    PRESSED_BUTTON
} BUTTON_OPER_TYPE;

/*
** Attributes for text displayed on screen
*/
typedef struct
{
   short   font;        /* font index of the talon_font table */
   short   fcolor;      /* foreground color */
   short   bcolor;      /* background color */
   short   xposn;       /* x position */
   short   yposn;       /* y position */
   short   align;       /* Alignment (relative to baseline or topleft)  */
   short   font_h;      /* height of font */
   short   text_w;      /* width of text string */
   char    *strptr;     /* text string ptr */
} SCRTEXT_T;

typedef struct
{
   image_type_t  camera;
   HAND_TYPE     hand;
   scan_type_t   fng_num;
   ULONG         store_addr;
   UBYTE         *src_addr;
   UBYTE         *dst_addr;
} IMAGE_ATTR_T;

#endif   /* GSP_DEFS_H */
```

【図１】

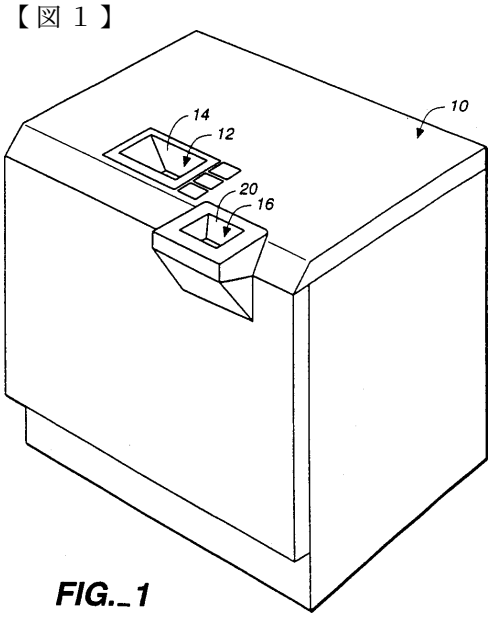

FIG._1

【図２】

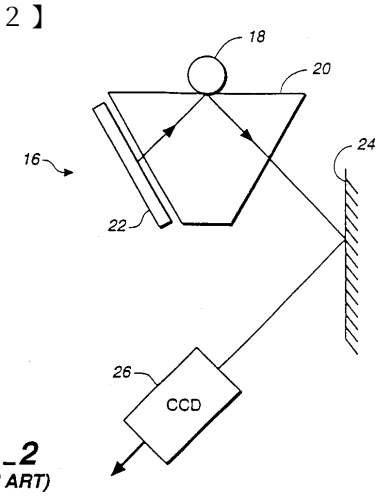

FIG._2
(PRIOR ART)

【図３】

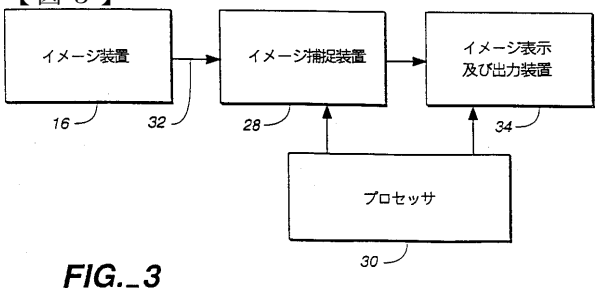

FIG._3

【図４】

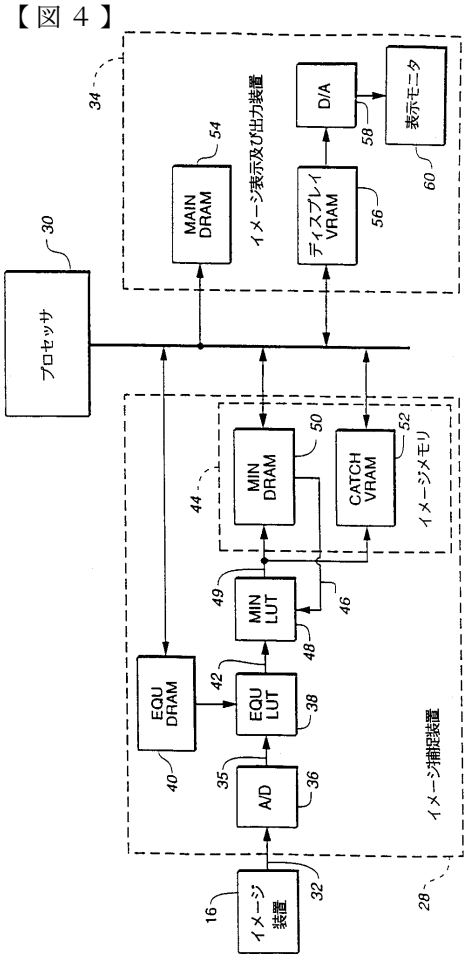

FIG._4

【図５Ａ】

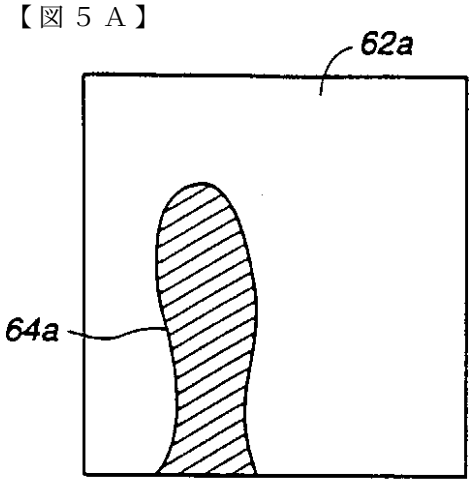

FIG._5A

【図５Ｂ】

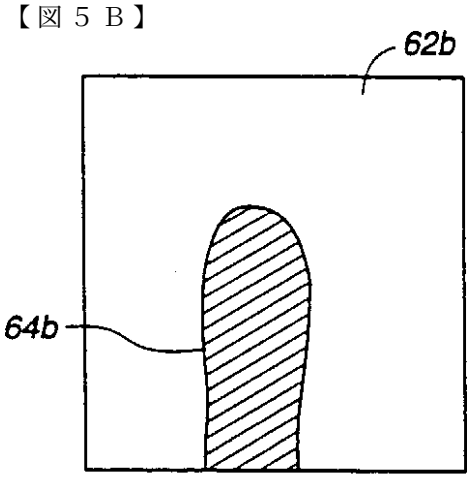

FIG._5B

【図５Ｃ】

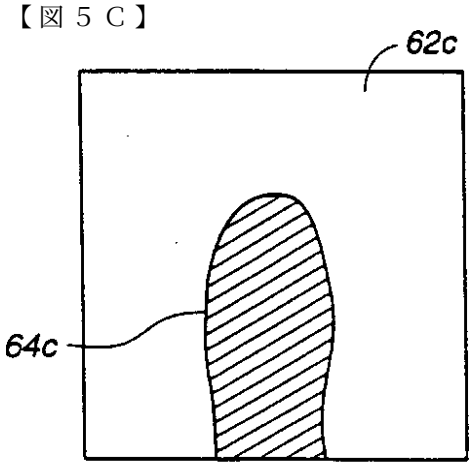

FIG._5C

【図５Ｄ】

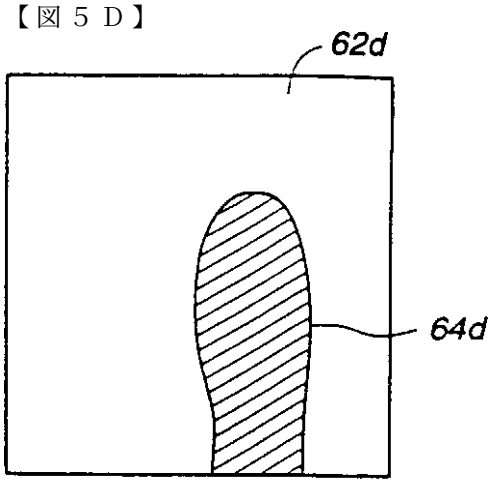

FIG._5D

【図５Ｅ】

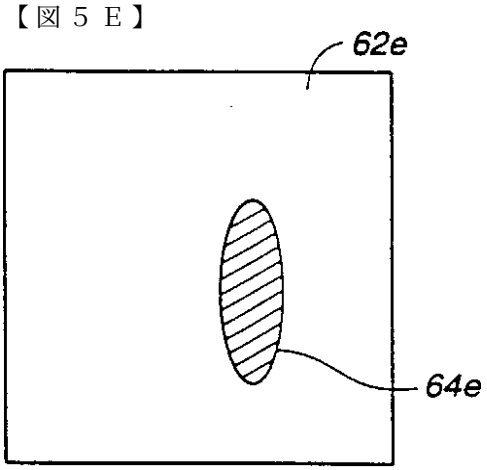

FIG._5E

【図６Ａ】

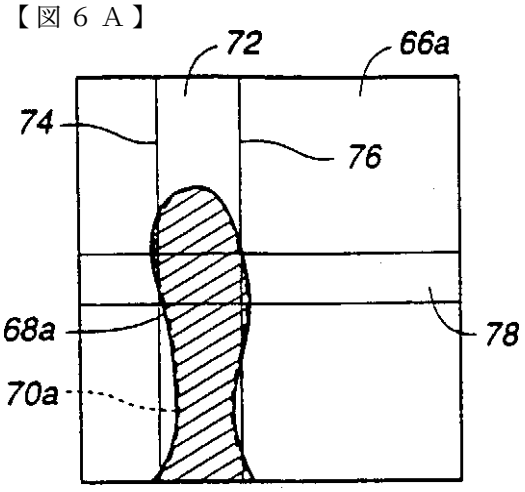

FIG._6A

【図６Ｂ】

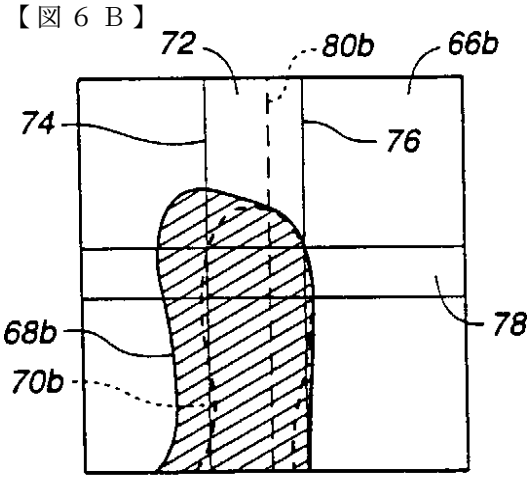

FIG._6B

【図６Ｃ】

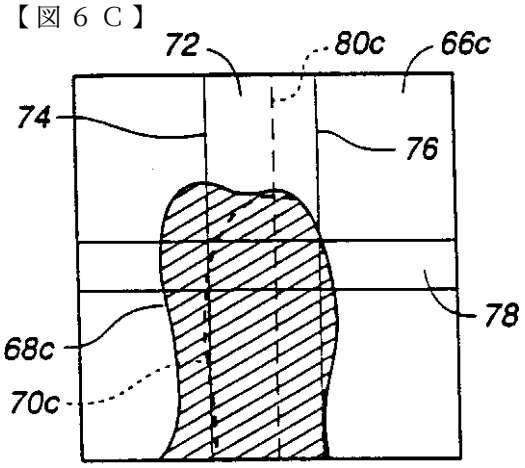

FIG._6C

【図６Ｄ】

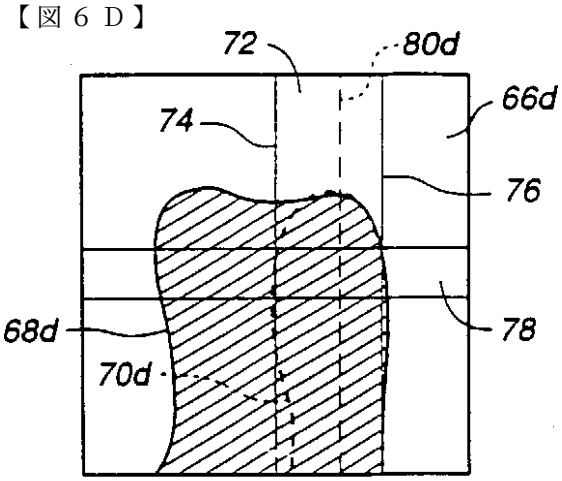

FIG._6D

【図６Ｅ】

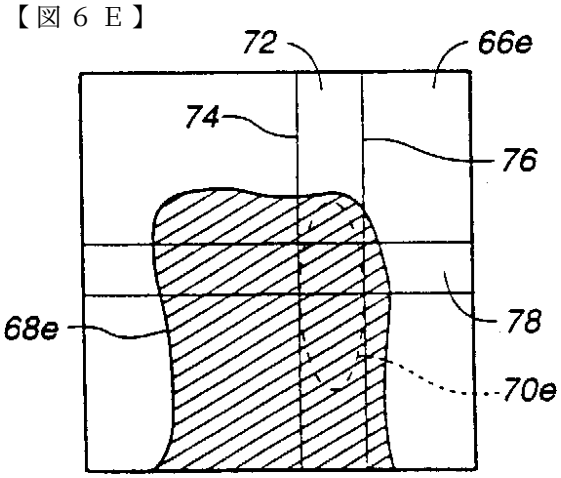

FIG._6E

【図７Ａ】

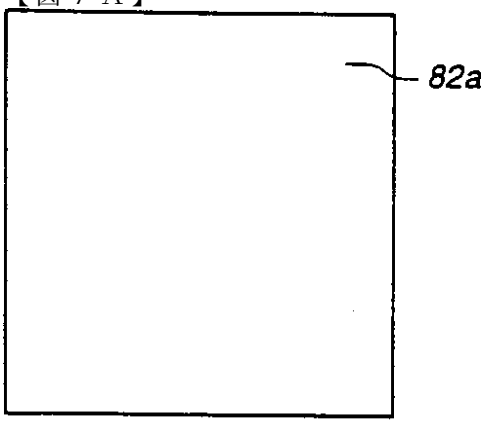

FIG._7A

【図７Ｂ】

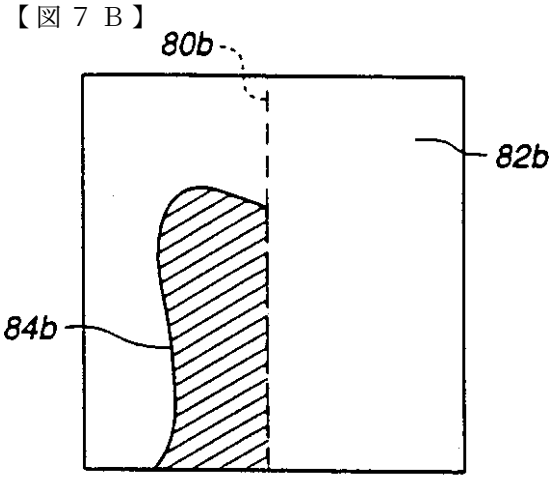

FIG._7B

【図７Ｃ】

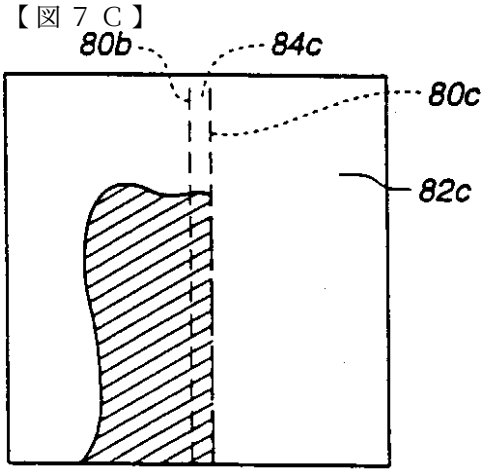

FIG._7C

【図７Ｄ】

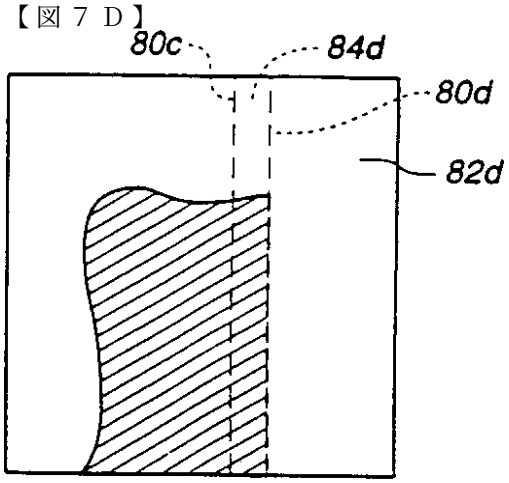

FIG._7D

【図７Ｅ】

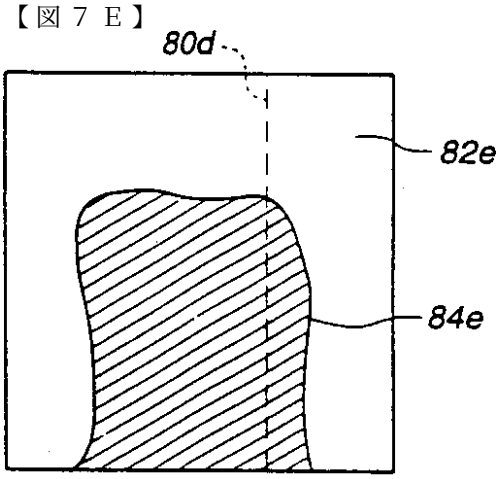

FIG._7E

【図８Ａ】

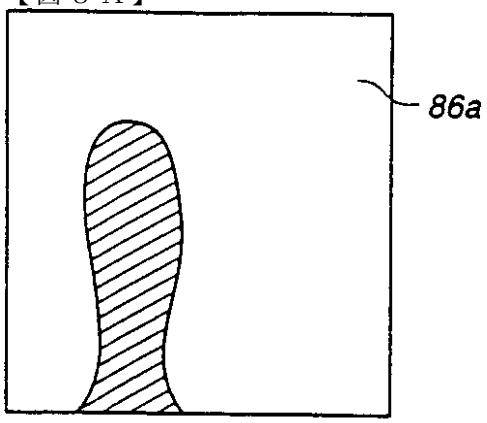

FIG._8A

【図８Ｂ】

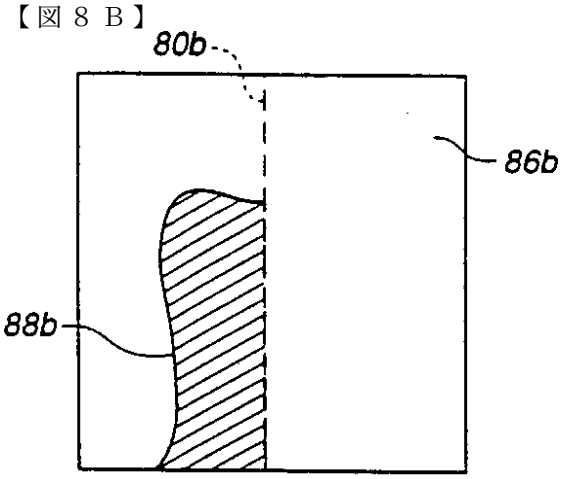

FIG._8B

【図８Ｃ】

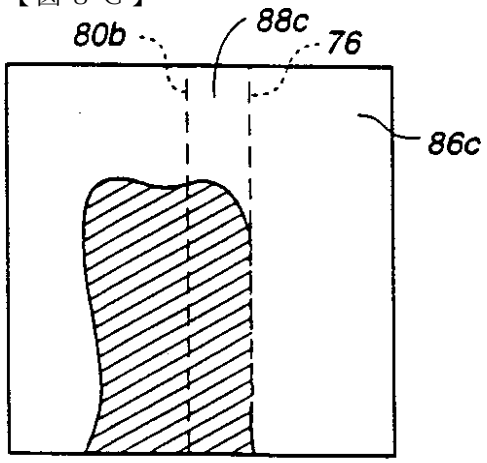

FIG._8C

【図８Ｄ】

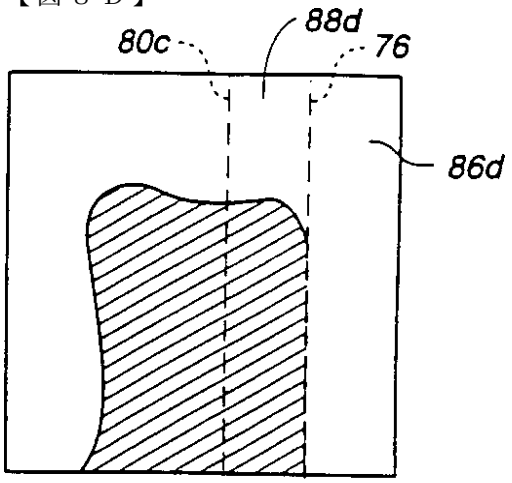

FIG._8D

【図８Ｅ】

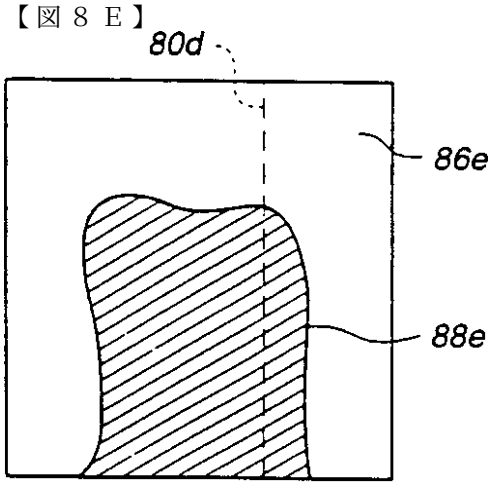

FIG._8E

フロントページの続き

(72)発明者　サーター、トーマス・フランス
アメリカ合衆国、カリフォルニア州　９４０８６、サニーベイル、フローラ・ヴィスタ　４６１

審査官　▲広▼島　明芳

(56)参考文献　特表平０６－５００６５４（ＪＰ，Ａ）
特開平０２－１７１８８１（ＪＰ，Ａ）
特開平０７－０９８７６４（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06T 1/00
A61B 5/117