SEKONIC

OPTICAL MARK READER

Windows 用

API リファレンス・マニュアル

目　次

参考:図番号一覧

1.はじめに

本書は、セコニック製 OMR SR-3500/6000/6500、SR-1800、SR-3500 HYBRID および SR-11000 を USB 通信 (Ver.1.1/2.0) で制御するために提供される API(Application Programming Interface)ライブラリーについて使用方法を解説したものです。(SR-11000 においては USB2.0)
プラットフォームは、Microsoft® Windows®を対象として、OMR 制御アプリケーション開発の際、その所要時間の短縮を目的としたものです。プラットフォームの詳細などは「4.動作環境」をご覧ください。
OMR を制御するためには、パーソナルコンピュータ(以下、PC)が USB で接続されている必要があります。
PC から OMR に対して、制御コマンドを送信することによって OMR が動作します。
この「制御コマンドの送信 (レスポンスの受信)」に対して個々に関数を作成し、またその関数を一括処理する関数を API リファレンスとして利用できるようになっています。
本書では、各関数に対して詳細な説明を与え、各関数に渡すことのできるパラメータを解説します。さらに、簡単な使用例も紹介しています。

各機能の使用例は簡単なものですが、変数の宣言の仕方と、その関数の機能を補足する関数の使い方も示されています。
OMR の取り扱い説明書とコマンドリファレンスをあわせてご利用ください。

本書において、OMR のことを「デバイス」と呼称する場合があります。
また、提供される API ライブラリーの構造はダイナミックリンクライブラリー形式であるため、一般的に「ライブラリー」や「DLL」と呼ばれることがあります。

ご注意

本書の内容は予告無く変更することがあります。
本書の内容について、万一、記述に誤りがあった場合でも、当社は一切の責任を負いかねます。
当社は、過失も含めた如何なる場合においても、ソフトウェアを使用または使用不能から生じた偶発的、特別、間接損害の責任を負わないものとします。

2.API ライブラリー構成

セコニック製 OMR を制御する為のプログラム構成は「図 1 プログラム構成」の通りになります。
お客様が開発するアプリケーションプログラムは、新 API ライブラリーもしくは旧 API ライブラリーの API 関数を使用し、デバイスドライバーを経由して OMR と通信を行います。新旧両方の API ライブラリーを使用することもできます。

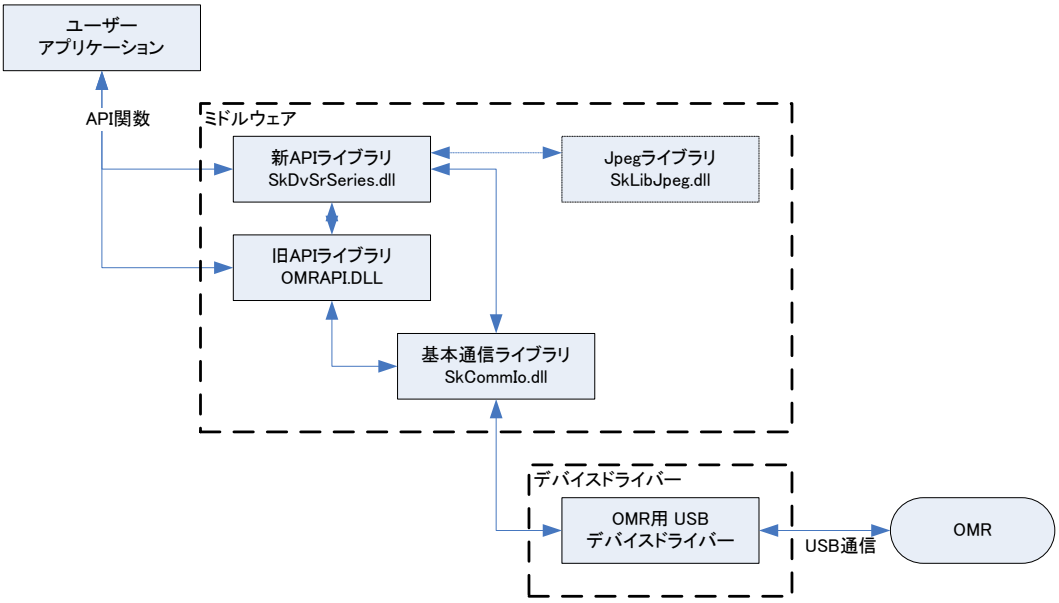


図 1 プログラム構成

	項目	内容
1	ユーザーアプリケーション	お客様が開発するアプリケーションプログラム。
2	新 API ライブラリー	SR-3500/6000/6500、SR-1800、 SR-3500 HYBRID および SR-11000 を制御する為の API ライブラリー。
3	旧 API ライブラリー	SR-3500/6000/6500、SR-1800 を制御する為の API ライブラリー。1つの API 関数につき1つのコマンドによる通信を行う。 画像の読み取りは行えない。
4	Jpeg ライブラリー	SR-3500 HYBRID を使用し画像を Jpeg 形式にてファイル保存するためのライブラリー。 使用するにあたっての条件がある為、「3.3 再配布可能ファイル」の注意事項を参照のこと。
5	基本通信ライブラリー	OMR 用 USB デバイスドライバーと通信を行う為のライブラリー。
6	OMR 用 USB デバイスドライバー	OMR と通信を行う為のデバイスドライバー。
7	OMR	SR-3500/6000/6500、SR-1800、 SR-3500 HYBRID および SR-11000。

3.提供されるファイル

セコニック製 OMR を USB 通信で制御するための API ライブラリーは、次のファイルで提供されます。

3.1. API ライブラリー

API ライブラリーは32bit プログラムです。
お客様がアプリケーションを開発される場合には、32bit プログラムで作成してください。

(1)　新 API ライブラリーを使用する場合

新 API ライブラリーを使用する場合、ユーザーアプリケーションは「SkDvSrSeries.dll」の API 関数を呼び出すことで OMR を制御します。

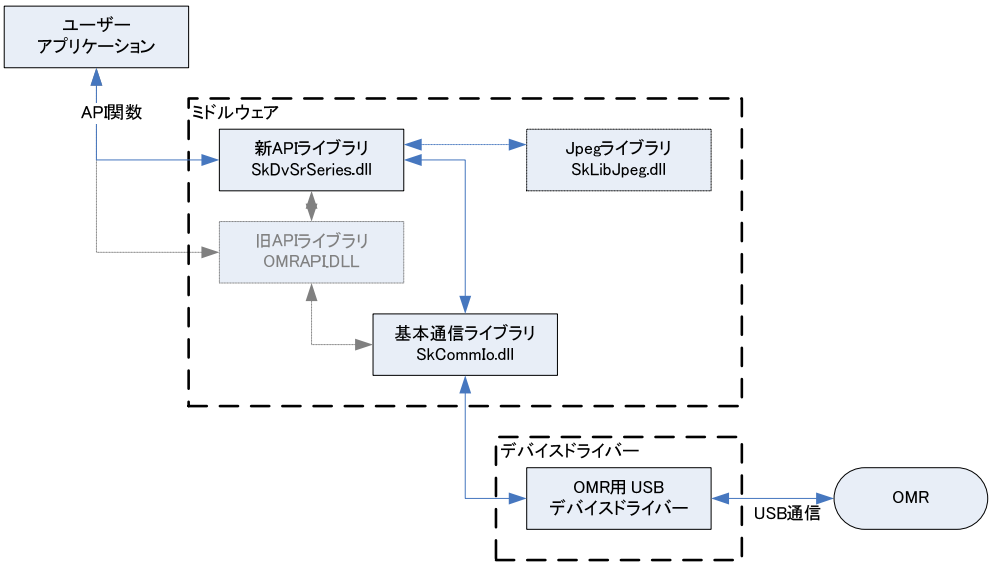


図 2 新 API ライブラリーを使用したプログラム構成

新 API ライブラリーでは、SR-3500 HYBRID の画像データ処理、SR-11000 の連続読み取り動作を実行することができます。
使用方法の詳細は、「5. API リファレンス(新 API ライブラリー)」を参照してください。

項目	ファイル	内容
実行ファイル	SkDvSrSeries.dll	新 API ライブラリー本体。ダイナミックリンクライブラリー。
	SkCommIo.dll	SkDvSrSeries.dll が使用する OMR と通信するためにデバイスドライバーへアクセスするダイナミックリンクライブラリー。
	SkLibJpeg.dll	SR-3500HYBRID を使用する際に画像ファイルを Jpeg ファイル保存に使用するライブラリー。
実行時生成ファイル	SkDvSrSeries.log	ライブラリーの実行時に生成されます。最大４KByte のサイズになります。
	SkCommIo.log	
ヘッダファイル	SkDvSr11000Api.h	SR-11000 専用の API ライブラリーのヘッダーファイル。
	SkCdSr3500ImgApi.h	SR-3500HYBRID 専用の API ライブラリーのヘッダファイル。
	SkDvSrOptionApi.h	オプションの API ライブラリーのヘッダファイル。
	SkDvSrBaseApi.h	OMR の基本となる API ライブラリーのヘッダファイル。
	SkDvSr11000Prm.h	SR-11000 専用の定数定義、構造体を定義しているヘッダファイル。
	SkDvSrSeriesPrm.h	定数定義、構造体を定義しているヘッダファイル。
	SkDvSrSeriesErr.h	SkDvSrSeries.dll のステータス定数を定義しているヘ

		ッダファイル。
	SkCommIoErr.h	SkCommIo.dll のステータス定数を定義しているヘッダファイル。
ライブラリー	SkDvSrSeries.lib	新 API ライブラリーを使用するときにリンクする導入ライブラリー。

(2)　　旧 API ライブラリーを使用する場合

旧 API ライブラリーを使用する場合、ユーザーアプリケーションは「OMRAPI.DLL」の API 関数を呼び出すことで OMR を制御します。

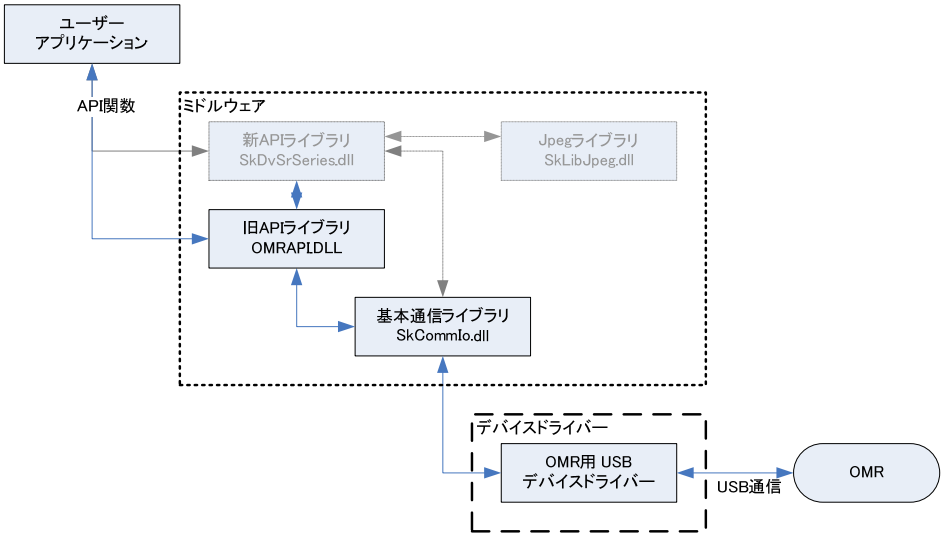


図 3 旧 API ライブラリーを使用したプログラム構成

旧 API ライブラリーでは、SR-3500 HYBRID にて画像データ、SR-11000 での連続読み取り動作を扱うことはできません。
旧 API ライブラリーの Ver4.1 以前は、「OMRAPI.DLL」のみで動作しておりましたが、Ver5.0 より「SkCommIo.dll」も必要になります。 ただし、処理内容は Ver4.1 と Ver5.0 で同等です。

使用方法の詳細は、「」を参照してください。

項目	ファイル	内容
実行ファイル	OMRAPI.DLL	API 機能を搭載した本体･ダイナミックリンクライブラリー。
	SkCommIo.dll	OMRAPI.DLL が使用するダイナミックリンクライブラリー。
実行時生成ファイル	SkCommIo.log	ライブラリーの実行時に生成されます。最大４KByte のサイズになります。
ヘッダファイル	OMRAPI.h	OMRAPI.DLL のヘッダファイル。
ライブラリー	OMRAPI.LIB	OMRAPI.DLL を使用するときにリンクする導入ライブラリー。

(3)　　新/旧 API ライブラリーを使用する場合

SR-3500 HYBRID の画像データの処理と SR-3500/6000/6500 でバーコードリーダーやプリンターを使用する場合には、「新 API ライブラリー」と「旧 API ライブラリー」の両方を使用してください。

使用方法の詳細は、「5. API リファレンス（新 API ライブラリー）」と「6. API リファレンス（旧 API ライブラリー）」を参照してください。

実行ファイルなどは、「新 API ライブラリー」と「旧 API ライブラリー」の両方が必要になります。

3.2. デバイスドライバー

以下は、セコニック製 OMR 用 USB デバイスドライバーインストーラです。
下記のファイルを実行し、表示されるメッセージに従いインストールを行ってください。

DriverInstaller4.3.exe

※1:「4.3」はインストールされるドライバーのバージョンを表わしています。
※2:このバージョンは更新される場合があります。

3.3. 再配布可能ファイル

下記ファイルの再配布は、お客様が本ライブラリーを用いて開発したアプリケーションとともに行われる場合に限り許諾されます。 下記ファイルのみを配布することはできません。

DriverInstaller*.*.exe　※「*.*」はバージョン番号
SkDvSrSeries.dll
OMRAPI.DLL
SkCommIo.dll
SkLibJpeg.dll　※注意

※注意
本ライブラリーに含まれる Jpeg ライブラリ(SkLibJpeg.dll)は、Independent JPEG Group のプログラムを使用しております。Jpeg ライブラリ(SkLibJpeg.dll)を使用し再配布される場合には、Independent JPEG Group のプログラムを使用している旨を記述する必要があります。

4.動作環境

4.1. 動作環境

本 API ライブラリーが動作する環境は次のとおりです。

OS

Microsoft Windows XP SP3以上 32bit 版
Microsoft Windows Vista 64bit / 32bit 版
Microsoft Windows 7 64bit / 32bit 版
Microsoft Windows 8 64bit / 32bit 版
※:64bit の OS では、WOW64 で動作します。
※:Windows7 の XP モードでは動作しません。
※:Windows8 のクライアント Hyper-V では動作しません。
※:Microsoft 社のサポート終了をもって、動作環境から除外します。

CPU

Intel Celeron 2GHz 以上またはそれに相当する CPU(推奨:Pentium4 3GHz 以上)

ディスク容量

10MB 以上の空きスペース
※:ユーザーアプリケーションやデータ保存用の容量は含みません。

メモリ容量

256MB 以上。(512MB 以上を推奨)
※:OS のシステム要件以上であること。

インターフェイス

USB1.1 および USB2.0。
※:SR-3500HYBRID で画像データを扱う際は、USB2.0 を使用すること。
※:SR-11000 では、USB2.0 を使用すること。

同時接続可能台数

1台

制限事項

- 本 API ライブラリーで OMR を制御中には OMR のパネル操作を行わないでください。正常に動作しない場合があります。
- 本 API ライブラリーで OMR を制御中に他のアプリケーションを起動したり操作したりしないでください。処理能力が極端に低下する場合があります。

4.2. 開発ツール

この DLL は Windows 標準の API を内部で使用しています。
その他のサードパーティ製のライブラリーなどは別途用意する必要はありませんので、一般的な Windows アプリケーション開発ツール(Visual C++、Visual Basic、Visual C#、C++ Builder など)から利用することができます。

開発ツールから DLL を利用する場合は、各ツールのマニュアル、参考書などをご覧ください。
また Win32API の LoadLibrary 関数を使ってコード中に DLL 呼び出しルーチンを記述することも可能です。
Win32API の参考書などをご覧ください。

本書では、C 言語で記述しています。実際にコードを書くときは使用するライブラリーに応じてヘッダファイルとライブラリーをインクルードするように記述します。
C 言語以外をご使用の場合には、その言語に合わせて読み替えてください。

作成したアプリケーションを実行するときは、OMRAPI.DLL/SkDvSrSeries.dll/SkCommIo.dll を実行ファイルと同じフォルダに置くか、システムフォルダなどパスの通っているフォルダに置いてください。

5.API リファレンス(新 API ライブラリー)

SR-11000 よりご提供しております「SkSrSeries.dll」の API リファレンスです。
SR-3500/6000/6500、SR-1800、SR-3500 HYBRID、 SR-11000 を USB コマンドにより制御することができます。

5.1. API 関数一覧

API 関数名のプレフィックスは下表を基準としています。

API関数名 プレフィックス	内　容
SkDv_	本ライブラリーの基本となる制御や各種一連の制御を行う関数。
SkDv_Req	デバイスとの通信を伴い、一連の制御を行う関数。 デバイスとの通信は複数のコマンドを伴う場合もある。
SkDv_Get/SkDv_Set	本ライブラリー内部で保持しているデータを制御する関数。 ただし、デバイスとの通信を行わなければ得られないデータの場合には、通信が行われる。
SkCd_	1つのコマンドの処理を行う関数。 ただし、帳票読み取り要求の場合、レスポンスが複数ある為に分割されている。

分類	名 称	API関数名	内　容
基本機能	モジュール情報取得	SkDv_GetModuleInfo	本ライブラリーの名称とバージョンを取得する。 ライブラリーがロードさせていれば、事前に開始処理など行う必要はない。
	開始処理	SkDv_OpenSingle	ＯＭＲの制御を開始する。 １台接続のみ対応し、既に開始処理を行っているデバイスがある場合エラーを返す。
	開始処理 （ＯＭＲＡＰＩ併用）	SkDv_OpenWithOmrapi	ＯＭＲＡＰＩを併用する場合の開始処理。
	終了処理	SkDv_Close	すべての処理を終了する際に実行し、各種作業用メモリの開放を行う。
	初期化要求	SkDv_ReqInit	デバイスの動作条件を初期化し、画像データや内部処理用のデータなども初期化される。また、デバイスに対しソフトリセットも実行される。
	デバイス情報取得	SkDv_GetInfo	デバイスのバージョン情報や所持する機能などの情報を取得する。
	動作モード取得	SkDv_GetMode	動作モードを取得する。動作モードは本ライブラリー内で保持される内容。
	動作モード設定	SkDv_SetMode	動作モードを設定する。動作モードは本ライブラリー内で保持される。
	マーク読み取り条件取得	SkDv_GetMarkConf	マーク読み取りの動作条件を取得する。動作条件は本ライブラリー内で保持される内容。
	マーク読み取り条件設定	SkDv_SetMarkConf	マーク読み取りの動作条件を設定する。動作条件はライブラリー内で保持される。
	ステータス情報要求	SkCd_GetStatus	通信を行いデバイスのステータスを取得する。

分類	名 称	API関数名	内　容
	センサ情報要求	SkCd_GetSensor	通信にてデバイスのセンサ状態を取得する。
	読み取り要求 （マークのみ）	SkDv_ReqFeedMarkSheet	マークの読み取りを行う。イメージの取得は行われない。 デバイス情報/マーク条件の取得通信が行われていない場合には、読み取り前に通信が行われる。
	排出要求	SkDv_ReqEjectForm	帳票の排出を行う。
	エラークリア要求	SkDv_ReqClearError	デバイスに対してエラークリアの通信を行う。
	マークデータ取得要求	SkDv_ReqGetMarkData	通信を行い、マークデータを取得する。
拡張機能	デバイス情報取得通信要求	SkDv_ReqGetInfo	デバイスに対し、情報を取得する為の通信を行う。
	動作モード取得通信要求	SkDv_ReqGetMode	デバイスに対し、動作モードを取得する為の通信を行う。
	動作モード設定通信要求	SkDv_ReqSetMode	デバイスに対し、動作モードを設定する為の通信を行う。
	マーク条件取得通信要求	SkDv_ReqGetMarkConf	デバイスに対し、マーク読み取りの動作条件を取得する為の通信を行う。
	マーク条件設定通信要求	SkDv_ReqSetMarkConf	デバイスに対し、マーク読み取りの動作条件を設定する為の通信を行う。

分類	名 称	API関数名	内　容
SR-3500HYBRID	イメージ読み取り条件取得	SkDv_GetImageConf	イメージ読み取りの動作条件を取得する。動作条件は本ライブラリー内で保持される内容。
	イメージ読み取り条件設定	SkDv_SetImageConf	イメージ読み取りの動作条件を設定する。動作条件は本ライブラリー内で保持される。
	イメージ条件取得通信要求	SkDv_ReqGetImageConf	デバイスに対し、イメージ読み取りの動作条件を取得する為の通信を行う。
	イメージ条件設定通信要求	SkDv_ReqSetImageConf	デバイスに対し、イメージ読み取りの動作条件を設定する為の通信を行う。
	読み取り要求（マークとイメージ）	SkDv_ReqScanForm	帳票の読み取りを行う。デバイス情報/マーク条件/イメージ条件/CIS 基準データの取得通信が行われていない場合には、読み取り前に通信が行われる。
	帳票サイズ取得	SkDv_GetFormSize	読み取りを行った帳票のサイズを取得する。
	画像データファイル保存	SkDv_SaveImageData	本ライブラリーが保持している画像データをファイルに保存する。切り出し位置、回転やファイル形式などを指定することができる。
	画像データファイル保存（マルチスレッド）	SkDv_SaveImageDataThread	指定した範囲のイメージデータを別のスレッドにてファイルに保存する。
	別スレッド動作中確認	SkDv_IsThreadRunning	SkDv_SaveImageDataThread 関数で実行された別スレッドが終了したか確認する。
	画像データ生成	SkDv_CreateImage	メモリを確保し、画像データを格納する。切り出し位置、回転やなどを指定することができる。
	画像データ開放	SkDv_DestroyImage	画像データ取得で確保したメモリを開放する。画像データ取得を行った場合、必ず行わなければならない。

分類	名 称	API関数名	内　容
SR-1000	ID 管理設定要求	SkDv_ReqSetLayoutManage	ウィンドウ制御を行うための ID 管理設定を行う。
	レイアウト設定終了要求	SkDv_ReqSetLayoutManageTerminate	ウィンドウ制御のレイアウト設定（ID データ設定、ウィンドウ領域データ設定）を終了する。
	帳票 ID 設定要求	SkDv_ReqSetLayoutId	帳票の ID を設定し、ウィンドウエリアを設定出来るようにする。
	ウィンドウエリア設定要求	SkDv_ReqSetWindowArea	ウィンドウエリアの設定を行う。
	レイアウト設定クリア	SkDv_ReqClearLayout	レイアウト設定の内容をすべてクリアする。
	LCD 表示要求	SkDv_ReqSetPanelUserEnable	OMR のパネルスイッチでの操作を禁止する。
	パネルスイッチ状態取得要求	SkDv_ReqGetPanelUserSwitch	OMR のパネルスイッチの状態を取得する。
	連続読み取り固定印字文字設定要求	SkDv_ReqSetPrintFixString	連続読み取りの固定印字文字を設定する。

	連続読み取り連番印字設定要求	SkDv_ReqSetContFeedPrint	連続読み取りの連番印字を設定する。
	連続読み取り開始要求	SkDv_ReqContFeedSheet	連続読み取りを開始する。
	連続読み取り停止要求	SkDv_ReqContFeed_Cancel	連続読み取りの動作をキャンセルする。
	連続読み取り動作中確認	SkDv_IsContFeedRunning	連続読み取りが動作しているか確認する。
	連続読み取り動作終了確認	SkDv_IsContFeedDataFinished	「連続読み取り要求」関数で開始した連続読み取りが終了しデータもなくなっているかどうかの状態を取得する。
	連続読み取りデータ蓄積確認	SkDv_ExistDoneData	「連続読み取り要求」関数で開始した連続読み取りのデータが蓄積されているか確認する。
	搬送状況取得	SkDv_GetContFeedCount	連続読み取りのワーカースレッドが受け取った現在の枚数状況を取得する。
	連続読み取りデータ準備	SkDv_PrepareContData	連続読み取りのワーカースレッドが受け取ったデータを取得。準備を行う。
	データ番号取得	SkDv_GetContDataNumber	連続読み取りデータの現在のデータ番号（枚数）を取得する。
	連続読み取りマークデータ取得	SkDv_GetContDataMarks	連続読み取りデータ準備で準備されたウィンドウデータを取得する。
	連続読み取りバーコードデータ取得	SkDv_GetContDataBarcodes	連続読み取りデータ準備で準備されたバーコードデータを取得する。
	連続読み取りのバーコードデータ(読取数)取得	SkDv_GetContDataBarcodesCount	連続読み取りデータ準備で準備された読み取ったバーコード数を取得する
	連続読み取りのバーコードデータ(データ)取得	SkDv_GetContDataBarcodesData	連続読み取りデータ準備で準備された読み取ったバーコードの指定した番号のデータを取得する
	連続読み取り帳票読み取り結果取得	SkDv_GetContDataSheetResult	連続読み取りデータ準備で準備された帳票読み取り結果を取得する。
	連続読み取り結果取得	SkDv_GetContFeetResult	連続読み取り結果を取得する。

分類	名 称	API関数名	内　容
オプション(バーコードリーダ)	読取条件取得	SkDv_GetBcrConf	ライブラリ内に格納されているバーコードリーダ読取条件を取得する。
	読取条件設定	SkDv_SetBcrConf	バーコードリーダ読取条件をライブラリ内に設定する。
	読取条件取得通信要求	SkDv_ReqGetBcrConf	デバイスに対し、バーコードリーダ読取条件を取得する為の通信を行う。
	読取条件設定通信要求	SkDv_ReqSetBcrConf	デバイスに対し、バーコードリーダ読取条件を設定する為の通信を行う。
	個数データ取得通信要求	SkDv_ReqGetBcrDataCount	読み取ったバーコードの数を通信で取得する。
	読み取りデータ取得通信要求	SkDv_ReqGetBcrData	読み取ったバーコードの指定された番号のデータを通信で取得する。
オプション(プリンタ)	印字条件取得	SkDv_GetPrinterConf	ライブラリー内に格納されているプリンタ印字条件を取得する。
	印字条件設定	SkDv_SetPrinterConf	プリンタ印字条件をライブラリー内に設定する。
	印字文字設定通信要求	SkDv_ReqPrintString	印字したい文字をデバイスに設定する。
	印字条件取得通信要求	SkDv_ReqGetPrinterConf	デバイスに対し、プリンタの印字条件を取得する為の通信を行う。
	印字条件設定通信要求	SkDv_ReqSetPrinterConf	デバイスに対し、プリンタの印字条件を設定する為の通信を行う。

5.2. 定数

5.2.1. ステータスコード概要

API の戻り値は SkDvStatus 型が基本となります。
SkDvStatus は、以下の通り定義されています。

```
typedef    DWORD    SkDvStatus;
```

ステータスコードを16進数で「0xAABBCCCC」とした場合、それぞれ AA, BB, CCCC の意味は以下の通りとなります。

	ビット位置	16進数	内容
1	31～24	0xAA------	固定(0x20)
2	23～16	0x--BB----	発生したデバイスやライブラリーを表すコード。 値 / 意味 0x2032---- / 本ライブラリーのステータスコード 0x2021---- / 下位通信ライブラリーのステータスコード 0x2002---- / デバイスのステータスコード
3	15～ 0	0x----CCCC	デバイスやライブラリー内のコード

デバイスのステータスコードは ST1 と ST2 の2つがありますが、優先度の高いコードを通知します。
ライブラリーのステータスコードでは、上位2バイトは 0x2002 固定で、下位2バイトを下記のように変換して通知します。
ただし、ステータスコード(ST1 および ST2)が「00」の場合は正常である為、ライブラリーのステータスコードとしては通知されません。

	ビット位置	2進数	内容
1	15～11	0bDDDDD----------	ステータスコード(STA)の1文字目。下表により変換される。 変換値 / ステータスコード文字 / ASCII コード 0b00000 / 0 / 0x30 0b00001 / A / 0x41 \ / \ / \ 0b11010 / Z / 0x5A 0b11111 / @ / 0x40
2	10～ 7	0b-----EEEE-------	ステータスコード(STA)の2文字目。下表により変換される。 変換値 / ステータスコード文字 / ASCII コード 0b0000 / 0 / 0x30 \ / \ / \ 0b1001 / 9 / 0x39 0b1111 / @ / 0x40
3	6～ 5	0b---------FF-----	ST1 と ST2 の発生状況。下表により変換される。 変換値 / 意味 0b00 / ST1 および ST2 で別々に発生 0b01 / ST1 で発生 0b10 / ST2 で発生 0b11 / ST1 および ST2 で発生
4	4～ 0	0b----------GGGGG	エラー詳細情報。ステータスコード(STA)の1文字目と同様に変換される。

5.2.2. ステータスコード一覧

〇ステータスコードの部位

定数名	実際の値	内容
SK_STS_MODULE_MASK	0xffff0000	モジュール定義部マスク
SK_STS_MODULE	0x20310000	本モジュール
SK_STS_MODULE_IO	0x20210000	下位 基本通信ライブラリー
SK_STS_DEVICE_3500	0x20020000	デバイスからのエラー

〇ライブラリーのステータスコード

定数名	実際の値	内容
SKDV_STS_SUCCESS	0x00000000	正常
SKDV_STS_EXECUTE_ERR	0x20310001	WindowAPI 実行エラー
SKDV_STS_CREATE_EVENT_ERR	0x20310006	イベント作成失敗
SKDV_STS_CREATE_THREAD_ERR	0x20310007	スレッド作成失敗
SKDV_STS_USED_DEVICE	0x2031000E	対象デバイス使用中
SKDV_STS_WAIT_TIMEOUT	0x2031000F	タイムアウト
SKDV_STS_HANDLE_ERR	0x20311001	ハンドルが異常
SKDV_STS_MEM_ALLOC_ERR	0x20311002	メモリ確保失敗
SKDV_STS_LOAD_SUBLIB_ERR	0x20311003	下位ライブラリーのロード失敗
SKDV_STS_PARAM_ERR	0x20311010	パラメータ異常
SKDV_STS_PARAM_ERR1	0x20311011	パラメータ 1 が異常
SKDV_STS_PARAM_ERR2	0x20311012	パラメータ 2 が異常
SKDV_STS_PARAM_ERR3	0x20311013	パラメータ 3 が異常
SKDV_STS_PARAM_ERR4	0x20311014	パラメータ 4 が異常
SKDV_STS_PARAM_ERR5	0x20311015	パラメータ 5 が異常
SKDV_STS_PARAM_ERR6	0x20311016	パラメータ 6 が異常
SKDV_STS_PARAM_ERR7	0x20311017	パラメータ 7 が異常
SKDV_STS_PARAM_ERR8	0x20311018	パラメータ 8 が異常
SKDV_STS_PARAM_ERR9	0x20311019	パラメータ 9 が異常
SKDV_STS_FUNCTION_ERR	0x2031101F	無効関数
SKDV_STS_PARAM_ERR_SCANNINGSIDE	0x20311011	裏面読み取り異常
SKDV_STS_PARAM_ERR_SCAN_CTRL	0x20311012	読み取りり方式異常
SKDV_STS_PARAM_ERR_FORMLENGTH_MAX	0x20311014	帳票長異常
SKDV_STS_PARAM_ERR_FORMLENGTH_MIN	0x20311015	帳票長異常
SKDV_STS_PARAM_ERR_CIS_LIGHT	0x20311016	光源異常(+iFace)
SKDV_STS_PARAM_ERR_CIS_LIGHT_F	0x20311016	光源異常(表面)
SKDV_STS_PARAM_ERR_CIS_LIGHT_B	0x20311017	光源異常(裏面)
SKDV_STS_PARAM_ERR_RESOLUTION	0x20311018	解像度異常
SKDV_STS_CIS_BASE_DATA_ERR	0x20311021	CIS 基準データ異常

定数名	実際の値	内容
SKDV_STS_NO_DATA	0x20311100	データがない
SKDV_STS_ROTATION_ERR	0x20311102	画像回転失敗
SKDV_STS_IMAGE_PROCESS_ERR	0x203111ff	画像データ処理失敗
SKDV_STS_FILE_ERR	0x20311200	ファイル処理失敗
SKDV_STS_FILE_SAVE_ERR	0x20311201	ファイル保存失敗
SKDV_STS_FILE_LOAD_ERR	0x20311211	ファイル読込失敗
SKDV_STS_FILE_FORMAT_ERR	0x20311231	ファイルフォーマット異常
SKDV_STS_DV_IMAGE_ERR	0x20311301	画像データ転送失敗(デバイスからの通知)
SKDV_STS_NO_IMAGE	0x20311311	画像データがない
SKDV_STS_OVER_LENGTH_IMAGE	0x20311312	画像データ量超過
SKDV_STS_DIFF_IMAGE_WIDTH	0x20311313	画像データの幅が異なる
SKDV_STS_DIFF_IMAGE_HEIGHT	0x20311314	画像データの長さが異なる
SKDV_STS_IMAGE_LENGTH_ERR	0x20311315	画像データの長さが異常
SKDV_STS_RECVED_NAK_CMD	0x20312101	NAK 受信(コマンド異常)
SKDV_STS_RECVED_NAK_PRM	0x20312102	NAK 受信(パラメータ異常)
SKDV_STS_RECVED_NAK_NON	0x20312103	NAK 受信(未サポート)
SKDV_STS_RECVED_NAK_OPE	0x20312104	NAK 受信(パネル操作中)
SKDV_STS_RECVED_NAK_DOING	0x20312105	NAK 受信(動作中)
SKDV_STS_RECVED_NAK_UNKNOWN	0x2031210F	未定義の NAK 受信
SKDV_STS_RECV_ACK_TIMEOUT	0x20312200	ACK 受信タイムアウト
SKDV_STS_RECV_RES_TIMEOUT	0x20312201	レスポンス受信タイムアウト
SKDV_STS_DIFF_ACK_CMD	0x20312210	ACK の CMD コードが異なる
SKDV_STS_DIFF_RES_CMD	0x20312211	レスポンスの CMD コードが異なる
SKDV_STS_SHORT_RESPONSE_ERR	0x20312220	レスポンス長が短い
SKDV_STS_LONG_RESPONSE_ERR	0x20312221	レスポンス長が長い
SKDV_STS_RESPONSE_FORMAT_ERR	0x20312222	レスポンスの形式が異常
SKDV_STS_DIFF_PRM	0x20312223	設定コマンドの結果、値が異なる
SKDV_STS_CALIB	0x20313000	キャリブレーションエラー
SKDV_STS_CALIB_CHK_BG_EVEN_BL_F_ERR	0x20313010	LED 消灯データのバラツキ
SKDV_STS_CALIB_CHK_BG_EVEN_BL_B_ERR	0x20313011	(裏面)
SKDV_STS_CALIB_CHK_BG_EVEN_WH_F_ERR	0x20313012	LED 点灯データのバラツキ
SKDV_STS_CALIB_CHK_BG_EVEN_WH_B_ERR	0x20313013	(裏面)
SKDV_STS_CALIB_CHK_BG_DIFF_F_ERR	0x20313014	点灯/消灯の差が大きい
SKDV_STS_CALIB_CHK_BG_DIFF_B_ERR	0x20313015	(裏面)

定数名	実際の値	内容
SKDV_STS_CALIB_CIS_BS_BLACK_EVEN_F_ERR	0x20313020	黒基準のバラツキ
SKDV_STS_CALIB_CIS_BS_BLACK_EVEN_B_ERR	0x20313021	(裏面)
SKDV_STS_CALIB_CIS_BS_WHITE_EVEN_F_ERR	0x20313022	白基準のバラツキ
SKDV_STS_CALIB_CIS_BS_WHITE_EVEN_B_ERR	0x20313023	(裏面)
SKDV_STS_CALIB_CIS_BS_RED_EVEN_F_ERR	0x20313024	赤基準のバラツキ
SKDV_STS_CALIB_CIS_BS_RED_EVEN_B_ERR	0x20313025	(裏面)
SKDV_STS_CALIB_CIS_BS_GREEN_EVEN_F_ERR	0x20313026	緑基準のバラツキ
SKDV_STS_CALIB_CIS_BS_GREEN_EVEN_B_ERR	0x20313027	(裏面)
SKDV_STS_CALIB_CIS_BS_BLUE_EVEN_F_ERR	0x20313028	青基準のバラツキ
SKDV_STS_CALIB_CIS_BS_BLUE_EVEN_B_ERR	0x20313029	(裏面)
SKDV_STS_CALIB_CIS_BS_BLACK_UPPER_F_ERR	0x20313030	黒基準の上限
SKDV_STS_CALIB_CIS_BS_BLACK_UPPER_B_ERR	0x20313031	(裏面)
SKDV_STS_CALIB_CIS_BS_WHITE_RANGE_F_ERR	0x20313032	白基準の有効範囲
SKDV_STS_CALIB_CIS_BS_WHITE_RANGE_B_ERR	0x20313033	(裏面)
SKDV_STS_CALIB_CIS_BS_RED_RANGE_F_ERR	0x20313034	赤基準の有効範囲
SKDV_STS_CALIB_CIS_BS_RED_RANGE_B_ERR	0x20313035	(裏面)
SKDV_STS_CALIB_CIS_BS_GREEN_RANGE_F_ERR	0x20313036	緑基準の有効範囲
SKDV_STS_CALIB_CIS_BS_GREEN_RANGE_B_ERR	0x20313037	(裏面)
SKDV_STS_CALIB_CIS_BS_BLUE_RANGE_F_ERR	0x20313038	青基準の有効範囲
SKDV_STS_CALIB_CIS_BS_BLUE_RANGE_B_ERR	0x20313039	(裏面)
SKDV_STS_CALIB_CHK_PL_UPPER_F_ERR	0x20313040	プラテンが白すぎる
SKDV_STS_CALIB_CHK_PL_UPPER_B_ERR	0x20313041	(裏面)
SKDV_STS_CALIB_LS_UPPER_LEVEL_ERR	0x20313101	左端が検索範囲の前(先頭 dot からレベル以上)
SKDV_STS_CALIB_LS_UNDER_LEVEL_ERR	0x20313102	左端が検索範囲の先(最終 dot までレベル未満)
SKDV_STS_USER_CANCEL	0x2031f001	上位の指示による処理の中断
SKIO_STS_SUCCESS	0x00000000	正常
SKIO_STS_EXECUTE_ERR	0x20210001	WindowAPI 実行エラー

定数名	実際の値	内容
SKIO_STS_NONE_DEVICE	0x20210002	対象デバイスなし
SKIO_STS_HANDLE_ERR	0x20210003	ハンドルエラー
SKIO_STS_PARAM_ERR	0x20210004	パラメータエラー
SKIO_STS_MEM_ALLOC_ERR	0x20210005	メモリ確保失敗
SKIO_STS_CREATE_EVENT_ERR	0x20210006	イベント作成失敗
SKIO_STS_USED_DEVICE	0x2021000e	対象デバイス使用中
SKIO_STS_WAIT_TIMEOUT	0x2021000f	タイムアウト

○デバイスのステータスコード

ステータス情報(ST1/ST2)に対してのみ定数を定義し、詳細コードについては定義しておりません。ビットにマスクをかけてご使用ください。

定数名	実際の値	内容
SKDV_STS_DEVICE_STA_MASK	0xffffff80	デバイスからのステータスコードマスク
SKDV_STS_DEVICE_ST_A1	0x20020880	"A1"本体部メモリエラー 1
SKDV_STS_DEVICE_ST_A2	0x20020900	"A2"本体部メモリエラー 2
SKDV_STS_DEVICE_ST_A3	0x20020980	"A3"本体部ホッパ駆動エラー
SKDV_STS_DEVICE_ST_A4	0x20020A00	"A4"本体部ダウンロードエラー
SKDV_STS_DEVICE_ST_A5	0x20020A80	"A5"本体部センサタイプエラー
SKDV_STS_DEVICE_ST_A6	0x20020B00	"A6"本体部オプションエラー
SKDV_STS_DEVICE_ST_A8	0x20020C00	"A8"本体部電源電圧エラー
SKDV_STS_DEVICE_ST_B1	0x20021080	"B1"読み取りユニット通信回線エラー
SKDV_STS_DEVICE_ST_B2	0x20021100	"B2"読み取りユニット内部通信エラー
SKDV_STS_DEVICE_ST_B3	0x20021180	"B3"読み取りユニットメモリエラー
SKDV_STS_DEVICE_ST_B4	0x20021200	"B4"読み取りユニット補正値エラー
SKDV_STS_DEVICE_ST_B5	0x20021280	"B5"読み取りユニットダウンロードエラー
SKDV_STS_DEVICE_ST_B6	0x20021300	"B6"読み取りユニット内部エラー
SKDV_STS_DEVICE_ST_B7	0x20021380	"B7"読み取りユニットバージョンエラー
SKDV_STS_DEVICE_ST_C1	0x20021880	"C1"バーコードユニット通信回線エラー
SKDV_STS_DEVICE_ST_C2	0x20021900	"C2"バーコードユニット内部通信エラー
SKDV_STS_DEVICE_ST_C3	0x20021980	"C3"バーコードユニットメモリエラー
SKDV_STS_DEVICE_ST_C4	0x20021A00	"C4"バーコードユニットセンサエラー
SKDV_STS_DEVICE_ST_C5	0x20021A80	"C5"バーコードユニットダウンロードエラー
SKDV_STS_DEVICE_ST_C6	0x20021B00	"C6"バーコードユニット内部エラー
SKDV_STS_DEVICE_ST_C7	0x20021B80	"C7"バーコードユニットバージョンエラー
SKDV_STS_DEVICE_ST_D1	0x20022080	"D1"プリンターユニット通信回線エラー
SKDV_STS_DEVICE_ST_D2	0x20022100	"D2"プリンターユニット内部通信エラー
SKDV_STS_DEVICE_ST_D3	0x20022180	"D3"プリンタユニットメモリエラー
SKDV_STS_DEVICE_ST_D4	0x20022200	"D4"プリンタユニットダウンロードエラー
SKDV_STS_DEVICE_ST_D5	0x20022280	"D5"プリンターユニット内部エラー
SKDV_STS_DEVICE_ST_D6	0x20022300	"D6"プリンターユニットバージョンエラー
SKDV_STS_DEVICE_ST_E1	0x20022880	"E1"スタッカユニット通信回線エラー
SKDV_STS_DEVICE_ST_E2	0x20022900	"E2"スタッカユニット内部通信エラー
SKDV_STS_DEVICE_ST_E3	0x20022980	"E3"スタッカユニットメモリエラー
SKDV_STS_DEVICE_ST_E4	0x20022A00	"E4"スタッカユニットダウンロードエラー
SKDV_STS_DEVICE_ST_E5	0x20022A80	"E5"スタッカユニット内部エラー
SKDV_STS_DEVICE_ST_E6	0x20022B00	"E6"スタッカユニットバージョンエラー
SKDV_STS_DEVICE_ST_E7	0x20022B80	"E7"駆動エラー(スタッカ)
SKDV_STS_DEVICE_ST_J1	0x20025080	"J1"画像読み取りユニットダウンロード異常

定数名	実際の値	内容
SKDV_STS_DEVICE_ST_J2	0x20025100	"J2"画像読み取りユニット内部通信エラー1
SKDV_STS_DEVICE_ST_J3	0x20025180	"J3"画像読み取りユニット内部通信エラー2
SKDV_STS_DEVICE_ST_J4	0x20025200	"J4"画像読み取りユニット外部 Flash 読込みエラー
SKDV_STS_DEVICE_ST_J5	0x20025280	"J5"画像読み取りユニット外部 Flash 書込みエラー
SKDV_STS_DEVICE_ST_J6	0x20025300	"J6"画像読み取りユニット E2PROM 読込みエラー
SKDV_STS_DEVICE_ST_J7	0x20025380	"J7"画像読み取りユニット E2PROM 書込みエラー
SKDV_STS_DEVICE_ST_J8	0x20025400	"J8"画像読み取りユニット IC 異常
SKDV_STS_DEVICE_ST_J9	0x20025480	"J9"画像読み取りユニット CIS 異常
SKDV_STS_DEVICE_ST_K1	0x20025880	"K1"画像読み取りユニット FPGA 異常
SKDV_STS_DEVICE_ST_K2	0x20025900	"K2"画像読み取りユニット電源電圧異常
SKDV_STS_DEVICE_ST_K3	0x20025980	"K3"画像読み取りユニットバージョンエラー
SKDV_STS_DEVICE_ST_F5	0x20023280	"F5"コマンドエラー
SKDV_STS_DEVICE_ST_F6	0x20023300	"F6"パラメータエラー
SKDV_STS_DEVICE_ST_F7	0x20023380	"F7"プロトコルエラー
SKDV_STS_DEVICE_ST_G1	0x20023880	"G1"本体カバーオープン
SKDV_STS_DEVICE_ST_G2	0x20023900	"G2"スタッカユニットカバーオープン
SKDV_STS_DEVICE_ST_H1	0x20024080	"H1"ノーフィード
SKDV_STS_DEVICE_ST_H2	0x20024100	"H2"給紙検知部ジャム
SKDV_STS_DEVICE_ST_H3	0x20024180	"H3"読み取り開始検知部ジャム
SKDV_STS_DEVICE_ST_H4	0x20024200	"H4"本体排紙検知部ジャム
SKDV_STS_DEVICE_ST_H5	0x20024280	"H5"本体シート間隔エラー
SKDV_STS_DEVICE_ST_I1	0x20024880	"I1"プリンタ印字開始検知部ジャム
SKDV_STS_DEVICE_ST_I2	0x20024900	"I2"メイン排紙部ジャム
SKDV_STS_DEVICE_ST_I3	0x20024980	"I3"セレクト排紙部ジャム
SKDV_STS_DEVICE_ST_I4	0x20024A00	"I4"搬送路上部ジャム(スタッカ)
SKDV_STS_DEVICE_ST_I5	0x20024A80	"I5"搬送路下部ジャム(スタッカ)
SKDV_STS_DEVICE_ST_I6	0x20024B00	"I6"スタッカシート間隔エラー
SKDV_STS_DEVICE_ST_P1	0x20028080	"P1"裏面読み取りユニット未接続
SKDV_STS_DEVICE_ST_P2	0x20028100	"P2"バーコードユニット未接続
SKDV_STS_DEVICE_ST_P3	0x20028180	"P3"プリンターユニット未接続
SKDV_STS_DEVICE_ST_P4	0x20028200	"P4"スタッカユニット未接続
SKDV_STS_DEVICE_ST_P5	0x20028280	"P5"画像読み取りユニット未接続
SKDV_STS_DEVICE_ST_Q1	0x20028880	"Q1"シートエンプティ
SKDV_STS_DEVICE_ST_Q2	0x20028900	"Q2"DF (ダブルフィード) エラー
SKDV_STS_DEVICE_ST_Q3	0x20028980	"Q3"左端スキューエラー
SKDV_STS_DEVICE_ST_Q4	0x20028A00	"Q4"マークスキューエラー
SKDV_STS_DEVICE_ST_R1	0x20029080	"R1"ホッパ緊急停止
SKDV_STS_DEVICE_ST_R2	0x20029100	"R2"引き抜きエラー
SKDV_STS_DEVICE_ST_R3	0x20029180	"R3"シート挿入タイムアウト
SKDV_STS_DEVICE_ST_R4	0x20029200	"R4"タイミングマークエラー
SKDV_STS_DEVICE_ST_R5	0x20029280	"R5"読み取り設定エラー
SKDV_STS_DEVICE_ST_R6	0x20029300	"R6"画像バッファオーバフロー
SKDV_STS_DEVICE_ST_R7	0x20029380	"R7"USB 接続エラー
SKDV_STS_DEVICE_ST_R8	0x20029400	"R8"シートサイズ設定エラー
SKDV_STS_DEVICE_ST_S2	0x20029900	"S2"黒レベルエラー
SKDV_STS_DEVICE_ST_S3	0x20029980	"S3"読み取りり開始検知センサ汚れエラー
SKDV_STS_DEVICE_ST_T1	0x2002A080	"T1"給紙検知部残紙
SKDV_STS_DEVICE_ST_T2	0x2002A100	"T2"読み取り開始検知部残紙
SKDV_STS_DEVICE_ST_T3	0x2002A180	"T3"本体排紙検知部残紙
SKDV_STS_DEVICE_ST_T4	0x2002A200	"T4"プリンタ印字開始検知部残紙

定数名	実際の値	内容
SKDV_STS_DEVICE_ST_T5	0x2002A280	"T5"メイン排紙部残紙
SKDV_STS_DEVICE_ST_T6	0x2002A300	"T6"セレクト排紙部残紙
SKDV_STS_DEVICE_ST_Z1	0x2002D080	"Z1"濃度補正スキューエラー
SKDV_STS_DEVICE_ST_Z2	0x2002D100	"Z2"濃度補正シートエラー
SKDV_STS_DEVICE_ST_Z3	0x2002D180	"Z3"濃度補正補正シート汚れ
SKDV_STS_DEVICE_ST_Z4	0x2002D200	"Z4"濃度補正調整不良
SKDV_STS_DEVICE_ST_Z5	0x2002D280	"Z5"読み取りり開始センサ調整エラー
SKDV_STS_DEVICE_ST_Z6	0x2002D300	"Z6"スキューセンサ調整エラー
SKDV_STS_DEVICE_ST_Z7	0x2002D380	"Z7"DFセンサ調整エラー

5.3. 構造体

5.3.1. SK_DV_MODULE_INFO

名称	モジュール情報構造体	
定義	typedef struct tag_SK_DV_MODULE_INFO { char szModel [SKDV_INFO_MODEL_LEN]; char szVersion [SKDV_INFO_VER_LEN]; }SK_DV_MODULE_INFO;	
メンバ	szModel	ライブラリーの名称文字列
	szVersion	ライブラリーのバージョン文字列
詳細	ライブラリーの情報を格納する構造体。	

5.3.2. SK_DEVICE_INFO

名称	基本デバイス情報構造体	
定義	typedef struct tag_SK_DEVICE_INFO { char szGuid [SKDV_GUID_LEN+1]; char szProduct [SKDV_PRODUCT_STR_MAX+1]; char szSerialNo [SKDV_SERIAL_LEN+1]; char szFirmVer [SKDV_FIRM_VER_STR_MAX+1]; char szFirmSum [SKDV_FIRM_SUM_STR_MAX+1]; char szHardVer [SKDV_HARD_VER_STR_MAX+1]; DWORD dwOption; }SK_DEVICE_INFO;	
メンバ	szGuid	固有の ID 文字列。
	szProduct	製品名文字列。
	szSerialNo	シリアル番号の文字列。
	szFirmVer	ファームウェアのバージョン文字列。
	szFirmSum	ファームウェアのチェックサム文字列。
	szHardVer	ハードウェアのバージョン文字列。
	dwOption	オプション情報群。ビットごとに意味を持つ。 詳細は「5.3.3.SK_DV_SR3500_INFO」を参照。
詳細	デバイスの情報を格納する構造体。 デバイス情報構造体のメンバとして使用される。 dwOption 以外のメンバは文字列で終端 NULL となる。	

5.3.3. SK_DV_SR3500_INFO

名称	デバイス情報構造体

定義

```
typedef struct   tag_SK_DV_SR3500_INFO
{
        SK_DEVICE_INFO          Main;
        SK_DEVICE_INFO          Reader[SKDV_FACE_NUM];
        SK_DEVICE_INFO          Bcr;
        SK_DEVICE_INFO          Printer;
        SK_DEVICE_INFO          Stacker;
        SK_DEVICE_INFO          Image1;
        SK_DEVICE_INFO          Image2;
}SK_DV_SR3500_INFO;
```

メンバ

Main	本体情報
Reader	読み取りユニット情報
Bcr	バーコードリーダーユニット情報（オプション）
Printer	プリンターユニット情報（オプション）
Stacker	スタッカユニット情報（オプション）
Image1	画像読み取りユニット情報１（イメージ機能搭載機のみ）
Image2	画像読み取りユニット情報２（イメージ機能搭載機のみ）

詳細

ＯＭＲの各種情報を格納する構造体。
製品やオプションの接続状態により、格納される項目が異なる。

製品ごとに格納されるメンバは以下の通り。（○：格納 / ×：無効）
オプションユニットの場合、接続されていない場合は無効。

	メンバ	SR-3500/6000/6500 SR-1800							SR-3500 HYBRID						
		szGuid	szProduct	szSerialNo	szFirmVer	szFirmSum	szHardVer	dwOption	szGuid	szProduct	szSerialNo	szFirmVer	szFirmSum	szHardVer	dwOption
1	Main	○	○	○	○	○	○	○	○	○	○	○	○	○	○
2	Reader[0]	○	×	×	○	○	×	○	○	×	×	○	○	×	○
3	Reader[1]	○	×	×	○	○	×	○	○	×	×	○	○	×	○
4	Bcr	○	×	×	○	○	×	○	×	×	×	×	×	×	×
5	Printer	○	×	×	○	○	×	○	×	×	×	×	×	×	×
6	Stacker	○	×	×	○	○	×	○	×	×	×	×	×	×	×
7	Image1	×	×	×	×	×	×	×	○	×	×	○	○	×	○
8	Image2	×	×	×	×	×	×	×	×	×	×	○	○	×	×

	メンバ	SR-11000													
		szGuid	szProduct	szSerialNo	szFirmVer	szFirmSum	szHardVer	dwOption	szGuid	szProduct	szSerialNo	szFirmVer	szFirmSum	szHardVer	dwOption
1	Main	○	○	○	○	○	○	○							
2	Reader[0]	○	×	×	○	○	×	○							
3	Reader[1]	○	×	×	○	○	×	○							
4	Bcr	×	×	×	×	×	×	×							
5	Printer	×	×	×	×	×	×	×							
6	Stacker	○	×	×	○	○	×	○							
7	Image1	×	×	×	×	×	×	×							
8	Image2	×	×	×	×	×	×	×							

各メンバの dwOption は、下記の意味を持つ。

○Main

定数名	値	意味
SKDV_OPT_UNIT_MASK	0xffff0000	ユニットビットマスク
SKDV_OPT_UNIT_READER_F	0x80000000	表面読み取りユニット有り。
SKDV_OPT_UNIT_READER_B	0x40000000	裏面読み取りユニット有り。
SKDV_OPT_UNIT_BCR	0x20000000	バーコードリーダーユニット有り。
SKDV_OPT_UNIT_PRINTER	0x10000000	プリンターユニット有り。
SKDV_OPT_UNIT_STACKER	0x08000000	スタッカユニット有り。
SKDV_OPT_UNIT_IMAGE_1	0x04000000	イメージ読み取りユニット１有り。
SKDV_OPT_UNIT_IMAGE_2	0x02000000	イメージ読み取りユニット２有り。

○Reader

定数名	値	意味
SKDV_OPT_SENSOR_TYPE_MASK	0x0000ff00	センサ種別のビットマスク
SKDV_OPT_SENSOR_TYPE_ERR	0x0000ff00	センサ種別のエラー
SKDV_OPT_SENSOR_TYPE_RED_VIS	0x00000000	可視光
SKDV_OPT_SENSOR_TYPE_INFRARED	0x00000100	近赤外光
SKDV_OPT_SENSOR_PITCH_MASK	0x000000ff	センサピッチのビットマスク
SKDV_OPT_SENSOR_PITCH_ERR	0x000000ff	センサピッチのエラー
SKDV_OPT_SENSOR_PITCH_1P6INCH	0x00000001	1/6 インチ
SKDV_OPT_SENSOR_PITCH_02INCH	0x00000002	0.2 インチ
SKDV_OPT_SENSOR_PITCH_02INCH_S	0x00000003	0.2 インチ S
SKDV_OPT_SENSOR_PITCH_025INCH	0x00000004	0.25 インチ
SKDV_OPT_SENSOR_PITCH_03INCH	0x00000005	0.3 インチ
SKDV_OPT_SENSOR_PITCH_03INCH_F	0x00000006	0.3 インチ F
SKDV_OPT_SENSOR_PITCH_6MM	0x00000007	6mm
SKDV_OPT_SENSOR_PITCH_02INCH_K	0x00000008	0.2 インチ K
SKDV_OPT_SENSOR_PITCH_02INCH_SP	0x00000009	0.2 インチ特殊
SKDV_OPT_SENSOR_PITCH_02INCH_C	0x0000000a	0.2 インチ C

○Bcr

定数名	値	意味
SKDV_OPT_BARCODE_MASK	0x000000ff	バーコードリーダーのビットマスク
SKDV_OPT_BARCODE_ERR	0x000000ff	バーコードリーダー未接続
SKDV_OPT_BARCODE_V	0x00000001	縦流し
SKDV_OPT_BARCODE_H	0x00000002	横流し

○Printer

定数名	値	意味
SKDV_OPT_PRINTER_CARTRIDGE	0x80000000	カードリッジ有り

○Image1

定数名	値	意味
SKDV_OPT_IMAGE_FACE_MASK	0xf0000000	面のビットマスク
SKDV_OPT_IMAGE_FACE_FRONT	0x80000000	表面有り
SKDV_OPT_IMAGE_FACE_BACK	0x40000000	裏面有り
SKDV_OPT_IMAGE_FACE_BOTH	0xc0000000	両面有り

5.3.4. SK_DV_SR3500_MODE

名称	デバイスモード構造体	

定義	typedef struct tag_SK_DV_SR3500_MODE
	{
	int iFeedMode;
	int iFeedTime;
	DWORD dwDisableWarning;
	struct MarkSkew{
	int iRow;
	int iLevel;
	};
	int iPanelOperation;
	struct Buzzer{
	int iVol;
	int iTone;
	};
	struct PSaving{
	int iSleep;
	int iStandby;
	};
	}SK_DV_SR3500_MODE;	

メンバ	iFeedMode	未使用。0固定。
	iFeedTime	未使用。0固定。
	dwDisableWarning	ワーニングエラー設定。 ビットごとに指定する。設定値は詳細を参照。
	MarkSkew	マークスキュー検出
	iRow	マークスキュー検出の検出欄を指定する。 設定範囲は０～１５５。０で検出は無効になる。
	iLevel	マークスキュー検出の検出レベル。設定範囲は１～１６。
	iPanelOperation	パネル操作の有効/無効設定。設定値は詳細を参照。
	Buzzer	ブザーの音量と音質を設定する構造体。
	iVol	ブザーの音量。設定範囲は０～５。０でブザーは OFF となる。
	iTone	ブザーの音質。設定範囲は１～３。
	PSaving	低消費電力モードを設定する構造体。
	iSleep	スリープタイマー。通常モードからスリープモードまでの時間。 設定範囲は０～６０［分］。０で設定は無効となる。
	iStandby	スタンバイタイマー。スリープモードからスタンバイモードまでの時間。 設定範囲は０～６０［分］。０で設定は無効となる。

詳細	デバイスの各種動作モードを格納する構造体。 動作モードの取得/設定関数「SkDv_GetMode/ SkDv_SetMode」にて使用する。 ○dwDisableWarning

定数名	値	意味
SKDV_WARN_AUTO_REJECT	0x00010000	自動排紙処理有効
SKDV_WARN_HOPPER_EMPTY	0x00020000	シートエンプティ検出有効
SKDV_WARN_TM_ERROR	0x00040000	タイミングマークエラー検出有効
SKDV_WARN_DF_ERROR	0x00080000	ダブルフィード検出有効
SKDV_WARN_LEFT_SKEW	0x00100000	左端スキュー検出有効

○iPanelOperation

定数名	値	意味
SKDV_DISABLE	0	パネルを操作できないようにする。
SKDV_ENABLE	1	パネルを操作できるようにする。

5.3.5. SK_DV_SR3500MARK_CONF

名称	マーク条件構造体	
定義	typedef struct tag_SK_DV_SR3500MARK_CONF { int iBackSideReading; int iColumns; int iReadingMethod; int iCtrlMultiple; int iThicknessType; }SK_DV_SR3500MARK_CONF;	
メンバ	iBackSideReading	裏面読み取りの有効/無効設定。設定値は詳細を参照。
	iColumns	読み取り行数。設定範囲は１～４８。 画像読み取り機能搭載機種の場合には０を設定可能。その場合にはマークの読み取りは行われない。
	iReadingMethod	読み取り方式。設定値は詳細を参照。
	iCtrlMultiple	制御倍数。読み取り方式が先端/後端制御型の場合に有効。 設定範囲は以下の通り。 先端制御型：１～９ 後端制御型：２～９
	iThicknessType	シート連量。ダブルフィードエラー検出時の判定基準に用いる、シートの連量（米坪）。設定値は詳細を参照。

詳細

マークシートのマークを読み取る条件を格納する構造体。

○iBackSideReading

定数名	値	意味
SKDV_DISABLE	0	裏面を無効にする。
SKDV_ENABLE	1	裏面を有効にする。

○iReadingMethod

定数名	値	意味
SKDV_READ_FRONT_EDGE	1	先端制御型
SKDV_READ_REAR_EDGE	2	後端制御型
SKDV_READ_DIRECT	3	直下型
SKDV_READ_FACOM	4	FACOM 型
SKDV_READ_BETWEEN_MARK_NS	5	マーク間型 （先端余白読み取り無し）
SKDV_READ_BETWEEN_MARK	6	マーク間型 （先端余白読み取り有り）

○iThicknessType

定数名	値	意味
SKDV_THICKNESS_AUTO_DETECT	0	自動検出
SKDV_THICKNESS_55_KG	1	64g/m^2 (55kg)
SKDV_THICKNESS_72_KG	2	84g/m^2 (72kg)
SKDV_THICKNESS_90_KG	3	105g/m^2 (90kg)
SKDV_THICKNESS_110_KG	4	128g/m^2 (110kg)
SKDV_THICKNESS_135_KG	5	157g/m^2 (135kg)

5.3.6. SK_DV_SR3500IMG_CONF

名称	イメージ条件構造体	
定義	typedef struct tag_SK_DV_SR3500IMG_CONF { int iScanningSide; int iScanCtrlType; int iFormLengthMin; int iFormLengthMax; int iLightColor[SKDV_FACE_NUM]; int iResoType; int iResoTypeY; }SK_DV_SR3500IMG_CONF;	
メンバ	iScanningSide	読み取り面の指定。設定値は詳細を参照。
	iScanCtrlType	未使用。
	iFormLengthMin	未使用。
	iFormLengthMax	未使用。
	iLightColor[0]	表面光源の指定。設定値は詳細を参照。
	iLightColor[1]	裏面光源の指定。設定値は表面光源と同じ。
	iResoType	解像度の指定。設定値は詳細を参照。 表裏個別に設定することはできない。
	iResoTypeY	未使用。

詳細

イメージ条件を格納する構造体。

○iScanningSide

定数名	値	意味
SKDV_SIDE_NONE	0	画像読み取りしない
SKDV_SIDE_BOTH	1	両面
SKDV_SIDE_FRONT	2	表面のみ
SKDV_SIDE_BACK	3	裏面のみ

○iLightColor

光源は、表裏で３色と単色を個別に設定できない。片面をカラーにするなら両面をカラーに設定する必要がある。

定数名	値	意味
SKDV_LIGHT_COLOR	0	カラー（３色）
SKDV_LIGHT_GRAYSCALE	1	グレースケール（単色）
SKDV_LIGHT_RED	2	赤色（単色）
SKDV_LIGHT_GREEN	3	緑色（単色）
SKDV_LIGHT_BLUE	4	青色（単色）

○iResoType

定数名	値	意味
SKDV_RESO_TYPE_300DPI	1	300dpi
SKDV_RESO_TYPE_200DPI	2	200dpi
SKDV_RESO_TYPE_150DPI	3	150dpi
SKDV_RESO_TYPE_100DPI	4	100dpi

5.3.7. SK_DV_SR3500_STATUS_CHAR

名称	デバイスステータスコード文字構造体	
定義	typedef struct tag_SK_DV_SR3500_STATUS_CHAR { char ST1[SKDV_ST_LEN]; char ST2[SKDV_ST_LEN]; char DIF1; char DIF2; }SK_DV_SR3500_STATUS_CHAR;	
メンバ	ST1	表面ステータス情報コード
	ST2	裏面ステータス情報コード
	DIF1	表面エラー詳細情報コード
	DIF2	裏面エラー詳細情報コード
詳細	デバイスのステータス情報コードを格納する構造体。 コードの詳細は、別紙「コマンドリファレンス」を参照。	

5.3.8. SK_DV_MARK_INFO

名称	マークデータ情報構造体	
定義	typedef struct tag_SK_DV_MARK_INFO { int iType; int iRows; int iColumns; }SK_DV_MARK_INFO;	
メンバ	iType	未使用。
	iRows	読み取ったマークの欄数。
	iColumns	読み取ったマークの行数。
詳細	読み取ったマークデータの情報を格納する構造体。	

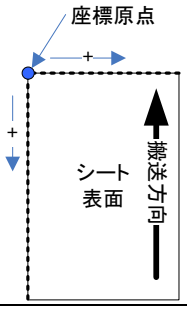
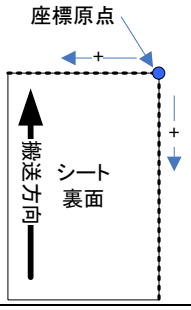
5.3.9. SK_DV_IMAGE_DATA_CONF

<table>
<tr><td>名称</td><td colspan="2">イメージデータ条件構造体</td></tr>
<tr><td>定義</td><td colspan="2">typedef struct tag_SK_DV_IMAGE_DATA_CONF
{
　　int　　iFace;
　　RECT　　rectCutout;
　　int　　iColorType;
　　int　　iRotate;
　　int　　iResolution;
　　DWORD　　dwOption;
}SK_DV_IMAGE_DATA_CONF;</td></tr>
<tr><td rowspan="6">メンバ</td><td>iFace</td><td>表裏の指定。設定値は詳細を参照。</td></tr>
<tr><td>rectCutout</td><td>切り出し位置の指定。0.1mm 単位で指定する。
座標系は詳細を参照。
RECT は MFC ライブラリリファレンスを参照。
typedef struct tagRECT {
　LONG left;
　LONG top;
　LONG right;
　LONG bottom;
} RECT;</td></tr>
<tr><td>iColorType</td><td>白黒２値、グレースケール（8bit）、カラー（24bit）から選択できる。
設定値は詳細を参照。</td></tr>
<tr><td>iRotate</td><td>画像の回転。９０度単位で選択できる。設定値は詳細を参照。</td></tr>
<tr><td>iResolution</td><td>解像度。設定範囲は５０～３００［dpi］。
通常は読み取り時の解像度を指定すること。
読み取り時の解像度より高く設定した場合、画素を補間する為に画像が劣化する。</td></tr>
<tr><td>dwOption</td><td>未使用。</td></tr>
</table>

詳細

イメージデータの取得条件を格納する構造体。

○rectCutout

表面(iFace=0)	裏面(iFace=1)
	

○iFace

定数名	値	意味
SKDV_FRONT	0	表面
SKDV_BACK	1	裏面

○iColorType

定数名	値	意味
SKIM_IMAGE_WB	0	白黒２値
SKIM_IMAGE_GRAY	1	８ビットグレースケール
SKIM_IMAGE_COLOR	2	２４ビットカラー

○iRotate

定数名	値	意味
SKIM_IMAGE_ROTATE_0	0	回転なし
SKIM_IMAGE_ROTATE_90	1	時計回りに９０度回転
SKIM_IMAGE_ROTATE_180	2	時計回りに１８０度回転
SKIM_IMAGE_ROTATE_270	3	時計回りに２７０度回転

5.3.10. SK_DV_IMAGE_FILE_CONF

名称	イメージファイル条件構造体	
定義	typedef struct tag_SK_DV_IMAGE_FILE_CONF { int iEncoder; int iParam; }SK_DV_IMAGE_FILE_CONF;	
メンバ	iEncoder	ファイル形式。設定値は詳細を参照。
	iParam	ファイル形式のパラメータ。ファイル形式により指定することが可能。 使用しないファイル形式の場合には、０を指定すること。 Jpeg 形式 ：画質の指定。設定範囲は１～１００。

詳細

スキャンしたイメージデータをファイルに保存する際のファイル形式を指定するための構造体。

○iEncoder

定数名	値	意味
SKIM_ENCODE_BMP	0	ビットマップ形式
SKIM_ENCODE_GIF	1	GIF 形式
SKIM_ENCODE_JPEG	2	Jpeg 形式
SKIM_ENCODE_PNG	3	PNG 形式
SKIM_ENCODE_TIFF	4	Tiff 形式

○iParam （iEncoder：SKIM_ENCODE_JPEG 指定時）
１～１００の範囲で指定。１が低画質、１００が高画質

○iParam （iEncoder：SKIM_ENCODE_TIFF 指定時）

定数名	値	意味
SKIM_ENCODE_TIFF_PRM_LZW	2	LZW 圧縮
SKIM_ENCODE_TIFF_PRM_NONE	6	圧縮なし

5.3.11. SK_LAYOUT_ID_PRM

名称	レイアウト ID 設定構造体	
定義	typedef struct tag_SK_LAYOUT_ID_PRM { UCHAR ucIdData[SKDV_LAYOUT_ID_PTN_LEN]; int iTmCount[SKDV_FACE_NUM]; int iBarcodeCount; }SK_LAYOUT_ID_PRM;	
メンバ	ucIdData	ID ウィンドウのマークパターンの指定
	iTmCount	表面と裏面のタイミングマーク数の指定。
	iBarcodeCount	未使用。0 固定。
詳細	ID 管理データ設定にて使用レイアウト ID の設定内容を格納する構造体。	

5.3.12. SK_WINDOW_PRM

名称	ウィンドウ設定構造体	
定義	typedef struct tag_SK_WINDOW_PRM { int iFace; struct { int iStart; int iNumber; int iStep; }Col; struct { int iStart; int iNumber; int iStep; }Row; int iDirection; int iPartition; struct { int iSensitivity; int iDifference; }Level; }SK_WINDOW_PRM;	
メンバ	iFace	表裏の指定。
	Col	行
	iStart	開始位置
	iNumber	ウィンドウの数
	iStep	ステップ数
	Row	列
	iStart	開始位置
	iNumber	ウィンドウの数
	iStep	ステップ数
	iDirection	方向
	iPartition	パーティション分割 SkDv_ReqSetLayoutManage 関数内ではこの値は使用しない。 （設定された場合は無視する。）
	Level	濃度

	iSensitivity	検出レベル
	iDifference	レベル差

詳細

ウィンドウ設定を指定構造体。

○iDirection

定数名	値	意味
SKDV_WIN_DIR_TL_DOWN	0	左上から下へ
SKDV_WIN_DIR_TR_DOWN	1	右上から下へ
SKDV_WIN_DIR_BL_UP	2	左下から上へ
SKDV_WIN_DIR_BR_UP	3	右下から上へ
SKDV_WIN_DIR_TL_RIGHT	4	左上から右へ
SKDV_WIN_DIR_TR_LEFT	5	右上から左へ
SKDV_WIN_DIR_BL_RIGHT	6	左下から右へ
SKDV_WIN_DIR_BR_LEFT	7	右下から左へ

○iPartition

定数名	値	意味
SKDV_WIN_PART_OFF	0	なし
SKDV_WIN_PART_ON_START0	1	パーティション分割あり (0 から開始)
SKDV_WIN_PART_ON_START1	2	パーティション分割あり (1から開始) 範囲チェックなどの演算処理が行われる場合に、先頭を 0 から開始か、1 から開始か指定できる。

5.3.13. SK_WINDOW_CHECK

名称	ウィンドウチェック構造体	
定義	typedef struct tag_SK_WINDOW_CHECK { int iOption; struct { int iMin; int iMax; }MarkCount; }SK_WINDOW_CHECK;	
メンバ	iOption	ノーマーク許可
	MarkCount	マーク数チェック
	iMin	マーク数最小
	iMax	マーク数最大
詳細	ウィンドウのマーク数をチェック構造体。	

5.3.14. SK_LAYOUT_OPTION

名称	レイアウトオプション構造体

定義

```
typedef struct   tag_SK_LAYOUT_OPTION
{
        int             iType;
        union
        {
                UCHAR   morimori
[SKDV_LAYOUT_OPT_LEN];
                struct
                {
                        DWORD           dwMin;
                        DWORD           dwMax;
                }Num;
        };
}SK_LAYOUT_OPTION;
```

メンバ

iType	オプション機能
ucData	オプション機能のパラメータ
Num	範囲
dwMin	最小
dwMax	最大

詳細

レイアウトのオプションを指定する構造体。

○iType
ID 設定時は SKDV_LAYOUT_OPT_NONE/ SKDV_LAYOUT_OPT_ORDER_ID
ウィンドウ設定時は SKDV_LAYOUT_OPT_NONE/ SKDV_LAYOUT_OPT_MASK/
SKDV_LAYOUT_OPT_FIXED_COMP/ SKDV_LAYOUT_OPT_CHECK_DIGIT/
SKDV_LAYOUT_OPT_ASCENDING_ORDER/ SKDV_LAYOUT_OPT_RANGE_CHECK/
SKDV_LAYOUT_OPT_MASK_PART が設定可能。

定数名	値	意味
SKDV_LAYOUT_OPT_NONE	0x00	無効(指定なし)
SKDV_LAYOUT_OPT_ORDER_ID	0x01	ID 順チェック(ID 設定時)
SKDV_LAYOUT_OPT_MASK	0x01	マークをマスク(ウィンドウ設定時)
SKDV_LAYOUT_OPT_FIXED_COMP	0x02	固定マーク
SKDV_LAYOUT_OPT_CHECK_DIGIT	0x03	チェックディジット
SKDV_LAYOUT_OPT_ASCENDING_ORDER	0x04	範囲チェック(昇順)
SKDV_LAYOUT_OPT_DESCENDING_ORDER	0x05	範囲チェック(降順)
SKDV_LAYOUT_OPT_RANGE_CHECK	0x06	範囲チェック(順番なし)
SKDV_LAYOUT_OPT_MASK_PART	0x07	マスク設定(パーティション共通)

○ucData

定数名	値	意味
SKDV_LAYOUT_OPT_LEN	16	オプションデータ長 [byte]

5.3.15. SK_LAYOUT_MANAGE_CONF

名称	ID データ管理構造体	
定義	typedef struct tag_SK_LAYOUT_MANAGE_CONF { BOOL blEnableId; SK_WINDOW_PRM IdWindowPrm; DWORD dwNgAction; SK_LAYOUT_OPTION LayoutOpt; }SK_LAYOUT_MANAGE_CONF;	
メンバ	blEnableId	レイアウト ID 切替
	IdWindowPrm	ID ウィンドウエリア
	dwNgAction	判定 NG 時の動作
	LayoutOpt	オプション指定
詳細	ID データ管理情報を格納する構造体。 ○LayoutOpt.iType	

定数名	値	意味
SKDV_LAYOUT_OPT_NONE	0	無効（指定なし）
SKDV_LAYOUT_OPT_ORDER_ID	1	ID 順チェック

5.3.16. SK_LAYOUT_ID_CONF

名称	ID データ構造体	
定義	typedef struct tag_SK_LAYOUT_ID_CONF { SK_LAYOUT_ID_PRM LayoutIdPrm; int iReject; SK_LAYOUT_OPTION LayoutOpt; }SK_LAYOUT_ID_CONF;	
メンバ	LayoutIdPrm	ID 設定
	iReject	強制リジェクト
	LayoutOpt	オプション指定
詳細	ID データを格納する構造体。	

5.3.17. SK_WINDOW_AREA_CONF

名称	ウィンドウ領域構造体	
定義	typedef struct tag_SK_WINDOW_AREA_CONF { SK_WINDOW_PRM WindowPrm; SK_WINDOW_CHECK WindowCheck; DWORD dwNgAction; SK_LAYOUT_OPTION LayoutOpt; }SK_WINDOW_AREA_CONF;	
メンバ	WindowPrm	ウィンドウエリア
	WindowCheck	ウィンドウチェック
	dwNgAction	判定 NG 時の動作
	LayoutOpt	オプション指定
詳細	ウィンドウ領域情報を格納構造体	

5.3.18. SK_CONT_FEED_PRN_OPT

名称	連続読み取り印字オプション構造体	
定義	typedef struct tag_SK_CONT_FEED_PRN_OPT { int iDigits; int iStartNumber; }SK_CONT_FEED_PRN_OPT;	
メンバ	iDigits	連番桁数指定
	iStartNumber	連番初期値
詳細	連続読み取り印字オプションを格納構造体	

5.3.19. SK_DV_WINDOW_RESULT

<table>
<tr><td>名称</td><td colspan="2">ウィンドウ読み取り結果結果構造体</td></tr>
<tr><td>定義</td><td colspan="2">typedef struct tag_SK_DV_WINDOW_RESULT
{
　　int　　iResult;
　　int　　iId;
　　int　　iCount;
}SK_DV_WINDOW_RESULT, *PSK_DV_WINDOW_RESULT;</td></tr>
<tr><td rowspan="3">メンバ</td><td>iResult</td><td>読み取り結果</td></tr>
<tr><td>iId</td><td>ID 認識結果</td></tr>
<tr><td>iCount</td><td>ウィンドウ数</td></tr>
<tr><td>詳細</td><td colspan="2">ウィンドウの読み取り結果情報を格納する構造体。

〇iId
<table>
<tr><th>定数名</th><th>値</th><th>意味</th></tr>
<tr><td>SKDV_RESULT_ID_MIN</td><td>0</td><td>レイアウト ID の最小値</td></tr>
<tr><td>SKDV_RESULT_ID_MAX</td><td>15</td><td>レイアウト ID の最大値</td></tr>
<tr><td>SKDV_RESULT_ID_NUM</td><td>16</td><td>レイアウト ID 数</td></tr>
<tr><td>SKDV_RESULT_ID_NONE</td><td>0</td><td>ID 設定なし時の ID</td></tr>
<tr><td>SKDV_RESULT_ID_UNKNOWN</td><td>99</td><td>ID 不一致</td></tr>
</table></td></tr>
</table>

5.3.20. SK_DV_BARCODE_RESULT

<table>
<tr><td>名称</td><td colspan="2">バーコード読み取り結果構造体</td></tr>
<tr><td>定義</td><td colspan="2">typedef struct tag_SK_DV_BARCODE_RESULT
{
　　int　　iResult;
　　int　　iCount;
}SK_DV_BARCODE_RESULT, *PSK_DV_BARCODE_RESULT;</td></tr>
<tr><td rowspan="2">メンバ</td><td>iResult</td><td>読み取り結果</td></tr>
<tr><td>iCount</td><td>バーコード数</td></tr>
<tr><td>詳細</td><td colspan="2">バーコードの読み取り結果を格納する構造体。</td></tr>
</table>

5.3.21. SK_SHEET_RESULT

名称	シート読み取り結果構造体	
定義	typedef struct tag_SK_SHEET_RESULT { SkDvStatus DvStatus; int iSheetCount; int iTmCount[SKDV_FACE_NUM]; int iBarcodeCount; int iId; int iPrinted; int iStackedId; DWORD dwResult; int iFinish; int iIdResult; struct { int iCount; int iIndex; int iPartIndex; int iReason; }NgWindow; }SK_SHEET_RESULT;	
メンバ	DvStatus	読み取り時のステータス
	iSheetCount	帳票カウント
	iTmCount	タイミングマーク数
	iBarcodeCount	バーコード数
	iId	ID 認識結果
	iPrinted	印字結果
	iStackedId	排紙先
	dwResult	搬送結果通知データ
	iFinish	終了通知データ
	iIdResult	ID 認識結果
	NgWindow	NG ウィンドウ詳細
	iCount	NG ウィンドウの総数
	iIndex	最初の NG ウィンドウ番号(0～)
	iPartIndex	最初の NG パーティション番号(0～)
	iReason	最初の NG 理由

詳細

シート読み取り結果を格納する構造体。

○iId

定数名	値	意味
SKDV_RESULT_ID_MIN	0	レイアウト ID の最小値
SKDV_RESULT_ID_MAX	15	レイアウト ID の最大値
SKDV_RESULT_ID_NUM	16	レイアウト ID 数
SKDV_RESULT_ID_NONE	0	ID 設定なし時の ID
SKDV_RESULT_ID_UNKNOWN	99	ID 不一致

○iPrinted

定数名	値	意味
SKDV_RESULT_PRN_NONE	0	印字なし/未印字
SKDV_RESULT_PRN_PRINTED	1	印字済み

○iStackedId

定数名	値	意味
SKDV_RESULT_STK_NONE	0	未排紙
SKDV_RESULT_STK_MAIN	1	メイン排紙
SKDV_RESULT_STK_SELECTION	2	セレクト排紙
SKDV_RESULT_STK_CNT	3	種別数

○dwResult

定数名	値	意味
SKDV_RESULT_STS_OK	0x00000000	正常
SKDV_RESULT_STS_ERROR_EJECT	0x00000004	エラー排紙
SKDV_RESULT_STS_ERROR_STOP	0x00000002	エラー停止
SKDV_RESULT_STS_WINDOW_NG	0x00000001	ウィンドウ判定 NG

○NgWidow.iReason

定数名	値	意味
SKDV_RESULT_REASON_OK	0	正常
SKDV_RESULT_REASON_MARK_COUNT_LOWER	1	マーク数が少ない
SKDV_RESULT_REASON_MARK_COUNT_UPPER	2	マーク数が多い
SKDV_RESULT_REASON_MARK_DISAGREE	3	固定データ不一致
SKDV_RESULT_REASON_CHECK_DIGIT_NG	4	チェックディジット NG
SKDV_RESULT_REASON_ASCENDING_ORDER_NG	5	範囲チェック(昇順)
SKDV_RESULT_REASON_DESCENDING_ORDER_NG	6	範囲チェック(降順)
SKDV_RESULT_REASON_RANGE_CHECK_NG	7	範囲チェック(方向なし)

○iFinish

定数名	値	意味
SKDV_RESULT_FINISH_CONT	0	継続
SKDV_RESULT_FINISH_DONE	1	終了

○iIdResult

定数名	値	意味
SKDV_RESULT_ID_OK	0	正常
SKDV_RESULT_ID_TM_CNT_DIFF_F	1	表面 TM 数不一致
SKDV_RESULT_ID_TM_CNT_DIFF_B	2	裏面 TM 数不一致
SKDV_RESULT_ID_TM_CNT_DIFF_FB	3	両面 TM 数不一致
SKDV_RESULT_ID_ID_NONE	4	レイアウト ID 全不一致
SKDV_RESULT_ID_BC_CNT_DIFF	5	BC 数不一致
SKDV_RESULT_ID_ID_ORDER_NG	6	レイアウト ID 順 NG

5.3.22. SK_CONT_FEED_RESULT

名称	連続読み取り結果構造体	
定義	typedef struct tag_SK_CONT_FEED_RESULT { SkDvStatus DvStatus; int iResult; struct { int iRead int iPrint; int iEject; }Remain; }SK_CONT_FEED_RESULT;	
メンバ	DvStatus	連続読み取りの最終的なステータス
	iResult	連続処理終了通知データ
	Remain	未処理枚数
	iRead	読み取り未処理枚数
	iPrint	印字未処理枚数
	iEject	排紙未処理枚数
詳細	連続読み取りの結果を格納する構造体。 ○iResult	

定数名	値	意味
SKDV_CFR_NORMAL	0	正常終了
SKDV_CFR_ERROR	1	異常終了
SKDV_CFR_WINDOW_NG	2	ウィンドウ判定による終了
SKDV_CFR_CANCEL	3	キャンセル終了

5.3.23. SK_CONT_FEED_COUNT

名称	連続読み取り枚数構造体	
定義	typedef struct tag_SK_CONT_FEED_COUNT { int iFeed; int iNormal; int iPrint; int iEject[SKDV_RESULT_STK_CNT]; int iUntreated; }SK_CONT_FEED_COUNT;	
メンバ	iFeed	給紙実施枚数
	iNormal	正常読み取り枚数
	iPrint	印字読み取り枚数
	iEject	排紙枚数
	iUntreated	未処理排紙枚数
詳細	連続読み取り枚数を格納する構造体。	

5.3.24. SK_DV_OPT_PRN_CONF

名称	印字設定構造体	
定義	typedef struct tag_SK_DV_OPT_PRN_CONF { int iEnable; int iStartPos; int iOrientation; int iFontSize; int iFontSpace; int iPrintMode; }SK_DV_OPT_PRN_CONF;	
メンバ	iEnable	プリンタの有効無効
	iStartPos	帳票先端からの印字位置 [mm]
	iOrientation	印字方向
	iFontSize	印字文字サイズ [0.1mm]
	iFontSpace	印字文字間隔 [0.1mm]
	iPrintMode	印字方法

詳細

印字設定を格納する構造体。

○iEnable

定数名	値	意味
SKDV_DISABLE	0	印字を無効にする。
SKDV_ENABLE	1	印字を有効にする。

○iOrientation

定数名	値	意味
SKDV_PRINT_ORIENT_0	1	通常印字
SKDV_PRINT_ORIENT_180	2	文字列を 180° 回転

○iPrintMode

定数名	値	意味
SKDV_PRINT_MODE_AFTER_FEED	1	読み取らせた後に印字する
SKDV_PRINT_MODE_FEED_AND_PRINT	2	読み取りと印字を同時に行う

5.3.25. SK_DV_OPT_BCR_CONF

名称	バーコード読み取り条件構造体	

定義	typedef struct tag_SK_DV_OPT_BCR_CONF
	{
	int iEnable;
	int iReadingArea[SKDV_BCR_READING_AREA_MAX];
	DWORD dwEnableBcType;
	
	struct tagCheckDigit{
	int iCode39;
	int iItf;
	int iNw7;
	int iReserved1;
	int iReserved2;
	int iReserved3;
	}CheckDigit;
	
	struct tagOption{
	int iUpcA;
	int iUpcE;
	int iReserved1;
	int iReserved2;
	int iReserved3;
	int iReserved4;
	}Option;
	}SK_DV_OPT_BCR_CONF;	

メンバ	iEnable	バーコード読み取りの有効/無効設定。
	iReadingArea	バーコードの読み取り範囲。 読み取らない範囲と読み取る範囲を１組として１０箇所の指定が可能。
	dwEnableBcType	読み取るバーコードの種類を設定。
	CheckDigit	チェックディジット
	iCode39	CODE39 のチェックディジットの指定
	iItf	ITF のチェックディジットの指定
	iNw7	NW7 のチェックディジットの指定
	iReserved1-3	（予備）
	Option	オプション
	iUpcA	UPC-A 出力桁数
	iUpcE	UPC-E システムコード付加
	iReserved1-4	（予備）

詳細

バーコード読み取り条件を格納する構造体。

○iEnable

定数名	値	意味
SKDV_DISABLE	0	読み取りを無効にする。
SKDV_ENABLE	1	読み取りを有効にする。

○iReadingArea

・配列の１番目に SKDV_BCR_READING_AREA_ENTIRE を指定すれば、全域を読み取る。
・配列の１番目に mm 単位で読み取らない範囲、２番目に SKDV_BCR_READING_AREA_ENTIRE を指定すると以降の領域を読み取り範囲とする。
・読み取らない範囲と読み取る範囲を１組として指定し、読み取り不要な組に０を指定することで、複数の読み取り範囲を指定することが出来る。

定数名	値	意味
SKDV_BCR_READING_AREA_ENTIRE	0	全域を読み取る。

○CheckDigit

バーコードの種類によってチェックディジットの種類を指定することが出来る。

・iCode39 / iItf

定数名	値	意味
SKDV_BCR_CD_NONE	0	検査しない
SKDV_BCR_CD_ENABLE	1	検査する

・iNw7

定数名	値	意味
SKDV_BCR_CD_NONE	0	検査しない
SKDV_BCR_CD_NW7_M16	1	モジュラス 16
SKDV_BCR_CD_NW7_M11	2	モジュラス 11
SKDV_BCR_CD_NW7_M10W2	3	モジュラス 10／2 ウェイト
SKDV_BCR_CD_NW7_M10W3	4	モジュラス 10／3 ウェイト
SKDV_BCR_CD_NW7_7DR	5	7 チェック DR
SKDV_BCR_CD_NW7_WM11	6	加重モジュラス 11
SKDV_BCR_CD_NW7_RUNES	7	ルーンズ

○Option

バーコードの種類によってオプション設定を指定することが出来る。

・iUpcA：UPC-A 出力桁数

定数名	値	意味
SKDV_BCR_CD_UPC_A_DIGIT_12	0	１２桁で出力する
SKDV_BCR_CD_UPC_A_DIGIT_13	1	１３桁で出力する

・iUpcE：UPC-E システムコード付加

定数名	値	意味
SKDV_BCR_CD_UPC_E_NO_CODE	0	システムコードを付加しない
SKDV_BCR_CD_UPC_E_ADD_CODE	1	システムコードを付加する

5.4. API 関数 ー 基本機能

5.4.1. SkDv_GetModuleInfo

処理	本ライブラリーの名称とバージョンを取得する。	
プロトタイプ	void SkDv_GetModuleInfo(SK_DV_MODULE_INFO* pModuleInfo)	
引数	pModuleInfo	情報を格納する為の構造体のアドレスを指定する。 構造体の内容は「5.3.1.SK_DV_MODULE_INFO」を参照。
戻り値	なし	
詳細	ライブラリーがロードさせていれば、事前に開始処理など行う必要はない。 実行後、引数 pModuleInfo に本モジュールの名称とバージョンが格納される。	

5.4.2. SkDv_OpenSingle

処理	USB に接続された OMR デバイスを検索し、そのデバイスと通信が可能な状態にする。	
プロトタイプ	SkDvStatus SkDv_OpenSingle(SkDvHandle *phSkDevice)	
引数	phSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗(オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数の実行により、作業用のメモリなどの確保とデバイスとの通信の事前準備が行われ、各種処理が実行できる状態となる。処理開始前に必ず実行する必要がある。 複数の OMR デバイスが接続されている場合、内部での管理で先頭の OMR デバイスと接続される。 正常に処理した場合には phSkDevice にデバイス制御ハンドルが格納され、処理に失敗した場合には phSkDevice に NULL(0)が格納される。 このハンドルを使用して本ミドルウェアの各種処理を行う。 ※:本関数を使用した場合、旧 API ライブラリーは使用することはできない。 SkDv_OpenSingle は1台接続のみ対応し、既に開始処理を行っているデバイスがある場合エラーを返す。	

5.4.3. SkDv_OpenWithOmrapi

処理	USB に接続された OMR デバイスを検索し、そのデバイスと通信が可能な状態にする。 旧 API ライブラリ(Omrapi.dll)からの制御が可能になる。	
プロトタイプ	SkDvStatus　SkDv_OpenWithOmrapi(SkDvHandle *phSkDevice)	
引数	phSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定する
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗(オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数内で旧 API ライブラリ（Omrapi.dll）の「OMR_OpenDeviceUSB」を実行するので、旧 API 関数を使用することができるようになる。よって「OMR_OpenDeviceUSB」は使用してはならない。それ以外は SkDv_OpenSingle と同じ。 なお、本関数を使用した場合には、旧 API ライブラリ（Omrapi.dll）のと「OMR_CloseDevice」を実行してはならない。	

5.4.4. SkDv_Close

処理	すべての処理を終了する際に実行し、各種作業用メモリの開放を行う。	
プロトタイプ	SkDvStatus　SkDv_Close(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗(オープンしているデバイスがない、又は既に切断されている。)
詳細	開始処理で受け取った制御ハンドルを使用すること。 同じデバイスを開きなおすと制御ハンドルは変わることに注意すること。	

5.4.5. SkDv_ReqInit

処理	デバイスに対しソフトウェアリセットを行い、各種動作条件を取得する。	
プロトタイプ	SkDvStatus SkDv_ReqInit(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	画像データや内部処理用のデータなどが初期化され、デバイスに対しソフトリセットも実行される。 また、デバイスからデバイス情報や各動作条件を取得する。	

5.4.6. SkDv_GetInfo

処理	デバイス制御のバージョン情報や所持する機能などの情報を取得する。	
プロトタイプ	SkDvStatus SkDv_GetInfo(SkDvHandle hSkDevice, SK_DV_SR3500_INFO* pDvInfo)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pDvInfo	デバイス情報を格納する為の構造体のアドレスを指定する。 構造体の内容は「5.3.3.SK_DV_SR3500_INFO」を参照。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	デバイス情報の取得が行われていなければ、デバイスと通信を行い、引数 pDvInfo にそのデバイス情報を格納する。 事前にデバイスとの通信を行いたい場合には、SkDv_ReqInit 関数または SkDv_ReqGetInfo 関数を呼び出すこと。	

5.4.7. SkDv_GetMode

処理	ライブラリー内に格納されている動作モードを取得する。	
プロトタイプ	SkDvStatus SkDv_GetMode(SkDvHandle hSkDevice, SK_DV_SR3500_MODE* pDvMode)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pDvMode	動作モードを格納する為の構造体のアドレスを指定する。 構造体の内容は「5.3.4.SK_DV_SR3500_MODE」を参照。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	取得する動作モードは本ライブラリー内で保持される内容。 本ライブラリーが保持する動作モードを引数 pDvMode に格納する。 デバイスとの通信は行わない。 起動直後などデバイスに設定されている内容を取得したい場合には、事前に SkDv_ReqInit 関数または SkDv_ReqGetMode 関数を呼び出すこと。	

5.4.8. SkDv_SetMode

処理	動作モードをライブラリー内に設定する。	
プロトタイプ	SkDvStatus SkDv_SetMode(SkDvHandle hSkDevice, SK_DV_SR3500_MODE* pDvMode)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pDvMode	動作モードを格納してある構造体のアドレスを指定する。 構造体の内容は「5.3.4.SK_DV_SR3500_MODE」を参照。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	デバイス情報の取得が行われていなければ、デバイスと通信を行い、そのデバイス情報をライブラリー内部に格納する。 引数 pDvMode で指定された動作モードをデバイス情報の内容に合わせチェックし、値に問題があればエラーを返す。 エラーが無ければ、ライブラリー内部に動作モードを格納する。 動作モードの設定内容は「読み取り要求」または SkDv_ReqSetMode 関数を実行することにより通信にてデバイスに設定する事ができる。	

5.4.9. SkDv_GetMarkConf

処理	ライブラリー内に格納されているマーク読み取りの動作条件を取得する。	
プロトタイプ	SkDvStatus SkDv_GetMarkConf(SkDvHandle hSkDevice, SK_DV_SR3500MARK_CONF* pDvMarkConf)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pDvMarkConf	マーク読み取りの動作条件を格納する為の構造体のアドレスを指定する。 構造体の内容は「5.3.5.SK_DV_SR3500MARK_CONF」を参照。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	取得するマーク読み取りの動作条件は本ライブラリー内で保持される内容。 本ライブラリーが保持するマーク読み取りの動作条件を引数 pDvMarkConf に格納する。 デバイスとの通信は行わない。 起動直後などデバイスに設定されている内容を取得したい場合には、事前に SkDv_ReqInit 関数または SkDv_ReqGetMarkConf 関数を呼び出すこと。	

5.4.10. SkDv_SetMarkConf

処理	マーク読み取りの動作条件をライブラリー内に設定する。	
プロトタイプ	SkDvStatus SkDv_SetMarkConf(SkDvHandle hSkDevice, SK_DV_SR3500MARK_CONF* pDvMarkConf)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pDvMarkConf	マーク読み取りの動作条件を格納してある構造体のアドレスを指定する。 構造体の内容は「5.3.5.SK_DV_SR3500MARK_CONF」を参照。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	デバイス情報の取得が行われていなければ、デバイスと通信を行い、そのデバイス情報をライブラリー内部に格納する。 引数 pDvMarkConf で指定されたマーク読み取りの動作条件をデバイス情報の内容に合わせチェックし、値に問題があればエラーを返す。 エラーが無ければ、ライブラリー内部にマーク読み取りの動作条件を格納する。 マーク読み取りの動作条件の設定内容は「読み取り要求」または SkDv_ReqSetMarkConf 関数を実行することにより通信にてデバイスに設定する事ができる。	

5.4.11. SkDv_ReqGetStatus

処理	通信を行いデバイスのステータス情報を取得する。	
プロトタイプ	SkDvStatus SkDv_ReqGetStatus(SkDvHandle hSkDevice, SK_DV_SR3500_STATUS_CHAR* pStatusChar)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pStatusChar	デバイスのステータス情報コードを格納する構造体のアドレスを指定する。 構造体の内容は「5.3.7.SK_DV_SR3500_STATUS_CHAR」を参照。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	エラー詳細情報要求コマンド「DE」を使用し、ステータス情報とエラー詳細情報を取得する。 デバイスのエラーは戻り値でも返す。	

5.4.12. SkDv_ReqGetSensor

処理	通信を行いデバイスのセンサ状態を取得する。	
プロトタイプ	SkDvStatus SkDv_ReqGetSensor(SkDvHandle hSkDevice, DWORD* pdwSensor)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pdwSensor	センサの状態を格納する変数のアドレスを指定する。 値は詳細を参照。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗

詳細

センサ情報要求コマンド「DS」を使用し、センサ状態を取得する。
引数 pdwSensor にビットに割り当てたセンサの状態を格納する。

○pdwSensor

定数名	値	意味
SKDV_SENSOR_BIT_OUTPS	0x00000020	OUTPS
SKDV_SENSOR_BIT_RDPS	0x00000010	RDPS
SKDV_SENSOR_BIT_INPS	0x00000008	INPS
SKDV_SENSOR_BIT_PS0	0x00000004	PS0
SKDV_SENSOR_BIT_UPPS	0x00000002	UPPS
SKDV_SENSOR_BIT_DWPS	0x00000001	DWPS
SKDV_SENSOR_BIT_SKS	0x00001000	SKS
SKDV_SENSOR_BIT_MAIN_CVR	0x00000100	MAIN-CVR
SKDV_SENSOR_BIT_SPS3	0x00200000	SPS3
SKDV_SENSOR_BIT_SPS2	0x00100000	SPS2
SKDV_SENSOR_BIT_SPS1	0x00080000	SPS1
SKDV_SENSOR_BIT_SPS	0x00040000	SPS
SKDV_SENSOR_BIT_MPS	0x00020000	MPS
SKDV_SENSOR_BIT_P2PS	0x00010000	P2PS
SKDV_SENSOR_BIT_SRPS	0x10000000	SRPS
SKDV_SENSOR_BIT_MRPS	0x08000000	MRPS
SKDV_SENSOR_BIT_SPS5	0x04000000	SPS5
SKDV_SENSOR_BIT_SPS4	0x02000000	SPS4
SKDV_SENSOR_BIT_STK_CVR1	0x01000000	STK-CVR1

5.4.13. SkDv_ReqFeedMarkSheet

処理	マークシートの読み取りを行う。	
プロトタイプ	SkDvStatus SkDv_ReqFeedMarkSheet(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	シート読み取りコマンド「SF」により、マークシートの読み取りを行う。 イメージデータの読み取りは行わない。 情報取得/条件設定の通信が行われていない場合には、読み取り前に通信が行われる。	

5.4.14. SkDv_ReqEjectForm

処理	デバイス内の帳票を排紙する。	
プロトタイプ	SkDvStatus SkDv_ReqEjectForm(SkDvHandle hSkDevice, int iDirection)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	iDirection	排紙動作を指定する。設定値は詳細を参照。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	排紙動作コマンド「ER」により、読み取ったシートの排紙動作を行う。 スタッカ(オプション)が接続されていない場合、メインスタッカとセレクトスタッカの切り替えは行われない。 SR-3500 HYBRID にてイメージ読み取りを行った場合には排紙も行われるため、本関数を実行しても排紙されない。	

○iDirection

定数名	値	意味
SKDV_EJECT_MAIN	1	停止位置にあるシートをメインスタッカに排紙 (即時処理)
SKDV_EJECT_SELECT	2	停止位置にあるシートをセレクトスタッカに排紙 (即時処理)
SKDV_EJECT_MAIN_NEXT	3	停止位置にあるシートを、次のシート読み取りコマンド実行時にメインスタッカに排紙
SKDV_EJECT_SELECT_NEXT	4	停止位置にあるシートを、次のシート読み取りコマンド実行時にセレクトスタッカに排紙

5.4.15. SkDv_ReqClearError

処理	デバイスで発生しているエラーをクリアする。	
プロトタイプ	SkDvStatus SkDv_ReqClearError(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	エラー解除コマンド「CE」により、デバイスで発生しているエラーをクリアする。 ただし、解除不能のエラーの場合にはそのエラーが返る。	

5.4.16. SkDv_ReqGetMarks

処理	通信を行い、マークデータを取得する。	
プロトタイプ	SkDvStatus SkDv_ReqGetMarks(SkDvHandle hSkDevice, int iFace, SK_DV_MARK_INFO *pMarkInfo, char* pMarks, int* piBufSize)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	iFace	表面か裏面かを指定する。
	pMarkInfo	マーク情報を格納する為の構造体のアドレスを指定する。 構造体の内容は「5.3.8.SK_DV_MARK_INFO」を参照。
	pMarks	マークデータを格納する為のバッファのアドレスを指定する。1バイト毎にマークの濃度が0～16で格納される。 NULL を指定した場合は、マークデータは格納されない。
	piBufSize	マークデータを格納する為のバッファのバイトサイズを指定する。 指定されたサイズまで pMarks で指定されたバッファにデータを格納する。 処理実行後は、必要なバッファサイズが格納される。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	マーク濃度データ要求コマンド「MD」を使用し、マークデータを取得する。 引数 pMarkInfo には確保済みのマーク情報構造体のアドレスを指定する。 引数 pMarks には得られるはずのマークデータ数分のメモリを確保したバッファの先頭アドレス、引数 piBufSize にはそのバッファのバイトサイズを指定し、本 API 関数を呼び出す。	

5.5. API 関数 ー 拡張機能

5.5.1. SkDv_ReqGetInfo

処理	デバイス情報を取得する為の通信を行う。	
プロトタイプ	SkDvStatus　SkDv_ReqGetInfo(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	デバイス情報を取得する為の通信を行い、ライブラリー内部に保持する。 使用するコマンドは以下の通り。 ・装置情報要求コマンド「DI」 ・機種名要求コマンド「MN」 ・ファームウェアバージョン要求コマンド「FV」 ・その他非公開コマンド	

5.5.2. SkDv_ReqGetMode

処理	動作モードを取得する為の通信を行う。	
プロトタイプ	SkDvStatus　SkDv_ReqGetMode(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	動作モードを取得する為の通信を行い、ライブラリー内部に保持する。 使用するコマンドは以下の通り。コマンドの詳細は別紙「コマンドリファレンス」参照。 ・ワーニングエラー設定コマンド「WE」 ・パネル操作設定コマンド「PO」 ・ブザー設定コマンド「BZ」 ・低消費電力設定コマンド「ES」	

5.5.3. SkDv_ReqSetMode

処理	動作モードを設定する為の通信を行う。	
プロトタイプ	SkDvStatus　SkDv_ReqSetMode(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	ライブラリー内部に保持されている動作モードをデバイスに設定する。 使用するコマンドは「SkDv_ReqGetMode」関数と同じ。	

5.5.4. SkDv_ReqGetMarkConf

処理	マーク条件を取得する為の通信を行う。	
プロトタイプ	SkDvStatus　SkDv_ReqGetMarkConf(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	マーク条件を取得する為の通信を行い、ライブラリー内部に保持する。 使用するコマンドは以下の通り。コマンドの詳細は別紙「コマンドリファレンス」参照。 ・読み取り行数設定コマンド「NC」 ・読み取り方式設定コマンド「RM」 ・裏面読み取り設定コマンド「BR」 ・シート連量設定コマンド「FT」	

5.5.5. SkDv_ReqSetMarkConf

処理	マーク条件の設定する為の通信を行う。	
プロトタイプ	SkDvStatus　SkDv_ReqSetMarkConf(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	ライブラリー内部に保持されているマーク条件をデバイスに設定する。 使用するコマンドは「SkDv_ReqGetMarkConf」関数と同じ。	

5.6. API 関数 ― SR-3500HYBRID

5.6.1. SkDv_GetImageConf

処理	ライブラリー内に格納されているイメージ読み取りの動作条件を取得する。	
プロトタイプ	SkDvStatus SkDv_GetImageConf(SkDvHandle hSkDevice, SK_DV_SR3500IMG_CONF* pDvImageConf)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pDvImageConf	イメージ読み取りの動作条件を格納する為の構造体のアドレスを指定する。 構造体の内容は「5.3.6.SK_DV_SR3500IMG_CONF」を参照。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	取得するイメージ読み取りの動作条件は本ライブラリー内で保持される内容。 本ライブラリーが保持するイメージ読み取りの動作条件を引数 pDvMarkConf に格納する。 デバイスとの通信は行わない。 起動直後などデバイスに設定されている内容を取得したい場合には、事前に SkDv_ReqInit 関数または SkDv_ReqGetImageConf 関数を呼び出すこと。 画像読み取り機能搭載機種専用であるため、未搭載の場合には使用してはならない。	

5.6.2. SkDv_SetImageConf

処理	イメージ読み取りの動作条件をライブラリー内に設定する。	
プロトタイプ	SkDvStatus SkDv_SetImageConf(SkDvHandle hSkDevice, SK_DV_SR3500IMG_CONF*)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pDvImageConf	イメージ読み取りの動作条件を格納してある構造体のアドレスを指定する。 構造体の内容は「5.3.6.SK_DV_SR3500IMG_CONF」を参照。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	デバイス情報の取得が行われていなければ、デバイスと通信を行い、そのデバイス情報をライブラリー内部に格納する。 引数 pDvImageConf で指定されたイメージ読み取りの動作条件をデバイス情報の内容に合わせチェックし、値に問題があればエラーを返す。 エラーが無ければ、ライブラリー内部にマーク読み取りの動作条件を格納する。 イメージ読み取りの動作条件の設定内容は「読み取り要求」または SkDv_ReqSetImageConf 関数を実行することにより通信にてデバイスに設定する事ができる。 画像読み取り機能搭載機種専用であるため、未搭載の場合には使用してはならない。	

5.6.3. SkDv_ReqGetImageConf

処理	イメージ条件をデバイスから取得する為の通信を行う。	
プロトタイプ	SkDvStatus　SkDv_ReqGetImageConf(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	イメージ条件を取得する為の通信を行い、ライブラリー内部に保持する。 使用するコマンドは以下の通り。コマンドの詳細は別紙「コマンドリファレンス」参照。 ・画像読み取り設定コマンド「IR」 画像読み取り機能搭載機種専用であるため、未搭載の場合には使用してはならない。	

5.6.4. SkDv_ReqSetImageConf

処理	イメージ条件をデバイスに設定する為の通信を行う。	
プロトタイプ	SkDvStatus　SkDv_ReqSetImageConf(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	ライブラリー内部に保持されているイメージ条件をデバイスに設定する。 使用するコマンドは「SkDv_ReqGetImageConf」関数と同じ。 画像読み取り機能搭載機種専用であるため、未搭載の場合には使用してはならない。	

5.6.5. SkDv_ReqScanForm

処理	マークシートの読み取りを行い、イメージデータを取り込む。	
プロトタイプ	SkDvStatus　SkDv_ReqScanForm(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	シート読み取りコマンド「NS」により、マークシートの読み取りを行う。 情報取得/条件設定/CIS 基準データ取得の通信が行われていない場合には、読み取り前に通信が行われる。 帳票を読み取り、イメージデータを本ライブラリー内に格納する。 画像読み取り機能搭載機種専用であるため、未搭載の場合には使用してはならない。	

5.6.6. SkDv_GetFormSize

処理	読み取りを行った帳票のサイズを取得する。	
プロトタイプ	SkDvStatus SkDv_GetFormSize(SkDvHandle hSkDevice, SIZE* pSizeForm)	
引数	hSkDevice	デバイス制御ハンドルを指定する。

	pSizeForm	帳票のサイズを格納する構造体のアドレスを指定する。
		格納された値の単位は0.1mm。
		構造体 SIZE は MFC ライブラリリファレンスを参照。
		`typedef struct tagSIZE`
		`{`
		`  LONG  cx;`
		`  LONG  cy;`
		`} SIZE, *PSIZE, *LPSIZE;`

戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	本ミドルウェアが保持しているイメージデータから帳票のサイズを取得する。 幅 cx はデバイスが読み取れる最大幅で、長さ cy はデバイスがセンサにより検出した長さとなる。よって、帳票にあわせた画像を使用したい場合には、ユーザーによる幅の指定などが必要となる。 保持しているイメージデータとは SkDv_ReqScanForm 関数にて読み取ったイメージデータ。 画像読み取り機能搭載機種専用であるため、未搭載の場合には使用してはならない。	

5.6.7. SkDv_SaveImageData

処理	指定した範囲のイメージデータをファイルに保存する。	
プロトタイプ	SkDvStatus SkDv_SaveImageData(SkDvHandle hSkDevice, SK_DV_IMAGE_DATA_CONF* pImageDataConf, SK_DV_IMAGE_FILE_CONF* pImageFileConf, char* pszFilename)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pImageDataConf	イメージデータの取得条件を指定した構造体のアドレスを指定する。 構造体の内容は「5.3.9.SK_DV_IMAGE_DATA_CONF」を参照。
	pImageFileConf	保存するファイル形式を指定した構造体のアドレスを指定する。 構造体の内容は「5.3.10.SK_DV_IMAGE_FILE_CONF」を参照。
	pszFilename	保存するファイル名を格納した文字列のアドレスを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	本ライブラリーが保持しているイメージデータをファイルに保存する。 保持している画像データとは SkDv_ReqScanForm 関数にて読み取った画像データ。 イメージ条件より高い解像度で保存を行った場合には画質が荒くなり、またイメージ条件でグレースケールなどの単色で読み取った場合にはカラーで保存しても画像はグレースケールになることに注意すること。 保存するファイル名について、上書きするかどうかや予約語や禁止文字のチェックを本関数の呼び出す前に行うこと。 画像読み取り機能搭載機種専用であるため、未搭載の場合には使用してはならない。	

5.6.8. SkDv_SaveImageDataThread

処理	指定した範囲のイメージデータを別のスレッドでファイルに保存する。	
プロトタイプ	SkDvStatus WINAPI SkDv_SaveImageDataThread(SkDvHandle hSkDevice, SK_DV_IMAGE_DATA_CONF* pImageDataConf, SK_DV_IMAGE_FILE_CONF* pImageFileConf, char* pszFilename, FN_SK_DV_SAVEIMAGE_CALLBACK fnSaveImageCallback, LPVOID lpParameter)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pImageDataConf	イメージデータの取得条件を指定した構造体のアドレスを指定する。 構造体の内容は「5.3.9.SK_DV_IMAGE_DATA_CONF」を参照。
	pImageFileConf	保存するファイル形式を指定した構造体のアドレスを指定する。 構造体の内容は「5.3.10.SK_DV_IMAGE_FILE_CONF」を参照。
	pszFilename	保存するファイル名を格納した文字列のアドレスを指定する。
	fnSaveImageCallback	ファイル保存完了後に呼び出されるコールバック関数のポインタを指定する。 NULL を指定するとコールバックしない。
	lpParameter	コールバック関数に引き渡されるユーザー指定値。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	本ライブラリーが保持しているイメージデータを別のスレッドに渡してファイルに保存する。この為、ファイル保存が完了する前に本関数は終了することに注意すること。なお、表面裏面でそれぞれ別のスレッドでファイル保存は動作するが前回のスレッドが完了していない場合には、本関数内で待機する。 本関数を実行すると次に読み取りを行うまで SkDv_SaveImageData/SkDv_CreateImage 関数で画像保存することは出来ない。 ファイルに保存する内容は、SkDv_SaveImageData 関数と同じ。 引数 fnSaveImageCallback を NULL にした場合にはコールバックが行われない為、保存に失敗しても通知されない。よって、ファイルが正しく保存されているかユーザーアプリケーション側で確認する必要がある。また、ファイル保存が完了したか判断するには SkDv_CreateImage 関数を使用する必要がある。 引数 fnSaveImageCallback にコールバック関数を指定した場合、引数にてエラーステータスを受け取ることが出来る。 コールバック関数の内容は「5.6.9.FN_SK_DV_SAVEIMAGE_CALLBACK」を参照。 画像読み取り機能搭載機種専用であるため、未搭載の場合には使用してはならない。	

5.6.9. FN_SK_DV_SAVEIMAGE_CALLBACK

処理	画像データファイル保存(別スレッド)用のコールバック関数	
プロトタイプ	BOOL (CALLBACK *FN_SK_DV_SAVEIMAGE_CALLBACK)(SK_DV_IMAGE_DATA_CONF* pImageDataConf, SK_DV_IMAGE_FILE_CONF* pImageFileConf, char* pszFilename, SkDvStatus Status, LPVOID lpParameter)	
引数	pImageDataConf	イメージデータの取得条件を指定した構造体のアドレスを指定する。
	pImageFileConf	保存するファイル形式を指定した構造体のアドレスを指定する。
	pszFilename	保存するファイル名を格納した文字列のポインタ。
	Status	別スレッド実行したイメージデータファイル保存の結果のステータスコードが格納されている。
	lpParameter	SkDv_SaveImageDataThread 関数で指定されたユーザー指定値が格納されている。
戻り値	TRUE	次の設定にて別スレッドでのイメージデータファイル保存を行う。
	FALSE	別スレッドでのイメージデータファイル保存を終了する。
詳細	SkDv_SaveImageDataThread 関数でのファイル保存の結果を受け取る為にユーザーアプリケーションに実装する関数。引数 Status には別スレッド実行したイメージデータファイル保存の結果のステータスコードが格納されているのでエラー処理を実装することが望ましい。 引数pImageDataConf/pImageFileConf/pszFilenameの内容を変更し、戻り値をTRUEにすれば、その内容で再びイメージデータファイル保存が行われ、戻り値を FALSE にすることでファイル保存の為の別スレッドが終了する。	

5.6.10. SkDv_IsThreadRunning

処理	SkDv_SaveImageDataThread 関数で実行された別スレッドが終了したか確認する。	
プロトタイプ	BOOL WINAPI SkDv_IsThreadRunning(SkDvHandle hSkDevice)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
戻り値	TRUE	イメージデータファイル保存のスレッドが実行中。
	FALSE	イメージデータファイル保存のスレッドがすべて終了している。
詳細	SkDv_SaveImageDataThread 関数で実行された別スレッド（ファイル保存するスレッド）がすべて終了したか確認する事が出来る。 SkDv_SaveImageDataThread 関数でコールバックを使用しない場合には、本関数にて終了を確認することを推奨する。 画像読み取り機能搭載機種専用であるため、未搭載の場合には使用してはならない。	

5.6.11. SkDv_CreateImage

処理	指定した範囲のイメージデータを生成する。	
プロトタイプ	SkDvStatus SkDv_CreateImage (SkDvHandle hSkDevice, SK_DV_IMAGE_DATA_CONF* pImageDataConf, void** ppBitmapBits)	
引数	hSkDevice	デバイス制御ハンドルを指定する。
	pImageDataConf	イメージデータの取得条件を指定した構造体のアドレスを指定する。 構造体の内容は「5.3.9.SK_DV_IMAGE_DATA_CONF」を参照。
	ppBitmapBits	イメージデータのビットデータが格納されたアドレスを取得する為のアドレスを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	イメージデータを格納するメモリを確保し、本ライブラリーが保持しているイメージデータを格納したアドレスを返す。 本関数内でメモリを確保している為、「SkDv_DestroyImage」にてメモリの開放を行わなければならない。 ただし、イメージデータは「読み取り要求」を実行しないと取得できない。 画像読み取り機能搭載機種専用であるため、未搭載の場合には使用してはならない。	

5.6.12. SkDv_DestroyImage

処理	SkDv_CreateImage で生成したイメージデータを廃棄する。	
プロトタイプ	SkDvStatus SkDv_DestroyImage (SkDvHandle hSkDevice, BITMAPINFO* pBitmapInfo)	
引数	hSkDevice	デバイス制御ハンドルを指定する
	pBitmapBits	SkDv_CreateImage で取得したビットデータが格納されているアドレスを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	SkDv_CreateImage で確保したメモリを開放する。 ただし、イメージデータは「読み取り要求」を実行しないと保存できない。 画像読み取り機能搭載機種専用であるため、未搭載の場合には使用してはならない。	

5.7. API 関数 ― SR-11000

5.7.1. SkDv_ReqSetLayoutManage

処理	ウィンドウ制御を行うための ID 管理設定を行う。	
プロトタイプ	SkDvStatus SkDv_ReqSetLayoutManage(SkDvHandle hSkDevice, SK_LAYOUT_MANAGE_CONF* pLayManConf)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pLayManConf	条件が設定されているID管理構造体のポインタを指定
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数により、OMRに「ID管理データ設定 [CMD:WM]」コマンドが送信される。 ID管理データ設定コマンドは、OMRのレイアウト設定がすべてクリアされ、帳票IDの使用/未使用やIDウィンドウの内容などウィンドウ制御のレイアウトIDを認識ための設定が行われる。 ウィンドウ制御を行う際には最初に実行必要があり、最後には「レイアウト定義終了」関数を実行しなければならない。 (「ID データ設定」「ウィンドウ領域データ設定」よりも先に実行する必要がある。) IDウィンドウにはパーティション分割を行うことは出来ないため、引数pLayManConfの IdWindowPrm.iPartitionは設定しても無視される。	

5.7.2. SkDv_ReqSetLayoutManageTerminate

処理	ウィンドウ制御のレイアウト設定（ID データ設定、ウィンドウ領域データ設定）を終了する。	
プロトタイプ	SkDvStatus SkDv_ReqSetLayoutManageTerminate(SkDvHandle hSkDevice, int* piLimit)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	piLimit	連続読み取り動作の制限状態を格納する為のポインタを指定。NULL が指定可能。NULL を指定と制限状態は格納されない。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数により、OMR に「ID 管理データ設定 [CMD:WM]」コマンドが送信される。ウィンドウ制御の設定を終了する。本関数を実行後は、「ID データ設定」「ウィンドウ領域データ設定」は実行してはならない。	

5.7.3. SkDv_ReqSetLayoutId

処理	帳票のIDを設定し、ウィンドウエリアを設定出来るようにする。	
プロトタイプ	SkDvStatus SkDv_ReqSetLayoutId(SkDvHandle hSkDevice, SK_LAYOUT_ID_CONF* pLayIdConf, int* piIdIdx)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pLayIdConf	条件が設定されているIDデータ構造体のポインタを指定。
	piIdIdx	設定したIDデータの0から始まるID番号を格納する為のポインタを指定。NULLが指定可能。NULLを指定と設定したID番号は格納されない。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数により、OMRに「IDデータ設定 [CMD:WL]」コマンドが送信される。「ID データ管理設定」にて指定したIDウィンドウ領域のIDパターンを指定し、そのIDの帳票のタイミングマーク数などを指定。 「ID データ管理設定」にてIDを使用しない設定にした場合には、本関数でのIDパターンは無視されるが、1回だけは呼び出さなければならない。本関数を実行後にそのIDの「ウィンドウ領域データ設定」を行うことが出来る。	

5.7.4. SkDv_ReqSetWindowArea

処理	ウィンドウエリアの設定を行う。	
プロトタイプ	SkDvStatus SkDv_ReqSetWindowArea(SkDvHandle hSkDevice, SK_WINDOW_AREA_CONF* pWinAreaConf, int* piWinIdx)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pWinAreaConf	条件が設定されているウィンドウ領域構造体のポインタを指定。
	piWinIdx	設定したウィンドウ領域の0から始まるウィンドウ番号を格納する為のポインタを指定 ウィンドウ番号はすべてのIDを含めた全体の番号となる。 NULLが指定可能。NULLを指定と設定したウィンドウ番号は格納されない。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数により、OMRに「ウィンドウ領域データ設定[CMD:WD]」コマンドが送信される。 最後に行った「ID データ設定」のウィンドウ領域として設定される。	

5.7.5. SkDv_ReqClearLayout

処理	レイアウト設定の内容をすべてクリアする。	
プロトタイプ	SkDvStatus　SkDv_ReqClearLayout(SkDvHandle hSkDevice)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数により、OMRに「ウィンドウ情報クリア [CMD:WC]」コマンドが送信される。「ID データ管理設定」「ID データ設定」「ウィンドウ領域データ設定」の内容がクリアされる。	

5.7.6. SkDv_ReqSetPanelUserEnable

処理	OMRのLCDに指定された文字を表示し、パネルスイッチでの操作を禁止する。	
プロトタイプ	SkDvStatus　SkDv_ReqSetPanelUserEnable(SkDvHandle hSkDevice, BOOL blEnable)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	blEnable	FALSE　:無効。LCDとパネルスイッチはOMRが制御。 TRUE　:有効。LCDとパネルスイッチをアプリケーションで制御。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数により、OMRに「パネルリモート設定[CMD:PC]」コマンドが送信される。 OMRのLCDの表示は消えるが「パネル表示設定」関数にてLCDの表示内容を指定ことが出来る。 また、OMRのパネル操作は出来なるなるが、その代わりにスイッチ情報取得。により、 パネルスイッチの状態を取得。ことが出来る。	

5.7.7. SkDv_ReqGetPanelUserSwitch

処理	OMRのパネルスイッチの状態を取得する。	
プロトタイプ	SkDvStatus　SkDv_ReqGetPanelUserSwitch(SkDvHandle hSkDevice, DWORD* pdwSwitch)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pdwSwitch	パネルスイッチの状態を格納する為のポインタを指定。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数により、OMRに「パネルリモート設定[CMD:PC]」コマンドが送信される。 「リモート設定」関数で有効にした場合に、パネルスイッチの状態を取得。できる。 ラッチ情報については、リモート設定の有効とスイッチ情報取得。後に押されたパネルスイッチの情報となる。	

5.7.8. SkDv_ReqSetPanelUserString

処理	OMRのLCDに表示文字を設定する。	
プロトタイプ	SkDvStatus SkDv_ReqSetPanelUserString(SkDvHandle hSkDevice, char* pPanelStr)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pPanelStr	OMRのLCDに表示NULL終端文字列のポインタを指定。40文字まで設定可能。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数により、OMRに「パネルリモート設定[CMD:PC]」コマンドが送信される。 「リモート設定」関数で有効制御可能にした場合に、LCDに表示文字を指定できる。 引数pPanelStrに指定した文字列が40文字未満の場合、不足分はスペースとなる。 引数pPanelStrに指定した文字列が40文字を超えた場合には、超えた分は無視される。	

5.7.9. SkDv_ReqSetPirntFixString

処理	連続読み取りの固定印字文字を設定する。	
プロトタイプ	SkDvStatus SkDv_ReqSetPrintFixString(SkDvHandle hSkDevice, char* pString)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pString	連続読み取りの固定印字NULL終端文字列のポインタを指定。20文字まで設定可能。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数により、OMRに「プリンタ設定 [CMD:PR]」コマンドが送信される。 フォントサイズなどは「プリンタ印字条件」の設定/取得。関数を使用こと。 引数pStringに指定した文字列が20文字を超えた場合には、超えた分は無視される。	

5.7.10. SkDv_ReqSetContFeedPrint

処理	連続読み取りの固定印字文字を設定する。	
プロトタイプ	SkDvStatus SkDv_ReqSetContFeedPrint (SkDvHandle hSkDevice, SK_CONT_FEED_PRN_OPT* pPrnOpt)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pPrnOpt	条件が設定されている印字オプション構造体のポインタを指定する
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数により、OMRに「プリンタ設定 [CMD:PR]」コマンドが送信される。	

5.7.11. SkDv_ReqContFeedSheet

<table>
<tr><td>処理</td><td colspan="2">連続読み取りを開始する。</td></tr>
<tr><td>プロトタイプ</td><td colspan="2">SkDvStatus SkDv_ReqContFeedSheet(
SkDvHandle hSkDevice, DWORD dwReqData, int iErrReject)</td></tr>
<tr><td rowspan="3">引数</td><td>hSkDevice</td><td>開いたデバイス制御ハンドルを受け取る変数のポインタを指定。</td></tr>
<tr><td>dwReqData</td><td>データ出力方式を指定。通常は
SKDV_CONT_FEED_REQ_NORMALを指定すること。</td></tr>
<tr><td>iErrReject</td><td>エラー発生時の排紙先を指定する。SKDV_DISABLE(0):メイン排紙 / SKDV_ENABLE(1):セレクト排紙(ﾘｼﾞｪｸﾄ)</td></tr>
<tr><td rowspan="2">戻り値</td><td>SKDV_STS_SUCCESS</td><td>成功</td></tr>
<tr><td>上記以外</td><td>失敗
(オープンできるデバイスがない、又は既に接続されている。)</td></tr>
<tr><td>詳細</td><td colspan="2">OMRに「シート連続読み取り [CMD:CF]」コマンドを送信し、連続読み取りを開始。
本関数では内部でワーカースレッドを生成し、マークデータなどの受信を自動で行い。
すべての読み取りが完了(エラー停止なども含む)したらそのワーカースレッドは自動で終了。
ワーカースレッドの動作状況は「連続読み取り動作中確認」で確認ことが出来る。
ワーカースレッドが動作中は「連続読み取りキャンセル要求」以外のOMRと通信関数を呼び出してはならない。
ワーカースレッドが動作中に使用してよい関数は以下の通り。

<table>
<tr><th>関数名</th><th>意味</th></tr>
<tr><td>SkDv_ReqContFeed_Cancel</td><td>連続読み取りキャンセル要求</td></tr>
<tr><td>SkDv_IsContFeedRunning</td><td>連続読み取り動作中確認</td></tr>
<tr><td>SkDv_IsContFeedDataFinished</td><td>連続読み取りデータ終了確認</td></tr>
<tr><td>SkDv_ExistDoneData</td><td>連続読み取り未取得。データ確認</td></tr>
<tr><td>SkDv_GetContFeedCount</td><td>連続読み取り読み取り状況(枚数)取得。</td></tr>
<tr><td>SkDv_PrepareContData</td><td>連続読み取りデータ準備</td></tr>
<tr><td>SkDv_GetContDataNumber</td><td>連続読み取りデータカレントデータ番号取得。</td></tr>
<tr><td>SkDv_GetContDataMarks</td><td>連続読み取りマークデータ取得。</td></tr>
<tr><td>SkDv_GetContDataWindows</td><td>連続読み取りウィンドウデータ取得。</td></tr>
<tr><td>SkDv_GetContDataBarcodes</td><td>連続読み取りバーコードデータ取得。</td></tr>
<tr><td>SkDv_GetContDataSheetResult</td><td>連続読み取り帳票読み取り結果取得。</td></tr>
<tr><td>SkDv_GetContFeedResult</td><td>連続読み取り結果取得。</td></tr>
</table>
</td></tr>
</table>

5.7.12. SkDv_ReqContFeed_Cancel

処理	連続読み取りの動作をキャンセルする。	
プロトタイプ	SkDvStatus SkDv_ReqContFeed_Cancel(SkDvHandle hSkDevice)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数により、OMRに「キャンセル動作 [CMD:CA]」コマンドが送信され、 OMRへ連続読み取りの中断を指示。 本関数から制御が戻っても連続読み取りが終了しているとは限らない。 動作状況は「連続読み取り動作中確認」で確認ことが出来る。	

5.7.13. SkDv_IsContFeedRunning

処理	連続読み取りが動作しているか確認する。	
プロトタイプ	BOOL SkDv_IsContFeedRunning(SkDvHandle hSkDevice)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
戻り値	TRUE	連続読み取りスレッドは実行中。
	FALSE	連続読み取りスレッドは終了している。
詳細	「連続読み取り要求」関数にて実行されているワーカースレッドが動作中か確認できる。 なお、「連続読み取りキャンセル要求」関数を実行してもワーカースレッドは直ちに終了しない。	

5.7.14. SkDv_IsContFeedDataFinished

処理	「連続読み取り要求」関数で開始した連続読み取りが終了しデータもなくなっているかどうかの状態を取得する。	
プロトタイプ	BOOL SkDv_IsContFeedDataFinished(SkDvHandle hSkDevice)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
戻り値	TRUE	連続読み取りスレッドは実行中。
	FALSE	連続読み取りスレッドは終了している。
詳細	「連続読み取り要求」関数にて実行されているワーカースレッドが終了し、 「連続読み取りデータ準備」関数にてすべての読み取ったデータの取得。が終わったか確認できる。 本関数は、「連続読み取り動作中確認」関数で連続読み取りのワーカースレッドが終了し、さらに「連続読み取り未取得。データ確認」関数で全てのデータがなくなっていた場合にTRUEを返している。	

5.7.15. SkDv_ExistDoneData

処理	「連続読み取り要求」関数で開始した連続読み取りのデータが蓄積されているか確認する。	
プロトタイプ	BOOL SkDv_ExistDoneData(SkDvHandle hSkDevice)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
戻り値	TRUE	連続読み取りのデータあり。
	FALSE	連続読み取りのデータなし。
詳細	「連続読み取り要求」関数にて実行されているワーカースレッドが蓄積した連続読み取りデータが、本ライブラリー内に存在か確認できる。 連続読み取りのワーカースレッドが動作中だが連続読み取りデータがまだ蓄積されていない場合は、FALSEを返すことがあることに注意こと。	

5.7.16. SkDv_GetContFeedCount

処理	連続読み取りのワーカースレッドが受け取った現在の枚数状況を取得する。	
プロトタイプ	SkDvStatus SkDv_GetContFeedCount(SkDvHandle hSkDevice, SK_CONT_FEED_COUNT* pContFeedCount)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pContFeedCount	現在の枚数状況を格納する為の連続読み取り枚数構造体のポインタを指定。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	「連続読み取り要求」関数にて実行されているワーカースレッドの現在の枚数状況を取得する。	

5.7.17. SkDv_PrepareContData

処理	連続読み取りのワーカースレッドが受け取ったデータを取得。準備を行う。	
プロトタイプ	SkDvStatus SkDv_PrepareContData(SkDvHandle hSkDevice)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
戻り値	SKDV_STS_SUCCESS	連続読み取り動作にて、実際に1枚の帳票を読み取った際のステータスが返る。
	上記以外	失敗 (オープンできるデバイスがない、又は既に接続されている。)
詳細	本関数を実行ことで、連続読み取りのワーカースレッドが受け取ったデータを取得できるように準備を行う。 連続読み取りデータがまだ蓄積されていない場合はワーカースレッドがデータを受け取るまで、本関数内で待機する。待機時間は状況により異なるが、十数秒掛かる可能性もあるため、処理を止めたくない場合には、「連続読み取りデータ蓄積確認」関数にて蓄積されたデータがあることを確認してから本関数を実行ことが望ましい。	

5.7.18. SkDv_GetContDataNumber

処理	連続読み取りデータの現在のデータ番号(枚数)を取得する。	
プロトタイプ	int SkDv_GetContDataNumber(SkDvHandle hSkDevice)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
戻り値	0	「連続読み取り要求」関数を実行した直後の値。
	上記以外	「連続読み取りデータ準備」関数を実行した値。 (実際に読みとった枚数+1)
詳細	「連続読み取り要求」関数を実行した直後に本関数を実行すると戻り値は0となる。 その後「連続読み取りデータ準備」を実行のたびに1ずつ繰り上がり、実際に読み取った枚数＋1まで進む。	

5.7.19. SkDv_GetContDataMarks

処理	連続読み取りデータ準備で準備されたマークデータを取得する。	
プロトタイプ	BOOL SkDv_GetContDataMarks(SkDvHandle hSkDevice, int iFace, SK_DV_MARK_INFO *pMarkResult, char* pMarks, int* piBufSize)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	iFace	表面か裏面かを指定。
	pMarkResult	マーク情報を格納する為の構造体のアドレスを指定。 構造体の内容は「SK_DV_MARK_INFO」を参照。
	pMarks	マークデータを格納する為のバッファーのアドレスを指定。1バイト毎にマークの濃度が0～16で格納される。NULLを指定した場合は、マークデータは格納されない。
	piBufSize	マークデータを格納する為のバッファーのバイトサイズを指定。指定されたサイズまで pMarks で指定されたバッファにデータを格納。処理実行後は、必要なバッファサイズが格納される。
戻り値	TRUE	正常
	FALSE	失敗 (引数に異常がある場合やデータがない場合)
詳細	「連続読み取りデータ準備」関数で準備されたマークデータを取得。引数の内容はSkDv_ReqGetMarks関数と同じである。	

5.7.20. SkDv_GetContDataBarcodes

処理	連続読み取りデータ準備で準備されたバーコードデータを取得する	
プロトタイプ	BOOL SkDv_GetContDataBarcodes(SkDvHandle hSkDevice, SK_DV_BARCODE_RESULT *pBarcodeResult, char* pBarcodes, int* piBufSize)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pBarcodeResult	バーコード情報を格納する為の構造体のアドレスを指定する。
	pBarcodes	バーコードデータを格納する為のバッファーのアドレスを指定する。 バーコードデータは種別/データのバイト数/データが読み取った数分が並ぶ。
	piBufSize	バーコードデータを格納する為のバッファーのバイトサイズを指定する。 指定されたサイズまで pBarcodes で指定されたバッファにデータを格納する。 処理実行後は、必要なバッファサイズが格納される。
戻り値	TRUE	正常
	FALSE	失敗 (引数に異常がある場合やデータがない場合)
詳細	「連続読み取りデータ準備」関数で準備されたバーコードデータを取得する。 バーコードデータは種別/データのバイト数/データが読み取った数分が並び、カンマで区切られている為、データを解析する必要がある。 1つ毎のデータを処理する場合には、SkDv_GetContDataBarcodesCount関数で読み取ったバーコードの数を取得し、SkDv_GetContDataBarcodesData関数で1つ毎のバーコードデータを取得する。	

5.7.21. SkDv_GetContDataBarcodesCount

処理	連続読み取りデータ準備で準備された読み取ったバーコード数を取得する	
プロトタイプ	BOOL SkDv_GetContDataBarcodesCount(SkDvHandle hSkDevice, int* piReadCount)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	piReadCount	読み取ったバーコードの数を格納する為の変数のアドレスを指定する。
戻り値	TRUE	正常
	FALSE	失敗 (引数に異常がある場合やデータがない場合)
詳細	「連続読み取りデータ準備」関数で準備されたバーコードの読み取った数を取得する。 SkDv_GetContDataBarcodes関数で取得できるpBarcodeResult->iCountと同じ。	

5.7.22. SkDv_GetContDataBarcodes

処理	連続読み取りデータ準備で準備された読み取ったバーコードの指定した番号のデータを取得する	
プロトタイプ	BOOL SkDv_GetContDataBarcodesData(SkDvHandle hSkDevice, int iIndex, char* pcBcType, char* pszBarcode, int* piDataLen)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	iIndex	1から始まる読み取ったバーコードのインデックス番号を指定する。
	pcBcType	iIndexで指定したバーコードの種類を格納する為の変数のポインタを指定する。
	pBarcode	iIndexで指定したバーコードデータを格納する為のバッファーのアドレスを指定する。
	piBufSize	iIndex で指定したバーコードデータを格納する為のバッファーのバイトサイズを指定する。 指定されたサイズまで pszBarcode で指定されたバッファにデータを格納する。 処理実行後は、必要なバッファサイズが格納される。
戻り値	TRUE	正常
	FALSE	失敗 (引数に異常がある場合やデータがない場合)
詳細	「連続読み取りデータ準備」関数で準備されたバーコードの指定した番号のデータを取得する。 SkDv_GetContDataBarcodes関数で取得できるバーコードデータを1つずつに分解した結果となる。	

5.7.23. SkDv_GetContDataSheetResult

処理	連続読み取りデータ準備で準備された帳票読み取り結果を取得する。	
プロトタイプ	BOOL SkDv_GetContDataSheetResult(SkDvHandle hSkDevice, SK_SHEET_RESULT *pSheetResult)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pSheetResult	連続帳票読み取り帳票結果構造体のアドレスを指定。
戻り値	TRUE	正常
	FALSE	失敗 (引数に異常がある場合やデータがない場合)
詳細	連続帳票読み取り帳票結果を取得する関数。	

5.7.24. SkDv_GetContFeedResult

処理	連続読み取り結果を取得する。	
プロトタイプ	BOOL SkDv_GetContFeetResult(SkDvHandle hSkDevice, SK_CONT_FEED_RESULT *pContFeedResult)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pContFeedResult	連続帳票読み取り結果構造体のアドレスを指定。
戻り値	TRUE	正常
	FALSE	失敗 (引数に異常がある場合やデータがない場合)
詳細	連続帳票読み取り結果を取得する関数。	

5.8. API 関数 ー オプション

5.8.1. SkDv_GetBcrConf

処理	ライブラリ内に格納されているバーコードリーダ読取条件を取得する。	
プロトタイプ	SkDvStatus SkDv_GetBcrConf (SkDvHandle hSkDevice, SK_DV_OPT_BCR_CONF* pBcrConf)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pBcrConf	バーコードリーダ読取条件を格納する為の構造体のポインタを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	取得するバーコードリーダ読取条件は本ライブラリ内で保持される内容。 本ライブラリが保持するバーコードリーダ読取条件を引数pBcrConfに格納する。 デバイスとの通信は行わない。 起動直後などデバイスに設定されている内容を取得したい場合には、事前に SkDv_ReqInit関数またはSkDv_ReqGetBcrConf関数を呼び出すこと。	

5.8.2. SkDv_SetBcrConf

処理	バーコードリーダ読取条件をライブラリ内に設定する。	
プロトタイプ	SkDvStatus SkDv_SetBcrConf (SkDvHandle hSkDevice, SK_DV_OPT_BCR_CONF* pBcrConf)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pBcrConf	バーコードリーダ読取条件が格納された構造体のポインタを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	デバイス情報の取得が行われていなければ、デバイスと通信を行い、そのデバイス情報をライブラリ内部に格納する。 引数pBcrConfで指定されたバーコードリーダ読取条件をデバイス情報の内容に合わせチェックし、値に問題があればエラーを返す。 エラーが無ければ、ライブラリ内部にバーコードリーダ読取条件を格納する。 バーコードリーダ読取条件の設定内容は「読み取り要求」またはSkDv_ReqSetBcrConf関数を実行することにより、通信にてデバイスに設定する事ができる。	

5.8.3. SkDv_ReqGetBcrConf

処理	デバイスに対し、バーコードリーダ読取条件を取得する為の通信を行う。	
プロトタイプ	SkDvStatus　SkDv_ReqGetBcrConf (SkDvHandle hSkDevice)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	バーコードリーダ読取条件を取得する為の通信を行い、ライブラリ内部に保持する。 使用するコマンドは以下の通り。コマンドの詳細は別紙「コマンドリファレンス」参照。 ・バーコード設定コマンド「BC」	

5.8.4. SkDv_ReqSetBcrConf

処理	デバイスに対し、バーコードリーダ読取条件を設定する為の通信を行う。	
プロトタイプ	SkDvStatus　SkDv_ReqSetBcrConf (SkDvHandle hSkDevice)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	ライブラリ内部に保持されているバーコードリーダ読取条件をデバイスに設定する。 使用するコマンドは「SkDv_ReqGetBcrConf」関数と同じ。	

5.8.5. SkDv_ReqGetBcrDataCount

処理	読み取ったバーコードの数を通信で取得する。	
プロトタイプ	SkDvStatus　SkDv_ReqGetBcrDataCount(SkDvHandle hSkDevice, int* piReadCount)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	piReadCount	読み取ったバーコードの数を格納する為の変数のアドレスを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	バーコードデータ要求「BD」により、読み取り要求で読み取ったバーコードの数を取得する。 連続読取要求で読み取った場合には取得することは出来ない。	

5.8.6. SkDv_ReqGetBcrData

処理	読み取ったバーコードの指定された番号のデータを通信で取得する。	
プロトタイプ	SkDvStatus　SkDv_ReqGetBcrData(SkDvHandle hSkDevice, int iIndex, char* pcBcType, char* pszBarcode, int* piDataLen)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	iIndex	１から始まる読み取ったバーコードのインデックス番号を指定する。
	pcBcType	iIndex で指定したバーコードの種類を格納する為の変数のポインタを指定する。
	pszBarcode	iIndex で指定したバーコードデータを格納する為のバッファーのアドレスを指定する。
	piDataLen	iIndex で指定したバーコードデータを格納する為のバッファーのバイトサイズを指定する。 指定されたサイズまでpszBarcode で指定されたバッファにデータを格納する。 処理実行後は、必要なバッファサイズが格納される。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	バーコードデータ要求「BD」により、読み取り要求で読み取ったバーコードデータを取得する。 連続読取要求で読み取った場合には取得することは出来ない。	

5.8.7. SkDv_GetPrinterConf

処理	ライブラリー内に格納されているプリンタ印字条件を取得する。	
プロトタイプ	SkDvStatus SkDv_GetPrinterConf(SkDvHandle hSkDevice, SK_DV_OPT_PRN_CONF* pPrnConf)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pPrnConf	プリンタ印字条件を格納する為の構造体のポインタを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	取得するプリンタ印字条件は本ライブラリ内で保持される内容。 本ライブラリが保持するプリンタ印字条件を引数pPrnConfに格納する。 デバイスとの通信は行わない。 起動直後などデバイスに設定されている内容を取得したい場合には、事前に SkDv_ReqInit関数またはSkDv_ReqGetPrinterConf関数を呼び出すこと。	

5.8.8. SkDv_SetPrinterConf

処理	プリンタ印字条件をライブラリー内に設定する。	
プロトタイプ	SkDvStatus SkDv_SetPrinterConf(SkDvHandle hSkDevice, SK_DV_OPT_PRN_CONF* pPrnConf)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pPrnConf	プリンタ印字条件が格納された構造体のポインタを指定する。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	ミドルウェアにプリンタ印字条件を設定する。	

5.8.9. SkDv_ReqPrintString

処理	印字したい文字をOMRに設定する。	
プロトタイプ	SkDvStatus SkDv_ReqPrintString(SkDvHandle hSkDevice, char* pString)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
	pString	印字したい文字列のアドレスを指定。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	印字したい文字を設定する。	

5.8.10. SkDv_ReqGetPrinterConf

処理	印字条件を取得する。	
プロトタイプ	SkDvStatus SkDv_ReqGetPrinterConf(SkDvHandle hSkDevice)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	印字条件を取得する。	

5.8.11. SkDv_ReqSetPrinterConf

処理	印字条件を設定する。	
プロトタイプ	SkDvStatus SkDv_ReqSetPrinterConf(SkDvHandle hSkDevice)	
引数	hSkDevice	開いたデバイス制御ハンドルを受け取る変数のポインタを指定。
戻り値	SKDV_STS_SUCCESS	成功
	上記以外	失敗
詳細	印字条件を設定する。	

5.9. 設定とデータ

5.9.1. 実行する API 関数による設定データの流れ

固定データのデバイス情報や設定データのデバイスモード/マーク条件/イメージ条件は、新 API ライブラリーと OMR 本体、設計によっては上位アプリケーションの3箇所にそれぞれで保持します。
それぞれのデータは API 関数の実行により、データの反映が行われます。その関係は下図の通りです。

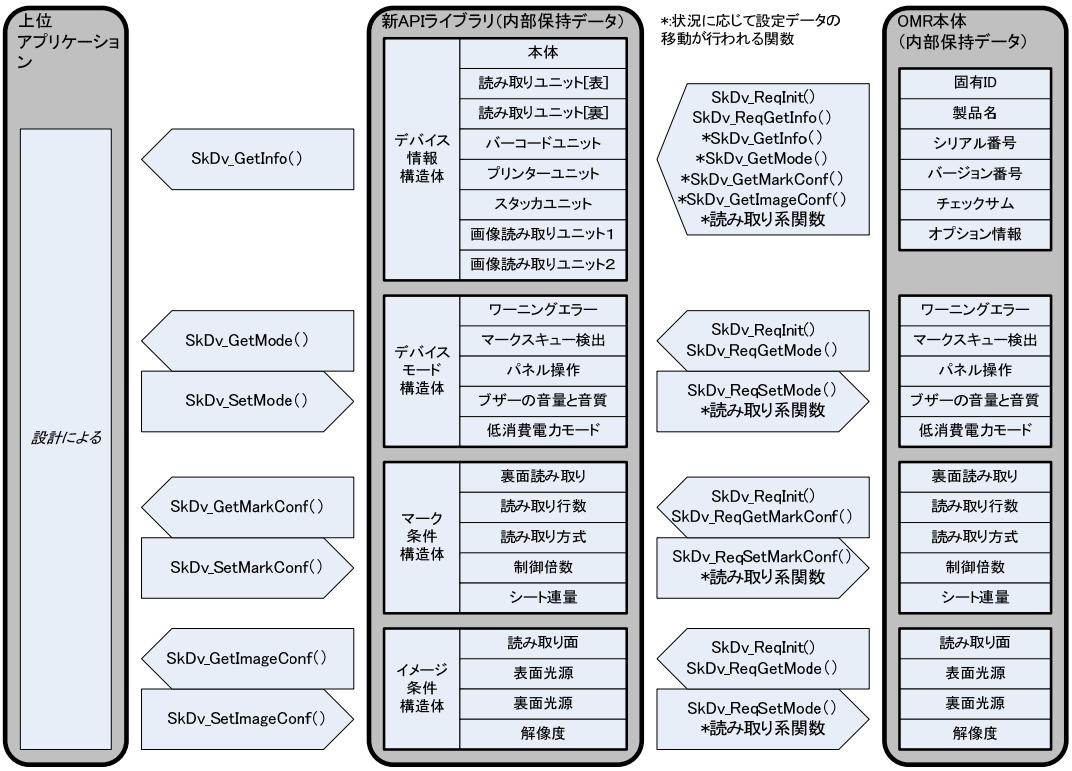


図 4 実行する API 関数による設定データの流れ

上位アプリケーションがデバイス情報を取得する「SkDv_GetInfo 関数」を行うと、新 API ライブラリーは必要に応じて自動的に OMR 本体からデバイス情報を受けり上位アプリケーションに返します。
また、デバイスモード設定「SkDv_SetMode 関数」などで設定を変更した後に読み取り系関数を実行した場合にも、新 API ライブラリーは必要に応じて自動的に OMR 本体にモードや条件を設定します。
よって、新 API ライブラリーを使用するにあたり、データの流れを意識せずに使用することも出来ますし、拡張機能の API 関数(SkDv_ReqGetMode/ SkDv_ReqSetMode 関数など)により制御することも出来ます。
ただし、旧 API ライブラリーと併用する場合には、旧 API ライブラリーを使用して設定を変更しても新 API ライブラリーの自動処理により再び設定が変更させてしまう場合があります。本章と「5.9.2 実行する API 関数による設定データの状態遷移」の内容を理解した上で使用してください。

5.9.2. 実行する API 関数による設定データの状態遷移

設定データはデバイスモード/マーク条件/イメージ条件の3つがあり、新 API ライブラリーと OMR 本体でそれぞれ保持しています。
設定データは API 関数の実行により差異が生じます。その状態は下図の通りです。

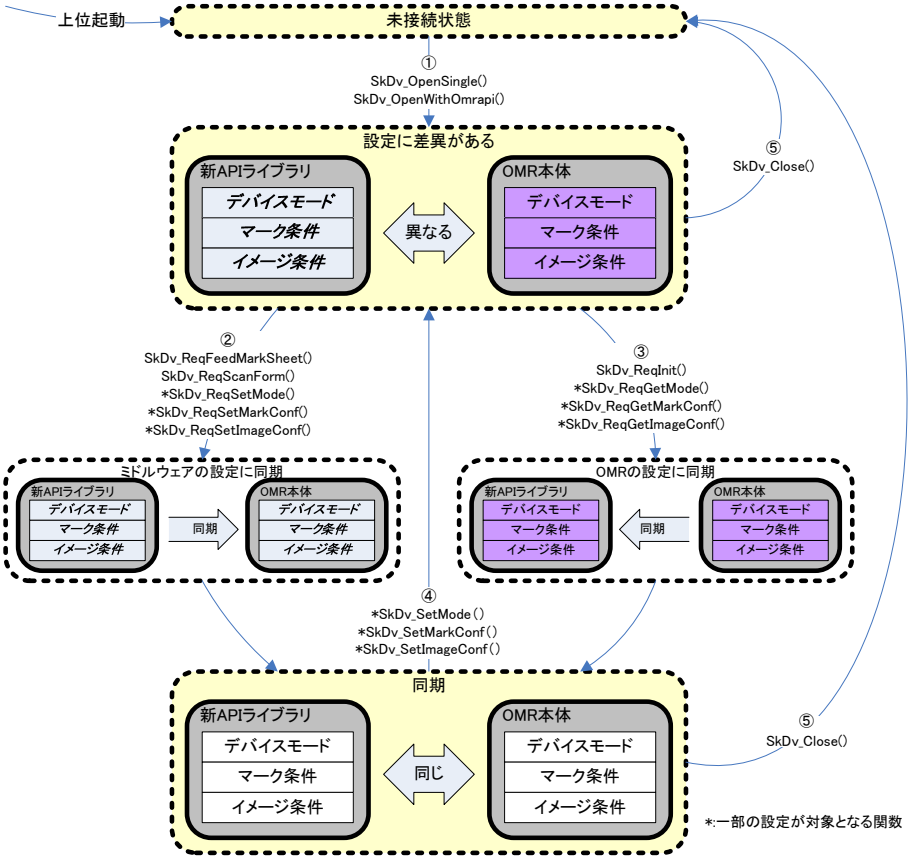


図 5 実行する API 関数による設定データの状態遷移

	実行する API 関数	解説
①	SkDv_OpenSingle() SkDv_OpenWithOmrapi()	開始処理を実行した直後は新 API ライブラリーでは初期値で、OMR 本体は不揮発メモリに保持された設定となり、設定内容は差異が生じている。
②	SkDv_ReqFeedMarkSheet() SkDv_ReqScanForm() *SkDv_ReqSetMode() *SkDv_ReqSetMarkConf() *SkDv_ReqSetImageConf()	読み取り要求関数を実行した場合には、新 API ライブラリーに保持されている設定内容が OMR に反映され、新 API ライブラリーと OMR の設定内容は同じ状態となる。 拡張機能の設定通信要求 API 関数(SkDv_ReqSetMode 関数など)を実行することで、その設定内容が反映される。
③	SkDv_ReqInit() *SkDv_ReqGetMode() *SkDv_ReqGetMarkConf() *SkDv_ReqGetImageConf()	SkDv_ReqInit 関数を実行した場合には、OMR に保持されている設定内容が新 API ライブラリーに反映され、新 API ライブラリーと OMR の設定内容は同じ状態となる。 拡張機能の取得通信要求 API 関数(SkDv_ReqGetMode 関数など)を実行することで、その設定内容が反映される。
④	*SkDv_SetMode() *SkDv_SetMarkConf() *SkDv_SetImageConf()	新 API ライブラリーと OMR の設定内容が同じ状態から、設定関数(SkDv_SetMode 関数など)を実行すると新 API ライブラリーの設定内容が変更され、設定内容は差異が生じる。
⑤	SkDv_Close()	終了処理を実行すると、新 API ライブラリーの設定内容は破棄される。よって、再び開始処理を実行しても以前の設定内容は保持されない。

5.9.3. イメージ条件と保存される画像ファイル

読み取りを行うとイメージデータは新 API ライブラリー内に格納されますが、解像度とカラー/グレースケールは表面/裏面で共通にする必要があります。使用目的によっては表と裏で個別の解像度にしたい場合などがありますので、読み取ったイメージデータから様々な設定で画像ファイルに保存することが出来ます。
カラー/300dpi で読み取りを行った場合には、低解像度やグレースケール/白黒2値で保存することが可能です。

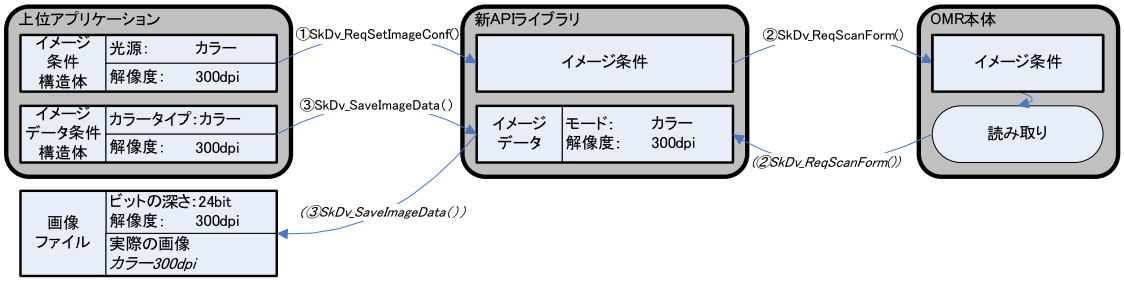


図 6 イメージ条件と保存される画像ファイル

しかし、グレースケール/低解像度で読み取りを行った場合には、カラー/300dpi で画像ファイルに保存するとファイル上はカラー(ビットの深さ24bit)で解像度が 300dpi になりますが、実際の画像を見るとグレースケールで解像度は荒くなります。(アップスケーリングはされません。)

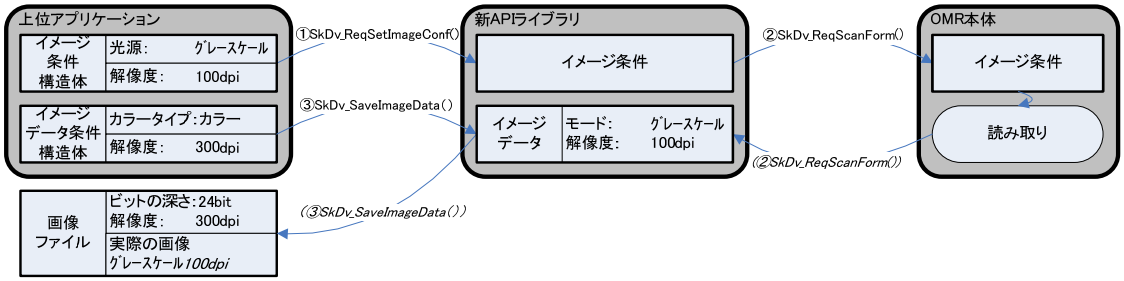


図 7 画像ファイルの形式と実際の画像との差異

5.10. 使用例

5.10.1. 新 API ライブラリーを使用する場合

1. 開始処理関数により、OMR と通信可能な状態にする。

2. 初期化要求関数により、OMR に設定されている各種動作モード/各種マーク読み取り設定/各種イメージ読み取り設定の取得がおこなわれ、ライブラリー内にその内容が格納される。
これにより、OMR とライブラリーで各設定内容が一致している状態になる。

3. 動作モードを変更する場合には、動作モード構造体に各値を格納し、動作モード設定関数に引数で渡す。この時ライブラリー内に動作モードが格納される。

4. マーク読み取りの条件を変更する場合には、マーク条件構造体に各値を格納し、マーク条件設定関数に引数で渡す。この時ライブラリー内にマーク条件が格納される。

5. イメージ読み取りの条件を変更する場合には、イメージ条件構造体に各値を格納し、イメージ条件設定関数に引数で渡す。
この時ライブラリー内にイメージ条件が格納される。

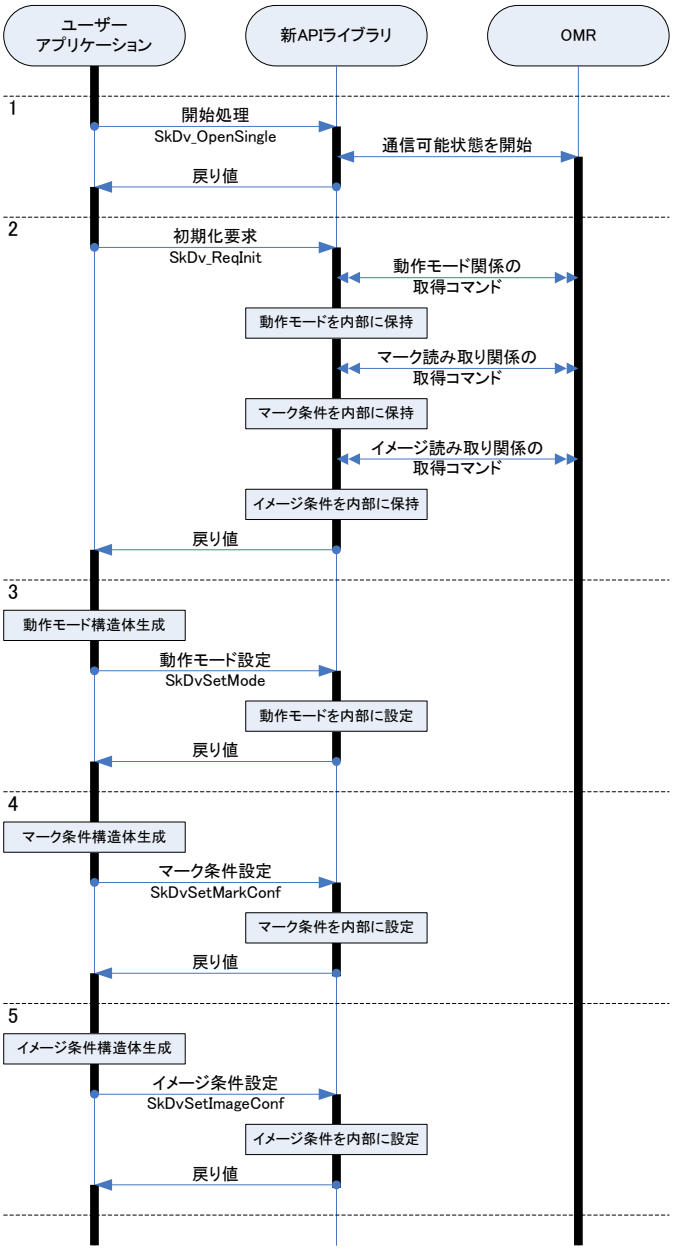


図 8 新 API ライブラリー制御イメージ(1)

6. 帳票の読み取りを行うには、読み取り要求関数を呼ぶ。
ライブラリーは、動作モード/マーク条件/イメージ条件が変更されている場合に各コマンドにより設定を行う。これにより、ライブラリーとOMR の設定が一致する。
その後、読み取りコマンドにより帳票の読み取りを行い、イメージデータをライブラリー内に格納する。

7. マークデータを得るには、マークデータを受け取る為のマーク情報構造体とマークデータを格納するバッファを用意し、マークデータ取得要求関数にて取得する。ライブラリーはマーク濃度データ要求コマンドにより受け取ったデータを返す。
受け取ったマークデータの処理を行う。

8. イメージをファイル保存するには、イメージデータ条件/イメージファイル条件構造体に値を設定し、イメージデータファイル保存関数を呼び出す。ライブラリー内にある全イメージデータを基に指定された条件によりファイル保存される。
必要に応じてデータ処理を行う。

（2枚目以降は6から繰り返す）
（設定を変更するには3から行う）

9. OMR の制御を終了するには終了処理関数を呼び、終了する。

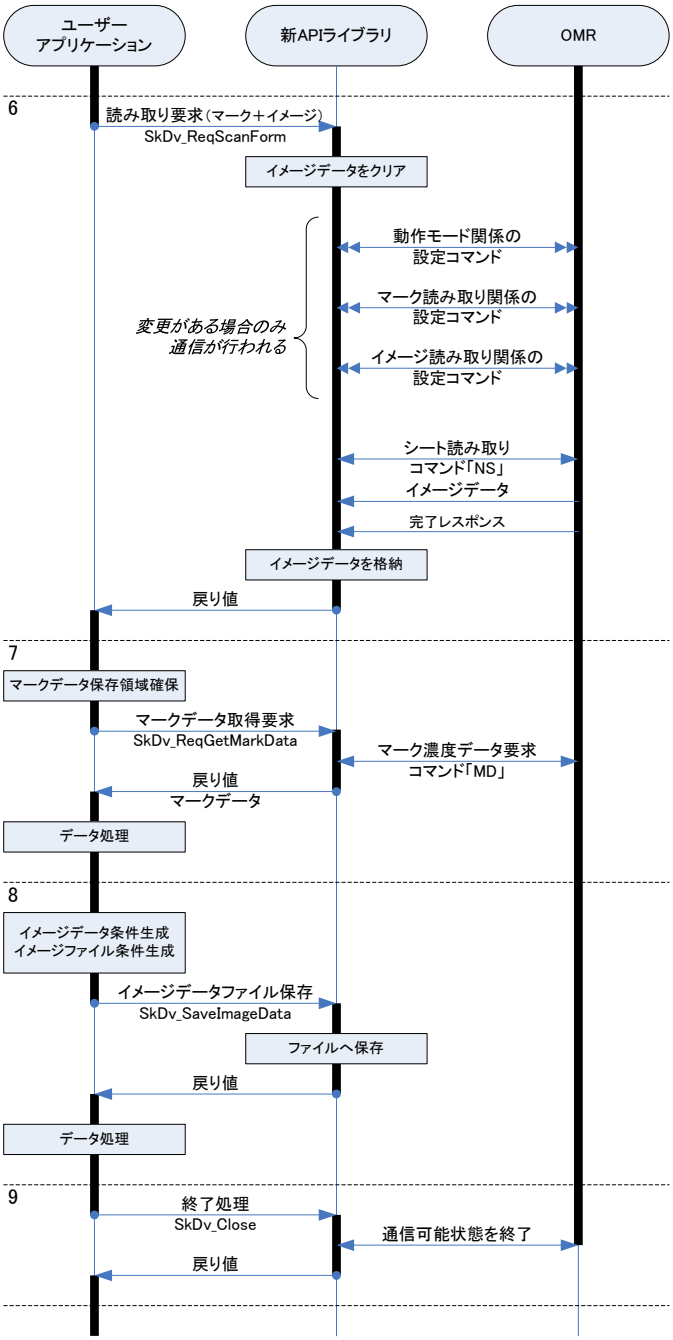


図 9 新 API ライブラリー制御イメージ（2）

5.10.2. イメージデータを別のスレッドでファイルに保存

新 API ライブラリーでは、処理能力を向上するためにイメージデータを別のスレッドでファイルに保存する機能を有しています。使用方法と内部動作について下記に説明します。
※この機能を使用するにはマルチスレッドやコールバックについての知識が必要です。理解したうえで使用することをおすすめします。

1. 帳票の読み取りを行うには、読み取り要求関数を呼ぶが、ファイル保存の方法に関わらず処理する。
一旦クリアした表面/裏面バッファに読み取ったイメージデータが格納される。

2. イメージデータをファイル保存する際、別スレッドにて行うには SkDv_SaveImageDataThread 関数を使用する。この関数は、イメージデータが格納されたバッファを別スレッド用にし読み取り用には新たにバッファを生成した後、スレッドを生成してイメージデータが格納されたバッファや必要なデータを渡して制御を戻す。よって、この時点では画像ファイルは保存されておらず、またイメージデータが格納されたバッファは別スレッド用に渡しているため SkDv_SaveImageData 関数などを使用しても画像ファイルに保存することは出来ない。

3. 読み取ったイメージデータは既に別のスレッドにて保存処理を行っている為、ここで次の帳票を読み取るために読み取り要求関数を呼び出すことが出来る。別スレッドが画像ファイルの保存が完了するとコールバック関数が呼び出される。右図においては読み取り中にコールバックしているが、パソコンのスペックや状況によってタイミングは異なる。なお、前回呼び出した別スレッドが終了するまで SkDv_SaveImageDataThread 関数は関数内部で待機する。

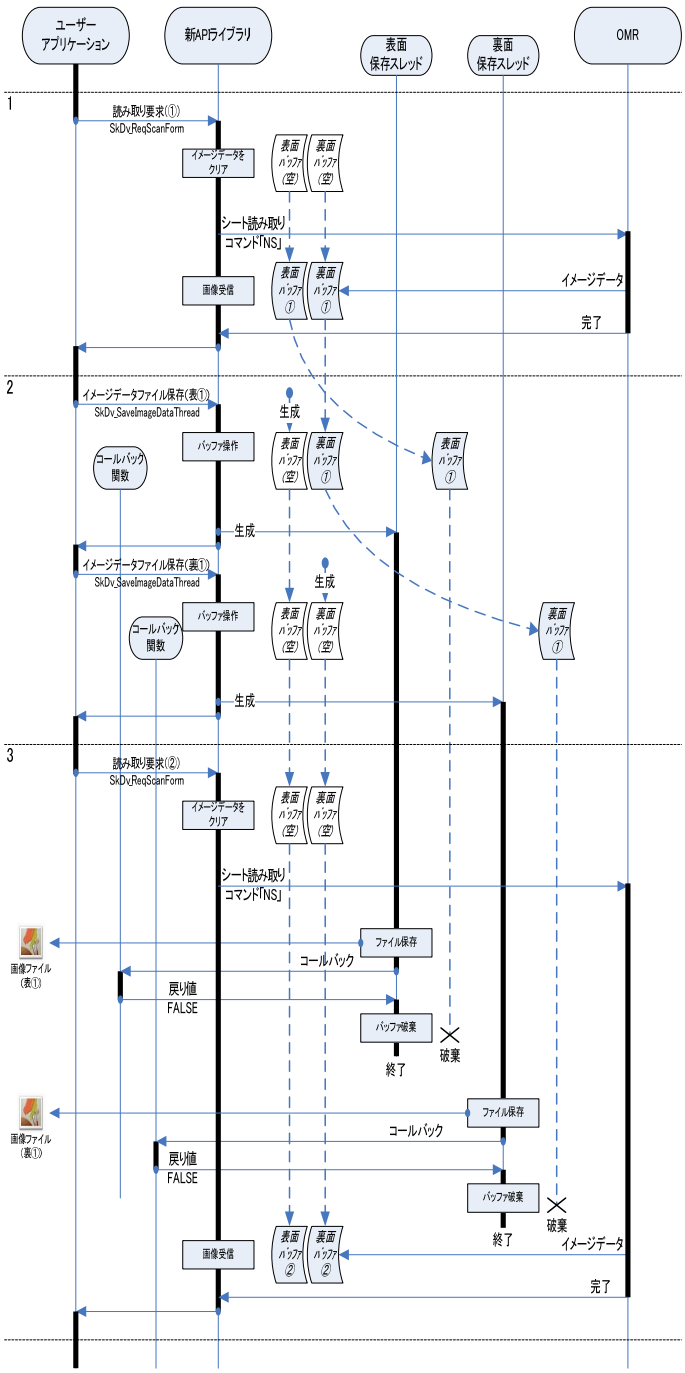


図 10 イメージデータを別のスレッドでファイルに保存

5.10.3. TM 数のチェックを行わずに帳票を読みとる場合

1. 開始処理関数により、OMR と通信可能な状態にする。

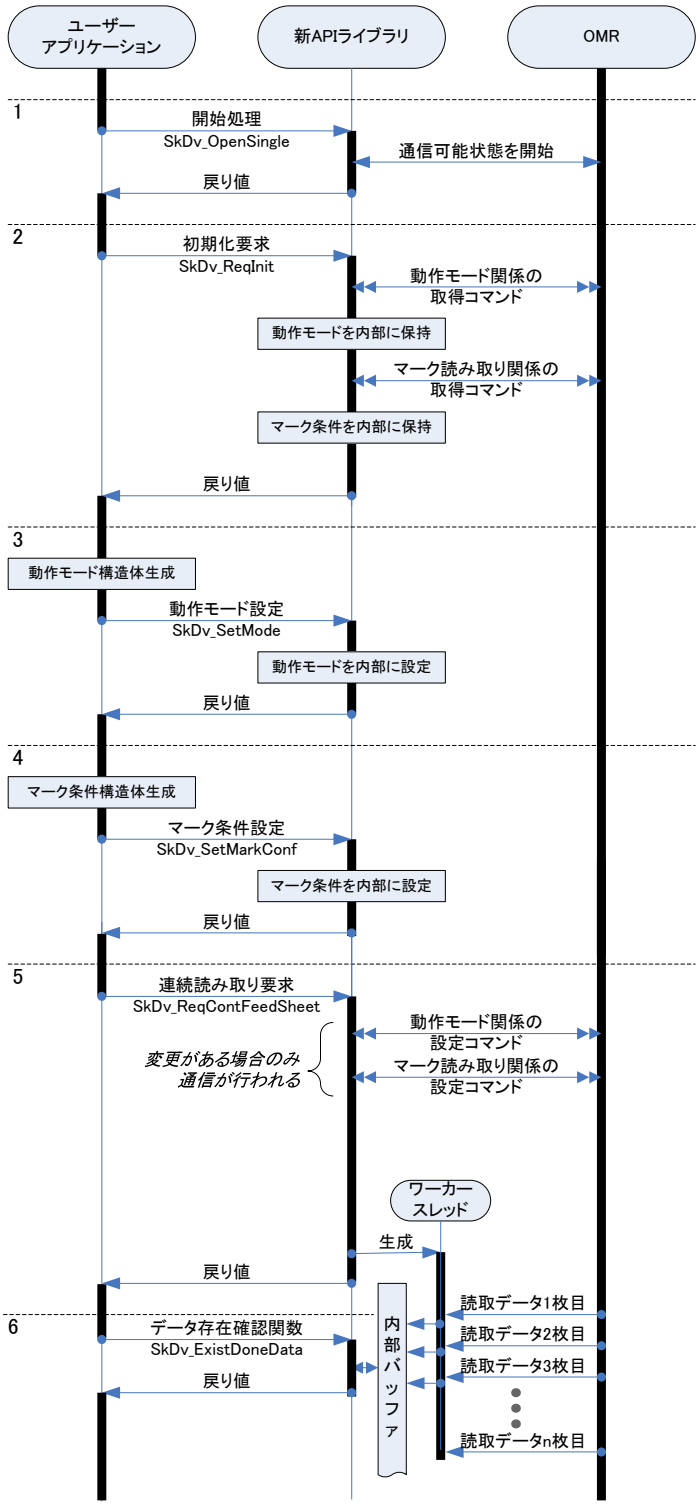
2. 初期化要求関数により、OMR に設定されている各種動作モード/各種マーク読み取り設定/各種イメージ読み取り設定の取得がおこなわれ、ライブラリー内にその内容が格納される。
これにより、OMR とライブラリーで各設定内容が一致している状態になる。

3. 動作モードを変更する場合には、動作モード構造体に各値を格納し、動作モード設定関数に引数で渡す。この時ライブラリー内に動作モードが格納される。

4. マーク読み取りの条件を変更する場合には、マーク条件構造体に各値を格納し、マーク条件設定関数に引数で渡す。この時ライブラリー内にマーク条件が格納される。

5. 帳票の読み取りを行うには、連続読み取り要求関数を呼ぶ。
ライブラリーは、動作モード/マーク条件が変更されている場合に各コマンドにより設定を行う。これにより、ライブラリーと OMR の設定が一致する。
その後、連続読み取りコマンドにより帳票の読み取りを行い、データを内部バッファに格納する。

6. データを使用するためには、データ存在確認関数を実行し、データが存在していることを確認する必要がある。



7. マークデータを得るには、データ取得準備関数を行う。データ取得準備関数を行うと、読みとった最初の帳票から順番にデータをバッファへ格納する。

8. マークデータを受け取る為のマーク情報構造体とマークデータを格納するバッファを用意し、連続読み取りマークデータ取得関数にて取得する。ライブラリーはマーク濃度データ要求コマンドにより受け取ったデータを返す。
受け取ったマークデータの処理を行う。

（2枚目以降は6～8を繰り返す）

9. 連続搬送完了の確認のために連続読み取り結果取得関数を実行する。

10. OMR の制御を終了するには終了処理関数を呼び、終了する。

（設定を変更するには3から行う）

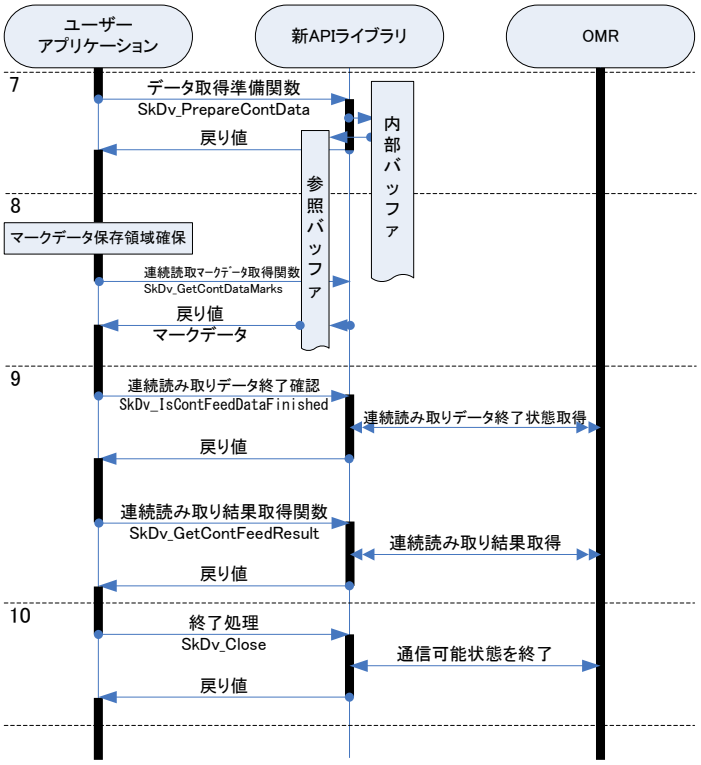


図 11 TM 数のチェックを行わずに帳票を読みとる処理の流れ

5.10.4. ID ウィンドウなしで TM 数が決まっている 1 種の帳票を読みとる場合

1. 開始処理関数により、OMR と通信可能な状態にする。

2. 初期化要求関数により、OMR に設定されている各種動作モード/各種マーク読み取り設定/各種イメージ読み取り設定の取得がおこなわれ、ライブラリー内にその内容が格納される。
これにより、OMR とライブラリーで各設定内容が一致している状態になる。

3. 動作モードを変更する場合には、動作モード構造体に各値を格納し、動作モード設定関数に引数で渡す。この時ライブラリー内に動作モードが格納される。

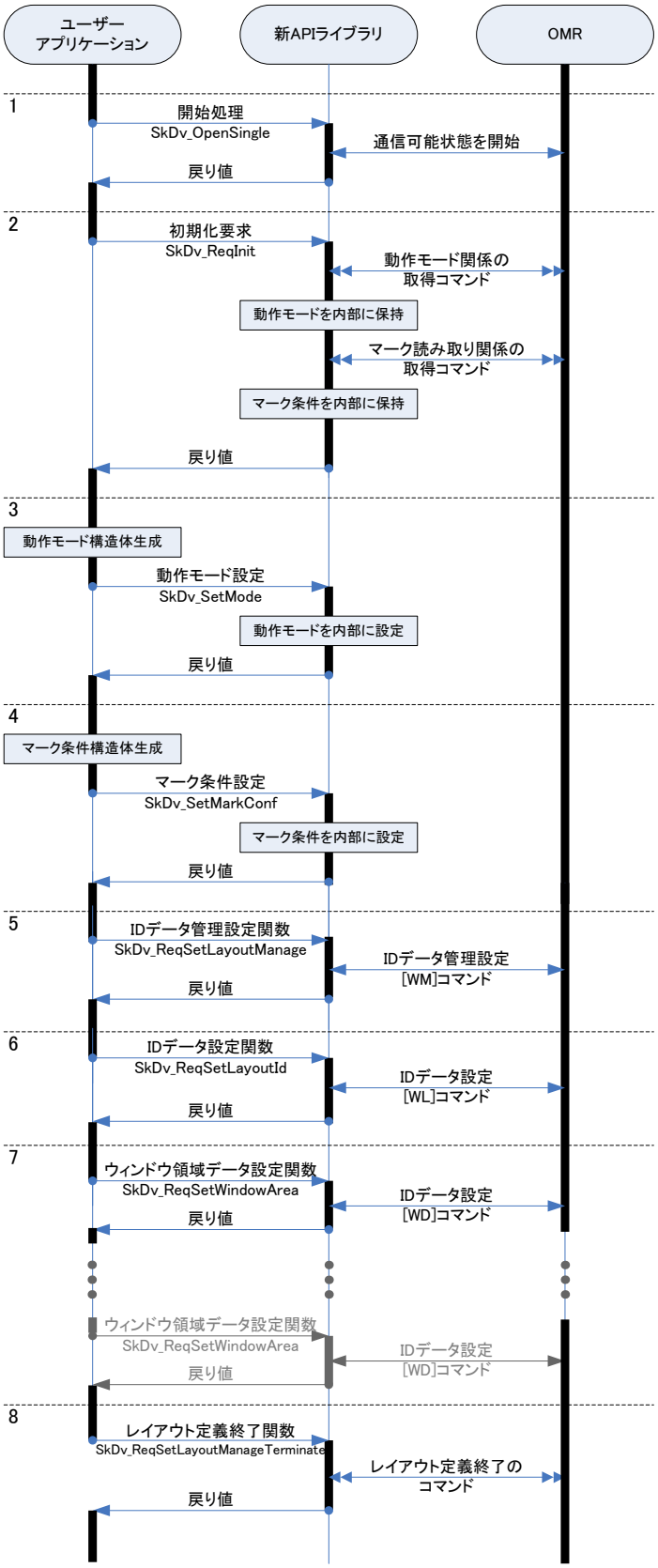
4. マーク読み取りの条件を変更する場合には、マーク条件構造体に各値を格納し、マーク条件設定関数に引数で渡す。この時ライブラリー内にマーク条件が格納される。

5. ウィンドウ設定を行う場合は、最初に ID データ管理設定関数を行う。引数に ID を使用しない設定を渡す。

6. ウィンドウ領域データ設定を行う前に ID データ設定関数を行う必要がある。ID を使用しない設定なので ID のパラメータは無視される。ここで読みとるタイミングマーク数を決定する。

7. ウィンドウ領域を設定する。
ウィンドウを設定する。直前に実行された ID データ設定関数で設定された ID にウィンドウが設定される。使用するウィンドウの数だけ設定する。

8. ウィンドウ設定を終了するときにはレイアウト定義終了関数を行う必要がある。



9. 帳票の読み取りを行うには、連続読み取り要求関数を呼ぶ。
ライブラリーは、動作モード/マーク条件が変更されている場合に各コマンドにより設定を行う。これにより、ライブラリーと OMR の設定が一致する。
その後、連続読み取りコマンドにより帳票の読み取りを行い、データを内部バッファ内に格納する。

10. データを使用するためには、データ存在確認関数を実行し、データが存在していることを確認する必要がある。

11. マークデータを得るには、データ取得準備関数を行う。データ取得準備関数を行うと、読みとった最初の帳票から順番にデータをユーザアプリへ渡す準備が行われる。

12. マークデータを受け取る為のマーク情報構造体とマークデータを格納するバッファを用意し、連続読み取りマークデータ取得関数にて取得する。ライブラリーはマーク濃度データ要求コマンドにより受け取ったデータを返す。
受け取ったマークデータの処理を行う。

（2枚目以降は10～12を繰り返す）

13. 連続搬送完了の確認のために連続読み取り結果取得関数を実行する。

14. OMR の制御を終了するには終了処理関数を呼び、終了する。
（設定を変更するには3から行う）
（ウィンドウ設定を変更するには5から行う。すべてのウィンドウがリセットされるので必要なウィンドウの数だけ繰り返す）

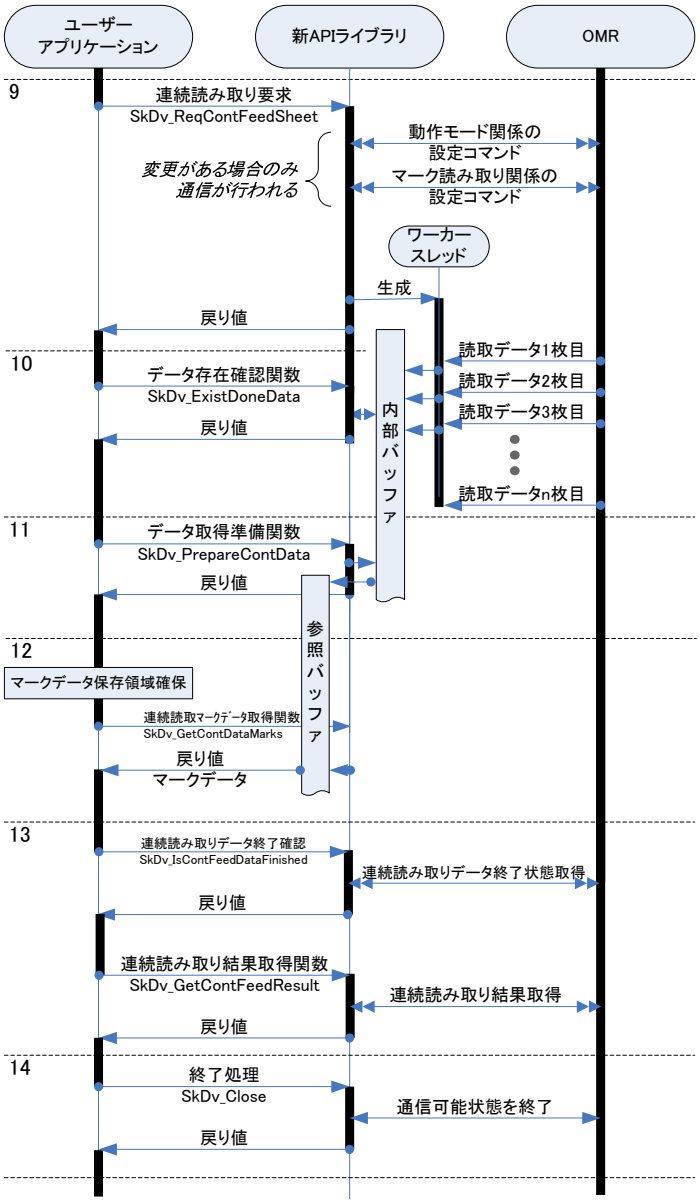


図 12 ID ウィンドウなしで TM 数が決まっている 1 種の帳票を読みとる処理の流れ

5.10.5. ID ウィンドウありで TM 数が決まっている数種の帳票を読みとる場合

1. 開始処理関数により、OMR と通信可能な状態にする。

2. 初期化要求関数により、OMR に設定されている各種動作モード/各種マーク読み取り設定/各種イメージ読み取り設定の取得がおこなわれ、ライブラリー内にその内容が格納される。
これにより、OMR とライブラリーで各設定内容が一致している状態になる。

3. 動作モードを変更する場合には、動作モード構造体に各値を格納し、動作モード設定関数に引数で渡す。この時ライブラリー内に動作モードが格納される。

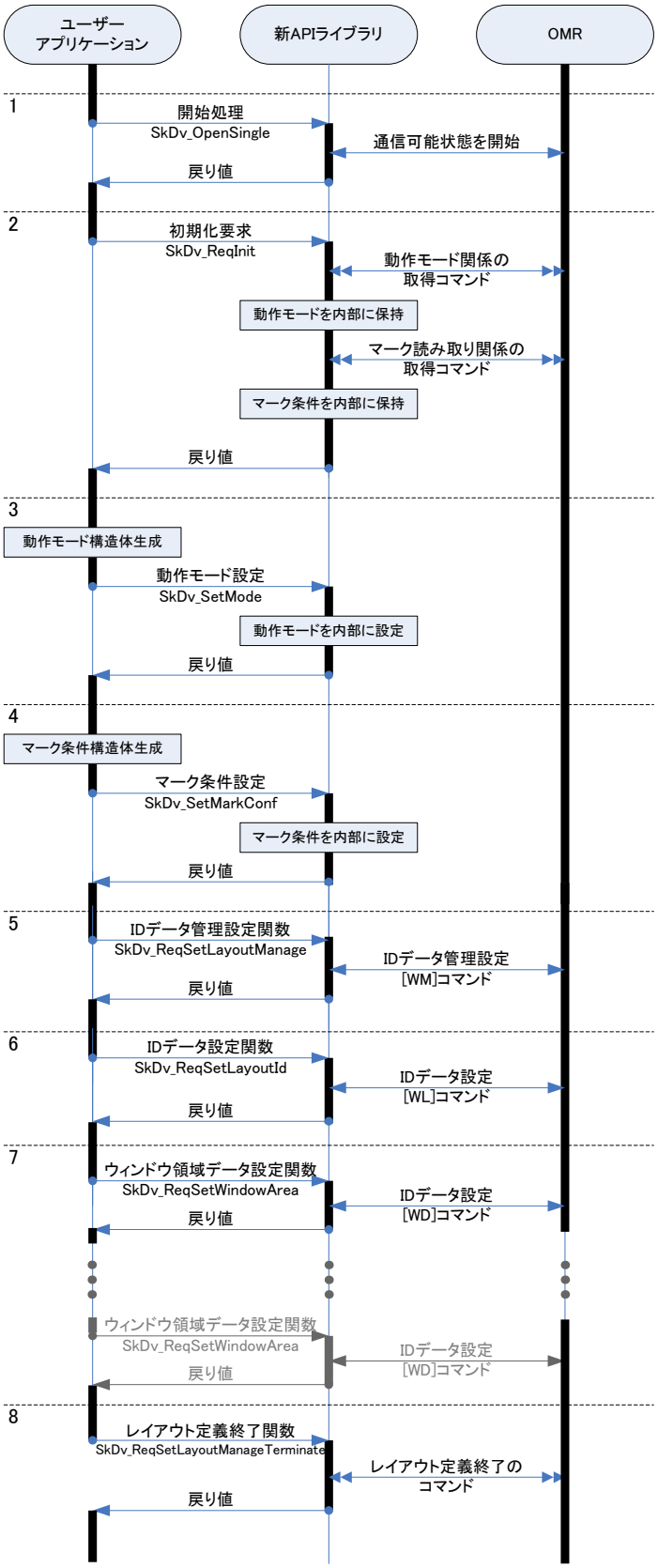
4. マーク読み取りの条件を変更する場合には、マーク条件構造体に各値を格納し、マーク条件設定関数に引数で渡す。この時ライブラリー内にマーク条件が格納される。

5. ウィンドウ設定を行う場合は、最初に ID データ管理設定関数を行う。ID を使用する場合は、ID データ管理設定関数の引数に ID を使用する設定を渡す。

6. ウィンドウ領域データ設定を行う前に ID データ設定関数を行う必要がある。ここで読みとるタイミングマーク数を設定する。使用する ID の数だけ設定する。

7. ウィンドウ領域を設定する。
ウィンドウを設定する。直前に実行された ID データ設定関数で設定された ID にウィンドウが設定される。使用するウィンドウの数だけ設定する。

8. ウィンドウ設定を終了するときにはレイアウト定義終了関数を行う必要がある。
（複数の ID を設定するには6，7を繰り返し、最後に8を行う）



9. 帳票の読み取りを行うには、連続読み取り要求関数を呼ぶ。
ライブラリーは、動作モード/マーク条件が変更されている場合に各コマンドにより設定を行う。これにより、ライブラリーと OMR の設定が一致する。
その後、連続読み取りコマンドにより帳票の読み取りを行い、データを内部バッファ内に格納する。

10. データを使用するためには、データ存在確認関数を実行し、データが存在していることを確認する必要がある。

11. マークデータを得るには、データ取得準備関数を行う。データ取得準備関数を行うと、読みとった最初の帳票から順番にデータをユーザアプリへ渡す準備が行われる。

12. マークデータを受け取る為のマーク情報構造体とマークデータを格納するバッファを用意し、連続読み取りマークデータ取得関数にて取得する。ライブラリーはマーク濃度データ要求コマンドにより受け取ったデータを返す。
受け取ったマークデータの処理を行う。

（2枚目以降は10～12を繰り返す）

13. 連続搬送完了の確認のために連続読み取り結果取得関数を実行する。

14. OMR の制御を終了するには終了処理関数を呼び、終了する。
（設定を変更するには3から行う）
（ウィンドウ設定を変更するには5から行う。すべてのウィンドウがリセットされるので必要なウィンドウの数だけ繰り返す）

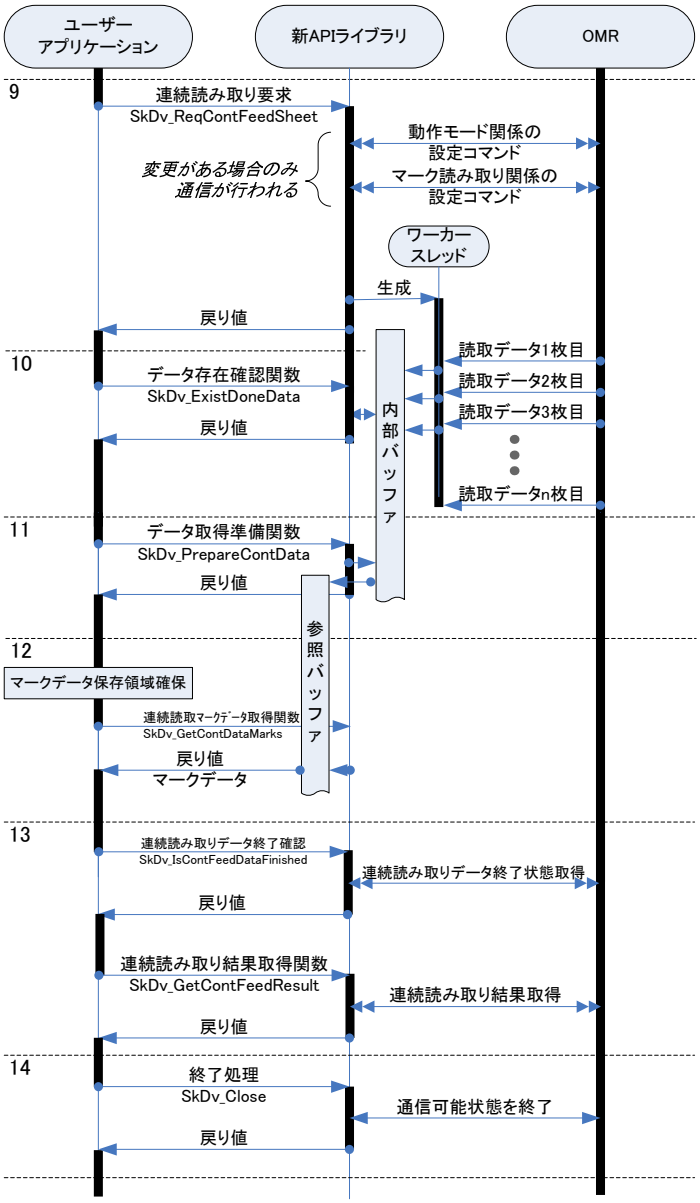


図 13 ID ウィンドウありで TM 数が決まっている数種の帳票を読みとる処理の流れ

5.11. 印字設定

5.11.1. 印字設定手順

SR-11000では印字を行うことが出来ます。印字を設定するためには複数の関数を実行する必要があります。その印字の設定手順は次の様になります。

① 現在の機体の印字条件を基に設定したい場合は印字条件取得要求関数(SkDv_ReqGetPrinterConf)と印字条件取得関数(SkDv_GetPrinterConf)を呼び、印字条件を取得します。
② 印字設定構造体(SK_DV_OPT_PRN_CONF)へ印字条件をセットします。
③ 印字条件設定関数(SkDv_SetPrinterConf)と印字条件設定要求関数(SkDv_ReqSetPrinterConf)を呼び、機体へ印字条件をセットします。
④ ～1 枚搬送(通常搬送)の場合～
印字文字を指定し、印字文字設定要求関数(SkDv_ReqPrintString)を呼び、印字文字を機体へセットします。

～連続搬送の場合～
固定印字文字を指定し、連続読み取り固定印字文字設定要求関数(SkDv_ReqSetPirntFixString)を呼び、固定印字文字を機体へセットします。
連続読み取り印字オプション構造体(SK_CONT_FEED_PRN_OPT)へ連番の書式の設定を行い、連続読み取り連番印字設定関数(SkDv_ReqSetContFeedPrint)を呼び、連番印字を機体へセットします。
(固定印字文字と連番印字は必ず両方必要なわけではなく、どちらかのみの設定でもかまいません。)

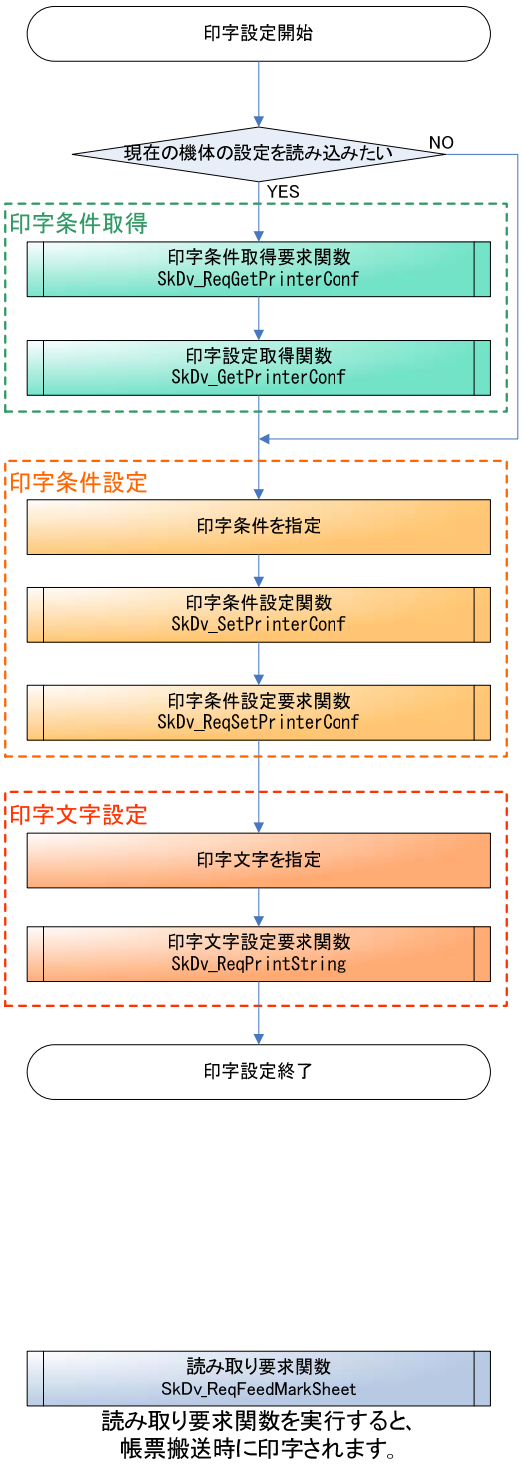
⑤ 搬送処理を開始すると、印字が行われます。
～1 枚搬送(通常搬送)の場合～
読み取り要求関数(SkDv_ReqFeedMarkSheet)

～連続搬送の場合～
連続読み取り要求関数(SkDv_ReqContFeedSheet)

5.11.2. 印字設定のフローチャート

印字設定手順をフローチャートであらわすと以下のようになります。

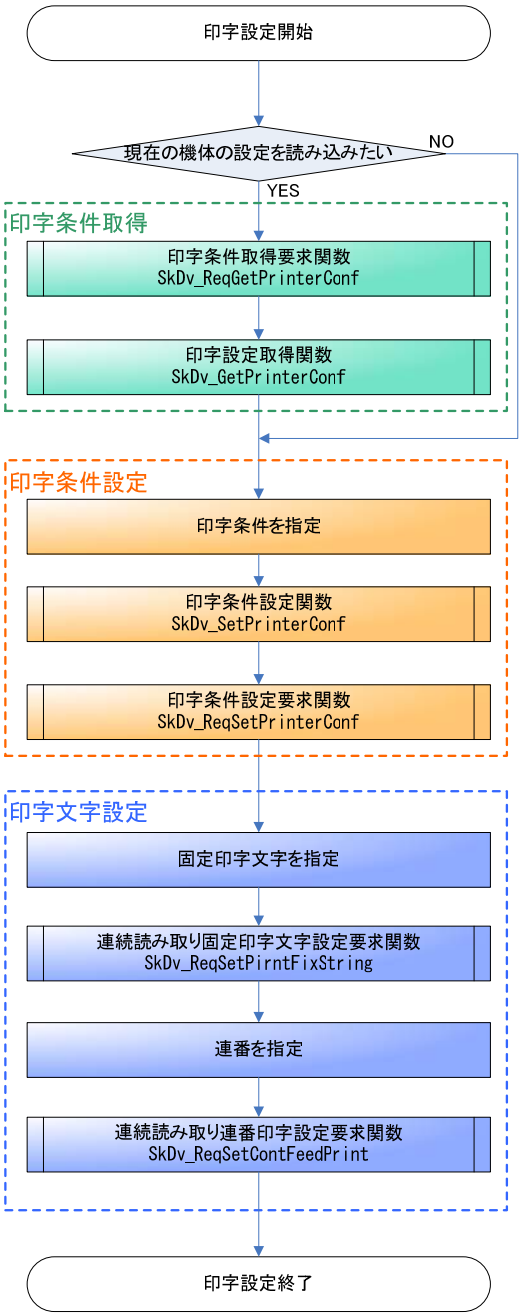
通常読み取りの場合



読み取り要求関数
SkDv_ReqFeedMarkSheet

読み取り要求関数を実行すると、
帳票搬送時に印字されます。

連続読み取りの場合



連続読み取り要求関数
SkDv_ReqContFeedSheet

連続読み取り要求関数実行すると、
帳票搬送時に印字されます。

5.11.3. 印字設定に関する構造体

スタッカのある機種やプリンターユニットのある機種では印字を行うことが出来ます。印字は細かな設定が可能で、あらゆる印字方法に応えるものとなっています。そのため、期待する印字を行うためには多くの変数の値を設定する必要があります。それらの変数は構造体内部へ指定します。印字設定に関する構造体は 2 種類、「SK_DV_OPT_PRN_CONF」「SK_CONT_FEED_PRN_OPT」があります。

「SK_DV_OPT_PRN_CONF」は印字有効無効/印字位置/印字方向/印字文字サイズ/印字文字間隔/印字方法の設定値をもつ構造体です。

「SK_CONT_FEED_PRN_OPT」は連番桁数/連番初期値の設定値を持つ構造体です。

構造体を使用する関数		
SkDv_ReqSetContFeedPrint		
構造体名	変数名	意味
SK_DV_OPT_PRN_CONF	iEnable	印字の[使用する]/[使用しない]を設定します。 [使用する]の場合は印字設定を使用します。 [使用しない]の場合は印字設定を無視します。
	iStartPos	印字設定(印字開始位置)を設定します。印字開始位置とは帳票の先端が基準で、0.0～355.0mm の範囲で設定できます。実際の印字では帳票先端に 3.0mm の余白が確保されるため、0.0mm を設定した場合、3.0mm から印字開始されます。
	iOrientation	印字設定(印字方向)を設定します。[通常印字]/[180°回転印字]の 2 種類の設定値があります。
	iFontSize	印字設定(印字文字サイズ)を設定します。印字文字サイズとは、印字文字の幅のことで、3.2～96.0mm の範囲(SR-11000 の場合は 4.0～96.0mm)で印字文字サイズを設定します。
	iFontSpace	印字設定(印字文字間隔)を設定します。印字文字間隔とは文字と文字の間のスペースのことで、0.8～92.0mm の範囲で設定できます。
	iPrintMode	印字方法を設定します。[読み取り後印字]/[読み取りながら印字]の 2 種類の設定値があります。

構造体を使用する関数		
SkDv_ReqSetContFeedPrint		
構造体名	変数名	意味
SK_CONT_FEED_PRN_OPT	iDigits	連番の桁数を指定します。1～8 桁まで指定できます。
	iStartNumber	連番初期値を指定します。連番桁数以上の値は指定できません。

5.11.4. 印字設定(印字文字)

印字は 42 文字(SR-11000 は 20 文字)まで設定することが可能です。印字可能な文字は 163 種あります。印字文字設定要求関数(SkDv_ReqPrintString)関数で設定可能です。ここで設定された印字文字は読み取り要求関数(SkDv_ReqFeedMarkSheet)実行時に印字されます。連続読み取り開始要求関数(SkDv_ReqContFeedSheet)実行時に印字する場合は、連続読み取り固定印字文字設定要求関数(SkDv_ReqSetPirntFixString)と連続読取連番印字設定関数(SkDv_ReqSetContFeedPrint)をご使用ください。

印字可能文字

	文字数	文字
数字	10	0123456789
アルファベット大文字	26	ABCDEFGHIJKLMNOPQRSTUVWXYZ
アルファベット小文字	26	abcdefghijklmnopqrstuvwxyz
カタカナ	55	アイウエオカキクケコサシスセソタチツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワヲンｱｨｳｴｵｬｭｮｯ
記号	46	SP(スペース)!”#$%&’ ()*+,ー./:;<=>?@[¥]^_ ■{}～○×△□。「」、・ﾞﾟ

5.11.5. 印字設定(印字開始位置)

印字開始位置は 0.0～355.0mm の範囲で設定できる、帳票の印字の起点となる位置を指定する設定値です。設定は 1.0mm ピッチ(0, 1.0, 2.0,…354.0, 355.0mm)で指定可能です。帳票の先端には余白 3.0mm をが確保されるため、実際に帳票に印字される位置は 3.0mm から となります。
また、帳票の後端にも余白 3.0mm が確保されます。そのため、帳票が短い時に印字開始位置を大きな値に設定すると、正常に印字できない可能性があります。実際の印字範囲は下図のようになります。

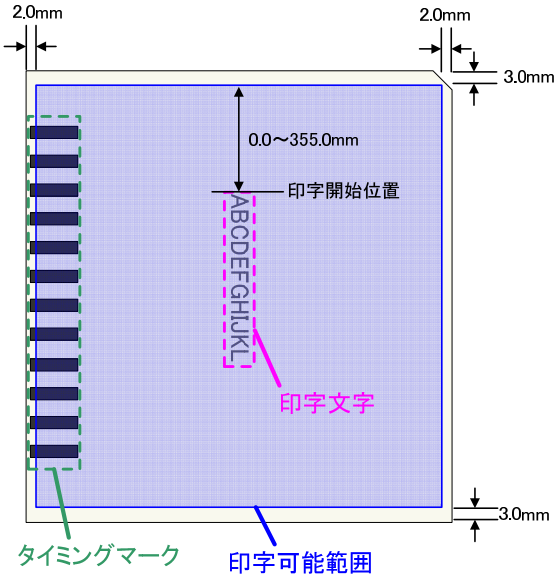


図 14 印字可能範囲

また、次の 3 つのすべての条件を満たすとき、印字可能範囲が通常とは異なります。

・プリンターユニットを使用して印字をする
・印字方法が[読み取り後印字]である
・帳票の長さが 233.0mm 以上、

この場合、印字開始位置の基準が((帳票の長さ)-230.0)の位置となります。

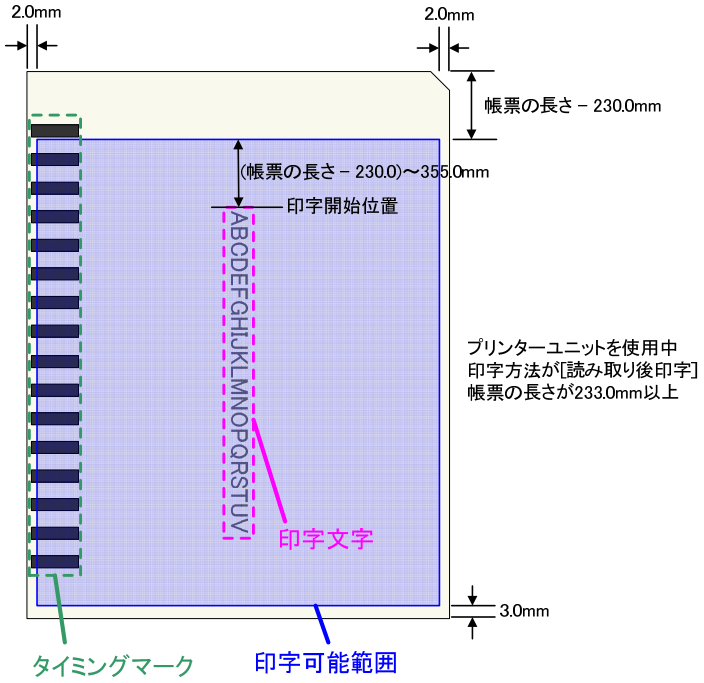


図 15 一定条件下の印字可能範囲

5.11.6. 印字設定(印字方向)

印字設定(印字方向)は [通常印字]/[180° 回転印字]の 2 種類の設定値があります。[通常印字]は帳票のタイミングマーク側を文字の底辺側として印字されます。[180° 回転印字]では帳票のタイミングマーク側が文字の頂点側として印字されます。

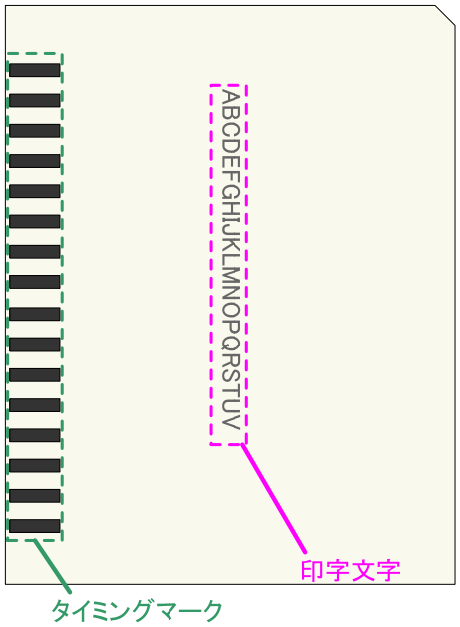


図 16 通常印字

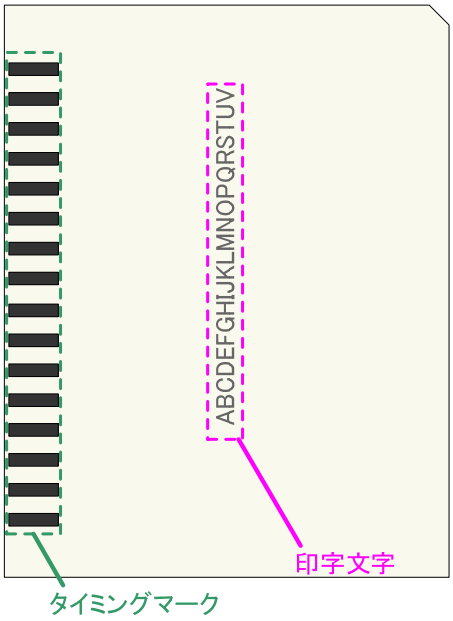
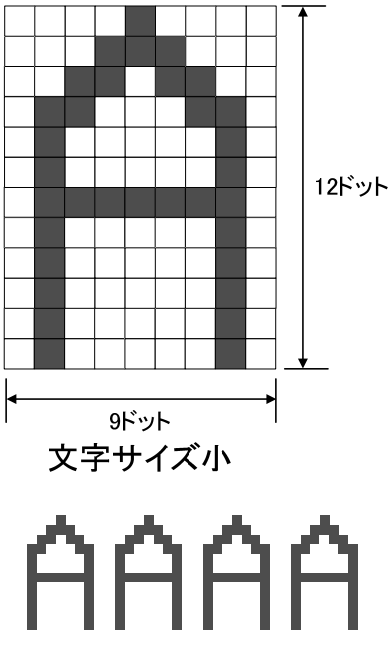


図 17 180° 回転

5.11.7. 印字設定(印字文字サイズ)

印字文字のサイズ(幅)は 3.2～96.0mm の範囲(SR-11000 の場合は 4.0～96.0mm)で設定します。設定は 0.8mm ピッチ(3.2, 4.0, 4.8,…95.2, 96.0mm)で指定可能です。印字文字はドットによって描かれ、文字に使用されるドット数は固定です。そのため、印字文字サイズを大きくすることによってドットの間隔が広くなり印字文字の色薄く感じられるようになります。



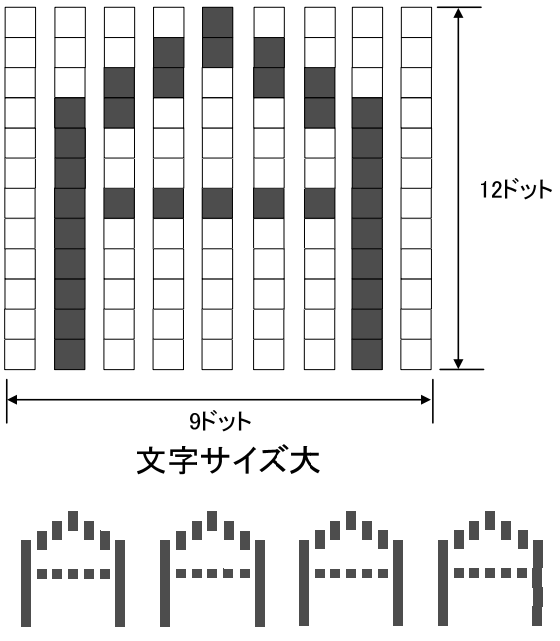


図 18 文字サイズが小の印字

図 19 文字サイズが大の印字

5.11.8.　　印字設定(印字文字間隔)

印字文字間隔とは文字と文字の間のスペースのことで、0.8～92.0mm の範囲で設定できます。設定は 0.1mm ピッチ(0.8, 0.9, 1.0,…94.9, 95.0mm)で指定可能です。

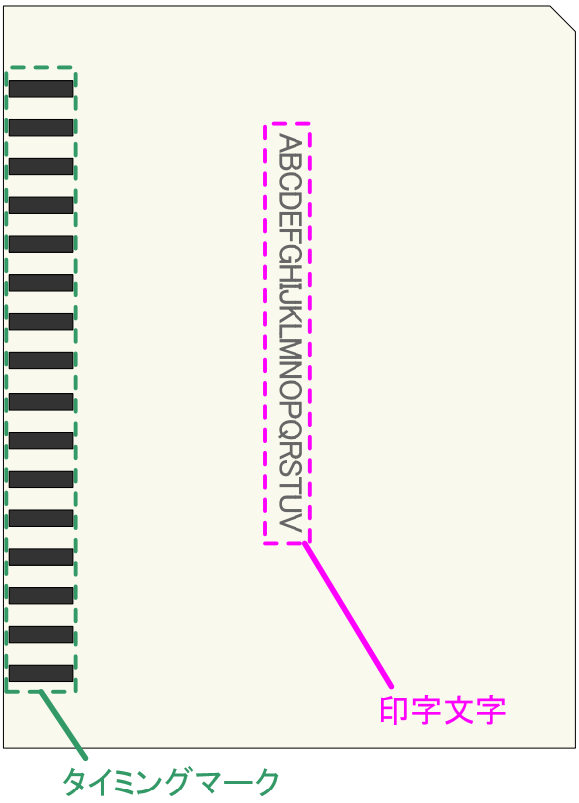


図 20 印字文字間隔 狭

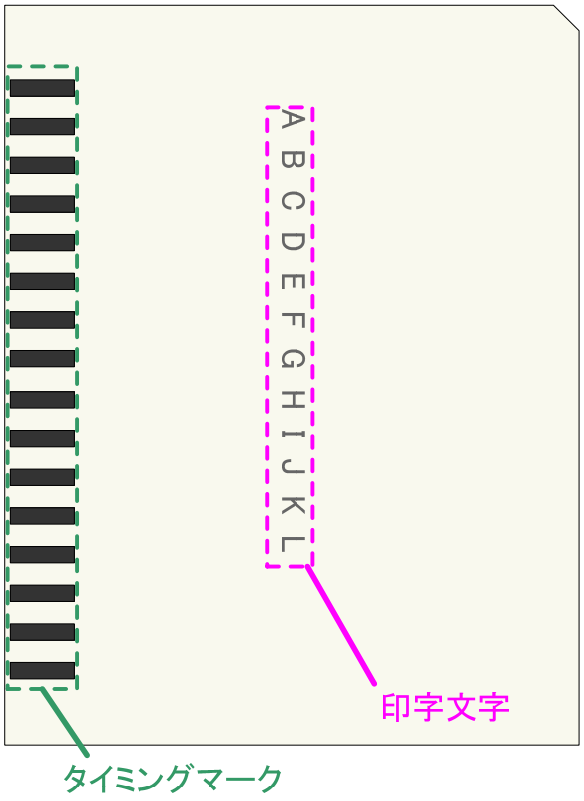


図 21 印字文字間隔 広

5.11.9.　　印字設定(固定印字･連番)

連続搬送時の印字は、固定印字+連番が印字されます。固定印字は毎回帳票に印字される文字です。連番は帳票を搬送するたびに値が 1 ずつ増える番号です。連番は指定桁数内の最大の数字になった場合は、0 から印字されます。

例)指定桁数 3 桁の時

… → 998 → 999 → 000 → 001 → …

固定印字の設定は連続読み取り固定印字文字設定要求関数(SkDv_ReqSetPirntFixString)で行います。連番の設定は連続読取連番印字設定関数(SkDv_ReqSetContFeedPrint)で行います。固定印字と連番は必ずどちらも必要というわけではないので、印字の利用状況に応じてご使用ください。

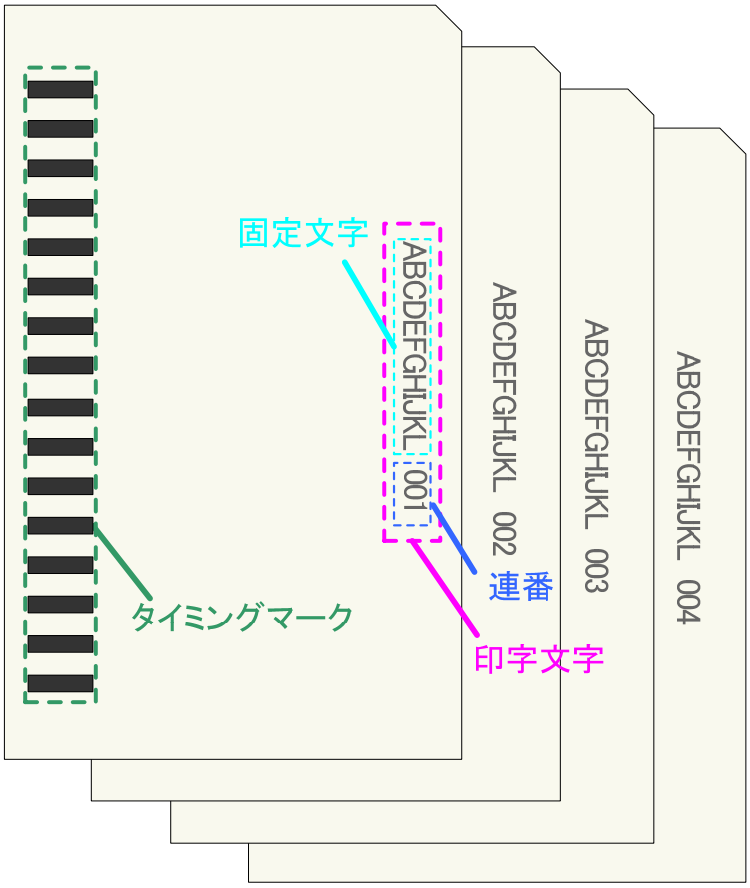


図 22 固定印字 + 連番のイメージ

5.12. レイアウト設定

5.12.1. レイアウト設定手順

SR-11000 ではレイアウトの設定を行うことが出来ます。レイアウトを設定するためには複数の関数を実行する必要があります。そのレイアウトの設定手順は次の様になります。

⑥ ID データ管理設定関数(SkDv_ReqSetLayoutManage)を呼び、ID のレイアウトを設定します。(帳票のどの範囲を ID とするか、ID をどのように読むかを決定します。)
⑦ ID データ設定関数(SkDv_ReqSetLayoutId)を呼び、ID を設定します。(ID の値をいくつにするか、タイミングマークの数はいくつか、エラー時の動作はどうするかを決定します。)
⑧ ウィンドウ領域データ設定関数(SkDv_ReqSetWindowArea)を呼び、ウィンドウを設定する。(帳票のどの範囲をウィンドウとするか、ウィンドウをどのように読むかを決定します。)
⑨ ③を繰り返し、1 つの ID に含まれるすべてのウィンドウを設定します。(ウィンドウは 1 つに ID に付き最大 80 個まで設定可能です。)
⑩ ②～④を繰り返し、必要なすべての ID を設定します。(ID は最大 16 個までとなります。)
⑪ レイアウト設定終了関数(SkDv_ReqSetLayoutManageTerminate)を呼び、レイアウト設定を終了します。

5.12.2. レイアウト設定手順のフローチャート

レイアウト設定手順をフローチャートであらわすと以下のようになります。

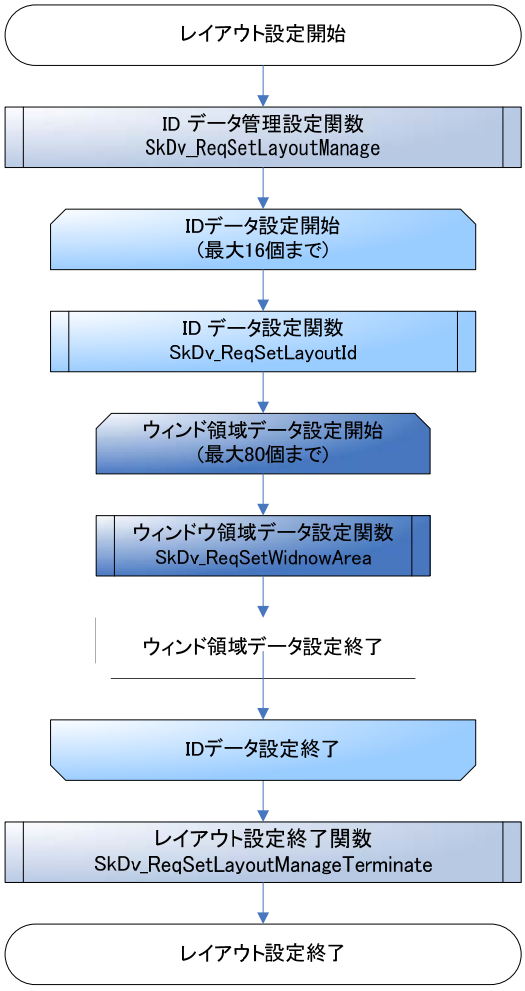


図 23 レイアウト設定手順のフローチャート

5.12.3. レイアウト設定に関する構造体

レイアウトは細かな設定が可能で、あらゆる読み取り要求に応えるものとなっています。そのため、期待する読み取りを行うためには多くの変数の値を設定する必要があります。それらの変数は構造体内部へ指定します。レイアウト設定に関する構造体は大きく分けて 3 種類、「SK_LAYOUT_MANAGE_CONF」「SK_LAYOUT_ID_CONF」「SK_WINDOW_AREA_CONF」があります。
「SK_LAYOUT_MANAGE_CONF」は ID をどう使用するかの設定値をもつ構造体です。
「SK_LAYOUT_ID_CONF」は ID 値をもつ構造体です。
「SK_WINDOW_AREA_CONF」はウィンドウをどう使用するかの設定値をもつ構造体です。

以下に構造体の変数とその意味をまとめます。

構造体を使用する関数		
SkDv_ReqSetLayoutManage		
構造体名	変数名	意味
SK_LAYOUT_MANAGE_CONF	blEnableId	レイアウトの[使用する]/[使用しない]を設定します。 [使用する]の場合はレイアウト設定を使用します。 [使用しない]の場合はレイアウト設定を無視します。(ただし、タイミングマークによるエラーチェックは行われます。タイミングマーク数の設定は SK_LAYOUT_ID_CONF の LayoutIdPrm によって行います。)
	IdWindowPrm	SK_WINDOW_PRM 構造体を使用して、[帳票の面]/[ID の範囲]/[ID の読み取り方法]/[ID の読み取り方向]/[読み取り感度レベル・濃度差レベル]を設定します。
	dwNgAction	ID が NG だったときの動作を設定します。 [セレクトスタッカへ排紙]/[搬送停止]の 2 種類設定できます。
	LayoutOpt	SK_LAYOUT_OPTION 構造体を使用して、ID のレイアウトのオプションを設定します。[オプションなし]/[(※)ID 順チェック]の 2 種類設定できます。

(※)ID 順チェックとは搬送された帳票が設定された ID の順序でなかった場合をエラーとする設定値です。
例①)ID が 3 つ設定されており、それが 0,1,2 の時。(下線部太字でエラー)

[搬送順序]
OK 0→1→2
NG 0→**2** ID1 をとばしているためエラーとなります
NG **1** ID0 をとばしているためエラーとなります

例②)ID が 4 つ設定されており、それが 3,5,7,9 の時。(下線部太字でエラー)

[搬送順序]
OK 3→5→7→9
NG 3→5→**9** ID7 をとばしているためエラーとなります
NG **5** ID3 をとばしているためエラーとなります

構造体を使用する関数		
SkDv_ReqSetLayoutId		
構造体名	変数名	意味
SK_LAYOUT_ID_CONF	LayoutIdPrm	SK_LAYOUT_ID_PRM 構造体を使用して、[ID ウィンドウのマークパターンの指定]/[タイミングマーク数]を指定できます。 ここでタイミングマーク数を設定することよって、異なったタイミングマーク数の帳票が搬送されたときにエラーとすることが出来ます。
	iReject	未使用。(0 固定にすること。)
	LayoutOpt	SK_LAYOUT_OPTION 構造体を使用して、ID ウィンドウのレイアウトのオプションを設定します。[無効(指定なし)]/[マークをマスク]/固定マーク]/[チェックディジット]/[範囲チェック(昇順)]/[範囲チェック(降順)]/[範囲チェック(順番なし)]/[マスク設定(パーティション共通]が設定できます。

構造体を使用する関数		
SkDv_ReqSetWindowArea		
構造体名	変数名	意味
SK_WINDOW_AREA_CONF	WindowPrm	SK_WINDOW_PRM 構造体を使用して、[帳票の面]/[ID の範囲]/[ID の読み取り方法]/[ID の読み取り方向]/[読み取り感度レベル・濃度差レベル]を設定します。
	WindowCheck	SK_WINDOW_CHECK 構造体を使用して、[ノーマーク許可設定]/[マーク数最小]/[マーク数最大]を設定します。これを設定することによってウィンドウ内のマーク個数が正しいかどうか判定することが出来ます。 [ノーマーク許可設定]が[許可する]の時、ウィンドウ内にマークがない場合でもエラーとなりません。[許可しない]の時は、ウィンドウ内にマークがない場合はエラーとします。 [マーク数最小]/[マーク数最大]を設定した場合、ウィンドウ内のマーク数が[マーク数最小]未満だった場合、ウィンドウ内のマーク数が[マーク数最大]より大きかった場合はエラーとします。
	dwNgAction	ウィンドウが NG だったときの動作を設定します。 [セレクトスタッカへ排紙]/[搬送停止]の 2 種類設定できます。
	LayoutOpt	SK_LAYOUT_OPTION 構造体を使用して、ウィンドウのレイアウトのオプションを設定します。[無効(指定なし)]/[マークをマスク]/固定マーク]/[チェックディジット]/[範囲チェック(昇順)]/[範囲チェック(降順)]/[範囲チェック(順番なし)]/[マスク設定(パーティション共通]が設定できます。

5.12.4. レイアウト ID の指定

レイアウトを設定するためには、レイアウトに個別の ID を指定する必要があります。ID はレイアウト ID 設定構造体(SK_LAYOUT_ID_PRM)の ucIdData に指定します。ucIdData は 8 バイトの配列で、ビットの 1 が ID のマーク有り、ビットの 0 が ID のマークなしとなります。

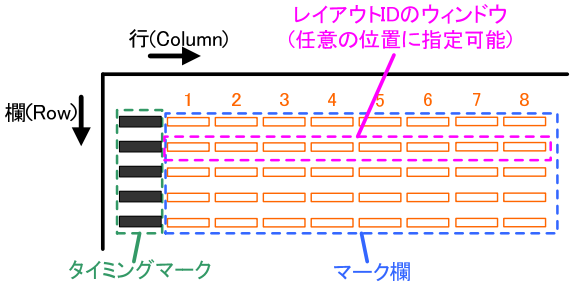


図 24 レイアウト ID と帳票のイメージ

	ucIdData							
	1Byte	2Byte	3Byte	4Byte	5Byte	6Byte	7Byte	8Byte
2 進数	1000	1000	0000	0000	0000	0000	0000	0000
16 進数	A	A	0	0	0	0	0	0

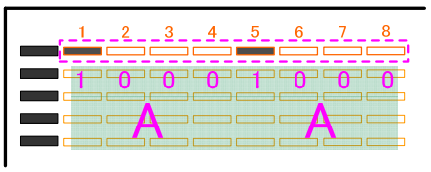


図 25 ID 設定例 1

	ucIdData							
	1Byte	2Byte	3Byte	4Byte	5Byte	6Byte	7Byte	8Byte
2 進数	0001	1100	0000	0000	0000	0000	0000	0000
16 進数	2	C	0	0	0	0	0	0

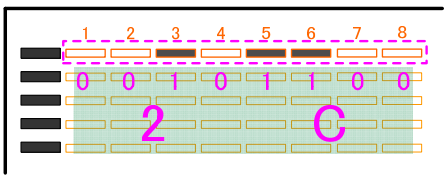


図 26 ID 設定例 2

ID の位置は帳票の隅だけでなく、ID データ管理構造体(SK_LAYOUT_MANAGE_CONF)内の Col 構造体と Row 構造体の値を変更することで位置や範囲を変更することが出来ます。詳しい設定方法は 5.12.6 レイアウトの範囲指定(切り出し)をご覧ください。

(※)ID のマークが同じ位置にあっても、ID の位置や範囲が異なると別の ID として認識します。

	開始位置 (Start)	マーク数 (Number)	間隔 (Step)
行(Column)	1	10	1
欄(Row)	3	1	1

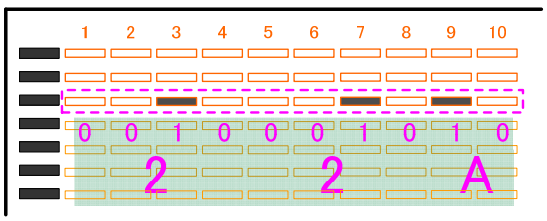


図 27 位置を変更した ID 設定例 1

	開始位置 (Start)	マーク数 (Number)	間隔 (Step)
行(Column)	3	8	1
欄(Row)	3	1	1

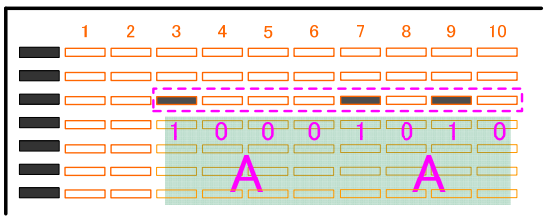
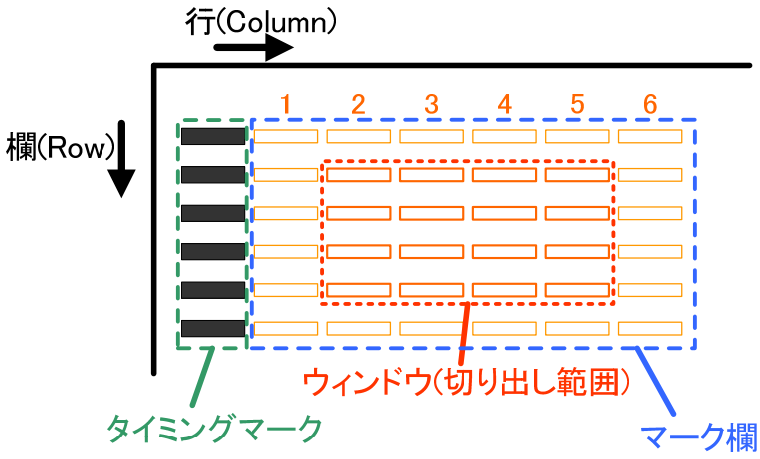


図 28 位置を変更した ID 設定例 2

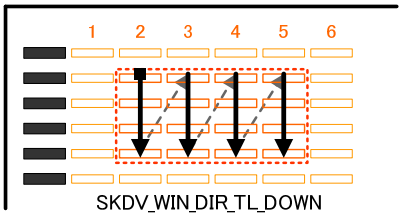
5.12.5. レイアウト読み取り方向

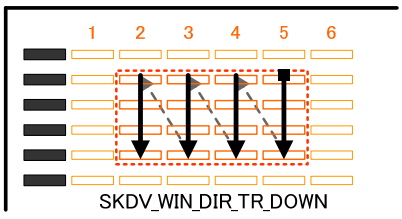
レイアウト設定時、レイアウトの読み取り方向を設定することが出来ます。ウィンドウ設定構造体（SK_WINDOW_PRM）の iDirection 変数の値によって設定できます。

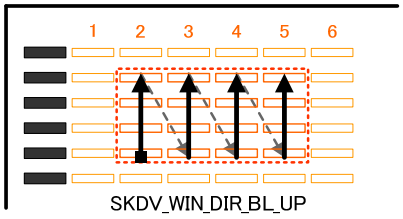


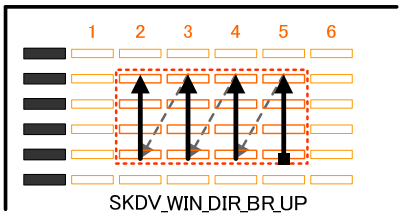
iDirection には 8 種類の定数があり、それぞれ次の様に定義されています。

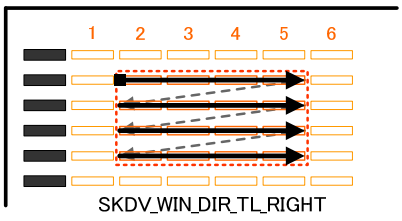
定数名	値	意味
SKDV_WIN_DIR_TL_DOWN	0	左上から下へ
SKDV_WIN_DIR_TR_DOWN	1	右上から下へ
SKDV_WIN_DIR_BL_UP	2	左下から上へ
SKDV_WIN_DIR_BR_UP	3	右下から上へ
SKDV_WIN_DIR_TL_RIGHT	4	左上から右へ
SKDV_WIN_DIR_TR_LEFT	5	右上から左へ
SKDV_WIN_DIR_BL_RIGHT	6	左下から右へ
SKDV_WIN_DIR_BR_LEFT	7	右下から左へ

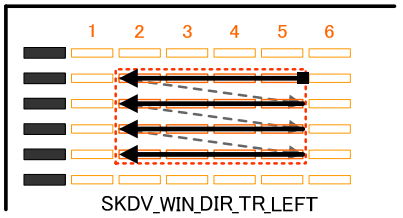


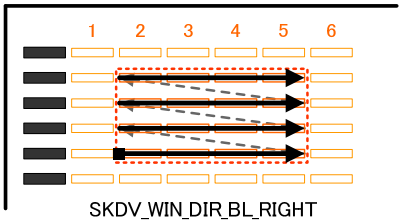


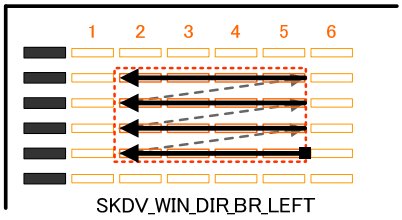












レイアウト読み取り方向が異なる場合、ID のマークが同じ位置にあっても、別の ID として認識します。

	開始位置 (Start)	マーク数 (Number)	間隔 (Step)
行(Column)	1	8	1
欄(Row)	3	1	1

レイアウト読み取り方向	左上から右

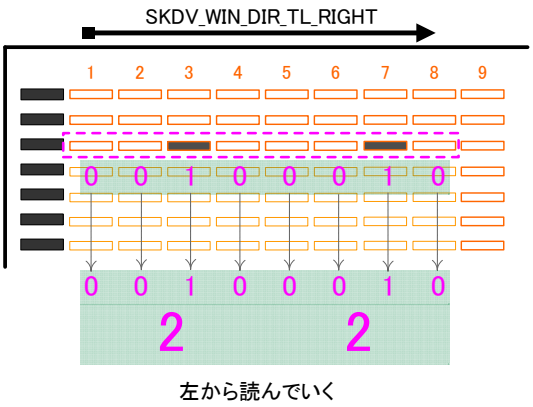


図 29 位置を変更した ID 設定例 1

	開始位置 (Start)	マーク数 (Number)	間隔 (Step)
行(Column)	1	8	1
欄(Row)	3	1	1

レイアウト読み取り方向	右上から左

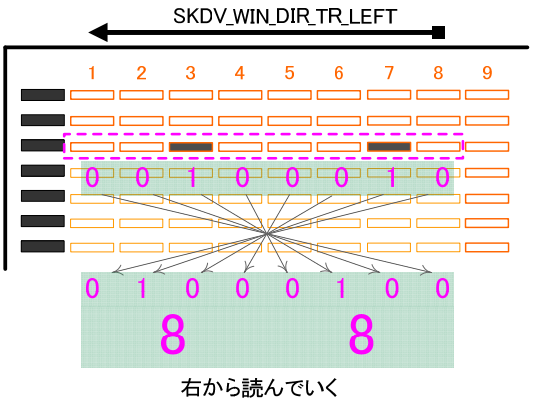


図 30 位置を変更した ID 設定例 2

5.12.6. レイアウトの範囲指定(切り出し)

SR-11000 では読みとり範囲を指定(切り出す)することが出来ます。これをレイアウト設定と言い、切り出された範囲をウィンドウと言います。

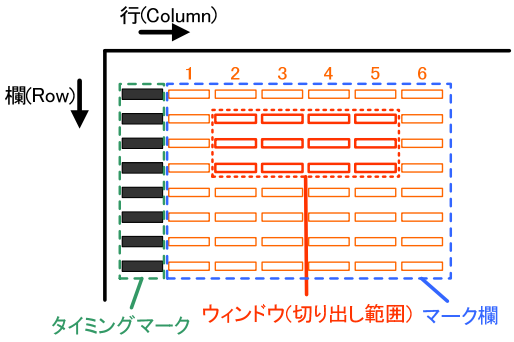


図 31 ウィンドウと帳票のイメージ

以下にレイアウト範囲指定の例を示します。

範囲指定はウィンドウ設定構造体(SK_WINDOW_PRM)によって行います。

	開始位置 (Start)	マーク数 (Number)	間隔 (Step)
行 (Column)	2	4	1
欄(Row)	2	3	1

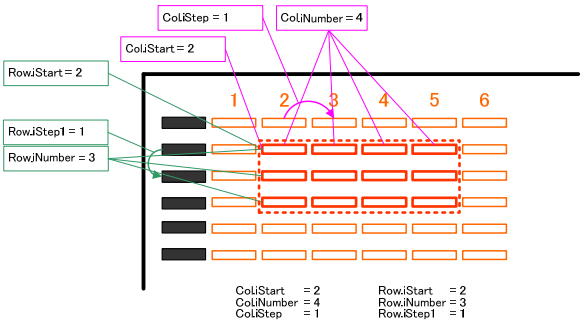


図 32 ウィンドウ設定例 1

ひとかたまりの範囲だけでなく、間隔を設けた範囲も設定できます。

	開始位置 (Start)	マーク数 (Number)	間隔 (Step)
行(Column)	1	6	1
欄(Row)	3	3	2

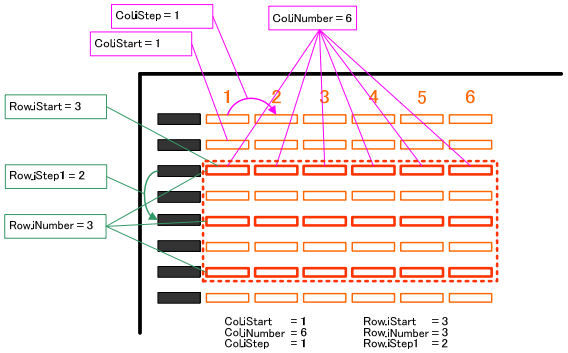


図 33 ウィンドウ設定例 2

行・欄をそれぞれ間隔開けた範囲も設定できます。

	開始位置 (Start)	マーク数 (Number)	間隔 (Step)
行(Column)	2	3	2
欄(Row)	2	3	3

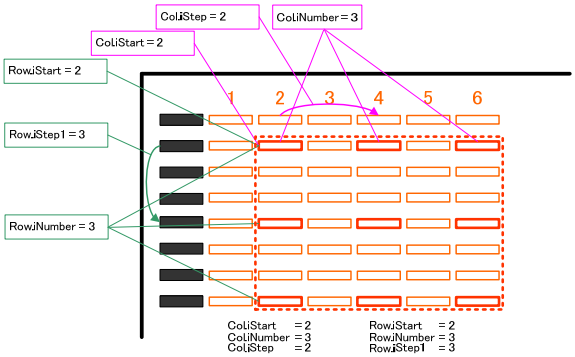
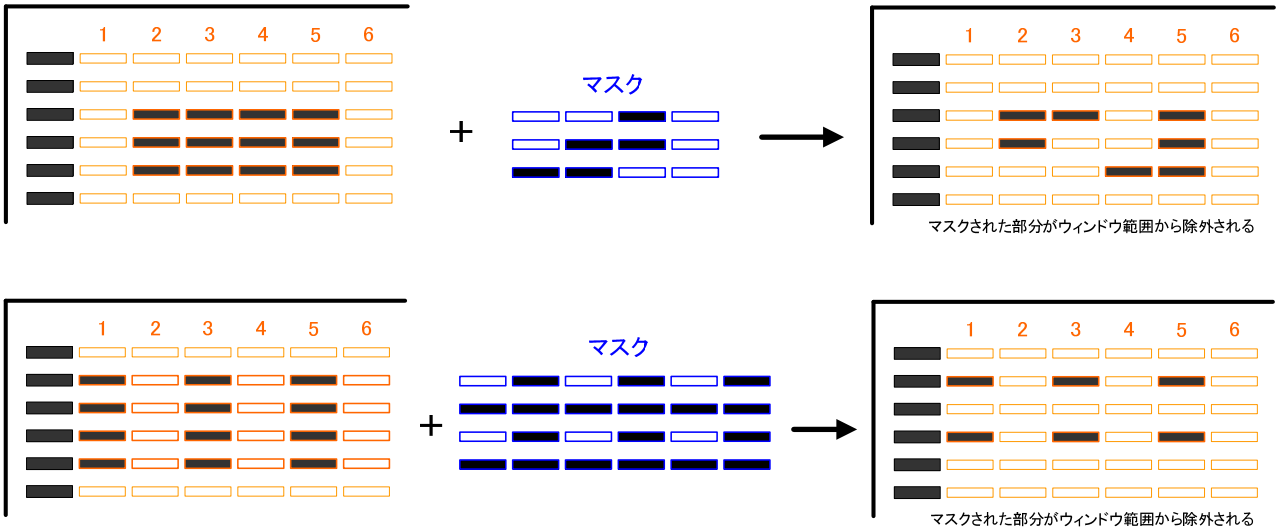


図 34 ウィンドウ設定例 3

5.12.7. マスク設定

ウィンドウではマスク設定を利用して読み取りパターンを指定することが出来ます。これによって規則性をもった範囲指定だけではなく、柔軟な指定方法も可能となる。
レイアウトオプション構造体(SK_LAYOUT_OPTION)の Type 変数で SKDV_LAYOUT_OPT_MASK 指定することでマスク設定を利用できます。



5.12.8. 感度レベル･濃度差レベル

ウィンドウ内のマークの読み取りの条件として感度レベル･濃度差レベルを設定することが出来ます。感度レベル･濃度差レベルを適切に設定することで、精確なマーク認識を実現します。
感度レベル･濃度差レベルはウィンドウごとに設定できます。

次の 2 つの条件に該当するとすべてのマークを認識します。

(1) 濃度レベルが感度レベル以上のマーク
(2) 濃度レベルが（ウィンドウ内の最大マーク濃度レベル）－（濃度差レベル）を超えるマーク

具体的な例を以下に記します。

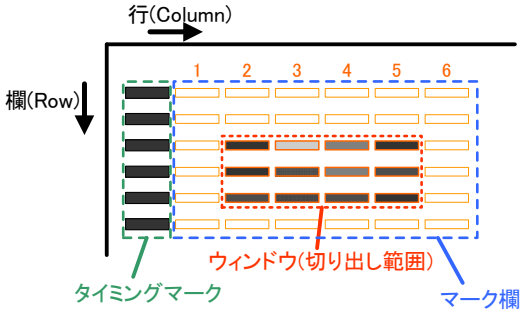


図 35 ウィンドウ内マーク記入イメージ

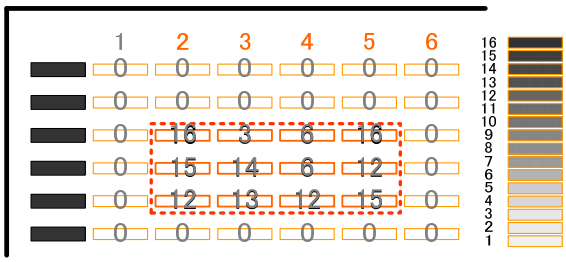


図 36 記入マーク濃度レベル

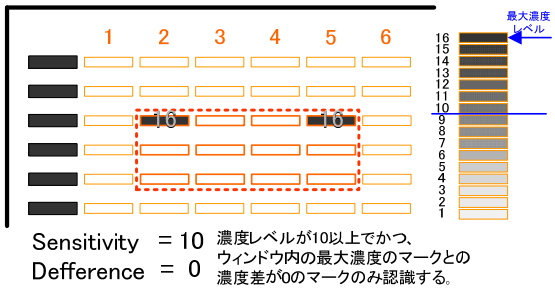


図 37 感度ﾚﾍﾞﾙ10、濃度差ﾚﾍﾞﾙ0の時のﾏｰｸ認識

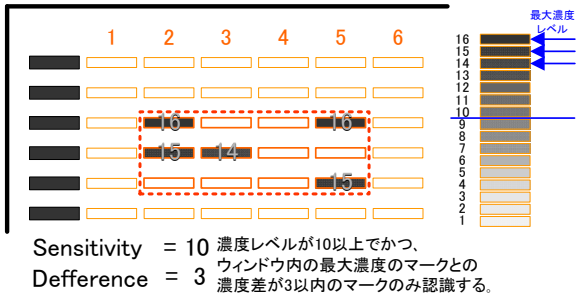


図 38 感度ﾚﾍﾞﾙ10、濃度差ﾚﾍﾞﾙ3の時のﾏｰｸ認識

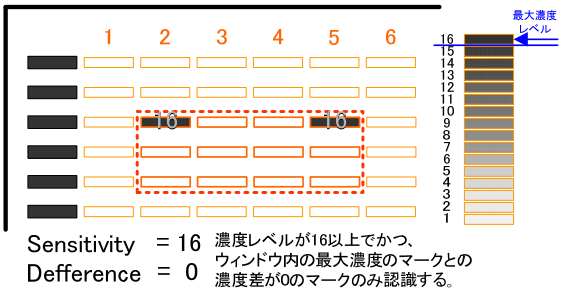


図 39 感度ﾚﾍﾞﾙ16、濃度差ﾚﾍﾞﾙ0の時のﾏｰｸ認識

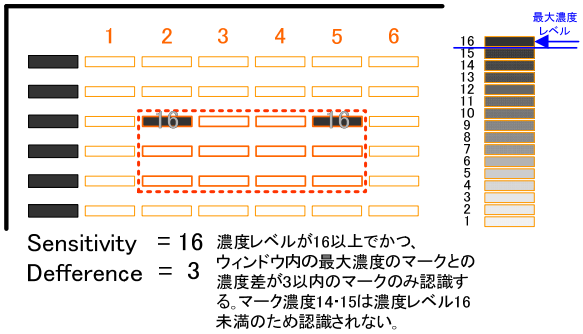


図 40 感度ﾚﾍﾞﾙ16、濃度差ﾚﾍﾞﾙ3の時のﾏｰｸ認識

6.API リファレンス(旧 API ライブラリー)

「OMRAPI.DLL」の API リファレンスです。
SR-3500/6000/6500 を USB コマンドにより制御することができます。
また、SR-3500 HYBRID のイメージ機能以外を USB コマンドにより制御することができます。

ひとつの API 関数で、USB コマンドを最大で1つ実行します。 USB コマンドによる通信の有無や使用するコマンドについては、各 API の説明をお読みください。

6.1. API 関数一覧

No.	API 関数	内容
1	OMR_OpenDeviceUSB	USB デバイスをオープンする
2	OMR_CloseDevice	USB デバイスハンドラをクローズする
3	OMR_GetLastError	OMR_STATUS の値を取得する
4	OMR_FormatMessage	OMR_STATUS の値を文字列に変換する
5	OMR_GetST	OMR_STATUS の値を取得する
6	OMR_SetNumberOfColumnsToRead	読み取り行数を設定する
7	OMR_GetNumberOfColumnsToRead	読み取り行数を取得する
8	OMR_SetReadingMethod	マーク読み取り方式を設定する
9	OMR_GetReadingMethod	マーク読み取り方式を取得する
10	OMR_SetBackSensorUnit	裏面読み取りユニットの使用/未使用を設定する
11	OMR_GetBackSensorUnit	裏面読み取りユニットの使用/未使用を取得する
12	OMR_SetSheetThickness	シートの連量を設定する
13	OMR_GetSheetThickness	シートの連量を取得する
14	OMR_SetWarningError	ワーニング発生の条件を設定する
15	OMR_GetWarningError	ワーニング発生の条件設定を取得する
16	OMR_SetPanelConfig	パネル操作の有効／無効を設定する
17	OMR_GetPanelConfig	パネル操作の有効／無効の情報を取得する
18	OMR_SetBuzzerConfig	ブザーの音量と音質を設定する
19	OMR_GetBuzzerConfig	ブザーの設定情報を取得する
20	OMR_SetID	識別コードを設定する
21	OMR_GetID	識別コードを取得する
22	OMR_SetSleepTime	スリープ/スタンバイ時間を設定する
23	OMR_GetSleepTime	スリープ/スタンバイ時間を取得する
24	OMR_Reset	電源投入時の状態にする
25	OMR_FeedSheet	シートを一枚搬送させる
26	OMR_MoveHopper	ホッパの上昇/下降を行なう
27	OMR_EjectSheet	シートの排紙動作を行なう
28	OMR_InitialSetting	各設定値を初期化する
29	OMR_CancelError	ステータス情報を取得する
30	OMR_GetMarks	マーク情報を取得する
31	OMR_GetStatus	ステータス情報を取得する
32	OMR_GetSensorInfo	ポジションセンサの ON/OFF 情報を取得する
33	OMR_GetDeviceInfo	接続されているユニットを確認する
34	OMR_GetMachineName	機種名を取得する
35	OMR_GetVersion	ファームウェアバージョンを取得する
36	OMR_SetPrinterUnit	プリンタ制御を設定する
37	OMR_GetPrinterUnit	プリンタ制御を取得する

No.	API 関数	内容
38	OMR_SetPrintOrder	印字順序を設定する
39	OMR_GetPrintOrder	印字方法を取得する
40	OMR_SetPrintMode	印字方法を設定する
41	OMR_GetPrintMode	印字方法を取得する
42	OMR_SetPrintPosition	印字位置を設定する
43	OMR_GetPrintPosition	印字位置を取得する
44	OMR_SetPrintAngle	印字方向を設定する
45	OMR_GetPrintAngle	印字方向を取得する
46	OMR_SetPrintFontSize	文字サイズを設定する
47	OMR_GetPrintFontSize	文字サイズを取得する
48	OMR_SetPrintFontPitch	印字文字間隔を設定する
49	OMR_GetPrintFontPitch	印字文字間隔を取得する
50	OMR_SetPrintString	バッファに文字列を設定する
51	OMR_GetPrintString	バッファの文字列を取得する
52	OMR_GetBarcodeInfo	読み取ったバーコードの枚数情報を取得する
53	OMR_GetBarcodeData	読み取ったバーコードデータを取得する
54	OMR_SetBarcodeReaderUnit	バーコードの読み取り許可/禁止を設定する
55	OMR_GetBarcodeReaderUnit	バーコードの読み取り許可/禁止を取得する
56	OMR_SetBarCodeArea	バーコードの読み取りエリアを設定する
57	OMR_GetBarCodeArea	バーコードの読み取りエリアを取得する
58	OMR_SetBarCodeReadType	読み取るバーコードの種類を設定する
59	OMR_GetBarCodeReadType	読み取るバーコードの種類を取得する
60	OMR_SetBarCodeCheckDigit	各バーコードのチェックディジットを設定する
61	OMR_GetBarCodeCheckDigit	各バーコードのチェックディジットを取得する
62	OMR_SetBarCodeOption	各バーコードの個別設定を行う
63	OMR_GetBarCodeOption	各バーコードの個別設定を取得する

6.2. 定数

6.2.1. ステータスコード

API の戻り値は OMR_STATUS 型が基本となります。
OMR_STATUS は、

```
typedef unsigned int OMR_STATUS;
```

と定義されており、以下の定数が格納されます。

定数名	実際の値	内容
SR_SUCCESS	0x00000000	正常
SR_UNSUCCESSFUL	0x00000001	エラー終了(詳細不明)
SR_DISCONNECTED	0x00000002	OMR 未接続
SR_WRONG_PARAMETER	0x00000003	パラメータが不正
SR_MEMORY_ERROR	0x00000004	DLL 内でメモリ領域の確保に失敗
SR_TIMEOUT	0x00000005	通信タイムアウト
SR_RECEIVE_NAK	0x00000006	OMR から NAK を受信
SR_WRONG_RESPONSE	0x00000007	レスポンス形式が不正
SR_ERROR_STATUS_A1	0x00010000	本体:メモリエラー1
SR_ERROR_STATUS_A2	0x00010001	本体:メモリエラー2
SR_ERROR_STATUS_A3	0x00010002	本体:ホッパ駆動エラー
SR_ERROR_STATUS_A4	0x00010003	本体:ダウンロードエラー
SR_ERROR_STATUS_A5	0x00010004	本体:センサタイプエラー
SR_ERROR_STATUS_A6	0x00010005	本体:オプションエラー
SR_ERROR_STATUS_A7	0x00010006	本体:スタッカユニット未接続エラー
SR_ERROR_STATUS_A8	0x00010007	本体:電源電圧エラー
SR_ERROR_STATUS_B1F	0x01020000	表面読み取りユニット:通信回線エラー
SR_ERROR_STATUS_B2F	0x01020001	表面読み取りユニット:内部通信エラー
SR_ERROR_STATUS_B3F	0x01020002	表面読み取りユニット:メモリエラー
SR_ERROR_STATUS_B4F	0x01020003	表面読み取りユニット:補正値エラー
SR_ERROR_STATUS_B5F	0x01020004	表面読み取りユニット:ダウンロードエラー
SR_ERROR_STATUS_B6F	0x01020005	表面読み取りユニット:内部エラー
SR_ERROR_STATUS_B7F	0x01020006	表面読み取りユニット:バージョンエラー
SR_ERROR_STATUS_B1B	0x02020000	裏面読み取りユニット:通信回線エラー
SR_ERROR_STATUS_B2B	0x02020001	裏面読み取りユニット:内部通信エラー
SR_ERROR_STATUS_B3B	0x02020002	裏面読み取りユニット:メモリエラー
SR_ERROR_STATUS_B4B	0x02020003	裏面読み取りユニット:補正値エラー
SR_ERROR_STATUS_B5B	0x02020004	裏面読み取りユニット:ダウンロードエラー
SR_ERROR_STATUS_B6B	0x02020005	表面読み取りユニット:内部エラー
SR_ERROR_STATUS_B7B	0x02020006	表面読み取りユニット:バージョンエラー
SR_ERROR_STATUS_C1	0x03030000	バーコードユニット:通信回線エラー
SR_ERROR_STATUS_C2	0x03030001	バーコードユニット:内部通信エラー
SR_ERROR_STATUS_C3	0x03030002	バーコードユニット:メモリエラー
SR_ERROR_STATUS_C4	0x03030003	バーコードユニット:センサエラー
SR_ERROR_STATUS_C5	0x03030004	バーコードユニット:ダウンロードエラー
SR_ERROR_STATUS_C6	0x03030005	バーコードユニット:内部エラー
SR_ERROR_STATUS_C7	0x03030006	バーコードユニット:バージョンエラー
SR_ERROR_STATUS_D1	0x04040000	プリンターユニット:通信回線エラー

定数名	実際の値	内容
SR_ERROR_STATUS_D2	0x04040001	プリンターユニット:内部通信エラー
SR_ERROR_STATUS_D3	0x04040002	プリンターユニット:メモリエラー
SR_ERROR_STATUS_D4	0x04040003	プリンターユニット:ダウンロードエラー
SR_ERROR_STATUS_D5	0x04040004	プリンターユニット:内部エラー
SR_ERROR_STATUS_D6	0x04040005	プリンターユニット:バージョンエラー
SR_ERROR_STATUS_E1	0x05050000	スタッカユニット:通信回線エラー
SR_ERROR_STATUS_E2	0x05050001	スタッカユニット:内部通信エラー
SR_ERROR_STATUS_E3	0x05050002	スタッカユニット:メモリエラー
SR_ERROR_STATUS_E4	0x05050003	スタッカユニット:ダウンロードエラー
SR_ERROR_STATUS_E5	0x05050004	スタッカユニット:内部エラー
SR_ERROR_STATUS_E6	0x05050005	スタッカユニット:バージョンエラー
SR_ERROR_STATUS_F1	0x1f060000	(予約)
SR_ERROR_STATUS_F2	0x1f060001	(予約)
SR_ERROR_STATUS_F3	0x1f060002	(予約)
SR_ERROR_STATUS_F4	0x1f060003	(予約)
SR_ERROR_STATUS_F5	0x1f060004	コマンドエラー
SR_ERROR_STATUS_F6	0x1f060005	パラメータエラー
SR_ERROR_STATUS_F7	0x1f060006	プロトコルエラー
SR_ERROR_STATUS_G1	0x20070000	本体:カバーオープン
SR_ERROR_STATUS_G2	0x25070001	スタッカユニット:カバーオープン
SR_ERROR_STATUS_H1	0x30080000	本体:ノーフィード
SR_ERROR_STATUS_H2	0x30080001	本体:給紙検知部ジャム
SR_ERROR_STATUS_H3	0x30080002	本体:読み取り開始検知部ジャム
SR_ERROR_STATUS_H4	0x30080003	本体:排紙検知部ジャム
SR_ERROR_STATUS_I1	0x35090000	スタッカユニット:プリンタ印字開始検知部ジャム
SR_ERROR_STATUS_I2	0x35090001	スタッカユニット:メイン排紙部ジャム
SR_ERROR_STATUS_I3	0x35090002	スタッカユニット:セレクト排紙部ジャム
SR_ERROR_STATUS_P1	0x42100000	裏面読み取りユニット:未接続
SR_ERROR_STATUS_P2	0x43100001	バーコードユニット:未接続
SR_ERROR_STATUS_P3	0x44100002	プリンターユニット:未接続
SR_ERROR_STATUS_P4	0x45100003	スタッカユニット:未接続
SR_ERROR_STATUS_Q1	0x40110000	シートエンプティ
SR_ERROR_STATUS_Q2	0x40110001	ダブルフィードエラー
SR_ERROR_STATUS_Q3	0x40110002	左端スキューエラー
SR_ERROR_STATUS_Q4	0x40110003	マークスキューエラー
SR_ERROR_STATUS_R1	0x40120000	本体:ホッパ緊急停止
SR_ERROR_STATUS_R2	0x40120001	本体:引き抜きエラー
SR_ERROR_STATUS_R3	0x40120002	本体:シート挿入タイムアウト
SR_ERROR_STATUS_R4M	0x40120003	表面/裏面読み取りユニット:タイミングマークエラー
SR_ERROR_STATUS_R4F	0x41120003	表面読み取りユニット:タイミングマークエラー
SR_ERROR_STATUS_R4B	0x42120003	裏面読み取りユニット:タイミングマークエラー
SR_ERROR_STATUS_R5M	0x40120004	表面/裏面読み取りユニット:読み取り設定エラー
SR_ERROR_STATUS_R5F	0x41120004	表面読み取りユニット:読み取り設定エラー
SR_ERROR_STATUS_R5B	0x42120004	裏面読み取りユニット:読み取り設定エラー
SR_ERROR_STATUS_S1F	0x41130000	表面読み取りユニット:白レベルエラー
SR_ERROR_STATUS_S2F	0x41130001	表面読み取りユニット:黒レベルエラー
SR_ERROR_STATUS_S1B	0x42130002	裏面読み取りユニット:白レベルエラー
SR_ERROR_STATUS_S2B	0x42130003	裏面読み取りユニット:黒レベルエラー
SR_ERROR_STATUS_S3	0x40130002	本体:読み取り開始センサ汚れエラー

定数名	実際の値	内容
SR_ERROR_STATUS_T1	0x40140000	本体:給紙検知部残紙
SR_ERROR_STATUS_T2	0x40140001	本体:読み取り開始検知部残紙
SR_ERROR_STATUS_T3	0x40140002	本体:排紙検知部残紙
SR_ERROR_STATUS_T4	0x45140003	スタッカユニット:プリンタ印字開始検知部残紙
SR_ERROR_STATUS_T5	0x45140004	スタッカユニット:メイン排紙部残紙
SR_ERROR_STATUS_T6	0x45140005	スタッカユニット:セレクト排紙部残紙
SR_ERROR_TERM	0xffffffff	未定義のステータス情報

OMR_STATUS の定数規則
原則として SR_SUCCESS を除き、より小さい数値の方が重要度の高いエラーを示します。
ステータス情報は表面読み取りユニット、裏面読み取りユニットの 2 種類があります。より重要度の高い方が選択され、重要度が同じ場合は表面読み取りユニットのステータス情報が優先されます。

Bit31 : : : : Bit28	重要度(4Bits) 0x0 :ハードエラー(解除不能) 0x1 :通信エラー 0x2 :カバーオープン 0x3 :ジャム発生 0x4 :警告/操作ミス
Bit27 : : : : : : Bit24	発生個所(4Bits) 0x0 :本体 0x1 :表面センサユニット 0x2 :裏面センサユニット 0x3 :バーコードユニット 0x4 :プリンターユニット 0x5 :スタッカユニット 0xf :その他
Bit23 : : : : Bit16	ページ番号(8Bits) ステータス情報 1 桁目のアルファベット別にページを割り振る 0x00 :通信状態でのエラー(ステータス情報を取る前にエラー発生) 0x01 :ステータス情報 1 桁目＝A : 0x1A :ステータス情報 1 桁目＝Z
Bit15 : Bit0	通し番号(16Bits) ページ番号ごとの通し番号

6.2.2. BOOL 型の値

API の戻り値や引数で BOOL 型を使用している場合、「Microsoft Visual C++」の場合には int 型(32bit)に定義されています。「Microsoft Visual C++」以外の言語でアプリケーションを作成する場合には注意してください。
本書にて使用している TRUE/FALSE の値は以下の通りです。

定数名	値	意味
TRUE	1	真。正常、成功など。
FALSE	0	偽。異常、失敗など。

BOOL 型とその値である TRUE と FALSE は以下のように定義されます。

	Visual C++ (本マニュアル)	Visual C#	Visual Basic 6.0	Visual Basic .net
BOOL 型 (32bit)	int	int	Long	Int32

6.3. API 関数 ー システム制御

6.3.1. OMR_OpenDeviceUSB

処理	USB に接続された OMR デバイスを検索し、そのデバイスをオープンする。	
プロトタイプ	OMR_STATUS OMR_OpenDeviceUSB(void)	
引数	なし	
戻り値	SR_SUCCESS	成功
	SR_UNSUCCESSFUL	失敗(オープンできるデバイスがない、又は既に接続されている。)
詳細	複数の OMR デバイスが接続されている場合、内部での管理で先頭の OMR デバイスと接続される。	

6.3.2. OMR_CloseDevice

処理	OMR_OpenDeviceUSB 関数でオープンしたデバイスハンドラをクローズする。 アプリケーションを終了するときに実行すること。	
プロトタイプ	OMR_STATUS OMR_CloseDevice(void)	
引数	なし	
戻り値	SR_SUCCESS	成功
	SR_UNSUCCESSFUL	失敗

6.3.3. OMR_GetLastError

処理	多くの制御 API は戻り値では成功／失敗しか取得できない。失敗した場合、原因を探る 1 つの手段としてこの関数を使用すると最後に記録された OMR_STATUS 値が取得できる。
プロトタイプ	OMR_STATUS OMR_GetLastError(void)
引数	なし
戻り値	最後に記録された OMR_STATUS 値
詳細	OMR_STATUS は typedef unsigned int OMR_STATUS と型が定義されており、格納される値は「定数」の項を参照。 OMR_OpenDeviceUSB/OMR_CloseDevice/OMR_GetLastError 関数を実行しても OMR_STATUS 値は記録されない。

6.3.4. OMR_FormatMessage

処理	OMR_STATUS の値を文字列に変換する。	
プロトタイプ	CHAR *OMR_FormatMessage(OMR_STATUS status, int iLanguageFlag)	
引数	status	変換する OMR_STATUS 値
	iLanguageFlag	出力言語指定 SR_STRING_NORMAL :指定無し(英語) SR_STRING_ENGLISH :英語(only ASCII Code) SR_STRING_JAPANESE :日本語(Shift-JIS Code)
戻り値	変換した文字列へのポインタ(固定値)	
詳細	変換結果は OMR_STATUS 定数の表を参照 使用例 OMR_GetLastError 関数と組み合わせて以下のような使い方が可能 printf(OMR_FormatMessage(OMR_GetLastError(), SR_STRING_NORMAL));	

6.3.5. OMR_GetST

処理	直前に受信したレスポンス中のステータス情報をそのまま出力する。	
プロトタイプ	const CHAR *OMR_GetST(int iPage)	
引数	iPage	SR_PAGE_FRONT :ST1 の情報を指定 SR_PAGE_BACK :ST2 の情報を指定
戻り値	ステータス情報文字列へのポインタ(固定値)。2バイトの文字列。 文字列が空の場合(文字列長が 0)はデータがないか、引数が不正	
詳細	使用例 printf(“%s”, OMR_GetST(SR_PAGE_FRONT));	

6.4. API 関数 ー 設定

6.4.1. OMR_SetNumberOfColumnsToRead

処理	読み取り行数設定コマンド NC を使用して、引数で指定された値を読み取り行数として設定する。	
プロトタイプ	BOOL OMR_SetNumberOfColumnsToRead(int iColumns)	
引数	iColumns	設定する行数設定する。0 を指定すると初期値に戻る。
戻り値	TRUE	成功
	FALSE	失敗
詳細	読み取り行数の設定範囲は OMR 本体に搭載されている読み取りユニットの種別によって異なる。	

読み取りユニットのセンサピッチ	設定範囲	初期値
1/6 inch	1～48	48
0.2 inch	1～40	40
0.2 inch S	1～40	40
0.25 inch	1～33	33
0.3 inch	1～27	27
0.3 inch F	1～24	24

6.4.2. OMR_GetNumberOfColumnsToRead

処理	読み取り行数設定コマンド NC を使用して、OMR に設定されている読み取り行数を取得する。	
プロトタイプ	int OMR_GetNumberOfColumnsToRead(void)	
引数	なし	
戻り値	0	失敗
	0 以外	OMR 本体に設定されている読み取り行数

6.4.3. OMR_SetReadingMethod

処理	読み取り方式設定コマンド RM を使用して、マーク読み取り方式を設定する。	
プロトタイプ	BOOL OMR_SetReadingMethod(int iControlType, int iMultipleValue)	
引数	iControlType	読み取り制御方式の指定する SR_READ_INITIAL :初期値に戻す (直下型,制御倍数＝3) SR_READ_FRONT_EDGE :先端制御型 SR_READ_REAR_EDGE :後端制御型 SR_READ_DIRECT :直下型 SR_READ_FACOM :FACOM 型 SR_READ_BETWEEN_MARK_NO_SPACE :マーク間型(先端余白読み取り無し) SR_READ_BETWEEN_MARK :マーク間型(先端余白読み取り有り)
	iMultipleValue	制御倍数 先端制御型は 1～9 を指定する。 後端制御型は 2～9 を指定する。 ただし、制御型の時以外はこの値は無視される。
戻り値	TRUE	成功
	FALSE	失敗

6.4.4. OMR_GetReadingMethod

処理	読み取り方式設定コマンド RM を使用して、設定されている値を取得する。	
プロトタイプ	BOOL OMR_GetReadingMethod(int *iControlType, int *iMultipleValue)	
引数	*iControlType	読み取った読み取り制御方式の値を格納するアドレス
	*iMultipleValue	読み取った制御倍数を格納するアドレス
戻り値	TRUE	成功
	FALSE	失敗

詳細	使用例
	int ctl_type, multi_val;
	
	//取得実行
	OMR_GetReadingMethod(&ctl_type, &multi_val);	

6.4.5. OMR_SetBackSensorUnit

処理	裏面読み取り設定コマンドBRを使用して、裏面読み取りユニットの使用/未使用を設定する。	
プロトタイプ	BOOL OMR_SetBackSensorUnit(int iDirective)	
引数	iDirective	設定するかどうかを指定する SR_INITIAL :初期値に戻す(使用する) SR_ENABLE :使用する SR_DISABLE :使用しない
戻り値	TRUE	成功
	FALSE	失敗

6.4.6. OMR_GetBackSensorUnit

処理	裏面読み取り設定コマンドBRを使用して、裏面読み取りユニットの使用/未使用を取得する。	
プロトタイプ	int OMR_GetBackSensorUnit(void)	
引数	なし	
戻り値	SR_ENABLE	使用する設定になっている
	SR_DISABLE	使用しない設定になっている

6.4.7. OMR_SetSheetThickness

処理	シートの連量設定コマンドFTを使用して、シートの連量（米坪）を設定する。	
プロトタイプ	BOOL OMR_SetSheetThickness(int iThickness)	
引数	iThickness	シートの連量（米坪)を指定。 SR_THICKNESS_AUTO_DETECT :自動検出 SR_THICKNESS_64_GPM2 :64g/m2(55kg紙) SR_THICKNESS_84_GPM2 :84g/m2(72kg紙) SR_THICKNESS_105_GPM2 :105g/m2(90kg紙) SR_THICKNESS_128_GPM2 :128g/m2(110kg紙) SR_THICKNESS_157_GPM2 :157g/m2(135kg紙) SR_THICKNESS_INITIAL :初期値に戻す（90kg紙）
戻り値	TRUE	成功
	FALSE	失敗

6.4.8. OMR_GetSheetThickness

処理	シートの連量設定コマンドFTを使用して、シートの連量（米坪）を取得する。	
プロトタイプ	int OMR_GetSheetThickness(void)	
引数	なし	
戻り値	SR_FUNCTION_FAIL	失敗
	上記以外	設定されているシートの連量（米坪） SR_THICKNESS_AUTO_DETECT :自動検出 SR_THICKNESS_64_GPM2 :64g/m2(55kg紙) SR_THICKNESS_84_GPM2 :84g/m2(72kg紙) SR_THICKNESS_105_GPM2 :105g/m2(90kg紙) SR_THICKNESS_128_GPM2 :128g/m2(110kg紙) SR_THICKNESS_157_GPM2 :157g/m2(135kg紙)

6.4.9. OMR_SetWarningError

処理	ワーニングエラー設定コマンド WE を使用して、ワーニング発生の条件を設定する。	
プロトタイプ	BOOL OMR_SetWarningError(DWORD dwConfigData, int iSkewCol, int iSkewLevel)	
引数	dwConfigData	ワーニングの有効/無効をビットで指定する SR_WARN_AUTO_REJECT SR_WARN_HOPPER_EMPTY SR_WARN_TM_ERROR SR_WARN_DF_ERROR SR_WARN_LEFT_SKEW SR_WARN_MARK_SKEW SR_WARN_INITIAL これらの値は論理和を使用して同時に駆動させることができるが、SR_WARN_INITIAL だけは単体で使用すること
	iSkewCol	マークスキュー検出欄 1～155 を指定。
	iSkewLevel	マークスキュー検出レベル 1～16 を指定。
戻り値	TRUE	成功
	FALSE	失敗
詳細	OMR_GetWarningError 参照 dwConfigData 初期値 SR_WARN_AUTO_REJECT :自動排紙処理(有効) SR_WARN_HOPPER_EMPTY :ホッパエンプティ検出(無効) SR_WARN_TM_ERROR :タイミングマークエラー検出(有効) SR_WARN_DF_ERROR :ダブルフィード検出(有効) SR_WARN_LEFT_SKEW :左端スキュー検出(有効) SR_WARN_MARK_SKEW :マークスキュー検出(無効) iSkewCol 初期値 :001 iSkewLevel 初期値 :4	

6.4.10. OMR_GetWarningError

処理	ワーニングエラー設定コマンド WE を使用して、ワーニング発生の条件設定を取得する。	
プロトタイプ	BOOL OMR_GetWarningError(DWORD *dwConfigData, int *iSkewCol, int *iSkewLevel)	
引数	*dwConfigData	ワーニングの有効/無効をビットで返す。 SR_WARN_AUTO_REJECT :自動排紙処理 SR_WARN_HOPPER_EMPTY :ホッパエンプティ検出 SR_WARN_TM_ERROR :タイミングマークエラー検出 SR_WARN_DF_ERROR :ダブルフィード検出 SR_WARN_LEFT_SKEW :左端スキュー検出 SR_WARN_MARK_SKEW :マークスキュー検出 これらの値の論理和で返す。
	*iSkewCol	マークスキュー検出欄を格納するアドレス。
	*iSkewLevel	マークスキュー検出レベルを格納するアドレス。
戻り値	TRUE	成功
	FALSE	失敗

詳細	使用例
	DWORD warn_info;
	int SkewCol;
	int SkewLevel;
	warn_info = OMR_GetWarningError();
	if (!OMR_GetWarningError(&warn_info, &SkewCol, &SkewLevel)){
	//ここにエラー時の処理を記述する
	}
	if ((warn_info & SR_WARN_DF_ERROR) != 0){
	//ここに処理が来たらダブルフィード検出が有効に設定されている
	}
	
	//現在の設定に加えてホッパエンプティ有効・タイミングマークエラー無効・マークスキュー検出欄数=13・検出レベル=7 に設定
	OMR_SetWarningError((warn_info | SR_WARN_HOPPER_EMPTY |
	SR_WARN_MARK_SKEW) & ~SR_WARN_TM_ERROR, 13, 7);	

6.4.11. OMR_SetPanelConfig

処理	パネル操作設定コマンド P0 を使用して、パネル操作の有効／無効を設定する。	
プロトタイプ	BOOL OMR_SetPanelConfig(int iPanelEnable)	
引数	iPanelEnable	パネル操作の有効／無効 SR_DISABLE :パネル操作の無効 SR_ENABLE :パネル操作の有効 SR_INITIAL :初期値に戻す(有効)
戻り値	TRUE	成功
	FALSE	失敗

6.4.12. OMR_GetPanelConfig

処理	パネル操作設定コマンド P0 を使用して、パネル操作の有効／無効の設定を取得する。	
プロトタイプ	int OMR_GetPanelConfig(void)	
引数	なし	
戻り値	SR_FUNCTION_FAIL	失敗
	それ以外	SR_DISABLE :パネル操作の無効に設定されている SR_ENABLE :パネル操作の有効に設定されている

6.4.13. OMR_SetBuzzerConfig

処理	ブザー設定コマンド BZ を使用して、ブザーの音量と音質を設定する。ブザーを無効に設定した場合は全てのブザー出力がなくなるため、エラー発生時はステータスおよびパネル表示のみの通知となる。	
プロトタイプ	BOOL OMR_SetBuzzerConfig(int iVolume, int iTone)	
引数	iVolume	ブザー音量 1～5 SR_BUZZER_DISABLE :ブザーの無効 SR_BUZZER_INITIAL :初期値に戻す (ブザー有効／音量:3／音質:2)
	iTone	ブザーの音質 1～3 iVolume が SR_BUZZER_DISABLE, SR_BUZZER_INITIAL の場合は無効。
戻り値	TRUE	成功
	FALSE	失敗

6.4.14. OMR_GetBuzzerConfig

処理	ブザー設定コマンド BZ を使用して、ブザーの音量と音質の設定情報を取得する。	
プロトタイプ	BOOL OMR_GetBuzzerConfig(int *piVolume, int *piTone)	
引数	piVolume	ブザー音量を格納するアドレス
	piTone	ブザーの音質を格納するアドレス
戻り値	TRUE	成功
	FALSE	失敗

6.4.15. OMR_SetID

処理	識別コード設定コマンド ID を使用して、OMR の識別コードを設定する。識別コードを設定することにより、アプリケーションとの識別 等を行なわせることが可能となる。	
プロトタイプ	BOOL OMR_SetID(CHAR *pID)	
引数	pID	文字列を格納した CHAR 型配列の先頭ポインタ。 文字コードは 0x00～0xFF。20 文字固定で設定。
戻り値	TRUE	成功
	FALSE	失敗

6.4.16. OMR_GetID

処理	識別コード設定コマンド ID を使用して、OMR の識別コードを取得する。	
プロトタイプ	BOOL OMR_GetID(CHAR *pID)	
引数	pID	識別コードを格納するアドレス （初期値） 'SEKONIC 機種名'
戻り値	TRUE	成功
	FALSE	失敗
詳細	初期値の例 : 'SEKONIC SR-2300' 20 文字固定で終端に NULL は付かない。	

6.4.17. OMR_SetSleepTime

処理	低消費電力設定コマンド ES を使用して、スリープ/スタンバイまでの時間を設定する。 機種により設定できない場合がある。取扱説明書を参照。	
プロトタイプ	BOOL OMR_SetSleepTime(int iSleepTime, int iStandbyTime)	
引数	iSleepTime	スリープするまでの時間 1～60[分] SR_SLEEPTIME_DISABLE :無効 SR_SLEEPTIME_INITIAL :初期値に戻す
	iStandbyTime	スタンバイするまでの時間 1～60[分] SR_STANDBYTIME_DISABLE :無効
戻り値	TRUE	成功
	FALSE	失敗

6.4.18. OMR_GetSleepTime

処理	低消費電力設定コマンド ES を使用して、スリープ/スタンバイまでの時間を取得する。 機種により取得できない場合がある。取扱説明書を参照。	
プロトタイプ	BOOL OMR_GetSleepTime(int* piSleepTime, int* piStandbyTime)	
引数	piSleepTime	スリープするまでの時間 1～60[分] SR_SLEEPTIME_DISABLE :無効
	piStandbyTime	スタンバイするまでの時間 1～60[分] SR_STANDBYTIME_DISABLE :無効
戻り値	TRUE	成功
	FALSE	失敗

6.5. API 関数 ー 動作要求

6.5.1. OMR_Reset

処理	ソフトウェアリセットコマンド SR を使用して、OMR を電源投入時の状態にする。	
プロトタイプ	BOOL OMR_Reset(void)	
引数	なし	
戻り値	TRUE	成功
	FALSE	失敗

6.5.2. OMR_FeedSheet

処理	シート読み取りコマンド SF を使用して、シートを一枚搬送させる。	
プロトタイプ	BOOL OMR_FeedSheet(void)	
引数	なし	
戻り値	TRUE	成功
	FALSE	失敗

6.5.3. OMR_MoveHopper

処理	ホッパ動作コマンド HU を使用して、ホッパの上昇/下降を行なう。	
プロトタイプ	BOOL OMR_MoveHopper(int iDirection)	
引数	iDirection	ホッパの動作方向 SR_HOPPER_DOWN :下降 SR_HOPPER_UP :上昇
戻り値	TRUE	成功
	FALSE	失敗

6.5.4. OMR_EjectSheet

処理	排紙動作コマンド ER を使用して、シートの排紙動作を行なう。	
プロトタイプ	BOOL OMR_EjectSheet(int iDirection)	
引数	iDirection	排紙動作の指定 SR_EJECT_MAIN :メインスタッカへ排紙 SR_EJECT_SELECT :セレクトスタッカへ排紙 SR_EJECT_MAIN_ON_NEXT :次の SF コマンドでメインスタッカへ排紙 SR_EJECT_SELECT_ON_NEXT :次の SF コマンドでセレクトスタッカへ排紙
戻り値	TRUE	成功
	FALSE	失敗

6.5.5. OMR_InitialSetting

処理	設定値初期化コマンド IS を使用して、OMR に設定された各設定値を工場出荷時に戻し保存する。	
プロトタイプ	BOOL OMR_InitialSetting(void)	
引数	なし	
戻り値	TRUE	成功
	FALSE	失敗

6.5.6. OMR_CancelError

処理	エラー解除コマンド CE を使用して、OMR 本体からステータス情報を取得する	
プロトタイプ	OMR_STATUS OMR_CancelError(void)	
引数	なし	
戻り値	OMR_STATUS	OMR_STATUS 定数の表を参照
詳細	OMR_GetLastError で取得できる OMR_STATUS にも反映される。	

6.6. API 関数 ー データ要求

6.6.1. OMR_GetMarks

処理	マーク情報を取得し、引数に渡された OMR_MARK 構造体へ格納する。	
プロトタイプ	BOOL OMR_GetMarks(int iPage, OMR_MARK_INFO* pMarkInfo, CHAR* pMarks)	
引数	iPage	表面/裏面識別フラグ SR_PAGE_FRONT :表面マーク情報 SR_PAGE_BACK :裏面マーク情報

	pMark	マーク情報格納用構造体へのポインタ
		本関数呼出前にメンバ変数 rows,columns には取得する最大値を設定しておくこと。第3引数の pMarks が示す変数は rows×columns 以上のメモリが確保されていること。
		呼出し後にはマーク情報が格納される。
		typedef struct tagOMR_MARK_INFO
		{
		long type; //未使用
		long rows; //欄数
		long columns; //行数
		}OMR_MARK_INFO, *POMR_MARK_INFO;

	pMarks	マーク情報は 1 マークあたり 1 バイト、0x0～0x10 のバイナリで格納される。 第2引数のメンバ変数 rows×columns 以上のメモリが確保されていること。
戻り値	TRUE	成功
	FALSE	失敗

詳細	本関数を呼び出す前に第2引数のメンバ変数 rows,columns に読み出しを行うマークの最大数を設定しておくこと。また、第3引数のポインタ pMarks が示す変数は rows×columns 以上のメモリが確保されていること。
	実際のマークデータが呼び出し前の rows×columns を超えた場合、pMarks には呼び出し前の(rows×columns)までのマークデータが格納されます。
	
	サンプル
	OMR_MARK_INFO MarkInfo;
	CHAR Marks[48*155];
	
	MarkInfo.columnss = 48;
	MarkInfo.rows = 155;
	if(!OMR_GetMarks(SR_PAGE_FRONT, &MarkData, Marks)){
	// エラー処理
	}
	else{
	// マークデータ処理
	}	

6.6.2. OMR_GetStatus

処理	ステータス情報要求コマンド ST を使用して、OMR 本体からステータス情報を取得する。	
プロトタイプ	OMR_STATUS OMR_GetStatus(void)	
引数	なし	
戻り値	OMR_STATUS	OMR_STATUS 定数の表を参照
詳細	OMR_GetLastError で取得できる OMR_STATUS にも反映される。	

6.6.3. OMR_GetSensorInfo

処理	センサ情報要求コマンド DS を使用して、読み取りセンサ以外の検知センサの ON/OFF 情報を取得する。	
プロトタイプ	DWORD OMR_GetSensorInfo(void)	
引数	なし	
戻り値	0xXXXXXXXX	成功(下記の詳細参照)
	SR_FUNCTION_FAIL	失敗

詳細

戻り値が SR_SENSOR_FAIL 以外の値をとった場合、DWORD 型 32bit は各ビットがそれぞれのセンサの ON/OFF 情報に対応している。
各ビットの対応は以下の表のとおり

Bit	対応するセンサ	マスク定数名	定数の値
Bit31	なし： 0（固定）		
Bit30	なし： 1（固定）		
Bit29	本体排紙検知	SR_SENSOR_OUTPS	0x20000000
Bit28	読み取り開始検知	SR_SENSOR_RDPS	0x10000000
Bit27	給紙検知	SR_SENSOR_INPS	0x08000000
Bit26	０枚検知	SR_SENSOR_PS0	0x04000000
Bit25	ホッパ上限検知	SR_SENSOR_UPPS	0x02000000
Bit24	ホッパ下限検知	SR_SENSOR_DWPS	0x01000000
Bit23	なし： 0（固定）		
Bit22	なし： 1（固定）		
Bit21	なし： 0（拡張用）		
Bit20	スキューセンサ	SR_SENSOR_SKS	0x00100000
Bit19	なし： 0（拡張用）		
Bit18	なし： 0（拡張用）		
Bit17	なし： 0（拡張用）		
Bit16	本体扉開閉検知	SR_SENSOR_MAIN_CVR	0x00010000
Bit15	なし： 0（固定）		
Bit14	なし： 1（固定）		
Bit13	なし： 0（拡張用）		
Bit12	なし： 0（拡張用）		
Bit11	なし： 0（拡張用）		
Bit10	セレクト排紙	SR_SENSOR_SPS	0x00000400
Bit9	メイン排紙検知	SR_SENSOR_MPS	0x00000200
Bit8	プリンタ 2 印字開始検知	SR_SENSOR_P2PS	0x00000100
Bit7	なし： 0（固定）		
Bit6	なし： 1（固定）		
Bit5	なし： 0（拡張用）		
Bit4	なし： 0（拡張用）		
Bit3	なし： 0（拡張用）		
Bit2	なし： 0（拡張用）		
Bit1	なし： 0（拡張用）		
Bit0	スタッカユニット扉開閉検知	SR_SENSOR_STK_CVR	0x00000001

使用例(本体排紙検知センサの取得)

```
DWORD sen_info;
sen_info = OMR_GetSensorInfo();
if ( sen_info == SR_FUNCTION_FAIL){
    //ここにエラー時の処理を記述する
}
if ( (sen_info & SR_SENSOR_OUTPS) != 0){
    //ここに本体排紙検知センサが ON になっている場合の処理を記述する
}
```

6.6.4. OMR_GetDeviceInfo

処理	装置情報要求コマンド DI を使用して、接続されているユニットを確認する。	
プロトタイプ	DWORD OMR_GetDeviceInfo(void)	
引数	なし	
戻り値	0xXXXXXXXX	成功(下記の詳細参照)
	SR_FUNCTION_FAIL	失敗

詳細

戻り値が SR_FUNCTION_FAIL 以外の値をとった場合、DWORD 型 32bit は各ビットがそれぞれのユニット情報に対応している。
各ビットの対応は以下の表のとおり

Bit	内容		マスク定数名	定数の値
Bit31 Bit30 Bit29 Bit28	スタッカユニット	0x0：接続なし 0x1：接続あり 0x8：カートリッジなし	SR_DEVICE_UNIT_STACKER_MASK SR_DEVICE_UNIT_STACKER SR_DEVICE_UNIT_STACKER_ERR	0xf0000000 0x10000000 0x80000000
Bit27 Bit26 Bit25 Bit24	プリンタユニット	0x0：接続なし 0x1：接続あり 0x8：カートリッジなし	SR_DEVICE_UNIT_PRINTER_MASK SR_DEVICE_UNIT_PRINTER SR_DEVICE_UNIT_PRINTER_ERR	0x0f000000 0x01000000 0x08000000
Bit23 Bit22 Bit21 Bit20	バーコードユニット	0x0：接続なし 0x1：縦流し接続あり 0x2：横流し接続あり 0x8：接続エラー	SR_DEVICE_UNIT_BARCODE_MASK SR_DEVICE_UNIT_BARCODE_V SR_DEVICE_UNIT_BARCODE_H SR_DEVICE_UNIT_BARCODE_ERR	0x00f00000 0x00100000 0x00200000 0x00800000
Bit19 Bit18 Bit17 Bit16	裏面ユニット	0x0：接続なし 0x1：接続あり	SR_DEVICE_UNIT_BACK_MASK SR_DEVICE_UNIT_BACK	0x000f0000 0x00010000
Bit15 ： Bit8	（未使用）			
Bit7 Bit6 Bit5 Bit4	センサタイプ	0x0：赤可視光 0x1：近赤外光 0x8：接続エラー	SR_DEVICE_SENSOR_TYPE_MASK SR_DEVICE_SENSOR_TYPE_ERR	0x000000f0 0x00000080
Bit3 Bit2 Bit1 Bit0	センサピッチタイプ	0x1：1/6 インチ 0x2：0.2 インチ 0x3：0.2 インチ S 0x4：0.25 インチ 0x5：0.3 インチ 0x6：0.3 インチ F 0x7：６mm 0x8：0.2 インチ K 0x9：0.2 インチ特殊 0xA：0.2 インチ C 0xF：接続エラー	SR_DEVICE_SENSOR_PITCH_MASK SR_DEVICE_SENSOR_PITCH_ERR	0x0000000f 0x0000000f

使用例(スタッカユニットの接続確認)

```
DWORD dev_info;
dev_info = OMR_GetDeviceInfo();
if ( dev_info == SR_FUNCTION_FAIL){
    //ここにエラー時の処理を記述する
}
if ( (dev_info & SR_DEVICE_UNIT_STACKER) != 0){
    //ここに処理が来たらスタッカユニットは接続されている
}
```

6.6.5. OMR_GetMachineName

処理	機種名要求コマンド MN を使用して、引数のポインタが指す領域へ機種名文字列が格納される。	
プロトタイプ	BOOL OMR_GetMachineName(CHAR * pResult)	
引数	pResult	機種名文字列を格納するアドレス
戻り値	TRUE	成功
	FALSE	失敗
詳細	使用例: CHAR pBuf[16]; OMR_GetMachineName(pBuf); printf(pBuf);	

6.6.6. OMR_GetVersion

処理	ファームウェアバージョン要求コマンド FV を使用して、引数のポインタが指す領域へ文字列が格納される。	
プロトタイプ	BOOL OMR_GetVersion(int iTarget, CHAR * pResult)	
引数	iTarget	バージョンを取得するユニットを指定 SR_UNIT_MAIN :本体 SR_UNIT_FRONT :表面読み取りユニット SR_UNIT_BACK :裏面読み取りユニット SR_UNIT_BARCODE :バーコードユニット SR_UNIT_PRINTER :プリンターユニット SR_UNIT_STACKER :スタッカユニット
	*pResult	バージョンを格納するアドレス
戻り値	TRUE	成功
	FALSE	失敗
詳細	使用例:本体のバージョンを取得する CHAR pBuf[16]; OMR_GetVersion(SR_UNIT_MAIN, pBuf); printf(pBuf);	

6.7. API 関数 ー プリンタ設定

6.7.1. OMR_SetPrinterUnit

処理	プリンタ設定コマンド PR を使用して、プリンタ制御を設定する。	
プロトタイプ	BOOL OMR_SetPrinterUnit (int iDirective)	
引数	iDirective	設定するかどうかを指定する SR_INITIAL :初期値に戻す(使用する) SR_ENABLE :プリンタ制御許可 SR_DISABLE :プリンタ制御禁止
戻り値	TRUE	成功
	FALSE	失敗

6.7.2. OMR_GetPrinterUnit

処理	プリンタ設定コマンド PR を使用して、プリンタ制御を取得する。	
プロトタイプ	int OMR_GetPrinterUnit (void)	
引数	なし	
戻り値	SR_ENABLE	プリンタ制御許可
	SR_DISABLE	プリンタ制御禁止
	SR_FUNCTION_FAIL	取得失敗

6.7.3. OMR_SetPrintOrder

処理	プリンタ設定コマンド PR を使用して、印字順序を設定する。 この印字順序の設定は印字が行われるとクリアされる。	
プロトタイプ	BOOL OMR_SetPrintOrder (int iFirst, int iSecond, int iThird)	
引数	iFirst	1番目に印字するバッファ番号。 SR_PRINT_BUFFER_NULL :印字しない SR_PRINT_BUFFER_1 :バッファ1 SR_PRINT_BUFFER_2 :バッファ2 SR_PRINT_BUFFER_3 :バッファ3
	iSecond	2番目に印字するバッファ番号。
	iThird	3番目に印字するバッファ番号。
戻り値	TRUE	成功
	FALSE	失敗

6.7.4. OMR_GetPrintOrder

処理	プリンタ設定コマンド PR を使用して、プリンタの印字方法を取得する。	
プロトタイプ	BOOL OMR_GetPrintOrder (int* pFirst, int* pSecond, int* pThird)	
引数	pFirst	1番目に印字するバッファ番号を格納するポインタ。
	pSecond	2番目に印字するバッファ番号を格納するポインタ。
	pThird	3番目に印字するバッファ番号を格納するポインタ。
戻り値	TRUE	成功
	FALSE	失敗

6.7.5. OMR_SetPrinterMode

処理	プリンタ設定コマンド PR を使用して、プリンタの印字方法を設定する。	
プロトタイプ	BOOL OMR_SetPrinterMode(int iMode)	
引数	iMode	印字方法 SR_PRINTER_MODE_AFTER_FEED :読み取り後印字する。 SR_PRINTER_MODE_FEED_AND_PRINT :読み取りながら印字する。
戻り値	TRUE	成功
	FALSE	失敗

6.7.6. OMR_GetPrinterMode

処理	プリンタ設定コマンド PR を使用して、プリンタの印字方法を取得する。	
プロトタイプ	BOOL OMR_GetPrinterMode(int* piMode)	
引数	piMode	印字方法を格納するアドレス
戻り値	TRUE	成功
	FALSE	失敗

6.7.7. OMR_SetPrintPosition

処理	プリンタ設定コマンド PR を使用して、プリンタの印字位置を設定する。	
プロトタイプ	BOOL OMR_SetPrintPosition (int iPosition)	
引数	iPosition	印字位置。(単位:mm) ※:設定値の範囲は各取扱説明書を参照してください。
戻り値	TRUE	成功
	FALSE	失敗

6.7.8. OMR_GetPrintPosition

処理	プリンタ設定コマンド PR を使用して、プリンタの印字位置を取得する。	
プロトタイプ	BOOL OMR_GetPrintPosition(int* piPosition)	
引数	piMode	印字位置を格納するアドレス
戻り値	TRUE	成功
	FALSE	失敗

6.7.9. OMR_SetPrintAngle

処理	プリンタ設定コマンド PR を使用して、プリンタの印字方向を設定する。	
プロトタイプ	BOOL OMR_SetPrintAngle(DWORD dwAngle)	
引数	dwAngle	印字方向。 SR_PRINT_ANGLE_0 :通常印字 SR_PRINT_ANGLE_180 :文字列を180度回転
戻り値	TRUE	成功
	FALSE	失敗

6.7.10. OMR_GetPrintAngle

処理	プリンタ設定コマンド PR を使用して、プリンタの印字方向を取得する。	
プロトタイプ	BOOL OMR_GetPrintAngle(DWORD *dwAngle)	
引数	dwAngle	印字方向を格納するアドレス
戻り値	TRUE	成功
	FALSE	失敗

6.7.11. OMR_SetPrintFontSize

処理	プリンタ設定コマンド PR を使用して、プリンタの文字サイズを設定する。	
プロトタイプ	BOOL OMR_SetPrintFontSize(int iSize)	
引数	iSize	印字の文字サイズ。(単位:0.1mm) ※:設定値の範囲は各取扱説明書を参照してください。
戻り値	TRUE	成功
	FALSE	失敗

6.7.12. OMR_GetPrintFontSize

処理	プリンタ設定コマンド PR を使用して、プリンタの文字サイズを取得する。	
プロトタイプ	BOOL OMR_GetPrintFontSize(int *iSize)	
引数	iSize	印字の文字サイズを格納するアドレス
戻り値	TRUE	成功
	FALSE	失敗

6.7.13. OMR_SetPrintFontPitch

処理	プリンタ設定コマンド PR を使用して、プリンタの印字文字間隔を設定する。	
プロトタイプ	BOOL OMR_SetPrintFontPitch(int iPitch)	
引数	iPitch	印字文字間隔。(単位:0.1mm) ※:設定値の範囲は各取扱説明書を参照してください。
戻り値	TRUE	成功
	FALSE	失敗

6.7.14. OMR_GetPrintFontPitch

処理	プリンタ設定コマンド PR を使用して、プリンタの印字文字間隔を取得する。	
プロトタイプ	BOOL OMR_GetPrintFontPitch(int *iPitch)	
引数	iPitch	印字文字間隔を格納するアドレス
戻り値	TRUE	成功
	FALSE	失敗

6.7.15. OMR_SetPrintString

処理	プリンタ設定コマンド PR を使用して、プリンタのバッファに文字列を設定する。	
プロトタイプ	OMR_SetPrintString(int iBufDirec, CHAR *pString)	
引数	iBufDirec	設定するバッファの番号を指定する。
	pString	設定する文字列のポインタ。 1から42文字まで設定できる。
戻り値	TRUE	成功
	FALSE	失敗

6.7.16. OMR_GetPrintString

処理	プリンタ設定コマンド PR を使用して、プリンタのバッファに格納されている文字列を取得する。	
プロトタイプ	BOOL OMR_GetPrintString(int iBufDirec, CHAR *pString)	
引数	iBufDirec	取得するバッファの番号を指定する。
	pString	取得する文字列を格納するポインタ。
戻り値	TRUE	成功
	FALSE	失敗

6.8. API 関数 － バーコードリーダー制御

6.8.1. OMR_GetBarcodeInfo

処理	バーコードデータ要求コマンド BD を使用して、読み取ったシートのバーコードデータ数を取得する。	
プロトタイプ	OMR_GetBarcodeInfo(int* piReadCount, int* piSettingCount)	
引数	piReadCount	読み取ったバーコードデータの数を格納する為の変数のポインタ。０～１０
	piSettingCount	読み取り位置指定されている数を格納する為の変数のポインタ。０～１０
戻り値	TRUE	成功
	FALSE	失敗

6.8.2. OMR_GetBarcodeData

処理	バーコードデータ要求コマンド BD を使用して、指定したバーコードデータを取得する。	
プロトタイプ	OMR_GetBarcodeData(int iIndex, CHAR* pBcType, int* piDataLen, CHAR* pBarcode)	
引数	iIndex	取得するバーコードデータの番号を設定する。１～１０
	pBcType	バーコードの種類を表す1バイトを格納する為のポインタ。 （値は詳細を参照）
	piDataLen	バーコードデータのバイト数を格納する為のポインタ。
	pBarcode	バーコードデータを格納する為のポインタ。
戻り値	TRUE	成功
	FALSE	失敗

詳細

バーコードの種類を表す定数は以下の通り。

定数名	値	意味
SR_BARCODE_TYPE_UNKNOWN	'@'	なし/不明
SR_BARCODE_TYPE_CODE39	'a'	CODE-39
SR_BARCODE_TYPE_ITF	'b'	ITF
SR_BARCODE_TYPE_NW7	'c'	NW-7(Codabar)
SR_BARCODE_TYPE_JAN_EAN_UPC	'd'	JAN/EAN/UPC
SR_BARCODE_TYPE_CODE128	'e'	CODE-128
SR_BARCODE_TYPE_INDUSTRIAL2OF5	'f'	Industrial 2of5
SR_BARCODE_TYPE_COOP2OF5	'g'	COOP 2of5
SR_BARCODE_TYPE_CODE93	'h'	CODE-93

6.8.3. OMR_SetBarcodeReaderUnit

処理	バーコード設定コマンド BC を使用して、バーコード読み取りの許可/禁止を設定する。	
プロトタイプ	OMR_SetBarcodeReaderUnit(int iDirective)	
引数	iDirective	設定するかどうかを指定する SR_INITIAL :初期値に戻す(使用する) SR_ENABLE :許可 SR_DISABLE :禁止
戻り値	TRUE	成功
	FALSE	失敗

6.8.4. OMR_GetBarcodeReaderUnit

処理	バーコード設定コマンド BC を使用して、バーコード読み取りの許可/禁止を取得する。	
プロトタイプ	OMR_GetBarcodeReaderUnit(void)	
引数	なし	
戻り値	SR_ENABLE	バーコードの読み取り許可
	SR_DISABLE	バーコードの読み取り禁止
	SR_FUNCTION_FAIL	取得失敗

6.8.5. OMR_SetBarCodeArea

処理	バーコード設定コマンド BC を使用して、バーコードの読み取り範囲を設定する。	
プロトタイプ	OMR_SetBarCodeArea(OMR_BARCODE_AREA* pAreaArray)	
引数	pAreaArray	読み取らない範囲と読み取る範囲を指定する構造体の配列へのポインタ。配列は10個固定。 (構造体、設定値については詳細を参照)
戻り値	TRUE	成功
	FALSE	失敗

詳細	読み取り範囲構造体
	typedef struct tagOMR_BARCODE_AREA
	{
	long unread; // 読み取らない範囲[mm]
	long read; // 読み取る範囲[mm]
	}OMR_BARCODE_AREA, *POMR_BARCODE_AREA;	

6.8.6. OMR_GetBarCodeArea

処理	バーコード設定コマンド BC を使用して、バーコードの読み取り範囲を取得する。	
プロトタイプ	OMR_GetBarCodeArea(OMR_BARCODE_AREA* pAreaArray)	
引数	pAreaArray	読み取らない範囲と読み取る範囲を指定する構造体の配列へのポインタ。配列は10個固定。 呼び出し側でメモリの確保を行う。
戻り値	TRUE	成功
	FALSE	失敗

6.8.7. OMR_SetBarCodeReadType

<table>
<tr><td>処理</td><td colspan="2">バーコード設定コマンド BC を使用して、読み取るバーコードの種類を設定する。</td></tr>
<tr><td>プロトタイプ</td><td colspan="2">OMR_SetBarCodeReadType(DWORD dwReadType)</td></tr>
<tr><td>引数</td><td>dwReadType</td><td>読み取りを行うバーコードの種類を DWORD 型32bit のビットで指定する。
(ビット定義は詳細を参照)</td></tr>
<tr><td rowspan="2">戻り値</td><td>TRUE</td><td>成功</td></tr>
<tr><td>FALSE</td><td>失敗</td></tr>
<tr><td>詳細</td><td colspan="2">読み取りを行うバーコードの種類を表す定数は以下の通り。
<table>
<tr><th>定数名</th><th>値</th><th>意味</th></tr>
<tr><td>SR_BARCODE_READ_CODE39</td><td>0x00000001</td><td>CODE-39</td></tr>
<tr><td>SR_BARCODE_READ_ITF</td><td>0x00000002</td><td>ITF</td></tr>
<tr><td>SR_BARCODE_READ_NW7</td><td>0x00000004</td><td>NW-7(Codabar)</td></tr>
<tr><td>SR_BARCODE_READ_JAN_EAN_UPC</td><td>0x00000008</td><td>JAN/EAN/UPC</td></tr>
<tr><td>SR_BARCODE_READ_CODE128</td><td>0x00000010</td><td>CODE-128
EAN-128</td></tr>
<tr><td>SR_BARCODE_READ_INDUSTRIAL2OF5</td><td>0x00000020</td><td>Industrial 2of5</td></tr>
<tr><td>SR_BARCODE_READ_COOP2OF5</td><td>0x00000040</td><td>COOP 2of5</td></tr>
<tr><td>SR_BARCODE_READ_CODE93</td><td>0x00000080</td><td>CODE-93</td></tr>
</table>
</td></tr>
</table>

6.8.8. OMR_GetBarCodeReadType

<table>
<tr><td>処理</td><td colspan="2">バーコード設定コマンド BC を使用して、読み取るバーコードの種類を取得する。</td></tr>
<tr><td>プロトタイプ</td><td colspan="2">OMR_GetBarCodeReadType(DWORD* pdwReadType)</td></tr>
<tr><td>引数</td><td>pdwReadType</td><td>読み取りを行うバーコードの種類を DWORD 型32bit のビットで表す。
(ビット定義は設定関数の詳細を参照)</td></tr>
<tr><td rowspan="2">戻り値</td><td>TRUE</td><td>成功</td></tr>
<tr><td>FALSE</td><td>失敗</td></tr>
</table>

6.8.9. OMR_SetBarCodeCheckDigit

<table>
<tr><td>処理</td><td colspan="2">バーコード設定コマンド BC を使用して、各バーコードの種類に対してのチェックディジットを設定する。
指定できないバーコードや読み取る設定となっていない場合、パラメータエラーが発生する。</td></tr>
<tr><td>プロトタイプ</td><td colspan="2">OMR_SetBarCodeCheckDigit(CHAR cBcType, int iCdType)</td></tr>
<tr><td rowspan="2">引数</td><td>cBcType</td><td>設定するバーコードの種類を設定する。
（設定値は詳細を参照）</td></tr>
<tr><td>iCdType</td><td>チェックディジットの検査の有無または種類を設定する。（設定値は詳細を参照）</td></tr>
<tr><td rowspan="2">戻り値</td><td>TRUE</td><td>成功</td></tr>
<tr><td>FALSE</td><td>失敗</td></tr>
<tr><td>詳細</td><td colspan="2">バーコードの種類を表す定数は以下の通り。
<table>
<tr><th>定数名</th><th>値</th><th>意味</th></tr>
<tr><td>SR_BARCODE_TYPE_CODE39</td><td>'a'</td><td>CODE-39</td></tr>
<tr><td>SR_BARCODE_TYPE_ITF</td><td>'b'</td><td>ITF</td></tr>
<tr><td>SR_BARCODE_TYPE_NW7</td><td>'c'</td><td>NW-7(Codabar)</td></tr>
</table>
チェックディジットの種類を表す定数は以下の通り。
検査の有無のみのバーコードの場合（“CODE-39”/“ITF”）
<table>
<tr><th>定数名</th><th>値</th><th>意味</th></tr>
<tr><td>SR_BARCODE_CD_INITIAL</td><td>SR_INITIAL</td><td>初期化</td></tr>
<tr><td>SR_BARCODE_CD_DISABLE</td><td>0</td><td>検査なし</td></tr>
<tr><td>SR_BARCODE_CD_ENABLE</td><td>1</td><td>検査あり</td></tr>
</table>
検査無/チェックディジットの種類が指定できるバーコードの場合（“NW-7”）
<table>
<tr><th>定数名</th><th>値</th><th>意味</th></tr>
<tr><td>SR_BARCODE_CD_INITIAL</td><td>SR_INITIAL</td><td>初期化</td></tr>
<tr><td>SR_BARCODE_CD_DISABLE</td><td>0</td><td>検査なし</td></tr>
<tr><td>SR_BARCODE_CD_MODUL016</td><td>1</td><td>モジュラス 16</td></tr>
<tr><td>SR_BARCODE_CD_MODUL011</td><td>2</td><td>モジュラス１１</td></tr>
<tr><td>SR_BARCODE_CD_MODUL010WAIT2</td><td>3</td><td>モジュラス１０/２ウェイト</td></tr>
<tr><td>SR_BARCODE_CD_MODUL010WAIT3</td><td>4</td><td>モジュラス１０/３ウェイト</td></tr>
<tr><td>SR_BARCODE_CD_7DR</td><td>5</td><td>７DR</td></tr>
<tr><td>SR_BARCODE_CD_MODUL011W</td><td>6</td><td>加重モジュラス１１</td></tr>
<tr><td>SR_BARCODE_CD_RUNES</td><td>7</td><td>ルーンズ</td></tr>
</table>
</td></tr>
</table>

6.8.10. OMR_GetBarCodeCheckDigit

処理	バーコード設定コマンド BC を使用して、各バーコードの種類に対してのチェックディジットを取得する。 指定できないバーコードや読み取る設定となっていない場合、パラメータエラーが発生する。	
プロトタイプ	OMR_GetBarCodeCheckDigit(CHAR cBcType, int* piCdType)	
引数	cBcType	取得するバーコードの種類を指定する。 (値は設定関数の詳細を参照)
	piCdType	チェックディジットの検査の有無または種類を格納するポインタ。(値は設定関数の詳細を参照)
戻り値	TRUE	成功
	FALSE	失敗

6.8.11. OMR_SetBarCodeUpcOption

処理	バーコード設定コマンド BC を使用して、各バーコードのを設定する。 設定のないバーコードや読み取る設定になっていない場合、パラメータエラーが発生する。	
プロトタイプ	OMR_SetBarCodeOption(CHAR cBcType, DWORD dwOption)	
引数	cBcType	設定するバーコードの種類を設定する。 (設定値は詳細を参照)
	dwOption	詳細設定を表す DWORD32bit の値。 (設定値は詳細を参照)
戻り値	TRUE	成功
	FALSE	失敗

詳細

バーコードの種類を表す定数は以下の通り。

定数名	値	意味
SR_BARCODE_TYPE_JAN_EAN_UPC	'd'	JAN/EAN/UPC

詳細設定を表す定数は以下の通り。
設定の初期化(単独で使用する)

定数名	値	意味
SR_BARCODE_OPT_INITIAL	0xffffffff	初期化

UPC-A 出力桁数設定(JAN/EAN/UPC の場合いずれかを設定)

定数名	値	意味
SR_BARCODE_OPT_UPC_12DIGIT	0x00000001	１２桁出力
SR_BARCODE_OPT_UPC_13DIGIT	0x00000002	１３桁出力

UPC-E システムコード設定(JAN/EAN/UPC の場合いずれかを設定)

定数名	値	意味
SR_BARCODE_OPT_UPC_NO_CODE	0x00000010	付加しない
SR_BARCODE_OPT_UPC_ADD_CODE	0x00000020	付加する

6.8.12. OMR_GetBarCodeUpcOption

処理	バーコード設定コマンド BC を使用して、UPC の出力形式を取得する。 UPC を読み取る設定になっていない場合、パラメータエラーが発生する。	
プロトタイプ	OMR_GetBarCodeOption(CHAR cBcType, DWORD* pdwOption)	
引数	cBcType	設定するバーコードの種類を設定する。 (設定値は詳細を参照)
	pdwOption	
戻り値	TRUE	成功
	FALSE	失敗

OPTICAL MARK READER
SR-3500/6000/6500, SR-1800, SR-3500 HYBRID, SR-11000
Windows 用 API リファレンス・マニュアル

発行年月　2013 年 05 月　第 6 版

〒178-8686 東京都練馬区大泉学園町 7 - 2 4 - 1 4
TEL 03-3978-2332 FAX 03-3922-2144
ホームページアドレス http://www.sekonic.co.jp
E-mail:omr@sekonic.co.jp