マニュアル

Algo Smart Panel用

『Windows Embedded CE 6.0』

について

ＡＰ－４４１０
ＡＰ－５４１０
ＡＰ－６４１０
ＡＰ－６５００
ＡＰ－７５００

目　次

はじめに

１）　お願いと注意 …… １

２）　対応機種について …… １

第１章　概要

１－１ 機能と特長 …… １－１

１－１－１　Algo Smart Panel用Windows Embedded CE 6.0 とは …… １－１

１－１－２　機能と特長 …… １－１

１－１－３　デバイスドライバ構成 …… １－４

１－２ システム構成 …… １－５

１－２－１　ファイルシステム構成 …… １－５

１－２－２　フォルダ・ファイル構成 …… １－６

第２章　システムの操作

２－１ OSの起動と終了 …… ２－１

２－１－１　OSの起動 …… ２－１

２－１－２　OSの終了 …… ２－１

２－２ ASD Config Tool …… ２－２

２－２－１　ASD Config Tool …… ２－２

２－２－２　LCD Setting …… ２－２

２－２－３　Buzzer Setting …… ２－３

２－２－４　Board Information …… ２－４

２－２－５　初期値 …… ２－４

２－３ アプリケーションの自動起動 …… ２－５

２－３－１　スタートアップフォルダによる自動起動 …… ２－５

２－３－２　Algo Smart Panel専用スタートアップ機能による自動起動 …… ２－５

２－４ ネットワークサーバ …… ２－７

２－４－１　Windows共有ファイルサーバ …… ２－７

２－４－２　FTPサーバ …… ２－８

２－４－３　TELNETサーバ …… ２－８
２－４－４　WEBサーバ …… ２－９

第３章　Algo Smart Panelについて

３－１ Algo Smart Panelに搭載された機能について …… ３－１
３－２ Windows標準インターフェース対応機能 …… ３－３
３－２－１　グラフィック …… ３－３
３－２－２　タッチパネル …… ３－３
３－２－３　シリアルポート …… ３－３
３－２－４　有線LAN …… ３－４
３－２－５　USBポート …… ３－４
３－２－６　SD/SDHCカード …… ３－４
３－２－７　RAMディスク …… ３－４
３－３ 組込みシステム機能 …… ３－５
３－３－１　ブザー …… ３－５
３－３－２　汎用入出力 …… ３－５
３－３－３　LCDバックライト …… ３－５
３－３－４　RAS機能 …… ３－５
３－３－５　シリアルコントロール機能 …… ３－５
３－３－６　バックアップSRAM …… ３－６

第４章　アプリケーション開発

４－１ アプリケーション開発について …… ４－１
４－２ 開発環境の構築 …… ４－２
４－２－１　必要なシステム …… ４－２
４－２－２　開発ツール …… ４－２
４－２－３　開発ツールのインストール …… ４－２
４－３ ターゲットとの接続 …… ４－３
４－３－１　IPアドレスの確認 …… ４－３
４－３－２　Visual Studio 2005のデバイス設定 …… ４－４
４－３－３　TCPトランスポートプログラムの起動 …… ４－５
４－３－４　接続確認 …… ４－６
４－４ ネイティブコードアプリケーション開発 …… ４－７

４－４－１　アプリケーションの作成 …… ４－７
４－４－２　アプリケーションのデバッグ …… ４－１１
４－５ マネージドコードアプリケーション開発 …… ４－１３
４－５－１　アプリケーションの作成 …… ４－１３
４－５－２　アプリケーションのデバッグ …… ４－１５
４－６ リモートツール …… ４－１８
４－６－１　リモートツールの起動と接続 …… ４－１８
４－６－２　リモート ズームイン …… ４－１９
４－６－３　リモート スパイ …… ４－１９
４－６－４　リモート ヒープ ウォーカ …… ４－２０
４－６－５　リモート ファイル ビューア …… ４－２０
４－６－６　リモート プロセス ビューア …… ４－２１
４－６－７　リモート レジストリ エディタ …… ４－２１

第５章　組込みシステム機能ドライバ

５－１ ドライバの使用について …… ５－１
５－１－１　サンプルコードについて …… ５－１
５－１－２　DeviceIoControlについて …… ５－２
５－２ バックライト …… ５－３
５－２－１　バックライトについて …… ５－３
５－２－２　バックライトドライバ …… ５－３
５－２－３　バックライトデバイス …… ５－４
５－２－４　リファレンス …… ５－５
５－２－５　サンプルコード …… ５－１２
５－３ ブザー …… ５－１７
５－３－１　ブザーについて …… ５－１７
５－３－２　ブザードライバ …… ５－１７
５－３－３　ブザーデバイス …… ５－１８
５－３－４　リファレンス …… ５－２０
５－３－５　サンプルコード …… ５－２９
５－４ 汎用入出力 …… ５－３６
５－４－１　汎用入出力について …… ５－３６
５－４－２　汎用出力ドライバ …… ５－３６
５－４－３　汎用出力デバイス …… ５－３７

５－４－４　汎用出力リファレンス ・・・５－３８
５－４－５　汎用入力ドライバ ・・５－４０
５－４－６　汎用入力デバイス ・・５－４１
５－４－７　汎用入力 IN1 割込みの使用手順・・５－４２
５－４－８　汎用入力リファレンス ・・・５－４３
５－４－９　サンプルコード・・５－４８
５－５ ･シリアルコントロール機能・・５－５６
５－５－１　シリアルコントロール機能について ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・５－５６
５－５－２　シリアルコントロールドライバについて ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・５－５６
５－５－３　SciCtlデバイス・・・５－５７
５－５－４　SciCtlリファレンス・・・５－５８
５－５－５　サンプルコード・・・５－６０

第６章　システムリカバリ

６－１ ･リカバリＳＤの準備・・・６－１
６－２ ･リカバリＳＤの起動・・６－１
６－３ ･システムの復旧・・・６－２

付録

A－1　マイクロソフト製品の組込み用OS（Embedded）について ・・・・・・・・・・・・・・・・・・・・・・・・・・１
A－2　参考文献・・・３

はじめに

この度は、アルゴシステム製品をお買い上げいただきありがとうございます。

弊社製品を安全かつ正しく使用していただくために、お使いになる前に本書を十分に理解していただくようお願い申し上げます。

１）　お願いと注意

本書では、Algo Smart Panel 用 Windows Embedded CE 6.0 について説明します。

Windows Embedded CE 6.0 は、マイクロソフト社によって開発された組み込み機器向けの 32 ビットのマルチタスク/マルチスレッドリアルタイムオペレーティングシステムです。組込み用 OS のため通常の PC 用 Windows とは動作が異なることがあります。詳しくは、「付録 マイクロソフト製品の組込み用 OS について」を参照してください。

本書は、アプリケーション開発、専用ドライバ仕様などの専門的な内容を含んでいます。これらの内容は、Windows アプリケーション開発、デバイス制御プログラミングに関する技術を必要とします。これらのプログラミング技術情報については、Microsoft Developer Network Library (MSDN Library)、市販の解説書などを参考にしてください。

２）　対応機種について

本書では、Algo Smart Panel AP-4410/5410/6410/6500/7500 について説明しています。その他の機種については、それぞれの機種に対応するマニュアルを用意しております。機種に対応したマニュアルを参照してご使用ください。

第１章　概要

本章では、Algo Smart Panel 用 Windows Embedded CE 6.0 の概要について説明します。

１－１　　機能と特長

１－１－１　Algo Smart Panel用Windows Embedded CE 6.0 とは

Windows Embedded CE 6.0 は、マイクロソフト社によって開発された組み込み機器向けの 32 ビットのマルチタスク/マルチスレッドリアルタイムオペレーティングシステムです。コンポーネント形式を採用し、汎用的な機能がテクノロジコンポーネントとして提供されています。ターゲットの機器に合わせてコンポーネントを選択することにより、より最適な組込み機器 OS へカスタマイズすることができます。

Algo Smart Panel 用 Windows Embedded CE 6.0 は、Algo Smart Panel 用にカスタマイズされた Windows Embedded CE 6.0 です。Algo Smart Panel 用に選定したコンポーネント、オンボード搭載デバイス用のドライバ及び設定ツールで構成されています。

１－１－２　機能と特長

Algo Smart Panel 用 Windows Embedded CE 6.0 は、Windows Embedded CE 6.0 の標準機能にオンボードに搭載されたデバイスのサポートが追加されています。このため、他の Windows で動作しているアプリケーションならば容易に移植することが可能です。

表 1-1-2-1 に Algo Smart Panel 用 Windows Embedded CE 6.0 に搭載されている主な機能を示します。

表 1-1-2-1. Algo Smart Panel用Windows Embedded CE 6.0 の主な機能

項目	機能
アプリケーションおよびサービスの開発	C ライブラリおよびランタイム
	LDAP クライアント
	SOAP Toolkit クライアント サーバー
	Standard SDK for Windows Embedded CE
	String Safe ユーティリティの関数
	MSXML 3.0 XML SAX XML クエリ言語 (XQL)
	アクティブ・テンプレート・ライブラリ(ATL)
	OBEX クライアント
	COM
	メッセージキュー(MSMQ)
	.NET Compact Framework 2.0
インターナショナル	各国語サポート(NLS)
	日本語サポート Monotype Imaging AC3 フォント圧縮 MS ゴシック、MS P ゴシックおよび MS UI Gothic かなソフトキーボード ローマ字/英語のソフトキーボード IME 3.1
	入力方式マネージャ(IMM)
インターネット・クライアント・サービス	Internet Explorer HTML/DHTML API
	Internet Explorer ブラウザ・コントロール・ホスト
	Internet Explorer 複数言語対応の基本 API

	URL モニカサービス
	Windows インターネットサービス
	コントロールパネルの[インターネットオプション]
	JScript 5.6
	Internet Explorer 6.0 サンプルブラウザ
グラフィック技術とマルチメディア技術	静止画像 Codec サポート(エンコードおよびデコード) エンコード: BMP, GIF, JPG, PNG デコード: BMP, GIF, JPG, PNG
	Windows Media Player
	WMA および MP3 ストリーミング
	WMA および MP3 ローカル再生
	オーディオ Codecs およびレンダラ MP3 Codec MPEG-1 レイヤ 1 および 2 オーディオ Codec Wave/AIFF/au/snd ファイルパーサー WMA Codec WMA Voice Codec
	ストリーミング メディアの再生
	ビデオコーデックおよびレンダラ DirectShow ビデオ レンダラ MPEG-1 ビデオ Codec WMV/MPEG-4 ビデオ Codec ビデオ/イメージ圧縮マネージャ
	メディアフォーマット MPEG-1 パーサー/スプリッタ
コア OS サービス	USB ホストサポート USB ヒューマン入力デバイス(HID)クラスドライバ USB ホストサポート USB 記憶域クラスドライバ
	Windows Embedded CE ドライバ開発キットサポートライブラリ
	インターネット機器(IABASE)のサポート
	カーネルモードドライバ用 UI プロキシ
	カーネル機能 FormatMessage API ターゲットコントロールサポート(Shell.exe) ファイバ API メッセージ キュー ポイント・ツー・ポイント メモリマップファイル
	シリアルポート・サポート
	ディスプレイ・サポート
	デバイス・マネージャ
	UI ベースの通知
	電源管理(完全)
シェルおよびユーザーインターフェイス	グラフィックス、ウィンドウおよびイベント 最小 GDI 構成 最小 GWES 構成 最小ウィンドウマネージャ構成 最小入力構成

	シェル 　AYGShell API セット 　標準のシェル 　コマンド・プロセッサ 　コンソール・ウィンドウ
	ユーザーインターフェイス 　コントロールパネル・アプレット 　ソフトウェアベースの入力パネルドライバ 　タッチ スクリーン(スタイラス) 　ネットワーク・ユーザーインターフェイス 　マウス 　共通コントロール 　共通ダイアログサポート
セキュリティ	高度な暗号化プロバイダのある暗号化サービス(CryptoAPI 1.0) 　証明書(CryptoAPI 2.0)
	資格情報マネージャ
	認証サービス(SSPI) 　NTLM 　Schannel(SSL/TLS)
ファイルシステムおよびデータストア	システムパスワード
	CEDB データベースエンジン
	RAM および ROM に適用されるファイルシステム
	Hive ベースのレジストリ
	圧縮
	記憶域マネージャ 　FAT ファイルシステム 　パーティションドライバ 　リリースディレクトリ・ファイルシステム 　記憶域マネージャ・コントロールパネル・アプレット
フォント	Courier New(サブセット1_30)
	Tahoma(サブセット1_07)
	Wingding
通信サービスおよびネットワーク	サーバー 　FTP サーバー 　Telnet サーバー 　Web サーバー 　コアサーバー・サポート 　ファイルサーバー(SMB/CIFS)
	ネットワーク ― ローカルエリア・ネットワーク 　ワイヤード(有線)ローカルエリア・ネットワーク(802.3、802.5)
	ネットワーク ― 広域ネットワーク(WAN) 　Telephony API(TAPI 2.0) 　ダイヤルアップ・ネットワーク (RAS/PPP) 　自動ダイヤル

	ネットワーク － 全般 NDIS ユーザーモード I/O プロトコルドライバ TCP/IP Windows ネットワーク API/リダイレクタ(SMB/CIFS) Winsock サポート ネットワークドライバ・アーキテクチャ (NDIS) ネットワーク・ユーティリティ(IpConfig、Ping、Route) 拡張 DNS クエリおよびアップデート(DNSAPI)

１－１－３　デバイスドライバ構成

Algo Smart Panel 用 Windows Embedded CE 6.0 は、各種デバイスドライバが標準搭載されています。標準搭載デバイスドライバの一覧を表 1-1-3-1 に示します。

各デバイスドライバの詳細は「第 5 章 組込みシステム機能ドライバ」を参照ください。

表 1-1-3-1. Algo Smart Panel用Windows Embedded CE 6.0 の標準搭載デバイスドライバ

デバイスドライバ	説明
グラフィック	・グラフィックドライバ
バックライト	・バックライトドライバ 輝度の調整、バックライトの ON/OFF を行うことができます。
ブザー	・ブザードライバ ブザーの周波数の変更、ブザーの ON/OFF、タッチパネルのタッチ音の ON/OFF を行うことができます。
シリアル	・オンボードシリアルポート 2 ポート(RS-232C/RS-422/RS-485)ドライバ ・シリアルポート切替えドライバ 各シリアルポートを RS-232C/RS-422/RS-485 に切替えることができます。
タッチパネル	タッチパネルドライバ
USB	・USB1.1/2.0 対応ホストドライバ ・HID ドライバ(マウス、キーボード対応) ・USB 記憶域クラスドライバ
LAN	・オンボード 100/10BASE-T の LAN ドライバ
汎用 I/O	・汎用入出力アクセスドライバ IN6 点、OUT4 点の ON/OFF を入出力することができます。 IN0BIT の ON トリガで電源リセットする機能があります。 IN1BIT の ON トリガで登録されたイベントをシグナル状態に設定することができます。
記憶装置	・512MByte NAND フラッシュメモリアクセスドライバ ・RAM ディスクドライバ（メインメモリの一部をストレージして使用） ・512KByte SRAM メモリアクセスドライバ ・SD/SDHC カードアクセスドライバ

１－２　　システム構成

１－２－１　　ファイルシステム構成

Windows Embedded CE 6.0 のファイルシステムは、他の Windows OS とは異なり、シングルルートファイルシステムとなっています。Windows Embedded CE 6.0 のファイルシステムの大きな特徴の 1 つです。Windows Embedded CE 6.0 は、他の Windows OS のように C:や D:といったドライブはありません。ハードディスクやコンパクトフラッシュ、USB フラッシュメモリなどの外部ストレージは、ファイルシステムルート「¥」以下に「¥ハード ディスク」、「¥ハード ディスク 2」、「¥Strage Card」のようにマウントされます。

OS 本体である OS システムファイルは、CE ランタイムイメージ（NK.bin）に格納されています。OS システムファイルは起動時に展開され「¥Windows」からアクセス可能となります。Algo Smart Panel 用 Windows Embedded CE 6.0 では、ファイルシステムルートとして NAND フラッシュメモリがマウントされます。OS システムファイル以外のファイルは、NAND フラッシュメモリ上に保存されます。

図 1-2-1-1 にファイルシステム構成の概要を示します。

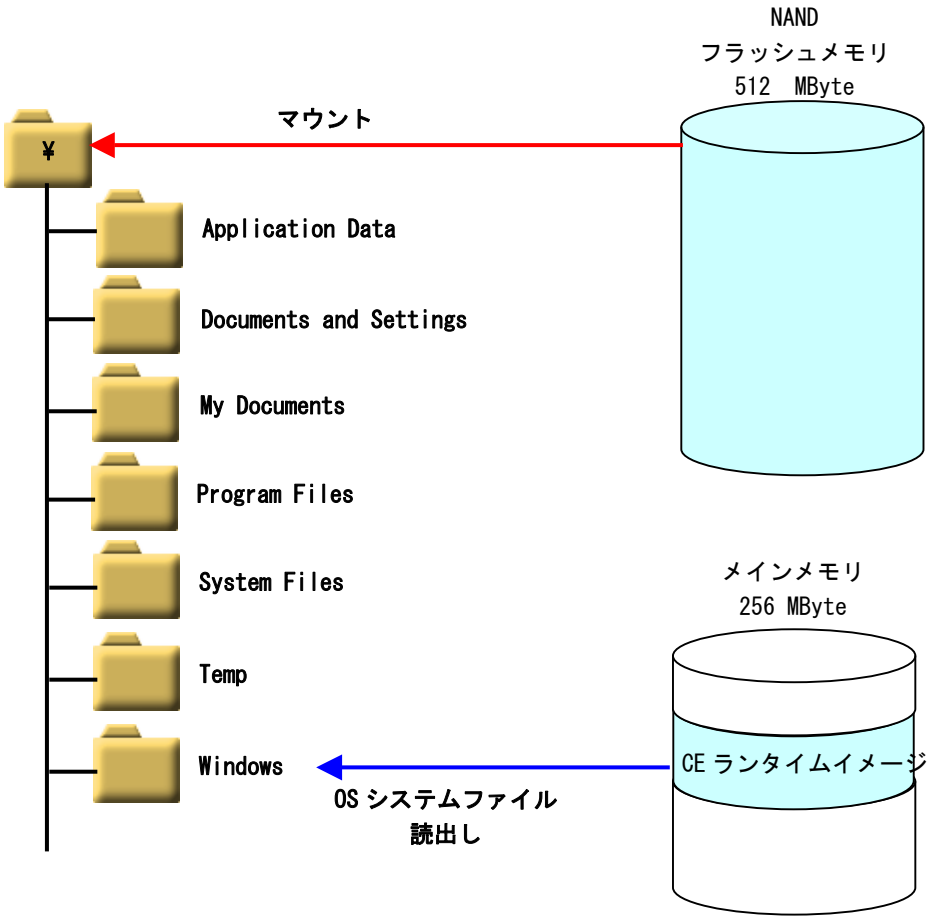

図 1-2-1-1.　ファイルシステム構成

１－２－２　フォルダ・ファイル構成

Algo Smart Panel 用 Windows Embedded CE 6.0 のフォルダ構成は、Windows Embedded CE 6.0 の標準的なフォルダ構成となっています。

ファイルシステムルート上のフォルダ、ファイルの構成を表 1-2-2-1 に示します。

表 1-2-2-1. Algo Smart Panel用Windows Embedded CE 6.0 フォルダ・ファイル構成

名前	種類	内容
Application Data	フォルダ	標準フォルダ。 アプリケーションデータを格納します。
Document and Settings	フォルダ	標準フォルダ。 ユーザー設定データを格納します。
My Documents	フォルダ	標準フォルダ。 ユーザーデータを格納します。
Program Files	フォルダ	標準フォルダ。 アプリケーションを格納します。
RAM Disk	ストレージ	Algo Smart Panel 専用 メインメモリの一部をストレージとしてマウント。
SRAM Disk	ストレージ	Algo Smart Panel 専用 SRAM デバイスをストレージとしてマウント。
System Files	フォルダ	Algo Smart Panel 専用 Algo Smart Panel 用のシステムファイルを格納します。
Temp	フォルダ	標準フォルダ。 テンポラリフォルダ。
Windows	フォルダ	標準フォルダ。 OS システムファイルを格納します。
ネットワーク	フォルダ	標準フォルダ。 ネットワークフォルダ。
NK.bin	ファイル	Algo Smart Panel 専用 CE ランタイムイメージ本体。起動時にメインメモリ上に展開されます。
コントロール パネル.lnk	ファイル	標準ファイル。 コントロールパネルへのショートカット。

※ システム属性、隠し属性のフォルダ・ファイルは表記していません。

第２章　システムの操作

本章では、Algo Smart Panel 用 Windows Embedded CE 6.0 の基本的な操作方法について説明します。

２－１　OSの起動と終了

２－１－１　OSの起動

本体の電源を投入します。電源が入ると POWER LED が点灯します。Windows Embedded CE 6.0 が起動します。正常に起動すると、図 2-1-1-1 の様なデスクトップが表示されます。

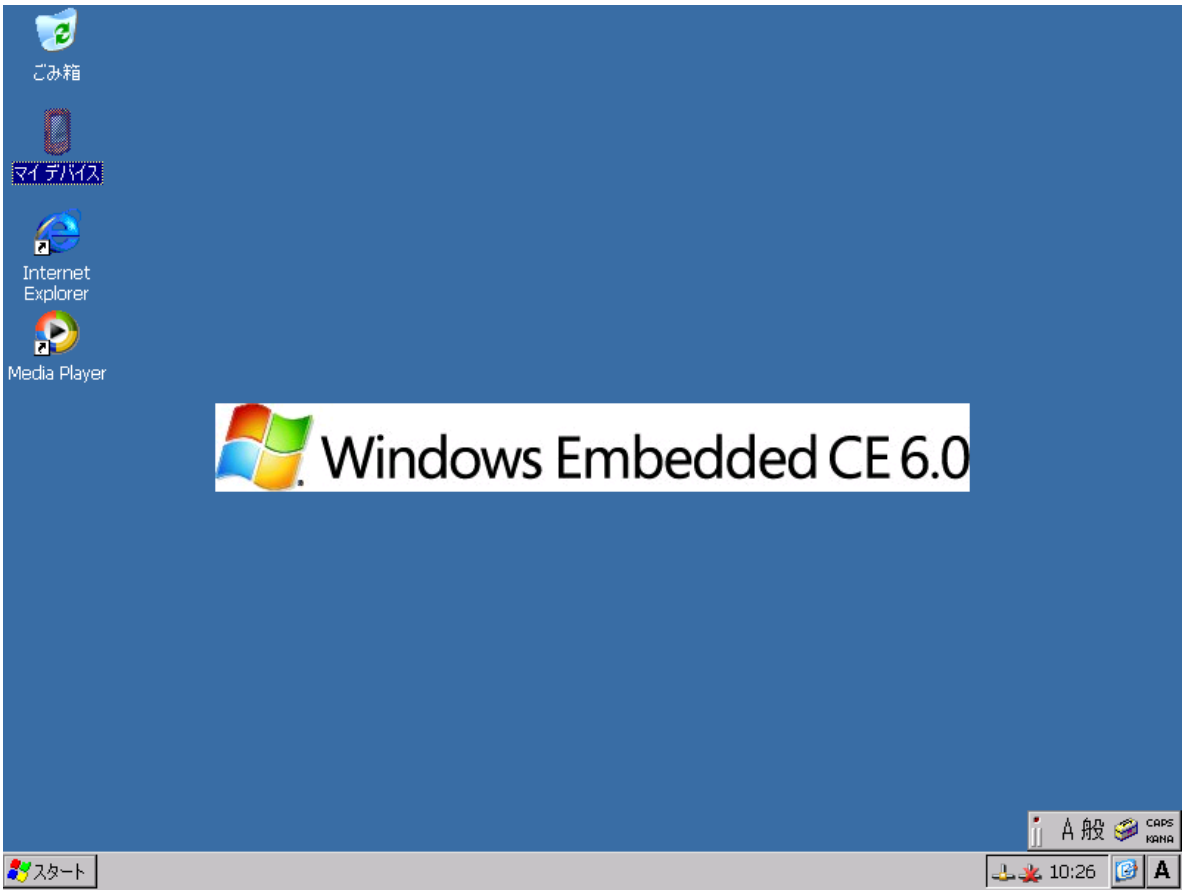

図 2-1-1-1. デスクトップ

２－１－２　OSの終了

ファイルシステムへのファイル書込み中でないことを確認します。本体の電源を落とします。電源が落ちると POWER LED が消灯します。

２－２　　ASD Config Tool

２－２－１　ASD Config Tool

「ASD Config Tool」は、Algo Smart Panel 専用のコントロールパネルアプリケーションです。スタートメニューから[コントロール パネル]を開き、[ASD Config Tool]から起動できます。

設定内容を表 2-2-1-1 に示します。

表 2-2-1-1. ASD Config Tool 設定/表示内容

タブ	設定/表示内容
LCD Setting	LCD バックライトの輝度を調整することができます。
Buzzer Setting	タッチブザーの設定を行うことができます。
Board Information	ハードウェア、ソフトウェアのバージョンを確認することができます。

２－２－２　LCD Setting

LCD バックライトの輝度を調整できます。

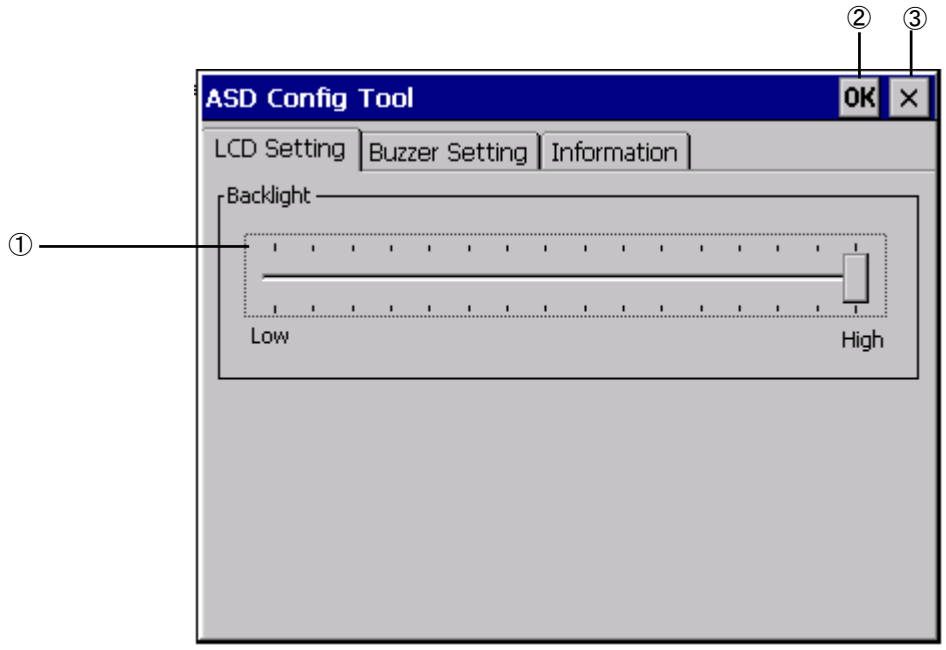

図 2-2-2-1. ASD Config – LCD Setting

① バックライトの輝度を 16 段階で調整できます。
② 設定を保存、適用して終了します。
③ 設定を破棄して終了します。

２－２－３　Buzzer Setting

ブザーの設定を行うことができます。

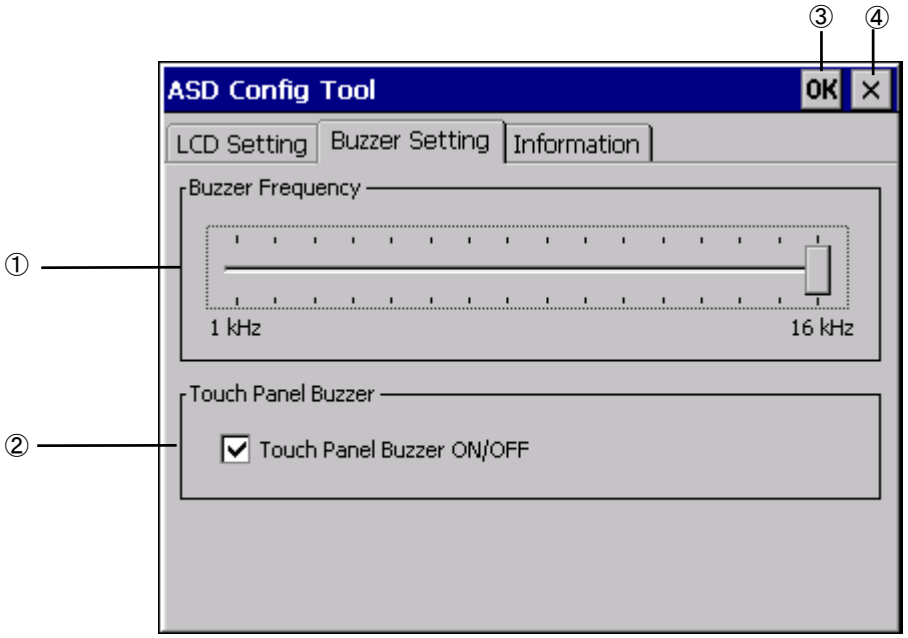

図 2-2-3-1. ASD Config – Buzzer Setting

① ブザー周波数を設定できます。(1～16kHz)
② タッチパネルブザーの ON/OFF を設定できます。(チェックあり: ON、チェックなし: OFF)
③ 設定を保存、適用して終了します。
④ 設定を破棄して終了します。

２－２－４　Board Information

ハードウェア、ソフトウェアのバージョンを確認することができます。

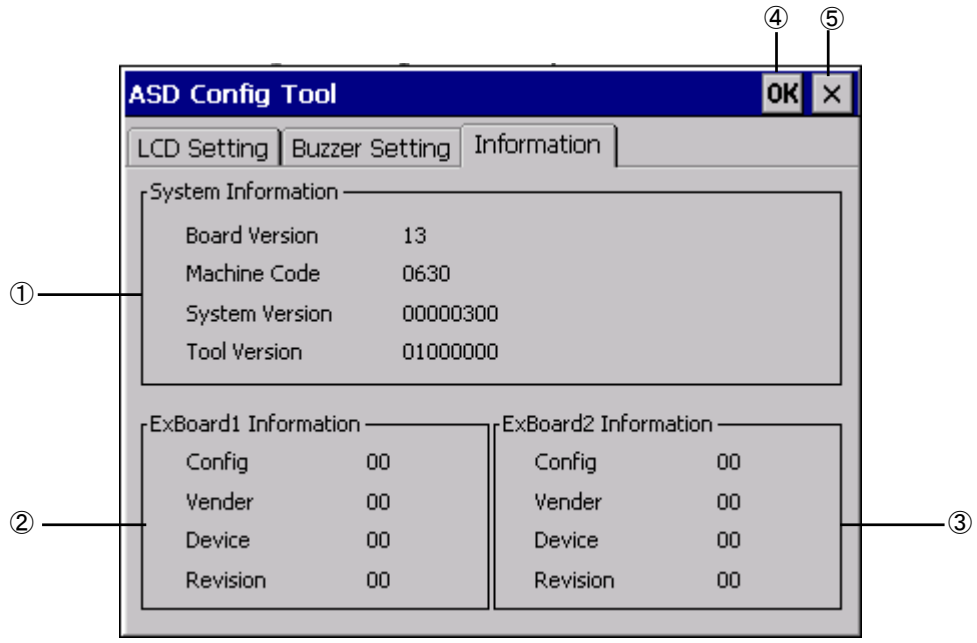

図 2-2-4-1. ASD Config – Board Information

① 各種バージョンを表示します。
② 拡張ボード 1 の情報を表示します。
③ 拡張ボード 2 の情報を表示します。
④ 設定を保存、適用して終了します。
⑤ 設定を破棄して終了します。

２－２－５　初期値

「ASD Config Tool」の設定初期値を表 2-2-5-1 に示します。

表 2-2-5-1. ASD Config Tool 設定初期値

タブ	設定項目	初期値
LCD Setting	Backlight	High 側最大値
Buzzer Setting	Buzzer Frequency	10kHz
	Touch Panel Buzzer	ON

２－３　　アプリケーションの自動起動

Algo Smart Panel では、OS 起動と同時にアプリケーションを起動させる方法として、スタートアップフォルダを利用した方法と、Algo Smart Panel 専用スタートアップ機能を使用した方法の 2 つ方法が利用できます。

２－３－１　スタートアップフォルダによる自動起動

PC 用の Windows OS と同様にスタートアップフォルダを利用した自動起動が可能です。スタートアップフォルダに実行可能ファイル（EXE ファイル、BAT ファイル、ショートカットなど）を保存します。次回起動時から OS 起動時に保存したファイルが実行されるようになります。

表 2-3-1-1 にスタートアップフォルダのパスを示します。

表 2-3-1-1. スタートアップフォルダパス

スタートアップフォルダパス
¥Windows¥スタートアップ

２－３－２　Algo Smart Panel専用スタートアップ機能による自動起動

Algo Smart Panel では、Algo Smart Panel 専用に独自のスタートアップ機能を用意しています。Algo Smart Panel 専用スタートアップ機能では、OS 起動時に実行するプログラムを設定ファイル（Startup.ini）で指定します。

表 2-3-2-1 に設定ファイルを示します。設定ファイルを作成し指定のフォルダに保存してください。

表 2-3-2-1. Algo Smart Panelスタートアップ機能 設定ファイル

ファイル名	格納フォルダ	設定ファイルフルパス
Startup.ini	¥System Files	¥System Files¥Startup.ini

● Startup.ini 書式

表 2-3-2-2 に Startup.ini の書式を示します。

表 2-3-2-2. Startup.ini書式

セクション名	キー名	内容
CONFIG	COUNT	起動処理の数を設定します。[0, 1, … N] 設定された数に従い、起動処理を[START01]、[START02]…と順に行います。 0 を指定した場合は起動処理は行われません。
START**	FROM	コピー元ファイル名を指定します。 ファイル名はフルパスで指定しください。
	TO	コピー先ファイル名を指定します。 ファイル名はフルパスで指定しください。
	START	起動するプログラムを指定します。 ファイル名はフルパスで指定しください。 ¥Windows 以下のファイルの場合はファイル名のみも可能です。
	PARAM	引数を指定します。
	WAIT	次の起動処理後の待機時間を指定します。 負の整数: 自セクションのプログラムが終了するまで待機します。 0: 待機しません。 正の整数: 指定時間待機します。[msec]

● 自動起動例 1

自動起動の例をリスト 2-3-2-1 に示します。この例では、以下の順で起動処理が行われます。

① 「¥User¥ProgA.exe」を「¥SRAM Disk¥Prog.exe」にコピーします。
② 「¥SRAM Disk¥Prog.exe」を起動します。
③ 「¥SRAM Disk¥Prog.exe」を起動後 100msec 待機します。
④ 「¥SRAM Disk¥ProgB.exe」を引数「100」を与えて起動します。
⑤ 「¥SRAM Disk¥ProgB.exe」が終了するまで待機します。
⑥ 「¥SRAM Disk¥ProgC.exe」を起動します。待機はしません。

リスト 2-3-2-1. 自動起動例 1 [Startup.ini]

```
[CONFIG]
COUNT=3

[START01]
FROM=¥User¥ProgA.exe
TO=¥SRAM Disk¥Prog.exe
START=¥SRAM Disk¥Prog.exe
WAIT=100

[START02]
START=¥SRAM Disk¥ProgB.exe
PARAM=100
WAIT=-1

[START03]
START=¥SRAM Disk¥ProgC.exe
WAIT=0
```

①, ②, ③

④, ⑤

⑥

● 自動起動例 2

バッチファイルを自動起動する場合の例をリスト 2-3-2-2 に示します。バッチファイルは、¥Windows¥cmd.exe に引数としてバッチファイルを渡すことで実行することができます。

リスト 2-3-2-2. 自動起動例 2 [Startup.ini]

```
[CONFIG]
COUNT=2

[START01]
START=cmd.exe
PARAM=/c "¥User¥init.bat"
WAIT=-1

[START02]
START=¥User¥ProgA.exe
```

２－４　　ネットワークサーバ

Algo Smart Panel 用 Windows Embedded CE 6.0 では、ネットワークサーバとして Windows 共有ファイルサーバ、FTP サーバ、TELNET サーバ、WEB サーバを組み込んでいます。本項では、ネットワークサーバを使用する基本的な方法を説明します。ネットワークサーバの設定には、レジストリの変更が必要です。レジストリの変更は、アプリケーションを作成するか、リモートツールを使用します。リモートツールの使用方法は、「4-6 リモートツール」を参照してください。

２－４－１　Windows共有ファイルサーバ

Algo Smart Panel 上のフォルダを Windows 共有で共有する例を示します。表 2-4-1-1 で示す共有環境を設定します。

表 2-4-1-1. Windows共有ファイルサーバ 共有環境

共有デバイス名（共有 PC 名）	cedevice
共有フォルダ名	ShareTemp
共有フォルダパス	¥Temp

● レジストリ設定

表 2-4-1-1 で示す共有環境のレジストリ設定を表 2-4-1-2 に示します。

設定変更後は再起動を行ってください。再起動後に設定が有効になります。

レジストリ設定内容の詳細は、MSDN Library（http://msdn.microsoft.com/en-us/library/ms898959.aspx）を参照してください。

表 2-4-1-2. Windows共有ファイルサーバ レジストリ設定

キー	[HKEY_LOCAL_MACHINE]-[Ident]		
	名前	種類	データ
	Name	REG_SZ	cedevice
	OrigName	REG_SZ	WindowsCE
キー	[HKEY_LOCAL_MACHINE]-[Services]-[Smbserver]		
	名前	種類	データ
	AdapterList	REG_SZ	*
	dll	REG_SZ	Smbserver.dll
	Keep	REG_DWORD	1
	Order	REG_DWORD	9
キー	[HKEY_LOCAL_MACHINE]-[Services]-[Smbserver]-[Shares]		
	名前	種類	データ
	UseAuthentication	REG_DWORD	0
キー	[HKEY_LOCAL_MACHINE]-[Services]-[Smbserver]-[Shares]-[ShareTemp]		
	名前	種類	データ
	Path	REG_SZ	¥Temp
	Type	REG_DWORD	0
	UserList	REG_SZ	@*;

２－４－２　FTPサーバ

Algo Smart Panel で FTP サーバを動作させる例を示します。表 2-4-2-1 で示す動作環境を設定します。

表 2-4-2-1. FTPサーバ 動作環境

FTP ルート	¥Temp
匿名ユーザ ログイン	許可
匿名ユーザ ファイル書込み	許可
匿名ユーザ 仮想ルート	有効

● レジストリ設定

表 2-4-2-1 で示す FTP サーバ動作環境のレジストリ設定を表 2-4-2-2 に示します。
設定変更後は再起動を行ってください。再起動後に設定が有効になります。
レジストリ設定内容の詳細は、MSDN Library (http://msdn.microsoft.com/en-us/library/ms898962.aspx) を参照してください。

表 2-4-2-2. FTPサーバ レジストリ設定

キー	[HKEY_LOCAL_MACHINE]-[COMM]-[FTPD]		
	名前	種類	データ
	IsEnabled	REG_DWORD	1
	UseAuthentication	REG_DWORD	0
	UserList	REG_SZ	@*;
	AllowAnonymous	REG_DWORD	1
	AllowAnonymousUpload	REG_DWORD	1
	AllowAnonymousVroot	REG_DWORD	1
	DefaultDir	REG_SZ	¥Temp

２－４－３　TELNETサーバ

Algo Smart Panel で TELNET サーバを動作させる例を示します。表 2-4-3-1 で示す動作環境を設定します。

表 2-4-3-1. TELNETサーバ 動作環境

ユーザー	なし
パスワード	なし

● レジストリ設定

表 2-4-3-1 で示す TELNET サーバ動作環境のレジストリ設定を表 2-4-3-2 に示します。
設定変更後は再起動を行ってください。再起動後に設定が有効になります。
レジストリ設定内容の詳細は、MSDN Library (http://msdn.microsoft.com/en-us/library/ms899559.aspx) を参照してください。

表 2-4-3-2. TELNETサーバ レジストリ設定

キー	[HKEY_LOCAL_MACHINE]-[COMM]-[TELNETD]		
	名前	種類	データ
	IsEnabled	REG_DWORD	1
	UseAuthentication	REG_DWORD	0
	UserList	REG_SZ	@*;

２－４－４　WEBサーバ

Algo Smart Panel で WEB サーバを動作させる例を示します。

● レジストリ設定

WEB サーバのレジストリ設定を表 2-4-4-1 に示します。

設定変更後は再起動を行ってください。再起動後に設定が有効になります。

レジストリ設定内容の詳細は、MSDN Library (http://msdn.microsoft.com/en-us/library/ms899583.aspx) を参照してください。

表 2-4-4-1. WEBサーバ レジストリ設定

キー	[HKEY_LOCAL_MACHINE]-[COMM]-[HTTPD]		
	名前	種類	データ
	IsEnabled	REG_DWORD	1
	UseAuthentication	REG_DWORD	0
	DirBrouse	REG_DWORD	1

第３章　Algo Smart Panelについて

本章では、Algo Smart Panel 本体に搭載されている機能について説明します。

３－１　Algo Smart Panelに搭載された機能について

Algo Smart Panel にはグラフィック表示機能、通信機能、USB 機能などが搭載されています。これらの機能は Windows の標準インターフェースを使用して操作することができます。また、Algo Smart Panel では組込みシステム向けに独自機能が追加されています。組込みシステム機能は、専用ドライバを使用して操作することが可能です。

シリーズ本体（AP-7500）の外形図を図 3-1-1 に示します。各部名称を表 3-1-1 に示します。

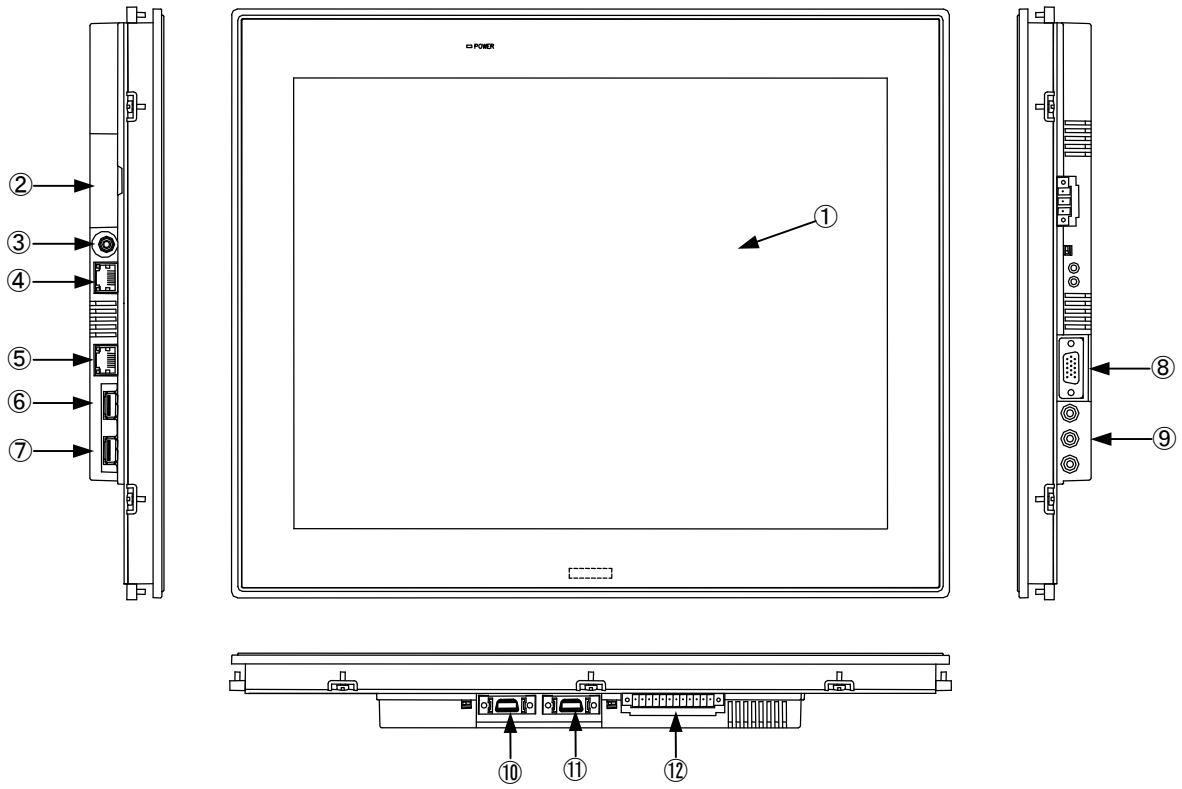

図 3-1-1. 外形図（AP-7500）

表 3-1-1. 各部名称

No.	名称	機能	説明
①	液晶タッチパネル	グラフィック タッチパネル	画面表示を行います。(65536 色) タッチパネルはポインティングデバイスとして使用可能です。
②	SD カード インターフェース	SD/SDHC カード	記憶領域として SD/SDHC カードを使用することができます。
③	ビデオ入力	ビデオキャプチャ	Windows Embedded CE 6.0 では使用できません。
④ ⑤	ネットワーク インターフェース	有線 LAN	ネットワークポートとして使用できます。 LAN1: SMSC91181 LAN2: SMSC91182
⑥ ⑦	USB インターフェース	USB ポート	USB1.1/2.0 の機器を接続することができます。
⑧	VGA 出力	グラフィック	画面表示を行います。 アナログディスプレイを接続可能です。
⑨	ライン入力 音声出力 マイク入力	サウンド	Windows Embedded CE 6.0 では使用できません。
⑩ ⑪	シリアル インタフェース	シリアルポート	シリアル通信が行えます。 組込みシステム機能のシリアルコントロール機能と併用すると RS-232C/422/485 の通信を行うことができます。 SIO1: COM1(RS-232C), COM3(RS-422/485) SIO2: COM2(RS-232C), COM4(RS-422/485)
⑫	DIO インターフェース	汎用入出力	汎用の入出力です。 入力 6 点、出力 4 点を制御できます。

３－２　　Windows標準インターフェース対応機能

本項では、Algo Smart Panel に搭載されている Windows 標準インターフェース対応機能について説明します。

３－２－１　グラフィック

一般的な Windows と同様に、デスクトップ表示、アプリケーション表示を行います。本シリーズでは LCD（図 3-1-1 ①）への表示と、VGA 端子（図 3-1-1 ⑧）を使用してモニタ、プロジェクタへの表示が可能となっています。

スタートメニューから[コントロールパネル]を選択し、[画面]を起動して設定を行います。

３－２－２　タッチパネル

タッチパネルをタッチすることにより､マウスなどのポインティングデバイス操作を行うことができます。本シリーズに搭載されたタッチパネルには以下の様な特徴があります。

- タッチパネルを操作することでマウスと同等な操作環境を実現することができます。
- マウスとの共存が可能なため、特別な設定を行うことなくタッチパネル、マウス双方を切替え使用することができます。
- クリック操作に関係する詳細な設定、精密なキャリブレーション機能などを提供します。

ダブルタップの設定､タッチスクリーンの補正機能などは[スタートメニュー]-[設定]-[コントロールパネル]-[スタイラス]から行ってください。

３－２－３　シリアルポート

一般的な Windows と同様に、COM ポートとしてシリアル通信に使用することができます。

搭載されている COM ポートの一覧を表 3-2-3-1 に示します。

表 3-2-3-1. シリアルポート

COM ポート	説明
COM1	SIO1（図 3-1-1 ⑩） RS-232C のシリアル通信に使用できます。組込みシステム機能のシリアルコントロール機能を使用することで、RS-232C/422/485 を切換えることができます。 **※　RS-422/485 で使用する場合は COM3 を使用してください。COM1 と COM3 の併用はできません。**
COM2	SIO2（図 3-1-1 ⑪） RS-232C のシリアル通信に使用できます。組込みシステム機能のシリアルコントロール機能を使用することで、RS-232C/422/485 を切換えることができます。 **※　RS-422/485 で使用する場合は COM4 を使用してください。COM2 と COM4 の併用はできません。**
COM3	SIO1（図 3-1-1 ⑩） RS-422/485 のシリアル通信に使用できます。組込みシステム機能のシリアルコントロール機能を使用することで、RS-232C/422/485 を切換えることができます。 **※　RS-232C で使用する場合は COM1 を使用してください。COM1 と COM3 の併用はできません。**
COM4	SIO2（図 3-1-1 ⑪） RS-422/485 のシリアル通信に使用できます。組込みシステム機能のシリアルコントロール機能を使用することで、RS-232C/422/485 を切換えることができます。 **※　RS-232C で使用する場合は COM2 を使用してください。COM2 と COM4 の併用はできません。**

３－２－４　有線LAN

100/10BASE-T イーサネット対応の有線 LAN ポートが 2 ポート用意されています。一般的な Windows と同様にネットワークポートとして使用することができます。表 3-2-4-1 にネットワーク名称と外部コネクタとの対応を示します。

表 3-2-4-1. 有線LANポート

ネットワーク名称	説明
SMSC91181	LAN1 （図 3-1-1 ④）
SMSC91182	LAN2 （図 3-1-1 ⑤）

３－２－５　USBポート

USB1.1/2.0 対応の USB ポートを外部コネクタとして 2 ポート用意しています。一般的な Windows と同様に USB 機器を接続して使用することができます。接続する USB 機器のドライバは、別途用意してください。

３－２－６　SD/SDHCカード

SD/SDHC に対応したカードスロットを搭載しています。データ記憶領域として SD/SDHC カードを使用することができます。

SD カードは、ストレージとして「¥Storage Card」にマウントされます。デスクトップからは、[マイデバイス]-[Storage Card]で SD/SDHC カードを使用することができます。

３－２－７　RAMディスク

メインメモリの一部をストレージとして「¥RAM Disk」にマウントしています。通常のストレージデバイスと同様にファイルの読書きが可能です。出荷状態の RAM ディスクの容量は 8MByte です。表 3-2-7-1 のレジストリ値を変更することによって容量を変更できます。設定は次回起動時に有効となります。再起動後、RAM ディスクが設定した容量になります。

レジストリの変更は、アプリケーションを作成するか、リモートツールを使用します。リモートツールの使用方法は、「4-6 リモートツール」を参照してください。

※　RAM ディスクは起動時に初期化されます。格納したファイルは消えてしまうので注意してください。

表 3-2-7-1. RAMディスク容量設定

キー	[HKEY_LOCAL_MACHINE]-[Drivers]-[BuiltIn]-[RAMDisk]		
	名前	種類	データ
	Size	REG_DWORD	RAM ディスク容量（1024 の倍数値を設定） 初期値: 8388608

３－３　　組込みシステム機能

Algo Smart Panel には、組込みシステム向けに独自の機能が搭載されています。本項では、組込みシステム機能について説明します。組込みシステム機能の一覧を表 3-3-1 に示します。

Algo Smart Panel 用 Windows Embedded CE 6.0 では、組込みシステム機能を使用するためにドライバを用意しています。ドライバの使用方法は「第 5 章 組込みシステム機能ドライバ」を参照してください。

表 3-3-1. 組込みシステム機能

機能	説明
ブザー	ブザーを鳴らすことができます。 タッチパネルのタッチ時のブザーも制御できます。
汎用入出力	汎用の入出力です。 入力 6 点、出力 4 点を制御できます。 （図 3-1-1 ⑫）
LCD バックライト	LCD バックライトを制御できます。 バックライトの ON/OFF、輝度調整ができます。
RAS 機能	汎用入力の IN0、IN1 にリセット機能、割込み機能があります。 IN0 リセット、IN1 割込みの制御ができます。
シリアルコントロール機能	シリアルポートの RS-232C/422/485 の切替えができます。
バックアップ SRAM	バックアップ付き SRAM を制御することができます。

３－３－１　　ブザー

ブザーの ON/OFF、ブザーの周波数の変更(16 段階)、タッチパネルのタッチ音の ON/OFF を行うことができます。

アプリケーションでブザーを鳴らすことができます。タッチパネルのタッチ音設定は、「ASD Config Tool」からも設定可能です。

３－３－２　　汎用入出力

汎用入出力が搭載されています。アプリケーションから入力 6 点、出力 4 点が制御可能です。

３－３－３　　LCDバックライト

LCD のバックライトを調整することができます。バックライトの ON/OFF と輝度を変更することができます。

アプリケーションで LCD バックライトの調整が可能です。バックライトの輝度は、「ASD Config Tool」からも設定可能です。

３－３－４　　RAS機能

ハードウェアによる IN0 リセット機能、IN1 割込み機能が実装されています。

IN0 入力時にハードウェアリセットを掛けることができます。アプリケーションでこの機能の有効/無効を制御できます。

IN1 入力時に割込みを発生させることができます。アプリケーションでこの機能の有効/無効を制御できます。また、割込み発生時にイベントを受けることができます。

３－３－５　　シリアルコントロール機能

COM ポートは、RS-232C 以外に RS-422、RS-485 通信を行う事ができます。COM ポートごとに通信のタイプを RS-232C/RS-422/RS-485 で切替えることができます。詳細は「3-2-3 シリアルポート」を参照ください。

アプリケーションでシリアル通信を行う前に、通信のタイプを切換えることができます。

３－３－６　バックアップSRAM

バックアップ電池が搭載されたSRAMが実装されています。SRAMにデータの読み書きを行うことができます。

SRAMは、ストレージとしてファイルシステムに「¥SRAM Disk」としてマウントされています。ファイルシステムからアクセスすることが可能です。デスクトップからは、[マイデバイス]-[SRAM Disk]でバックアップSRAMを使用することができます。

第４章　アプリケーション開発

４－１　　アプリケーション開発について

Windows Embedded CE 5.0 まではアプリケーション開発に eMbedded Visual C++が使われていました。Windows Embedded CE 6.0 でも根本的な開発工程は変わりませんが、Windows Embedded CE 6.0 では Visual Studio 2005 を使用してアプリケーションを開発できるようになっています。

Algo Smart Panel 用 Windows Embedded CE 6.0 では、アプリケーションとしてネイティブコードアプリケーションと.NET Compact Framework 2.0 SP3 を使用したマネージドコードアプリケーションを動作させることが可能です。両アプリケーションとも Visual Studio 2005 を使用して開発することが可能です。ただし、ネイティブコードアプリケーションを作成するには Algo Smart Panel 用の SDK が必要になります。

● アプリケーションの作成

Windows オペレーティングシステムが動作している PC を使用してアプリケーションの作成を行います。アプリケーションの作成には Visual Studio 2005 を使用します。ネイティブアプリケーションを作成する場合は、作成を行う PC に Algo Smart Panel 用の SDK をインストールする必要があります。

● アプリケーションのデバッグ

Visual Studio 2005 には、開発用 PC とターゲットデバイスを接続する機能があります。開発用 PC からターゲットデバイスへのファイル転送、リモートデバッグなどを行うことができます。リモートデバッグでは、PC 上で動作する Windows アプリケーションと同じように、リモートでステップ実行や変数の監視が可能です。

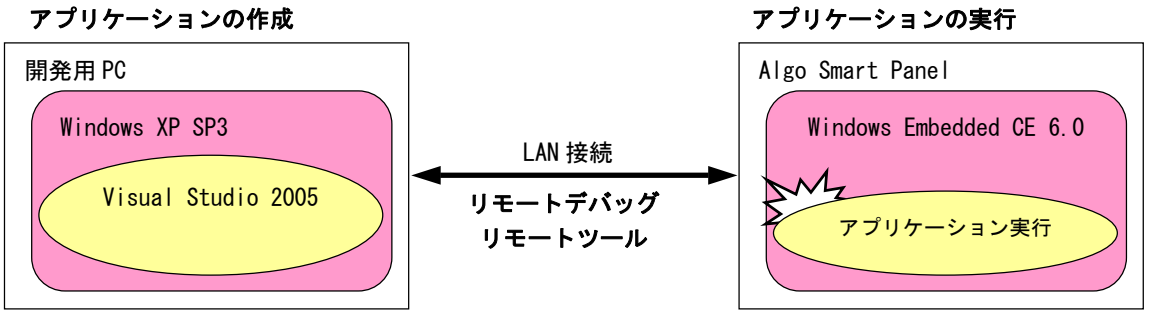

図 4-1-1. アプリケーション開発

４－２　　開発環境の構築

アプリケーションの開発には開発環境が必要です。開発用 PC と開発ツールが必要となります。

４－２－１　必要なシステム

開発用 PC は以下のシステム用件を満たす必要があります。

- CPU:　933MHz 以上（推奨: 2GHz）
- RAM:　512MByte 以上（推奨: 1GByte）
- OS:　Windows 2000 Professional Service Pack 4
  Windows XP Professional Service Pack 2
  Windows Vista Business/Ultimate
- HDD:　18GByte 以上
- DVD-ROM ドライブ
- モニタ: 1024 x 768、16Bit カラー対応

４－２－２　開発ツール

開発ツールとして以下のソフトウェアが必要となります。

- Visual Studio 2005
- Visual Studio 2005 Service Pack 1
- .NET Compact Framework Service Pack 2
- Algo Smart Panel 用 SDK（弊社提供）

Service Pack が手元にない場合は、Microsoft Update、または Microsoft Download Center を利用して入手、インストールすることができます。

「Algo Smart Panel 用 SDK」は、「Algo Smart Panel AP-4410/5410/6410/6500/7500 Windows Embedded CE 6.0 リカバリ/SDK DVD」で提供しています。表 4-2-2-1 に「Algo Smart Panel 用 SDK」のインストーラを示します。

表 4-2-2-1. Algo Smart Panel用 SDK

Algo Smart Panel 用 SDK インストーラ
<DVD-DRIVE>¥SDK¥Algo¥SDK¥G3N_SDK_RELEASE.msi

４－２－３　開発ツールのインストール

以下の順序で開発 PC に開発ツールをインストールします。

① Visual Studio 2005
[選択した機能をインストールします]ダイアログで[既定]または[すべて]を選択してインストールします。[カスタム]を選択した場合は、利用する言語の[スマートデバイスプログラマビリティ]を選択してください。

② Visual Studio 2005 Service Pack 1
Microsoft Update を利用してインストールできます。

③ .NET Compact Framework Service Pack 2
Microsoft Download Center から入手、インストールできます。

④ Algo Smart Panel 用 SDK
弊社提供のインストーラを使用してインストールしてください。

４－３　　ターゲットとの接続

本項では、アプリケーションの開発の前に、ターゲットである Algo Smart Panel との接続確認を行います。リモートデバッグ、リモートツールを使用する場合は、必ず接続確認を行ってください。

４－３－１　　IPアドレスの確認

リモートデバッグ、リモートツールの使用には、開発用 PC と Algo Smart Panel を LAN 経由で接続する必要があります。LAN ポートを使用して Algo Smart Panel をネットワークに接続し、環境に合わせて LAN の設定を行ってください。

Visual Studio 2005 と接続する前に、本体の IP アドレスを確認します。

① ステータスバーの LAN 接続アイコンをダブルクリックします。

図 4-3-1-1. LAN接続アイコン

② IP 情報のダイアログが表示されます。IP アドレスを確認します。図 4-3-1-2 に IP 情報の例を示します。例では DHCP を利用しています。

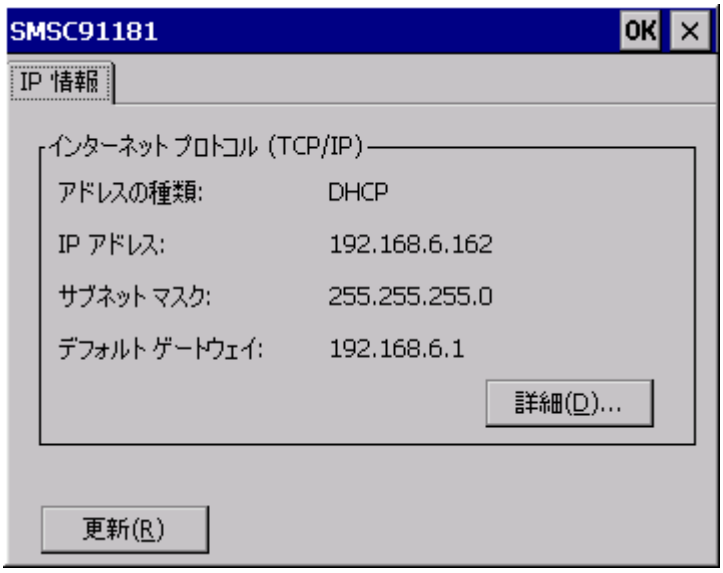

図 4-3-1-2. IP情報例

４－３－２　Visual Studio 2005 のデバイス設定

Algo Smart Panel と Visual Studio 2005 を接続するために Visual Studio 2005 の設定を行います。

① Visual Studio 2005 を起動し、[ツール]-[オプション]を選択します。
② [オプション]ダイアログが開きます。[デバイスツール]-[デバイス]を選択します。[デバイスを表示するプラットフォーム]から[ASP G3N SDK Release]を選択します。(図 4-3-2-1)

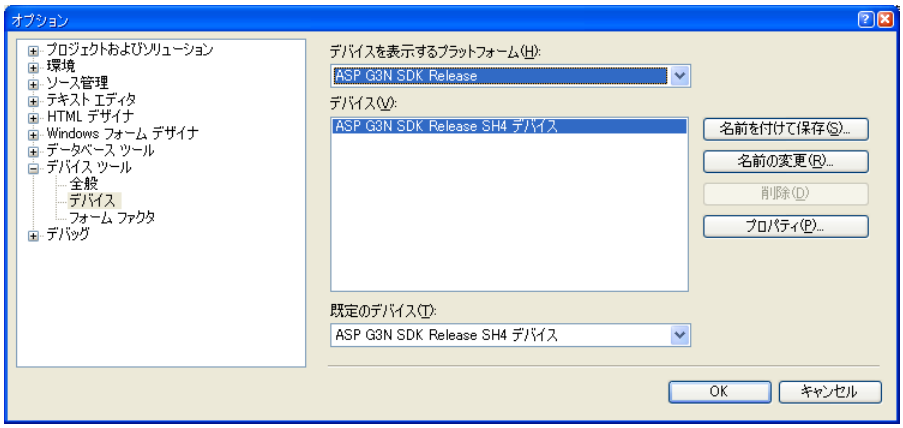

図 4-3-2-1. オプションダイアログ

③ [プロパティ]を押すと、[ASP G3N SDK Release SH4 デバイスのプロパティ]ダイアログが表示されます。(図 4-3-2-2)

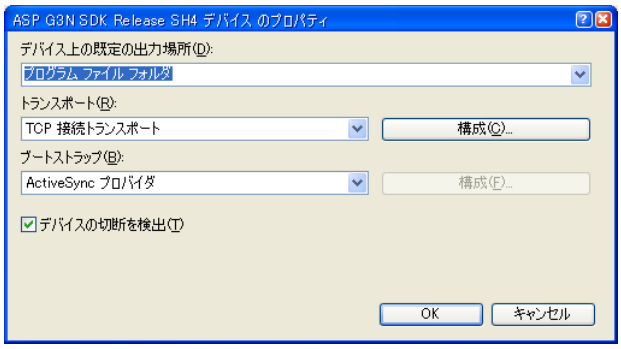

図 4-3-2-2. デバイスのプロパティ

④ [トランスポート]の[構成]を押します。[TCP/IP トランスポートの構成]ダイアログが開きます。(図 4-3-2-3)
[デバイス IP アドレス]で[特定の IP アドレス使用]を選択し、「4-3-1 IP アドレスの確認」で確認した IP アドレスを入力します。

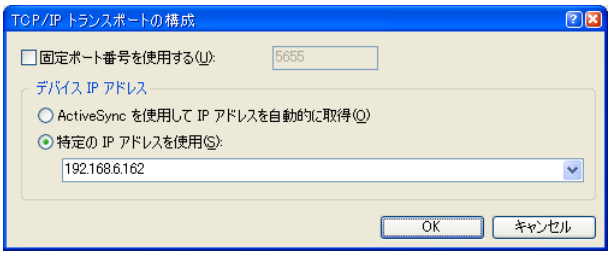

図 4-3-2-3. TCP/IP トランスポートの構成

⑤ [OK]ボタンを順に押して、すべてのダイアログを閉じます。

４－３－３　TCPトランスポートプログラムの起動

Algo Smart Panel 側で Visual Studio 2005 と通信するためのプログラムを起動します。

① 開発用 PC の「¥Program Files¥Common Files¥microsoft shared¥CoreCon¥1.0¥Target¥wce400¥sh4」フォルダ内の下記のファイルを Algo Smart Panel 側の「¥Windows」フォルダへコピーします。

- Clientshutdown.exe
- ConmanClient2.exe
- CMaccept.exe
- eDbgTL.dll
- TcpConnectionA.dll

② TCP トランスポートプログラムを起動します。[¥Windows]フォルダを開き、以下の 2 つのファイルを順にクリックして起動します。(バックグラウンドでの実行なので、クリックしても結果はありません)

- ConmanClient2.exe
- CMaccept.exe

③ これで接続のための準備は終了です。開発用 PC でデバッグ、リモートツールを使用します。ただし、接続は CMaccept.exe を起動してから 3 分間までに行う必要があります。3 分間までに接続できない場合は、再度 CMaccept.exe を起動して接続するようにします。

４－３－４　接続確認

Algo Smart Panel と Visual Studio 2005 が正しく接続できるかどうか確認します。

① 「4-3-3 TCP トランスポートプログラムの起動」の作業を行います。
② Visual Studio 2005 のメニューから[ツール]-[デバイスへの接続]を選択します。
③ [デバイスへの接続]ダイアログが開きます。（図 4-3-4-1）

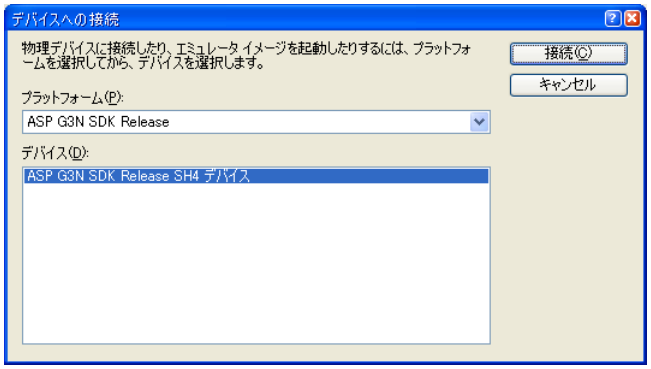

図 4-3-4-1. デバイスへの接続

④ [プラットフォーム]から[ASP G3N SDK Release]を選択し、[接続]ボタンを押します。
⑤ 接続に成功すると図 4-3-4-2 の様なダイアログが表示されます。

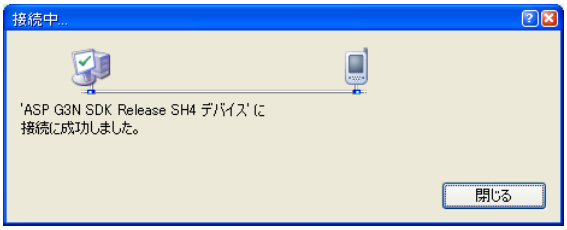

図 4-3-4-2. 接続成功

４－４　　ネイティブコードアプリケーション開発

本項では、ネイティブコードアプリケーションの開発方法を説明します。

４－４－１　　アプリケーションの作成

Visual Studio 2005 のアプリケーションウィザードを使用して、Algo Smart Panel 用のネイティブコードアプリケーションを作成します。ここでは、サンプルとして MFC を使用したダイアログベースのアプリケーションを作成します。

ここで作成するサンプルコードは、「Algo Smart Panel AP-4410/5410/6410/6500/7500 Windows Embedded CE 6.0 リカバリ/SDK DVD」で提供しています。（表 4-4-1-1）

表 4-4-1-1. ネイティブコードアプリケーションサンプル

DVD-ROM のフォルダ	内容
¥SDK¥Algo¥Sample¥Sample_MFC	ネイティブコードアプリケーションのサンプルコードです。

① Visual Studio 2005 を起動します。
② メニューから[ファイル]-[新規作成]-[プロジェクト]を選択し、[新しいプロジェクト]ダイアログを開きます。（図 4-4-1-1）
③ 左ペインの[プロジェクトの種類]で[Visual C++]-[スマートデバイス]を選択します。右ペインの[テンプレート]から[MFC スマートデバイスアプリケーション]を選択します。

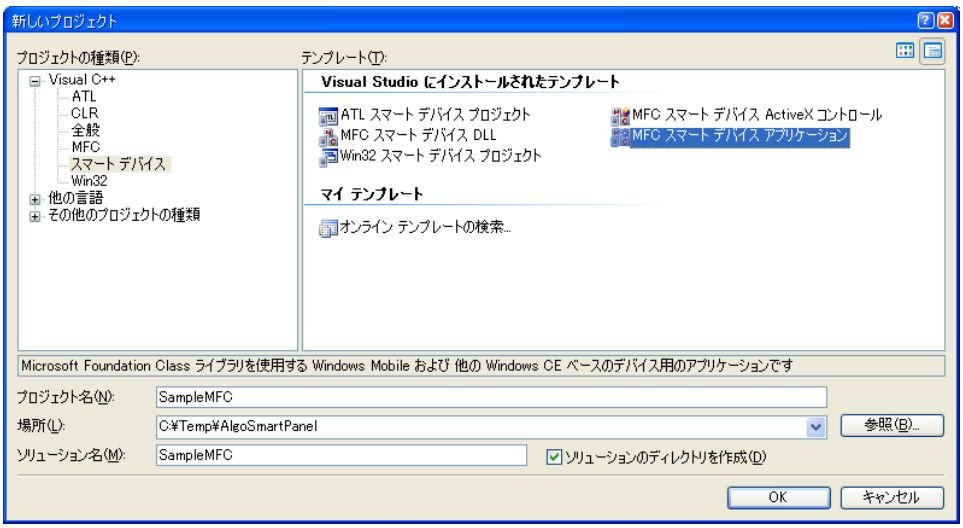

図 4-4-1-1. 新しいプロジェクト

④ [プロジェクト名]、[場所]、[ソリューション名]を指定し、[OK]ボタンを押します。ここでは、以下の様に設定しています。

[プロジェクト名]　：　SampleMFC
[場所]　：　C¥Temp¥AlgoSmartPanel
[ソリューション名]：　SampleMFC

⑤　アプリケーションウィザードが開きます。(図 4-4-1-2)
[次へ]を押します。

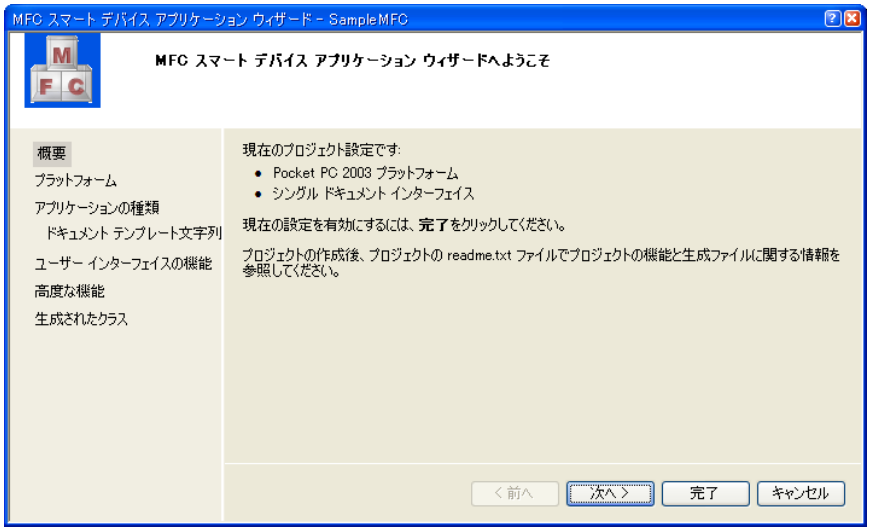

図 4-4-1-2. アプリケーションウィザード

⑥　[選択された SDK]に[ASP G3N SDK Release]を追加します。Algo Smart Panel でのみ動作するアプリケーションの場合は、[ASP G3N SDK Release]のみを選択します。(図 4-4-1-3)
[次へ]を押します。

図 4-4-1-3. SDKの選択

⑦ [アプリケーションの種類]で[ダイアログベース]を選択します。[MFC の使用方法]で[スタティック ライブラリで MFC を使用]を選択します。(図 4-4-1-4)

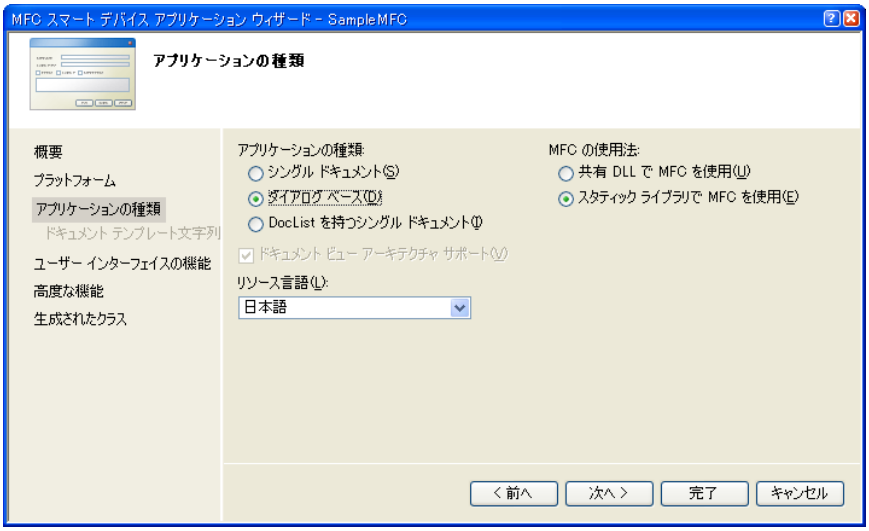

図 4-4-1-4. ダイアログベースアプリケーションの選択

⑧ [完了]を押してウィザードを終了するとプロジェクトが作成されます。
⑨ 通常の Windows アプリケーションと同様にアプリケーションの作成を行います。(図 4-4-1-5)

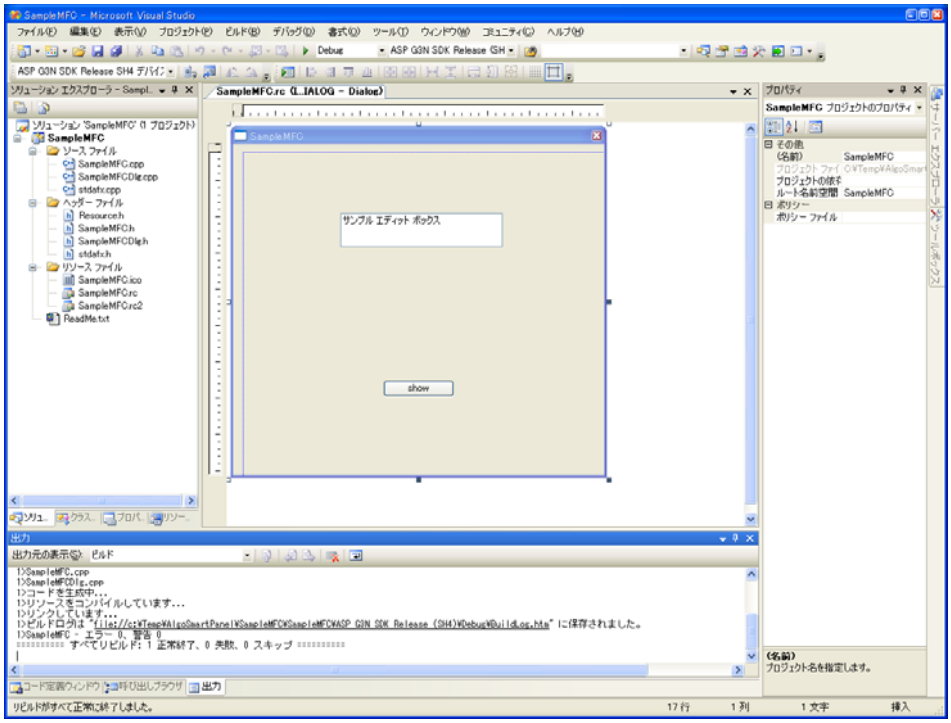

図 4-4-1-5. アプリケーションの作成

⑩　メニューから[ビルド]-[ソリューションのビルド]を選択しプロジェクトをビルドします。

※　ビルドの際、文字列関連の関数に関して、下記の様なワーニングメッセージが表示されます。このメッセージが表示されても作成されるアプリケーションには特に問題はありません。
気になる場合は、プロジェクトのプロパティから[C/C++プロパティ]-[プリプロセッサ]-[プリプロセッサの定義]に「 _CRT_SECURE_NO_DEPRECATE 」を追加してください。

```
1>C:¥Program Files¥Microsoft Visual Studio 8¥VC¥ce¥atlmfc¥include¥atlchecked.h(291) :
                    warning C4996: '_gcvt' が古い形式として宣言されました。
1>        C:¥Program Files¥Windows CE Tools¥wce600¥ASP G3N SDK Release¥include¥SH4¥stdlib.h(529) :
                    '_gcvt' の宣言を確認してください。
1>        メッセージ: 'This function or variable may be unsafe. Consider using _gcvt_s instead.
                    To disable deprecation, use _CRT_SECURE_NO_DEPRECATE. See online help for details.'
```

４－４－２　アプリケーションのデバッグ

「4-4-1 アプリケーションの作成」で作成したネイティブコードアプリケーションを使ってアプリケーションをデバッグします。デバッグを開始する前に「4-3 ターゲットとの接続」を参考に事前にターゲットとの接続が可能であることを確認してください。

① Visual isual Studio 2005 を起動し、「4-4-1 アプリケーションの作成」で作成したプロジェクトを開きます。
② メニューから[ビルド]-[構成マネージャ]を選択し、[アクティブ ソリューション構成]を[Debug]にします。
③ ソリューションをビルドします。ブレークポイントなどを設定し、デバッグの準備をします。（図 4-4-2-1）

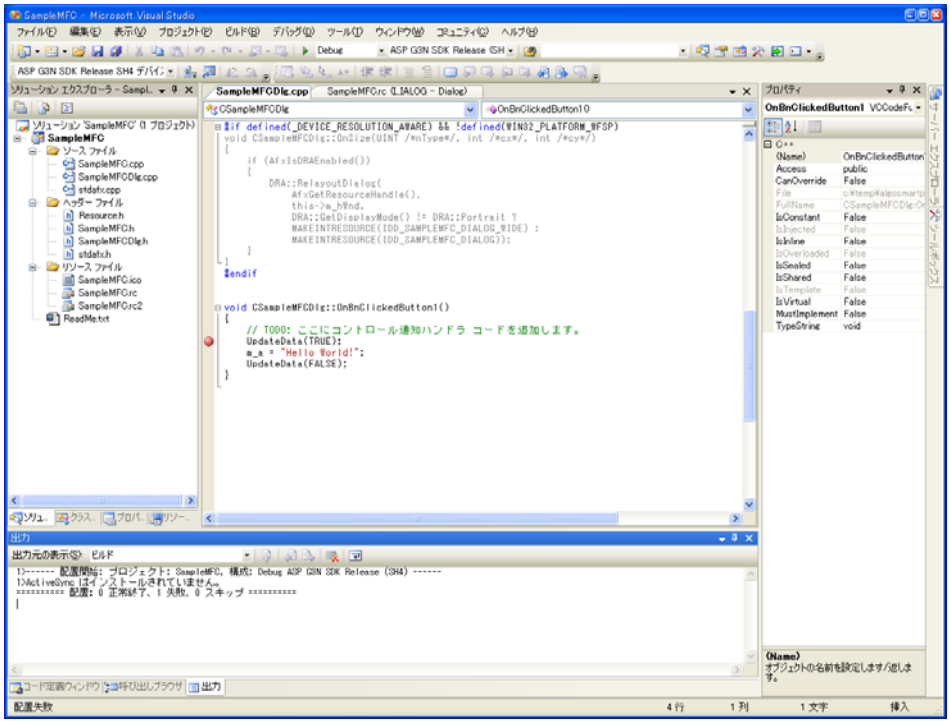

図 4-4-2-1. デバッグ準備

④　メニューから[デバッグ]-[デバッグの開始]を選択すると、Algo Smart Panel にプログラム転送され、プログラム起動がします。(図 4-4-2-2)
転送に失敗する場合は、「4-3 ターゲットとの接続」を参考に接続を確認してください。

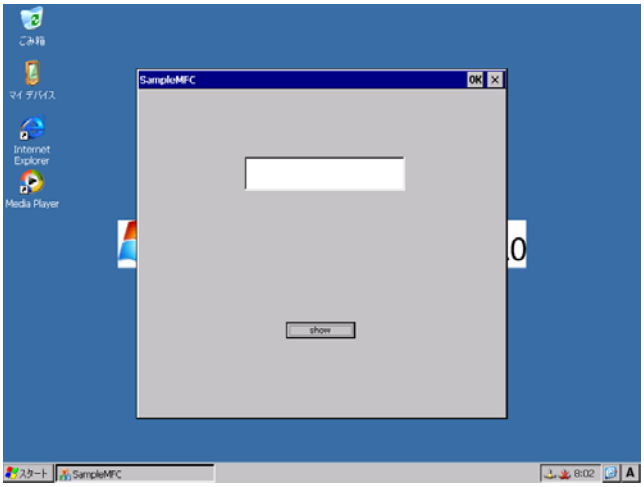

図 4-4-2-2. アプリケーションの起動

⑤　デバッグが開始されます。通常の Windows アプリケーションと同様にデバッグを行います。(図 4-4-2-3)

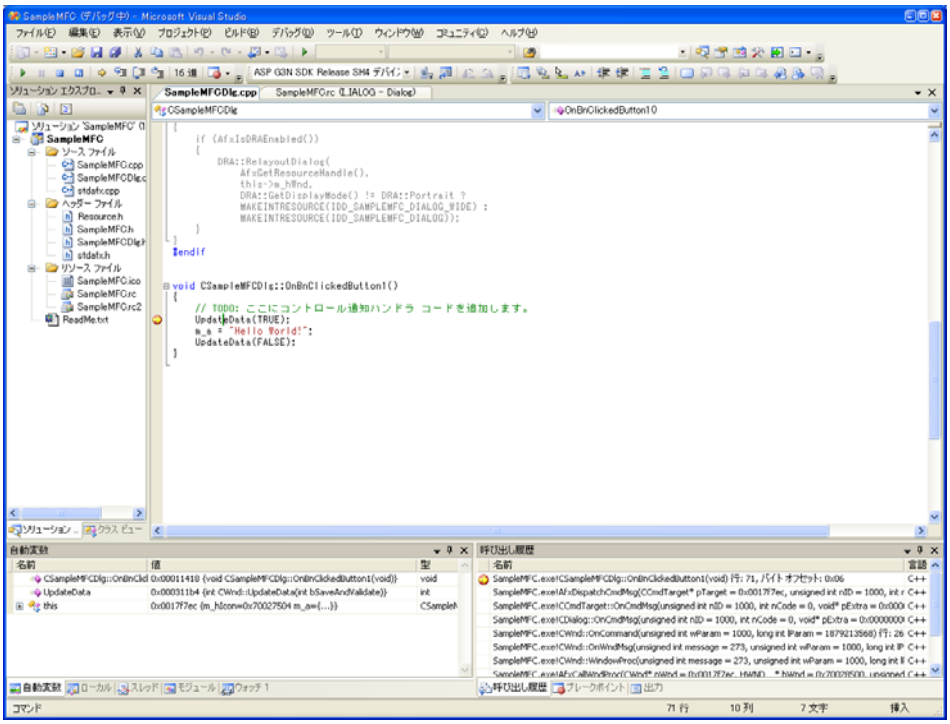

図 4-4-2-3. デバッグの開始

４－５　　マネージドコードアプリケーション開発

本項では、.NET Compact Framework を使用したマネージドコードアプリケーションの開発方法を説明します。

４－５－１　　アプリケーションの作成

Visual Studio 2005 のアプリケーションウィザードを使用して、Algo Smart Panel 用のマネージドコードアプリケーションを作成します。ここでは、サンプルとして C#を使用したダイアログベースのアプリケーションを作成します。

ここで作成するサンプルコードは、「Algo Smart Panel AP-4410/5410/6410/6500/7500 Windows Embedded CE 6.0 リカバリ/SDK DVD」で提供しています。（表 4-5-1-1）

表 4-5-1-1. ネイティブコードアプリケーションサンプル

DVD-ROM のフォルダ	内容
¥SDK¥Algo¥Sample¥Sample_C#	マネージドコードアプリケーションのサンプルコードです。

⑪ Visual Studio 2005 を起動します。

⑫ メニューから[ファイル]-[新規作成]-[プロジェクト]を選択し、[新しいプロジェクト]ダイアログを開きます。（図 4-5-1-1）

⑬ 左ペインの[プロジェクトの種類]で[Visual C#]-[SmartDevice]-[WindowsCE]を選択します。右ペインの[テンプレート]から[デバイスアプリケーション]を選択します。

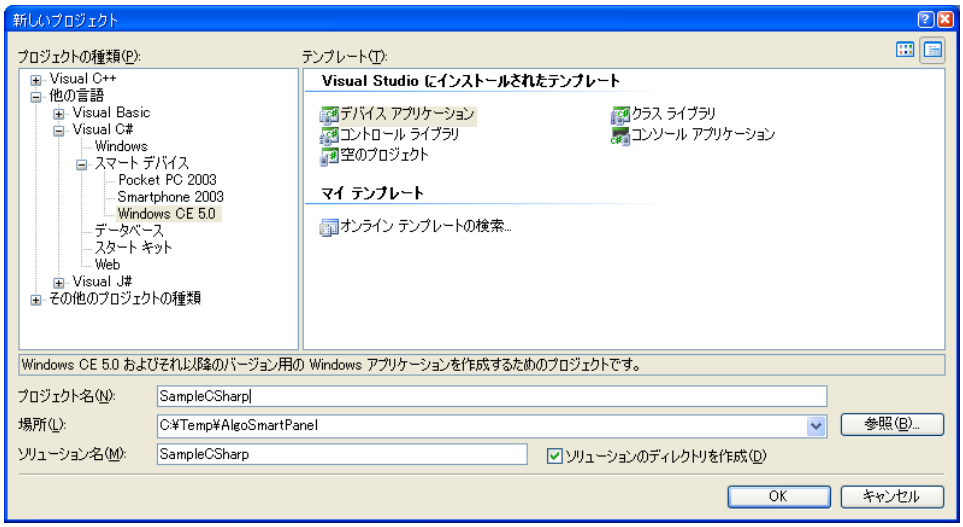

図 4-5-1-1. 新しいプロジェクト

⑭ [プロジェクト名]、[場所]、[ソリューション名]を指定します。ここでは、以下の様に設定しています。

[プロジェクト名]　：　SampleCSharp
[場所]　：　C¥Temp¥AlgoSmartPanel
[ソリューション名]：　SampleCSharp

⑮ [OK]ボタンを押すとプロジェクトが作成されます。
⑯ 通常の Windows アプリケーションと同様にアプリケーションの作成を行います。(図 4-5-1-2)

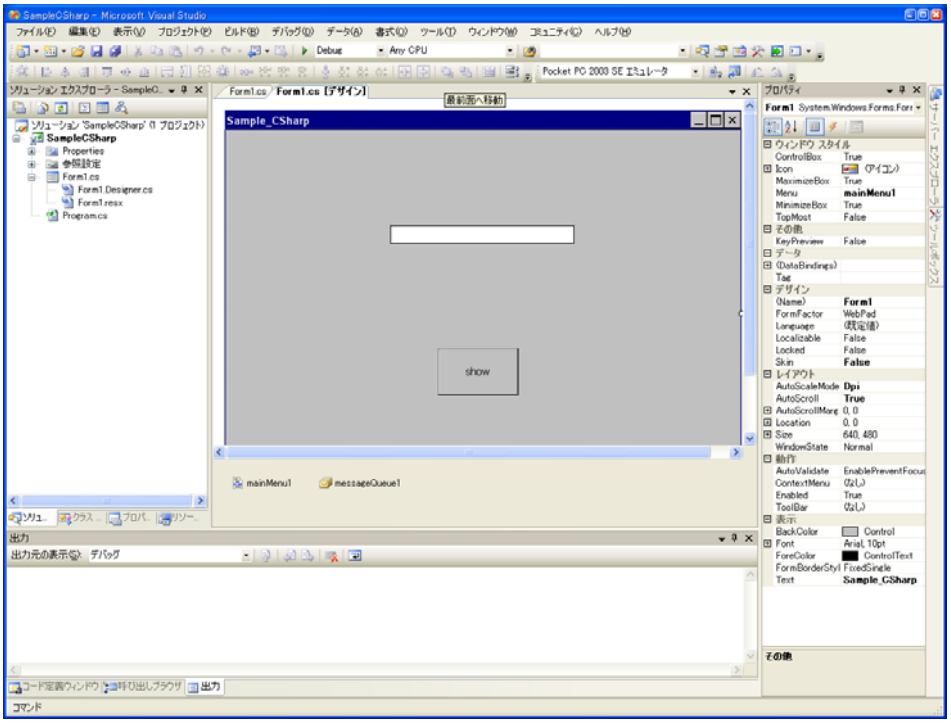

図 4-5-1-2. アプリケーションの作成

４－５－２　アプリケーションのデバッグ

「4-5-1 アプリケーションの作成」で作成したマネージドコードアプリケーションを使ってアプリケーションをデバッグします。デバッグを開始する前に「4-3 ターゲットとの接続」を参考に事前にターゲットとの接続が可能であることを確認してください。

① Visual Studio 2005 を起動し、「4-5-1 アプリケーションの作成」で作成したプロジェクトを開きます。
② メニューから[ビルド]-[構成マネージャ]を選択し、[アクティブ ソリューション構成]を[Debug]にします。
③ メニューから[プロジェクト]-[SampleCSharp のプロパティ]を選び、プロパティを開きます。[デバイス]タブを開きます。[ターゲットデバイス]で[ASP G3N SDK Release SH4 デバイス]を選択します。（図 4-5-2-1）

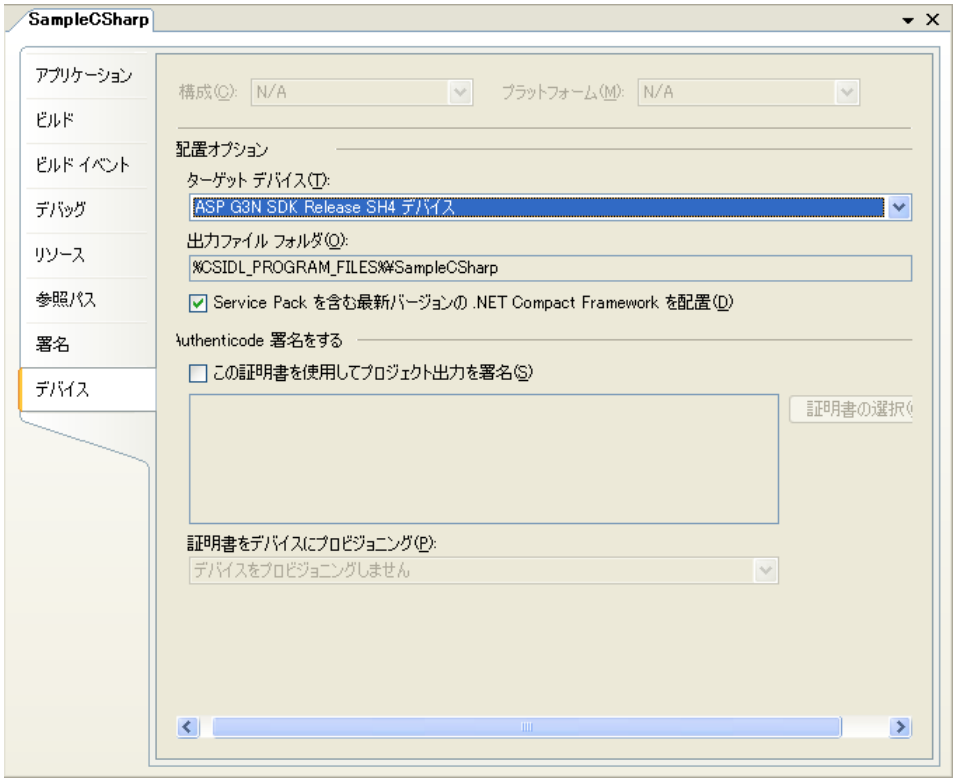

図 4-5-2-1. ターゲットデバイスの設定

④ ソリューションをビルドします。ブレークポイントなどを設定し、デバッグの準備をします。（図 4-5-2-2）

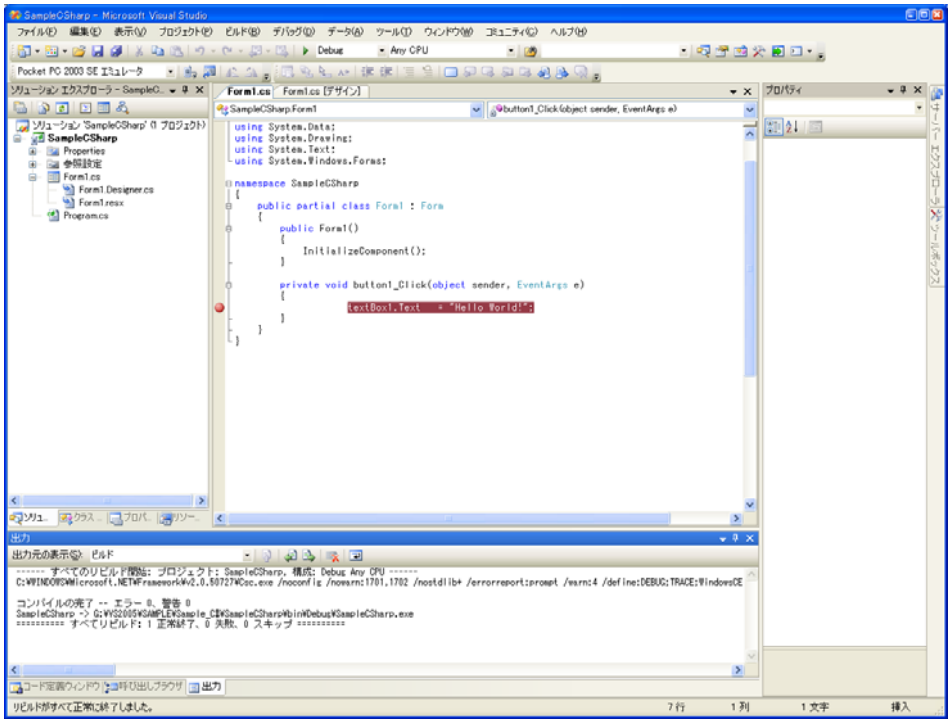

図 4-5-2-2. デバッグの準備

⑤ メニューから[デバッグ]-[デバッグの開始]を選択します。転送先デバイスを指定するダイアログが表示されますので、[ASP G3N SDK Release SH4 デバイス]を選択し[配置]ボタンを押します。（図 4-5-2-3）
接続に失敗する場合は、「4-3 ターゲットとの接続」を参考に接続を確認してください。

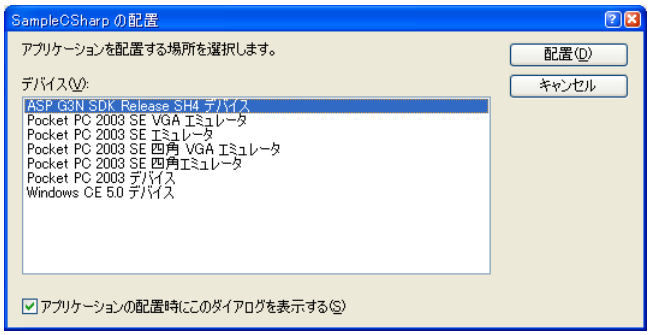

図 4-5-2-3. 転送先デバイスの指定

⑥ Algo Smart Panel にプログラム転送され、プログラム起動がします。（図 4-5-2-4）

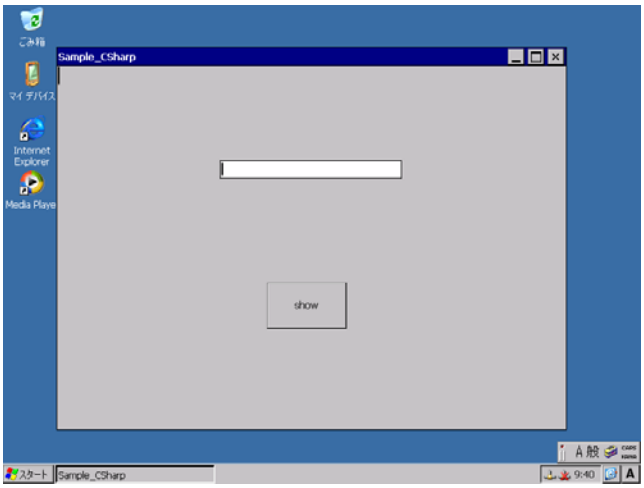

図 4-5-2-4. アプリケーションの起動

⑦ デバッグが開始されます。通常の Windows アプリケーションと同様にデバッグを行います。（図 4-5-2-5）

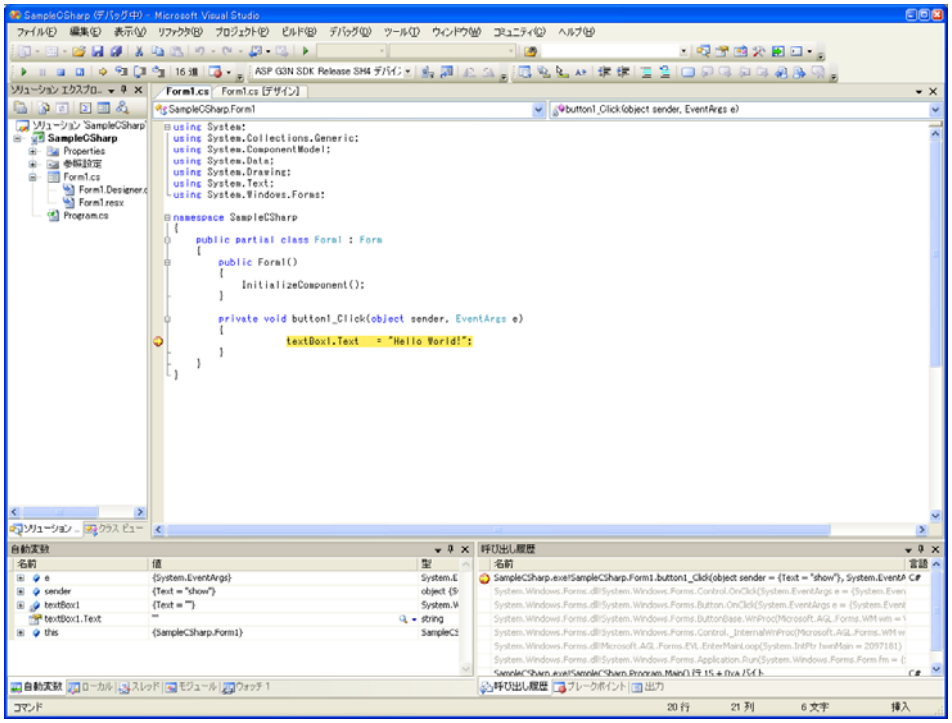

図 4-5-2-5. デバッグの開始

４－６　　リモートツール

Windows Embedded CE 6.0 は、Visual Studio 2005 に付属のリモートツールを使用してターゲットと接続し、ターゲットの状態確認、データの転送を行うことができます。
本項では、リモートツールの基本的な使用方法を説明します。

４－６－１　　リモートツールの起動と接続

リモートツールは、[スタートメニュー]-[すべてのプログラム]-[Microsoft Visual Studio 2005]-[Visual Studio Remote Tools]に用意されています。リモートツールは以下の 6 種類が用意されています。

- ■　リモート ズームイン
- ■　リモート スパイ
- ■　リモート ヒープ ウォーカ
- ■　リモート ファイル ビューア
- ■　リモート プロセス ビューア
- ■　リモート レジストリ エディタ

リモートツールを使用するには、「4-3 ターゲットとの接続」で説明している方法で Algo Smart Panel と接続できる常態である必要があります。
リモートツールを起動すると、図 4-6-1-1 のような接続先デバイスの選択ダイアログが表示されます。[ASP G3N SDK Release SH4 デバイス]を選択し[OK]ボタンを押します。正常に接続されるとリモートツールの各機能が実行されます。

図 4-6-1-1. CEデバイスの選択

４－６－２　リモート ズームイン

「リモート ズームイン」を使用すると、ターゲット上の画面イメージをキャプチャすることができます。キャプチャイメージの保存、印刷などを行うことができます。（図 4-6-2-1）

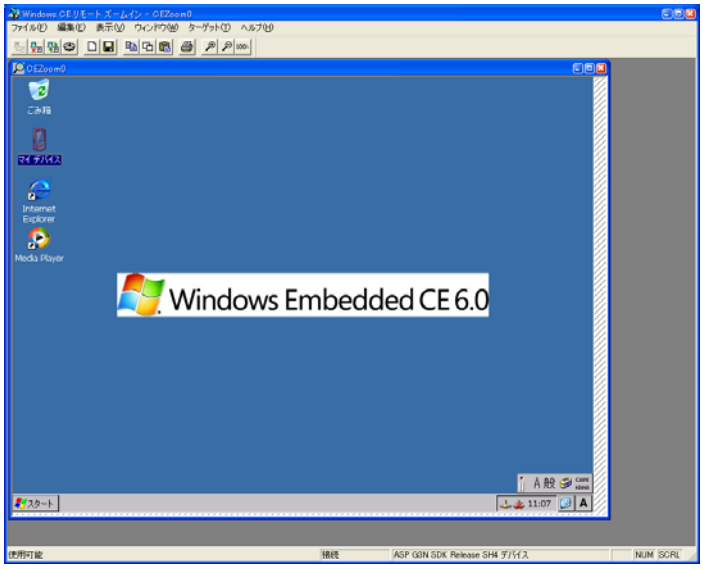

図 4-6-2-1. リモート ズームイン

４－６－３　リモート スパイ

「リモート スパイ」を使用すると、ターゲット上で実行中のアプリケーション・ウィンドウが受信したメッセージを表示することができます。（図 4-6-3-1）

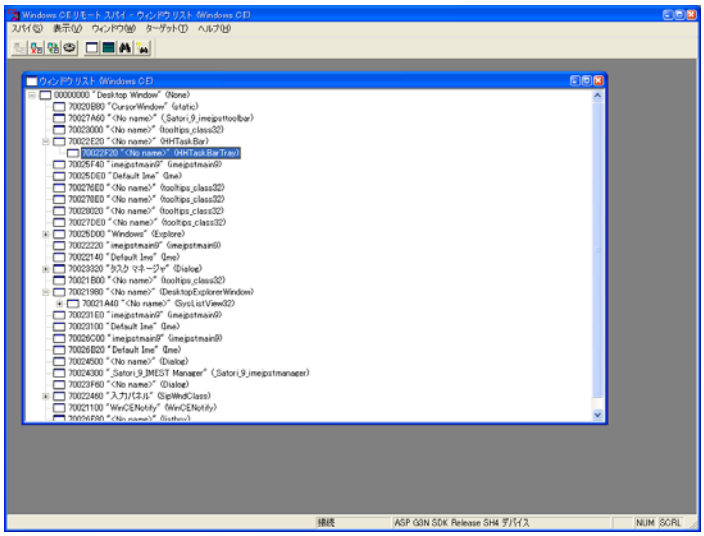

図 4-6-3-1. リモート スパイ

４－６－４　リモート ヒープ ウォーカ

「リモート ヒープ ウォーカ」を使用すると、ターゲット上で実行中の各プロセスのヒープ ID およびフラグに関する情報が表示することができます。（図 4-6-4-1）

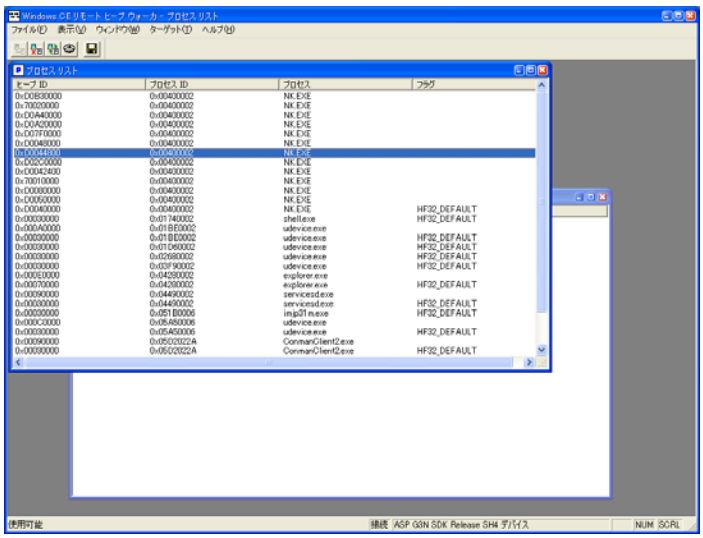

図 4-6-4-1. リモート ヒープ ウォーカ

４－６－５　リモート ファイル ビューア

「リモート ファイル ビューア」は、Windows エクスプローラと似ています。ターゲットのディレクトリの階層構造を表示することができます。ファイルの削除、ファイルのプロパティの表示、ファイルの検索が可能です。（図 4-6-5-1）

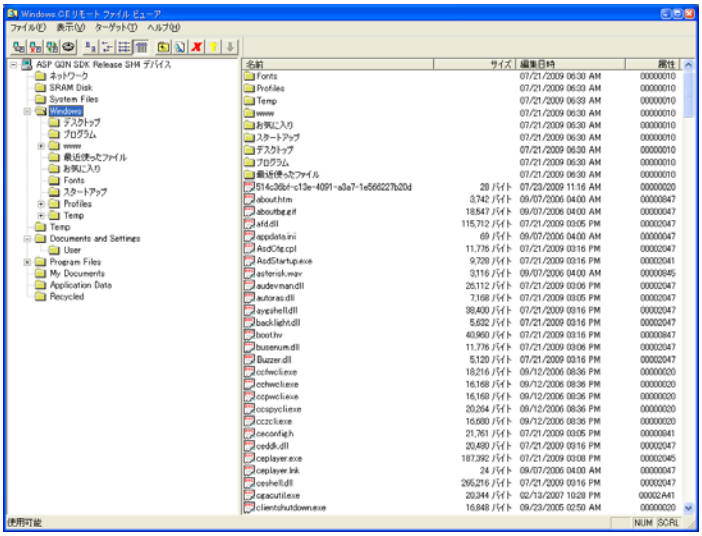

図 4-6-5-1. リモート ファイル ビューア

４－６－６　リモート プロセス ビューア

「リモート プロセス ビューア」を使用すると、ターゲット上で実行中の各プロセスに関連付けられた情報を表示することができます。（図 4-6-6-1）

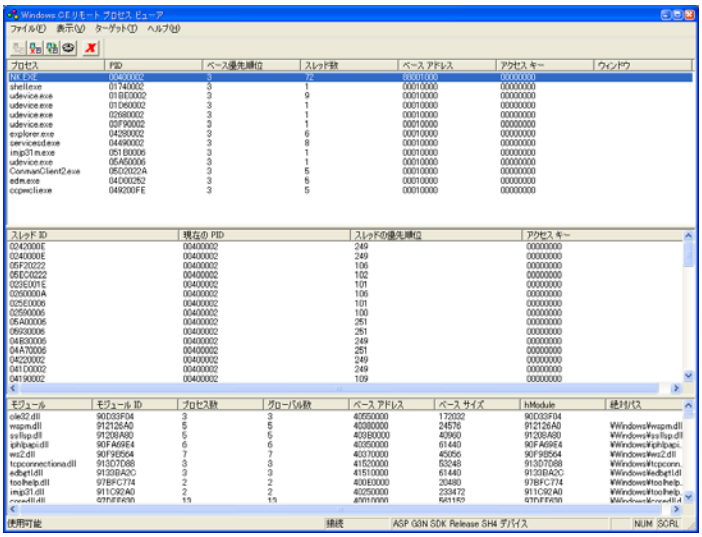

図 4-6-6-1. リモート プロセス ビューア

４－６－７　リモート レジストリ エディタ

「リモート レジストリ エディタ」を使用すると、ターゲットのレジストリの閲覧とレジストリキーの追加、削除、編集ができます。（図 4-6-7-1）

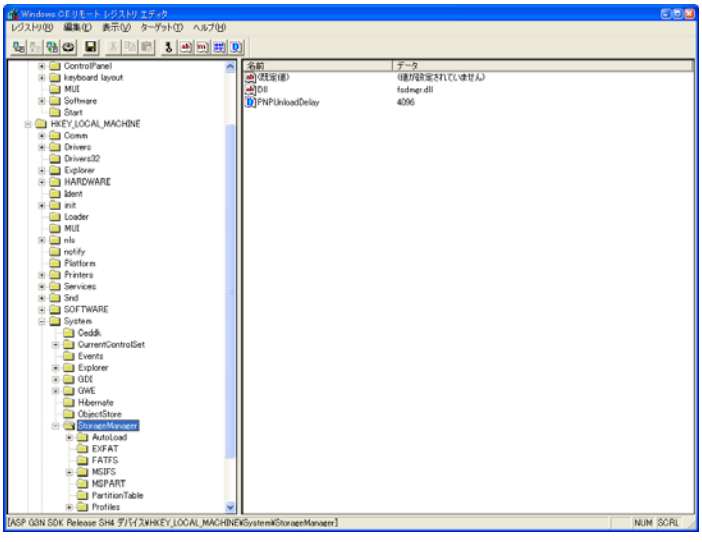

図 4-6-7-1. リモート レジストリ エディタ

第５章　組込みシステム機能ドライバ

Algo Smart Panel には、組込みシステム向けに独自の機能が搭載されています。Algo Smart Panel 用 Windows CE 6.0 には、これらの機能にアクセスするためのドライバが用意されています。このドライバを使用することでアプリケーションからこれらの機能を使用することができます。本章では、組込みシステム機能ドライバの使用方法について説明します。

５－１　　ドライバの使用について

５－１－１　サンプルコードについて

「Algo Smart Panel AP-4410/5410/6410/6500/7500 Windows Embedded CE 6.0 リカバリ/SDK DVD」にドライバを使用したサンプルコードを用意しています。DVD に含まれるサンプルコードの一覧を表 5-1-1-1 に示します。

表 5-1-1-1. サンプルコード一覧

DVD-ROM のフォルダ	内容
¥SDK¥Algo¥Sample¥Sample_BackLight	LCD バックライト制御のサンプルコードです。
¥SDK¥Algo¥Sample¥Sample_Buzzer	ブザー制御のサンプルコードです。
¥SDK¥Algo¥Sample¥Sample_GenIO	汎用入出力制御のサンプルコードです。
¥SDK¥Algo¥Sample¥Sample_Interrupt	タイマ割込み機能 IN1 割込み機能サンプルコードです。
¥SDK¥Algo¥Sample¥Sample_Reset	IN0 リセット機能のサンプルコードです。
¥SDK¥Algo¥Sample¥Sample_SerialControl	シリアルコントロール機能のサンプルコードです。
¥SDK¥Algo¥Sample¥Sample_C#	C#アプリケーションのサンプルコードです。
¥SDK¥Algo¥Sample¥Sample_MFC	MFC アプリケーションのサンプルコードです。

５－１－２　DeviceIoControlについて

組込みシステム機能のドライバは、ドライバの機能にアクセスするためにDeviceIoControl関数を使用します。以下にその書式を示します。関数仕様の詳細は、Windows APIの仕様を参照してください。

コントロールコードに対応する動作及び引数は、ドライバ毎にリファレンスを用意していますので、各ドキュメントを参照してください。

関数書式

```
BOOL DeviceIoControl (
      HANDLE        hDevice,
      DWORD         dwIoControlCode,
      LPVOID        lpInBuf,
      DWORD         nInBufSize,
      LPVOID        lpOutBuf,
      DWORD         nOutBufSize,
      LPDWORD       lpBytesReturned,
      LPOVERLAPPED  lpOverlapped
);
```

パラメータ

hDevice	：デバイス、ファイル、ディレクトリいずれかのハンドル
dwIoControlCode	：実行する動作のコントロールコード
lpInBuf	：入力データを供給するバッファへのポインタ
nInBufSize	：入力バッファのバイト単位のサイズ
lpOutBuf	：出力データを受け取るバッファへのポインタ
nOutBufSize	：出力バッファのバイト単位のサイズ
lpBytesReturned	：lpOutBufに格納されるバイト数を受け取る変数へのポインタ
lpOverlapped	：非同期動作を表す構造体へのポインタ

５－２　　バックライト

５－２－１　　バックライトについて

バックライトの ON/OFF、輝度を変更することができます。

５－２－２　　バックライトドライバ

バックライトドライバはバックライトの ON/OFF、輝度をユーザーアプリケーションから変更できるようにします。

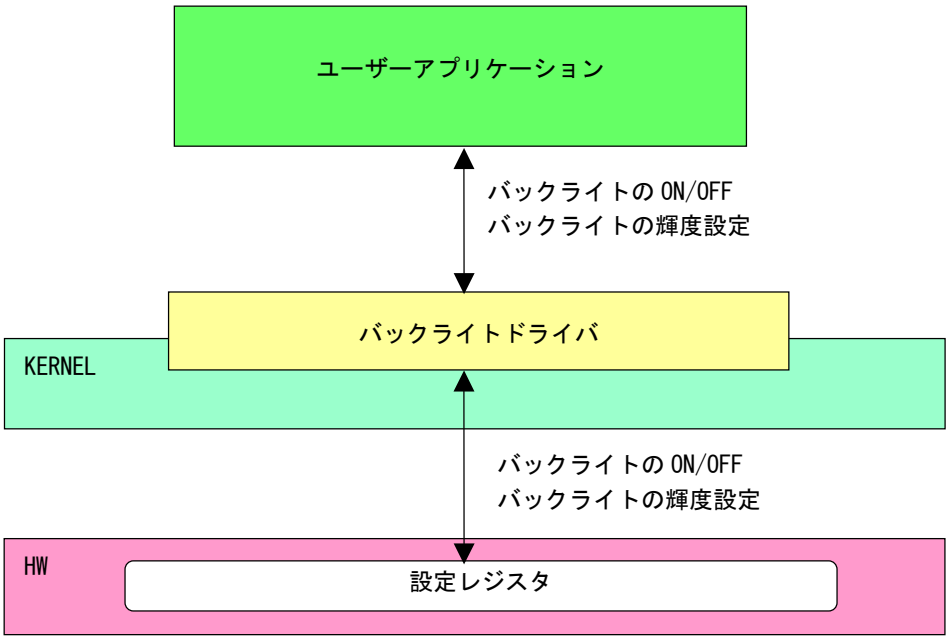

図 5-2-2-1. バックライトドライバ

５－２－３　バックライトデバイス

バックライトドライバはバックライトデバイスを生成します。ユーザーアプリケーションは、デバイスファイルにアクセスすることによってバックライトの ON/OFF、輝度を操作します。

バックライトデバイス

デバイスファイル

BKL1:

説明

バックライトのON/OFF、輝度を変更することができます。

レジストリ設定

[KEY]　　HKEY_LOCAL_MACHINE¥Drivers¥BuiltIn¥Backlight
[VALUE:DWORD]　　Brightness

バックライトの輝度を設定します。ドライバ起動時(OS起動時)にこの値を参照します。
(デフォルト値: 0)

CreateFile

デバイスファイル(BKL1:)をオープンし、デバイスハンドルを取得します。

```
hBacklight = CreateFile(
    _T("BKL1:"),
    GENERIC_READ | GENERIC_WRITE,
    FILE_SHARE_READ | FILE_SHARE_WRITE,
    NULL,
    OPEN_EXISTING,
    0,
    NULL
);
```

CloseHandle

デバイスハンドルをクローズします。

```
CloseHandle(hBacklight);
```

DeviceIoControl

- **IOCTL_BKL_SETSTATE**
  バックライトのON/OFFを設定します。
- **IOCTL_BKL_GETSTATE**
  バックライトのON/OFFを取得します。
- **IOCTL_BKL_SETBRIGHTNESS**
  バックライトの輝度を設定します。
- **IOCTL_BKL_GETBRIGHTNESS**
  バックライトの輝度を取得します。
- **IOCTL_BKL_GETMAXBRIGHTNESS**
  バックライトの輝度の最大値を取得します。
- **IOCTL_BKL_SAVEREG**
  バックライトの輝度設定をレジストリに書込みます。
- **IOCTL_BKL_LOADREG**
  バックライトの輝度設定をレジストリから読込みます。

５－２－４　リファレンス

IOCTL_BKL_SETSTATE　[DeviceIoControl]

機能

バックライトの ON/OFF を設定します。

パラメータ

lpInBuf	:	バックライト ON/OFF 情報を格納するポインタ
nInBufSize	:	バックライト ON/OFF 情報のデータサイズ
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

バックライト ON/OFF 情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	0: OFF, 1: ON

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

バックライトの ON/OFF の設定を行います。
バックライトを ON する場合は、バックライト ON/OFF 情報に 1、OFF にする場合は 0 を設定し DeviceIoControl を実行します。

IOCTL_BKL_GETSTATE　[DeviceIoControl]

機能

バックライトの ON/OFF を取得します。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	バックライト ON/OFF 情報を格納するポインタ
nOutBufSize	:	バックライト ON/OFF 情報のデータサイズ
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

バックライト ON/OFF 情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	0: OFF, 1: ON

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

バックライトの ON/OFF を取得します。処理が成功するとバックライト ON/OFF 情報にバックライト ON/OFF の状態が格納されます。バックライト ON/OFF 情報が 1 で ON、0 で OFF となります。

IOCTL_BKL_SETBRIGHTNESS　[DeviceIoControl]

機能

バックライトの輝度を設定します。

パラメータ

lpInBuf	:	バックライト輝度情報を格納するポインタ
nInBufSize	:	バックライト輝度情報のデータサイズ
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

バックライト輝度情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	0（暗い）～15（明るい）

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

バックライトの輝度の設定を行います。
バックライトの輝度は 0（暗い）～15（明るい）の 16 段階で設定できます。バックライト輝度情報を設定して DeviceIoControl を実行してください。

IOCTL_BKL_GETBRIGHTNESS　[DeviceIoControl]

機能

バックライトの輝度を取得します。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	バックライト輝度情報を格納するポインタ
nOutBufSize	:	バックライト輝度情報のデータサイズ
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

バックライト輝度情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	0（暗い）～15（明るい）

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

バックライトの輝度の取得を行います。処理が成功するとバックライト輝度情報にバックライト輝度が格納されます。

IOCTL_BKL_GETMAXBRIGHTNESS　[DeviceIoControl]

機能

バックライトの輝度設定の最大値を取得します。

パラメータ

lpInBuf	:	バックライト輝度情報を格納するポインタ
nInBufSize	:	バックライト輝度情報のデータサイズ
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

バックライト輝度情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	輝度最大値（15）

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

バックライトの輝度設定の最大値を取得します。処理が成功するとバックライト輝度情報にバックライト輝度の最大値が格納されます。本シリーズでは 15 が格納されます。

IOCTL_BKL_SAVEREG　[DeviceIoControl]

機能

バックライトの輝度設定をレジストリに書込みます。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

現在のバックライトの輝度設定をレジストリに書込みます。
レジストリ情報は OS 起動時、IOCTL_BLK__LOADREG（DeviceIoControl）実行時に反映されます。

IOCTL_BKL_LOADREG　[DeviceIoControl]

機能

バックライトの輝度設定をレジストリから読込みます。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

レジストリからバックライトの輝度設定を読込み、バックライトの状態を変更します。

５－２－５　サンプルコード

●バックライト輝度

「¥SDK¥Algo¥Sample¥Sample_BackLight¥BacklightBrightness」にバックライト輝度の取得と設定のサンプルコードを用意しています。リスト 5-2-5-1 にサンプルコードを示します。

リスト 5-2-5-1. LCDバックライト輝度

```
/**
    バックライト輝度制御サンプルソース
**/

#include "stdafx.h"
#include <windows.h>
#include <commctrl.h>

#include <asp_g3n¥backlight.h>

#define DRIVER_FILENAME     _T("BKL1:")

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE  m_hDevice;                      // デバイスハンドル
    BOOL    ret;
    DWORD   retlen;
    DWORD   set_data;
    DWORD   get_data;

    /*
     * 起動引数からバックライト光量変更値を取得
     *   0～15 の範囲で設定します
     *   0：暗い　～　15：明るい
     */
    if(argc != 2){
        _tprintf(_T("invalid arg¥n"));
        _tprintf(_T("BacklightBrightness.exe [Brightness 0~15]¥n"));
        return -1;
    }
    _stscanf(*(argv + 1), _T("%u"), &set_data);

    /*
     *デバイスの Open
     */
    m_hDevice = CreateFile(
                    DRIVER_FILENAME,
                    GENERIC_READ | GENERIC_WRITE,
                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL
                );
    if(m_hDevice == INVALID_HANDLE_VALUE){
```

```
        _tprintf(_T("CreateFile: NG¥n"));
        return FALSE;
    }

    /*
     *バックライトの輝度を設定
     */
    ret = DeviceIoControl(
                m_hDevice,
                IOCTL_BKL_SETBRIGHTNESS,
                &set_data,
                sizeof(DWORD),
                NULL,
                0,
                &retlen,
                NULL);
    if(!ret){
        _tprintf(_T("DeviceIoControl: IOCTL_BKL_SETBRIGHTNESS NG¥n"));
        CloseHandle(m_hDevice);
        return -1;
    }

    /*
     *バックライトの輝度を取得
     */
    ret = DeviceIoControl(
                m_hDevice,
                IOCTL_BKL_GETBRIGHTNESS,
                NULL,
                0,
                &get_data,
                sizeof(DWORD),
                &retlen,
                NULL);
    if(!ret){
        _tprintf(_T("DeviceIoControl: IOCTL_BKL_GETBRIGHTNESS NG¥n"));
        CloseHandle(m_hDevice);
        return -1;
    }
    _tprintf(_T("Get LCD Backlight Brightness: %x¥n"), get_data);

    /*
     *デバイスの Close
     */
    CloseHandle(m_hDevice);

    return 0;
}
```

●バックライト ON/OFF

「¥SDK¥Algo¥Sample¥Sample_BackLight¥BacklightOnOff」にバックライト ON/OFF 制御のサンプルコードを用意しています。リスト 5-2-5-2 にサンプルコードを示します。

リスト 5-2-5-2. バックライトON/OFF

```
/**
    バックライト OnOff 制御サンプルソース
**/

#include "stdafx.h"
#include <windows.h>
#include <commctrl.h>

#include <asp_g3n¥backlight.h>

#define DRIVER_FILENAME     _T("BKL1:")

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE  m_hDevice;                      // デバイスハンドル
    BOOL    ret;
    DWORD   retlen;
    DWORD   set_data;
    DWORD   get_data;

    /*
     * 起動引数からバックライトの ON/OFF 変更値を取得します。
     *   1 : バックライト ON
     *   0 : バックライト OFF
     */
    if(argc != 2){
        _tprintf(_T("invalid arg¥n"));
        _tprintf(_T("BacklightOnOff.exe [OnOff(0:OFF, 1:ON)]¥n"));
        return -1;
    }
    _stscanf(*(argv + 1), _T("%u"), &set_data);

    /*
     *デバイスの Open
     */
    m_hDevice = CreateFile(
                    DRIVER_FILENAME,
                    GENERIC_READ | GENERIC_WRITE,
                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL
                );
    if(m_hDevice == INVALID_HANDLE_VALUE){
        _tprintf(_T("CreateFile: NG¥n"));
        return FALSE;
```

```
}

/*
 *バックライトの ON/OFF を設定
 *       バックライト  1: ON
 *                     0: OFF
 */
ret = DeviceIoControl(
            m_hDevice,
            IOCTL_BKL_SETSTATE,
            &set_data,
            sizeof(DWORD),
            NULL,
            0,
            &retlen,
            NULL);
if(!ret){
    _tprintf(_T("DeviceIoControl: IOCTL_BKL_SETSTATE NG¥n"));
    CloseHandle(m_hDevice);
    return -1;
}

/*
 *バックライトの ON/OFF を取得
 */
ret = DeviceIoControl(
            m_hDevice,
            IOCTL_BKL_GETSTATE,
            NULL,
            0,
            &get_data,
            sizeof(DWORD),
            &retlen,
            NULL);
if(!ret){
    _tprintf(_T("DeviceIoControl: IOCTL_BKL_GETSTATE NG¥n"));
    CloseHandle(m_hDevice);
    return -1;
}

if(get_data == 1){
    _tprintf(_T("status=ON¥n"));
}
else if(get_data == 0){
    _tprintf(_T("status=OFF¥n"));
}

/*
 *デバイスの Close
 */
CloseHandle(m_hDevice);
```

```
    return 0;
}
```

５－３　　ブザー

５－３－１　　ブザーについて

ブザーの ON/OFF、ブザーの周波数の変更(16 段階)、タッチパネルのタッチ音の ON/OFF を行うことができます。

５－３－２　　ブザードライバ

ブザードライバは、ユーザーアプリケーションからブザーを操作出来るようにします。ユーザーアプリケーションから、ブザーの ON/OFF 設定、ブザー周波数の設定、タッチパネルのタッチ音の ON/OFF を変更することができます。

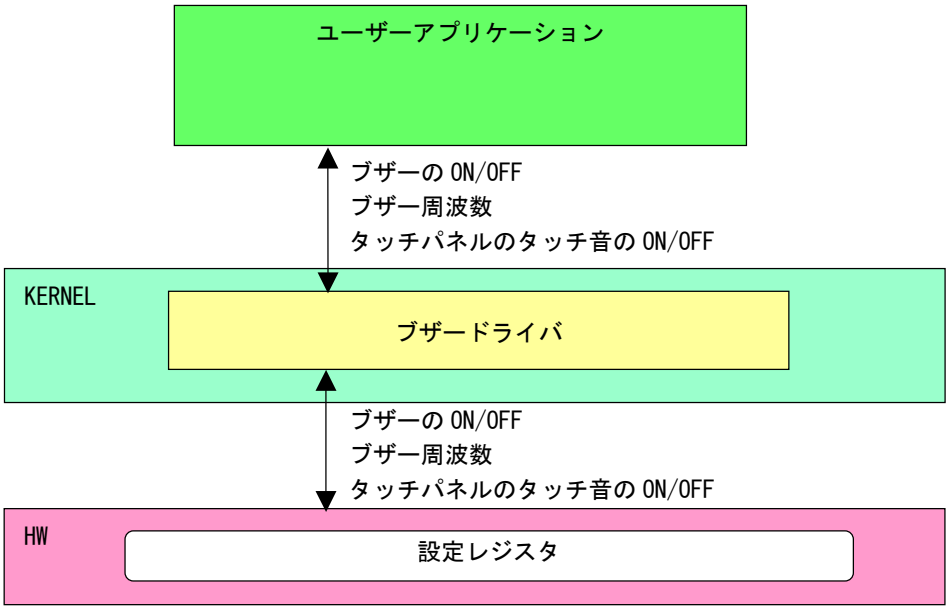

図 5-3-2-1. ブザードライバ

５－３－３　ブザーデバイス

ブザードライバはブザーデバイスを生成します。ユーザーアプリケーションは、デバイスファイルにアクセスすることによってブザー機能を操作します。

ブザーデバイス

デバイスファイル

BUZ1:

説明

ブザーのON/OFF、ブザーの周波数の設定、タッチパネルのタッチ音のON/OFFを
行うことができます。
ブザーデバイスはシステム全体で同時に利用できる数は15までとなっています。

レジストリ設定

[KEY]　　HKEY_LOCAL_MACHINE¥Drivers¥BuiltIn¥Buzzer
[VALUE:DWORD]　Hz　　0～15
　ブザー周波数を設定します。ドライバ起動時(OS起動時)にこの値を参照しブザー周波数を
　設定します。(デフォルト値: 9)
[VALUE:DWORD]　TouchBuzzer　　0:OFF, 1:ON
　タッチパネルのタッチ音のON/OFFを設定します。ドライバ起動時(OS起動時)に
　この値を参照し、タッチ音を設定します。(デフォルト値: 1)

CreateFile

デバイスファイル(BUZ1:)をオープンし、デバイスハンドルを取得します。
ブザーデバイスはシステム全体で同時に利用できる数は15までとなっています。
15以上のハンドルを取得しようとするとCreateFileはエラーとなります。

```
hBuzz = CreateFile(
    _T("BUZ1:"),
    GENERIC_READ | GENERIC_WRITE,
    FILE_SHARE_READ | FILE_SHARE_WRITE,
    NULL,
    OPEN_EXISTING,
    0,
    NULL
);
```

CloseHandle

デバイスハンドルをクローズします。

```
CloseHandle(hBuzz);
```

DeviceIoControl

- **IOCTL_BUZ_SETTOUCH**
  タッチパネルのタッチ音のを設定します。
- **IOCTL_BUZ_GETTOUCH**
  タッチパネルのタッチ音を取得します。
- **IOCTL_BUZ_SETHZ**
  ブザー周波数を設定します。
- **IOCTL_BUZ_GETHZ**
  ブザー周波数を取得します。
- **IOCTL_BUZ_SAVEREGHZ**
  現在のタッチ音設定、ブザー周波数設定をレジストリに書込みます。
- **IOCTL_BUZ_LOADREGHZ**
  タッチ音設定、ブザー周波数設定をレジストリから読込みます。
- **IOCTL_BUZ_ON**
  指定した時間ブザーをONします。
- **IOCTL_BUZ_OFF**
  ブザーをOFFします。
- **IOCTL_BUZ_STATUS**
  ブザーの状態を取得します。

５－３－４　リファレンス

IOCTL_BUZ_SETTOUCH　[DeviceIoControl]

機能

タッチパネルのタッチ音の ON/OFF の設定を行います。

パラメータ

lpInBuf	:	タッチ音情報を格納するポインタ
nInBufSize	:	タッチ音情報のデータサイズ
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

タッチ音情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	0: OFF, 1: ON

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

タッチパネルのタッチ音の ON/OFF の設定を行います。
タッチ音を ON にする場合は、タッチ音情報に 1、OFF にする場合は 0 を設定してから DeviceIoControl を実行してください。

IOCTL_BUZ_GETTOUCH　[DeviceIoControl]

機能

タッチパネルのタッチ音の ON/OFF の取得を行います。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	タッチ音情報を格納するポインタ
nOutBufSize	:	タッチ音情報のデータサイズ
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

タッチパネル音情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	0: OFF, 1: ON

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

タッチパネルのタッチ音の ON/OFF の取得を行います。処理が成功するとタッチ音情報にタッチ音の状態が格納されます。

IOCTL_BUZ_SETHZ　[DeviceIoControl]

機能

ブザー周波数の設定を行います。

パラメータ

lpInBuf	:	ブザー周波数情報を格納するポインタ
nInBufSize	:	ブザー周波数情報のデータサイズ
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

ブザー周波数情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	0 ～ 15 [(設定値+1)kHz]

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

ブザー周波数の設定を行います。
ブザー周波数情報に 0～15 の値を入力して DeviceIoControl を実行してください。
15 以上の値を設定した場合は 15 に設定されます。

IOCTL_BUZ_GETHZ　[DeviceIoControl]

機能

ブザー周波数の取得を行います。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	ブザー周波数情報を格納するポインタ
nOutBufSize	:	ブザー周波数情報のデータサイズ
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

ブザー周波数情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	0 ～ 15 [(設定値+1)kHz]

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

現状のブザー周波数の取得を行います。処理が成功するとブザー周波数情報に現在の周波数設定が格納されます。

IOCTL_BUZ_SAVEREGHZ　[DeviceIoControl]

機能

現在のタッチ音設定、ブザー周波数設定をレジストリに書込みます。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

現在のタッチ音設定、ブザー周波数設定をレジストリに書込みます。
レジストリ情報は OS 起動時、IOCTL_BUZ_LOADREGHZ（DeviceIoControl）実行時にブザー設定に反映されます。

IOCTL_BUZ_LOADREGHZ　[DeviceIoControl]

機能

レジストリからタッチ音設定、ブザー周波数設定を読込みます。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

レジストリからタッチ音設定、ブザー周波数設定を読込み、ブザーの状態を変更します。

IOCTL_BUZ_ON　[DeviceIoControl]

機能

指定した時間ブザーを ON します。

パラメータ

lpInBuf　　：　ブザーON 時間情報を格納するポインタ
nInBufSize　　：　ブザーON 時間情報のデータサイズ
lpOutBuf　　：　NULL
nOutBufSize　　：　0
lpBytesReturned　：　実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped　　：　NULL

ブザーON 時間情報

データタイプ	：	DWORD
データサイズ	：	4 バイト
内容	：	ON 時間 [msec] 0 を指定した場合、IOCTL_BUZ_OFF で止めるまで ON のままです。

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

ON 時間情報に指定した時間、ブザーが ON となります。ON 時間情報に 0 を指定した場合は、ON 時間は無効となり、IOCTL_BUZ_OFF（DeviceIoControl）を実行してブザーを OFF するまで ON のままとなります。

注意

ブザーデバイスのハンドル毎にブザーの ON/OFF が可能です。ハンドル 1 がブザーを OFF しても、ハンドル 2 が ON している場合はブザーが鳴ったままになります。アプリケーションでブザーを ON した状態でアプリケーションを終了(ブザーデバイスのハンドルをクローズ)した場合は、強制的に OFF になります。

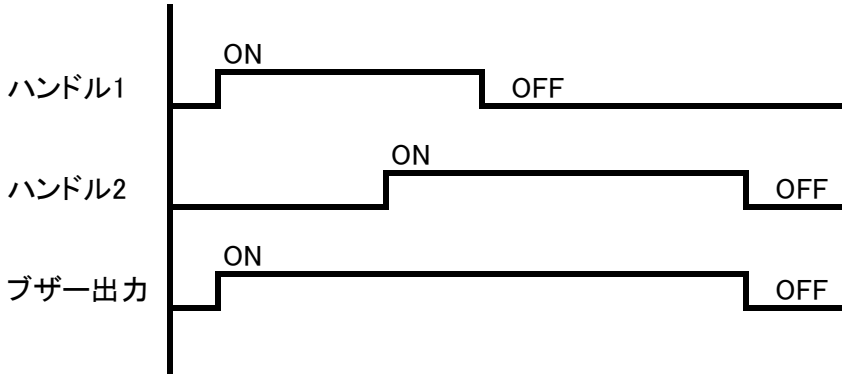

図 5-3-4-1. タイミングチャート

IOCTL_BUZ_OFF　[DeviceIoControl]

機能

ブザーを OFF します。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

ブザーを OFF します。

注意

ブザーデバイスのハンドル毎にブザーの ON/OFF が可能です。ハンドル 1 がブザーを OFF しても、ハンドル 2 が ON している場合はブザーが鳴ったままになります。アプリケーションでブザーを ON した状態でアプリケーションを終了(ブザーデバイスのハンドルをクローズ)した場合は、強制的に OFF になります。

IOCTL_BUZ_STATUS　[DeviceIoControl]

機能

ブザーの状態を取得します。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	ブザー状態情報を格納するポインタ
nOutBufSize	:	ブザー状態情報のデータサイズ
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

ブザー状態情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	0: OFF, 1: ON

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

ブザーの状態を取得します。ブザーが鳴っている場合は、ブザー状態情報に 1 が格納されます。ブザーが鳴っていない場合は 0 が格納されます。

注意

ブザーデバイスのハンドル毎にブザーの ON/OFF が可能です。システム上でブザーが鳴っている状態（タッチ音または、オープンされているハンドルの中の何れかがブザーを ON している場合）に 1 が取得されます。

５－３－５　サンプルコード

●ブザー周波数

「¥SDK¥Algo¥Sample¥Sample_Buzzer¥BuzzerHz」にブザー周波数取得と設定のサンプルコードを用意しています。リスト 5-3-5-1 にサンプルコードを示します。

リスト 5-3-5-1. ブザー周波数

```
/**
    ブザー周波数制御サンプルソース
**/

#include "stdafx.h"
#include <windows.h>
#include <commctrl.h>

#include <asp_g3n¥buzzer.h>

#define DRIVER_FILENAME     _T("BUZ1:")

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE  m_hDevice;                      // デバイスハンドル
    BOOL    ret;
    DWORD   retlen;
    DWORD   set_data;
    DWORD   get_data;

    /*
     * 起動引数から変更を行うブザー音設定を取得
     *   ブザー音の周波数を設定します。周波数は『設定値+1kHz』になります
     *   設定範囲は、『0～15』です
     */
    if(argc != 2){
        _tprintf(_T("invalid arg¥n"));
        _tprintf(_T("BuzzerHz.exe [Hz(0~15)]¥n"));
        return -1;
    }
    _stscanf(*(argv + 1), _T("%u"), &set_data);

    /*
     *デバイスの Open
     */
    m_hDevice = CreateFile(
                    DRIVER_FILENAME,
                    GENERIC_READ | GENERIC_WRITE,
                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL
                    );
    if(m_hDevice == INVALID_HANDLE_VALUE){
```

```
        _tprintf(_T("CreateFile: NG¥n"));
        return FALSE;
    }

    /*
     *ブザーの周波数を設定
     */
    ret = DeviceIoControl(
                m_hDevice,
                IOCTL_BUZ_SETHZ,
                &set_data,
                sizeof(DWORD),
                NULL,
                0,
                &retlen,
                NULL
                );
    if(!ret){
        _tprintf(_T("DeviceIoControl: IOCTL_BUZ_SETHZ NG¥n"));
        CloseHandle(m_hDevice);
        return -1;
    }
    /*
     *ブザーの周波数を取得
     */
    ret = DeviceIoControl(
                m_hDevice,
                IOCTL_BUZ_GETHZ,
                NULL,
                0,
                &get_data,
                sizeof(DWORD),
                &retlen,
                NULL
                );
    if(!ret){
        _tprintf(_T("DeviceIoControl: IOCTL_BUZ_GETHZ NG¥n"));
        CloseHandle(m_hDevice);
        return -1;
    }
    _tprintf(_T("Buzzer Hz=%d¥n"), get_data);

    /*
     *デバイスの Close
     */
    CloseHandle(m_hDevice);

    return 0;
}
```

●ブザーON/OFF

「¥SDK¥Algo¥Sample¥Sample_Buzzer¥BuzzerOnOff」にブザーON/OFF 制御を行うサンプルコードを用意しています。リスト 5-3-5-2 にサンプルコードを示します。

リスト 5-3-5-2. ブザーON/OFF

```
/**
    ブザーON/OFF 制御サンプルソース
**/

#include "stdafx.h"
#include <windows.h>
#include <commctrl.h>

#include <asp_g3n¥buzzer.h>

#define DRIVER_FILENAME     _T("BUZ1:")

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE  m_hDevice;                      // デバイスハンドル
    BOOL    ret;
    DWORD   retlen;
    DWORD   set_data;
    DWORD   get_data;
    DWORD   ctrl_code;

    /*
     * 起動引数からブザーの ON/OFF 変更値を取得します。
     *   1 : ブザーON
     *   0 : ブザーOFF
     */
    if(argc != 2){
        _tprintf(_T("invalid arg¥n"));
        _tprintf(_T("BuzzerOnOff.exe [OnOff(0:OFF, 1:ON)]¥n"));
        return -1;
    }
    _stscanf(*(argv + 1), _T("%u"), &set_data);

    /*
     *デバイスの Open
     */
    m_hDevice = CreateFile(
                    DRIVER_FILENAME,
                    GENERIC_READ | GENERIC_WRITE,
                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL
                    );
    if(m_hDevice == INVALID_HANDLE_VALUE){
        _tprintf(_T("CreateFile: NG¥n"));
```

```
        return FALSE;
    }

    /*
     *ブザーの ON/OFF を設定
     */
    if(set_data == 1){
        ctrl_code       = IOCTL_BUZ_ON;
        set_data        = 1000;
    }else{
        ctrl_code       = IOCTL_BUZ_OFF;
    }

    ret = DeviceIoControl(
                m_hDevice,
                ctrl_code,
                &set_data,
                sizeof(DWORD),
                NULL,
                0,
                &retlen,
                NULL
                );
    if(!ret){
        if(ctrl_code == IOCTL_BUZ_ON){
            _tprintf(_T("DeviceIoControl: IOCTL_BUZ_ON NG¥n"));
        }
        else{
            _tprintf(_T("DeviceIoControl: IOCTL_BUZ_OFF NG¥n"));
        }
        CloseHandle(m_hDevice);
        return -1;
    }

    /*
     *ブザーの ON/OFF を取得
     */
    ret = DeviceIoControl(
                m_hDevice,
                IOCTL_BUZ_STATUS,
                NULL,
                0,
                &get_data,
                sizeof(DWORD),
                &retlen,
                NULL
                );
    if(!ret){
        _tprintf(_T("DeviceIoControl: IOCTL_BUZ_STATUS NG¥n"));
        CloseHandle(m_hDevice);
        return -1;
    }
```

```
    _tprintf(_T("Buzzer OnOff=%d¥n"), get_data);

    for(;set_data > 0;set_data--) {
        Sleep(1);
    }

    /*
     *デバイスの Close
     */
    CloseHandle(m_hDevice);

    return 0;
}
```

●タッチブザー

「¥SDK¥Algo¥Sample¥Sample_Buzzer¥BuzzerTouch」にブザーの取得と設定を行うサンプルコードを用意しています。リスト 5-3-5-3 にサンプルコードを示します。

リスト 5-3-5-3. タッチブザー

```
/**
    タッチパネル音制御サンプルソース
**/

#include "stdafx.h"
#include <windows.h>
#include <commctrl.h>

#include <asp_g3n¥buzzer.h>

#define DRIVER_FILENAME     _T("BUZ1:")

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE  m_hDevice;                      // デバイスハンドル
    BOOL    ret;
    DWORD   retlen;
    DWORD   set_data;
    DWORD   get_data;

    /*
     * 起動引数からタッチパネル音の ON/OFF 変更値を取得します。
     *   1 : タッチパネル音 ON
     *   0 : タッチパネル音 OFF
     */
    if(argc != 2){
        _tprintf(_T("invalid arg¥n"));
        _tprintf(_T("BuzzerAuto.exe [auto(0:OFF, 1:ON)]¥n"));
        return -1;
    }
    _stscanf(*(argv + 1), _T("%u"), &set_data);

    /*
     *デバイスの Open
     */
    m_hDevice = CreateFile(
                    DRIVER_FILENAME,
                    GENERIC_READ | GENERIC_WRITE,
                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL
                    );
    if(m_hDevice == INVALID_HANDLE_VALUE){
        _tprintf(_T("CreateFile: NG¥n"));
        return FALSE;
```

```
    }

    /*
     *タッチパネル音 ON/OFF を設定
     */
    ret = DeviceIoControl(
                m_hDevice,
                IOCTL_BUZ_SETTOUCH,
                &set_data,
                sizeof(DWORD),
                NULL,
                0,
                &retlen,
                NULL
                );
    if(!ret){
        _tprintf(_T("DeviceIoControl: IOCTL_BUZ_SETHZ NG¥n"));
        CloseHandle(m_hDevice);
        return -1;
    }

    /*
     *タッチパネル音 ON/OFF を取得
     */
    ret = DeviceIoControl(
                m_hDevice,
                IOCTL_BUZ_GETTOUCH,
                NULL,
                0,
                &get_data,
                sizeof(DWORD),
                &retlen,
                NULL
                );
    if(!ret){
        _tprintf(_T("DeviceIoControl: IOCTL_BUZ_GETHZ NG¥n"));
        CloseHandle(m_hDevice);
        return -1;
    }
    _tprintf(_T("Touch Sound OnOff=%d¥n"), get_data);

    /*
     *デバイスの Close
     */
    CloseHandle(m_hDevice);

    return 0;
}
```

５－４　　汎用入出力

５－４－１　　汎用入出力について

本シリーズには、汎用入力６点、汎用出力４点の汎用入出力があります。

汎用入力にはハードウェアによる IN0 リセット機能、IN1 割込み機能が実装されています。IN0 入力時にハードウェアリセットを掛けることができます。また、IN1 入力時に割込みを発生させることができます。

５－４－２　　汎用出力ドライバ

汎用出力ドライバは汎用出力を、ユーザーアプリケーションから利用できるようにします。

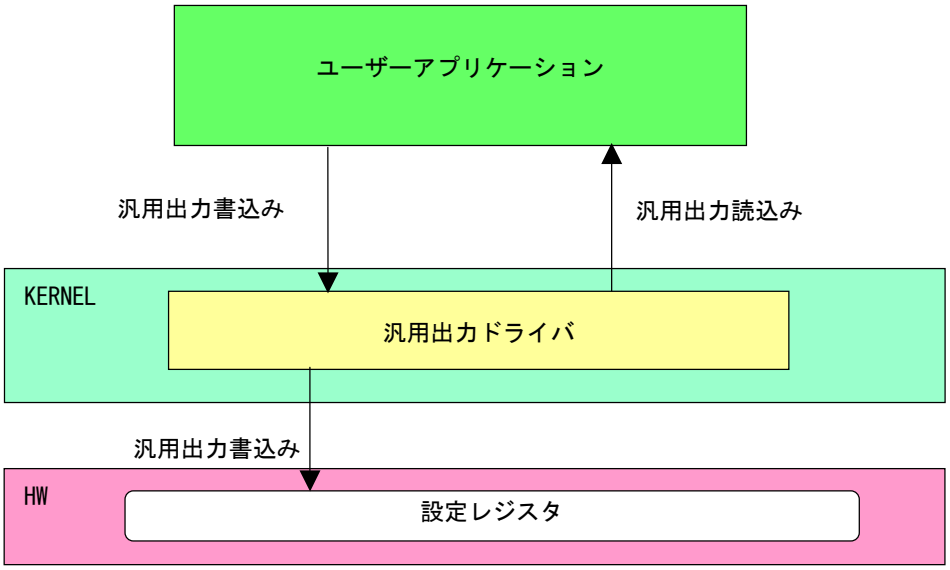

図 5-4-2-1. 汎用出力ドライバ

５－４－３　汎用出力デバイス

汎用出力ドライバは汎用出力デバイスを生成します。ユーザーアプリケーションは、デバイスファイルにアクセスすることによって汎用出力を制御します。

汎用出力デバイス

デバイスファイル

GNO1:

説明

汎用出力の制御を行うことができます。

CreateFile

デバイスファイル(GNO1:)をオープンし、デバイスハンドルを取得します。

```
hGenOUT = CreateFile(
    _T("GNO1:"),
    GENERIC_READ | GENERIC_WRITE,
    FILE_SHARE_READ | FILE_SHARE_WRITE,
    NULL,
    OPEN_EXISTING,
    0,
    NULL
);
```

CloseHandle

デバイスハンドルをクローズします。

```
CloseHandle(hGenOUT);
```

ReadFile

出力データを読込みます。
読込みサイズは1Byteまでです。1Byte以上の場合はエラーとなります。

```
res = ReadFile(hGenOUT, &OutData, 1, NULL);
```

WriteFile

出力データを書込みます。
書込みサイズは1Byteまでです。1Byte以上の場合はエラーとなります。

```
res = WriteFile(hGenOUT, &OutData, 1, NULL);
```

５－４－４　汎用出力リファレンス

ReadFile

機能

出力データを読込みます。

書式

```
BOOL ReadFile(
    HANDLE hFile,
    LPVOID lpBuffer,
    DWORD nNumberOfBytesToRead,
    LPDWORD lpNumberOfBytesRead,
    LPOVERLAPPED lpOverlapped
);
```

パラメータ

hFile	：	汎用出力デバイスのハンドル
lpBuffer	：	出力データバッファのポインタ
nNumberOfBytesToRead	：	出力データバッファのサイズ
lpNumberOfBytesRead	：	読込んだバイト数を格納するためのポインタ
lpOverlapped	：	NULL

出力データバッファ

データタイプ　：　UCHAR
データサイズ　：　１バイト
内容　　：　出力データ

bit	7	6	5	4	3	2	1	0
OUT	／	／	／	／	OT4	OT3	OT2	OT1

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

出力データを読込みます。現在の出力状態を確認することができます。
汎用出力デバイスは 1Byte の読込みしか出来ません。NNumberOfBytesToRead に 1Byte 以上を指定した場合はエラーとなります。

WriteFile

機能

出力データを書込みます。

書式

```
BOOL WriteFile(
    HANDLE hFile,
    LPVOID lpBuffer,
    DWORD nNumberOfBytesToRead,
    LPDWORD lpNumberOfBytesRead,
    LPOVERLAPPED lpOverlapped
);
```

パラメータ

hFile	：	汎用出力デバイスのハンドル
lpBuffer	：	出力データバッファのポインタ
nNumberOfBytesToWrite	：	出力データバッファのサイズ
lpNumberOfBytesWritten	：	書込んだバイト数を格納するためのポインタ
lpOverlapped	：	NULL

出力データバッファ

データタイプ	：	UCHAR
データサイズ	：	1 バイト
内容	：	出力データ

bit	7	6	5	4	3	2	1	0
OUT					OT4	OT3	OT2	OT1

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

出力データを書込みます。
汎用出力デバイスは 1Byte の書込みしか出来ません。nNumberOfBytesToWrite に 1Byte 以上を指定した場合はエラーとなります。

５－４－５　汎用入力ドライバ

汎用入力ドライバは汎用入力をユーザーアプリケーションから利用できるようにします。また、IN0 リセット機能、IN1 割込み機能を、ユーザーアプリケーションから利用できるようにします。ユーザーアプリケーションから、IN0 リセット機能と IN1 割込み機能の ON/OFF 設定とイベントによる IN1 割込み通知の機能を使用することができます。

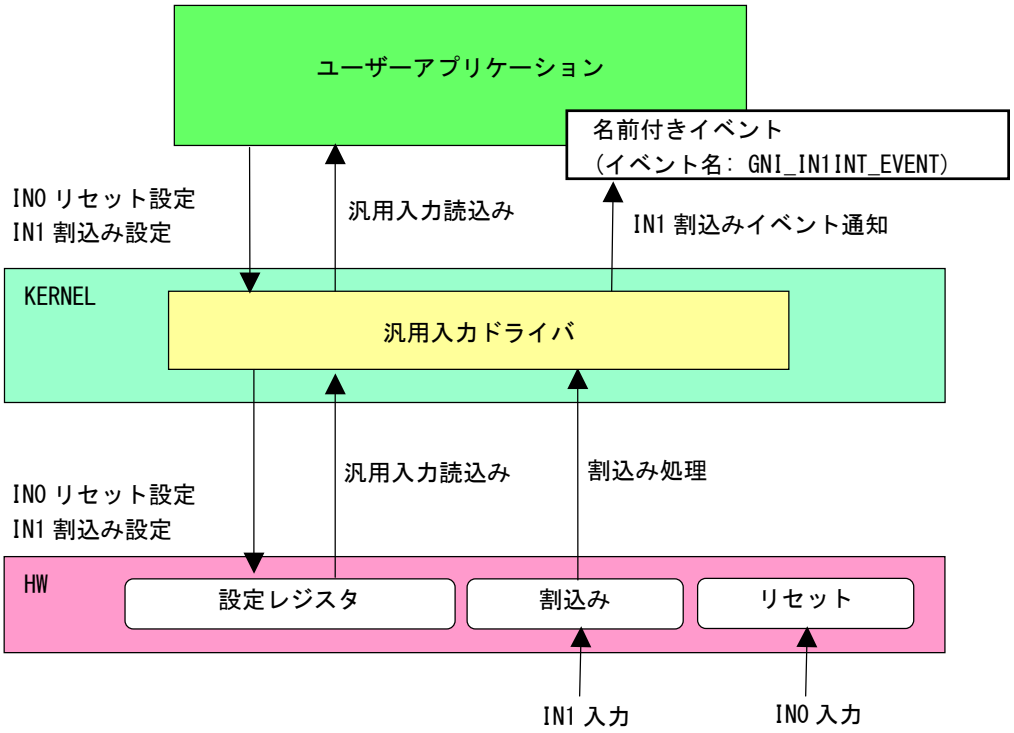

図 5-4-5-1. 汎用入力ドライバ

５－４－６　汎用入力デバイス

汎用入力ドライバは汎用入力デバイスを生成します。ユーザーアプリケーションは、デバイスファイルにアクセスすることによって汎用入力を制御します。

汎用入力デバイス

デバイスファイル

GNI1:

説明

汎用入力の読込み、IN0リセット設定、IN1割り込み設定を行うことができます。

CreateFile

デバイスファイル(GNI1:)をオープンし、デバイスハンドルを取得します。

```
hGenIN = CreateFile(
    _T("GNI1:"),
    GENERIC_READ | GENERIC_WRITE,
    FILE_SHARE_READ | FILE_SHARE_WRITE,
    NULL,
    OPEN_EXISTING,
    0,
    NULL
);
```

CloseHandle

デバイスハンドルをクローズします。

```
CloseHandle(hGenIN);
```

ReadFile

入力データを読込みます。
読込みサイズは1Byteまでです。1Byte以上の場合はエラーとなります。

```
res = ReadFile(hGenIN, &InData, 1, NULL);
```

DeviceIoControl

- **IOCTL_GNI_SET_IN1INT**
  IN1割り込みを設定します。
- **IOCTL_GNI_GET_IN1INT**
  現在のIN1割り込み設定を取得します。
- **IOCTL_GNI_SET_INORST**
  IN0リセットを設定します。
- **IOCTL_GNI_GET_INORST**
  現在のIN0リセット設定を取得します。

５－４－７　汎用入力 IN1 割込みの使用手順

基本的な使用手順を以下に示します。

IN1 割込み通知用イベントハンドルを作成後、IN1 割込み通知用イベントハンドルでのイベント待ち準備が整ったところで、RAS-IN デバイスに IN1 割込み有効を設定します。

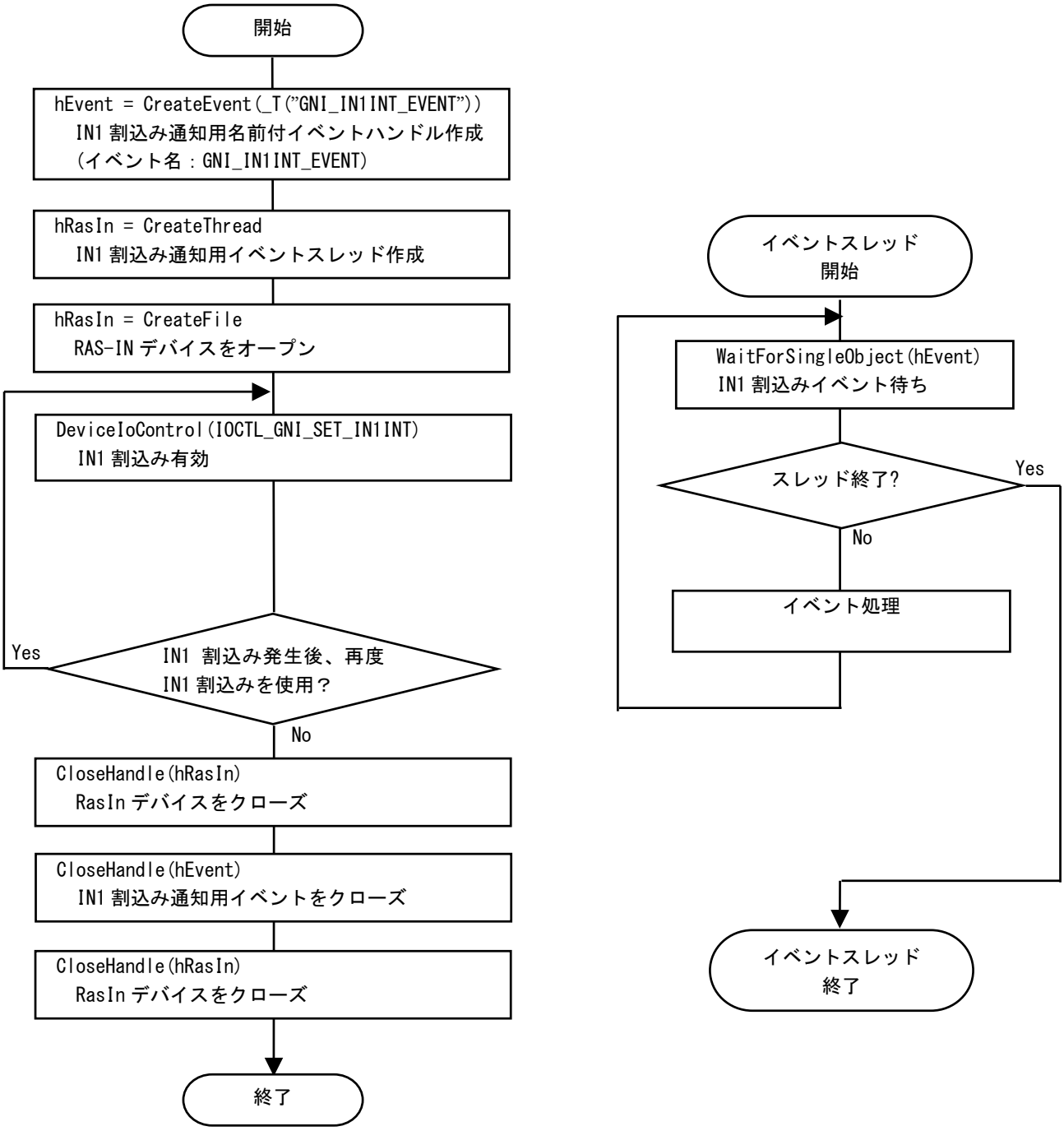

図 5-4-7-1. IN1 割込み仕様手順

５－４－８　汎用入力リファレンス

ReadFile

機能

入力データを読込みます。

書式

```
BOOL ReadFile(
    HANDLE hFile,
    LPVOID lpBuffer,
    DWORD nNumberOfBytesToRead,
    LPDWORD lpNumberOfBytesRead,
    LPOVERLAPPED lpOverlapped
);
```

パラメータ

hFile	：	汎用出力デバイスのハンドル
lpBuffer	：	入力データバッファのポインタ
nNumberOfBytesToRead	：	入力データバッファのサイズ
lpNumberOfBytesRead	：	読込んだバイト数を格納するためのポインタ
lpOverlapped	：	NULL

出力データバッファ

データタイプ	：	UCHAR
データサイズ	：	1 バイト
内容	：	入力データ

bit	7	6	5	4	3	2	1	0
IN	／	／	IN6	IN5	IN4	IN3	IN2	IN1

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

入力データを読込みます。現在の入力状態を確認することができます。
汎用入力デバイスは 1Byte の読込みしか出来ません。NNumberOfBytesToRead に 1Byte 以上を指定した場合はエラーとなります。

IOCTL_GNI_SET_IN1INT　[DeviceIoControl]

機能

IN1 割込みを設定します。

パラメータ

lpInBuf	:	IN1 割込み情報を格納するポインタ
nInBufSize	:	IN1 割込み情報のデータサイズ
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

IN1 割込み情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	1: IN1 割込み有効, 0: IN1 割込み無効

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

IN1 割込みの設定を行います。
IN1 割込みを有効にする場合は、IN1 割込み情報に 1、無効にする場合は 0 を設定して DeviceIoControl を実行してください。

IOCTL_GNI_GET_IN1INT　[DeviceIoControl]

機能

IN1 割込み設定を取得します。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	IN1 割込み情報を格納するためのポインタ
nOutBufSize	:	IN1 割込み情報のデータサイズ
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

IN1 割込み情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	1: IN1 割込み有効, 0: IN1 割込み無効

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

現在の IN1 割込み設定値を取得します。

IOCTL_GNI_SET_INORST　[DeviceIoControl]

機能

INO リセットを設定します。

パラメータ

lpInBuf　　：　INO リセット情報を格納するポインタ
nInBufSize　　：　INO リセット情報のデータサイズ
lpOutBuf　　：　NULL
nOutBufSize　　：　0
lpBytesReturned　：　実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped　　：　NULL

INO リセット情報

データタイプ	：	DWORD
データサイズ	：	4 バイト
内容	：	1: INO リセット有効, 0: INO リセット無効

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

INO リセットを設定します。
INO リセットを有効にする場合は、INO リセット情報に 1、無効にする場合は 0 を設定して DeviceIoControl を実行してください。

IOCTL_GNI_GET_INORST　[DeviceIoControl]

機能

INO リセット設定を取得します。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	INO リセット情報を格納するポインタ
nOutBufSize	:	INO リセット情報のデータサイズ
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

INO リセット情報

データタイプ	:	DWORD
データサイズ	:	4 バイト
内容	:	1: INO リセット有効, 0: INO リセット無効

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

現在の INO リセット設定値を取得します。

５－４－９　サンプルコード

「¥SDK¥Algo¥Sample¥Sample_GenIO¥GenIo」に汎用入出力を使用したサンプルコードを用意しています。リスト 5-4-9-1 にサンプルコードを示します。

リスト 5-4-9-1. 汎用入出力

```
/**
    汎用入出力制御サンプルソース
**/

#include "stdafx.h"
#include <windows.h>
#include <commctrl.h>

#define GENOUT_DRIVERNAME   _T("GNO1:")
#define GENIN_DRIVERNAME    _T("GNI1:")

BOOL Read(HANDLE hDevice, BYTE *pReadBuffer)
{
    BOOL    ret;
    DWORD   retlen;

    ret = ReadFile(hDevice, pReadBuffer, 1, &retlen, NULL);

    if(!ret){
        return FALSE;
    }
    return TRUE;
}

BOOL Write(HANDLE hDevice, BYTE Data)
{
    BOOL    ret;
    DWORD   retlen;

    ret = WriteFile(hDevice, &Data, 1, &retlen, NULL);

    if(!ret){
        return FALSE;
    }
    return TRUE;
}

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE  hGenIn, hGenOut;
    BOOL    ret;
    DWORD   i;
    BYTE    outdata=0x0001;
    BYTE    indata=0x0000;

    /* 汎用出力
        汎用出力の 4 点を 1 点ずつ出力を行います。
```

```
*/

/* 汎用出力ﾃﾞﾊﾞｲｽのｵｰﾌﾟﾝ */
hGenOut = CreateFile(
                GENOUT_DRIVERNAME,
                GENERIC_READ | GENERIC_WRITE,
                FILE_SHARE_READ | FILE_SHARE_WRITE,
                NULL,
                OPEN_EXISTING,
                0,
                NULL
            );
if(hGenOut == INVALID_HANDLE_VALUE){
    _tprintf(_T("GenOut: CreateFile: NG¥n"));
    return -1;
}

for (i=0; i<4; i++){
    ret = Write(hGenOut, outdata);
    if ( !ret ) {
        _tprintf(_T("GenOut: Write NG¥n"));
        CloseHandle(hGenOut);
        return -1;
    }
    outdata <<= 1;
    Sleep(500);
}
outdata = 0x0000;
ret = Write(hGenOut, outdata);
if ( !ret ) {
    _tprintf(_T("GenOut: Write NG¥n"));
    CloseHandle(hGenOut);
    return -1;
}

/* 汎用出力ﾃﾞﾊﾞｲｽのｸﾛｰｽﾞ */
CloseHandle(hGenOut);

/* 汎用入力
    汎用入力の入力を行います。
*/

/* 汎用入力ﾃﾞﾊﾞｲｽのｵｰﾌﾟﾝ */
hGenIn = CreateFile(
                GENIN_DRIVERNAME,
                GENERIC_READ | GENERIC_WRITE,
                FILE_SHARE_READ | FILE_SHARE_WRITE,
                NULL,
                OPEN_EXISTING,
                0,
                NULL
            );
```

```
    if(hGenIn == INVALID_HANDLE_VALUE){
        _tprintf(_T("GenIn: CreateFile: NG¥n"));
        return -1;
    }

    ret = Read(hGenIn, &indata);
    if ( !ret ) {
        _tprintf(_T("GenIn: Read NG¥n"));
        CloseHandle(hGenIn);
        return -1;
    }
    else{
        indata &= 0x3F;
        /* IN0 状態 */
        if (indata & 0x01) _tprintf(_T("IN0: ON¥n"));
        else               _tprintf(_T("IN0: OFF¥n"));
        /* IN1 状態 */
        if (indata & 0x02) _tprintf(_T("IN1: ON¥n"));
        else               _tprintf(_T("IN1: OFF¥n"));
        /* IN2 状態 */
        if (indata & 0x04) _tprintf(_T("IN2: ON¥n"));
        else               _tprintf(_T("IN2: OFF¥n"));
        /* IN3 状態 */
        if (indata & 0x08) _tprintf(_T("IN3: ON¥n"));
        else               _tprintf(_T("IN3: OFF¥n"));
        /* IN4 状態 */
        if (indata & 0x10) _tprintf(_T("IN4: ON¥n"));
        else               _tprintf(_T("IN4: OFF¥n"));
        /* IN5 状態 */
        if (indata & 0x20) _tprintf(_T("IN5: ON¥n"));
        else               _tprintf(_T("IN5: OFF¥n"));
    }

    /* 汎用入力ﾃﾞﾊﾞｲｽのｸﾛｰｽﾞ */
    CloseHandle(hGenIn);

    return 0;
}
```

●IN0 リセット

「¥SDK¥Algo¥Sample¥Sample_Reset¥In0Reset」に IN0 リセット機能のサンプルコードを用意しています。リスト 5-4-9-2 にサンプルコードを示します。

リスト 5-4-9-2. IN0 リセット

```
/**
    汎用入力 IN0 リセット制御方法サンプルソース
**/

#include "stdafx.h"
#include <windows.h>
#include <commctrl.h>

#include <asp_g3n¥GenIO.h>

#define GENIN_DRIVERNAME    _T("GNI1:")

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE hGenIn;
    BOOL ret;
    DWORD retlen;
    DWORD reset;

    /* 汎用入力ﾃﾞﾊﾞｲｽのｵｰﾌﾟﾝ */
    hGenIn = CreateFile(
                    GENIN_DRIVERNAME,
                    GENERIC_READ | GENERIC_WRITE,
                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL
                    );
    if(hGenIn == INVALID_HANDLE_VALUE){
        _tprintf(_T("GenIn: CreateFile: NG¥n"));
        return -1;
    }

    /*
     * IN0 リセットを有効にする
     *
     * reset: 1    有効
     *      : 0    無効
     */
    reset = 1;
    ret = DeviceIoControl(
                hGenIn,
                IOCTL_GNI_SET_INORST,
                &reset,
                sizeof(DWORD),
```

```
                NULL,
                0,
                &retlen,
                NULL
                );
    if(!ret){
        _tprintf(_T("IOCTL_GNI_SET_INORST failed¥n"));
        CloseHandle(hGenIn);
        return -1;
    }
    else {
        _tprintf(_T("ioctl set INORST success: %d¥n"), reset);
    }

    /*
     * INO リセットを確認する
     */
    ret = DeviceIoControl(
                hGenIn,
                IOCTL_GNI_GET_INORST,
                NULL,
                0,
                &reset,
                sizeof(DWORD),
                &retlen,
                NULL
                );
    if(!ret){
        _tprintf(_T("IOCTL_GNI_GET_INORST failed¥n"));
        CloseHandle(hGenIn);
        return -1;
    }
    else {
        _tprintf(_T("ioctl get INORST success: %d¥n"), reset);
    }

    CloseHandle(hGenIn);

    return 0;
}
```

●IN1 割込み

「¥SDK¥Algo¥Sample¥Sample_Interrupt¥In1Interrupt」に IN1 割込み機能を使用したサンプルコードを用意しています。リスト 5-4-9-3 にサンプルコードを示します。

リスト 5-4-9-3. IN1 割込み

```
/**
    汎用入力 IN1 割込み制御サンプルソース
**/

#include "stdafx.h"
#include <windows.h>
#include <commctrl.h>

#include <asp_g3n¥GenIO.h>

#define GENIN_DRIVERNAME    _T("GNI1:")

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE hEvent;
    HANDLE hGenIn;
    BOOL ret;
    DWORD retlen;
    DWORD Interrupt;

    /* イベントオブジェクトの作成
     *   複数アプリケーションでイベントを共有する場合は、
     *   CreateFile でドライバオブジェクトを作成するより前に
     *   CreateEvent で手動リセットを有効にした名前付き
     *   イベントを作成する必要があります。
     *   単アプリケーションの場合、自動リセットで問題ありません。
     */
    hEvent = CreateEvent(NULL, FALSE, FALSE, GNI_IN1INT_EVENTNAME);
    if(hEvent == NULL){
        _tprintf(_T("Event Create failed¥n"));
        return -1;
    }

    /* 汎用入力ﾃﾞﾊﾞｲｽのｵｰﾌﾟﾝ */
    hGenIn = CreateFile(
                    GENIN_DRIVERNAME,
                    GENERIC_READ | GENERIC_WRITE,
                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL
                    );
    if(hGenIn == INVALID_HANDLE_VALUE){
        _tprintf(_T("GenIn: CreateFile: NG¥n"));
        return -1;
    }
```

```
/*
 * IN1 割り込みを有効にする
 * Interrupt: 1    有効
 *          : 0    無効
 */
Interrupt = 1;
ret = DeviceIoControl(
            hGenIn,
            IOCTL_GNI_SET_IN1INT,
            &Interrupt,
            sizeof(DWORD),
            NULL,
            0,
            &retlen,
            NULL
            );
if(!ret){
    _tprintf(_T("IOCTL_GNI_SET_IN1INT failed¥n"));
    CloseHandle(hEvent);
    CloseHandle(hGenIn);
    return -1;
}
_tprintf(_T("SET IN1INT: %d¥n"), Interrupt);

/*
 * 現在の設定を取得
 *
 * Interrupt: 1    有効
 *          : 0    無効
 */
ret = DeviceIoControl(
            hGenIn,
            IOCTL_GNI_GET_IN1INT,
            NULL,
            0,
            &Interrupt,
            sizeof(DWORD),
            &retlen,
            NULL
            );
if(!ret){
    _tprintf(_T("IOCTL_GNI_GET_IN1INT failed¥n"));
    CloseHandle(hEvent);
    CloseHandle(hGenIn);
    return -1;
}
_tprintf(_T("GET IN1INT: %d¥n"), Interrupt);

_tprintf(_T("Wait For IN1Interrupt Event¥n"));
if(WaitForSingleObject(hEvent, INFINITE) != WAIT_OBJECT_0){
    _tprintf(_T("WaitForEvent Error?¥n"));
```

```
    }
    _tprintf(_T("Detect IN1Interrupt Event¥n"));

    CloseHandle(hEvent);
    CloseHandle(hGenIn);

    return 0;
}
```

５－５　　シリアルコントロール機能

５－５－１　　シリアルコントロール機能について

本シリーズには、RS-232C 以外に RS-422、RS-485 通信を行う事ができるシリアルポートが 2 ポート実装されています。シリアルコントロール機能により、指定したポートを RS-232C/RS-422/RS-485 に切替えることができます。

５－５－２　　シリアルコントロールドライバについて

シリアルコントロール(以下、SciCtl と称します)ドライバはシリアルコントロール機能を、ユーザーアプリケーションから利用できるようにします。ユーザーアプリケーションから、ポートの設定と送信イネーブルタイムアウト時間を設定することができます。

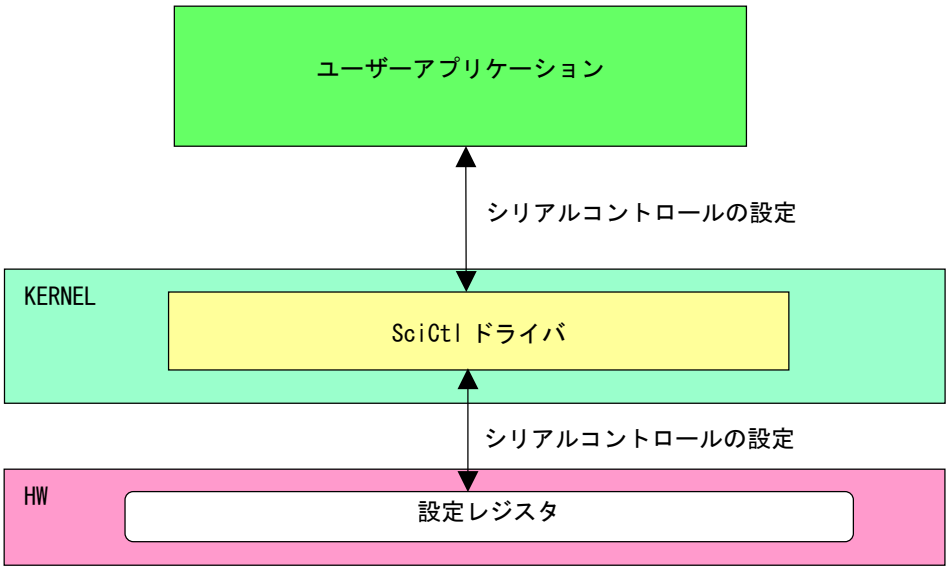

図 5-5-2-1. SciCtl ドライバ

５－５－３　SciCtlデバイス

SciCtl ドライバは SciCtl デバイスを生成します。ユーザーアプリケーションは、デバイスファイルにアクセスすることによってシリアルコントロール機能を操作します。

SciCtlデバイス

デバイスファイル

SCL1:

説明

シリアルコントロール機能の設定をすることができます。

レジストリ設定

[KEY]　　HKEY_LOCAL_MACHINE¥Drivers¥BuiltIn¥SciCtl
[VALUE:DWORD]　　Com1Type, Com2Type
　シリアルポートタイプを設定します。[0: RS-232C, 1: RS-422, 2: RS-485]
　ドライバ起動時(OS起動時)にこの値を参照します。(デフォルト値: 0)
[VALUE:DWORD]　　Com1Timer, Com2Timer
　RS-485の送信イネーブル時間を設定します。マイクロ秒単位で設定します。
　ドライバ起動時(OS起動時)にこの値を参照します。(デフォルト値: 0)

CreateFile

デバイスファイル(SCL1:)をオープンし、デバイスハンドルを取得します。

```
hSciCtl = CreateFile(
    _T("SCL1:"),
    GENERIC_READ | GENERIC_WRITE,
    FILE_SHARE_READ | FILE_SHARE_WRITE,
    NULL,
    OPEN_EXISTING,
    0,
    NULL
);
```

CloseHandle

デバイスハンドルをクローズします。

```
CloseHandle(hSciCtl);
```

DeviceIoControl

- **IOCTL_SCICTL_SETCONFIG**
  シリアルコントロールを設定します。
- **IOCTL_SCICTL_GETCONFIG**
  シリアルコントロールを取得します。

５－５－４　SciCtlリファレンス

IOCTL_SCICTL_SETCONFIG　[DeviceIoControl]

機能

シリアルコントロールの設定を行います。

パラメータ

lpInBuf	:	SCICTL_CONF_PAR を格納するためのポインタ
nInBufSize	:	SCICTL_CONF_PAR のサイズ
lpOutBuf	:	NULL
nOutBufSize	:	0
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

SCICTL_CONF_PAR

```
typedef struct {
    WORD ch;
    WORD type;
    WORD timer;
} SCICTL_CONF_PAR, *P_SCICTL_CONF_PAR;
```

ch	: チャンネル [0: SIO1, 1: SIO2]
type	: シリアルポートタイプ [0: RS-232C, 1: RS-422, 2: RS-485]
timer	: RS-485 の送信イネーブル時間 [0～65535us]

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

シリアルコントロールの設定を行います。シリアルの通信を開始する前に、このコントロールを実行してシリアルコントロールの設定を行うようにしてください。

IOCTL_SCICTL_GETCONFIG　[DeviceIoControl]

機能

シリアルコントロール設定を取得します。

パラメータ

lpInBuf	:	NULL
nInBufSize	:	0
lpOutBuf	:	SCICTL_CONF_PAR を格納するためのポインタ
nOutBufSize	:	SCICTL_CONF_PAR のサイズ
lpBytesReturned	:	実際の出力バイト数を受け取る変数へのポインタ
lpOverlapped	:	NULL

SCICTL_CONF_PAR

```
typedef struct {
    WORD ch;
    WORD type;
    WORD timer;
} SCICTL_CONF_PAR, *P_SCICTL_CONF_PAR;
```

ch	: チャンネル [0: SIO1, 1: SIO2]
type	: シリアルポートタイプ [0: RS-232C, 1: RS-422, 2: RS-485]
timer	: RS-485 の送信イネーブル時間 [0～65535us]

戻り値

処理が成功すると TRUE を返します。失敗の場合は FALSE を返します。

説明

現在のシリアルコントロール設定値を取得します。

５－５－５　サンプルコード

「¥SDK¥Algo¥Sample¥Sample_SerialControl¥SerialControlConfig」にシリアルポートの RS-232C/422/485 切替えのサンプルコードを用意しています。リスト 5-5-5-1 にサンプルコードを示します。

リスト 5-5-5-1. RS-232C/422/485 切替え

```
/**
    シリアルポート RS422/RS485/RS232C 切替え制御方法サンプルソース
**/

#include "stdafx.h"
#include <windows.h>
#include <commctrl.h>

#include <asp_g3n¥SciCtl.h>

#define DRIVER_FILENAME     _T("SCL1:")

int _tmain(int argc, _TCHAR* argv[])
{
    HANDLE hSciCtl;
    ULONG retlen;
    ULONG errno;
    BOOL ret;
    int ch, type, timer;
    SCICTL_CONF_PAR conf;

    if(argc != 4){
        printf("invalid arg¥n");
        printf("SerialControlConfig.exe [ch(0,1)] [type] [timer]¥n");
        return -1;
    }

    _stscanf(*(argv + 1), _T("%u"), &ch);
    _stscanf(*(argv + 2), _T("%u"), &type);
    _stscanf(*(argv + 3), _T("%u"), &timer);

    /*
     * ドライバオブジェクトの作成
     */
    hSciCtl = CreateFile(
                    DRIVER_FILENAME,
                    GENERIC_READ | GENERIC_WRITE,
                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                    NULL,
                    OPEN_EXISTING,
                    0,
                    NULL
                    );
    if(hSciCtl == INVALID_HANDLE_VALUE){
        printf("CreateFile: NG¥n");
        return -1;
    }
```

```
/*
 *  シリアルコントロール情報を書き込む
 *
 *   conf.ch
 *       ポート番号  0: COM3
 *                   1: COM4
 *
 *   conf.type
 *       ポートタイプ  0: RS232C
 *                     1: RS422
 *                     2: RS485
 *
 *   conf.timer
 *       RS485 用 TX ディセーブルタイマ[マイクロ秒]
 *       送信完了から設定時間の間、再送信がなければ
 *       TX がディセーブルされます。
 */
conf.ch = ch;
conf.type = type;
conf.timer = timer;

ret = DeviceIoControl(
            hSciCtl,
            IOCTL_SCICTL_SETCONFIG,
            &conf,
            sizeof(SCICTL_CONF_PAR),
            NULL,
            0,
            &retlen,
            NULL
      );
if(!ret){
    errno = GetLastError();
    fprintf(stderr, "ioctl set IOCTL_SCICTLDRV_SETCONFIG error: %d¥n", errno);
    CloseHandle(hSciCtl);
    return -1;
}

/*
 * シリアルコントロール情報を読み出す
 */
ret = DeviceIoControl(
            hSciCtl,
            IOCTL_SCICTL_GETCONFIG,
            &conf,
            sizeof(SCICTL_CONF_PAR),
            &conf,
            sizeof(SCICTL_CONF_PAR),
            &retlen,
            NULL
      );
```

```
    if(!ret){
        errno = GetLastError();
        fprintf(stderr, "ioctl set IOCTL_SCICTLDRV_GETCONFIG error: %d¥n", errno);
        CloseHandle(hSciCtl);
        return -1;
    }
    printf("SerialControlConfig.exe ch=%u, type=%u, timer=%u¥n",
            conf.ch, conf.type, conf.timer);

    CloseHandle(hSciCtl);
    return 0;
}
```

第６章　システムリカバリ

Algo Smart Panel 用 Windows Embedded CE 6.0 は SD カードを使用して工場出荷状態に戻すことが出来ます。本章では、Algo Smart Panel 用 Windows Embedded CE 6.0 のシステムのリカバリ方法について説明します。

６－１　　リカバリＳＤの準備

Algo Smart Panel 用 Windows Embedded CE 6.0 では、リカバリ SD を作成するためにリカバリ SD のディスクドライブイメージ（dd イメージ）を用意しています。リカバリ SD の dd イメージは、「Algo Smart Panel AP-4410/5410/6410/6500/7500 Windows Embedded CE 6.0 リカバリ/SDK DVD」（以降、「リカバリ/SDK DVD」と表記します。）で提供しています。（表 6-1-1-1）

表 6-1-1-1. リカバリSDイメージ

dd イメージファイル	サイズ [Byte]
¥Recovery¥G3N_CE_Recovery_v*.**.ddi	513,277,952

※　「*.**」はバージョンをあらわします。例：1.00→Ver1.00

「リカバリ/SDK DVD」に提供されているリカバリ SD の dd イメージを DDforWindows(SiliconLinux 社)等 dd イメージ書込みツールで SD カードに書込みます。

リカバリ SD に使用する SD カードは、表 6-1-1-1 で示したイメージサイズ以上の容量が必要となります。

６－２　　リカバリＳＤの起動

「6-1 リカバリ SD の準備」で作成したリカバリ SD を使用してリカバリソフトを起動します。リカバリ SD で起動させるには、SD スロットにリカバリ SD を入れ、MODE スイッチを押しながら AP シリーズ本体の電源を入れます。

正常に起動すると図 6-2-1 のリカバリメイン画面が表示されます。

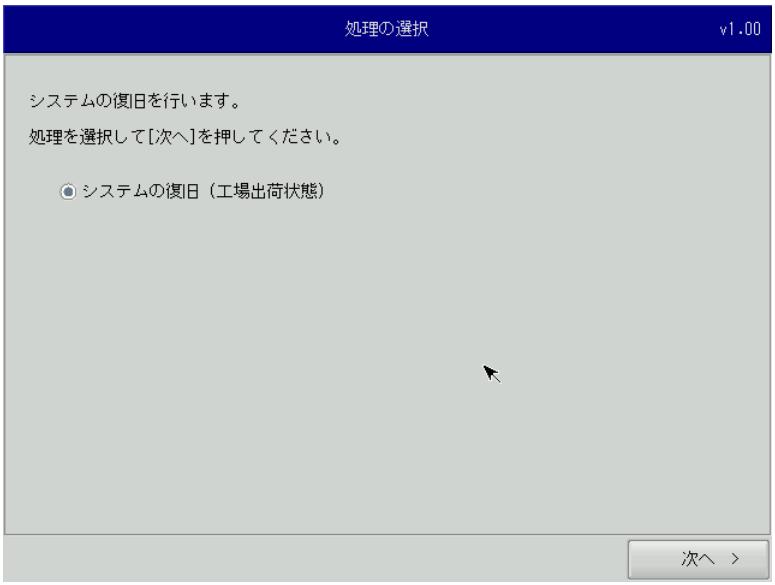

図 6-2-1. リカバリメイン画面

６－３　　システムの復旧

工場出荷イメージを NAND フラッシュメモリに書込むことで、システムを工場出荷状態に復旧することができます。

※　システムを工場出荷状態へ復旧すると NAND フラッシュメモリにあるデータはすべて消えてしまいます。必要なデータがある場合は、復旧作業を行う前に保存してください。

※　工場出荷状態へのシステム復旧には、ソフトウェア使用許諾契約に同意していただく必要があります。ソフトウェア使用許諾契約は、製品に同梱されています。システム復旧を行う場合は、内容を確認するようにしてください。

● システム復旧の手順

① リカバリメイン画面（図 6-2-1）で[次へ]ボタンを押します。
② ソフトウェア使用許諾契約確認画面（図 6-3-1）が表示されます。使用許諾契約を確認し、使用許諾契約の諸条件に同意できる場合は[次へ]ボタンを押します。

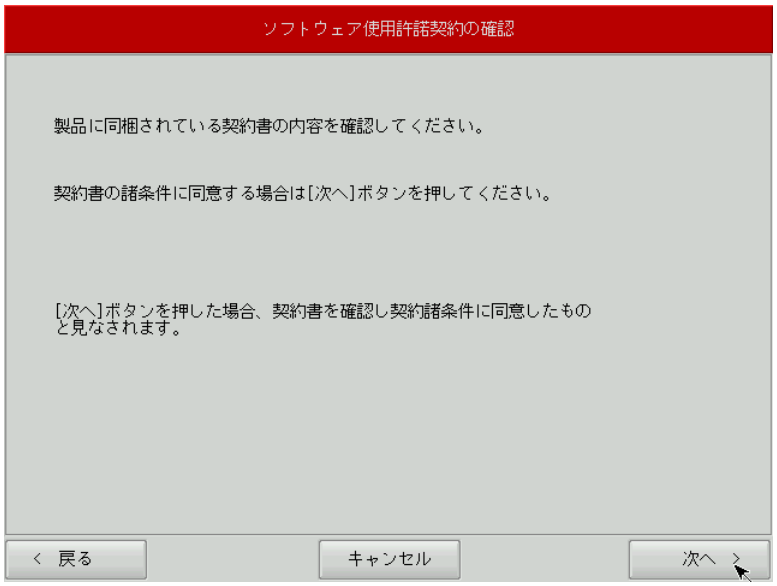

図 6-3-1. ソフトウェア使用許諾契約確認画面

③　確認画面（図 6-3-2）が表示されますので、[次へ]ボタンを押して処理を開始します。

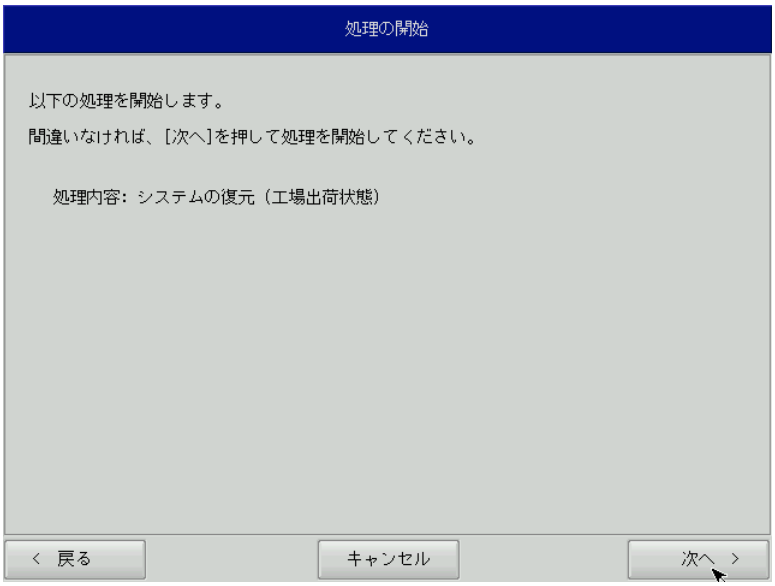

図 6-3-2. 確認画面

④　処理が始まります（図 6-3-3）。実行中は SD カードを外したり、電源を落としたりしないでください。

図 6-3-3. 実行中画面

⑤　終了画面（図 6-3-4）が表示されると復旧処理は完了です。電源を落として SD カードを外します。

図 6-3-4. 終了画面

⑥　電源を入れ起動を確認します。

付録

A－1　マイクロソフト製品の組込み用OS（Embedded）について

【OS の注意事項】

本製品に搭載している OS は組込み用（Embedded）であり、一般的なパソコンで使用される OS とは異なります。そのためソフトウェアや接続デバイスの動作が異なる（または動作しない、インストールできない）等の可能性がありますので、お客様にて十分な動作確認を実施して頂きますようお願いいたします。

【OS 種別】

本製品には以下のマイクロソフト製品の OS が存在します。各々下記の意味で使用しておりますので、ご認識ください。

○　Windows Embedded CE 6.0

：マイクロソフト社が組込み用に独自に開発したコンポーネント形式の OS

【OS 用のアプリケーション開発】

・　Windows Embedded CE6.0 は、AP 開発環境として Visual Studio 2005 と SDK が必要です。尚、SDK は本製品と一緒に提供されますが、Visual Studio 2005 は別途ご用意下さい。

・　クロス環境によるアプリケーション開発をお願いします。実機での開発はできません。

【I/O 機器の接続について】

本製品のインターフェースを介して周辺デバイスを接続する場合、組込み用（Embedded）OS では通常 OS とは機能が異なります。十分な動作確認を実施してください。

各種 I/O 機器の動作において動作不良が発生した場合には、機器提供メーカーへお問い合わせをお願いいたします。

【提案に際して】

・お客様への提案に際して、事前に装置の寿命年数と条件、保守条件（有寿命部品）等の諸条件の説明をお願いいたします。

・本装置は、医療機器･原子力設備や機器、航空宇宙機器・輸送設備など、人命に関わる設備や機器および高度な信頼性を必要とする設備や機器などへの組込み、また、これらの機器の制御などを目的とした使用は意図されておりません。これらの設備や機器、制御システムなどに本装置を使用した結果、人身事故、財産損害などが生じてもいかなる責任も負いかねます。

・本装置（ソフトウエアを含む）が、外国為替および外国貿易法の規定により、輸出規制品に該当する場合は、海外輸出の際に日本国政府の輸出許可申請等必要な手続きをお取り下さい。また、米国再輸出規制等外国政府の規制を受ける場合には、所定の手続きを行ってください

＜制約事項＞

- Embedded ライセンスは、組込機器や特定業務用機器の OS としてのみご使用いただけます。汎用目的（「市販のアプリケーションをエンドユーザがインストールして使用する」など）ではご使用いただけません。
- OS はお客様で開発した専用アプリケーションとあわせて利用しなければなりません。必ず専用アプリケーションをインストールしてご使用ください。
- 医療機器･原子力設備や機器、航空宇宙機器・輸送設備など、誤動作により被害が想定される装置への使用はできません。
- 製品構成に関する制限
  —Embedded ライセンス契約であることを示す「Certificate of Authenticity」ステッカーを装置本体に貼り付けて出荷します。
  —OS のインストール媒体は添付できません。復旧媒体（アプリケーション込み）の添付は可能です。
- 専用アプリケーションに関する制限
  —専用アプリケーションのユーザインタフェースからのみ、他のアプリケーションやファンクションにアクセスできるようにしなければなりません。
  —Microsoft のユーザインタフェース（ロゴ、ブートアップ、デスクトップ画面、フォルダ、ツールバーなど）を一切表示してはいけません。
- 組込みシステムの再販売･再頒布に際しマイクロソフト社の OS 製品の COA と APM を各システムに必ず貼付・添付しなければなりません。
- 組込み型システムとは別にマイクロソフト社の OS 製品またはその製品の一部を宣伝したり､価格提示したり、あるいは販売したり再頒布したりしてはなりません。
- ここに定めた条件を守っていないことが判明した場合、株式会社アルゴシステムはマイクロソフト株式会社との契約に従って状況報告をマイクロソフト株式会社に行い、出荷停止・状況改善依頼・調査を行うことができます。

＜用語の説明＞

- 「APM」とは「関連製品資料」のことで、使用許諾製品の再配布可能コンポーネントとしてマイクロソフト社が適宜指定する、使用許諾製品に関連するドキュメント、ソフトウェアや他の有形資料を含んだ外部メデイアを意味します。なお、APM に COA は含まれません。
- 「COA」すなわち「Certificate of Authenticity」は、マイクロソフト社が使用許諾製品のみに作成した削除不可のステッカーを意味します。
- 「組込み型アプリケーション」とは、業界または業務固有のソフトウエアプログラムを意味し、以下の属性をすべて備えているものです。
  （１） 組込み型システムの主要な機能がある。
  （２） 組込み型システムが販売される業界に特有な機能要求に合わせて設計されている。
  （３） 使用許諾製品ソフトウェアに加えて重要な機能がある。
- 「組込み型システム」とは、組込み型アプリケーション用に設計され、組込み型アプリケーションと共に配布され且つ､汎用的なパーソナルコンピューテイングデバイス（パーソナルコンピュータなど）または多機能サーバとして販売もしくは使用されません。また、これらシステムの代替品として商業的に実現不可能な、お客様のイメージ付きコンピューテイングシステムまたはデバイスを意味します。

A－2 参考文献

● Microsoft Developer Network Library (MSDN Library)
マイクロソフト社の開発者向け WEB サイト
Windows Embedded CE 関連情報 URL: http://msdn.microsoft.com/en-us/library/bb847932.aspx

●「はじめての Windows Embbeded CE6」
著者 大川 善邦
発行所 株式会社 工学社
発行年 2007 年

●「Windows EmbeddedCE 6.0 組み込み OS 構築技術法入門」
監修 松岡 正人
著者 伊藤 優、岩崎 平、江島 午郎、大場 孝仁、Kasar Mahesh、杉本 拓也、
高根 英哉、田靡 哲也、中山 宏之、松井 俊訓、好井 智章
発行所 日経 BP ソフトプレス
発行年 2008 年

このマニュアルについて

(1)本書の内容の一部又は全部を当社からの事前の承諾を得ることなく、無断で複写、複製、掲載することは固くお断りします。
(2)本書の内容に関しては、製品改良のためお断りなく、仕様などを変更することがありますのでご了承下さい。
(3)本書の内容に関しては万全を期しておりますが、万一ご不審な点や誤りなどお気付きのことがございましたらお手数ですが巻末記載の弊社までご連絡下さい。その際、巻末記載の書籍番号も併せてお知らせ下さい。

７７W０１０００３Ａ　　２００９年　７月　初版

株式会社アルゴシステム

本社
〒５８７－００２１　大阪府堺市美原区小平尾６５６番地　　ＴＥＬ（０７２）３６２－５０６７
ＦＡＸ（０７２）３６２－４８５６

東京支社
〒１０４－００６１　東京都中央区銀座７－１５－８　銀座堀ビル２Ｆ　　ＴＥＬ（０３）３５４１－７１７０
ＦＡＸ（０３）３５４１－７１７５

大阪支社
〒５４２－００８１　大阪府大阪市中央区南船場１－１２－３　船場グランドビル３Ｆ　　ＴＥＬ（０６）６２６３－９５７５
ＦＡＸ（０６）６２６３－９５７６

名古屋営業所
〒４６１－０００４　愛知県名古屋市東区葵２－３－１５　ふぁみーゆ葵ビル５０３　　ＴＥＬ（０５２）９３９－５３３３
ＦＡＸ（０５２）９３９－５３３０

ホームページ　http://www.algosystem.co.jp