

マニュアル

Algo Smart Panel用

『Windows Embedded CE 6.0』

について

AP-3300

# 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	3-3
3-2-5	USBポート	3-3
3-2-6	RAMディスク	3-4
3-3	組み込みシステム機能	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	バックアップSRAM	3-5

## 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 ウォーカー	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 組み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	汎用入力ドライバについて	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

## 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。組み込み用 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み込み用 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が必要とします。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-3300 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、より最適な組み込み機器 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
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

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



	<p>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</p> <p>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</p>
セキュリティ	<p>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)</p> <p>資格情報マネージャ                  認証サービス (SSPI)                  NTLM                  Schannel (SSL/TLS)</p>
ファイルシステムおよびデータストア	<p>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</p>
フォント	<p>Courier New (サブセット 1_30)                  Tahoma (サブセット 1_07)                  Wingding</p>
通信サービスおよびネットワーク	<p>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)</p> <p>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)</p> <p>ネットワーク - 広域ネットワーク (WAN)                  Telephony API (TAPI 2.0)                  ダイアルアップ・ネットワーク (RAS/PPP)                  自動ダイヤル</p>

	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 1 ポートドライバ
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 * <b>USB-HUB には非対応です。</b>
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ccess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ccess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。

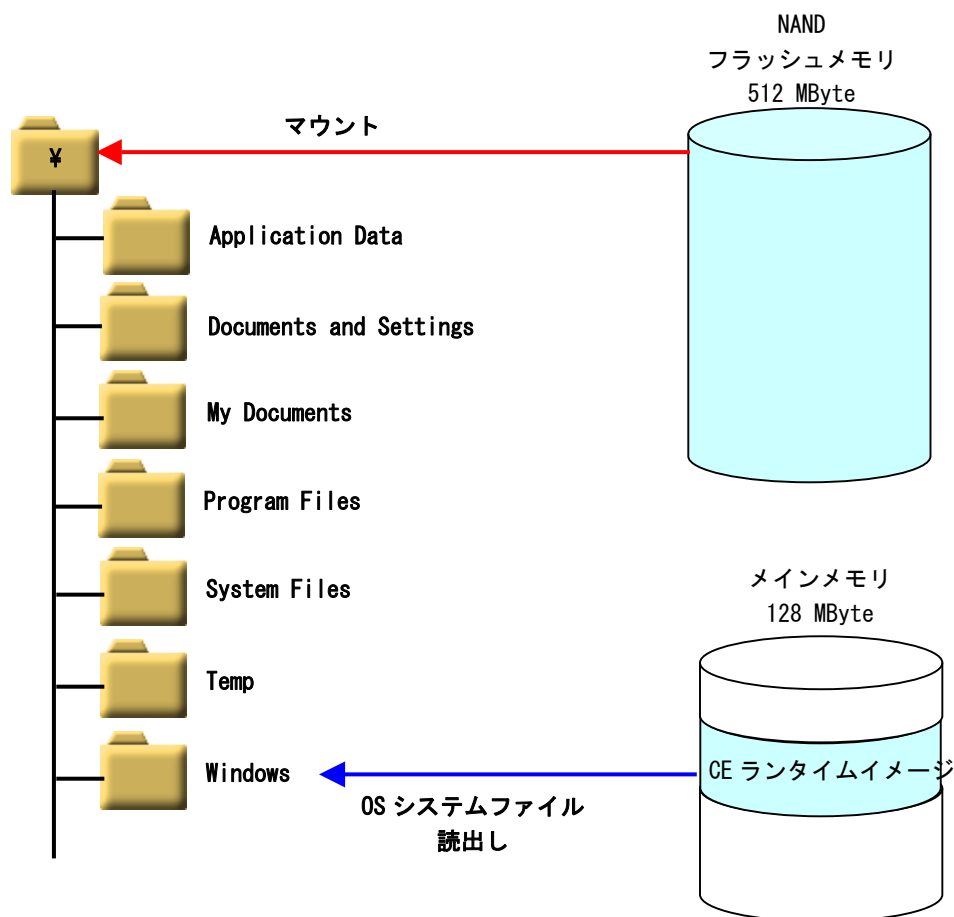


図 1-2-1-1. ファイルシステム構成

## 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。

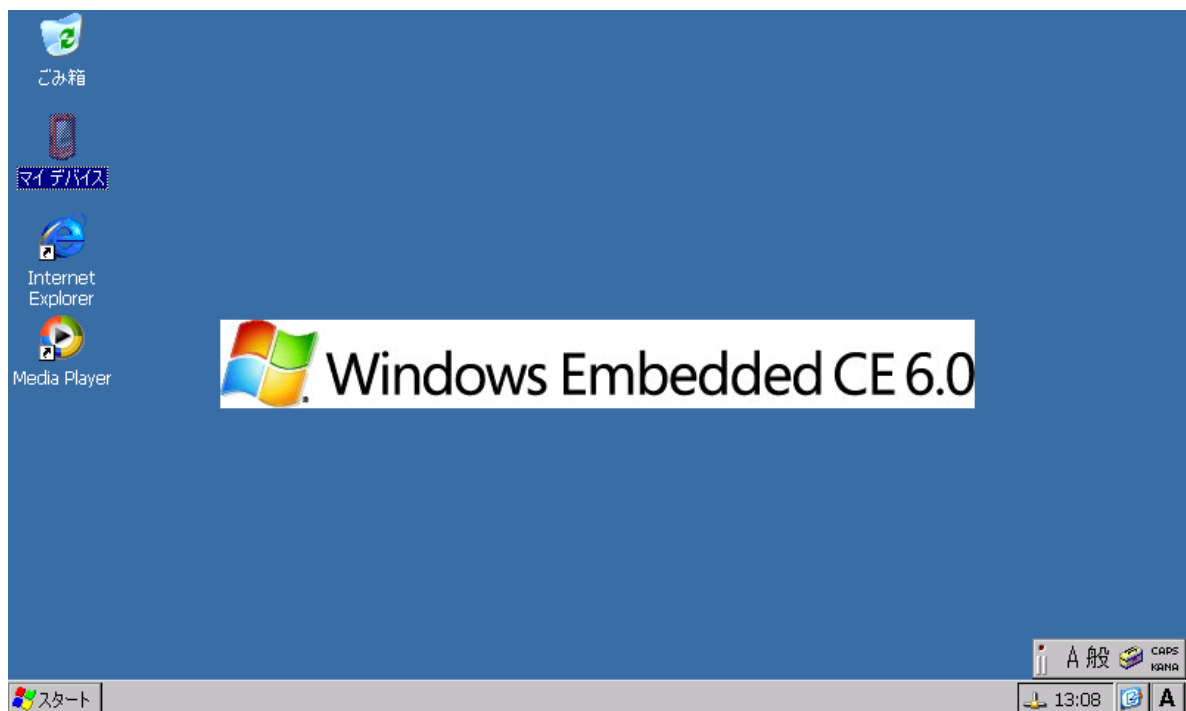


図 2-1-1-1. デスクトップ

#### 2-1-2 OSの終了

ファイルシステムへのファイル書き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。

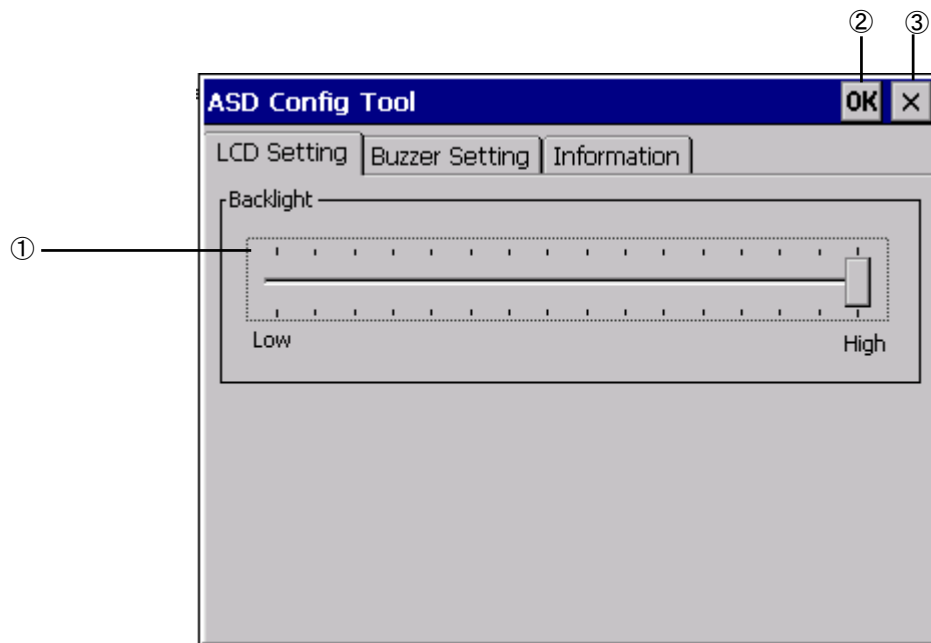


図 2-2-2-1. ASD Config – LCD Setting

- 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。

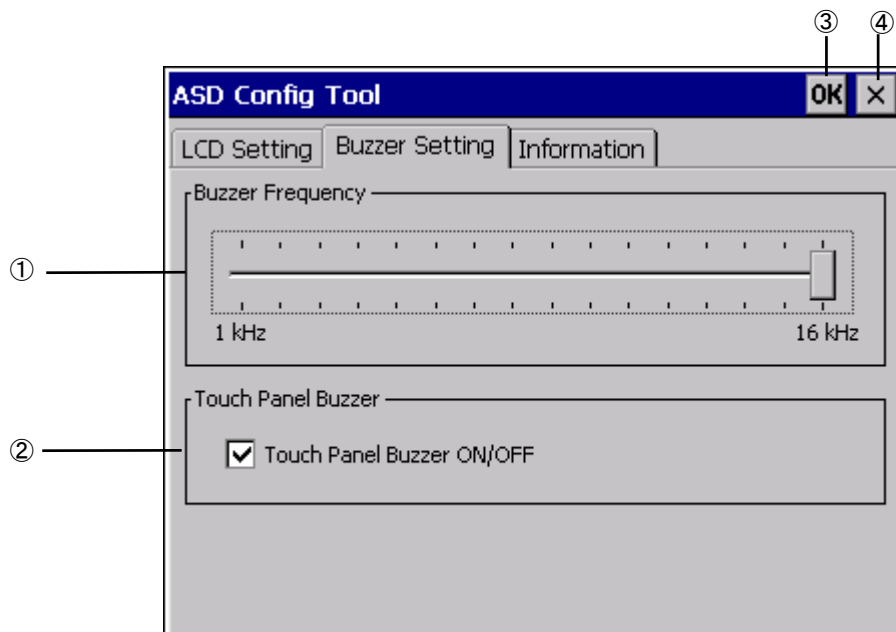


図 2-2-3-1. ASD Config – Buzzer Setting

- 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。

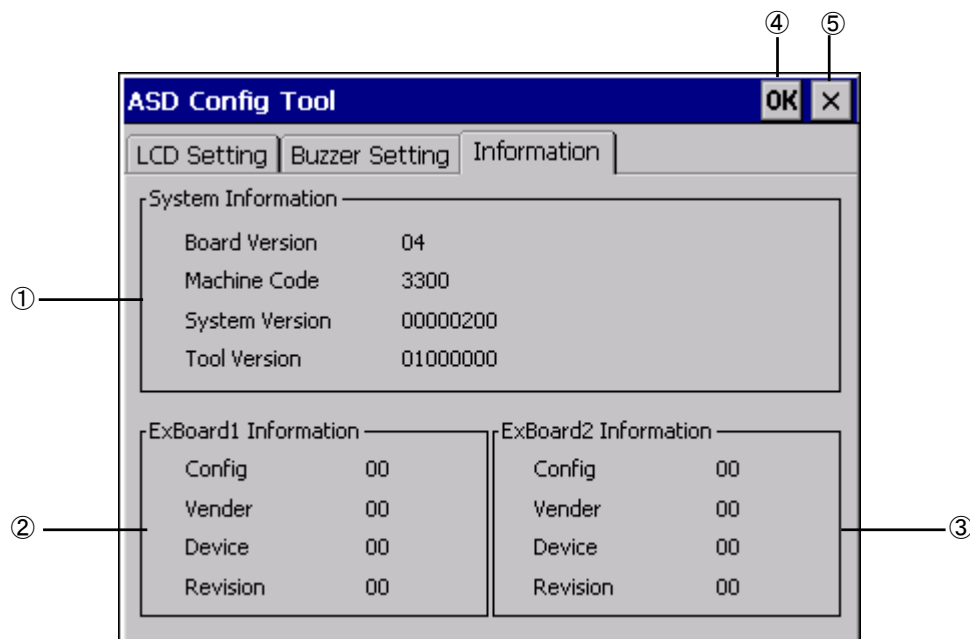


図 2-2-4-1. ASD Config – Board Information

- 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てください。
	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てください。
	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てください。 ¥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
} ①, ②, ③

[START02]
START=¥SRAM Disk¥ProgB.exe
PARAM=100
WAIT=-1
} ④, ⑤

[START03]
START=¥SRAM Disk¥ProgC.exe
WAIT=0
} 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
	DirBrowse	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。

本体の外形図を図 3-1-1 に示します。各部名称を表 3-1-1 に示します。

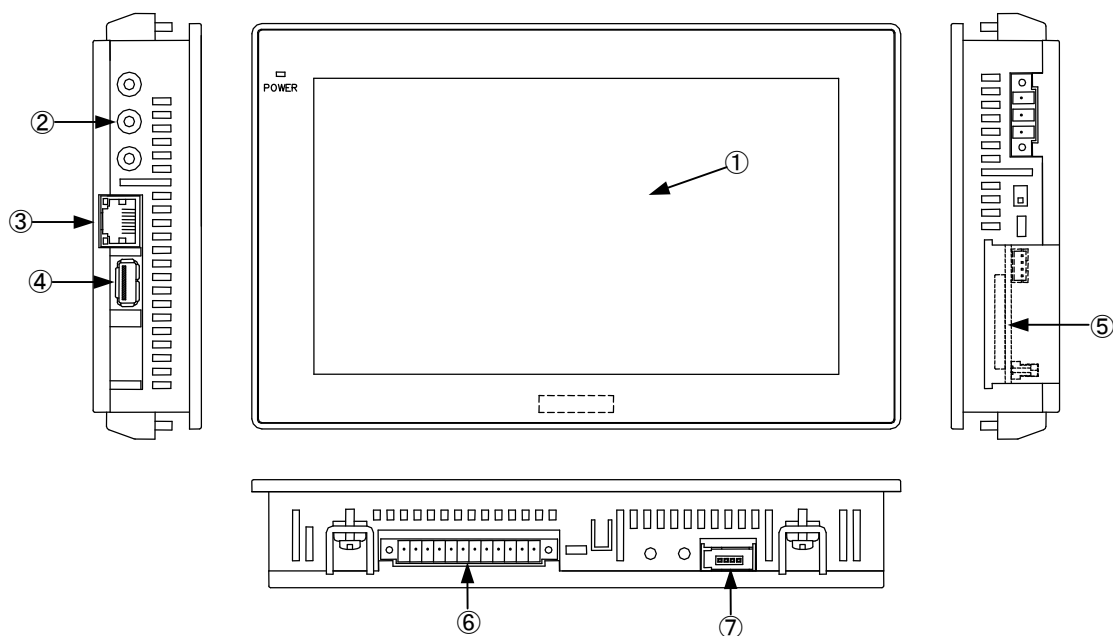


図 3-1-1. 外形図

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	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。
③	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。 LAN: SMC9118
④	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。 <b>※ USB-HUB には対応していません。</b>
⑤	SD カード インターフェース	SD/SDHC カード	Windows Embedded CE 6.0では使用できません。
⑥	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。
⑦	シリアル インターフェース	シリアルポート	シリアル通信が行えます。 SIO: COM1

## 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（図 3-1-1 ①）。  
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のため、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	SI0 (図 3-1-1 ⑦)

### 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
SMSC9118	LAN (図 3-1-1 ③)

### 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 1 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。USB-HUB には対応していません。

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 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3-2-6-1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 3-2-6-1. 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 ⑥)
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。
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 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。

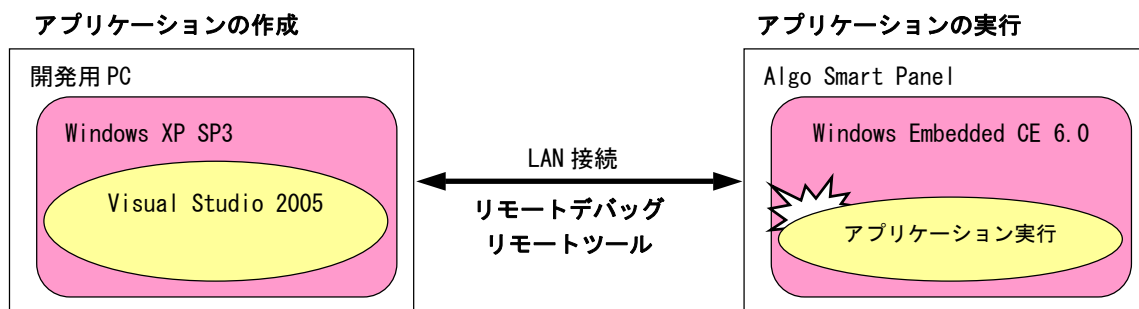


図 4-1-1. アプリケーション開発

## 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 は以下のシステム要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 が手元がない場合は、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-3300 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¥G2MP_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。

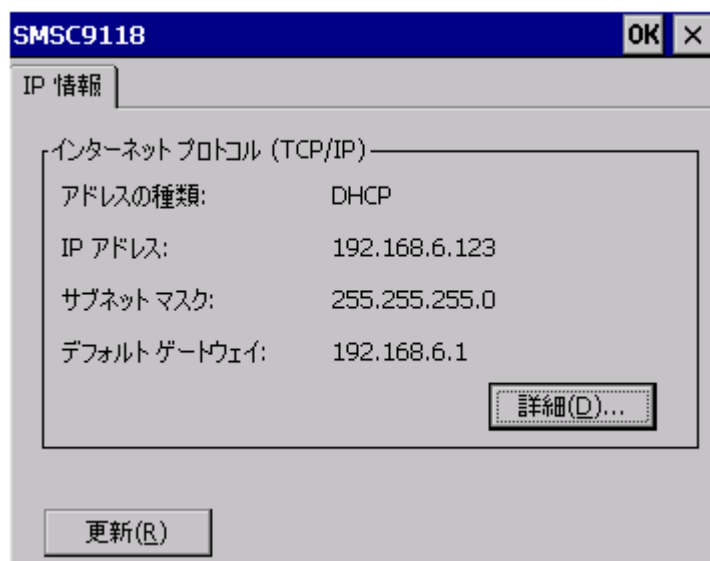


図 4-3-1-2. IP情報例

### 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 G2MP SDK Release]を選択します。(図 4-3-2-1)

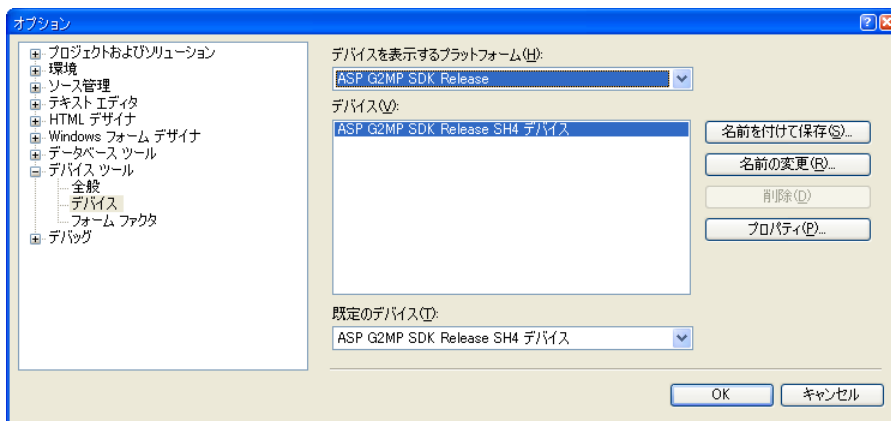


図 4-3-2-1. オプションダイアログ

- ③ [プロパティ]を押すと、[ASP G2MP 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)

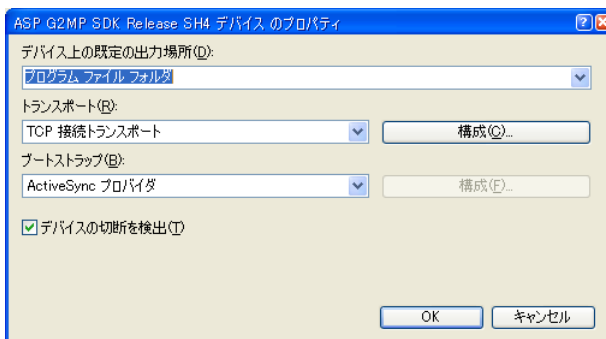


図 4-3-2-2. デバイスのプロパティ

- 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。

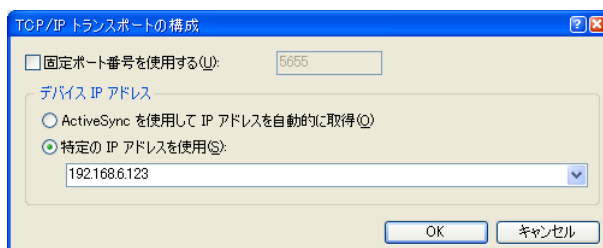


図 4-3-2-3. TCP/IP トランスポートの構成

- 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。
- Clientsshutdown.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)

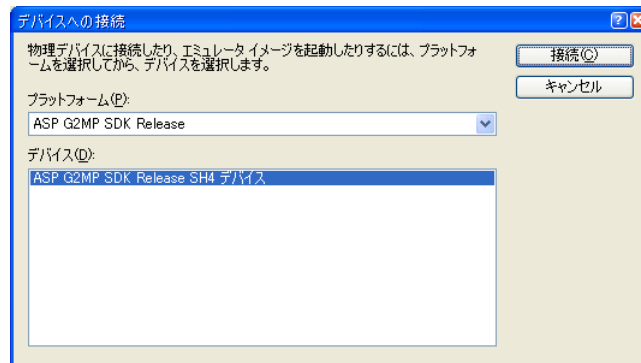


図 4-3-4-1. デバイスへの接続

- ④ [プラットフォーム] から [ASP G2MP 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 のようなダイアログが表示されます。

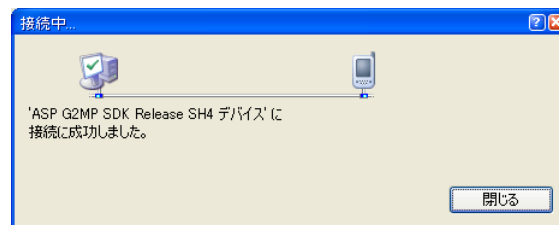


図 4-3-4-2. 接続成功



## 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-3300 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。

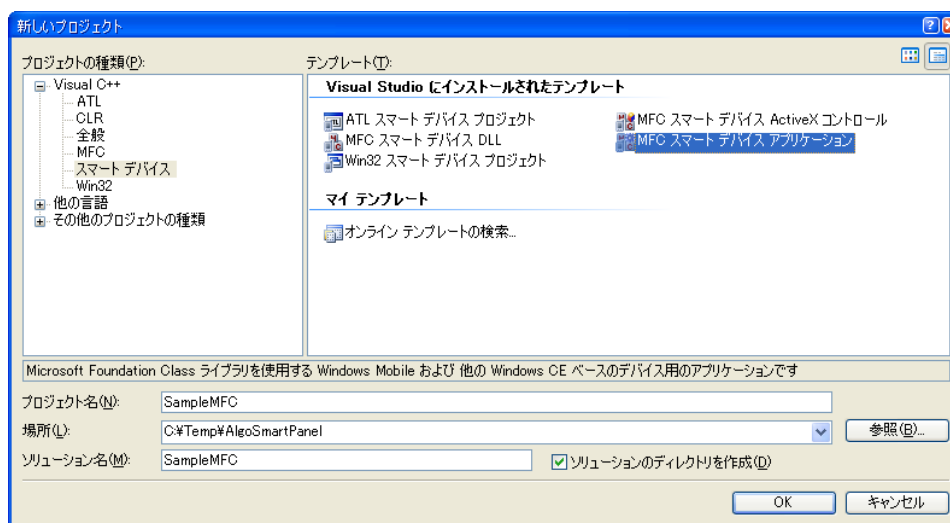


図 4-4-1-1. 新しいプロジェクト

- 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。

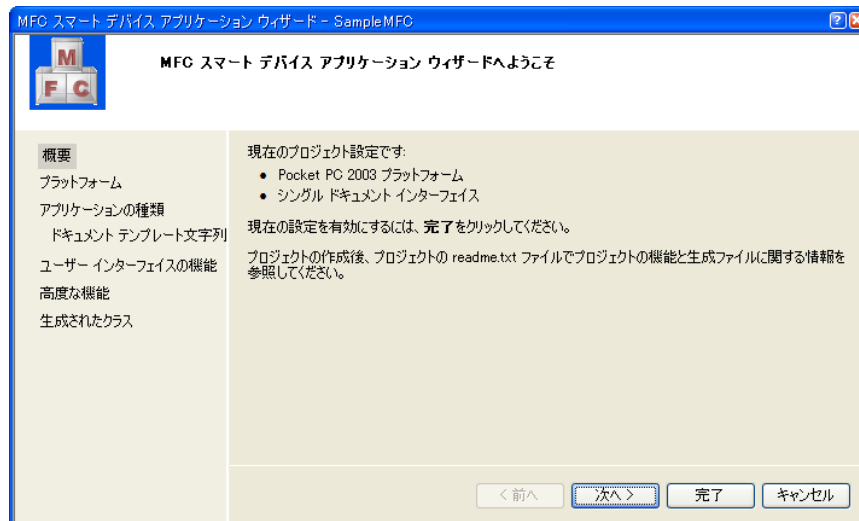


図 4-4-1-2. アプリケーションウィザード

- ⑥ [選択された SDK]に[ASP G2MP 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 G2MP SDK Release]のみを選択します。(図 4-4-1-3)  
[次へ]を押します。

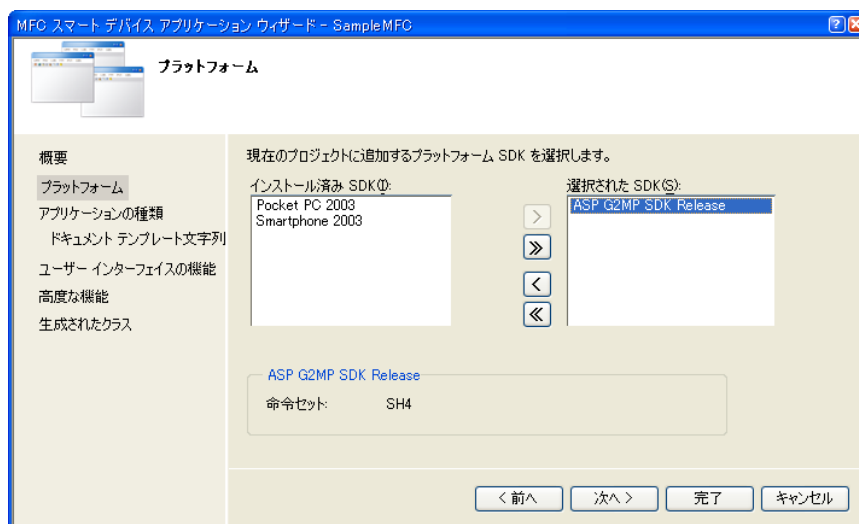


図 4-4-1-3. SDKの選択

- 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)

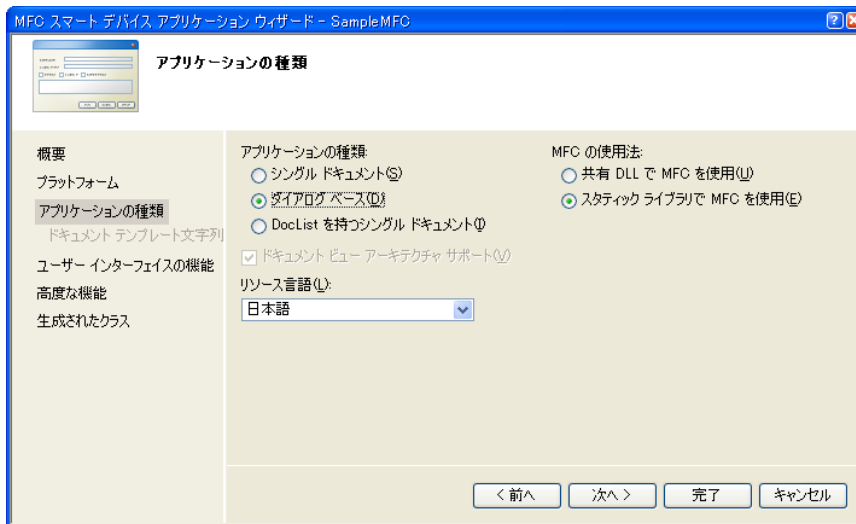


図 4-4-1-4. ダイアログベースアプリケーションの選択

- 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)

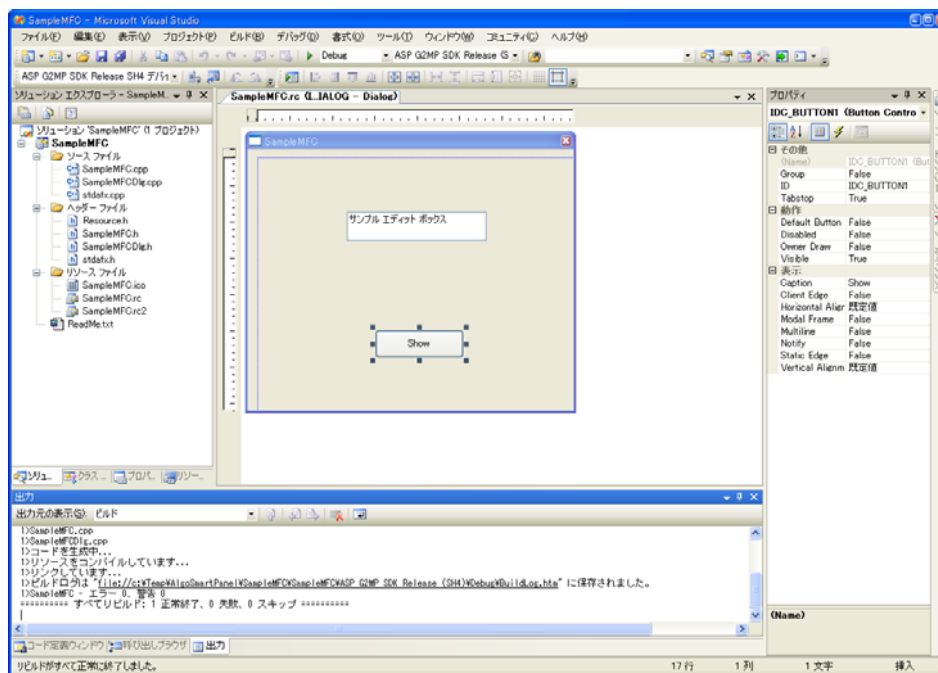


図 4-4-1-5. アプリケーションの作成

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、下記のよう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\_DEPRECATED」を追加してください。

```
1>C:\Program Files\Microsoft Visual Studio 8\VC\ce\atl\mf\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_G2MP_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_DEPRECATED. 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 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)

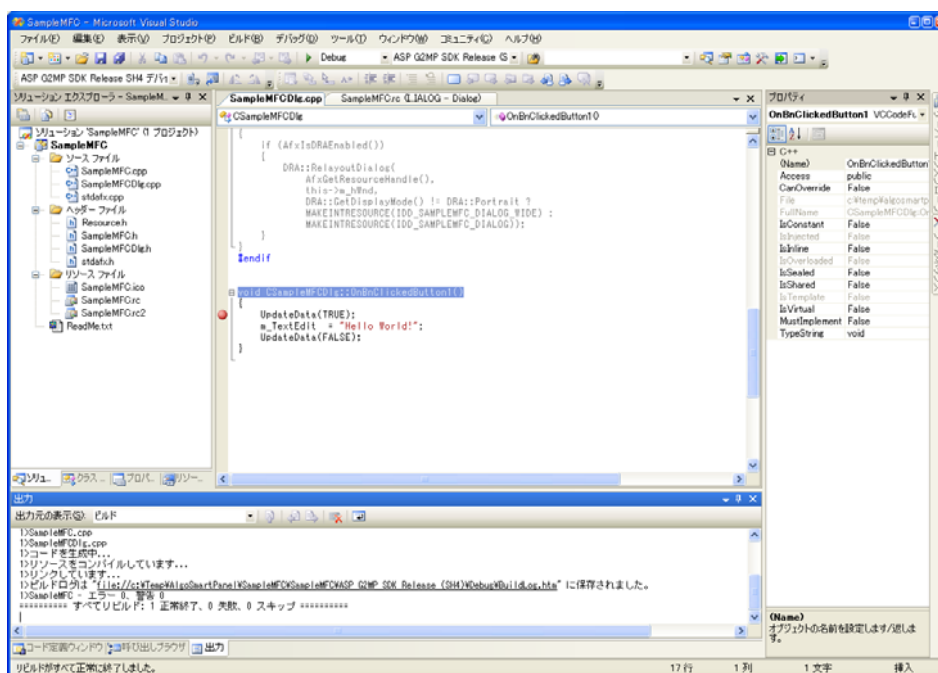


図 4-4-2-1. デバッグ準備

- 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。

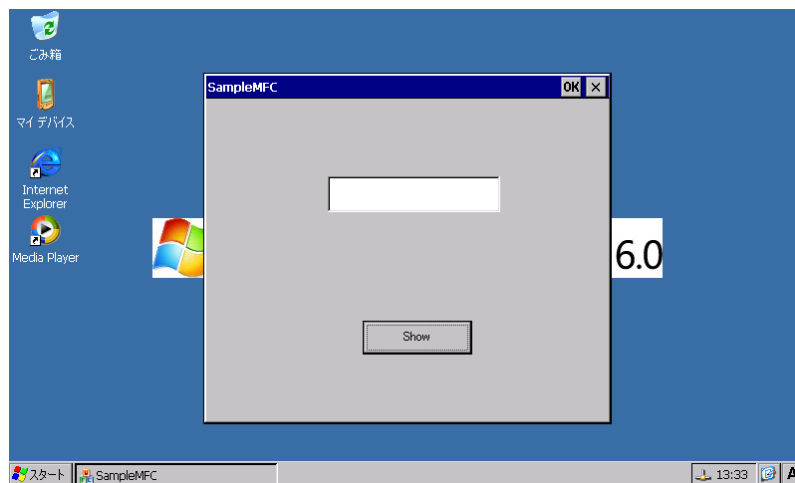


図 4-4-2-2. アプリケーションの起動

- 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)

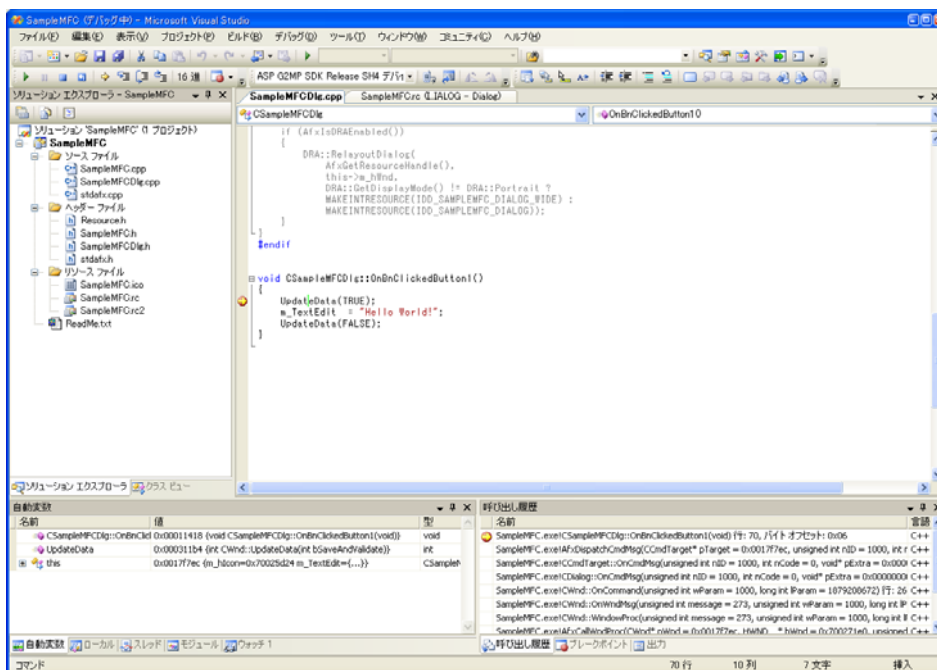


図 4-4-2-3. デバッグの開始

## 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-3300 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。

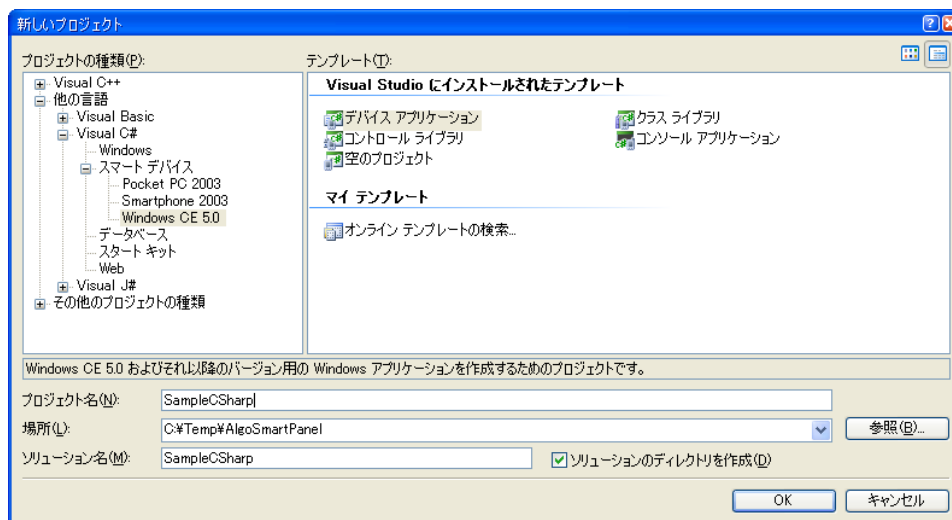


図 4-5-1-1. 新しいプロジェクト

- 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)

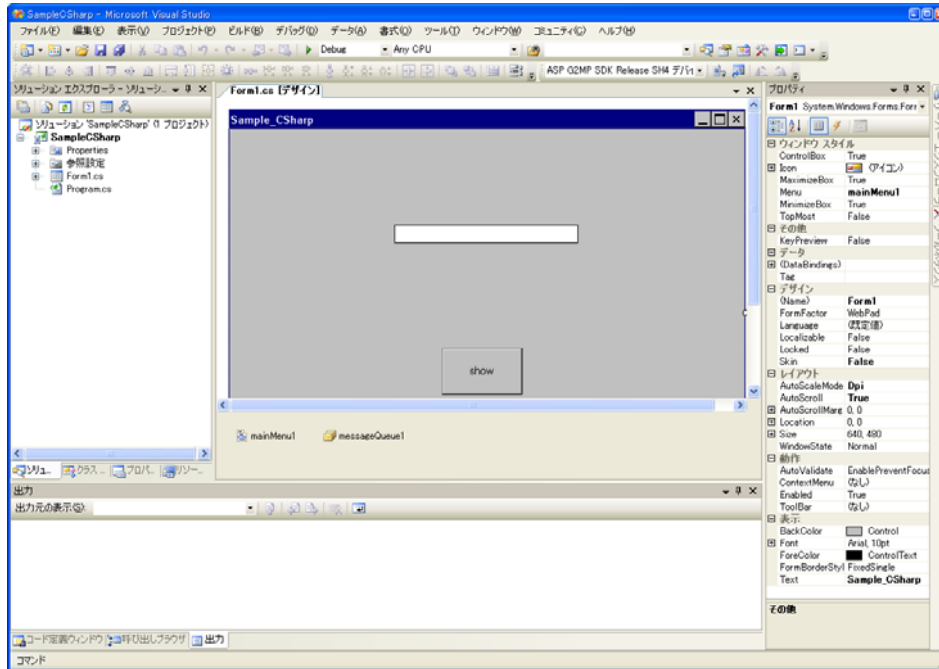


図 4-5-1-2. アプリケーションの作成



#### 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 G2MP SDK Release SH4 デバイス]を選択します。  
(図 4-5-2-1)

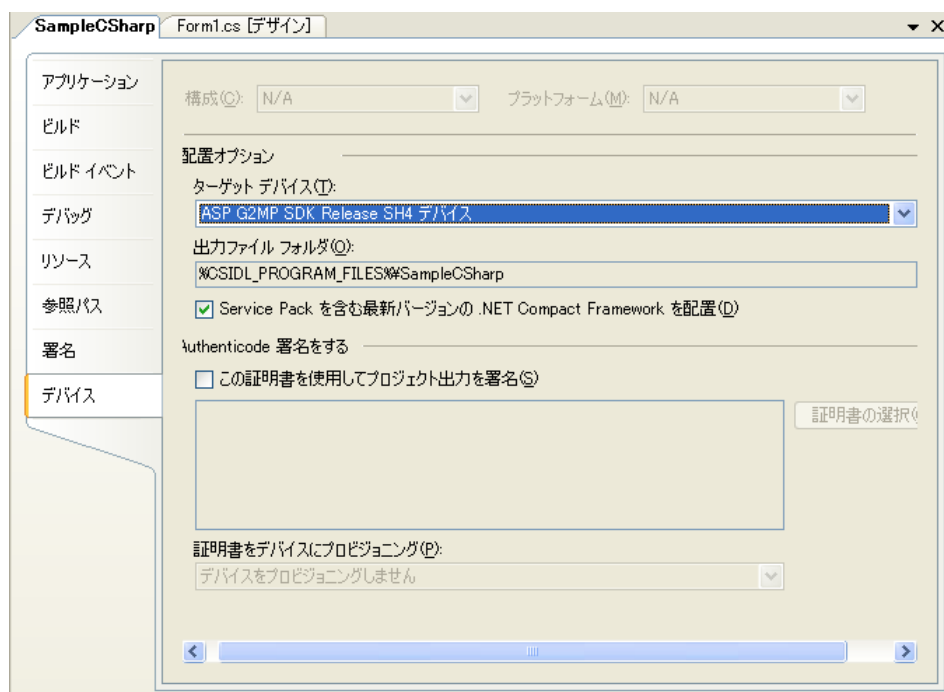


図 4-5-2-1. ターゲットデバイスの設定

- 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)

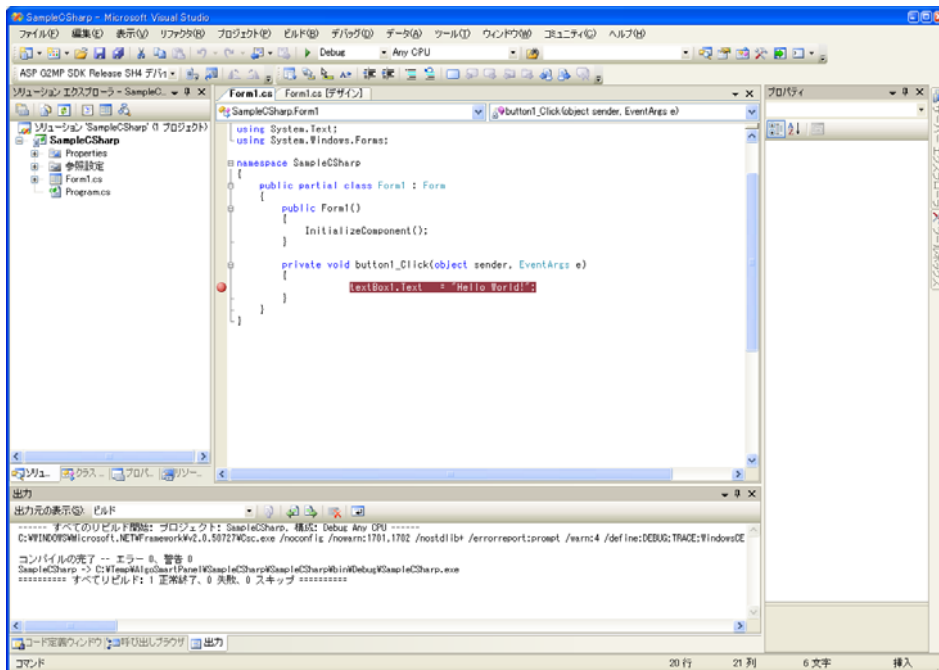


図 4-5-2-2. デバッグの準備

- 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 G2MP 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。

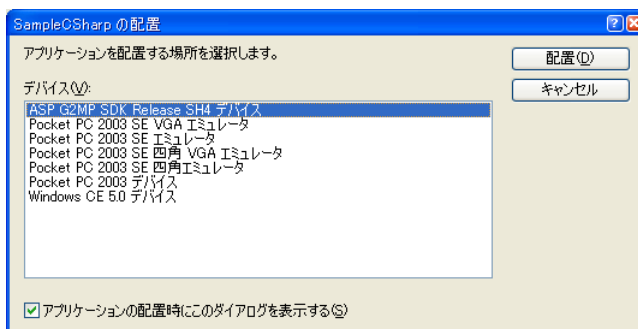


図 4-5-2-3. 転送先デバイスの指定

- 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)

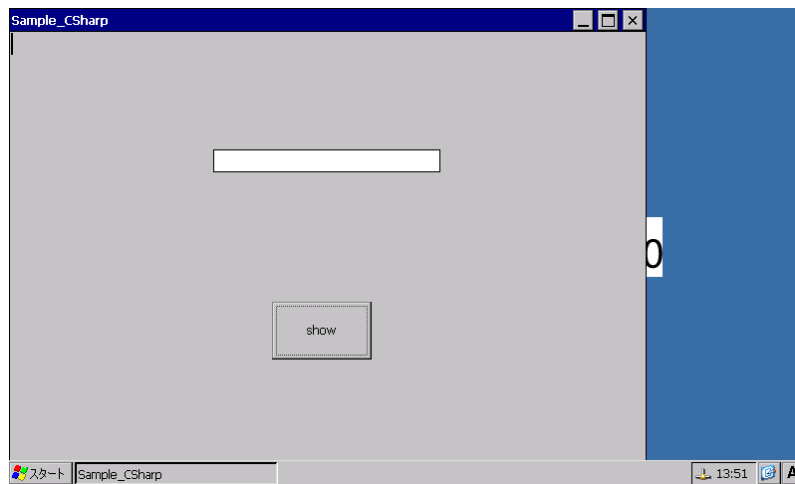


図 4-5-2-4. アプリケーションの起動

- 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)

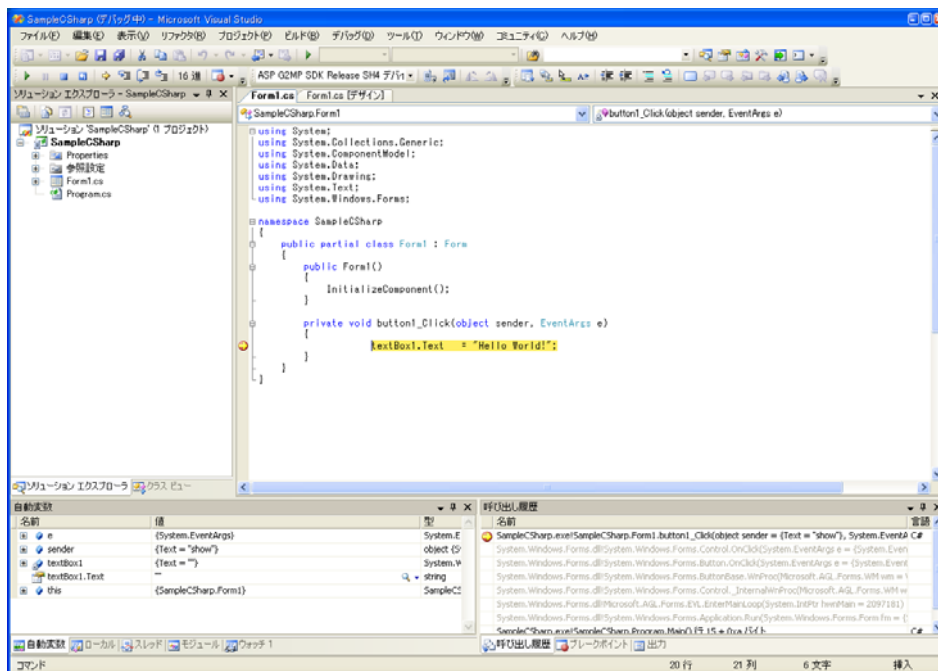


図 4-5-2-5. デバッグの開始

## 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 G2MP 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。

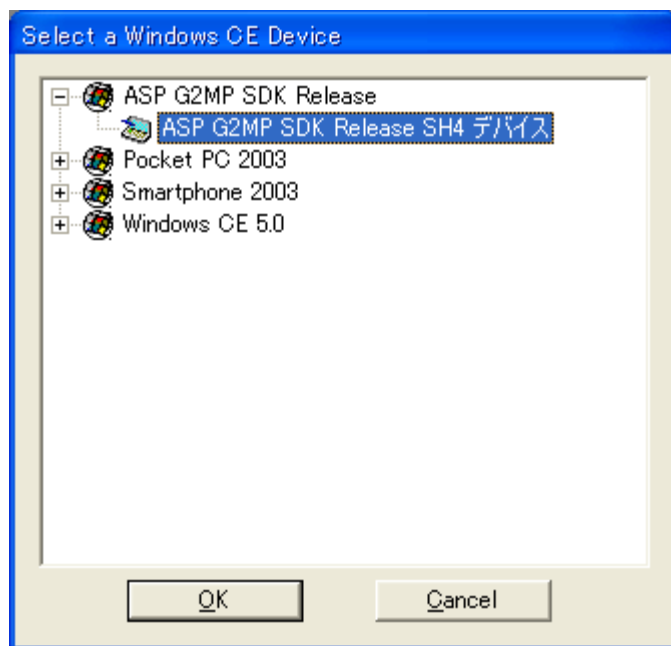


図 4-6-1-1. CEデバイスの選択



4-6-4 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)

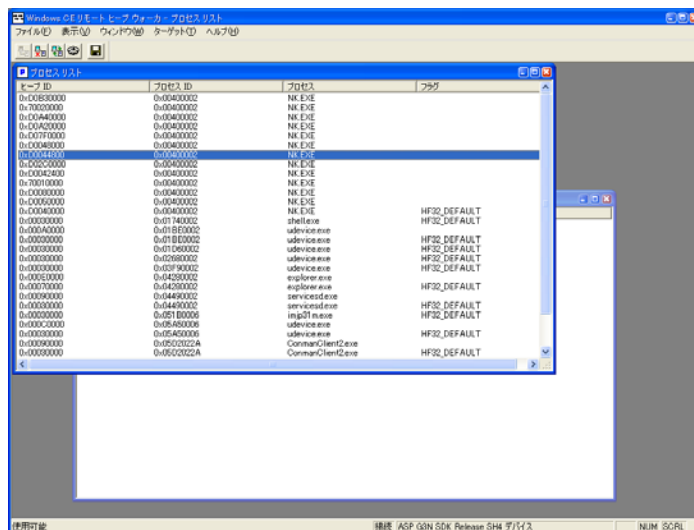


図 4-6-4-1. リモート ヒープ ウォーカ

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)

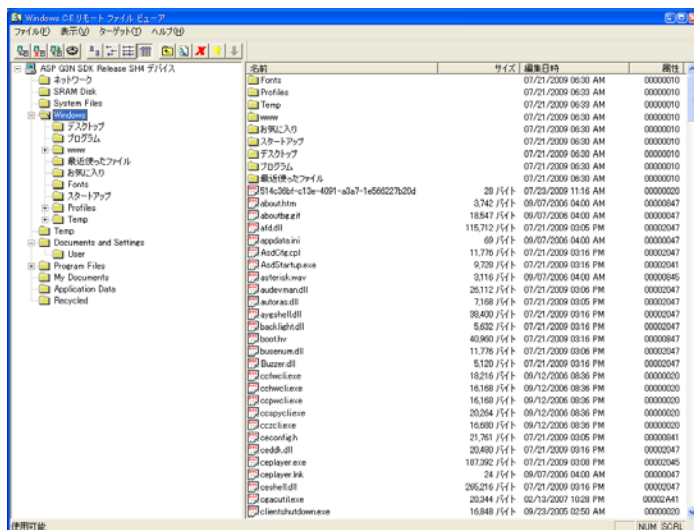


図 4-6-5-1. リモート ファイル ビューア

#### 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)

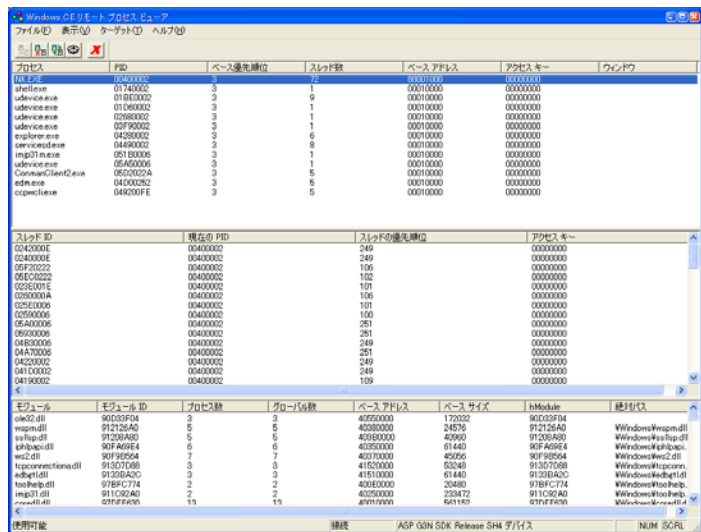


図 4-6-6-1. リモート プロセス ビューア

#### 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)

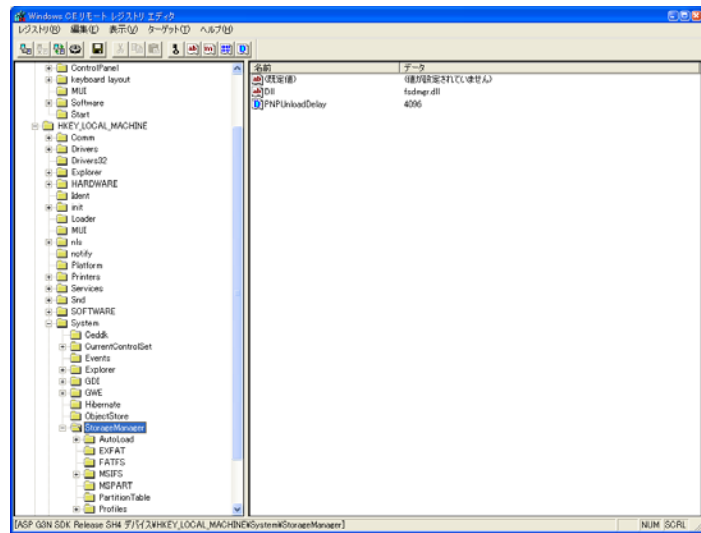


図 4-6-7-1. リモート レジストリ エディタ

## 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-3300 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	INO リセット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。

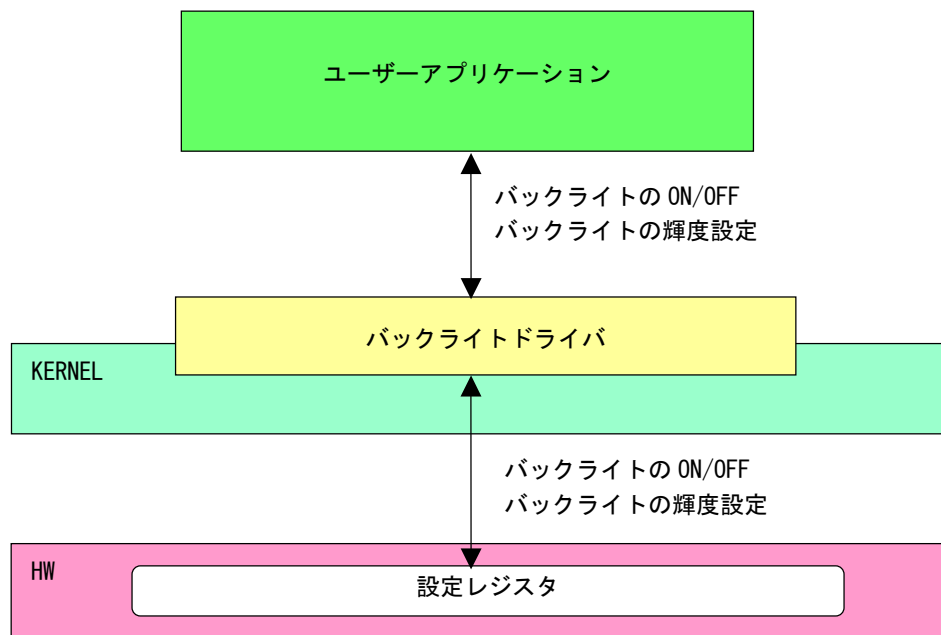


図 5-2-2-1. バックライトドライバ

## 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	<p>[KEY] HKEY_LOCAL_MACHINE¥Drivers¥BuiltIn¥Backlight</p> <p>[VALUE:DWORD] Brightness</p> <p>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)</p>
CreateFile	<p>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。</p> <pre>hBacklight = CreateFile(     _T("BKL1:"),     GENERIC_READ   GENERIC_WRITE,     FILE_SHARE_READ   FILE_SHARE_WRITE,     NULL,     OPEN_EXISTING,     0,     NULL );</pre>
CloseHandle	<p>デバイスハンドルをクローズします。</p> <pre>CloseHandle(hBacklight);</pre>
DeviceIoControl	<ul style="list-style-type: none"> <li>● IOCTL_BKL_SETSTATE バックライトのON/OFFを設定します。</li> <li>● IOCTL_BKL_GETSTATE バックライトのON/OFFを取得します。</li> <li>● IOCTL_BKL_SETBRIGHTNESS バックライトの輝度を設定します。</li> <li>● IOCTL_BKL_GETBRIGHTNESS バックライトの輝度を取得します。</li> <li>● IOCTL_BKL_GETMAXBRIGHTNESS バックライトの輝度の最大値を取得します。</li> <li>● 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。</li> <li>● IOCTL_BKL_LOADREG バックライトの輝度設定をレジストリから読み込みます。</li> </ul>

## 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** : 実際の実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** : 実際の実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** : 実際の実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** : 実際の実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** : 実際の実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` : 実際の実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み込みます。

**パラメータ**

`lpInBuf` : NULL  
`nInBufSize` : 0  
`lpOutBuf` : NULL  
`nOutBufSize` : 0  
`lpBytesReturned` : 実際の実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み込み、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_g2mp¥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_g2mp¥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。

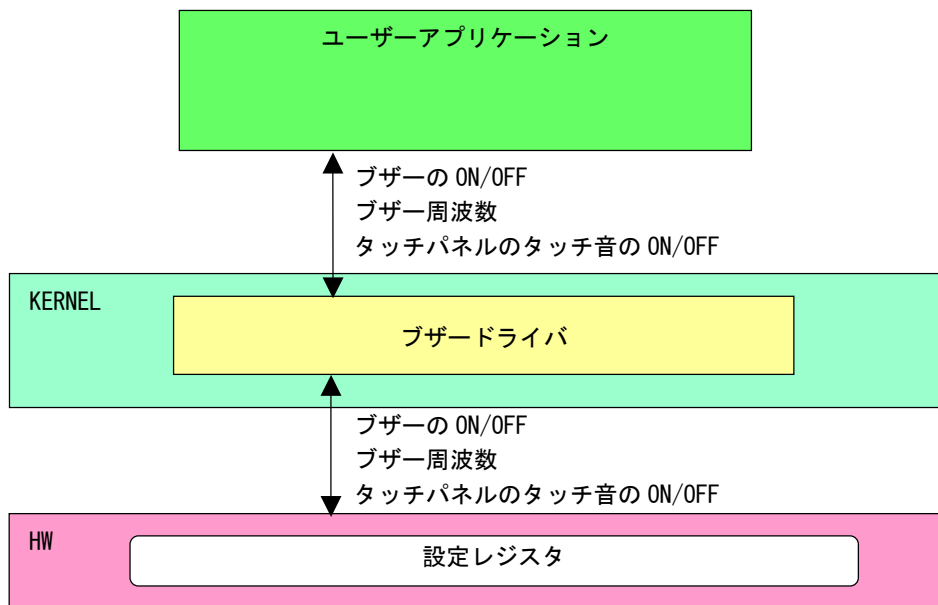


図 5-3-2-1. ブザードライバ



## 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进行操作します。

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	<p>[KEY] HKEY_LOCAL_MACHINE¥Drivers¥BuiltIn¥Buzzer</p> <p>[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)</p> <p>[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)</p>
CreateFile	<p>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CreateFileはエラーとなります。</p> <pre> hBuzz = CreateFile(     _T("BUZ1:"),     GENERIC_READ   GENERIC_WRITE,     FILE_SHARE_READ   FILE_SHARE_WRITE,     NULL,     OPEN_EXISTING,     0,     NULL ); </pre>
CloseHandle	<p>デバイスハンドルをクローズします。</p> <pre>CloseHandle(hBuzz);</pre>

**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み込みます。
- **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** : 実際の実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** : 実際の実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** : 実際の実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** : 実際の実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` : 実際の実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み込みます。

**パラメータ**

`lpInBuf` : NULL  
`nInBufSize` : 0  
`lpOutBuf` : NULL  
`nOutBufSize` : 0  
`lpBytesReturned` : 実際の実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み込み、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 : 実際の実出力バイト数を受け取る変数へのポインタ  
 lpOver lapped : 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。

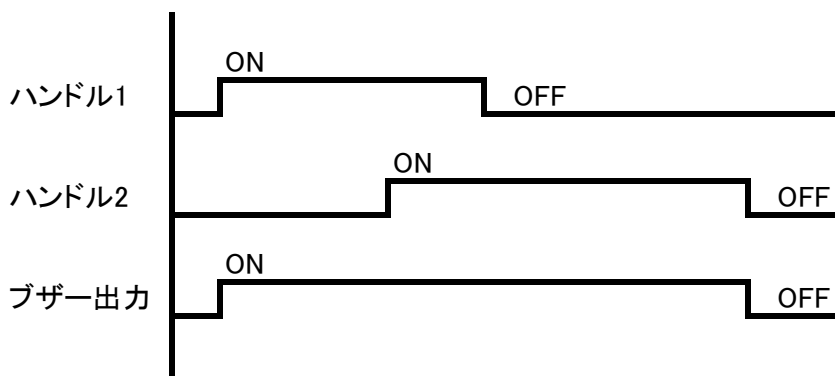


図 5-3-4-1. タイミングチャート

---

**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** : 実際の実出力バイト数を受け取る変数へのポインタ  
**lpOver lapped** : 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** : 実際の実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_g2mp¥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_g2mp¥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_g2mp¥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。

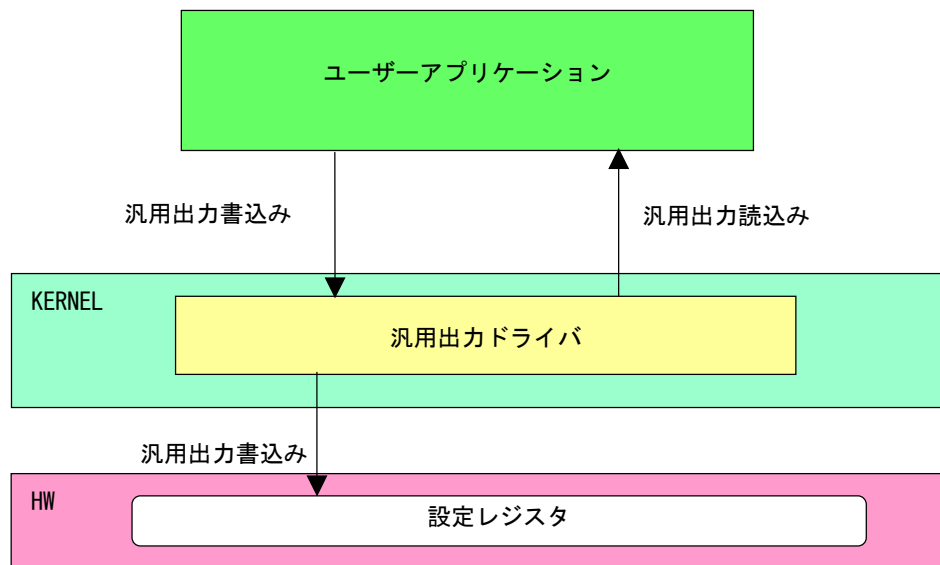


図 5-4-2-1. 汎用出力ドライバ

## 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	GN01:
説明	汎用出力の制御を行うことができます。
CreateFile	<p>デバイスファイル(GN01:)をオープンし、デバイスハンドルを取得します。</p> <pre> hGenOUT = CreateFile(     _T("GN01:"),     GENERIC_READ   GENERIC_WRITE,     FILE_SHARE_READ   FILE_SHARE_WRITE,     NULL,     OPEN_EXISTING,     0,     NULL ); </pre>
CloseHandle	<p>デバイスハンドルをクローズします。</p> <pre>CloseHandle(hGenOUT);</pre>
ReadFile	<p>出力データを読み込みます。 読み込みサイズは1Byteまでです。1Byte以上の場合はエラーとなります。</p> <pre>res = ReadFile(hGenOUT, &amp;OutData, 1, NULL);</pre>
WriteFile	<p>出力データを書込みます。 書き込みサイズは1Byteまでです。1Byte以上の場合はエラーとなります。</p> <pre>res = WriteFile(hGenOUT, &amp;OutData, 1, NULL);</pre>

5-4-4 汎用出力リファレンス

ReadFile

**機能**

出力データを読みみます。

**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**

出力データを読みみます。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 の読みみしか出来ません。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 汎用入力ドライバについて

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。

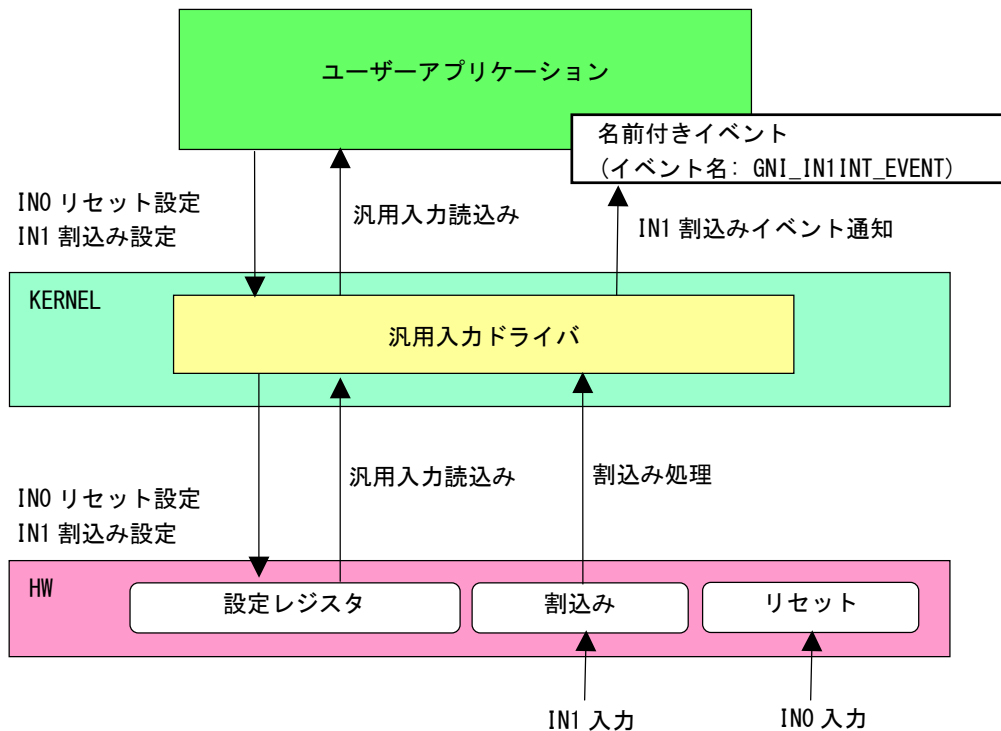


図 5-4-5-1. 汎用入力ドライバ

## 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	汎用入力の読み込み、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	<p>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。</p> <pre> hGenIN = CreateFile(     _T("GNI1:"),     GENERIC_READ   GENERIC_WRITE,     FILE_SHARE_READ   FILE_SHARE_WRITE,     NULL,     OPEN_EXISTING,     0,     NULL ); </pre>
CloseHandle	<p>デバイスハンドルをクローズします。</p> <pre>CloseHandle(hGenIN);</pre>
ReadFile	<p>入力データを読み込みます。 読み込みサイズは1Byteまでです。1Byte以上の場合はエラーとなります。</p> <pre>res = ReadFile(hGenIN, &amp;InData, 1, NULL);</pre>
DeviceIoControl	<ul style="list-style-type: none"> <li>● IOCTL_GNI_SET_IN1INT IN1割り込みを設定します。</li> <li>● IOCTL_GNI_GET_IN1INT 現在のIN1割り込み設定を取得します。</li> <li>● IOCTL_GNI_SET_INORST IN0リセットを設定します。</li> <li>● IOCTL_GNI_GET_INORST 現在のIN0リセット設定を取得します。</li> </ul>



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。

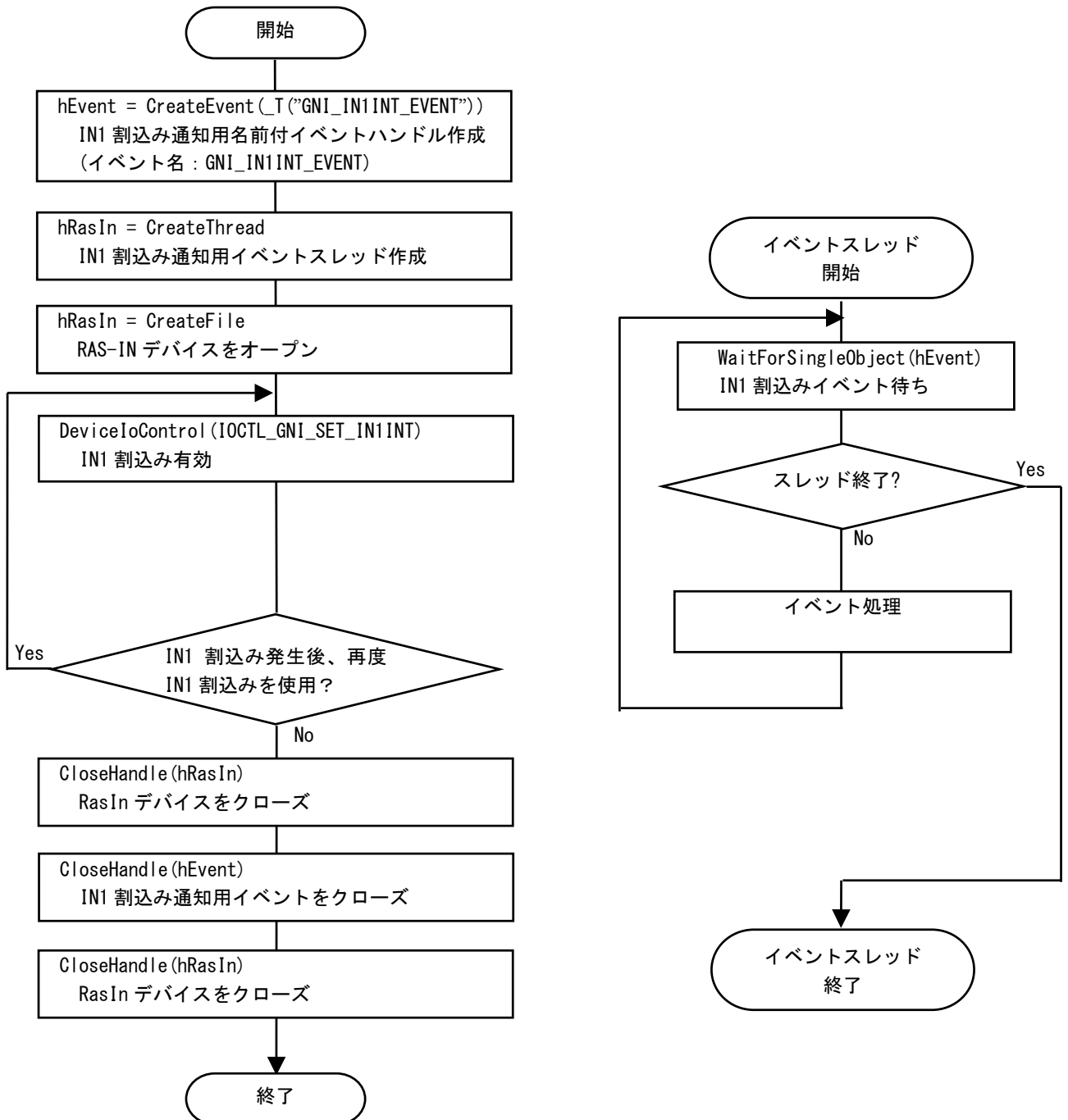


図 5-4-7-1. IN1 割込み仕様手順

5-4-8 汎用入力ファレンス

ReadFile

**機能**

入力データを読みみます。

**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**

入力データを読みみます。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 の読みみしか出来ません。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** : 実際の実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** : 実際の実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** : 実際の実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** : 実際の実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("GN01:")
#define GENIN_DRIVERNAME  _T("GN11:")

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。
    */
}
```

```
*/

/* 汎用出力デバイスへのオープン */
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;
}

/* 汎用出力デバイスへのクローズ */
CloseHandle(hGenOut);

/* 汎用入力
   汎用入力の入力を行います。
*/

/* 汎用入力デバイスへのオープン */
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"));
}

/* 汎用入力ポートのクローズ */
CloseHandle(hGenIn);

return 0;
}
```

## ● INO リセット

「¥SDK¥Algo¥Sample¥Sample\_Reset¥In0Reset」に INO 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. INO リセット

```

/**
 汎用入力 INO 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_g2mp¥GenIO.h>

#define GENIN_DRIVERNAME  _T("GNI1:")

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

    /* 汎用入力デバイスへのオープン */
    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;
    }

    /*
     * INO 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_g2mp¥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;
    }

    /* 汎用入力デバイスへのオープン */
    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;  
}
```

## 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できます。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-3300 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¥G2MP_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 イメージ書き込みツールで 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。

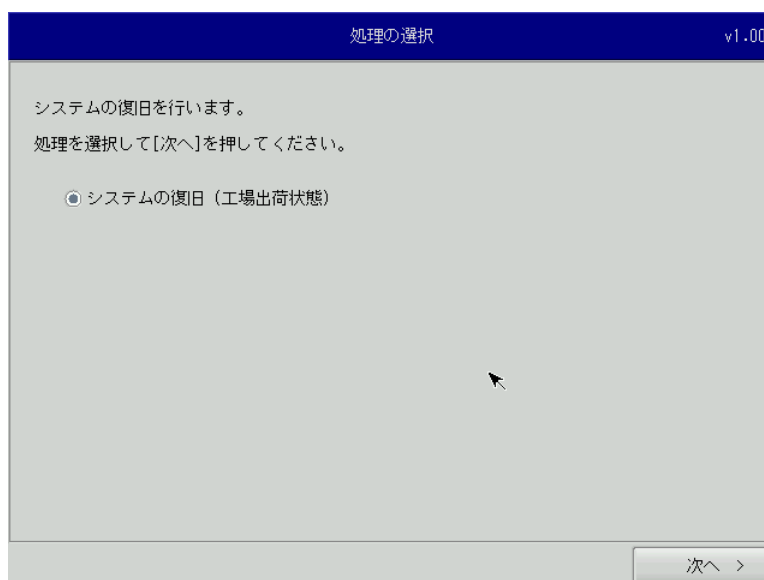


図 6-2-1. リカバリメイン画面

### 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。

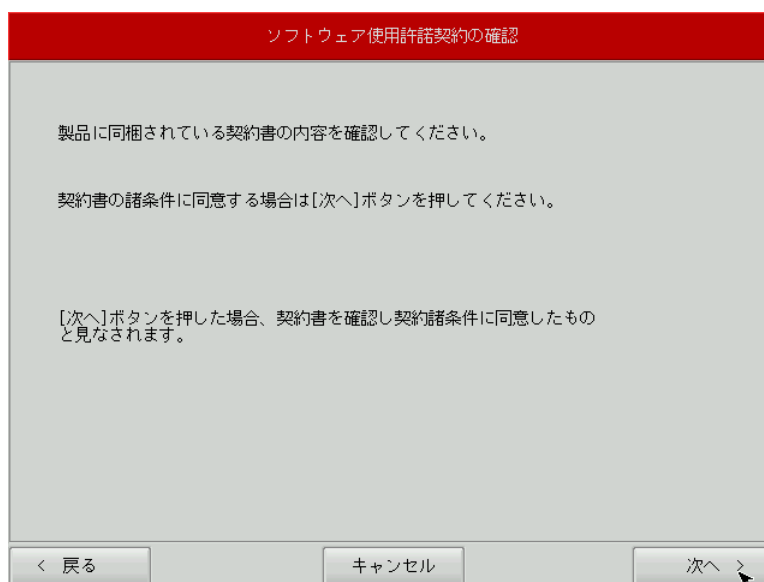


図 6-3-1. ソフトウェア使用許諾契約確認画面



- 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。

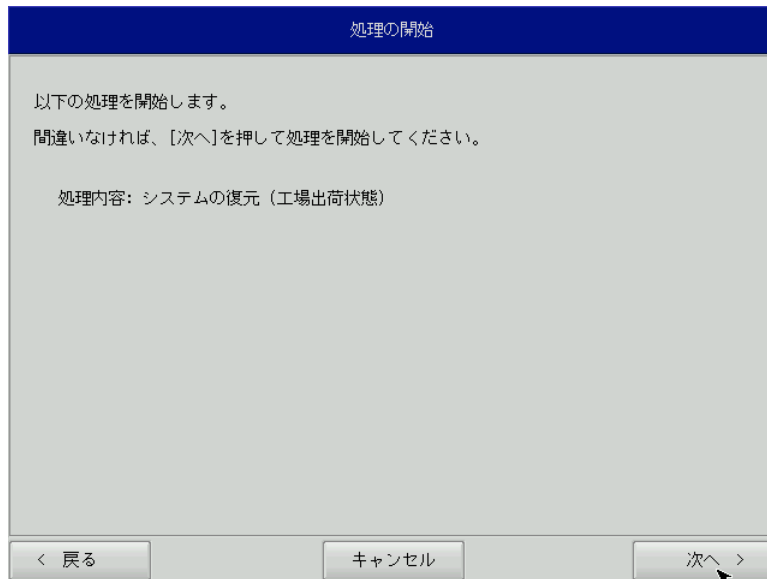


図 6-3-2. 確認画面

- 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。

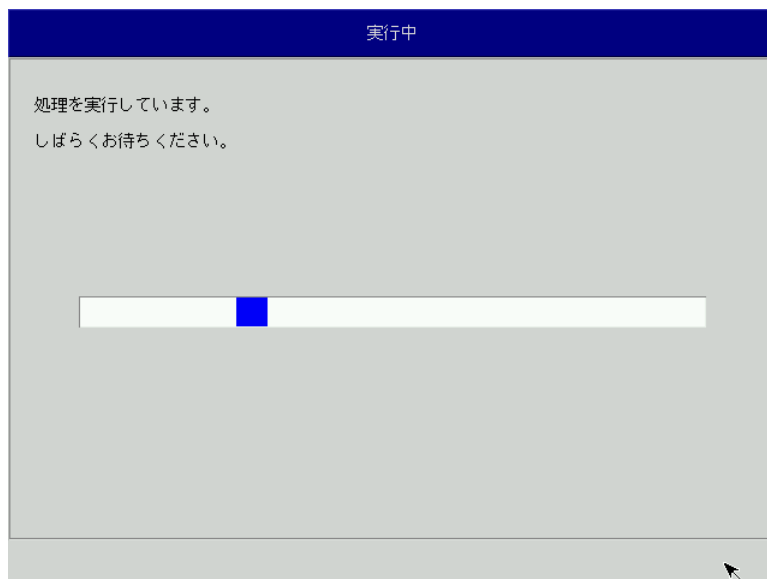


図 6-3-3. 実行中画面

- 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。

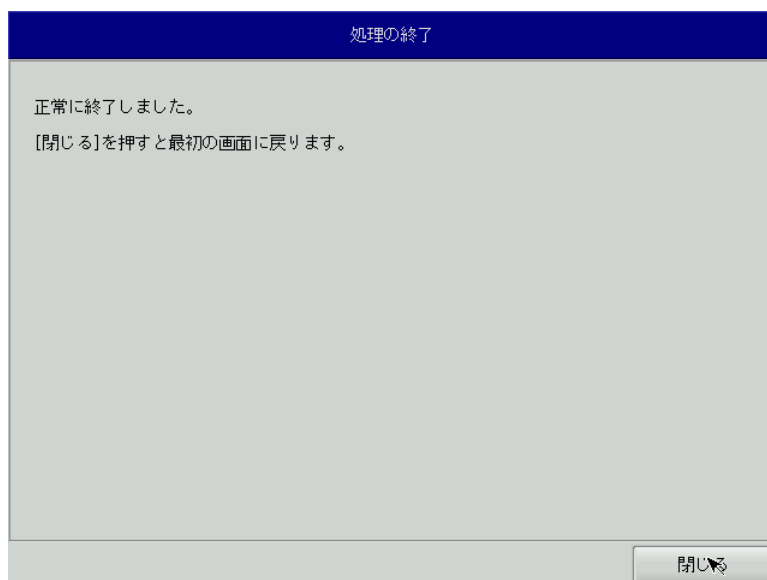


図 6-3-4. 終了画面

- 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み込み用OS (Embedded) について

### 【OS の注意事項】

本製品に搭載している OS は組み込み用 (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み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、組み込み用 (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み込み、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 (ソフトウェアを含む) 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 はお客様が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ィアを意味します。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で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ェ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ィングデバイス（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ィ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 Embedded 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、田藤 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づ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。

77W010004A

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>