お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

RENESAS

ユーザーズ・マニュアル

CAN ソフトウエア・ドライバ

資料番号　U16844JJ3V0UM00（第 3 版）
発行年月　July 2007　NS

〔メ　モ〕

目 次 要 約

Windows および Windows XP は，米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

Multi は，米国 Green Hills Software, Inc.の商標です。

PC/AT は米国 IBM 社の商標です。

その他，記載の会社名／製品名は，各社の商標，または，登録商標です。

- 本資料に記載されている内容は2007年7月現在のもので，今後，予告なく変更することがあります。量産設計の際には最新の個別データ・シート等をご参照ください。
- 文書による当社の事前の承諾なしに本資料の転載複製を禁じます。当社は，本資料の誤りに関し，一切その責を負いません。
- 当社は，本資料に記載された当社製品の使用に関連し発生した第三者の特許権，著作権その他の知的財産権の侵害等に関し，一切その責を負いません。当社は，本資料に基づき当社または第三者の特許権，著作権その他の知的財産権を何ら許諾するものではありません。
- 本資料に記載された回路，ソフトウエアおよびこれらに関する情報は，半導体製品の動作例，応用例を説明するものです。お客様の機器の設計において，回路，ソフトウエアおよびこれらに関する情報を使用する場合には，お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し，当社は，一切その責を負いません。
- 当社は，当社製品の品質，信頼性の向上に努めておりますが，当社製品の不具合が完全に発生しないことを保証するものではありません。当社製品の不具合により生じた生命，身体および財産に対する損害の危険を最小限度にするために，冗長設計，延焼対策設計，誤動作防止設計等安全設計を行ってください。
- 当社は，当社製品の品質水準を「標準水準」，「特別水準」およびお客様に品質保証プログラムを指定していただく「特定水準」に分類しております。また，各品質水準は，以下に示す用途に製品が使われることを意図しておりますので，当社製品の品質水準をご確認ください。

  標準水準：コンピュータ，OA機器，通信機器，計測機器，AV機器，家電，工作機械，パーソナル機器，産業用ロボット

  特別水準：輸送機器（自動車，電車，船舶等），交通用信号機器，防災・防犯装置，各種安全装置，生命維持を目的として設計されていない医療機器

  特定水準：航空機器，航空宇宙機器，海底中継機器，原子力制御システム，生命維持のための医療機器，生命維持のための装置またはシステム等

  当社製品のデータ・シート，データ・ブック等の資料で特に品質水準の表示がない場合は，標準水準製品であることを表します。意図されていない用途で当社製品の使用をお客様が希望する場合には，事前に当社販売窓口までお問い合わせください。

(注)

（1）本事項において使用されている「当社」とは，NECエレクトロニクス株式会社およびNECエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいう。

（2）本事項において使用されている「当社製品」とは，（1）において定義された当社の開発，製造製品をいう。

M8E 02.11

はじめに

対 象 者　このマニュアルは，CAN ソフトウエア・ドライバの機能を理解し，それを用いた応用システムを設計するユーザを対象とします。

目　　的　このマニュアルは，次の構成に示す CAN ソフトウエア・ドライバの機能をユーザに理解していただくことを目的としています。

構　　成　このマニュアルでは，大きく分けて次の内容で構成しています。

- **製品概要**
- **インストレーション**
- **システム構築**
- **コンフィギュレーション**
- **ドライバ関数**
- **サンプル・プログラム**

読 み 方　このマニュアルを読むにあたっては，電気，論理回路，マイクロコントローラに関する一般的知識が必要となります。

CAN ソフトウエア・ドライバの機能を理解しようとするとき

→目次に従ってお読みください。本文欄外の★印は，本版で改訂された主な箇所を示しています。この"★"を PDF 上でコピーして「検索する文字列」に指定することによって，改版箇所を容易に検索できます。

凡　　例　データ表記の重み　：左が上位桁，右が下位桁

注　：本文中につけた注の説明

注意　：気をつけて読んでいただきたい内容

備考　：本文中の補足説明

数の表記　：2 進数…××××または 0××××B

10 進数…××××

16 進数…××××H

2 のべき数を示す接頭語（アドレス空間，メモリ容量）：

K（キロ）…2^{10} = 1024

M（メガ）…2^{20} = 1024^2

G（ギガ）…2^{30} = 1024^3

データ・タイプ　：ワード…32 ビット

ハーフ・ワード…16 ビット

バイト…8 ビット

関連資料　関連資料は暫定版の場合がありますが，この資料では「暫定」の表示をしておりません。あらかじめご了承ください。

デバイスに関する資料

・対象デバイスに関する内容は，各製品の**ユーザーズ・マニュアル　ハードウエア編**を参照してください。

開発ツールに関する資料（ユーザーズ・マニュアル）

関連資料		資料番号
CA850　Ver.3.00　**Cコンパイラ・パッケージ**	**操作編**	U17293J
	C言語編	U17291J
	アセンブリー言語編	U17292J
	リンク・ディレクティブ編	U17294J
PM+ Ver.6.30　**プロジェクト・マネージャ**		U18416J
ID850　Ver.3.00　**統合デバッガ**	**操作編**	U17358J
ID850QB　Ver.3.40　**統合デバッガ**	**操作編**	U18604J
RX850　Ver.3.20 **以上　リアルタイム** OS	**基礎編**	U13430J
	インストレーション編	U17419J
	テクニカル編	U13431J
	タスク・デバッガ編	U17420J
RX850 Pro　Ver.3.21　**リアルタイム** OS	**基礎編**	U18165J
	インストレーション編	U17421J
	テクニカル編	U13772J
	タスク・デバッガ編	U17422J
RD850　Ver.3.01　**タスク・デバッガ**		U13737J
AZ850　Ver.3.30　**システム・パフォーマンス・アナライザ**		U17423J
SM+　**システム・シミュレータ**	**操作編**	U18601J
	ユーザ・オープン・インタフェース編	U18212J
RA78K0　Ver.3.80　**アセンブラ・パッケージ**	**操作編**	U17199J
	言語編	U17198J
	構造化アセンブリー言語編	U17197J
CC78K0　Ver.3.70　C **コンパイラ**	**操作編**	U17201J
	言語編	U17200J
ID78K0-QB　Ver.3.00　**統合デバッガ**	**操作編**	U18492J
ID78K0　**統合デバッガ**　EWS **ベース**	**レファレンス編**	U11151J
PM+ Ver.6.30		U18416J
PG-FP4　**フラッシュ・メモリ・プログラマ**		U15260J

注意　上記関連資料は，予告なしに内容を変更することがあります。設計などには，必ず最新の資料をご使用ください。

目　次

図の目次

図番号　　タイトル，ページ

表の目次

表番号　　タイトル，ページ

第1章　製品概要

1.1　概　　要

CAN ソフトウエア・ドライバは，NEC エレクトロニクス製の CAN 通信機能付き V850 32 ビット・マイクロコントローラ，78K0 8 ビット・マイクロコントローラで通信を実現させるためのアプリケーション・プログラム・インタフェース関数（API 関数）を提供しています。

1.2　特　　徴

1.2.1　高い移植性

ユーザが CAN のハードウエア依存箇所，CPU コアを意識せずに CAN 通信プログラムの記述を可能としています。これにより，実行環境への移植性，カスタマイズを容易なものとしています。

1.2.2　コンフィギュレーション・ツールの提供

ユーザの使用するデバイス，環境に則した CAN のハードウエア初期設定，メッセージの静的生成が GUI を中心とした操作により容易に設定可能です。

1.3　CAN ソフトウエア・ドライバの種類

★　CAN ソフトウエア・ドライバは，ソース・ファイルで提供されます。CAN コントローラの種類，パラメータ・チェック機能の有無を選択し，ソース・ファイルを出力させることができます。

種　別	説　明
ハードウエア依存	CPU コア，CAN コントローラごとに用意されています。
パラメータ・チェック機能	CAN ソフトウエア・ドライバ関数内で，ユーザ指定のパラメータをチェックし，エラーとして認識する機能を持つものと，持たないものを選択可能です。 パラメータ・チェック機能は，CAN ソフトウエア・ドライバ関数内でエラー・チェックを行うため，不正なパラメータ指定が防げる反面，コード容量が増加します。したがって，デバッグ，評価時の使用を推奨します。

1. 4　実行環境

CAN ソフトウエア・ドライバは，次のハードウエアを備えたターゲット・システム上で動作します。

★　(1) 対象 CPU

V850 マイクロコントローラ	V850ES/FE2, V850ES/FF2, V850ES/FG2, V850ES/FJ2, V850ES/FE3, V850ES/FF3, V850ES/FG3, V850ES/FJ3, V850ES/FK3, V850ES/SG2, V850ES/SJ2, V850ES/SG3, V850ES/SJ3, V850E/RS1, V850E/RS2, V850E/PG2, V850E/DJ3, V850E/DL3, V850E/IA1
78K0 マイクロコントローラ	78K0/FC2, 78K0/FE2, 78K0/FF2, μPD780822B

(2) 対象 CAN コントローラ

aFCAN

DCAN

FCAN

(3) メモリ容量

メモリ容量は，ユーザが使用する関数（API）の数，メッセージ・バッファとチャネル数，CAN コントローラおよびレジスタ・モードなどにより変化します。特に，関数（API）の数に大きく依存します。

★　全関数（16 関数）と第 6 章　サンプル・プログラムで使用する関数（8 関数）のメモリ容量は，次のとおりです。

★　(a) 全 16 関数のメモリ容量

- コンパイラ：NEC エレクトロニクス製　CA850　Ver. 3.00
- 最適化：サイズ最適化
- CAN コントローラ：aFCAN
- レジスタ・モード：32 レジスタ

★　- 使用関数（API）：全 16 関数
- 使用テーブル：チャネル数：2

送受信メッセージ・バッファ：各チャネル 10 メッセージ・バッファ

（合計：20 メッセージ・バッファ）

★

	ROM 容量	RAM 容量
全 16 関数（API）	約 1.46 K バイト	最大 8 バイト
テーブル	272 バイト	−
合　計	約 1.73 K バイト	

備考　上記容量はコンパイラのバージョンに依存します。

RAM 容量はスタックで使用します。

★ (b) サンプル・プログラムの8関数のメモリ容量

- コンパイラ：NEC エレクトロニクス製　CA850　Ver. 3.00
- 最適化：サイズ最適化
- CAN コントローラ：aFCAN
- レジスタ・モード：32 レジスタ

★
- 使用関数（API）：8 関数
- 使用テーブル：チャネル数：1

送受信メッセージ・バッファ：2 メッセージ・バッファ

★

	ROM 容量	RAM 容量
サンプル・プログラムの8関数（API）	約1.01 K バイト	最大8 バイト
テーブル	116 バイト	−
合　計	約1.13 K バイト	

備考　上記容量はコンパイラのバージョンに依存します。

RAM 容量はスタックで使用します。

★ 1. 5　開発環境

CAN ソフトウエア・ドライバを使用したアプリケーション・システムを開発する上で必要となる環境を，次に示します。

(1) ハードウエア

ホスト・マシン

- IBM-PC/AT ™ 互換機

次の Windows® に対応　：Windows 98, Windows Me, Windows 2000, Windows XP®

(2) ソフトウエア

コンパイラ・パッケージ

- CA850　：NEC エレクトロニクス製
- CCV850　：Green Hills Software, Inc.製
- CC78K0　：NEC エレクトロニクス製

(3) デバッガ

- ID850　：NEC エレクトロニクス製
- Multi ™　：Green Hills Software, Inc.製
- PARTNER　：京都マイクロコンピュータ製
- ID78K0　：NEC エレクトロニクス製
- ID78K0-NS　：NEC エレクトロニクス製

(4) シミュレータ

- SM850　：NEC エレクトロニクス製
- SM78K0　：NEC エレクトロニクス製

第2章　インストレーション

2. 1　概　　要

CAN ソフトウエア・ドライバの提供媒体は，Windows ベースで用意されています。
また，CAN ソフトウエア・ドライバ関数はソース・ファイルで提供されます。

★ 表 2－1　CAN ソフトウエア・ドライバの提供方法

提供形式	提供内容	提供媒体
全インストール	• サンプル・プログラム • CAN コンフィギュレータ／CAN ソフトウエア・ドライバ （CAN コンフィギュレータ，ドライバ・ソース・ファイル，ヘッダ・ファイル，デバイス・データ・ベース） • コンフィギュレータ • ドキュメント（このドキュメント）	CD-ROM
データ・ベース	• サンプル・プログラム	Web（提供予定），FD

★ 2. 2　インストール手順

CAN ソフトウエア・ドライバの提供媒体に格納されているファイル群をホスト・マシンにインストールする際の手順を次に示します。

2. 2. 1　Windows の起動

ホスト・マシンおよび周辺機器などの電源を投入し，Windows を起動します。

2. 2. 2　提供媒体のセット

CAN ソフトウエア・ドライバの提供媒体をホスト・マシンの該当デバイス装置（CD-ROM）にセットすることにより，セットアップ・プログラムが自動実行されます。

インストーラは，大きく分けて次の 3 つのファイルを出力します。

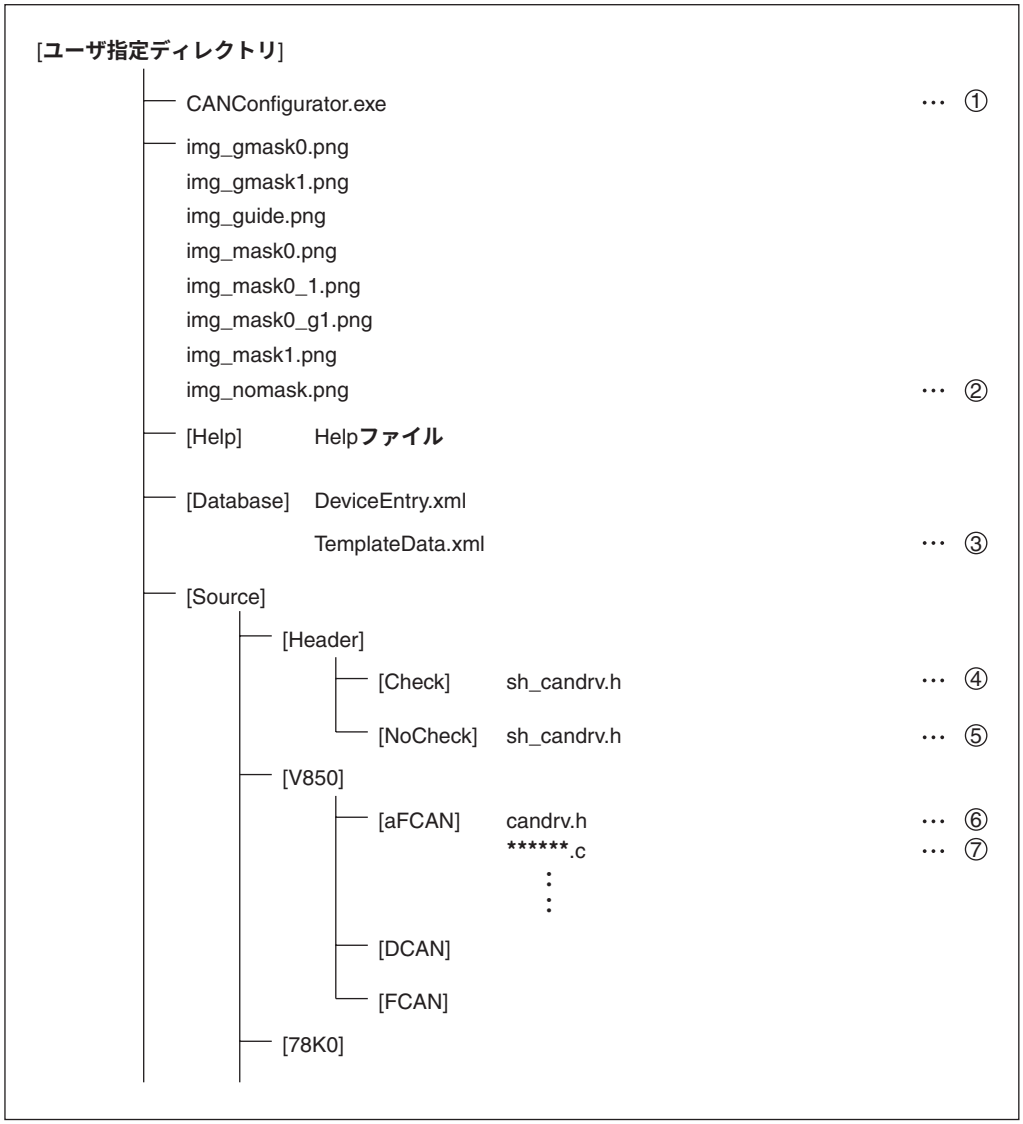
★
- CAN コンフィギュレータ／CAN ソフトウエア・ドライバ
- ドキュメント
- サンプル・プログラム

各出力ファイルとも，モニタ画面に表示されるメッセージに従ってインストール作業を実行します。

2. 3　ディレクトリ構成

2. 3. 1　CAN ソフトウエア・ドライバ

★ **図2－1　CAN ソフトウエア・ドライバのディレクトリ構成**



①　：　コンフィギュレータ本体

②　：　コンフィギュレータ・イメージ・ファイル

③　：　バージョン情報ファイル

④　：　パラメータ・チェックありヘッダ・ファイル

⑤　：　パラメータ・チェックなしヘッダ・ファイル

⑥　：　CAN ソフトウエア・ドライバ・ヘッダ・ファイル

⑦　：　CAN ソフトウエア・ドライバ本体ソース・ファイル

2.3.2　ドキュメント

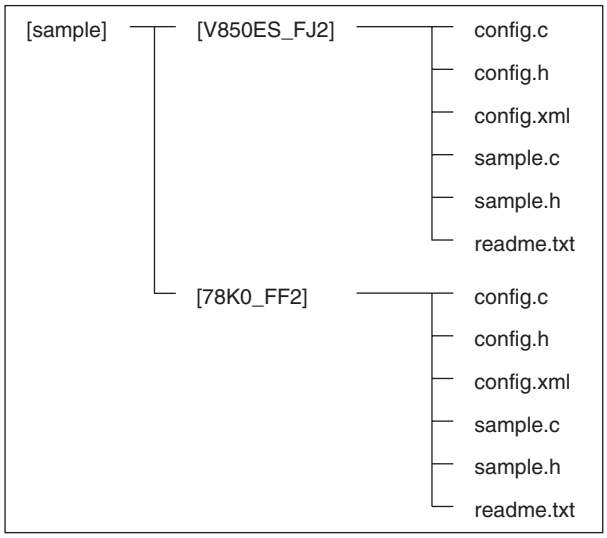
このドキュメントを任意のディレクトリ上にコピーしてください。

2.3.3　サンプル・プログラム

★　V850ES/FJ2 と 78K0/FF2 のサンプル・プログラムが用意されています。任意のディレクトリに 図 2−2のディレクトリ構成でインストールされます。

各サンプル・プログラムの動作については，各ディレクトリ内の readme.txt を参照してください。

★　**図 2−2　サンプル・プログラムのディレクトリ構成**



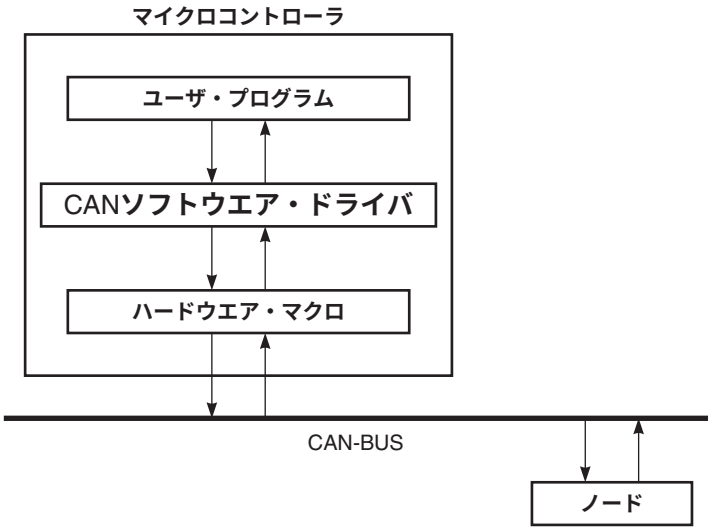
第3章　システム構築

3. 1　CAN ソフトウエア・ドライバの位置づけ

CANソフトウエア・ドライバは，システム内においてユーザ・アプリケーションとハードウエアの間に位置しています（図 3－1参照）。また，ハードウエア制御を行うためのユーザ・インタフェースを持っています。

ユーザは，アプリケーション内に CAN ソフトウエア・ドライバ関数を記述することによって，ハードウエアのレジスタ制御を意識する必要がなくなります。

図 3－1　システム概要



3.2　システム構築手順

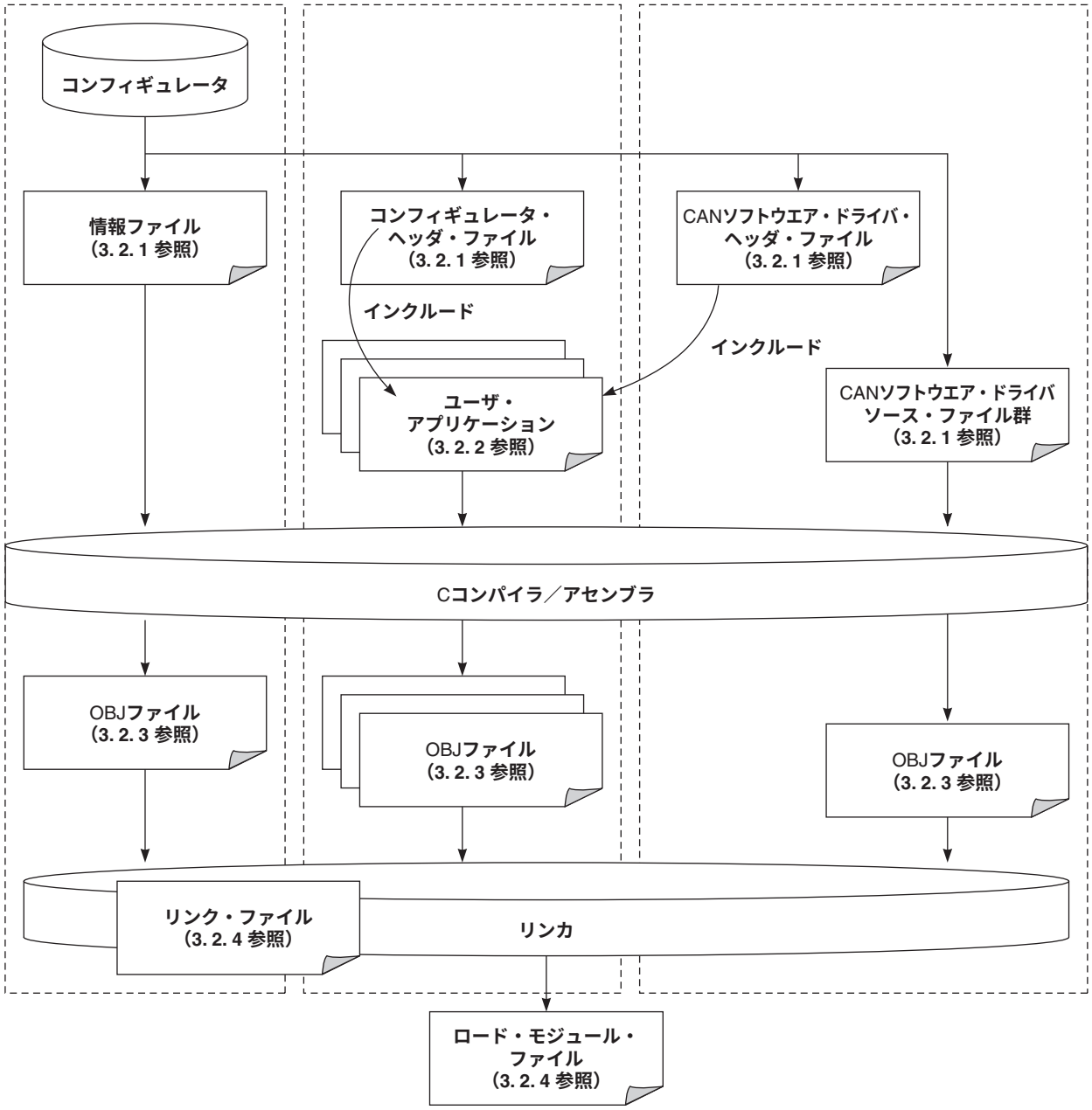
システム構築とは，CAN ソフトウエア・ドライバの提供媒体からユーザの開発環境（ホスト・マシン）上にインストールしたファイル群を用いてロード・モジュール・ファイルを生成することです。

ここでは次の略称を使用して説明しています。

- OBJ ファイル：オブジェクト・ファイル
- LINK ファイル：リンク・ディレクティブ・ファイル

図にシステム構築の手順を示します。

図 3－2　システム構築手順



3. 2. 1　コンフィギュレータによるファイル生成

コンフィギュレータにより，CAN のボー・レート，メッセージ・バッファの送受信割付，割り込み，マスクなどの初期値を設定し，情報ファイル，ヘッダ・ファイルの形で生成します。

コンフィギュレータによる設定手順の詳細については**第4章　コンフィギュレーション**を参照してください。

(1) コンフィギュレータの入力データ

コンフィギュレータによるファイル生成を行う前には,最低次に示す内容を決定しておく必要があります。

- 使用するデバイス（マイクロコントローラ名称，デバイス名）
  例　V850ES/FJ2 マイクロコントローラのμPD70F3239
- デバイスのシステム・クロック
- 使用するチャネル
- モジュールごとの CAN システム・クロック
- モジュールごとのボー・レート，データ・ビット・タイムの構成
- モジュールごとのメッセージ・バッファ設定
  送信／受信割付，CAN-ID，標準／拡張フレーム，割り込み通知使用／未使用，マスク設定など
- モジュールごとの割り込み許可／禁止
  エラー通知，送受信通知

(2) コンフィギュレータの出力データ

コンフィギュレータで出力されるファイルは次の 4 種類です。

- 情報ファイル
  ユーザにより入力された CAN の設定情報のテーブル群です。CAN ソフトウエア・ドライバが参照します。したがって，ユーザ・アプリケーション・プログラムとともに，コンパイル／アセンブル，リンクを実行させる必要があります。
- コンフィギュレータ・ヘッダ・ファイル
  ユーザが設定したメッセージ名称がマクロ定義されています。ユーザはこのマクロ名を使用して CAN ソフトウエア・ドライバ関数への引数などに使用します。
  したがって，ユーザ・アプリケーション内にインクルードさせる必要があります。
- CAN ソフトウエア・ドライバ・ソース・ファイル群
  使用するデバイスに適合する CAN ソフトウエア・ドライバ・ソース・ファイル群を出力します。ユーザ・アプリケーション・プログラムとともに，コンパイル／アセンブル，リンクを実行させる必要があります。
- CAN ソフトウエア・ドライバ・ヘッダ・ファイル（candrv.h）
  使用される CAN ソフトウエア・ドライバ関数と組み合わせて使用するヘッダ・ファイルを出力します。

3. 2. 2　ユーザ・アプリケーション

CANのAPI関数を使用してCAN通信アプリケーションを作成します。ドライバ関数を使用しているファイルには，コンフィギュレータで作成されたヘッダ・ファイルのインクルードが必要です。

また，CANのハードウエアおよびCANソフトウエア・ドライバ関数を使用するために，次の設定をユーザ自身でコーディングしてください。

次の（1）から（4）までは初期化ルーチン，（5）と（6）はCAN制御アプリケーションに関する記述です。

（1）システムの設定

使用するデバイスによって設定する項目が異なります。

たとえば，次のような設定レジスタがあります。

V850 マイクロコントローラの場合

- システム・ウエイト・コントロール・レジスタ（VSWC）
- PLL コントロール・レジスタ（PLLCTL）

78K0 マイクロコントローラの場合

- システム・ウエイト・コントロール・レジスタ（VSWC）
- メモリ・サイズ切り替えレジスタ（IMS）
- 内部拡張 RAM サイズ切り替えレジスタ（IXS）

（2）ポートの設定

使用する CAN コントローラの送受信に割り当てられた端子を CAN 送受信モードに指定します。

（3）BPC の設定（デバイスによっては設定不要）

プログラマブル周辺 I/O レジスタ領域の設定を行います。この領域は CAN コントローラ用の周辺 I/O レジスタが割り当てられている領域です。

（4）割り込みの設定

CAN コントローラで割り込みハンドラを使用する場合，割り込みハンドラ・ルーチンへの割り込み先アドレスの記述が必要です。

（5）ヘッダ・ファイルのインクルード

次の 2 つのヘッダ・ファイルを CAN ソフトウエア・ドライバが使用しているソース・ファイルにインクルードさせます。

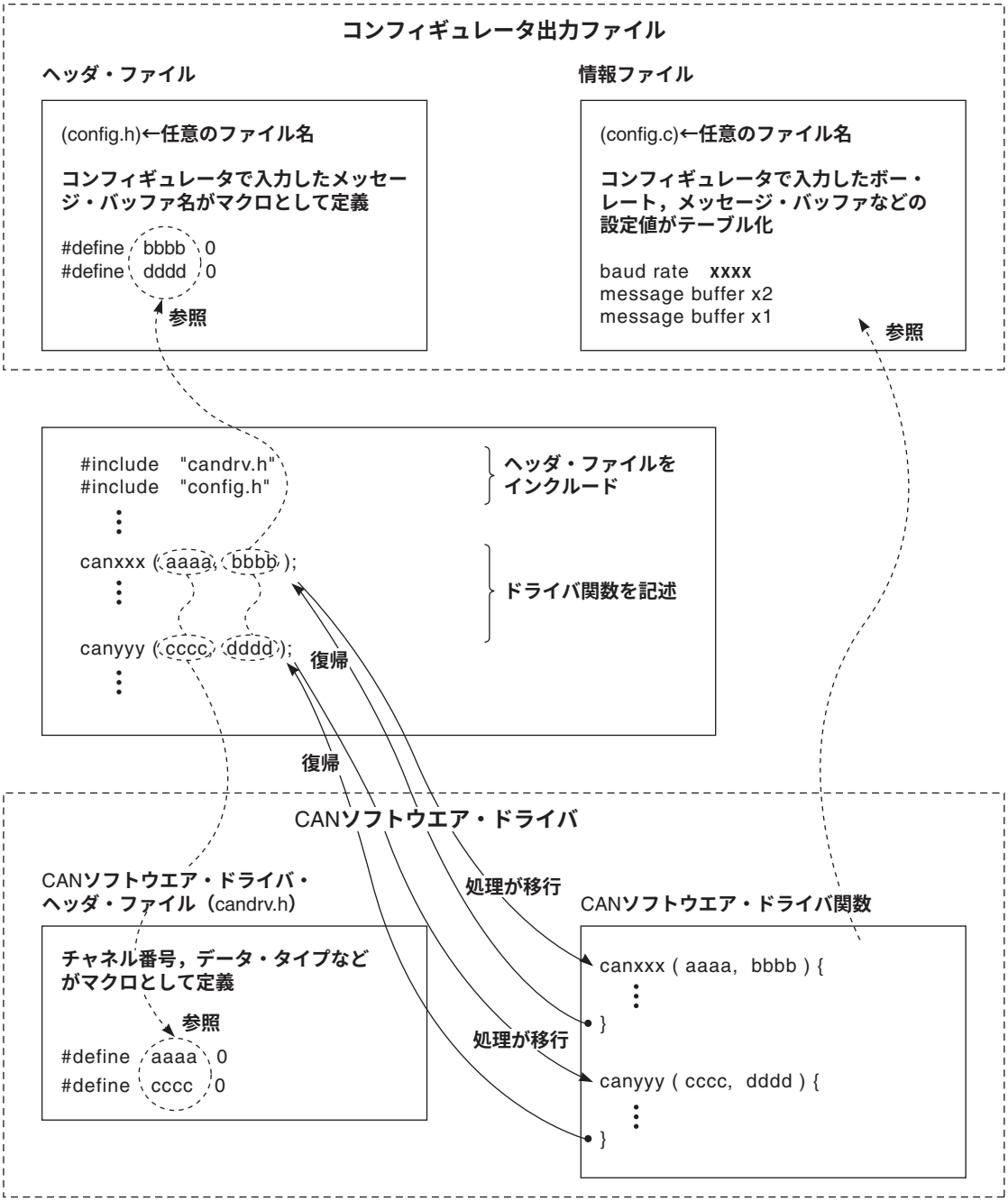
- candrv.h　：CAN ソフトウエア・ドライバとともに提供されるヘッダ・ファイル
- *****.h　　：コンフィギュレータで作成されたヘッダ・ファイル（*****はユーザが任意に決定）

（6）CAN ソフトウエア・ドライバ関数（API）の記述

アプリケーション・プログラム中にドライバ API 関数を使用した CAN の制御コードを記述します。

CAN ソフトウエア・ドライバの API 関数については**第 5 章　ドライバ関数**を参照してください。

図3−3　アプリケーション・プログラムとCANソフトウエア・ドライバ／コンフィギュレータとの相関関係



3.2.3　オブジェクト・ファイルの生成

ユーザ・アプリケーションと CAN コンフィギュレータで作成された情報ファイル，CAN ソフトウエア・ドライバ・ソース・ファイル群にコンパイル／アセンブルを実行し，リロケータブルなオブジェクト・ファイルを生成します。

備考　C コンパイラ／アセンブラの起動オプション，および実行方法についての詳細は，各ツールのユーザーズ・マニュアルを参照してください。

3.2.4　ロード・モジュール・ファイルの生成

次のファイル群に対して，リンクを実行してロード・モジュール・ファイルを生成します。

- ユーザ・アプリケーションをコンパイル／アセンブルしたオブジェクト・ファイル
- CAN コンフィギュレータで作成された情報ファイルをコンパイルしたオブジェクト・ファイル
- CAN コンフィギュレータで作成された CAN ソフトウエア・ドライバ・ソース・ファイル群をコンパイルしたオブジェクト・ファイル
- リンク・ディレクティブ・ファイル
- C コンパイラ・パッケージが推奨するライブラリ・ファイル

備考　リンク・エディタの起動オプション，および実行方法についての詳細は，各ツールのユーザーズ・マニュアルを参照してください。なお，上記の手順は，NEC エレクトロニクス製コンパイラの例ですので，その他のツールを使う場合は各ツールのユーザーズ・マニュアルを参照してください。

第4章　コンフィギュレーション

4.1　概　　要

コンフィギュレータは，ユーザがCAN機能を組み込んだシステム構築において，CANの初期化値を設定するための開発支援ツールです。ユーザが使用するデバイスに則したレジスタの初期設定や，システム中で使用するメッセージの静的生成を行うための機能を提供します。

使用するデバイスの選択とシステム・クロック値の入力やチャネルごとのボー・レート値の指定などから，そのデバイスに則したレジスタの初期設定を行うことができます。また，クロックやボー・レートの設定にとどまらず，割り込みの使用／未使用といった設定，メッセージ・バッファの機能設定も可能です。

ボー・レートを設定するための別ウインドウでは，ユーザの入力した値から設定可能な数値を自動計算し，一覧表にまとめて提示します。ユーザがその中から任意の設定値を選択すると，その値に応じた棒グラフが表示されます。このことにより，ユーザはグラフィカルにボー・レート関連の値を選択，設定することができます。

メッセージの静的生成については，送信，受信メッセージへの割り振り，メッセージ名称，CAN-ID，割り込みの許可／禁止などを設定することができます。

備考　CANコントローラの機能に関する詳細は，各デバイスのドキュメントのCANコントローラの章を参照してください。

4.2　プロジェクト・ファイルによる入力情報管理

コンフィギュレータはプロジェクト・ファイルにより，ユーザが入力した各種の情報を保存，管理しています。プロジェクト・ファイルを保存，読み出すことにより，何度でもドライバ関数が使用するヘッダ・ファイル，情報ファイルなどを生成させることが可能です。

4.3　ファイル生成手順

使用するデバイスを選択すれば，その他は順不同で設定することが可能です。

一般的には，4.4　コンフィギュレータの起動の手順で設定することを推奨します。

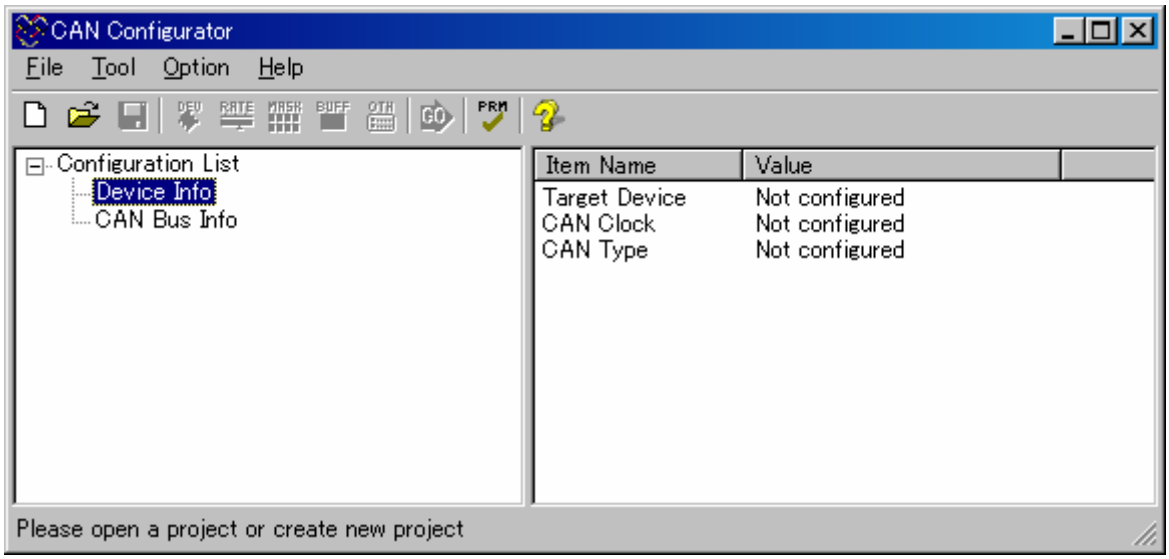
4.4　コンフィギュレータの起動

CANConfigurator.exe を起動させると，図のようなメイン画面が現れます。

画面の左側に大項目が表示され，右側には項目ごとの設定内容の概要が表示されます。

設定されていない項目は，Value 欄に Not configured と表示されます。

★　図4－1　メイン画面



4.4.1　デバイスの選択

デバイス選択メニューからデバイス情報を登録します。

＜起動方法＞

- メニューの[File] – [New]から起動
- メイン画面の Device Info の箇所をダブル・クリックして起動
- 設定項目をあとから変更したい場合は，メニューの[Tool] – [Device Setup]により起動します。

＜設定項目＞

- デバイスのマイクロコントローラ名称，対象デバイスの選択

  デバイスを選択すると，そのデバイスの CAN に関わる情報が Device Information 欄に表示されます。

  また，CAN のチャネル数も選択されているデバイスに合わせた数に変更されます。

★ - CAN クロックの入力

  CAN モジュールに供給するクロックの値を入力します。

  チャネルごとに CAN クロックを指定可能なデバイスを選択した場合は CAN Clock... ボタンが表示され，ボタンの押下により複数クロック入力用のダイアログが表示されます。

- プログラマブル I/O 領域の設定

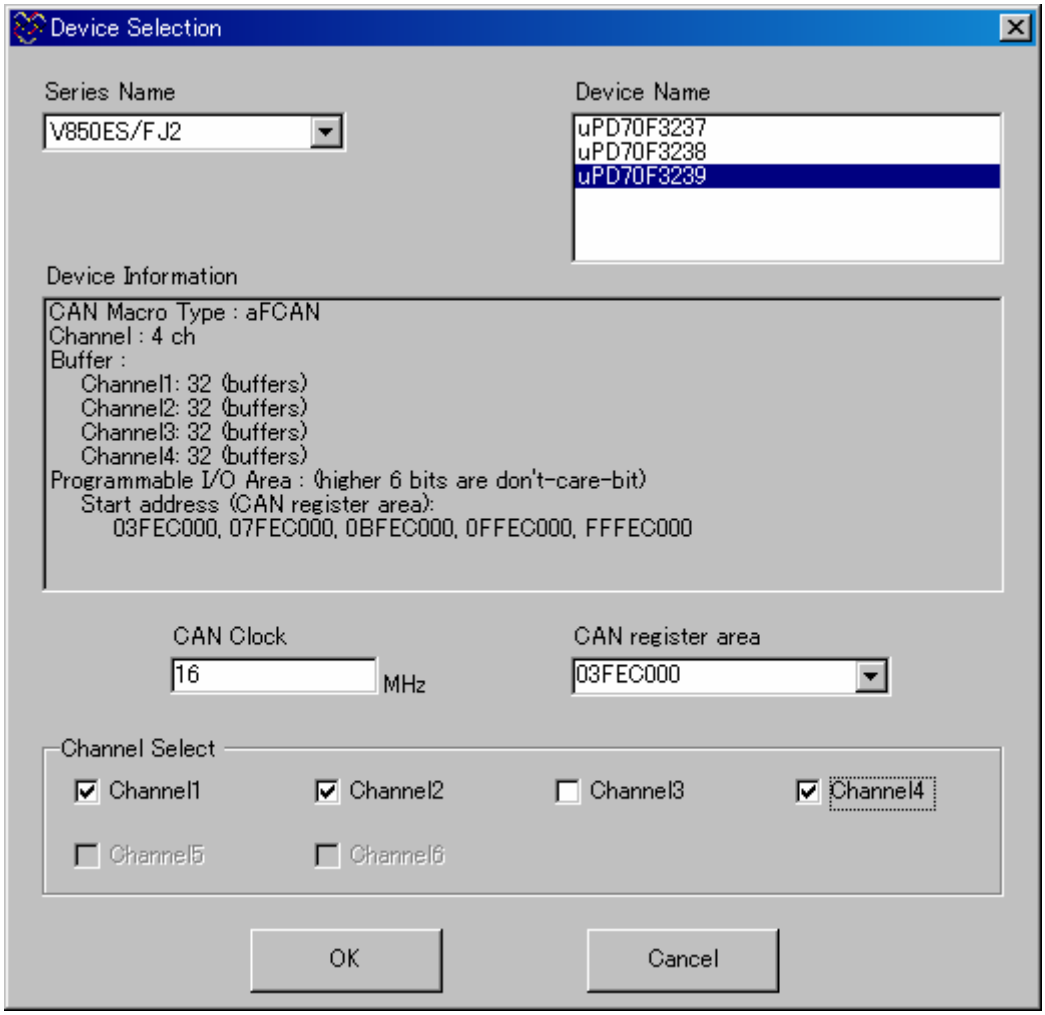
  プルダウン・メニューから選択します。プルダウン・メニュー一覧にない I/O 領域を設定したい場合は直接入力します。

  なお，使用するデバイスによっては，選択または入力ができない（領域が固定されている）場合があります。

- 使用する CAN のチャネルを選択

  使用する CAN のチャネルをチェック・ボックスで選択します。

★ **図4－2　デバイス選択メニュー画面**



4.4.2　ボー・レートの設定

各チャネルのボー・レートを設定します。

＜起動方法＞

- メニューの[Tool] – [Baud Rate Setup]から起動

＜設定項目＞　（V850-aFCAN, V850-DCAN, 78K0-aFCAN）

- CAN モジュール・システム・クロックの選択

  プルダウン・メニューから選択します。このモジュール・システム・クロックの設定により，選択できるボー・レートが決まります。

- ボー・レートの選択

  使用する CAN バス条件に合うボー・レートをプルダウン・メニューから選択します。

  プルダウン・メニューには使用頻度の高いボー・レートが表示されています。該当するボー・レートがない場合は，直接，値を入力します。入力後に Refresh ボタンを押下すると自動計算を行います。

- データ・ビット・タイムの設定

  ビット・タイムはコンフィギュレータで自動的に算出されます。該当する組み合わせのビット・タイムをリスト上から選択します。なお，設定の目安は SamplePoint が 75%前後で，SJW がなるべく大きいもの（最大は 4）です。

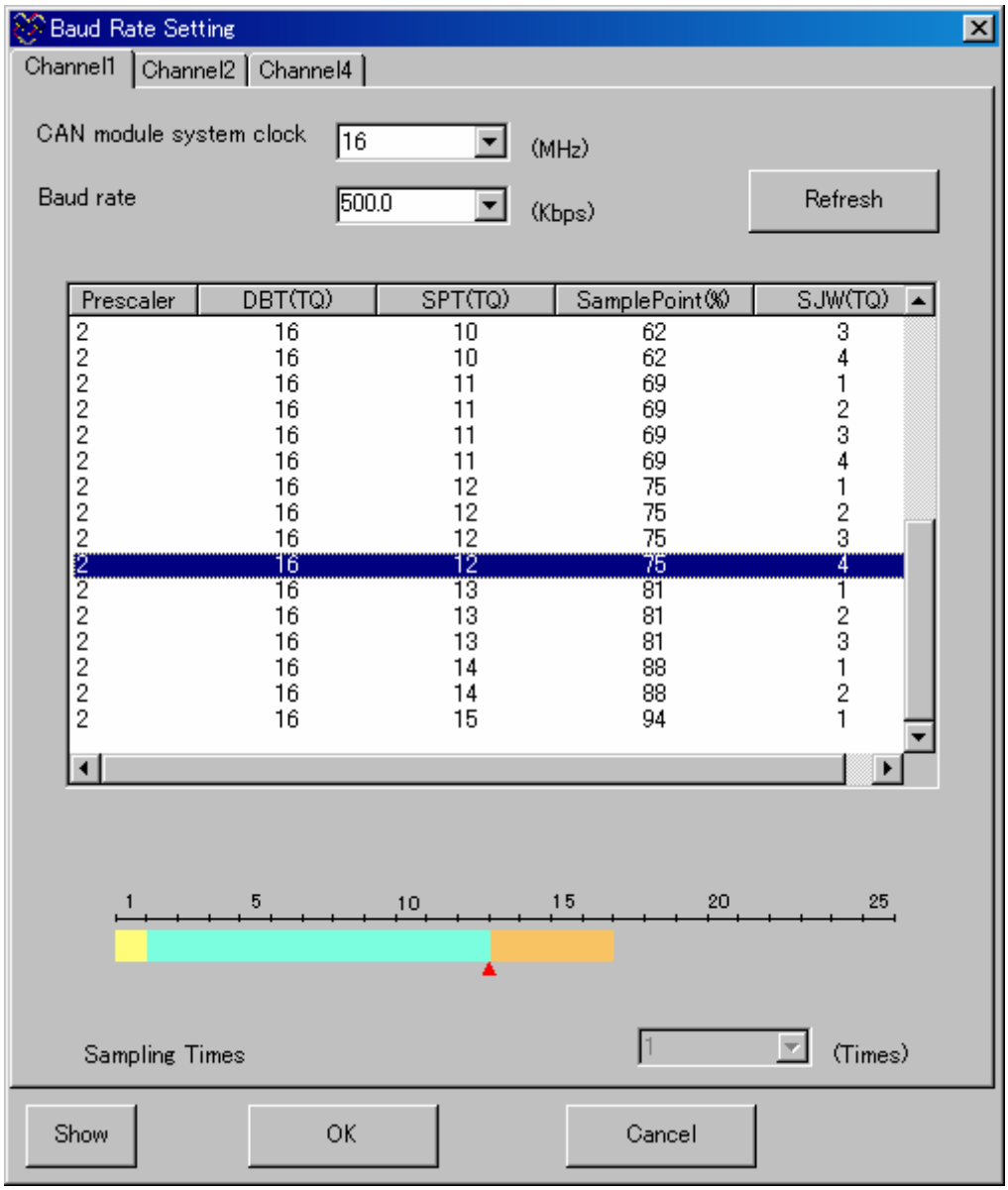
  各項目（Prescaler/DBT/SPT/SamplePoint/SJW）の項目欄を押下することにより，ソート機能が働きます。また，リスト下部にサンプル・ポイント（赤い三角印）を中心としたビット・タイムがグラフィカルに表示されます。

  Show ボタンを押下すると，レジスタに設定される値が表示されます。

- 設定状態の初期化

  Refresh ボタンを押下することで，選択していたデータ・ビット・タイムが非選択状態になり，またソート機能がディフォールト状態に戻ります。

★　**図4－3　ボー・レート設定画面（**V850-aFCAN, V850-DCAN, 78K0-aFCAN**）**



＜設定項目＞　（V850-FCAN）

- CAN グローバル・タイマ・クロックの選択

  プルダウン・メニューから選択します。このグローバル・タイマ・クロックの設定により，選択できるボー・レートが決まります。すべてのチャネルで共通の設定となります。

- タイム・スタンプ・クロックの設定

  タイム・スタンプ・クロックを入力します。単位は 

ボタンで選択します。入力後に 

ボタンを押下すると自動計算を行います。

- ボー・レートの選択

  使用する CAN バス条件に合うボー・レートをプルダウン・メニューから選択します。

  プルダウン・メニューには使用頻度の高いボー・レートが表示されています。該当するボー・レートがない場合は，直接，値を入力します。入力後に 

ボタンを押下すると自動計算を行います。

- データ・ビット・タイムの設定

  ビット・タイムはコンフィギュレータで自動的に算出されます。該当する組み合わせのビット・タイムをリスト上から選択します。

  各項目（Prescaler/DBT/SPT/SamplePoint/SJW）の項目欄を押下することにより，ソート機能が働きます。また，リスト下部にサンプル・ポイント（赤い三角印）を中心としたビット・タイムがグラフィカルに表示されます。

ボタンを押下することで TL モードを使用してより詳細な設定が可能になります。

ボタンを押下すると，レジスタに設定される値が表示されます。

- 設定状態の初期化

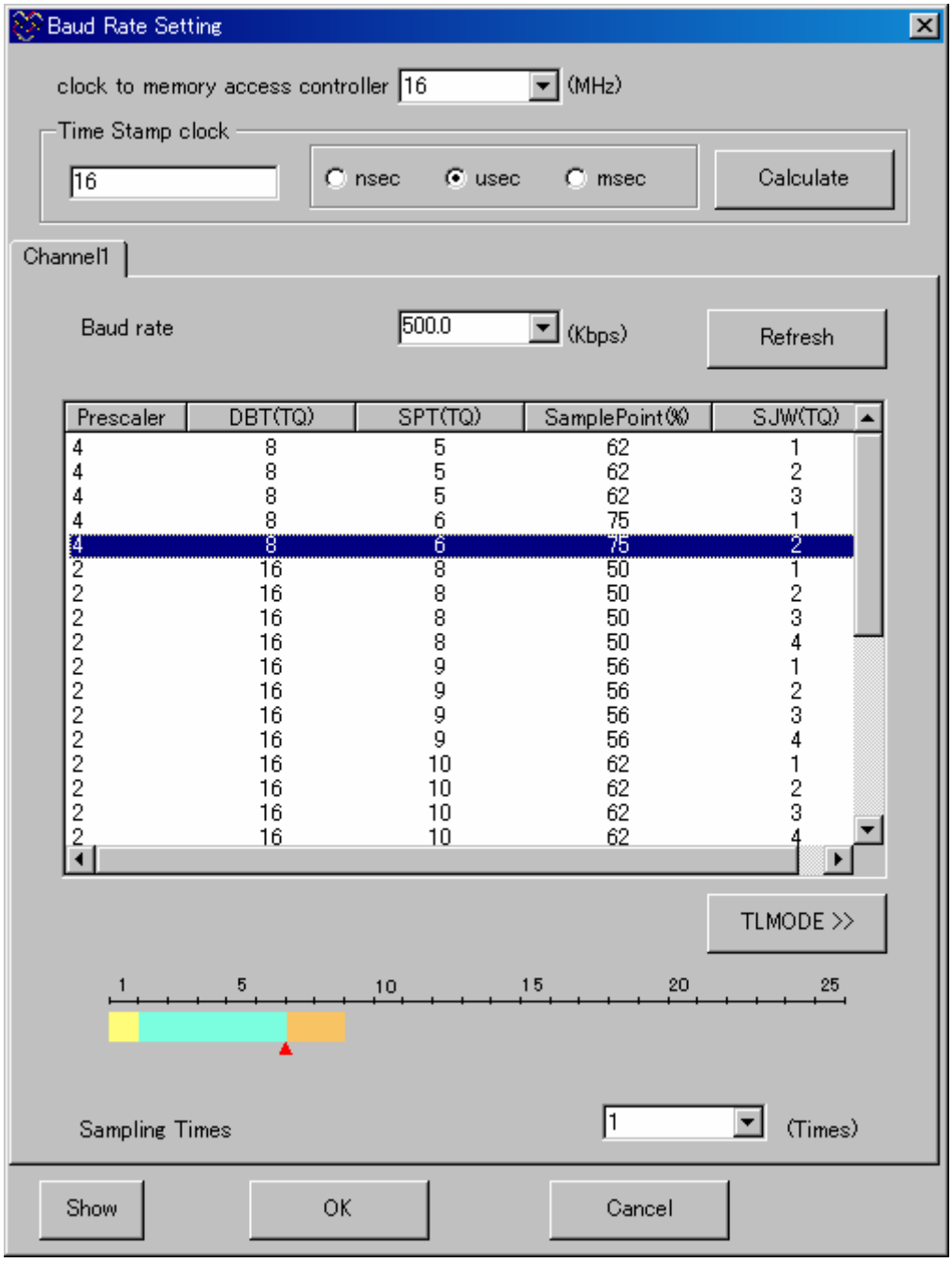
  

ボタンを押下することで，選択していたデータ・ビット・タイムが非選択状態になり，またソート機能がディフォールト状態に戻ります。

- サンプリング回数の選択

  プルダウン・メニューから選択します。サンプリング・ポイントで 1 回サンプリングか，受信データを 3 回サンプリングし，多数決で決定するかを選択します。

★ **図4－4　ボー・レート設定画面（V850-FCAN）**



＜設定項目＞（78K0-DCAN）

- CAN モジュール・システム・クロックの選択

  プルダウン・メニューから選択します。このモジュール・システム・クロックの設定により，選択できるボー・レートが決まります。

  外部クロックを使用する場合は 

をチェックして，外部クロック周波数を直接モジュール・システム・クロックへ入力してください。

- ボー・レートの選択

  使用する CAN バス条件に合うボー・レートをプルダウン・メニューから選択します。

  プルダウン・メニューには使用頻度の高いボー・レートが表示されています。該当するボー・レートがない場合は，直接，値を入力します。入力後に 

ボタンを押下すると自動計算を行います。

- データ・ビット・タイムの設定

  ビット・タイムはコンフィギュレータで自動的に算出されます。該当する組み合わせのビット・タイムをリスト上から選択します。

  各項目（Prescaler/DBT/SPT/SamplePoint/SJW）の項目欄を押下することにより，ソート機能が働きます。また，リスト下部にサンプル・ポイント（赤い三角印）を中心としたビット・タイムがグラフィカルに表示されます。

ボタンを押下することで TL モードを使用してより詳細な設定が可能になります。

ボタンを押下すると，レジスタに設定される値が表示されます。

- 設定状態の初期化

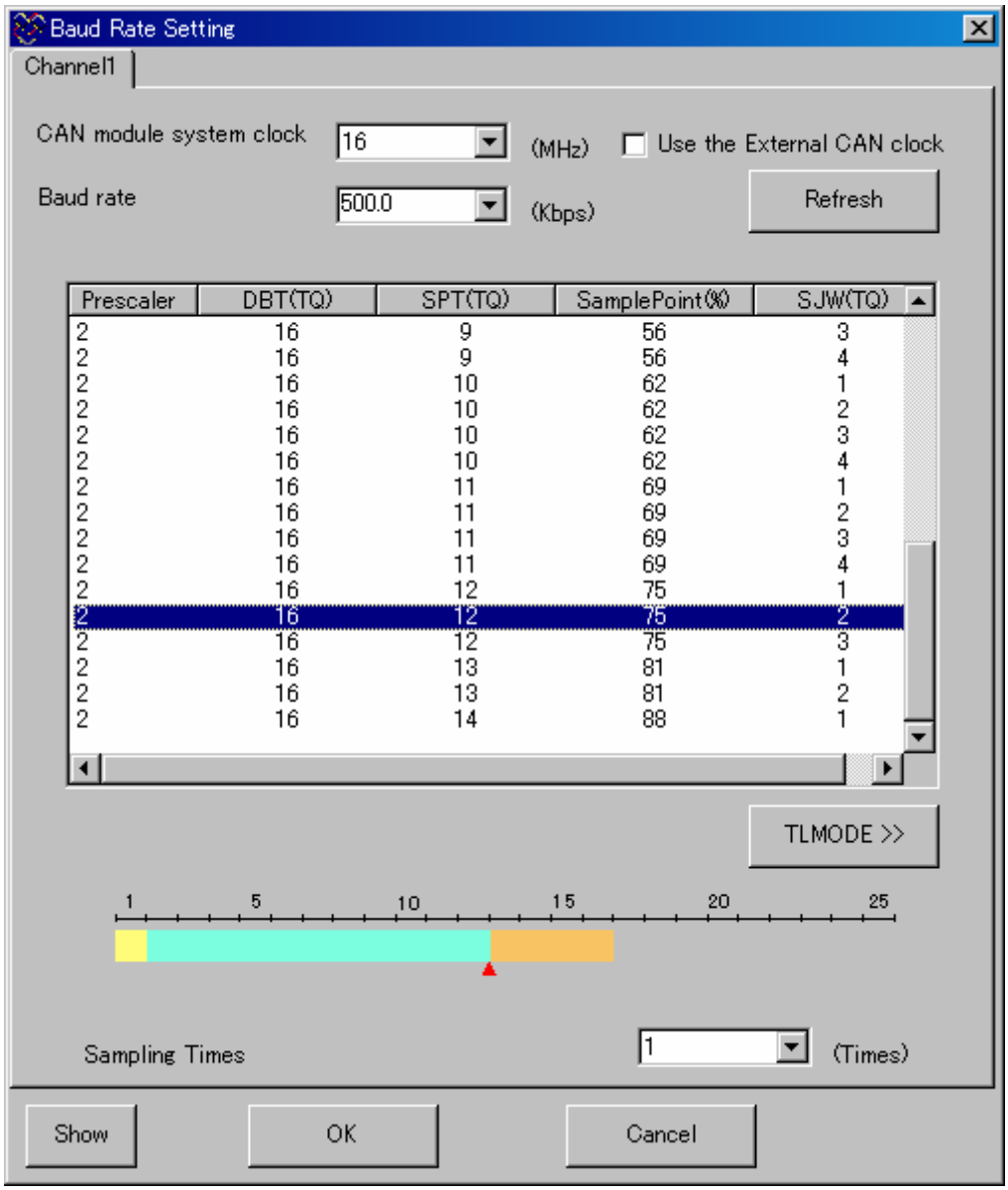
  

ボタンを押下することで，選択していたデータ・ビット・タイムが非選択状態になり，またソート機能がディフォールト状態に戻ります。

- サンプリング回数の選択

  プルダウン・メニューから選択します。サンプリング・ポイントで受信データを 1 回サンプリングするか，3 回サンプリングし，多数決で決定するかを選択します。

★ **図4－5　ボー・レート設定画面（78K0-DCAN）**



4.4.3　マスクの設定

チャネルごとのマスク設定をします。

各受信メッセージ・バッファとマスクのリンクは，メッセージ・バッファの設定項目内に存在します。

＜起動方法＞

- メニューの[Tool] – [Mask Setup]から起動

＜設定項目＞　（V850-aFCAN, 78K0-aFCAN）

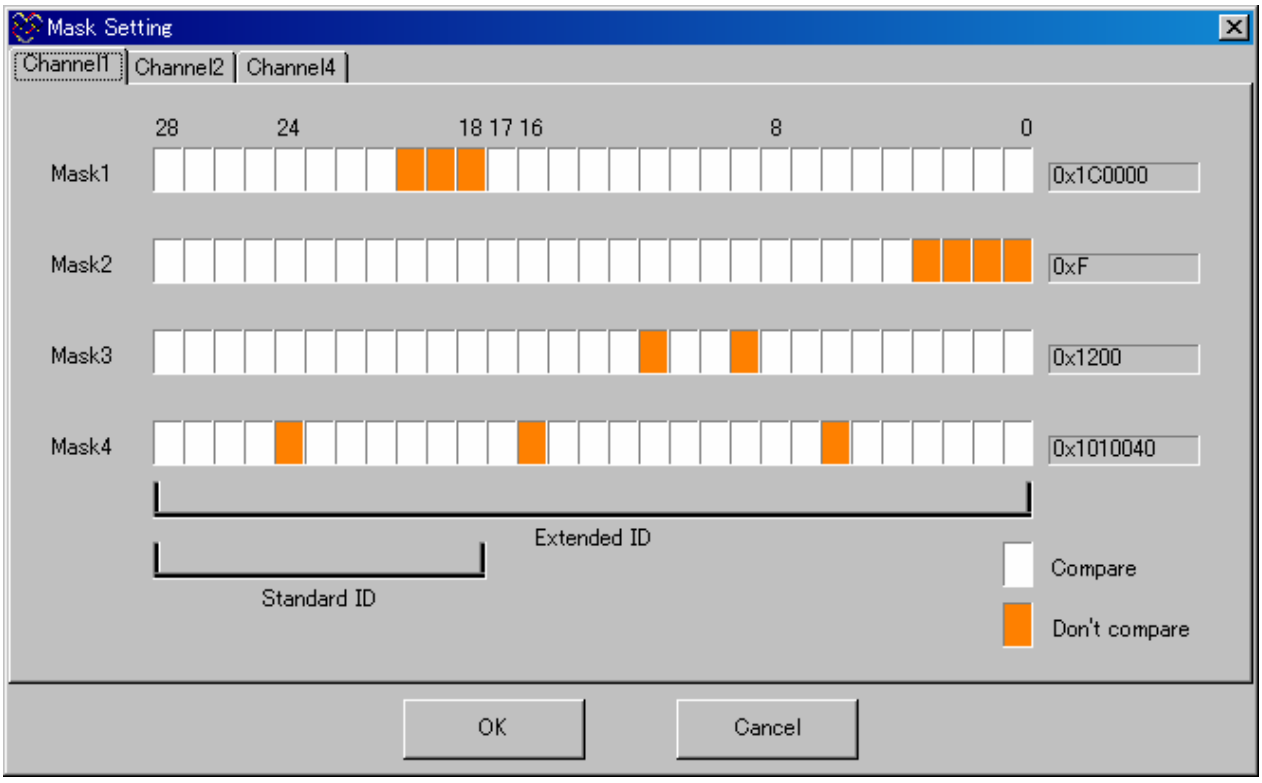
- マスクの設定

  マスク・レジスタはビット・イメージで表示されています。

  受信したメッセージ・バッファの ID とメッセージ・バッファの ID を比較しない（マスクする）場合，ビットをセットします。

備考　マスク機能： 複数の CAN-ID を 1 つのメッセージ・バッファに格納することができるので，CAN-ID をグループで管理する際に便利な機能です。

図4－6　マスク設定画面（V850-aFCAN, 78K0-aFCAN**）**



＜設定項目＞　(V850-FCAN)

- マスクの設定

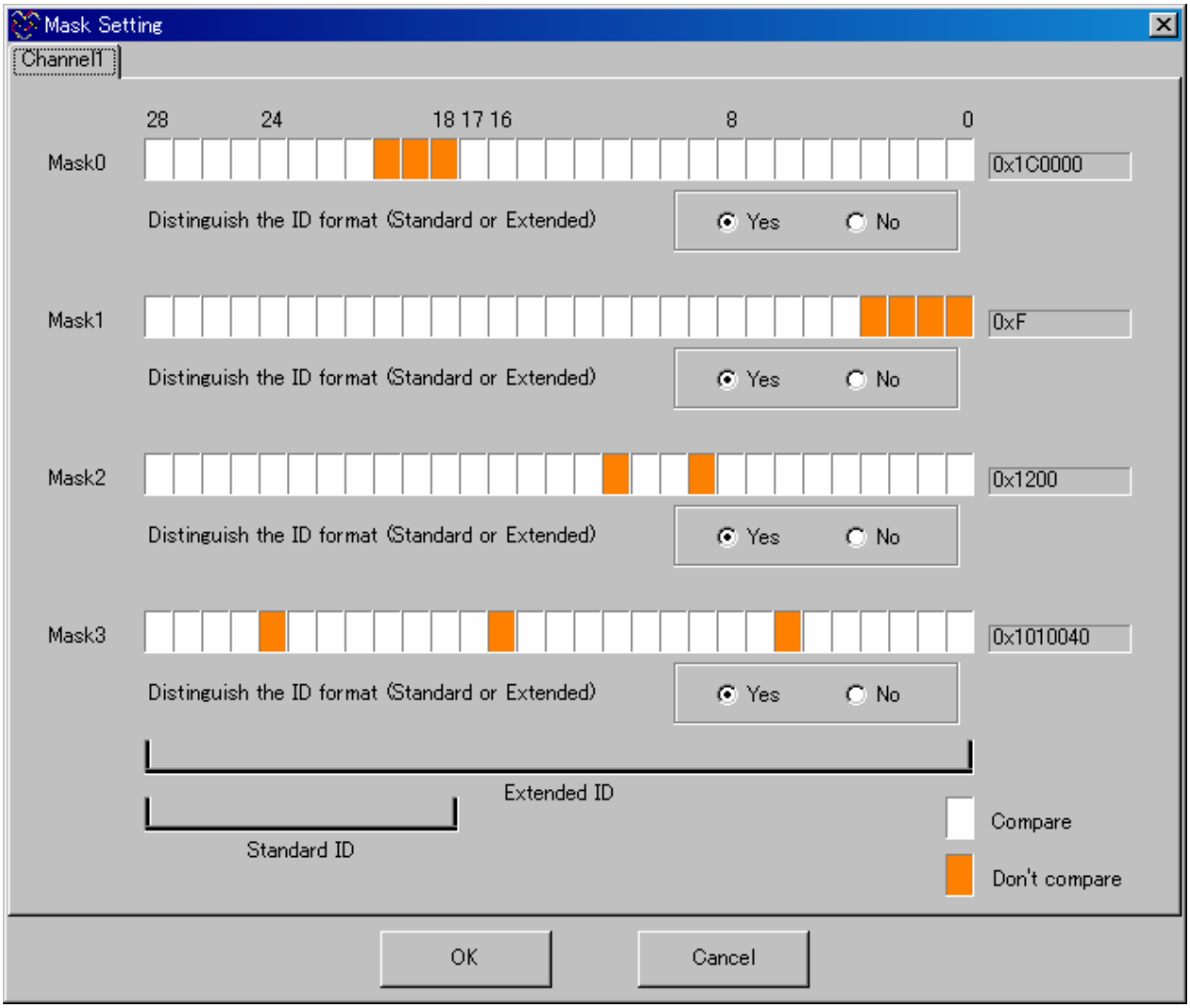
  マスク・レジスタはビット・イメージで表示されています。

  受信したメッセージ・バッファの ID とメッセージ・バッファの ID を比較しない（マスクする）場合，ビットをセットします。

- アイデンティファイア（ID）形式のマスク設定

  マスク・レジスタ毎に Yse No ボタンで ID 形式（標準または拡張）のチェックをする，しないを選択します。

図 4－7　マスク設定画面（V850-FCAN）



＜設定項目＞　（V850-DCAN, 78K0-DCAN）

- マスクの設定

  マスク・レジスタはビット・イメージで表示されています。

  受信したメッセージ・バッファの ID とメッセージ・バッファの ID を比較しない（マスクする）場合，ビットをセットします。

- フレーム形式のマスク設定

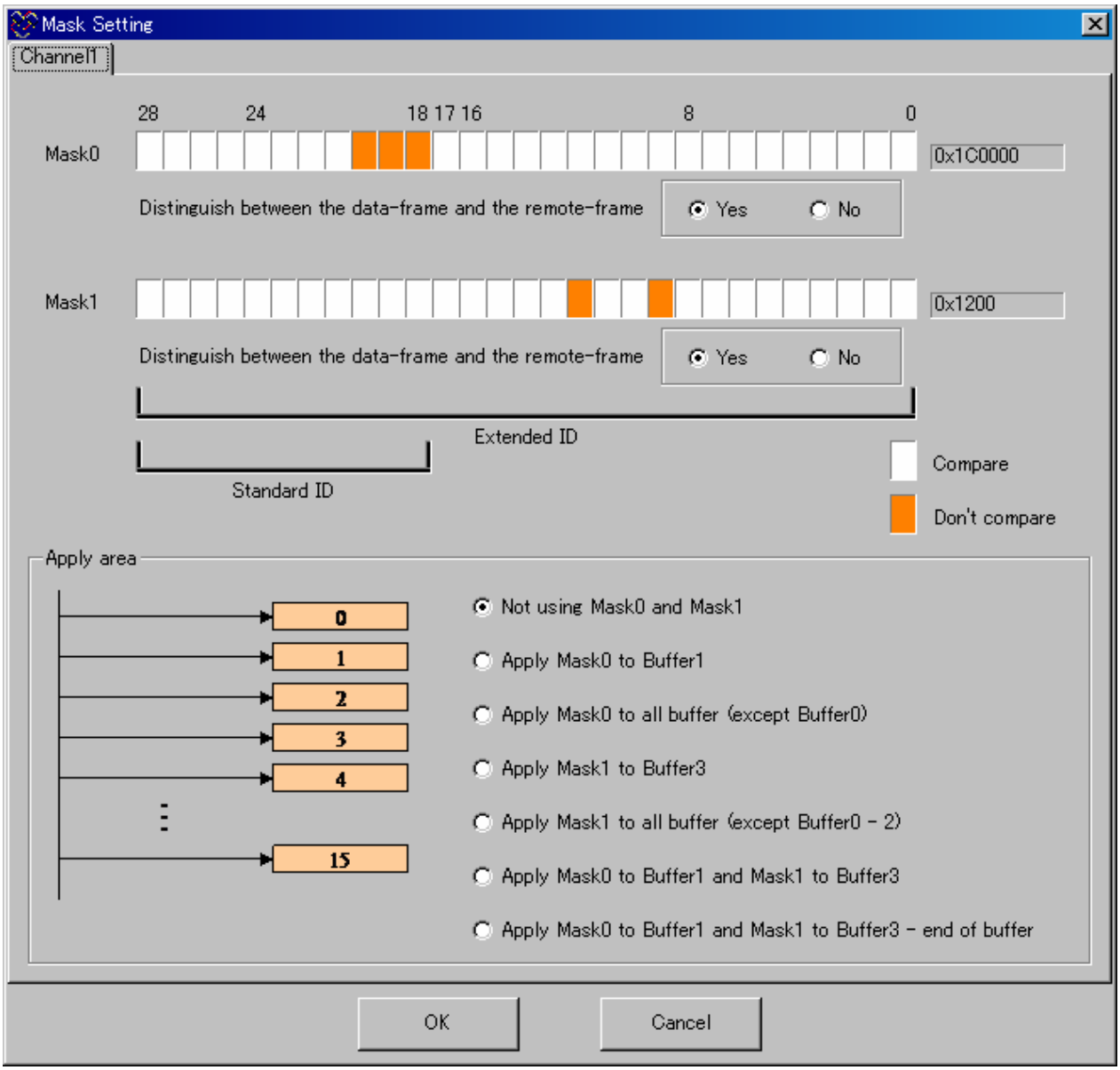
  マスク・レジスタ毎に 

ボタンでフレーム形式（データ・フレームまたはリモート・フレーム）のチェックをする，しないを選択します。

- グローバル・マスク機能の選択

  マスク・レジスタの使用，未使用，グローバル・マスクの選択をします。ボタンをクリックすると左下部分にマスクのイメージが表示されます。

図4－8　マスク設定画面（V850-DCAN, 78K0-DCAN）



4.4.4　メッセージ・バッファの設定

各チャネルのメッセージ・バッファを送信と受信に割り振り，各メッセージのID，フレーム・タイプなどの設定を行います。

(1) メッセージ・バッファの割り振り

＜起動方法＞

- メニューの[Tool] – [Message Buffer Setup]から起動

＜設定項目＞　(V850-aFCAN, 78K0-aFCAN)

- 未使用メッセージ・バッファの送信／受信機能への振り分け
  メッセージ・バッファの未使用リスト欄（No Used Buffer list）から << ボタンで送信側のリスト（Tx Message）あるいは受信側のリスト（Rx Message）に移動させます。
- 自動ブロック送信（ABT）機能[注]の使用有無の設定
  ABT機能を使用する場合は，Use側のラジオ・ボタンを選択します。
  これにより，メッセージ・バッファの0～7が送信メッセージ・バッファとしか使えないようにガードをかけておくことができます。

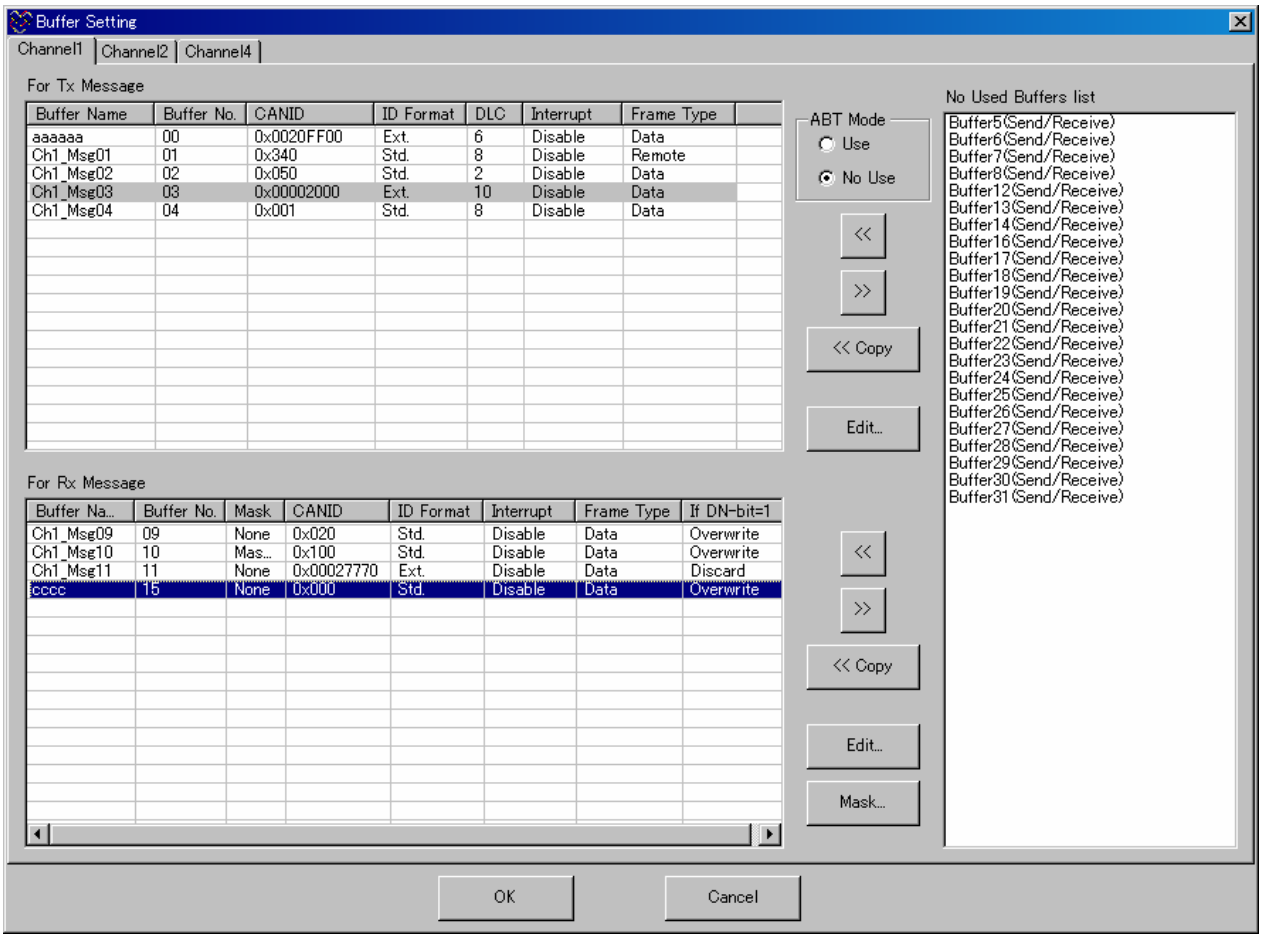
  注　自動ブロック送信（ABT）機能は，CPUを介さずに複数のデータ・フレームを連続的に送信することができる機能です。ABT用に割り付けられる送信メッセージ・バッファ数は，メッセージ・バッファ0からメッセージ・バッファ7までの8メッセージ固定です。
  なお，ABTモードは，標準ではCANソフトウエア・ドライバの機能には含まれていないため，ユーザ自身でコーディングする必要があります。

- マスク機能の設定
  4.4.3　マスクの設定のマスク設定機能が Mask ボタン押下でも設定できます。

＜その他の機能＞

- 割り振ったメッセージ・バッファを未使用メッセージに戻す方法
  Tx（Rx）リスト欄のメッセージを選択して >> ボタンを押下すると，再び未使用リスト欄にメッセージが戻ります。
- メッセージコピー機能
  未使用リスト欄のメッセージ・バッファを既に設定したメッセージ・バッファの設定と同じ内容にして割り振りたい場合は，未使用リスト欄のメッセージと，コピーしたいメッセージの両方を選択し << Copy ボタンを押下します。

図4－9　メッセージ・バッファ設定画面（V850-aFCAN, 78K0-aFCAN**）**



＜設定項目＞（V850-FCAN）

- 未使用メッセージ・バッファの送信／受信機能への振り分け

  メッセージ・バッファの未使用リスト欄（List of Unused Buffers）から << ボタンで送信側のリスト（Tx Message）あるいは受信側のリスト（Rx Message）に移動させます。

- 送信時のプライオリティ制御の選択

  アイデンティファイア（ID）によるプライオリティ制御とメッセージ番号によるプライオリティ制御をラジオ・ボタンによって選択します。

- オーバライト・モードの選択

  すでに受信しているメッセージ・バッファに対して新しいメッセージを受信した場合，上書きするか，上書きしないで破棄するかの選択を行います。Overwrite（ディフォールト）が選択されていると，メッセージが上書きされます。

  なお，Discard を選択した場合は，すでに受信しているメッセージ・バッファ[注]に対して新しく受信したメッセージは，上書しないで破棄します。

  注　DN ビットがセット“1”されているメッセージ・バッファを意味します。

- マスク機能の設定

  4. 4. 3　マスクの設定のマスク設定機能が Mask ボタン押下でも設定できます。

＜その他の機能＞

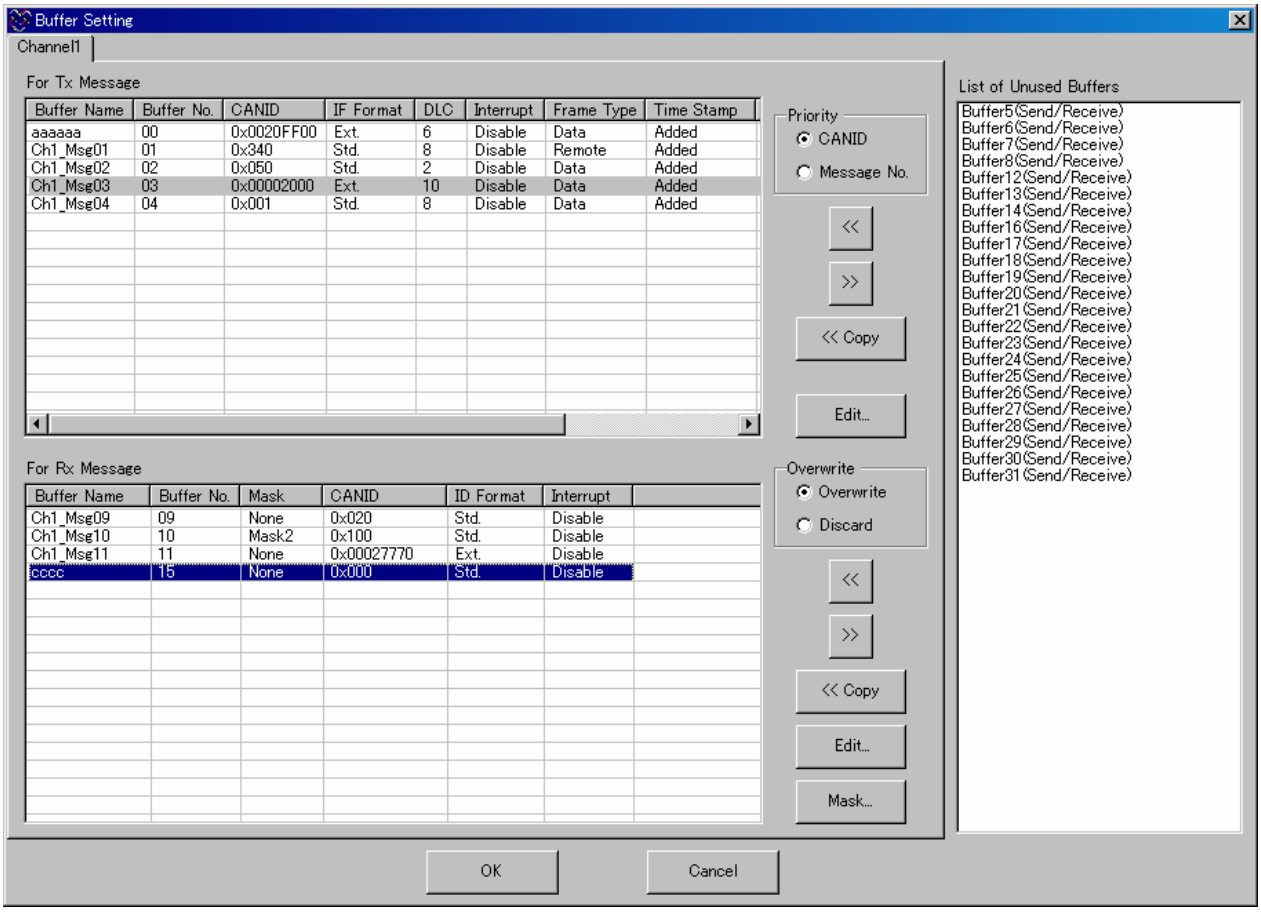
- 割り振ったメッセージ・バッファを未使用メッセージに戻す方法

  Tx（Rx）リスト欄のメッセージを選択して >> ボタンを押下すると，再び未使用リスト欄にメッセージが戻ります。

- メッセージコピー機能

  未使用リスト欄のメッセージ・バッファを既に設定したメッセージ・バッファの設定と同じ内容にして割り振りたい場合は，未使用リスト欄のメッセージと，コピーしたいメッセージの両方を選択し << Copy ボタンを押下します。

図4－10　メッセージ・バッファ設定画面（V850-FCAN）



＜設定項目＞　（V850-DCAN, 78K0-DCAN）

- 送信時のプライオリティ制御の選択

  優先送信メッセージ・バッファを送信バッファ 0 または 1 のどちらにするか，ラジオ・ボタンによって選択します。尚，送信メッセージ・バッファ数は 2 つに固定されています。

- 使用する受信メッセージ・バッファ数の登録

  使用する受信メッセージ・バッファ数（Using buffer number）を直接入力するか，プルダウン・メニューから選択し，Refresh ボタン押下で受信側のリスト（Rx Message）へ登録します。

- マスク機能の設定

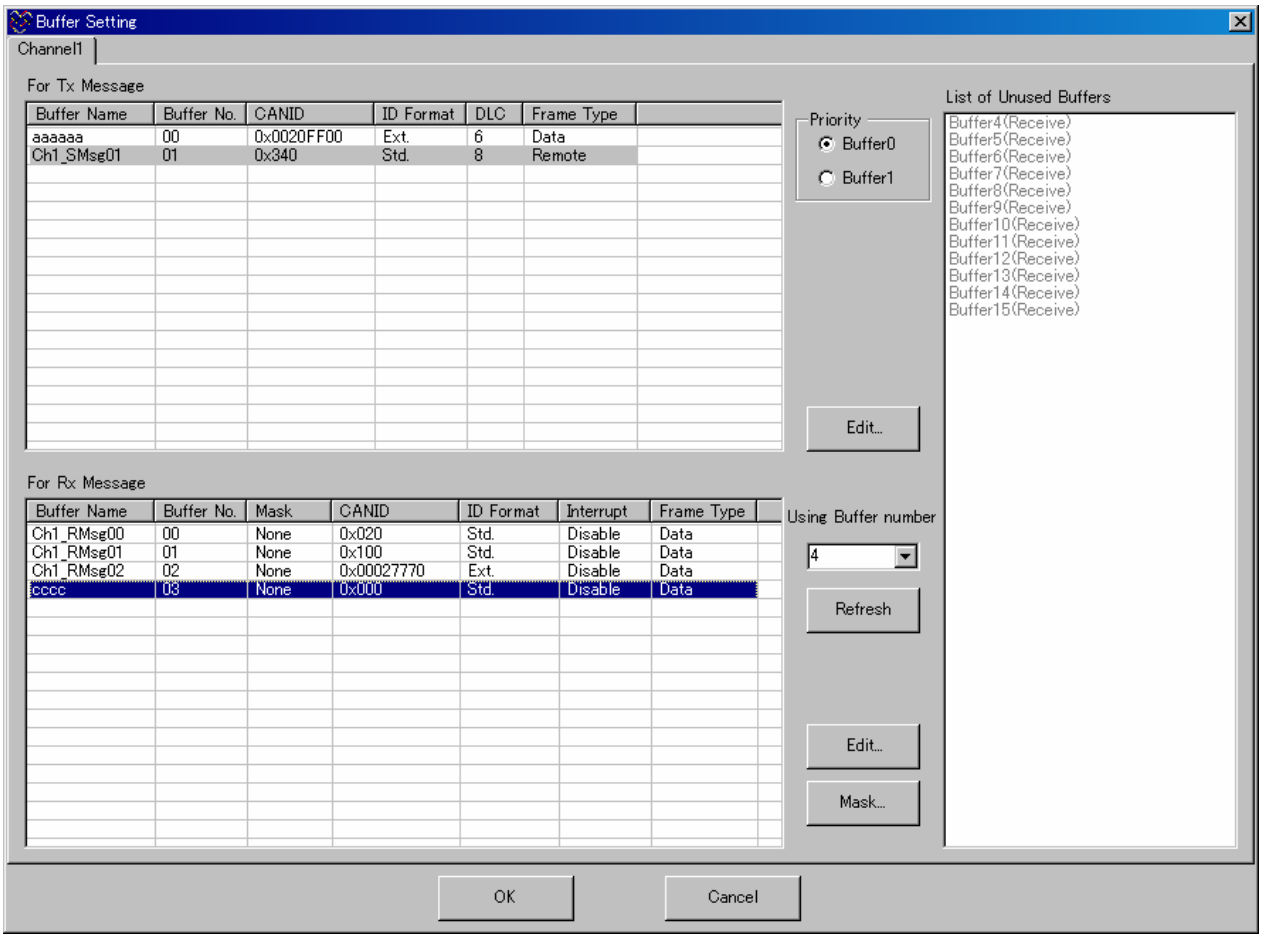
  4. 4. 3　マスクの設定のマスク設定機能が Mask ボタン押下でも設定できます。

＜その他の機能＞

- 登録した受信メッセージ・バッファを未使用メッセージに戻す方法

  使用する受信メッセージ・バッファ数（Using buffer number）を減らして Refresh ボタンを押下すれば，再び未使用リスト欄にメッセージが戻ります。

図 4－11　メッセージ・バッファ設定画面（V850-DCAN, 78K0-DCAN）



(2) メッセージ・バッファの設定

＜起動方法＞

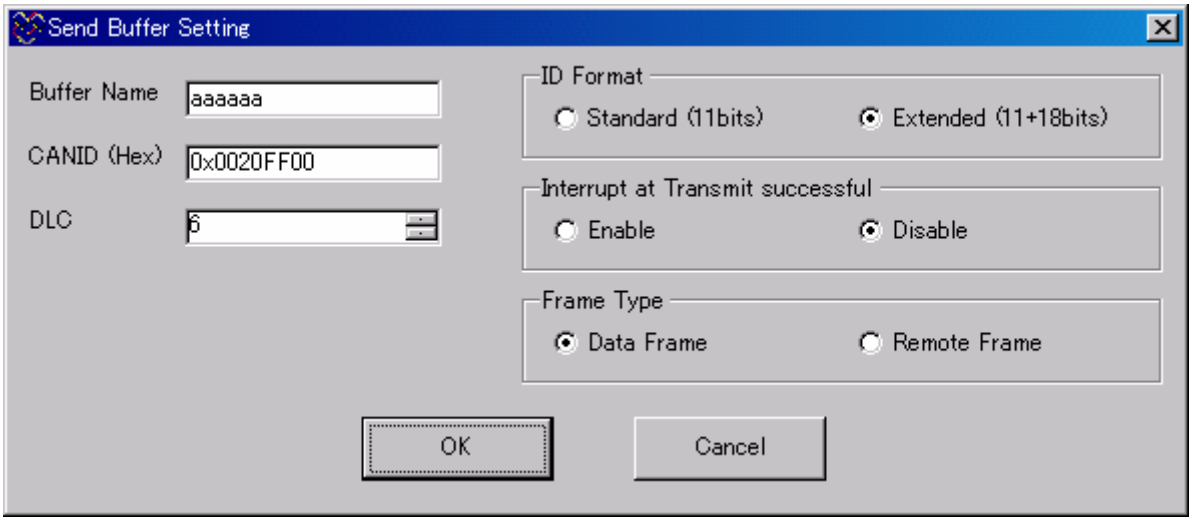
(1) のメッセージ・バッファの割り振りを行う画面から起動します。

- メッセージ・バッファを選択した状態で Edit ボタンを押下することで起動
- メッセージ・バッファをダブル・クリックすることで起動

＜設定項目＞ (V850-aFCAN, 78K0-aFCAN)

(a) 送信メッセージ・バッファの設定項目

図4－12　送信メッセージ・バッファの設定画面（V850-aFCAN, 78K0-aFCAN）



- バッファ名

Buffer Name 欄にメッセージ・バッファ名称を入力します。

あらかじめディフォールトのメッセージ・バッファ名が入っています。

ユーザは，ここで設定したメッセージ・バッファ名を CAN ソフトウエア・ドライバ関数（API）の引数（メッセージ・バッファ指定）として使用します。

注意　次のメッセージ名は，コンフィギュレータで予約済みです。メッセージ名を入力する際は次の予約済み以外の名称を使用してください。

ChX_MsgYY　(X : 1～6　AND　Y : 00～63)

- アイデンティファイアの設定

CANID（アイデンティファイア）を 16 進数で入力します。先頭に“0x”はコンフィギュレータでも自動的に付加されます。

ディフォールトで 0x000 の値が入力されています。

また，設定範囲はフレーム・フォーマットにより異なります。

- **データ長の設定**

  DLC **の欄に** CAN **メッセージのデータ長を指定します。直接に入力しても上下ボタンを押下しても指定することが可能です。**

  指定は，10 **進数で，指定範囲は** 0～15 **バイトです。**

  なお，8 **バイトを越えるデータ長の指定をした場合は，注意を促すメッセージが出ます。**

  備考　8 **バイト以上のデータ長を設定した場合,実際に** CAN **バスに送信されるデータ長はここで設定した値になります（初期設定時)。ただし，送信されるデータ長は最大** 8 **バイトです。**

- ID **フォーマットの選択**

  ID Format **の欄で，標準** ID（Standard）**か拡張** ID（Extended）**かの選択を行います。**

  ディフォールトは標準 ID **が選択されています。**

- **送信完了割り込みの許可／禁止の選択**

  メッセージの送信完了割り込みの許可／禁止の選択を行います。

  割り込み許可を選択すると，割り込み要求が発生します。

  ディフォールトは禁止が選択されています。

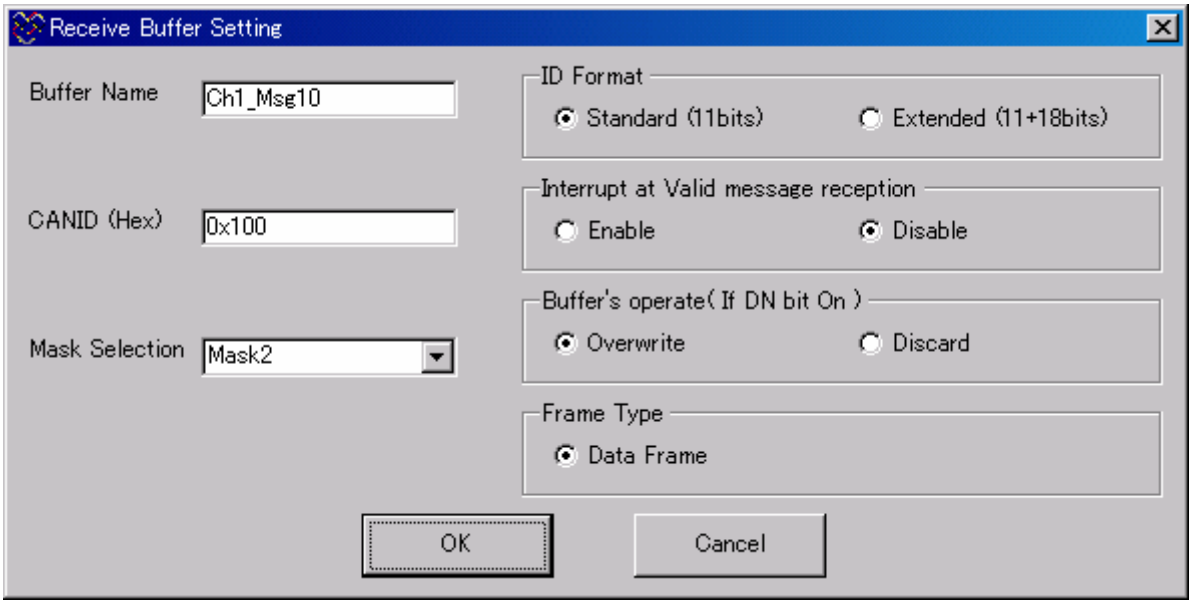
- **フレーム・タイプの選択**

  データ・フレームか，リモート・フレームかの選択を行います。

  ディフォールトはデータ・フレームが選択されています。

(b) 受信メッセージ・バッファの設定項目

図4－13　受信メッセージ・バッファの設定画面（V850-aFCAN, 78K0-aFCAN）



- **バッファ名**

  Buffer Name 欄にメッセージ・バッファ名称を入力します。

  あらかじめディフォールトのメッセージ・バッファ名が入っています。

  ユーザは，ここで設定したメッセージ・バッファ名を CAN ソフトウエア・ドライバ関数（API）の引数（メッセージ・バッファ指定）として使用します。

  注意　次のメッセージ名は，コンフィギュレータで予約済みです。メッセージ名を入力する際は次の予約済み以外の名称を使用してください。

  ChX_MsgYY　(X : 1～6　AND　Y : 00～63)

- **アイデンティファイアの設定**

  CANID（アイデンティファイア）を 16 進数で入力します。先頭に“0x”はコンフィギュレータでも自動的に付加されます。

  ディフォールトで 0x000 の値が入力されています。

  また，設定範囲はフレーム・フォーマットにより異なります。

- **マスク設定のリンク**

  マスク機能を使用する場合，4. 4. 3　マスクの設定で設定したマスク 1～4 のいずれか設定をリンクさせます。

  Mask Selection のプルダウン・メニューから選択します。

  ディフォールトは，マスク設定のリンクなし（None）が選択されています。

- ID フォーマットの選択

  ID Format の欄で，標準 ID（Standard）か拡張 ID（Extended）かの選択を行います。

  ディフォールトは標準 ID が選択されています。

- 受信完了割り込みの許可／禁止の選択

  メッセージの受信完了割り込みの許可／禁止の選択を Interrupt at Valid message reception の欄で行います。

  割り込み許可を選択すると，割り込み要求が発生します。

  ディフォールトは禁止が選択されています。

- データ・フレームの上書き選択

  すでに受信しているメッセージ・バッファに対して新しいメッセージを受信した場合，上書きするか，上書きしないで破棄するかの選択を行います。Buffer's operate の Overwrite（ディフォールト）が選択されていると，メッセージが上書きされます。

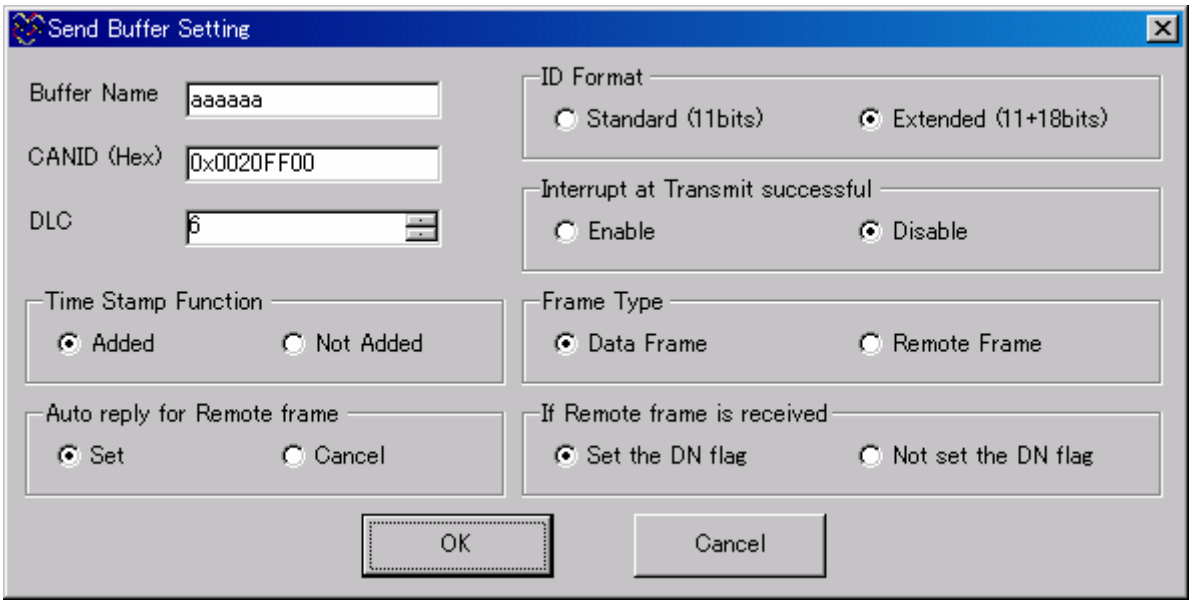
  なお，Overwrite の選択を外した場合は，すでに受信しているメッセージ・バッファ[注]に対して新しく受信したデータ・フレームは，上書しないで破棄します。

  注　DN ビットがセット“1”されている受信メッセージ・バッファを意味します。

＜設定項目＞　（V850-FCAN）

（a）送信メッセージ・バッファの設定項目

図4－14　送信メッセージ・バッファの設定画面（V850-FCAN）



- バッファ名

  Buffer Name 欄にメッセージ・バッファ名称を入力します。

  あらかじめディフォールトのメッセージ・バッファ名が入っています。

  ユーザは，ここで設定したメッセージ・バッファ名を CAN ソフトウエア・ドライバ関数（API）の引数（メッセージ・バッファ指定）として使用します。

  注意　次のメッセージ名は，コンフィギュレータで予約済みです。メッセージ名を入力する際は次の予約済み以外の名称を使用してください。

  ChX_MsgYY　(X : 1〜6　AND　Y : 00〜63)

- アイデンティファイアの設定

  CANID（アイデンティファイア）を 16 進数で入力します。先頭に“ 0x ”はコンフィギュレータでも自動的に付加されます。

  ディフォールトで 0x000 の値が入力されています。

  また，設定範囲はフレーム・フォーマットにより異なります。

- **データ長の設定**

  DLC **の欄に** CAN **メッセージのデータ長を指定します。直接に入力しても上下ボタンを押下しても指定することが可能です。**

  指定は，10 **進数で，指定範囲は** 0〜15 **バイトです。**

  なお，8 **バイトを越えるデータ長の指定をした場合は，注意を促すメッセージが出ます。**

  備考　8 **バイト以上のデータ長を設定した場合，実際に** CAN **バスに送信されるデータ長はここで設定した値になります（初期設定時）。ただし，送信されるデータ長は最大** 8 **バイトです。**

- **送信時のタイム・スタンプの付加の指定**

  Time Stamp Function **の欄で，送信時にタイム・スタンプを付加するか，しないかの選択を行います。**

  ディフォールトは付加する，が選択されています。

- **リモート・フレーム自動応答機能の設定／解除を指定**

  Auto reply for Remote frame **の欄で，リモート・フレーム自動応答機能の設定／解除の選択を行います。ディフォールトは設定が選択されています。**

- ID **フォーマットの選択**

  ID Format **の欄で，標準** ID（Standard）**か拡張** ID（Extended）**かの選択を行います。**

  ディフォールトは標準 ID **が選択されています。**

- **送信完了割り込みの許可／禁止の選択**

  メッセージの送信完了割り込みの許可／禁止の選択を行います。

  割り込み許可を選択すると，割り込み要求が発生します。

  ディフォールトは禁止が選択されています。

- **フレーム・タイプの選択**

  データ・フレームか，リモート・フレームかの選択を行います。

  ディフォールトはデータ・フレームが選択されています。

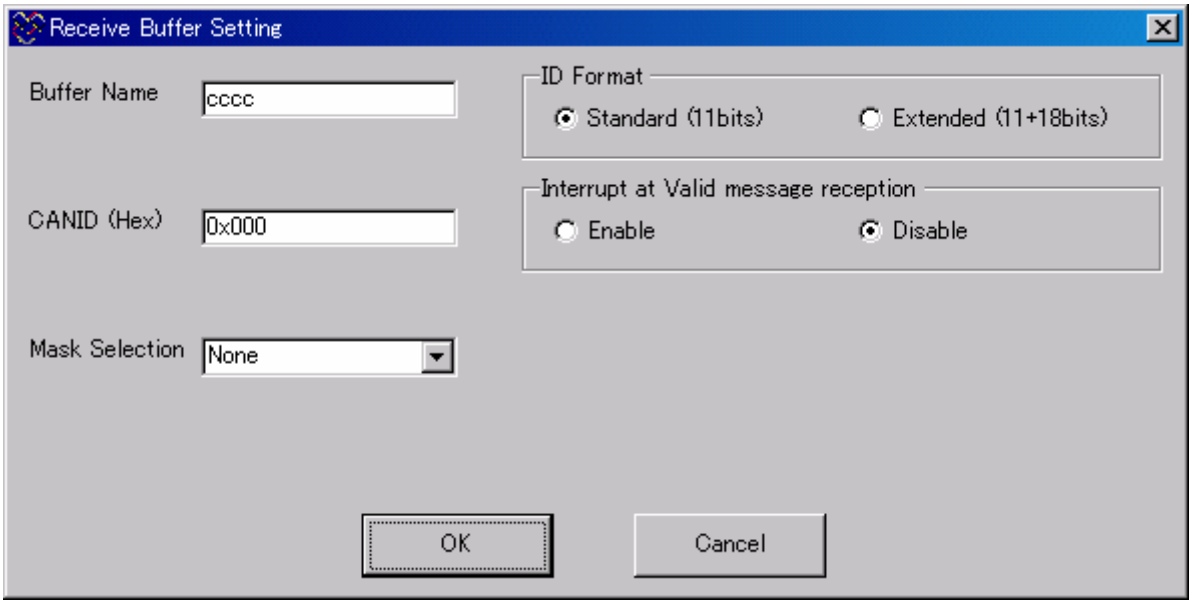
- **送信メッセージ・バッファ上にリモート・フレームを受信した場合の** DN **フラグの動作指定**

  リモート・フレームを受信した場合，DN **フラグをセットする，しないの選択を行います。**

  ディフォールトはセットするが選択されています。

(b) 受信メッセージ・バッファの設定項目

図4－15　受信メッセージ・バッファの設定画面（V850-FCAN）



- **バッファ名**

  Buffer Name 欄にメッセージ・バッファ名称を入力します。

  あらかじめディフォールトのメッセージ・バッファ名が入っています。

  ユーザは，ここで設定したメッセージ・バッファ名を CAN ソフトウエア・ドライバ関数（API）の引数（メッセージ・バッファ指定）として使用します。

  注意　次のメッセージ名は，コンフィギュレータで予約済みです。メッセージ名を入力する際は次の予約済み以外の名称を使用してください。

  ChX_MsgYY　(X : 1～6　AND　Y : 00～63)

- **アイデンティファイアの設定**

  CANID（アイデンティファイア）を 16 進数で入力します。先頭に“0x”はコンフィギュレータでも自動的に付加されます。

  ディフォールトで 0x000 の値が入力されています。

  また，設定範囲はフレーム・フォーマットにより異なります。

- **マスク設定のリンク**

  マスク機能を使用する場合，4. 4. 3　マスクの設定で設定したマスク 0～3 のいずれかをリンクさせます。診断処理モード時に使用する場合はDiagnosticを選択します。

  Mask Selection のプルダウン・メニューから選択します。

  ディフォールトは，マスク設定のリンクなし（None）が選択されています。

- ID フォーマットの選択

  ID Format の欄で，標準 ID（Standard）か拡張 ID（Extended）かの選択を行います。

  ディフォールトは標準 ID が選択されています。

- 受信完了割り込みの許可／禁止の選択

  メッセージの受信完了割り込みの許可／禁止の選択を Interrupt at Valid message reception の欄で行います。

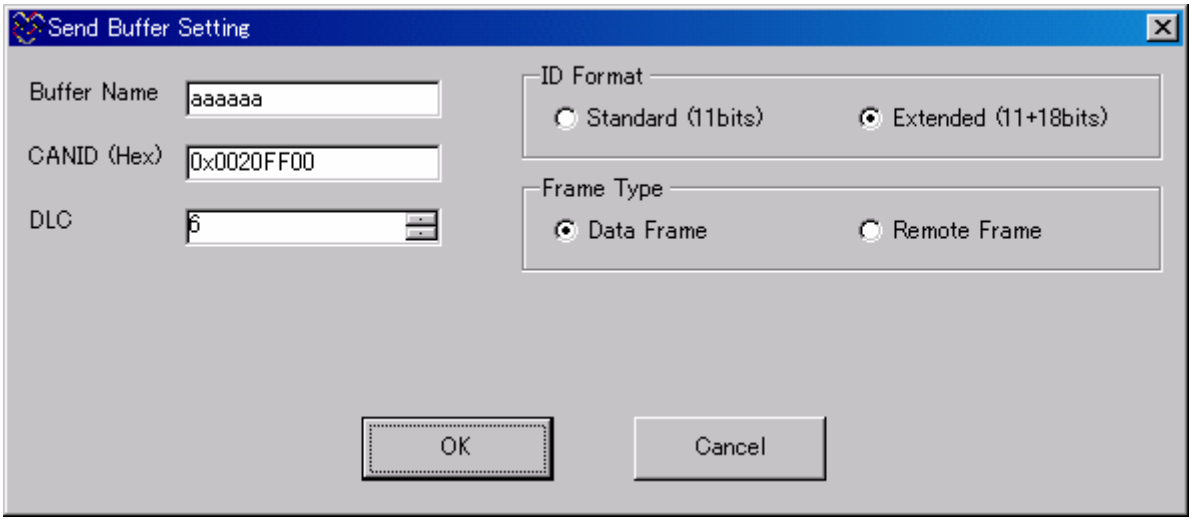
  割り込み許可を選択すると，割り込み要求が発生します。

  ディフォールトは禁止が選択されています。

＜設定項目＞　（V850-DCAN, 78K0-DCAN）

（a）送信メッセージ・バッファの設定項目

図4－16　送信メッセージ・バッファの設定画面（V850-DCAN, 78K0-DCAN）



- バッファ名

  Buffer Name 欄にメッセージ・バッファ名称を入力します。

  あらかじめディフォールトのメッセージ・バッファ名が入っています。

  ユーザは，ここで設定したメッセージ・バッファ名を CAN ソフトウエア・ドライバ関数（API）の引数（メッセージ・バッファ指定）として使用します。

  注意　次のメッセージ名は，コンフィギュレータで予約済みです。メッセージ名を入力する際は次の予約済み以外の名称を使用してください。

  ChX_MsgYY　(X : 1～6　AND　Y : 00～63)

- アイデンティファイアの設定

  CANID（アイデンティファイア）を 16 進数で入力します。先頭に“ 0x ”はコンフィギュレータでも自動的に付加されます。

  ディフォールトで 0x000 の値が入力されています。

  また，設定範囲はフレーム・フォーマットにより異なります。

- **データ長の設定**

  DLC **の欄に** CAN **メッセージのデータ長を指定します。直接に入力しても上下ボタンを押下しても指定することが可能です。**

  指定は，10 **進数で，指定範囲は** 0～15 **バイトです。**

  なお，8 **バイトを越えるデータ長の指定をした場合は，注意を促すメッセージが出ます。**

  備考　8 **バイト以上のデータ長を設定した場合，実際に** CAN **バスに送信されるデータ長はここで設定した値になります（初期設定時）。ただし，送信されるデータ長は最大** 8 **バイトです。**

- ID **フォーマットの選択**

  ID Format **の欄で，標準** ID（Standard）**か拡張** ID（Extended）**かの選択を行います。**

  ディフォールトは標準 ID **が選択されています。**

- **送信完了割り込みの許可／禁止の選択**

  メッセージの送信完了割り込みの許可／禁止の選択を行います。

  割り込み許可を選択すると，割り込み要求が発生します。

  ディフォールトは禁止が選択されています。

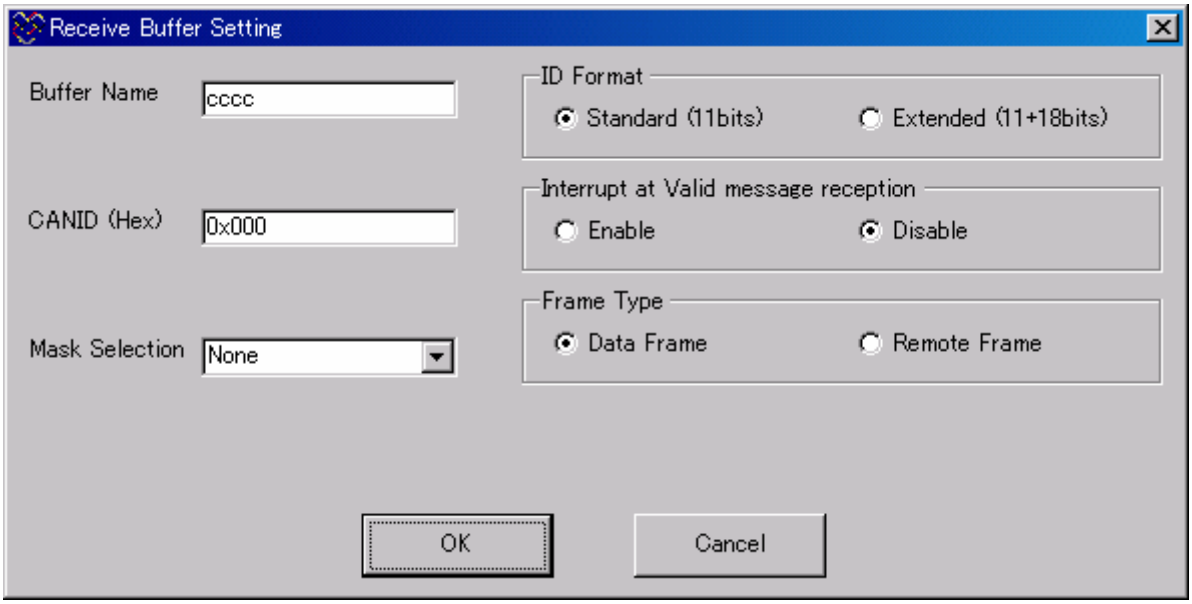
- **フレーム・タイプの選択**

  データ・フレームか，リモート・フレームかの選択を行います。

  ディフォールトはデータ・フレームが選択されています。

(b) 受信メッセージ・バッファの設定項目

図4－17　受信メッセージ・バッファの設定画面（V850-DCAN, 78K0-DCAN）



- バッファ名

  Buffer Name 欄にメッセージ・バッファ名称を入力します。

  あらかじめディフォールトのメッセージ・バッファ名が入っています。

  ユーザは，ここで設定したメッセージ・バッファ名を CAN ソフトウエア・ドライバ関数（API）の引数（メッセージ・バッファ指定）として使用します。

  注意　次のメッセージ名は，コンフィギュレータで予約済みです。メッセージ名を入力する際は次の予約済み以外の名称を使用してください。

  ChX_MsgYY　(X : 1～6　AND　Y : 00～63)

- アイデンティファイアの設定

  CANID（アイデンティファイア）を 16 進数で入力します。先頭に“0x”はコンフィギュレータでも自動的に付加されます。

  ディフォールトで 0x000 の値が入力されています。

  また，設定範囲はフレーム・フォーマットにより異なります。

- マスク設定のリンク

  4. 4. 3　マスクの設定で設定したマスク 1～2，Noneのいずれかがプルダウン・メニューに表示されます。このプルダウン・メニューからは変更できません。

- ID フォーマットの選択

  ID Format の欄で，標準 ID（Standard）か拡張 ID（Extended）かの選択を行います。

  ディフォールトは標準 ID が選択されています。

- **受信完了割り込みの許可／禁止の選択**
  メッセージの受信完了割り込みの許可／禁止の選択を Interrupt at Valid message reception **の欄で行います。**
  割り込み許可を選択すると，割り込み要求が発生します。
  ディフォールトは禁止が選択されています。

- **フレーム・タイプの選択**
  データ・フレームか，リモート・フレームかの選択を行います。
  ディフォールトはデータ・フレームが選択されています。

4.4.5　その他の設定

その他の CAN モジュール機能，モジュール割り込みの設定を行います。

＜起動方法＞

- メニューの[Tool] – [Other Setup]から起動します。

＜設定項目＞　（V850-aFCAN, 78K0-aFCAN）

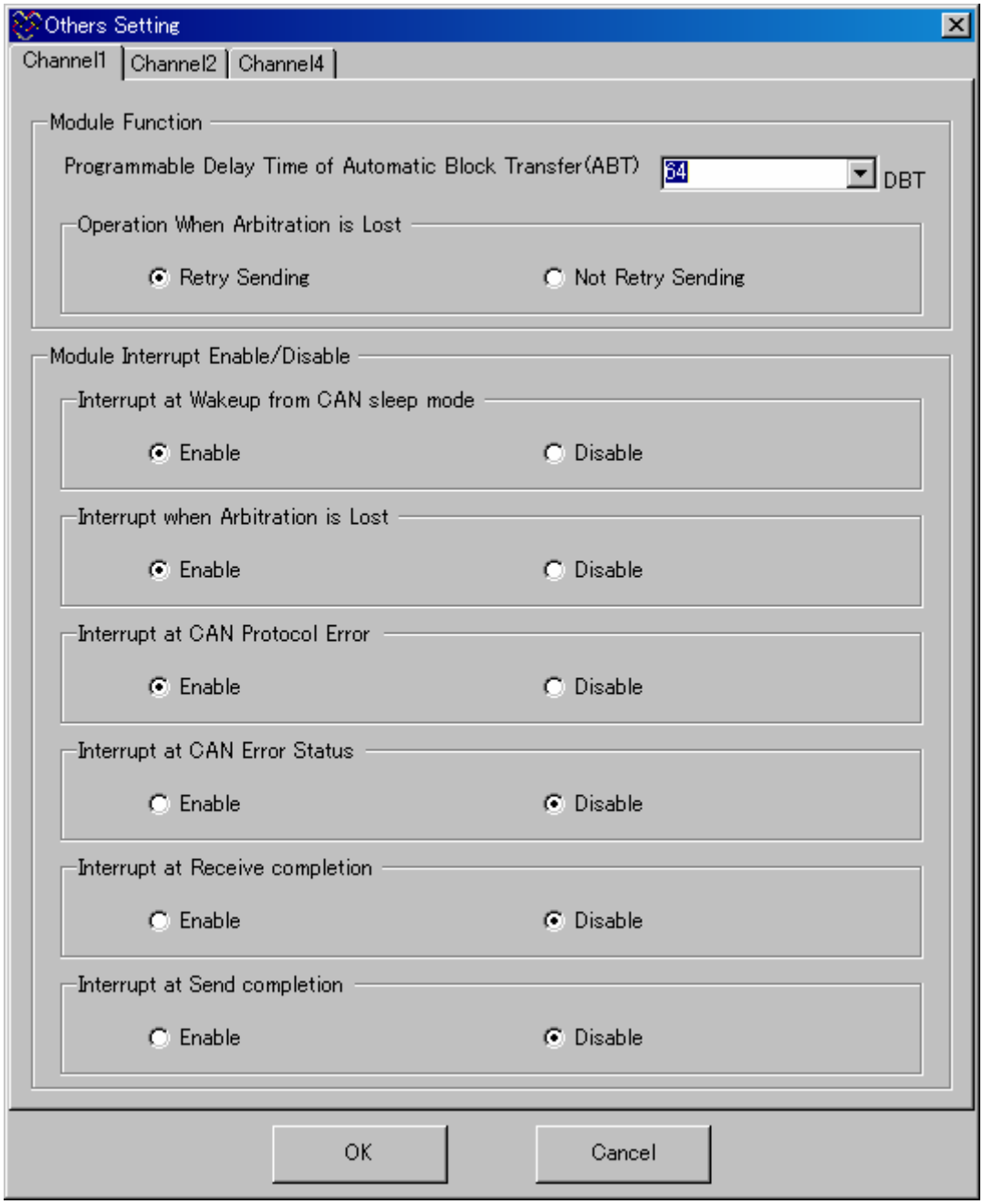
- 自動ブロック送信（ABT）の送信遅延時間の設定
  ABT 機能を使用する場合，送信遅延時間の設定をここで行います。
  設定単位はデータ・ビット・タイムになります。

- アービトレーション・ロスト発生時の動作設定
  シングル・ショット・モードにおいてアービトレーション・ロスト発生時，再送信するか，再送信しないかの選択を行います

- モジュール割り込みの許可／禁止
  次の各項目について，割り込みを許可するか禁止にするかの選択を行います。

  - スリープ・モードからのウエイクアップ割り込み
  - アービトレーション・ロスト発生割り込み
  - CAN プロトコル・エラー割り込み
  - CAN エラー・ステータス割り込み
  - メッセージ・バッファからの有効なメッセージ・フレームの受信完了割り込み
  - メッセージ・バッファからのメッセージ・フレームの正常な送信完了割り込み

図4－18　その他設定画面（V850-aFCAN, 78K0-aFCAN**）**



＜設定項目＞　（V850-FCAN）

- グローバル割り込みの許可／禁止

  次の各項目について，割り込みを許可するか禁止にするかの選択を行います。

  - 使用不可アドレスへのメモリ・アクセスの割り込み
  - 不正ライト・アクセス（テンポラリ・バッファなど）の割り込み

- タイム・スタンプ機能の使用／未使用の選択

  Time Stamp Function の欄で，タイム・スタンプ機能の使用／未使用の選択を行います。ディフォールトは未使用が選択されています。

- 送信端子のドミナント・レベルの選択

  Dominant of sending の欄で，送信端子からロウ・レベルをドミナントとして送信するか，ハイ・レベルをドミナントとして送信するかの選択を行います。ディフォールトはロウ・レベルが選択されています。

- 受信端子のドミナント・レベルの選択

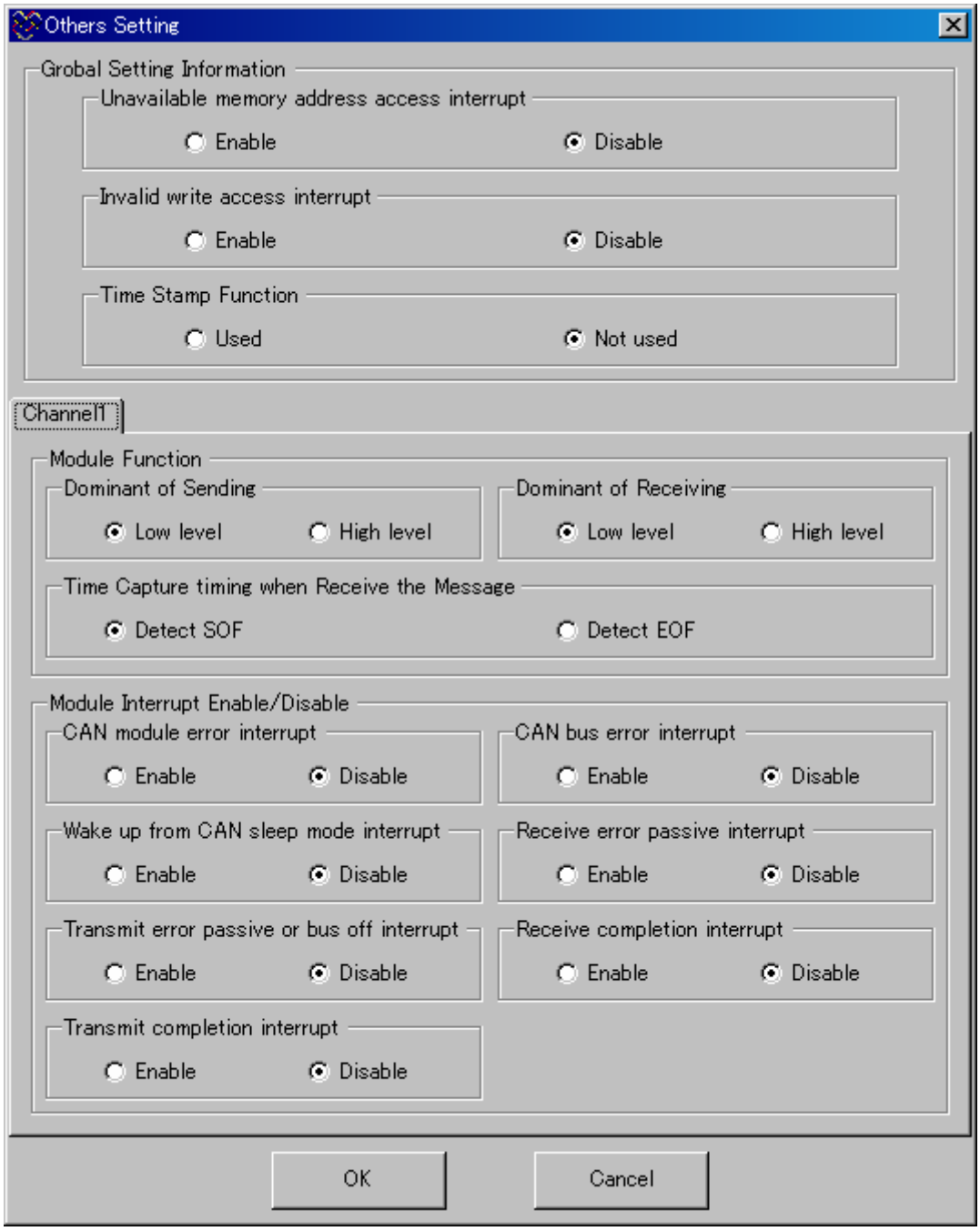
  Dominant of Receiving の欄で，受信端子へのロウ・レベルをドミナントとして認識するか，ハイ・レベルをドミナントとして認識するかの選択を行います。ディフォールトはロウ・レベルが選択されています。

- モジュール割り込みの許可／禁止

  次の各項目について，割り込みを許可するか禁止にするかの選択を行います。

  - CAN モジュール・エラー割り込み
  - CAN バス・エラー割り込み
  - スリープ・モードからのウエイクアップ割り込み
  - 受信エラー・パッシブ割り込み
  - 送信エラー・パッシブまたはバス・オフ割り込み
  - 受信完了割り込み
  - 送信完了割り込み

図4－19　その他設定画面（V850-FCAN）



＜設定項目＞　（V850-DCAN, 78K0-DCAN）

- アービトレーション・ロスト発生時の動作設定

  シングル・ショット・モードにおいてアービトレーション・ロスト発生時，再送信するか，再送信しないかの選択を行います

- タイム・スタンプ機能の使用／未使用の選択

  Use or not Function の欄で，タイム・スタンプ機能の使用／未使用の選択を行います。ディフォールトは未使用が選択されています。

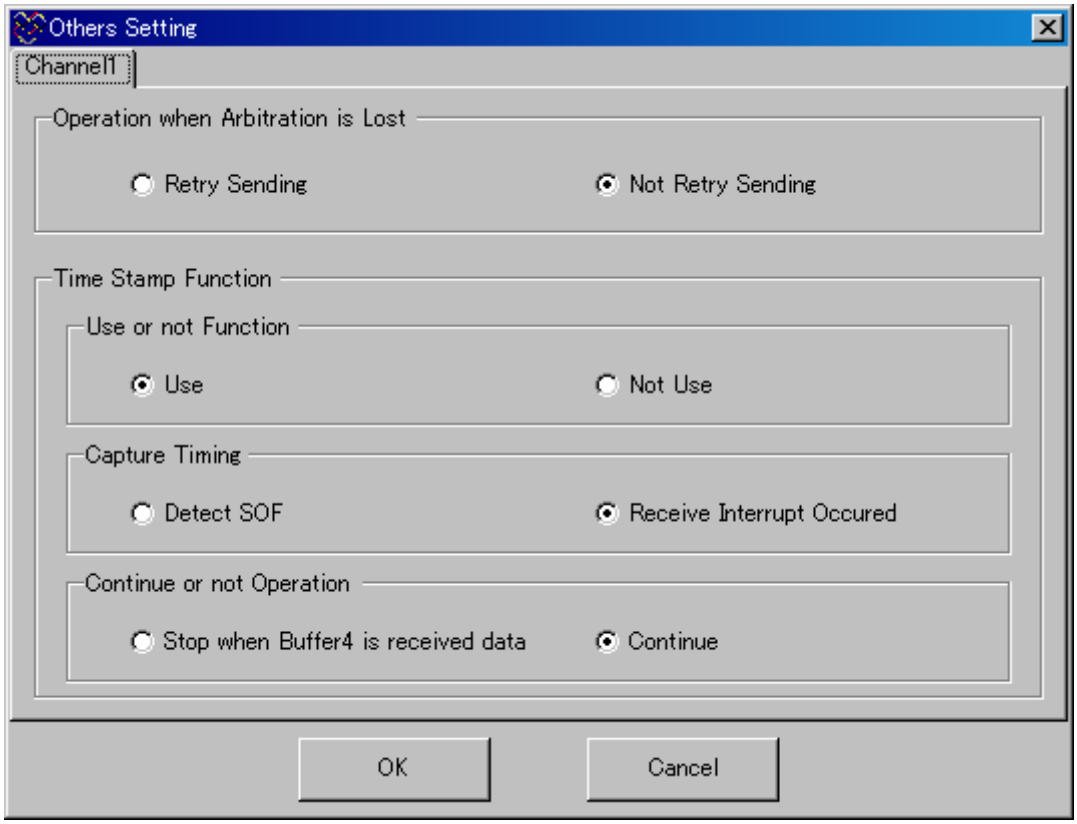
- SOFOUT 出力形式の選択

  Capture Timeing の欄で，SOFOUT を受信割り込みでトグル（タイム・スタンプ・モード）にするか，スタート・オブ・フレーム受信ごとにトグル（グローバル・タイム・モード）にするかを選択します。ディフォールトはグローバル・タイム・モードが選択されています。

- SOFOUT 機能（SOFEn フラグ動作）の選択

  Continue or not Operation の欄で，SOFEn ビットは CAN バス動作に依存しない（トグル動作継続）か，受信メッセージ・バッファ 4 へメッセージが格納されはじめたときに，SOFEn ビットをクリアする（トグル動作停止）かを選択します。ディフォールトはトグル動作継続が選択されています。

図4－20　その他設定画面（V850-DCAN, 78K0-DCAN**）**



4.4.6　コード生成

情報ファイル，コンフィギュレータ・ヘッダ・ファイルのソース・コードおよび CAN ソフトウエア・ドライバ・ソース・ファイル群，CAN ソフトウエア・ドライバ・ヘッダ・ファイル・コードを生成します。

(1) CAN ソフトウエア・ドライバ・ソース・ファイルの出力オプション設定

＜起動方法＞

- メニューの[Option]から起動します。

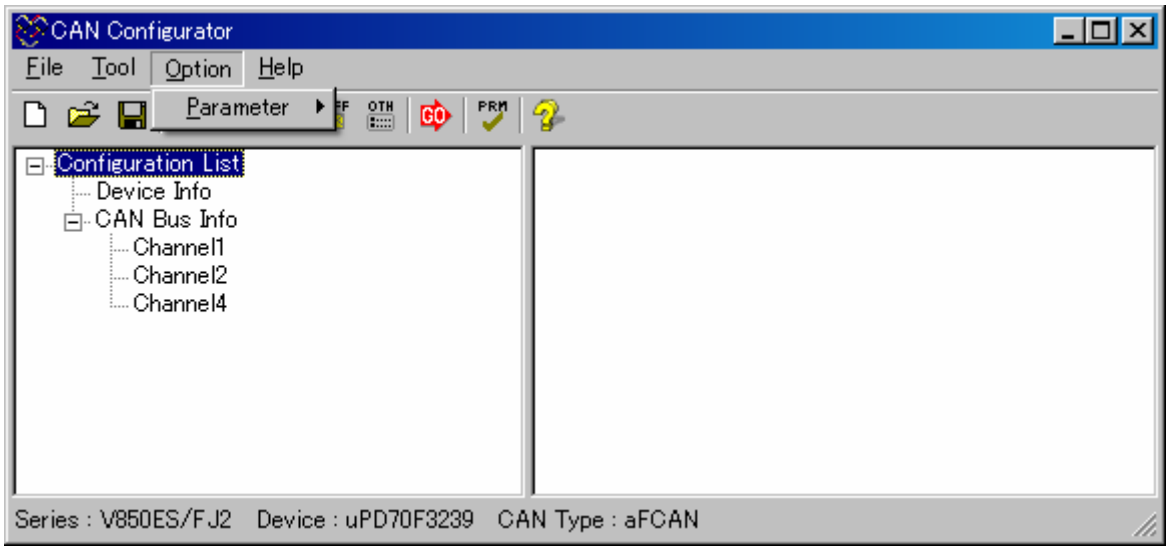
＜設定項目＞

- [Option] - [Parameter]

  出力する CAN ソフトウエア・ドライバ・ソース・ファイル群のパラメータ・チェック機能の有無を選択します。

Check	パラメータ・チェックありの CAN ソフトウエア・ドライバ・ソース・ファイル群を出力します
No Check	パラメータ・チェックなしの CAN ソフトウエア・ドライバ・ソース・ファイル群を出力します

★ **図 4－21　出力オプション設定画面**



(2) ファイルの出力

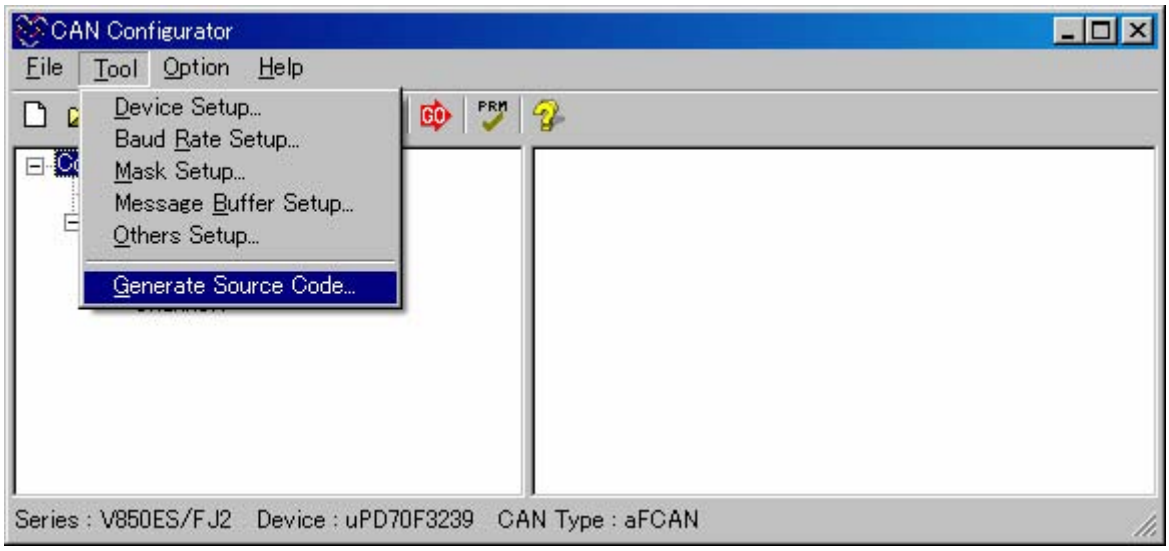
情報ファイル，コンフィギュレータ・ヘッダ・ファイル，CAN ソフトウエア・ドライバ・ソース・ファイル群，CAN ソフトウエア・ドライバ・ヘッダ・ファイルを出力します。

ユーザは，これらのファイルの格納先を任意のフォルダに指定することができます。

＜起動方法＞

- メニューの[Tool]-[Generate source code]から起動します。

★ **図4－22　コード生成起動画面**



4.4.7　プロジェクト・ファイルの保存，呼び出し

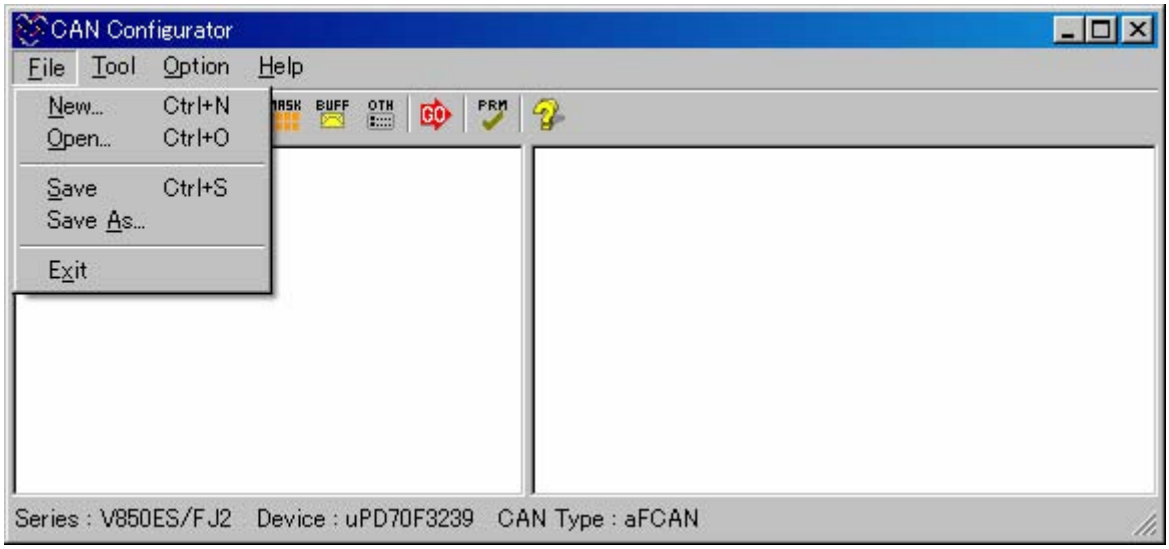
情報ファイル，ヘッダ・ファイルを管理するプロジェクト・ファイルを保存，呼び出しをします。

プロジェクト・ファイルは XML 形式になっています。

＜起動方法＞

- [File] – [Save As]でプロジェクト・ファイルを新規に保存します
- [File] – [Save]でプロジェクト・ファイルを上書き保存します。
- [File] – [Open]でプロジェクト・ファイルを呼び出します。

★　図4－23　プロジェクト・ファイル保存，呼び出し画面



4.5　エラー／ワーニング・メッセージ一覧

範囲外の設定を行った場合や，未設定の項目があった場合，表に示すようなメッセージを出力します。

表4－1　エラー・コード一覧

(1/3)

	コード名	メッセージ	説　明	対　処
	(E)0101	Device was not selected. Select a device.	デバイス選択が行われていない状態で[Tool]-[Generate Source Code...]（ソース・コードの生成）を行った場合に表示	[File]-[New]または[File]-[Open]でデバイスの選択を行ってください
	(E)0102	Channel was not selected. Select a channel.	デバイス選択ダイアログで使用するチャネルが選択されていない状態で[Tool]-[Generate Source Code...]（ソース・コードの生成）を行った場合に表示	[Tool]-[Device Setup...]で使用するチャネルの選択を行ってください
	(E)0103	Baud rate was not selected. Select a baud rate.	ボー・レート設定ダイアログでボー・レートの設定が行われていない状態で[Tool]-[Generate Source Code...]（ソース・コードの生成）を行った場合に表示	[Tool]-[Baud Rate Setup...]でボー・レート値の設定の選択を行ってください
★	(E)0104	The CAN clock is wrong value.	デバイス選択ダイアログでボー・レート値を設定できない値のCANクロック値が設定された場合に表示	0＜CANクロック＜1310の範囲でCANクロック値を設定してください
	(E)0105	The baud rate value is incorrect.	ボー・レート設定ダイアログでボー・レート値として，10進数の数値以外を入力した場合に表示	10進数の数値を入力してください
	(E)0106	Enter a different baud rate value (from 5 to 1000 Kbps).	ボー・レート設定ダイアログで5～1000 Kbpsの範囲以外のボー・レート値を選択した場合に表示	5～1000 Kbpsの範囲でボー・レート値を設定してください
	(E)0107	Channel X not set to Baud Rate list. Select from list.	ボー・レート設定ダイアログで各設定の組み合わせリストを選択していないチャネルがある場合に表示	リストから設定値の組み合わせを選択してください
	(E)0108	Buffer name 'XXX' already exists.	送信（または受信)メッセージ・バッファ設定ダイアログで既に使用されているバッファ名を設定しようとした場合に表示	他のバッファと重複しないバッファ名を設定してください
	(E)0109	CANID 'XXX' is incorrect.	送信（または受信)メッセージ・バッファ設定ダイアログでCANIDに16進数以外の文字列を入力した場合に表示（ただし，先頭から二文字分の0xはのぞく）	0-9，A-Fの値を入力してください

(2/3)

	コード名	メッセージ	説　明	対　処
	(E)010A	CANID 'XXX' is outside setting range.	送信（または受信)メッセージ・バッファ設定ダイアログで CANID に設定可能範囲外の値を入力した場合に表示	標準 ID の場合： 0x000-0x7FF 拡張 ID の場合： 0x00000000-0x1FFFFFFF
	(E)010B	Buffer name 'XXX' is a reserved name. Set another name.	送信（または受信)メッセージ・バッファ設定ダイアログでバッファ名に他のバッファのディフォールト名を設定しようとした場合に表示	他のバッファのディフォールト名ではないバッファ名を設定してください
	(E)010E	DLC 'XXX' is incorrect.	送信メッセージ・バッファ設定ダイアログで DLC の値に数字以外の文字が設定された場合に表示	0-15 の範囲で DLC の値を設定してください
	(E)010F	DLC 'XXX' is out of range. Enter a value in the range from 0 to 15.	送信メッセージ・バッファ設定ダイアログで DLC の値に 0-15 以外の値が設定された場合に表示	0-15 の範囲で DLC の値を設定してください
	(E)0111	CAN register area address is incorrect.	デバイス選択ダイアログで CAN レジスタ領域に設定可能アドレス以外の値が設定された場合に表示	設定可能なアドレスを設定してください
★	(E)0112	Enter the CAN clock value.	デバイス選択ダイアログで CAN クロック値が設定されていない（空欄）場合に表示	0＜CAN クロック＜1310 の範囲で CAN クロック値を設定してください
★	(E)0113	Enter the CAN register area address.	デバイス選択ダイアログで CAN レジスタ・エリアの値が設定されていない(空欄)場合に表示	設定可能なアドレスを設定してください
	(E)0114	Enter a buffer name.	送信(または受信)メッセージ・バッファ設定ダイアログでバッファ名の設定がされていない（空欄）場合に表示	バッファ名を設定してください
	(E)0115	Enter a CANID value.	送信(または受信)メッセージ・バッファ設定ダイアログで CANID の値が設定されていない（空欄）場合に表示	CANID を設定してください
	(E)0116	Enter a DLC value.	送信メッセージ・バッファ設定ダイアログで DLC の値が設定されていない（空欄）場合に表示	DLC を設定してください
	(E)0117	Install database file DeviceEntry.xml.	デバイス依存情報用データベース・ファイル（DeviceEntry.xml）が見つからない場合に表示	デバイス依存情報用データベース・ファイル（DeviceEntry.xml）をインストールしてください
	(E)0118	Install database file TemplateData.xml.	コード生成用データベース・ファイル（TemplateData.xml）が見つからない場合に表示	コード生成用データベース・ファイル（TemplateData.xml）をインストールしてください

(3/3)

コード名	メッセージ	説　明	対　処
(E)0119	The file name [candrv.h] is library header name. Set another name.	コンフィグレーション結果の出力ファイル名として candrv.c/h を設定したときに表示。	candrv.c/h 以外のファイル名を出力ファイル名として設定してください
(E)011A	XXX is incorrect. (XXX は設定した受信メッセージのバッファ数)	DCAN 用のバッファ登録ダイアログで，受信メッセージ・バッファの使用バッファ数設定欄に数字以外の文字を設定したときに表示	受信メッセージ・バッファの使用バッファ数設定欄に数字を設定してください
(E)011B	This Channel doesn't have XXX receive buffers. Input the number less than N. (XXX は設定した受信メッセージのバッファ数，N は受信メッセージ・バッファとして登録できる最大バッファ数)	DCAN 用のバッファ登録ダイアログで，受信メッセージ・バッファの使用バッファ数設定欄に登録できる数以上の値を設定したときに表示	受信メッセージ・バッファの使用バッファ数設定欄に登録できる数以内の数字を設定してください
(E)011C	This device must use Rx buffer. Input the number of 1 or over.	DCAN 用のバッファ登録ダイアログで受信メッセージ・バッファの使用バッファ数設定欄に 0 以下の値を設定したときに表示 (78K0)	受信メッセージ・バッファの使用バッファ数設定欄に登録できる数以内の数字を設定してください
(E)011D	The external CAN clock value is incorrect. Enter a different external CAN clock value.	ボー・レート設定ダイアログで，外部クロックを CAN モジュール・システム・クロックとして使用する場合，CAN モジュール・システム・クロックに数字以外を入力したときに表示 (78K0)	CAN モジュール・システム・クロックに数字でクロック値を入力してください
(E)011E	XXX doesn't exist, select ohter file! (XXX はファイル名)	開こうとしたプロジェクト・ファイルが存在しない場合に表示	存在するプロジェクト・ファイルを選択してください
(E)011F	Target CAN message buffer has not been selected. Select a message buffer.	メッセージ・バッファの設定が行われていない状態で [Tool]-[Generate Source Code...]（ソース・コードの生成）を行った場合に表示	[Tool]-[Message Buffer Setup...]でメッセージ・バッファの設定を行ってください。
★ (E)0120	The series name XXX or the device name YYY is not in the database file DeviceEntry.xml. (XXX はマイクロコントローラ名，YYY はデバイス名)	開いたプロジェクト・ファイルのマイクロコントローラ名またはデバイス名がデバイス依存情報用データベース・ファイル（DeviceEntry.xml）に存在しない場合に表示	マイクロコントローラ名およびデバイス名が存在するプロジェクト・ファイルを選択してください

表4−2　ワーニング・コード一覧

（1/2）

	コード名	メッセージ	出力箇所	備　考
	(W)0001	Do you want to create a new project without saving the current project?	ファイルへ保存していない設定がある状態で，[File]-[New]（プロジェクトの新規作成）を行った場合に表示	現在の設定を保存しない場合：Yes ボタンを押してください 現在の設定を保存する場合：No → [File]-[Save]/[Save as]で保存
★	(W)0002	The CAN Clock has been changed, the Baud rate setting should be updated!	[Tool]-[Device Setup…]（デバイス選択ダイアログの表示）で CAN クロックの値を変更した場合に表示	ボー・レート設定ダイアログで，ボー・レート値を再設定してください
	(W)0003	The channel being used has been changed: settings must be updated!	[Tool]-[Device Setup...]（デバイス選択ダイアログの表示）で使用チャネル数を追加した場合に表示	追加したチャネルに対して，各ダイアログ（ボー・レート設定，マスク設定，バッファ登録,その他の設定ダイアログ）で設定を行ってください
	(W)0007	BufferX can be used as a send message! （X はバッファ番号）	バッファ登録ダイアログで，ABT モード使用を選択している状態で，バッファ 0-7 のいずれかを "For Rx Message"リストに登録しようとした場合に表示	BufferX を Rx Message として使用する場合：ABT モード未使用を選択してください
	(W)0008	XXX is set at Rx message. Delete it from the Rx Message list. （XXX はバッファ名）	バッファ登録ダイアログで，バッファ 0-7 のいずれかが"For Rx Message" リストに登録されて いる状態で ABT モード使用を選択しようとした場合に表示	ABT モードを使用する場合："For Rx Message"リストに登録されているバッファ 0-7 を"For Rx Message"リストから削除してください
★	(W)0009	The DLC value in the message frame has to be transferred as programmed but only 8 data bytes are transferred in the data field. Set the DLC value (XX).? （XX は DLC の値）	送信メッセージ・バッファ設定ダイアログで，DLC に 9-15 の値を設定しようとした場合に表示	9-15 を DLC 値にしない場合は，No ボタンを押し，再設定を行ってください（9-15 を設定した場合，実際に CAN バスに送信される DLC は設定値になりますが,送信されるデータは設定した DLC 値に関わらず 8 バイトデータとなります）。
	(W)000A	Do you want to save the current project?	ファイルへ保存していない設定がある状態で，[File]-[Exit]または,ウインドウ右上の×ボタンを押した場合に表示	現在の設定を保存してアプリケーションを終了する場合：Yes ボタンを押してください 現在の設定を保存しないでアプリケーションを終了する場合：No ボタンを押してください アプリケーションを終了しない場合：Cancel ボタンを押してください

(2/2)

コード名	メッセージ	出力箇所	備　考
(W)000B	Select the using device or read the project file.	使用するデバイスが選択されていない状態でヘルプを開こうとした場合に表示	[File]-[New]で使用するデバイスを選択するか，[File]-[Open]で既存のプロジェクト・ファイルを開いてください
(W)000C	Check and update the message buffer setting!	DCAN 用のマスク設定ダイアログで，"Apply area" 欄のマスク動作設定を変更した場合に表示	[Tool]-[Message Buffer Setup…]でメッセージ・バッファを確認し，必要な場合は再設定を行ってください
(W)000D	Do you want to load the project without saving the current project?	ファイルへ保存していない設定がある状態で，[File]-[Open]（既存プロジェクトの読み込み）を行った場合に表示	現在の設定を保存しない場合：Yes ボタンを押してください 現在の設定を保存する場合：No → [File]-[Save]/[Save as]で保存

第5章　ドライバ関数

5.1　ドライバ関数一覧

ドライバ関数一覧を次に示します。

5.1.1　初期化／設定関連（6 種類）

	関数名	機能概要
	CanChEnable	CAN イネーブル（チャネル指定）
	CanAllEnable	CAN イネーブル（全チャネル指定）
	CanChInit	CAN チャネル初期化（チャネル指定再初期化）
	CanAllInit	CAN チャネル初期化（全チャネル再初期化）
★	CanChShutdown	強制シャットダウン（チャネル指定）
★	CanAllShutdown	強制シャットダウン（全チャネル指定）

5.1.2　動作モード関連（3 種類）

	関数名	機能概要
	CanChSetNrmMode	通常動作モード設定
	CanChGetMode	動作モードおよびパワー・セーブ・モード状態取得
★	CanChSetInitMode	初期化モード設定

5.1.3　バッファ・データ取得関連（4 種類）

関数名	機能概要
CanMsgGetDatDlc	データ，データ長の取得
CanMsgGetIdDatDlc	CAN-ID，データ，データ長の取得
CanMsgGetDatDlc_DSx	データ，データ長の取得[注]
CanMsgGetIdDatDlc_DSx	CAN-ID，データ，データ長の取得[注]

5.1.4　バッファ・データ設定関連（4 種類）

関数名	機能概要
CanMsgSetDat	データの設定
CanMsgSetIdDatDlc	CAN-ID，データ，データ長の設定
CanMsgSetDat_DSx	データの設定[注]　（x = 1～8）
CanMsgSetIdDatDlc_DSx	CAN-ID，データ，データ長の設定[注]　（x = 1～8）

5.1.5　送受信確認関連（4 種類）

関数名	機能概要
CanMsgTxReq	送信要求
CanMsgGetTxInfo	送信情報取得
CanChSrcRxInfo	受信情報検索（DN サーチ）
CanChSrcRxInfo_MSxx	受信情報検索（DN サーチ）[注]　（xx = 01～16）

注　78K0-DCAN 専用パフォーマンス改善関数。

5. 1. 6　CAN チャネル状態取得関連（3 種類）

関数名	機能概要
CanChGetStatus	CAN チャネル状態取得
CanChClrStatus	CAN チャネル状態クリア
CanChGetBusStatus	CAN バス状態取得

5.2　データ・タイプ

CAN ソフトウエア・ドライバを使用するアプリケーションが使用するすべてのデータ・タイプは，typedef を使用して特別なデータ・タイプとして candrv.h に宣言されています。

CANソフトウエア・ドライバで使用するデータ・タイプを 表 5－1に示します。

表5－1　データ･タイプ一覧

CAN ソフトウエア・ドライバにおけるデータ・タイプ	実際のデータ・タイプ	説　明
CD_ER	unsigned int[注]	エラー・コード，戻り値
CD_ID	unsigned long	CAN-ID
CD_DLC	signed char	データ長
CD_DAT	unsigned char	CAN データ
CD_CHNO	signed char	CAN チャネル番号
CD_BUFNO	unsigned char	CAN メッセージ・バッファ番号

注　int 型のサイズは，CA850/CCV850（V850）では 4 バイト。CC78K0（78K0）では 2 バイトとなります。

各パラメータ値に指定できる値の範囲は 表 5－2のとおりです。この範囲を越えた値を指定した場合，動作は不定となるので注意してください。

表5－2　パラメータ範囲

データ・タイプ	指定できる範囲
CD_ID	0x0-0x1FFFFFFF
CD_DLC	0-15
CD_DAT	0x00-0xFF
CD_CHNO	0～デバイスに搭載されているチャネル数に依存[注]
CD_BUFNO	0～デバイスに搭載されているバッファ数に依存[注]

注　詳細はデバイスごとのユーザーズ・マニュアルを参照してください。

また，表5－3のとおりチャネル番号など各パラメータ値に指定するマクロが用意されています。

表5－3　パラメータ用マクロ

マクロ名	値	説　　明
CD_CAN1	0	CAN1 指定用のマクロ
CD_CAN2	1	CAN2 指定用のマクロ
CD_CAN3	2	CAN3 指定用のマクロ
CD_CAN4	3	CAN4 指定用のマクロ
CD_CAN5	4	CAN5 指定用のマクロ
CD_CAN6	5	CAN6 指定用のマクロ
CD_ERR_CLR_STS	0x0004	CAN エラー・ステータス　クリア指定マクロ
CD_ERR_CLR_PRT	0x0008	CAN プロトコル・エラー・ステータス　クリア指定マクロ
CD_ERR_CLR_ABL	0x0010	アービトレーション・ロスト・ステータス　クリア指定マクロ
CD_ERR_CLR_WAK	0x0020	CAN スリープ・モードからのウエイクアップ・ステータス　クリア指定マクロ
CD_ERR_CLR_OVR	0x0040	CAN オーバラン・エラー・ステータス　クリア指定マクロ
CD_ERR_CLR_TXP	0x0080	CAN 送信エラー・パッシブまたは，バス・オフ状態　クリア指定マクロ
CD_ERR_CLR_RXP	0x0100	CAN 受信エラー・パッシブ状態　クリア指定マクロ

5.3　戻り値（エラー・コード）

CANソフトウエア・ドライバ関数はCD_ER型のエラー・コード（戻り値）を返します。エラー・コードに使用されるシンボルはヘッダ・ファイルcandrv.hで宣言されています。表5－4に，ドライバ関数が返す戻り値の一覧を示します。

表5－4　エラー・コード用マクロ

シンボル	値	意　味
CD_TRUE	1	－
CD_FALSE	0	－
CD_E_OK	0x0	正常終了
CD_E_FLG	MSB = 1	最上位ビットが1。 値がエラー・コードであることを示す。
CD_E_PRM	CD_E_FLG + 0x1	パラメータ・エラー
CD_E_STS	CD_E_FLG + 0x2	CAN モジュール・ステータス・エラー
CD_E_ALRDY	CD_E_FLG + 0x3	すでに設定済み
CD_E_NOMSG	CD_E_FLG + 0x4	メッセージ未受信

5. 4　CAN-ID 変換マクロ

CANソフトウエア・ドライバ関数ではCAN-IDをCANソフトウエア・ドライバ・フォーマットで取り扱いますので，candrv.hに宣言されている 表 5－5の変換マクロを使用してください。

備考　CAN-ID マクロの使用方法は，CanMsgGetIdDatDlc 関数の使用例および CanMsgSetIdDatDlc 関数の使用例を参照してください。

表 5－5　CAN-ID 変換マクロ一覧

マクロ名	値	説　　明
CD_SET_STD_ID(id)	(id << 18)	標準 CAN-ID フォーマット → CAN ソフトウエア・ドライバ・フォーマット
CD_SET_EXT_ID(id)	(id | 0x80000000)	拡張 CAN-ID フォーマット → CAN ソフトウエア・ドライバ・フォーマット
CD_GET_STD_ID(id)	(id = (id >> 18) & 0x000007ff)	CAN ソフトウエア・ドライバ・フォーマット → 標準 CAN-ID フォーマット
CD_GET_EXT_ID(id)	(id = id & 0x1fffffff)	CAN ソフトウエア・ドライバ・フォーマット → 拡張 CAN-ID フォーマット

5.5　単チャネル仕様 CAN ソフトウエア・ドライバ関数

78K0 マイクロコントローラ用 CAN ソフトウエア・ドライバ関数は，単チャネル仕様（チャネル固定仕様）となっており，チャネルの指定はできません。単チャネル仕様の関数は基本関数と API が異なりますが，candrv.h 内の定義により次のように関数名が置き換えられるので，基本関数名で呼び出すことができます。

表5－6　単チャネル仕様 CAN ソフトウエア・ドライバ関数一覧

基本関数名（置き換え前）	78K0 マイクロコントローラ専用関数名（置き換え後）
CanChEnable(chno)	CanChEnable_CH1( )
CanChInit(chno)	CanChInit_CH1( )
CanChSetNrmMode(chno)	CanChSetNrmMode_CH1( )
CanChGetMode(chno)	CanChGetMode_CH1( )
★ CanChSetInitMode(chno)	CanChSetInitMode_CH1( )
CanMsgGetIdDatDlc(chno, bufno, p_canid, p_data, p_dlc)	CanMsgGetIdDatDlc_CH1(bufno, p_canid, p_data, p_dlc)
CanMsgGetDatDlc(chno, bufno, p_data, p_dlc)	CanMsgGetDatDlc_CH1(bufno, p_data, p_dlc)
CanMsgSetIdDatDlc(chno, bufno, canid, p_data, dlc)	CanMsgSetIdDatDlc_CH1(bufno, canid, p_data, dlc)
CanMsgSetDat(chno, bufno, p_data)	CanMsgSetDat_CH1(bufno, p_data)
CanMsgTxReq(chno, bufno)	CanMsgTxReq_CH1(bufno)
CanMsgGetTxInfo(chno, bufno)	CanMsgGetTxInfo_CH1(bufno)
CanChSrcRxInfo(chno, bufno)	CanChSrcRxInfo_CH1(bufno)
CanChGetStatus(chno)	CanChGetStatus_CH1( )
CanChClrStatus(chno, clrdat)	CanChClrStatus_CH1(clrdat)
CanChGetBusStatus(chno)	CanChGetBusStatus_CH1( )

【注　　意】

78K0 マイクロコントローラの場合，上記どちらの関数名でも呼び出すことができますが，基本関数名で呼び出す場合は，ダミーでチャネル番号を指定しなければなりません。

5. 6　パフォーマンス改善版 CAN ソフトウエア・ドライバ関数

一部の関数において，78K0-DCAN 専用にパフォーマンス（処理速度）を改善した関数が用意されています。

表 5－7　パフォーマンス改善版 CAN ソフトウエア・ドライバ関数一覧

基本関数名（V850, 78K0 共通）	パフォーマンス改善関数名（78K0-DCAN 専用）
CanMsgGetIdDatDlc(chno, bufno, p_canid, p_data, p_dlc)	CanMsgGetIdDatDlc_DSx( )
CanMsgGetDatDlc(chno, bufno, p_data, p_dlc)	CanMsgGetDatDlc_DSx( )
CanMsgSetIdDatDlc(chno, bufno, canid, p_data, dlc)	CanMsgSetIdDatDlc_DSx( )
CanMsgSetDat(chno, bufno, p_data)	CanMsgSetDat_DSx( )
CanChSrcRxInfo(chno, bufno)	CanChSrcRxInfo_MSxx( )

x ：1～8

xx：01～16

【注　　意】

パフォーマンス改善関数は，基本関数とは API が異なり，引数を持っていません。データの受け渡しはすべてグローバル変数によって行います。詳細は各関数の仕様を確認してください。

5. 7　ドライバ関数の解説

これより，ドライバ関数について 図 5－1の形式に従って解説します。

図 5－1　ドライバ関数の記述フォーマット

1 ⟶ ○○○○○○

2 ⟶ 【概　　要】

● ●

3 ⟶ 【C言語記述様式】

● ●

4 ⟶ 【パラメータ】

I/O	パラメータ	説　　明

5 ⟶ 【説　　明】

● ●
● ●
● ●
● ●
● ●
● ● ● ● ● ● ● ●

6 ⟶ 【戻り値】

● ● ●　　　● ● ● ● ● ● ● ● ●
● ● ● ●　　● ●
● ● ●　　　● ●

7 ⟶ 【注　　意】

● ●

8 ⟶ 【使用例】

● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●
● ●
● ● ● ● ● ● ● ●

1. **名　　称**

ドライバ関数の名称を示しています。

2. **【概　　要】**

ドライバ関数の機能概要を示しています。

3. **【C 言語記述形式】**

ドライバ関数を C 言語で発行する際の記述形式を示しています。

4. **【パラメータ】**

ドライバ関数のパラメータを以下の形式で示しています。

I/O	**パラメータ**	**説　　明**
A	B	C

A**：パラメータの入出力の区分**

I　**…　入力パラメータ**

O　**…　出力パラメータ**

B**：パラメータの型および名称**

C**：パラメータの説明**

5. **【説　　明】**

ドライバ関数の機能を説明しています。

6. **【戻 り 値】**

ドライバ関数からの戻り値をマクロと数値で示しています。

7. **【注　　意】**

ドライバ関数に関する注意事項です。主に実装依存による注意点について説明しています。

8. **【使 用 例】**

ドライバ関数ごとの使用例です。

5.8　ドライバ関数

5.8.1　初期化／設定

ここでは，表5－8に示すドライバ関数について説明しています。

表5－8　初期化／設定

	関数名	機能概要
	CanChEnable	CAN イネーブル（チャネル指定）
	CanAllEnable	CAN イネーブル（全チャネル指定）
	CanChInit	CAN チャネル初期化（再初期化）
	CanAllInit	CAN チャネル初期化（全チャネル再初期化）
★	CanChShutdown	強制シャットダウン（チャネル指定）
★	CanAllShutdown	強制シャットダウン（全チャネル指定）

CanChEnable

★ 【概　　要】

指定されたチャネルの CAN クロックを設定し，CAN コントローラを起動します。

【C 言語記述形式】

```
CD_ER CanChEnable(CD_CHNO chno);
```

【パラメータ】

I/O	パラメータ	説　　明
I	CD_CHNO　　chno	チャネル番号

★ 【説　　明】

このドライバ関数は，指定されたチャネルの CAN クロックを設定し，CAN コントローラの起動を行います[注]。正常終了した場合は，CAN モジュールは初期化モードになります。

すでに CAN コントローラが起動されているときにこの関数を発行した場合，CAN クロックの設定は行われず，CAN モジュールの動作モードも変化しません。

この関数は，ほかのドライバ関数を使用する前に発行する必要があります。

★　　注　DCAN コントローラは CAN クロックの設定は行わず，CAN コントローラの起動のみを行います。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	正常終了
CD_E_PRM	CD_E_FLG + 0x1	チャネル番号の指定が不正
CD_E_ALRDY	CD_E_FLG + 0x3	すでに CAN コントローラは起動されており，CAN クロックの設定ができない

【注　　意】

本関数は，FCAN コントローラでは使用できません。

【使 用 例】

```
CD_ER ret;

ret = CanChEnable(CD_CAN1);
if (ret == CD_E_OK){
 "正常終了処理を記述";
}
else if (ret == CD_E_PRM){
 "エラー処理記述" ;                    /* チャネル番号の指定が不正 */
                                       /* パラメータ・チェックありのドライバ使用時でパラメータ・エラー発生 */
}
else{                                  /* (ret == CD_E_ALRDY) */
 "エラー処理を記述" ;                  /* CAN コントローラは起動済み */
}
```

CanAllEnable

【概　　要】

すべてのチャネルの CAN クロックを設定し，CAN コントローラを起動します。

【C 言語記述形式】

```
CD_ER CanAllEnable();
```

【パラメータ】　　なし

【説　　明】

このドライバ関数は，すべてのチャネルの CAN クロックを設定し，CAN コントローラの起動を行い，タイム・スタンプ・カウンタの動作設定を行います。正常終了した場合は，すべての CAN モジュールは初期化モードになります。

すでに一つでも CAN コントローラが起動されているときにこの関数を発行した場合，CAN クロックの設定などは行われず，CAN モジュールの動作モードも変化しません。

この関数は，ほかのドライバ関数を使用する前に発行する必要があります。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	正常終了
CD_E_ALRDY	CD_E_FLG + 0x3	すでに CAN コントローラは起動されており，CAN クロックの設定などができない

【注　　意】

本関数は，FCAN コントローラのみで使用できます。

【使 用 例】

```
CD_ER ret;

ret = CanAllEnable();
if (ret == CD_E_OK){
  "正常終了処理を記述";
}
else{                                   /* (ret == CD_E_ALRDY) */
  "エラー処理を記述" ;                  /* CAN コントローラは起動済み */
}
```

CanChInit

【概　　要】

指定されたチャネルの初期設定（再初期化）を行います。

【C 言語記述形式】

```
CD_ER CanChInit(CD_CHNO chno);
```

【パラメータ】

I/O	パラメータ	説　　明
I	CD_CHNO　　chno	チャネル番号

【説　　明】

このドライバ関数は，指定されたチャネルに対して，次の設定を行います。

- **ボー・レートの設定**
- **サンプル・ポイントの設定**
- DBT **の設定**
- SJW **の設定**
- **チャネル割り込み許可禁止の設定**[注]
- **マスク・レジスタの設定**
- **メッセージ・バッファの設定（属性の設定，データのクリアなど）**
- **アービトレーション・ロスト時の動作設定**
- ABT **遅延設定**[注]

注　DCAN **コントローラを除く。**

上記設定値は別途コンフィギュレーション・ファイルで指定します。

この関数は CAN **モジュールが初期化モード時に発行する必要があります。**

初期化モードでないときにこの関数を発行した場合，上記の設定は行われません。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	**正常終了**
CD_E_PRM	CD_E_FLG + 0x1	**チャネル番号の指定が不正**
CD_E_STS	CD_E_FLG + 0x2	**初期化モードでないときに発行された**

【注　　意】

本関数は，FCAN **コントローラでは使用できません。**

【使 用 例】

```
CD_ER ret;

ret = CanChInit(CD_CAN1);
if (ret == CD_E_OK){
  "正常終了処理を記述";
}
else if (ret == CD_E_PRM){
  "エラー処理記述";                    /* チャネル番号の指定が不正 */
                                       /* パラメータ・チェックありのドライバ使用時でパラメータ・エラー発生 */
}
else{                                  /* (ret == CD_E_STS) */
  "エラー処理を記述";                  /* 初期化モードでない時に発行された */
}
```

CanAllInit

【概　　要】

すべてのチャネルの初期設定（再初期化）を行います。

【C 言語記述形式】

```
CD_ER CanAllInit();
```

【パラメータ】

なし

【説　　明】

このドライバ関数は，指定されたチャネルに対して，次の設定を行います。

- ボー・レートの設定
- サンプル・ポイントの設定
- DBT の設定
- SJW の設定
- チャネル割り込み許可禁止の設定
- マスク・レジスタの設定
- メッセージ・バッファの設定（属性の設定，データのクリアなど）

上記設定値は別途コンフィギュレーション・ファイルで指定します。

この関数はすべての CAN モジュールが初期化モード時に発行する必要があります。

一つでも初期化モードでない CAN モジュールがある場合，上記の設定は行われません。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	正常終了
CD_E_STS	CD_E_FLG + 0x2	初期化モードでないときに発行された

【注　　意】

本関数は，FCAN コントローラのみで使用できます。

【使 用 例】

```
CD_ER ret;

ret = CanAllInit();
if (ret == CD_E_OK){
  "正常終了処理を記述";
}
else{                                   /* (ret == CD_E_STS) */
  "エラー処理を記述";                   /* 初期化モードでない時に発行された */
}
```

★ CanChShutdown

【概　　要】

指定されたチャネルの強制シャットダウンを行います。

【C 言語記述形式】

```
CD_ER CanChShutdown(CD_CHNO chno);
```

【パラメータ】

I/O	パラメータ	説　　明
I	CD_CHNO　　chno	**チャネル番号**

【説　　明】

このドライバ関数は，指定されたチャネルに対して，強制シャットダウンを行います。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	**正常終了**
CD_E_PRM	CD_E_FLG + 0x1	**チャネル番号の指定が不正**
CD_E_STS	CD_E_FLG + 0x2	**強制シャットダウンが失敗した**

【注　　意】

本関数は，aFCAN **コントローラのみで使用できます。**

【使 用 例】

```
CD_ER ret;

ret = CanChShutdown(CD_CAN1);
if (ret == CD_E_OK){
  "正常終了処理を記述";
}
else if (ret == CD_E_PRM){
  "エラー処理記述";                    /* チャネル番号の指定が不正 */
                                       /* パラメータ・チェックありのドライバ使用時でパラメータ・エラー発生 */
}
else{                                  /* (ret == CD_E_STS) */
  "エラー処理を記述";                  /* 強制シャットダウンが失敗した */
}
```

★ CanAllShutdown

【概　　要】

すべてのチャネルの強制シャットダウンを行います。

【C 言語記述形式】

```
CD_ER CanAllShutdown();
```

【パラメータ】　　なし

【説　　明】

このドライバ関数は，すべてのチャネルに対して，強制シャットダウンを行います。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	**正常終了**
CD_E_STS	CD_E_FLG + 0x2	**強制シャットダウンが失敗した**

【注　　意】

本関数は，FCAN **コントローラのみで使用できます。**

【使 用 例】

```
CD_ER ret;

ret = CanAllShutdown();
if (ret == CD_E_OK){
  "正常終了処理を記述";
}
else{                                   /* (ret == CD_E_STS) */
  "エラー処理を記述";                   /* 強制シャットダウンが失敗した */
}
```

5.8.2　動作モード

ここでは，表5－9に示すドライバ関数について説明しています。

表5－9　動作モード

	関数名	機能概要
	CanChSetNrmMode	通常動作モード設定
	CanChGetMode	動作モードおよびパワー・セーブ・モード状態取得
★	CanChSetInitMode	初期化モード設定

CanChSetNrmMode

【概　　要】

指定されたチャネルを通常動作モードへ設定します。

【C 言語記述形式】

```
CD_ER CanChSetNrmMode(CD_CHNO chno);
```

【パラメータ】

I/O	パラメータ	説　　明
I	CD_CHNO　　chno	**チャネル番号**

【説　　明】

このドライバ関数は，指定されたチャネル番号の CAN **モジュールを初期化モードから通常動作モードに設定します。この関数は** CanChEnable **および** CanChInit **発行後，メッセージの送受信を行う前に発行する必要があります。**

この関数は CAN **モジュールが初期化モード時に発行する必要があります。**

初期化モードでないときにこの関数を発行した場合，動作モードは変化しません。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	**正常終了**
CD_E_PRM	CD_E_FLG + 0x1	**チャネル番号の指定が不正**
CD_E_STS	CD_E_FLG + 0x2	**初期化モードでないときに発行された**

【使 用 例】

```
CD_ER ret;

ret = CanChSetNrmMode(CD_CAN1);
if (ret == CD_E_OK){
  "正常終了処理を記述";
}
else if (ret == CD_E_PRM){
  "エラー処理記述" ;                    /* チャネル番号の指定が不正 */
                                        /* パラメータ・チェックありのドライバ使用時でパラメータ・エラー発生 */
}
else{                                   /* (ret == CD_E_STS) */
  "エラー処理を記述";                   /* 初期化モードでない時に発行された */
}
```

CanChGetMode

【概　　要】

指定されたチャネルの動作モードおよびパワー・セーブ・モードの状態を取得します。

【C 言語記述形式】

```
CD_ER CanChGetMode(CD_CHNO chno);
```

【パラメータ】

I/O	パラメータ	説　　明
I	CD_CHNO　　chno	チャネル番号

【説　　明】

このドライバ関数は，指定したチャネル番号の動作モードおよび，パワー・セーブ・モードの状態を取得し，戻り値として返却します。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_PRM	CD_E_FLG + 0x1	チャネル番号の指定が不正
−	MSB = 0 の場合	MSB = 0 時の各ビットの説明参照

- MSB = 0 **時の各ビットの説明**

ビット位置	内　　容
bit2 - bit0	動作モード状態（OPMODE） 000：初期化モード 001：通常動作モード 010：自動ブロック送信機能付き通常動作モード[注] 011：受信専用モード 100：シングル・ショット・モード 101：セルフ・テスト・モード[注]
bit4 - bit3	パワー・セーブ・モード状態（PSMODE） 00：パワー・セーブ・モードは選択されていません。 01：CAN スリープ・モード 11：CAN ストップ・モード
MSB - bit5	0 固定

注　aFCAN **コントローラのみ。**

【使 用 例】

```
CD_ER ret;
CD_ER pmmode;
CD_ER psmode;

ret = CanChGetMode(CD_CAN1);
if (ret == CD_E_PRM){
  "エラー処理記述"                          /* チャネル番号の指定が不正 */
                                            /* パラメータ・チェックありのドライバ使用時でパラメータ・エラー発生 */
}
else{
  pmmode = ret & 0x07;
  switch (pmmode){
       case 0:
            "初期化モードの処理";                    /* 現在、初期化モード */
            break;
       case 1:
            "通常動作モードの処理";                  /* 現在、通常動作モード */
            break;
       case 2:
            "ABT モードの処理";                      /* 現在、ABT モード */
            break;
       case 3:
            "受信専用モードの処理";                  /* 現在、受信専用モード */
            break;
       case 4:
            "シングル・ショット・モード時の処理";    /* 現在、シングル・ショット・モード */
            break;
       case 5:
            "セルフ・テスト・モード時の処理";        /* 現在、セルフ・テスト・モード */
            break;
  }
  psmode = (ret >> 3) & 0x02;
  switch (psmode){
       case 0:
            "パワー・セーブ・モードではない時の処理";
            break;
       case 1:
            "CAN スリープ・モードの処理";            /* CAN スリープ・モード中 */
            break;
       case 3:
            "CAN ストップ・モードの処理";            /* CAN ストップ・モード中 */
            break;
  }
}
```

★ CanChSetInitMode

【概　　要】

指定されたチャネルを初期化モードへ設定します。

【C 言語記述形式】

```
CD_ER CanChSetInitMode(CD_CHNO chno);
```

【パラメータ】

I/O	パラメータ	説　　明
I	CD_CHNO　　chno	**チャネル番号**

【説　　明】

このドライバ関数は，指定されたチャネル番号の CAN **モジュールを初期化モード以外の動作モードから初期化モードに設定します。**

すでに CAN **モジュールが初期化モードであるときにこの関数を発行した場合，動作モードは変化しません。**

本関数を呼び出したあとは，必ず CanChGetMode **で初期化モードに移行したことを確認してください。**

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	**正常終了**
CD_E_PRM	CD_E_FLG + 0x1	**チャネル番号の指定が不正**
CD_E_STS	CD_E_FLG + 0x2	**すでに初期化モードに設定されている**

【使 用 例】

```
CD_ER ret;

ret = CanChSetInitMode (CD_CAN1);
if (ret == CD_E_OK){
  ret = CanChGetMode (CD_CAN1);
  if (ret == 0x00){
      "正常終了処理を記述";
  }
  else{                                  /* 初期化モードに移行していない */
  }
}
else if (ret == CD_E_PRM){
  "エラー処理記述" ;                     /* チャネル番号の指定が不正 */
                                         /* パラメータ・チェックありのドライバ使用時でパラメータ・エラー発生 */
}
else{                                    /* (ret == CD_E_STS) */
  "エラー処理を記述";                    /* すでに初期化モードに設定されている */
}
```

5.8.3　バッファ・データ取得

ここでは，表5－10に示すドライバ関数について説明しています。

表5－10　バッファ・データ取得

関数名	機能概要
CanMsgGetDatDlc	データ，データ長の取得
CanMsgGetIdDatDlc	CAN-ID，データ，データ長の取得
CanMsgGetDatDlc_DSx	データ，データ長の取得[注]　(x = 1～8)
CanMsgGetIdDatDlc_DSx	CAN-ID，データ，データ長の取得[注]　(x = 1～8)

注　78K0-DCAN専用パフォーマンス改善関数。

CanMsgGetDatDlc

【概　　要】

指定されたチャネルのメッセージ・バッファから，データ，データ長を取得します。

【C 言語記述形式】

```
CD_ER CanMsgGetDatDlc(CD_CHNO chno, CD_BUFNO bufno,
                      CD_DAT* p_data, CD_DLC* p_dlc);
```

【パラメータ】

I/O	パラメータ		説　　明
I	CD_CHNO	chno	チャネル番号
I	CD_BUFNO	bufno	バッファ番号
O	CD_DAT*	p_data	メッセージ・データを格納する領域の先頭アドレス
O	CD_DLC*	p_dlc	メッセージ長を格納する領域の先頭アドレス

【説　　明】

このドライバ関数は，指定されたメッセージ・バッファから次のデータを取得します。

- **データ**（DLC **バイト数のみ取得。最大** 8 **バイト**）
- DLC **値（取得した値そのもの）**

この関数は初めに新規データがあるか確認し，新規データがない場合はデータの取得は行いません。
データを取得した場合，同時にデータ更新ビット（DN **ビット**）**をクリアします。**

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	正常終了
CD_E_PRM	CD_E_FLG + 0x1	チャネル番号またはバッファ番号の指定が不正
CD_E_STS	CD_E_FLG + 0x2	データ取得失敗
CD_E_NOMSG	CD_E_FLG + 0x4	新規データなし

【注　　意】

DCAN **コントローラの場合，本関数でデータを取得できるのは受信バッファのみです。**

【使 用 例】

```
/* 事前にコンフィギュレータで以下の設定をします。
   ・メッセージ・バッファの１つを受信用に割り当てる
     (ch1 のメッセージ・バッファ 1 を使用してバッファ名を Ch1_Msg01 と定義)
   ・割り当てたメッセージ・バッファのフレーム・フォーマットを標準 ID に設定
   ・受信する CAN-ID を任意の値に設定
   ・マスクの設定（必要時）
*/

CD_DLC canRdlc;
CD_DAT canRdata[8];
CD_DAT tempdata[8];
CD_ER  ret;
int i;

ret = CanMsgGetDatDlc(CD_CAN1,Ch1_Msg01,canRdata,&canRdlc);
if (ret == CD_E_OK){
  for (i=0; i<canRdlc ; i++ ){
       tempdata[i] = canRdata[i];                    /* DLC 分データを別バッファに移す */
  }
}
else if (ret == CD_E_NOMSG){
  "メッセージ未受信時の処理";                        /* 新規メッセージなし */
}
else{
  "パラメータ・エラー時の処理";                      /* パラメータ・チェックありのドライバ使用時で */
                                                    /* パラメータ・エラー (ret == CD_E_PRM)発生 */
}
```

CanMsgGetIdDatDlc

【概　　要】

指定されたチャネルのメッセージ・バッファから，CAN-ID，データ，データ長を取得します。

【C 言語記述形式】

```
CD_ER CanMsgGetIdDatDlc(CD_CHNO chno, CD_BUFNO bufno,
                        CD_ID* p_canid, CD_DAT* p_data, CD_DLC* p_dlc);
```

【パラメータ】

I/O	パラメータ		説　　明
I	CD_CHNO	chno	チャネル番号
I	CD_BUFNO	bufno	バッファ番号
O	CD_ID*	p_canid	CAN-ID を格納する領域の先頭アドレス
O	CD_DAT*	p_data	メッセージ・データを格納する領域の先頭アドレス
O	CD_DLC*	p_dlc	メッセージ長を格納する領域の先頭アドレス

【説　　明】

このドライバ関数は，指定されたメッセージ・バッファから次のデータを取得します。

- CAN-ID（CAN ソフトウエア・ドライバ・フォーマットでセットされます）
- データ（DLC バイト数のみ取得。最大 8 バイト）
- DLC 値（取得した値そのもの）

CAN-ID は CAN ソフトウエア・ドライバ・フォーマットでセットされますので，CAN-ID 変換マクロを使用して参照してください。

この関数は初めに新規データがあるか確認し，新規データがない場合はデータの取得は行いません。

データを取得した場合，同時にデータ更新ビット（DN ビット）をクリアします。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	正常終了
CD_E_PRM	CD_E_FLG + 0x1	チャネル番号またはバッファ番号の指定が不正
CD_E_STS	CD_E_FLG + 0x2	データ取得失敗
CD_E_NOMSG	CD_E_FLG + 0x4	新規データなし

【注　　意】

DCAN コントローラの場合，本関数でデータを取得できるのは受信バッファのみです。

【使 用 例】

```
/* 事前にコンフィギュレータで以下の設定をします。
   ・メッセージ・バッファの１つを受信用に割り当てる
     (ch1 のメッセージ・バッファ 1 を使用してバッファ名を Ch1_Msg01 と定義)
   ・割り当てたメッセージ・バッファのフレーム・フォーマットを標準 ID に設定
   ・受信する CAN-ID を任意の値に設定
   ・マスクの設定（必要時）
*/

CD_ID canRid;
CD_DLC canRdlc;
CD_DAT canRdata[8];
CD_DAT tempdata[8];
CD_ER  ret;
int i;

ret = CanMsgGetIdDatDlc(CD_CAN1,Ch1_Msg01,&canRid,canRdata,&canRdlc);
if (ret == CD_E_OK){
  CD_GET_STD_ID(canRid);                              /* 取得した CAN-ID を標準 ID フォーマット形式に変換 */
  for (i=0; i<canRdlc ; i++ ){
       tempdata[i] = canRdata[i];                     /* DLC 分データを別バッファに移す */
  }
}
else if (ret == CD_E_NOMSG){
  "メッセージ未受信時の処理";                          /* 新規メッセージなし */
}
else{
  "エラー処理";          /* データ取得失敗 (ret == CD_E_STS)    あるいは,  */
                         /* パラメータ・チェックありのドライバ使用時でパラメータ・エラー (ret == CD_E_PRM)発生 */
}
```

CanMsgGetDatDlc_DSx　（x = 1～8）

【概　　要】

78K0-DCAN 専用パフォーマンス改善関数です。

指定されたチャネルのメッセージ・バッファから，データ，データ長を取得しグローバル変数へ格納します。

【C 言語記述形式】

```
CD_ER CanMsgGetDatDlc_DS1();    （取得データ・サイズ = 1）
CD_ER CanMsgGetDatDlc_DS2();    （取得データ・サイズ = 2）
CD_ER CanMsgGetDatDlc_DS3();    （取得データ・サイズ = 3）
CD_ER CanMsgGetDatDlc_DS4();    （取得データ・サイズ = 4）
CD_ER CanMsgGetDatDlc_DS5();    （取得データ・サイズ = 5）
CD_ER CanMsgGetDatDlc_DS6();    （取得データ・サイズ = 6）
CD_ER CanMsgGetDatDlc_DS7();    （取得データ・サイズ = 7）
CD_ER CanMsgGetDatDlc_DS8();    （取得データ・サイズ = 8）
```

【パラメータ】

なし

【グローバル変数】

I/O	グローバル変数名		説　　明
I	unsigned char*	u1gp_rxbuf_addr	対象受信バッファの DSTAT レジスタのアドレス。 CanChSrcRxInfo_MSxx() の検索結果がそのまま使用できます。 【使用例】を参照してください。
O	CD_DAT	u1g_rxdata	取得したメッセージ・データが格納されるグローバル変数 （データ・サイズは使用する関数に合わせて設定）
O	CD_DLC	u1g_rxdlc	取得した DLC が格納されるグローバル変数

【説　　明】

このドライバ関数は，指定されたメッセージ・バッファから次のデータを取得しグローバル変数へ格納します。

- データ（取得するデータ・サイズは関数ごとに固定です）
- DLC 値（取得した値そのもの）

この関数は初めに新規データがあるか確認し，新規データがない場合はデータの取得は行いません。

データを取得した場合，同時にデータ更新ビット（DN ビット）をクリアします。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	**正常終了**
CD_E_STS	CD_E_FLG + 0x2	**データ取得失敗**
CD_E_NOMSG	CD_E_FLG + 0x4	**新規データなし**

【注　　意】

本関数でデータを取得できるのは受信バッファのみです。

【使 用 例】

```
/* 事前にコンフィギュレータで以下の設定をします。
   ・メッセージ・バッファの１～１６を受信用に割り当てる
     (ch1 のメッセージ・バッファ xx を使用してバッファ名を Ch1_Msgxx と定義)
   ・割り当てたメッセージ・バッファのフレーム・フォーマットを標準 ID に設定
   ・受信する CAN-ID を任意の値に設定
   ・マスクの設定（必要時）
以下のグローバル変数を定義します。
    unsigned char * u1gp_rxbuf_addr;
    CD_DAT          u1g_rxdata[8];
    CD_DLC          u1g_rxdlc;
*/

CD_DAT tempdata[8];
CD_ER  ret1;
CD_ER  ret2;
int i;

ret1 = CanChSrcRxInfo_MS01();                /* 受信しているバッファを検索 */
                                             /* 検索結果はグローバル変数 u1gp_rxbuf_addr へ格納される */
if(!(ret1 & CD_E_FLG)) {
 ret2 = CanMsgGetDatDlc_DS8();               /* u1gp_rxbuf_addr で指定されるバッファから */
                                             /* 8 バイト分のデータを取得 */
 if (ret2 == CD_E_OK){
     for (i=0; i<u1g_rxdlc ; i++ ){
         tempdata[i] = u1g_rxdata[i];        /* DLC 分データを別バッファに移す */
     }
 }
 else if (ret2 == CD_E_NOMSG){
     "メッセージ未受信時の処理";             /* 新規メッセージなし */
 }
 else{
     "パラメータ・エラー時の処理";           /* パラメータ・チェックありのドライバ使用時で */
                                             /* パラメータ・エラー (ret == CD_E_PRM)発生 */
 }
}
```

CanMsgGetIdDatDlc_DSx　（x = 1～8）

【概　　要】

78K0-DCAN 専用パフォーマンス改善関数です。

指定されたチャネルのメッセージ・バッファから，CAN-ID，データ，データ長を取得しグローバル変数へ格納します。

【C 言語記述形式】

```
CD_ER CanMsgGetIdDatDlc_DS1();    (取得データ・サイズ = 1)
CD_ER CanMsgGetIdDatDlc_DS2();    (取得データ・サイズ = 2)
CD_ER CanMsgGetIdDatDlc_DS3();    (取得データ・サイズ = 3)
CD_ER CanMsgGetIdDatDlc_DS4();    (取得データ・サイズ = 4)
CD_ER CanMsgGetIdDatDlc_DS5();    (取得データ・サイズ = 5)
CD_ER CanMsgGetIdDatDlc_DS6();    (取得データ・サイズ = 6)
CD_ER CanMsgGetIdDatDlc_DS7();    (取得データ・サイズ = 7)
CD_ER CanMsgGetIdDatDlc_DS8();    (取得データ・サイズ = 8)
```

【パラメータ】

なし

【グローバル変数】

I/O	グローバル変数名		説　　明
I	unsigned char*	u1gp_rxbuf_addr	対象受信バッファの DSTAT レジスタのアドレス。 CanChSrcRxInfo_MSxx() の検索結果がそのまま使用できます。 【使用例】を参照してください。
O	unsigned char	u1g_rxid[5]	取得した CAN-ID が格納されるグローバル変数
O	CD_DAT	u1g_rxdata	取得したメッセージ・データが格納されるグローバル変数 （データ・サイズは使用する関数に合わせて設定）
O	CD_DLC	u1g_rxdlc	取得した DLC が格納されるグローバル変数

【説　　明】

このドライバ関数は，指定されたメッセージ・バッファから次のデータを取得しグローバル変数へ格納します。

- CAN-ID（レジスタ・イメージでセットされます）
- データ（取得するデータ・サイズは関数ごとに固定です）
- DLC 値（取得した値そのもの。ただし最上位ビットは CAN-ID のフォーマットを表します。
  0：標準 CAN-ID, 1：拡張 CAN-ID）

この関数は初めに新規データがあるか確認し，新規データがない場合はデータの取得は行いません。

データを取得した場合，同時にデータ更新ビット（DN ビット）をクリアします。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	**正常終了**
CD_E_STS	CD_E_FLG + 0x2	**データ取得失敗**
CD_E_NOMSG	CD_E_FLG + 0x4	**新規データなし**

【注　　意】

本関数でデータを取得できるのは受信バッファのみです。

【使 用 例】

```
/* 事前にコンフィギュレータで以下の設定をします。
   ・メッセージ・バッファの０～１５を受信用に割り当てる
     (ch1 のメッセージ・バッファ xx を使用してバッファ名を Ch1_Msgxx と定義)
   ・割り当てたメッセージ・バッファのフレーム・フォーマットを標準 ID に設定
   ・受信する CAN-ID を任意の値に設定
   ・マスクの設定（必要時）
以下のグローバル変数を定義します。
      unsigned char *   u1gp_rxbuf_addr;
    unsigned char       u1g_rxid[5]
    CD_DAT              u1g_rxdata[8];
    CD_DLC              u1g_rxdlc;
*/

unsigned char tempid[5];
CD_DAT tempdata[8];
CD_ER  ret1;
CD_ER  ret2;
int i;

ret1 = CanChSrcRxInfo_MS01();               /* 受信しているバッファを検索 */
                                            /* 検索結果はグローバル変数 u1gp_rxbuf_addr へ格納される */
if(!(ret1 & CD_E_FLG)) {
 ret2 = CanMsgGetIdDatDlc_DS8();            /* u1gp_rxbuf_addr で指定されるバッファから */
                                            /* 8 バイト分のデータを取得 */
 if (ret2 == CD_E_OK){
     for (i=0; i<u1g_rxdlc ; i++ ){
         tempdata[i] = u1g_rxdata[i];       /* DLC 分データを別バッファに移す */
     }
     for (i=0; i<5 ; i++ ){
         tempid[i] = u1g_rxid[i];           /* CAN-ID を別バッファに移す */
     }
 }
```

```
  else if (ret2 == CD_E_NOMSG){
      "メッセージ未受信時の処理";                    /* 新規メッセージなし */
  }
  else{
      "パラメータ・エラー時の処理";                  /* パラメータ・チェックありのドライバ使用時で */
                                                    /* パラメータ・エラー (ret == CD_E_PRM)発生 */
  }
}
```

5.8.4　バッファ・データ設定

ここでは，表5－11に示すドライバ関数について説明しています。

表5－11　バッファ・データ設定

関数名	機能概要
CanMsgSetDat	データの設定
CanMsgSetIdDatDlc	CAN-ID，データ，データ長の設定
CanMsgSetDat_DSx	データの設定[注]　(x = 1～8)
CanMsgSetIdDatDlc_DSx	CAN-ID，データ，データ長の設定[注]　(x = 1～8)

注　78K0-DCAN専用パフォーマンス改善関数。

CanMsgSetDat

【概　　要】

指定されたチャネルのメッセージ・バッファへ，データを設定します。

【C 言語記述形式】

```
CD_ER CanMsgSetDat(CD_CHNO chno, CD_BUFNO bufno, CD_DAT* p_data);
```

【パラメータ】

I/O	パラメータ		説　　明
I	CD_CHNO	chno	**チャネル番号**
I	CD_BUFNO	bufno	**バッファ番号**
I	CD_DAT*	p_data	**メッセージ・データを格納する領域の先頭アドレス**

【説　　明】

このドライバ関数は，指定されたメッセージ・バッファへ次のデータを設定します。

- **データ（データ長は，そのときのバッファ設定値（**DLC**））**

送信 CAN-ID**，メッセージ長は，メッセージ・バッファに設定された（初期化完了時はコンフィギュレーション時の初期設定）値を使用します。**

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	**正常終了**
CD_E_PRM	CD_E_FLG + 0x1	**チャネル番号またはバッファ番号の指定が不正**
CD_E_STS	CD_E_FLG + 0x2	**データ設定失敗**

【注　　意】

CanMsgSetIdDatDlc **関数で** CAN-ID, DLC **をコンフィギュレーション時の初期設定値から変更した場合，**CanMsgSetDat **関数においても，以降はその変更した設定値で送信されます。**

DCAN **コントローラの場合，本関数でデータを設定できるのは送信バッファのみです。**

【使 用 例】

```
/* 事前にコンフィギュレータで以下の設定をします。
   ・メッセージ・バッファの１つを送信用に割り当てる
     (ch1 のメッセージ・バッファ 0 を使用してバッファ名を Ch1_Msg00 と定義)
   ・DLC を 8 バイトに設定
   ・送信する CAN-ID を設定
   ・割り当てたメッセージ・バッファのフレーム・フォーマットを標準 ID に設定
   ・割り当てたメッセージ・バッファのフレーム・タイプをデータ・フレームに設定
*/

CD_DAT canTdata[8];
CD_ER  ret;

canTdata[0] = 0x00;                                  /* Tx data byte1 */
canTdata[1] = 0x11;                                  /* Tx data byte2 */
canTdata[2] = 0x22;                                  /* Tx data byte3 */
canTdata[3] = 0x33;                                  /* Tx data byte4 */
canTdata[4] = 0x44;                                  /* Tx data byte5 */
canTdata[5] = 0x55;                                  /* Tx data byte6 */
canTdata[6] = 0x66;                                  /* Tx data byte7 */
canTdata[7] = 0x77;                                  /* Tx data byte8 */

ret = CanMsgSetDat(CD_CAN1,Ch1_Msg00,canTdata);
                                                     /* 送信データをメッセージ・バッファにセット */
if(ret == CD_E_OK){
  ret = CanMsgTxReq(CD_CAN1,Ch1_Msg00);              /* メッセージ・バッファの送信要求ビットをセット */
  if(ret == CD_E_OK){
       "送信要求が成功したときの処理";               /* 送信要求ビットセット */
  }
  else{
       "送信要求が失敗したときの処理";               /* 送信要求ビットセットの失敗 */
  }
}
else{
  "パラメータ・エラー時の処理";                      /* パラメータ・チェックありのドライバ使用時で */
                                                     /* パラメータ・エラー (ret == CD_E_PRM)発生 */
}
```

CanMsgSetIdDatDlc

【概　　要】

指定されたチャネルのメッセージ・バッファへ，CAN-ID**，データ，データ長を設定します。**

【C 言語記述形式】

```
CD_ER CanMsgSetIdDatDlc(CD_CHNO chno, CD_BUFNO bufno,
                        CD_ID canid, CD_DAT* p_data, CD_DLC dlc);
```

【パラメータ】

I/O	パラメータ		説　　明
I	CD_CHNO	chno	**チャネル番号**
I	CD_BUFNO	bufno	**バッファ番号**
I	CD_ID	canid	CAN-ID（CAN **ソフトウエア・ドライバ・フォーマット**）
I	CD_DAT*	p_data	**メッセージ・データを格納する領域の先頭アドレス**
I	CD_DLC	dlc	**メッセージ長**

【説　　明】

このドライバ関数は，指定されたメッセージ・バッファへ次のデータを設定します。

- CAN-ID（CAN **ソフトウエア・ドライバ・フォーマットでセットします**）
- **データ**（DLC **バイト数のみ設定。最大** 8 **バイト**）
- DLC **値（下位** 4 **ビットのみ有効）**

CAN-ID **は** CAN **ソフトウエア・ドライバ・フォーマットでセットしなければならないため，**CAN-ID **変換マクロを使用して設定してください。**

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	**正常終了**
CD_E_PRM	CD_E_FLG + 0x1	**チャネル番号またはバッファ番号の指定が不正**
CD_E_STS	CD_E_FLG + 0x2	**データ設定失敗**

【注　　意】

DCAN **コントローラの場合，本関数でデータを設定できるのは送信バッファのみです。**

【使 用 例】

```
/* 事前にコンフィギュレータで以下の設定をします。
   ・メッセージ・バッファの１つを送信用に割り当てる
     (ch1のメッセージ・バッファ0を使用してバッファ名をCh1_Msg00と定義)
   ・割り当てたメッセージ・バッファのフレーム・フォーマットを標準IDに設定
   ・割り当てたメッセージ・バッファのフレーム・タイプをデータ・フレームに設定
*/

CD_ID canTid;
CD_DLC canTdlc;
CD_DAT canTdata[8];
CD_ER  ret;

canTdata[0] = 0x00;                          /* Tx data byte1 */
canTdata[1] = 0x11;                          /* Tx data byte2 */
canTdata[2] = 0x22;                          /* Tx data byte3 */
canTdata[3] = 0x33;                          /* Tx data byte4 */
canTdata[4] = 0x44;                          /* Tx data byte5 */
canTdata[5] = 0x55;                          /* Tx data byte6 */
canTdata[6] = 0x66;                          /* Tx data byte7 */
canTdata[7] = 0x77;                          /* Tx data byte8 */

canTid = CD_SET_STD_ID(0x100);               /* IDが0x100の標準IDフレームをドライバフォーマット形式に変換*/
canTdlc = 8;                                 /* DLC 8バイト */

ret = CanMsgSetIdDatDlc(CD_CAN1,Ch1_Msg00,canTid,canTdata,canTdlc);
                                             /* 送信データをメッセージ・バッファにセット */
if(ret == CD_E_OK){
  ret = CanMsgTxReq(CD_CAN1,Ch1_Msg00);      /* メッセージ・バッファの送信要求ビットをセット */
  if(ret == CD_E_OK){
      "送信要求が成功したときの処理";        /* 送信要求ビットセット */
  }
  else{
      "送信要求が失敗したときの処理";        /* 送信要求ビットセットの失敗 */
  }
}
else{
      "データ設定、パラメータ・エラー時の処理";      /* データ設定失敗 (ret == CD_E_STS)か */
                                                    /* パラメータ・チェックありのドライバ使用時で */
                                                    /* パラメータ・エラー (ret == CD_E_PRM)発生 */
}
```

CanMsgSetDat_DSx　（x = 1～8）

【概　　要】

78K0-DCAN 専用パフォーマンス改善関数です。

指定されたメッセージ・バッファへ，グローバル変数で指定されたデータを設定します。

【C 言語記述形式】

```
CD_ER CanMsgSetDat_DS1();    (設定データ・サイズ = 1)
CD_ER CanMsgSetDat_DS2();    (設定データ・サイズ = 2)
CD_ER CanMsgSetDat_DS3();    (設定データ・サイズ = 3)
CD_ER CanMsgSetDat_DS4();    (設定データ・サイズ = 4)
CD_ER CanMsgSetDat_DS5();    (設定データ・サイズ = 5)
CD_ER CanMsgSetDat_DS6();    (設定データ・サイズ = 6)
CD_ER CanMsgSetDat_DS7();    (設定データ・サイズ = 7)
CD_ER CanMsgSetDat_DS8();    (設定データ・サイズ = 8)
```

【パラメータ】

なし

【グローバル変数】

I/O	グローバル変数名		説　　明
I	unsigned char*	u1gp_txbuf_addr	対象送信バッファの TCON レジスタのアドレス
I	CD_DAT	u1g_txdata	設定するメッセージ・データが格納されているグローバル変数 （データ・サイズは使用する関数に合わせて設定）

【説　　明】

このドライバ関数は，指定されたメッセージ・バッファへ次のデータを設定します。

- データ（データ・サイズは使用する関数に合わせて設定）

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	正常終了
CD_E_STS	CD_E_FLG + 0x2	データ設定失敗

【注　　意】

本関数でデータを設定できるのは送信バッファのみです。

【使 用 例】

```
/* 事前にコンフィギュレータで以下の設定をします。
   ・メッセージ・バッファの１つを送信用に割り当てる
     (ch1のメッセージ・バッファ0を使用してバッファ名をCh1_Msg00と定義)
   ・DLCを8バイトに設定
   ・送信するCAN-IDを設定
   ・割り当てたメッセージ・バッファのフレーム・フォーマットを標準IDに設定
   ・割り当てたメッセージ・バッファのフレーム・タイプをデータ・フレームに設定
以下のグローバル変数を定義します。
    unsigned char *     u1gp_txbuf_addr;
    CD_DAT              u1g_txdata[8];
以下のマクロを定義します。(設定するメッセージ・バッファのアドレス)
    #define             TXBUFADD_00       (0xf600)
*/

CD_ER  ret;

u1gp_txbuf_addr = (unsigned char *)(TXBUFADD_00);
u1g_txdata[0] = 0x00;                                   /* Tx data byte1 */
u1g_txdata[1] = 0x11;                                   /* Tx data byte2 */
u1g_txdata[2] = 0x22;                                   /* Tx data byte3 */
u1g_txdata[3] = 0x33;                                   /* Tx data byte4 */
u1g_txdata[4] = 0x44;                                   /* Tx data byte5 */
u1g_txdata[5] = 0x55;                                   /* Tx data byte6 */
u1g_txdata[6] = 0x66;                                   /* Tx data byte7 */
u1g_txdata[7] = 0x77;                                   /* Tx data byte8 */

ret = CanMsgSetDat_DS8();                               /* 送信データをメッセージ・バッファにセット */
if(ret == CD_E_OK){
  ret = CanMsgTxReq(CD_CAN1,Ch1_Msg00);                 /* メッセージ・バッファの送信要求ビットをセット */
  if(ret == CD_E_OK){
      "送信要求が成功したときの処理";                   /* 送信要求ビットセット */
  }
  else{
      "送信要求が失敗したときの処理";                   /* 送信要求ビットセットの失敗 */
  }
}
else{
  "データ設定失敗時の処理";                             /* データ設定失敗 (ret == CD_E_STS)発生 */
}
```

CanMsgSetIdDatDlc_DSx　（x = 1～8）

【概　　要】

78K0-DCAN 専用パフォーマンス改善関数です。

指定されたメッセージ・バッファへ，グローバル変数で指定された CAN-ID，データ，データ長を設定します。

【C 言語記述形式】

```
CD_ER CanMsgSetIdDatDlc_DS1();    (設定データ・サイズ = 1)
CD_ER CanMsgSetIdDatDlc_DS2();    (設定データ・サイズ = 2)
CD_ER CanMsgSetIdDatDlc_DS3();    (設定データ・サイズ = 3)
CD_ER CanMsgSetIdDatDlc_DS4();    (設定データ・サイズ = 4)
CD_ER CanMsgSetIdDatDlc_DS5();    (設定データ・サイズ = 5)
CD_ER CanMsgSetIdDatDlc_DS6();    (設定データ・サイズ = 6)
CD_ER CanMsgSetIdDatDlc_DS7();    (設定データ・サイズ = 7)
CD_ER CanMsgSetIdDatDlc_DS8();    (設定データ・サイズ = 8)
```

【パラメータ】

なし

【グローバル変数】

I/O	グローバル変数名		説　　明
I	unsigned char*	u1gp_txbuf_addr	対象送信バッファの TCON レジスタのアドレス
I	unsigned char	u1g_txid[5]	設定する CAN-ID が格納されているグローバル変数
I	CD_DAT	u1g_txdata	設定するメッセージ・データが格納されているグローバル変数 （データ・サイズは使用する関数に合わせて設定）
I	CD_DLC	u1g_txdlc	設定する DLC が格納されているグローバル変数

【説　　明】

このドライバ関数は，指定されたメッセージ・バッファへ次のデータを設定します。

- CAN-ID（レジスタ・イメージでセットします）
- データ（データ・サイズは使用する関数に合わせて設定）
- DLC 値（下位 4 ビットのみ有効。ただし最上位ビットは CAN-ID のフォーマットを表します。
  0:標準 CAN-ID，1:拡張 CAN-ID）

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	正常終了
CD_E_STS	CD_E_FLG + 0x2	データ取得失敗
CD_E_NOMSG	CD_E_FLG + 0x4	新規データなし

【注　　意】

本関数でデータを設定できるのは送信バッファのみです。

【使 用 例】

```
/* 事前にコンフィギュレータで以下の設定をします。
   ・メッセージ・バッファの１つを送信用に割り当てる
     (ch1 のメッセージ・バッファ 0 を使用してバッファ名を Ch1_Msg00 と定義)
   ・DLC, CAN-ID, フレーム・フォーマットは任意に設定
   ・割り当てたメッセージ・バッファのフレーム・タイプをデータ・フレームに設定
以下のグローバル変数を定義します。
    unsigned char *     u1gp_txbuf_addr;
    unsigned char       u1g_txid[5];
    CD_DAT              u1g_txdata[8];
    CD_DLC              u1g_txdlc;
以下のマクロを定義します。(設定するメッセージ・バッファのアドレス)
    #define             TXBUFADD_00       (0xf600)
*/

CD_ER  ret;

u1gp_txbuf_addr = (unsigned char *)(TXBUFADD_00);          /* Message buffer address */
u1g_txdata[0] = 0x00;                                      /* Tx data byte1 */
u1g_txdata[1] = 0x11;                                      /* Tx data byte2 */
u1g_txdata[2] = 0x22;                                      /* Tx data byte3 */
u1g_txdata[3] = 0x33;                                      /* Tx data byte4 */
u1g_txdata[4] = 0x44;                                      /* Tx data byte5 */
u1g_txdata[5] = 0x55;                                      /* Tx data byte6 */
u1g_txdata[6] = 0x66;                                      /* Tx data byte7 */
u1g_txdata[7] = 0x77;                                      /* Tx data byte8 */
u1g_txid[0] = 0x00;                                        /* ID byte0 (IDTX0) */
u1g_txid[1] = 0x00;                                        /* ID byte1 (IDTX1) */
u1g_txid[2] = 0x00;                                        /* ID byte2 (IDTX2) */
u1g_txid[3] = 0x00;                                        /* ID byte3 (IDTX3) */
u1g_txid[4] = 0x00;                                        /* ID byte4 (IDTX4) */
u1g_txdlc = 8;                                             /* DLC */
```

```
ret = CanMsgSetIdDatDlc_DS8();                    /* 送信データをメッセージ・バッファにセット */
if(ret == CD_E_OK){
  ret = CanMsgTxReq(CD_CAN1,Ch1_Msg00);           /* メッセージ・バッファの送信要求ビットをセット */
  if(ret == CD_E_OK){
      "送信要求が成功したときの処理";               /* 送信要求ビットセット */
  }
  else{
      "送信要求が失敗したときの処理";               /* 送信要求ビットセットの失敗 */
  }
}
else{
  "データ設定失敗時の処理";                         /* データ設定失敗 (ret == CD_E_STS)発生 */
}
```

5.8.5　送受信確認

ここでは，表5－12に示すドライバ関数について説明しています。

表5－12　送受信確認

関数名	機能概要
CanMsgTxReq	送信要求
CanMsgGetTxInfo	送信情報取得
CanChSrcRxInfo	受信情報検索（DNサーチ）
CanChSrcRxInfo_MSxx	受信情報検索（DNサーチ）[注]　（xx = 01～16）

注　78K0-DCAN専用パフォーマンス改善関数。

CanMsgTxReq

【概　　要】

指定されたチャネルのメッセージ・バッファの送信要求ビットをセットします。

【C 言語記述形式】

```
CD_ER CanMsgTxReq(CD_CHNO chno, CD_BUFNO bufno);
```

【パラメータ】

I/O	パラメータ		説　　明
I	CD_CHNO	chno	**チャネル番号**
I	CD_BUFNO	bufno	**バッファ番号**

【説　　明】

このドライバ関数は，指定されたチャネルのメッセージ・バッファの送信要求ビットをセットします。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	**正常終了**
CD_E_PRM	CD_E_FLG + 0x1	**チャネル番号またはバッファ番号の指定が不正**
CD_E_STS	CD_E_FLG + 0x2	**送信要求ビットの設定失敗**

【使 用 例】

CanMsgSetIdDatDlc **の項を参照してください。**

CanMsgGetTxInfo

【概　　要】

指定されたチャネルのメッセージ・バッファの送信要求ビットを取得します。

【C 言語記述形式】

```
CD_ER CanMsgGetTxInfo(CD_CHNO chno, CD_BUFNO bufno);
```

【パラメータ】

I/O	パラメータ		説　　明
I	CD_CHNO	chno	**チャネル番号**
I	CD_BUFNO	bufno	**バッファ番号**

【説　　明】

このドライバ関数は，指定されたチャネルのメッセージ・バッファの送信要求ビットを取得します。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_PRM	CD_E_FLG + 0x1	**チャネル番号またはバッファ番号の指定が不正**
CD_TRUE	1	**送信要求ビットがセットされている**
CD_FALSE	0	**送信要求ビットがセットされていない**

【使 用 例】

```
/* 事前にコンフィギュレータで以下の設定をします。
   ・メッセージ・バッファの１つを送信用に割り当てる
     (ch1 のメッセージ・バッファ 0 を使用してバッファ名を Ch1_Msg00 と定義)
   ・割り当てたメッセージ・バッファのフレーム・フォーマットを標準 ID に設定
   ・割り当てたメッセージ・バッファのフレーム・タイプをデータ・フレームに設定
*/

CD_ER  ret;

ret = CanMsgGetTxInfo(CD_CAN1,Ch1_Msg00);
if (ret == CD_TRUE){
      "送信要求ビットがセットされている時の処理";       /* 未送信データが残っている */
}
else if (ret == CD_FALSE){
      "送信要求ビットがセットされていない時の処理";     /* 未送信データはなし */
}
else{
  "パラメータ・エラー時の処理";                        /* パラメータ・チェックありのドライバ使用時で */
                                                       /* パラメータ・エラー (ret == CD_E_PRM)発生 */
}
```

CanChSrcRxInfo

【概　　要】

指定されたチャネルのセットされているデータ更新ビット（DN ビット）を検索します。

【C 言語記述形式】

```
CD_ER CanChSrcRxInfo(CD_CHNO chno, CD_BUFNO bufno);
```

【パラメータ】

I/O	パラメータ		説　　明
I	CD_CHNO	chno	**チャネル番号**
I	CD_BUFNO	bufno	**検索開始バッファ番号**

【説　　明】

このドライバ関数は，セットされているデータ更新ビット（DN ビット）を検索します。

検索は，指定されたチャネル内の指定されたバッファ番号から昇順に行われ，バッファ番号の最大値まで検索します。最初に発見したバッファ番号を戻り値として返却します。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_PRM	CD_E_FLG + 0x1	**チャネル番号またはバッファ番号の指定が不正**
CD_E_NOMSG	CD_E_FLG + 0x4	**検索した範囲でデータ更新ビット（DN ビット）がセットされているバッファはなし**
−	MSB = 0 **の場合**	**最初に発見したバッファ番号**

【使 用 例】

```
/* 事前にコンフィギュレータで以下の設定をします。
   ・メッセージ・バッファの 0～15 を受信用に割り当てる
     (ch1 のメッセージ・バッファ xx を使用してバッファ名を Ch1_Msgxx と定義)
   ・割り当てたメッセージ・バッファのフレーム・フォーマットを標準 ID に設定
   ・受信する CAN-ID を任意の値に設定
   ・マスクの設定（必要時）
*/

CD_ER  ret;

ret = CanChSrcRxInfo(CD_CAN1,Ch1_Msg00);                    /* バッファ番号 0 から DN ビット検索 */
if (ret == CD_E_NOMSG){
      "データ更新ビットがセットされてないときの処理";        /* DN ビットは未セット */
}
else if ( ret == CD_E_PRM){
      "パラメータ・エラー時の処理";                          /* パラメータ・チェックありのドライバ使用時で */
                                                            /* パラメータ・エラー発生 */
}
else{
  switch (ret){
      case 0:
           "バッファ番号 0 がセットされていたときの処理";
           break;                                           /* バッファ番号 0 の DN ビットがセット*/
      case 1:
           "バッファ番号 1 がセットされていたときの処理";
           break;                                           /* バッファ番号 1 の DN ビットがセット*/
      case 2:
           "バッファ番号 2 がセットされていたときの処理";
           break;                                           /* バッファ番号 2 の DN ビットがセット*/
      case 3:
           break;
           /* …… */
  }
}
```

CanChSrcRxInfo_MSxx　（xx = 01～16）

【概　　要】

78K0-DCAN 専用パフォーマンス改善関数です。

セットされているデータ更新ビット（DN ビット）を検索します。

【C 言語記述形式】

```
CD_ER CanChSrcRxInfo_MS01();    (検索受信バッファ・サイズ = 1)
CD_ER CanChSrcRxInfo_MS02();    (検索受信バッファ・サイズ = 2)
CD_ER CanChSrcRxInfo_MS03();    (検索受信バッファ・サイズ = 3)
CD_ER CanChSrcRxInfo_MS04();    (検索受信バッファ・サイズ = 4)
CD_ER CanChSrcRxInfo_MS05();    (検索受信バッファ・サイズ = 5)
CD_ER CanChSrcRxInfo_MS06();    (検索受信バッファ・サイズ = 6)
CD_ER CanChSrcRxInfo_MS07();    (検索受信バッファ・サイズ = 7)
CD_ER CanChSrcRxInfo_MS08();    (検索受信バッファ・サイズ = 8)
CD_ER CanChSrcRxInfo_MS09();    (検索受信バッファ・サイズ = 9)
CD_ER CanChSrcRxInfo_MS10();    (検索受信バッファ・サイズ = 10)
CD_ER CanChSrcRxInfo_MS11();    (検索受信バッファ・サイズ = 11)
CD_ER CanChSrcRxInfo_MS12();    (検索受信バッファ・サイズ = 12)
CD_ER CanChSrcRxInfo_MS13();    (検索受信バッファ・サイズ = 13)
CD_ER CanChSrcRxInfo_MS14();    (検索受信バッファ・サイズ = 14)
CD_ER CanChSrcRxInfo_MS15();    (検索受信バッファ・サイズ = 15)
CD_ER CanChSrcRxInfo_MS16();    (検索受信バッファ・サイズ = 16)
```

【パラメータ】

なし

【グローバル変数】

I/O	グローバル変数名	説　　明
O	unsigned char*　u1gp_rxbuf_addr	最初に発見した受信バッファの DSTAT レジスタのアドレスが格納されるグローバル変数。 CanMsgGetDatDlc_DS1( ) などで検索結果がそのまま使用できます。CanMsgGetDatDlc_DSx の【使用例】を参照してください。

【説　　明】

このドライバ関数は，セットされているデータ更新ビット（DN ビット）を検索します。

検索は，常に受信バッファ 0 から昇順に行われ，関数ごとに決定されている検索受信バッファ・サイズまで検索します。最初に発見したバッファの DSTAT レジスタのアドレスをグローバル変数に格納します。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	0x0	検索した範囲で，データ更新ビット（DN ビット）がセットされているバッファあり。
CD_E_NOMSG	CD_E_FLG + 0x4	検索した範囲でデータ更新ビット（DN ビット）がセットされているバッファなし。

【使 用 例】

CanMsgGetDatDlc_DSx の項を参照してください。

5. 8. 6　CAN チャネル状態取得

ここでは，表 5－13に示すドライバ関数について説明しています。

表 5－13　CAN チャネル状態取得

関数名	機能概要
CanChGetStatus	CAN チャネル状態取得
CanChClrStatus	CAN チャネル状態クリア
CanChGetBusStatus	CAN バス状態取得

CanChGetStatus

【概　　要】

指定されたチャネルの CAN 状態を取得します。

【C 言語記述形式】

```
CD_ER CanChGetStatus(CD_CHNO chno);
```

【パラメータ】

I/O	パラメータ	説　　明
I	CD_CHNO　　chno	チャネル番号

【説　　明】

このドライバ関数は，指定されたチャネルの CAN スリープ・モードからのウエイクアップ，アービトレーション・ロスト，CAN プロトコル・エラー，CAN エラー・ステータスなどの CAN 状態を取得し，戻り値として返却します。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_PRM	CD_E_FLG + 0x1	チャネル番号の指定が不正
−	MSB = 0 の場合	MSB = 0 時の各ビットの説明参照

• MSB = 0 時の各ビットの説明

ビット位置	内　　容	各 CAN コントローラ対応		
		aFCAN	FCAN	DCAN
bit1, bit0	0 固定	−	−	−
bit2	CAN エラー・ステータス （0：イベント保留なし，1：イベント保留あり）	対応	未対応	未対応
bit3	CAN プロトコル・エラー （0：イベント保留なし，1：イベント保留あり）	対応	対応	未対応
bit4	アービトレーション・ロスト （0：イベント保留なし，1：イベント保留あり）	対応	未対応	未対応
bit5	CAN スリープ・モードからのウエイクアップ （0：イベント保留なし，1：イベント保留あり）	対応	対応	対応
bit6	CAN オーバラン・エラー （0：イベント保留なし，1：イベント保留あり）	未対応	対応	対応
bit7	CAN 送信エラー・パッシブまたは，バス・オフ状態 （0：イベント保留なし，1：イベント保留あり）	未対応	対応	未対応
bit8	CAN 受信エラー・パッシブ状態 （0：イベント保留なし，1：イベント保留あり）	未対応	対応	未対応
MSB - bit9	0 固定	−	−	−

【使 用 例】

```
CD_ER ret;

ret = CanChGetStatus(CD_CAN1);
if (ret == CD_E_PRM){
  "パラメータ・エラー時の処理";              /* パラメータ・チェックありのドライバ使用時でパラメータ・エラー発生 */
}
else{
  if ((ret & 0x04) == 0x04){
       "CAN エラー・ステータスのイベント保留あり時の処理";
  }
  if ((ret & 0x08) == 0x08){
       "CAN プロトコル・エラーのイベント保留あり時の処理";
  }
  if ((ret & 0x10) == 0x10){
       "アービトレーション・ロストのイベント保留あり時の処理";
  }
  if ((ret & 0x20) == 0x20){
       "CAN スリープ・モードからのウエイクアップのイベント保留あり時の処理";
  }
}
```

CanChClrStatus

【概　　要】

指定されたチャネルの CAN **状態をクリアします。**

【C 言語記述形式】

```
CD_ER CanChClrStatus(CD_CHNO chno, unsigned char clrdat);
```

【パラメータ】

I/O	パラメータ		説　　明
I	CD_CHNO	chno	**チャネル番号**
I	unsigned char	clrdat	**クリア・ビット指定データ**

【説　　明】

このドライバ関数は，指定されたチャネルの CAN **状態をクリアします。クリアする** CAN **状態は，クリア・ビット指定データで指定します。**

クリア・ビット指定データは，クリア・ビット指定データ用マクロ定義を使用します。一度の関数コールで，複数のビットをクリアするときは，目的のマクロ定義を“ | ”（論理和演算子）で結合して指定します。

- **クリア・ビット指定データ用マクロ定義**

マクロ名	内　　容	各 CAN コントローラ対応		
		aFCAN	FCAN	DCAN
CD_ERR_CLR_STS	CAN **エラー・ステータス　クリア指定マクロ**	**対応**	**未対応**	**未対応**
CD_ERR_CLR_PRT	CAN **プロトコル・エラー・ステータス　クリア指定マクロ**	**対応**	**対応**	**未対応**
CD_ERR_CLR_ABL	**アービトレーション・ロスト・ステータス　クリア指定マクロ**	**対応**	**未対応**	**未対応**
CD_ERR_CLR_WAK	CAN **スリープ・モードからのウエイクアップ・ステータス　クリア指定マクロ**	**対応**	**対応**	**対応**
CD_ERR_CLR_OVR	CAN **オーバラン・エラー・ステータス　クリア指定マクロ**	**未対応**	**対応**	**対応**
CD_ERR_CLR_TXP	CAN **送信エラー・パッシブまたは，バス・オフ状態　クリア指定マクロ**	**未対応**	**対応**	**未対応**
CD_ERR_CLR_RXP	CAN **受信エラー・パッシブ状態　クリア指定マクロ**	**未対応**	**対応**	**未対応**

【戻 り 値】

エラー・コード	値	意　　味
CD_E_OK	CD_E_FLG + 0x0	**正常終了**
CD_E_PRM	CD_E_FLG + 0x1	**チャネル番号の指定が不正**

【使 用 例】

```
CD_ER ret;
unsigned char clrdat;

clrdat = CD_ERR_CLR_STS | CD_ERR_CLR_PRT;
ret = CanChClrStatus(CD_CAN1,clrdat);          /* エラー・ステータスとプロトコル・エラー・ステータスのクリア指定*/
if (ret == CD_E_OK){
  "正常終了時の処理";
}
else{
  "パラメータ・エラー時の処理";                /* パラメータ・チェックありのドライバ使用時で */
                                               /* パラメータ・エラー (ret == CD_E_PRM)発生 */
}
```

CanChGetBusStatus

【概　　要】

指定されたチャネルの CAN バス状態を取得します。

【C 言語記述形式】

```
CD_ER CanChGetBusStatus(CD_CHNO chno);
```

【パラメータ】

I/O	パラメータ	説　　明
I	CD_CHNO　　chno	チャネル番号

【説　　明】

このドライバ関数は，指定されたチャネルのバス・オフ状態ビット，送信／受信エラー・カウンタ状態ビットなどの CAN バス状態を取得し，戻り値として返却します。

【戻 り 値】

エラー・コード	値	意　　味
CD_E_PRM	CD_E_FLG + 0x1	チャネル番号またはバッファ番号の指定が不正
−	MSB = 0 の場合	MSB = 0 **時の各ビットの説明**参照

• MSB = 0 時の各ビットの説明

ビット位置	内　　容
bit1, bit0	受信エラー・カウンタ状態ビット ★ 00：受信エラー・カウンタはワーニング・レベル未満（〜95） ★ 01：受信エラー・カウンタはワーニング・レベル範囲（96〜127） （DCAN コントローラの場合はワーニング・レベル範囲，またはエラー・パッシブ範囲） 10：未定義 11：受信エラー・カウンタはエラー・パッシブ範囲（128〜） （DCAN コントローラの場合は未定義）
bit3, bit2	送信エラー・カウンタ状態ビット ★ 00：送信エラー・カウンタはワーニング・レベル未満（〜95） ★ 01：送信エラー・カウンタはワーニング・レベル範囲（96〜127） （DCAN コントローラの場合はワーニング・レベル範囲，またはエラー・パッシブ範囲，またはバス・オフ範囲） 10：未定義 11：送信エラー・カウンタはエラー・パッシブまたはバス・オフ範囲（128〜） （DCAN コントローラの場合は未定義）
bit4	バス・オフ状態ビット 0：バス・オフ状態ではありません（送信エラー・カウントが 256 未満） 1：バス・オフ状態（送信エラーのカウントが 256 以上）
MSB - bit5	0 固定

【使 用 例】

```
CD_ER ret;
CD_ER rx_err;
CD_ER tx_err;

ret = CanChGetBusStatus(CD_CAN1);
if (ret == CD_E_PRM){
 "パラメータ・エラー時の処理";              /* パラメータ・チェックありのドライバ使用時でパラメータ・エラー発生 */
}
else{
 rx_err = ret & 0x03;
 switch (rx_err){
     case 0:
★            "受信エラー・カウンタはワーニング・レベル未満(～95)";
          break;
     case 1:
★            "受信エラー・カウンタはワーニング・レベル範囲(96～127)";
          break;
     case 3:
          "受信エラー・カウンタはエラー・パッシブ範囲(128～)";
          break;
 }
 tx_err = (ret >> 2) & 0x03;
 switch (rx_err){
     case 0:
★            "送信エラー・カウンタはワーニング・レベル未満(～95)";
          break;
     case 1:
★            "送信エラー・カウンタはワーニング・レベル範囲(96～127)";
          break;
     case 3:
          "送信エラー・カウンタはエラー・パッシブ範囲(128～)";
          break;
 }
 if ((ret & 0x10) == 0x10){
     "バス・オフ状態(送信エラーのカウントが 256 以上)";
 }
 else{
     "バス・オフ状態ではない";
 }
}
```

★ # 第6章　サンプル・プログラム

6. 1　V850ES/FJ2

ここでは，V850ES/FJ2 **のサンプル・プログラムについて説明します。**

6. 1. 1　動作環境

ターゲット・デバイス：μPD70F3239（V850ES/FJ2）

ターゲット・ボード：COSMO **製** CEB-V850ES/FJ2-SJ2

6. 1. 2　動作概要

- **デバイスの初期処理が正常終了すると，**7seg-LED **の表示が「**0**」から「**1**」に変わります。**
- **割り込みで受信データの検出を行います。**
- **タイマ** M **のタイマ周期で送信します。**
- **データの受信完了ごとに，**8bit-LED **に受信データの第** 1 **バイトが表示されます。**
- **送信データは送信** 1 **回ごとに変化します。**

  1 **回目：**"0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07"**の** 8 **バイト・データ**

  2 **回目：**"0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f"**の** 8 **バイト・データ**

  3 **回目：**"0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17"**の** 8 **バイト・データ**

  -
  -
  -

6.1.3　コンフィギュレータでの事前設定

コンフィギュレータにおいて，次の設定内容で設定ファイルを生成しています。

- **生成したファイル名称**

  情報ファイル：config.c

  ヘッダ・ファイル：config.h

Device selection	: Device Name	uPD70F3239(V850/FJ2)
	: CAN Clock	20MHz
	: CAN register area	0x03FEC000
	: Channel Select	Cannel 1 のみ
Baud rate setting	: Can module system clock	20MHz
	: Baud rate	500Kbps
	: データ・ビット設定	
	Prescaler	2
	DBT	20
	SPT	15
	Sammple point	75
	SJW	2
Mask Setting	: なし	
Buffer Setting(Tx)	: 送信メッセージ・バッファ	
	Buffer name	TxData_00
	Buffer No	00
	CAN ID	100
	ID Type	Std
	DLC	8
	Interrupt	Disable
	Flame Type	Data
(Rx)	: 受信メッセージ・バッファ	
	Buffer name	RxData_00
	Buffer No	01
	Mask	None
	CAN ID	200
	ID Type	Std
	Interrupt	Enable
	Flame Type	Data
	If DN-bit=1	Overwite
Other setting	: Default setting	

6.1.4　サンプル・プログラム（NEC ツール用）

ヘッダ・ファイル：sample.h

```
/*
 * #include
 */
#include<candrv.h>                                      /* Header file for CAN software driver */



/* Wait macro */
#define     WAIT( val )       {                                                                 \
                                                                                                \
                                  unsigned int  ctr ;                                       \
                                                                                                \
                                   for( ctr = ( unsigned int )( 0 ) ; ctr < val ; ctr++ ) ;  \
                                                                                                \
                              }

/* Display wait */
#define     WAIT_DISP     ( ( unsigned int )( 2000000 ) )

/* 8bit LED port info. */
#define     OUT_LED_7_2       ( P6H )
#define     OUT_LED_1_0       ( PCT )

/* LED mask info */
#define     MSK_LED_7_2       ( ( unsigned char )( 0x03 ) )
#define     MSK_LED_1_0       ( ( unsigned char )( 0xf3 ) )

/* 7seg LED port info. */
#define     SFR_LED0_L        ( PCD )
#define     SFR_LED0_H        ( PCS )

/* individual segments (0:on, 1:off) */
#define     SEG7_A            ( ( unsigned char )( ~0x01 ) )
#define     SEG7_B            ( ( unsigned char )( ~0x02 ) )
#define     SEG7_C            ( ( unsigned char )( ~0x04 ) )
#define     SEG7_D            ( ( unsigned char )( ~0x08 ) )
#define     SEG7_E            ( ( unsigned char )( ~0x10 ) )
#define     SEG7_F            ( ( unsigned char )( ~0x20 ) )
#define     SEG7_G            ( ( unsigned char )( ~0x40 ) )
#define     SEG7_DP           ( ( unsigned char )( ~0x80 ) )

/* all segments ON and OFF */
#define     LED_ON            ( ( unsigned char )( 0x00 ) )
#define     LED_OFF           ( ( unsigned char )( 0xff ) )
```

```
/* pattern */
#define     LED_PAT_0    ( SEG7_A&SEG7_B&SEG7_C&SEG7_D&SEG7_E&SEG7_F        )
#define     LED_PAT_1    (        SEG7_B&SEG7_C                             )
#define     LED_PAT_2    ( SEG7_A&SEG7_B&       SEG7_D&SEG7_E&       SEG7_G )
#define     LED_PAT_3    ( SEG7_A&SEG7_B&SEG7_C&SEG7_D&              SEG7_G )
#define     LED_PAT_4    (        SEG7_B&SEG7_C&              SEG7_F&SEG7_G )
#define     LED_PAT_5    ( SEG7_A&       SEG7_C&SEG7_D&       SEG7_F&SEG7_G )
#define     LED_PAT_6    ( SEG7_A&       SEG7_C&SEG7_D&SEG7_E&SEG7_F&SEG7_G )
#define     LED_PAT_7    ( SEG7_A&SEG7_B&SEG7_C&              SEG7_F        )
#define     LED_PAT_8    ( SEG7_A&SEG7_B&SEG7_C&SEG7_D&SEG7_E&SEG7_F&SEG7_G )
#define     LED_PAT_9    ( SEG7_A&SEG7_B&SEG7_C&SEG7_D&       SEG7_F&SEG7_G )
#define     LED_PAT_A    ( SEG7_A&SEG7_B&SEG7_C&       SEG7_E&SEG7_F&SEG7_G )
#define     LED_PAT_B    (               SEG7_C&SEG7_D&SEG7_E&SEG7_F&SEG7_G )
#define     LED_PAT_C    ( SEG7_A&              SEG7_D&SEG7_E&SEG7_F        )
#define     LED_PAT_D    (        SEG7_B&SEG7_C&SEG7_D&SEG7_E&       SEG7_G )
#define     LED_PAT_E    ( SEG7_A&              SEG7_D&SEG7_E&SEG7_F&SEG7_G )
#define     LED_PAT_F    ( SEG7_A&                     SEG7_E&SEG7_F&SEG7_G )
#define     LED_PAT_H    (        SEG7_B&SEG7_C&       SEG7_E&SEG7_F&SEG7_G )
#define     LED_PAT_G    ( SEG7_A&SEG7_B&       SEG7_D&SEG7_E&SEG7_F        )
#define     LED_PAT_J    (        SEG7_B&SEG7_C&SEG7_D                       )
#define     LED_PAT_L    (                     SEG7_D&SEG7_E&SEG7_F        )
#define     LED_PAT_N    (               SEG7_C&       SEG7_E&       SEG7_G )
#define     LED_PAT_O    (               SEG7_C&SEG7_D&SEG7_E&       SEG7_G )
#define     LED_PAT_P    ( SEG7_A&SEG7_B&              SEG7_E&SEG7_F&SEG7_G )
#define     LED_PAT_Q    ( SEG7_A&SEG7_B&SEG7_C&              SEG7_F&SEG7_G )
#define     LED_PAT_R    (                             SEG7_E&       SEG7_G )
#define     LED_PAT_U    (        SEG7_B&SEG7_C&SEG7_D&SEG7_E&SEG7_F        )
```

ソース・ファイル：sample.c

```
/*
 * #include
 */
#include"sample.h"                                  /* Header file for can sample program */
#include"config.h"                                  /* Configuration file */

/*
 * #pragma
 */
#pragma     ioreg
#pragma     interruptINTTP0CC0v0i_candrv_tx          /* Set interrupt vector address of Timer-P */
#pragma     interruptINTC0REC v0i_candrv_rx          /* Set interrupt vector address of CAN0 receive */

/*
```

```
 * Prototype declaration
 */
        void     main( void ) ;
static  void     v0s_candrv_init( void ) ;
static  void     v0i_candrv_tx( void ) ;
static  void     v0i_candrv_rx( void ) ;
static  void     v0s_start( void ) ;
static  void     v0s_ledout( unsigned char ) ;
static  void     v0s_7segout( unsigned char ) ;
static  void     v0s_error( void );

/*
 * Main function
 */
void
main( void )
{

    v0s_start() ;                                   /* Initialize sample program */
    v0s_7segout( LED_PAT_0 ) ;                      /* Output 7seg-LED */
    v0s_candrv_init() ;                             /* Initialize CAN */

    TPOCE    = 1 ;                                  /* Start TMP0 */

    while( 1 ){                                     /* Main loop */

        v0s_7segout( LED_PAT_1 ) ;                  /* Output 7seg-LED */

    }

}

/*
 * CAN initialize operation
 * (CAN software driver)
 */
static
void
v0s_candrv_init( void )
{

    CD_ER    u4t_ret ;

    u4t_ret  = CanChEnable( CD_CAN1 ) ;                     /* Enable CAN1 */
    if( u4t_ret != CD_E_OK ){
```

```
        v0s_error() ;

    }

    u4t_ret  = CanChInit( CD_CAN1 ) ;                          /* Initialize CAN1 */
    if( u4t_ret != CD_E_OK ){

        v0s_error() ;

    }

    u4t_ret  = CanChSetNrmMode( CD_CAN1 ) ;                    /* Set normal mode CAN1 */
    if( u4t_ret != CD_E_OK ){

        v0s_error() ;

    }

    return ;

}

/*
 * CAN transmit interrupt operation
 * (CAN software driver)
 */
__interrupt
void
v0i_candrv_tx( void )
{

        CD_ER             u4t_ret ;
        CD_DAT            u1t_TxDatabuf[ 8 ] ;
static  unsigned char u1t_txdata = ( unsigned char )( 0x00 ) ;

    u4t_ret  = CanMsgGetTxInfo( CD_CAN1, TxData_00 ) ;         /* Check TRQ bit */
    if( u4t_ret == CD_TRUE ){

        return ;

    }

    {                                                          /* Set Tx data */
```

```
        signed int    s4t_ctr;

        for( s4t_ctr = ( signed int )( 0 ) ; s4t_ctr < ( signed int )( 8 ) ; s4t_ctr++ ){

            u1t_TxDatabuf[ s4t_ctr ]   = u1t_txdata++ ;

        }

    }

                                                        /* Set Tx message */
    u4t_ret  = CanMsgSetDat( CD_CAN1, TxData_00, &u1t_TxDatabuf[ 0 ] ) ;
    if( u4t_ret != CD_E_OK ){

        v0s_error() ;

    }

    u4t_ret  = CanMsgTxReq( CD_CAN1, TxData_00 ) ;              /* Set Tx request */
    if( u4t_ret != CD_E_OK ){

        v0s_error() ;

    }

    return ;

}

/*
 * CAN receive interrupt operation
 * (CAN software driver)
 */
__interrupt
void
v0i_candrv_rx( void )
{

    CD_ER    u4t_ret ;
    CD_DAT   u1t_RxDatabuf[ 8 ] ;
    CD_DLC   s1t_RxDlc ;

    u4t_ret = CanChSrcRxInfo( CD_CAN1, RxData_00 ) ;            /* Search Rx message */
    if( ( u4t_ret & CD_E_FLG ) == ( unsigned int )( 0x00000000 ) ){
```

```
        CD_BUFNO u1t_bufno ;

        u1t_bufno= u4t_ret;
                                                        /* Get Rx message */
        u4t_ret = CanMsgGetDatDlc( CD_CAN1, u1t_bufno, &u1t_RxDatabuf[ 0 ], &s1t_RxDlc ) ;
        if( u4t_ret != CD_E_OK ){

            v0s_error() ;

        }

        v0s_ledout( u1t_RxDatabuf[ 0 ] ) ;                  /* Display Rx message */

    }

    return ;

}

/*
 * Device initialization function
 */
static
void
v0s_start( void )
{

extern  signed int        _rcopy( unsigned long *, signed long ) ;
extern  unsigned long _S_romp;

    VSWC     = ( unsigned char  )( 0x01 ) ;                 /* System clock = 20MHz */
    RCM          = ( unsigned char  )( 0x01 ) ;             /* Stop ring-OSC */
    WDTM2        = ( unsigned char  )( 0x00 ) ;             /* Stop WDT2 */
    BPC          = ( unsigned short )( 0x8ffb );            /* Set programmable peripheral I/O area */

    ( void )_rcopy( &_S_romp, ( signed long )( -1 ) ) ;     /* Copy ".data" section in ROM to RAM */

    v0s_7segout( LED_OFF ) ;                                /* clear 7seg-LED */
    v0s_ledout( ( unsigned char )( 0x00 ) ) ;               /* clear 8bit-LED */

    PMCD     = ( unsigned char )( 0xf0 ) ;                  /* Port CD (7seg-LED) */
    PMCCS        = ( unsigned char )( 0x00 ) ;              /* Port CS (7seg-LED) */
    PMCS     = ( unsigned char )( 0x00 ) ;
    PMCCT        = ( unsigned char )( 0x00 ) ;              /* Port CT (8bit-LED) */
    PMCT     = ( unsigned char )( 0x00 ) ;
```

```
    PMC6H         = ( unsigned char )( 0x00 ) ;              /* Port 6H (8,1bit-LED) */
    PM6H      = ( unsigned char )( 0x00 ) ;

                                                             /* Port setting for CAN1 */
    PFC3L         &= ( unsigned char  )( 0xe7 ) ;            /* Clear PFC33,34 */
    PFCE3L        |= ( unsigned char  )( 0x18 ) ;            /* Set PFCE33,34 */
    PMC3L         |= ( unsigned short )( 0x00d8 ) ;          /* Set PMC37,36,34,33 */

                                                             /* Setup Timer P */
    TP0CE         = 0 ;                                      /* Stop TMP0 */
    TP0CTL0       = ( unsigned char  )( 0x07 ) ;
    TP0CTL1       = ( unsigned char  )( 0x00 ) ;
    TP0IOC0       = ( unsigned char  )( 0x00 ) ;
    TP0IOC1       = ( unsigned char  )( 0x00 ) ;
    TP0IOC2       = ( unsigned char  )( 0x00 ) ;
    TP0OPT0       = ( unsigned char  )( 0x00 ) ;
    TP0CCR0       = ( unsigned short )( 0xffff ) ;           /* TMP0 : 419.424(msec) */
                                                             /*        = ( 1 / ( 20MHz / 128 ) * 65535 */
    SELCNT0       = ( unsigned char  )( 0x00 ) ;
    SELCNT1       = ( unsigned char  )( 0x00 ) ;

    __EI() ;                                                 /* Enable global interrupt */
    CORECMK       = 0 ;                                      /* Enable receive complete interrupt */
    TP0CCMK0 = 0 ;                                           /* Enable Timer-P interrupt */

}

/*
 * 8bit-LED port output function
 */
static
void
v0s_ledout(                                                  /* 8bit-LED output */
    unsigned char u1t_dat
    )
{

    OUT_LED_7_2   = ( ( ~u1t_dat ) | MSK_LED_7_2 ) ;         /* 8bit-LED : bit7-bit2 */
    OUT_LED_1_0   = ( ( ( ~u1t_dat ) << 2 ) | MSK_LED_1_0 ) ;  /* 8bit-LED : bit1-bit0 */

    return ;

}

/*
```

```
 * 7seg-LED port output function
 */
static
void
v0s_7segout(                                                    /* 7seg-LED output */
    unsigned char u1t_pat
    )
{

                                                                /* 7seg-LED : a - d */
    SFR_LED0_L    = ( ( unsigned char )( 0xf0 ) | u1t_pat & ( unsigned char )( 0x0f ) ) ;
                                                                /* 7seg-LED : e - dot */
    SFR_LED0_H    = ( ( unsigned char )( 0x0f ) | u1t_pat & ( unsigned char )( 0xf0 ) ) ;

    return ;

}

/*
 * Error function
 */
static
void
v0s_error( void )                                               /* Error message output */
{

    while( 1 ){

        v0s_7segout( LED_PAT_E ) ;                              /* Output 7seg-LED "E" */
        WAIT( WAIT_DISP ) ;
        v0s_7segout( LED_PAT_R ) ;                              /* Output 7seg-LED "R" */
        WAIT( WAIT_DISP ) ;
        v0s_7segout( LED_PAT_R ) ;                              /* Output 7seg-LED "R" */
        WAIT( WAIT_DISP ) ;
        v0s_7segout( LED_PAT_0 ) ;                              /* Output 7seg-LED "0" */
        WAIT( WAIT_DISP ) ;
        v0s_7segout( LED_PAT_R ) ;                              /* Output 7seg-LED "R" */
        WAIT( WAIT_DISP ) ;
        v0s_7segout( LED_OFF ) ;                                /* Output 7seg-LED " " */
        WAIT( WAIT_DISP ) ;

    }

}
```

付録　改版履歴

本文欄外の★印は，本版で改訂された主な箇所を示しています。
この"★"をPDF上でコピーして「検索する文字列」に指定することによって，改版箇所を容易に検索できます。

付. 1　本版で改訂された主な箇所

(1/2)

箇　所	内　　容
p. 11	1. 3　CAN**ソフトウエア・ドライバの種類**の説明を修正
p. 12, 13	1. 4　**実行環境**の説明を修正
p. 14	1. 5　**開発環境**の説明を修正
p. 15	**表**2－1　CAN**ソフトウエア・ドライバの提供方法**を修正 2. 2. 2　**提供媒体のセット**の説明を修正 2. 2. 3　CAN**ソフトウエア・ドライバのインストール**を削除
p. 16	**図**2－1　CAN**ソフトウエア・ドライバのディレクトリ構成**を修正，**備考**を削除
p. 17	2. 3. 3　**サンプル・プログラム**の説明を修正 **図**2－2　**サンプル・プログラムのディレクトリ構成**を修正
p. 25	**図**4－1　**メイン画面**を修正
p. 26	4. 4. 1　**デバイスの選択**の説明を修正
p. 27	**図**4－2　**デバイス選択メニュー画面**を修正
p. 29	**図**4－3　**ボー・レート設定画面**（V850-aFCAN, V850-DCAN, 78K0-aFCAN）を修正
p. 31	**図**4－4　**ボー・レート設定画面**（V850-FCAN）を修正
p. 33	**図**4－5　**ボー・レート設定画面**（78K0-DCAN）を修正
p. 60	**図**4－21　**出力オプション設定画面**を修正
p. 61	**図**4－22　**コード生成起動画面**を修正
p. 62	**図**4－23　**プロジェクト・ファイル保存，呼び出し画面**を修正
p. 63-65	**表**4－1　**エラー・コード一覧**を修正
p. 66	**表**4－2　**ワーニング・コード一覧**を修正
p. 68	5. 1. 1　**初期化／設定関連**（6**種類**）を修正 5. 1. 2　**動作モード関連**（2**種類**）を修正
p. 74	**表**5－6　**単チャネル仕様**CAN**ソフトウエア・ドライバ関数一覧**を修正
p. 78	**表**5－8　**初期化／設定**を修正
p. 79	5. 8. 1　**初期化／設定** CanChEnable **【概要】から注**を削除 **【説明】の注**1を修正，**注**2を削除
p. 86	5. 8. 1　**初期化／設定** CanChShutdown**を追加**
p. 87	5. 8. 1　**初期化／設定** CanAllShutdown**を追加**
p. 88	**表**5－9　**動作モード**を修正

（2/2）

箇　所	内　　容
p. 92	5. 8. 2　**動作モード** CanChSetInitMode**を追加**
p. 125, 126	5. 8. 6　CAN**チャネル状態取得** CanChGetBusStatus**の•** MSB = 0**時の各ビットの説明，【使用例】を変更**
p. 127	**第**6**章　サンプル・プログラムを修正**

付. 2　前版までの改訂履歴

前版までの改版履歴を次に示します。なお，適用箇所は各版での章を示します。

(1/2)

版　数	内　　容	適用箇所
第2版	対象デバイスにV850/SF1, V850/DB1, μPD780824B(A), 780826B(A), 780828B(A), 78F0828B, 78F0876を追加	全般
	ドライバ・ライブラリをドライバ・ソース・ファイルに変更	
	デバイスに関する資料を変更	はじめに
	開発ツールに関する資料（ユーザーズ・マニュアル）を変更	
	1. 3　CANソフトウエア・ドライバの種類の記述を変更	第1章　製品概要
	1. 4（3）（a）全14関数のメモリ容量の記述を変更	
	1. 4（3）（b）サンプル・プログラムの6関数のメモリ容量の記述を変更	
	図2－1　CANソフトウエア・ドライバのディレクトリ構成を変更	第2章　インストレーション
	図2－2　サンプル・プログラムのディレクトリ構成を変更	
	図3－2　システム構築手順を変更	第3章　システム構築
	3. 2. 1（2）コンフィギュレータの出力データの記述を変更	
	図3－3　アプリケーション・プログラムとCANソフトウエア・ドライバ／コンフィギュレータとの相関関係を変更	
	3. 2. 4　ロード・モジュール・ファイルの生成の記述を変更	
	図4－1　メイン画面を変更	第4章　コンフィギュレーション
	図4－2　デバイス選択メニュー画面を変更	
	4. 4. 2　ボー・レートの設定を変更	
	4. 4. 3　マスクの設定を変更	
	4. 4. 4　メッセージ・バッファの設定を変更	
	4. 4. 5　その他の設定を変更	
	4. 4. 6（1）CANソフトウエア・ドライバ・ソース・ファイルの出力オプション設定を変更	
	図4－23　プロジェクト・ファイル保存，呼び出し画面を変更	
	4. 5　エラー／ワーニング・メッセージ一覧を変更	
	5. 1. 1　初期化・設定関連（4種類）に関数を追加	第5章　ドライバ関数
	5. 1. 3　バッファ・データ取得関連（4種類）に関数を追加	
	5. 1. 4　バッファ・データ設定関連（4種類）に関数を追加	
	5. 1. 5　送受信確認関連（4種類）に関数を追加	
	表5－3　パラメータ用マクロにマクロを追加	
	5. 5　単チャネル仕様CANソフトウエア・ドライバ関数を追加	
	5. 6　パフォーマンス改善版CANソフトウエア・ドライバ関数を追加	
	5. 8. 1　初期化／設定 次の関数に記述を追加 CanChEnable, CanChInit	
	5. 8. 2　動作モード CanChGetModeに注を追加	
	5. 8. 3　バッファ・データ取得 次の関数に記述を追加 CanMsgGetDatDlc, CanMsgGetIdDatDlc	

（2/2）

版　数	内　　容	適用箇所
第2版	5. 8. 4　**バッファ・データ設定** **次の関数に記述を追加** CanMsgSetDat, CanMsgSetIdDatDlc	**第5章　ドライバ関数**
	5. 8. 6　CAN**チャネル状態取得** **次の関数の記述を変更** CanChGetStatus, CanChClrStatus, CanChGetBusStatus	
	6. 1 .3　**スタートアップ・ファイルでの設定（**V850ES/FJ2**のデバイス依存箇所）の記述**を変更	**第6章　サンプル・プログラム**

〔メ　モ〕

【発　行】
NECエレクトロニクス株式会社
〒211-8668　神奈川県川崎市中原区下沼部1753
電話（代表）：044(435)5111

お問い合わせ先

【ホームページ】

NECエレクトロニクスの情報がインターネットでご覧になれます。
URL(アドレス)　http://www.necel.co.jp/

【営業関係，技術関係お問い合わせ先】

半導体ホットライン
（電話：午前 9:00～12:00，午後 1:00～5:00）
電　話　：044−435−9494
E-mail　：info@necel.com

【資料請求先】

NECエレクトロニクスのホームページよりダウンロードいただくか，NECエレクトロニクスの販売特約店へお申し付けください。