JobGen Plus ユーザマニュアル

For Microsoft Windows

Version 5.8

Document Revision 5.0, May 2001

翻訳: ユニテック・ジャパン株式会社、2005 年 12 月

目　　次

JobGen Plus の基礎

このセクションでは以下を学習します:

- JobGen Plus の概要
- JobGen Plus のインストールと必要なシステム
- JobGen Plus の動作環境

第 1 章 概要

JobGen Plus の概要

JobGen Plus は皆様が独自のデータ収集アプリケーションを作ることができるプログラミングツールです。これは紙にプランを書くようにできるだけ簡単にデータ収集のアプリケーションを開発することができるようにするという考えに沿って設計されています。JobGen Plus はユーザフレンドリーなプログラミング環境、完全なプログラミングツールのセット、そしてプログラマでない方にでも多彩なデータ収集のソリューションを提供いたします。

JobGen Plus は Windows アプリケーションなので、Windows の良く知られたユーザフレンドリーな操作環境の利点を受けることができます。ユーザは単にマウスボタンをクリックするだけで、 JobGen Plus のプログラミングツールのすべてにアクセスすることができます。この強力なプログラミングツールのセットは、データ収集アプリケーションの設計に含まれるすべてのステップを深く検討することによって開発されました。これは JobGen Plus ユーザの誰もが、工業、学校、小売店舗、倉庫 – ほとんどどこでも – 何らプログラミングの経験なしに、御自分のデータ収集アプリケーションを容易に開発できるようにします。

また、JobGen Plus はより複雑なデータ収集アプリケーションを必要とする上級ユーザに**プログラミング言語**オプションを提供します。このオプションはデータ収集アプリケーションで他の開発ツールによって加えられた制限からプログラマを解放します。しかし、ほとんどのユーザは、各種のデータ収集アプリケーション、例えば、倉庫管理から売り上げの記録までを、JobGen Plus だけを使用することによって開発することができます。

JobGen Plus のコンポーネント

JobGen Plus によって作られたデータ収集アプリケーションは二つの主要なコンポーネントを含んでいます: 最初の一つは**ノード**と呼ばれ、データ収集アプリケーション中の**プロセス**を表します。

二つ目は**リンク**と呼ばれ、あるプロセスから他のプロセスへの**移動**を表します。これらの JobGen Plus アプリケーションの主要なエレメントは、以下の章で詳しく説明します。 このセクションでは、ノードとリンクの概念を紹介することにとどめておきます。

各ノード はデータ収集における一つのタスクを表します。データ収集アプリケーションにおけるタスクの例は、データ収集のメニューの選択; データコレクタに項目番号を入力; あるいはデータコレクタからホストコンピュータへ収集した在庫情報を送信する等を含んでいます。JobGen Plus はデータ収集アプリケーションにおける各種のタスクを実行するために 9 つのタイプのノードを持っています。これらは、コメントノード、メニューノード、収集ノード、演算ノード、編集ノード、消去ノード、アップロードノード、プログラムノード、そしてジョブ実行ノードです。

コメントノードはフローチャートに直接注記を貼り付けることができます。

メニューノードはメッセージ、指示あるいはユーザが選択するアプリケーションの主要な機能の表示のためのスペースを提供します。

収集ノードはデータコレクタでユーザによって入力されたバーコードスキャナまたはキーパッドからの情報を保存するために使用されます。

演算ノードはデータの計算に使用されます。

編集ノードはデータコレクタに記録されたデータの表示あるいは変更を可能にします。

消去ノードはデータコレクタに記録されたデータを削除します。

アップロードノードはデータコレクタからホストコンピュータへ収集した情報を送信します。

プログラムノードはアプリケーション開発者が C 言語モジュールを追加することによってプロジェクトを強化することを可能にします。

ジョブ実行ノードは他の JobGen Plus ジョブを読み込んで実行するために使用します。

リンクは二つあるいはそれ以上のノード(プロセス)間の移動を行います。リンクはすべてのプロセスを一つの JobGen Plus アプリケーションに接続するだけではなく、一つのプロセス(ノード）から他へパスと方向をも提供します。どのようにして、そして何時移動が起こるかについての情報は各リンクの中に記載されます。

第2章 インストール

必要なシステム

以下は JobGen Plus のインストールに必要な最低条件です。:

- DOS/V 互換コンピュータ、Intel Pentium 互換 CPU 以上
- 128 MB メモリ
- Hard Disk に 10 MB 以上の空き容量があること
- VGA モニター
- 1 シリアルポート（COM1、COM2、COM3 等々）
- 1 マウス（あるいは MS Windows を操作するためのポインティングデバイス）
- Windows Me/2000/XP

MS Windows Me/2000/XP へのインストール

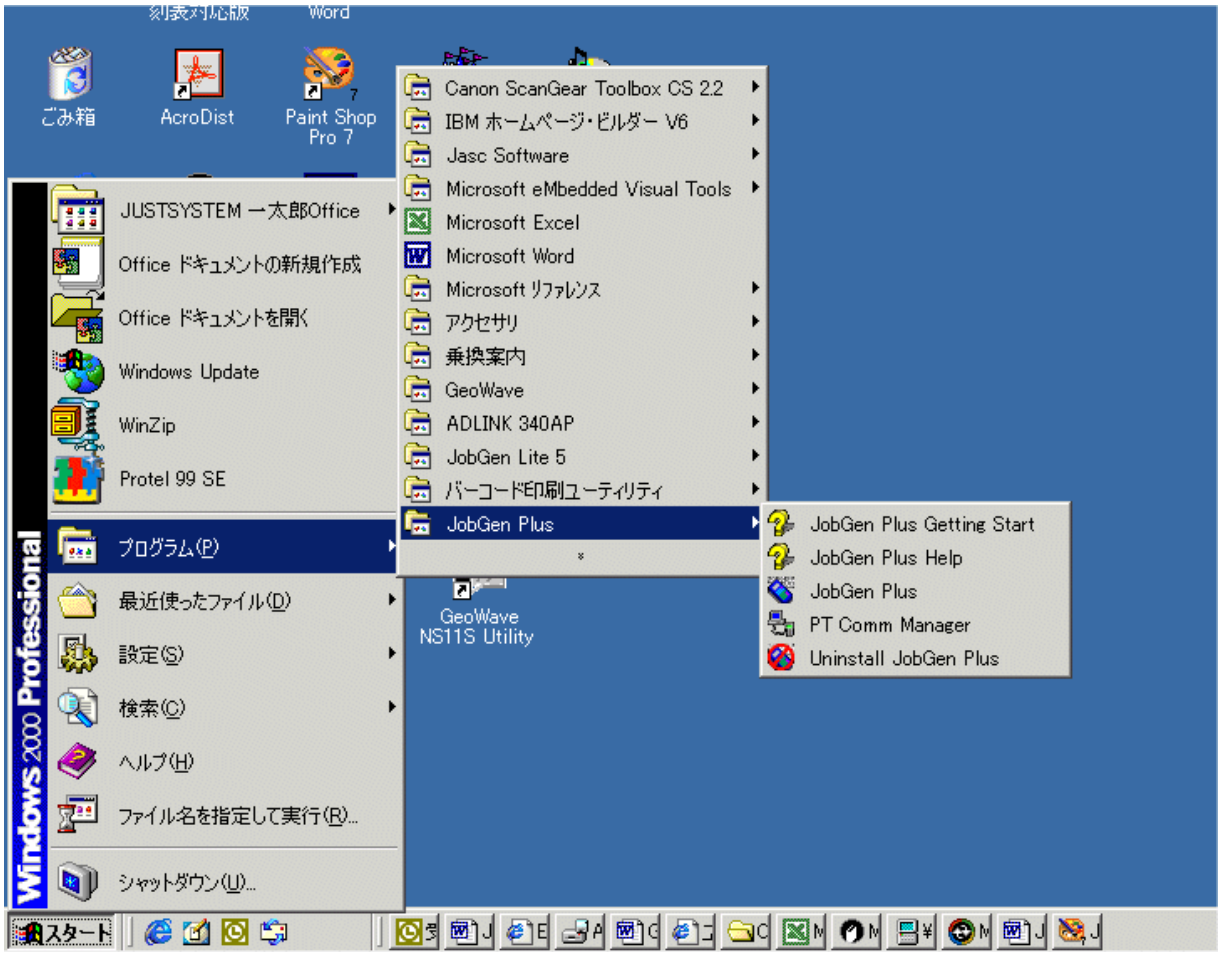
1. JobGen Plus の CD-ROM を CD-ROM ドライブに入れて下さい。または、ユニテック社の Web サイトから JobGen Plus プログラムダウンロードして適当なフォルダに置いて下さい。
2. **スタート**をクリックし、そして**ファイル名を指定して実行**を選択して下さい。
3. JobGen Plus CD-ROM の入っているドライブまたはフォルダをタイプして、“**Setup**”をタイプして下さい。
4. **Enter** キーを押し、 **OK** をクリックして下さい。
5. JobGen Plus は設定の初期化を実行し、JobGen Plus システムファイルを保存するディレクトリ名を入力するように求めます。標準のディレクトリ名は JGPLUS5 です。
6. セットアッププログラムはファイルを指定したディレクトリへコピーを始めます。
7. セットアッププログラムが終了したときに、JobGen Plus のプロクラムフォルダとアイコンを作成します。インストールが終わります。

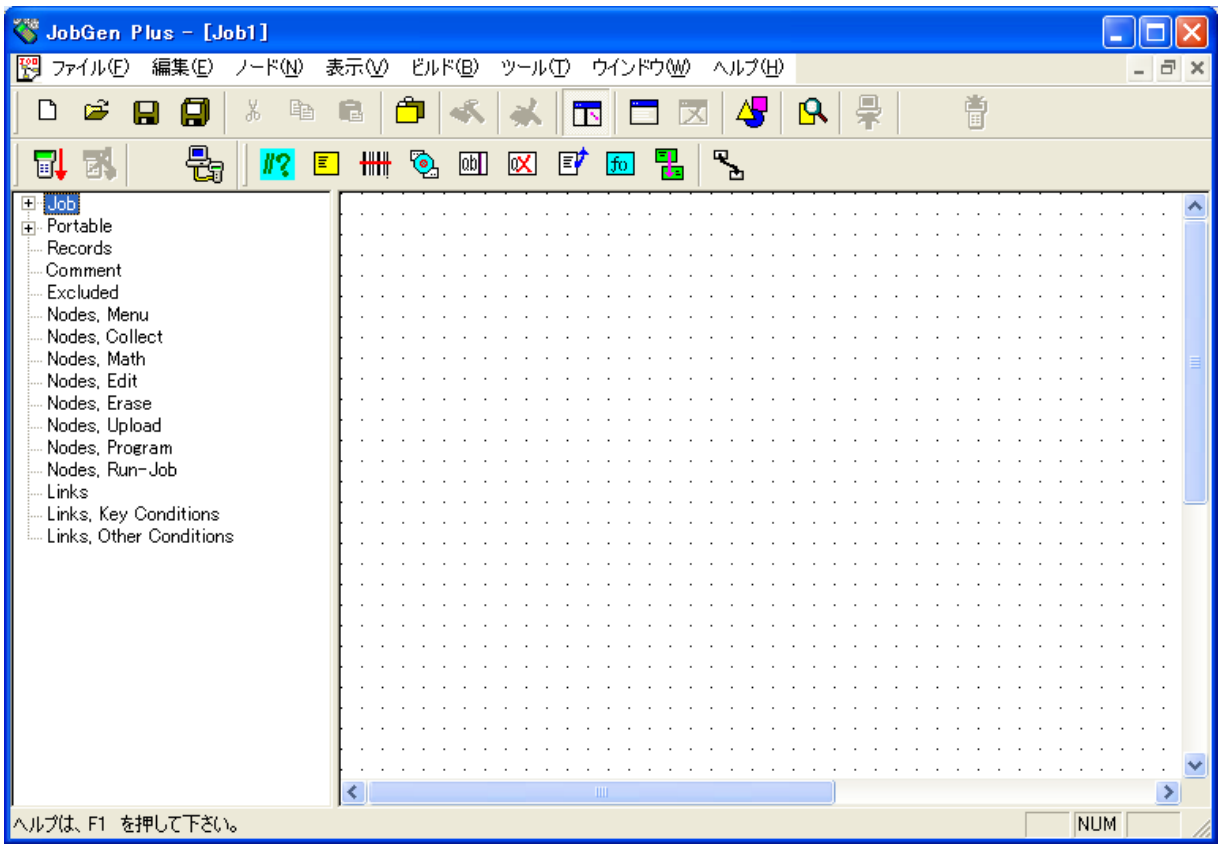
第 3 章 JobGen Plus を使う

MS Windows Me/2000/XP から JobGen Plus を実行

1. タスクバーのスタートボタンをクリックして、プログラムを選択します。
2. マウスカーソルを JobGen Plus プログラムを含むプログラムグループに移動します。
3. プログラムを実行するために JobGen Plus アイコンをクリックします。

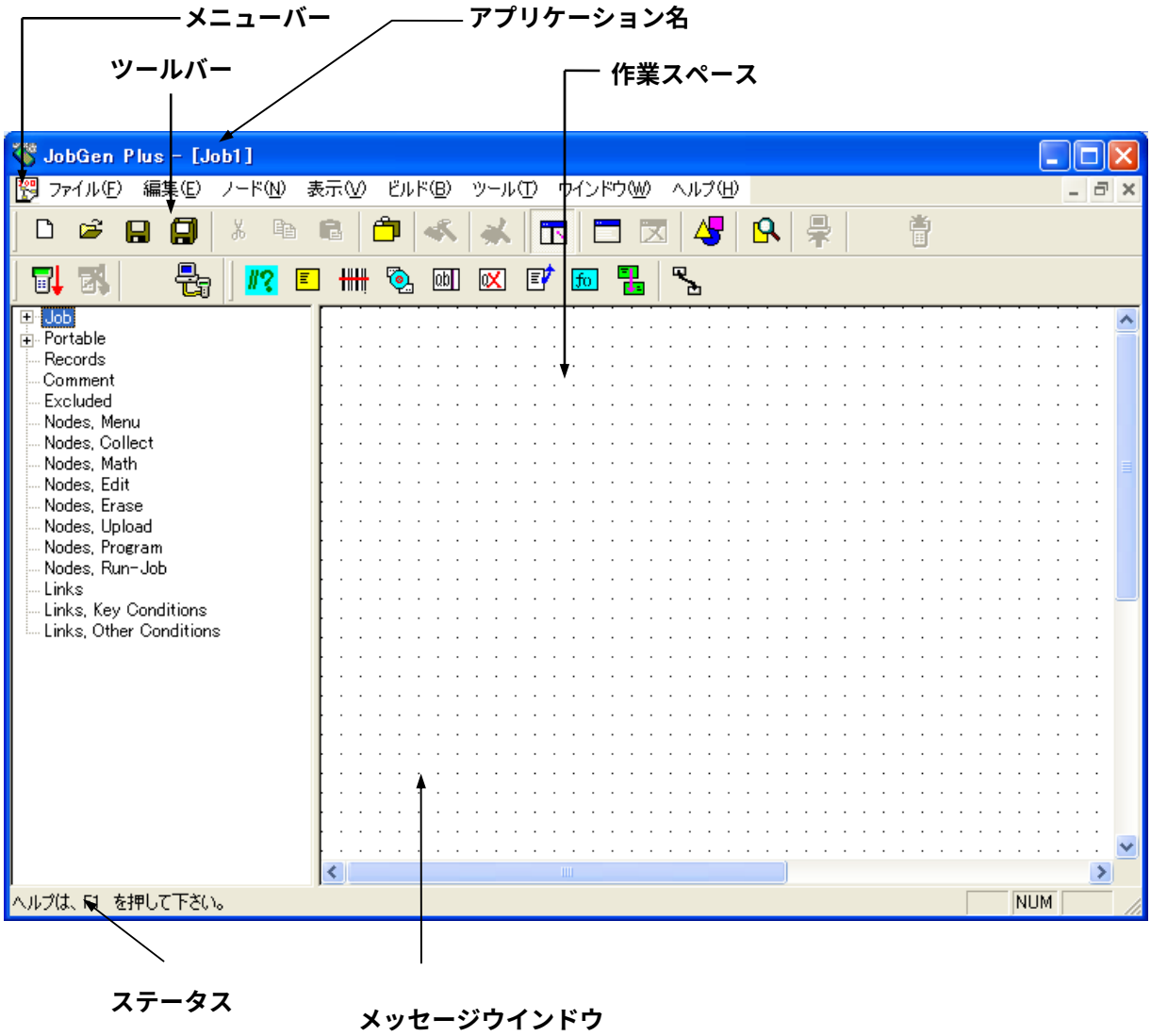
(あるいは、デスクトップにショートカットアイコンを作っておき、これをダブルクリックします。)



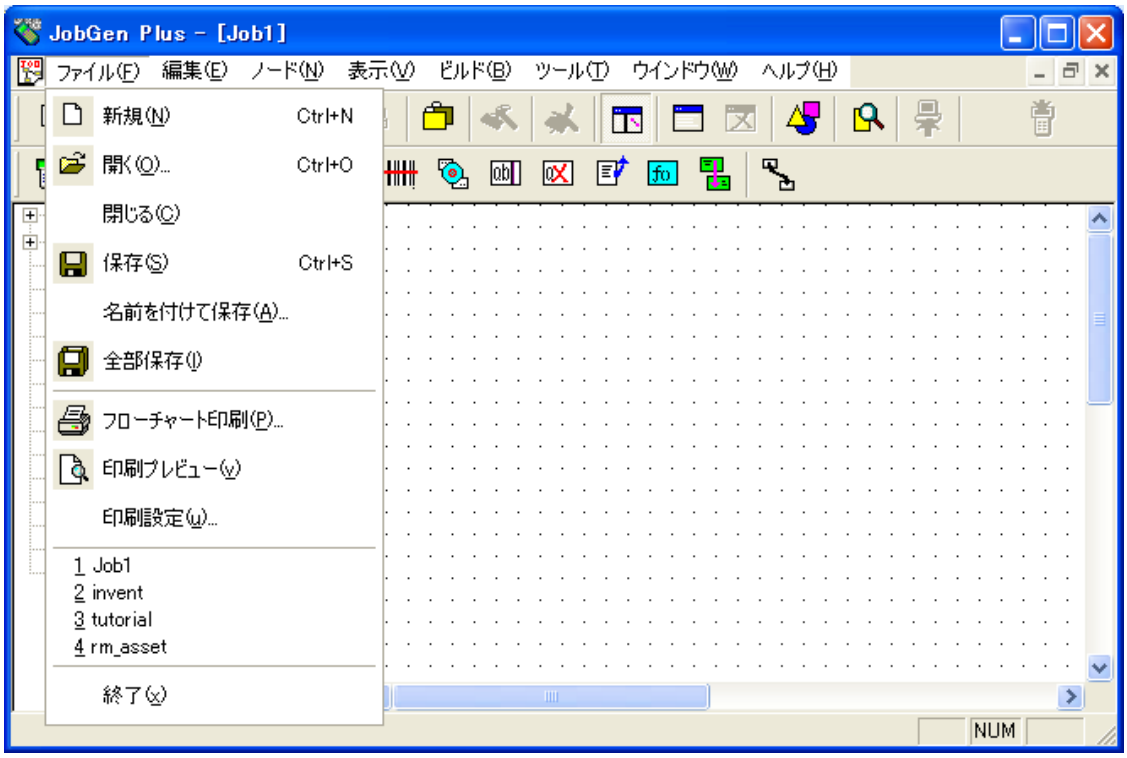


ここで、JobGen Plus はデータ収集のアプリケーション開発の準備ができました。

JobGen Plus の作業環境

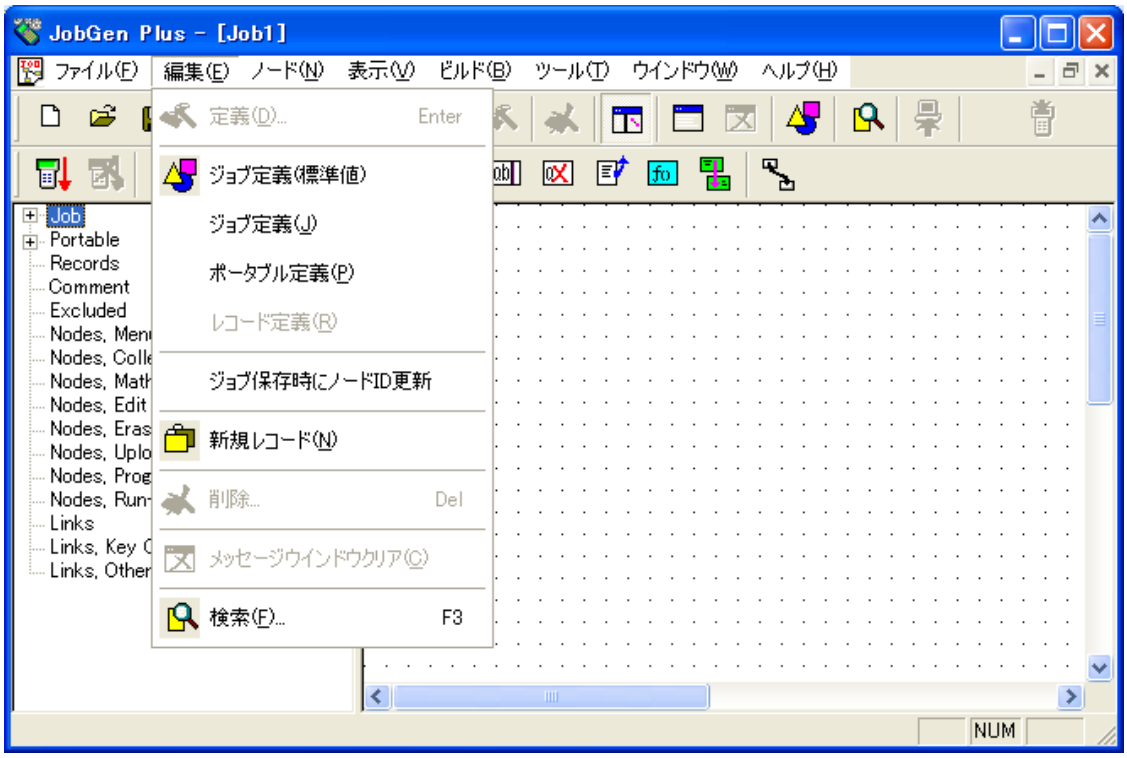


1. ファイルメニュー



新規	新しいジョブを作成する。
開く	すでにあるジョブファイルを開く。
閉じる	現在のジョブウインドウを閉じる。
保存	現在のジョブの内容を保存する。
名前を付けて保存	現在のジョブを別なファイル名で保存する。
全部保存	すべての開いたジョブの内容を保存する。
-	
フローチャート印刷	現在のジョブのフローチャートを印刷する。
印刷プレビュー	印刷出力をスクリーン上で見る。
印刷設定	印刷オプションの設定。
-	
最近使ったファイル	最近使ったファイルをリストする。
-	
終了	JobGen Plus プログラムを終了する。

2. 編集メニュー



定義 ノード、リンク、あるいはレコードのいずれかの現在選択されているオブジェクトを定義する。

-

ジョブ定義(標準値) ジョブについて、標準値の設定を定義する。

ジョブ定義 現在のジョブを定義する。

ポータブル定義 ポータブルターミナルの設定を定義する。

レコード定義 レコード設定を定義する。

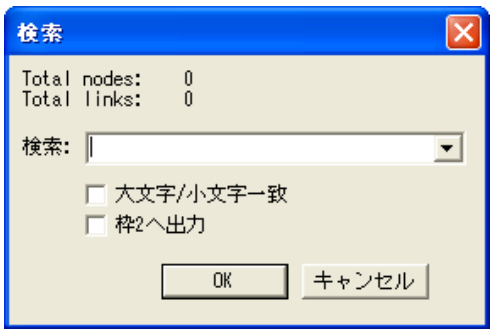
-

新規レコード 新しいレコードを作成する。JobGen Plus は複数レコードをサポートしている。

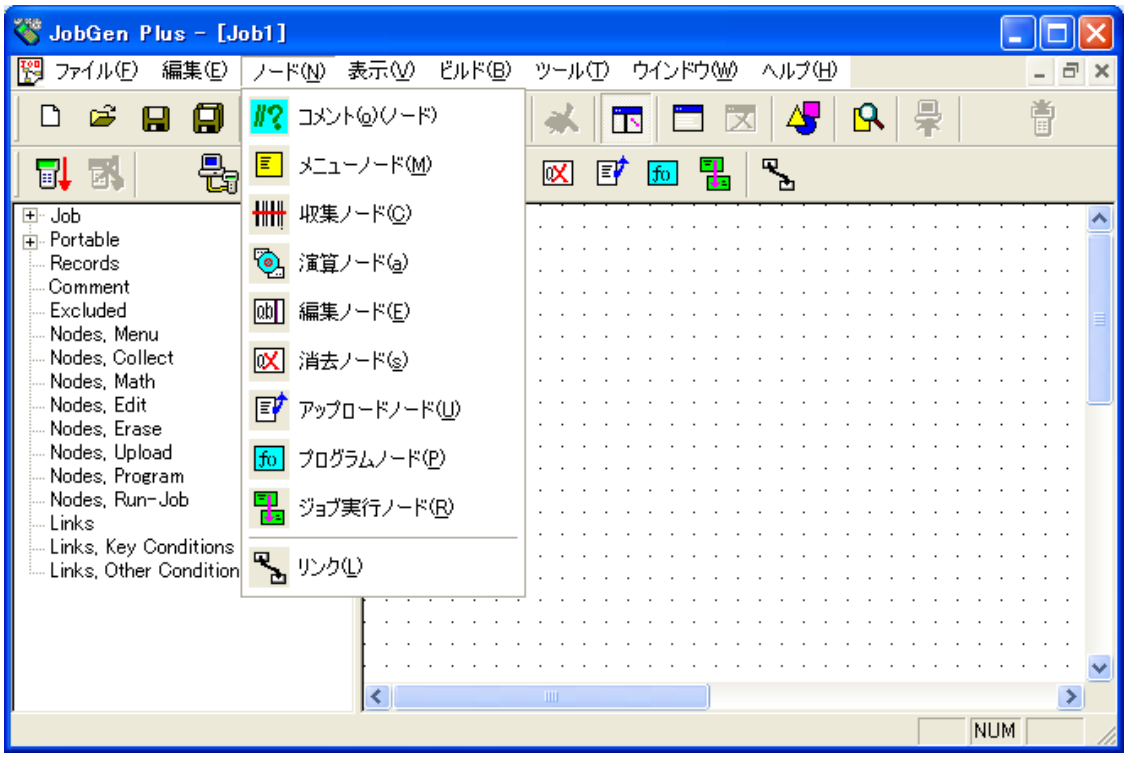
削除 ノード、リンク、あるいはレコードのいずれかの現在選択されているオブジェクトを削除する。

メッセージウインドウクリア メッセージウインドウに表示されているすべてのメッセージをクリアします。

検索 現在のジョブ中のテキストを検索する。結果はメッセージスクリーン検索に表示されます。“枠 2 へ出力”がチェックされた場合、すべての結果はメッセージスクリーン検索 2 に表示されます。メッセージウインドウで見つかった結果をダブルクリックすると関連の定義ダイアローグ・ウインドウが開きます。

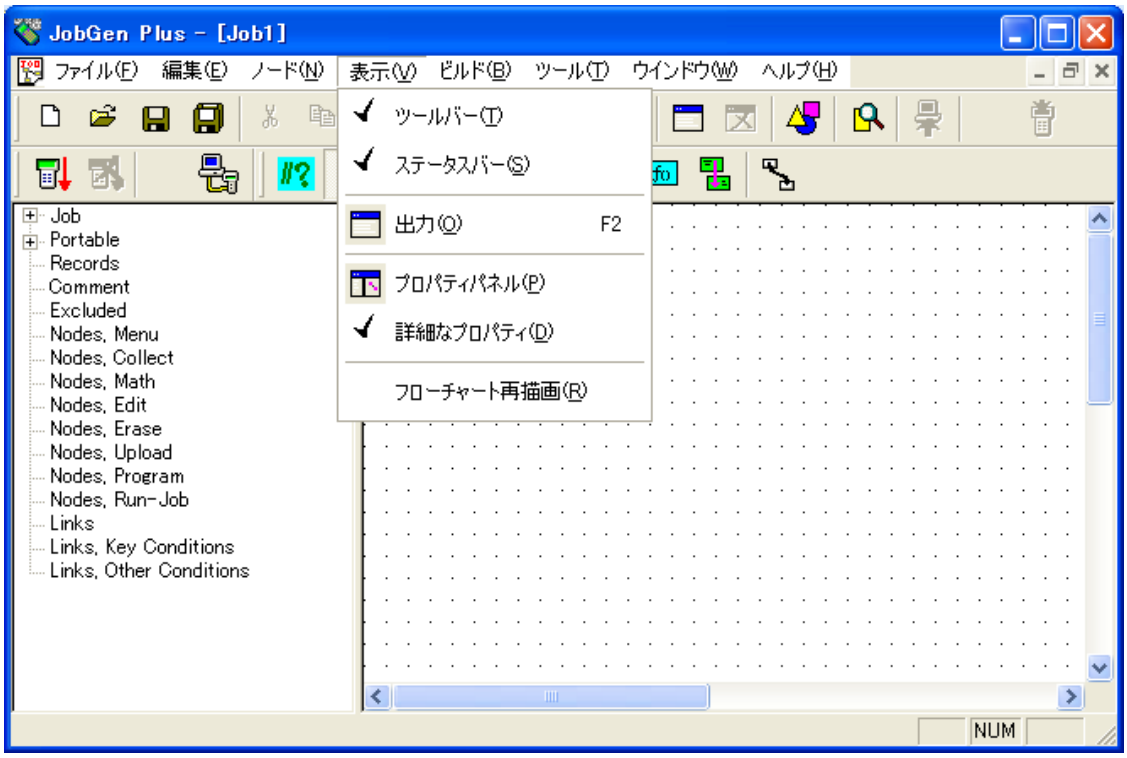


3. ノードメニュー



コメント(ノード) コメントノードを作成する。
メニューノード メニューノードを作成する。
収集ノード 収集ノードを作成する。
演算ノード 演算ノードを作成する。
編集ノード 編集ノードを作成する。
消去ノード 消去ノードを作成する。
アップロードノード アップロードノードを作成する。
プログラムノード プログラムノードを作成する。
ジョブ実行ノード ジョブ実行ノードを作成する。
-
リンク リンクを作成する。

4．表示メニュー



ツールバー　　このコマンドがチェックされた場合、ツールバーを表示する。

ステータスバー　　このコマンドがチェックされた場合、ステータスバーを表示する。

-

出力　　メッセージウインドウを開く。

-

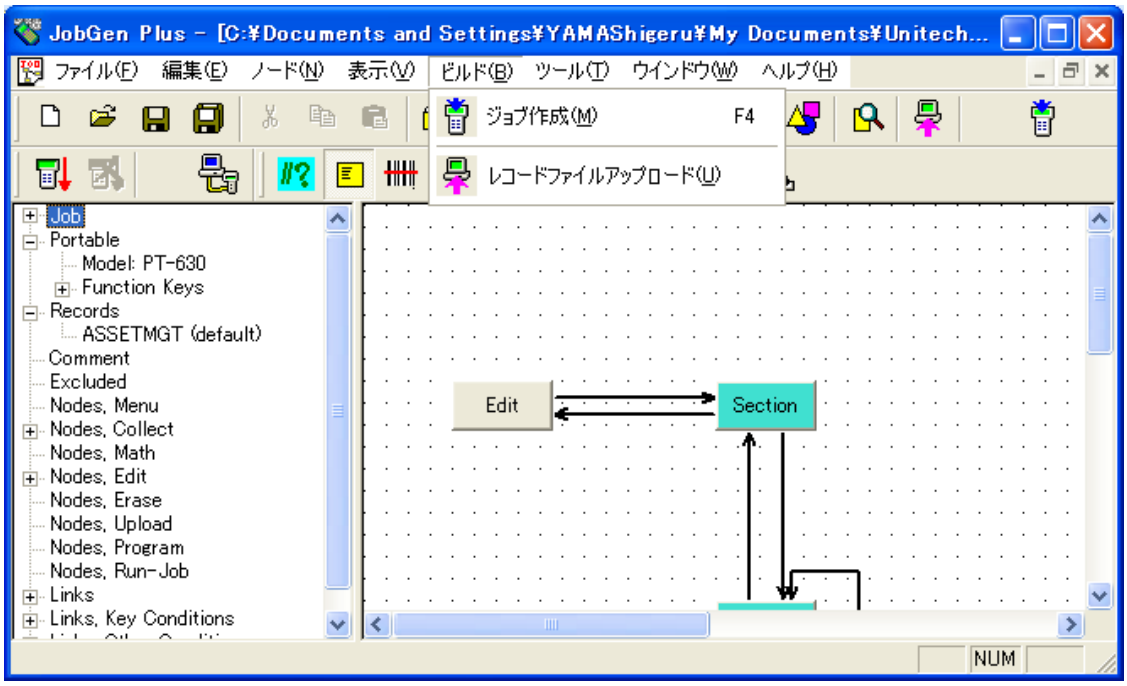
プロパティパネル　　プロパティパネルを開く。

詳細なプロパティ　　プロパティパネルで詳細な情報の表示を可能にする。

-

フローチャート再描画　　全体のフローチャートを再描画する。

5. ビルドメニュー

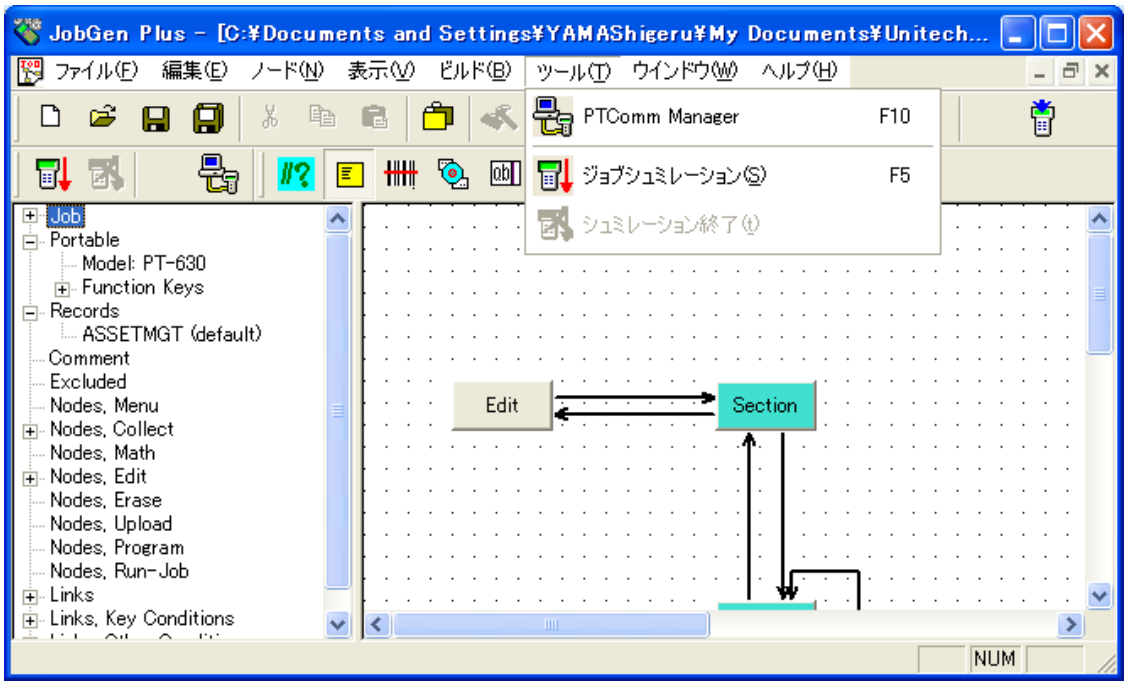


ジョブ作成　　ジョブの実行コードを生成し、そしてコードとすべてのデータファイルを含むファイルをポータブルターミナルにダウンロードする。

-

レコードファイルアップロード　　レコードファイルをアップロードする。

6. ツールメニュー



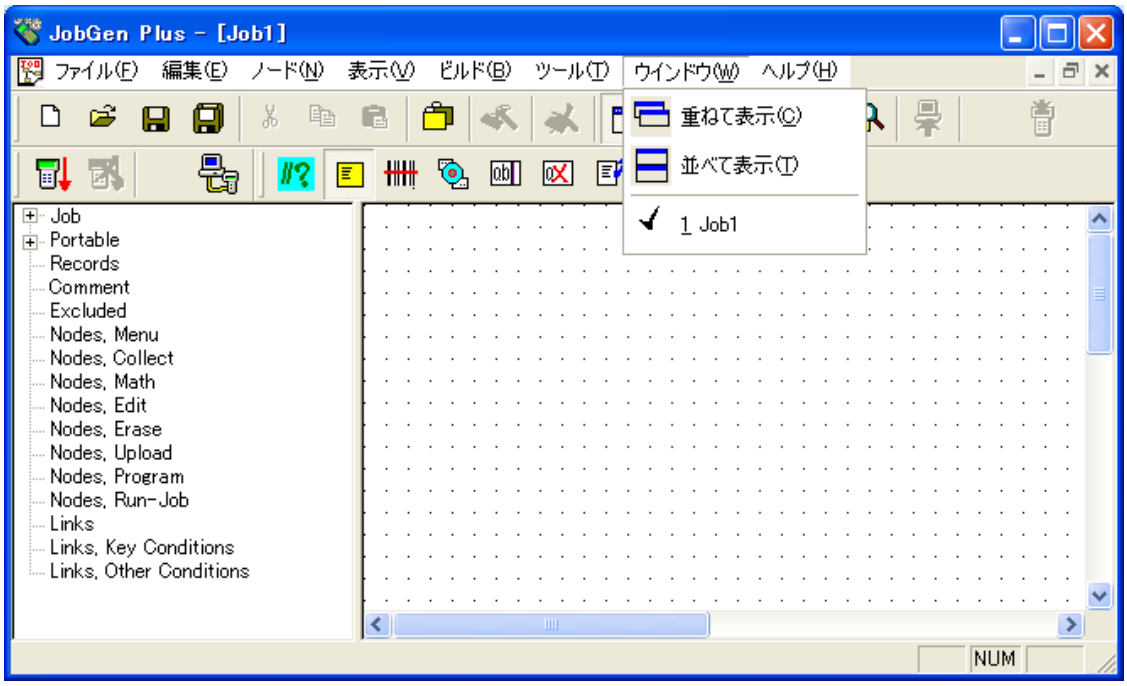
PTComm Manager　　PTComm マネージャ(通信管理)アプリケーションを実行する。

-

ジョブシュミレーション　　ジョブシュミレーションを開始する。

シュミレーション終了　　ジョブシュミレーションを終了する。

7. ウインドウメニュー



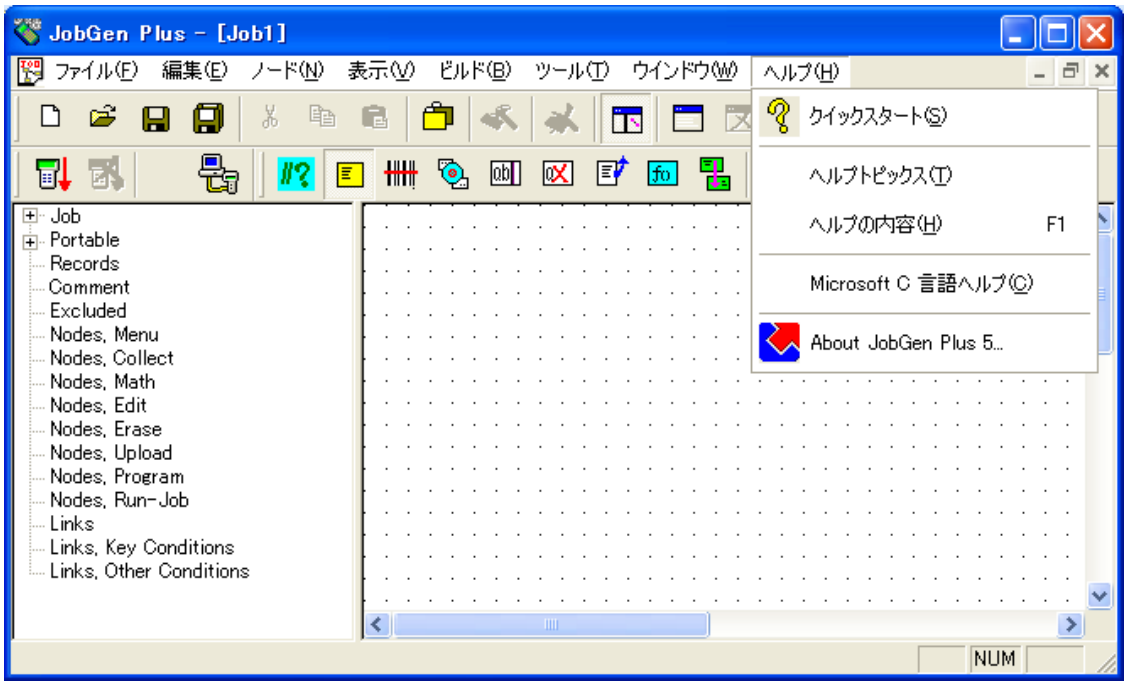
重ねて表示　　すべてのウインドウを重ねて表示する。

並べて表示　　すべてのウインドウを並べて表示する。

-

開かれたウインドウ　　すべての開かれたウインドウのタイトルをリストする。

8. ヘルプメニュー



クイックスタート JobGen Plus の概要を表示する。

-

ヘルプトピックス ヘルプのトピックスを表示する。

ヘルプの内容 JobGen Plus のヘルプを表示する。

-

Microsoft C 言語ヘルプ *Microsoft C Language Run-Time Reference* のヘルプを表示する。

-

About JobGen Plus 5 JobGen Plus のバージョン番号と版権を表示する。

ジョブの作成

このセクションでは以下を学習します:

- ジョブの設計の仕方
- ジョブの作成方法
- 実行可能なジョブの作成方法

第 4 章 ジョブの設計

ジョブを作成する前に、ユーザは作成に必要な各種のタスク(仕事)のすべてについてまず分析をしなければなりません。タスクの例としては、データの収集、選択を行う、データを送信する、データのチェック、メッセージの表示などです。次に、ユーザはこれらのタスクをどのように実行するかのプランを立てなければなりません。あるリストされたタスクは、タスクからタスクへの移動を伴う順序を持った手順で構成されます。一方では、あるタスクはジョブ全体のサポート機能として独立して働くこともあるでしょう。最後ステップは JobGen Plus ですべてのタスクと、タスク間の移動を実装することです。

タスクの実装とタスク間の移動は簡単です。ユーザは JobGen Plus のノードに合った各タスクを単に実装するだけです。例えば、データ収集のタスクは収集ノードによって実装することができます。 以下の表は JobGen Plus で相当するノードによって実装することのできるタスクのタイプを示しています。

ノード	実装することのできるタスク
コメント	• 注記と説明を表示
メニュー	• ユーザに対する選択を表示
	• エラーメッセージを表示
	• 命令・指示を表示
収集	• キーボードからデータを入力
	• バーコードスキャナからデータを入力
	• データコレクタのタイマーからデータを入力
	• ルックアップファイルからデータを入力
	• 式からデータを入力
	• データ入力の確認
演算	• 式からデータを入力
	• データの計算
	• データの比較
消去	• 収集したデータ内部のレコードを削除
アップロード	• データコレクタからホストコンピュータへデータを送信

編集	• ポータブルターミナルのデータファイルに保存されたデータを表示 • データを検索 • データの変更
プログラム	• 上記にリストされたノードによって実装することのできないタスクの種類
ジョブ実行	• 他のジョブを実行

あるタスクから別のタスクへの移動は JobGen Plus ではリンクによって行うことができます。

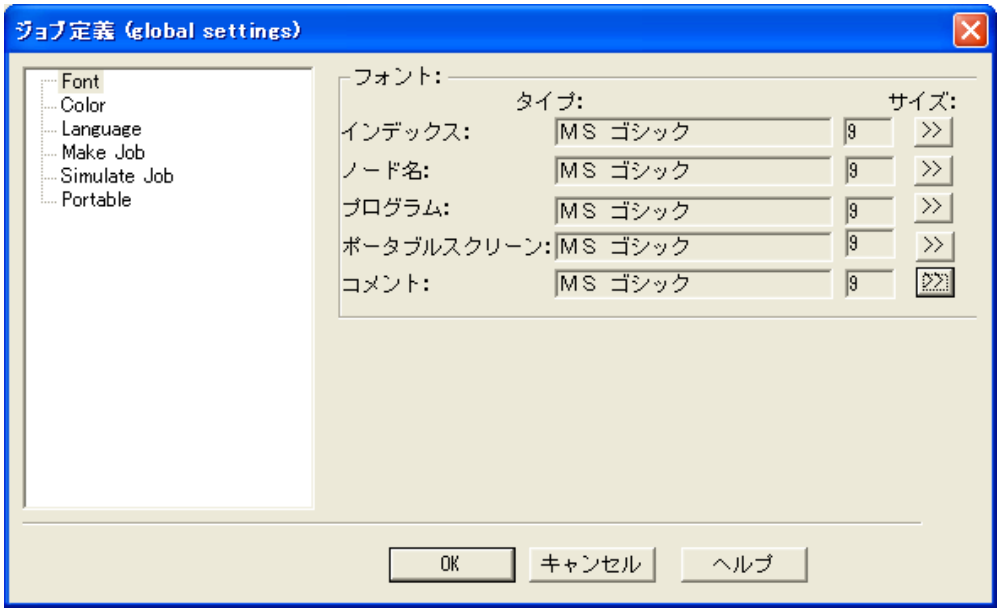
第 5 章 ジョブのプログラミング

ジョブ[1]の構築はマウスボタンをクリックするように簡単です。JobGen Plus は MS Windows の利点を利用しており、したがってほとんどのユーザは JobGen Plus によって提供されているツールの使用を視覚的に行うことができます。この章はジョブの作成に必要なすべてのプロセスを詳しく説明しています。

ジョブ設定の標準値を定義する

ジョブ設定の標準値定義のダイアローグ・ウインドウを開くために、メニューの**編集 > 書部定義(標準値)**を選択して下さい。これらの設定はすべての作業中のジョブに影響します。

フォントプロパティ



左のプロパティウインドウ、ノード名、プログラムエディタ、そしてコメントを表示するフォントを定義します。

[1] ジョブはデータ収集アプリケーションです。

カラープロパティ

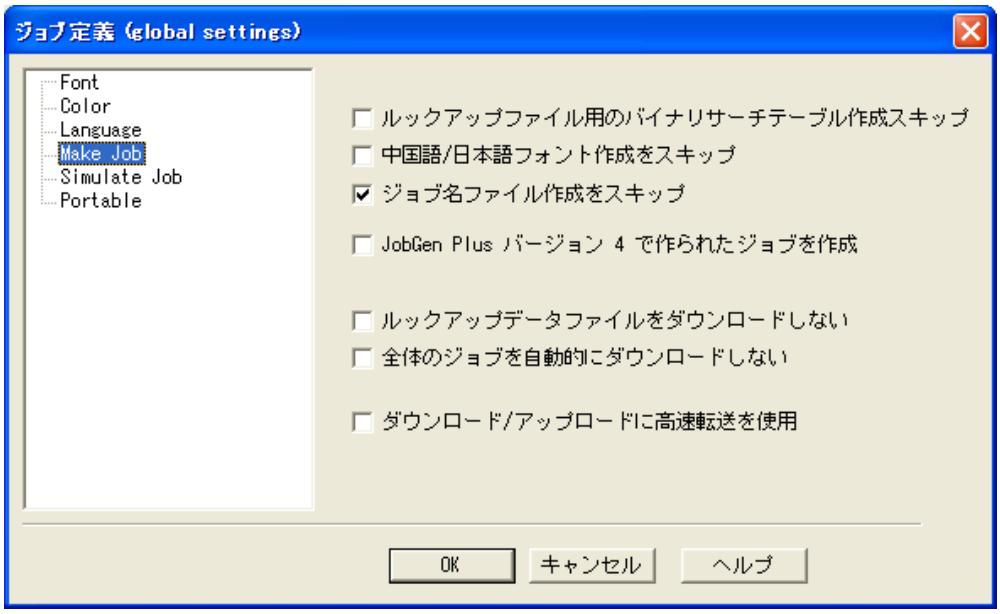
ノード、リンク、そして選択についてのカラーを定義します。表示されているのはノードについてのカラー設定の標準値です。それぞれのノードの色はその基本プロパティ設定で変更することができます。

Language(言語)プロパティ

ジョブの言語を選択します。

日本語フォントを選択します。

ジョブ作成プロパティ



ルックアップファイル用のバイナリサーチテーブル作成スキップ

ルックアップファイルの内容が変わらない場合、ジョブを作成するたびにコンパイルする必要はありません。これはジョブ作成のプロセスをスピードアップします。

中国語/日本語フォント作成をスキップ

ルックアップファイルとデータファイルに含まれるすべてのテキストが変わらない場合、そのたびごとに漢字フォントを作る必要はありません。

ジョブ名ファイル作成をスキップ

ジョブファイル名がどのジョブを実行するかを決めるためにジョブエンジンに対して使用されます。ジョブの実行可能な名前を選択することによってジョブを実行する場合、ジョブ名ファイルは不要になります。

JobGen Plus バージョン 4 で作られたジョブを作成

古い JobGen Plus のバージョンでは動いていた古いジョブで問題が起こった場合、このオプションをチェックして、ジョブを再度作成して下さい。これは以前のバージョンとの互換性を持たせることができます。

ルックアップデータファイルをダウンロードしない

すべてのルックアップファイルの内容が変更されない場合、そのたびごとにダウンロードする必要はありません。

全体のジョブを自動的にダウンロードしない

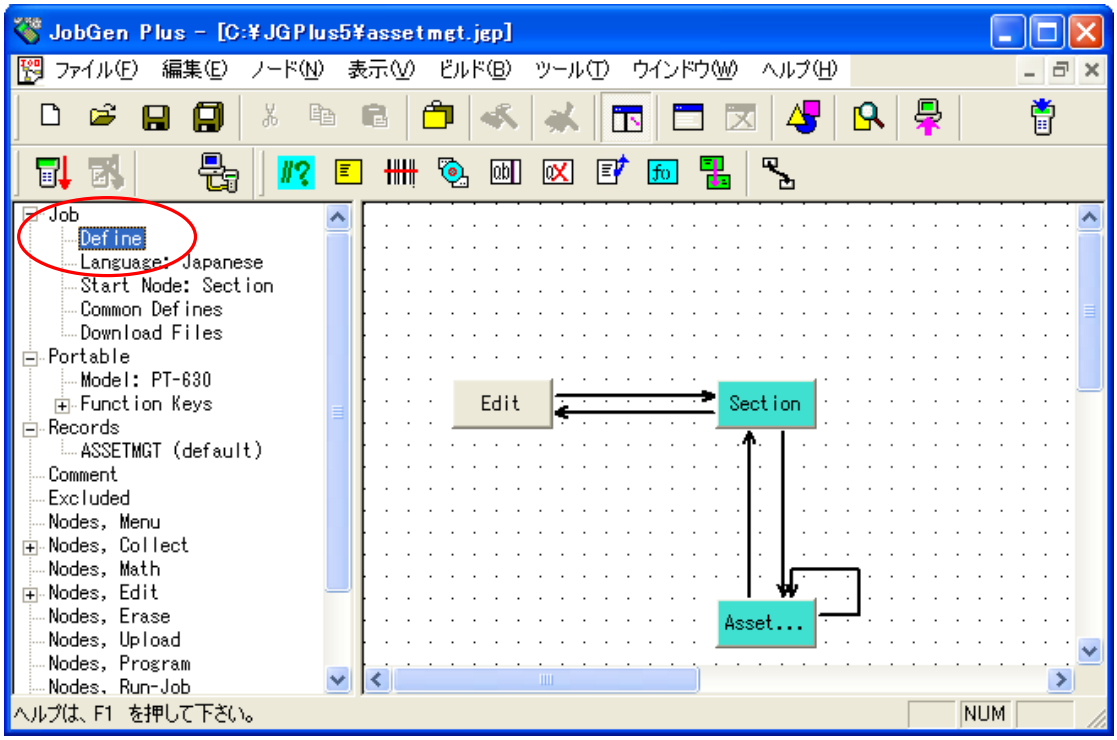
JobGen Plus は、ジョブ実行コードを正しく生成できたときにポータブルターミナルに実行可能ジョブをダウンロードすることができます。ダウンロードを停止するためにはこのオプションをチェックして下さい。JobGen Plus は後でダウンロードするために Job エンジン（JENG.exe）をジョブ・フォルダにコピーします。

ダウンロード/アップロードに高速転送を使用

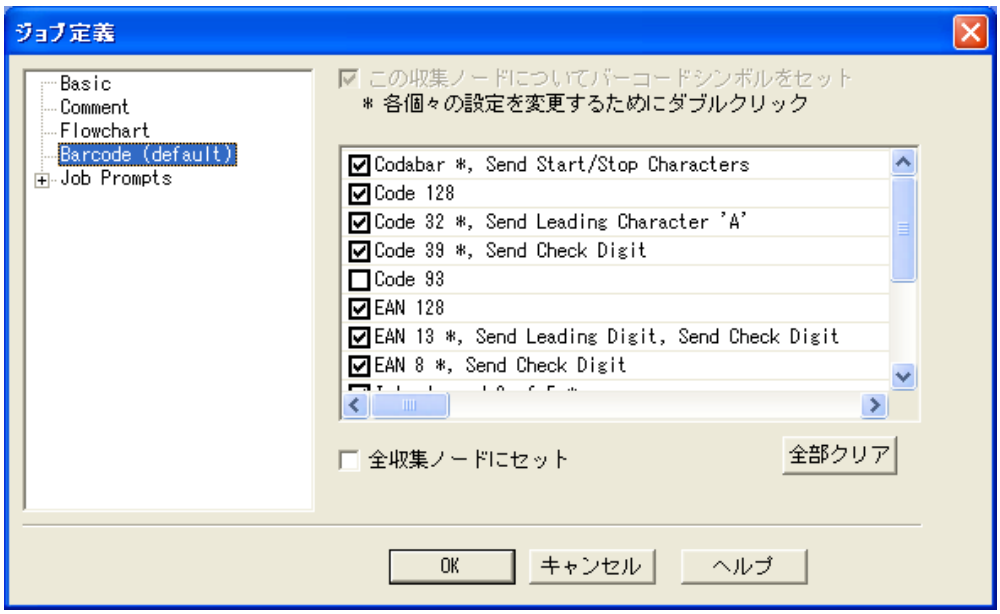
JobGen Plus は PTComm マネージャが最も速い速度で通信するようにします。

ジョブ設定の標準値を定義

ジョブ定義設定ウインドウを開くために、ジョブの左側のプロパティウインドウで Job > Define を選択します。

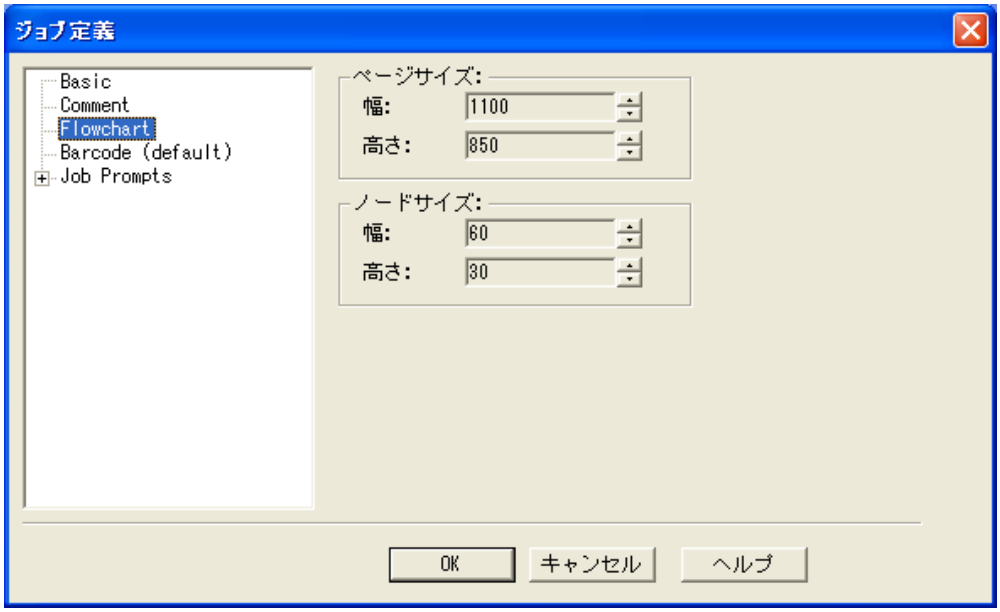


Barcode (default) プロパティ



すべての収集ノードについて標準のバーコードシンボル設定を定義します。それぞれの収集ノードについての設定は、Input > Barcode プロパティで変更することができます。

Flowchart(フローチャート)プロパティ



フローチャートのページサイズとすべてのノードサイズを変更します。

Language（言語）プロパティ

このジョブの言語設定を変更します。

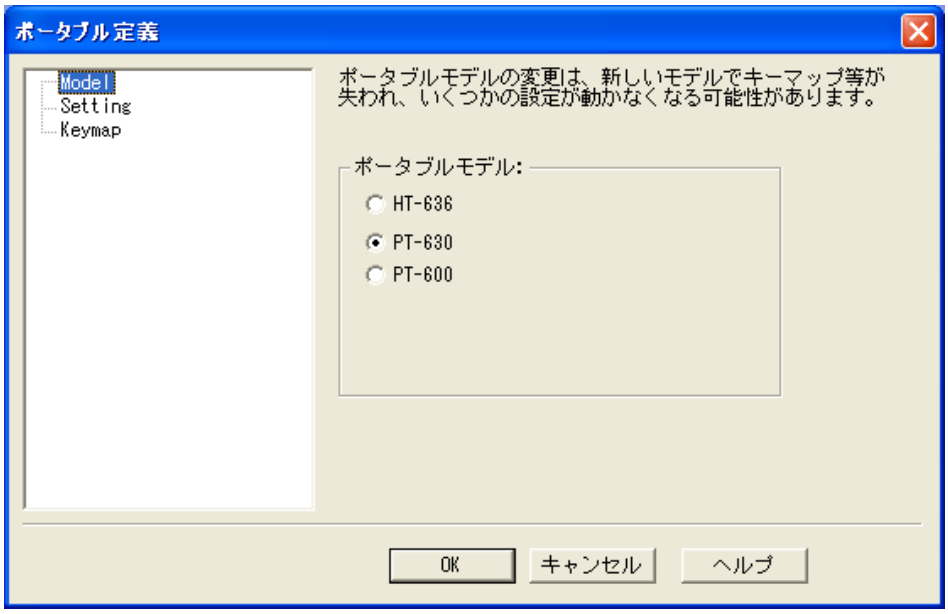
ポータブルターミナルの定義

デバイスの選択は、メニューノード、収集ノード、演算ノード そしてリンクの表示に影響があるので JobGen Plus では最も重要です。メニューノード、収集ノードと演算ノードの“Screen”(スクリーン)部分のサイズは選択したデバイスの表示と完全に同じサイズにセットする必要があります。また、リンクのキー入力は、選択したデバイスと同じキーパッドレイアウトを表示します。JobGen Plus は選択したジョブを適当なポータブルターミナルにダウンロードします。

ポータブルターミナルのモデルを選択するには、ジョブのプロパティパネルの、プロパティの項: Portable > Model を開きます。ウインドウをポップアップするために Model の項目をダブルクリックします。 ポータブルターミナルのタイプを定義するために Model プロパティを再度選択します。

ポータブル定義プロパティ

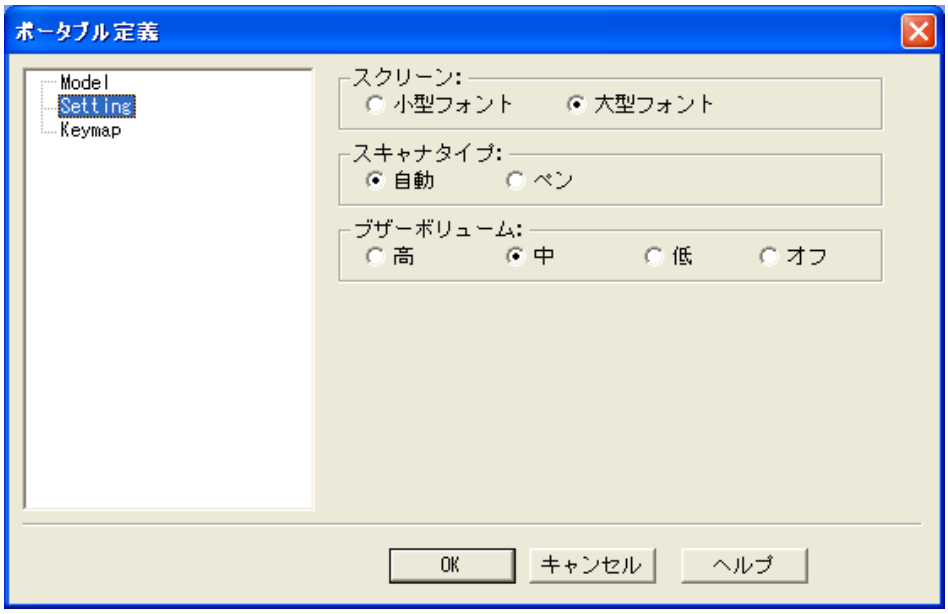
モデルの定義:



ポータブルモデル　例えば、PT630 のようにターゲットとするポータブルターミナルのモデルを選択します。PT-630 はポータブルターミナルの標準のモデルです。

ポータブルモデル設定の変更はジョブのあるプロパティ設定に影響します。例えば、PT-630 は PT600 よりも大きなスクリーンなので、PT-630 から PT600 へポータブルモデルの変更を行うと、スクリーンのレイアウトはマニュアルで変更する必要があるか、あるいは小さなスクリーンに合うように再設計する必要があります。

ポータブルターミナル設定(Setting)

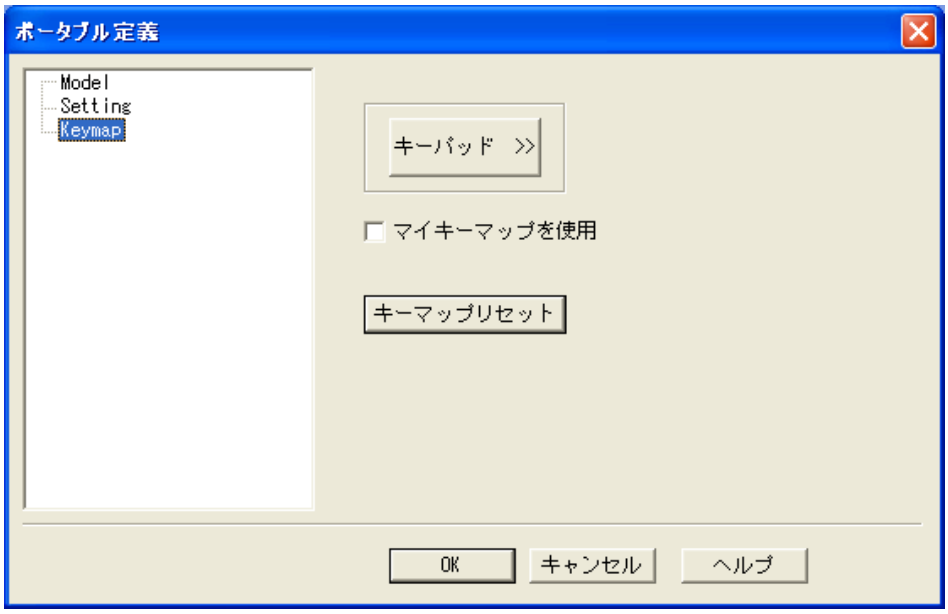


スクリーン　ディスプレィのフォントサイズを定義。日本語を表示させる場合は大型フォントを選択して下さい。

スキャナタイプ　スキャナタイプの定義。標準は自動です。

ブザーボリューム　ブザーの音量を定義。

キーパッド設定の定義(Keymap):

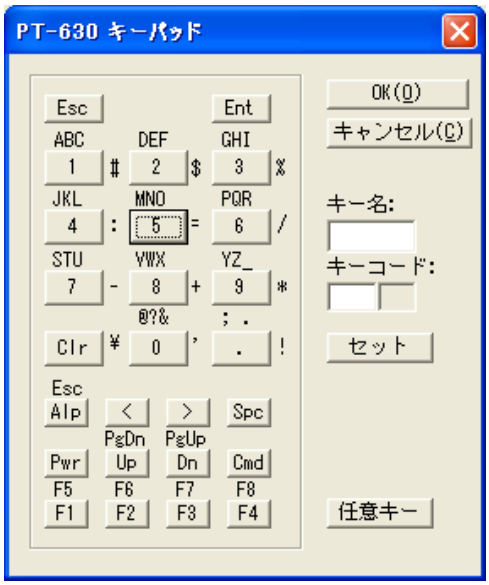


キーマップリセット　　ポータブルターミナルの標準設定を使用。

キーパッド　　キーマップの設定を定義するためにウインドウをポップアップ。

マイキーマップを使用　　キーパッド設定を適用することをジョブエンジンに伝える。

標準値からキーマップを変更する方法:



キーマップを変更したいキーボタンをクリックします。キーの名前が右側のキーボックスに表示され、そして ASCII コードが印刷可能な文字でキー名の下の、二つのキーコードボックスに表示されます。

キーマップを変更するために、キーの ASCII コードの値を変更し、そして“セット”をクリックします。ジョブの実行時にカスタマイズしたキーマップ設定を適用するためには“マイキーマップを使用”をチェックしなければなりません。

メニューノード)の作成

メニューノード)は、メッセージの表示方法を提供します。メニューノード を作成した後で、ユーザはターゲットデバイスのディスプレィに表示するメッセージをタイプインすることができます

メニューノードの作成方法: ノードツールバーのメニューアイコンをクリックすることによって、メニューノードとなるノードタイプを選択するか、あるいはノードメニューでメニューノードを選択します。そして、フローチャートに新しいノードを置きたい場所を指すためにカーソルを使用して、ダブルクリックして下さい。新しいメニューノードが現れます。

あるいは、より簡単な方法では: フローチャートに新しいノードを置きたい場所を指すためにカーソルを使用して、コンテキストメニューを表示するために右クリックし、そしてメニューノードを選択します。新しいメニューノードが作成されます。

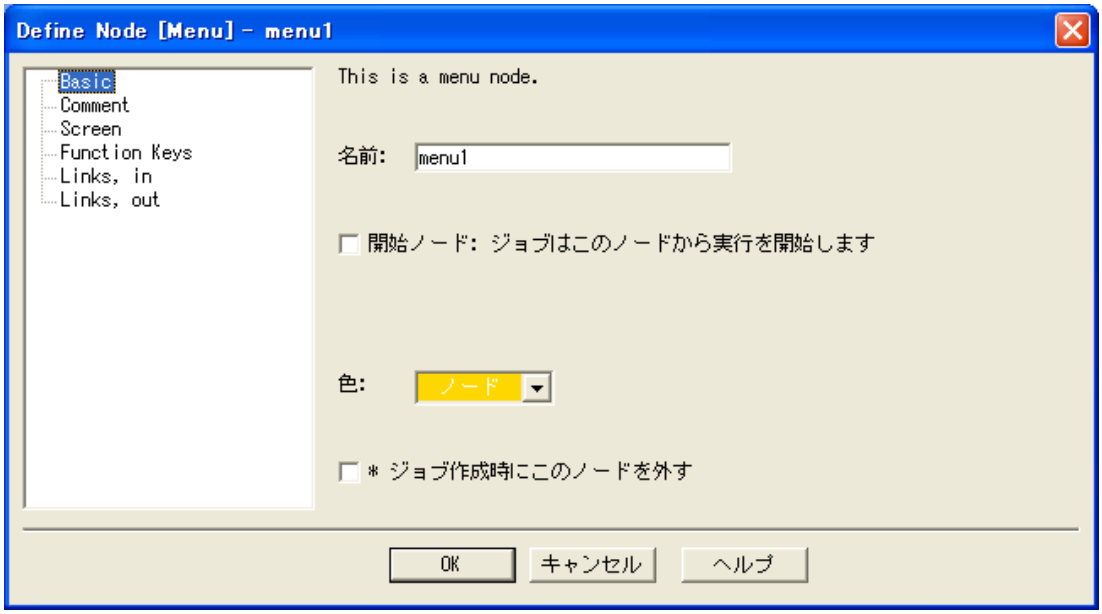
他のノードタイプの作成には、操作は同じです – 別なノードタイプを選択するだけです。

そのプロパティを定義するためにメニューノードをダブルクリックして下さい。

メニューノードの定義

各ノードは独自のプロパティグループを持っており – そして、basic、 comment、 function keys、links in、 そして links out プロパティなどの共通なプロパティがあります。

Basic プロパティ



名前

ノードの名前。アルファベット、数字と下線文字のみを名前に使用できます。

収集または演算ノードの場合、これらは実際にデータを持っているので、ノード名にプリフィックス“_”（下線）を追加し、データにアクセスするためにＣプログラムによって参照できるようにします。

開始ノード

このボックスをチェックすることによって、このノードは開始ノードになります（JobGen Plus アプリケーションの最初のポイント）。

各 JobGen Plus アプリケーションは一つだけの開始ノードを持つことができます。

色

ノードの色を定義します。

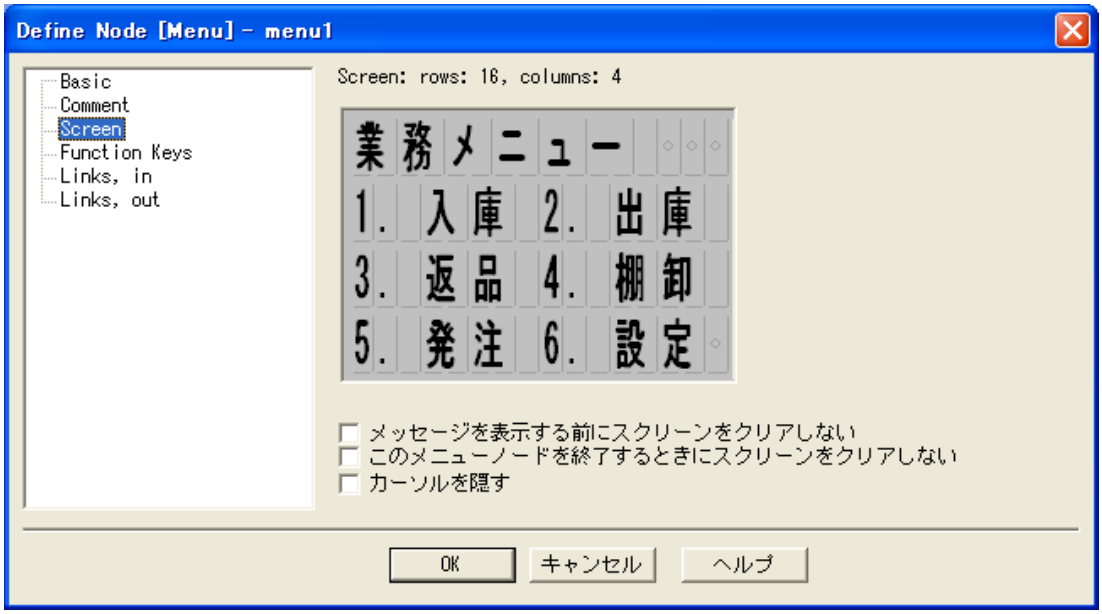
＊ ジョブ作成時にこのノードを外す

このノードはまだ作っている最中であることを JobGen Plus に知らせます。その設定はまだ終わっていないので、JobGen Plus はこのノードを実行コードに生成しません。

Comment(コメント)プロパティ

コメントボックス　　コメントを書きます。

Screen(スクリーン)プロパティ



画面表示　　実際の画面で見える通りに表示するウインドウです。これはポータブルターミナルの表示と同じように表示します。

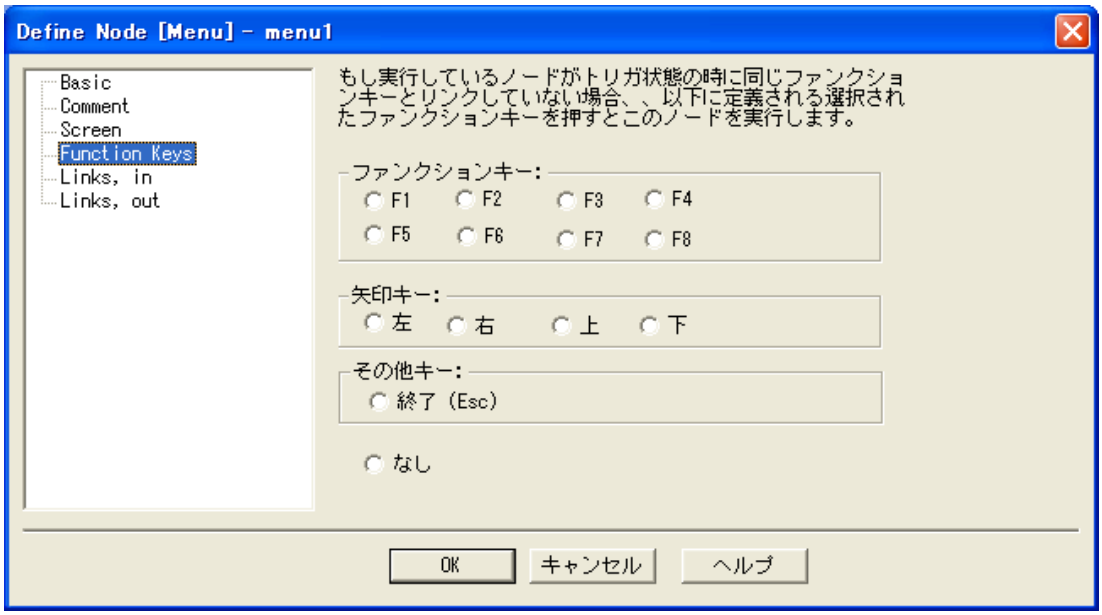
表示したいテキストをウインドウに直接タイプします。

ディスプレィの大きさは、ポータブル定義で選択したポータブルターミナルによって決まります。

メッセージを表示する前にスクリーンをクリアしない　　前のスクリーンからのメッセージを保持します。

このメニューノードを終了するときにスクリーンをクリアしない　　次のスクリーンにメッセージを保持します。

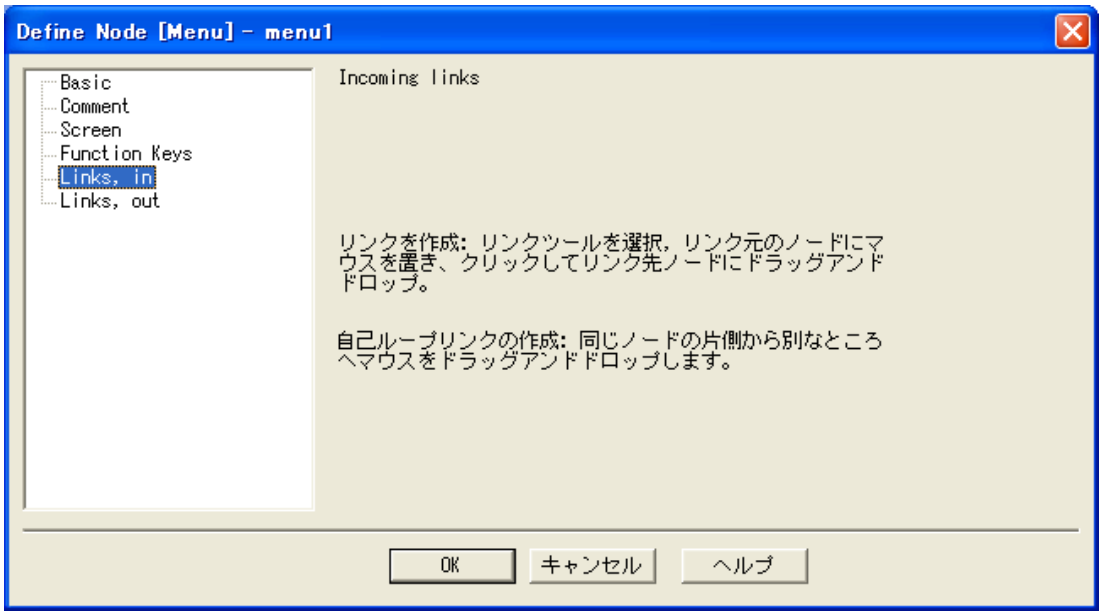
Function Keys(ファンクションキー)プロパティ



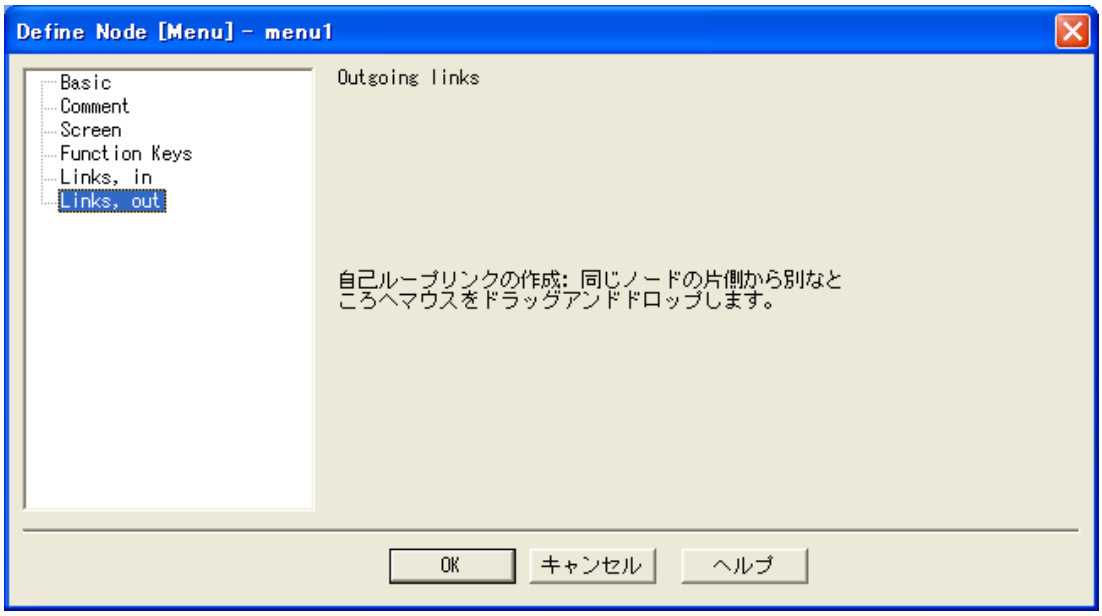
ジョブを実行中に、このノードで定義したファンクションキーを押すことによりこのノードの実行フローを指示します。しかし、現在のノードがこのファンクションキーのキー入力状態にリンクされている場合、このリンクはその相手先ノードを実行するために高い優先度を持ちます。

特定ノードへファンクションキーを定義することは大変簡単なフローチャートの設計です。 一方、一つのノードへ同じファンクションキーの入力状態を指示している多数のリンクがあり、これらのリンクは相手先ノードのファンクションキー定義によって置き換えることができます。

Links In(リンクイン)プロパティ



Links Out(リンクアウト)プロパティ



Links In/Out(リンクイン/リンクアウト)プロパティは、このノードに入る、または出るすべてのリンクをリストします。Define Link(リンク定義)ダイアローグを開くためにリンク上をクリックして下さい。

収集ノードの作成

収集ノードはユーザが情報の入力するための最初の場所です。ユーザからの入力を必要とする JobGen Plus アプリケーションの任意のポイントは、収集ノードによって表されます。

収集ノードの定義

Basic プロパティ

Comment プロパティ

Screen プロパティ

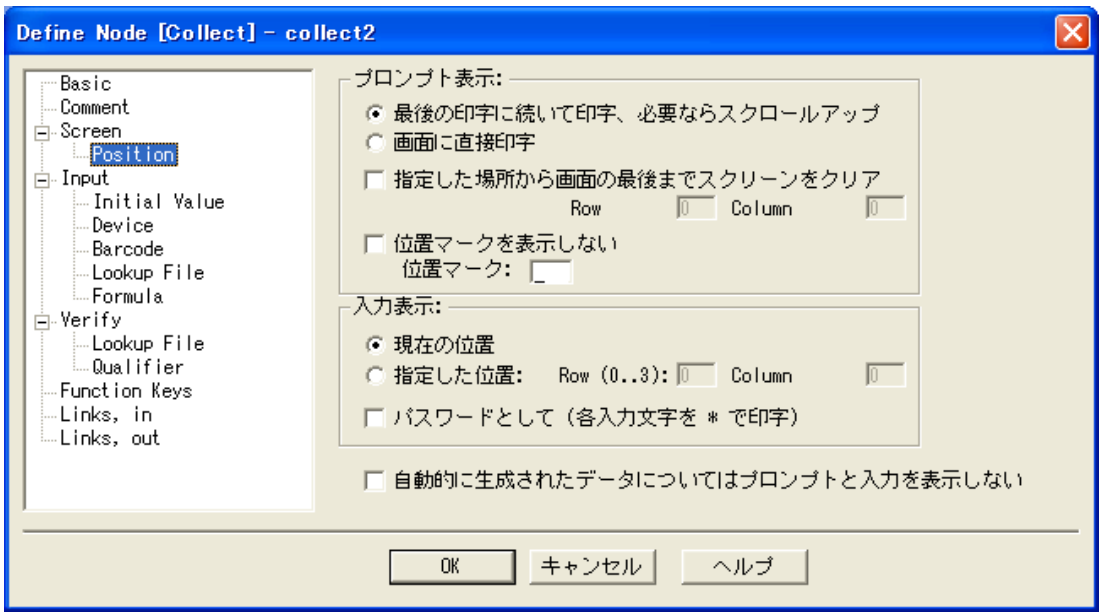
Function Keys プロパティ

Links In プロパティ

Links Out プロパティ

これらのプロパティはメニューノードと同じです。

Screen > Position プロパティ



プロンプト表示

最後の印字に続いて表示、必要ならスクロールアップ　現在のカーソル位置で行ごとに表示し、そしてスクリーンをスクロールアップします。

画面に直接印字
スクリーンに表示されたテキストをポータブルターミナルのスクリーンに表示します。すべてのテキストはその指定された位置になければなりません。

指定した場所から画面の最後までスクリーンをクリア
表示する前にスクリーンの一部または全部をクリアします。

row と column ボックスで行と列を指定します。

位置マークを表示しない
新しい入力のスタート時に、前の入力がブランクになります。現在の入力は以降の入力が始まるまでスクリーン上に残ります。

位置マーク
データ入力のためにブランクのスペースを埋めるために使用される英文字。ユーザがデータを入力したら、これらの文字は入力データに置き換わります。

入力表示

現在の位置
現在のカーソル位置に表示します。

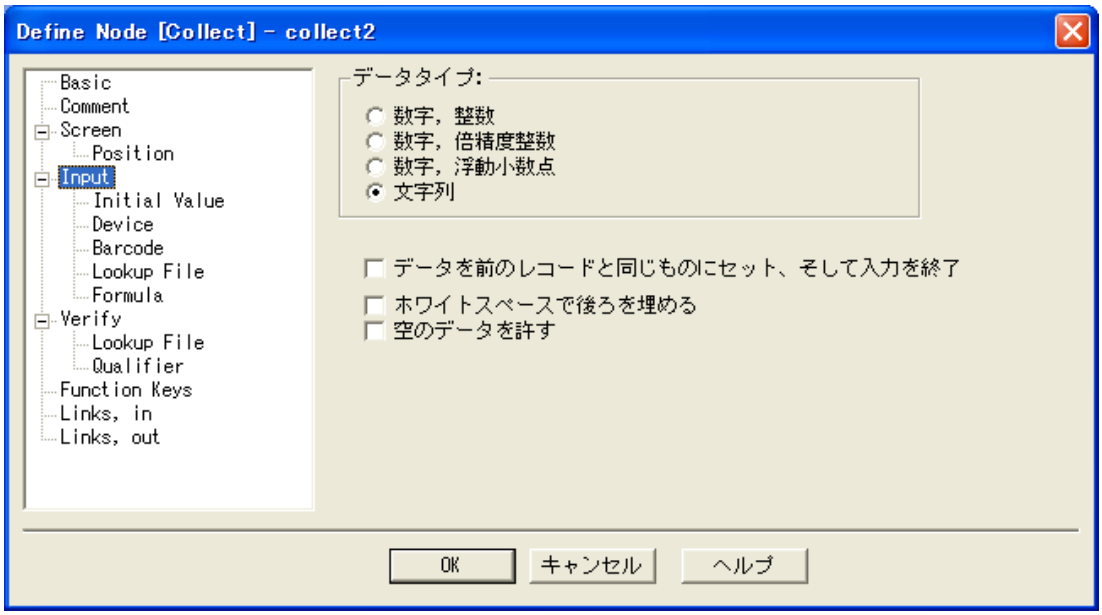
指定した位置
Row(行)と column(列)で指定された位置に表示します。

row と column ボックスで行と列を指定します。

パスワードとして
* を表示することによって入力を隠します。

自動的に生成されたデータについてはプロンプトと入力を表示しない
入力はユーザとの会話なしに入れられます。

Input(入力)プロパティ



データタイプ

すべてのデータは、数値または文字列の二つの基本データタイプです。数値データは整数、 倍精度整数と浮動小数点の三つのタイプに分けられます。

数値データは数値演算を実行するために使用されます。 三つの数値タイプは整数の範囲と小数点以下の数があるかどうかが異なっています。

倍精度整数を表すために、数の最後に‘L’ を追加します。例えば: 100000L。 (100000 は整数の制限値を越えています。)

浮動小数点を表すために、小数点の右に常に入力があります。例えば: 10.0。

文字列はテキストです。

データを前のレコードと同じものにセット、そして入力を終了

データは最後のレコードからのデータで作られ(コピーされ)、そして入力はすぐに自動的に完了します。言い換えると、新しい入力は前のレコードと完全に同じです。

ホワイトスペースで後ろを埋める

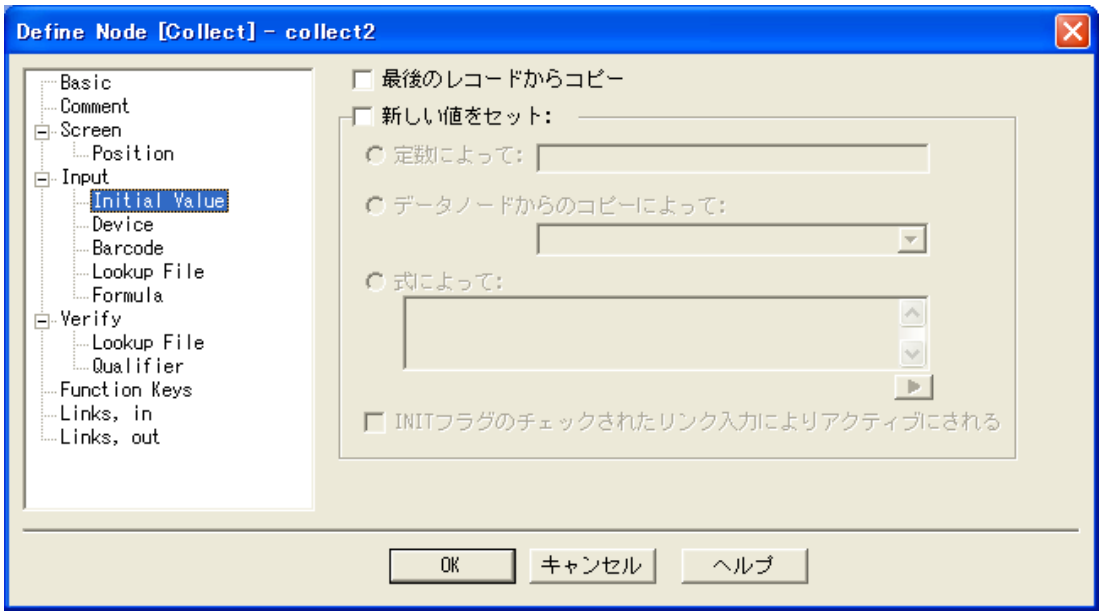
空白文字(ASCII コード、0x20)でデータの後部を自動的に埋めます。これは固定長データを生成するためのものです。

空のデータを許す　データ入力のスキップを可能にします。レコード中のこの特別なフィールドの値はヌルで、何もデータはありません。標準では、すべてのデータは 1 またはそれ以上の文字を含まなければなりません。

数の範囲

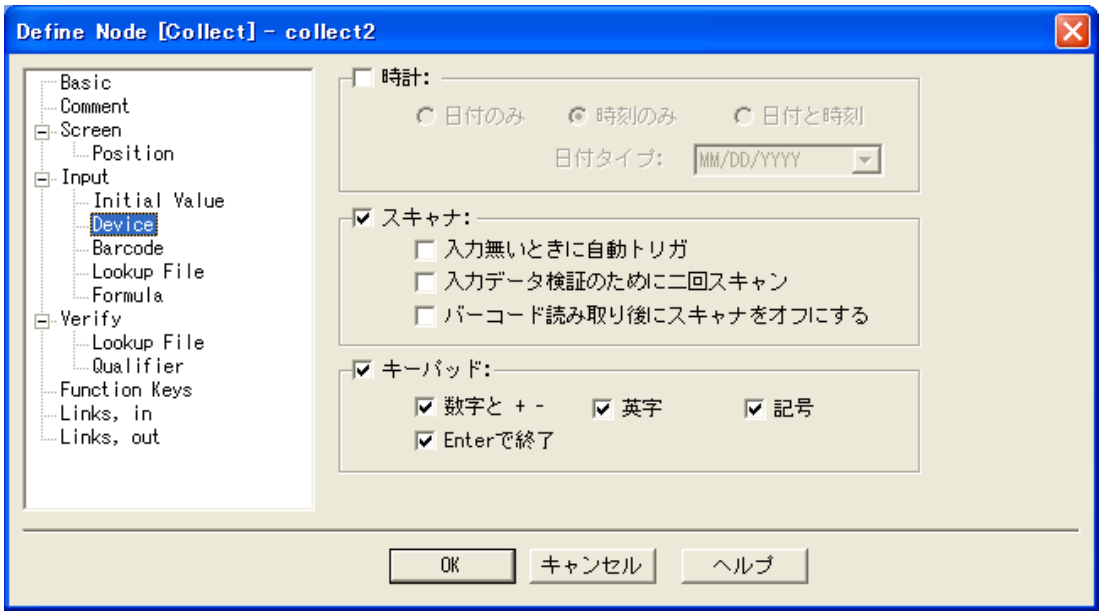
整数:	-32767	から 32767
倍精度整数:	-2147483647	から 2147483647
浮動小数点:	1.175494351e-38F	から 3.402823466e+38F

Input > Initial Value プロパティ



最後のレコードからコピー	データの値は最後のレコードからのデータで(コピーで)埋められます。データはこの後変更することができます。このオプションは最後の結果からデータを受け継ぎたい場合に使用されます。
新しい値をセット	新しい値をセットします。
定数によって	データに定数を指定します。
データノードからのコピーによって	他のデータノード(収集ノードまたは演算ノード)からデータをコピーします。数値タイプの収集ノードだけがデータソースとして選択することができることに注意して下さい。
式によって	値を指定するのに式を使用。
INIT フラグのチェックされたリンク入力によりアクティブにされる	**新しい値をセット**の初期値は **INIT フラグのチェックされたリンク入力**のチェックによってオンにされた場合にのみ処理されます。

Input > Device プロパティ



ポータブルターミナル装置へのデータ入力方法が三つあります。時計、 スキャナそしてキーパッドです。複数の選択が可能な場合、実際の入力は一度に一つの装置からだけ入ります。

- **時計**

自動的に時計からデータを得て、入力を埋めます。三つのデータフォーマットがあります。これらは 日付のみ(MM/DD/YYYY)、時刻のみ(HH:MM:SS) と日付と時刻(MM/DD/YYYY HH:MM:SS)です。

- **スキャナ**

スキャナの入力を可能にします。バーコードシンボルは Barcode プロパティページ でセットすることができます。個々の収集ノードはバーコードシンボルについて独自の設定を持つことができます。ジョブ中のすべての収集ノードについての標準のバーコードシンボル設定はジョブ定義、Barcode (default) プロパティページで定義されます。

- **バーコード読み込み後スキャナをオフにする**

バーコードを正しく読んだ後でスキャナをオフにするためにこのオプションをチェックします。いつオンにするかを気にする必要はありません – 収集ノードはこれが選択した装置の一つである場合にスキャナを自動的にオンにします。

通常は、スキャナは収集ノードでバーコードを読んだ後オン状態のままですので、リリースしてスキャナのトリガを再度引くことなしには以降の収集ノードでデータのスキャニングをしません。各スキャンはトリガを引くことによってオンにしなければなりません。しかし、オン状態のままのスキャナはデータ入力のない時間に間違ったスキャンをすることになります。ジョブエンジンは現在の実行ノードが収集ノードあるいは演算ノード、またはプログラムノードでなければ、スキャナを自動的にオフにします。

- **キーパッド**

入力のためにキーパッドを使用可能にします。

- **数字と + - .**

数字キー: 0 – 9、小数点:. 、そして正負符号 + と – のみがキー入力として受け入れられます。

- **英字**

大文字と小文字の英字キー: a – z 、 A – Z、とスペースがキー入力として受け入れられます。

- **記号**

$、 #、 !、 等の記号だけがキー入力として受け入れられます。

- **Enter で終了**

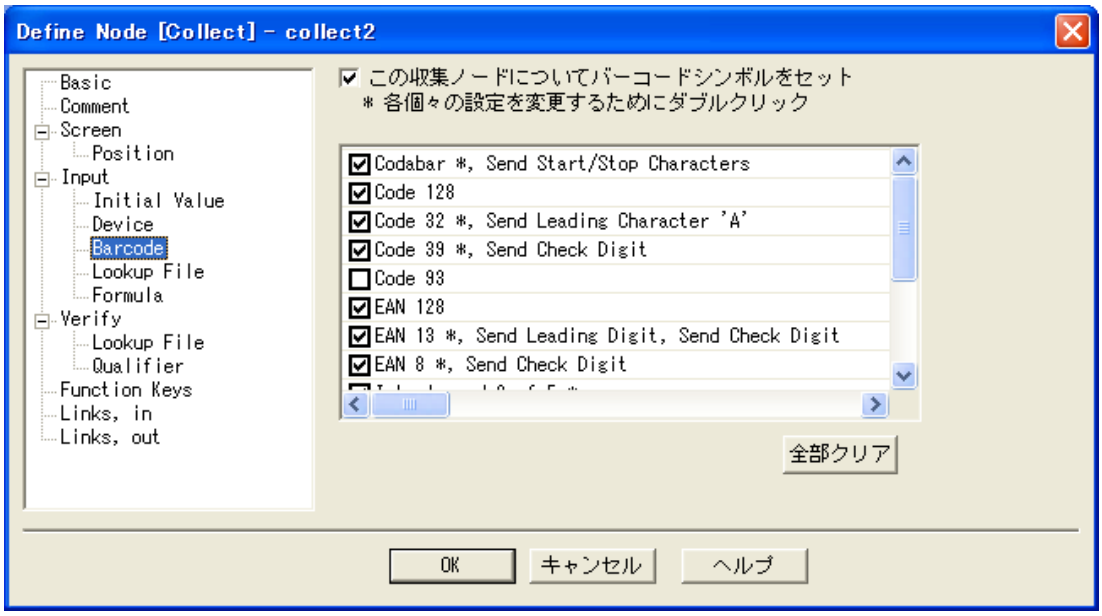
入力を終了するためには、Enter キーを押さなければなりません。これをチェックしない場合、入力が最大長に達したときに自動的に終了します。

キーは三つの種類: 通常キー (0 から 9 数字キー、そして A から Z 英字キー)、ファンクションキー (F1 から F8、そして矢印キー) および特殊キー(CMD、スキャンと電源キー)に分けられます。

ファンクションキーはデータ入力の操作で特別な機能を持っています。入力操作は、ノード、あるいは現在のノードに入る、あるいは出るリンクに対して定義されたファンクションキーが押された場合キャンセルするようにすることができます。通常は、ファンクションキーは特別なノードによる操作手順の変更に使用されます。

通常キーとファンクションキーはキーマップ でコードを再定義することによって他のキーに割り当て直すことができます。

Input > Barcode プロパティ

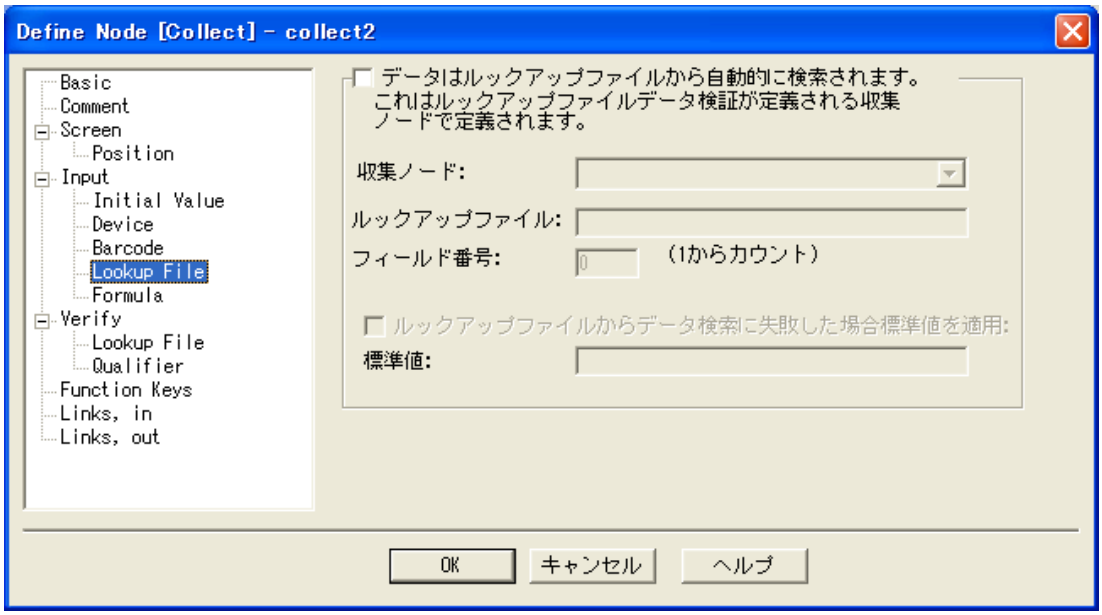


選択された場合、それぞれの収集ノードは、それ自身のバーコードシンボル設定を持っているので、別な収集ノードは別なスキャナの動作をします。

アスタリスク（*）が付く名前のバーコードは詳細な設定が必要です。設定ウインドウを開くために名前をダブルクリックして下さい。チェックされたサブ設定は名前の次に表示されます。

標準のバーコードシンボル設定は Job Define、 Barcode（default）ページで行われます。

Input > Lookup File プロパティ

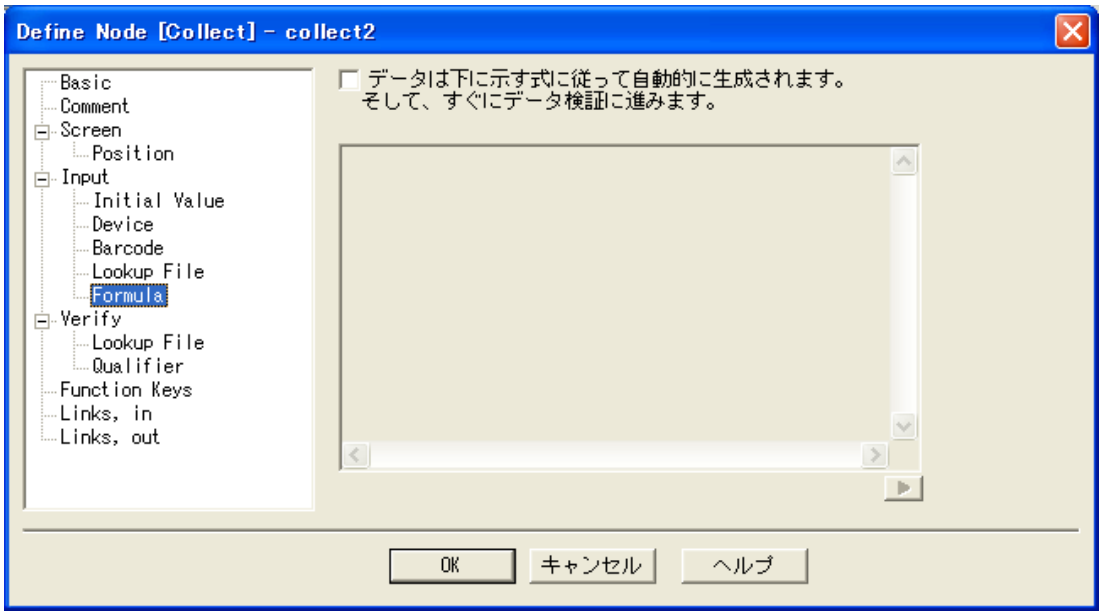


多くの場合、ルックアップ検証における以前の成功を元にしたルックアップファイルからデータを検索する必要があり、関連するデータをベースにしてデータを処理することができます。

ルックアップファイルを定義するためにルックアップ検証を実行するすべての収集ノードから名前を選択します。成功したルックアップ検証を必要とするのでルックアップファイルを直接指定することはできないことに注意して下さい。そして、フィールド番号を指定します。レコード番号は、ノードのルックアップ検証から一致したレコード番号によってジョブ実行時に決定されます。

収集ノードがルックアップ検証に失敗した場合、ルックアップファイルからの入力は終わらず、そして入力方法がキーボード入力に変更されます。

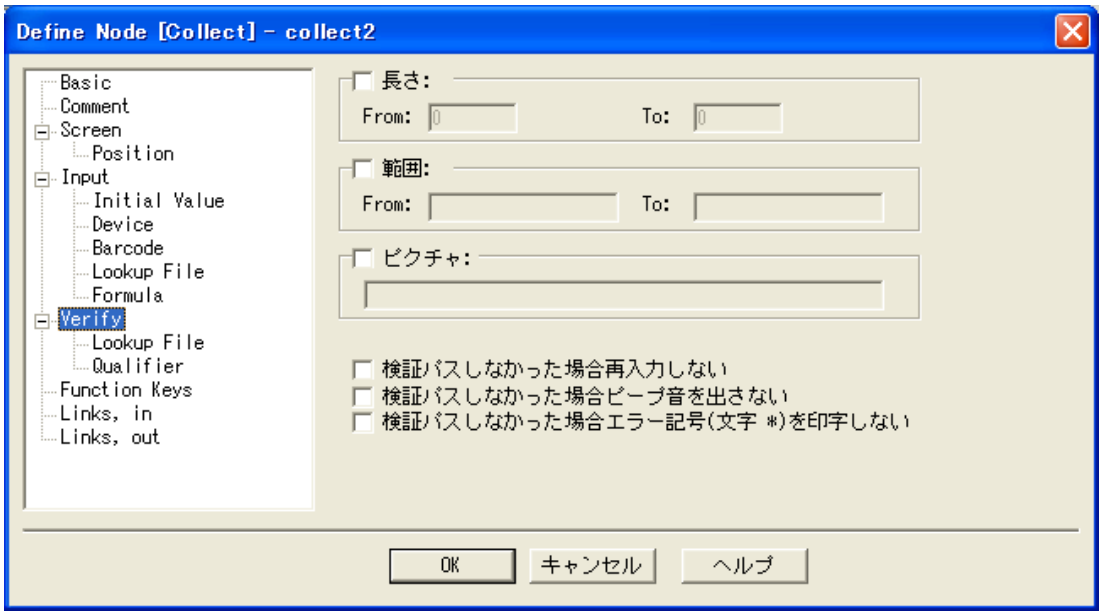
Input > Formula(式) プロパティ



データを生成するための式として C 言語で記述して下さい。

収集ノードは実際に他によってアクセスすることのできるデータの変数を持っています。アクセスするには、下線('_')のプリフィックスを持つノード名であるその変数名を使用します。

Verify (検証)プロパティ



長さ	入力データの長さ(文字数)を指定します。
範囲	入力データの値の範囲を指定します。これはその文字の ASCII コードと比較されます。
ピクチャ	入力文字タイプのマスクパターンを定義します。
検証パスしなかった場合再入力しない	普通は、基本的な検証、長さ、範囲とピクチャは入力データの基本的なチェックとして扱われます。入力データは、プロセスを進める前に、これらの三つの状態に合わなければなりません。そうでなければ、ユーザはデータが正しく渡されるまで再入力させられます。
検証パスしなかった場合ビープ音を出さない	検証が失敗した場合、ユーザの警告のためにブザーが鳴ります。検証が失敗した場合にブザーを切るためにこのオプションをチェックして下さい。
検証パスしなかった場合、エラー記号(文字*)を印字しない	検証に失敗した場合、間違ったデータ入力のエラー記号はスクリーンに表示されません。

ピクチャパターン

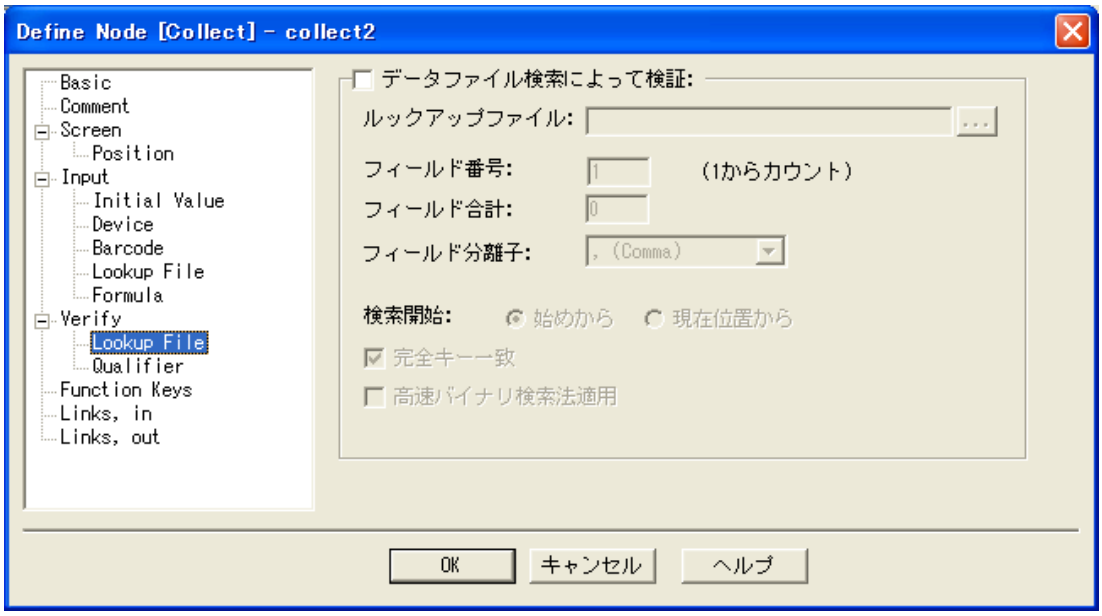
a	英文字 (A-Z, a-z) のみ可
n	数字(0-9) と ‘+’, ’-‘, ‘.’ (ドット), ‘E’ と ‘e’のみ
0	数度 (0-9) と ‘+’, ’-‘ のみ
p	数字 (0-9)のみ
l	小文字 (a-z)のみ
u	大文字 (A-Z)のみ
x	印刷可能な文字 (0x32--0x7f)
z	全 ASCII コード
_	全 ASCII コード
#	文字を削除
@	@ と @間の文字列
%	インジケータ、パターンにシンボルを追加
?	インジケータ、非固定長のパターン
!	インジケータ、検証結果を逆に
(space)	パターンのターミネータ

ピクチャ検証の例:

ピクチャ	入力データ	結果
ppp-pppp	1112222	1112222
ppp-pppp	111-2222	1112222
ppp-pppp	111.2222	***エラー***
ppp-pppp	11-22222	***エラー***
aaa-aaaa	1112222	***エラー***
#pp-pppp	1112222	112222
#pp-pppp	111-2222	112222
aa#aaaa	PC-WAND	PCWAND
###-zzzz	1112222	2222
###-zzzz	111-2222	2222

pp@bcd@pp	11bcd22	11bcd22
pp@bcd@pp	11abc22	***エラー***
$ppp-pp	11133	11133
$ppp-pp	$11133	11133
$ppp-pp	111-33	11133
$ppp-pp	$111-33	11133
%ppp-pppp	1112222	111-2222
%ppp-pppp	111-2222	111-2222
%”ppp-pppp”	1112222	“111-2222”
%$ppp.pp	11133	$111.33
%$ppp.##	11133	$111.
?p	1	1
?p	111	111
?ppp	11	***エラー***
?aaap	abc123	abc123
?aaap	abc	***エラー***
?%$ppp	123	$123
?%$ppp	12345	$12345
?%$ppp	12	***エラー***

Verify > Lookup File (ルックアップファイル)プロパティ



これはデータファイル(データベース)の入力データをチェックするのに使用される便利な機能です。

フィールド番号は 1 からカウントが始まります。これがパスしたとき、リンク状態に使用される成功フラグが上がります。そうでなければ、異常フラグが上がります。

JobGen Plus はジョブファイルとルックアップファイルを自動的にダウンロードします。これが大きなファイルで、ジョブ開発中に変更されない場合は、毎回ダウンロードする必要はありません。 ジョブ定義(標準設定) 、ジョブ作成ページで、ルックアップファイルの自動ダウンロードを禁止して下さい。

- **検索開始**

最初から、もしくは現在位置から、のいずれかからデータファイルを検索します。

- **完全キー一致**

完全なデータの一致を指定します。例えば:

“abc” は “abc”に完全に一致します;

“abc” は “abcdefg”と完全には一致しません;

“abc” は “1abc23”には一致しません。

- **高速バイナリ検索法適用**

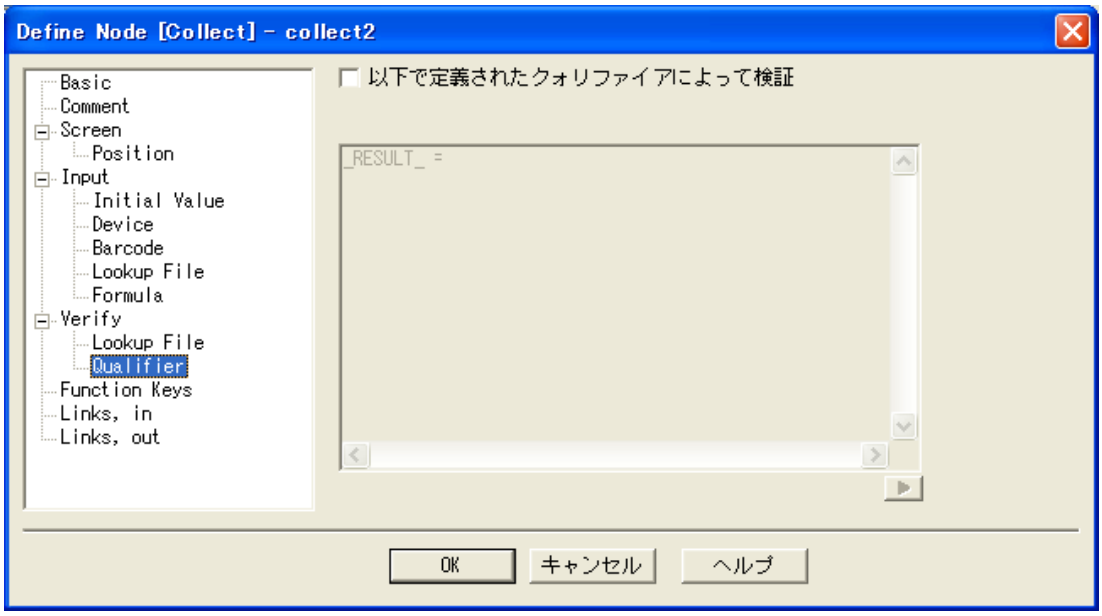
10 万件のデータを含むような大きなデータベースを検索する場合、高速検索を可能にするためにこのオプションをチェックして下さい。JobGen Plus は自動的にファイル – バイナリ検索テーブル – を追加し、そして元のデータファイルとジョブファイルと共にダウンロードします。

バイナリ検索の欠点は、データファイルを実行時に変更できないことです。データファイルの変更はデータファィル中ですべてのレコードの正確な位置を記録しているレコードであるバイナリ検索テーブルを壊してしまいます。バイナリ検索テーブルの間違った情報で、検索は予期しない結果が戻ります。

しかし、ある規則を加えることによって、バイナリ検索中にデータファイル中のあるデータの変更が可能になります。この規則は以下の通りです:

- キーフィールドは変更することができない。
- レコードの長さは一定。各レコードは同じ長さである必要はありません。レコードを変更した後で、実際のレコード長が変更していないことを保証します。

Verify > Qualifier (クォリファイア)プロパティ



これは C の式です。事前定義変数 _RESULT_ に値がゼロでなければ True(真)、そうでなければ False(偽)を結果にセットします。これがパスした場合、リンク状態として使用される成功フラグが上がります。そうでなければ、異常フラグが上がります。

演算ノードの作成

演算ノードは、数値の処理または単にデータフィールドに定数をセットするために使用されます。収集ノードと演算ノードの両方はデータノードで、これは実際にデータを保持し、そしてレコードのフィールドとして選択することができます。

Basic プロパティ

Comment プロパティ

Screen プロパティ

Screen > Position プロパティ

Calculate > Formula プロパティ

Compare > Qualifier プロパティ

Function Keys プロパティ

Links In プロパティ

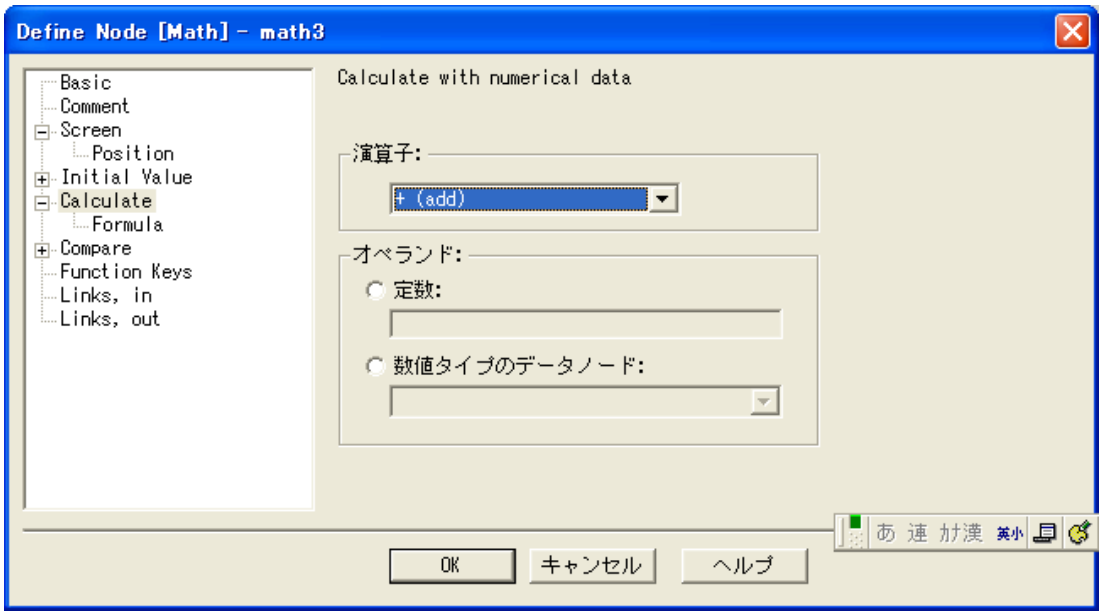
Links Out プロパティ

これらのプロパティは収集ノードのものと同じです。

Initial Value > Data Type プロパティ

演算ノードは計算を目的としているので 数値データのみを持つことができます。違うデータタイプをいっしょに扱う場合、結果は自動的に大きなデータタイプに変換されます。

Calculate プロパティ



計算の演算子とオペランドを定義します。

- **演算子:**

  +(add), -(minus), *(multiply), /(divide), と None(なし).

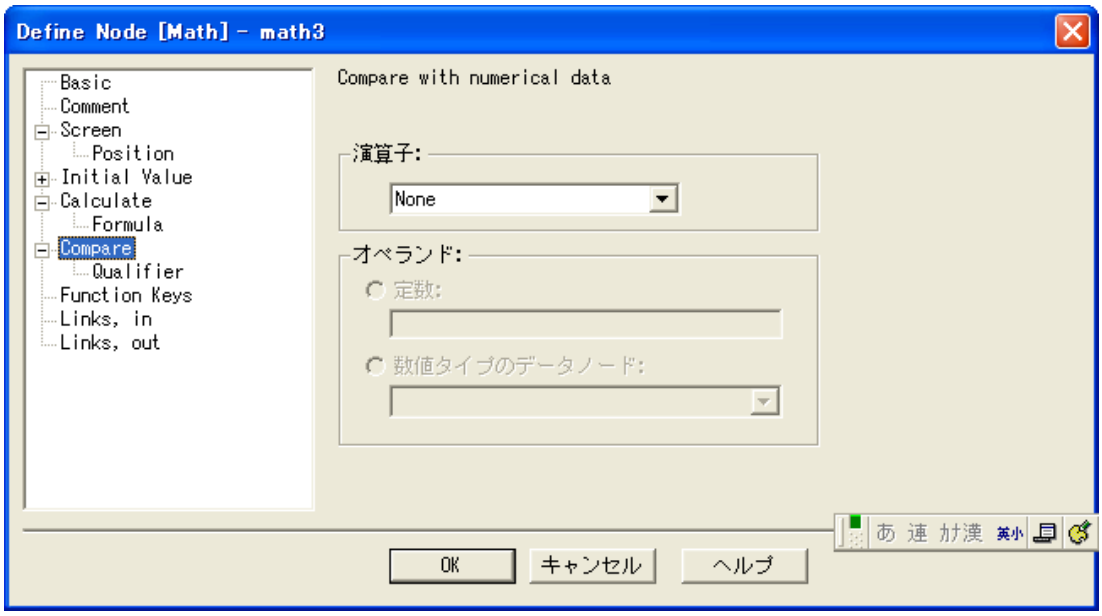
- **オペランド:**
  - **定数:**

    オペランドとして定数を指定。

  - **数値タイプのデータノード:**

    データノード、これは収集ノードまたは演算ノードからデータを得ます。**注意** – 数値データタイプの収集ノードだけをここで選択することができます。

Compare (比較)プロパティ



比較の演算子とオペランドを定義します。比較の結果はリンクの成功または失敗状態として使用することができます。

- **演算子:**

  <(less)、<=(less or equal)、>(greater)、>=(greater or equal)、==(equal)、!=(not Equal)、と None(なし)。

- **オペランド:**
  - **定数:**

    オペランドとして定数を指定。

  - **数値タイプのデータノード:**

    データノード、これは収集ノードまたは演算ノードからデータを得ます。
    注意 – 数値データタイプの収集ノードだけをここで選択することができます。

編集ノードの作成

編集ノードは、実行時にポータブルターミナルのデータファイルに含まれるデータを表示し、変更するために使用します。

編集ノードの定義

Basic プロパティ

Comment プロパティ

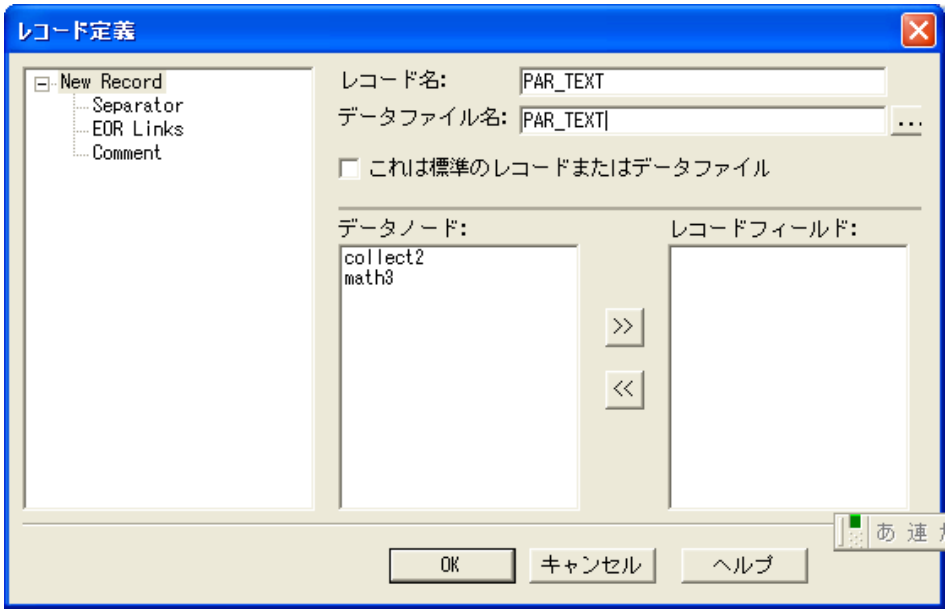
Function Keys プロパティ

Links In プロパティ

Links Out プロパティ

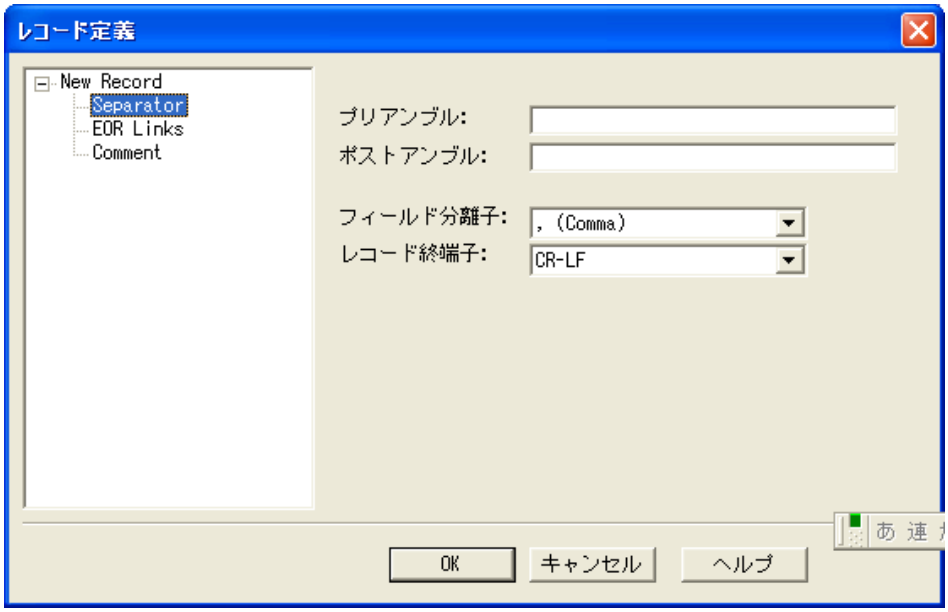
これらのプロパティはメニューノードのものと同じです。

Edit プロパティ



レコード名 データファイル名を定義するためにレコードを選択します。

データファイル名 実行時に編集ノードで操作するファイルを定義します。



フィールド分離子　　フィールド区切り文字を定義します。

編集ノードは、いくつかの異なる作業ノードを持っています。 これはスタート時に**ブラウズモード(表示モード)**に入り、そして最初のレコードの、最初のフィールドにあるデータを表示します。スクリーンは"B,R:xx,F:xx"でユーザに入力要求(プロンプト)を示し、ここで B はブラウズ(Browse)モード、そして R: と F: はそれぞれ現在のフィールドの、レコードとフィールドの座標を表します。

行(レコード)中のフィールドを前方または後方に移動するには右または左の矢印キーを押して下さい。列(フィールド)中のフィールドを移動するには、上（または F1 ファンクションキー）あるいは下（または F2 ファンクションキー)を押して下さい。

フィールドに直接移動(ジャンプ)するには、新しい R:と F:の座標を入力するために F3 ファンクションキーを押して下さい。R の後の、かっこ中の数はファイル中のレコード数の合計です。

ブラウズ(表示)モードで、**編集モード**に入るために Enter を押して下さい。これは入力要求を"E,R:xx,F:xx"に変えます。E は編集(Edit)モードを表します。編集モードでのみユーザはフィールドデータを変更することができます。 ファイルに変更を保存するためにデータを変更した後で再度 Enter を押して下さい。編集ノードの選択はブラウズ(表示)モードに戻ります。

データを変更中に、左矢印キーはバックスペースキーのように働き、前の文字を削除します。右矢印キーを押すと、すべての削除した文字を復元します。このファンクションは、データの内容を覚えなくてもよいので変更を容易にします。

検索モードに入るために F4 ファンクションキーを押して下さい。検索は現在のフィールド番号で定義された欄について検索を現在のレコードから開始します。入力要求は"S,R:xxx,F:xxx"を表示し、ここで R は開始レコード番号を表し、そして F は列番号を表します。入力要求の後で、ユーザに対して検索するための入力データ(標準値は現在の表示データ)を待ちます。

検索は、最初からの部分的なデータが入力データと全く同じ場合に一致します。データを見つけた後で、 SEARCH NEXT? を質問します。YES の答えは次のレコードから続けて検索を始め、あるいは NO は、検索を終了して、一致しているフィールドを表示するために自動的にブラウズ(表示)モードに戻ります。 データが見つからなかった場合、入力要求(プロンプト)は "Not found" を表示し、そして次の検索のための入力待ちに戻ります。検索モードをやめるには ESC を押して下さい。

編集ノードを終了するためにはブラウズモードで ESC キーを押します。

消去ノードの作成

消去ノードはデータファイルからレコードを削除するため、あるいはデータファイル全体を消去するために使用されます。

消去ノード定義

Basic プロパティ

Comment プロパティ

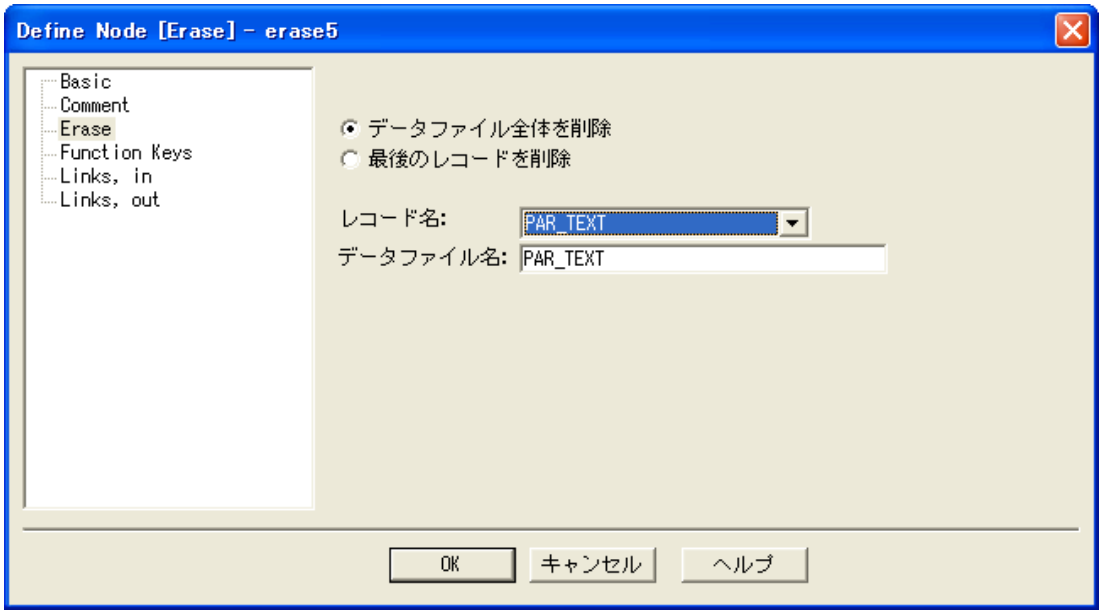
Function Keys プロパティ

Links In プロパティ

Links Out プロパティ

これらのプロパティはメニューノードのものと同じです。

Erase プロパティ



データファイル全体を削除 ポータブルターミナルからデータファイル全体を消去します。

最後のレコードを削除 データファイルから最後のレコードを削除します。

レコード名 データファイル名を定義するために、レコードを選択します。

データファイル名　ファイルを定義します。これは実行時に消去ノードで操作されます。

アップロードノードの作成

アップロードノードは、ターゲットのポータブルターミナルからホストコンピュータへデータファイルを送信するために使用されます。

アップロードノード定義

Basic プロパティ

Comment プロパティ

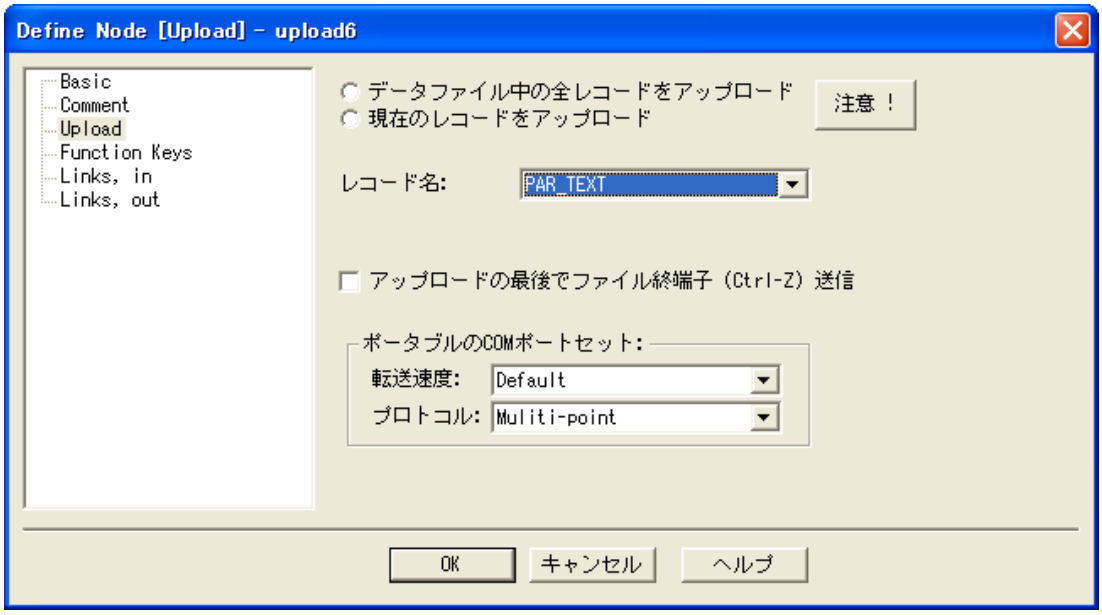
Function Keys プロパティ

Links In プロパティ

Links Out プロパティ

これらのプロパティはメニューモードのものと同じです。

Upload プロパティ



アップロードノードの機能は、**マルチポイントプロトコル** でポータブルターミナルからホストへデータを転送することです。アップロードノードの実行で、送るために定義されたすべてのデータは連続的にデータをポーリングしているホストアプリケーションに即座に送られます。

データファイル注の全レコードをアップロード　ファイル中のすべてのレコードをアップロード。

現在のレコードをアップロード　現在のレコードをアップロード。

レコード名　アップロードするレコードを選択。レコードはどのフィールドをアップロードするかを定義します。

アップロードの最後でファイル終端子送信　アップロードの最後でファイル終端子(Ctrl-Z) を送信。②

プログラムノードの作成

プログラムノード はCプログラムを書くためのスペースをユーザに提供します。

プログラムノードの定義

Basic プロパティ

Comment プロパティ

Function Keys プロパティ

Links In プロパティ

Links Out プロパティ

これらのプロパティはメニューノードのものと同じです。

Basic プロパティ

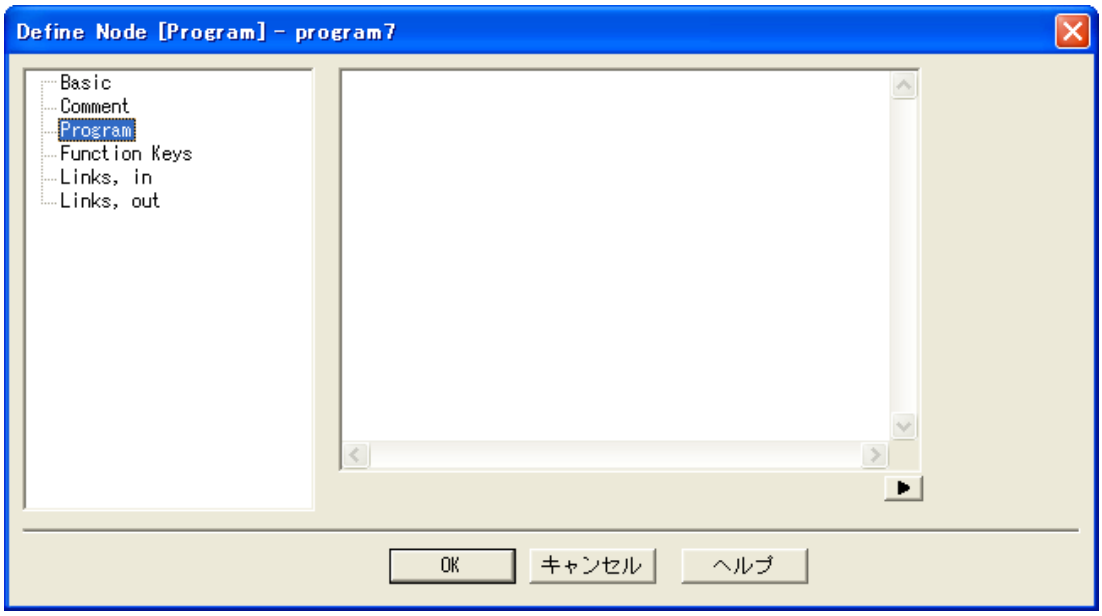
プログラムノードはBasicプロパティに追加の選択があります。

- **グローバルアクセス**

このプログラムで定義された機能に他からアクセス(呼び出す)することを可能にします。例えば、入力式の、検証修飾子(クォリファイア)の、あるいはリンクの他のプログラムノードによって呼ばれます。すべてのグローバルアクセスプログラムノードは左のプロパティパネルのJob ¥ Common Define エントリにリストされています。

C 関数は呼ばれる前に定義されなければなりません。 JobGen Plus はコードを生成するときにすべての、他のノードの前に常にグローバルアクセスノードを置きます。そしてアルファベット順に処理されます。

Program プロパティ



プログラムノードは、C コードで皆さん独自の関数を書くことのできる場所を実際に提供します。編集ボックスにコードを書くか、あるいは大きな編集ウインドウを開くために右下の矢印ボタンをクリックすることができます。戻るには単にウインドウを閉じます。

各プログラムノードは、関数のエントリを持っており、このプログラムノードの実行時に呼び出されます。エントリはプリフィックスの下線('_')が付いたプログラムノードの名前です。例えば、プログラムノード名が "ring"の場合、"_ring()' の名前の関数がこのプログラムの実行時に呼び出されます。

多数の関数がプログラムノードで呼び出されます。これらは、ジョブエンジン関数、ある標準の C ライブラリ関数、そしてファームウェア関数の三種類に分けられます。

ジョブエンジン関数の詳細な説明については、*ヘルプの内容* の 関数ライブラリ をチェックして下さい。標準 C 関数については *Microsoft C 言語ヘルプ* のランタイムルーチンをチェックして下さい。ファームウェア関数の詳細な説明については、ポータブルターミナルのプログラミングマニュアル をご覧下さい。

ジョブ実行ノードの作成

一つのジョブは他のジョブの中で実行することができます。大きなジョブを複数の小さな独立したもの(モジュール)に分割するためにこのテクニックを使用して下さい。小さなジョブはメインテナンスが簡単であることを意味します。

注意、ジョブ実行は、ジョブ呼び出しとは違っています。ジョブ実行は呼び出した側には戻りません。ポータブルターミナルのジョブエンジンは一度にメモリ中に一つのジョブだけを持っています。現在のジョブはジョブ実行が新しいジョブを読み込んだときに消えます。ジョブ実行を実行する前にすべてのデータファイルを保存することを忘れないで下さい。

サブジョブは呼び出したジョブを実行することによって呼び出したジョブに戻ることができます。開始ノード名を指定することによって、ジョブエンジンは、元の開始ノードに代わって指定したノードからスタートすることができます。

プログラムノード定義

Basic プロパティ

Comment プロパティ

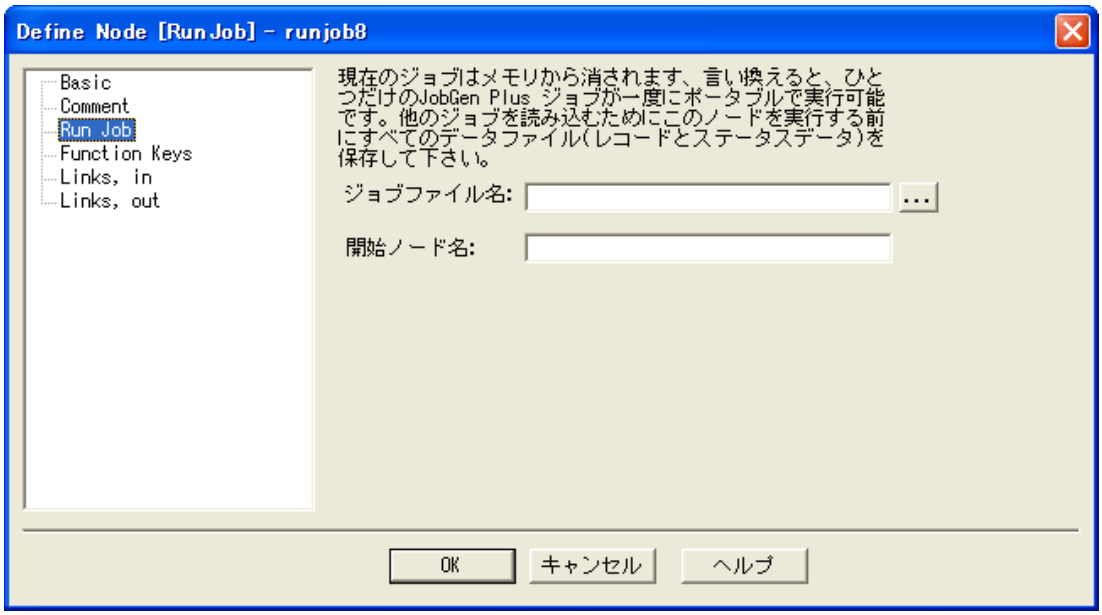
Function Keys プロパティ

Links In プロパティ

Links Out プロパティ

これらのプロパティはメニューノードのものと同じです。

Run Job (ジョブ実行)プロパティ

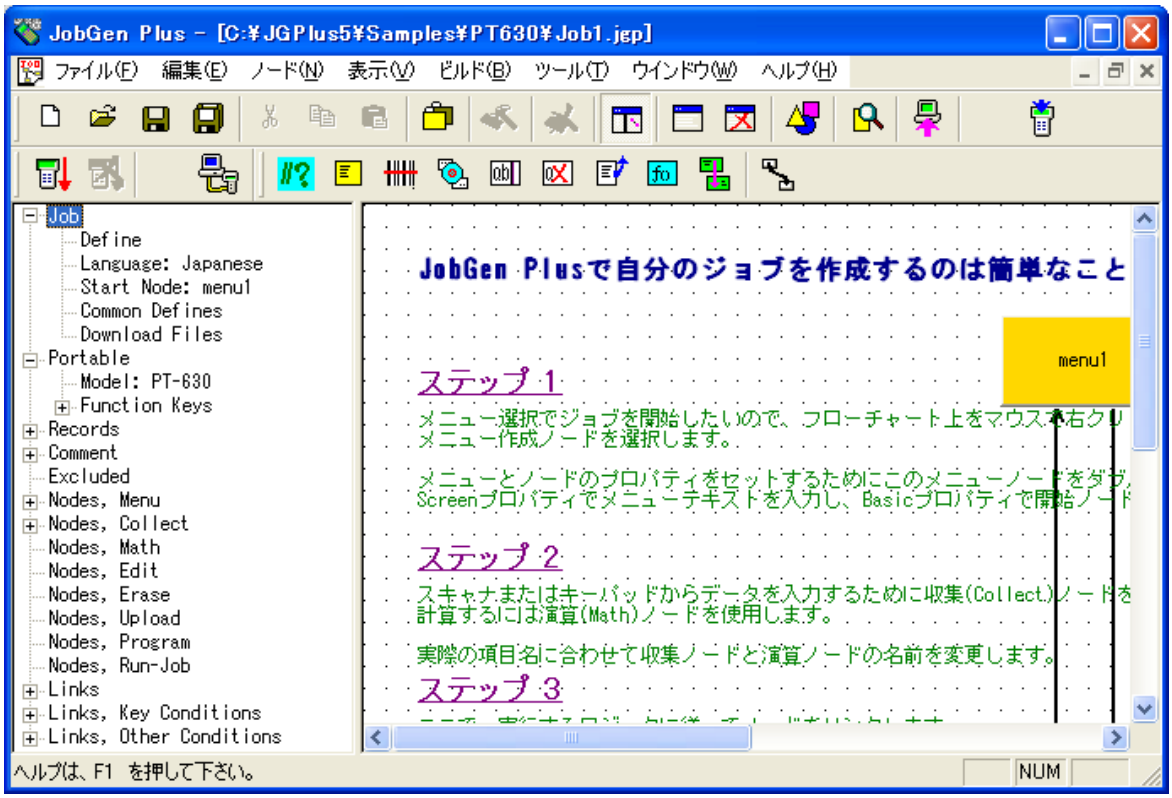


ジョブファイル名　　ジョブ実行ファイル名の定義

開始ノード名　　開始ノード名の定義。標準では、ジョブエンジンはジョブで定義された開始ノードから実行します。これは別なノード名を入力することによって変更できるので、ジョブは指定したノードから開始します。

コメントノードの作成

コメントノードはフローチャート上に直接コメントを貼り付けるために使用されます。



リンクの作成

リンクは JobGen Plus アプリケーションを構築する基本的なコンポーネントです。これらはあるプロセス(ノード)から他へのパスだけではなく、プログラム実行のフローです。

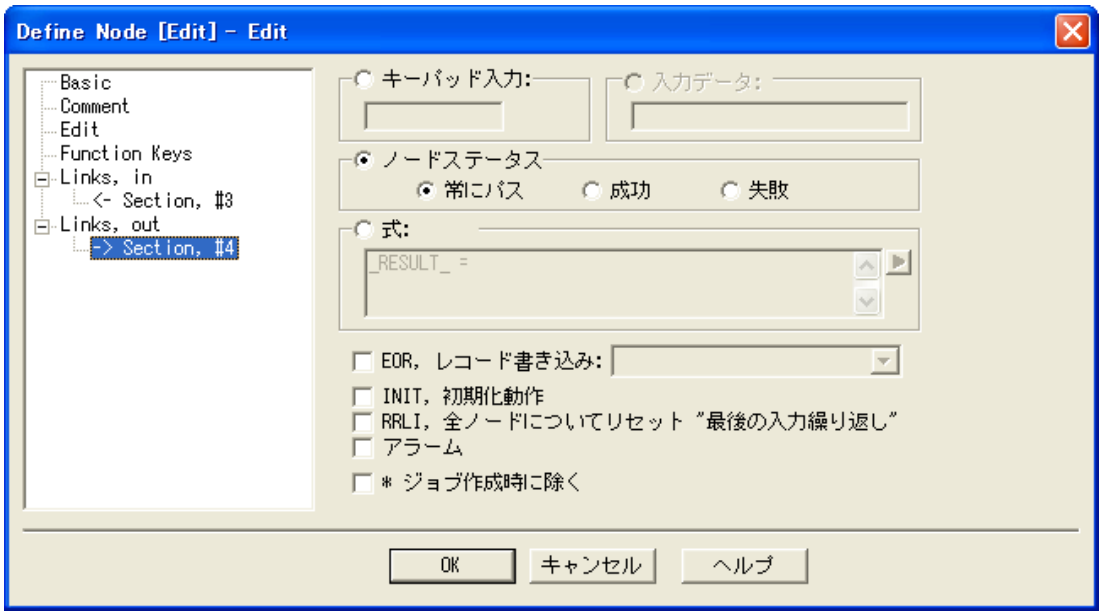
リンクの作成方法: ツールバーのリンクアイコンをクリックするか、あるいはノードメニューでリンクを選択することによってリンクツールを選択します。開始ノードを指して、マウスカーソルを相手先ノードにドラッグして、マウスボタンを離します。

自己ループするリンクを作る方法: ノードボタンを指して、カーソルを一方の側に近づけ、ノードボタン中の別の側にマウスをドラッグして、マウスボタンを離します。自己ループが作成されます。

ノードは複数の出ていくリンクと入ってくるリンクがあります。同じノードに複数のリンクがあるだけでなく、各リンクは異なる条件を持ちます。

接続元のノードから複数のリンクを決めるリンクの条件が選択されます。リンクはリンクの条件が TRUE と評価された時に選択されます。

リンク定義プロパティ



- **キーパッド入力**

このオプションを選択するためにクリックし、そしてポータブルキーパッドウインドウを立ち上げるためにその編集ボックスをクリックします。それをクリックすることによってキーを選択します。リンクについては一つのキーだけを受け入れることができます。ジョブが接続元のノードを終了して、リンク選択についての入力を待っているとき、ここで定義したキーを押すとこのリンク状態が TRUE(真)にセットされます。

- **入力データ**

このオプションは接続元のノードが収集ノードまたはカウントタイプの場合にのみ有効です。これは個々で指定したデータとノードのデータの値を比較して、これらが同じなら結果をTRUE(真)にセットします。

- **ノードステータス**

このオプションは、接続元ノードの実行結果チェックのためのものです。メニュー、消去そして編集ノードについては、その実行は常に成功です。収集ノードについては、その実行は入力データが検証をパスした場合にのみ成功です。アップロードノードについては、タイムアウトが起こった場合、その実行結果は失敗です。

検証は 4 つのオプション常に、成功、失敗、そしてなしを提供しています:

常にパス	接続元ノードの実行結果のステータスには関係ない- これは常に 真(TRUE)
成功	接続元ノードの実行結果が成功の場合にのみこの条件は真(TRUE) になる。
失敗	接続元ノードの実行結果が失敗の場合にのみこの条件は真(TRUE) になる。

- **式**

これは C の式です。これは収集ノードの検証定義のクォリファイア(Qualifier) と同じフォーマットです。事前に定義された変数 **_RESULT_** の値は条件の結果として評価されます。

- **EOR、レコード書き込み(End of Record，レコード終了)**

現在の指定したレコードの入力が完了し、そしてこのリンクが実行された時にデータファイルに保存されることを示すためにこのオプションをクリックします。レコードが定義された場合EOR を定義したリンクが少なくとも一つあります。複数の EOR(これは複数のリンクが EOR 選択を持っている)はレコードを終了するために異なる条件によって適用されます。

ジョブエンジンは、実行があるノードでループしている場合には終了やレコードの保存をしません。ジョブエンジンがレコードを保存するただ一つの方法はリンクの実行時に EOR に合うことです。

どのレコードがデータファイルに保存されるかをジョブエンジンに知らせるためにレコード名を指定して下さい。

- **INIT, 初期化動作**

収集と演算ノードの初期処理で、**新しい値をセット**オプションを有効にする時間であることを示すためにこのオプションをクリックします。

- **RRLI、全ノードについてリセット、”最後の入力繰り返し”**

次のレコードが新しい入力を必要としていることを示すためにこのオプションをクリックします。これは収集ノードの入力セッションで、**最後の入力を繰り返す**設定と一緒に使用されます。ジョブエンジンが新しい入力レコードのチェックされたリンクに会った場合、最後の入力繰り返しのチェックされた収集ノードを実行する次の時に新しいデータの入力を行います。新しい入力レコードの機能は次のレコードのみに影響し、最後の入力繰り返しを持つ収集ノードの後、リンクで再度新しい入力レコードに会うまで繰り返すことができます。これは欄中の繰り返されたフィールドが繰り返し新しい入力を得ることができる一つの方法です。

- **アラーム**

このリンクを実行しているときにブザー音をならすためにこのオプションをクリックします。

リンク条件の優先度

ジョブがノードでの実行を終えた後で(この時点で、このノードは接続元のノード)、どのノードを次に実行するノードかを決めます。この決定はすべての出ていくリンクのすべての条件をチェックすることによって、そしてどれがリンク状態に合うかを決定することによって行われます。 これらの条件の評価は以下の優先度順に行われます。

1)．ノードが実行の中間で入力を待っている場合に（例えば、収集モードでのキーパッド入力)、特殊キー を押すと現在の実行を終了します。ジョブエンジンは、ちょうど押された特殊キーがこれらのリンクの一つで**キーパッド入力** 条件で定義されている場合に決定するためにこのノードから接続されているすべてのリンクをチェックします。見つけた場合、このリンクが選択され、そして実行フローはこのリンクの相手先ノードに行きます。このようなキーがこれらのリンクで定義されていない場合、特殊キーを押すと無視され、ノードの実行が続きます。

注意： 特殊キーはこれが別なノードで定義されたものかどうかを見るためにもチェックされます。ジョブエンジンがこの特別なファンクションキー定義されているノードを見つけた場合、ノードは実行する次のノードとして選択されます。

2)．接続元ノードの実行が終わった場合、ジョブ・エンジンは**ノードステータス**条件でアクティブにセットされた**成功**または**失敗**があるかどうかを決めるためにノードから接続されているすべてのリンクをチェックします。リンクが**失敗**に定義されており、そして接続元ノードの実行結果が**成功**の場合、このリンクは選択されません。しかし、もし結果が**失敗**の場合、このリンクが選択され、そして実行はこのリンクの相手先ノードに行きます。

3)．**式**の条件があり、そして論理表現の計算結果が真(TRUE)、そしてリンクが選択された場合、実行はこのリンクの相手先ノードに行きます。

4)．**入力データ**条件が定義されており、そしてデータが定義と一致し、そしてこのリンクが選択された場合、実行はこのリンクの相手先ノードに行きます。

5)．**常にパス**オプションが**ノードステータス）**条件で定義されている場合、ジョブはこのリンクの相手先ノードに行きます。**常にパス**オプションを定義しているリンクがない場合、このステップではリンクは選択されず、そしてジョブは入力を待ちます。

6)．**キーパッド入力**条件が定義されており、そして押されるキーがこの定義と一致し、そしてこのリンクが選択された場合、実行はこのリンクの相手先ノードに行きます。

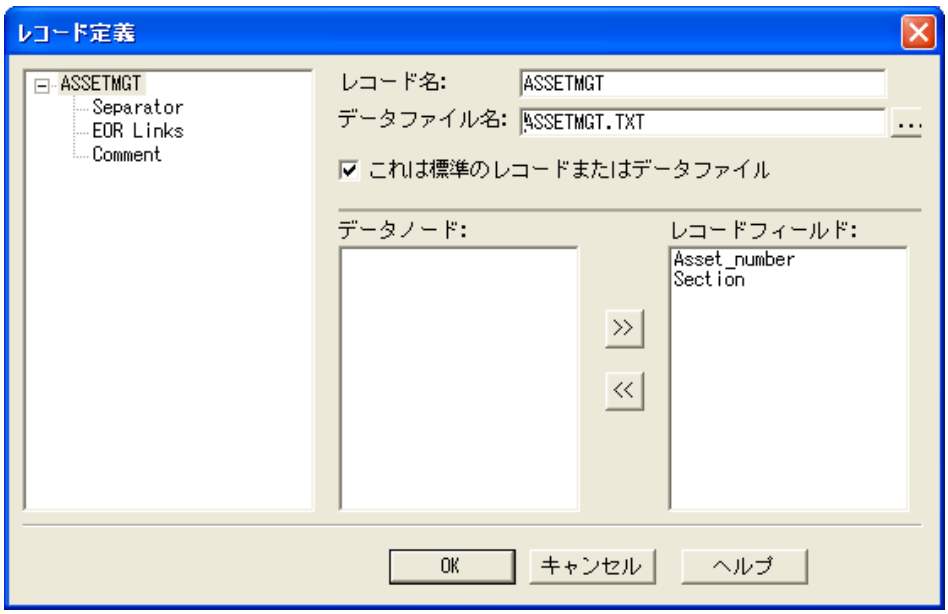
リンクを選択する 6 つの優先度があり、そしてすべてのリンクはリンクのシーケンス番号順に評価されます。リンクが選択された場合、実行は低い優先度の条件をチェックしないで直ちに相手先ノードに行きます

これらの優先度のステップを実行した後で、リンクが見つからなかった場合、ジョブはここで停止します。これが不適切なプログラミングの結果起こった場合、ジョブを終了するために ESC キーを押して下さい。

このノードから接続されているリンクが全くない場合、ジョブの実行は自動的に終了します。

レコードの定義

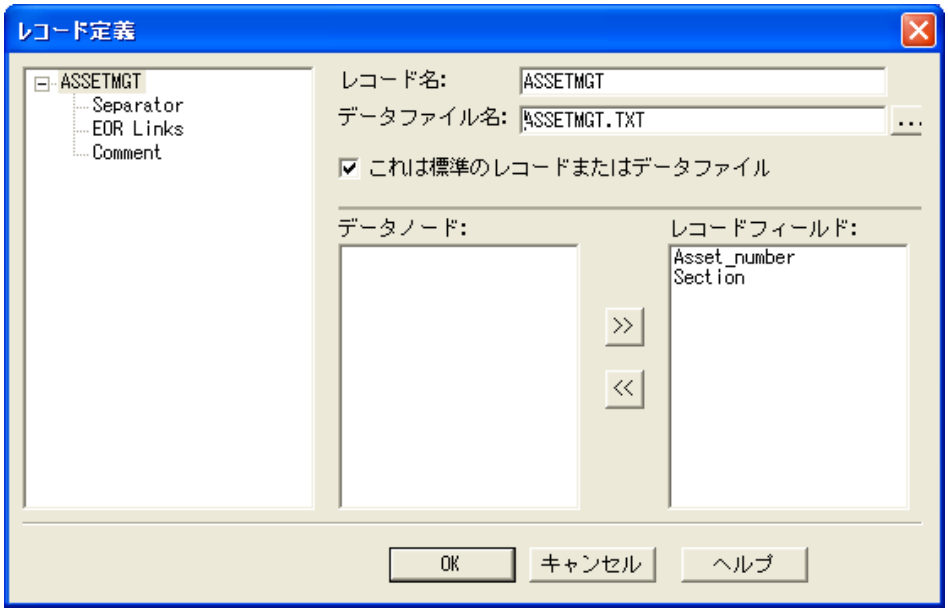
各ジョブは複数のレコードを持つことができます。レコードはフィールド、レコード区切り、プリアンブルとポストアンブルで構成されています。



レコードの属性を書く**フィールド**は、収集ノードです。収集ノードを定義して、そしてレコードフィールドのチェックボックスを選択した場合、 このノードは自動的にフィールドになります。標準では、JobGen Plus はすべての定義された収集ノードをレコードのフィールドとしてセットし、そしてレコード中のフィールドの順序はノードの定義の時刻 – 最初に定義された収集ノードが最初のフィールド、次の収集ノードが二番目、等々 - で決定されます。**フィールド分離子**はフィールドの分離のために使用され、そして**レコード終端子**はレコードを分離するために使用されます。プリアンブルとポストアンブルは文字列です。プリアンブルはレコードのデータの前にセットされ、ポストアンブルはレコードのデータの後にセットされます。

各レコードは、データを保存するために TXT ファイルを持っています。データファイル名はユーザによって定義することができ、そしてその標準名はジョブ名に拡張子.TXT が付いています。例えば、 demo .txt はジョブ demo のデータファイルです。データファイルはデータファイルフォーマットで定義された多数のレコードを保存します。

レコードを定義する二つの方法があります。最初の方法は収集ノードの定義時に行われますが、レコードのフィールドだけがこの方法で定義されます。他の方法はレコード定義ダイアログボックスを出すために編集メニューからレコード定義を選択することです。



レコード定義ダイアローグボックスには、二つのスクロール・ウインドウ: **データノード:** 、左のウインドウと**レコードフィールド:** 、右のウインドウがあります。ジョブで作成されたすべての収集ノードは、**データノード:** ウインドウにリストされ、そしてレコード中のすべてのフィールドは、**レコードフィールド:** ウインドウにリストされます。これらの二つのウインドウ間には**追加(>>)** と **削除(<<)** ボタンがあります。

収集ノードをレコードのフィールドになるようにしたい場合は、選択するために**データノード:** ウインドウでノードをクリックし、そして **追加(>>)** ボタンをクリックします。このノードはレコードのフィールドとなり、そして **レコードフィールド:** ウインドウに現れます。

レコードのフィールドを削除するには、**レコードフィールド:** ウインドウのノードをクリックし、そして **削除(<<)** ボタンをクリックし、そしてフィールドは削除され、そしてウインドウから除かれます。

フィールド と **レコード分離子(区切り文字)**を定義するための二つのプルダウンウインドウがあります。カンマ、セミコロン、タブ(ASCII 9)、そして LF&CR 文字のみがフィールドまたはレコード区切り文字として選択することができます。一つの文字を同時にフィールドとレコードの両方を区切り文字として選択することはできません。

フィールドの順序はフィールドが作成される順に従っています。

プリアンブルとポストアンブルは適当な編集ボックスで文字列をタイプすることによって定義することができます。

最終的に、作業を保存するために OK をクリックして下さい。

レコードフィールドチェックボックスで変更の効果を見るために、Define Collect (収集定義)ダイアローグボックスを立ち上げるために変更した収集ノードをクリックします。

ジョブエンジンは、実行があるノードの輪でループしている場合、レコードの完了と保存しません。これは単に古いデータに上書きします。ジョブエンジンがレコードを保存するただ一つの方法はリンクの実行中で EOR(End Of Record)に出会ったときです。

データファイルフォーマット

データはレコードごとにデータファイルに保存されます。レコードの最大長は 1024 バイトです。レコード・フォーマットは以下の通りです:

[Pre]<f1><fd><f2><fd>...<fn>[<fd>[Post]]<EOR>[CR&LF]

ここで:

<f1><f2>...<fn>:	フィールド 1、フィールド 2、 ...フィールド n フィールドの最大長は 64 バイトです。 N はレコードのフィールド番号です。最大値はフィールドの長さ、プリアンブルとポストアンブルの長さに依存します。
<fd>:	ユーザ定義のフィールド区切り文字(分離子)。
[Pre]:	ユーザ定義のプリアンブル。
[<fd>[Post]]:	[Post] はユーザ定義のポストアンブル。ポストアンブルがある場合、fd はその前に挿入しなければなりません。
<EOR>:	ユーザ定義のレコード区切り文字(終端子)。
[CR&LR]:	EOR が CR&LF でない場合、JobGen Plus はレコードの最後に自動的に CR&LF を追加します。EOR が CR&LF の場合、 JobGen Plus はそれを無視します。レコードの最後には CR&LF がひとつだけでなければなりません。

第 6 章 ジョブの作成

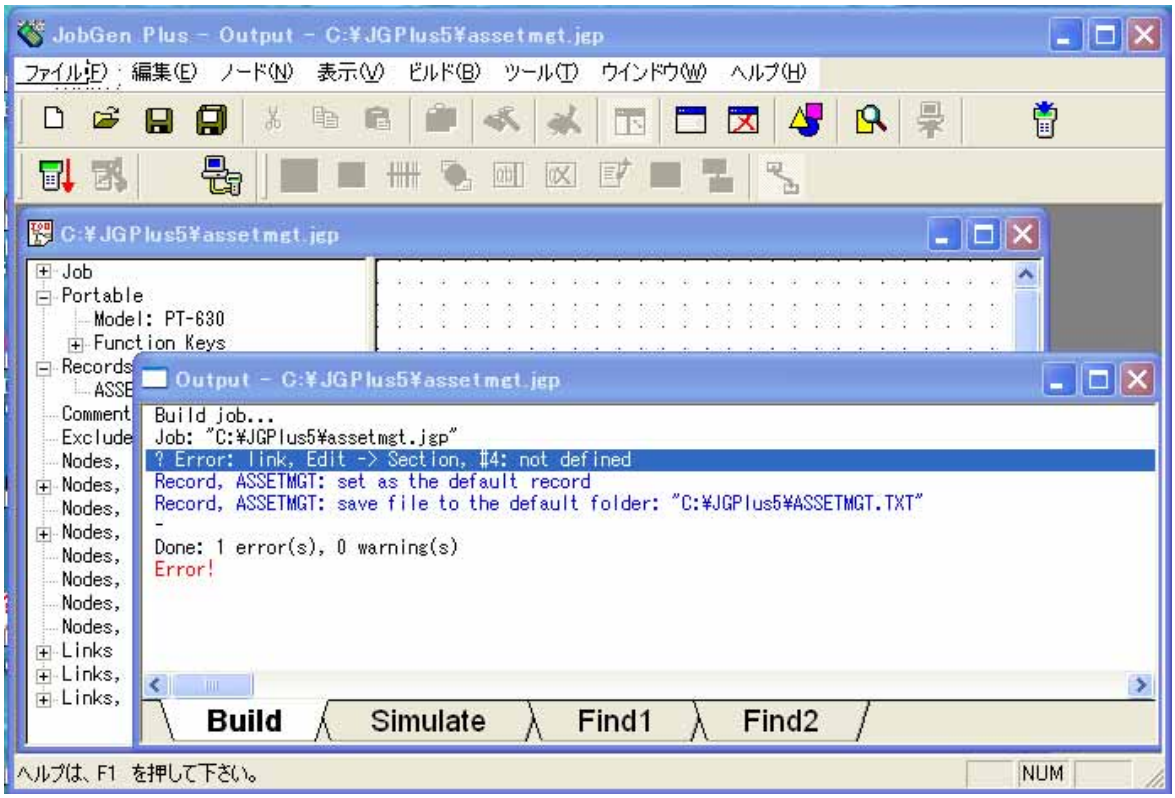
すべてのノードとリンクを作成した後で、次のステップは実行可能なジョブを作ることです。

JobGen Plus は Job の実行形式コードを作るために二つのステップを行います。最初は、フローチャートですべての設定に従ったソースコードを生成します。メッセージウインドウにエラーと警告があれば表示します。そして、最終的に実行可能なコードを生成するためにソースコードをコンパイルします。メッセージウインドウにまたエラーと警告があれば表示します。

エラーがあった場合(赤文字でメッセージウインドウに表示される)、その上を単にダブルクリックすると、関係する定義ダイアローグボックスがポップアップします。問題を修正するために設定を変更して、**ジョブ実行**を再度実行して下さい。

ジョブ実行の途中で、JobGen Plus はバイナリインデックスサーチ・インデックステーブルの構築）または二バイト文字フォントファイルの構築などいくつかの他の操作を起動します。これらすべての動作はメッセージウインドウにレポートされます。

ジョブの実行形式が正しく生成された後で、JobGen Plus はジョブをポータブルターミナルにダウンロードするために PTComm マネージャを起動します。ダウンロードは、すべてのルックアップファイル、バイナリサーチインデックステーブル、そして二バイトフォントファイルなどのすべての関連するファイルを含んでいます。



実行形式がターゲットのポータブルターミナルに正しくダウンロードダウンロードした後で、ユーザがポータブルターミナルで ”RUN” コマンドの実行ができる状態となり、JobGen Plus のジョブアプリケーションを開始します。

第 7 章 ジョブのシュミレーション

ジョブが設計され、定義された後で、これはホスト PC でシュミレーションをすることができます。シュミレーションをスタートするにはツール > ジョブシュミレーションを選択するか、あるいはツールバーのシュミレートアイコンをクリックします。

ジョブシュミレーションは実際のポータブルターミナルがなくてもジョブの確認をする容易な方法を提供します。 シミュレーションはジョブで選択したポータブルターミナルのモデルによく似たポータブルターミナルのウインドウを表示、そしてジョブと Windows の下で DOS セッションのジョブエンジンを実行します。これはすべての入力と出力をシュミレーションします。すべての操作は実際のポータブルターミナルとほとんど同じです。

ジョブのシュミレーションにはある制限があります。ルックアップ検証におけるファイル検索はファイルの最初から 64K バイトを越えることはできません。

ジョブシュミレーションはジョブの開発の助けになります。実際のポータブルターミナルにダウンロードすることによってすべてのジョブのテストを行い、そして各機能の実行をされることを推奨いたします。これはジョブが正しいことを確認するただ一つの方法です。

第 8 章 言語サポート

JobGen Plus はポータブルターミナルのスクリーンで複数の言語をサポートしています。インストール時に言語を選択して下さい。JobGen Plus はフォントのビットマップファイルなどのサポートファイルを自動的に生成し、またジョブを作成し、そしてジョブ実行ファイルと共にポータブルターミナルにこれらをダウンロードします。

日本語の表示は 大型フォントのみをサポートしています。ジョブのプロパティパネルの、プロパティの項: Portable >> Setting を開きます。スクリーンから大型フォントを選択します。

漢字サポート

JobGen Plus は Windows 日本語版のフォントライブラリから漢字のビットマップを探し、コードとビットマップの両方を含むファイルを作成します。このファイルはジョブの実行形式と一緒にポータブルターミナルにダウンロードされます。フォント・タイプ、スタイル、そしてサイズを選択することができます。ポータブルターミナルに表示される漢字のビットマップサイズは 16 x 16 ピクセルです。ビットマップの文字位置のレイアウトを調整することもできます。

漢字ファイルはジョブで共有することができます。複数のジョブが一つの漢字ファイルを使用することができるので、システムのメモリを節約することができます。ジョブエンジンはジョブの漢字ファイル(jobname.ccb)。 を最初に開こうとします。これに失敗すると、標準の CCB ファイルを開こうとします(jeng.ccb)。