JobGen Plus ユーザマニュアル

For Microsoft Windows

Version 5.2

Document Revision 5.0, May 2001

翻訳: ユニテック・ジャパン株式会社、2002 年 2 月

目　次

第 1 章　概要 2
JobGen　Plus の概要 2
JobGen　Plus のコンポーネント 2

第 2 章　インストール 4
必要なシステム 4
MS　Windows　95/98/Me/NT/2000 へのインストール 4

第 3 章　JOBGEN　PLUS を使う 5
MS　Windows　95/98/Me/NT/2000 から JobGen　Plus を実行 5
JobGen　Plus の作業環境 7
1.　File　コマンド 8
2.　Edit　コマンド 9
3.　Nodes　コマンド 11
4.　View　コマンド 12
5.　Build　コマンド 13
6.　Tools　コマンド 14
7.　Window　コマンド 15

第 4 章　ジョブの設計 18

第 5 章　ジョブのプログラミング 20

INTEGER　(整数)の制限値 40
レコードの定義 70
データファイル・フォーマット 72

第 6 章　ジョブの作成 73

第 7 章　ジョブのシュミレート 75

第 8 章) 言語サポート 77

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古い制限を取り除くことによって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-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(Node)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(Link)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・オプションの選択; 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 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 (Comment Node), メニューノード (Menu Node), 収集ノード (Collect Node), 演算ノード (Math Node), 編集ノード (Edit Node), 消去ノード (Erase Node), アップロードノード (Upload Node), プログラムノード (Program Node), そしてジョブ実行ノード (Run-Job Node)です。

> **コメントノード (Comment Node)** はフローチャートに直接注記を貼り付ける方法を提供します。

メニューノード (Menu Node) はメッセージ、指示あるいはユーザによって選択されたアプリケーションの主要な機能の表示のためにアプリケーション・デザイナーに対してスペースを提供します。

収集ノード (Collect Node) はデータコレクタでユーザによって入力された情報を保存するために使用されます。

演算ノード (Math Node) はデータの計算に使用されます。

編集ノード (Edit Node) 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 (Erase Node) 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 (Upload Node) 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 (Program Node) 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 (Run-Job Node) 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 (Links) は二つあるいはそれ以上のプロセス (ノード) 間の移動を行います。リンク (Links) 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 (Node)) 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 (Link) の中に記録されます。

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)
- 32) MB メモリ)
- Hard) Disk に 5MB 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,) または) COM4)))
- 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) 95/98/Me/NT/2000)

)

MS) Windows) 95/98/Me/NT/2000 へのインストール)

1.) **JobGen) Plus**) システムディスクを) 3.5") フロッピー・ディスク・ドライブに入れて下さい。)

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**) ディスク) (**A:**) または) **B:**)) の入っているフロッピー・ディスク・ドライブをタイプして、) "**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) Setup) Initialization(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 Directory) Name) (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) JGPLUS) 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。)

)

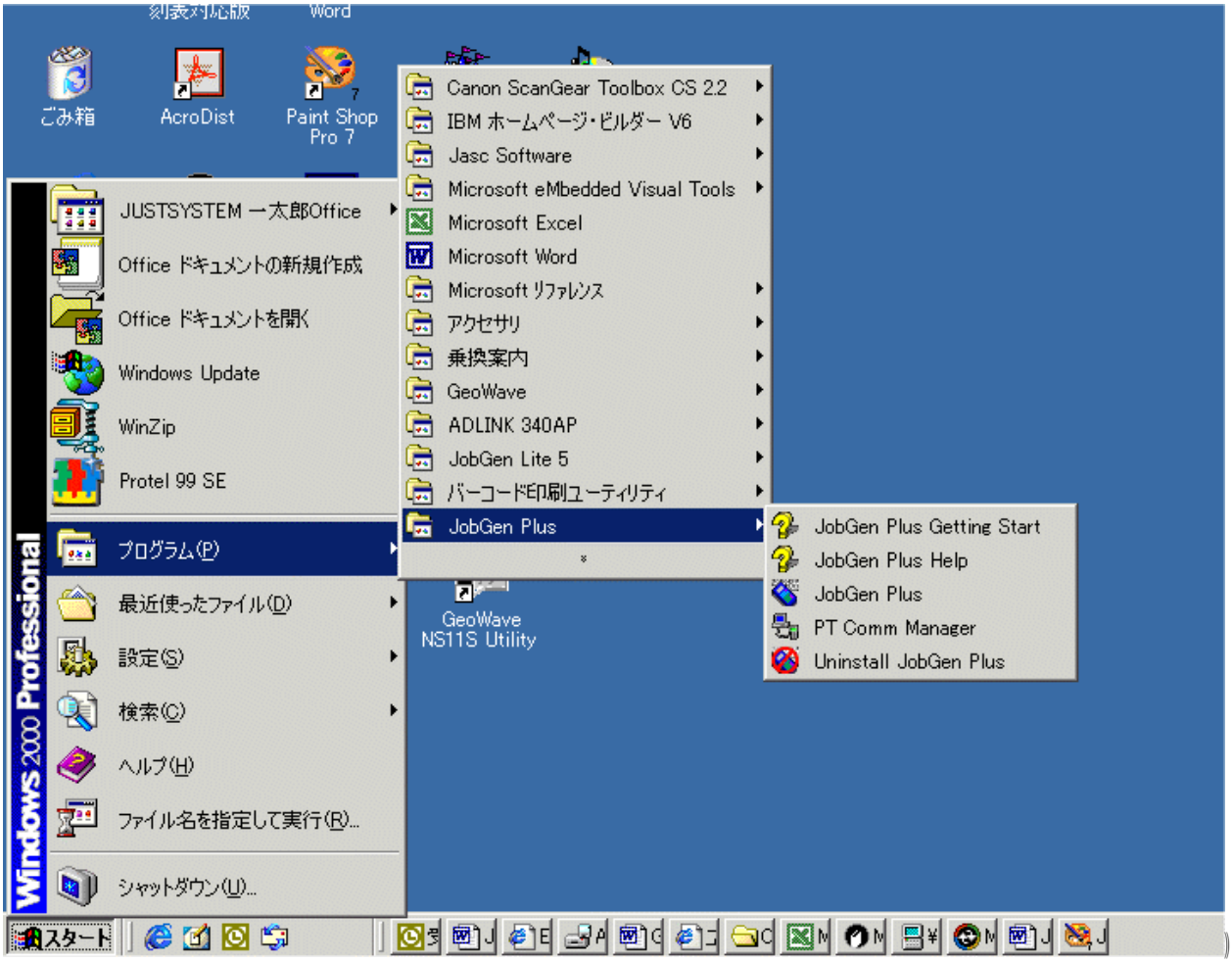
)

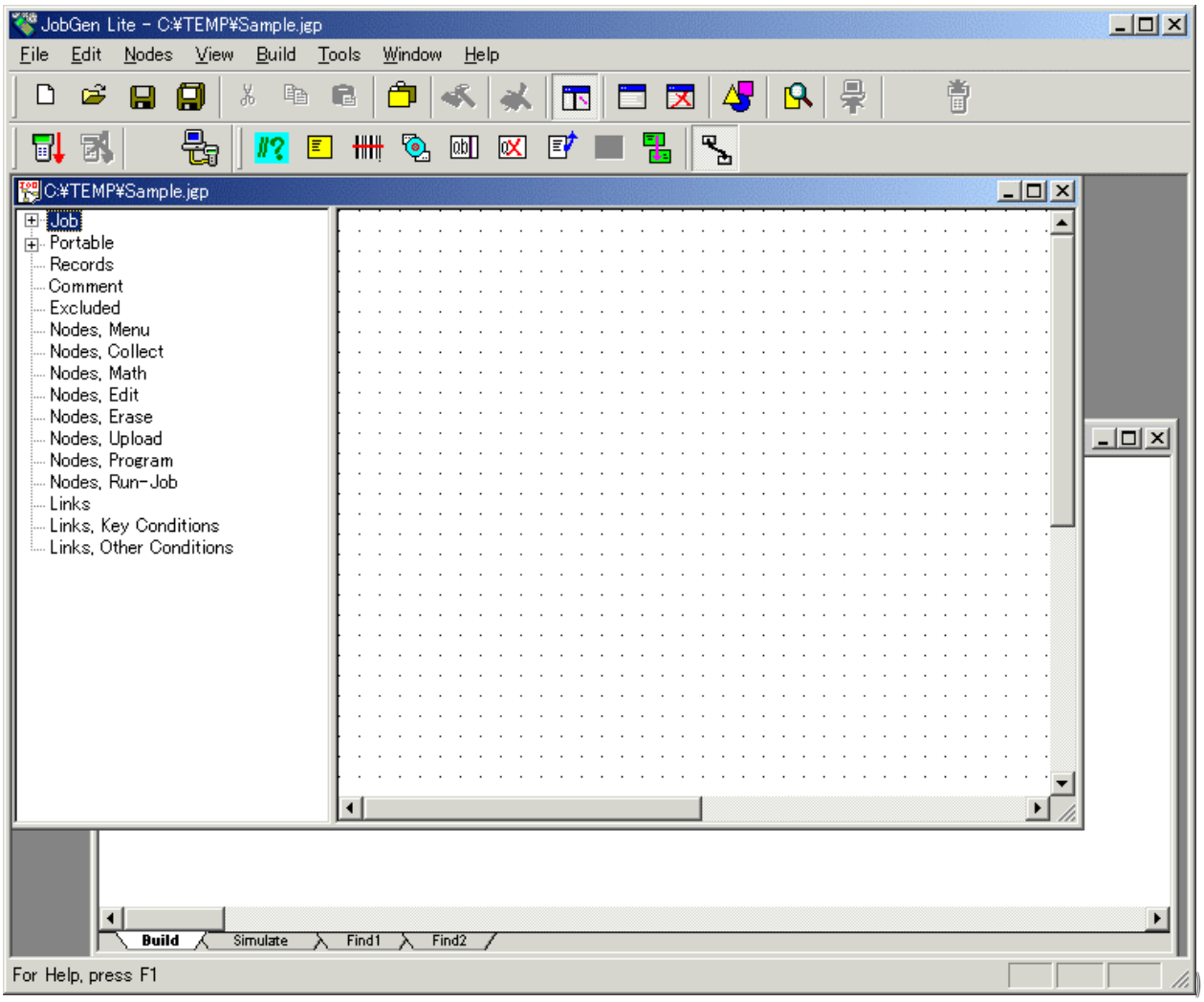
第 3 章 JobGen Plus を使う

MS Windows 95/98/Me/NT/2000 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。)

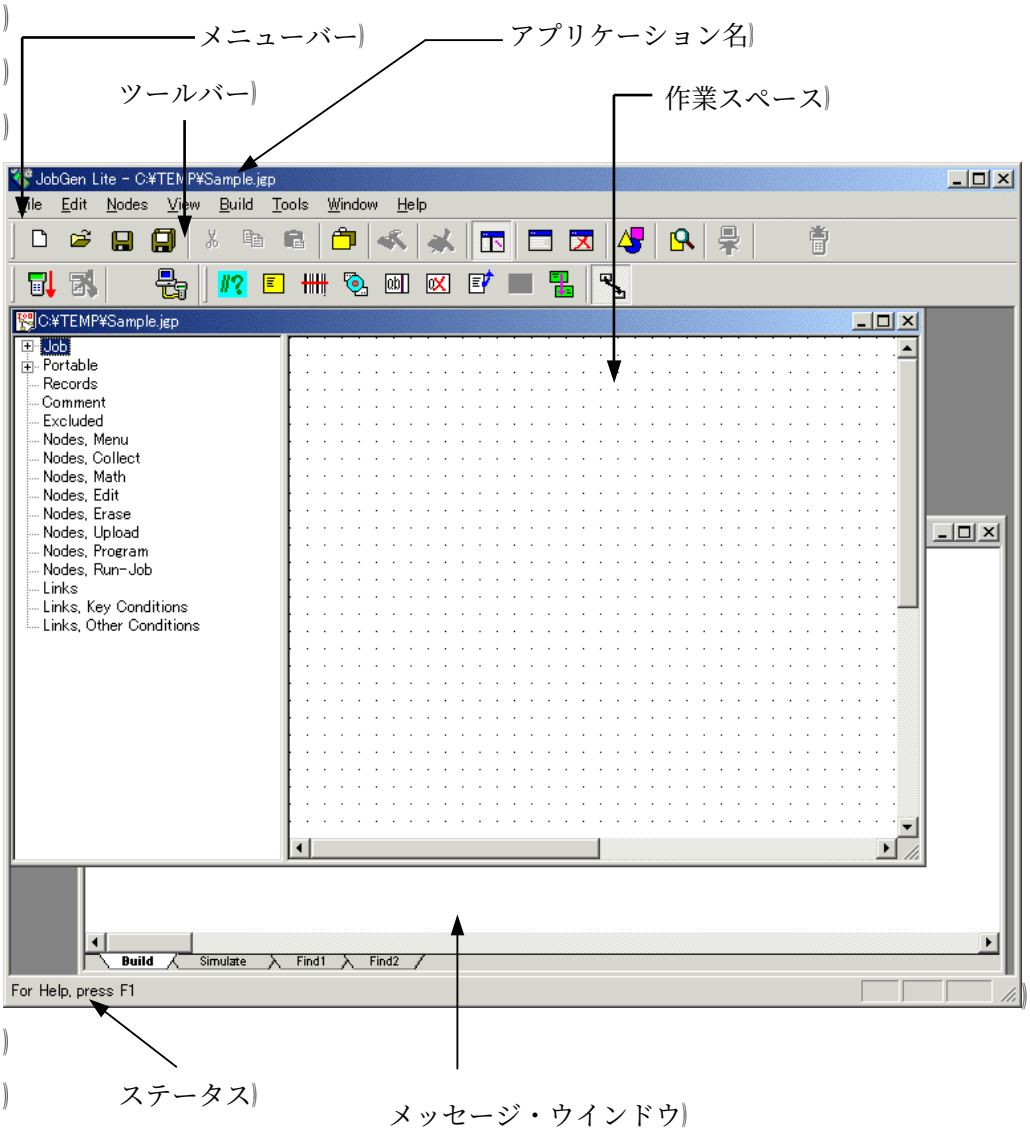




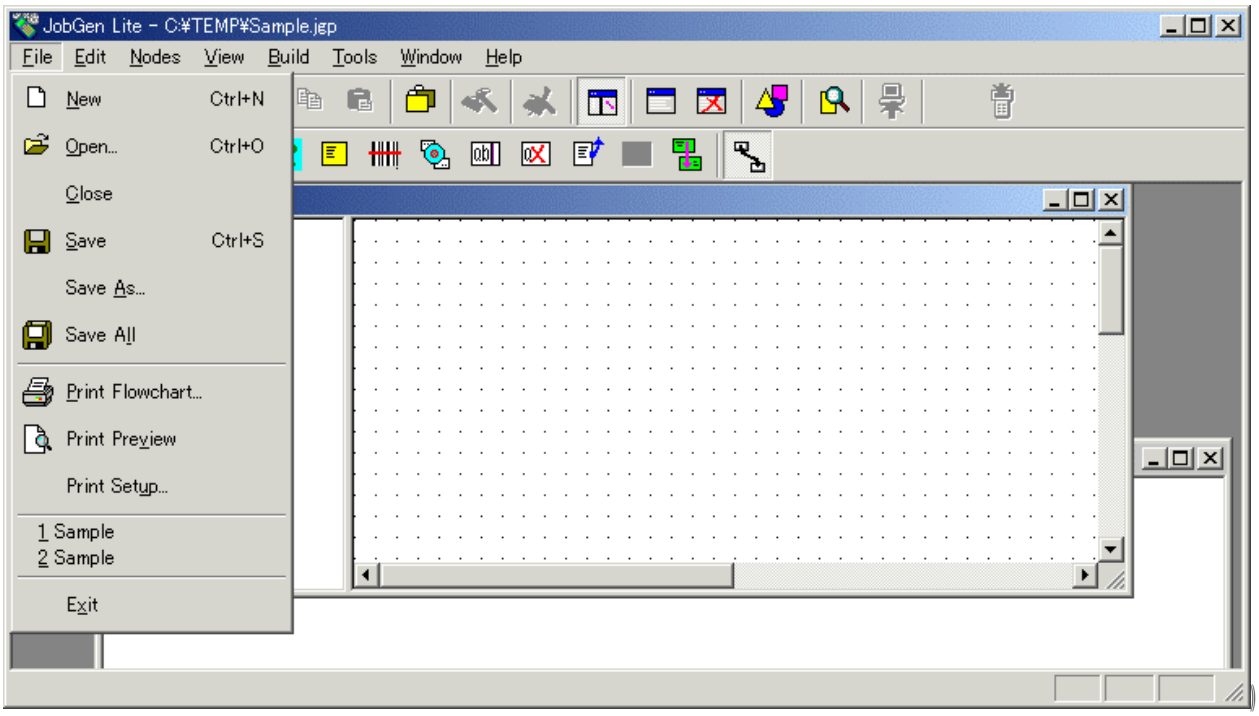
)

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.) File) コマンド)



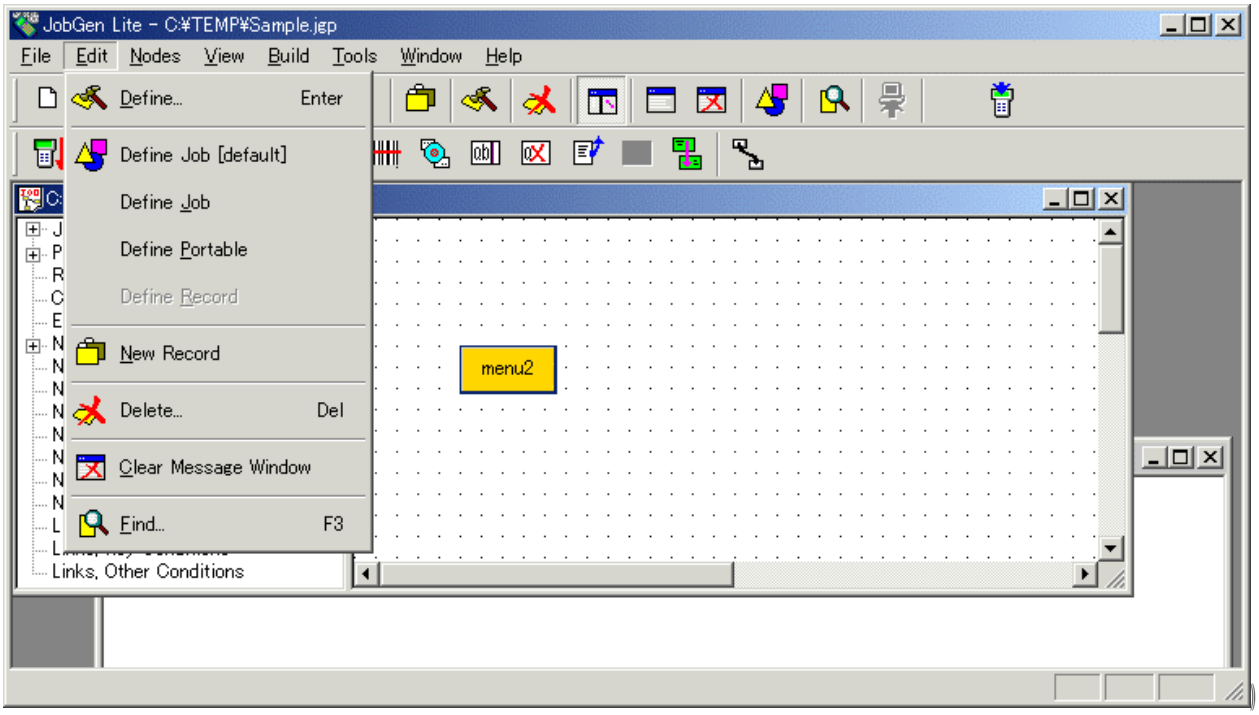
)

New)	新しいジョブを作成する。)
Open)	すでにあるジョブ・ファイルを開く。)
Close)	現在のジョブ・ウインドウを閉じる。)
Save)	現在のジョブの内容を保存する。)
Save) As)	現在のジョブを別なファイル名で保存する。)
Save) All)	すべての開いたジョブの内容を保存する。)
-)	)
Print) Flowchart)	現在のジョブのフローチャートを印刷する。)
Print) Preview)	印刷出力をスクリーン上で見る。)
Print) Setup)	印刷オプションのセットアップ。)
-)	)
Recent) Files)	最近使ったファイルをリストする。)
-)	)
Exit)	**JobGen) Plus)** プログラムを終了する。)

)

)

2. Edit コマンド

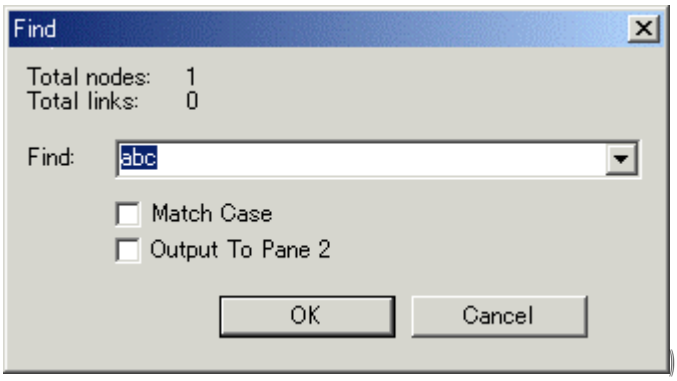


Define	ノード、リンク、あるいはレコードのいずれかの現在選択されているオブジェクトを定義する。
-	
Define Job [default]	ジョブについて、標準値の設定を定義する。
Define Job	現在のジョブを定義する。
Define Portable	ポータブル・ターミナルの設定を定義する。
Define Record	レコード設定を定義する。
-	
New record	新しいレコードを作成する。JobGen Plus は複数レコードをサポートしている。
Delete	ノード、リンク、あるいはレコードのいずれかの現在選択されているオブジェクトを削除する。
Clear Message Window	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する。

Find) 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 Find1 に表示される。) “Output) To) Pane) 2”) 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 Find2 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。)

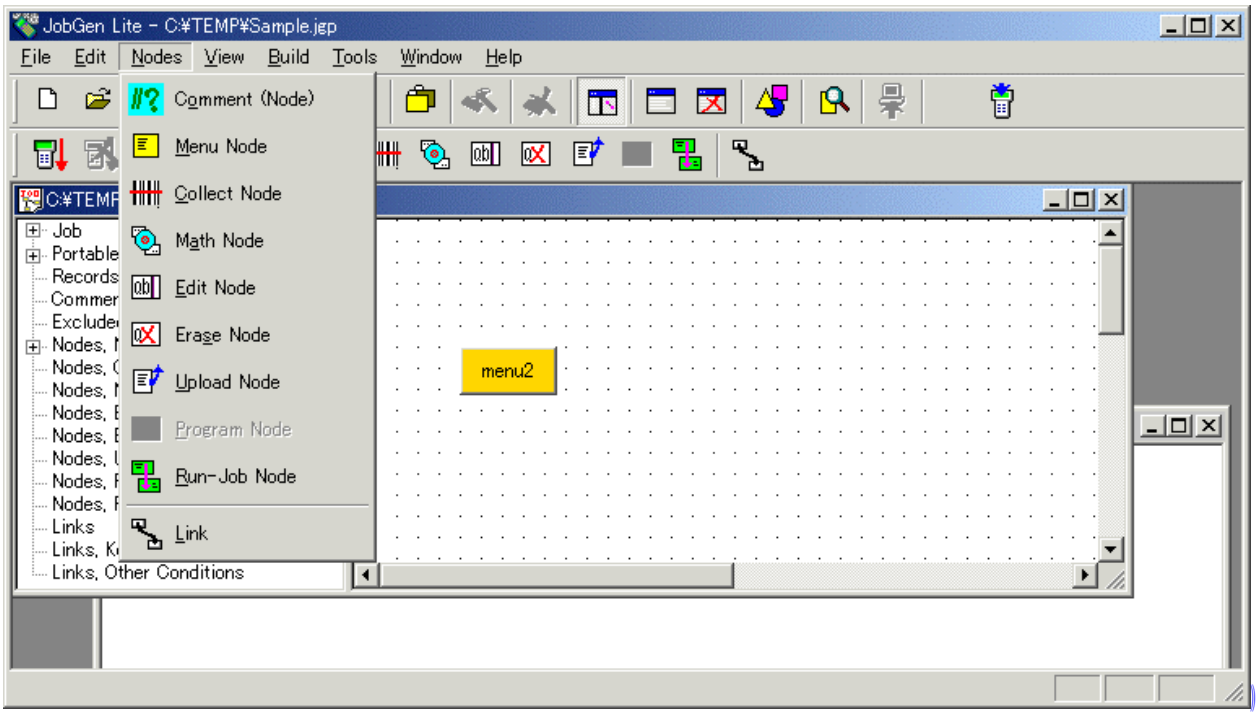
/)

)



)

3.) Nodes) コマンド)



)

Comment (Node)) コメント・ノードを作成する。)

Menu) Node) メニュー・ノードを作成する。)

Collect) Node) 収集ノードを作成する。)

Math) Node) 演算ノードを作成する。)

Edit) Node) 編集ノードを作成する。)

Erase) Node) 消去ノードを作成する。)

Upload) Node) アップロード・ノードを作成する。)

Program) Node) プログラム・ノードを作成する。)

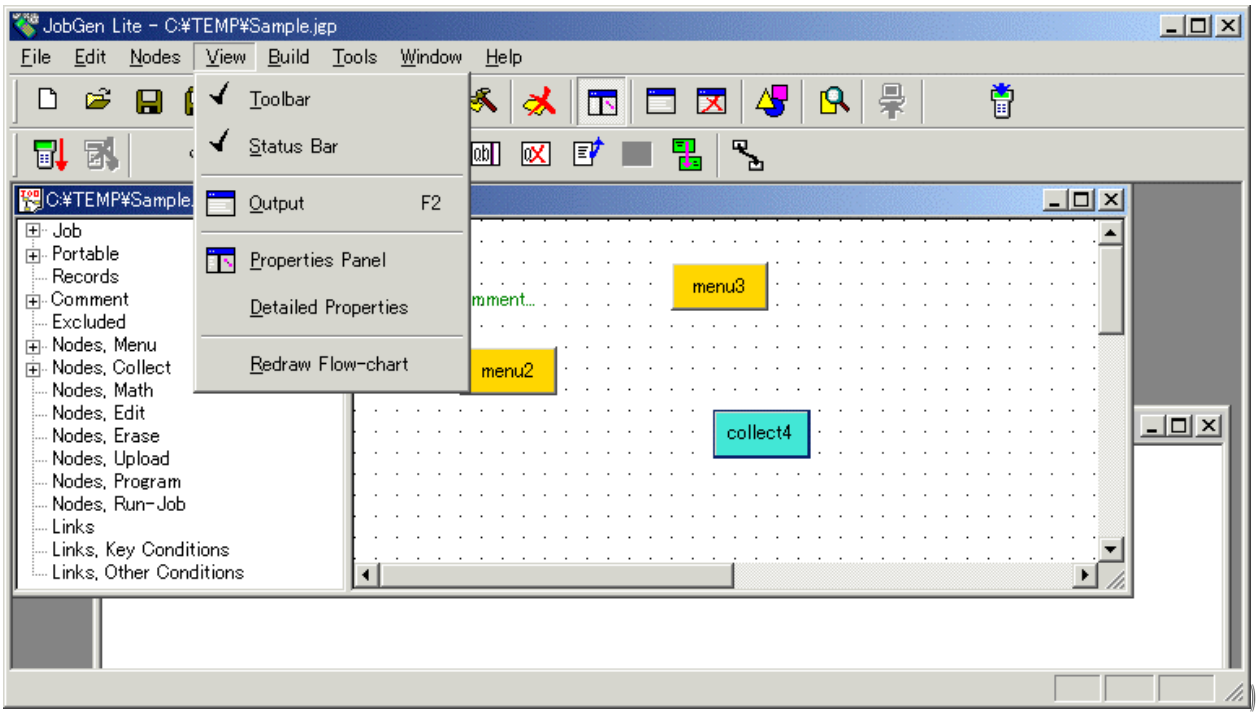
Run-Job) Node) ジョブ実行ノードを作成する。)

-))

Link) リンクを作成する。)

)

4.) View) コマンド)



)

Toolbar) 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。)

Status) Bar) 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。)

-))

Output) メッセージ・ウインドウを開く。)

-))

Properties) Panel) プロパティ・パネルを開く。)

Detailed) Properties) プロパティ・パネルで詳細な情報の表示を可能にする。)

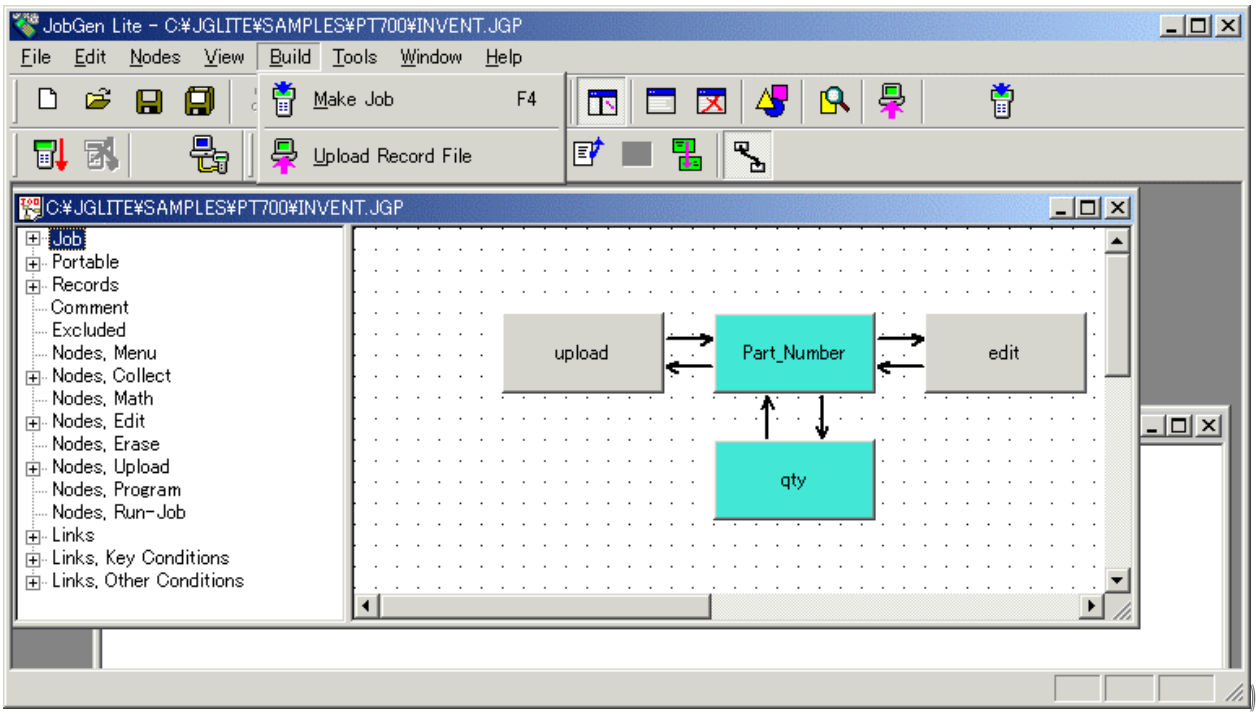
-))

Redraw) Flowchart) 全体のフローチャートを再描画する。)

))

)

5.) Build) コマンド)



)

Make) Job) 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。)

-))

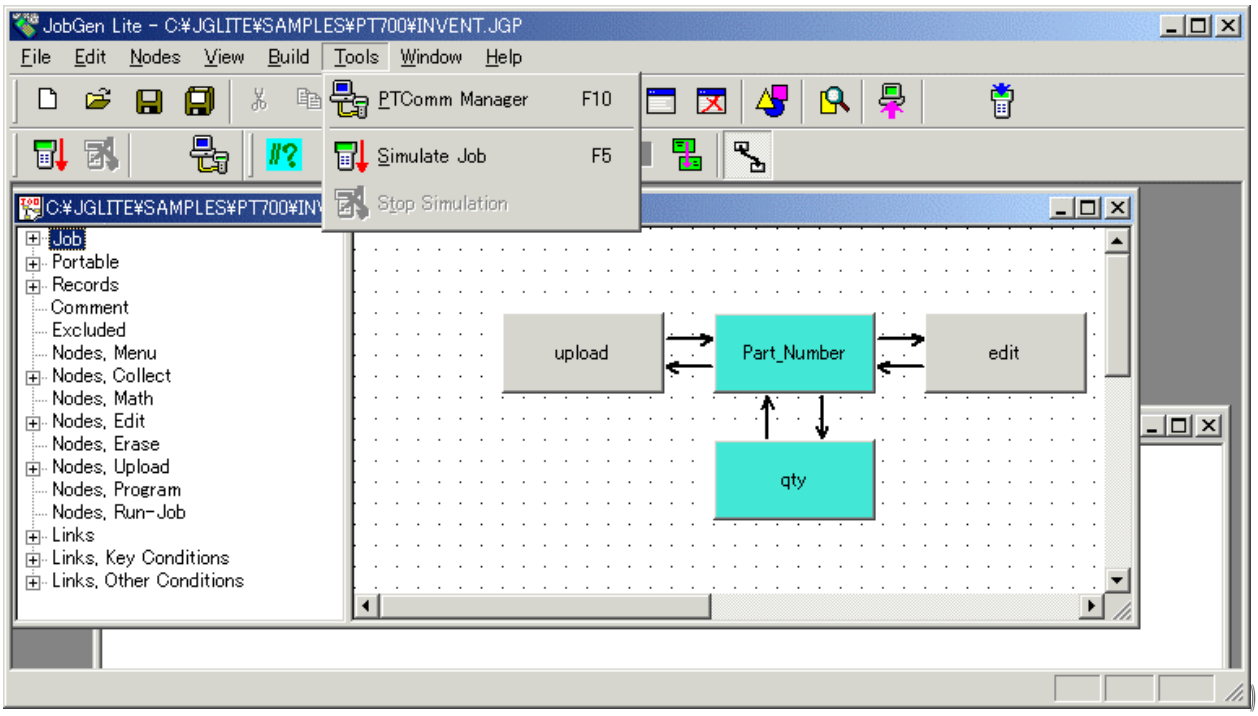
Upload) Record) File) レコード・ファイルをアップロードする。)

))

)

)

6.) Tools) コマンド)



)

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

-))

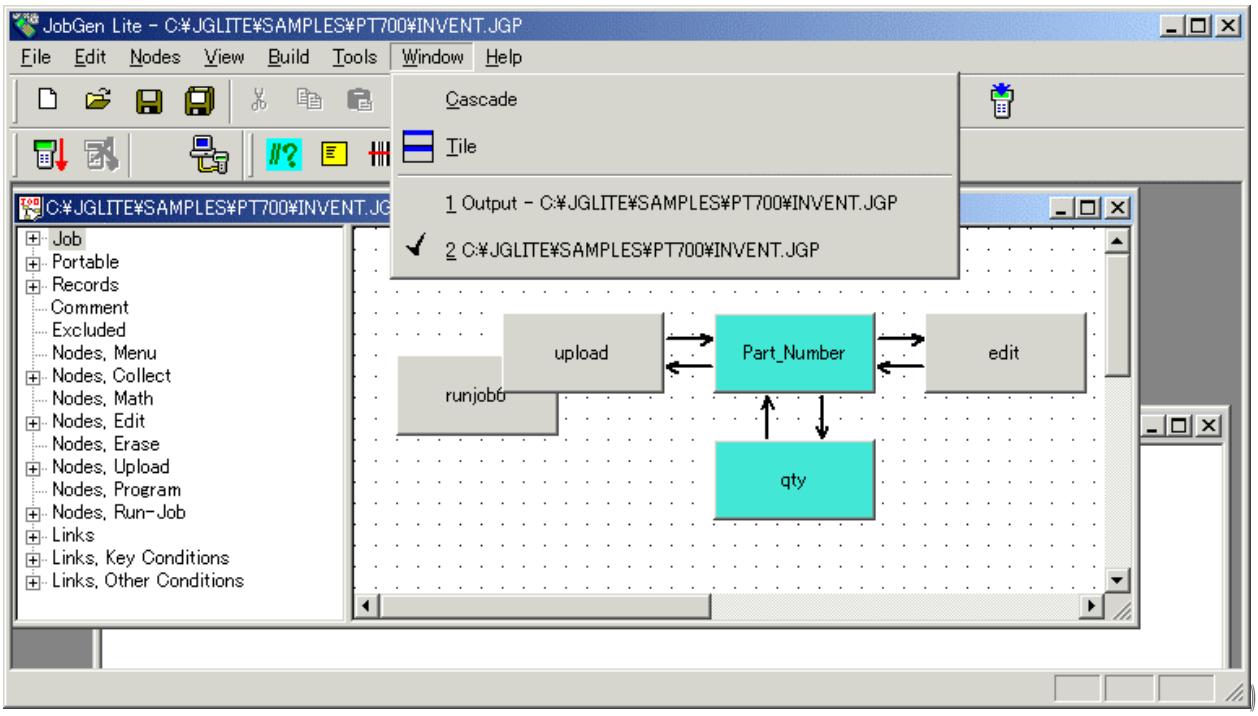
Simulate) Job) ジョブ・シュミレーションを開始する。)

Stop) Simulation) ジョブ・シュミレーションを停止する。)

)

)

7. Window コマンド



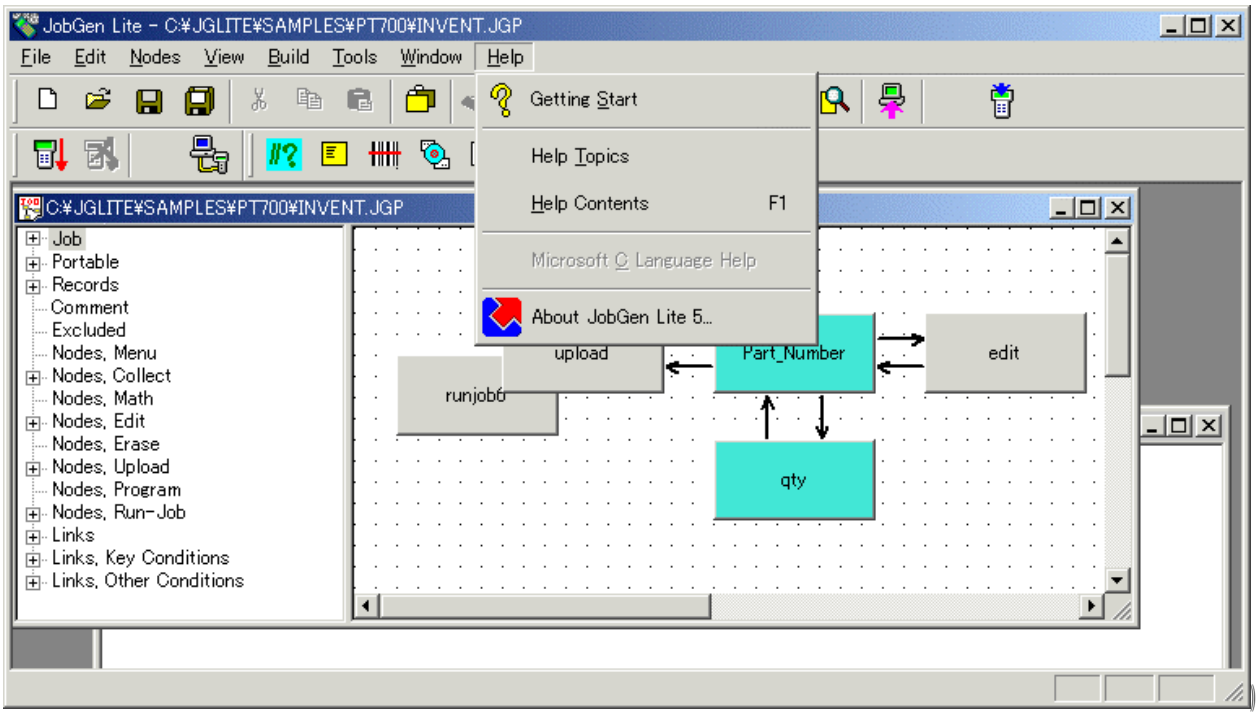
Cascade Window 全てウインドウを重ねて表示する。

Tile 全てのウインドウをタイル表示する。

-

Opened Window *Title* 全ての開かれたウインドウのタイトルをリストする。

8.) Help) コマンド)



)

Getting) Started)	JobGen) Plus の概要を表示する。)
-)	)
Help) Topics)	ヘルプのトピックスを表示する。)
Help) Contents)	JobGen) Plus のヘルプを表示する。)
-)	)
Microsoft) C) Language) Help)	*Microsoft) C) Language) Run-Time) Reference* のヘルプを表示する。)
-)	)
About) JobGen) Plus)	**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 Collect(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)**
Comment) (コメント))	• 注記と説明を表示)
Menu) (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)
Collect) (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) • 表現式からデータを入力) • データ入力の確認)
Math) (演算))	• 表現式からデータを入力) • データの計算) • データの比較)
Erase) (消去))	• 収集したデータ内部のレコードを削除))
Upload) (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)

Edit (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
Program (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
Run-Job (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 Links(リンク)によって行うことができます。

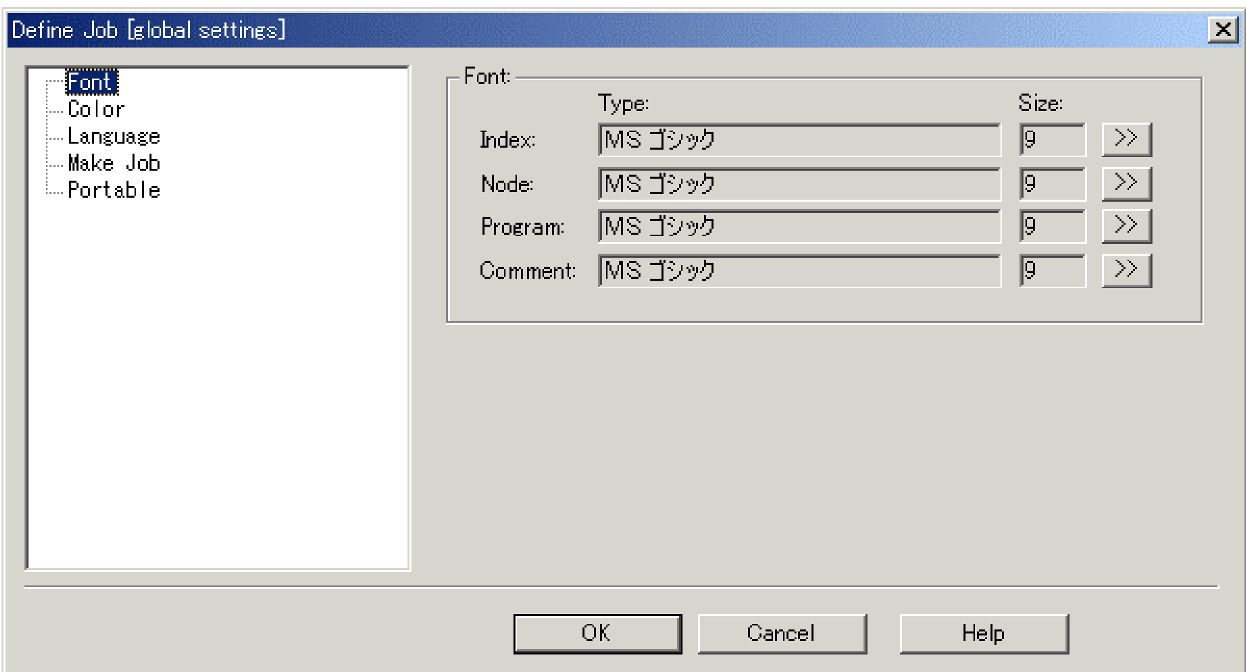
第 5 章 ジョブのプログラミング

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 Edit > Define Job [Default] 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。

Font Property(フォント・プロパティ)

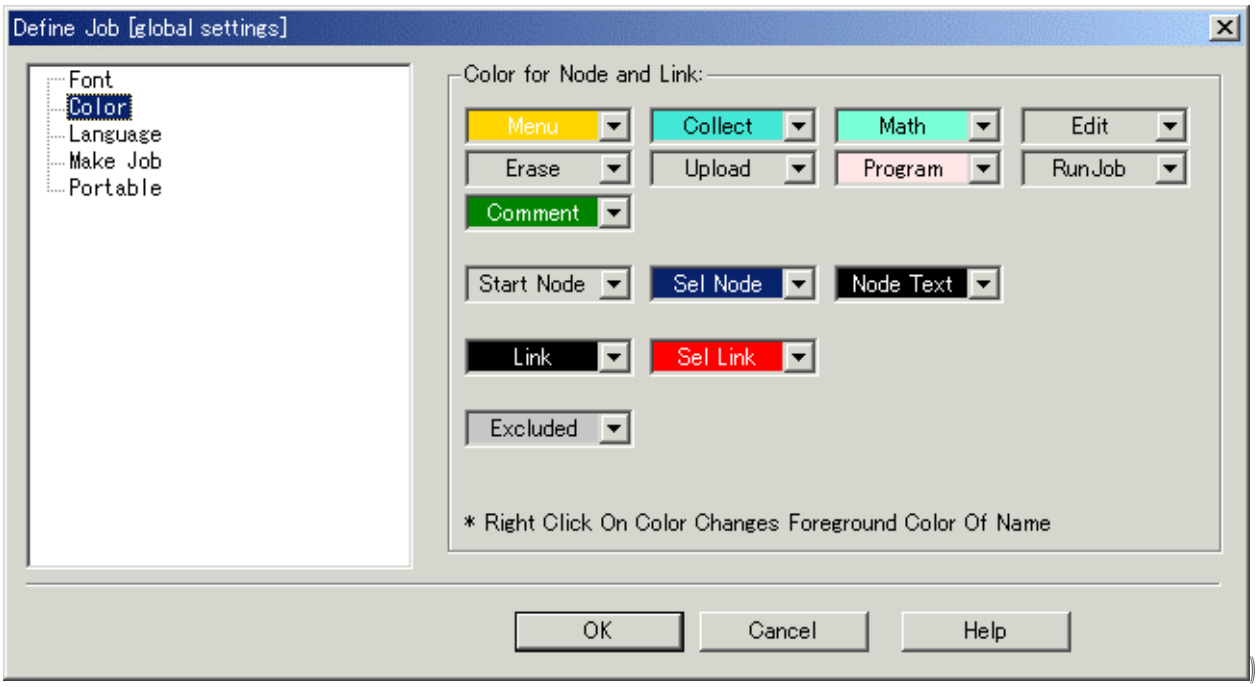


左のプロパティ・ウインドウ、ノード名、プログラム・エディタ、そしてコメントについてフォントを定義します。

---

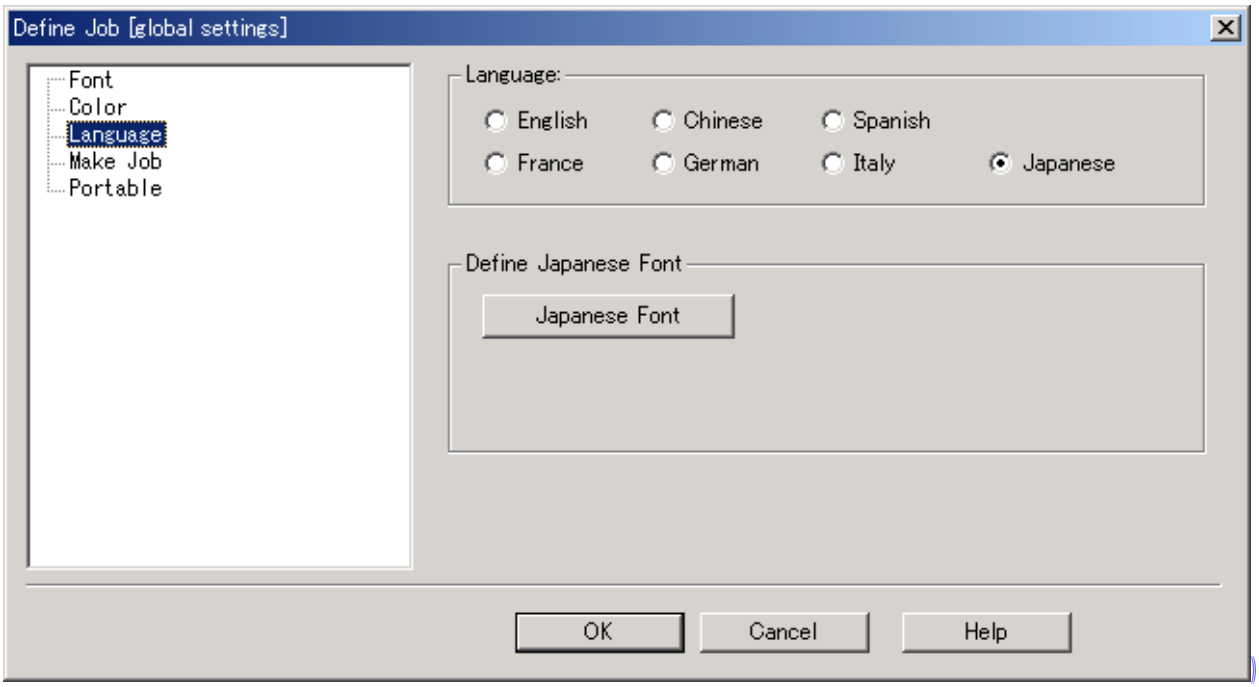
[1] ジョブはデータ収集アプリケーションです。

Color) Property(カラー・プロパティ)



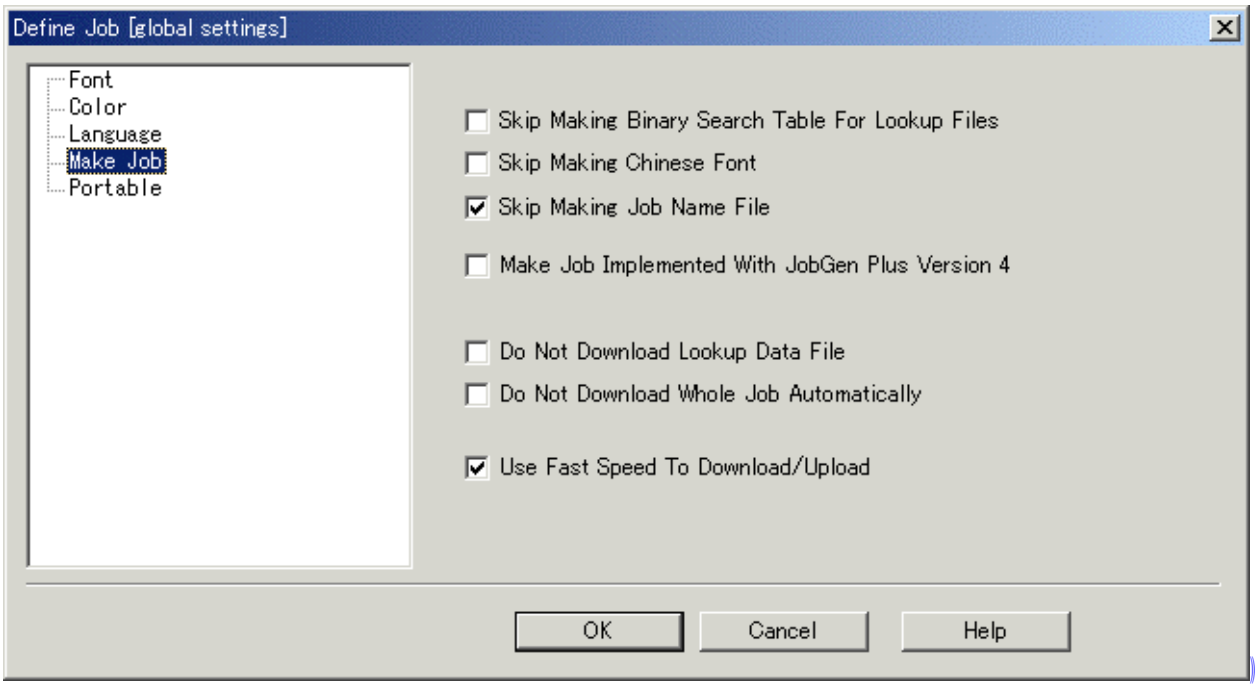
ノード、リンク、そして選択についてのカラーを定義します。表示されているのはノードについてのカラー設定の標準値です。それぞれのノードの色はその基本プロパティ設定で変更することができます。

Language) Property(言語プロパティ)



ジョブの言語を選択します。

Make Job Property(ジョブ作成プロパティ)

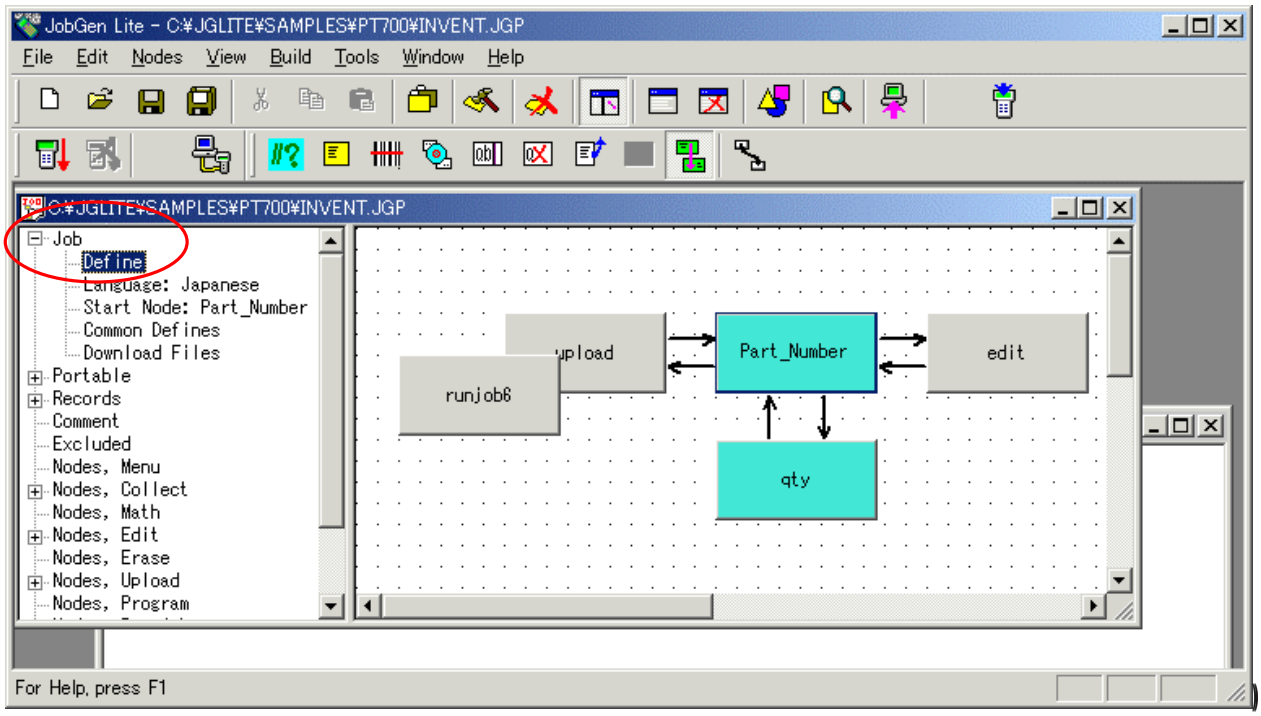


Skip Making Binary Search Table For Lookup Files	ルックアップ・ファイルの内容が変わらない場合、ジョブを作成するたびにコンパイルする必要はありません。これはジョブ作成のプロセスをスピードアップします。
Skip Making Chinese Font.	ルックアップ・ファイルとデータファイルに含まれるすべてのテキストが変わらない場合、そのたびごとに漢字フォントを作る必要はありません。
Skip Making Job Name File	Job Name File(ジョブ・ファィル名)がどのジョブを実行するかを決めるためにジョブ・エンジンに対して使用されます。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、Job Name File は不要になります。
Make Job Implemented With JobGen Plus Version 4	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。
Do Not Download Lookup Data File	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。

Do Not Download Whole Job Automatically	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。
Use Fast Speed To Download/Upload	JobGen Plus は PTComm Manager が最も速い速度で通信するようにします。

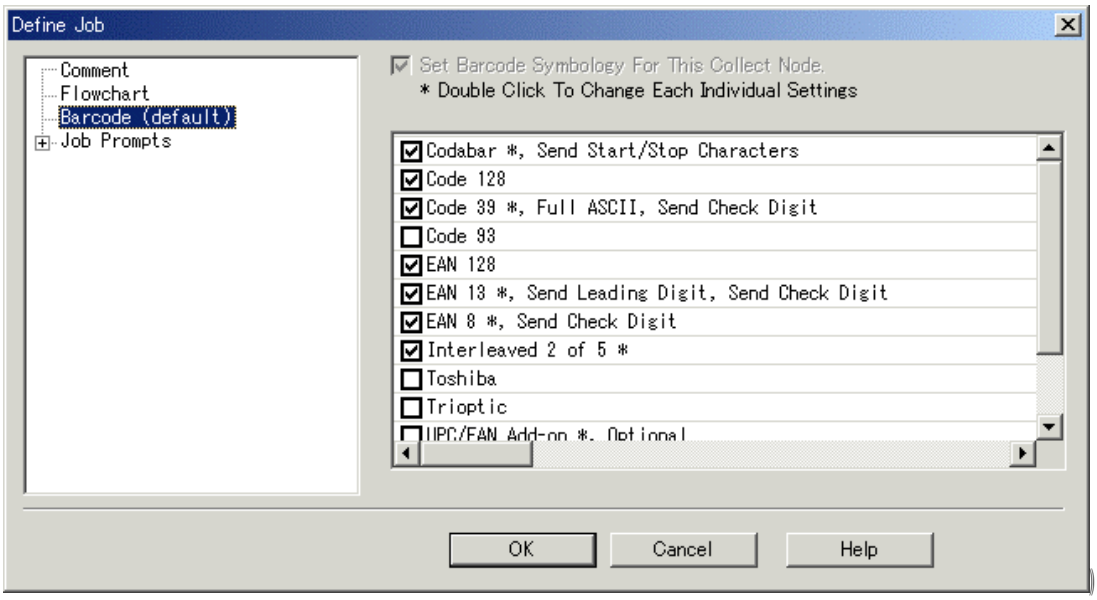
ジョブ設定の標準値を定義

Define Job (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。

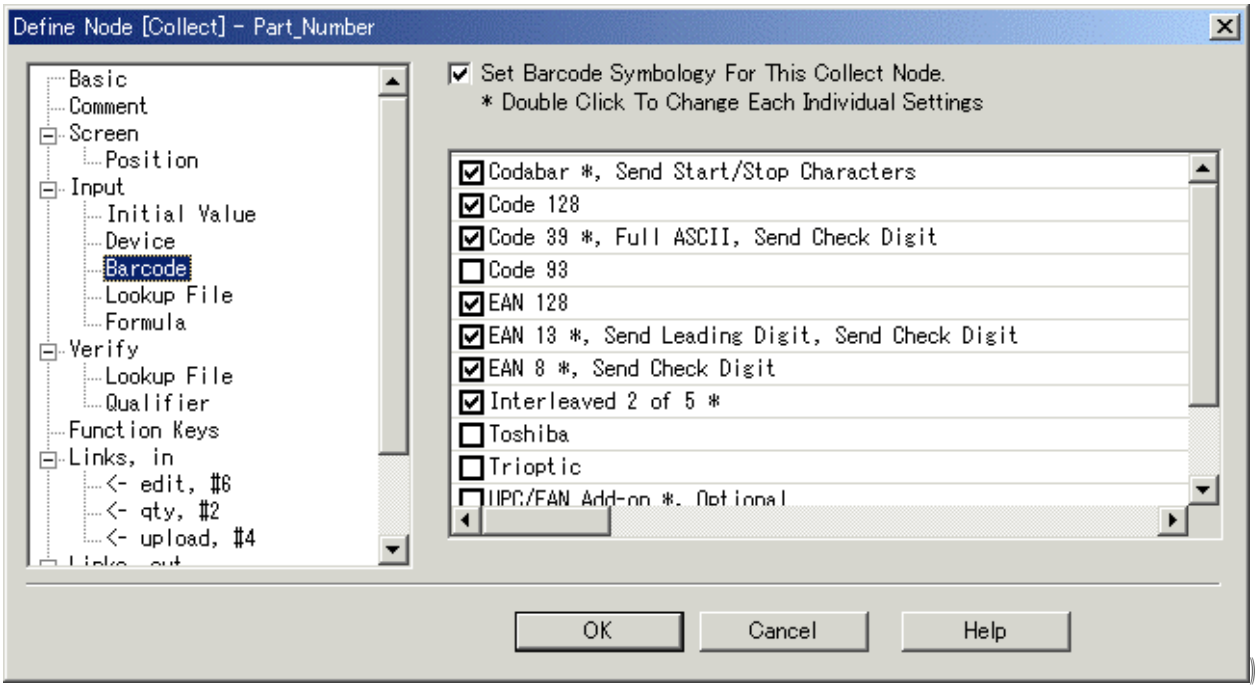


)

Barcode (default) プロパティ

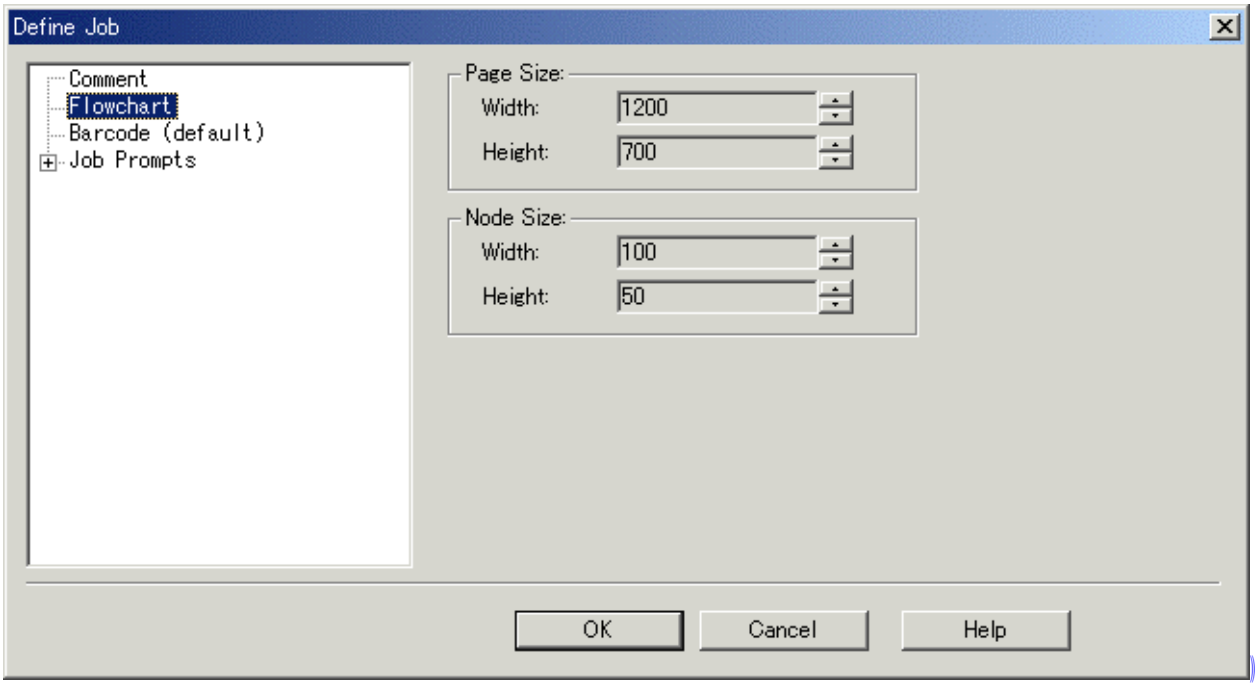


すべての Collect(収集)ノードについて標準のバーコード・シンボル設定を定義します。それぞれの Collect ノード設定は、Input > Barcode プロパティで変更することができます。



)

Flowchart) 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。

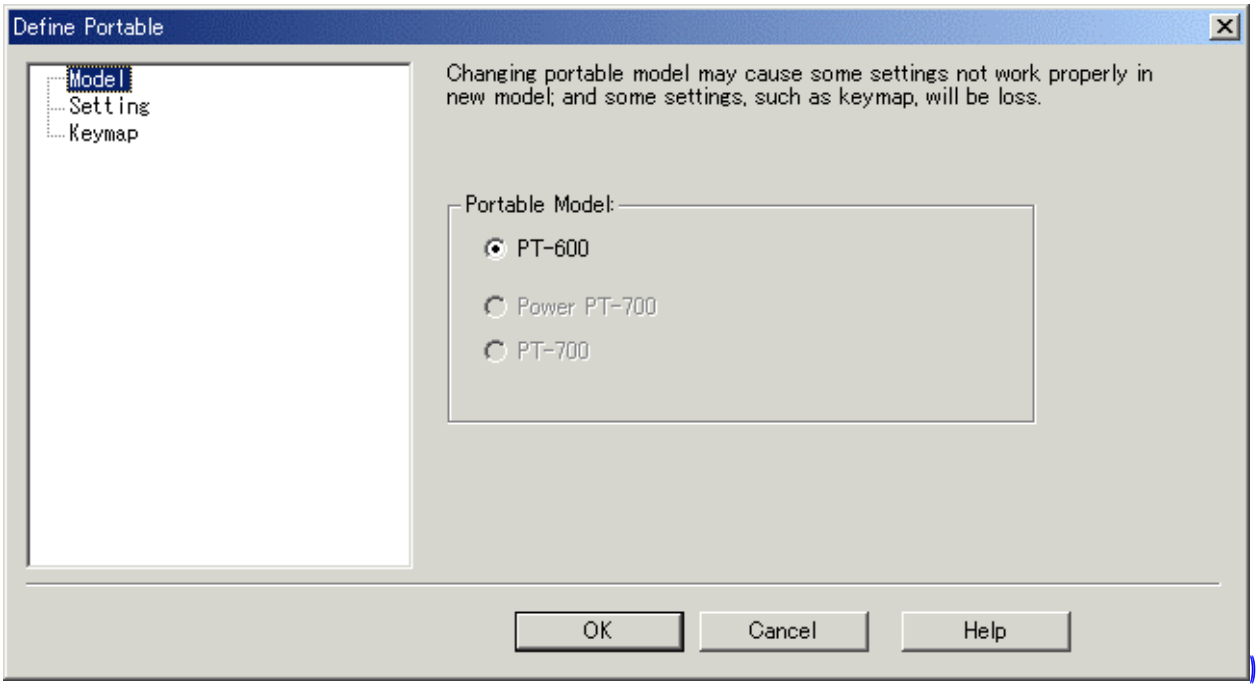
ポータブル・ターミナルの定義

デバイスの選択は、Menu (メニュー) ノード、Collect (収集) ノード、Math (演算) ノード、そして Links (リンク) の表示に影響があるので **JobGen Plus** では最も重要です。Menu ノード、Collect ノードと Math ノードの"Screen:" 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、Link の "Key Input" (キー入力) 確認は、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。

Portable (ポータブル) プロパティ

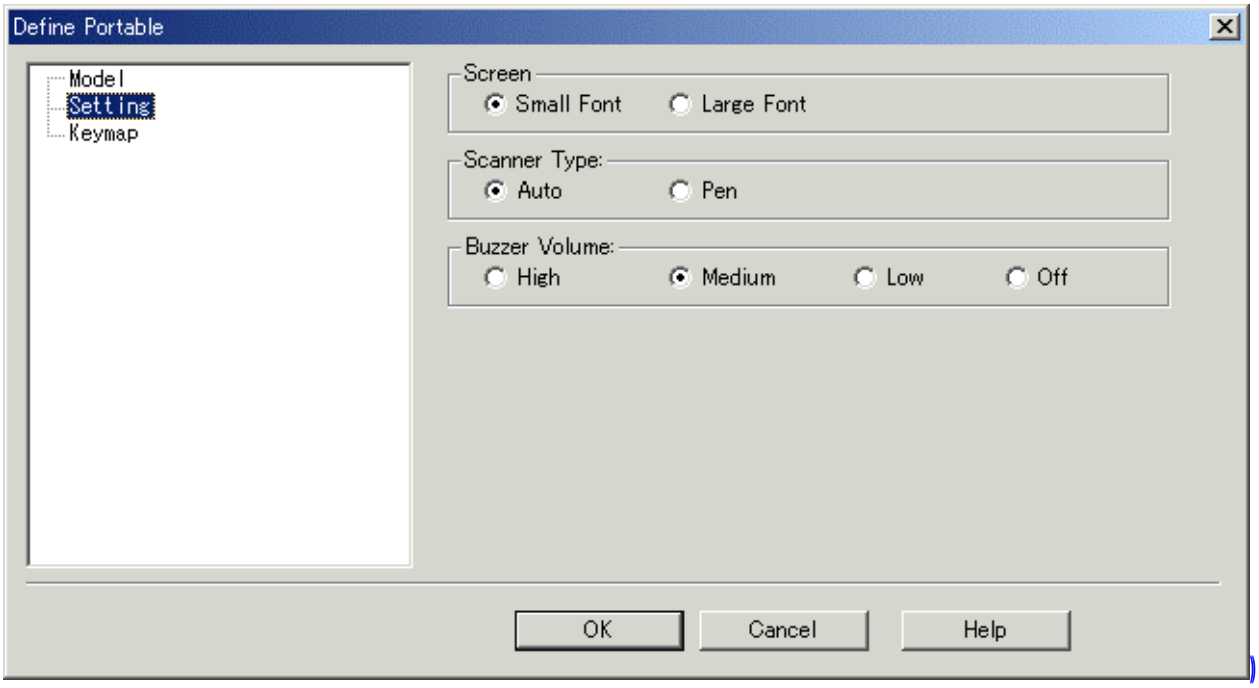
モデルの定義:



Portable Model	例えば、PT600 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-600 はポータブル・ターミナルの標準のモデル・タイプです。

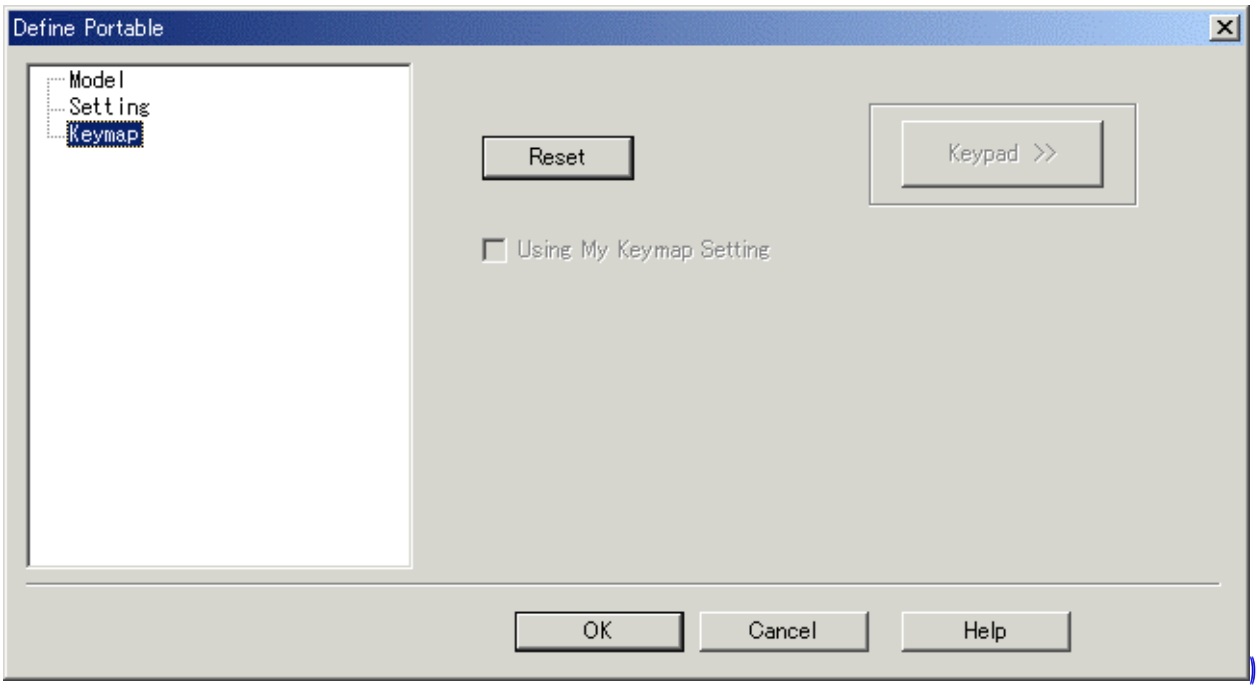
Portable model タイプ設定の変更はジョブのあるプロパティ設定に影響します。例えば、PT-700 は PT600 よりも大きなスクリーンなので、PT-700 から PT600 へ Portable model タイプの変更を行うと、スクリーンのレイアウトはマニュアルで変更する必要があるか、あるいは小さなスクリーンに合うように再設計する必要があります。

Portable Settings(ポータブル・ターミナル設定)の定義



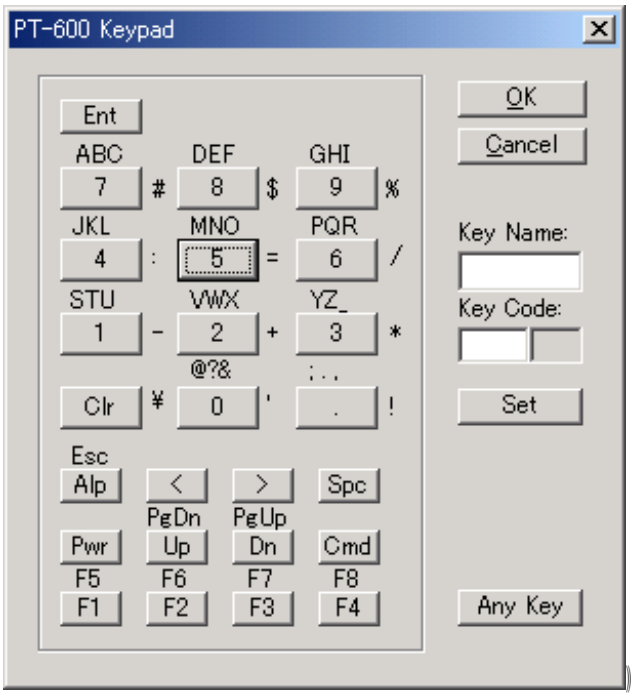
Screen	ディスプレィのフォント・サイズを定義。日本語を表示させる場合は Large Font を選択して下さい。
Scanner Type	スキャナ・タイプの定義。
Buzzer Volume	ブザーの音量を定義。

キーパッド設定の定義:



Reset	ポータブル・ターミナルの標準設定を使用。
Keypad	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。
Use My Keypad Setting	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 "Set" 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 "Use My Keypad Setting" をチェックしなければなりません。

Menu Node (メニュー・ノード) の作成

Menu Node (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: Node (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 Node メニューで Menu 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。

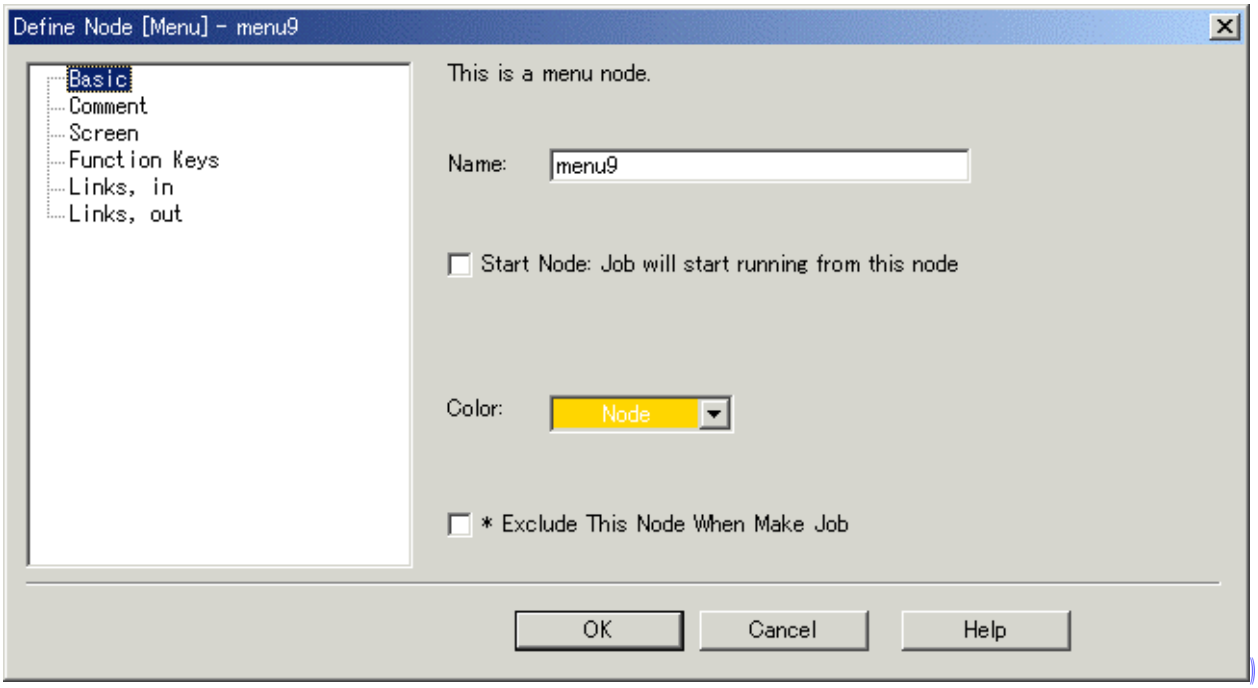
他のノードタイプの作成には、操作は同じです – 別なノードタイプを選択するだけです。

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

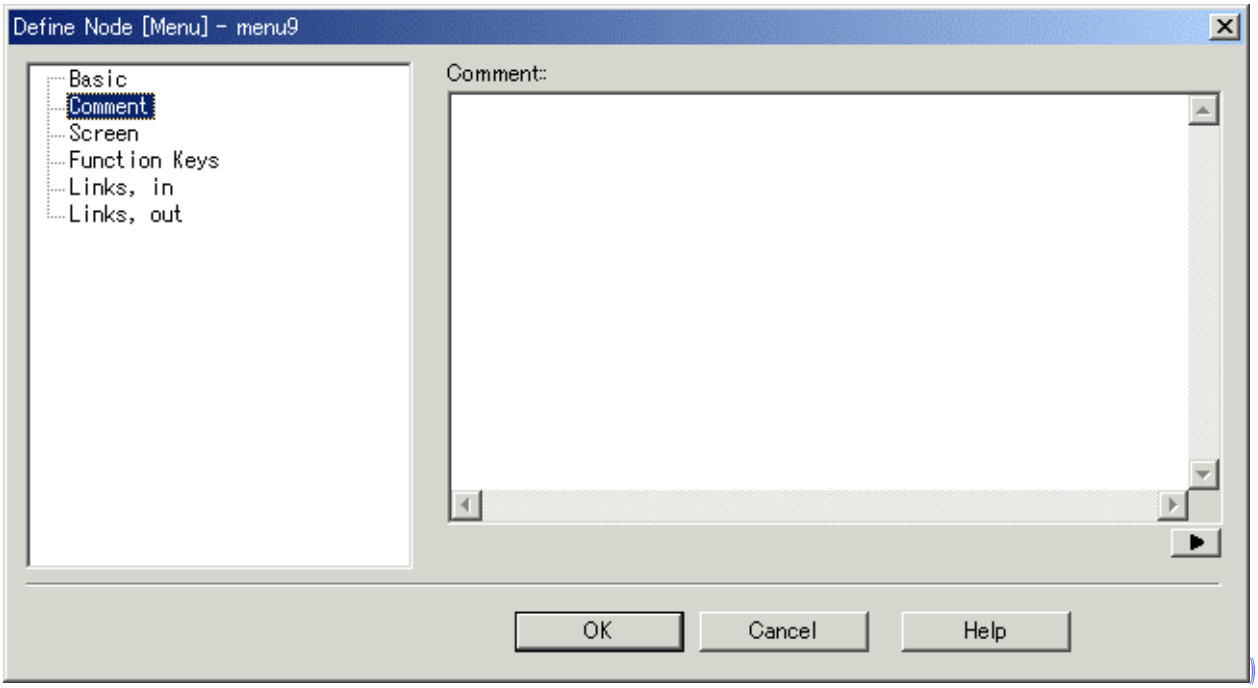


Node Name	ノードの名前。アルファベット、数字と下線文字のみを名前に使用できます。 Collect(収集)または math(演算)ノードの場合、これらは実際にデータを持っているので、ノード名にプリフィックス"_"(下線)を追加し、データにアクセスするために C プログラムによって参照できるようにします。
Start Node	このボックスをチェックすることによって、このノードは開始ノードになります(JobGen Plus アプリケーションの最初のポイント)。 各 JobGen Plus アプリケーションは一つだけの開始ノードを持つことができます。
Color	ノードの色を定義します。

* Exclude This Node When Make Job	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) プロパティ)

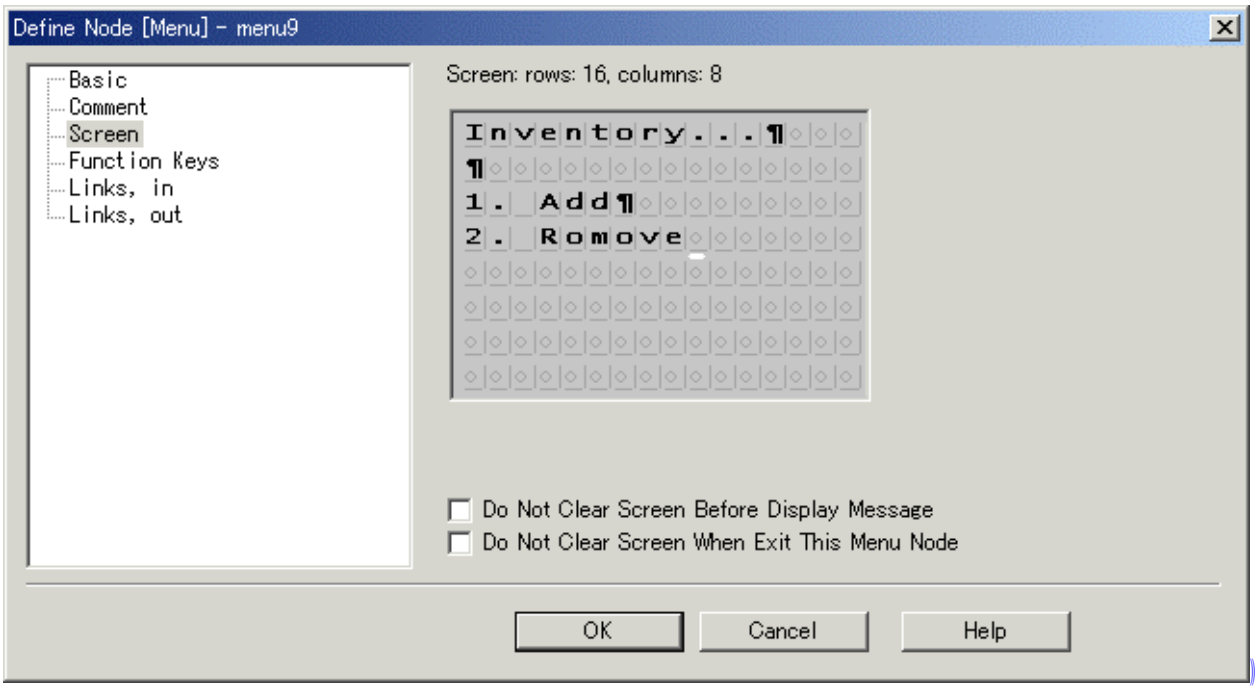


)

Comment Editor Box	コメントを書きます。)

)

Screen プロパティ



画面表示　これは WYTWYS (what you type is what you will see、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。

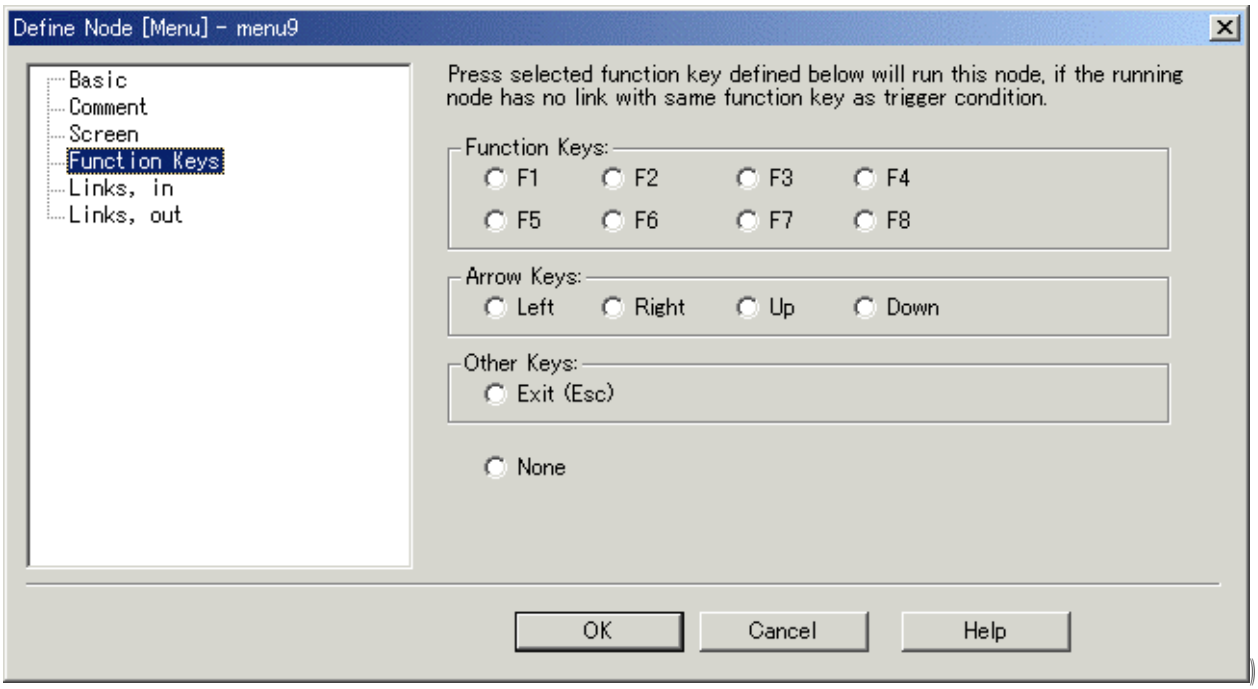
表示したいテキストをウインドウに直接タイプします。

ディスプレィの大きさは、Portable プロパティで定義したフォント設定によって決まります。

Do Not Clear Screen Before Display Message　前のスクリーンからのメッセージを保持します。

Do Not Clear Screen When Exit This Menu Node　次のスクリーンにメッセージを保持します。

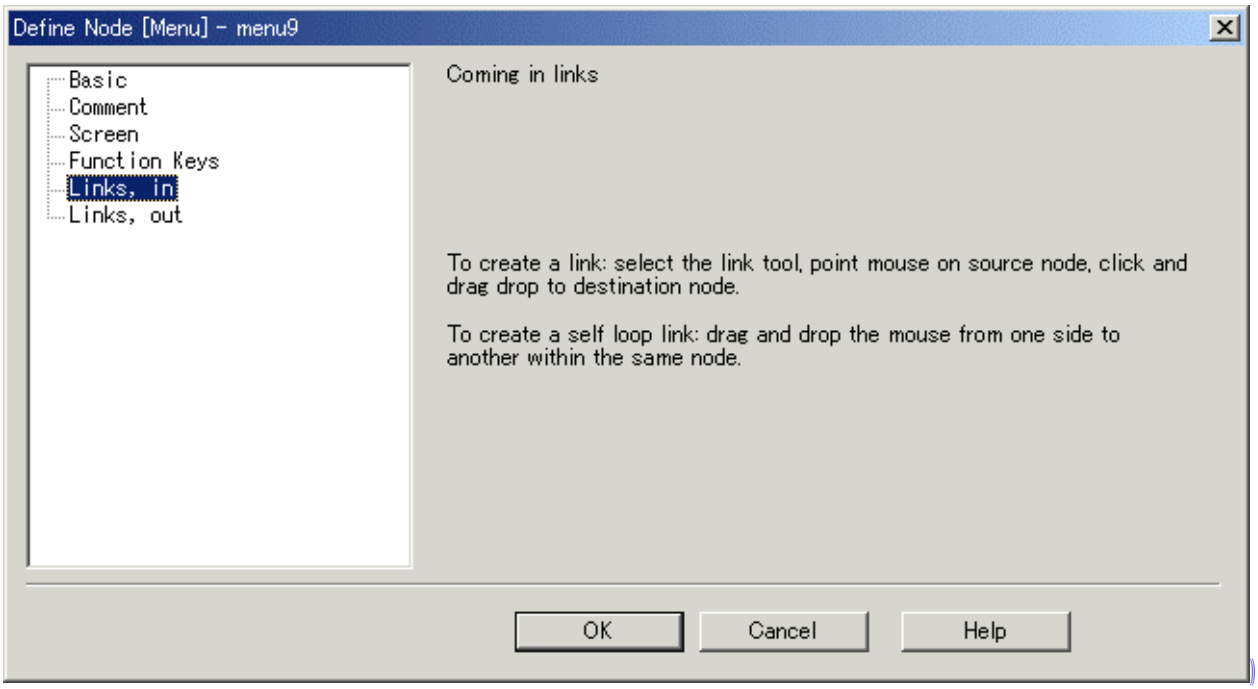
Function) Keys) プロパティ)



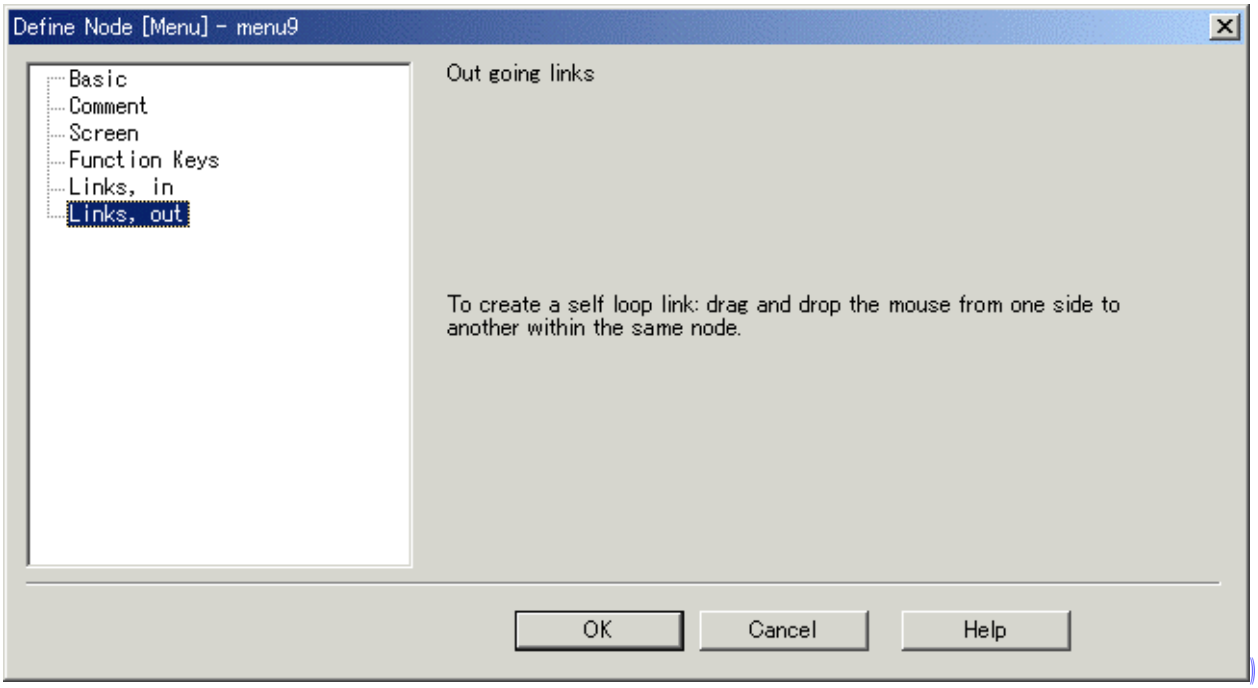
ジョブを実行中に、このノードで定義したファンクション・キーを押すことによりこのノードの実行フローを指示します。しかし、現在のノードがこのファンクション・キーのキー入力状態にリンクされている場合、このリンクはその相手先ノードを実行するために高い優先度を持ちます。

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 プロパティ



Links Out プロパティ



Links In/Out 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 Dialog ウインドウを開くためにリンク上をクリックして下さい。

Collect(収集)ノードの作成

Collect Node 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、Collect Node によって表されます。

Define Collect Node(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

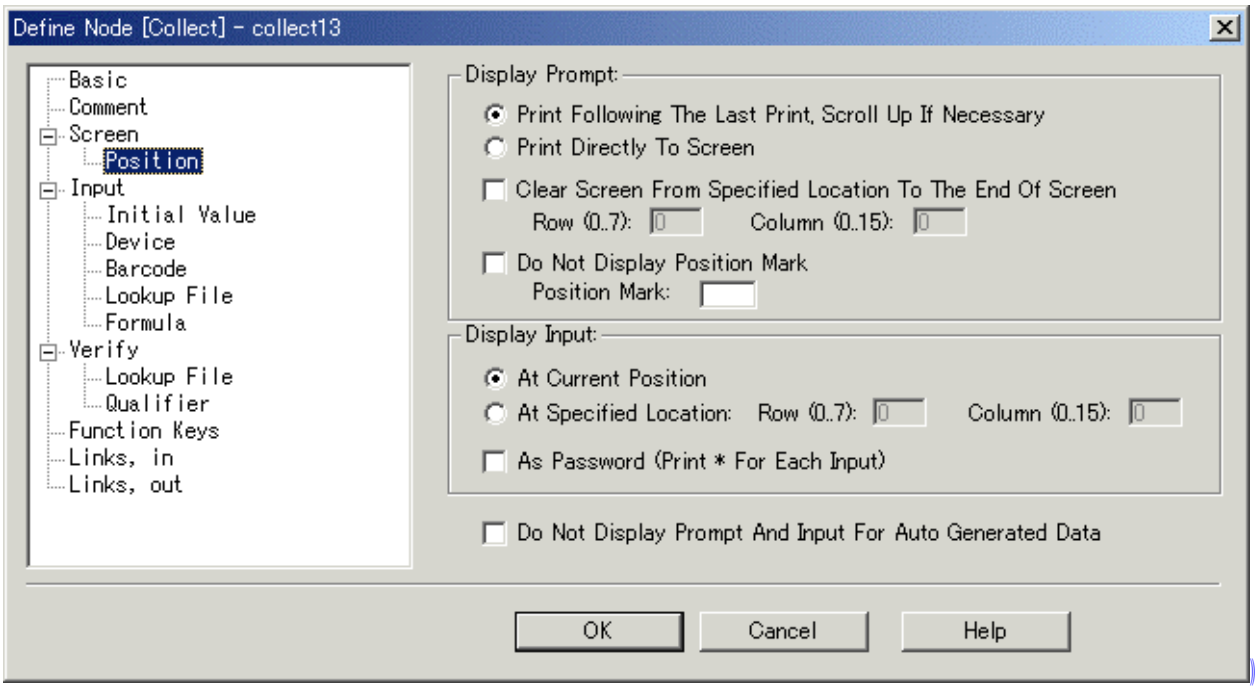
Function Keys プロパティ

Links In プロパティ

Links Out プロパティ

これらのプロパティは Menu(メニュー)ノードと同じです。

Screen >> Position プロパティ



Display Prompt(表示プロンプト)

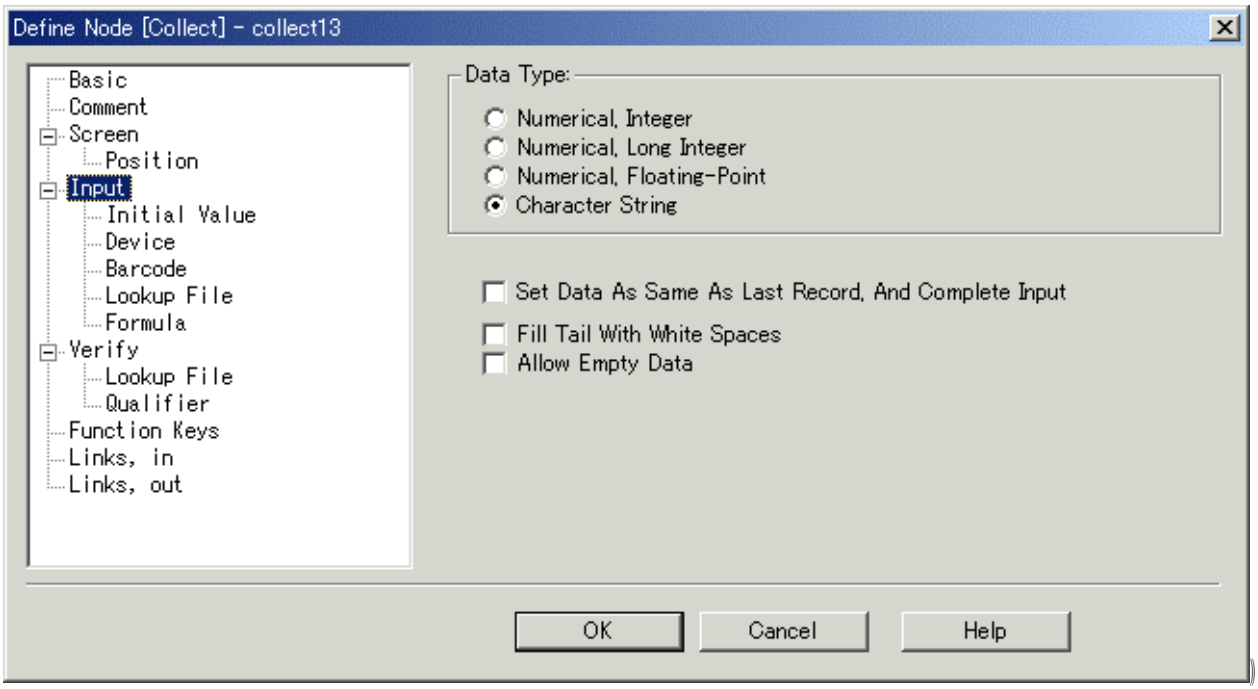
Print Following The Last Print, - Scroll Up If Necessary	現在のカーソル位置で行ごとに表示し、そしてスクリーンをスクロールアップします。
Print Directly To Screen	スクリーンに表示されたテキストをポータブル・ターミナルのスクリーンに表示します。すべてのテキストはその指定された位置になければなりません。
Clear Screen From Specified Location To The End Of Screen	表示する前にスクリーンの一部または全部をクリアします。 row と column ボックスで行と欄を指定します。
Do Not Display Position Mark	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。
Position Mark	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。

Display) Input(表示入力))

At Current Position	現在のカーソル位置で表示します。)
At Specified Location	Row(行)と column(欄)で指定された位置で表示します。) row)と)column)ボックスで行と欄を指定します。)
As Password	*)を表示することによって入力を隠します。)
Do Not Display Prompt And Input For Auto Generated data	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



Data Type	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 Integer、long integer と floating-point 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制限値と小数点以下の数があるかどうかが異なっています。 long integer 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。) floating-point 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。
Set Data As Same As Last Record, And Complete Input	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。)

Fill Tail With White Spaces	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。》
Allow Empty Data	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。》

》 》

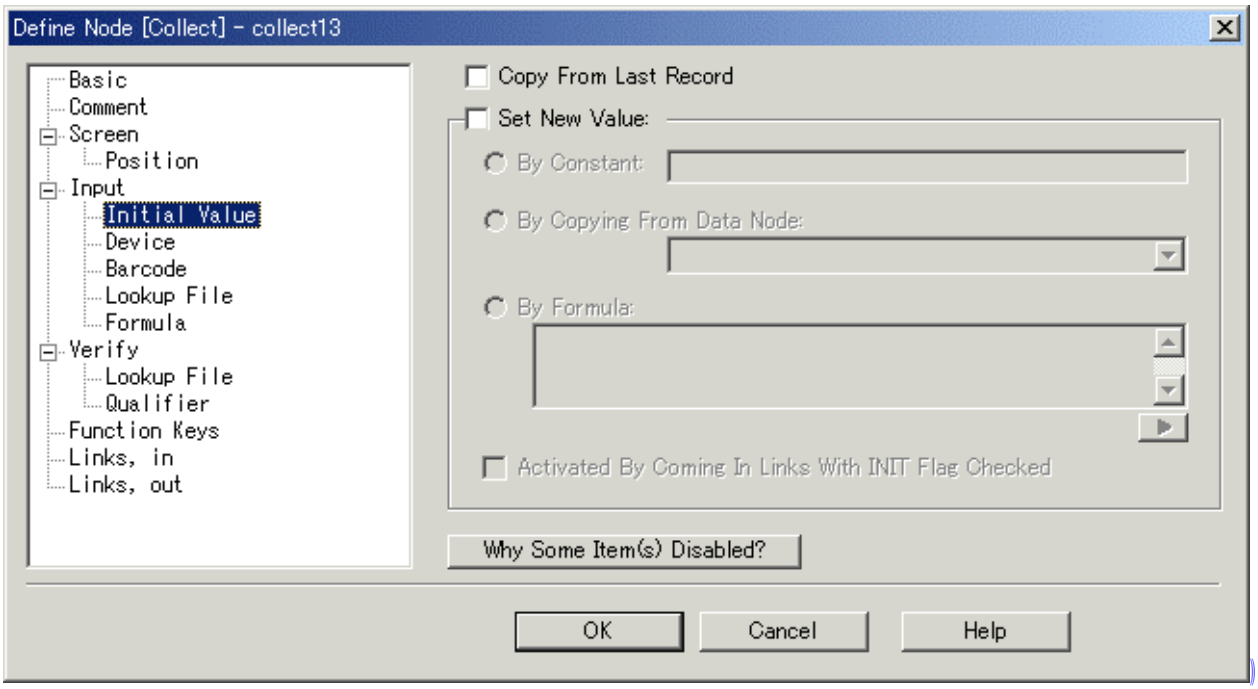
》

Integer》(整数)の制限値》

Integer:	-32767	から 32767
Long integer:	-2147483647	から 2147483647
Floating-point:	1.175494351e-38F	から 3.402823466e+38F

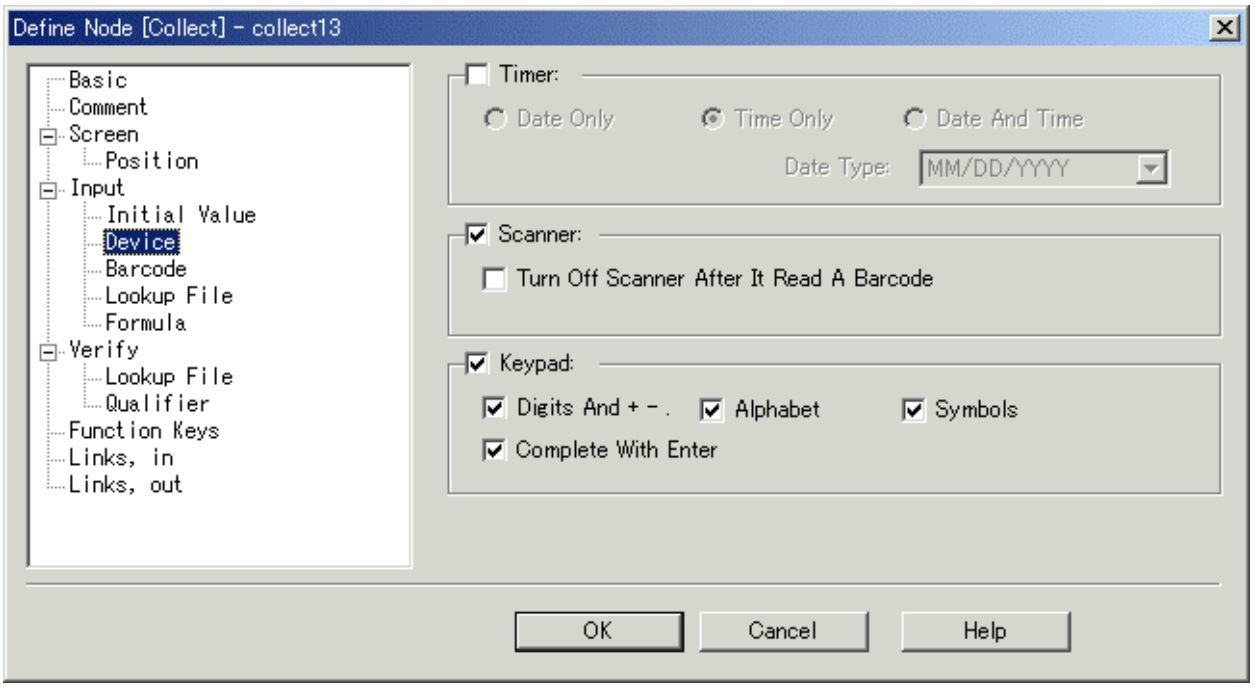
》

Input > Initial Value プロパティ



Copy From Last Record	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。
Set New Value	新しい値をセットします。
By Constant	データに定数を指定します。
By Copying From Data Node	他のデータノード(collect ノードまたは math ノード)からデータをコピーします。数値タイプの collect 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。
By Formula	値を指定するのに表現式を使用。
Activated By Coming In Links With INIT Flag Checked	Set New Value(新しい値をセット)の初期値は Coming In Link with INIT 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。Timer(時計)、) Scanner(スキャナ)) そして Keypad(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。)

- **Timer**

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) date) (MM/DD/YYYY)、) time) (HH:MM:SS)) と) date-time) (MM/DD/YYYY) HH:MM:SS)です。)

- **Scanner**

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 collect(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 Collect(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 Job) Define,) Barcode) (default)) プロパティ・ページで定義されます。.)

注意:) データバッファが空でない場合(例、文字がキーパッドから入力される)、新しくスキャンするデータは排除されます。すべての入力をクリアしてスキャンを再度行うには) Clear) ボタンを押して下さい。)

- **Turn Off Scanner After It Reads A Barcode(バーコード読み込み後スキャナをオフ)**

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。collect(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 collect ノードでバーコードを読んだ後 ON 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 collect 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、ON 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 collect(収集)ノードあるいは math(演算)ノード、または program(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。

- **Keypad**

入力のためにキーパッドを使用可能にします。

- **Digits And + - .**

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。

- **Alphabet**

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。

- **Symbols**

$, #, ! , 等の記号だけがキー入力として受け入れられます。

- **Complete With Enter**

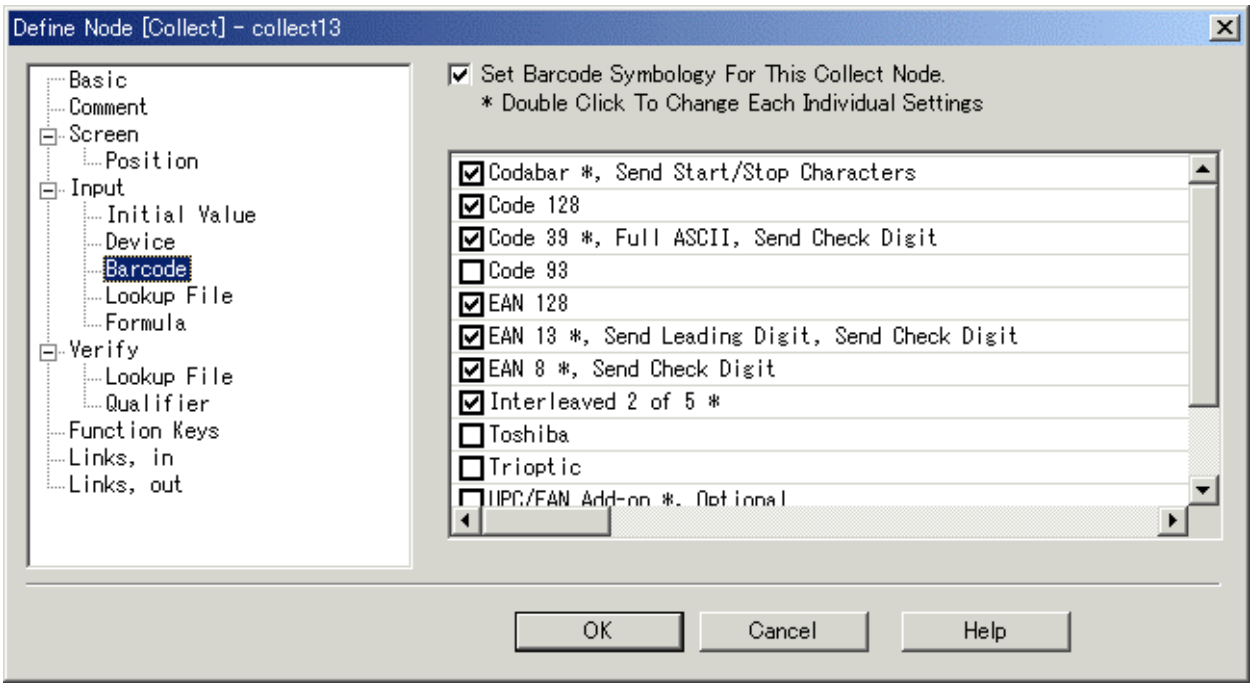
入力を終了するためには、Enter キーを押さなければなりません。これをチェックしない場合、そして入力が最大長に達したときに自動的に終了します。

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 keymap (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)



)

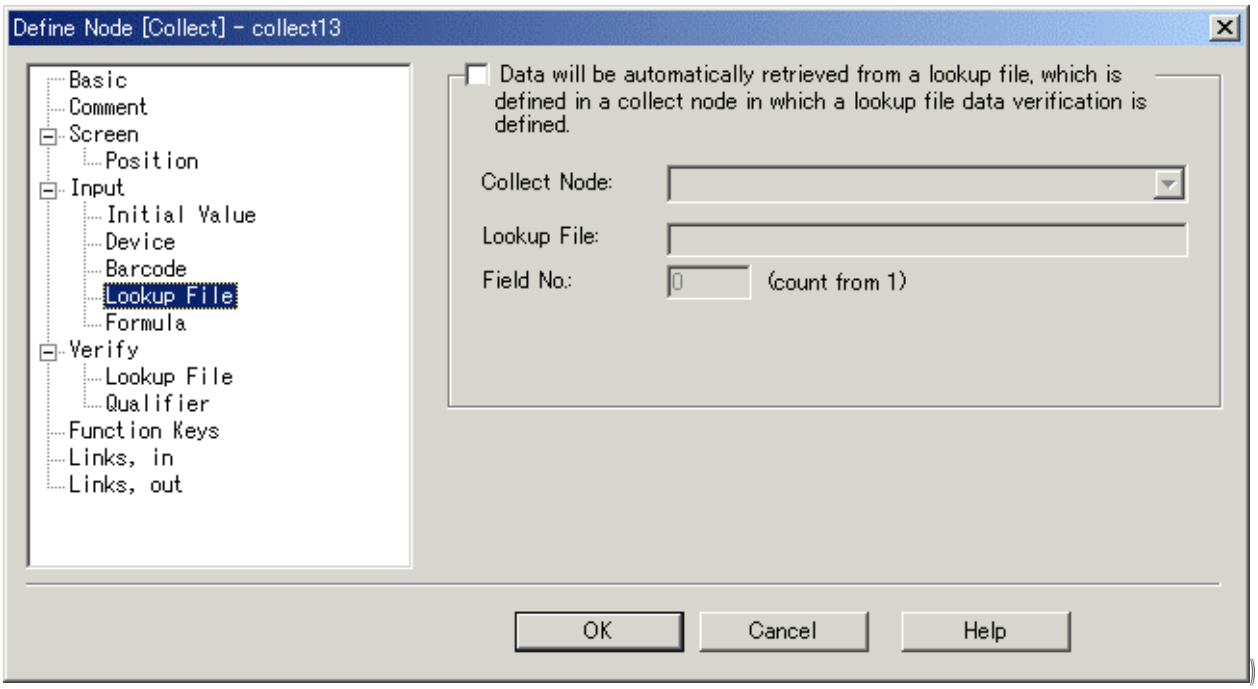
選択された場合、それぞれの collect) (収集) ノードは、それ自身のバーコード・シンボル設定を持っているので、別な) collect (収集) ノードは別なスキャナの動作をします。))

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)



)

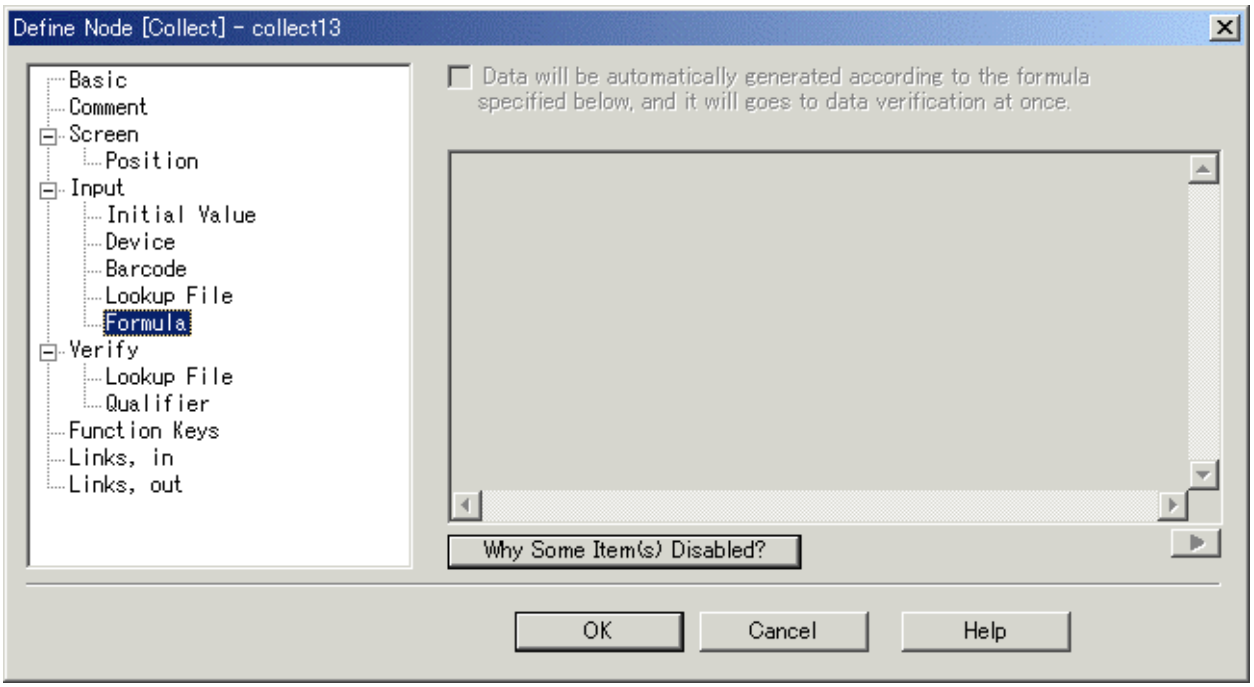
多くの場合、ルックアップ検証における以前の成功を元にしたルックアップ・ファイルからデータを検索する必要があり、関連するデータをベースにしてデータを処理することができます。)

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 collect(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。))

collect(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。)

)

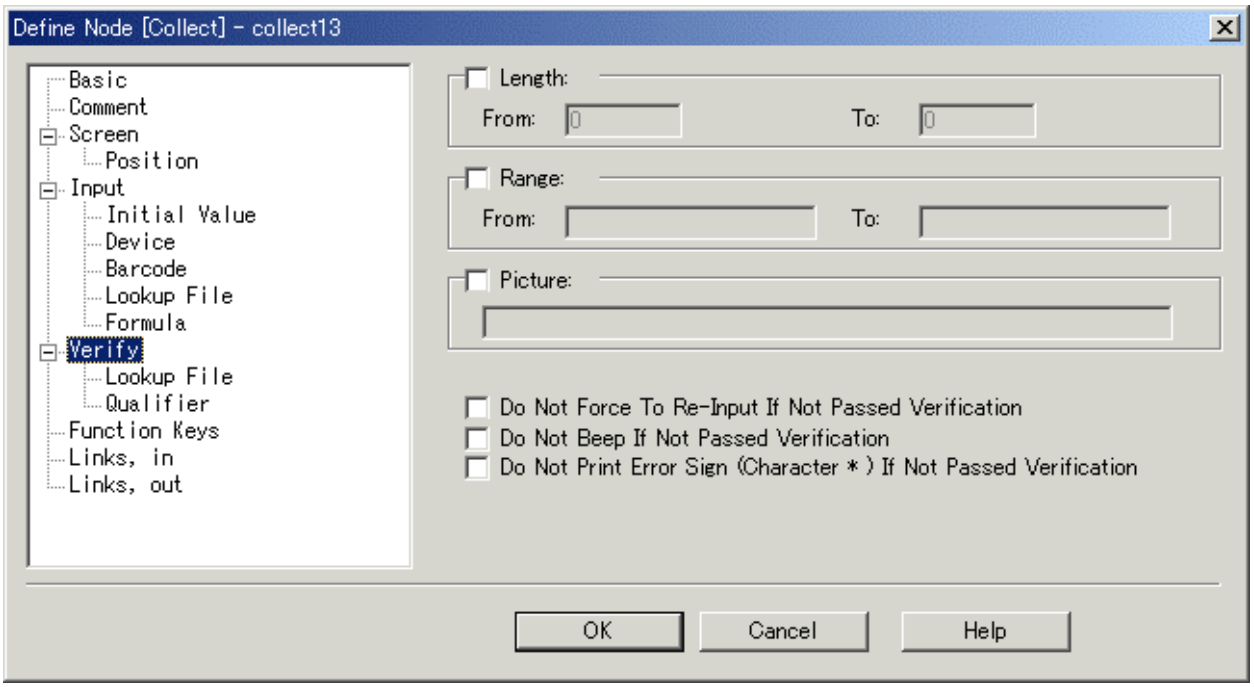
Input) >) Formula) プロパティ)



データを生成するために表現式として) C) の表現を指定して下さい。)

Collect(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単に使用します。)

Verify プロパティ



Length	入力データの長さ(文字数)を指定します。
Range	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。
Picture	入力文字タイプのマスクパターンを定義します。
Do Not Force To Re-Input If Not Passed Verification	普通は、基本的な検証、長さ(Length)、範囲(Range)とピクチャ(picture)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、あるなら、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。
Do Not Beep If Not Passed Verification	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。
Do Not Print Error Sign (*) If Not Passed Verification	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。

Picture Pattern(ピクチャ・パターン)

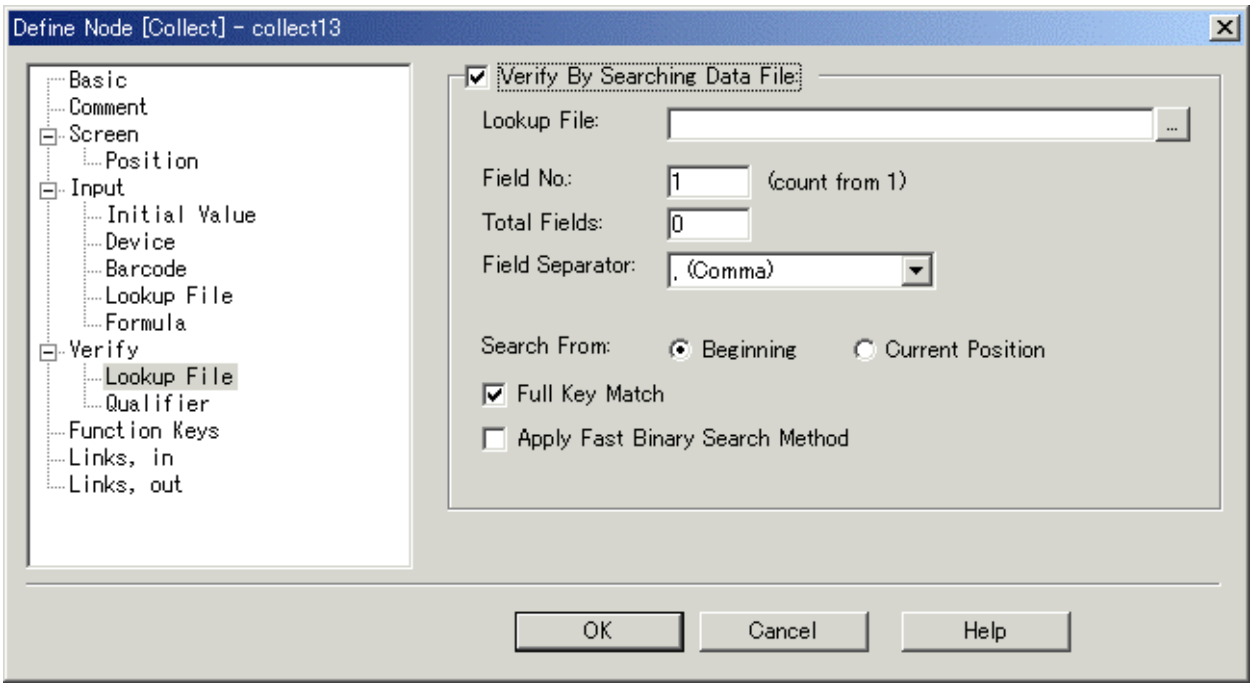
a	英文字 (A-Z, a-z) のみ可
n	数字(0-9) と ‘+’, ’-‘, ‘.’ (ドット), ‘E’ と ‘e’のみ
0	数度 (0-9) と ‘+’, ’-‘ のみ
p	数字 (0-9)のみ
l	小文字 (a-z)のみ
u	大文字 (A-Z)のみ
x	印刷可能な文字 (0x32--0x7f)
z	全 ASCII コード
_	全 ASCII コード
#	文字を削除
@	@ と @間の文字列
%	インジケータ、パターンにシンボルを追加
?	インジケータ、非固定長のパターン
!	インジケータ、検証結果を逆に
(space)	パターンのターミネータ

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) 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。))

Field) No.) 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) success) flag(成功フラグ)) が上がります。) そうでなければ、) fail) flag(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。) Define) Job) [default) settings]) 、) Make) Job) 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。)

- **Search From(検索)**

最初から、もしくは最後に一致したレコードの後のレコード、のいずれかからデータファイルを検索します。

- **Full Key Match(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。

- **Apply Fast Binary Search Method(最も速いバイナリ検索法を適用)**

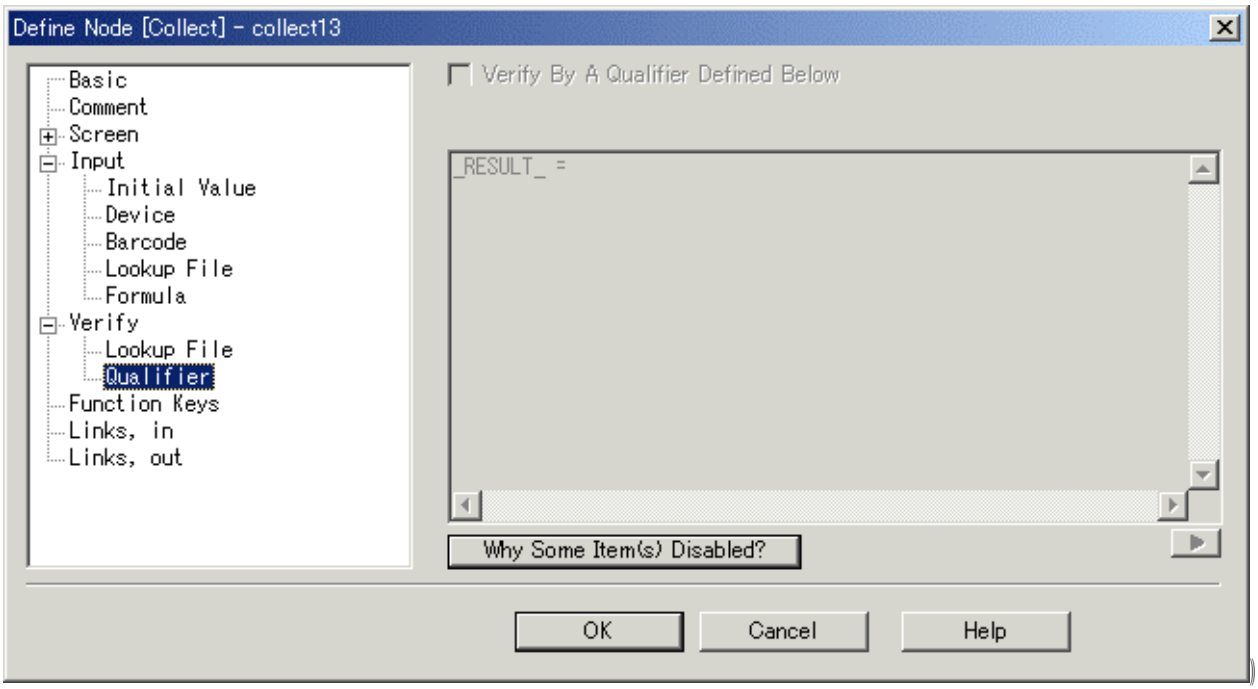
10 万件のデータを含むような大きなデータベースを検索する場合、高速検索を可能にするためにこのオプションをチェックして下さい。JobGen Plusは自動的にファイル–バイナリ検索テーブル-を追加し、そして元のデータファイルとジョブ・ファイルと共にダウンロードします。

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 プロパティ



これは C の表現式です。事前定義変数 _RESULT_ に値がゼロでなければ true、そうでなければ false 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 success flag(成功フラグ)が上がります。そうでなければ、fail flag(異常フラグ)が上がります。

Math(演算)ノードの作成

Math(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。Collect(収集)ノードと Math(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

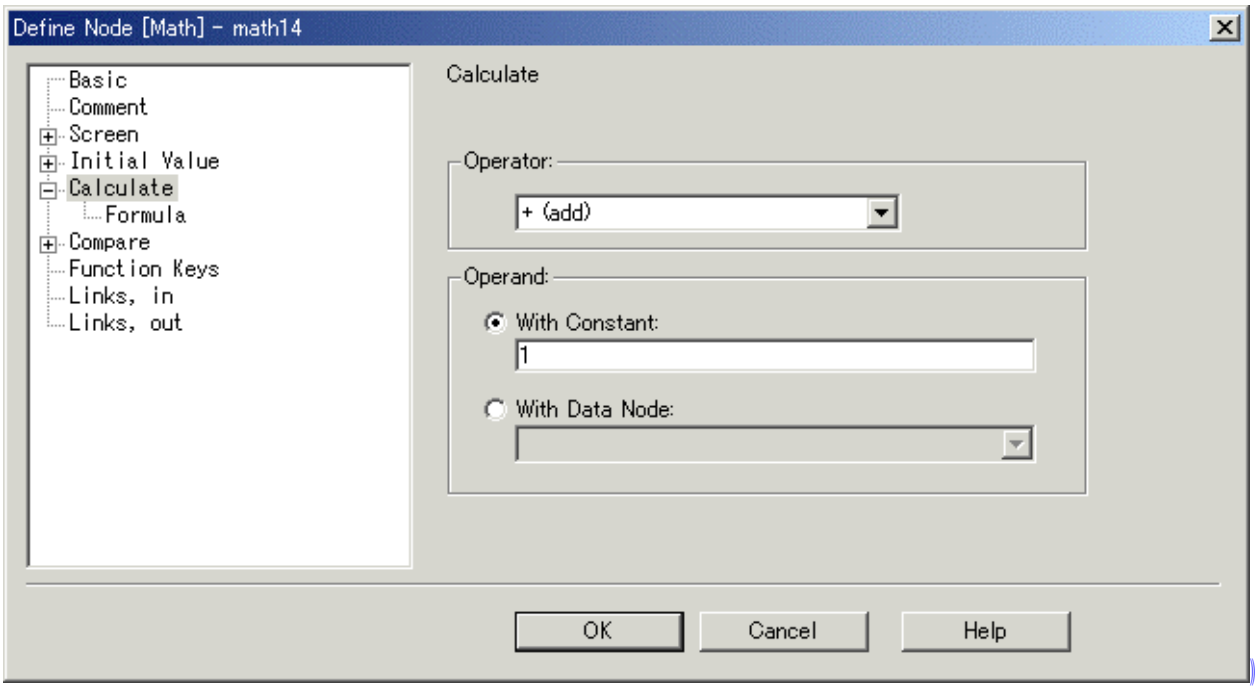
Links Out プロパティ

これらのプロパティは collect(収集)ノードのものと同じです。

Initial Value > Data Type プロパティ

Math(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。

- **Operators:(演算子)**

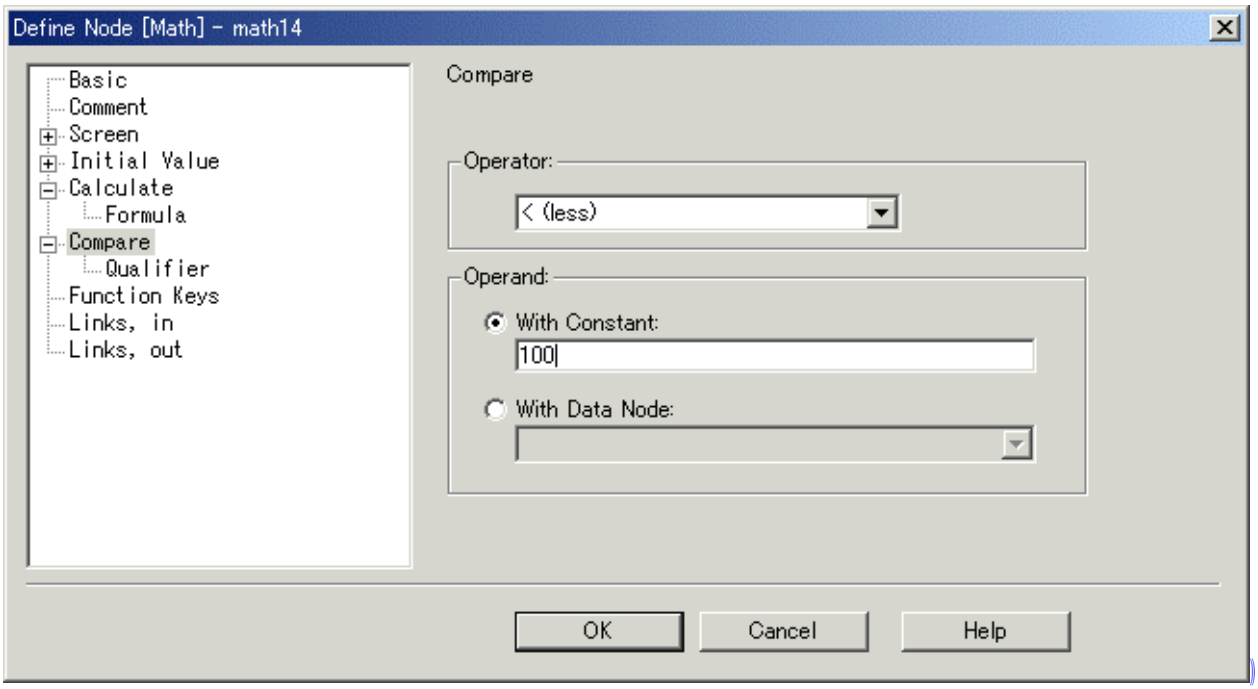
  Add, Minus, Multiple, Divide, と None.

- **Operands: (オペランド)**
  - **With Constant:**

    オペランドとして定数を指定。
  - **With Data Node:**

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

Compare Property.



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 Success(成功)または Fail(失敗)状態として使用することができます。

- **Operators: (演算子)**

  Less, Less or Equal, Greater, Greater or Equal, Equal, Not Equal, と None。

- **Operands: (オペランド)**
  - **With Constant:**

    オペランドとして定数を指定。
  - **With Data Node:**

    データノード、これはCollect(収集)ノードまたはMath(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Collect(収集)ノードだけをここで選択することができます。

Edit(編集)ノードの作成

Edit Node(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。

Edit ノードの定義

Basic プロパティ

Comment プロパティ

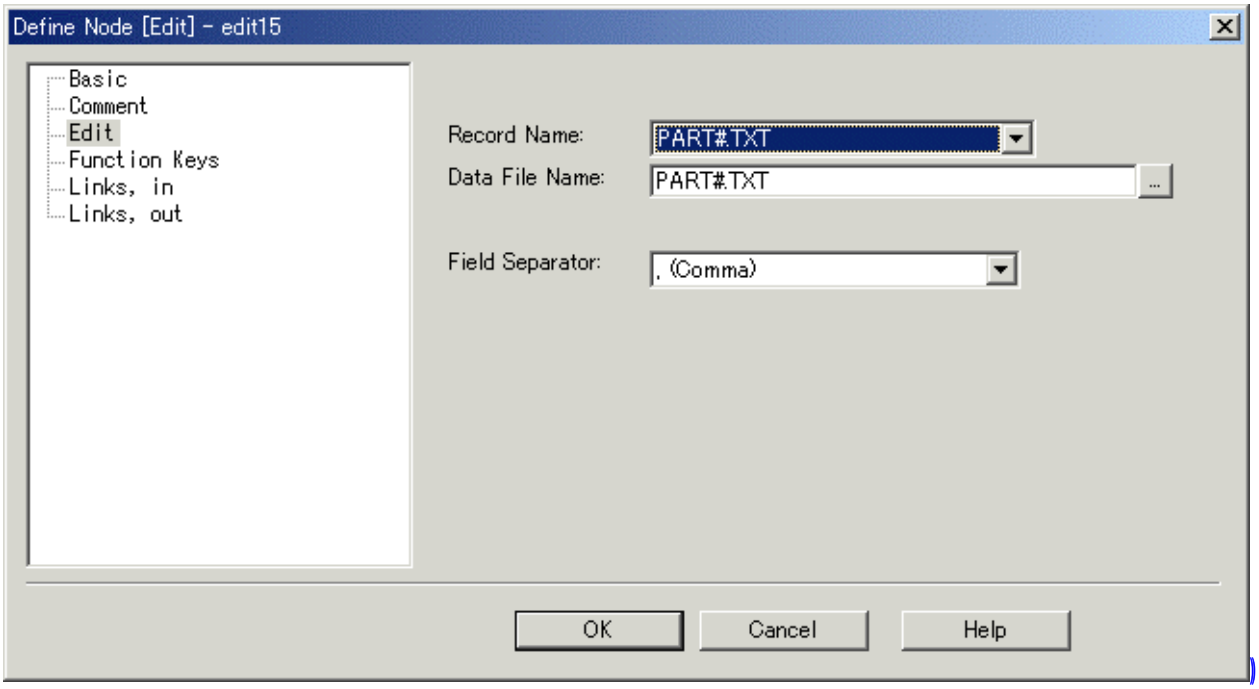
Function Keys プロパティ

Links In プロパティ

Links Out プロパティ

これらのプロパティは menu(メニュー)ノードのものと同じです。

Edit プロパティ



Record Name	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。
Data File Name	実行時に edit(編集)ノードで操作するファイルを定義します。
Field Separator	データファイルで使用されるフィールド・セパレータ(デリミタ、分離記号)を指定します。

Edit(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**Browse mode(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。欄(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。

Browse(表示)モードで、**Edit(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 Editモードを表します。Edit(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。Edit(編集)ノードの選択はBrowse(表示)モードに戻ります。

データを変更中に、左矢印キーはbackspace(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。

Search(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は欄番号を表します。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Browse(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EXITを押して下さい。

Edit(編集)ノードから終了するためにはBrowseモードで**Exit**キーを押します。

Erase(消去)ノードの作成

Erase Node(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。

Define Erase Node

Basic プロパティ

Comment プロパティ

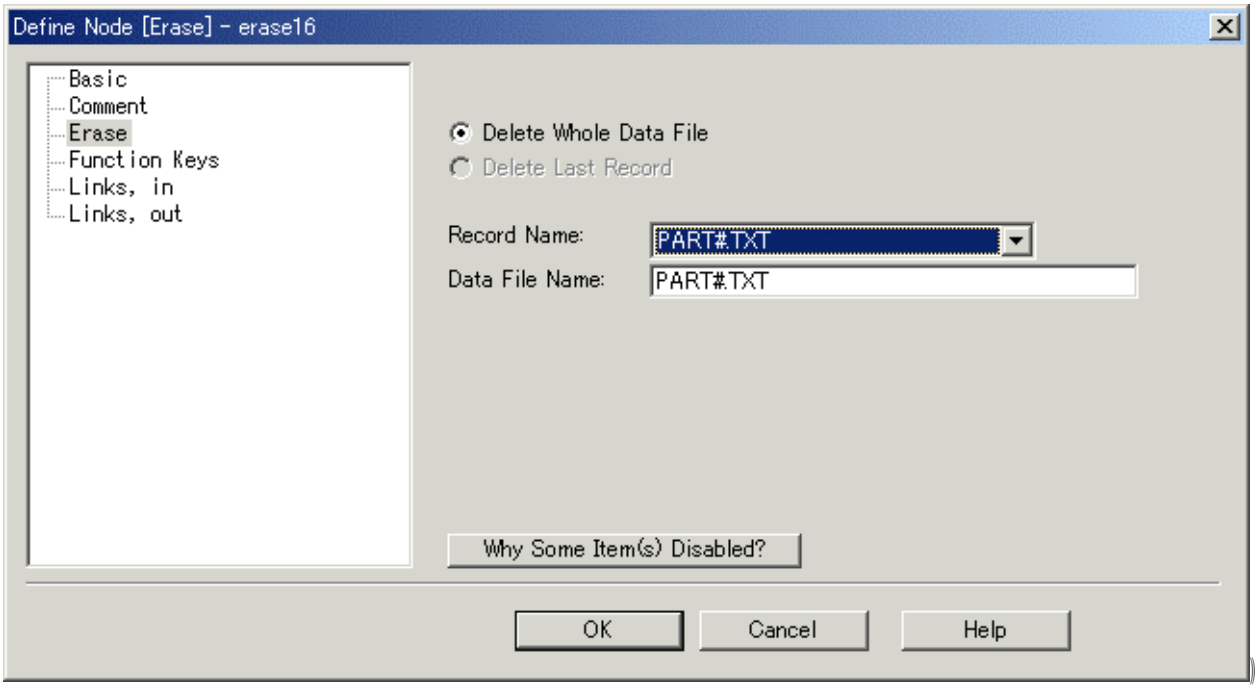
Function Keys プロパティ

Links In プロパティ

Links Out プロパティ

これらのプロパティは Menu(メニュー)ノードのものと同じです。

Erase プロパティ



- Delete Whole Data File　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。
- Delete Last Record　データファイルから最後のレコードを削除します。
- Record Name　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。

Data File Name	ファイルを定義します。これは実行時に erase(消去)ノードで操作されます。

Upload(アップロード)) ノードの作成)

Upload(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。)

)

Define) Upload) Node)

Basic) プロパティ)

Comment) プロパティ)

Function) Keys) プロパティ)

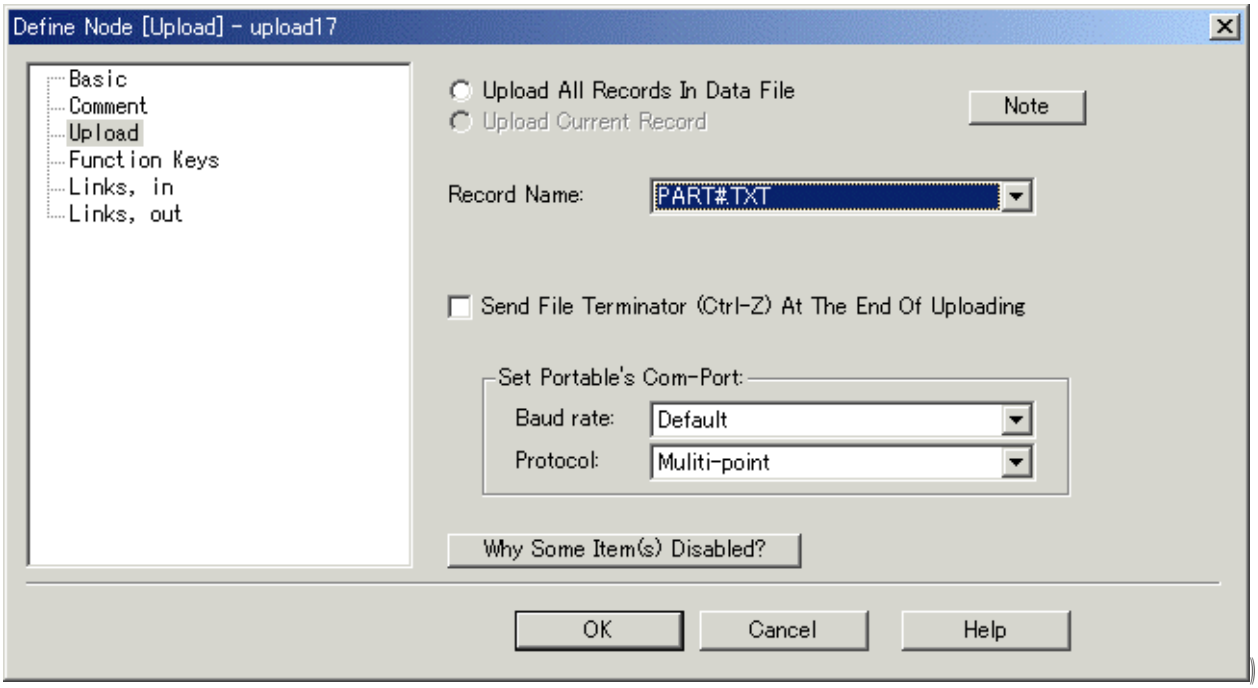
Links) In) プロパティ)

Links) Out) プロパティ)

これらのプロパティは menu(メニュー)モードのものと同じです。)

)

Upload) プロパティ)



)

Upload(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。) Upload(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。)

))

)

Upload All Records In Data File	ファイル中のすべてのレコードをアップロード。
Upload Current Record	現在のレコードをアップロード。
Record Name	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。
Send File Terminator At The End Of Uploading	アップロードの最後でファイル・ターミネータ(Ctrl-Z)を送信。

Program(プログラム)ノードの作成)

Program) Node(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。))

)

Program(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 menu(メニュー)ノードのものと同じです。)

)

Basic) プロパティ)

Program(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。))

)

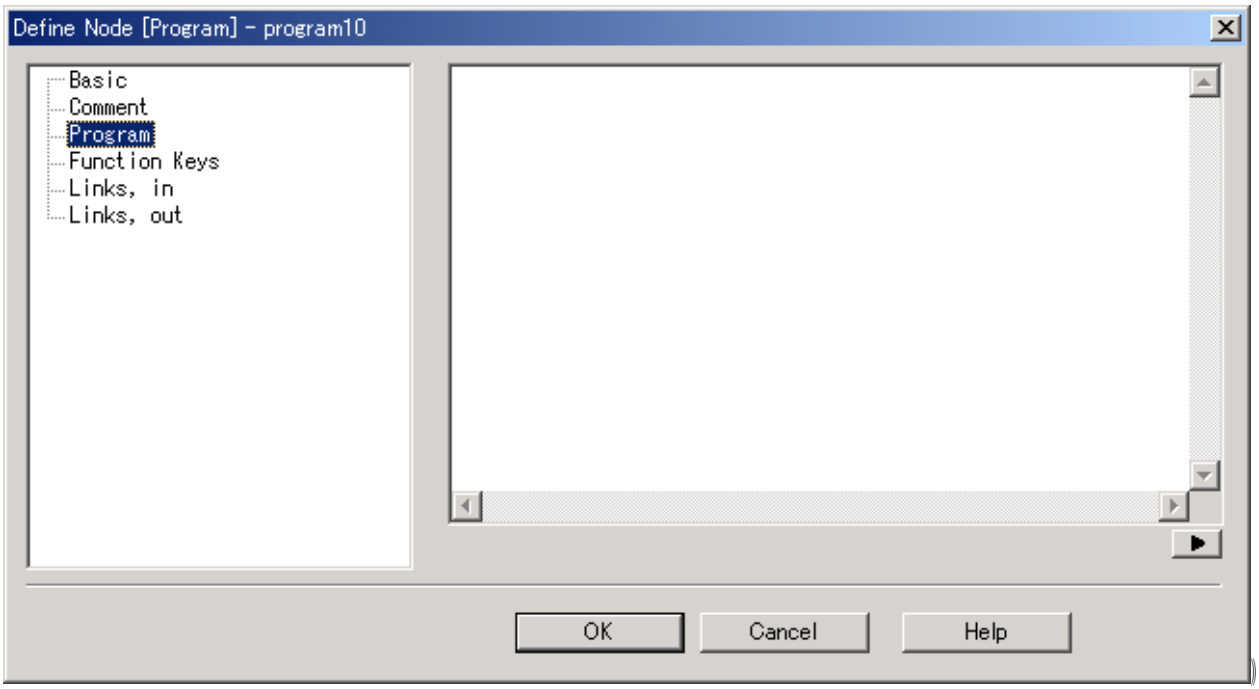
- **Global Access(グローバル・アクセス)**

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、入力表現式の、検証修飾子の、あるいはリンク表現の他のプログラム・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) ファンクション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ファンクション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、ファンクション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、そして"_ring()'の名前のファンクションがこのプログラムの実行時に呼び出されます。

多数のファンクション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・ファンクション、ある標準の C ライブラリ・ファンクション、そしてファームウェア・ファンクションの三種類に分けられます。

ジョブエンジン・ファンクションの詳細な説明については、*Help/Contents* の Function Library をチェックして下さい。標準 C ファンクションについては *Microsoft/C/Language/Help* のランタイム・ルーチンをチェックして下さい。ファームウェアファンクションの詳細な説明については、ポータブル・ターミナルの *Technical/Binder* をご覧下さい。

Run-Job) (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。

警告、run job (ジョブ実行)は call job(ジョブ呼び出し) とは違っています。Run job 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 run-job が新しいジョブを読み込んだときに消えます。Run-job 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。

"sub" job (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。Start Node Name(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。

Define Program Node

Basic プロパティ

Comment プロパティ

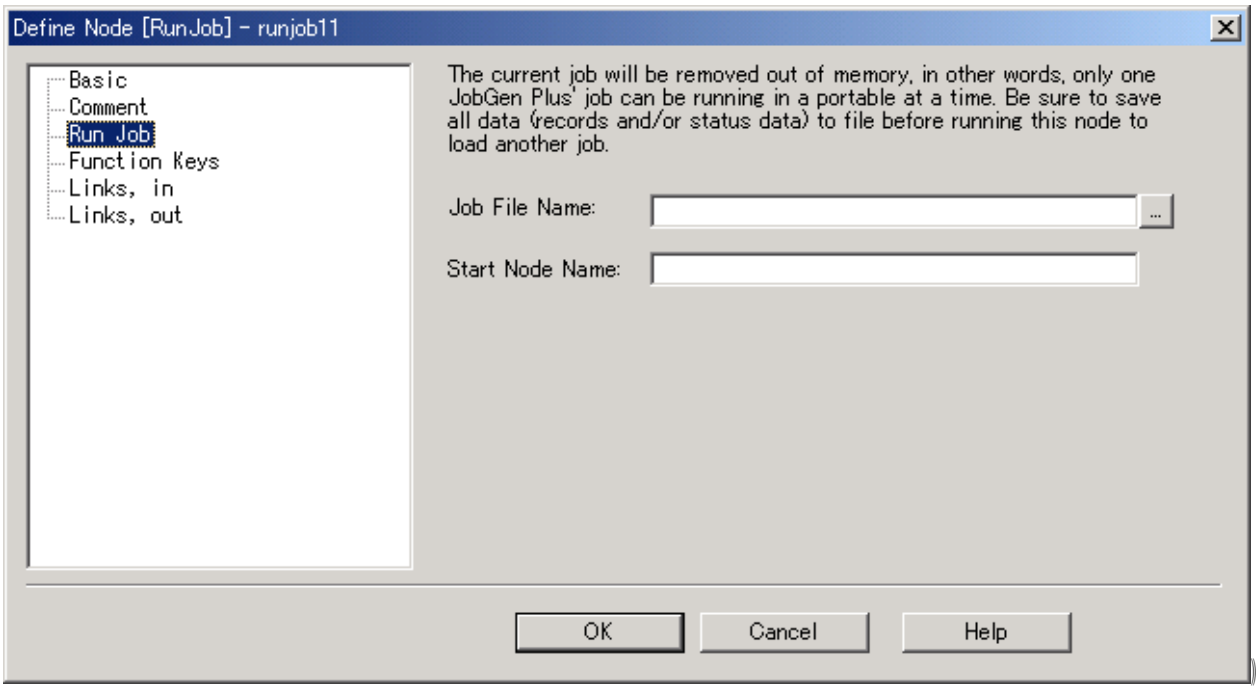
Function Keys プロパティ

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 プロパティ



- Job File Name　ジョブ実行ファイル名の定義
- Start Node Name　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(Comment node)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。

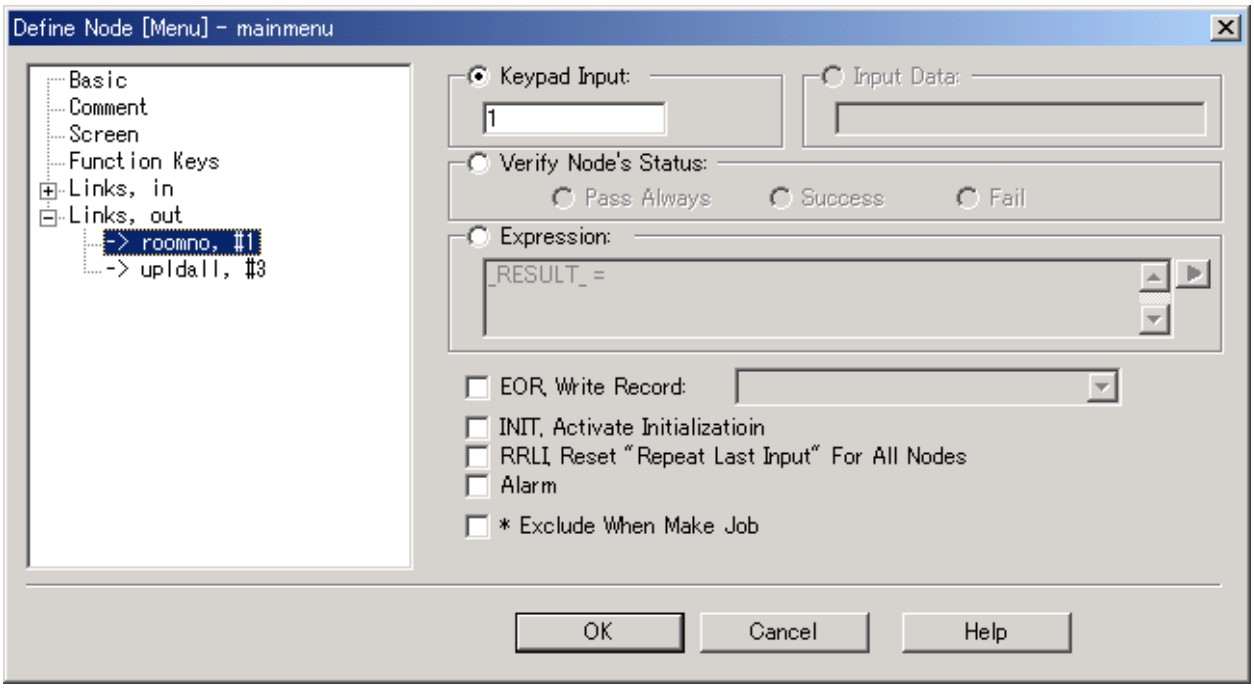
リンクの作成方法: ツールバーのリンクアイコンをクリックするか、あるいはノードメニューでリンクを選択することによってリンクツールを選択します。開始ノードを指して、マウスカーソルを相手先ノードにドラッグして、マウスボタンを離します。

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。

Define) Link) プロパティ



- **Keypad Input(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 にセットされます。

- **Input Data(入力データ)**

このオプションは接続元のノードが収集(collect)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 にセットします。

- **Verify Node's Return(ノードの戻り値の確認)**

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(menu)、消去(erase)そして編集(edit)ノードについては、その実行は常に Success(成功)です。収集(collect)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(Success)です。アップロード(upload)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 Failure(失敗)です。

検証(Verify)は 4 つのオプション Always(常に)、Success(成功)、Failure(失敗)、そして None(なし)を提供しています:

Always	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 TRUE.
Success	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 TRUE になる。
Failure	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 TRUE になる。

- **Expression(表現)**

これは C の表現です。これは収集(collect)ノードの検証定義の修飾子(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 (End of Record,レコード終了),Write Record(レコード書き込み)**

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た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 信号に合うことです。

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, Activate Initialization(初期化を行う)**

収集(collect)と演算(math)ノードの初期処理で、**Set) New) Value(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。)

- **Reset "Repeat Last Input" For All Nodes(すべてのノードに対して " 最後の入力繰り返し " をリセット)**

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(collect)ノードの入力セッションで、**Repeat) Last) Input(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) New) Input) Record) のチェックされたリンクに合った場合、Repeat) Last) Input) 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。) New) Input) Record) の機能は次のレコードのみに影響し、Repeat) last) Input を持つ収集(collect)ノードの後、リンクで再度) New) Input) Record) 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。)

- **Alarm(警告)**

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(collect)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**Verify Node's Return(ノードの戻り検証)**条件でアクティブにセットされた**Success(成功)**または**Failure(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**Fail(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**Success(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**Fail(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). **Expression(表現)**条件があり、そして論理表現の計算結果が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). **Input Data(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). **Always(常に)**オプションが**Verify Node's Return(ノードの戻り検証)**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。**Always**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). **Keypad Input(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。

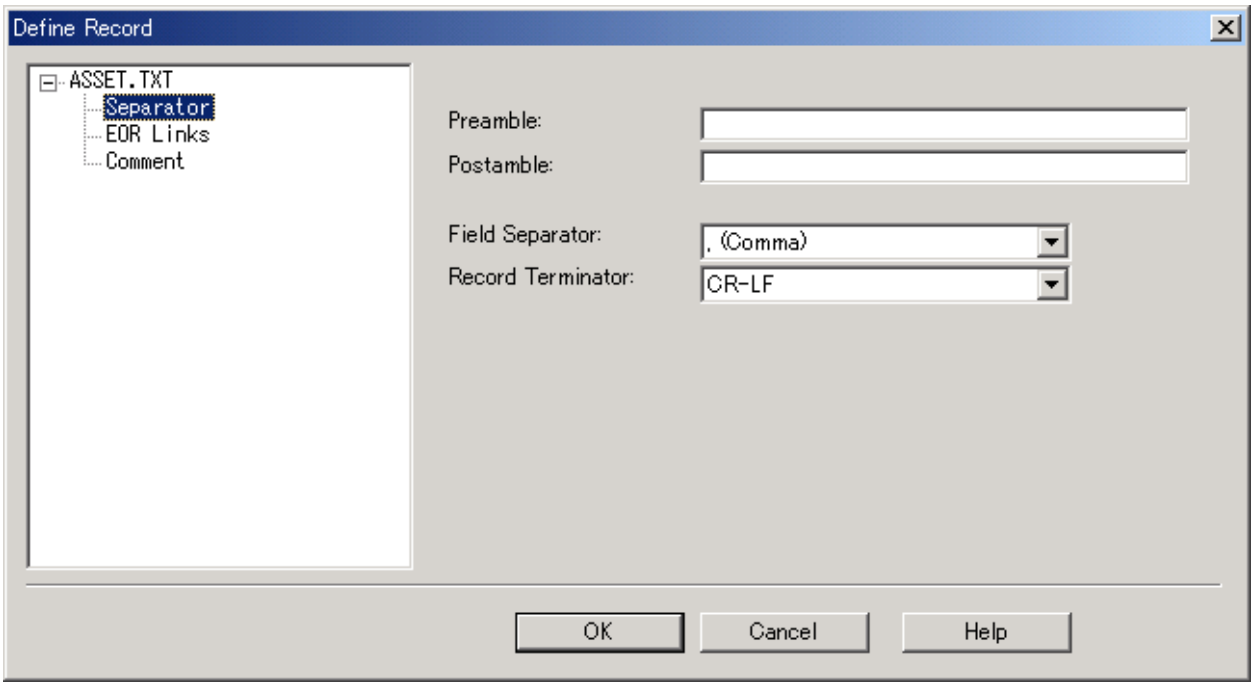
リンクを選択する6つの優先度があり、そしてすべてのリンクはリンクのシーケンス番号順に評価されます。リンクが選択された場合、実行は低い優先度の条件をチェックしないで直ちに相手先ノードに行きます

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**Exit**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/

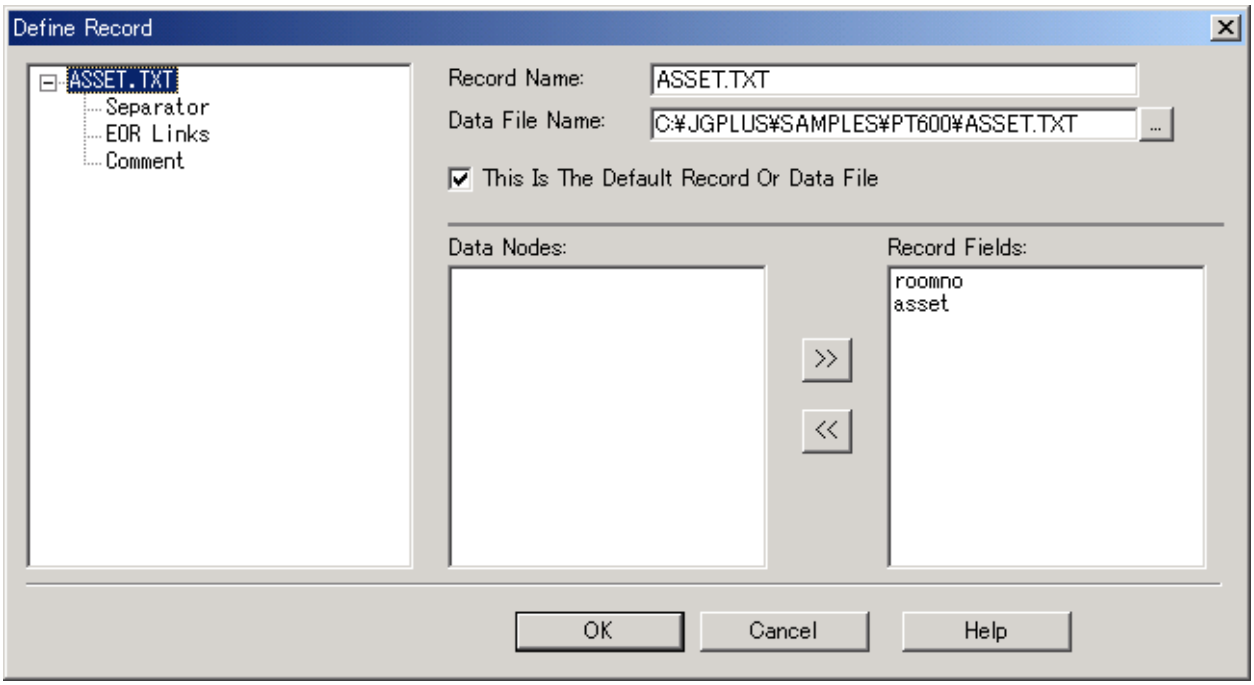
各ジョブは複数のレコードを持つことができます。レコードはフィールド、レコード区切り、プリアンブルとポストアンブルで構成されています。



レコードの属性を書く**フィールド(Field)**は、収集(collect)ノードです。収集(collect)ノードを定義して、そして Record Field(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(collect)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。**フィールドデリミタ(Field) delimiter)** はフィールドの分離のために使用され、そして**レコードデリミタ(Record) delimiter)**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(Data File Format)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(collect)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 Define Record ダイアログボックスを出すために Edit(編集)メニューから Define Record 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:) **All) Fields:**) (全フィールド、左のウインドウ)と **Fields) in) Record:**) (レコード中のフィールド、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(collect)ノードは、**All) Fields:**) 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、**Fields) in) Record:**) 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) **Add(追加)**) と) **Delete(削除))** ボタンがあります。)

収集(collect)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) **All) Fields:**) ウインドウでノードをクリックし、そして) **Add**) 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) **Fields) in) Record:**) 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、**Fields) in) Record:**) ウインドウのノードをクリックし、そして) **Delete**) 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。)

Field) と) **Record) Delimiters** 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デリミタ(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デリミタ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。)

Record) Field) チェックボックスで変更の効果を見るために、Define) Collect) 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(collect)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 信号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デリミタ。)
[**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デリミタ。)
[**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

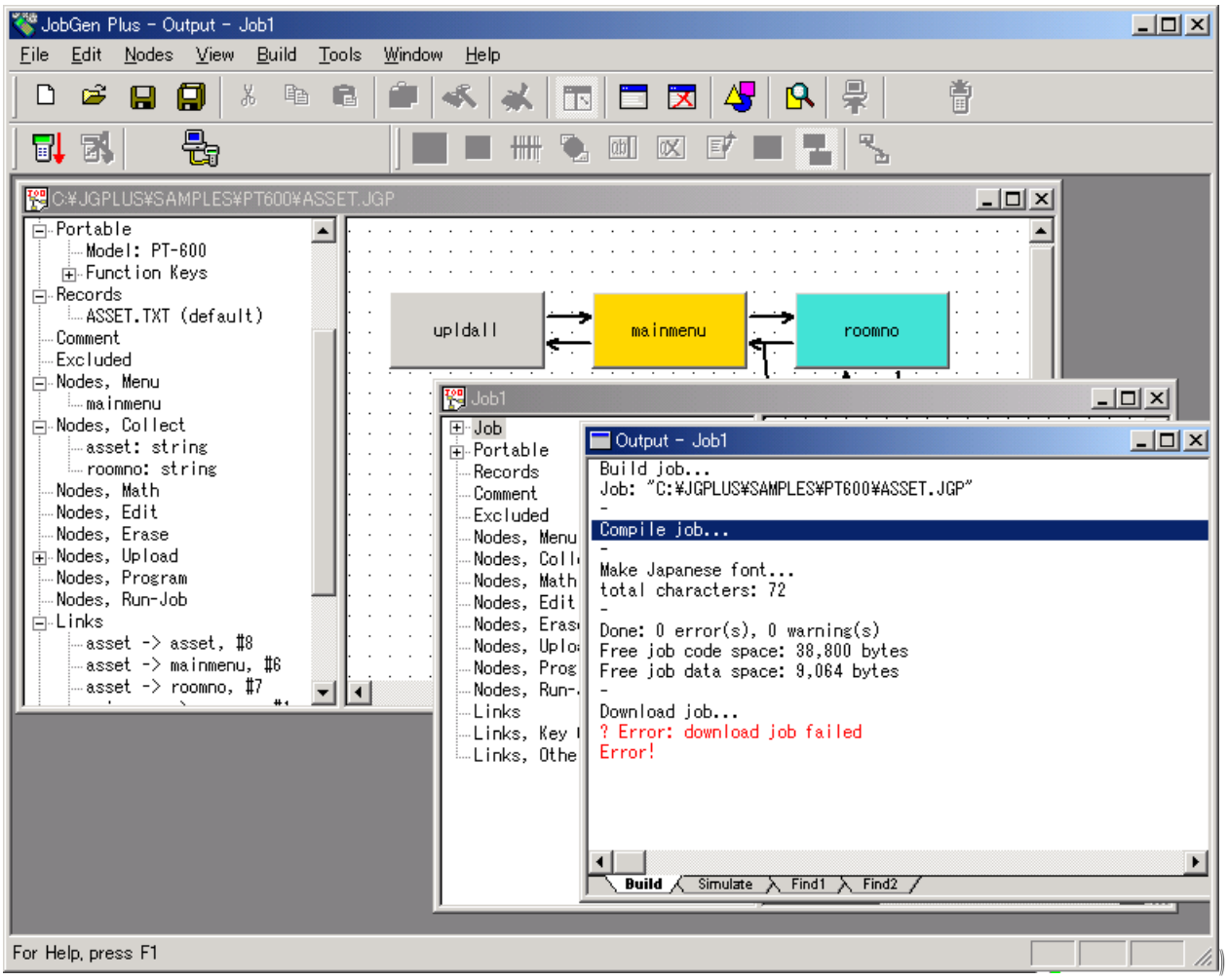
すべてのノードとリンクを作成した後で、次のステップは実行可能なジョブを作ることです。

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して、そして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、Make Job を再度実行して下さい。

Make Job の途中で、JobGen Plus は Make Binary Search Index Table(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 Make Japanese Characters Font File(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 Manager 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ィ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 "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ト

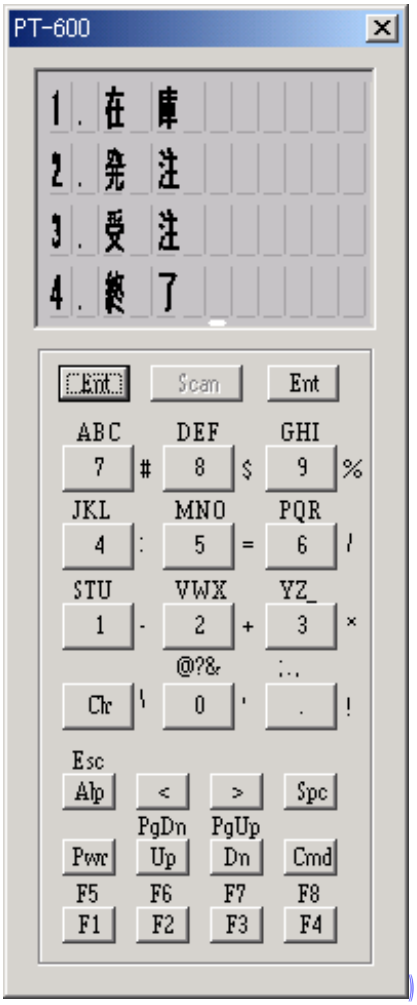
ジョブが設計され、定義された後で、これはホスト PC でシュミレートすることができます。シュミレーションをスタートするには Run > Simulate Job を選択するか、あるいはツールバーのシュミレートアイコンをクリックします。

シュミレーションは Windows 95/98/Me でのみ使用可能です。Windows 3.x、NT と 2000 はサポートされていません。

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トします。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。

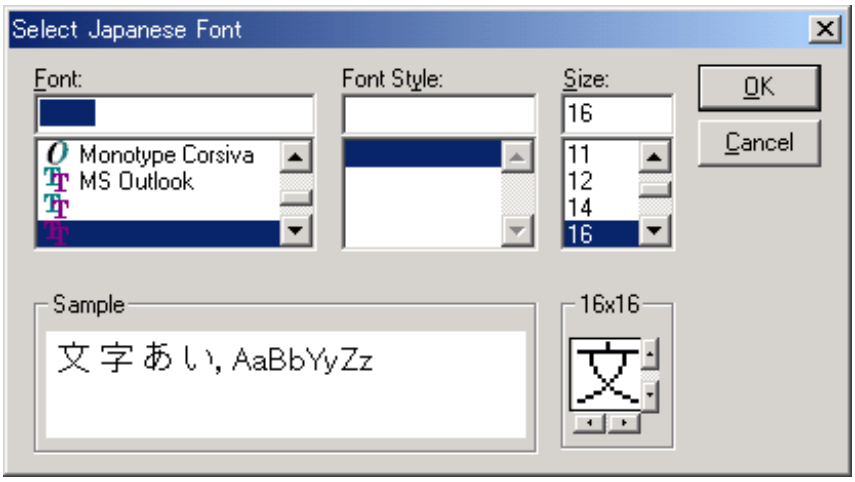

PT-600
1. 在 庫
2. 発 注
3. 受 注
4. 終 了
Ent
Scan
Ent
ABC
DEF
GHI
7
#
8
$
9
%
JKL
MNO
PQR
4
:
5
=
6
/
STU
VWX
YZ_
1
-
2
+
3
×
@?&
;.,
Chr
\
0
'
.
!
Esc
Alp
<
>
Spc
PgDn
PgUp
Pwr
Up
Dn
Cmd
F5
F6
F7
F8
F1
F2
F3
F4

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 Large Font 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。Screen から Large Font 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)。)