JobGen Plus User's Manual

JobGen Plus

For Microsoft Windows
Version 6.0

Document Revision 6.0, June 2011

Table of Contents

JobGen Plus Essentials

In this section you will learn:

- *Overview of JobGen Plus*
- *Installing JobGen Plus and the system requirements*
- *Working environment of JobGen Plus*

Chapter 1 Introduction

Overview of JobGen Plus

JobGen Plus is a programming tool that will enable users to generate their own data collection applications. It was designed along the concept that users should be able to develop data collection applications as easily as scribbling a plan on a piece of paper. **JobGen Plus** provides a user-friendly programming environment, a complete set of programming tools, and an ultra-versatile data collection solution for even non-programmers.

Because **JobGen Plus** is a Windows application, it can take full advantage of Windows' well-known user-friendly operating environment. Users can access all of **JobGen Plus**' programming tools by simply clicking a mouse button. This comprehensive set of programming tools was developed by careful consideration of all the steps involved in designing a data collection application. This makes it possible for any user of **JobGen Plus** to easily develop their own sophisticated data collection applications for use in industry, schools, retail stores, warehouses - pretty much anywhere - without any prior programming experience.

Also, **JobGen Plus** provides a **programming language** option for advanced users who need to write even more complex data collection applications. This option frees up programmers by eliminating the old limits imposed by other development tools on the types of data collection applications possible. However, most users can now implement a very wide variety of data collection applications - from warehouse management to recording business transactions – by using only **JobGen Plus**.

Components of JobGen Plus

A data collection application created by **JobGen Plus** is comprised of two major components: The first one, called a *Node*, represents a **process** in the data collection application.
The second one, called a *Link*, represents the **transition** from one process to another. These two major elements of **JobGen Plus**' application will be discussed in great detail in the following chapters. This section only serves to introduce the concepts of *Nodes* and *Links*.

Each *Node* represents one task in data collection. Examples of tasks in a data collection application include: selecting menu options on the data collector; entering item numbers on the data collector; or sending collected inventory information from the data collector to the host computer. **JobGen Plus** provides nine types of *Nodes* to perform various tasks in the data collection application: *Comment Node, Menu Node, Collect Node, Math Node, Edit Node, Erase Node, Upload Node, Program Node, and Run-Job Node*.

Comment Node provides a way to paste notes directly onto the flowchart.

Menu Node supplies a space for application designers to display messages, instructions, or the application's major functions for selection by the users.

Collect Node is used for storing information entered by the user on the data collector.

Math Node is used to calculate data.

Edit Node allows review or modification of the data recorded on the data collector.

Erase Node deletes the data recorded on the data collector.

Upload Node sends the collected information from the data collector to the host computer.

Program Node allows the application developer to enhance their project by adding C language modules.
Run-Job Node is used to load and run another JobGen Plus job.

Links render transitions between two or more processes (*Nodes*). *Links* not only connect all processes into one **JobGen Plus** application, but they also provide the path and direction from one process (*Node*) to another. Information about how or when transitions occur is recorded inside each *Link*.

Chapter 2 Installation

System Requirements

Following are the minimum requirements for installation of **JobGen Plus**:

- IBM AT Compatible Computer with Intel 80386 (or compatible) CPU and above
- 8 Megabytes of Memory
- 5 Megabytes free space on Hard Disk
- VGA monitor or better
- One serial port (COM1, COM2, COM3, or COM4) available for communication
- One mouse (or other pointing device that operates under MS Windows)
- Windows 95/98/Me/NT/2000/XP/Vista/7

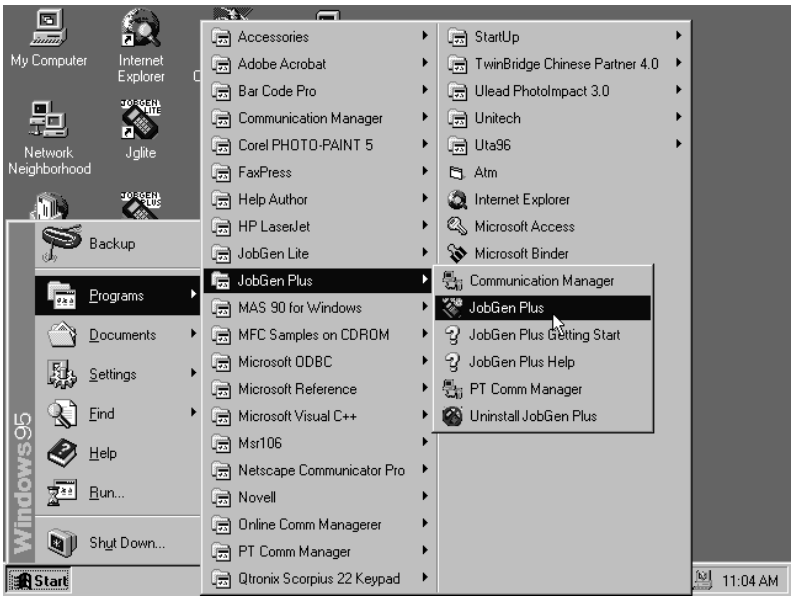
Installation on MS Windows OS

1. Get **JobGen Plus** installation file ***"JobGenPlus-vx.x.xxxxx.msi"*** into any folder on your PC.
2. Double click "***.msi"** to run it. (if there is previous JobGen Plus version installed on the same PC, it is necessary to un-install it first)
3. Press the **Next** button
4. Select target drive/folder y or click **Next**.
5. Click **Next** to start instllation
6. **JobGen Plus** will run *Setup Initialization* and then ask the user to enter the *Directory Name* for storing **JobGen Plus** system files. The default directory name is: *JGPLUS*.
7. The Setup program will begin copying files to the designated directory.
8. When the setup program has finished creating the Program Folder and Icons for **JobGen Plus**, installation is complete.

Chapter 3 Working with JobGen Plus

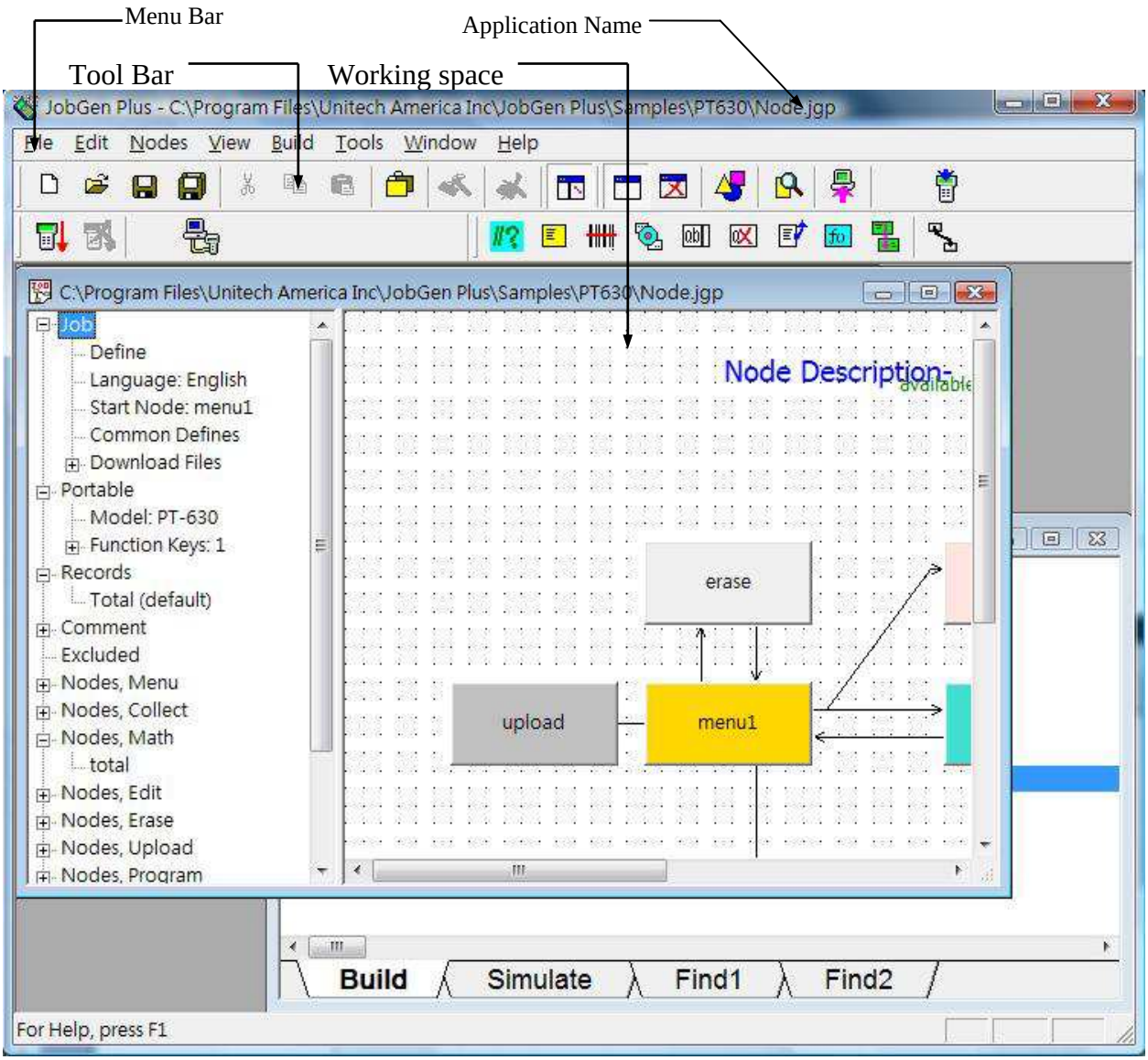
Running JobGen Plus from MS Windows

1. Click the Start button on the taskbar and select Programs.
2. Move the mouse cursor to the program group containing the **JobGen Plus** program.
3. Click on the Job Gen Plus icon to run.
   (or, click on the desktop shortcut icon if you've already set it up)

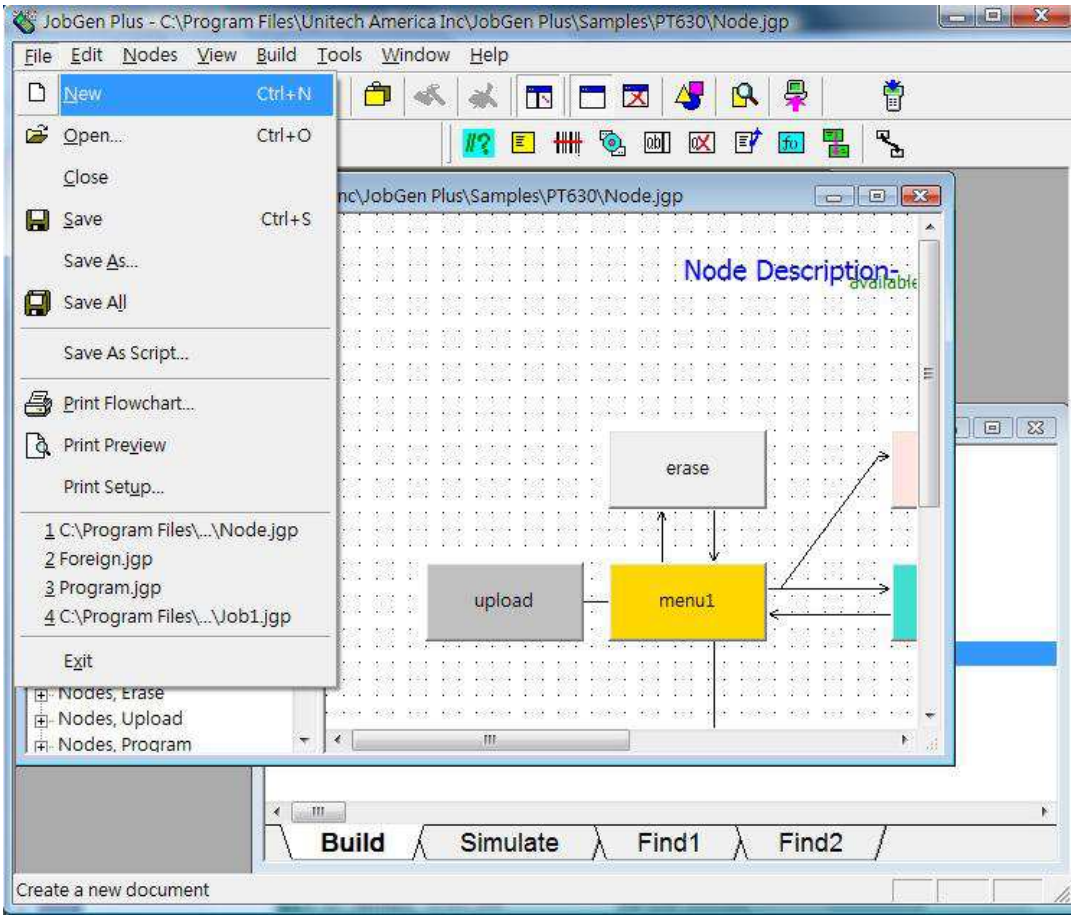


Now **JobGen Plus** is ready to develop applications for data collection.

Working Environment of JobGen Plus

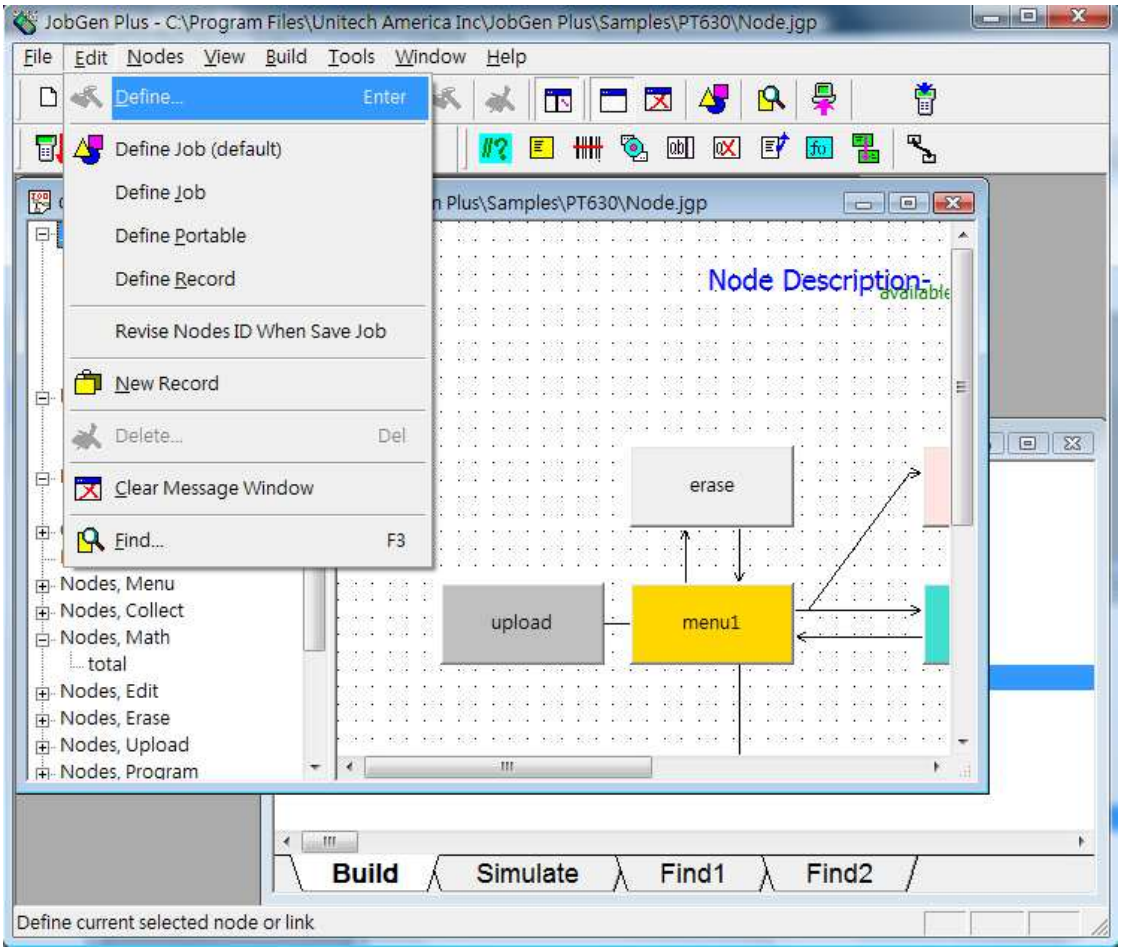


1. **File** Commands:



New	Create a new Job.
Open	Open an existing Job file.
Close	Close the current job window.
Save	Save the content of the current Job.
Save As	Save the content of the current Job to a different file name.
Save All	Save the content of all opened Jobs.
-	
Print Flowchart	Print the flowchart of the current Job.
Print Preview	Preview the printing output on the screen.
Print Setup	Setup printing options.
-	
Recent Files	List recently opened files
-	
Exit	Exit **JobGen Plus** program.

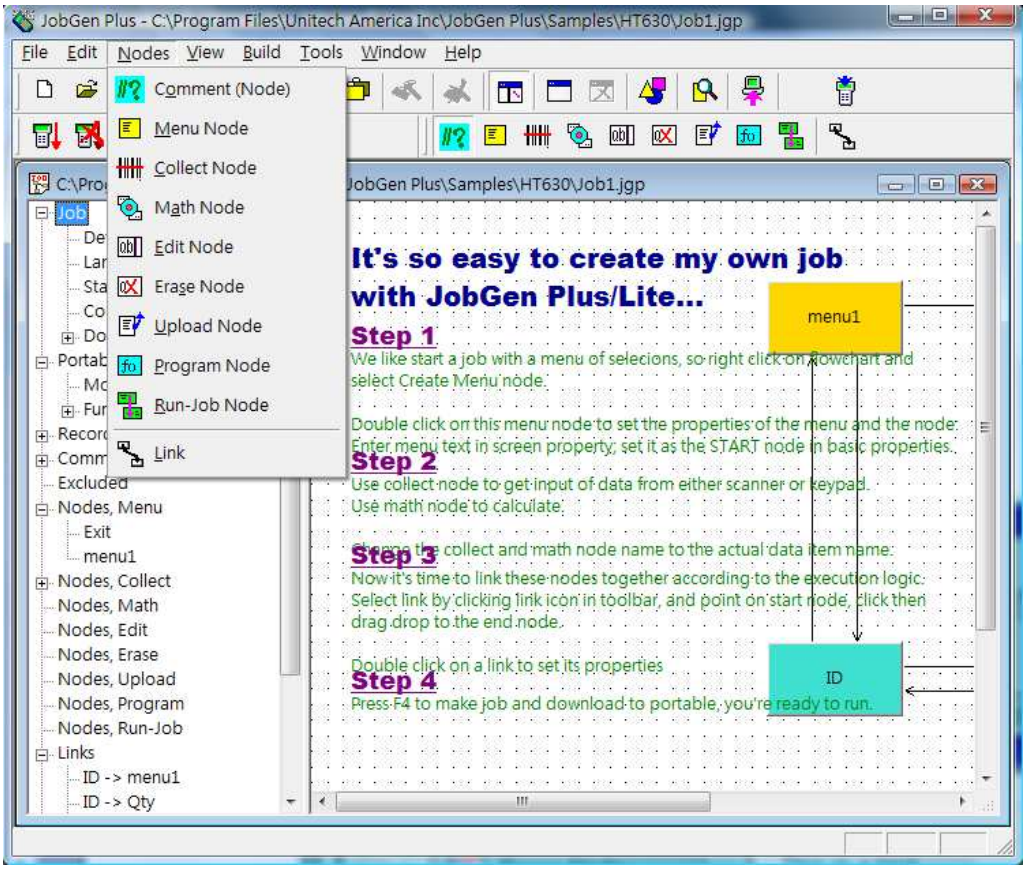
2. Edit commands:



Define	Define current selected object, which can either be a node, a link, or a record.
-	
Define Job [default]	Define default settings for a Job.
Define Job	Define the current Job.
Define Portable	Define portable settings.
Define Record	Define records settings.
-	
New record	Create a new record. JobGen Plus supports multiple records.
Delete	Delete the current selected object, which can either be a node, a link, or a record.
Clear Message Window	Clear all messages displayed in the message window.

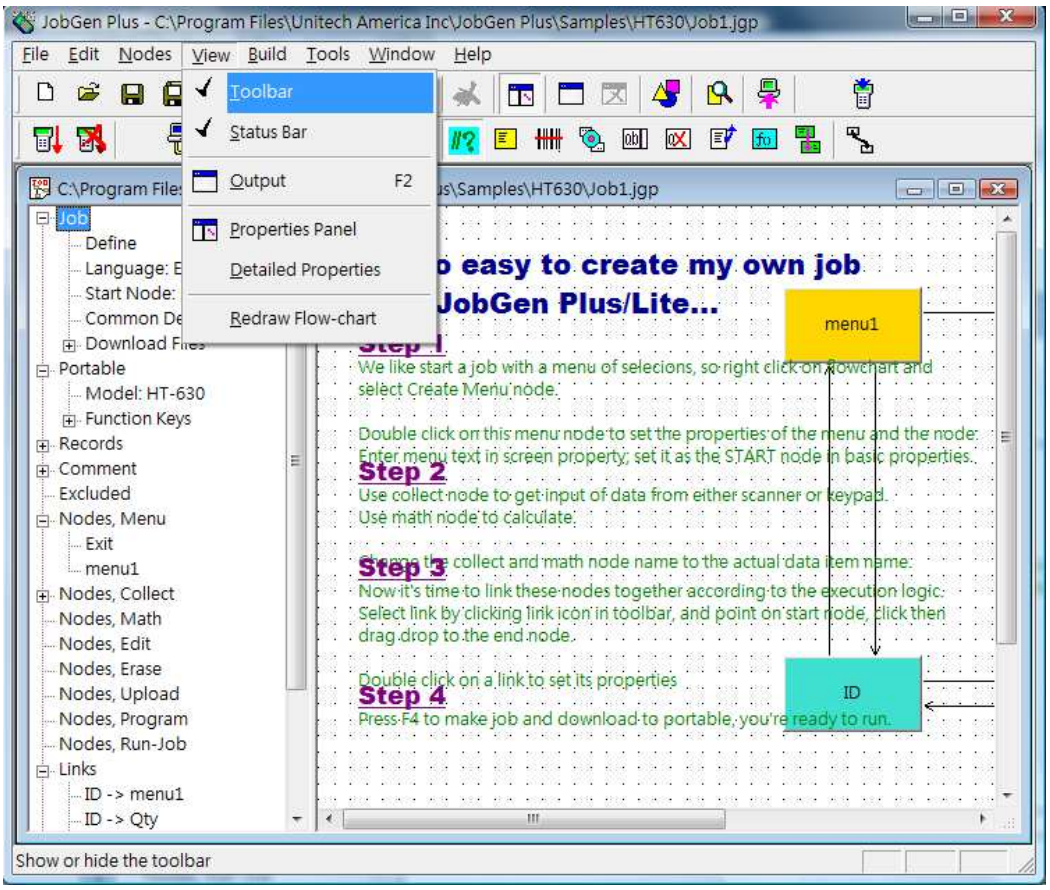
Find	Find text in the current Job. The results will be displayed in the message screen Find1. If “Output To Pane 2” is checked, all the results will be displayed in message screen Find2. Double clicking on the found result in message window will open a related definition dialog window.

3. Nodes commands:



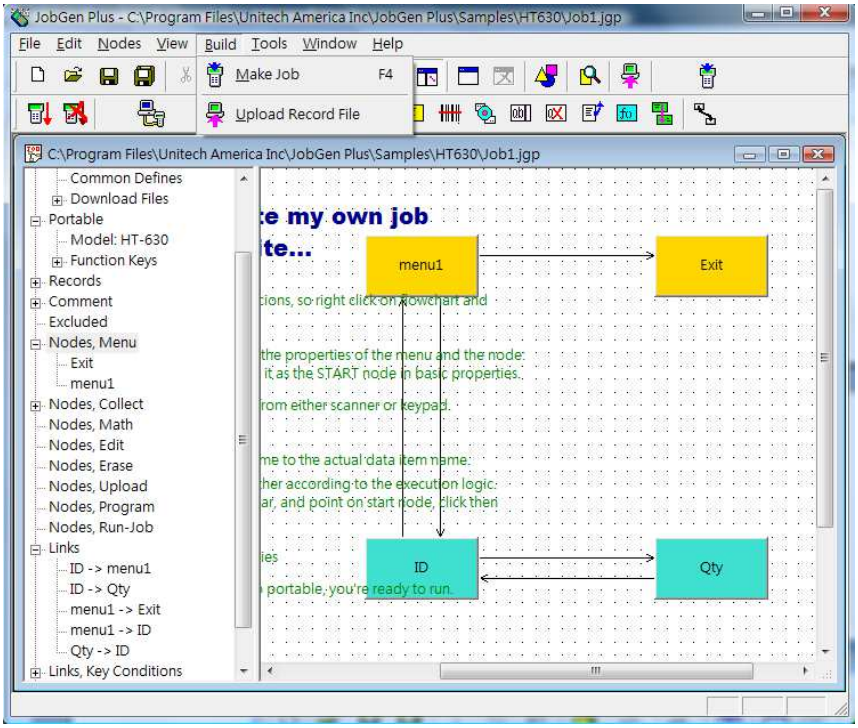
Comment	Make Comment Node ready for implementation.
Menu	Make Menu Node ready for implementation.
Math	Make Math Node ready for implementation.
Collect	Make Collect Node ready for implementation.
Upload	Make Upload Node ready for implementation.
Erase	Make Erase Node ready for implementation.
Edit	Make Edit Node ready for implementation.
Program	Make Program Node ready for implementation.
Run-Job	Make Run-Job Node ready for implementation
-	
Link	Make the Link ready for implementation.

4. View commands:



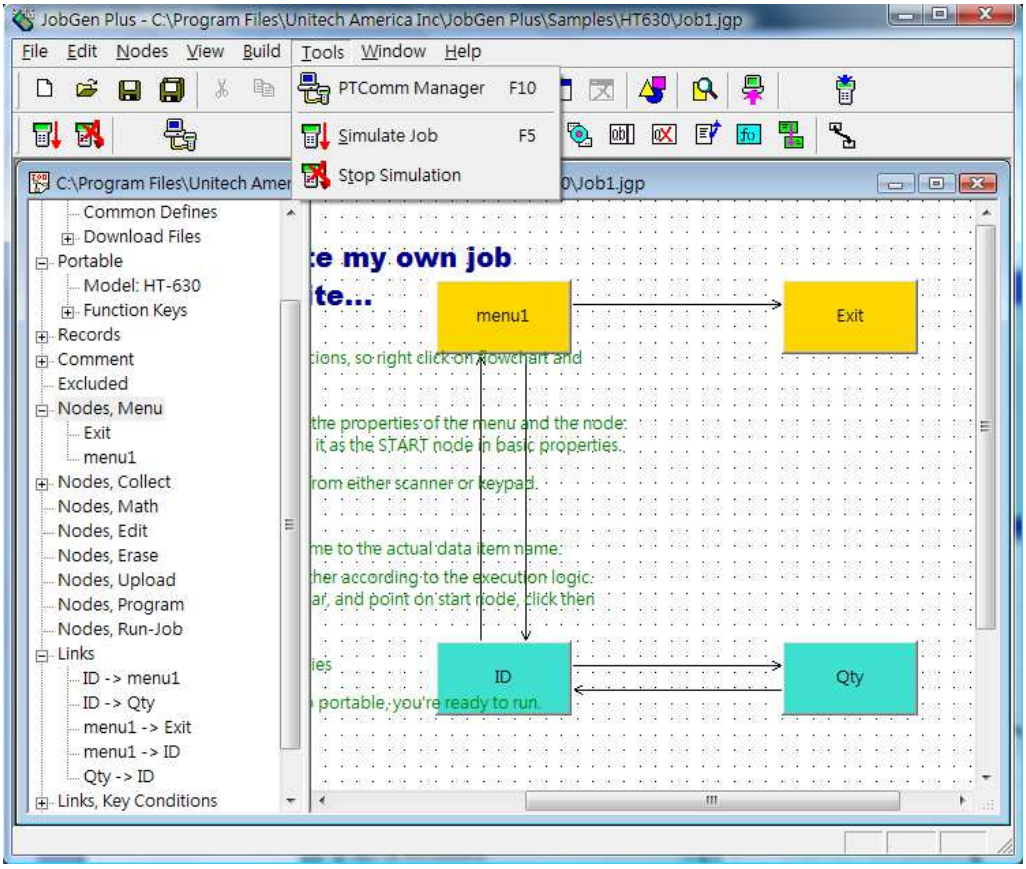
Toolbar	Display the Tool Bar when this command is checked.
Status Bar	Display the Status Bar when this command is checked.
-	
Output	Open message window.
-	
Properties Panel	Open properties panel.
Detailed Properties	Enable display of detailed information in the properties panel.
-	
Redraw Flowchart	Redraw whole flowchart.

5. Build commands:



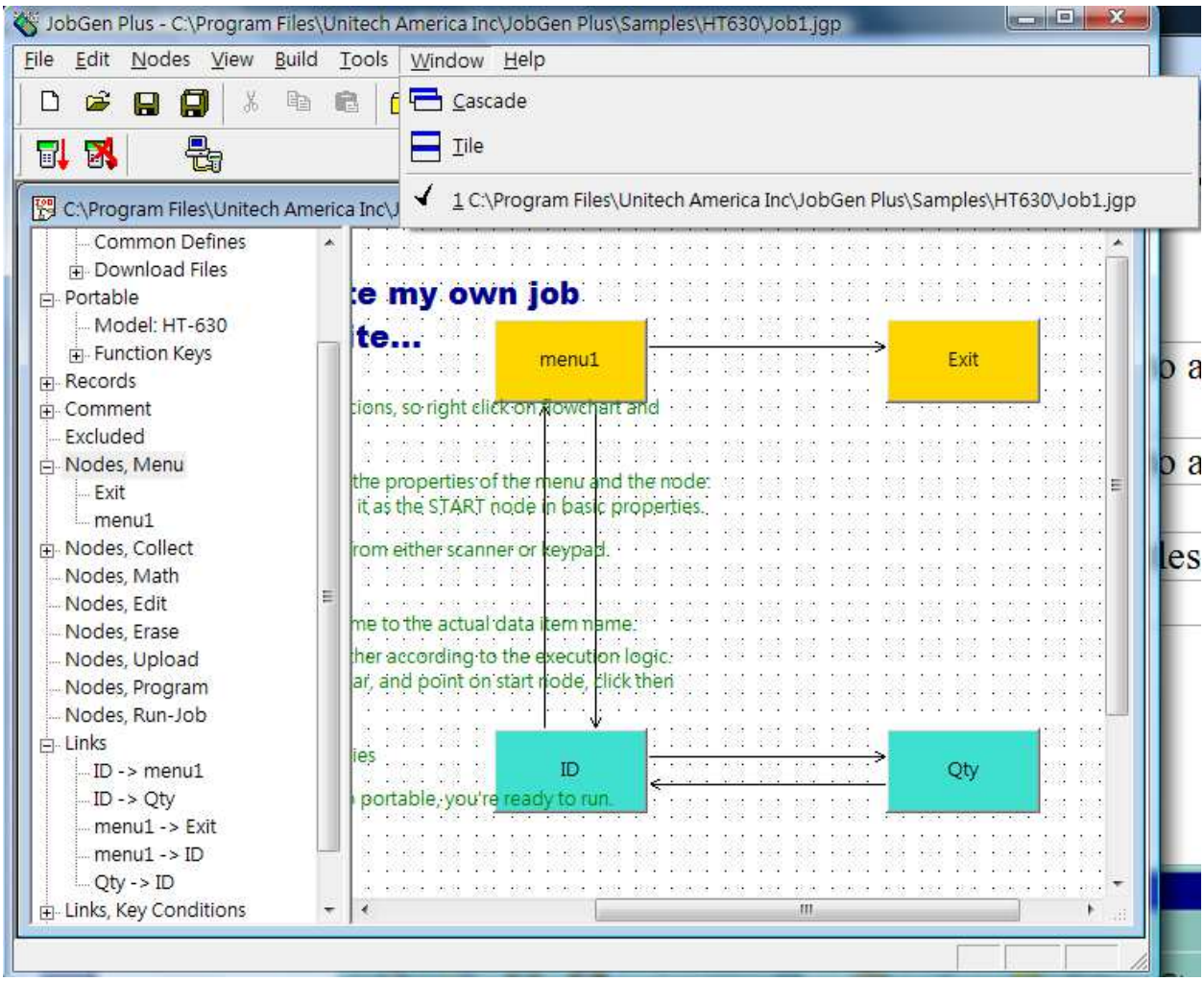
Make Job	Generate execution code for the Job, and download files including code and all data files to the portable.
-	
Upload Record File	Upload a record file.

6. Tools commands:



PTComm Manager	Run PTComm manager application.
-	
Simulate Job	Start job simulation.
Stop Simulation	Cancel job simulation.

7. Window commands:



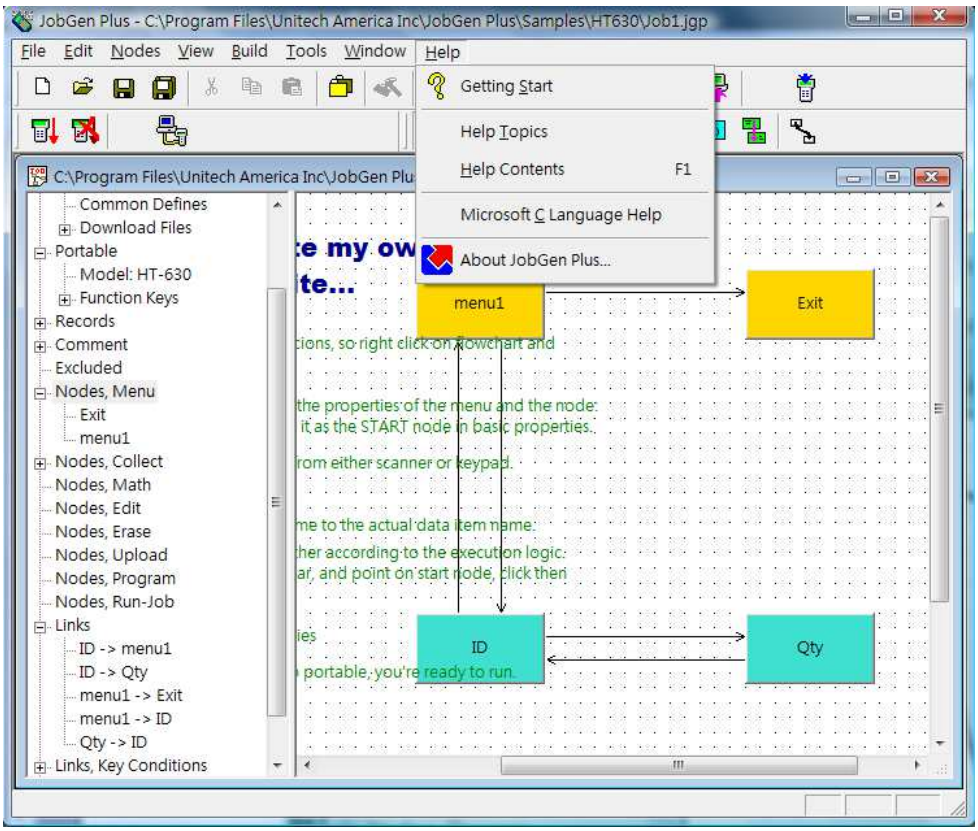
Cascade Window Rearrange all windows into a cascaded display.

Tile Rearrange all windows into a tiled display.

-

Opened Window Title List all opened window titles

8. Help commands:



Getting Started	Display the “Getting Started” instructions.
-	
Help Topics	Display help topics.
Help Contents	Display helps for JobGen Plus.
-	
Microsoft C Language Help	Display helps for *Microsoft C Language Run-Time Reference.*
-	
About JobGen Plus	Display the version number and copyright of **JobGen Plus**.

Job Implementation

In this section you will learn:

- *How to design a Job*
- *How to create a Job*
- *How to create an executable Job*

Chapter 4 Job Design

Before creating a *Job*[1], users must first identify all the various tasks needing to be accomplished. Examples of tasks include the following: collecting data, making selections, sending data, checking data, printing messages, . . . etc. Secondly, users should plan how to execute these tasks. Some listed tasks can be organized into an orderly procedure, with transitions from one task to another. Meanwhile, some tasks may work independently as supporting functions of the whole *Job*. The last step is to implement all the tasks and the transitions between tasks into **JobGen Plus**.

Transforming the tasks and the transitions between tasks is easy. Users simply replace each task with the appropriate corresponding *Node* in **JobGen Plus**. For example, a task for collecting data can be replaced by a *Collect Node*. The following table shows the types of tasks that can be replaced by corresponding *Nodes* in **JobGen Plus**:

Nodes	**Tasks that can be replaced**
Comment	• Display notes and descriptions
Menu	• Display selections for users
	• Display error messages
	• Display instruction
Collect	• Data entry from keypad
	• Data entry from bar code scanner
	• Data entry from the data collector's timer
	• Data entry from Look-up file
	• Data entry from a formula
	• Verify data entry
Math	• Data entry from a formula
	• Calculate data
	• Compare data
Erase	• Delete records inside the collected data
Upload	• Send data from the data collector to the host computer
Edit	• View data stored in data file in portable
	• Search data
	• Modify data
Program	• Any type of task that cannot be replaced by the *Nodes* listed above
Run-Job	• Run another job

Transitions from one task to another can be substituted by *Links* in **JobGen Plus**.

[1] A *Job* is a data collecting application.

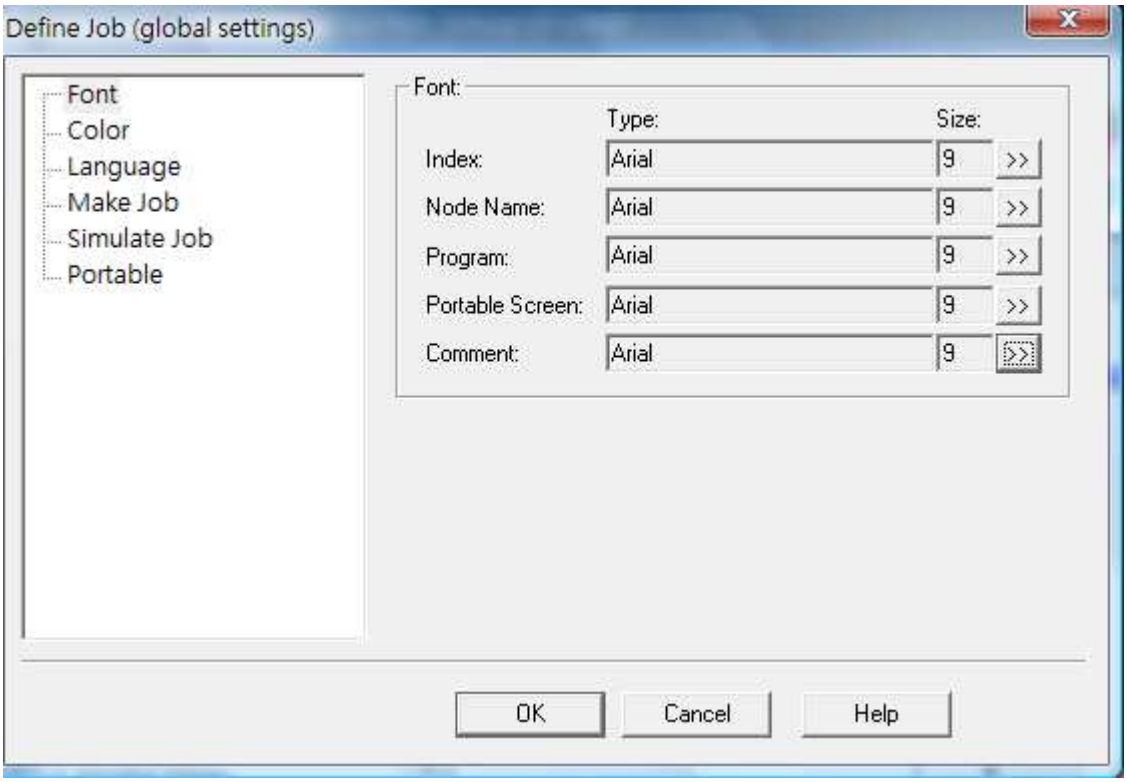
Chapter 5 Job Programming

To build a *Job*[1] is as easy as clicking a mouse button. **JobGen Plus** takes full advantage of MS Windows, so therefore most users can intuitively use the tools provided by **JobGen Plus**. This chapter will describe in detail all the processes necessary to create a *Job*.

Define Default Job Settings

Select menu Edit > Define Job [Default] to open the Define Default Job settings dialog window. These settings will affect all working Jobs.

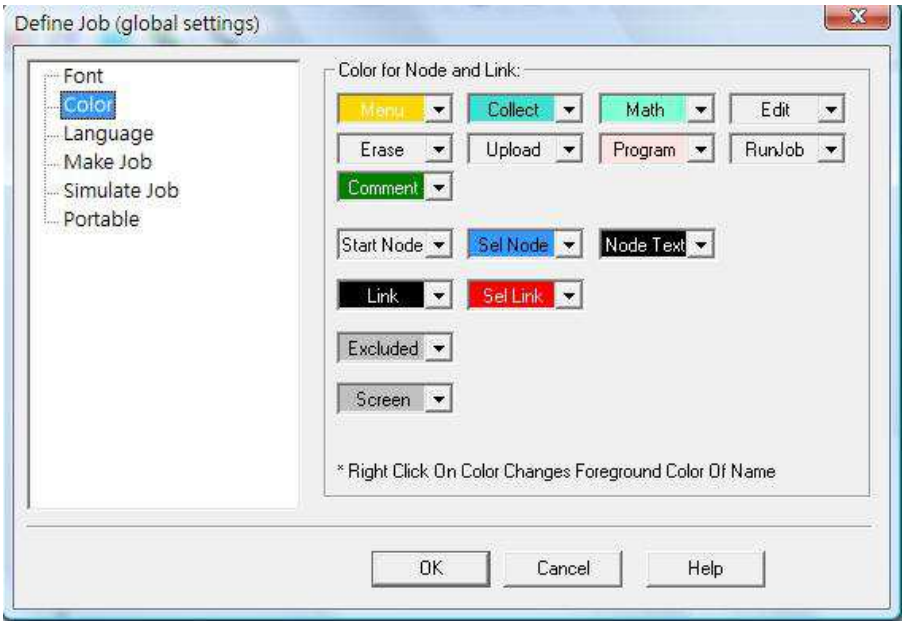
Font Property:



Define fonts for the left properties window, node name, program editor and comment.

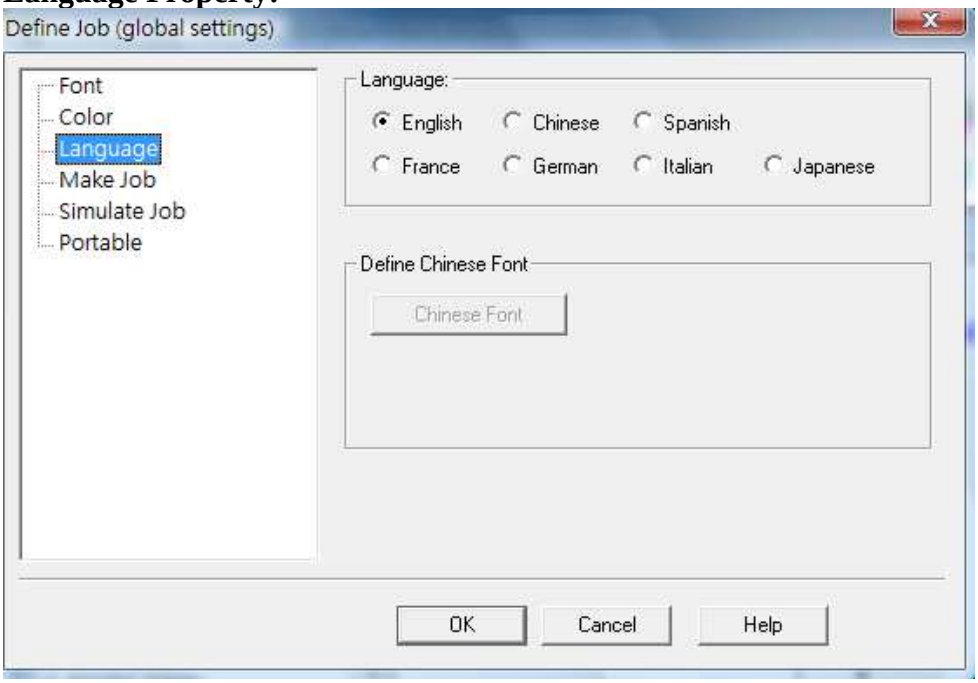
[1] A *Job* is a data collection application.

Color Property:



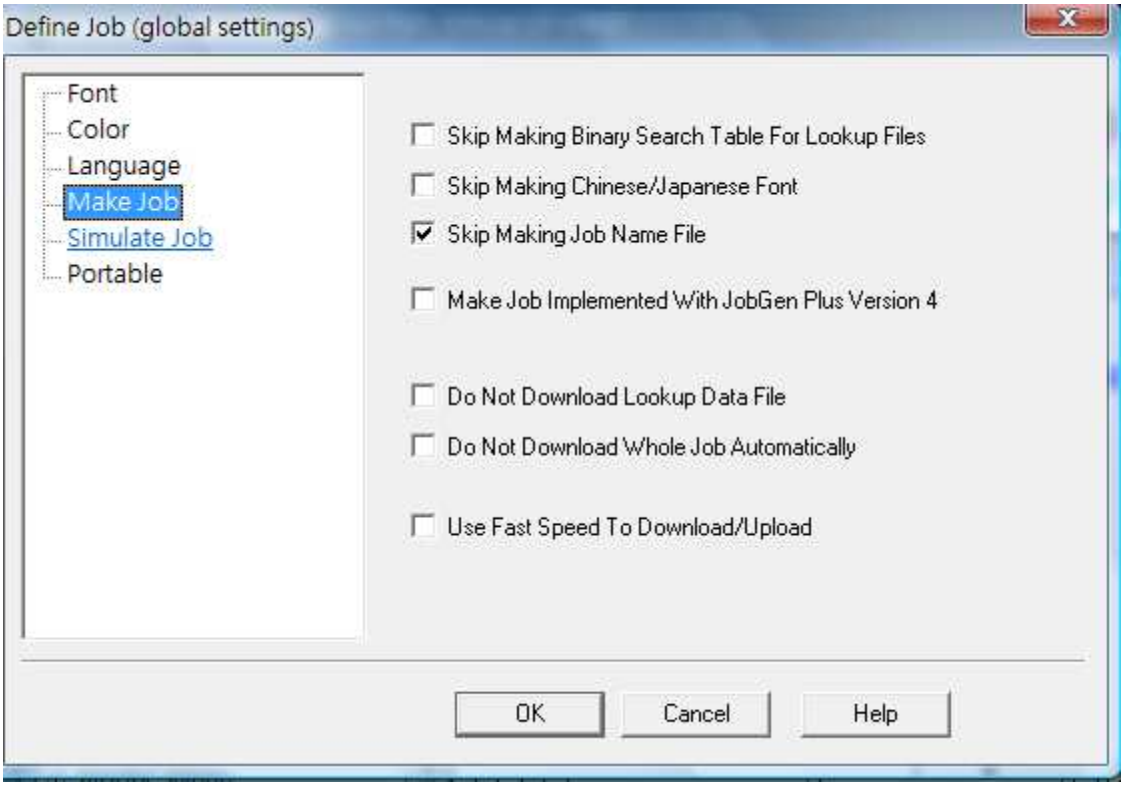
Define colors for nodes, links, and selections. Shown are the default color settings for nodes. Each individual node color can be changed in its basic property setting.

Language Property:



Select language for Job.

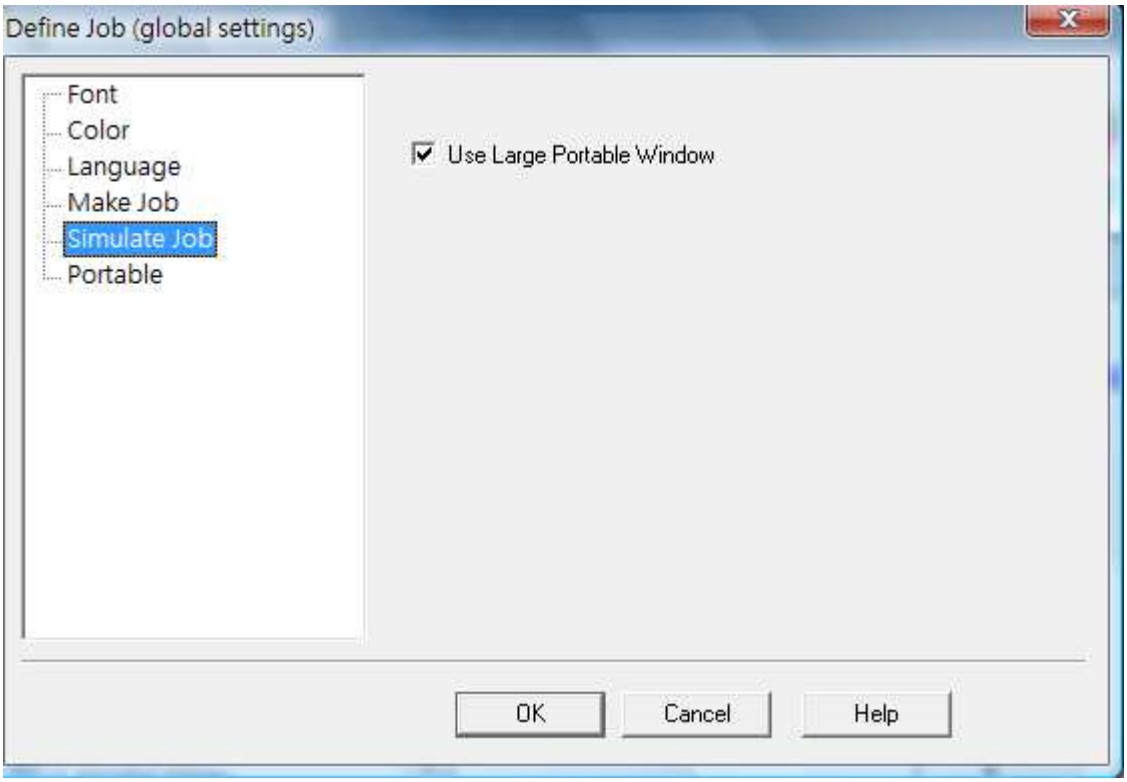
Make Job Property:



Skip Making Binary Search Table For Lookup Files	If the content of the lookup file does not change, you needn't compile it each time you make a Job. This speeds up the process of making a Job.
Skip Making Chinese Font.	If all text, including in the lookup files and data files does not change, you needn't make a Chinese font each time.
Skip Making Job Name File	The Job Name File is used for the job engine to determine which Job to run. If you execute Jobs by selecting the Job executable name, Job Name File becomes unnecessary.
Make Job Implemented With JobGen Plus Version 4	If you encounter a problem with an old Job - which was working OK in an old JobGen Plus version - check this option and try making the Job again. This allows downward compatibility with previous versions.
Do Not Download Lookup Data File	If the contents of all lookup files do not change, you needn't download them every time.

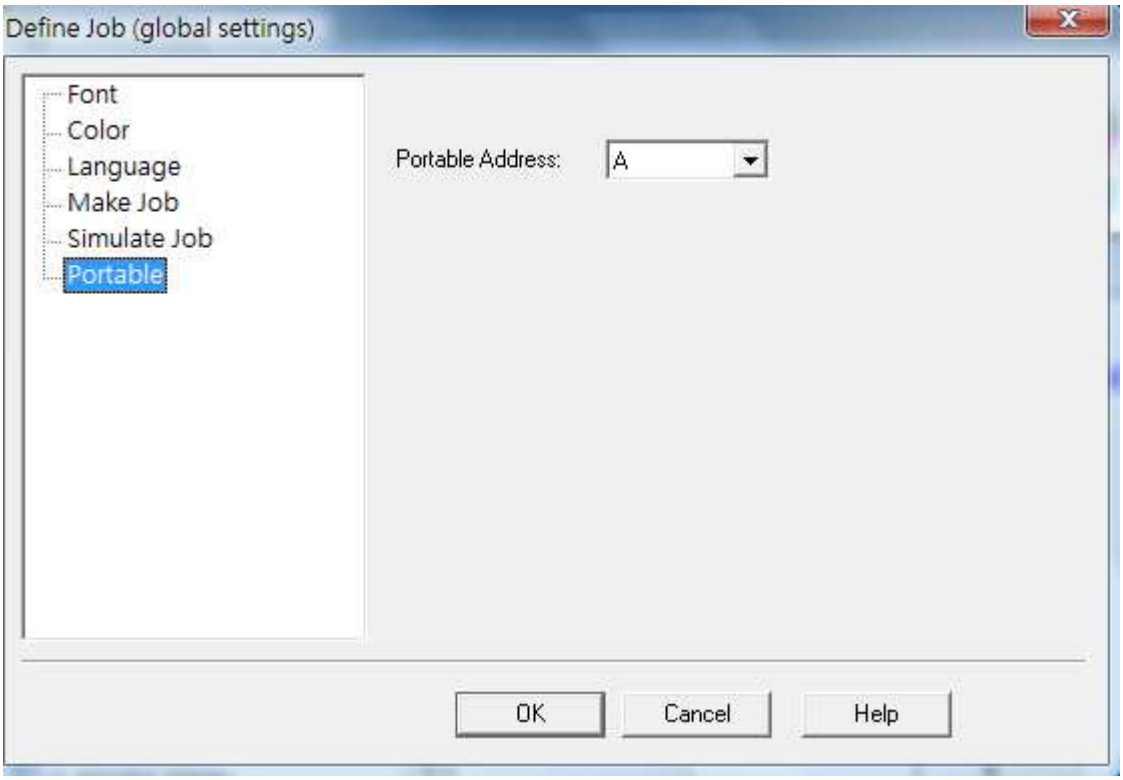
Do Not Download Whole Job Automatically	JobGen Plus will download an executable Job to the portable when it has successfully generated the Job execution code. Check this option to stop downloading. JobGen Plus will then copy the Job engine (jeng.exe) to the Job folder for later downloading.
Use Fast Speed To Download/Upload	JobGen Plus will force PTComm Manager to communicate at the fastest possible speed.

Make Job Property:



Define if using large screen for simulation

Portable

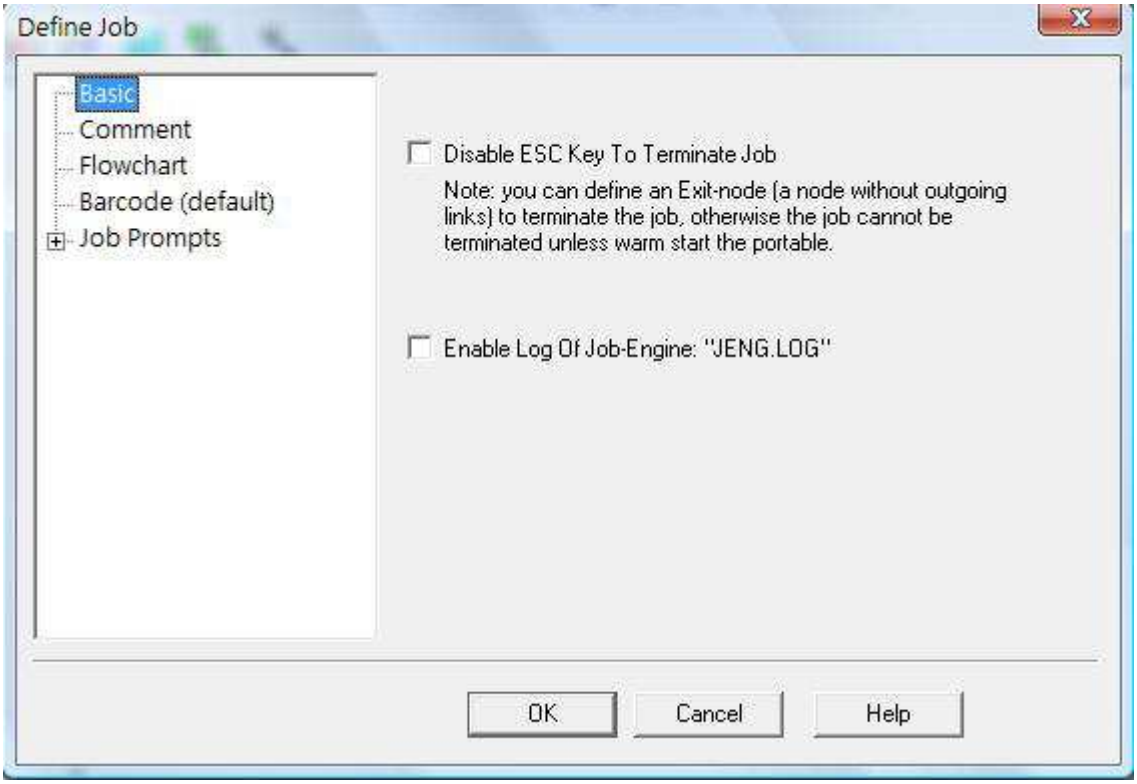


Define default portable terminal's address ("A"~"Y", "0"~"6")

Define Default Job Settings

Select Job> Define in Job's left Properties Window to open the Define Job settings dialog window.

Basic Property

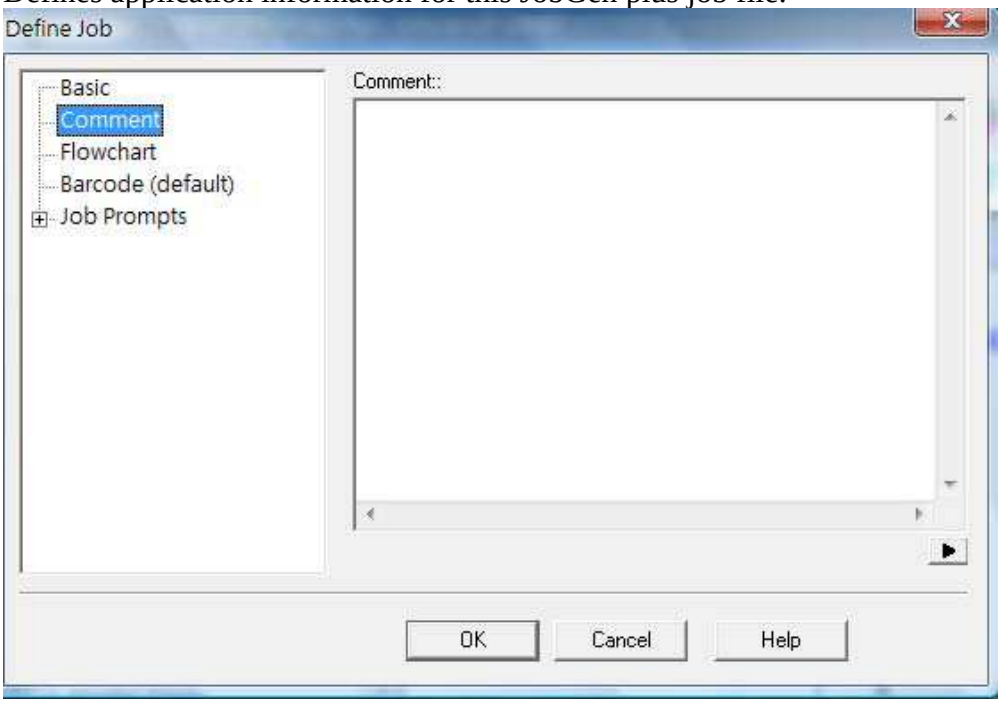


Defines if disable ESC hot key to prevent operator to exist JobGen Plus program to HT-630's menu screen.

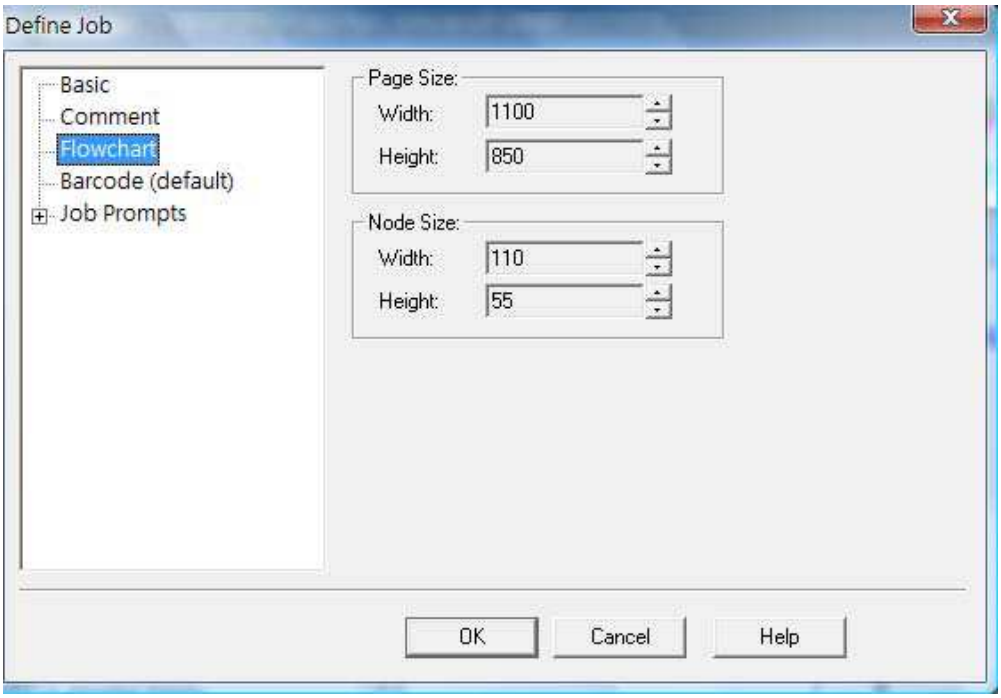
Define if enable log tracking. It will be useful for Unitech to trace abnormal problem. You can send this log back to Unitech to us to do debugging.

Comment Property

Defines application information for this JobGen plus job file.



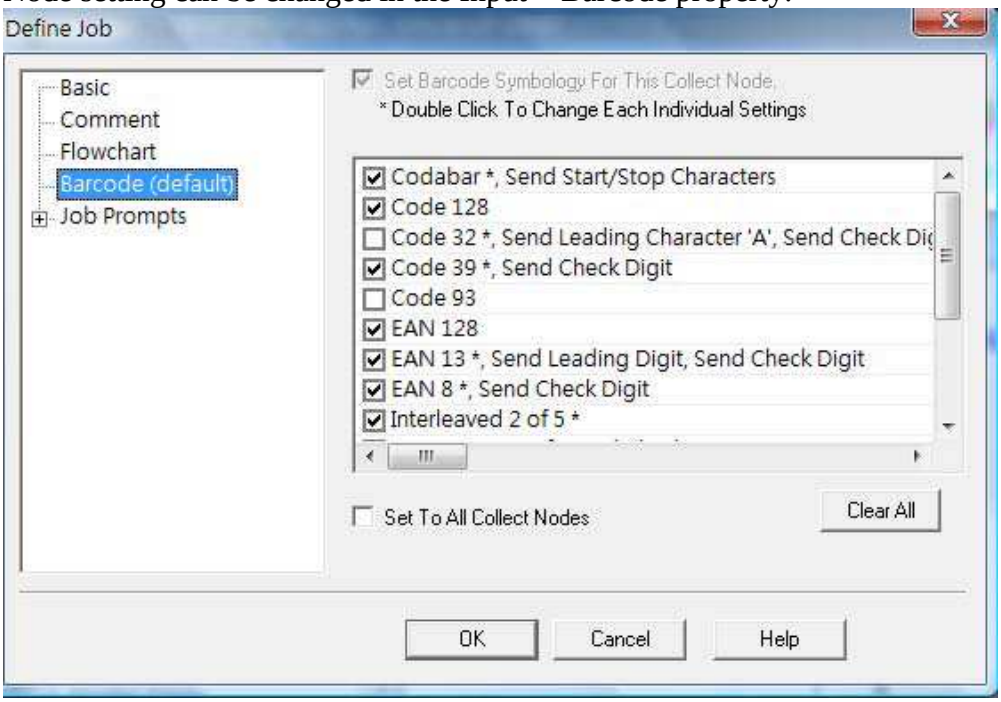
Flowchart Property:



Change flowchart page size and all node sizes.

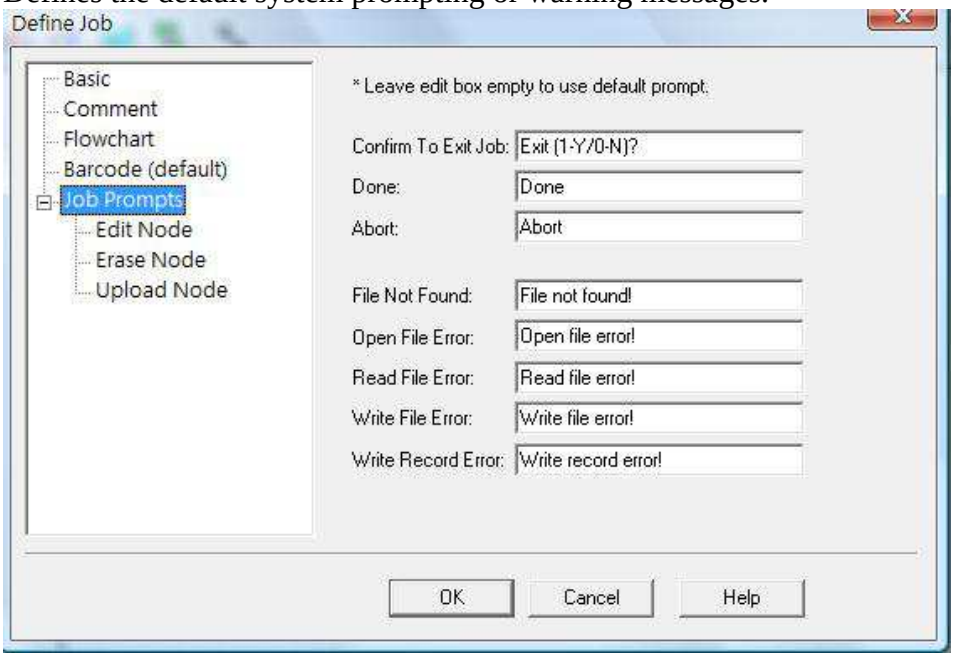
Barcode (default) Property

Defines the default barcode symbology settings for all Collect Nodes. Each individual Collect Node setting can be changed in the Input > Barcode property.



Job Prompt Property

Defines the default system prompting or warning messages.



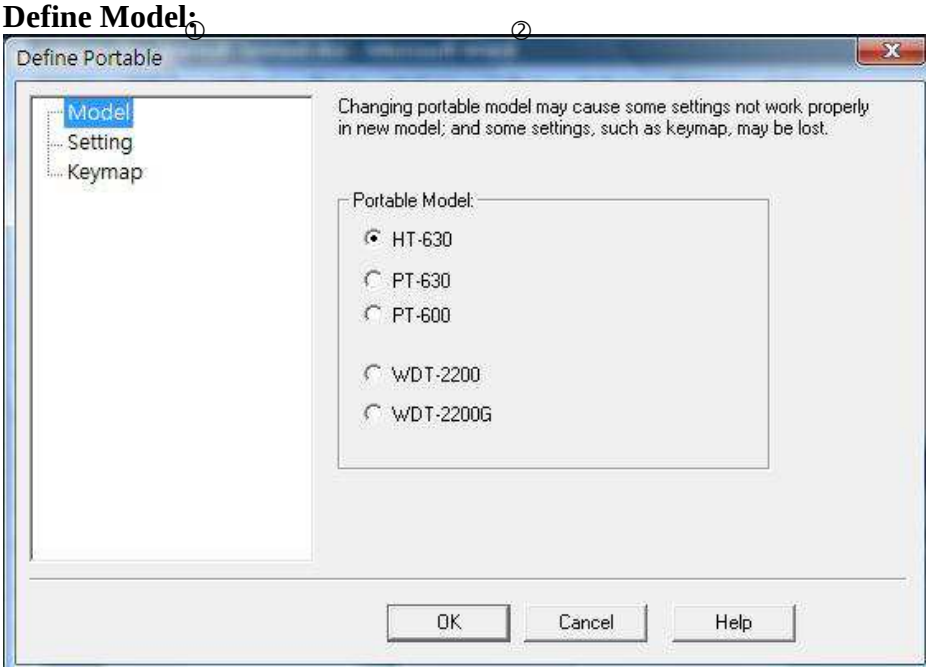
Define a Portable Terminal

Device selection is important in **JobGen Plus** because it will influence the appearance of *Menu Node*, *Collect Node*, *Math Node* and the *Links*. The size of the "Screen:" portion on *Menu Node*, *Collect Node and Math Node* need to be set exactly the same as the display of the selected device. Also, the "Key Input" verification in the *Link* will display the same keypad layout as the selected device. **JobGen Plus** also downloads selected jobs to the appropriate portables.

To select the portable model: open property item in job's property panel: Portable > Model. Double click Model item to pop up a window. Select model property again to define portable model type.

Portable Property:

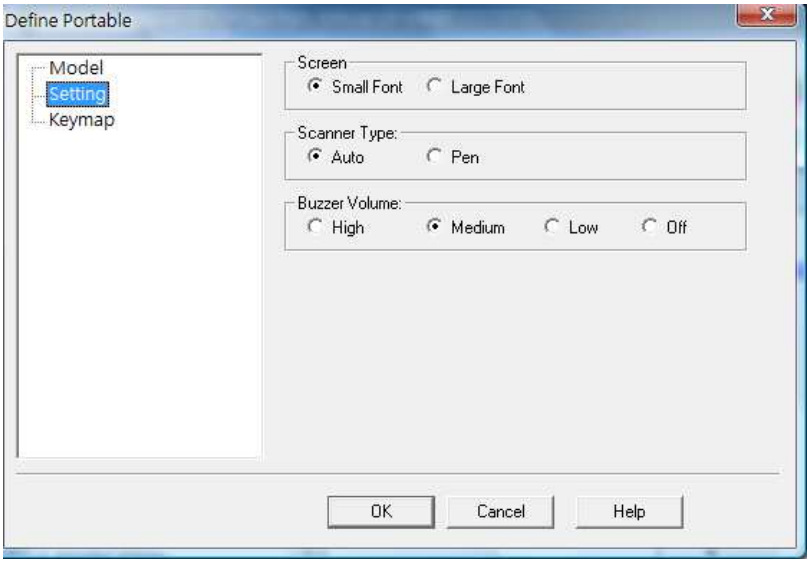
Define Model:



Portable Model	Select a target portable model type, for example: PT-600. HT-630 is the default portable model type.

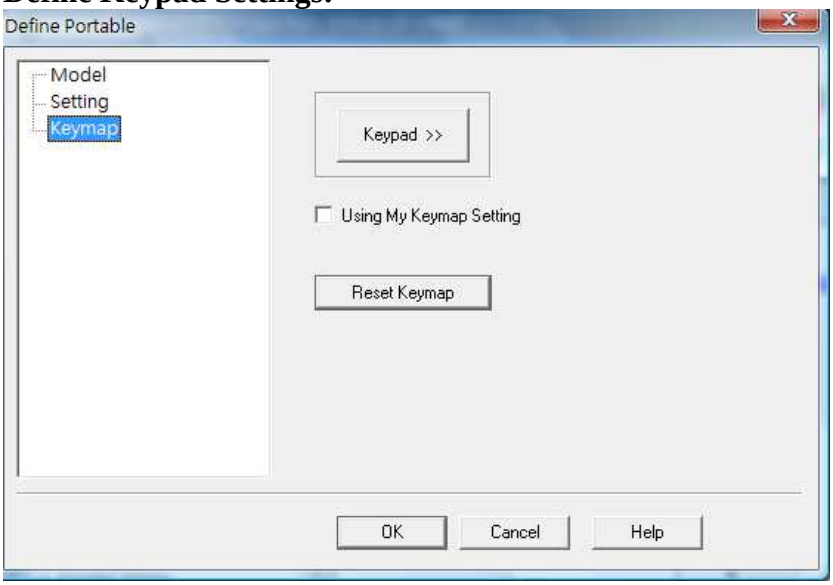
Changing the portable model type setting may affect some property settings in the Job.

Define Portable Settings:



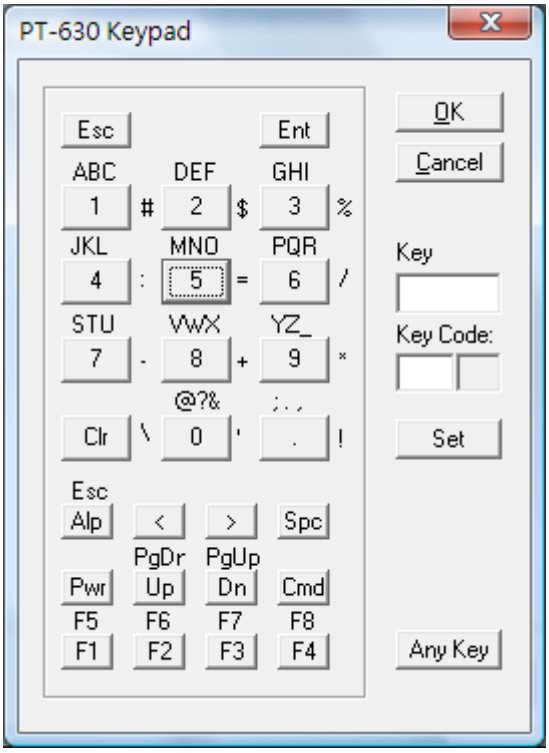
Screen	Define font size of display.
Scanner Type	Define scanner type.
Buzzer Volume	Define buzzer volume.

Define Keypad Settings:



Reset	Use portable's default settings.
Keypad	Popup a window to define key map setting.
Use My Keypad Setting	Tell job engine to apply the keypad setting.

Modify key map setting from default:



Click the key button for which you want to change the key map - the key name will show up in the key box on the right side, and the key's ASCII code with its printable character will also be displayed in the two key code box under the key name.

Change the value of a key's ASCII code to modify the key map, and click "set" to submit it. Remember, you have to check "Use My Keypad Setting" to apply this customized key map setting in Job running time.

Creating a Menu Node

The *Menu Node* provides a way to display messages. After the *Menu Node* is created, users can type in messages that will show up on the target device's[1] display.

To create a menu node: select current node type to become a menu node by clicking the menu icon on the node toolbar - or select menu item in Nodes menu. Then use the cursor to point to where you want to place the new node in the flowchart and double click. A new menu node will appear.

Or, there is an easier method: point to where you want to place the new node in the flowchart and right click to display a context menu, and select menu node. A new menu node will be created.

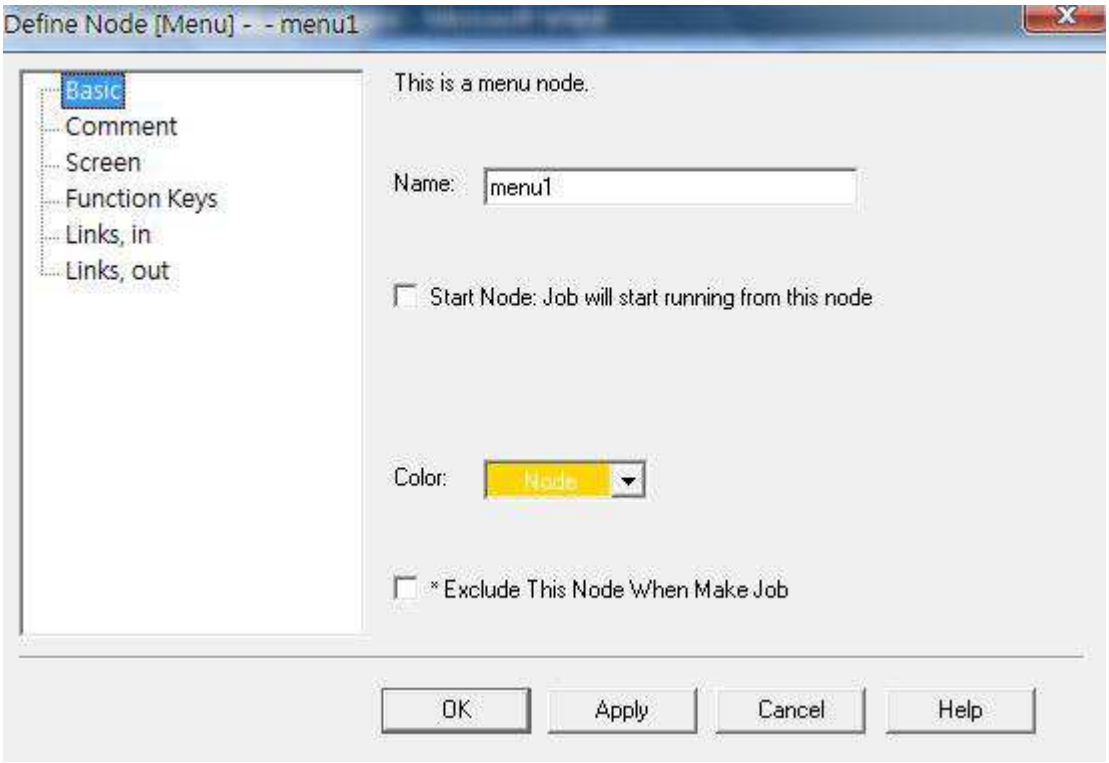
To create another type of node, the operation is same - just select a different node type.

Double click on the menu node to define its properties.

Each node has its own group of properties - and has some common properties, such as basic, comment, function keys, links in, and links out properties.

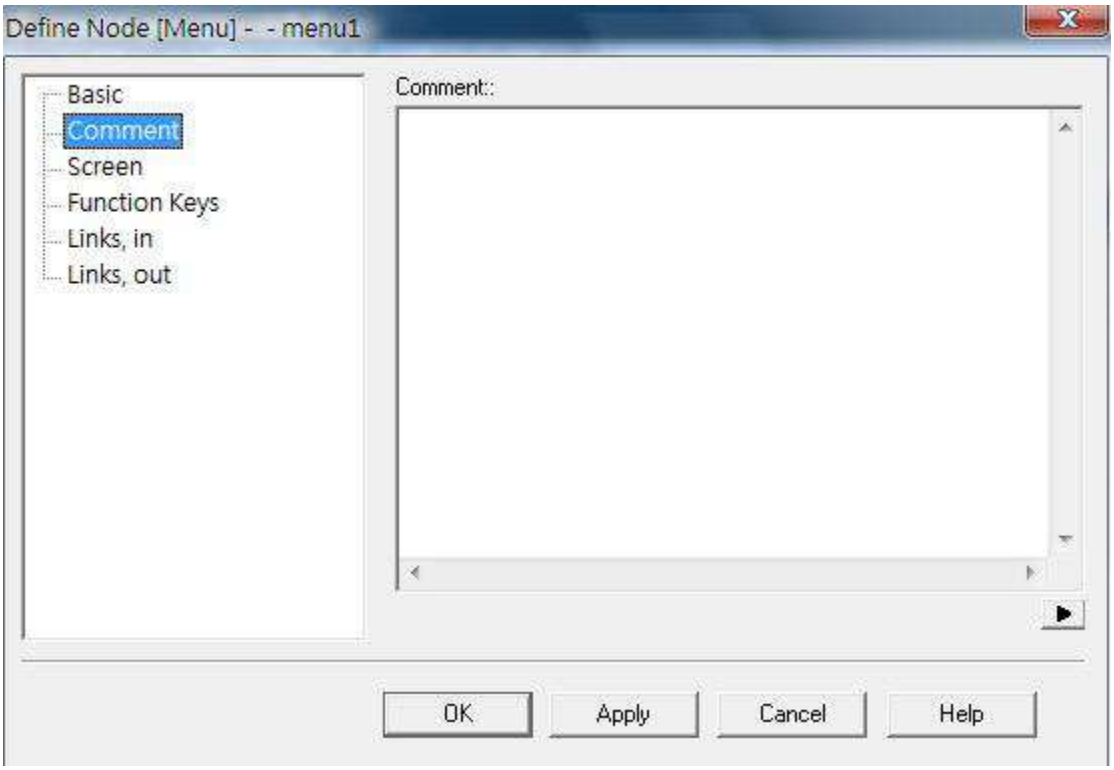
[1]A target device is the device that will run **JobGen Plus** application.

Basic properties:



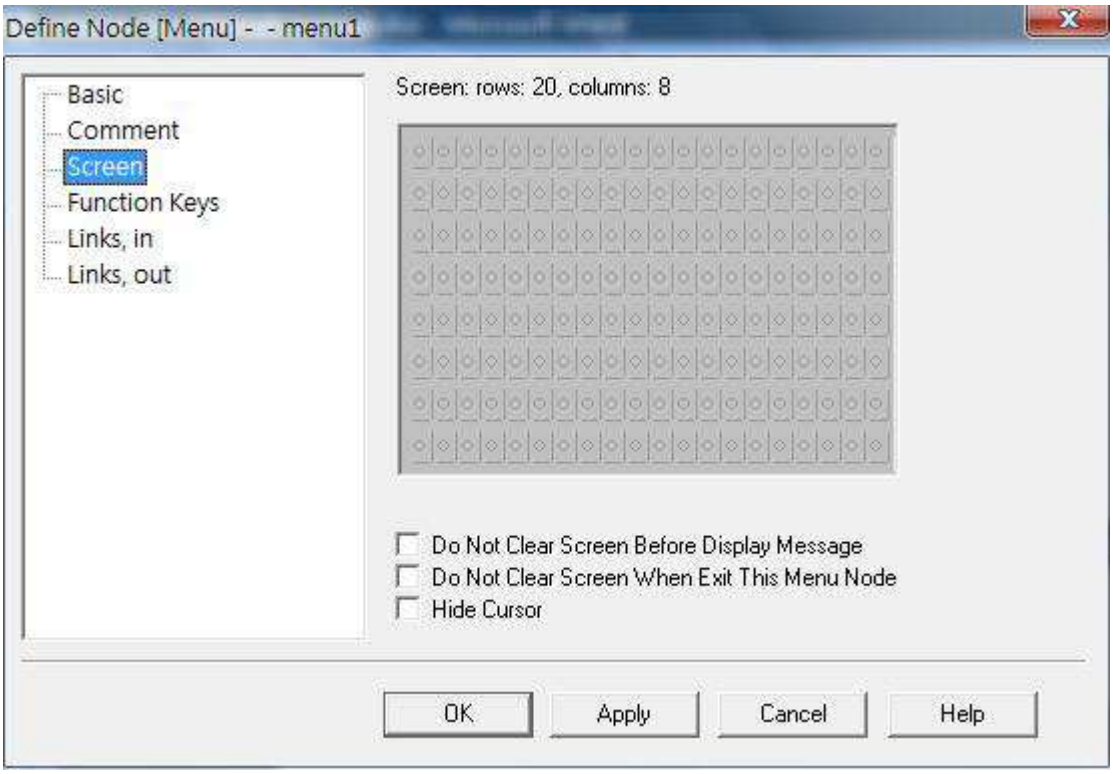
Node Name	The name of the node. Only alphabetic, numeric, and underscore characters are allowed in the name. In the case of collect or math node, because they actually hold data, add the prefix "_" (underscore) to the node name so they can be referenced by the C program in order to access the data.
Start Node	By checking this box, this node will become the starting node (the beginning point of the **JobGen Plus** application). Each **JobGen Plus** application can only have one starting node.
Color	Defines the node color.
* Exclude This Node When Make Job	Tells JobGen Plus that this node is still under construction. As its settings are not complete yet, JobGen Plus will NOT generate this node into the execution code.

Comment property:



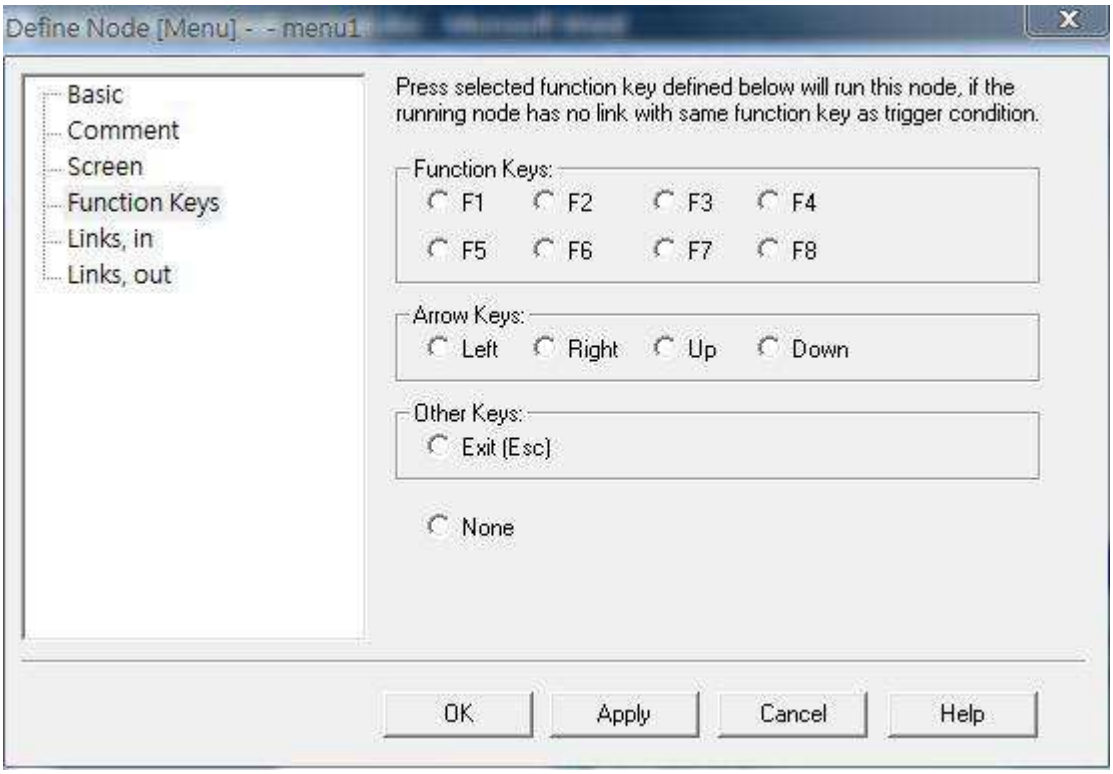
Comment Editor Box	Write comments.

Screen property:



Screen Display	This is a WYTWYS (what you type is what you will see) window. It is exactly the same display as the one in the portable. Type the text you want to display directly into the window. The dimension of the display is determined by the font setting defined in the portable properties.
Do Not Clear Screen Before Display Message	Retain messages from the previous screen.
Do Not Clear Screen When Exit This Menu Node	Retain messages to the next screen.

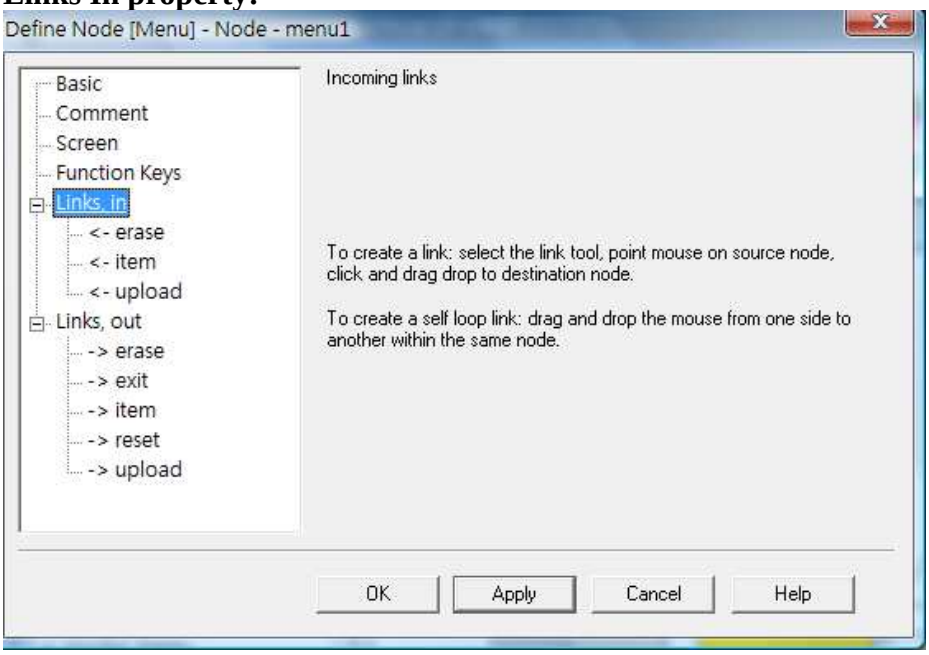
Function Keys property:



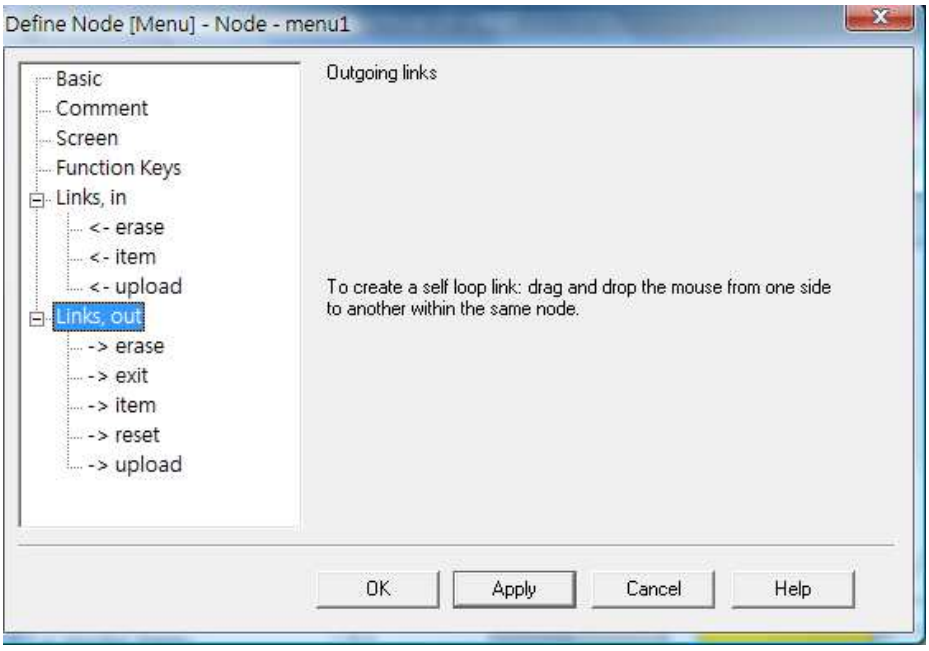
While executing a Job, pressing a function key defined for this node will direct the execution flow to this node. However, if the current node has a link with a key input condition of this function key, this link takes top priority to run to its destination node.

Defining function keys to particular nodes can significantly simplify flow chart design. While there are many links sharing the same function key input condition directing to one node, these links can be replaced by a function key definition in the destination node.

Links In property:



Links Out property:



Links In/Out Properties list all incoming and outgoing links of this node separately. Click on the link to open Define Link Dialog window.

Creating Collect Node

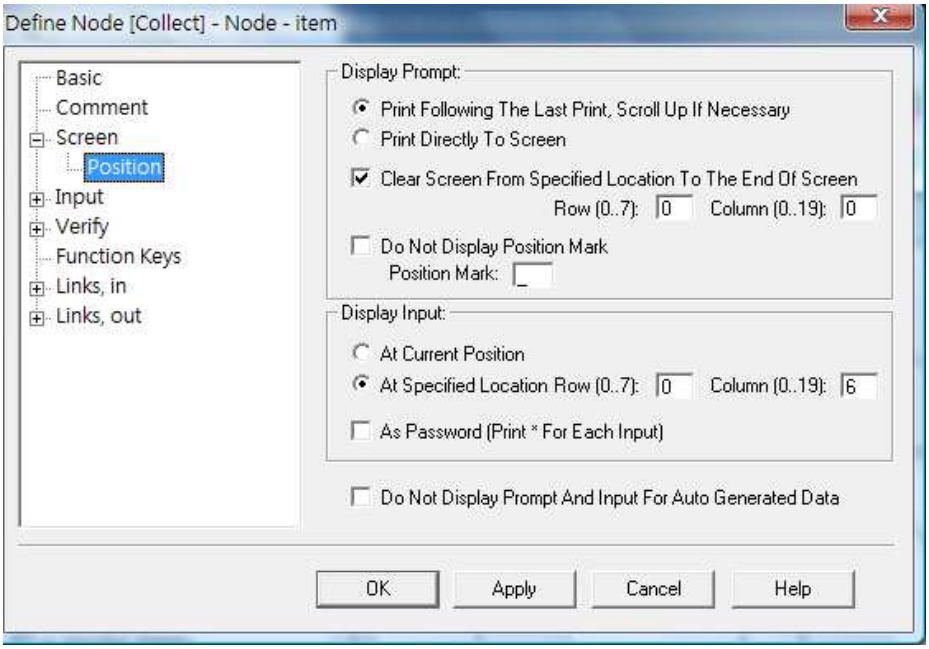
The *Collect Node* is the primary place for users to enter information. Any point in the **JobGen Plus** application that requires input from users will be represented by the *Collect Node.*

Define Collect Node

Basic Property.
Comment Property.
Screen Property.
Input Property
Verify Property
Function Keys Property.
Links In Property.
Links Out Property.

These properties are as same as the ones in menu node.

Screen > Position Property.



Display Prompt:

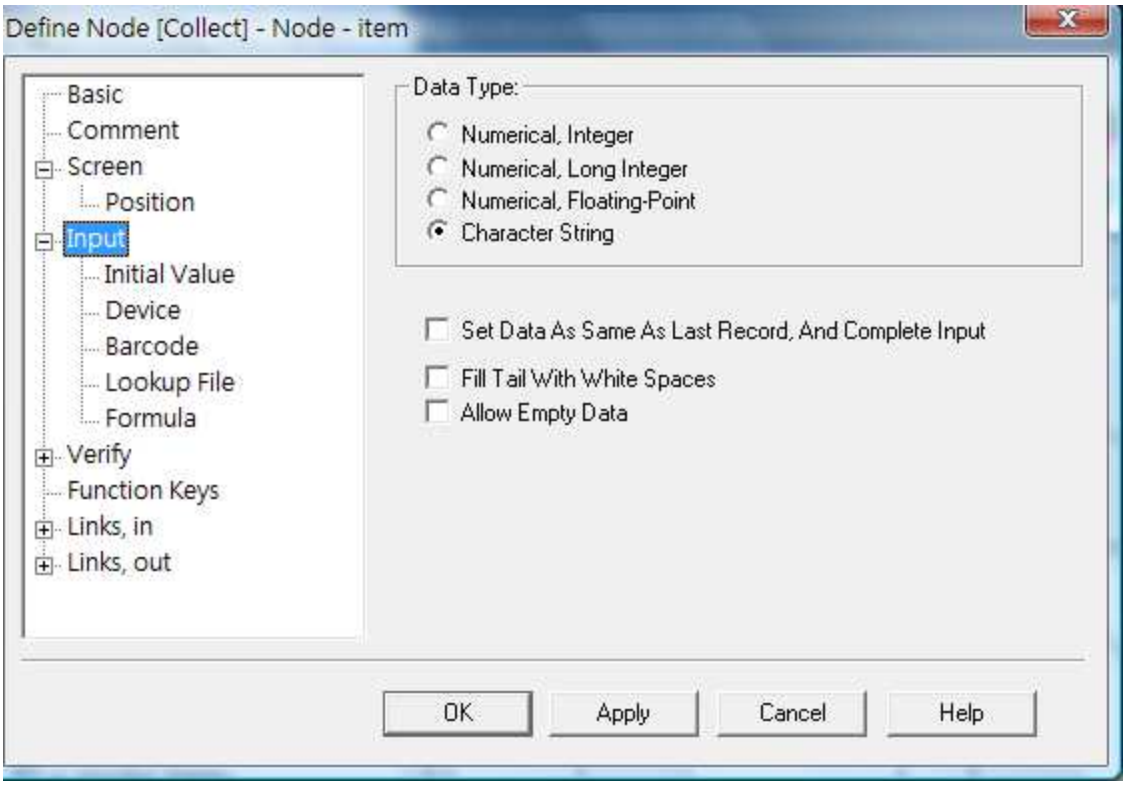
Print Following The Last Print, - Scroll Up If Necessary	Prints line by line at the current cursor position, then scrolls up the screen.

Print Directly To Screen	Print text displayed on the screen to the portable screen. All the text should be at its designated position.
Clear Screen From Specified Location To The End Of Screen	Clear portion or entire screen before print. Specify row and column position in row and column box.
Do Not Display Position Mark	At the start of a new input, the previous input will become blank. The now current input will then remain on the screen until a subsequent input begins.
Position Mark	Alphabetic characters used to occupy blank spaces for data input. Once users enter data, these characters will be replaced by the input data.

Display Input:

At Current Position	Print at the current cursor position.
At Specified Location	Print at the position specified in row and column. Specify row and column position in row and column box.
As Password	Hide input by printing * instead.
Do Not Display Prompt And Input For Auto Generated data	Input enters without interaction with user.

Input Property.



Data Type

All data are two basic data types: numerical or character string. Numerical data can be further divided into three types: integer, long integer and floating-point.

Numerical data can be used to perform math calculations. The three different numerical types differ in their limits and if they can accept fractions.

To represent a constant long integer, add 'L' at the end of the number. For example: 100000L. (100000 exceeds limit of integer.)

To represent floating-point, always include input to the right of the decimal point, for instance: 10.0.

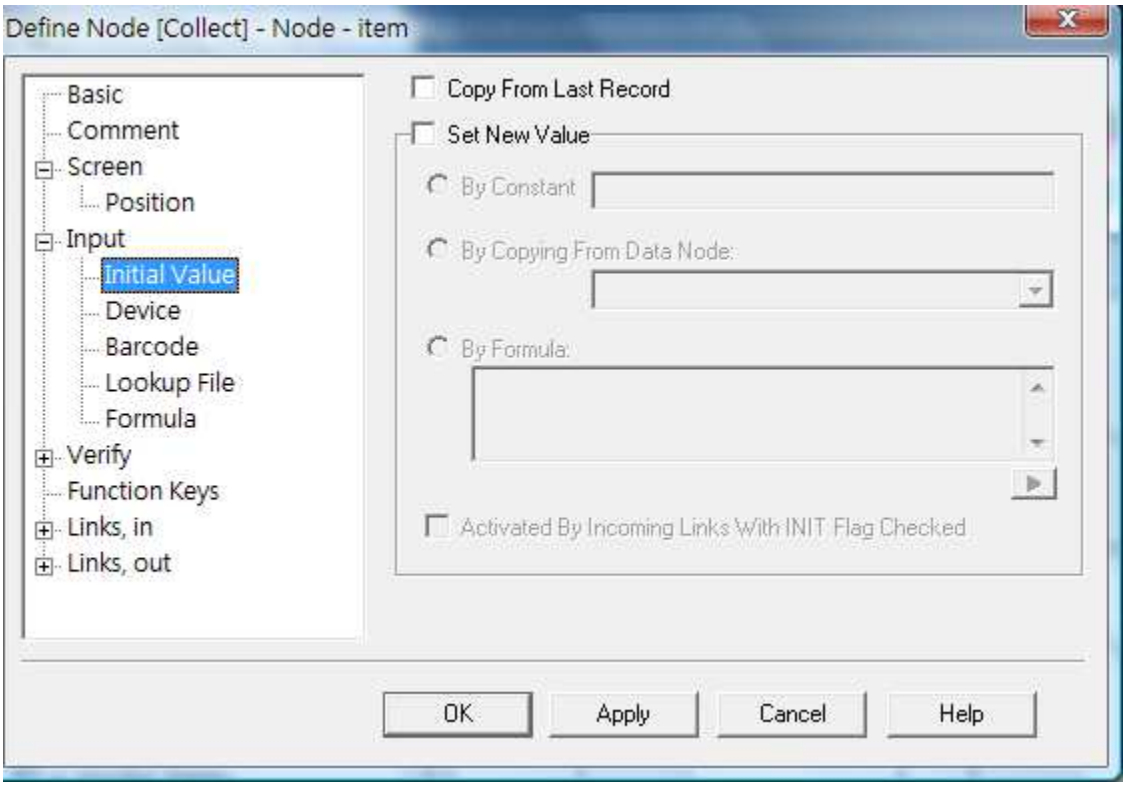
Character string is pure text.

Set Data As Same As Last Record, And Complete Input	The data is prepared with (copied from) the data from the last record, and will complete the input automatically at once. In other words, the new input will be exactly same as in the previous record.
Fill Tail With White Spaces	Automatically fills tail of data with white space character (ASCII code: 0x20 in hexadecimal). This is to generate fixed-length data.
Allow Empty Data	Enables skip of data input. The value of this particular field in the record will be null, which means no data. By default, all data should include one or more characters.

Integer Limits

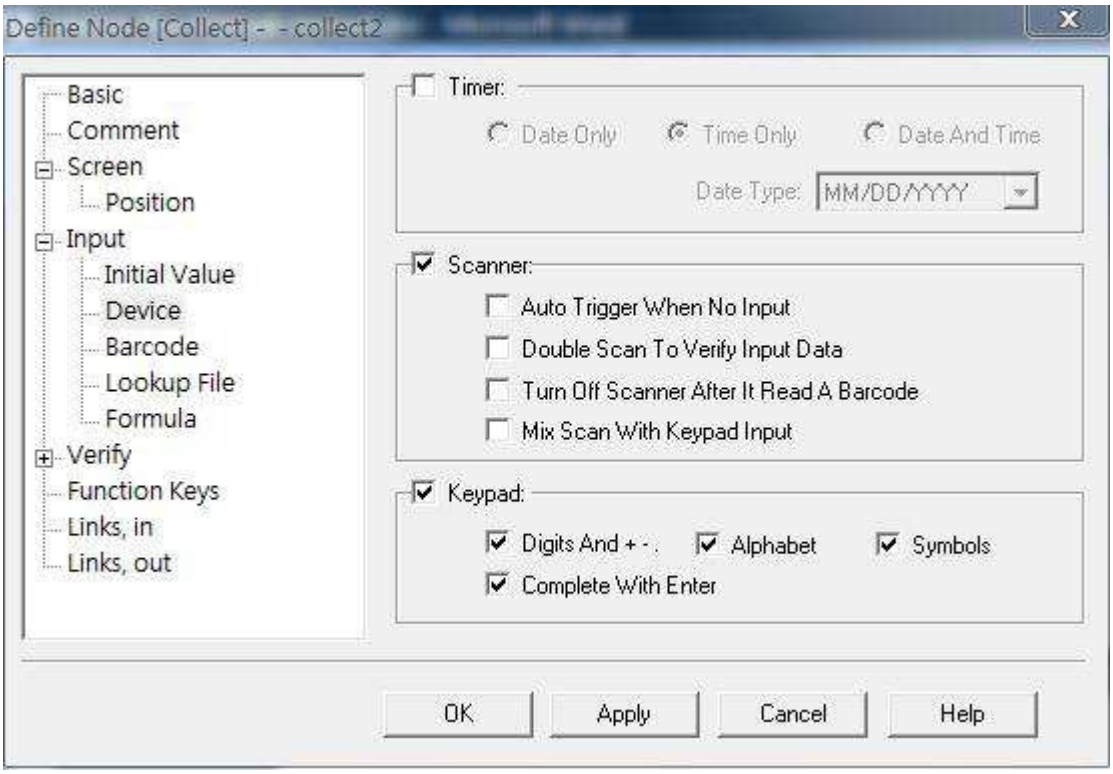
Integer:	-32767	to	32767
Long integer:	-2147483647	to	2147483647
Floating-point:	1.175494351e-38F	to	3.402823466e+38F

Input > Initial Value Property.



Copy From Last Record	Data value will be prepared with (copied from) the data from the last record. The data can then be modified. This option is used when you want to inherit data from last result.
Set New Value	
By Constant	Assign a constant to the data.
By Copying From Data Node	Copy data from another data node (collect node or math node). Note, only the collect nodes of numerical types can be selected as a data source.
By Formula	Use a formula to assign a value.
Activated By Coming In Links With INIT Flag Checked	The initial of *Set New Value* will be processed only when it has been activated by a Coming In Link with INIT checked.

Input > Device Property.



Three methods of data input are available to portable devices: Clock, Scanner and Keypad. When multiple selections are possible, the actual input can come from only one device at a time.

- **Timer**

Automatically get data from clock and completes the input. Three data formats are available. They are date (MM/DD/YYYY), time (HH:MM:SS) and date-time (MM/DD/YYYY HH:MM:SS).

- **Scanner**

Enables scanner for input. Barcode symbologies can be set in the Barcode property page . Individual collect nodes can have their own settings for barcode symbologies. The default barcode symbology settings for all collect nodes in a job is defined in the Job Define, Barcode (default) property page.

Note: if the data buffer is not empty (for example, because characters are input through the keypad), new scanning data will be rejected. Press the Clear button to clear all input and then scan again.

- **Turn Off Scanner After It Reads A Barcode**

  Check this option to turn the scanner off after it successfully reads a barcode. You don't have to worry about when to turn it on – the collect node will automatically turn the scanner on if it's one of the selected input devices.

  Usually the scanner will be left in ON status after it reads a barcode in collect node, so that it will prevent the subsequent collect node from scanning data without a release and repress of the scan trigger. That means a continuous scan cannot occur. Each scan must be activated by pressing the trigger. However, leaving scanner in ON status may cause incorrect scanning at none data input time. Job engine will automatically turn off the scanner if the current running node is not a collect node or math node or program node.

- **Keypad**

Enable keypad for the input.

- **Digits And + - .**

  Only digital keys: 0 – 9, decimal point:., and sign symbols + and – are acceptable as key input.

- **Alphabet**

  Only alphabet keys, both upper and lower cases: a – z, A – Z, and space are acceptable as key input.

- **Symbols**

  Only symbols, such as $, #, !, etc, are acceptable as key input.

- **Complete With Enter**

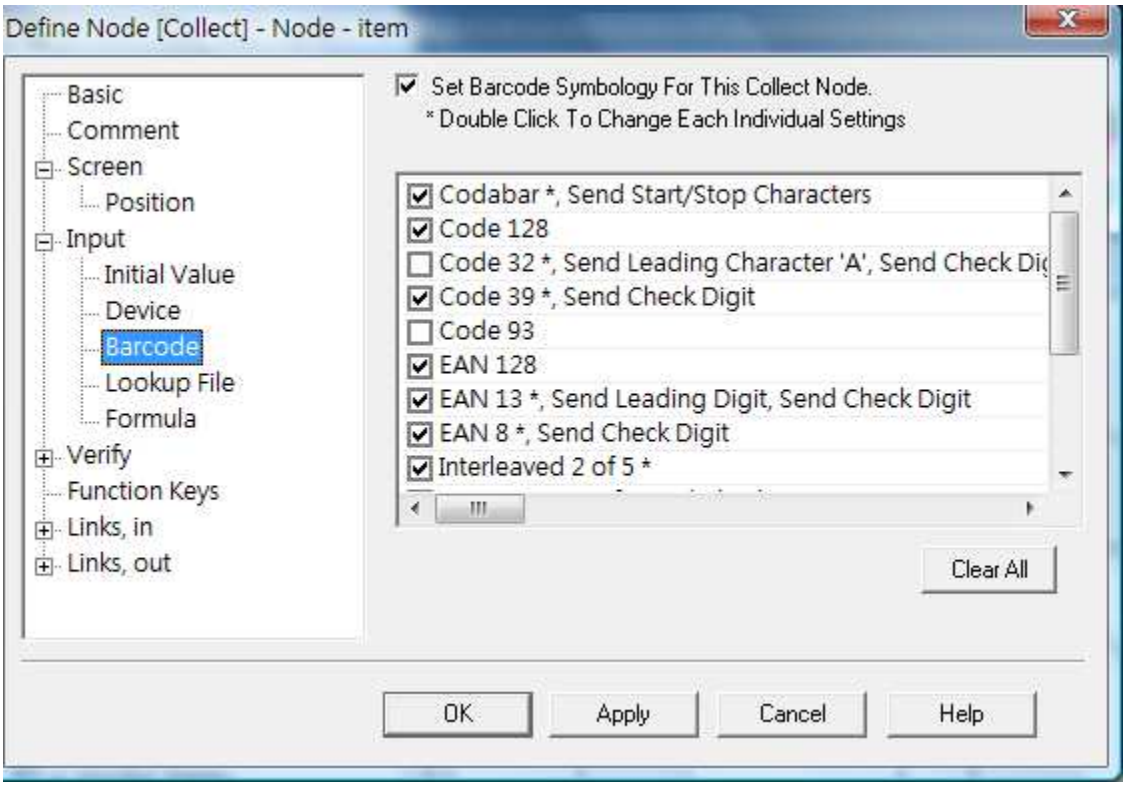
  To complete the input, the Enter key must be pressed. If it is not checked, then input will finish automatically when input reaches the maximum length.

Keys are divided into three categories: normal keys (such as 0 to 9 numerical keys, and A to Z alphabetical keys), function keys (such as F1 to F8, and arrow keys) and special keys (such as CMD, scan and power keys).

Function keys have a special function in data input operation. The input operation could be canceled if a function key is pressed that is defined for a node, or in an out-going link of the current node. Normally, function keys are used to change the operation procedure depending upon special needs.

Normal keys and function keys can be re-mapped to other keys by redefining its code in keymap .

Input > Barcode Property.

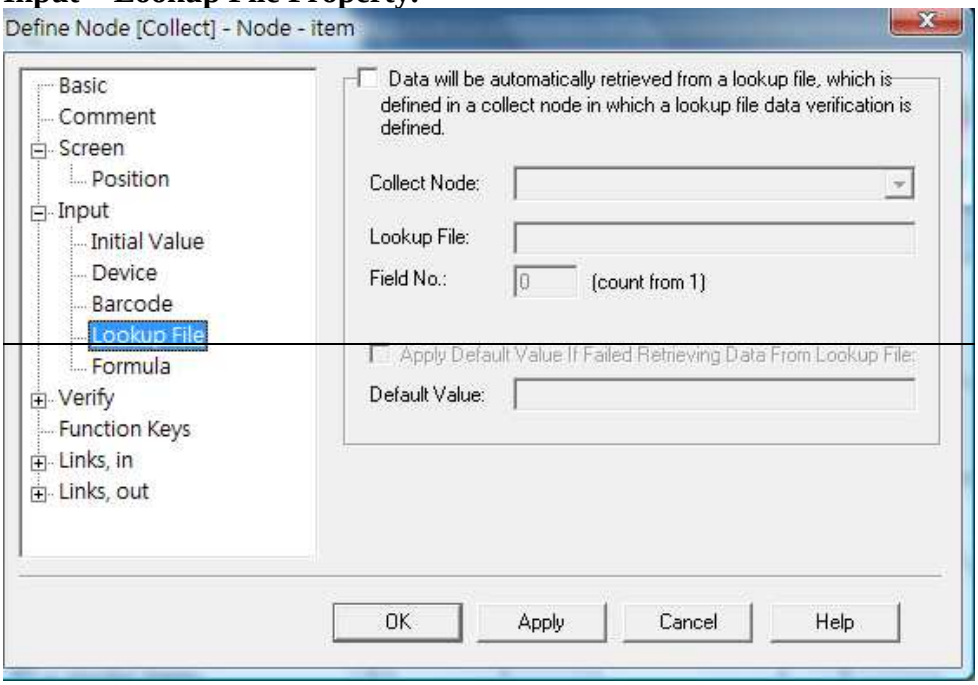


When selected, each individual collect node can have its own barcode symbology settings, so that different collect nodes may have different scanner behavior.

The barcode whose name is followed by an asterisk (*) requires its own detailed settings. Double click on the name to open the setting window. Checked sub-settings will be printed next to the name.

Default barcode symbology settings are made in the Job Define, Barcode (default) page.

Input > Lookup File Property.

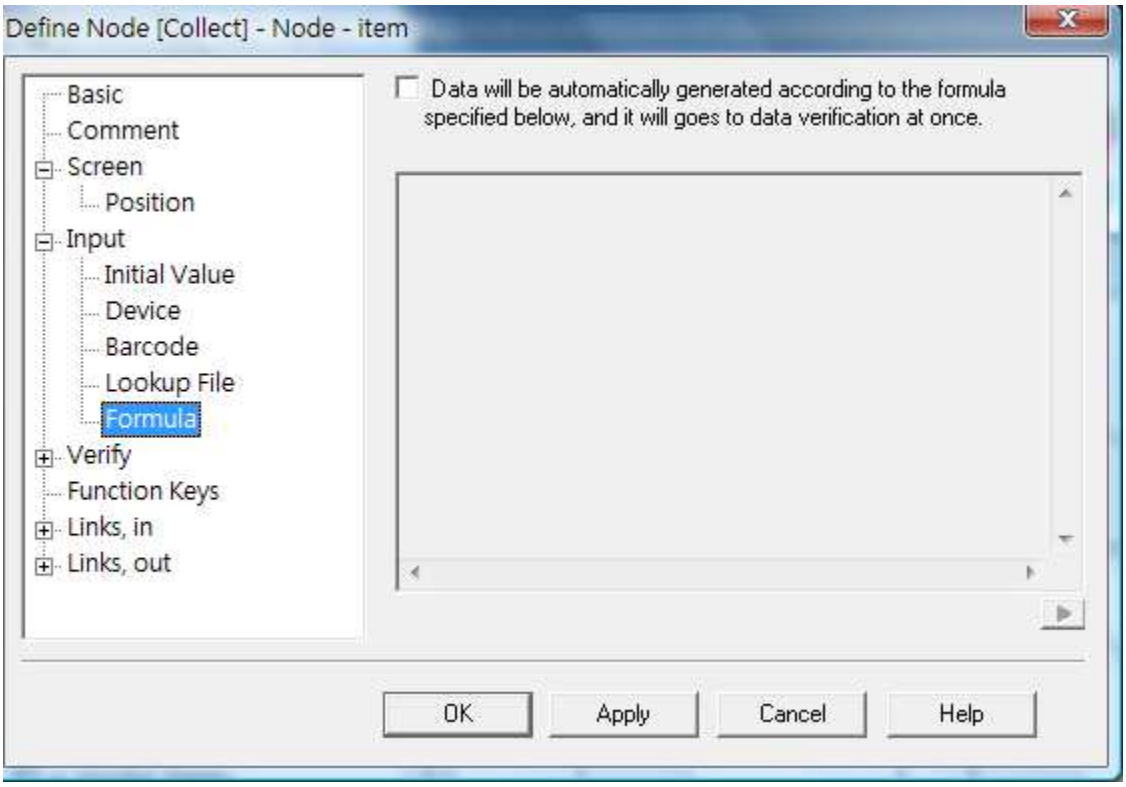


In many cases, you need to retrieve data from a lookup file based upon a previous success in lookup verification, so that you can process data on a related data basis.

Select the name from among all the collect nodes that perform lookup verifications in order to define the lookup file. Note that it does not specify the lookup file directly because it requires a successful lookup verification. A field number is then specified. The record number is determined at job runtime by the matched record number from the lookup verification of the node.

If the collect node fails the lookup verification, then the input from the lookup file cannot be completed, and the input method will be changed to keypad input.

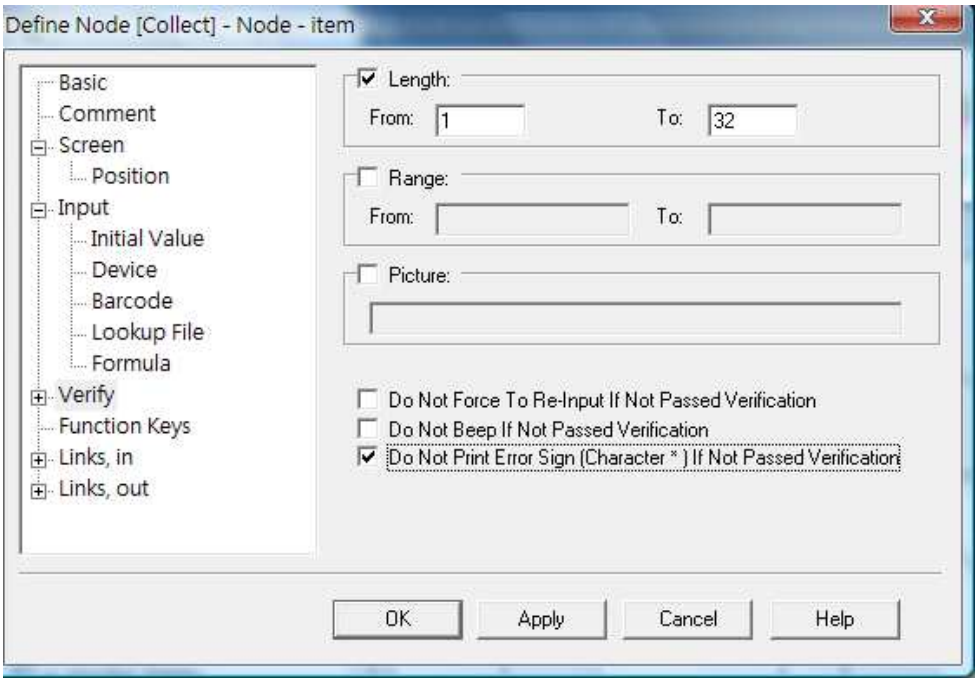
Input > Formula Property.



Specify a C expression as the formula to generate the data.

A collect node actually holds a variable of data that can be accessed by others. To access it, simply use its variable name, which is its node name with the prefix of underscore ('_').

Verify Property:



Length	Specify the length (number of characters) of the input data.
Range	Specify the value range of the input data. This is compared by its character's ASCII codes.
Picture	Defines a mask pattern of input character's type.

Do Not Force To Re-Input If Not Passed Verification	Normally, basic verifications, length, range and picture, are treated as basic checking for the input data. The input data must meet these three conditions, if any, before proceeding with the process. Otherwise, the user is forced to re-input until the data is passed correctly.
Do Not Beep If Not Passed Verification	If verification fails, a buzzer will sound to warn user. Check this option to turn off the buzzer when verification fails.
Do Not Print Error Sign (*) If Not Passed Verification	If verification fails, no error sign of wrong data input will be printed on screen.

Picture Pattern:

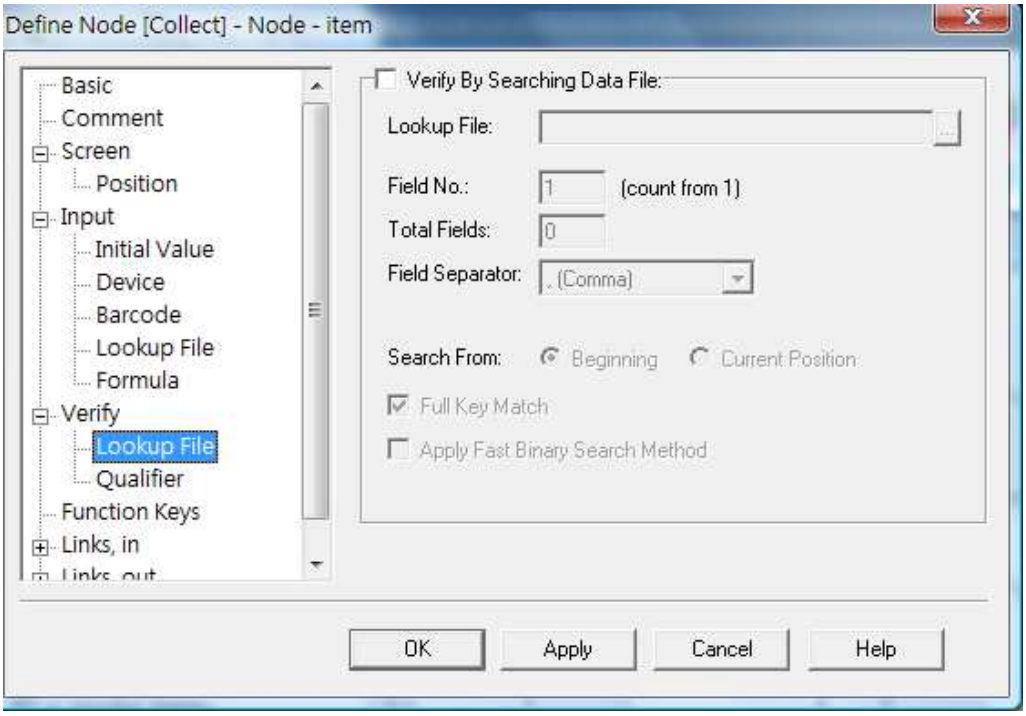
a	only alpha characters are accepted (A-Z, a-z)
n	only digits(0-9) and '+', '-', '.' (dot), 'E' and 'e'
0	only digits (0-9) and '+', '-'
p	only digits (0-9)
l	only lower case alpha (a-z)
u	only upper case alpha (A-Z)
x	printable characters (0x32--0x7f)
z	full ASCII codes
_	full ASCII codes
#	remove a characters
@	constant string between @ and @
%	indicator, append symbols in the pattern
?	indicator, non-fixed length of the pattern
!	indicator, reverse the verify result
(space)	terminator of the pattern

Example of picture verifications:

Picture	**Data Enter**	**Result**
ppp-pppp	1112222	1112222
ppp-pppp	111-2222	1112222
ppp-pppp	111.2222	***error***
ppp-pppp	11-22222	***error***
aaa-aaaa	1112222	***error***
#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	***error***
$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	$111.
?p	1	1
?p	111	111
?ppp	11	***error***
?aaap	abc123	abc123
?aaap	abc	***error***
?%$ppp	123	$123
?%$ppp	12345	$12345
?%$ppp	12	***error***

Verify > Lookup File Property:



This is a useful function that is used to check the input data with a data file (database).

Field No. starts counting with 1. If it is passed, the *success flag* used as link condition will raise. Otherwise the *fail flag* will raise.

JobGen Plus will download the lookup file automatically with job files. If it is a big file and remains unchanged while developing the job, then there is no need to download it every time. Disable auto download lookup file in Define Job [default settings] , Make Job page.

- **Search From**

Search data file either from the beginning or from the record following the last matching record.

- **Full Key Match**

Specifies full data matches. For example:
“abc” is full match with “abc”;
“abc” is not full match with “abcdefg”;
“abc” is not match with “1abc23”.

- **Apply Fast Binary Search Method**

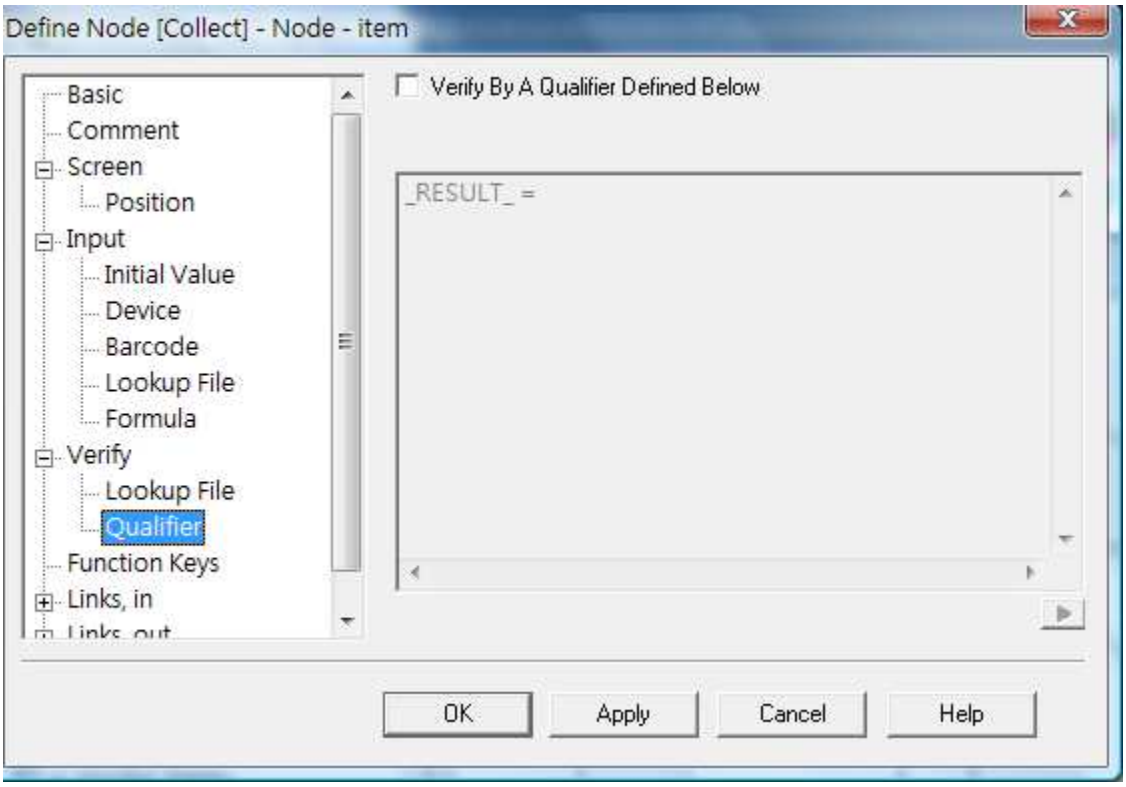
When searching larger databases, such as those containing hundreds of thousands of pieces of data, check this option to enable really fast search. JobGen Plus will automatically generate an additional file - the binary search table - and download it along with the original data file and job file.

The drawback of binary search is that the data file cannot be changed at run-time. Changes in the data file may corrupt corresponding binary search table files, which actually records the exact position of all records in data file. With incorrect information in binary search table, the search will return unpredictable results.

However, by adhering to certain rules, it is possible to change some data in data file while doing binary search. The rules are:

- Key field can never be changed.
- Length of record must be constant. Each record may not necessarily be same length. After modifying a record, insure that the actual length of the record remains unchanged.

Verify > Qualifier Property:



This is a C expression. Set the result to predefined variable _RESULT_ which is true if its value is non zero, and otherwise is false. If it passes, the *success flag* used as link condition will raise. Otherwise the *fail flag* will raise.

Creating math Node

The Math node is used to process numbers or simply to set a constant number to a data field. Both collect node and math node are data nodes, which actually holds data and can be selected as a field of record.

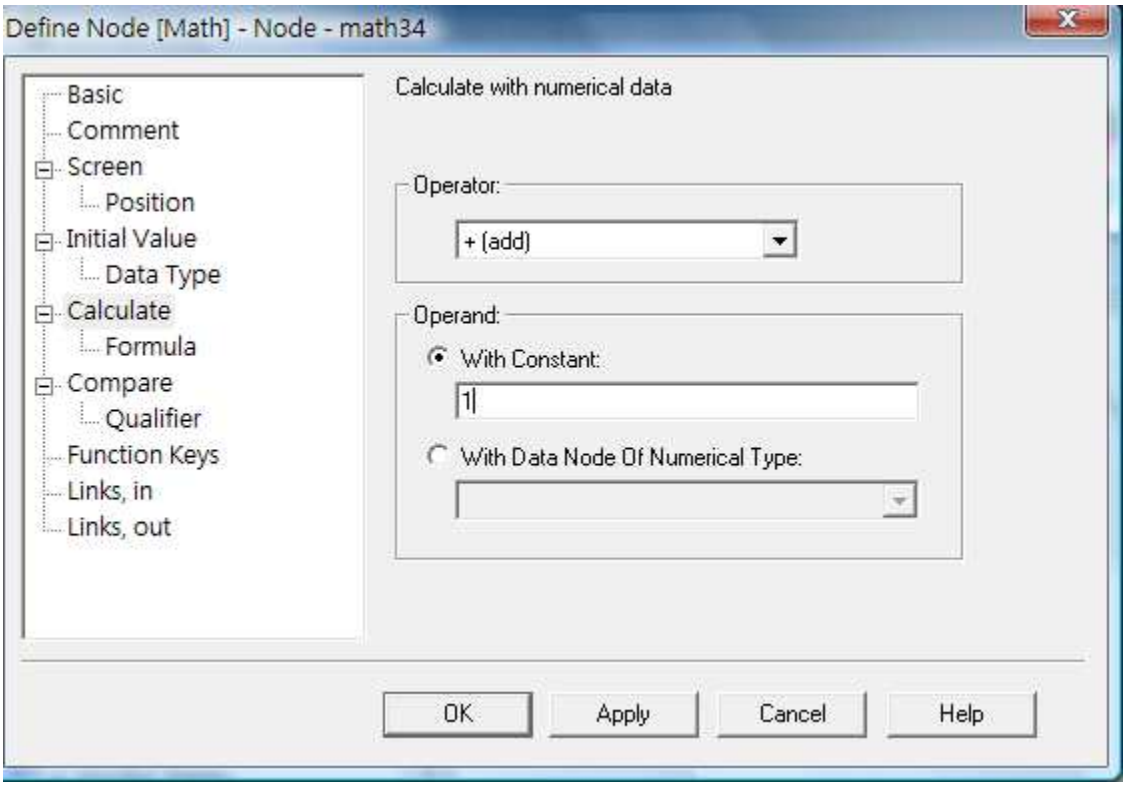
Basic Property.
Comment Property.
Screen > Position Property.
Initial Valus
Calculate > Formula Property.
Compare > Qualifier Property.
Function Keys Property.
Links In Property.
Links Out Property.

These properties are as same as the ones in collect node.

Initial Value > Data Type Property.

Math node can only have numerical data, as it is intended to do calculations. When different data types work together, the result will be automatically converted to large data type.

Calculate Property.



Define operator and operand of the calculation.

- **Operators:**

Add, Minus, Multiple, Divide, and None.

- **Operands:**

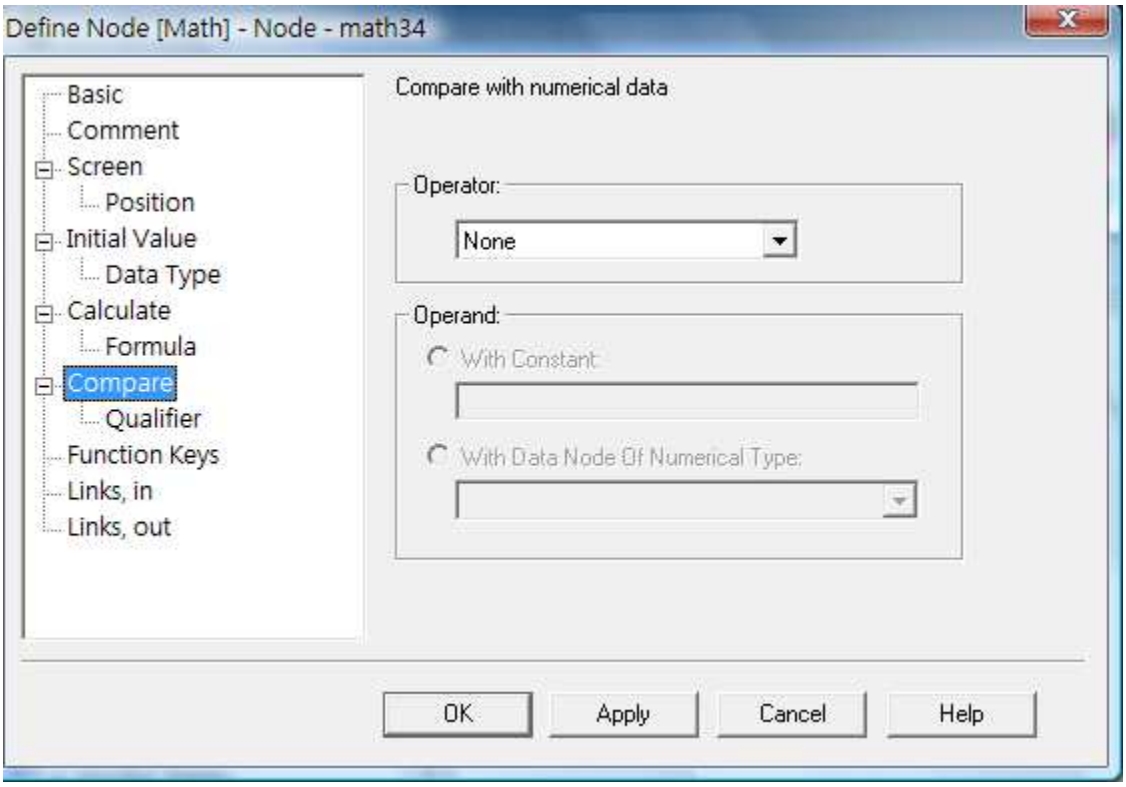
  - **With Constant:**

    Assign a constant as the operand.

  - **With Data Node:**

    Gets the data from a data node, which can be a collect node or a math node. Note - only the collect node of numerical data types can be selected here.

Compare Property.



Defines operator and operand of the comparison. The result of comparison can be used as the Success or Fail condition of links.

- **Operators:**

  Less, Less or Equal, Greater, Greater or Equal, Equal, Not Equal, and None.

- **Operands:**

  - **With Constant:**

    Assign a constant as the operand.

  - **With Data Node:**

    Get data from a data node, which can be a collect node or a math node. Note, only the collect node of numerical data types can be selected here.

Creating Edit Node

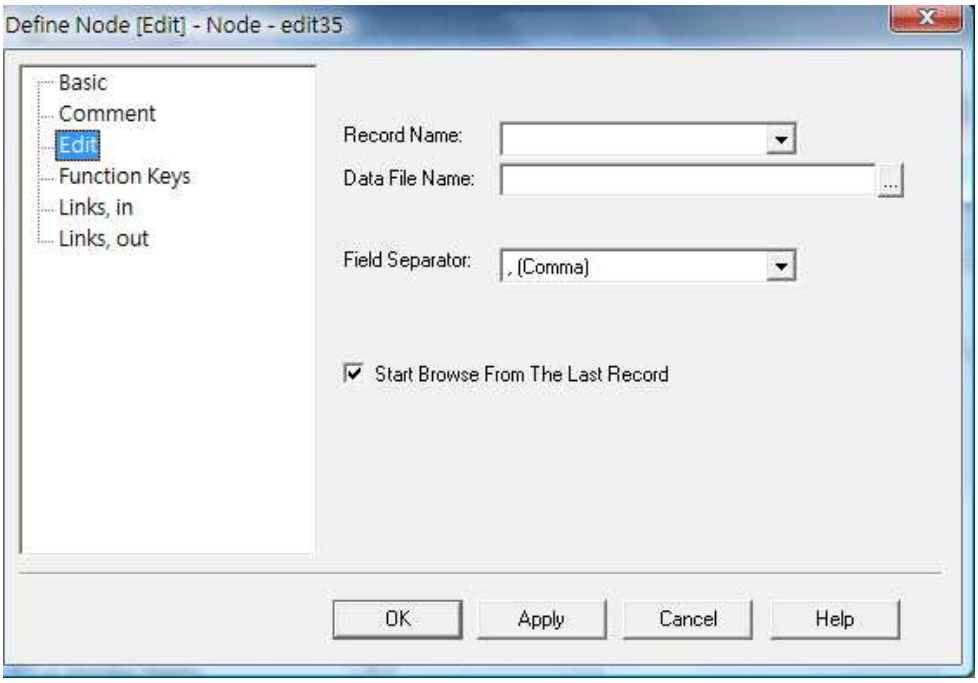
Edit Node is used to view and modify the data contained in the data file in the portable at run time.

Define Edit Node

Basic Property.
Comment Property.
Function Keys Property.
Links In Property.
Links Out Property.

These properties are as same as the ones in menu node.

Edit Property.



Record Name	Select a record to define the data file name.
Data File Name	Define the file, which will be operated in edit node at run time.
Field Separator	Specify the field separator (delimiter) that is used in the data file.

Edit node has several different working modes. It enters into **Browse mode** at the start time and displays the data in the first field of the first record. The screen prompts the user with "B,R:xx,F:xx", in which B refers to Browse mode, and R: and F: refer to the coordinate numbers of the record and field respectively of the current field.

Press Left or Right arrow keys to go backwards and forwards through fields in a row (record). Press Up (or **F1** function key) or Down (or **F2** function key) to go through fields in a column (fields).

Press **F3** function key to input new R: and F: coordinates in order to go (jump) to a field directly. The number in parentheses after R is the total number of records in the file.

In Browse mode, press Enter to enter the **Edit mode**, which will change the prompt to "E,R:xx,F:xx". E refers to Edit mode. Only in the edit mode can the user modify the field data. Press Enter again after modifying the data in order to save the changes to the file. Selecting edit node will return to Browse mode.

While changing data, the Left arrow key acts like a backspace key, deleting a previous character. Pressing the Right arrow key will restore all the deleted characters. This function makes modification easier because you don’t have to remember the contents of the data.

Press **F4** function key to enter into **Search mode**. Search starts from the current record looking FORWARD for the column defined by the current field number. The prompt displays "S,R:xxx,F:xxx", in which R indicates the start record number, and F indicates column number. After a prompt it waits for user to input data (default value is currently browsed data) to search.

Search will match a field if partial data from the beginning is exactly same as the input data. After successfully finding data, it will ask SEARCH NEXT? A YES answer causes it to search continually starting from the next record, or a NO stops the search and switches back to the browse mode automatically to display the matching field. If the data is not found, the prompt "Not found" will be displayed, and then returns to wait for an input for a next search. Press EXIT to quit from search mode.

Press **Exit** key in the Browse mode to quit from the edit node.

Creating Erase Node

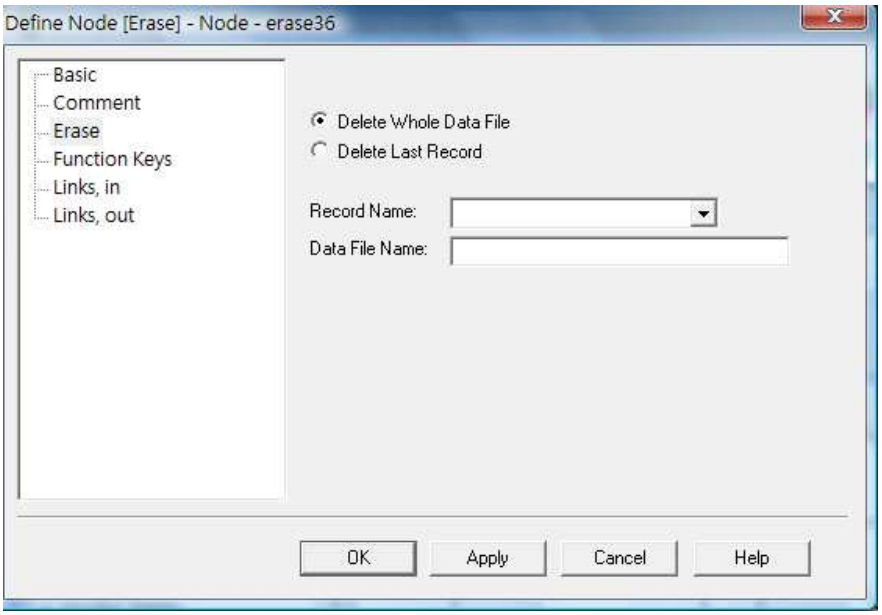
Erase Node is used to delete records from the data file, or the whole data file.

Define Erase Node

Basic Property.
Comment Property.
Function Keys Property.
Links In Property.
Links Out Property.

These properties are the same as the ones in menu node.

Erase Property.



Delete Whole Data File	Delete whole data file from portable.
Delete Last Record	Delete last record from the data file.
Record Name	Select a record to define the data file name.
Data File Name	Define the file, which will be operated in erase node at run time.

Creating Upload Node

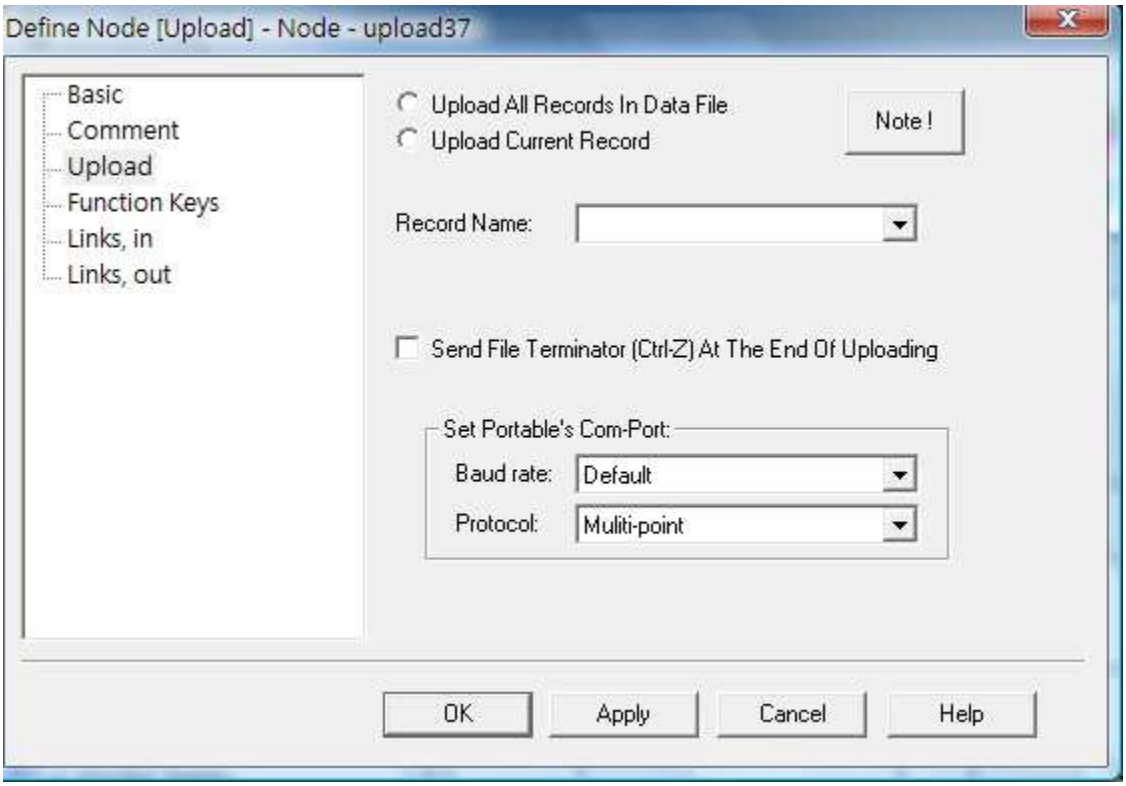
Upload Node is used to send data file from the target portable to the host computer.

Define Upload Node

Basic Property.
Comment Property.
Function Keys Property.
Links In Property.
Links Out Property.

These properties are as same as the ones in menu node.

Upload Property.



The function of the upload node is to transfer data from the portable to the host with ***multi-point protocol***. On the execution of an upload node, all the data defined to be sent will be sent immediately to the host application which is continuously polling the data.

Upload All Records In Data File	Upload all the records in the file.
Upload Current Record	Upload the current record.
Record Name	Select a record to upload. The record defines which fields to upload.
Send File Terminator At The End Of Uploading	Send a file terminator (Ctrl-Z) at the end of uploading.

Creating Program Node

Program Node provides the space for users to write C programs.

Define Program Node

Basic Property.
Comment Property.
Function Keys Property.
Links In Property.
Links Out Property.

These properties are as same as the ones in menu node.

Basic Property.

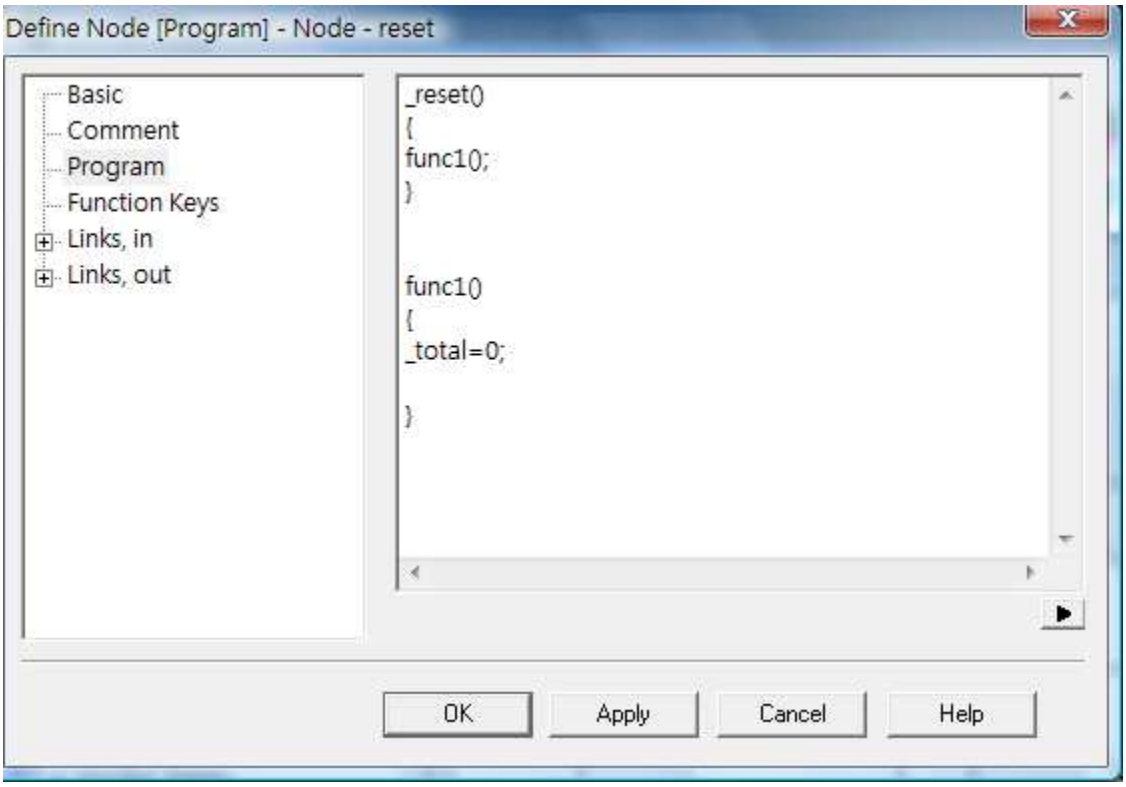
Program node has an additional selection in basic property.

- **Global Access**

Allows the functions defined in this program node to be accessed (called) by others, for example, called by another program node, in input formula, in verify qualifier, or in link expression. All global access program nodes are listed under Job \ Common Define entry in left properties panel.

A C function must be defined before it can be called. JobGen Plus will always put global access nodes before all other nodes when generating the code, and they will be processed in alphabetical order.

Program Property.



Program node actually provides a place where you can write your own functions in C code. You can write code in the editor box, or click the arrow button at right bottom to open a larger editor window. Simply close the window to return back.

Each program node has an entry of a function, which will be called when executing this program node. The entry is the name of the program node proceeded with the prefix underscore ('_'). For example, if the program node name is "ring", then a function with the name "_ring()' will be called when executing this program node.

Many functions can be called in program code. They are divided into three categories: job-engine functions, some standard C library functions, and firmware functions.

Check Function Library in *Help Contents* for detailed descriptions of job-engine functions. Check Run-Time Routines in *Microsoft C Language Help* for detailed descriptions for standard C functions. Refer to portable's *Technical Binder* for detailed description for firmware functions.

Creating Run-Job node

One job can be run within another job. Use this technique to divide a large job into several smaller independent ones (modules). Smaller Jobs mean simpler maintenance.

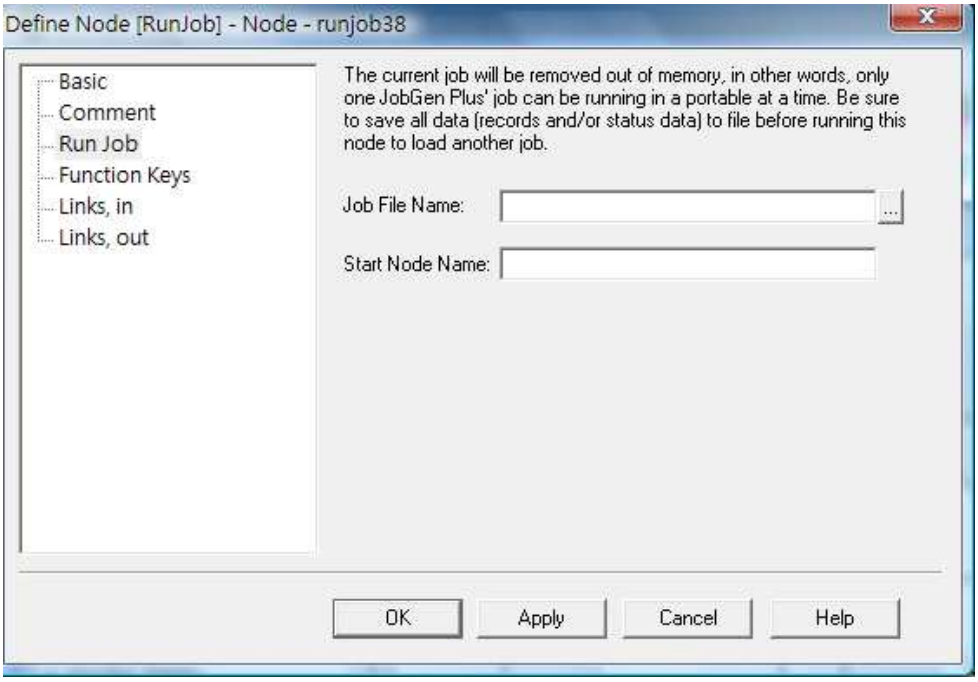
WARNING, run job is different than call job. Run job will never return to the caller. The job engine in the portable only keeps one job instance in memory at a time. The current job will be removed from memory when run-job loads a new one. Remember to save all data to file before executing run-job.

A “sub” job can return back to the caller by rerunning the caller job. By specifying a Start Node Name, the Job engine can start the Job from the specified node instead of from the original start node.

Define Program Node

Basic Property.
Comment Property.
Function Keys Property.
Links In Property.
Links Out Property.
①
These properties are as same as the ones in menu node.

Run Job Property.



Job File Name Define job execution file name.

Start Node Name Define start node name. By default, Job

engine will run the Job from start node defined in the Job. This can be changed by entering another node name, so that the Job will start from the specified node.

Creating Comment node

Comment node is used to paste comments directly on flow-chart.

Creating Links

Links are the essential components that construct the **JobGen Plus** application. They demonstrate not only the path from one process (*Node*) to another, but also the flow of program execution.

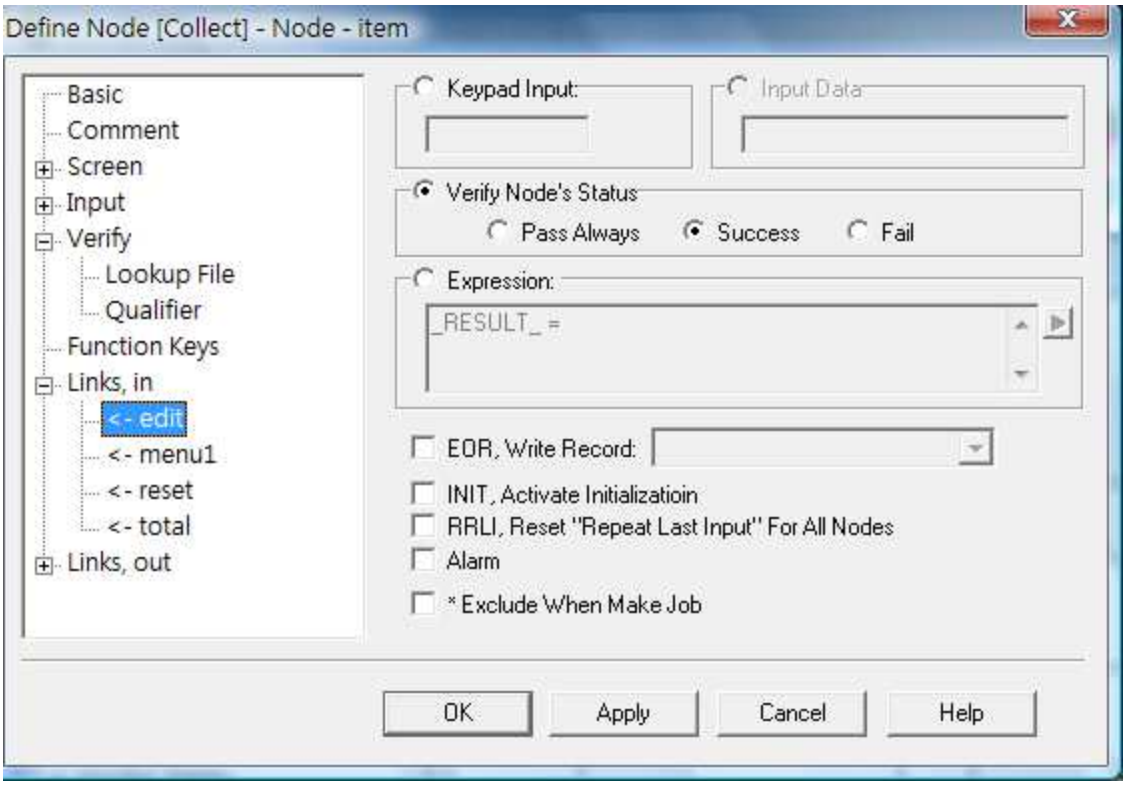
To create a link: select the link tool by clicking the link icon in the toolbar, or selecting link in nodes menu. Point to the start node, click and drag the mouse cursor to the destination node, release the mouse button. A link will be created.

To create a self-looping link: Point to the node button, move the cursor to close one side, click and drag mouse to another side within the node button, release the mouse button. A self loop link will be created.

A node can have multiple out going and in coming links. It can also have multiple links to the same node but each link should have different conditions.

The conditions of a link determines which of multiple links from a source node will be selected. A link is selected when the conditions of the link is evaluated as TRUE.

Define Link Property:



- **Keypad Input:**

Click to select this option, then click on its edit box to bring up the portable keypad window. Select a key by clicking on it. Only one key is acceptable for a link. When the Job has finished execution of the source node and is waiting for an input for a link selection, a key press of the key defined here will set this link condition to be TRUE.

- **Input Data:**

This option is only available when the source node is a collect or count type. It compares the data value of the node with data specified here, and sets the result to be TRUE if they are identical.

- **Verify Node's Return:**

This option is for checking the execution result of a source node. For the menu, erase, and edit nodes, the execution is always Success. For the collect node, the execution is Success only if the input data passes the verification. For the upload node, if a time-out occurs, the execution is Failure.

Verify offers four options: Always, Success, Failure, and None:

Always	The status of the execution result of the source node is irrelevant - it is always TRUE.
Success	The condition is TRUE only if the execution result of the source node is Success.
Failure	The condition is TRUE only if the execution result of the source node is Failure.

- **Expression:**

This is a C expression. It has the same format as a Qualifier in verification definition in collect node. The value of predefined variable _**RESULT**_ is evaluated as the result of the condition.

- **EOR (End of Record), Write Record:**

Click this option to indicate that the input of the current specified record will be completed and saved into the data file when this link is executed. There is at least one link with an EOR defined in a job if the record is defined. Multiple EORs (that is more than one link have EOR selected) is applicable depending on different conditions to complete the record.

The Job engine won't complete and save a record if the execution is looping in a circle of some nodes. It will simply overwrite on the old data. The only way the job engine will save a record is when it encounters an EOR signal in the execution of a link.

Specify a record name to tell job engine which record will be saved into the data file.

- **INIT, Activate Initialization**

  Click this option to indicate that it is time to activate the ***Set New Value*** option in the initial processing of collect and math nodes.

- **Reset "Repeat Last Input" For All Nodes**

  Click this option to indicate that the next record needs a new input. It is used to cooperate with the setting ***Repeat Last Input*** in the input session in the collect node. After the job engine encounters a link with New Input Record checked, it will be forced to input a new data the next time it executes a collect node that has Repeat Last Input checked. The function of New Input Record only affects the next record, after which a collect node with Repeat last Input can repeat again until it encounters New Input Record in a link again. This is a way that repeated fields in a column are able to get a new input to repeat.

- **Alarm:**

  Click this option to sound the buzzer when executing this link.

The priority of link conditions

After a job has finished executing at a node (at this time, this node is a source node), it will decide which node is the next one to execute. This decision is made by checking all the conditions of all the out going links, and determining which one meets the link condition. The evaluation of these conditions is under the following priority sequence.

1). When a node is waiting for an input in the middle of execution (for example, keypad input in a collect node), pressing a Special Key will abort the current execution. The job engine will check all the links dispatching from this node to determine if the special key just pressed is defined in the **Keypad Input** condition in one of these links. If it finds one, this link will be selected and the execution flow will go to the destination node of this link. If no such a key is defined in these links then the press of this special key is ignored and the node execution continues.

Note, the special key will also be checked to see if it's defined in a separate node. If the job engine finds a node that defines this special function key, then that node is selected as the next node to run.

2). When the execution of the source node has completed, the job engine will check all the links dispatching from it to determine if there is a **Success** or **Failure** setting activated in the **Verify Node's Return** conditions. If a link has defined **Fail**, and the execution result of the source node was **Success**, this link will not be selected. But if the result is **Fail**, this link is chosen and execution would then go to the destination node of this link.

3). If there is an **Expression** condition and the result of the calculation of the logic expression is TRUE, then the link is selected and execution will go to the destination node of this link.

4). If there is an **Input Data** condition defined, and the data matches with this definition, then this link is selected and execution will go to the destination node of this link.

5). If the **Always** option is defined in the **Verify Node's Return** condition, the job will go to the destination node of this link. If no link has defined the **Always** option, no link is selected at this step, and the job will wait for an input.

6). If there is a **Keypad Input** condition defined, and a key press matches with this definition, then this link is selected and execution will go to the destination node of this link.

There are six priorities to select a link, and all links are evaluated in the order of the link's sequence number. If a link selected, then the execution will go to the destination node immediately without further checking for the lower priority conditions.

After these priority steps have been sequenced, if no link is found, the job will pause there. If this should happen as a result of inadequate programming, press **Exit** key to quit the job.

If there is not a link defined dispatching from this node at all, then the execution of the job will be terminated automatically.

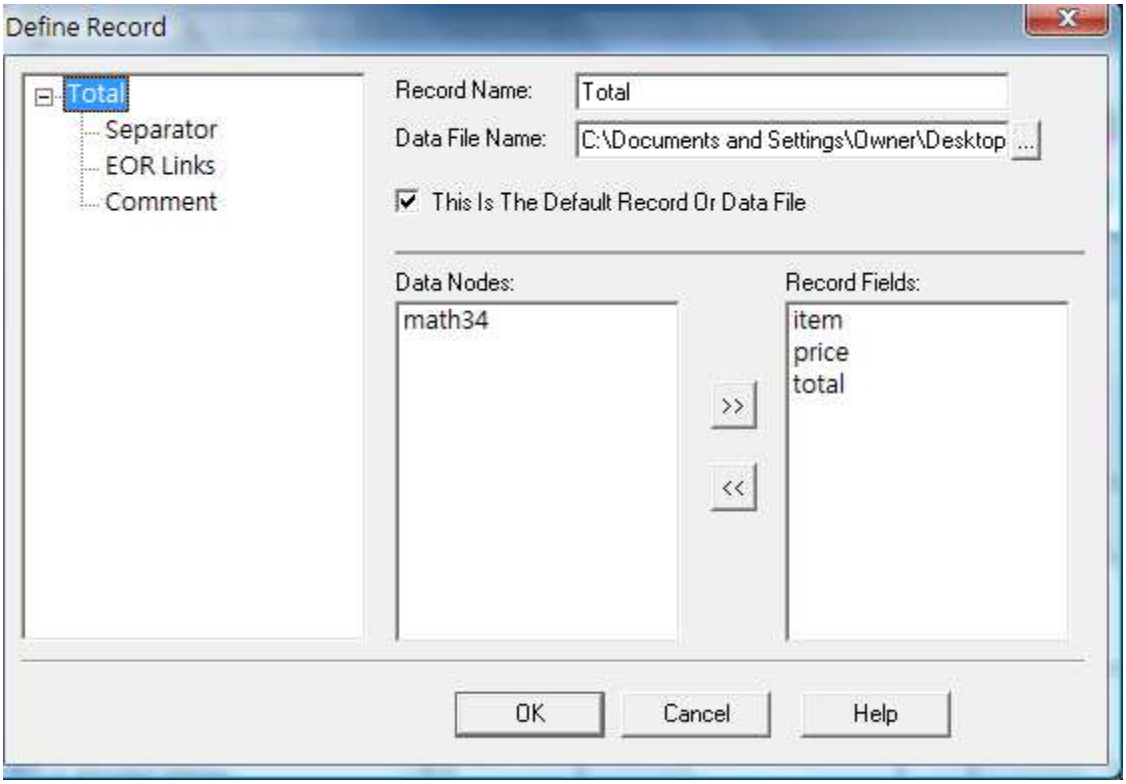
Define the record

Each job can have several records. A record consists of **field**s, field and record **delimiters**, preamble and postamble.

A **field** that subscribes an attribute of the record is a collect node. When you define a collect node and select the checkbox of the Record Field, this node becomes a field automatically. By default, JobGen Plus sets all defined collect nodes as fields of the record, and the sequence of the fields in the record is determined at the time of the node's definition - the collect node first defined is the first field, the next collect node defined is the second, ..etc. The **Field delimiter** is used to separate fields and the **record delimiter** is used to separate records. Preamble and postamble are char-strings. The preamble is set before the data of a record and the postamble is set after the data of the record.

Each record can have a TXT file to store data. The data filename can be defined by the user, and its default name is Job name with **.TXT** extension name - for instance: demo .txt is the job demo's data file. The data file stores many records as defined by the Data File Format.

There are two ways to define a record. The first way is done when defining a collect node, but only the fields of the records are defined this way. Another way is to select the Define Record from the Edit menu to activate the Define Record dialog box.



In the Define Record dialog box, there are two scroll-windows: **All Fields:** (window at the left), and **Fields in Record:** (window at the right). All the collect nodes that have been created in the Job are listed in the **All Fields:** window and all fields in the record are listed in the **Fields in Record:** window. Between these two windows are the **Add** and **Delete** buttons.

When you want a collect node to become a field of the record, click that node in the **All Fields:** window to select it, and then click the **Add** button. This node will become a field of the record and appear in the **Fields in Record:** window.

To delete a field of the record, click the node in the **Fields in Record:** window, then click the **Delete** button, and the field will be deleted and moved out of the window.

There are two other pull-down windows for defining **Field** and **Record Delimiters**. Only the characters: Comma, Semicolon, Tab (ASCII 9), and LF&CR can be selected as field or record delimiters. No single character can be selected as both field and record delimiter at the same time.

The sequence of fields follows the order in which the fields are created.

The Preamble and Postamble can be defined by typing the char-string in the appropriate edit box.

Finally, click OK to save your work.

To see the effect of your changes on the Record Field checkbox, click on the collect node that you modified to bring up the Define Collect dialog box.

The Job engine will not complete and save a record if the execution is looping in a circle of some nodes. It will simply overwrite on the old data. The only way the job engine will save a record is when it encounters an EOR signal in the execution of a link.

Data File Format

The data is stored in the data file record by record. The maximum length of a record is 1024 bytes. The record format is:

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

in which:

<f1><**f2**>...<**fn**>:	field 1, field 2, ...field n The maximum length of a field is 64 bytes. N is the field number in a record. The maximum value of n depends on the lengths of fields, the lengths of preamble and postamble.
<**fd**>:	Field delimiter user defined.
[**Pre**]:	Preamble user defined.
[<**fd**>[**Post**]]:	[Post] is the postamble user defined. If there is Postamble, an fd must be inserted before it.
<**EOR**>:	Record delimiter user defined.
[**CR&LR**]:	If the **EOR** isn't a CR&LF, JobGen Plus will automatically add a CR&LF

at the end of a record. If the **EOR** is a CR&LF, JobGen Plus will ignore it. There must be one and only one CR&LF at the end of a record.

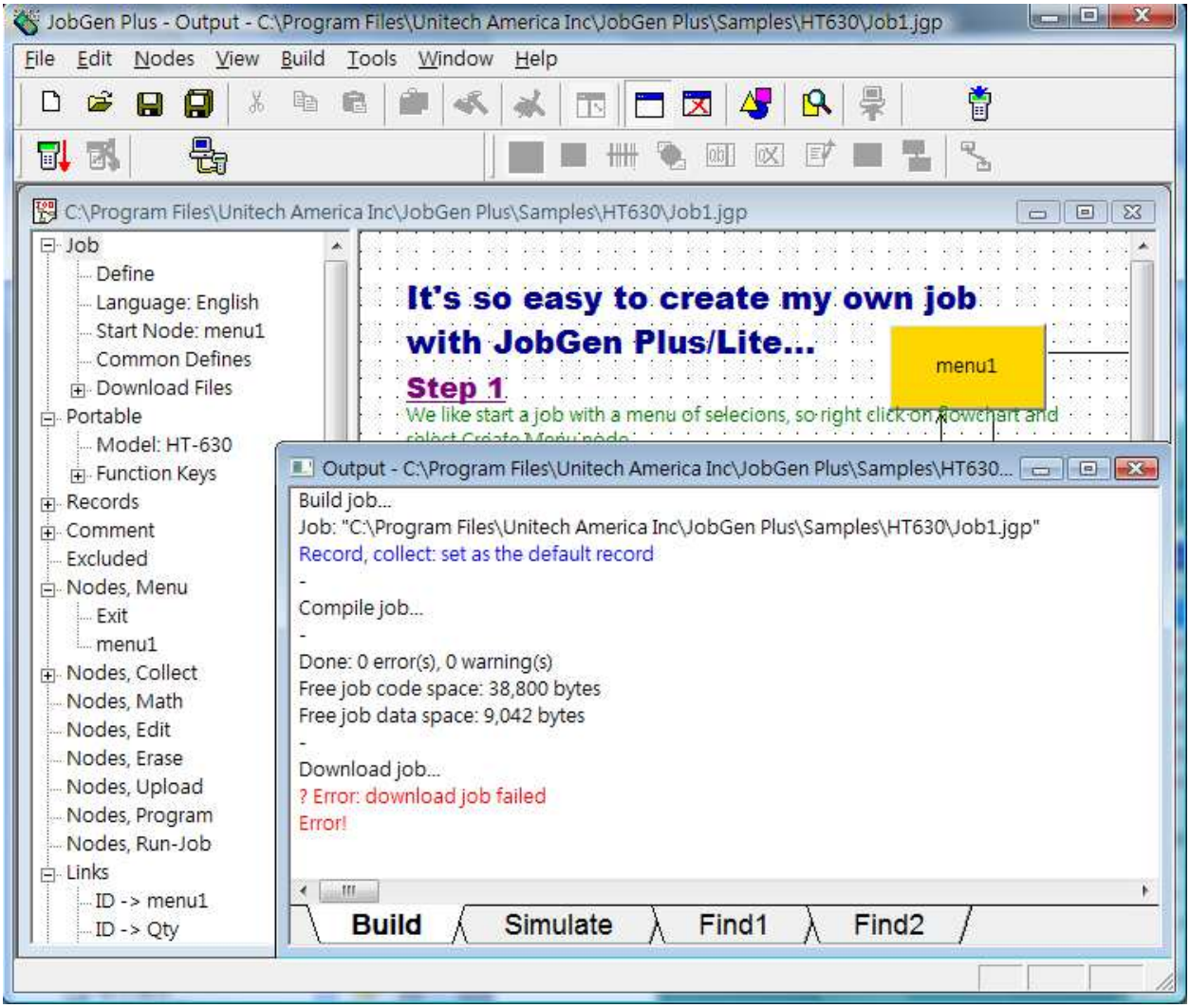
Chapter 6 Make Job

After all the nodes and links are created, the next step is to make the executable for the Job. JobGen Plus takes two steps to make Job executable code. First it generates source code according to all the settings in the flowchart. It will report errors and warnings, if any, in the message window. Then it will compile the source code to finally generate the executable code. It will report errors and warnings too, if any, in the message window.

If you get errors (printed in the message window in red letters), simply double click on it and a related definition dialog box will pop up. Change the setting to fix the problem, and run Make Job again.

In the middle of Make Job, JobGen Plus may activate several other operations, such as Make Binary Search Index Table or Make Chinese Characters Font File. All these activities will be reported in the message window.

After Job Executable has successfully been generated, JobGen Plus will activate PTComm Manager to download the job to the portable. The download may include all related files, such as all lookup files, binary search index tables, and Chinese font file.



After the executable is successfully downloaded to the target portable, it is ready for users to issue the "RUN" command and start the **JobGen Plus** Job application.

Chapter 7 Simulate Job

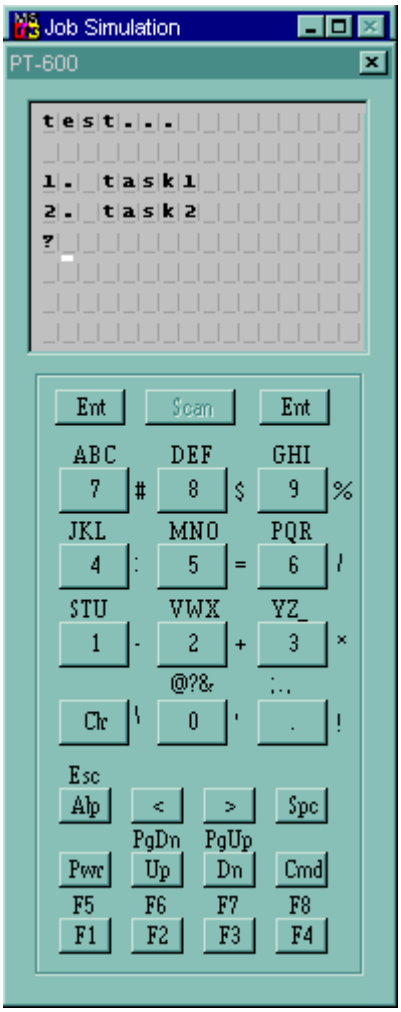
After a Job has been designed and defined, it can be simulated in the host PC. Select Run > Simulate Job, or click the simulate icon on toolbar to start simulation.

Simulation is available for Windows 95/98/Me only. Windows 3.x and NT and 2000 are not supported.

Job simulation provides an easy way to examine the job without having a real portable on hand. Simulation displays a portable window that is very similar to the portable model selected for the job, and it runs the job and job engine in DOS session under the Windows. It simulates all the inputs and outputs. All the operations are exactly same as they would be on a real portable.

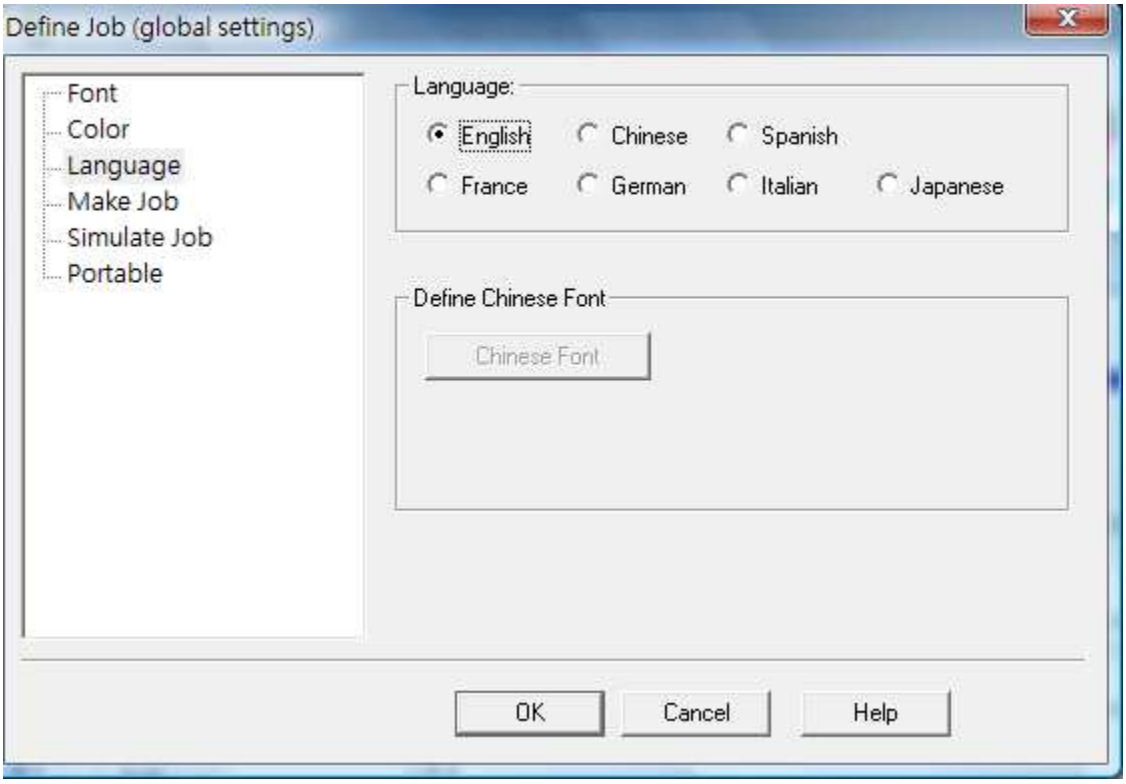
There are certain limitations in the simulation of a Job. The file search in Lookup verification cannot exceed 64K bytes from the beginning of the file.

Job simulation is helpful in developing a Job. It is highly recommended to test all Jobs by downloading them to a real portable and running them through every function. This is the only way to verify that a Job has been produced correctly.



Chapter 8 **Language Support**

JobGen Plus supports multiple language output on the portable screen. Choose the language at installation time. JobGen Plus will automatically generate additional support files, like font bitmap files, while making a Job and downloading them together with the Job execution file to the portable.



Check “Share Chinese Font With Font File” to share the font with other jobs. Having multiple jobs use a single font file eliminates duplication of Chinese characters. This will save precious portable memory.

Chinese Character Support:

JobGen Plus retrieves the bitmaps of Chinese characters from the font libraries in Windows with Chinese support, and creates a CCB (Chinese Character Bitmap) file that contains both codes and bitmaps. This file will be downloaded to the portable along with Job Executable as well. Font type, style and size can be selected. The size of the Chinese character bitmap displayed in the portable is 16 x 16 pixels. The layout of the position of a character in the bitmap can be adjusted too. Both GB and BIG5 code type are supported.

CCB file can be shared between jobs. That is, several jobs can use a single CCB file so that it can save precious system memory. The Job engine will first try to open the job's own CCB file (jobname.ccb). If it fails then it will try to open the default CCB file (jeng.ccb).