

1. Overview

RFID Inspection: If you are producing RFID labels from label stock and inlays, you need to inspect your labels and verify that the inlays are still good after you convert them. The easiest thing to do is to run the finished labels past a reader, and check for the reader's "green light." That's slow, but it's the minimum investment path, and many companies start out that way.

Automated RFID Inspection: As you begin to automate your testing, you first need to look at the steps involved:

- Monitor the web position so you know when to read
- Trigger the reader when the tag is in the read zone
- Complete some operations on the tag before it leaves the read zone
- Determine whether the tag was good or bad
- Mark the bad tag before it gets away

Each of those steps involves instrumentation that may or not already be on your system. Since it's your system, you will need to develop the exact sensor placement and reader setup commands for every tag you handle. You will also need to work out the exact timing sequences for your PLC programmers to implement. And if you don't own the PLC, you'll be negotiating with the equipment company for every change.

Interference and Overruns: So, imagine you fight past all those issues and actually start running your system. What then? Well, first you find that you can't tell exactly which tag you're talking to. So now you're an antenna enclosure designer. Finally, everything is running great, and you start cranking the speed up, only to find that you are marking the wrong tags bad when the web exceeds some threshold speed.

LineLogix: LineLogix is a standard platform for RFID conversion line automation. Using a combination of PC and realtime embedded software, it performs all the tasks outlined above, on different tag types, at high speed. It can tell when your line is running faster than your reader. The LineLogix Timing Control Unit (TCU) establishes the correct timing windows for reading and marking based on gap and position sensors. The TCU remembers how far your marker is from your reader, so the correct tag is marked, every time. LineLogixTCU also supports peel-and-replace equipment. LineLogixPC software configures it all, monitors the job, and reports the results.

2. Installation

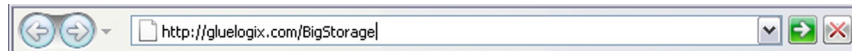
New installations of LineLogixPC are packaged as a self installing EXE file. It is too big to email. Most users will receive a PC with software preloaded and jobs set up for their application.

2.1 *Downloading the Installer*

Each customer has a password protected customer folder on the GlueLogix website, accessible only by that customer. You will be provided a user name and password to access this folder to get the latest available version of LineLogixPC.

In the following example we will use the user name “BigStorage” with the password “Big14store!” to show how to obtain the latest version. We will also use MSIE 7 as the browser, so your browser may look a bit different.

First, open your web browser and enter “http://GlueLogix.com/<user_name>” into the address bar.



A login screen similar to below will appear.

A screenshot of a login dialog box. The dialog box has a blue header with a key icon. The main area is white with a grey border. It contains the following text: "The server www.gluelogix.com at Only for authorized personnel requires a username and password." followed by a warning: "Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection)." Below the warning are two input fields: "User name:" with a dropdown menu showing "BigStorage" and "Password:" with a masked password field (dots). There is a checkbox labeled "Remember my password" below the password field. At the bottom are "OK" and "Cancel" buttons.

The server www.gluelogix.com at Only for authorized personnel requires a username and password.

Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection).

User name:




Password:

☐ Remember my password

OK Cancel

Enter your user name and the password you were provided, and then click OK.

If your user name and password are entered correctly, you will be presented with a directory listing which will include the latest version of the installer for you to download.

Index of [REDACTED]			
Name	Last modified	Size	Description
 Parent Directory	13-Nov-2007 10:04	-	
 .htaccess	03-Oct-2007 09:00	1k	
 LineLogixPC 313 03Oc..>	03-Oct-2007 09:02	5.0M	

Apache/1.3.31 Server at gluelogix.com Port 80

Click on the “LineLogixPC_<version_number>.exe” link and save the file to your desktop.

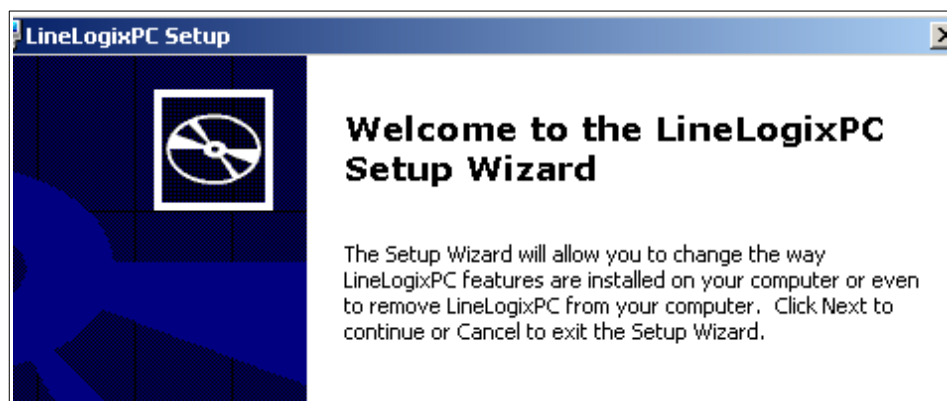
Once you have successfully downloaded the installer, you can close your browser.

2.2 Starting the Installation

Once the installer has been acquired, double-click on the file to initiate the install.

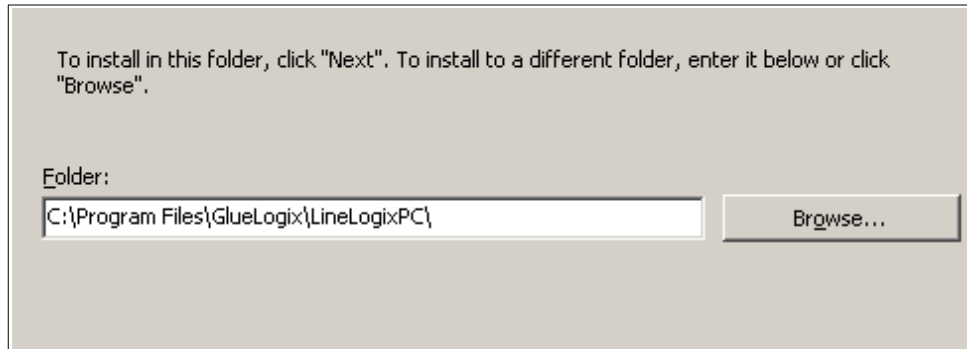
2.3 Completing the Installation

The first page presented by the installer is a Welcome page.



Click 'Next' to continue.

The second page presented allows you to view or change the folder in which LineLogixPC will be installed.

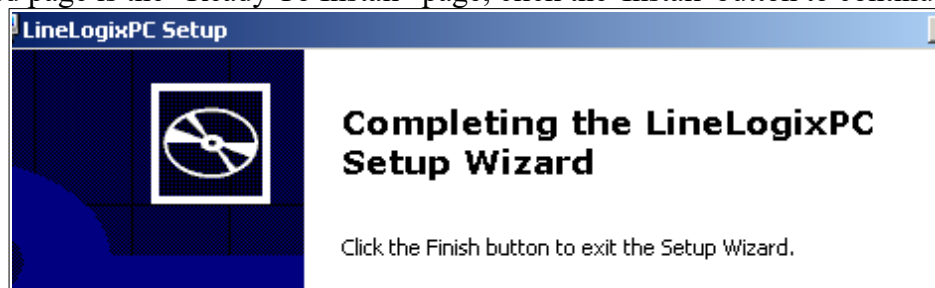


The default installation folder is “C:\Program Files\GlueLogix\LineLogixPC\”.

It's a good idea to take note of this folder location, as it is where most of the output files and folders created by LineLogixPC will be located. Starting with Windows7, the permissions of the install folder must be adjusted in a later step.

Once a suitable folder has been selected, click 'Next' to continue.

The third page is the “Ready To Install” page; click the 'Install' button to continue.



After the installer finishes copying files, and initializing the appropriate settings, you will be presented with the final screen of the installer.

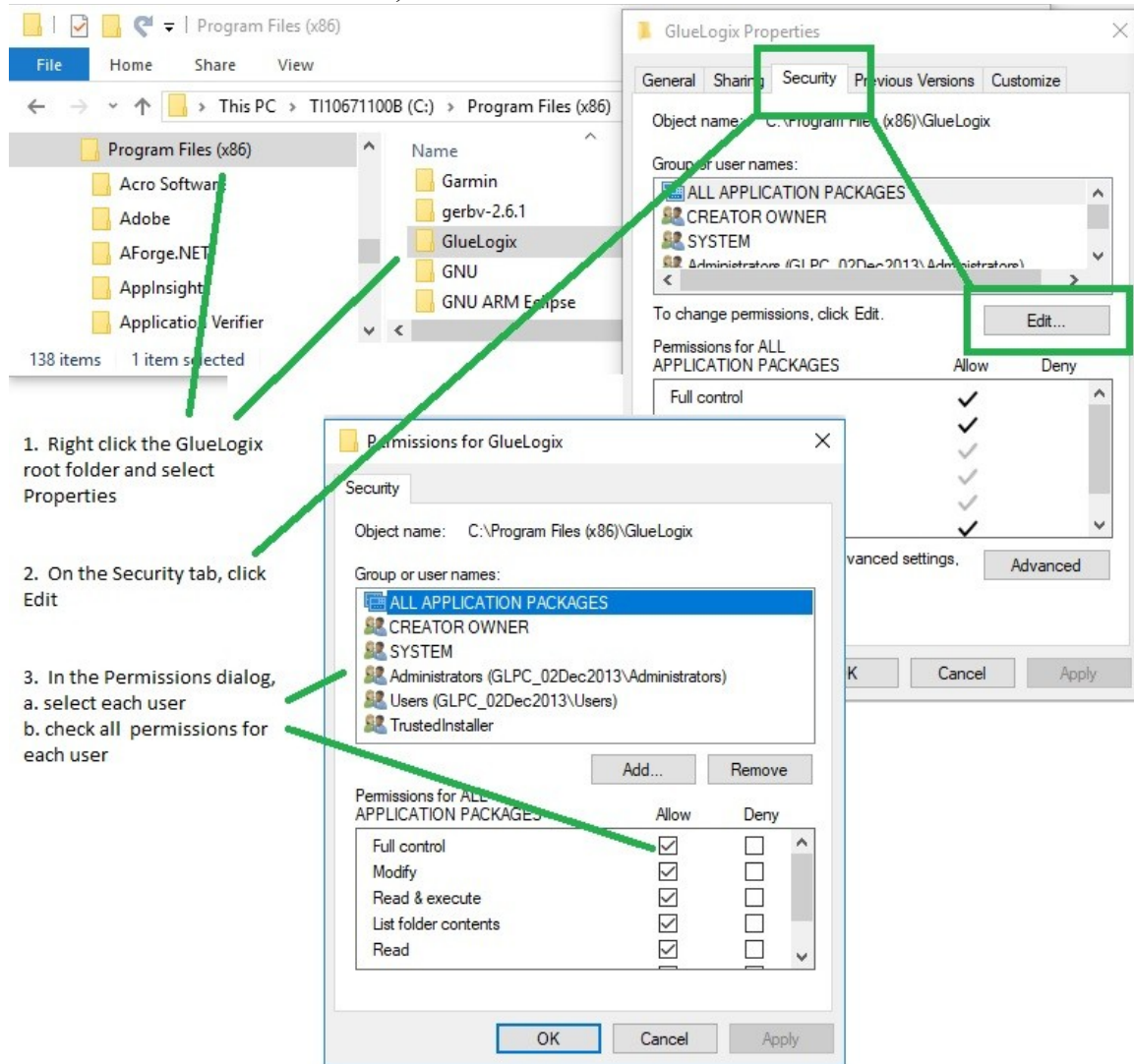
Click the 'Finish' button to exit the installer.

LineLogixPC will now be available in your Start menu, as well as on the desktop (depending on settings used during install).

2.3.1 Folder Permissions

After installation to Program Files on Win7 and above platforms, the following steps must be performed in order to support job and log file creation in the install folder. These steps can be skipped if LineLogixPC is installed to the desktop or other unprotected location.

1. Right click the GlueLogix root folder and select Properties
2. On the Security tab, click Edit
3. Click each User in turn, and enable all Permissions for each User.



Note: These instructions have been distributed as text files Win7Instructions.txt and Win10Instructions.txt.

2.4 *Updating LineLogixPC*

Updates to LineLogixPC are normally distributed as a “patch,” which is a ZIP archive containing updated files for the program folder.

To install a patch release, follow these steps:

1. In Program Files\GlueLogix\LineLogixPC, make a folder named "Backup"
2. Move LineLogixPC.exe and Library.zip to folder Backup. Your desktop and taskbar icons should now be inoperable.
3. Open the new patch ZIP. Observe two files inside with the same names as the files you just moved.
4. In the next step, DO NOT EXPAND Library.zip. The python runtime module expects to work with a ZIP file of that name.
5. Extract the new LineLogixPC.exe and Library.zip to Program Files\GlueLogix\LineLogixPC (2 files)
6. Start the new program from your desktop or taskbar icon.
7. Once logged in, check version using Help, About. It should match the filename of your release package.

Note: These instructions have been distributed as text file
PatchInstructions.txt.

2.5 *Removing LineLogixPC*

LineLogixPC can be removed from the system by utilizing the Windows' “Add or Remove Programs” wizard. This wizard is available to you by clicking Start, Settings, Control Panel in Windows.

By utilizing this wizard you will have the ability to Remove, Modify or Repair the LineLogixPC installation.

2.6 *Customizing*

The distribution contains a copy of text files called `operators.dat`, `supervisors.dat` and `site.dat`. You may customize the contents of these files to match your own needs. Instructions are given below in the sections that describe the various files.

File Tip: Over time, you will be developing your own local copies of files like `operators.dat` and the LineLogixPC job files. Create a new folder `C:\Program Files\GlueLogix\Local` and maintain your files in there. Then after extracting each new distribution into `C:\Program Files\GlueLogix\LineLogixPC`, you can copy in the contents of your Local folder and be sure you are using the latest version of your own jobs.

3. LineLogixPC Reference

3.1 Command Line

LineLogixPC has several command line parameters as of September 2022, which are formatted as key=value pairs, and can be presented in any order:

`csv=<CSV filename>` must include extension. Set the CSV name as if it were entered in the Start Roll button's CSV Select dialog. The Select CSV dialog at Start Roll time (including `autostartjob=True`) will be skipped if the CSV is specified on the command line.

`dat=<Sector filename.dat>` must include “.dat” at the end. Select the DAT file as if it were specified in the JOB (as `usermemoryfile=filename.dat`)

`job=<JOB filename.job>` must include “.job” at the end. Set the JOB name as if it were selected in the login screen. If the JOB file is set on the command line, the Login screen will be skipped.

`macro=<MACRO filename.py>` must include “.py” at the end.

`op=<Operator Name from operators.dat>` Set the operator name as if it were selected in the Login screen. If the JOB is set on the command line but the Operator is not, the Login screen will be skipped and the session will run as the Default operator. Default may or may not have JOB change privileges, see `Supervisors.dat` below.

`key=value readerN.key=value` Any value set up by the JOB file can be overridden on the command line. The first form overrides values in the top global section of the JOB file. The second form overrides values in a reader stanza. For example, to set UHF power for the `[reader0]` JOB file stanza to 10 dBm on the command line, use this command line parameter: `reader0.z=10`. Note that this substitution is made at program startup, before `site.dat` limits and force values are applied, so make sure `site.dat` is compatible with your command line settings.

Samples:

```
LineLogixPC.exe job=SomethingFromTheJobsheet.job
```

3.2 Startup

To start LineLogixPC, double click the desktop icon. As LineLogixPC starts up, a number of things happen:

- If a JOB file is specified on the command line, that JOB is used, even if `default.job` is present.
- If LineLogixPC is started from a button or shortcut, without command line parameters, it first checks for a text file called `default.job`. Job files contain settings for LineLogixPC and for the LineLogix system. If `default.job` is present, LineLogixPC reads settings from that file and logs in as an operator

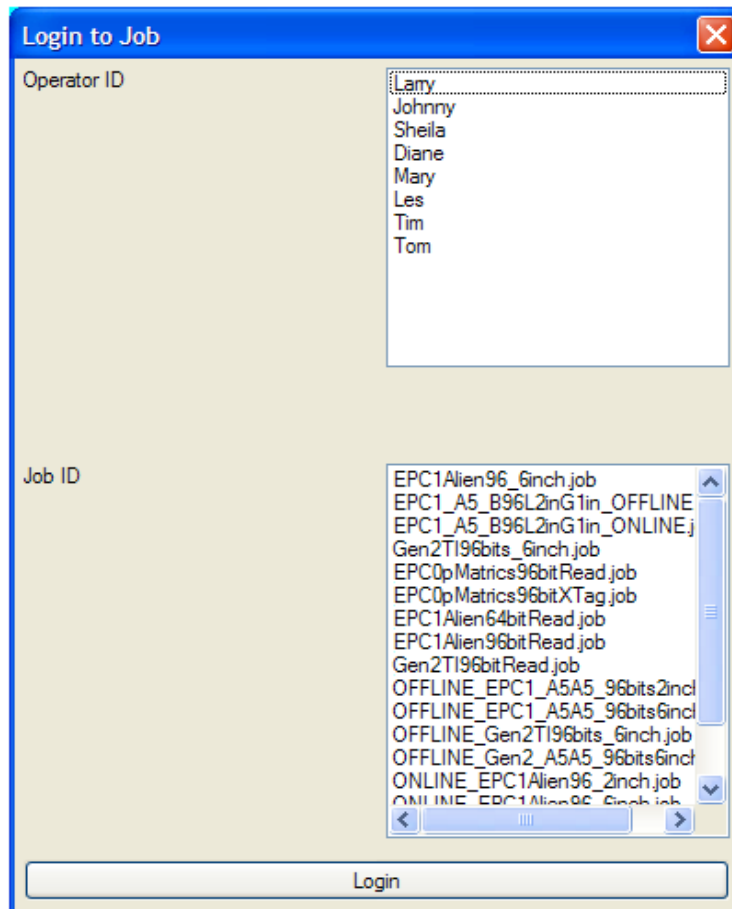
named “default.”

- If `default.job` is not present, LineLogixPC checks for a text file called `operators.dat`, which contains the names of the operators at your facility, and it also checks the current directory for job files.
- If `default.job` is not present, and if either `operators.dat` is absent or no jobs are available, LineLogixPC exits with an error message.
- If `default.job` is not present, but both `operators.dat` and at least one job are available, LineLogixPC presents a Login screen which allows the operator name and job to be picked from lists. When the Login operation is complete, LineLogixPC reads settings from the selected job file.
- LineLogixPC opens the serial or network ports in the selected job, contacts the TCUs and RFID readers configured in that job, and sets those devices up for the job.
- LineLogixPC sets up logging based on job settings. If logging is enabled, LineLogixPC checks for the presence of a subdirectory called `Output`. If `Output` is missing, it will be created. All log files are placed in `Output` unless overridden by global setting `LogDirectory` in the jobfile.
- LineLogixPC sets up its display based on job settings.

3.3 Screens

3.3.1 Login Screen

This screen will not be displayed if file `default.job` is present.



You must select an Operator ID and a Job, then press Login.

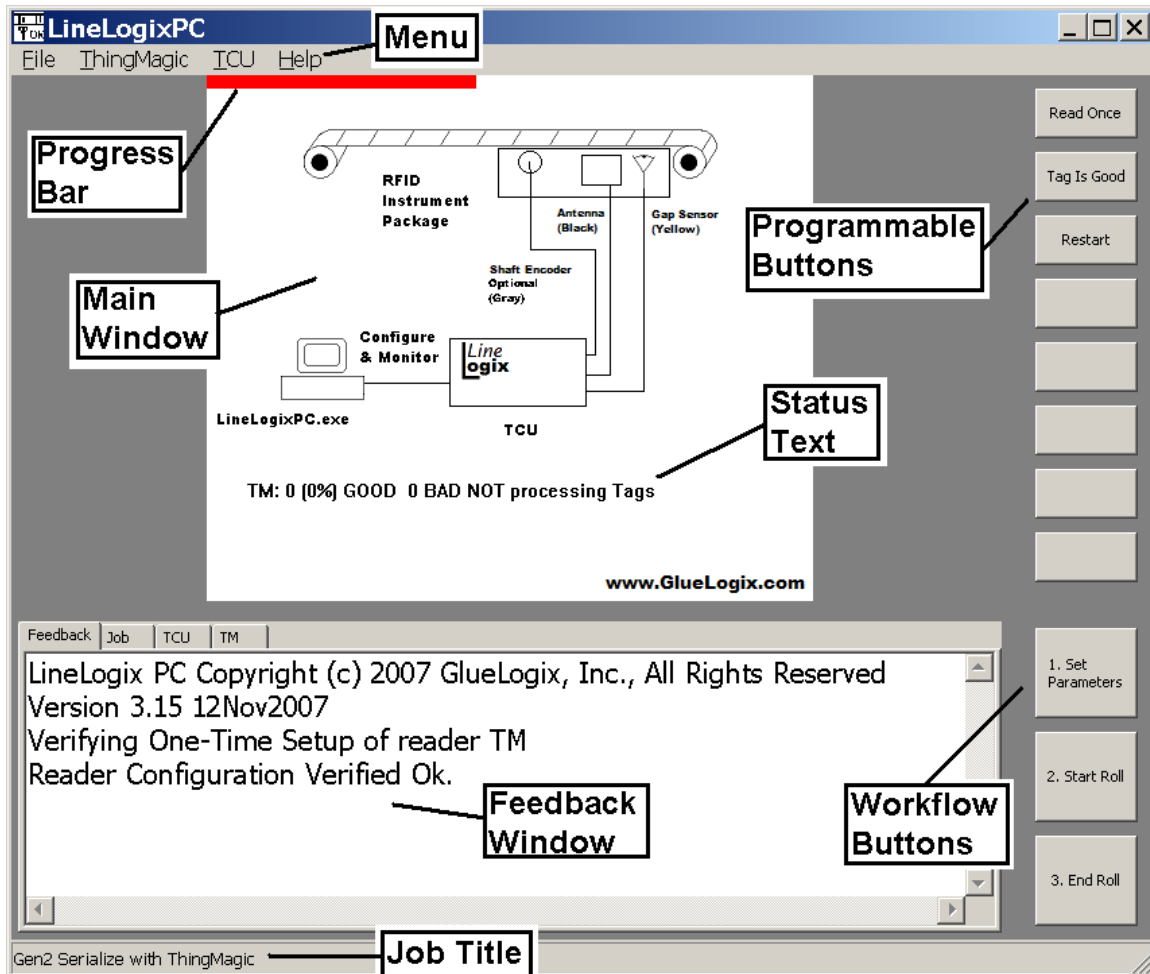
The Operator ID list is filled from file `operators.dat`. A short reference section is given below for this file.

The Job ID list is filled by taking a directory of all job files in the current directory. Job files are identified by the suffix “.job”.

The title bar and exit box may be suppressed, making your PC seem more like a machine controller. Set `UsePCControls=False` in the Global section of `site.dat`.

3.3.2 Typical Interface

The appearance of LineLogixPC is highly configurable and may vary greatly based on the current job. This section describes some visual features of LineLogixPC based on a typical job file.



3.4 Menu Reference

Computer Tip: Many programming languages count things starting with 0 instead of 1. Programmers who use those languages a lot start doing the same thing. Many of LineLogixPC's features are set up for up to 4 readers numbered 0 through 3.

3.4.1 File Menu

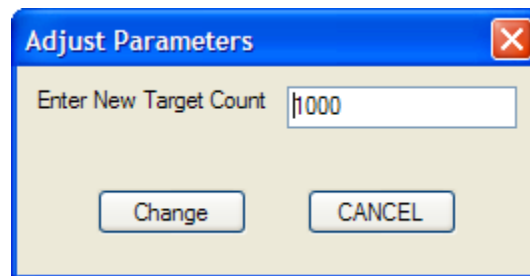
Operator Options:

Debug Report: Close the current job and collect the job file, log and the contents of the Feedback Window into one large file, named automatically and saved in the Output folder. Then attempt to send the file to GlueLogix via a private email relay. As of September 2022, that relay is hosted at mailgun.org.

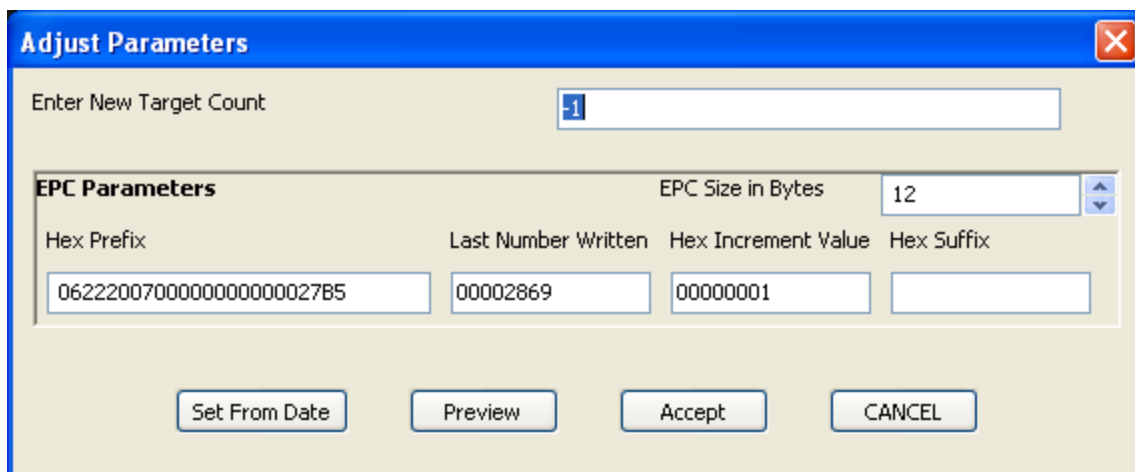
Save Feedback: Save the contents of the Feedback Window to a file. A standard file selection dialog is presented so you may specify the filename.

Clear Feedback: Clear the Feedback Window. During a long run, your screen may flicker as the Feedback Window history takes up PC memory. You can make that stop with this menu item.

Adjust Parameters: Most job parameters are set by the job file and are not available for change. Some parameters can be changed at runtime. This menu item brings up a dialog that presents those parameters for adjustment:



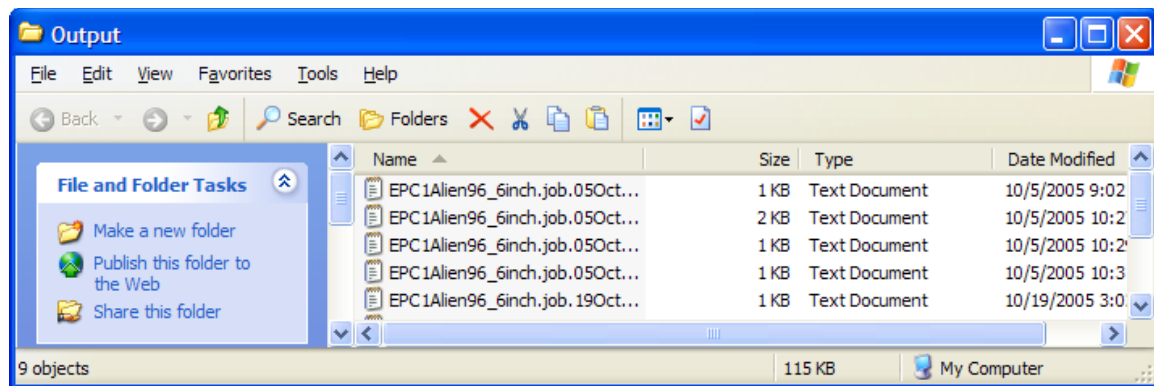
If a Serialize (Write Increment) job is selected, the Parameters dialog looks more like this:



Last Log: Opens the last-saved log file in a text editor. The editor is configured in job

file Global option `logeditor`. If no editor is configured, Notepad is used as the default.

Logfiles: Opens the Output folder (which contain the log files) in a file manager window:



Logout: Ends the current job if one is active and returns to the Login screen.

Exit: Exits the program. Closes any current roll.

Supervisor Options (these options appear when the current Operator name appears in `supervisors.dat`):

Operators: Opens `operators.dat` in the configured JobEditor.

Supervisors: Opens `supervisors.dat` in the configured JobEditor.

Site Settings: Opens `site.dat` in the configured JobEditor.

Apply Job: Send the current job tab contents to all readers and TCUs and save the JOB file.

Name Job: Specify a name for the current job, for saving after changes. Default is the current job name with a plus sign (“+”) in front. If you want to overwrite the current job, remove the leading plus sign.

Write Job: Save current job settings to disk. If no job filename has been specified, you will be prompted for one.

Edit Job: Opens the current job file in a text editor. The editor is configured in job file Global option `JobEditor`. If no editor is configured, Notepad is used as the default.

3.4.2 TCU Menu

This menu contains actions appropriate for the LineLogix TCU, and will be hidden if no TCUs are configured in the current job. Multiple TCUs are differentiated by the Reader Name. The following options appear for each configured TCU.

Operator Options:

Version: Queries one TCU for its software version and displays the result in the Feedback window.

Setup: Sets up one TCU for the current job.

Mark: Sends a message to the TCU to Mark a label per its current configuration. This will not work until the TCU has been **Setup**.

Stop: Sets the Stop output per the current configuration. This will not work until the TCU has been **Setup**.

Restart: Clears the Stop output per the current configuration. This will not work until the TCU has been **Setup**.

Pause: Sends a message to the TCU to pause automatic processing, i.e., ignore incoming triggers.

UnPause: Sends a message to the TCU to restart automatic processing, i.e., process incoming triggers. This may not work correctly before the TCU is **Setup**.

Supervisor Options (these options appear when the current Operator name appears in `supervisors.dat`):

Query: Queries a TCU for current values of tag position, count and other important quantities. Displays the result in the Feedback window. See the TCU section of the system reference for an explanation of the displayed values.

Cycle: Sends the Q0C! command and formats the result. This is a Cycle diagnostic that maintains a

Debug: Sends a single text command to the TCU. Displays the result in the Feedback window. See the TCU section of the system reference for an explanation of the available commands.

3.4.3 ThingMagic Menu

This menu contains actions appropriate to the ThingMagic M5e module (as packaged in the Vega automotive reader), and will appear when these readers are configured in the job file.

Show Serial: Scan the first 10 serial ports and try to find a reader. Results are displayed in the Feedback Window.

Multiple Readers:

Multiple readers are differentiated by the Reader Name. The following options appear for each configured reader.

Operator Options:

Version: Displays the Version string captured when the reader was opened in the Feedback Window.

Read Once: Sends a Read Tag ID Single command to the reader and displays the result in the Feedback window. Useful for debugging.

Verbose: Start printing of verbose data for debugging.

End Verbose: End printing of verbose data for debugging.

Supervisor Options (these options appear when the current Operator name appears in supervisors.dat):

None.

3.4.4 Help Menu

About: Displays the software version. Job and operator names along with the version information are printed to the Feedback Window. Please have this information ready if you need to contact GlueLogix for technical support.

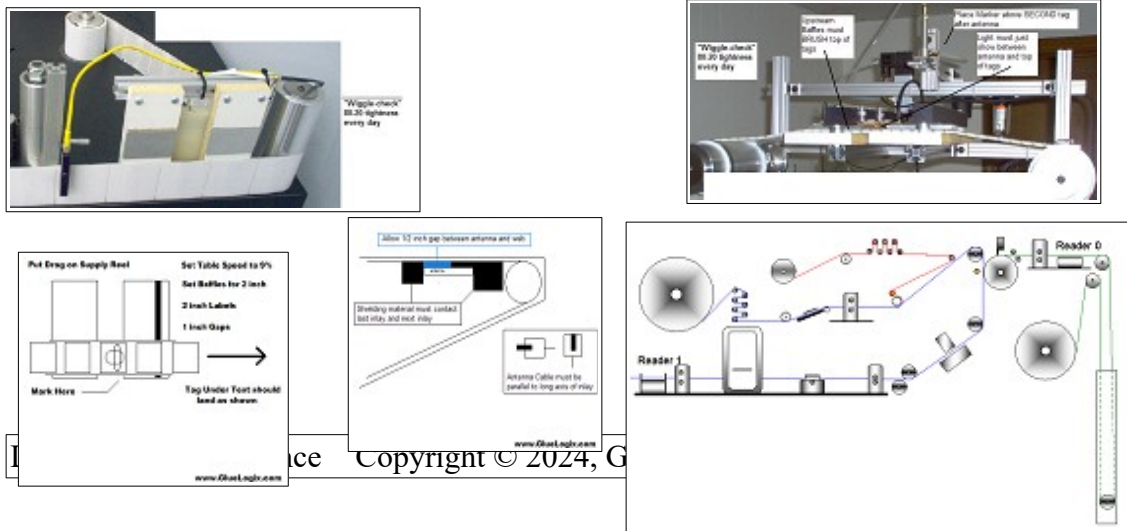
3.5 Main Window

Displays a graphic, specified in the job file, in Windows Bitmap (BMP) format.

The Main window graphic is the most visual element of LineLogixPC. Site administrators are free to change the main window graphic to include:

- Diagram of the installation
- Instructions for the operator
- Photo of the correct setup for the job
- Photo of the correct tag for the job

Here are some bitmap files in daily use:



3.7 Programmable Buttons

Eight blank buttons are available on the LineLogixPC interface. The content of these buttons is read from `site.dat` and the individual job files. See the Job File Reference section, below, for more details.

The Programmable Buttons will not be displayed if the Button section is missing from both `site.dat` and the job file.

A Programming Model for these buttons is presented below.

3.8 Job Title

The Status Bar of LineLogixPC displays the `title` entry from the current job file.

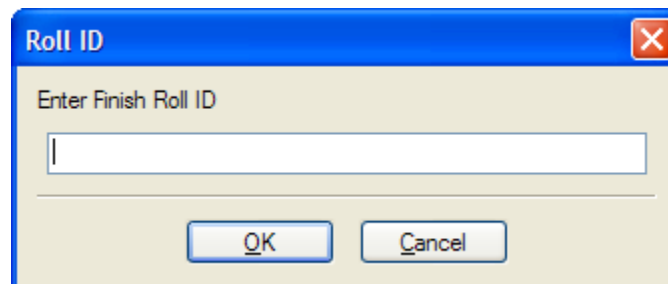
3.9 Workflow Buttons

LineLogixPC enables a workflow model based on finished label rolls. The buttons marked `Set Parameters`, `Start Roll` and `End Roll` are always present and cannot be changed or configured out by the job file.

The `Set Parameters` button performs the same function as menu `File, Set Parameters` – see above. Setting parameters is mandatory before `Serialize (Write Increment)` jobs. If Parameters are not set when `Start Roll` is clicked on a `Serialize` job, the Parameters dialog will be displayed anyway.

The `Start Roll` button initializes all internal counters and configures all readers and TCUs in the current job. Parts of the main window may go blank while these actions are being performed. If Logging is configured in the job file, a new log file is opened. When this button is successful, the Status Text changes to **Processing Tags** for each reader in the job.

The `End Roll` button closes the current log file and changes the Status text to **Not Reading Tags**. If the current job is configured to capture a roll ID, this dialog is displayed to do so:



The Roll ID will be saved in the log file. If so configured in the job file, the log file will also be renamed to the Roll ID with a `.txt` extension. Future releases of LineLogixPC may print the Roll ID on a label.

3.10 Job Setup Tabs

When a Supervisor is logged in, tabbed controls are visible allowing most jobfile settings to be made.

3.10.1 Job Tab (one per job)

The screenshot shows a software interface with four tabs: Feedback, Job (selected), TCU, and Standalone. The Job tab contains the following settings:

Field	Value
Author	Larry
Statusbar Text	Gen2 Serialize with ThingMac
Number of Readers	1
Prompt for RollID at end of roll	True
Inlay Type	Gen2
Gap Length in Decimal Inches	0.0625
How Many Labels on a Roll? (-1 disables)	Set by Operator in Parameters
Stop the System after Consecutive Errors? (-1 disables)	-1
Change Date	02Feb10
Bitmap File Name	LL_On_TwinCat.bmp
Frequency Range	FCC
Rename Log File to RollID	True
Inlay Repeat in Decimal Inches	1.75
Start Job as soon as Program Loads	False
Include Bad tags in the Roll Count?	False
Reader Index for Consecutive Errors	0

A "Download Job" button is located at the bottom right of the form.

The Apply (formerly Download Job) button sends changes to all TCUs and Readers. Often used settings are presented in **Boldface**.

Control	Note
Author	Operator Name from Login Screen
Date	Read from PC by LineLogixPC
Number of Readers	Global setting NumberOfReaders
Frequency Range	Always FCC
Prompt for RollID at end of of roll	Global setting GetRollID
Rename Log File to RollID	Global setting RenameLogToRollID
Inlay Type	Global setting TagName
Inlay Repeat in Decimal Inches	Global setting LabelLength
Gap Length in Decimal Inches	Global setting GapLength
Start Job as soon as Program Loads	Global setting AutoStartJob
How Many Labels on a Roll?	Global setting TargetCount . Set to -1 (in Parameters screen) to disable StopOnCount.

Include Bad Tags in the Roll Count?	Global setting TargetBadTags
Stop the System after Consecutive Errors	Global setting StopOnErrorNumber. Set to -1 to disable StopOnCount.
Reader Index for Consecutive Errors	Global setting StopOnErrorReader
Application (not shown in Rev G)	Global setting application. A catchall string for configuring Standalone, Barcode and other PC based features.

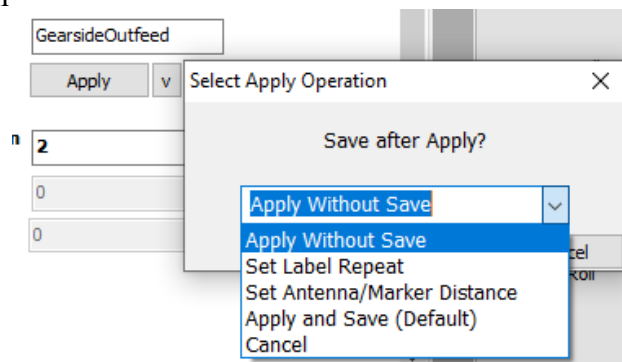
3.10.2 Download and Apply Buttons

Changes to values in configuration tabs are not immediately sent to connected devices like TCU boards and RFID readers. If every change were immediately sent to the the devices, a series of changes might bog down due to communication lag. The software assumes that a setup person will make all required changes, and then press an Apply button (formerly called Download). The Apply buttons act as follows:

- collect every parameter from the configuration tabs
- check each parameter against optional limits in `site.dat` and correct as necessary
- update the software's JOB structure with the parameter values
- Write the updated JOB structure to a file
- download the updated JOB structure to all connected devices

Additional Apply options are available on a small pulldown button next to each Apply button:

- Apply without saving the JOB data to a file.
- Set **Label Repeat** directly with no other changes. Affects only PC software.
- Set **Antenna/Marker Distance** in label edges, with no other changes. Sends change to the connected TCU.



3.10.3 Reader Tabs (one per reader)

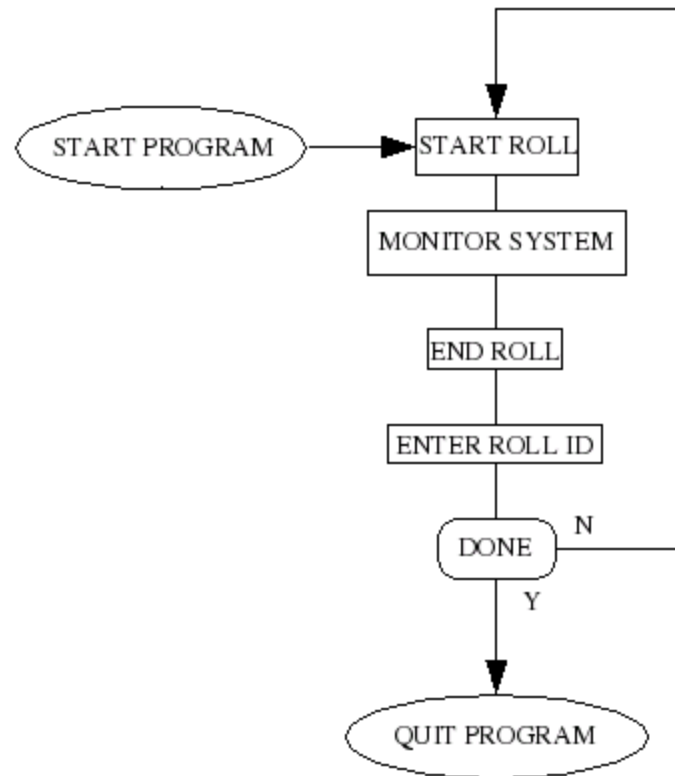
The `Redisplay` button moves the Status Text display after changing its X and Y coordinates.

The `Reopen Reader` button resets all software interfaces to this reader, including the serial port, and sends the current parameters to this reader.

The `Apply` button sends changes to all TCUs and Readers.

Control	Note
Reader Type	Reader setting <code>Type</code>
Log Counts	Reader setting <code>LogCounts</code>
Log IDs	Reader setting <code>LogIDs</code>
Connection Type	Reader setting <code>Conn</code>
Serial Port	Reader setting <code>Port</code> Through Version 3.x, Unix numbering is used, so Windows COM1 is entered as “0”. In Version 4.x and above, COM port numbering is displayed in Windows platforms but the number is still saved in Unix format.
Comm Parameters	Reader setting <code>Baud</code>
IP Address	Reader setting <code>IP</code> . Only available when the Ethernet Connection Type is selected.
Test	Reader setting <code>RdrSetupOneTime</code>
Power in dbm	Reader setting <code>Z</code>
Timeout	Reader setting <code>T</code>
TCU Present?	Reader setting <code>TCUPresent</code>
TCU Port	Reader setting <code>TCUPort</code>
TCU Baud	Reader setting <code>TCUBaud</code>
X coordinate of Status Display	Reader setting <code>DisplayX</code>
Y coordinate of Status Display	Reader setting <code>DisplayY</code>
Reader Name	Reader setting <code>Name</code>

3.11 *LineLogixPC Operation with default.job*



1. Start LineLogixPC with the Windows desktop icon. LineLogixPC will find all readers needed for the job, check their configuration, and notify you of any problems. If all goes well, the Status Text will say **Not Reading Tags**.
2. Set Parameters by pressing the Set Parameters button.
3. Start a Finished Roll by pressing the Start Roll button. If all goes well, the Status Text will change to **Reading Tags**.
4. Run the machine. Watch the Status Line counts to make sure they agree with the counts provided by your machine.
5. When the roll is complete, stop the machine and press the End Roll button.
6. If prompted for a Roll ID, enter one. Use the work order number or other value from your tracking system. A USB barcode scanner that enumerates as a Human Machine Interface (HMI) may be used to enter barcode values into this dialog.
7. Repeat steps 2 through 5 until you are out of tags.
8. Exit LineLogixPC.

4. LineLogixPC Button Programming Model

LineLogixPC's Programmable Buttons are programmed by the job file. Each button has a one line entry in the job file. All text up to the first comma appears as the label of the button. All text after the comma is executed by LineLogixPC as if it were a line of source code. This creates a very powerful and flexible system, but one that requires some explanation. Notes:

- The button commands are Python statements that execute within LineLogixPC.
- You can set up more than one button command on a button. Separate the individual commands by semicolon (;) characters.
- An “at” (@) character at the start of a button command forces LineLogixPC to print the result of the command in the Feedback tab. A result of None in Python prints an empty line.

If, after reviewing this material, you have trouble creating buttons for your users, feel free to contact GlueLogix tech support for help.

Here is the job file entry that creates a “Read Once” button for the first reader:

```
Read Once, self.OnReadOnce()
```

The string `Read Once` to the left of the first comma has become the label of the button. The string to the right of the first comma is the action taken by LineLogixPC when the button is pressed. The text to the right of the comma is interpreted by the Python runtime system in the context of the top level of LineLogixPC.

Notes:

- Button text is case sensitive.
- Multiline button programs are not supported.

4.1 Top Level Functions

Several functions of the menu and windowing system are available through the Button mechanism. Due to the syntax of the Python programming language (used to write LineLogixPC), the top level of the application is referred to as “self”.

4.1.1 Menu Functions

File: File menu functions are available on Programmable Buttons by calling:

```
self.OnReport()  
self.OnSaveFeedback()  
self.OnClearFeedback()  
self.Params()  
self.LastLog()  
self.ManageLogs()  
self.TimeToQuit()  
self.OnDownloadTabs()
```

```
self.SetNewJobFileName()  
self.EmitJob()  
self.EditJob()
```

Reader: Generic reader routines are available on Programmable Buttons by calling:

```
self.OnReaderVersion()  
self.OnReadOnce()
```

Note: These functions take an optional parameter:

- `Empty()`: operate on the first configured reader, usually reader 0
- `Index(0), (1), etc.:` operate on that reader
- `All(-1)`: operate on all configured readers

TCU: TCU menu functions are available on Programmable Buttons by calling:

```
self.OnTCUVer()  
self.OnTCUSetup()  
self.OnTCUQuery()  
self.OnTCUParams()  
self.OnTCUMark()  
self.OnTCURead()  
self.OnTCUStop()  
self.OnTCUUnstop()
```

Note: when called without parameters, the last functions operate on the first reader (Reader0).

Help: Help menu functions are available on Programmable Buttons by calling:

```
self.OnAbout()
```

4.2 *Application Level*

The next level of the programming model is the application logic (as opposed to the windowing logic of the Top level). For historical reasons, this level is called the Read Only Application and abbreviated `roa`.

4.2.1 **ROA Functions**

AdjustCounts: Accepts one parameter, a number that will be added to the Good count and adjusted from the Bad count. To go the other way, pass in a negative number. This function is intended for “doctoring” applications where bad tags are recovered through button driven interaction with the RFID reader, or replaced by hand.

```
self.roa.AdjustCounts(1)
```

GetJobId: Returns the name of the job file selected in the login screen as a string.

```
self.roa.GetJobId()
```

GetLoginName: Returns the operator name selected in the login screen as a string.

```
self.roa.GetLoginName()
```

VerifyReadersAgainstJob: Verify each configured reader against its job parameters. This is the same function called automatically when LineLogixPC starts up.

```
self.roa.VerifyReadersAgainstJob()
```

4.2.2 NFC Editor Buttons

Tag Seek, self.roa.jr[0].TagSeek()

New Tag, self.OpenCloneTagDlg(self.roa.jr[0], "new+ndef")

Clone Tag, self.OpenCloneTagDlg(self.roa.jr[0])

4.2.3 Job Readers

The ROA layer owns the RFID readers configured in the job. Up to three readers may be configured. They are stored in array `jr` (short for Job Reader). Buttons can access all readers configured in the job file. The mapping is as follows:

<i>Job File Reader</i>	<i>Button Reader</i>
[Reader0]	self.roa.jr[0]
[Reader1]	self.roa.jr[1]
[Reader2]	self.roa.jr[2]

Useful elements of the Reader structure are described below in section 4.3.

4.3 Readers

Each RFID Reader object encapsulates connection information and functionality for one reader. The Reader objects are owned by the ROA layer and are organized as described above. This section describes useful elements of the Reader object.

4.3.1 Reader Object Functions

GetCounts: Returns a 3-tuple containing the number of Good, Bad, and Error replies from the reader since the job was started.

```
self.roa.jr[n].GetCounts()
```

SerialWriteLine: Passes a string to the serial or network port configured for the reader object. Appends line termination (CRLF). Any reply from target is displayed in the Feedback window by other code.

```
self.roa.jr[n].SerialWriteLine("string")
```

4.3.2 Reader Object Variables

Name: The “Name” string from the job file, as displayed in the Main window's Status Text. You can assign a new Name to any reader by changing this string.

```
self.roa.jr[n].Set("Name", "newname")
```

StatusText: Running status of Reader object. Possible values: "READING Tags", "NOT Reading Tags", "Tag WRITE Ok", "Tag Write FAILED", "Cannot find reader", "Cannot connect to Network Reader", "Cannot open Reader serial port", "No Reader Setup Data".

```
self.roa.jr[n].TCU.StatusText
```

type: The “Type” string from the job file.

```
self.roa.jr[n].type
```

writeid: The “WriteID” string from the job file. You can assign new base values to Symbol write jobs by setting this string.

```
self.roa.jr[n].writeid
```

4.3.3 Reader Object TCUs

Each reader object can own one TCU object. If no TCU is configured for a given reader, the value of the TCU object is the Python reserved value `None`.

Standalone mode readers do not own a TCU object, nor do `subreader` or `childreader` objects.

4.4 TCUs

Each TCU object encapsulates connection information and functionality for one TCU. Each TCU object is owned by a Reader object. This section describes useful elements of the TCU object.

4.4.1 TCU Object Functions

ClearStop: Sends the TCU serial command that clears the TCU's Stop output. Useful for a “Restart Web” button.

```
self.roa.jr[n].TCU.ClearStop()
```

SerialWriteLine: Passes a string to the serial or network port configured for the TCU object. Appends line termination (CRLF). Any reply from target is displayed in the Feedback window by other code.

```
self.roa.jr[n].TCU.SerialWriteLine("string")
```

ShowVersion: Gets software version from TCU and displays it in the Feedback window. This function has a side effect: StatusText is set based on the TCU's response.

```
self.roa.jr[n].TCU.ShowVersion()
```

TriggerMark: Sends the TCU serial commands that toggle the TCU's Mark output.

```
self.roa.jr[n].TCU.TriggerMark()
```

TriggerRead: Sends the TCU serial commands that toggle the TCU's Read output.

```
self.roa.jr[n].TCU.TriggerRead()
```

TriggerStop: Sends the TCU serial command that sets the TCU's Stop output. Unlike the Mark and Read functions, this one just sets the Stop output. Function ClearStop must be called to restart the web.

```
self.roa.jr[n].TCU.TriggerStop()
```

Query: Queries the TCU for its current state and displays the replies in the Feedback window. See the TCU manual for an explanation of the replies.

```
self.roa.jr[n].TCU.Query()
```

4.4.2 TCU Object Variables

StatusText: Running status of TCU object. Possible values: "TCU Ok", "Cannot find TCU", "Cannot open TCU serial port".

```
self.roa.jr[n].TCU.StatusText
```

5. Operator File Reference

The Operator File, `operators.dat`, is a simple text file containing the name of each operator who is allowed to run the machine. Each line of this text file becomes a line in the Operator ID section of the Login dialog (see section 3.3.1 above).

Here is the `operators.dat` file that created the display of section 3.3.1:

```
Larry
Johnny
Sheila
Diane
Mary
Les
Tim
Tom
```

The operator's name is placed in each log file created while that operator is logged in.

Tip: If you need to place some information in your log files that is not supported by LineLogixPC, and you don't care about the operator's name, then you can use the `operators.dat` file to hold that information, and it will appear in the log file. For example, the customer's name may be more important to you than the operator's name. If so, use `operators.dat` to hold your customer list instead of your operator list.

If LineLogixPC starts up with `default.job`, the operator name is assumed to be “default”, and is so recorded in the log file.

6. Supervisor File Reference

The file `supervisors.dat` contains names from `operators.dat` that should be allowed to setup and save jobs. The operator name `default` can also appear in `supervisors.dat`. Operators who are not also supervisors are not shown the Job and Reader setup tabs, or the extended setup menus; but they can still set Parameters using the first Workflow button.

Sample:

```
default
Supervisor
```

Supervisor accounts can be protected with a simple password, stored unencrypted in `supervisors.dat`

Sample:

```
default,pass1
Supervisor,pass2
```

THIS IS NOT A SECURITY MEASURE, JUST AN EXTRA CHECK.

7. Job File Reference

The job file is organized by “stanzas,” delimited by lines beginning with a left square bracket (“[“). Everything up to the first stanza is considered a global parameter.

Settings are key, value pairs, one per line. For Boolean settings, the values “yes”, “true” and “1” will be read as True – all other values will be read as False.

7.1 Global Settings

All job file entries up to the first square bracket (“[“) are stored by LineLogixPC as global settings. Values are shown here in CamelCase for readability. Most JOB files are all lower case.

Application: A catchall configuration string, described in a separate section below.

Author: Informational, should be updated when the job is changed.

AutoStartJob: If TRUE, the operator will not have to press “Start Roll” before starting the first roll.

Bitmap: Sets the name of the bitmap file to display in the Main window.

DebugStr: A collection of debug parameters that may sometimes help make a job work temporarily. Any use of this parameter in the field should be reported to GlueLogix for proper analysis.

GetRollId: If TRUE, Roll ID is captured and placed in the Log file when the operator presses End Roll

Helper: Sets up a Helper app. See the Helper reference for details.

JobEditor: Sets the name of the text editor to use on Job files when accessed through LineLogixPC menu choices. If not specified, Notepad is used as a default.

LabelLength: Repeat of the label in inches. Used to develop FPM from TCU traffic. The number of TCU messages in a second is multiplied by the LabelLength.

LastChangeDate: Informational, should be updated when the job is changed. Set automatically when job is Written by LineLogixPC.

LogDirectory: Sets the directory for Log files to be written. Provided so Logs may be saved to a network drive. If not provided, logs are written to the Output subdirectory of the install location. **As of Rev G**, there is no facility for remote locating jobfiles, and Debug Reports are always written to the Output folder.

LogEditor: Sets the name of the text editor to use on Log files when accessed through LineLogixPC menu choices. If not specified, Notepad is used as a default.

NumberOfReaders: Sets LineLogixPC up for the number of [ReaderN] stanzas that follow in the job file.

PrintRollId: Reserved

RenameLogToRollID: If TRUE, each log file will be renamed to the roll ID entered by

the operator after pressing End Roll.

StopOnErrorNumber: If present and greater than zero, stops the web after N consecutive errors on Reader0 or the reader identified by **StopOnErrorReader**. A value of -1 disables the feature.

`StopOnErrorNumber=5`

StopOnErrorReader: Sets the reader that controls the stop on error function. See **StopOnErrorNumber**.

`StopOnErrorReader=0`

TagName: Trade name of your inlay. Can be “Variable By Reader” in multifrequency jobs. In those cases, each Reader stanza MUST contain a TagName key.

TargetBadTags: If TRUE, includes bad tags in the **TargetCount**. If FALSE, the roll is made bigger to compensate for bad tags.

TargetCount: Sets the target count for the job. On full LineLogix installations, the host machine can be stopped after the correct number of tags are tested. This value can be adjusted through the Parameter dialog. It is not exact at high speeds.

TimerInterval: Can be used to change the rate at which LineLogixPC checks for information from the TCU. Units are whole number milliseconds, default is 1000 (check once per second).

Title: Sets the text to appear in the status bar of LineLogixPC. This is freeform text with no formatting requirements.

UsePCControls: When False, the Windows title bar will be hidden along with the Minimize and Exit buttons. This is meant to make LineLogixPC look more like a machine controller. In this mode, AltTab, CtlF4 and other standard Windows keyboard shortcuts are available.

UserMemSize: Sets the length of Ultralight/NTAG block data. May be set by the TagSeek function.

UseSmallFonts: The Feedback Window normally uses an oversized font for visibility. When TRUE, this setting keeps the font at the normal size for the PC.

7.1.1 Application String

The application string is a catchall configuration string. It is most commonly used to set up Standalone operation and PC based barcode capture.

`>raction` in each reader's configuration is executed in turn. Results are logged as if a hardware trigger had been received at the TCU. Standalone jobs require the `>raction` to be specified because there is no TCU.

`barcode` – sets up PC based barcode capture. For each cycle, a Barcode Dialog is presented. The dialog can be empty for debugging, or its contents can be filled in from an external scanner or Helper application. Sample lines follow:

barcode

Presents an empty Barcode Dialog on each cycle. Useful for debugging, or integrated applications that present data over a USB HID endpoint.

barcode:USB,L6

Sets up to receive barcodes from a USB connected HID scanner. The Barcode Dialog is set up to return after 6 characters are received, so you can use a scanner that does not append line ends.

barcode:com7,115200

Sets up to receive barcodes from an RS232 connected scanner on COM7 at 115200 baud, no parity, 8 data, 1 stop ("8N1"). The data is edited to the last 9 received characters

barcode:com7,115200 substring=-9 trigger=0

In this form, incoming data is edited to the last 9 received characters. In other words, a negative substring value filters data right justified, and a positive substring value filters data left justified. The optional trigger parameter causes a trigger string to be sent to the serial connected barcode (not implemented for USB barcode scanners as of Rev G).

csv helper – Enable selection of a CSV for a Helper program

delaystart – Do not start stepper based transport, or release the Stop line of standard transport, after Start Roll. The operator must start the transport explicitly using the TCU menu or a user button. This is the same function as delaystart=True in a JOB file reader stanza.

noparams – suppress the Params dialog on Parameters and Start Roll button press.

nottranslate – suppress barcode translation when it would otherwise be run

onpress – signals the user interface code to present some data in a form expected by press operators as opposed to rewind encoders.

probefirsttag – on Start Roll, read the tag on the antenna to determine its type and memory size.

prompt – used with barcode, sets the software to prompt for manual entry of a barcode when none is available via normal mechanisms.

readback – used with any >raction encode job, readback causes LineLogixPC to not encode, but just read back and verify tag data.

tagfirst – in jobs that are both standalone and barcode, LineLogixPC looks for a barcode first then for a tag. The tagfirst option reverses this flow.

standalone – sets LineLogixPC to self trigger based on arrival of new tags at the antenna.

ucode9 – configure nonstandard locking for NXP ucode 9.

Other - Custom special functions are often set up in the Application string, and documented to the customer in email.

7.1.2 Debug String

A collection of debug parameters that may sometimes help make a job work temporarily. Any use of this parameter in production should be reported to GlueLogix for proper analysis. Example:

```
debugstr=s_ilent,c_onstant,d_atatest,n_oreadback,s_howcmp,fo  
rgetuid
```

Note that many job files contain this string with all options disabled using underscores to break the pattern.

Parameters:

`forgetuid` – LLPC normally remembers all tag IDs seen in the last minute. In Standalone, the software refuses to act on known tags when they appear in the Inventory reply. The `forgetuid` string suppresses the software's storage of tag IDs for this purpose. It is not available unless a file named `developer.txt` exists in the same folder as `LineLogixPC.exe`. See the explanation of `tagagemax`.

7.2 TCU Setup for Each Roll

The contents of this stanza are stored as a single list of commands to be sent to the TCU at the start of a roll, or whenever TCU Everytime Setup is triggered by menu selection or programmable button press.

See the TCU reference for an explanation of these commands.

Through Version 3.x, there was only one global TCU section and all TCUs had the same settings.

In version 4.x, the TCU controls were moved into the Reader section. Now each reader's TCU is set up individually. There is no automatic coordination of parameters between TCUs. If this section is read from a jobfile by LineLogixPC 4.x, the parameters are copied to every reader in the job.

7.2.1 [ReaderN_TCUSetupEveryTime]

Since the Job file is not hierarchical, each reader's TCU parameters are saved in a stanza containing the reader name.

7.3 [Reader0] Through [Reader9]

The job file can contain up to ten stanzas named [Reader0] through [Reader9]. The number of these stanzas must agree with the global **NumberOfReaders** setting. Each line in this section is stored as a parameter for retrieval by the code that manages readers and TCUs. Values are shown here in CamelCase for readability. Most JOB files are all lower case.

>raction: If present, the value will be evaluated by LineLogixPC as part of each tag cycle. Usually used with the SuppressTargetCommand TCU flag and the WriteBlockData tag test. The most common >ractions are `self.FeigMiFareWriteSectorsFromTextFile()` for HF and `self.PcSerialize()` for UHF. Others are undocumented, and provided for special or custom applications.

BarcodeAsCsv: If less than 0, do not use the current barcode value as a CSV field. If 0 or above, substitute the current barcode value as the corresponding CSV field.

BarcodeIsAscii: In on-TCU encoding (the `Serialize` tag test), use the barcode data as presented and do not parse it as a number.

BarcodeLookup: Use the current Barcode value to look up actual data in the selected CSV. In Lookup jobs, the lookup field is always 0, with the other CSV fields shifted one place to the right.

Baud: Sets the baudrate and protocol bits for the serial connection to an RFID reader. TCU baud is set separately. Typical values: 115200 N81 for TCUV4 and 38400 N81 for TCUV3. Some ThingMagic-only setups with TCUV3 used 57600 N81.

ChildReader: This parameter identifies the index of a reader that will receive data from the current reader. The most common setup is that Reader0 will have `childreader=1` and Reader1 will have `parentreader=0`. Mostly used in multifrequency encode jobs.

Conn: Sets the physical connection to the reader. If `SharedSerial` is specified, the reader must also have a `parentreader` line pointing to a reader with a `Serial` (unshared) connection.

CsvDupCheck: In CSV jobs, run a check to see if the current CSV has been used before.

CsvDupField: In CSV jobs with `CsvDupCheck True`, specify the CSV data field to search for in saved LOG files, in order to determine previous use of the CSV.

CsvFileName: In CSV jobs, holds the name of and optionally the path to the CSV file selected at job start time.

CsvMacro: If `True`, triggers Macro behavior. See that section for more information.

CsvMerge: If `True`, enables CSV features. See that section for more information.

CsvSkipLineOnFailedWrites: If `False`, a CSV line will be applied to successive tags until one succeeds. If `True`, the next CSV line will always be fetched for a new cycle, even if the previous cycle failed.

CustomSetup: drives a variety of application specific behaviors for individual readers, as opposed to the Application String which is global to the JOB.

ReadTidAsTagData: For ThingMagic Vega and Nano jobs (NOT Feig jobs), changes the Trigger Command set up in the TCU to read the TID of each tag using a Read Tag Data (28h) command. The default behavior is to set up a fast Inventory with the Read Tag Single (21h) command. The Trigger Command is sent by the TCU to the connected RFID reader on each trigger, usually with some configurable level of retries.

SendMarkDeltaInMsBad: In WriteBlockData applications with >raction specified, on rewind, the PC sends the message >00! at the end of every good tag, to pass control back to the machine and advance to the next label. Bad tags on the main reader are signaled by >01!; upon receipt, the TCU sets its mark queue based on the low nibble of its SenseMarkDelta (parameter 0x0A). Bad tags on a secondary reader can be signaled by >02!, which sets Mark Queue from the high nibble of SenseMarkDelta (this feature is little used outside of one or two special projects). Starting in Jan2024, some TCU variants accept the **SenseMarkDelta** in the > message if the high bit is set. This option tells LineLogixPC to do so. For example, If the **SenseMarkDelta** of a reader tab is 0x03, and this option is set, the bad tag message will become >83!<CRLF>.

DisplayFallout: Originally changed the Status Text to display the percent bad instead of percent good. As of RevG, this setting is unused but still shows up in JOB files.

DisplayX, DisplayY: Sets the X-Y coordinates of the Status Text for this reader within the main window's bitmap. More information is given above in the description of the Main window.

DispBarDirection: Sets the direction of the Progress Bar (see 3.2.2) for this reader. Possible values:

- | | |
|---|---------------|
| 0 | Left to Right |
| 1 | Top to Bottom |
| 2 | Right to Left |
| 3 | Bottom to Top |

DispBarX, DispBarXSize, DispBarY, DispBarYSize: Sets the origin and cell size of the progress Bar for this reader.

Dynamic: When True, reloads the Sector File (the “DAT” file) set by parameter UserMemoryFile on every cycle. Defaults to False, in which the DAT file is loaded once at Roll Start time. This option is provided to support integration with external applications that may manipulate DAT files to get information into LineLogixPC.

ExFlags: A list of 16 small integers meant to fill TCU Parameter locations 16 through 32 (0x10..1F). See the TCU Reference section of the LineLogix User Guide for details of the TCU parameters.

FeedbackOptions: Sets optional extra information fields for the end of each OK/BAD line as displayed in the Feedback window and logged to file. As of Rev G, possible

values are:

BlockDataHex: Display and log block data as a punctuated ASCII-HEX string. For example, data 0x31323334 would show as the string “31-32-33-34”. Without this option, the data would show as the string “1234”, but unprintable bytes would not show up at all.

CsvLine: If true, appends the 0 starting line number of the CSV file for the current cycle. This behavior is undefined in CSV Lookup jobs.

Field1Only: Display and log only the HF UID or UHF EPC, not the barcode, block data, or other fields that are normally logged for the job.

NoFeedback: Suppress dynamic feedback over the Bitmap, including OK and BAD counts.

Nxporig: Read, display and log the NXP Originality Signature for Ultralight EVn NTAG21x, or NTAG242, depending on the JOB tagname.

ReadbackASCII[start:end]: cut the bytes [start:end] from tag return data and display it as ASCII text. Very useful for NFC applications, which have a mix of binary header and ASCII payload data.

ReadbackHex: Format, display and log readback data as ASCII-HEX. For example, data 0x31323334 would show as the string “31323334”.

Showlost: In Ultrafast configurations, display the count of “lost” tags, i.e. tags that are never assigned to a place on the web but are reported as they are about to shift out of memory.

Tamperstate25: Read, display and log ST25TV (ISO15693) dynamic tamper state

Timestamp: If True, appends a timestamp of the format “,Fri Jun 8 16:42:44 2018”.

Trust25: Read, display and log the TruST25 chip signature.

Ufmt=[ueak]: Display and log User, Epc, Access or Kill data in UHF jobs

Ultrafast: Enables display of the “jellyfish” multilane status diagram for Ultrafast configurations.

I: For Serialize (Write Increment) jobs, this is the value to increment the write data by after each tag. This will usually be 1, but some numbering schemes include a nonincrementing field at the end. If your job includes a 4-bit constant field at the end of tag data, then this parameter must become 10 (always read as hex).

IconSize: Meant to set the size of the Standalone color box, a Red/Green/Yellow block on screen that substitutes for a stack light in Standalone mode. **As of RevG**, this setting is unused but still shows up in JOB files.

ID: Sets an internal variable to the index (0 through 9) of the reader. This number **MUST** agree with the numeral in the name of the stanza. That is, stanza [Reader0] **MUST** contain the line ID=0. This is set up automatically when the job file is Written

by LineLogixPC.

IP: IP Address and port for Ethernet connected readers. The IP address and port number should be separated by a colon (e.g. 127.0.0.1:8080).

LogCounts: If TRUE, sets logging of summary counts for this reader at the end of the log file.

LogIDs: If TRUE, sets logging of individual replies from this reader in the body of the log file.

Login: Login name for Ethernet connected readers.

M: Formerly a modulation setting for SAMSys readers. **As of RevG**, this setting is unused but still shows up in JOB files.

MacroFileName: When `CsvMacro` is True, sets the name of the Macro file. See the Macro reference section for details.

Name: Sets the logical name of the reader. This name is used in the Main window's Status Text, in the Feedback window, the Menu and Tab titles, and in the log files. If the Name is missing, then the reader is referred to by the name of the stanza, i.e., "Reader0," "Reader1" or "Reader2."

OpTable: Set of instructions that should be executed by TCU on each inlay. See section the TCU Reference section of the Linelogix User Manual for more information on the Operation Table. OpTables only work with TCUv3.

ParentReader: This parameter identifies the index of a reader that will send data to the current reader. The most common setup is that Reader0 will have `childreader=1` and Reader1 will have `parentreader=0` Mostly used in multifrequency encode jobs. When a reader has a parent reader, it is also usually a **subreader**.

Passwd: Password for Ethernet connected RFID readers. **As of RevG**, this setting is unused but still shows up in JOB files.

Port: Sets the serial port for the RFID reader. This entry must be a number ≥ 0 . A 0 maps to COM1 on Windows and `/dev/ttyS0` on Linux. A 1 maps to COM2, and so on. For Ethernet connected readers, this parameter can also select the network port for TCP connection. Beware of conflicts with the port specified by the **IP** parameter, if any.

R: Remainder value for Serialize (Write Increment) jobs. If your job includes a constant field at the end of tag data, then this parameter must hold the value of that field, and the **i** parameter must be left shifted to the start of your incrementing field.

RdrSetupOneTime: Sets the test to be performed at each trigger from the host machine or TCU. May be different for each reader in the job. Values for all readers are:

- Read – Single Read command per trigger. TCUv4 has programmable retries on Read, to regain some of the benefit of the TCUv3 Count job.
- Serialize – In TCU based encoding, write incrementing values to successive tags.
- Count – Repeated Read operations in a window. TCUv3 only.

- Sequence – Associated with the optable option and UseOpTable TCU flag. Sets up a sequence of low level actions within the TCU in response to each trigger. TCUv3 only.
- WriteBlockData – Used with >raction to implement PC based encoding. In Standalone mode, this flag is used with TCUPresent set False. In normal mode, this flag is used with the SuppressReaderCommand, NotifyOnRead and StopOnRead TCU flags to force the TCU to turn control over to the PC on every tag.

Write and Alert appear in the pick list but are no longer used as of Rev G.

RssiFloor: Sent to the TCU to filter weak UHF reads. For ThingMagic Vega, values are 1..120. For Feig UHF readers, the value is a dbm value more negative than -15. In both cases, value of 0 disables the feature. A value of -1 suppresses display of the associated control in the Reader tab. The value is ignored for HF readers.

RxAntenna: Set the Receive Antenna. Possible values are 1 through 4. The software attempts to use any configured value, so it is up to the user to make sure this number is reasonable for the reader in use.

SerialDelay: When present and greater than 0, enforces a minimum number of milliseconds between the start of any outgoing serial command and the end of the previous command.

SubReader: When present and True, engages application specific functionality on readers that are not in the zero position. Subreaders may encode different parts of an RFID chip in a pipelined process.

T: Sets the timeout for each command in milliseconds. For ThingMagic readers, this number is inserted into each command, and the reader will retry their commands until they succeed or the timeout expires. Feig readers use this number as the global Tag Reply Timeout (TRTO) setting, and return false BAD status 0x83 when this number is too small.

Warning: Due to proprietary design constraints, ThingMagic write commands may take up to 500 mS regardless of the Timeout setting.

Tagname: In multifrequency jobs in which the global tagname is `VariesByReader`, this sets the tag for the given reader.

TCUBaud: Sets the baudrate for the TCU attached to this reader. This is just a number (115200, 57600 or 38400), since all TCU firmware uses the “N81” RS232 protocol bits.

TcuMarkWindowStart: As of RevG, this setting is unused but still shows up in JOB files.

TCUPort: Sets the serial port (0-based, so COM1 equals port 0) for the TCU attached to this reader. For standard ThingMagic applications, this is the same as the reader.

TCUPresent: Configures a TCU attached to this reader. For Standalone applications, this setting is FALSE.

TcuStepperRamp: Sets the number of steps over which the dynamic TcuStepperSpeed will ramp from a maximum value down to the configured value.

TcuStepperSpeed: Sent to the TCU when TCU flag UseStepper is set. The TCU stepper function works by pulsing the stepper controller at high speed. This number sets the number of TCU control cycles between state changes of the stepper pulse line. In most TCU firmwares, the relevant control cycle is around 100 uS, so a setting of 16 makes each stepper pulse around 3 mS ($0.1 \text{ mS} * 16 * 2$). Values are from 1 to 255, but most setups stall below a setting of 10 because the step motor is trying to move too fast.

TxAntenna: Set the Transmit Antenna. Possible values are 1 through 4. The software attempts to use any configured value, so it is up to the user to make sure this number is reasonable for the reader in use.

Type: Sets the type of the reader. Possible values as of **Rev G**:

- ThingMagic Vega
- ThingMagic Nano
- FEIG LRU UHF
- FEIG ISC/MR HF
- FEIG CPR HF
- Voyantic Tagsurance
- TriggerSpy

UidAsCsv: When CsvMerge is True and this value is 0 or higher, the current UID value is substituted for the corresponding CSV field. A value of -1 disables the feature. Other negative values are undefined.

UidLookup: When True, the current UID is used to look up the desired line of the selected CSV. The UID will be in field 0. The rest of the CSV line will be copied to a new CSV line shifted left one comma from the original. The new line will be used as if it were the next line in a standard CSV job.

UserMemoryFile: The DAT file for the reader, which sets contents of the UHF User bank or HF data blocks.

Verbose: When True, sets the display of verbose debug data including all commands between PC and TCU or RFID Reader.

W: The most recent variable value written to a chip in Serialize mode.

WriteId: The most recent complete EPC written to a chip in Serialize mode.

X, Y: As of RevG, this setting is unused but still shows up in JOB files.

Z: (UHF only) Sets the power for the test. May be entered as a floating point number, e.g. $Z=12.0$ and $Z=12$ both set power to 12 dbm. Range for ThingMagic Vega is 5..30 dbm. Range for ThingMagic Nano is 0..30 dbm. Range for Feig LRU1002 is 20..33 dbm.

7.4 **[ReaderN_TCUCompareBuffer]**

The data in this stanza is transmitted to the reader's TCU and used with certain Operation Table commands. See the LineLogix User Guide and OpTable Supplement documents for more information on Operation Table jobs.

7.5 **[Buttons]**

The first 8 lines of this stanza are processed per the Programmable Button Reference in section 4. Any blank line is mapped to an empty button.

If the stanza is shorter than 8 lines, only those lines present will be processed. If longer, only the last 8 lines will be displayed as buttons.

Any buttons in `site.dat` are processed first. If there are more than 8 buttons in `site.dat` and the job file together, the first ones will be unavailable as buttons.

7.6 **Site.dat**

One of the biggest problems in LineLogixPC site administration has been a large collection of job files with many common elements and just a few differences. As of LineLogixPC version 1.12, you can put the information common to all your jobs into a single job file called `site.dat`. LineLogixPC now reads this file before the job file you specified. For example, if you want all your jobs to have the same readers and buttons, you can move those sections of the job files to `site.dat` and remove them from the individual job files. The individual files then contain just the important information: tag type, test parameters, etc.

In the event of a conflict between `site.dat` and an individual job file, the job file usually wins, with the job file value overwriting the `site.dat` value. Numeric values can have minimum and maximum levels set, and initial values forced, by special `site.dat` entries. Note the leading dollar sign ('\$') and trailing command ("min", "max", or "force"):

```
$tcustepperspeedmin=1 sets a low limit on TCU stepper (nip) speed.  
$tcustepperspeedmax=250 sets an upper limit on the same parameter  
$serialdelayforce=5 sets this parameter to 5 and ignores the job file value.
```

8. **Log File Reference**

8.1 **Heading**

Each log file begins with some heading information, as shown in the example. Heading lines are identified with source "SUMMARY."

8.2 **Body**

For each reader configured with `LogIDs=TRUE` in the job file, LineLogixPC records every line sent by the reader to the PC in the log file, identified by the reader's name. In

the example below, Reader0 was given the name “TM” in the job file.

Notes:

- The placeholder “NoId” is provided on the “bad” lines instead of a tag ID to make parsing a little easier.
- A tag ID will be provided if the tag responds even one time, even if the tag fails the configured test.

8.3 Summary

Each log file ends with some Summary information, as shown in the example. The Summary section starts with the End Time line and extends to the end of the log file.

8.4 Log File Location and Naming Convention

Log files end with the suffix “.txt.” They are stored in the directory given by global setting LogDirectory. If that setting is not given, log files are stored in a subdirectory named “Output.” Each log file is initially named for the job name and start date, for example:

```
Gen2TI96bits_6inch.job.26Aug05_1410.txt
```

If the job file is so configured, log files will be renamed to “RollID.txt” where “RollID” is entered by the operator after pressing End Roll, for example:

```
RollID_POxxxxyyzzzz.txt
```

8.5 Sample Log File

```
0, SUMMARY, Job, +++simulate.job,
0, SUMMARY, Operator, Larry,
0, SUMMARY, Start Time, Thu Feb 04 16:43:28 2010,
0, SUMMARY, LineLogixPC Version, Version 4.01 02Feb2010,
0, Reader0, Versions, Simulated Reader Active, SimulatedIndustRFID
0, Upstream, 1, OK, 062220070000000000000033EB
1, Upstream, 1, OK, 062220070000000000000033EC
2, Upstream, 1, OK, 062220070000000000000033ED
3, Upstream, 0, BAD, 062220070000000000000033EE
4, Upstream, 1, OK, 062220070000000000000033EF
...
19, Upstream, 1, OK, 062220070000000000000033FE
20, Upstream, 1, OK, 062220070000000000000033FF
21, Upstream, 1, OK, 06222007000000000000003400
22, Upstream, 1, OK, 06222007000000000000003401
23, Upstream, 1, OK, 06222007000000000000003402
0, SUMMARY, End Time, Thu Feb 04 16:43:31 2010
0, SUMMARY, GOOD, 22
0, SUMMARY, BAD, 2
```

1, SUMMARY, No Reader 1
2, SUMMARY, No Reader 2
3, SUMMARY, No Reader 3
4, SUMMARY, No Reader 4
5, SUMMARY, No Reader 5
6, SUMMARY, No Reader 6
7, SUMMARY, No Reader 7
8, SUMMARY, No Reader 8
9, SUMMARY, No Reader 9

This log file was produced by a Simulated ThingMagic Serialize job with
LogIDs=TRUE. X and Y do not apply to ThingMagic, so the maximum Good count is
1.

9. Helper, Master, and Launcher Applications

LineLogixPC has a number of Application Program Interfaces (APIs) accessible via network sockets. These APIs are useful in:

- Expanding LineLogixPC functionality via Helper applications,
- Allowing a Master application to launch LineLogixPC with desired command line parameters, then monitor and control LineLogixPC via socket messages.

9.1 *Helper Specification*

Helper connections are defined in the JOB file. There can be multiple Helpers in a job as long as they have unique command lines. If all jobs at a given site require the same Helper or Master specification, the lines can be moved to `site.dat`.

```
helper=(127.0.0.1,8082)Helper\GLCsvHelper  
app=csvfeed;gagemax=7200;bcdelta=8;comout=44;baudout=38400
```

In this sample Helper specification:

- LineLogixPC will open a TCP Client socket with the parameters in parentheses: 127.0.0.1 (localhost) and port 8082
- then run the command line (from Helper to the end)
- monitor the process and restart it if necessary
- communicate over the socket

The Helper application must open a TCP Server socket at the corresponding address. Both sides use heartbeat messages to monitor socket health. Both sides tear down and rebuild sockets in response to a socket fault.

Most Helper applications have been provided by GlueLogix for application specific functionality. It is possible for third party applications to be integrated by GlueLogix via the Helper interface, at customer request.

9.1.1 **Helper Message Protocol**

Most Helpers use a text protocol and process at least the “startroll” and “endroll” messages. Further message protocol features are application specific and documented with specific Helper programs.

9.2 *Master Specification*

Master connections are defined in the JOB file. There can be only one Master in a job. If all jobs at a given site require the same Helper or Master specification, the lines can be moved to `site.dat`.

```
master="(127.0.0.1,8081)"
```

In this sample Master specification:

- LineLogixPC opens a TCP Client socket with the parameters in parentheses: 127.0.0.1 (localhost) and port 8081,
- and maintain the socket, reopening it as necessary.

The Helper application must open a TCP Server socket at the corresponding address.

9.2.1 Master Command Protocol

Master,StartRoll<CRLF> to LineLogixPC

Master,EndRoll<CRLF> to LineLogixPC

Master,EncodeDataN,<"barcode" data><CRLF> where N is the reader number (usually 0 for single reader setups). To LineLogixPC. The data can start with a letter U, which is a destination code, for Gen2 USER bank or HF tag data memory. Other destination codes will be added as needed by applications. Data in this message can be further parsed by a python Macro.

LineLogix,UTidN,<UID/TID in ASCII-HEX><CRLF> To Master. N is the reader number.

Feigl: 1 OK 008029c3f2097204,12345678 Wed Nov 04 12:27:27 2020 To Master. This is an example of the echo of Feedback tab lines to the Master.

Heartbeats: Every 5 seconds when idle, LineLogixPC sends "?\n" as a heartbeat message. If that causes an exception, the socket is closed and reopened. LineLogixPC does not check for an echo of the heartbeat.

The normal flow of a master session follows this pattern:

- Start LineLogixPC and set up the Master socket
- Wait for Start Roll from the Master
- Loop for each tag:
 - send the UID or TID to the Master
 - receive the computed encode data from from the Master
 - encode the data
 - echo the Feedback line to the Master
- Repeat until the Master sends End Roll.

9.3 Launching LineLogixPC

A Launcher program would set up JOB, DAT, MACRO, or CSV files for LineLogixPC, then run it with command line parameters (described above) as needed. A Launcher can continue to use the Master socket interface, or simply allow the LineLogixPC human interface to function as configured.

Most modern programming languages provide convenient interfaces to start and monitor operating system processes:

- C, C++: `system`, `fork`, `spawn`
- Python: `subprocess.Popen`
- C#: `class Process` with its events and methods

9.4 *Sample Code*

The Master Socket Server sample project is available from GlueLogix on request, and illustrates best practices for launching and managing LineLogixPC via a master Socket, and for maintaining a robust TCP socket using heartbeats.

10. Encoding

10.1 *TCU Based Encoding*

The LineLogix system can execute triggered encoding in real time on the TCU board. The Serialize and Sequence jobs both support encoding in this mode. Data transformations are minimal in the standard firmware. Even custom code is limited by the capacity of the Parallax Propeller chip used in the TCU.

Complex data transformations can be handled in real time by the GlueLogix Barcode Translator.

Triggered encoding with tags in continuous motion must be carefully optimized and may only work in special cases. The general solution is `EncodeInMotion`:

<http://gluelogix.com/EncodeInMotion.php>

TCU based encoding is described further in the LineLogix User Guide.

10.2 *PC Based Encoding*

Most LineLogix encoding systems are implemented on the GlueLogix nip driven rewind machine, or other platform that can stop each label over an antenna and transfer control to the PC with a Read message ("`>Rxxxxn`" where `xxxx` is the Gap Count). The PC then executes arbitrarily complex data transformations including database lookups and encryption computations. After encoding the computed data, LineLogixPC transmits status to the transport so it can manage bad tag marking, and restarts the transport.

In both Standalone and Rewind modes, PC Based Encoding happens when the `WriteBlockData` test is selected and an `>raction` is present in the job file reader stanza. In most cases, TCU flag `SuppressReaderCommand` is used to prevent the TCU from running any commands before passing control to the PC. One exception would be a job that reads HF UID, then passes control to encode a downstream UHF tag.

10.2.1 *The Sector (DAT) File*

The file that defines static chip memory contents was named the Sector file in development, and has come to be called the DAT file in deployment. It consists of data and metadata. All metadata must appear at the top of the file. The data is ASCII text, but

whitespace is ignored, so whitespace must be represented as hex bytes.

Mifare Ultralight NFC Data

```
[tagname Mifare UltraLight]
[lockbytes 00 00]
\x99H\x00\x00\xE1
\x10\x06\x0F\x03
\x0D\xD1\x01\x09
U\x04gogo.com\xFE
\x00\x00\x00\x00
\x00\x00\x00\x00
\x00\x00\x00\x00
\x00\x00\x00\x00
```

Tagname choices are:

- UHF Gen2
- HF ISO15693
- Mifare Classic/1K
- Mifare UltraLight
- NXP NTag203
- NXP Ntag21x
- NXP Ntag4xx
- HF Gen2
- Mifare Plus SL0
- Mixed by Reader

Lockbytes vary slightly by chip. For Ultralight and NTag, the two bytes correspond to the two Lock bytes in page 02 of the chip. “FF FF” locks all pages up to page 15 (64 bytes). Some chip variants have additional lock bytes near the end of memory; these values are set by memory mapped data bytes.

Ultralight/NTag data starts at page 03, the One Time Programmable (OTP) page, which holds the Capability Container (CC) in NDEF usage. This chip also supports RAW metadata, described below, which lets you set a different start point.

The Data section in this file starts with the hex byte \x99 (0x99 in C notation, or 153 decimal). The DAT record here was created with the LineLogixPC NDEF Editor. It contains some binary flags and two length bytes. The length bytes have to change whenever the NDEF record length changes.

The NDEF payload in this file is `https://gogo.com` (a made up website). The `https:` prefix is set by the \x04 byte before the URL. The NDEF Editor manages that abbreviation. Other choices are given in the NFC Forum NDEF documentation.

Since this DAT file does not include the RAW metadata tag, its data section is interpreted as NDEF data. LineLogixPC parses the DAT file contents and isolates the original NDEF record. For each chip, any Barcode (or OCR or other vision data) or CSV data is substituted for the original payload. If necessary, the start of the substituted string is

trimmed to take out the text represented by the prefix. That is, in the example URL `\x04gogo.com`, with substitution data `https://gogo.com?123456`, the final encoded data would be `\x04gogo.com?123456` and the NDEF record's length bytes would be adjusted.

If incoming Barcode or CSV data includes a URL prefix, as when scanning QR codes, the DAT file URL prefix MUST agree with the incoming data. That is, you cannot use a DAT file set for https with barcode data that is just http. Bad URLs result.

This DAT file's NDEF record data ends at `\xFE` on line 6. The `\x00` bytes after that are meant to erase old data on the chip. Nonzero data in this region can be proprietary signatures, or end-of-memory configuration data. Such data is not changed by Vision/Barcode or CSV substitution features, but can be set chip-by-chip using the Macro feature.

ISO15693 Raw Data

```
[tagname HF ISO15693]
[lockbytes 00]
[format RAW@0]
0000
```

For ISO15693 chips, lockbytes `\xFF` would lock all written 4-byte pages.

The RAW metadata tag in this file has an optional address, expressed in hexadecimal. To start data at page 04, i.e. byte 16, the tag would be `[format RAW@10]`, where 10 is the hex equivalent of 16 decimal.

The data in this file is written as four ASCII zero '0' characters, the same as `\x30\x30\x30\x30`. Barcode/Vision or CSV data would be substituted directly with no NDEF formatting.

The ISO15693 chip does support NDEF data, although it is not shown in this example.

UHF Gen2 User Bank Data

For UHF Gen2 chips, one of two standard DAT files must be selected:

`RawUhfGen2.dat`: specifies USER bank data per the rules for RAW data representation in HF chips. This file is required if Barcode/Vision or CSV data is to be written to the USER bank, but breaks the encoding process on chips with no USER bank.

`EmptyUhfGen2.dat`: specifies that no USER bank is available. This DAT file is REQUIRED for chips with no USER bank.

Mifare Classic and Plus

DAT files for Mifare Classic and Plus use the [sector metadata key to set up access to specific parts of the chip. Here is a sample stanza that writes an NFC Capability Container (CC) to sector 0 of a Classic chip:

```
[sector 0]
[lockbytes 00 00 00 04]
[extrabyte C1]
[key A FFFFFFFFFFFFFF]
[new key A A0A1A2A3A4A5]
[new key B FFFFFFFFFFFFFF]
\x14\x01\x03\xE1\x03\xE1\x03\xE1
\x03\xE1\x03\xE1\x03\xE1\x03\xE1
\x03\xE1\x03\xE1\x03\xE1\x03\xE1
\x03\xE1\x03\xE1\x03\xE1\x03\xE1
```

Lockbytes in this case are a LineLogixPC specific representation of Mifare Classic security bits, set by the LineLogixPC NDEF Editor. [lockbytes 00 00 00 04] does not lock the sector.

[extrabyte sets up byte 08 of the sector trailer, which has no special function in the Classic chip.

[key A sets up the key to authenticate the sector at the start of the process. If this key fails, a range of well known keys is tried, including all-zero, all-one (shown), and the standard NFC keys.

[new key A and [new key B set up the keys to be encoded.

NTAG 21x

The Ntag21x chip type serves all NXP NTAG and Ultralight EVn variants with special function bytes at the end of memory, including dynamic lock, mirror, password and tamper setup.

NTAG 4xx

See <https://gluelogix.com/LineLogixNtag424dna.php>

Mifare Plus SL0

Undocumented, will be disclosed as needed.

DesFire and up

As of 2022, all GlueLogix code for encoding DesFire and payment chips has been custom and outside the LineLogix system.

10.2.2 PC Based Barcode Capture

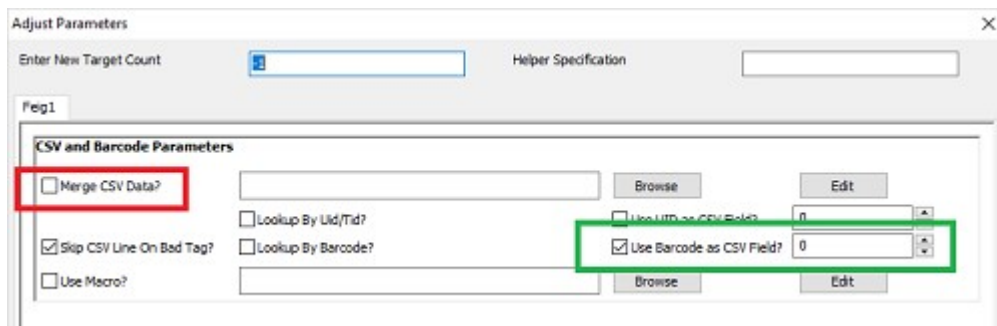
Depending on settings, a barcode (including QR, datamatrix or OCR string) can be provided to LineLogixPC by a number of mechanisms:

- USB Connected Camera with external Helper application

- USB Connected Barcode Scanner
- RS232 connected Barcode Scanner
- TCU store and forward code from upstream of the antenna
- Dialog Box for hand entry

TCU Store and Forward

To set up a job in which the barcode contents are written to each chip in turn, you must go to the Parameter screen and check “Use Barcode as CSV Field”. Leave the number at 0 and DO NOT check “Merge CSV Data”. This will treat each barcode in turn as if it had been read as the first field of a CSV line. See the discussion below of CSV schema for LineLogixPC encoding.



Note: If you were to check “Merge CSV Data”, the system would prompt you for a CSV file and read each line of the CSV in turn. Then it would substitute barcode data if present for the numbered field of the current CSV line when it went to encode each chip.

This excerpt from the TCU Check diagnostic illustrates how barcode data is stored:

```
Q0E!
Slot Barcode UID EPC Compare Gap BarCode CmdDT GapDT
0: X 0000000000000000 00000000000000000000000000000000 0 0 0 0 0
1: X206312370 0000000000000000 00000000000000000000000000000000 0 4022145 4022178 0 4022145
2: X206312369 0000000000000000 00000000000000000000000000000000 0 4031440 4031473 0 9295
3: X206312391 0000000000000000 00000000000000000000000000000000 0 4033216 3809881 0 1776
4: X206312367 0000000000000000 00000000000000000000000000000000 0 4035472 4035504 0 2256
5: X206312366 0000000000000000 00000000000000000000000000000000 0 4037893 4037926 0 2421
6: X206312365 0000000000000000 00000000000000000000000000000000 0 4040088 4040121 0 2195
7: X206312364 0000000000000000 00000000000000000000000000000000 0 4042731 4042763 0 2643
8: X206312363 0000000000000000 00000000000000000000000000000000 0 4043473 4043505 0 742
9: X206312362 0000000000000000 00000000000000000000000000000000 0 4044301 4044334 0 828
10: X206312361 0000000000000000 00000000000000000000000000000000 0 4064301 4064336 0 20000
11: X206312360 0000000000000000 00000000000000000000000000000000 0 4065047 4065079 0 746
```

Where each line describes a machine cycle. The X at the start of each code means it has already been sent to the PC. The TCU tracks machine cycles based on its GAP input, usually a label sensor. In order to forward the correct code to the PC for each cycle, the TCU must be set for the distance between camera and antenna. That setting is made in TCU parameter 0 (see the LineLogix System User Guide for the TCU reference). In each LineLogixPC Reader tab, that parameter is normally called “Inlay Repeat”, but that usage only applies to encoder based systems, which are rare. The prompt can be changed with this job file entry in the [reader0] stanza:

lengthprompt=Barcode Delta

LineLogixPC

File Feig ICU Help

Feig1: 0 (0.00%) GOOD 0 BAD NOT processing Tags

Feedback Job Feig1

Redisplay Set Power

TCU

Units for this tab

Decimal Counts Barcode Delta 6

Web Travel from Label Start to Inlay Entering Test Section 0

Web Travel from Label Start to Inlay Exiting Test Section 0

Web Travel from Label Start 2048

Flags. See LineLogix System Manual for Evaluation

UsePeripheral

This technique only works if the barcode contains the entire data to substitute. For example, if you have a DAT file with the NFC URL <http://someplace.com>, and your stock is printed with codes for <http://elsewhere.com>, then you could use this technique to encode your barcode data as NFC URLs.

If, however, your DAT file describes <http://myplace.com> and your barcodes are just a number, and you want to end up with URLs like <http://someplace.com/barcodeNumber>, then you need a Macro. See below.

10.2.3 PC Based CSV Operations

In HF NFC jobs, the CSV schema depends on the number of CSV fields. Most NFC jobs have just one field, a URL or TEXT record set up in the DAT file. Each CSV line will therefore have one field. If it has more, the extra fields will be ignored. If the CSV line has fewer data fields than the NDEF message has records, the data in the DAT file will be

programmed.

HF schema for NFC:

```
NFC1[,NFC2[...]]
```

As of Rev G, LineLogixPC supports just one field for RAW data substitution. It is a simple replacement of the CSV data for the DAT file contents at the offset stated in the DAT file.

HF schema for RAW data:

```
Data
```

The UHF schema is different because of the memory architecture of the UHF Gen2 chip.

UHF schema:

```
[user data],epc data[,access[,kill]]
```

example line with User data, 12 byte EPC and Access data but no Kill password:

```
123456789abcdef,0102030405060708090a0b0c,87654321
```

In CsvLookup and BarcodeLookup jobs, the lookup value is always in CSV field 0, and fields 1 and above have the functions of fields 0 and above in non-lookup jobs:

HF Lookup schema for NFC:

```
Barcode/UID,NFC1[,NFC2[...]]
```

HF Lookup schema for RAW data:

```
Barcode/UID,Data
```

UHF Lookup schema:

```
Barcode/TID,[user data],epc data[,access[,kill]]
```

example of barcode to EPC with no User data:

```
XYZ,,0102030405060708090a0b0c
```

10.2.4 PC Based Encoding Macros

A LineLogixPC Macro is a file of Python code that prepares data for encoding. These comments from a actual macro document the API and illustrate its usage:

```
#API: rc,self.szBarcodeTagId,ldata = self.DoMacro(self.LastTag, self.szBarcodeTagId, ldata)
# def DoMacro(self, pszUid, pszBarcode, plData):
#     rc = False
#     tagid = pszUid
#     barcode = pszBarcode
#     ldata = plData
#     ... exec macro ...
#     return rc, barcode, ldata
```

At a key point in the encoding process, the Macro is called with three parameters:

1. the current UID, EPC or TID as appropriate for the job

2. the current barcode
3. the current CSV line, separated at the commas into a list of strings

The Macro must set three values for its evaluating function to return:

1. a return code “rc,” True for success
2. the current “barcode.” which can be changed by the macro
3. the current CSV line “ldata”, which can be changed by the macro

As of **Rev G**, the python program that runs the macro imports Python hashlib, Crypto, sql, sql.aggregate and sql.conditionals. All functions of those modules are available to Macros.

Note: Each line in the file is evaluated separately by LineLogixPC, so multiline control structures are not allowed and “else” clauses do not seem to work. An if statement normally rendered as:

```
if 1 > 0:
    print "ok"
else:
    print "crazy"
```

would have to be entered in the macro file as:

```
if 1 > 0: print "ok"
if 1 <= 0: print "crazy"
```

This sample macro reverses the bytes of an HF UID, reformats it into a 12 byte EPC and generates a new UHF CSV line with the new EPC data:

```
rc = True
oldTagid = self.LastTag.upper()
tagid = ""
for bix in range(len(oldTagid)-2, 0, -2): tagid += oldTagid[bix:bix+2]
ldata = ["" ,"0000000000" + tagid]
```

The Macro name is set by-reader in the Parameters dialog. In the JOB file, the feature is enabled by csvmacro=True and the file name is saved as macrofilename.

10.2.5 PC Based Dual Frequency Encoding

Set up a Standalone or Rewind job that does what you want on one reader. The first reader should be the data source. That will typically be the HF reader since the HF UID cannot be changed, and is often an input to UHF encoding computations.

Once the first part of the application is working, make a second job with the second reader and debug that one.

Once both jobs are working, copy the [reader0] stanzas of the second job to the first job, changing reader0 to reader1 in the headings. Then continue to:

- Set global NumberOfReaders to 2
- Under [reader0], set childreader=1

- Under [reader1], set parentreader=0

The childreader setting makes the Reader0 object save its last result in variable LastTcuMessage, which you can inspect using File, Debug:

```
self.roa.jr[0].LastTcuMessage
```

The parentreader setting makes the Reader1 object reach out to Reader0 for information rather than reading its own CSV or Barcode data.

Another setting that may be helpful is readtidinstandalone=12, which forces the capture of 12 bytes of UHF TID data for display and logging

11. Networking

Although LineLogixPC has no intrinsic facilities for remote management, job file maintenance or log file maintenance, it is a PC program. All major PC operating systems have some facilities for file sharing. Some have facilities for remote login and remote windowing. Site administrators are free to use those facilities to create the systems they need. GlueLogix is available to consult on specific applications.

As of 2022, all LineLogix systems are shipped with an Internet connection to a remote desktop solution. Such PCs should be connected to the Internet before contacting GlueLogix for help that might lead to the need to log in remotely.

12. Quality Assurance

This is the most important section of the manual. Setup errors will always be found quickly if proper QA procedures are followed.

Failure to perform QA can lead you to ship defective product.

Please be sure to read, understand and follow the instructions in this section.

Although LineLogix is a highly capable system, incorrect setup can result in incorrect results. After reviewing all reference materials, possibly consulting GlueLogix, and running your first few tags, you are ready to begin your QA activity.

In short, the first 100 tags you process after any setup change or new inlay batch need to be checked. If any are wrong, fix the setup and try another 100. Once you can get through 100 tags without errors, start checking a percentage of your product.

Recommended progression:

Out of the first:	Read this many:	For this level of coverage:
100	100	100%
10,000	1000	10%
1,000,000	10,000	1%
10,000,000	A different 10,000	0.1%

12.1 Validation Testing

The 100% and 10% checks can be thought of as validation tests because they are validating your setup. Here are the most common validation tests:

- Each tag should be read for value. If the tags are serialized, the sequence should be checked.
- Each tag should be checked for strength. A nonmetallic platform or other position marker should be provided some distance away from the test antenna. Every tag not marked bad should be readable on the platform. The reader power and read distance will vary by product, but one typical setup involves reading pallet tags at 18 inches and 12 dbm with a patch antenna.
- Each tag with a bad mark should be dead, incorrectly coded, or weak per the strength test.

12.1.1 Typical Results

LineLogix systems with ThingMagic have been validated to less than 0.5% error. The most common errors with ThingMagic are false positive Write operations on moving webs when the web is too fast or the power is too high. When the web is too fast, the LineLogix counter is often wrong, i.e., lower than the machine's counter. When the power is too high, the finish roll will have repeated tag IDs coupled with uninitialized or

partially encoded tags.

LineLogix is likely to pass (i.e., not mark) inlays from some manufacturers whose tests are more restrictive. If that happens a lot, the operator should consider lowering the job's power. If that action is taken, extra QA should be performed against the risk of falsely marking good inlays bad.

12.1.2 Calibration Rolls

It is smart to keep a short validated roll of your most common product, with a few known-bad tags. New operators should run one of these calibration rolls at the start of every shift to check their technique as well as to check the system. Experienced operators should run the calibration roll once per week, or before using the system after it has been idle for a while.

Because some errors appear only rarely, the calibration roll should hold at least 500 inlays.

12.1.3 Inspection Lamp

Many facilities that use LineLogix on-press position a lamp under the web, so that the press operator can compare LineLogix' marks with marks from the inlay supplier. In most cases, these marks should agree.

12.2 *Production Testing*

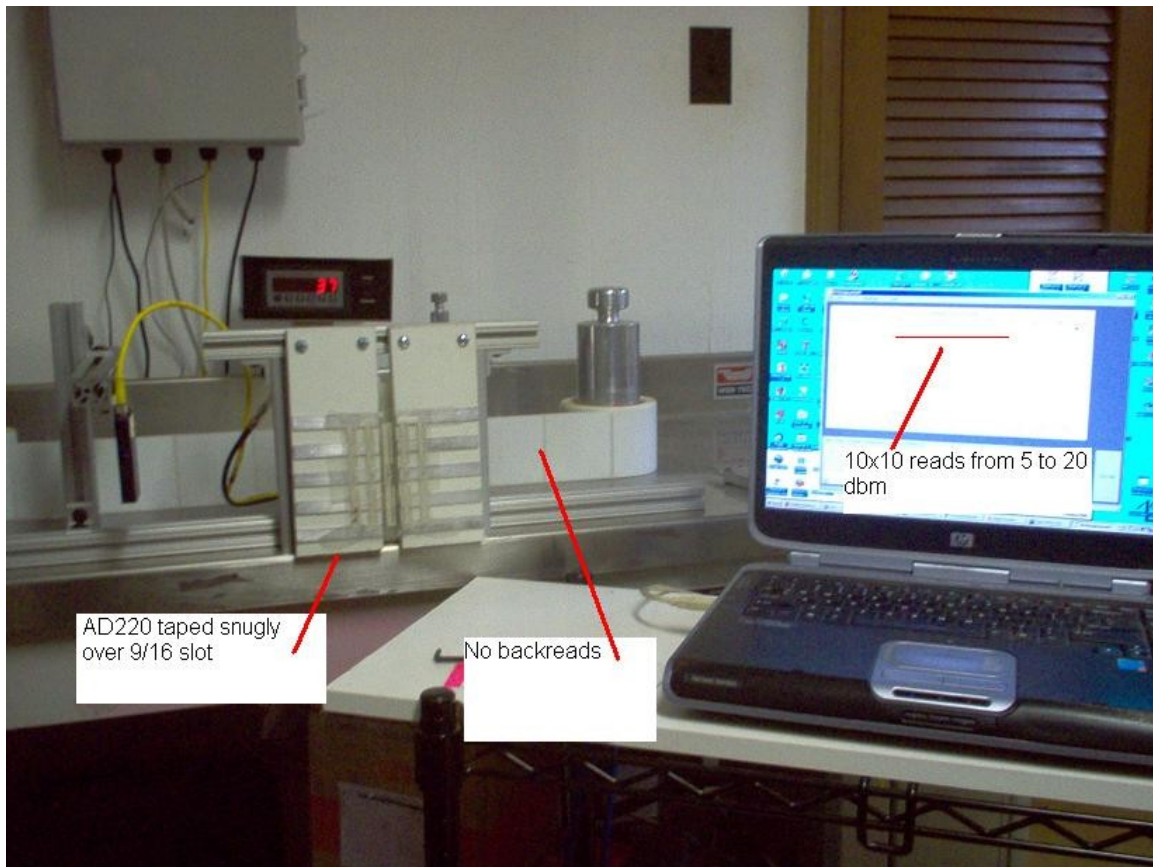
One simple way to accomplish the production-time 1% and 0.1% checks is to read the last tag in every finished roll using an RFID handheld. The operator could compare that read with the work order, and fill in a check box on the work order showing that QA is complete.

13. Job Setup

You should create a new job for each inlay type you process. The structure of job files, bitmaps, setup tabs and menu items are all described in detail above. You must be logged in as a Supervisor in order to change job settings.

13.1 Static Tests

Job setup begins before paper motion starts. Position an inlay or finished label on the paper web over the antenna, and use RFIDiagnostix to determine the best combination of power settings and antenna position for the new inlay.



This photo shows an actual setup, using RFIDiagnostix to determine the best power for an application. Note that additional inlays are present behind the antenna as well as on each side of the test section. The Inventory test of RFIDiagnostix will show when such “adjacent” tags are detected, because the number of reads will exceed the number of tries.

For this level of testing, the inlays must be encoded with unique IDs. RFIDiagnostix cannot distinguish responses from multiple inlays with the same tag ID.

Record the settings for transfer to LineLogixPC. If it fits your process and security regime, take a digital photo of this static setup.

13.2 *Trigger Setup*

With paper in place as for production, and the inlay in its final place in the label, position the web so that the inlay is slightly upstream of its ideal position over the antenna. That position was determined in the last step. The definition of “slightly upstream” will vary by target web speed. At 200 FPM, use about ¼ inch. At 30 FPM, do not worry about leading.

Once the web is in position, adjust the gap sensor or host PLC trigger (as on Tamarack units) so that the trigger signal is generated when the web reaches this position. For PLC systems including Tamarack, see your operator documentation for those instructions. For gap sensor systems, see the LineLogix System Manual and/or the documentation that came with your sensor.

If it fits your process and security regime, take a digital photo of this static setup.

13.3 *Job Setup*

1. Start LineLogixPC.
2. Load the existing job for the inlay closest to your new inlay.
3. Save the job (File, Write Job) with a new name that makes sense in your process.
4. Transfer the Static Test settings from RFIDiagnostix.
5. Edit the bitmap file.
6. Save the bitmap file to a new name.
7. Add notes to the bitmap file, documenting the setup for operators. Use the digital photos of your test setup if you took them.
8. In LineLogixPC, select the new bitmap file.
9. On the Reader tab, use the Redisplay button to make sure the new bitmap looks right.
10. Save the job again (File, Write Job).

13.4 *First Motion*

As you run the first tags with the new job, perform the Quality Assurance recommended above. Adjust job parameters as needed. Remember to Write the Job to disk (File, Write Job) after making changes.

14. *Troubleshooting*

14.1 *Startup Errors*

The startup processing performed by LineLogixPC is described in detail above. Once a

host PC and job is properly set up, most errors associated with initial configuration should not be seen again.

Here are the most likely Startup errors:

If you see this Symptom:	You may have this problem:
LineLogixPC crashes after Login. The Low Level Error message described below is displayed.	Bitmap file missing or specified incorrectly in the jobfile. Check the job file in a text editor like Notepad. If you cannot resolve the problem, retrieve the Low Level Error logfile described below and send it to GlueLogix along with the job file.
Cannot Open Serial Port	If your PC uses an internal serial port, this may indicate a serious internal error. With USB serial ports, check that they are plugged in. Try unplugging the USB serial port for a few seconds and plugging it back in. Use Device Manager to check the COM Port number and driver status of the USB serial port.
Cannot Find Reader	With LineLogixPC 4.x and above, try reading the TCU version to verify the communication link. Earlier versions of LineLogixPC will not allow this check. If this check fails, investigate cabling between PC and TCU, and power to TCU.
Cannot Find TCU	Try reading version from the reader. If that succeeds, investigate the wiring at the serial header within the TCU. Contact GlueLogix for detailed guidance.
Cannot Start Roll	If all the above are correct, the job should start. If you get this error, generate a Debug Report (see below) and contact GlueLogix.
Roll Starts, Tags are Moving, All are Bad	Stop the paper transport and repeat the Job Setup steps, above.
Roll Starts, Tags are Moving, All are Good	If you are sure some of the supposed good inlays are bad, check that there is no metal over the antenna that may be causing reflections. If that does not help, repeat the Job Setup steps above.
Roll Starts, Tags are Moving, LineLogixPC Counts do Not Change	Make sure that LineLogixPC says “Counting (or Reading or Writing) Tags”, then check the Trigger input. Open the LineLogix enclosure and check the LED next to the Gap Sensor input. It should be flashing with each label. See the LineLogix User Guide for more details.

14.2 Runtime Errors

Once the system is running, there are only a few symptoms that you are likely to detect;

however, those symptoms can have multiple causes.

If your setup has this error:	You may see this problem:
Power too high	Write: False positives, slid IDs, blank tags Read: False negatives
Power too low	False negatives
Overhead Reflections from Metal	Same as Power too high
Antenna baffles too wide	Same as Power too high
Antenna baffles too narrow	Same as Power too low
Trigger too soon or too late	Same as Power too low, possibly slid IDs on write
Web not touching or within 1/16 inch of baffle surface	Same as Power too high
Web too fast	Same as Power too high but also bad counts on the PC

14.3 *Bad Tags*

All operators should be trained to take bad-marked tags seriously. If a large number of tags are being marked bad by LineLogix, production should be stopped and the problem isolated.

The Stop On Error feature can be used to enforce this, but is not well suited to on-press installations because of the consequences of “stopping the presses.”

14.4 *Symptom Documentation*

When you contact GlueLogix for assistance, we may ask for information from your PC. This section tells you how to gather that information.

14.4.1 *Debug Reports*

If you run into trouble, or think you might be in trouble, generate a Debug Report using the LineLogixPC file menu. If this information is needed later for failure analysis, it will then be available.

14.4.2 *Screen Shots*

If you are asked to send a screen shot, these steps will work for most Windows PCs:

1. Look at your keyboard's Print Screen button. If the button just says PrintScreen (or PrtScr or other variant), just press it. If the key says something else as well, you probably have to press a function key to enable the Print Screen function. On many laptops, that is literally a key labelled “Fn”. Figure out what the function

- key is and use it with the Print Screen key.
2. Start Windows Paint:
 - On XP, click Start, then Run, then enter `pbrush`.
 - On Vista or Win7, click Start, then enter `paint`.
 3. Paste the screenshot into Paint using the menu Edit, Paste
 4. If the screen appears in Paint, save it. Otherwise, repeat steps 1 and 3 until the screenshot appears in Paint.
 5. Send the file to GlueLogix via email.