Spring 2013

Brass Ring Carousel Control System

Daniel Karwatka
Governors State University

Follow this and additional works at: http://opus.govst.edu/capstones
Part of the Computer Sciences Commons

Recommended Citation
http://opus.govst.edu/capstones/7

For more information about the academic degree, extended learning, and certificate programs of Governors State University, go to
http://www.govst.edu/Academics/Degree_Programs_and_Certifications/

Visit the Governors State Computer Science Department
This Project Summary is brought to you for free and open access by the Student Capstone Projects at OPUS Open Portal to University Scholarship. It has been accepted for inclusion in All Capstone Projects by an authorized administrator of OPUS Open Portal to University Scholarship. For more information, please contact opus@govst.edu.
Brass Ring

Carousel Control System

By

Daniel Karwatka

B.S., Governor’s State University, 2003

Master's Project/Graduate Project

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,

With a Major in Computer Science

Governor’s State University

University Park, IL 60484

2013
Abstract

In this age of e-commerce, use of automated industrial machines in warehousing is necessary for lowering labor costs and increasing productivity. In many cases the high cost of implementation and slow return on investment impede the installation of these systems. Furthermore, the ongoing maintenance costs of these system may also be a dissuading factor. BrassRing is an attempt to lower the software implementation and maintenance costs for industrial horizontal carousels. By utilizing open and free platforms along with inexpensive computer equipment a business may lower its installation and customization costs significantly while also reducing future maintenance and upgrade fees.
# Table of Contents

1.0 Purpose:..................................................................................................................... 5  
1.1 Business Case:........................................................................................................ 5  
1.2 Goals:....................................................................................................................... 5  
1.3 Development Environment:.................................................................................. 6  
1.4 Industrial Equipment Environment:....................................................................... 6  
2.0 Brass Ring Operations.............................................................................................. 7  
2.1 Pick Multi.................................................................................................................. 7  
2.2 Pick Singles............................................................................................................. 11  
2.3 Pick All.................................................................................................................... 14  
2.4 Put Batch............................................................................................................... 14  
2.5 Count Batch........................................................................................................... 14  
3.0 Import Transactions................................................................................................. 15  
3.1 Auto Import Transactions:..................................................................................... 16  
3.2 Pending Tasks Dashboard...................................................................................... 16  
3.3 Purge Transaction................................................................................................... 16  
4.0 System Configuration.............................................................................................. 17  
4.1 Pod Configuration:................................................................................................. 17  
4.2 CommChannel Configuration................................................................................. 19  
4.3 Carousel Configuration.......................................................................................... 21  
4.4 Light Tree Configuration....................................................................................... 23  
4.5 Sort Bar Configuration............................................................................................ 25  
4.6 Task Button Configuration..................................................................................... 27  
4.7 Configure Users...................................................................................................... 29  
4.8 Reason Configuration............................................................................................. 30  
5.0 Test Devices........................................................................................................... 31  
5.1 Testing Carousels................................................................................................... 31  
5.2 Testing Light Trees.................................................................................................. 34  
5.3 Testing Sort Bars..................................................................................................... 37  
5.4 Testing Task Buttons............................................................................................. 40  
6.0 Transaction Maintenance....................................................................................... 44  
6.1 View Edit Batch:.................................................................................................... 44  
6.2 View Edit Detail...................................................................................................... 45  
6.3 View Edit Header.................................................................................................... 47  
7.0 Database.................................................................................................................. 49  
7.1 Database UML........................................................................................................ 50  
7.2 Database Schema.................................................................................................... 51  
8.0 Utilities and Tools.................................................................................................... 55  
8.1 Serial....................................................................................................................... 55  
8.2 Serial to TCP/IP...................................................................................................... 55  
9.0 Security.................................................................................................................... 56  
9.1 Database Security................................................................................................... 56  
9.2 Serial Port Security................................................................................................. 56  
Appendix A – Pictures................................................................................................. 57
1.0 Purpose:
Create a carousel control system software that will reduce the implementation costs and realize a quicker Return On Investment (ROI) for businesses trying to employ an industrial carousel in their warehousing operation.

1.1 Business Case:
The company I work for installed a used industrial carousel and contracted with a firm to provide customized software for interfacing it with our Warehouse Management Software (WMS). Complete software installation was contracted to be $15k. Although the software was basically operational, a number of deficiencies were never corrected. We are unable to get any other contractor to correct the original software since it's closed source. The software is registered/locked to the computer on which the software was installed which prevents us from upgrading or replacing the equipment without having to employ the original vendor.

1.2 Goals:
• Control industrial carousels, light trees, sort bars and related equipment
• No operating system lock in
• No computer equipment lock in
• No industrial equipment lock in
• Open Source
• Failure tolerant
• Simple disaster recovery
• Run on any operating system supported by Java
• Run on commodity equipment (cheap)
• Build core classes for use in creating pod servers that can support any variety of clients
• Design equipment classes based on interfaces so additional class may be created for new equipment models without needing to change how applications address the pod
• Create a Java Swing application as an example of how to implement
• Unlimited number of carousels
• Unlimited number of light trees
• Unlimited number of pods
### 1.3 Development Environment:
- NetBeans IDE 7.3 (Build 201302132200)
- Glassfish-3.1.2.2
- Java 7 (1.7.0_17; Java HotSpot(TM) Client VM 23.7-b01)
- Java(TM) SE Runtime Environment 1.7.0_17-b02
- Apache Derby Network Server - 10.8.1.2 – (1095077)
- Super CSV 2.0.1
- CommTunnel 2.1.0.35
- ELTIMA Software Serial Port Monitor 4.1

### 1.4 Industrial Equipment Environment:
- Diamond Carousels with CCS-4 Controllers
- Diamond LTS6 Light Trees
- Diamond LTS6 Sort Bar
- Optional: RS232/485 to TCP/IP Converter
2.0 Brass Ring Operations

2.1 Pick Multi

Any Doc Id that has more than one pick is considered a Pick Multi. The number of multi-picks that can be processed at one time is limited to the number of locations on your sort bar.

1. Select Transaction->Pick Multi

2. Select the documents you wish to process. They will be displayed in priority order and document number order.

Note: You will be limited to the maximum number of position on your sort bar. You may re-sort the list by clicking on the column header.
3. Click on OK to proceed. Pressing cancel will abort the picking process and return you to the menu.

4. Register each document to a tote and to the sort bar. Totes are not required. This screen automatically fills sort bar locations in ascending order. Document IDs are also pre-filled in ascending order.

5. When you reach the last document to register selecting Finish will assign these orders to a batch and start the picking process.

6. All light displays and buttons will be cleared and the carousels will move to their next upcoming pick.
7. A light tree will illuminate indicating the location and quantity of the first pick. The screen will also indicate the active transaction quantity along with the item and location information.

8. The sort bar locations that will receive this product will also be illuminated with the quantity to place in the tote.

9. If all totes are satisfied press any task complete button and proceed to step 14.

10. When the user is unable to fulfill the request they should press Control-Q or click on the Qty Exception button.

11. The user must short the appropriate quantities. When complete select Done to proceed.
12. You will now be back at the Active Task windows where the new quantity will be displayed and the quantities on the sort bar will be refreshed with updated quantities.

13. Press any task complete button and proceed.

14. The light trees and sort bar displays will now be cleared.

15. The carousel that we just picked from will move to its next location.

16. The light tree(s) will now display the pick location and quantity on the next carousel in the round-robin sequence. We will go back to step 8 and repeat this process until all items have been picked.

17. When the last item is finally picked all displays will show DONE.

18. Once the sort bar is clear and all totes passed or removed the user presses task complete one final time to clear the displays and the carousels will be homed (sent back to bin 1) if this option is enabled.
2.2 Pick Singles

In this time of e-commerce, warehouses are receiving many more small orders than ever before. The worst of these is the single item order. When these are in abundance they have the ability to dilute the order pool causing an extremely low density wave.

To counteract this we isolate all orders with single picks into their own wave separate from multi picks. We further gain efficiencies by assigning a single sort bar location for these transactions. By eliminating sort bar registration we can speed up the assign orders to totes process.

Large Pick Single batches are exceedingly efficient and can rapid fire product out of the pod toward its next destination.

1. Transaction->Pick Singles
2. Select the documents you wish to process. They will be displayed in priority order and document number order.

![Select Documents](image)

Note: The is NO limit and the elect first button is effectively a select ALL button.
You may re-sort the list by clicking on the column header.

3. Select the documents you wish to process. They will be displayed in priority order and document number order.

4. Click on OK to proceed. Pressing cancel will abort the picking process and return you to the menu.
5. As each item is presented, you will be prompted to register each order to a tote if tote tracking is enabled.

6. Each active transaction will be presented to the user.

7. After pressing task complete the item is cleared from the sort bar.

8. The light trees and sort bar displays will now be cleared.

9. The carousel that we just picked from will move to it's next location

10. The light tree(s) will now display the pick location and quantity on the next carousel in the round-robin sequence. We will go back to step 5 and repeat this process until all items have been picked.

11. When the last item is finally picked all displays will show DONE.
12. Once the sort bar is clear and all totes passed or removed the user presses task complete one final time to clear the displays and the carousels will be homed (sent back to bin 1) if this option is enabled.

2.3 Pick All
Pick All operates the same as Pick Multi except that both multi picks and single picks.

2.4 Put Batch
Put Batch operates just like Pick All except product will be removed from the totes and placed on the carousel.

2.5 Count Batch
The Count Batch operates like the Pick All option except product is not removed from the carousel
3.0 Import Transactions

One of the key features needed to make this operate is the importing of transaction requests from the WMS system. While we could have easily used JDBC to import data from other databases, I've added a csv importer just to show some added flexibility beyond a flat file and JDBC.

The reality is that data imports are expected to be customized for every different installation. We deliberately wanted this product to be Open Source to allow this type of customization by any contractor. Even so, a basic csv import is included.

Default Import File Format: Comma Separated Variables, comma delimiter, quoted text.
Default File Path Name: \ImportExport\ctrequest.csv

Field/Column Layout:

<table>
<thead>
<tr>
<th>Col</th>
<th>Column Name</th>
<th>Data Format</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Transaction Type</td>
<td>Char(3)</td>
<td>PIC, PUT, or CNT</td>
<td>Carousel Transaction Type</td>
</tr>
<tr>
<td>B</td>
<td>Priority</td>
<td>smallint</td>
<td>0-32767</td>
<td>Ascending integer to allow transactions to be handled out of sequence.</td>
</tr>
<tr>
<td>C</td>
<td>Transaction ID</td>
<td>Char(3)</td>
<td>OE:12345:10:2</td>
<td>Unique transaction identifier</td>
</tr>
<tr>
<td>D</td>
<td>Doc ID</td>
<td>Varchar(30)</td>
<td>OE:12345</td>
<td>A number that identifies a group of transactions.</td>
</tr>
<tr>
<td>E</td>
<td>Transaction Location</td>
<td>Varchar(30)</td>
<td>01030134D1</td>
<td>Encoded location that directs transactions to a specific pod, carousel, bin, shelf and cell.</td>
</tr>
<tr>
<td>F</td>
<td>Item</td>
<td>Varchar(100)</td>
<td>SKU12345</td>
<td>Item SKU or other unique identifier</td>
</tr>
<tr>
<td>G</td>
<td>Request Quantity</td>
<td>Integer</td>
<td>5</td>
<td>The needed for the pic, put or count</td>
</tr>
<tr>
<td>H</td>
<td>User Defined</td>
<td>Varchar</td>
<td>Y N</td>
<td>Field for adding custom flags like create label, serialized, weigh_count, lot_no, serial_no</td>
</tr>
<tr>
<td>I</td>
<td>Description</td>
<td>Varchar(100)</td>
<td>Widget</td>
<td>Simple description of this product</td>
</tr>
</tbody>
</table>

Natural Order Multi Pick Test Data

<table>
<thead>
<tr>
<th>PIC</th>
<th>00:OE:2123:1:1</th>
<th>00:OE:2123:01020202B1</th>
<th>SKUITEM1</th>
<th>2</th>
<th>Multi Pick 2 Rec 1</th>
<th>03-01-2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC</td>
<td>00:OE:2123:2:2</td>
<td>00:OE:2123:01020305C1</td>
<td>SKUITEM2</td>
<td>5</td>
<td>Multi Pick 5 Rec 2</td>
<td>03-01-2013</td>
</tr>
<tr>
<td>PIC</td>
<td>00:OE:2123:3:4</td>
<td>00:OE:2123:01020305C1</td>
<td>SKUITEM3</td>
<td>8</td>
<td>Multi Pick 8 Rec 3</td>
<td>03-01-2013</td>
</tr>
<tr>
<td>PIC</td>
<td>99:OE:2234:7:5</td>
<td>00:OE:2234:01010557E1</td>
<td>SKUITEM7</td>
<td>10</td>
<td>Multi Pick 10 Rec 7</td>
<td>03-01-2013</td>
</tr>
<tr>
<td>PIC</td>
<td>00:OE:2345:8:8</td>
<td>00:OE:2345:0103014561</td>
<td>SKUITEM8</td>
<td>9Y</td>
<td>Multi Pick 9 Rec 8</td>
<td>03-01-2013</td>
</tr>
<tr>
<td>PIC</td>
<td>00:OE:2345:9:9</td>
<td>00:OE:2345:01030134D1</td>
<td>SKUITEM9</td>
<td>6</td>
<td>Multi Pick 6 Rec 9</td>
<td>03-01-2013</td>
</tr>
<tr>
<td>PIC</td>
<td>00:OE:2345:10:10</td>
<td>00:OE:2345:01030121C1</td>
<td>SKUITEM10</td>
<td>3Y N</td>
<td>Multi Pick 3 Rec 10</td>
<td>03-01-2013</td>
</tr>
</tbody>
</table>
3.1 Auto Import Transactions:
Launches a timed import to run in the background every 1 minute.

3.2 Pending Tasks Dashboard
The pod window displays a running total of all pending tasks. By default 0 is normal priority. Any number greater than zero is high priority.

These numbers are updated after every import and when a new batch is started.

Note: The background also displays the name of the Pod that is in use.

3.3 Purge Transaction
The purge feature allows removal of completed transactions. Without this the database may grow beyond a reasonable limit.

Note: In this alpha version it is set to remove all records. This is needed to be able to run and rerun the same data.
4.0 System Configuration

System configuration is available only to system administrators.

A pod consists of all of the equipment that must interact with the operator for the efficient processing of transactions. Within a pod we typically have, but are not limited to, carousels, light trees, sort bars, task buttons, etc.

It's recommended that all equipment be configured and tested prior to normal operations. This will reduce frustration and confusion. This is especially true when a device is configured to send back carriage returns causing communication failures that affect further transactions for other devices.

If the admin knows all of the configuration parameters he/she may configure them all at once or one device at a time. When using either method the pod must be defined first.

4.1 Pod Configuration:

The pod configuration is used to customize the user requirements at the pod workstation. This includes the user requirement to confirm the item and/or location on each transaction, homing of carousels, and where on the sort bar to place single picks. All other configuration settings are dependent upon this so it must be created first.
Pod No: This is a non editable field used only for the java persistence system to be able to retrieve and save entries. This is automatically assigned by the database.

Pod Id: The string sent by the WMS system to address this pod. Often this is seen as a numeric code within a larger location code. In some systems this is also known as the zone number.

Pod Name: This is the name we want to call this pod. This name is used on the command line to tell the software which configuration to load.

Item Pic Conf: Require item confirmation at the time of picking an item? When set to “Y” the user will be required to enter the item code to complete the transaction. Item code may be entered using bar code scanner, voice, etc.

Item Put Conf: Require item confirmation at the time of putting an item away? When set to “Y” the user will be required to enter the item code to complete the transaction.

Item Cnt Conf: Require item confirmation at the time of counting an item? When set to “Y” the user will be required to enter the item code to complete the transaction.

Home After Pick Batch: Send carousels back to location 1 after pick batch is complete.

Home After Put Batch: Send carousels back to location 1 after put batch is complete.

Home After Cnt Batch: Send carousels back to location 1 after count batch is complete.

Single Pick Sortbar No: Enter the sort bar number to use for single picks. During single picks only one sortbar location is used.

Single Pick Sortbar Display: The display location within the sort bar to use for single picks. Single picks keep up a high transaction rate by not requiring the assignment of totes to different sortbar locations.

Export Completed Batch: Export completed batch information back to the WMS system? “Y” will export the completed transaction to a file for use by the WMS.
4.2 CommChannel Configuration

To “talk” to a device it must be addressable via a commchannel. A commchannel is defined for each communication path to a device. Commchannels may be assigned to each individual device, shared communication “party” line, or to each device network. There is no limit to the number of channels a pod may address.

Note: Diamond uses a shared communication “party” line.

Channel Id: This is a non editable field used only for the java persistence system to be able to retrieve and save entries.

Channel No: Each pod has its own channel numbers. You may define any number from 1 to 32767.

Pod No: Enter the pod number to which this channel will be available.

Channel Desc: A verbose description for this channel. This will make it easier to identify a channel when there are multiples for a pod. It may help during problem diagnosis.

Address: Device TCP/IP address. This may be an address like 192.168.0.2 or DNS name like com1.pod1.com. localhost (127.0.0.1) is also supported.

Port: Device TCP/IP port.
Timeout: The length of time the network connection will wait until it assumes a failure. The number is specified in milliseconds. Valid entries may range from -1 to 32767. When -1, the system default will be used. When 0, wait time will be infinity.
4.3 Carousel Configuration

The carousel is the machine on which all product it stored. It's the machine that brings the items to the operator (also known as goods to person).

We define the equipment parameters here so that we know which pod it belongs to and how to communicate with it.

The practical maximum carousels for a pod is about 8 and most commonly will be set to 3 or 4. The actual number of carousels that can be associated with a single pod is limited to 32767 but due to speed there may be a limit of 100 or so. Further more, walking distances and physical space considerations probably place the extreme limit at 16 for a human controlled pod.

Carousel Id: This is a non editable field used only for the java persistence system to be able to retrieve and save entries.

Carousel No: This is the number embedded in the larger location code. In some systems this is equivalent to the aisle or row within a zone. This number is also used to sequence the round robin access, usually in ascending order.

Pod No: This is the pod number you want this carousel to belong to.

Address: This is the hardware device address used for communication. There may be different limits defined by different manufacturers. Diamond defines the address to be from 00 to 99.

Bin Count: This is the number of bins, or shelving units, installed on the carousel. All calls addressing bins higher than this number will be ignored.
**Retries:** This is the number of times the software will retry sending a command. Retries only occur when the carousel doesn't acknowledge successful receipt of the command. Failure may due to communication issue, bad command, or system busy (not in host mode).

**Retry Delay:** The amount of time between retries to allow the error to clear.

**Comm Channel No:** The number of a commchannel defined previously.

**Model:** This is a model number used to differentiate between different carousel Java classes. New custom classes may be integrated into this project without affecting its general operation.

The main reason for adding new models is to support new features and different communication protocols.
4.4 Light Tree Configuration

Light trees visually direct the operator as to the quantity and location of the product on the carousel. Each display on a light tree should line up with a shelf on the carousel.

Tree Id: This is a non editable field used only for the java persistence system to be able to retrieve and save entries.

Tree No: This is an arbitrary number assigned to differentiate between trees in a pod.

Pod No: This is the pod number you want this light tree to belong to.

Address: This is the “base” hardware device address used for communication. There may be different limits defined by different manufacturers. Diamond defines the address to be from 001 to 999.

Note: The shelf number is added to the base to get the actual display address.

Comm Channel No: The number of a commchannel defined previously.
Retries: This is the number of times the software will retry sending a command. Retries only occur when the tree doesn't acknowledge successful receipt of the command. Failure may due to communication issue, bad command, or system busy

Retry Delay: The amount of time between retries to allow the error to clear.

Left Carousel No: This is the carousel number physically to the left of this tree. Leave blank/null if there isn't one. Entering the same carousel number in Left and Right indicates center positioning like in a twin bin.
Brass Ring Carousel Control System

Right Carousel No: This is the carousel number physically to the right of this tree. Leave blank/null if there isn't one. Entering the same carousel number in Left and Right indicates center positioning like in a twin bin.

Note: In this software light trees do not belong to a carousel but to the pod. This allows us to implement an expanded number of features like pointing to a single carousel with more than one tree, or sharing trees.

Display Digits: The number of characters that can be shown on the physical displays. Display data will be trimmed down to fit this size. Any characters beyond this size will be dropped.

Orientation: Use V for Vertical and H for Horizontal. This is used to determine the set of chevrons to use. “<” and “>” are used for vertical trees while horizontal trees will not display any.

First Left Cell: The first cell number (far left cell) in the bin on the left. Not used at this time.

First Right Cell: The first cell number in the bin on the right. Any cell number >= to this cell will get a right “>” chevron. All others will get the left “<” chevron.

Note: this implementation assumes a grid assignment of cells within a carousel.

Model: This is a model number used to differentiate between different light tree Java classes. New custom classes may be integrated into this project without affecting its general operation. (The main reason for adding new models is to support new features and different communication protocols.)
4.5 Sort Bar Configuration

Sort Bars visually direct the operator as to the quantity and location of the product on the sort station. Each display on a light tree should line up with a tote position in the sort station. Essentially this is a light tree assembled in a horizontal design.

Bar Id: This is a non editable field used only for the java persistence system to be able to retrieve and save entries.

Bar No: This is an arbitrary number assigned to differentiate between trees in a pod.

Pod No: This is the pod number you want this light tree to belong to.

Address: This is the “base” hardware device address used for communication. There may be different limits defined by different manufacturers. Diamond defines the address to be from 001 to 999.

Note: The tote position number is added to the base to get the actual display address.

Comm Channel No: The number of a commchannel defined previously.

Retries: This is the number of times the software will retry sending a command. Retries only occur when the tree doesn't acknowledge successful receipt of the command. Failure may due to communication issue, bad command, or system busy.

Retry Delay: The amount of time between retries to allow the error to clear.

Displays: The number of tote positions with displays in the sort station.
Levels: Used to determine the number of displays on each level and to center the totes when less that the full number will be used.

Display Digits: The number of characters that can be shown on the physical displays. Display data will be trimmed down to fit this size. Any characters beyond this size will be dropped.

Orientation: Use V for Vertical and H for Horizontal. This is used to determine the set of chevrons to use. “<” and “>” are used for vertical trees while horizontal trees will not display any.

Model: This is a model number used to differentiate between different light tree Java classes. New custom classes may be integrated into this project without affecting its general operation. A sort bar is a special type of light tree. (The main reason for adding new models is to support new features and different communication protocols.)
4.6 Task Button Configuration

Task Buttons are a physical device that, when pressed, indicate completion of a task. Task Buttons may be located anywhere within a pod and may or may not be associated to a light tree location.

<table>
<thead>
<tr>
<th>Task Button Id</th>
<th>Task Button No</th>
<th>Pod No</th>
<th>Tree No</th>
<th>Address</th>
<th>Comm Channel No</th>
<th>Retries</th>
<th>Retry Delay</th>
<th>Model</th>
<th>Sort Bar No</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>909</td>
<td>1</td>
<td>0</td>
<td>909</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>911</td>
<td>1</td>
<td>0</td>
<td>911</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>912</td>
<td>1</td>
<td>0</td>
<td>912</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>914</td>
<td>1</td>
<td>1</td>
<td>914</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>104</td>
<td>1</td>
<td>1</td>
<td>104</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>204</td>
<td>1</td>
<td>2</td>
<td>204</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>304</td>
<td>1</td>
<td>3</td>
<td>304</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Task Button Id**: This is a non-editable field used only for the Java persistence system to be able to retrieve and save entries.

**Task Button No**: This is an arbitrary number that is used to order the polling sequence. These numbers are set on the hardware and may be preset by the manufacturer.

**Pod No**: This is the pod number you want this light tree to belong to.

**Tree No**: This should be the same as an existing tree no. When the button is associated to a tree certain commands sent to trees will affect both the tree and buttons. Because this command has already been executed for both tree and button it will not need to be issued to the button again.

**Sort Bar No**: This should be the same as an existing Sort Bar No. When the button is associated to a tree certain commands sent to bars will affect both the bar and buttons. Because this command has already been executed for both tree and button it will not need to be issued to the button again.
**Brass Ring Carousel Control System**

**Address**: The device address used for communications.

**CommChannel No**: The number of a commchannel defined previously.

**Retries**: This is the number of times the software will retry sending a command. Retries only occur when the button doesn't acknowledge successful receipt of the command. Failure may due to communication issue, bad command, or system busy.

**Retry Delay**: The amount of time between retries to allow the error to clear.

**Model**: This is a model number used to differentiate between different task button Java classes. New custom classes may be integrated into this project without affecting its general operation. (The main reason for adding new models is to support new features and different communication protocols.)
4.7 Configure Users

There are two different types of users. The first is the Admin who has all privileges and the other is the User that may do anything except items under the Tools menu.

User Id: A code used to identify the operator. This helps define access and is also stamped on each transaction. This should allow metric reports to be run later.

User Name: A longer or verbose user name.

Admin: Y/N field to determine if this user has access to the Tools Menu items.
4.8 Reason Configuration

Reasons are exception codes that are sent with a transaction when the requested amount is not available. This code is sent back to the WMS system through the export. This will allow the WMS to take further action based on this feedback.

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Display Order</th>
<th>Pod No</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>Wrong Item</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>Bad or Broken</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S</td>
<td>Shortage</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Code**: Simple 1 character code to send back to the WMS system.

**Description**: A longer more verbose description displayed to the operator for selection.

**Display Order**: The sequence the should be presented. The lowest ordered item will be the default.

**Pod No**: The pod number to associate the codes to. Different pods may need different lists and defaults based upon type of product being stored.
5.0 Test Devices

While configuring the equipment it's best to test your devices. This option is available to the Admin.

5.1 Testing Carousels

Recommended testing sequence:

1. Get status for carousel. This usually addresses the carousel not the bin. Make sure the command sent and the response are as you expect.
2. Clear the carousel. This also only addresses the carousel not the bin. Make sure the command sent and the response are as you expect.

3. Send the carousel one bin from where it's currently located. Make sure the command sent and the response are as you expect.
4. Send the carousel to the last bin. Make sure the command sent and the response are as you expect. Make sure the carousel presents (moves to) the appropriate bin.

5. Send the carousel one bin after the last bin. This should fail by not sending a command and response will show “ffff”. Make sure the command sent and the response are as you expect.
5.2 Testing Light Trees:

Recommended testing sequence:

1. Display a message on a middle light tree display. The message sent should be trimmed down the length of your display. Confirm that the message was illuminated on the correct display. If not the base address could be off, the equipment is not configured properly, or the model selected is not compatible. Make sure the command sent and the response are as you expect.
2. Clear the message. Make sure the command sent and the response are as you expect.

3. Display a message on every display. All displays should be illuminated. If any fail confirm that there truly is a device at that address.
4. Clear All displays. This will test the ability of any and all displays to be cleared. Depending upon the communication configuration and manufacturer's features, this may also clear other displays within the pod.

![Test Light Tree Communications](image)
5.3 Testing Sort Bars

Recommended Testing Sequence:

1. Display a message on a middle light tree display. The message sent should be trimmed down the length of your display. Confirm that the message was illuminated on the correct display. If not the base address could be off, the equipment is not configured properly, or the model selected is not compatible. Make sure the command sent and the response are as you expect.
2. Clear the message. Make sure the command sent and the response are as you expect.

3. Display a message on every display. All displays should be illuminated. If any fail confirm that there truly is a device at that address. Make sure the command sent and the response are as you expect.
4. Clear All displays. This will test the ability of any and all displays to be cleared. Depending upon the communication configuration and manufacturer's features, this may also clear other displays within the pod. Make sure the command sent and the response are as you expect.
5.4 Testing Task Buttons

Recommended Testing Sequence:
1. Get the status of a single button. It should return Button Not Pressed if the buttons have not been activated. Make sure the command sent and the response are as you expect.

![Image of test task complete button communications interface]
2. Press the button and get the status again. This time it should show Button Pressed. Make sure the command sent and the response are as you expect.

3. Clear the button. Make sure the command sent and the response are as you expect.
4. Repeat test #1 and the button should once again show Button Not Pressed.

5. Test every button individually.

6. Test polling of all buttons. Press every button. Poll All should get the status of each button in order. It will stop and clear the button detected.
7. Repeat Poll All for each button. You will notice that as each button is cleared more buttons will be queried before encountering one that is triggered.

8. When all buttons have been encountered the polling will need to be stopped manually or by pressing a task complete button.
6.0 Transaction Maintenance

Although it should not normally be necessary there is a view Edit feature for examining records.

6.1 View Edit Batch:

Groups of picks, puts, and counts are grouped into batches. New unique batches are created when users select items to process. At that time the header is assigned a batch number and the item may not be processed by any other batch.

**Batch No:** Unique number assigned by the database and is not editable.

**Batch Status:** The status (aka stage) of a batch. Values may be N - New or D – Done. Only batches at done stage may be purged.
6.2 View Edit Detail

As items are imported from the WMS system they are added to the transdetail table. This is a list of every individual item that needs to be processed.

| I | This is a non editable field used only for the java persistence system to be able to retrieve and save entries. |
| Pod No | This is the pod number where this transaction is to be executed. |
| Trans Type | This may be PIC, PUT, or CNT. PIC removes items from the carousel, PUT adds items to the carousel, and CNT audits the quantity on the carousel. |
| Trans Id | A code/number that uniquely identifies a transaction for a single item in the WMS system. Example: OE:345:10:10 may mean sales order document 345 line 10 shipment 10. |
| Priority | Any number ranging from 0 to 32767 that can be used to promote transaction over others. 0 is the lowest priority. Example: Normal picks are set to 50, those for air shipments are 99, and product scrapping 0. |
| Request Qty | The quantity to be picked, put, or counted. |
| Request Date | The calendar date on which this request should take place. |
| Doc Id | A number that identifies a group of transactions, like an order number, that should be kept together. This tends to be a shortened version of the trans_id. Example: OE:345 all items from order 345. |
Brass Ring Carousel Control System

Location: The 10 character encoded location where the transaction should occur. Characters 1-2 = Pod/Zone, 3-4 = Carousel/Row, 5-6 = Bin/Shelving Unit, 7-8 = Shelf, 9-10 = Cell.

Item: The number of/on the product related to this transaction.

Description: The description of the item related to this transaction.

Serialized: Will a serial number need to be recorded. *This is for future use.*

Print Label: Print a label for this transaction? Used for bulk items. *This is for future use.*

Weight Count: Count items using a scale. *This is for future use.*

Status Code: The status of this transaction. This will be filled with an exception code when the quantity requested cannot be filled. Null/blank means no problems.

Create Dtm: Time stamp of when this transaction was actually ingested into the BrassRing database.

Presented Dtm: Time stamp of when this transaction was presented to the operator for picking, put, or count.

Completed Dtm: Time stamp of when the this transaction was completed. This may be used to compare to the presented date time to gauge how long it take a user to pick certain items.

Completed Qty: The actual quantity affected by this transaction. This might be the actual quantity picked from the carousel, the actual quantity put on the carousel, or the actual quantity counted on the carousel.
6.3 View Edit Header

After each import, header records are created for each unique Doc Id in the detail records. The header is equivalent to a single tote, order, or sort bar position.

I: This is a non editable field used only for the java persistence system to be able to retrieve and save entries.

Pod No: This is the pod number where this transaction is to be executed.

Trans Type: This may be PIC, PUT, or CNT. PIC removes items from the carousel, PUT adds items to the carousel, and CNT audits the quantity on the carousel.

Doc Id: A number that identifies a group of transactions, like an order number, that should be kept together. This tends to be a shortened version of the trans_id. Example: OE:345 all items from order 345.

Item Count: The number of items assigned to this Doc Id. This is used to determine the difference between single and multi picks.

Tote Id: A unique number or “license plate” on the containers that bring items to/from the carousel.

User Id: The identity of the person performing this transaction. This can be used along with the time stamps in transdetail to determine operator efficiency.

Batch No: A unique grouping number assigned to every wave of transaction processed on the carousel.
Stage: A code indicating whether the transaction has been completed or not. Stages may be N = New, P = Paused, and D = Done.

Priority: The highest priority of the items for this Doc Id. Because these items travel together they are all raised up to the highest priority in the group.

Sortbar Location: The location on the sort bar for a tote used to contain items for a document. The items are placed in this location during a pick or removed from this location during a put. This is stored to support pausing of batches and recovery from system crashes.
7.0 Database

Apache Derby was selected for this project for the following reasons:

- Runs on any platform that supports Java
- Low memory requirements (approximately 2.6 MB)
- Can run in a standalone client/server mode
- Can be run as embedded
- SQL Standard

The data structures and features were designed for portability. These tables may be created in any standard SQL-99 database. Again this prevents vendor and platform lock-in.
7. 1 Database UML
7.2 Database Schema

CREATE TABLE "APP"."REASON" ("CODE" CHAR(10) NOT NULL, "DESCRIPTION" VARCHAR(60) NOT NULL, "DISPLAY_ORDER" SMALLINT, "POD_NO" SMALLINT NOT NULL DEFAULT 0);

CREATE TABLE "APP"."LIGHTTREE" ("TREE_ID" INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1), "TREE_NO" SMALLINT NOT NULL, "POD_NO" SMALLINT NOT NULL, "ADDRESS" SMALLINT NOT NULL, "COMM_CHANNEL_NO" SMALLINT NOT NULL, "RETRIES" SMALLINT NOT NULL DEFAULT 0, "RETRY_DELAY" SMALLINT NOT NULL DEFAULT 0, "LEFT_CAROUSEL_NO" SMALLINT, "RIGHT_CAROUSEL_NO" SMALLINT, "DISPLAY_DIGITS" SMALLINT NOT NULL DEFAULT 8, "ORIENTATION" CHAR(1) NOT NULL DEFAULT 'V', "FIRST_LEFT_CELL" CHAR(2), "FIRST_RIGHT_CELL" CHAR(2), "MODEL" CHAR(30));

CREATE TABLE "APP"."TRANSDETAIL" ("I" INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1), "POD_NO" SMALLINT NOT NULL DEFAULT 0, "TRANS_TYPE" CHAR(3) NOT NULL, "DOC_ID" VARCHAR(30) NOT NULL, "TRANS_ID" VARCHAR(30) NOT NULL, "LOCATION" CHAR(10) NOT NULL, "REQUEST_QTY" INTEGER NOT NULL DEFAULT 0, "REQUEST_DATE" DATE NOT NULL, "PRIORITY" SMALLINT NOT NULL DEFAULT 0, "ITEM" VARCHAR(100) NOT NULL, "DESCRIPTION" VARCHAR(100), "SERIALIZED" CHAR(1), "PRINTLABEL" CHAR(1), "WEIGHT_COUNT" CHAR(1) DEFAULT 'N', "CREATED_DTM" TIMESTAMP, "PRESENTED_DTM" TIMESTAMP, "COMPLETED_DTM" TIMESTAMP, "COMPLETED_QTY" INTEGER, "STATUS_CODE" CHAR(1), "ROW" SMALLINT, "BIN" SMALLINT, "SHELF" SMALLINT, "CELL" CHAR(2));

CREATE TABLE "APP"."TASKBUTTON" ("TASK_BUTTON_ID" INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1), "TASK_BUTTON_NO" SMALLINT NOT NULL, "POD_NO" SMALLINT NOT NULL, "TREE_NO" SMALLINT NOT NULL, "ADDRESS" SMALLINT NOT NULL, "COMM_CHANNEL_NO" SMALLINT NOT NULL, "RETRIES" SMALLINT NOT NULL DEFAULT 0, "RETRY_DELAY" SMALLINT NOT NULL DEFAULT 0, "MODEL" CHAR(30) DEFAULT '0', "SORTBAR_NO" SMALLINT NOT NULL DEFAULT 0);

CREATE TABLE "APP"."CAROUSEL" ("CAROUSEL_ID" INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1), "CAROUSEL_NO" SMALLINT NOT NULL, "POD_NO" SMALLINT NOT NULL, "ADDRESS" SMALLINT NOT NULL, "BIN_COUNT" SMALLINT NOT NULL, "RETRIES" SMALLINT NOT NULL DEFAULT 0, "RETRY_DELAY" SMALLINT NOT NULL DEFAULT 0, "BIN_COUNT" SMALLINT NOT NULL DEFAULT 0);
NOT NULL DEFAULT 0, "COMM_CHANNEL_NO" SMALLINT NOT NULL, "MODEL" CHAR(30));

CREATE TABLE "APP"."POD" ("POD_NO" SMALLINT NOT NULL, "POD_ID" CHAR(2) NOT NULL, "POD_NAME" VARCHAR(50) NOT NULL, "ITEM_PIC_CONF" CHAR(1) NOT NULL DEFAULT 'N', "ITEM_PUT_CONF" CHAR(1) NOT NULL DEFAULT 'N', "ITEM_CNT_CONF" CHAR(1) NOT NULL DEFAULT 'N', "LOCATION_PIC_CONF" CHAR(1) NOT NULL DEFAULT 'N', "LOCATION_PUT_CONF" CHAR(1) NOT NULL DEFAULT 'N', "LOCATION_CNT_CONF" CHAR(1) NOT NULL DEFAULT 'N', "SP_SORTBAR_NO" SMALLINT, "SP_SORTBAR_DISPLAY" SMALLINT, "EXPORT_COMPLETED_BATCH" CHAR(1) NOT NULL DEFAULT 'Y', "HOME_AFTER_PIC_BATCH" CHAR(1) NOT NULL DEFAULT 'N', "HOME_AFTER_PUT_BATCH" CHAR(1) NOT NULL DEFAULT 'N', "HOME_AFTER_CNT_BATCH" CHAR(1) NOT NULL DEFAULT 'N');

CREATE TABLE "APP"."BATCH" ("BATCH_NO" INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1), "BATCH_STATUS" CHAR(1) NOT NULL DEFAULT 'Y');

CREATE TABLE "APP"."TRANSHEADER" ("I" INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1), "POD_NO" SMALLINT NOT NULL DEFAULT 0, "TRANS_TYPE" CHAR(3) NOT NULL, "DOC_ID" VARCHAR(30) NOT NULL, "ITEM_COUNT" SMALLINT NOT NULL DEFAULT 0, "TOTE_ID" VARCHAR(20), "USER_ID" VARCHAR(20), "BATCH_NO" INTEGER, "STAGE" CHAR(1), "PRIORITY" SMALLINT NOT NULL DEFAULT 0, "SORTBAR_LOCATION" SMALLINT);

CREATE TABLE "APP"."SORTBAR" ("BAR_ID" INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1), "BAR_NO" SMALLINT NOT NULL, "POD_NO" SMALLINT NOT NULL, "ADDRESS" SMALLINT NOT NULL, "COMM_CHANNEL_NO" SMALLINT NOT NULL, "RETRIES" SMALLINT NOT NULL DEFAULT 0, "RETRY_DELAY" SMALLINT, "DISPLAYS" SMALLINT, "LEVELS" SMALLINT, "DISPLAY_DIGITS" SMALLINT NOT NULL DEFAULT 8, "ORIENTATION" CHAR(1) NOT NULL DEFAULT 'V', "MODEL" CHAR(30));

CREATE TABLE "APP"."USERS" ("USER_ID" VARCHAR(20) NOT NULL, "USER_NAME" VARCHAR(50) NOT NULL, "ADMIN" SMALLINT);

CREATE TABLE "APP"."TEST1" ("ANUMERIC" NUMERIC(5,0), "AVARCHARBIT" VARCHAR() FOR BIT DATA, "AVARCHAR" VARCHAR(10), "ATIME" TIME, "ACHARFORBIT" CHAR() FOR BIT DATA, "AREAL" REAL, "ADATE" DATE, "ALONGVCHAR" LONG VARCHAR, "ABIGINT" BIGINT, "AINT" INTEGER,
"ATIMESTAMP" TIMESTAMP, "ADouble" DOUBLE, "ADECIMAL" DECIMAL(12,0), "ABOOL" BOOLEAN);

CREATE TABLE "APP"."COMMCHANNEL" ("CHANNEL_ID" INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1), "CHANNEL_NO" SMALLINT NOT NULL DEFAULT 1, "POD_NO" SMALLINT NOT NULL, "CHANNEL_DESC" VARCHAR(100) NOT NULL, "ADDRESS" VARCHAR(40) NOT NULL, "PORT" INTEGER NOT NULL, "TIMEOUT" INTEGER NOT NULL DEFAULT 0);

-- DDL Statements for indexes
-- ----------------------------------------------
CREATE UNIQUE INDEX "APP"."POD_IDX" ON "APP"."POD" ("POD_ID");
CREATE UNIQUE INDEX "APP"."SBAR_IDX1" ON "APP"."SORTBAR" ("BAR_NO", "POD_NO");
CREATE UNIQUE INDEX "APP"."COM_IDX1" ON "APP"."COMMCHANNEL" ("CHANNEL_NO", "POD_NO");
CREATE UNIQUE INDEX "APP"."TREE_IDX1" ON "APP"."LIGHTTREE" ("TREE_NO", "POD_NO");
CREATE UNIQUE INDEX "APP"."CRSL_IDX1" ON "APP"."CAROUSEL" ("CAROUSEL_NO", "POD_NO");
CREATE UNIQUE INDEX "APP"."TBTN_IDX1" ON "APP"."TASKBUTTON" ("TASK_BUTTON_NO", "POD_NO");

-- DDL Statements for keys
-- ----------------------------------------------
-- primary/unique
ALTER TABLE "APP"."POD" ADD CONSTRAINT "SQL130411201702740" PRIMARY KEY ("POD_NO");
ALTER TABLE "APP"."SORTBAR" ADD CONSTRAINT "SQL130416222247170" PRIMARY KEY ("BAR_ID");
ALTER TABLE "APP"."COMMCHANNEL" ADD CONSTRAINT "SQL130416221444190" PRIMARY KEY ("CHANNEL_ID");
ALTER TABLE "APP"."LIGHTTREE" ADD CONSTRAINT "SQL130416222052090" PRIMARY KEY ("TREE_ID");

ALTER TABLE "APP"."TRANSDETAIL" ADD CONSTRAINT "SQL130411210717010" PRIMARY KEY ("I");

ALTER TABLE "APP"."TRANSHEADER" ADD CONSTRAINT "SQL130411211131950" PRIMARY KEY ("I");

ALTER TABLE "APP"."CAROUSEL" ADD CONSTRAINT "SQL130416221936410" PRIMARY KEY ("CAROUSEL_ID");

ALTER TABLE "APP"."USERS" ADD CONSTRAINT "SQL130412001524550" PRIMARY KEY ("USER_ID");

ALTER TABLE "APP"."TASKBUTTON" ADD CONSTRAINT "SQL130416222505960" PRIMARY KEY ("TASK_BUTTON_ID");

ALTER TABLE "APP"."BATCH" ADD CONSTRAINT "SQL130429234148240" PRIMARY KEY ("BATCH_NO");

ALTER TABLE "APP"."REASON" ADD CONSTRAINT "SQL130425192521470" PRIMARY KEY ("CODE");

ALTER TABLE "APP"."REASON" ADD CONSTRAINT "SQL130512180337300" UNIQUE ("DISPLAY_ORDER");

-- foreign
ALTER TABLE "APP"."TRANSDETAIL" ADD CONSTRAINT "SQL130411210717011" FOREIGN KEY ("POD_NO") REFERENCES "APP"."POD" ("POD_NO") ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE "APP"."TRANSHEADER" ADD CONSTRAINT "SQL130412001916450" FOREIGN KEY ("USER_ID") REFERENCES "APP"."USERS" ("USER_ID") ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE "APP"."TRANSHEADER" ADD CONSTRAINT "SQL130411211131951" FOREIGN KEY ("POD_NO") REFERENCES "APP"."POD" ("POD_NO") ON DELETE NO ACTION ON UPDATE NO ACTION;
8.0 Utilities and Tools
A few special utilities were used during development

8.1 Serial
Serial Port Monitor was used to inspect the command being sent to/from the carousel system on the serial line.

8.2 Serial to TCP/IP
CommTunnel was a free tool used to build a connection between the RS-232/485 port on the computer and our TCP/IP port. This is not intended for production use but provides a data display in hex that is invaluable when trying to debug protocol issues.
http://www.serialporttool.com/CommTunnel.htm

There are a number of free utilities that provide similar functionality. Also, there are industrial hardware products that provide this same functionality and may be more suitable for a production environment.
9.0 Security

Security is of some importance especially since there are humans in close proximity to large industrial machines. The following need to be taken into account during implementation.

9.1 Database Security

- When running the database in a client/server mode it is exposed to the network
- When running in embedded mode only the client may access it
- The database connection is not encrypted
- The passwords used are embedded and NOT secure

9.2 Serial Port Security

- Local serial to TCP/IP software may keep connections private
- The industrial equipment may be exposed to the network when shared via TCP/IP
- Many serial to TCP/IP products do not ask for passwords for access
- Many serial to TCP/IP products do not restrict what addresses may connect

Appendix A – Pictures

Looking at carousels from behind the sort bar. Carousels 1 and 2 are fully visible along with the controllers.
Looking at carousels from behind the sort bar. The controllers and carousel 3 are fully visible. Notice the light tree in the center of carousel 3 that has a task complete button.
Sort bar showing only 6 of its 16 positions. Notice the handy task complete buttons.
Brass Ring Carousel Control System

Close up of a light tree display and task complete button.
Close up of the controllers to which the software communicates.
Overhead view with multiple pods.

Full sort station with motor controller behind.
Rear view of pod from above.
Appendix B - Diamond Communication Protocols
- See Attached