

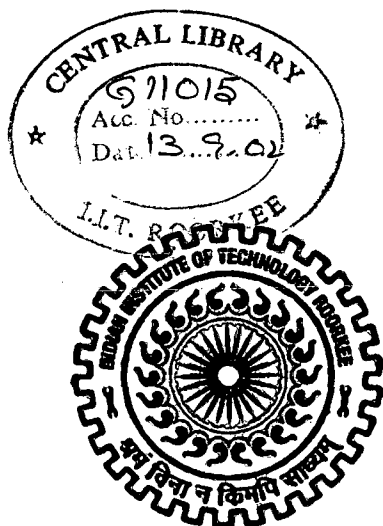
INVENTORY-BARCODE MANAGEMENT SYSTEM

A DISSERTATION

*Submitted in partial fulfilment of the
requirements for the award of the degree
of*
MASTER OF COMPUTER APPLICATIONS

By

SHAKLENDRA SINGH



**DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247 667 (INDIA)**

MAY, 2002

CANDIDATE'S DECLARATION

I hereby declare that the work which is being presented in this project work entitled **Inventory - Barcode Management System** in partial fulfillment of the requirement for the award of the degree of Master of Computer Applications (MCA) in the department of Mathematics, Indian Institute Technology, Roorkee, is an authentic record of my own work carried out by me for a period of five months from January to May 2002 under the supervision and guidance of **Dr. R.C. Mittal**, Professor, Department of Mathematics, **Mr. Sujit Pal Singh**, Project Manager (Cosmic Softech Ltd.), **Mr. Gurmit Singh Ahluwalia**, Project Leader (Cosmic Softech Ltd.).

I have not submitted the matter embodied in this project work for the award of any other degree or diploma.



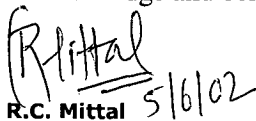
Shaklendra Singh

MCA IIIrd Year

IIT Roorkee

CERTIFICATE

This is to certify that the above statements made by the candidate are correct to the best of our knowledge and belief.



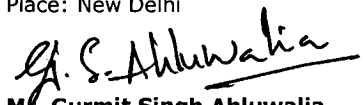
Dr. R.C. Mittal

Professor,
Department of Mathematics,
IIT, Roorkee.
Date:
Place: Roorkee



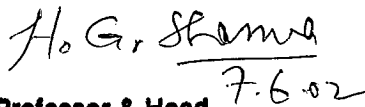
Mr. Sujit Pal Singh

Project Manager
Cosmic Softech Ltd.
New Delhi
Date:
Place: New Delhi



Mr. Gurmit Singh Ahluwalia

Project Leader
Cosmic Softech Ltd.
New Delhi
Date: 4/6/2002
Place: New Delhi



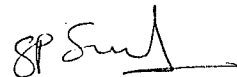
Professor & Head
Department of Mathematics
I.I.T. Roorkee - 247 687

Cosmic InfoTech Solutions
B1/E-11, Mohan Industrial Estate,
Mathura Road,
New Delhi-110044, India
Tel. : +91-11-6941934
Fax : +91-11-6945323
www.citesindia.com

COMPANY CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that the project titled “**Inventory-BarCode Management System**” being submitted by **Mr. Shaktendra Singh**, a student of Indian Institute of Technology, Roorkee in partial fulfilment of the award of the degree of Masters of Computer Applications, has been completed under our supervision at Cosmic Softech Ltd, New Delhi during the period from Jan 2002 to May 2002.



Sujit Pal Singh

Project Manager,
Cosmic Softech Ltd.,
New Delhi

Date: 20th JUNE 2002

ACKNOWLEDGEMENT

It is my proud privilege to express my profound gratitude to my guide **Dr. R.C. Mittal**, Professor, Department of Mathematics, **Mr. Sujit Pal Singh**, Project Manager, Cosmic Softech Ltd., **Mr. Gurmit Singh Ahluwalia**, Project Leader, Cosmic Softech Ltd., for their invaluable inspiration, guidance and continuous encouragement throughout this project work.

I also acknowledge Team Member of the project team for providing the support and inspiration throughout the project work.

I would like to thank all my friends for their help and constructive criticism during the development of this work.



Shaklendra Singh

MCA IIIrd Year
IIT Roorkee

Abstract

The client Richa Creation is a jewellery importer and wholesaler in the United States of America. The company has its manufacturing facilities in New Delhi from where it imports the jewellery. Richa creation decided to make its inventory management, sales analysis and other operations very effective. The assignment for the development of the application has been awarded to Cosmic Softech ltd. *Inventory-Sales Management and Analysis System* provides the user and manager a fast way to handle multi store inventory transfers from their representative at any position to Home (Head Office) and from Home to their representative at any place, manage physical inventory, maintain turnover of sales and provide trend/analysis of the style in demand along with the reasons behind it. The administrator can maintain the database and also import the new inventory details

This report gives development environment and application development of the Inventory and Barcode Management. The Inventory management aims to help the user to enter or import the inventory details. Once the details are in the system the user can control the inventory flow for proper inventory management. This module gives ease to the user to enter the inventory details. User can import the data coming from the vendor/factory on the ASCII file directly into the system. Inventory Management also deals in generating BarCode for the new items coming from Vendors. BarCode are also generated for items whose any information is edited or updated in the database on server. BarCodes allows data to be collected rapidly and with extreme accuracy. There are various symbologies available for generating BarCodes and symbology used for this particular project in Code 128B. The Code 128 character set includes the digits 0-9, the letters A-Z (upper and lower case), and all standard ASCII symbols and control codes. BarCode Printer are special type of printer and they have their own instruction set available for printing BarCodes. Instructions are executed at printer only. BarCodes are printed for Inventory Number of items for easy identification, apart from this information like item's Price, Style, Description and Vendor Name from whom Item has come is also printed on the Label.

Microsoft Visual Basic 6.0 is used at Client Side to give application better look and feel, whereas MSSQL Server 7 is used at Back End for Database Handling. This application has been developed on Intel Platform with 128 MB RAM with Windows 98 and Windows ME as Operating System

Table of Contents

Company Profile	1
1. Introduction	3
1.1 Background	3
1.2 Intended audience and their responsibilities	3
2. General description of Product	4
2.1 Product Perspective	4
2.2 SRS	5
2.3 Scope	5
2.4 Definitions, Acronyms, and Abbreviations	6
2.5 Modular Diagram	7
2.5.1 Inventory Module	7
2.5.2 Sales Module	7
2.5.3 Search Module	8
2.5.4 Reports and Data Analysis Module	8
2.5.5 Orders Module	8
2.5.6 Administrator Module	9
2.6 User Characteristics	9
3 Specific Requirement for Inventory Management	10
3.1 Functional Requirements	10
3.1.1 Introduction	10
3.1.2 Scope	10
3.1.3 DFD	11
3.1.4 Inputs	12
3.1.4.1 New Inventory	12
3.1.4.2 Import of Inventory and Cost Sheet	13
3.1.4.3 Print Tag	14
3.1.5 Outputs	14
3.1.5.1 New Inventory	14
3.1.5.2 Import of Inventory	15
3.1.5.3 Print Tag	15
3.1.6 Processing	15
3.1.6.1 New Inventory	15
3.1.6.2 Import of Inventory	16
3.1.6.3 Print Tag	16
3.1.7 Validations	17
3.1.7.1 New Inventory	17
3.1.7.2 Import of Inventory	17
3.2 Security Considerations	17
3.3 Error Recovery	17
3.4 External Interface /Dependencies on other functions	17

4	BarCode	18
4.1	Barcode Basics	18
4.2	BarCode Symbologies	19
4.3	Code 128	21
4.4	BarCode Reader	26
4.5	BarCode Printing	28
4.5.1	Outline of Command System	28
4.5.2	Outline of Interpreter	29
4.5.3	System Level Immediate Execution Commands	30
4.5.4	Label Format Commands	31
4.6	Printer status	38
4.7	Communication control flowchart	39
5.	Data Base Handling	40
5.1	ADO	40
5.2	The ADO objects	41
5.2.1	Recordset Object Basics	42
5.2.2	Using the Open Method of a Recordset Object	43
5.2.3	Connecting a Recordset Object Variable	44
5.2.4	Specifying Cursor Types	44
5.2.5	Specifying Locking	45
5.3	Data Access Using ActiveX Data Objects (ADO)	46
6.	Standards And Guidelines	47
6.1	Introduction	47
6.2	File Naming Conventions	47
6.3	Coding Standards And Guidelines	48
7.	Software testing	52
7.1	Testing Objectives	52
7.2	Types of Testing	53
7.2.1	Unit Testing	53
7.2.2	Integration Testing	55
7.2.3	Stress Testing	55
7.2.4	Performance Testing	55
8.	Conclusion	56
	Appendix-A ScreenShots for Inventory Management	57
	Appendix- B Defect Report	61
	Appendix- C Screen Shot for Printing BarCode	67
	References	69

Company Profile

Cosmic Softech Ltd.(CITeS) formally Cosmic Infotech Solutions is a part of the Maharishi Group, acclaimed for its philanthropic activities worldwide. The diversified operations of the group are managed from its establishment in more than 26 countries, including India, USA, UK, Germany, Holland, Russia, Japan, Brazil and Australia. CITeS' dizzying range of cutting-edge software solutions include all kind of web-related customised software application, e-commerce, on-site consultation, offshore development, and telecommunications. Recently it entered into a strategic alliance with CIT Solutions of Iowa, USA--the globally acclaimed developer of state-of-the-art telecom systems. Cosmic Softech Ltd.(CITeS) provides e-Business Solutions and Services including web-based customised software applications and offshore development services. CITeS focuses on the Internet domain and services its customers through a **ZDD** delivery model for onsite and offshore delivery. CITeS's delivery methodology makes a highest quality delivery process. Cites help its customers solve business problems by providing effective technology solutions. It is a customer-focused company looking for and offering practical, innovative applications of technology to solve the real-world needs of our clients as they strive to compete in a constantly changing business environment.

CITeS is well positioned to support its global customers, anywhere in the world with minimal lead-time.

- CITeS's competencies exist in the following service areas:
- E-Business & Mobile Computing Solutions
- Application Maintenance
- Application Reengineering/Migration Solutions
- Onsite Consultancy
- Customized Software Development Solutions

CITeS is the Microsoft certified partner, which helps the company in bagging high profile projects and develop on present outlook in a considerable manner. The strategic partnership with Elcom Inc. helped in building a platform for providing solutions for eProcurement and eMarketplace and improving the client's business efficiency. CITeS is also the Off-Shore Development Partner for USA-based Global Online, the world's first global gateway to international life and world trade, providing B2B and B2C e-commerce services through its 5,680 web sites in 18 languages and 29 currencies in over 120 countries worldwide.

In today's dynamic and competitive market, successful business communication is only made possible by developing e-Business Strategy. With its rich experience in deploying e-Business Solutions and Services, CITEs is designed to deliver enterprise client-server/ multi-tier and web-based solutions across the entire value chain, spanning on-site consulting services to turnkey software projects. Expertise in state-of-the-art Internet application tools, database management systems and other software, allow to design and deploy the best solution for your business.

CITEs offers effective solutions for information access, control, delivery, transformation, and sharing to customers anywhere across the globe. In short, CITEs offers competitive and razor-sharp management solutions to its customers. CITEs offers:

- e-Procurement
- e-Marketplace
- e-Sale
- e-Recruiting
- Portal Solutions
- Knowledge Management
- e-Assessment
- Mobile Computing Solutions
- Palm-based Publishing Solution

1. Introduction

1.1 Background

Richa Creation[3][4]jewelry importer and wholesaler in the United States of America. The company has its manufacturing facilities in New Delhi from where it imports the jewelry. Richa Creation has its network of salespersons called REP, who are assigned to take the inventory to the customers in their respective regions. They display the items to the customers, accept orders, and raise the Bill of Sale to the customers. The HOME office in return generates the invoice after receiving the Bill of Sale from the REP.

HOME office has a computerized system of generating the invoices and managing the inventory. Lately Richa Creation realized that they have outgrown this system, which was posing problems for them in expanding their business. Hence, they decided to get a new system developed for it's expanding business.

Richa Creation has decided to make its inventory management, sales analysis and other operations very effective. The company plans to give all their REPs a Palmtop /Laptop with new application loaded on them to simplify their work of generating Bill of Sale from their own system and post all details through wireless to the HOME Server. Thus delays in making invoices and analyzing data can be drastically cut down.

The project will have three phases: firstly, automating the Inventory-Sales Management and Analysis System; secondly, making available the inventory and company information on the Internet; and thirdly, automation of the Manufacturing Unit.

The assignment for the development of the application has been awarded to Cosmic Softech Ltd.. This Project is being developed after detailed study of the existing system and discussions with Richa Creation.

1.2 Intended audience and their responsibilities

- Project Team, Richa Creation.
 - To review the document for validity and completeness
 - To give the sign off to ensure the acceptance of this document
- Project Team, Cosmic Softech Ltd.
 - To take the Software Requirement Specifications as frozen for further Design and Development of project

2. General Description of Product

2.1 Product Perspective

Inventory-Sales Management and Analysis System[5] provides the user and manager a fast way to handle multi store inventory transfers from REPs to NSM, NSM to REP and from NSM to HOME, manage physical inventory, maintain turnover of sales and provide trend/analysis of the style in demand along with the reasons behind it. The administrator can maintain the database and also import the new inventory details

This is the integrated diagram for all the three phases of the project.

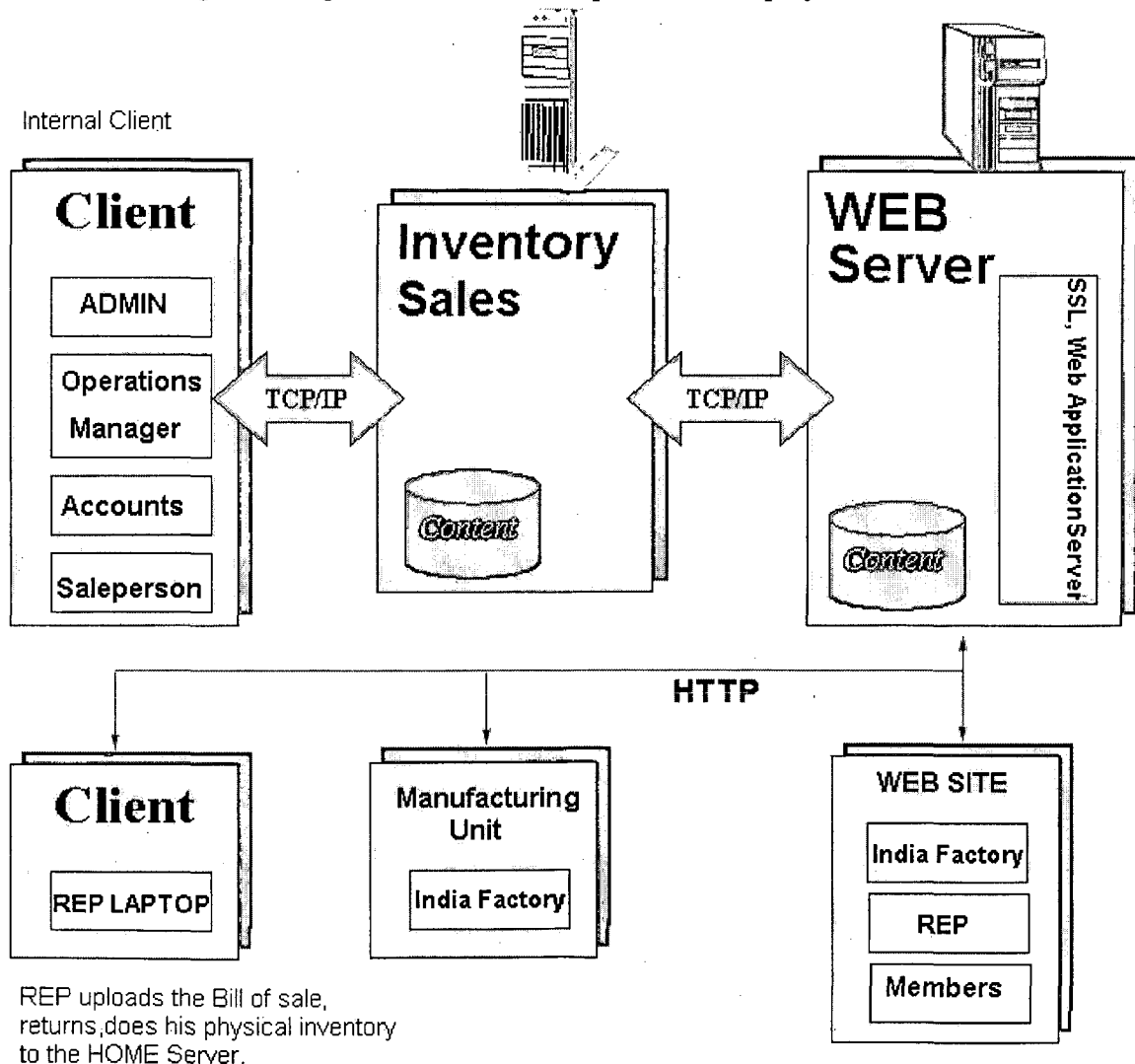


Figure 2.1 Integrated Diagram of all three phases of Inventory Sales Management and Analysis System

2.2 SRS

Requirement Analysis has been done based on the detailed discussion with Richa Creation and collecting the required data after studying their present working structure. A detailed document providing all the findings of Requirement Analysis is titled as “Software Requirement Specifications Document” (SRS Document)[5]

Requirement Analysis for different modules has been conducted to finalize the following aspects:

1. Data Capture and Maintenance Requirements
2. Processing Requirements
3. Online Information Extraction for Decision Making
4. Administration Requirements.

The objective of the Software Requirement Specifications Document is to define the functionalities and scope of the modules. SRS Document identifies all user requirements, which need to be Designed and Developed. Richa Creation’s project team should verify that all the contents are complete and exhaustive. The feedback on the proposed system needs to be known at this stage, as the cost of making any changes later grows exponentially as the system is developed and implemented.

2.3 Scope

The scope of Designing and Development of the “Inventory-Sales Management and Analysis System “ has been envisaged in the following modules:

- *Phase One:*
 - Inventory
 - Sales & Returns
 - Search
 - Orders
 - Reports and Data Analysis
 - Administration
- *Phase Two:*
 - Web Site Hosting
- *Phase Three:*
 - Automation of the Manufacturing Unit

The following areas are not in the present scope of work:

- Day to day administration of the application
- Web site hosting at the web sever
- Web site administration
- Domain name registration
- Data porting to the new system from the old system.
- Data entry / verification of existing data.

2.4 Definitions, Acronyms, and Abbreviations[5]

S. No.	Term/Acronym/Abbr eviation	Meaning
1.	CITeS	Cosmic Softech Ltd.
2.	User	Person responsible for particular module
3.	Administrator	User with power to manage other users and do administrative jobs
4.	RDBMS	Relational Database Management System
5.	Search String	Set of words (customer/product) used by user/administrator for getting more information about the same
6.	REP	Salesperson who is assigned a territory for the sale operations
7.	NSM	National Sales Manager, person to whom a REP reports
8.	Credit Memo	Return invoice generated when a Customer makes a Return.
9.	Bill of Sale	REP and HOME generates the Bill of Sale when any sale is made to a customer this is not a invoice
10.	Return Bill of Sale	REP and HOME generates the Bill of Sale when any Return is made by a customer which is not an invoice
11.	Transfer	When items are transferred from HOME to a REP's line or NSM's line/ or NSM transfers to the REP/ REPs or NSM transfer to HOME
12.	CASE#	A unique number called as the CASE number (CASE#) identifying every REP.
13.	HOME	Head Office of Richa Creation.
14.	SKU	Unique number of each inventory as generated by the system.
15.	DFD	Data Flow Diagram.
16.	ASCII	American Standard Code for Information and Interchange.
17.	Closeout	Ageing inventory or inventory when analyzed by Richa Creation as non-saleable is sold on discounted price.
18.	Major Class	All inventories are categorized in a broad level head called as Major class. The list of major class is defined in the system.

2.5 Modular Diagram

The scope of Designing and Development of the “Inventory-Sales Management and Analysis System “ has been envisaged in the following modules:

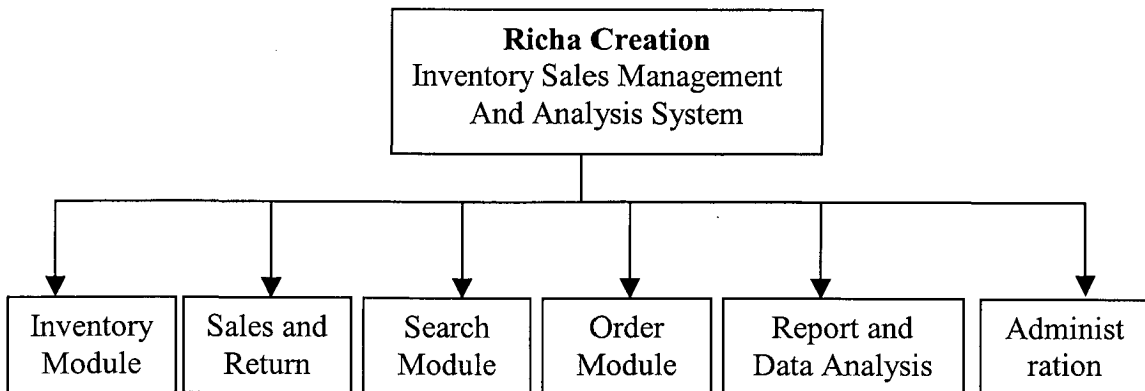


Fig. 2.2 HLD Diagram For Inventory Sales Management And Analysis System

2.5.1 Inventory Module

The Inventory module aims to help the user to enter or import the inventory details. Once the details are in the system the user can control the inventory flow for proper inventory management. The inventory module will give different access level to different users.

This module gives ease to the user to enter the inventory details. User can import the data coming from the vendor/factory on the ASCII file directly into the system. System validates the ASCII data for any errors and generates an error log file. Inventory module has six different sub modules categorized into New Inventory Module, Import Module, Physical Inventory Module, Transfer of inventory Module, Print Tag, and Inventory Room Transfer.

Data coming from the vendors should be in the pre-defined ASCII format.

2.5.2 Sales Module

The sales module captures the details of the sales made to a customer by the REP. When the REP sells a product to the customer he generates a Bill of Sale which is send to the HOME location from his palmtop/laptop and a copy of the same is mailed. The HOME location generates an invoice from the Bill of Sale in the name of the customer and the details of the sale are captured in it. REP has to make separate Bill of Sale for Closeouts, Memorandum and general sales. The HOME server updates the closeout details in REP's system periodically.

The REP uploads the Bill of Sale to the HOME location. This is when the HOME location generates the Sales Invoice. When the Sale is made from the HOME location, the Bill of Sale is also generated.

The RETURN sub-module captures return of Inventory. Customer can only return the items that are present in the Memorandum Bill and that also limited to the percentage specified. Sale Returns can happen in three different ways:

1. Customer directly sends the returned inventory to HOME.
2. REP reconciles the inventory from the Customer.
3. REP collects the inventory from the customer but keeps a part of the return with him as his line.

User can lookup /search for a sales information in the “LOOKUP” sub-module. Here the user has to enter his search criteria and the result will be fetched from the corresponding database.

Returns can be categorized as Memorandum and Manufacturing Defects. Customer can either return the inventory directly to the HOME office or to the REPs. If returned to the REP, he will generate a Return Bill of Sale from his palmtop/Laptop for the returns due to the manufacturing defect (for which he would have to take a Return authorization number from the Home Office), he will generate a Memorandum Return Bill of Sale for the returns against the memorandum Bill of Sale; his system maintains the customer data while he scans the inventory in his system and generates a return Bill of Sale or a Memorandum Return Bill Of Sale. The Return Bill of Sale will have the SKU number of the inventory returned and a note for return.

2.5.3 Search Module

User can perform an easy search on the Inventory-Sales Management and Analysis System by specifying the search criteria in the search module or sub-module. The search module is divided in different sub-module so as to make the search easy.

Search Module facilitates the User to find about the Inventory, Sales, Vendors, Customer, REP Data and Transfer Details. The User enters the search criteria in the selected search screen. Search result is displayed depending on the search criteria.

2.5.4 Reports and Data Analysis Module

Purpose of this module is to generate Reports for analysis and recording purposes. User can select the report to be generated from the sub-menu for a particular module. The behavior of this module will depend on the USER Privilege level i.e. Data analysis can only be done by Administrator and Managers.

Various Modules in the Inventory-Sales Management and Analysis System have a reporting function. User can click on the Report Module to generate formatted reports. Reports will be printed on A4 size paper in landscape or portrait style. Report Module can also be called from other modules.

2.5.5 Orders Module

Purpose of this module is to monitor and place orders to the vendors. Orders can be classified into general orders and Special orders. Inventory-Sales Management and

Analysis System systems analysis model will be able to tell the manager or the administrator about the reorder level of the inventory. The Special orders sub-module will capture the special orders placed by the customer.

Orders Module will have two sub-modules i.e. General Orders and the Special Orders. This module in future will integrate with the system coming up at the India factory level. Manager at any time can view the status of Special Orders from this system, but the status information will primarily be the responsibility of the India factory system.

The Data Analysis module in the Inventory-Sales Management and Analysis System will provide the Orders Module with decision-making information regarding the reorder level of the inventory i.e. if a fast moving inventory reaches its lower limit then the manager can make decision of reordering the style from India factory. Customers with specific requirements for a style can order and user can capture these details in the Special Orders module. The managers can check Special Order status by clicking on the status button.

2.5.6 Administrator Module

Purpose of this module is to do the administration of the Inventory-Sales Management and Analysis System application. User with specific privileges can only do the Administrator function. Administrator is involved in maintaining the Database i.e. taking backups, making tables, running the scripts. Administrator also takes decision on flagging an inventory as Closeout or "must have" for the company.

Administrator will have the right to operate this module. General functions like maintaining the database, taking backups, defining the user rights will all be part of his job and he will be taking decisions on recommendations from manager to make an inventory as a closeout item or declare an item as "must have" for the REP or "never have" for the REP.

The Administrator will generate the reports informing the manager about the profitability of a style and will be making programs and delivery in which fast moving inventories will be added.

2.6 User Characteristics

The user should be familiar with the operations of Richa Creation and must understand the terms given in the document and also have knowledge of operating a computer.

3. Specific Requirement for Inventory Management

3.1 Functional Requirements

3.1.1 Introduction

The Inventory module aims to help the user to enter or import the inventory details. Once the details are in the system the user can control the inventory flow for proper inventory management. The inventory module will give different access level to different users. The Screens involved in Inventory management are given in Appendix –A and details description is given below in section 3.1.3,3.1.4,3.1.5,3.1.6 and 3.1.7, which shows input, output, processing, validation to be done while generating Screens. Figure A.1 shows that when some new Item arrives in Inventory what all details should be entered and saved in the database. Figure A.2 shows how inventory will be imported from a file and if there is any error in the file it returns the rows where errors are found. Figure A.3 shows list of error generated while checking the imported file and these errors can be corrected in this screen and then this inventory details are saved in to the database. Figure A.4 Shows Search Criteria for searching any item in a Inventory. Details of all these screens are given in following sections.

3.1.2 Scope

This module gives ease to the user to enter the inventory details. User can import the data coming from the vendor/factory on the ASCII file directly into the system. System validates the ASCII data for any errors and generates an error log file. Inventory module has six different sub modules categorized into New Inventory Module, Import Module, Physical Inventory Module, Transfer of inventory Module, Print Tag, and Inventory Room Transfer.

Data coming from the vendors should be in the pre-defined ASCII format.

3.1.3 DFD

A high-level data flow diagram for the module: **New Inventory**[5]

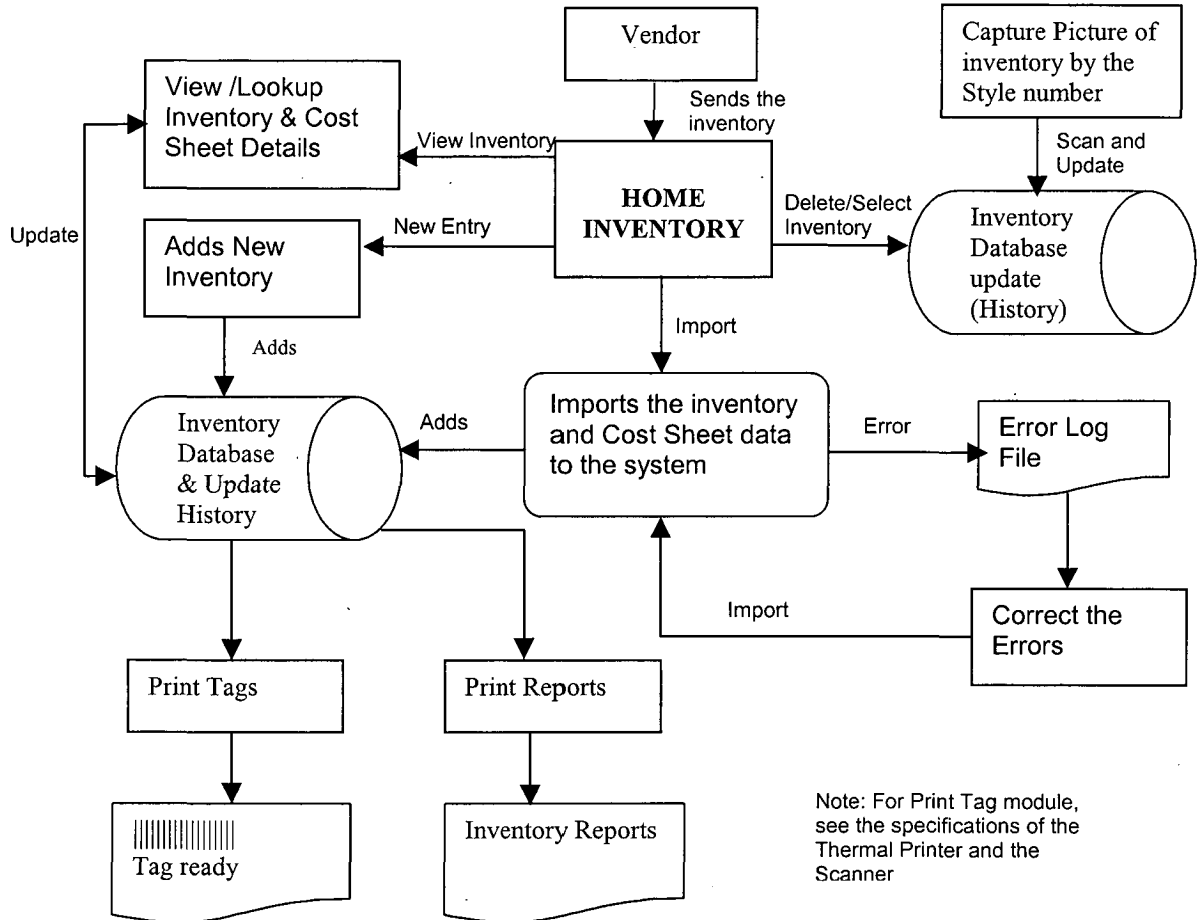


Figure 3.1 DFD of Inventory Module

3.1.4 Inputs

3.1.4.1 New Inventory

1. User Name: Displayed on the screen; depends on the user who logs in the system; used for maintaining the history of the new user added.
2. Date: System date is displayed on the screen.
3. Major Class: Select the item's Major Class in which the item will be categorized from the drop down list.
4. Minor Class: Select the item's Minor Class from the dropdown list. Minor class is also mandatory.
5. Inventory Number: Auto-generate the inventory number as MMM-NNN-XXXXXX where M is Major Class, N is Minor Class and X is the running number.
6. Style Number: Select the style number from the drop down list or just write it in the text box (optional).
7. Primary Material: Select the item's primary material. This can be chosen between "14K Yellow Gold", "18K Yellow Gold", "Sterling Silver", or Other like Platinum and more.
8. Item weight: Enter the item's weight; unit can be select from the drop down list.
9. Style: Select "Ladies", "Gents", "Unisex", or "None" from the style option box.
10. Ring Width: Enter the width of the ring in mm.
11. Quantity: For a new inventory the quantity can only be one.
12. Quantity in stock: Quantity in stock defines the number of pieces for a particular inventory that can be zero or one.
13. Per Quantity Cost: Enter per quantity cost for the item.
14. % Markup Retail: Enter the % markup for calculating the retail price for the item. Explain??
15. % Markup Wholesale: Enter the % markup for calculating the wholesale price for the item
16. Per Quantity List Price Retail: Calculated by the system as per the details of items cost price, %markup.
17. Per Quantity List Price Wholesale: Calculated by the system as per the details of items cost price, %markup.
18. CASE: CASE is the head on which the item is assigned by default. A new inventory item's CASE is HOME.
19. Program ID: Inventory can be put in a defined program by selecting from the drop down list. The Administrator in a program table defines Programs. Programs are made to make the sales of items easy like "12 Best Seller" program has different styles of item in it.
20. Overhead Cost: Display field computed by the system based on the cost for carrying out the transfer and insurance of the inventory, which is calculated by a formula. Every time the inventory is moved the cost accumulated and is shown

21. Closeout Status: Displays field only and the checked status declare this item as a closeout. The Administrator in the Administrator module decides closeouts.
22. Stone Definition: For each style of diamond a stone definition has to be entered like number of diamonds, carat weight, clarity, and minor class.
23. Vendor Details: Enter the vendor details for a new item coming from a new vendor. The user can select the vendor id from the drop down list if the vendor is an existing one.
 - 23.1 Vendor Code: Enter the Vendor's Code, which is alpha numeric.
 - 23.2 Vendor Name: Enter the vendor's name.
 - 23.3 Vendor Invoice #: Enter the vendor's invoice number.
 - 23.4 Vendor SKU: Enter the vendor's SKU number (In case of GDPL this is necessary otherwise this can be ignored). Every item of a vendor has a SKU number
 - 23.5 Style number: Enter the style number.
 - 23.6 Address: Enter the vendor's address in City/State/Zip and Country Fields format.
 - 23.7 Contact Name: Enter the name of the contact person in first/last/title name format.
 - 23.8 Phone number: Enter the Vendor's phone number.
 - 23.9 Fax Number: Enter the facsimile number (optional)
 - 23.10 E-mail: Enter the e-mail address of vendor (optional).
24. Save (Button): Pressing this saves the item in the database.
25. Find/look up (Button): Pressing this button brings up the lookup screen for the Inventory Module. Note: See the Search–Inventory Module for this.
26. Delete (Button): On pressing this button the selected item will be deleted from the inventory database. The process will ask the user for confirmation and the history of the record is updated in the history table.
27. Duplicate Record (Button): While entering similar details for more than one inventory, user can copy the details from the previous item by pressing this button.

3.1.4.2 Import of Inventory and Cost Sheet

This sub-module is used to import new inventory details and the cost sheet details directly into the system. Cost sheet contains the costing details of the entire inventory mentioned in the inventory sheet sent along with it. The data to be imported in the system is sent by the vendor in ASCII file along with his invoice number and Vendors item SKU number for all the records/items for the inventory sheet and cost sheet.

- 1 User Name: Displayed on the screen, user name depends on the user who logs in the system.
2. Date: System date is displayed on the screen.
3. Import inventory: Select this check box to import the inventory details.

4. Import Cost Sheet: Select this check box to import the cost sheet details for the inventory
5. Import: Pressing this button brings up the import screen.
6. File: Enter the name and location of file to be imported.
7. Browse: Takes to the directory where the file is located.
8. OK: Starts the import.
9. Cancel: Interrupts the process in between when pressed and no record is imported.

3.1.4.3 Print Tag

This sub-module to print the Barcode Tags for the inventory

1. User Name: Displayed on the screen, depends on the user who logs in the system.
2. New Inventory: Prints all the tags for new inventory.
3. Select Inventory: Select items from the drop down list for which the tags are to be generated.
4. Empty list: Empty the list of inventory from the print tag list.
5. Adds Row: Adds an inventory to the print tag list.
6. Delete Row: Deletes the selected row from the print tag list.
7. Print All: Prints the entire item in the print tag list.
8. Print Selected: Prints the selected item to the thermal printer.
9. Edit Row: Select inventory details can be edited. Reason for modification is mentioned.
10. Cancel: Cancels the process.
11. Export Tag List: Menu option used to export the tag list to a text file.

3.1.5 Outputs

3.1.5.1 New Inventory

1. Vendor Details: Enter all vendor details like vendor id, address, and vendor invoice number and the system-generated messages if they are not entered.
2. Invalid CASE: Item should be assigned to a CASE.
3. Error per Qty Cost: Message for an item which does not have a per quantity cost.
4. Error % markup: Item should have the retail and wholesale markups.
5. Save: Saves the inventory in the database.
6. Do you want to delete: This message asks the user for confirmation before deletion of any record.
7. Lookup process generates the result of the lookup based on the lookup criteria.
8. Delete process updates the Inventory and History Database.

9. Add new process adds new items in the Inventory and History Database.
10. Update process updates the Inventory and History Database.
11. Print Tag module is called when the “print tag “ button / menu is clicked.
12. Capture Picture Module is called when the “Capture Picture” button /menu is clicked.
13. Import Module is called by clicking the “Import” button / menu.

3.1.5.2 Import of Inventory

1. The inventory details will be updated in the system after the successful completion of the process.
2. Cost sheet data will be updated in the cost sheet table. The identifying key to a record is the factory SKU number.
3. Warning Message: Displayed before starting the import process so as to warn the user about the impact of the process on the application / database.
4. Error message and Error location of the incorrect records will be reported by the system.

3.1.5.3 Print Tags.

1. Prints the Thermal barcode tags.
2. Generates a file containing the Tag details of Export tag.
3. Adds new records, deletes a record, and edits a record to update the print tag list.

3.1.6 Processing

After starting the application User will enter his ‘User Id’ and ‘password’. If User Id and password are incorrect then Error Message will be displayed. Depending on his privileges user will be able to perform on the enabled modules/functions in the application.

3.1.6.1 New Inventory

1. User with the privilege to enter a new inventory will be able to enter the inventory details. All Inventory items are categorized by the way of defining a major class and minor class (optional) for it.
2. Once the user has entered the major class and the minor class, the inventory number will be generated as MMMM-NNNN-XXXXX where M is major class, N is Minor class and X is the running number.
3. The inventory will be associated to a style number, if the user wants. Style number can be selected from a drop down list. Note: A style number B10114, which is associated with major class Bracelets and minor class Bagguttes D can have many inventory associated with it.
4. A new Inventory by default will have the CASE as “HOME”
5. By default the quantity in the inventory can only be one.

6. After entering the broad level details of the inventory, user can press the “ADD” button to move to a new screen, which captures the micro level details of the inventory.
7. Pressing the “Item components” button opens a new screen where the details of Stone Definition in the inventory can be entered. By clicking the “OK” button the screen will capture the details.
8. User enters the micro level details in this screen. User will be prompted to enter the compulsory details like item weight, vendors invoice number, cost price, % markup for retails and wholesale, etc.
9. Clicking on the “Save” button will save the new inventory details to the database.
10. Clicking on the “Delete“ button will delete the selected inventory from the database for which the history will be maintained in the history database.
11. User can view the details of an inventory by entering the lookup criteria, which will call the search–lookup module. User with the privilege of updating the record will be able to do so in lookup module.

3.1.6.2 Import of Inventory

1. User with the privilege to import the inventory will be able to perform this process.
2. User will have to select the option of “importing inventory “ or “importing Cost Sheet” Details.
3. The data to be imported has to be present in ASCII format file with specific delimiters between two fields of a record. Note: Richa Creation will give the ASCII file format and position of the fields in a record.
4. Browsing the file location and then clicking on it can specify the location of the file. Once the file is selected, the process can be initiated by pressing the import button.
5. Import process can be cancelled in between by pressing the “CANCEL” button.
6. During the whole process if there are any bad records or errors, the system will generate a log file for this.
7. Error report can be generated to find the error location or the cause of error.

3.1.6.3 Print Tags.

1. User with the privilege to Print tags for the inventory will be able to perform this process.
2. User can select the inventory numbers or can add new inventory number for which he needs to print the tag.
3. The inventory information to be printed on the tags can be modified through this module.
4. Print Button when pressed will print the tags to the thermal printer.
5. Pause Button when pressed will pause the printing.
6. Cancel button will cancel the printing job.

3.1.7 Validations

1. For users, the user name and password are validated with the user database and corresponding menu options shall be displayed to the user.
2. Correct data type shall be validated at the client side before the data is posed to the database.
3. Compulsory fields cannot be left blank; error messages will be generated if done so.

3.1.7.1 New inventory

1. Quantity of inventory can only be one or zero, as the case might be.
2. For every new inventory, there will be a CASE associated with it. It is HOME location by default.
3. System will generate error if wrong Style number is associated with any inventory.
4. Major class is required; system will generate error if no major class is entered for a new inventory.
5. Privileged users will update cost details of the item only; system will generate messages for invalid users.

3.1.7.2 Import inventory

1. System performs the data type check on all the fields of a record in the import file.

3.2 Security Considerations

1. Password will be encrypted.
2. User Menu will be assigned to each User according to the privileges assigned by the administrator.

3.3 Error Recovery

1. Data type mismatch: The User shall be made alert about the mismatch type and he can go back to the field and reenter the information in the field. Data entered in the fields will be retained. For the incorrect entry, error message will be displayed and on the same screen data capture fields will be available with the already entered correct entries. The incorrect entry fields will be blank.

3.4 External Interface /Dependencies on other functions

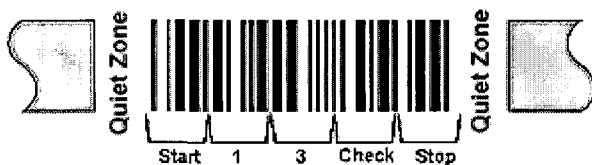
1. In case of any errors while importing the ASCII file, system will generate a log file. User can find the cause of error by viewing this file.

4 BarCode

4.1 Barcode Basics

Bar codes[6][7] provide a simple and inexpensive method of encoding text information that is easily read by inexpensive electronic readers. Bar coding also allows data to be collected rapidly and with extreme accuracy. A bar code consists of a series of parallel, adjacent bars and spaces. Predefined bar and space patterns or "symbologies" are used to encode small strings of character data into a printed symbol. Bar codes can be thought of as a printed type of the Morse code with narrow bars (and spaces) representing dots, and wide bars representing dashes. A bar code reader decodes a bar code by scanning a light source across the bar code and measuring the intensity of light reflected back by the white spaces. The pattern of reflected light is detected with a photodiode which produces an electronic signal that exactly matches the printed bar code pattern. This signal is then decoded back to the original data by inexpensive electronic circuits. Due to the design of most bar code symbologies it does not make any difference if you scan a bar code from right to left or from left to right.

The basic structure of a bar code consists of a leading and trailing quiet zone, a start pattern, one or more data characters, optionally one or two check characters and a stop pattern.



There are a variety of different types of bar code encoding schemes or "symbologies", each of which were originally developed to fulfill a specific need in a specific industry. Several of these symbologies have matured into de-facto standards that are used universally today throughout most industries.

The different symbologies[8] have different capabilities for encoding data. For example the UPC symbology used to identify retail products always contains 12 numeric digits whereas the general purpose Code 39 or Code 128 bar code symbologies can encode variable length alphanumeric data up to about 30 characters in length. These types of bar codes are called "linear symbologies" because they are made up of a series of lines of different widths. Most commercially available bar code scanners are able to read all of the different linear bar code symbologies therefore you do not need different readers for different types of bar codes.

New "2-Dimensional" bar code symbologies like PDF417, Aztec Code and Data Matrix are also now available that can encode several thousand bytes of data in a single bar code symbol including text or binary data. The newer 2D bar code symbologies typically require special bar code readers that are designed specifically for reading them.

The primary purpose of a bar code is to identify something by labeling the item with a bar code containing a unique number or character string. Bar codes are typically used with a database application where the data encoded in the bar codes is used as an index to a record in the database that contains more detailed information about the item that is being scanned. For example, when a checkout clerk scans a bar code on a product in a grocery store, the bar code data is fed to a computer that looks up the information in a central database and returns more detailed information about the item that was scanned including possibly a description of the item and a price. By using bar codes, the grocery store does not need to put a price tag on each item in the store and they can also change the price for a particular item by modifying a single entry in the central database. They can also track how much of a product is currently in stock so that they know when to re-order more of each item as the number of items in stock falls.

Bar codes also provide a quick and error free means for inputting the data into an application running on a computer. By using bar codes, the potential for errors from manual data input is eliminated. Another typical application for bar codes is therefore for inputting data without having to type. For example you could encode name or address data in a bar code on an ID badge and then scan the ID badges to input a persons name into a computer program instead of typing the information.

The different bar code symbologies support different types and amounts of data therefore you normally choose a particular symbology based on the type and amount of data that you want to encode in your bar codes. You are generally free to use any type of bar code that you like and encode whatever data that you like for applications in a closed system.

4.2 BarCode Symbologies[8]

Bar codes are like a printed version of the Morse code. Different bar and space patterns are used to represent different characters. Sets of these patterns are grouped together to form a "symbology". There are many types of bar code symbologies each having their own special characteristics and features. Most symbologies were designed to meet the needs of a specific application or industry. For example the UPC symbology was designed for identifying retail and grocery items and PostNET was designed to encode Zip Codes for the US Postal Service.

A Barcode Symbology defines the technical details of a particular type of barcode: the width of the bars, character set, method of encoding, checksum specifications, etc. Most users are more interested in the general capabilities of a particular symbology (how much and what kind of data can it hold, what are its common uses, etc) than in the excruciating technical details.

Numeric-only barcodes

- **EAN-13** (European Article Numbering international retail product code)
- **EAN-8** (compressed version of EAN code for use on small products)
- **UPC-A** (Universal Product Code seen on almost all retail products in the USA and Canada)
- **UPC-E** (compressed version of UPC code for use on small products)
- **Code 11** (used primarily for labeling telecommunications equipment)
- **Interleaved 2 of 5** (compact numeric code, widely used in industry, air cargo, other applications)
- **Industrial 2 of 5** (older code not in common use)
- **Standard 2 of 5** (older code not in common use)
- **Codabar** (older code often used in library systems, sometimes in blood banks)
- **Plessey** (older code commonly used for retail shelf marking)
- **MSI** (variation of the Plessey code commonly used in USA)
- **PostNet** (used by U.S. Postal Service for automated mail sorting)

Alphanumeric barcodes

- **Code 39** (general-purpose code in very wide use world-wide)
- **Code 93** (compact code similar to Code 39)
- **Code 128** (very capable code, excellent density, high reliability; in very wide use world-wide)
- **LOGMARS** (same as Code 39, this is the U.S. Government specification)

2-Dimensional barcodes

- **PDF417** (excellent for encoding large amounts of data)
- **DataMatrix** (can hold large amounts of data, especially suited for making very small codes)
- **Maxicode** (fixed length, used by United Parcel Service for automated package sorting)
- **QR Code** (used for material control and order confirmation)
- **Data Code**
- **Code 49**
- **16K**

Industry Standards for Barcodes and Labels

- **Bookland EAN encodes ISBN numbers**, (used internationally to mark books)
- **ISSN and the SISAC Barcode** (International Standard Serial Numbering)
- **OPC Optical Industry Association** barcode for marking retail optical products
- **UPC Shipping Container Symbol (ITF-14)**
- **Co-Operative labels** (located under software)

Accuracy of Different Symbologies - How Accurate is Accurate?

It's commonly known that the best-trained data entry operator will make a keystroke entry error once every 300 keystrokes. Each of these keystroke errors represents an error in your decision-making data. This leads to wasted time, misappropriated capital and ultimately, lost revenues. It is for this reason that most companies make the decision to adopt AIDC technologies.

In studies conducted by the University of Ohio, common bar code symbologies were tested to determine real life accuracy. The worst bar code for data accuracy in the test proved to be one of the most common - the UPC. The UPC had a worst-case error rate of 1 error in 394K characters. The best-tested symbologies were the DataMatrix and PDF417, with a worst case error rate of 1 error in 10.5M characters. All the results from the University of Ohio study are listed below.

Symbology	Worst Case	Best Case
<i>DataMatrix</i>	1 error in 10.5M	1 error in 612.9M
<i>PDF417</i>	1 error in 10.5M	1 error in 612.4M
<i>Code 128</i>	1 error in 2.8M	1 error in 37M
<i>Code 39</i>	1 error in 1.7M	1 error in 4.5M
<i>UPC</i>	1 error in 394K	1 error in 800K

4.3 CODE 128

The following is a detailed description of the most commonly used bar code symbology Code 128 which is used for generation Barcode for items of inventory of Richa Creation.



Code 128[7][8] is a variable length, high density, alphanumeric symbology. Code 128 has 106 different bar and space patterns and each pattern can have one of three different meanings, depending on which of three different character sets is employed. Special start characters tell the reader which of the character sets is initially being used and three special shift codes permit changing character sets inside a symbol. One character sets encodes all upper case and ASCII control characters, another encodes all upper and lower case characters and the third set encodes numeric digit pairs 00 through 99. This third character set effectively doubles the code density when printing numeric data. Code 128 also employs a check digit for data security. In addition to ASCII characters, Code 128 also allows encoding of four special function codes

(FNC1 - FNC4). The meaning of function code FNC1 and FNC4 were originally left open for application specific purposes. Recently an agreement was made by the Automatic Identification Manufacturers Assoc. (AIM) and the European Article Numbering Assoc. (EAN) to reserve FNC1 for use in EAN applications. FNC4 remains available for use in closed system applications. FNC2 is used to instruct a bar code reader to concatenate the message in a bar code symbol with the message in the next symbol. FNC3 is used to instruct a bar code reader to perform a reset. When FNC3 is encoded anywhere in a symbol, any data also contained in the symbol is discarded.

The Code 128 character set includes the digits 0-9, the letters A-Z (upper and lower case), and all standard ASCII symbols and control codes. The codes are divided into three subsets A, B, and C. There are three separate start codes to indicate which subset will be used; in addition, each subset includes control characters to switch to another subset in the middle of a barcode. Subset A includes the standard ASCII symbols, digits, upper case letters, and control codes. Subset B includes standard ASCII symbols, digits, upper and lower case letters. Subset C compresses two numeric digits into each character, providing excellent density. Here is a sample that contains 12 digits; compare its size to the sample at the top of the page that contains 12 assorted characters:



Each character is 11 times the width of the narrowest bar; using a minimum bar width of 0.010" each character would be 0.11" wide. Using the 0.010" figure, 20 data characters plus start code, check digit, and stop code would measure 2.55" wide (the stop code is 13 times as wide as a narrow bar). Using Subset C with all-numeric data provides 2:1 compression of the data for a total width of 1.45".

Each character consists of 3 bars and 3 spaces, each of which may be 1, 2, or 3 elements wide (1 element = 1/11th of the character width). The bars always use an even number of elements and the spaces use an odd number. This provides the basis for a character-by-character consistency check during scanning. In addition, each Code 128 barcode includes a Modulo 103 checksum.

Three different start characters are used in the Code 128 bar codes to tell the barcode reader which character set is being used. The table below illustrates the three different character sets and the ASCII location to print the character. If you are using character sets A or B you can create the Code 128 barcode output simply by selecting the Code 128 font and typing the appropriate letter from the keyboard with the exception of the space character barcode, extended functions and the start / stop characters. It is possible to copy and paste these extended characters from the Character Map application.

Code A	Code B	Code C	ASCII	Unicode	Value	Code A	Code B	Code C	ASCII	Unicode	Value
Space	Space	00	0194	00C2	00	V	V	54	0086	0056	54
!	!	01	0033	0021	01	W	W	55	0087	0057	55
"	"	02	0034	0022	02	X	X	56	0088	0058	56
#	#	03	0035	0023	03	Y	Y	57	0089	0059	57
\$	\$	04	0036	0024	04	Z	Z	58	0090	005A	58
%	%	05	0037	0025	05	[[59	0091	005B	59
&	&	06	0038	0026	06	\	\	60	0092	005C	60
'	'	07	0039	0027	07]]	61	0093	005D	61
((08	0040	0028	08	^	^	62	0094	005E	62
))	09	0041	0029	09	_	_	63	0095	005F	63
*	*	10	0042	002A	10	nul	`	64	0096	0060	64
+	+	11	0043	002B	11	soh	a	65	0097	0061	65
,	,	12	0044	002C	12	stx	b	66	0098	0062	66
-	-	13	0045	002D	13	etx	c	67	0099	0063	67
.	.	14	0046	002E	14	eot	d	68	0100	0064	68
/	/	15	0047	002F	15	eno	e	69	0101	0065	69
0	0	16	0048	0030	16	ack	f	70	0102	0066	70
1	1	17	0049	0031	17	bel	g	71	0103	0067	71
2	2	18	0050	0032	18	bs	h	72	0104	0068	72
3	3	19	0051	0033	19	ht	i	73	0105	0069	73
4	4	20	0052	0034	20	lf	j	74	0106	006A	74
5	5	21	0053	0035	21	vt	k	75	0107	006B	75
6	6	22	0054	0036	22	ff	l	76	0108	006C	76
7	7	23	0055	0037	23	cr	m	77	0109	006D	77
8	8	24	0056	0038	24	s0	n	78	0110	006E	78
9	9	25	0057	0039	25	s1	o	79	0111	006F	79
:	:	26	0058	003A	26	dle	p	80	0112	0070	80
;	;	27	0059	003B	27	dc1	q	81	0113	0071	81
<	<	28	0060	003C	28	dc2	r	82	0114	0072	82
=	=	29	0061	003D	29	dc3	s	83	0115	0073	83
>	>	30	0062	003E	30	dc4	t	84	0116	0074	84
?	?	31	0063	003F	31	nak	u	85	0117	0075	85
@	@	32	0064	0040	32	syn	v	86	0118	0076	86
A	A	33	0065	0041	33	etb	w	87	0119	0077	87
B	B	34	0066	0042	34	can	x	88	0120	0078	88
C	C	35	0067	0043	35	em	y	89	0121	0079	89

D	D	36	0068	0044	36	sub	z	90	0122	007A	90
E	E	37	0069	0045	37	esc	{	91	0123	007B	91
F	F	38	0070	0046	38	fs		92	0124	007C	92
G	G	39	0071	0047	39	gs	}	93	0125	007D	93
H	H	40	0072	0048	40	rs	~	94	0126	007E	94
I	I	41	0073	0049	41	us	del	95	0195	00C3	95
J	J	42	0074	004A	42	fnc 3	fnc 3	96	0196	00C4	96
K	K	43	0075	004B	43	fnc 2	fnc2	97	0197	00C5	97
L	L	44	0076	004C	44	Shift	Shift	98	0198	00C6	98
M	M	45	0077	004D	45	code C	code C	99	0199	00C7	99
N	N	46	0078	004E	46	code B	fnc 4	code B	0200	00C8	100
O	O	47	0079	004F	47	fnc 4	code A	code A	0201	00C9	101
P	P	48	0080	0050	48	fnc 1	fnc 1	fnc 1	0202	00CA	102
Q	Q	49	0081	0051	49	Start A	Start A	Start A	0203	00CB	103
R	R	50	0082	0052	50	Start B	Start B	Start B	0204	00CC	104
S	S	51	0083	0053	51	Start C	Start C	Start C	0205	00CD	105
T	T	52	0084	0054	52	Stop	Stop	Stop	0206	00CE	
U	U	53	0085	0055	53						

* It is necessary to print the Code 128 space character from ASCII 194 instead of ASCII 32 because Windows cannot print a symbol instead of a space character from Visual Basic and most other development environments.

The Checksum[8]

Calculating the checksum can be a little tricky. Each Code 128 character has a numeric value from 0 to 102. In Subset A and B, the numeric value of a character is its ASCII code minus 32. For instance, a space (ASCII 32) has a value of 0, the exclamation point (ASCII 33) has a value of 1, etc.

Subset A permits printing of ASCII control characters, those with ASCII codes between 0 and 31. For these characters add 64 to the ASCII code to obtain the numeric value. For example, the value of NUL (ASCII 0) is 64, SOH (ASCII 1) is 65, STX (ASCII 2) is 66, etc.

Subset C prints numeric digits in pairs, and the value of the character for checksum purposes is the numeric value of the pair (00, 01, 02... 99).

To calculate the checksum, follow these steps:

1. Reference the table above to obtain the value of the start character and all data characters.
2. Assign a weighting to each data character (not the start character just the data characters.) The weighting starts at 1 and increases by one for each data character.
3. Multiply the character values by their weights for the data characters.
4. Add these all together including the start character, divide by 103 and obtain the remainder.
5. Use the table above to locate the character that has the value of the remainder, use this as the check character.

Calculating Character Set A or B:

The following table is an example of how to obtain the check character for the data "biz" using Code 128 character set B.

	Start B	b	I	z	STOP
weighting		1	2	3	
Values	104	66	73	90	
Totals	104	66	146	270	

1. Calculate Total: $104 + (66*1) + (73*2) + (90*3) = 586$
2. Calculate Checksum: $586 \text{ divided by } 103 = 5 \text{ remainder of } 71$. Check digit = value of 71. The character to print for the value of 71 is "g" or ASCII 103.

To print extended ASCII characters from your keyboard that do not have keys defined, you can use the **ALT+<xxxx>** key combination where xxxx equals the character's ASCII location in 4 digits. For example, to type ASCII character 104 into Microsoft Word, Select the font in Word, Press the ALT key and while holding it down type **0104** on the keypad of your keyboard. You must use the keypad to do this and you must enter 4 digits.

Therefore to print the data "biz" as a barcode, you would need to type: ALT 0204, b, i, z, g, ALT 0206 or **ïbizĝ**

Calculating Character Set C:

The following table is an example of how to obtain the check character for the number "667390" using Code 128 character set C. First we have to interleave the numbers into pairs and then choose the appropriate character that represents the number pair.

	Start C	66	73	90	STOP
weighting		1	2	3	
values	105	66	73	90	
totals	105	66	146	270	

1. Calculate Total: $105 + (66*1) + (73*2) + (90*3) = 587$
2. Calculate Checksum: 587 divided by $103 = 5$ remainder of 72 . Check digit = value of 72 . The character to print for the value of 72 is ASCII 104 .

Therefore to print the data "667390" as a barcode, you would need to type: ALT 0205, b, i, z, h, ALT 0206 or **ÍbizhÍ** .

The details of converting the numeric checksum to an ASCII character may vary depending on the particular font being used. Special handling may be required in cases where the resulting ASCII character code equals zero or is above 95.

Generally speaking, in Subset A if the checksum is between 0 and 63 inclusive, add the checksum and the ASCII code for a space (32) to obtain the character code. If the checksum is 64 or higher, subtract 64 to obtain the character code. In Subset B, add the checksum and the ASCII code for a space (32). In Subset C, the checksum is the ASCII value of the character.

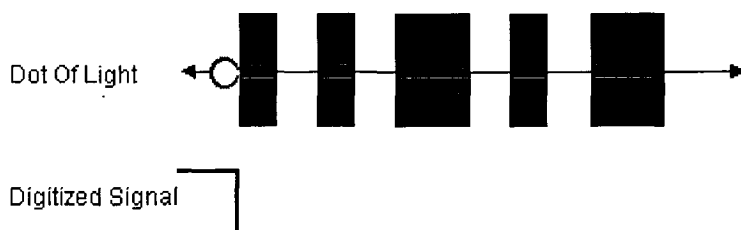
4.4 BarCode Reader

How a BarCode Reader Works[6]

There are currently four different types of bar code readers available. Each uses a slightly different technology for reading and decoding a bar code. There are pen type readers (i.e. bar code wands), laser scanners, CCD readers and camera based readers.

Pen Types Readers and Laser Scanners

Pen type readers consist of a light source and a photo diode that are placed next to each other in the tip of a pen or wand. To read a bar code, you drag the tip of the pen across all the bars in a steady even motion. The photo diode measures the intensity of the light reflected back from the light source and generates a waveform that is used to measure the widths of the bars and spaces in the bar code. Dark bars in the bar code absorb light and white spaces reflect light so that the voltage waveform generated by the photo diode is an exact duplicate of the bar and space pattern in the bar code. This waveform is decoded by the scanner in a manner similar to the way Morse code dots and dashes are decoded.



Laser scanners work the same way as pen type readers except that they use a laser beam as the light source and typically employ either a reciprocating mirror or a rotating prism to scan the laser beam back and forth across the bar code. Just the same as with the pen type reader, a photo diode is used to measure the intensity of the light reflected back from the bar code. In both pen readers and laser scanners, the light emitted by the reader is tuned to a specific frequency and the photo diode is designed to detect only this same frequency light.

Pen type readers and laser scanners can be purchased with different resolutions to enable them to read bar codes of different sizes. The scanner resolution is measured by the size of the dot of light emitted by the reader. The dot of light should be equal to or slightly smaller than the narrowest element width ("X" dimension). If the dot is wider than the width of the narrowest bar or space, then the dot will overlap two or more bars at a time thereby causing the scanner to not be able to distinguish clear transitions between bars and spaces. If the dot is too small, then any spots or voids in the bars can be misinterpreted as light areas also making a bar code unreadable. The most commonly used X dimension is 13 mils (roughly 4 printer dots on a 300 DPI printer). Because this X dimension is so small, it is extremely important that the bar code is created with a program that creates high resolution graphics (like B-Coder).

CCD Readers

CCD (Charge Coupled Device) readers use an array of hundreds of tiny light sensors lined up in a row in the head of the reader. Each sensor can be thought of as a single photo diode that measures the intensity of the light immediately in front of it. Each individual light sensor in the CCD reader is extremely small and because there are hundreds of sensors lined up in a row, a voltage pattern identical to the pattern in a bar code is generated in the reader by sequentially measuring the voltages across each sensor in the row. The important difference between a CCD reader and a pen or laser scanner is that the CCD reader is measuring emitted ambient light from the bar code whereas pen or laser scanners are measuring reflected light of a specific frequency originating from the scanner itself.

Camera Based Readers

The fourth and newest type of bar code reader currently available are camera based readers that use a small video camera to capture an image of a bar code. The reader then uses sophisticated digital image processing techniques to decode the bar code. Video cameras use the same CCD technology as in a CCD bar code reader except that instead of having a single row of sensors, a video camera has hundreds of rows of sensors arranged in a two dimensional array so that they can generate an image.

The factors that make a bar code readable are: an adequate print contrast between the light and dark bars and having all bar and space dimensions within the tolerances for the symbology. It is also helpful to have sharp bar edges, few or no spots or voids, a smooth surface and clear margins or "quiet zones" at either end of the printed symbol.

4.5 BarCode Printing

There are various printers available which can print Barcode. These printers are of special type, because Barcode cannot be printed with any ordinary printer. These are thermal printer made for printing Barcode. Every Barcode Printer has its own Instruction sets or language which it can understand. There are Zebra Printers which are used to print Barcode and they use ZPL(Zebra Programming Language) for developing and printing Barcode

The printer, which is used for Richa Creation project, is C.ITOH.This printer incorporate many functions such as a variety of fonts, bar code generators, and graphic commands along with high speed processing, so high-quality labels can be printed easily at high speeds when simple commands are transmitted from the host computer. The computer processing in generating labels is reduced enabling it to undertake more processing.

The Figure C.1 in Appendix-C shows various details of Items in Inventory, which is to be collected and printed on Labels, and depending upon type of printing whether Batch Printing or Single Item printing is done. Some of the Barcode generated are shown Figure C.2 which shows in what manner various details which is to be printed in Barcode are printed. There are various options in which details can put at different places or in different orientation on the label.

4.5.1 Outline of Command System[2]

Commands for this printer consist of a string of ASCII characters and end with a "CR" (decimal: 13, hex: 0D). Generally, commands are classified into two types, that is, system level commands and label format commands.

System level commands are used in system level operations, including printer output, sensor selection and memory card maintenance. Label format commands are used in the definition of printing contents such as character data, bar code data, printing speed, and print density.

System level commands start with ASCII "SOH" (\$01) or ASCII "STX" (\$02). Commands which start with "SOH" are requested for the realtime execution. When received, they are executed immediately even during printing. Commands which start with "STX" enter the buffer area and are executed in the order of data reception.

Label format commands follow the system level commands' "STX" + "L" and end with a "CR."

Command summary

System level commands	Commands which start with "SOH" Executed as soon as they are received (For example: printing halt, output of printer status, etc.)
Start with "SOH" or "STX"	Commands which start with "STX" Executed in order after they are received into the reception buffer (For example: sensor switching, memory card maintenance, etc.)

"STX" + "L"

"E" (with printing)
"X" (without printing)

Label format commands Print parameter control End with "CR"	Character data definition commands Bar code definition commands Graphic commands Other commands
---	--

4.5.2 Outline of Interpreter [2]

Two types of interpreters are used for this printer; system level and label format interpreters. When power is turned on, the system level interpreter is selected and the data received is processed in the system level interpreter and system level commands are executed.

Changing to the label format interpreter to start generating label data is executed with system level commands. When the system level interpreter receives the system level command "STX" + "L," it changes to the label format interpreter. The commands after this are regarded as label format commands and label format starts.

The label format interpreter does not need headers such as "SOH" and "STX." The data for printing data format is delimited by a "CR" and then transmitted. Changing to the system level interpreter from the label format interpreter is executed by the label format command "E" or "X." When label format ends with "E," defined data is printed and the system level interpreter is started. When label format ends with "X," the system level interpreter is started without printing.

4.5.3 System Level Immediate Execution Commands

These commands are executed as soon as they are received by the printer. They begin with "SOH," i.e. [01].

Command reset	[01] #
Printer status transmission request) (8-byte packet)	[01] A
Pause	[01] B
Stop/cancel	[01] C
SOH command shutdown	[01] D
Transmission of number of remaining sheets to be issued	[01] E
Printer status transmission request (1-byte packet)	[01] F

System Level Occasional Execution Commands

These commands are executed as soon as they are received by the printer. They begin with "STX," i.e. [02].

Setting date and time	[02] A
Setting feedback character transmission validness	[02] a
Date and time transmission request	[02] B
Setting paper length for continuous paper	[02] c
Setting two-page edit mode (double buffer)	[02] d
Changing number of prints for edited format	[02] E
Setting edge sensor selection	[02] e
Label one sheet feed	[02] F
Setting peeling (cutting) position	[02] f
Printing edited or formerly-printed format	[02] G
Graphics data block input command	[02] I
Pause per label printing	[02] J
Extension system command (printer settings)	[02] KD
Extension system command(setting peeling or cuttingposition)	[02] Kf
Setting Y-code-transmission-to-serial-port request	[02] k
Specifying printing contents setting start	[02] L
Setting maximum label length	[02] M
Changing units from inch to metric system	[02] m
Changing units from metric to inch system	[02] n
Setting printing position	[02] O
Paper cut	[02] o
Setting dump mode start	[02] P
Pause in occasional execution	[02] p
Clearing all memory module contents	[02] Q
Clearing memory module contents	[02] q
Setting reflective paper sensor selection	[02] r
Setting paper feed speed	[02] S
Setting one-page edit mode (single buffer)	[02] s
Printing quality test pattern	[02] T

Rewriting specified format register contents	[02] U
Setting memory switch contents	[02] V
Printer version number transmission request	[02] v
Information-in-memory-module transmission request	[02] W
Testing memory card (flash memory)	[02] w
Default module selection	[02] X
Clearing memory module contents (in file units)	[02] x
Sensor level issued to port	[02] Y
Printing printer status	[02] Z
Packing memory module contents	[02] z
Paper position detection sensor voltage transmission request	[02][1B] S
Head disconnection detection	[02][1B] T
Setting ejection (tear-off)	[02][1B] t

Example :

Setting date and time

Code [02] A, w, mm, dd, yyyy, hh, MM, jjj

Setting w Sun 0 Mon 1 Tue 2 Wed 3 Thu 4 Fri 5 Sat 6

mm Month 01 Ð 12

dd Day 01 Ð 31

yyyy Year 4 digits

hh Hour (24-hour display)

MM Minute 00 Ð 59

jjj Spare 000 fixed

Function Sets date and time on the calendar stored in the printer.

Example Input data below represents 15:30 Saturday 1 July 1995.

Input data [02]A6070119951530000

4.5.4 Label Format Commands[2]

The following commands will be valid if the label format command interpreter is turned on with "STX" + "L," i.e. [02] L.

Specifying development method	A
Specifying development method	[1B]B
Setting offset in direction of column	C
Setting number of cuts (2-digit)	c
Setting pixel size in horizontal and vertical direction	D
Completion of setting printing contents (field preparation) and printing labels	E
Entering previous-defined field character string into global table	G
Setting print density (head heat factor)	H
Changing units from inch to metric system	m
Changing units from metric to inch system	n
Setting printable area speed	P
Setting backfeed speed	p
Specifying space between characters	[1B]P

Setting number of prints	Q
Setting offset in direction of row	R
Calling label format	r
Setting unprintable area speed	S
Storing label format	s
Specifying ending code	T
Setting previous field to character-string-replacementmode field	U
Completion of setting printing contents (field preparation)	X
Setting addition of previous-defined printing contents (field data) 1	+
Setting subtraction of previous-defined printing contents (field data) 1	Ð
Setting subtraction of previous-defined printing contents (field data) 2	<
Setting number of prints for same label	^
Setting number of cuts (4-digit)	:
Setting slash zero Character field definition Bar code field definition Ruled line definition Box definition Graphics printing definition Reading out from global register Polygon definition Circle definition Date and time printing definition	z

Character field definition

Code rotate, font, hexp, vexp, point, row, column, d1, d2,.....

Setting

- rotate Sets rotation direction for character data 1, 2, 3, 4
1: 0; 2: 90; 3: 180; 4: 270;
- Font Sets type of character (see table 1)
- hexp Sets expansion rate in horizontal direction 1 Ð 9, A Ð O (A Ð O corresponding to 10 Ð 24)
- vexp Sets expansion rate in vertical direction 1 Ð 9, A Ð O (A Ð O corresponding to 10 Ð 24)
- point Sets size of smooth font
A06 Ð A48 (corresponding to 6pt Ð 48pt)
Sets downloading font ID 100 Ð 999 (see table 2)
Setting of this item is valid only when font is set to 9.
- Row Row address 0000 Ð 9999 Unit: 0.01 inch
- column Column address 0000 Ð 0410 Unit: 0.01 inch
- d1, d2,.. Printing character data Character data ending with CR
By setting [02]Sn (n is a calling character string parameter specifying A Ð P), character string data stored in the global register is picked out and printed.

Function Prints characters for contents data entered with items such as rotation, vertical and horizontal expansion rate, type of font and printing position.

Example Input data below represents that data "123" is printed with system Font 6, vertical and horizontal expansion rate 1 and row and Column address 0.5 inch, and data "ABC" is printed with Smooth font 30pt and vertical and horizontal expansion rate row Address 1.0 inch and column address 0.5 inch.

Input data

[02] n	Sets units to inch
[02] L	Starts label format mode
D22	Sets pixel size
16 11 000 0050 0050 123	Sets data "123" with system font 6
19 11 A30 0100 0050 ABC	Sets data "ABC" with smooth font 30pt
E	Ends label format mode and prints

Bar code field definition

Code Setting rotate, font, thick, narrow, hight, row, column, d1, d2,.....
 rotate Sets rotation direction for bar code 1, 2, 3, 4
 1: 0j 2: 90j 3: 180j 4: 270j

Font Sets type of bar code (see table 3)
 Thick Sets thick bar width in 1-dot units (0.005 inch)
 1 Ð 9, A Ð O (A Ð O corresponding to 10 Ð 24)
 narrow Sets narrow bar width in 1-dot units (0.005 inch)
 1 Ð 9, A Ð O (A Ð O corresponding to 10 Ð 24)
 height Sets height of bar code data by using 3-digit numeric 001 Ð 999 Unit: 0.01 inch
 row Row address 0000 Ð 9999 Unit: 0.01 inch
 column Column address 0000 Ð 0410 Unit: 0.01 inch
 d1, d2,.. Bar code data See table 3

Function Encodes contents data specified with items such as rotation, size of bar code data and printing position into bar code and prints.

Example Input data below is prepared with the following setting and printed.

font	EAN-13
bar ratio (thick:narrow)	3:3
height	0.6 inch
row, column	0.5 inch, 0.5 inch
d1, d2,..	490123456789

Input data

[02] n	Sets units to inch
[02] L	Starts label format mode
D11	Sets pixel size
1F3306000500050490123456789	Sets EAN13 bar code for data "490123456789"
E	Ends label format mode and prints

Graphics printing definition

Code rotate, Y, hexp, vexp, 000, row, column, graphic
Setting rotate Sets graphic data rotation direction to 1 (fixed)
 Y Y fixed
 hexp Sets expansion rate in horizontal direction 1 ÷ 9, A ÷ O (A ÷ O corresponding to 10 ÷ 24)
 vexp Sets expansion rate in vertical direction 1 ÷ 9, A ÷ O (A ÷ O corresponding to 10 ÷ 24)
 000 000 fixed
 row Row address 0000 ÷ 9999
 column Column address 0000 ÷ 0398
 graphic Specifies graphic file name to be printed

Function Picks out file name from the memory module and prints.
Caution Graphic file to be printed with this command must be stored in the memory module.

Example Input data below represents that file name "IMAGE" in the memory module is printed.

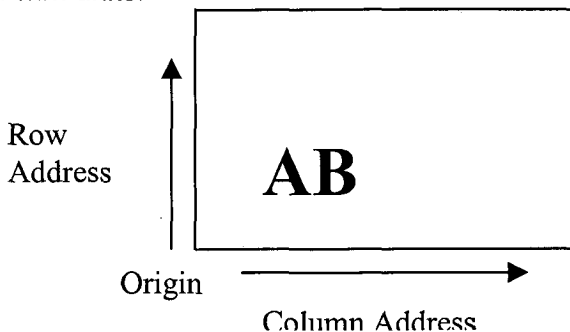
Input data [02] n Sets units to inch
 [02] L Starts label format mode
 1Y1100000500050IMAGE Sets graphic file name "IMAGE"
 E Ends label format mode and prints

Printing Position Specification

The origin for positioning bar codes or characters to be printed on labels is at the bottom left of label, and with the distance from that point, the printing position is designated. The distance upward from the point is called the row address, while the distance rightward from the point is called the column address. Units of 0.01 inch or 0.1 mm are used. Changing units is provided with the m command. In this explanation, the address is specified in 0.1 mm units.

Relevant command: m

After receiving this command, all length specification commands are in 0.1 mm units.



The origin for row address is 2 mm from the top of the paper.

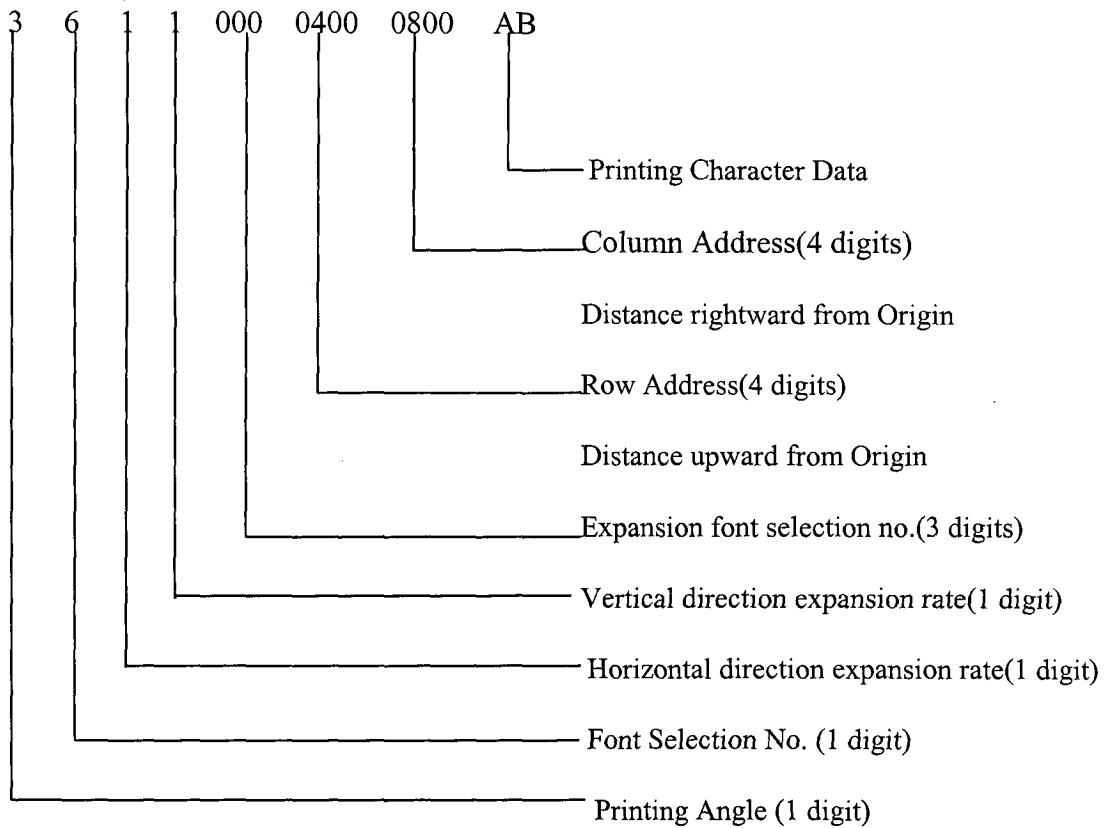
Note: Data in [] is hex.

Character Printing[2]

When characters or strings are to be printed, attribute data such as printing direction and printing position must be added to the top of the strings. With the program example printing "AB" on the previous page, the contents of the character printing are described below.

Program example

[02] m	Sets units to metric system
[02] L	Starts label format mode
D11	Sets pixel size
361100004000800AB	Character data
E	Ends label format mode and prints



- | | | |
|---|----------|-------------------|
| 1 | 0 deg. | Horizontal |
| 2 | 90 deg. | Vertical upward |
| 3 | 180 deg. | Reversed upward |
| 4 | 270 deg. | Vertical downward |

Font selection (ANK/alphabet):

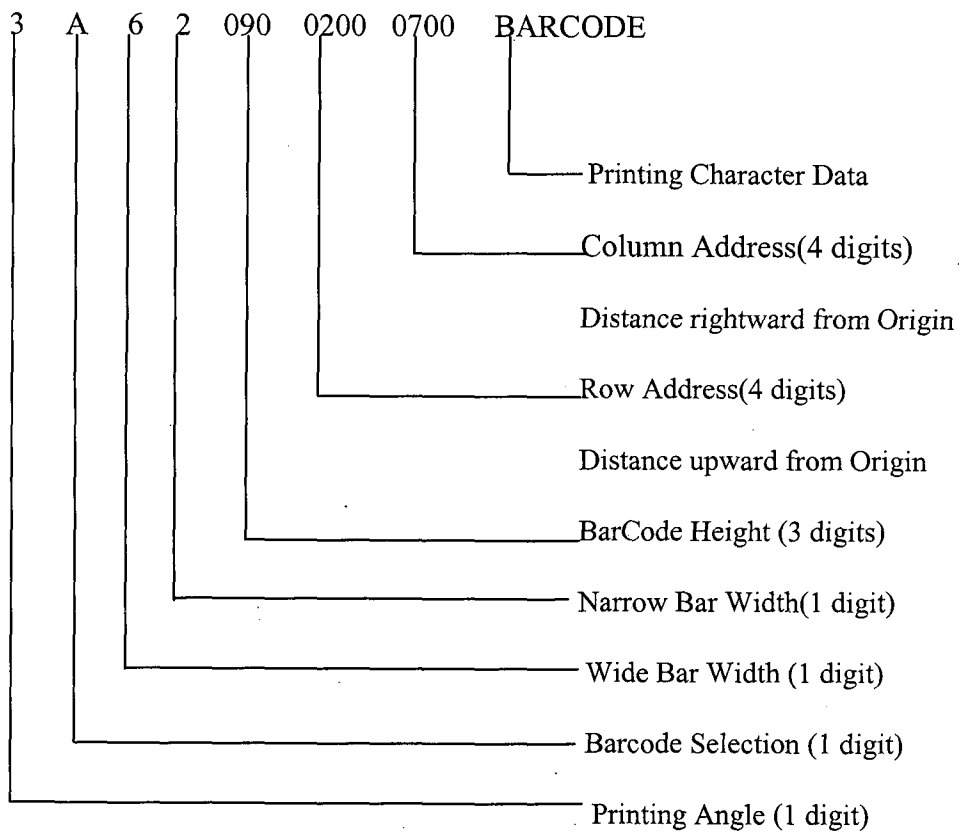
To select the font, specify a font selection number from 1 to 8 or 9.
When font number 9 is selected, the expansion font is selected according to the contents of the expansion font selection number.

391100504000800AB

Bar Code Printing

When bar codes are to be printed, attribute data such as printing position and bar code types must be included in the command.

Program description



- | | | |
|---|----------|-------------------|
| 1 | 0 deg. | Horizontal |
| 2 | 90 deg. | Vertical upward |
| 3 | 180 deg. | Reversed upward |
| 4 | 270 deg. | Vertical downward |

Bar code selection number

Number	Bar code name
A	3 OF 9
D	1 2 OF 5
H	HIBC
I	CODABAR
J	12 OF 5 W/BARS
K	PLESSEY
L	CASECODE
B	UPC-A
C	UPC-E
E	CODE 128 (B)
F	EAN-13
G	EAN-8
M	UPC 2 DIG ADD
N	UPC 5 DIG ADD
O	CODE 93
p	ZIP
Q	UCC/EAN 128
R	UCC/EAN 128 (for KMART)
S	UCC/EAN/128 Random
	Weight
T	Telepen
u	UPS MaxiCode
v	FIM
z	DF417

4.6 Printer status

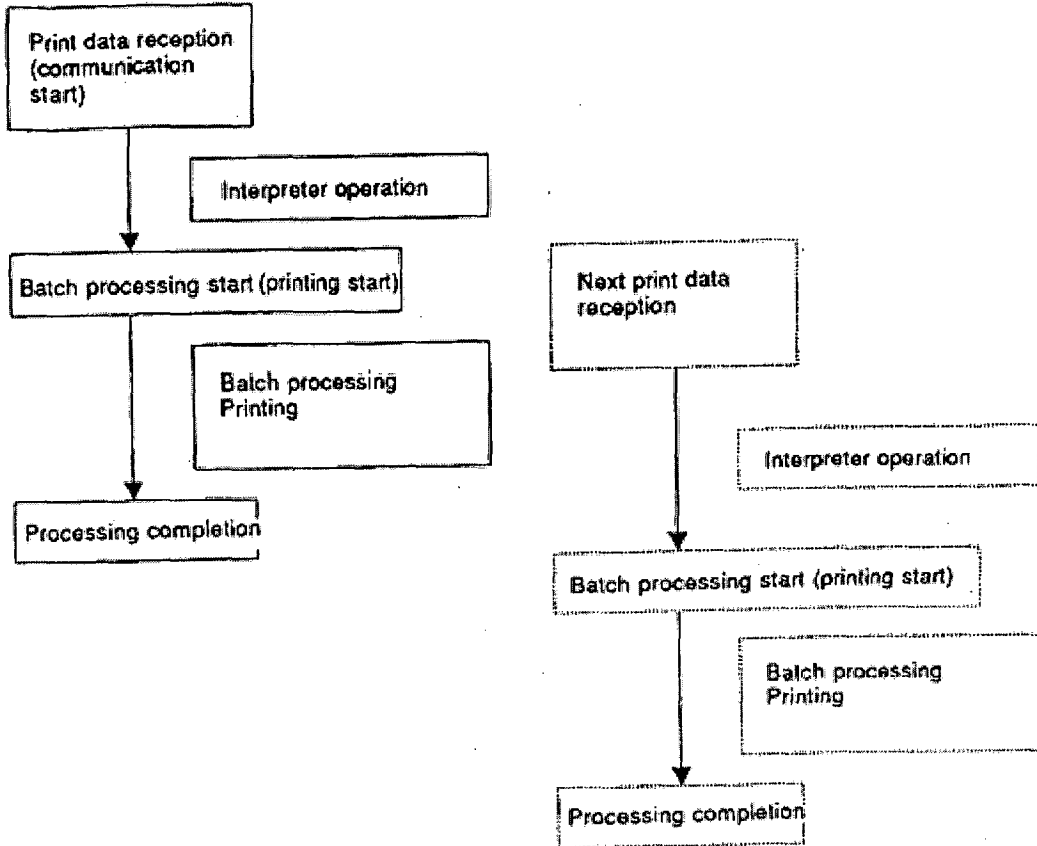


Figure 4.1 Showing Relationship between command interpreter, batch processing and printing[2]

Normal label printing puts the printer in the above status. The printer, however, operates with a double buffer, so if the next printing data is received during batch processing, both interpreter operation and batch processing (printing) may be performed simultaneously.

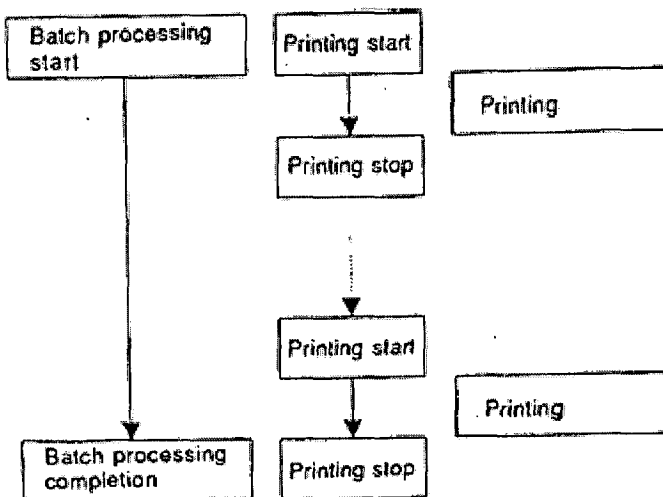


Figure 4.2 Showing Difference between batch processing and printing

As shown in the diagram, printing start and stop may be repeated within a single cycle of batch processing. Therefore, use the operations properly (peeling, auto-cutter, etc.) as needed.

4.7 Communication control flowchart

The following is a reference flowchart for sending and receiving data by using printer transmission request command (01+A, or 01+F). (In XON/XOFF protocol and compatible machine ON mode)

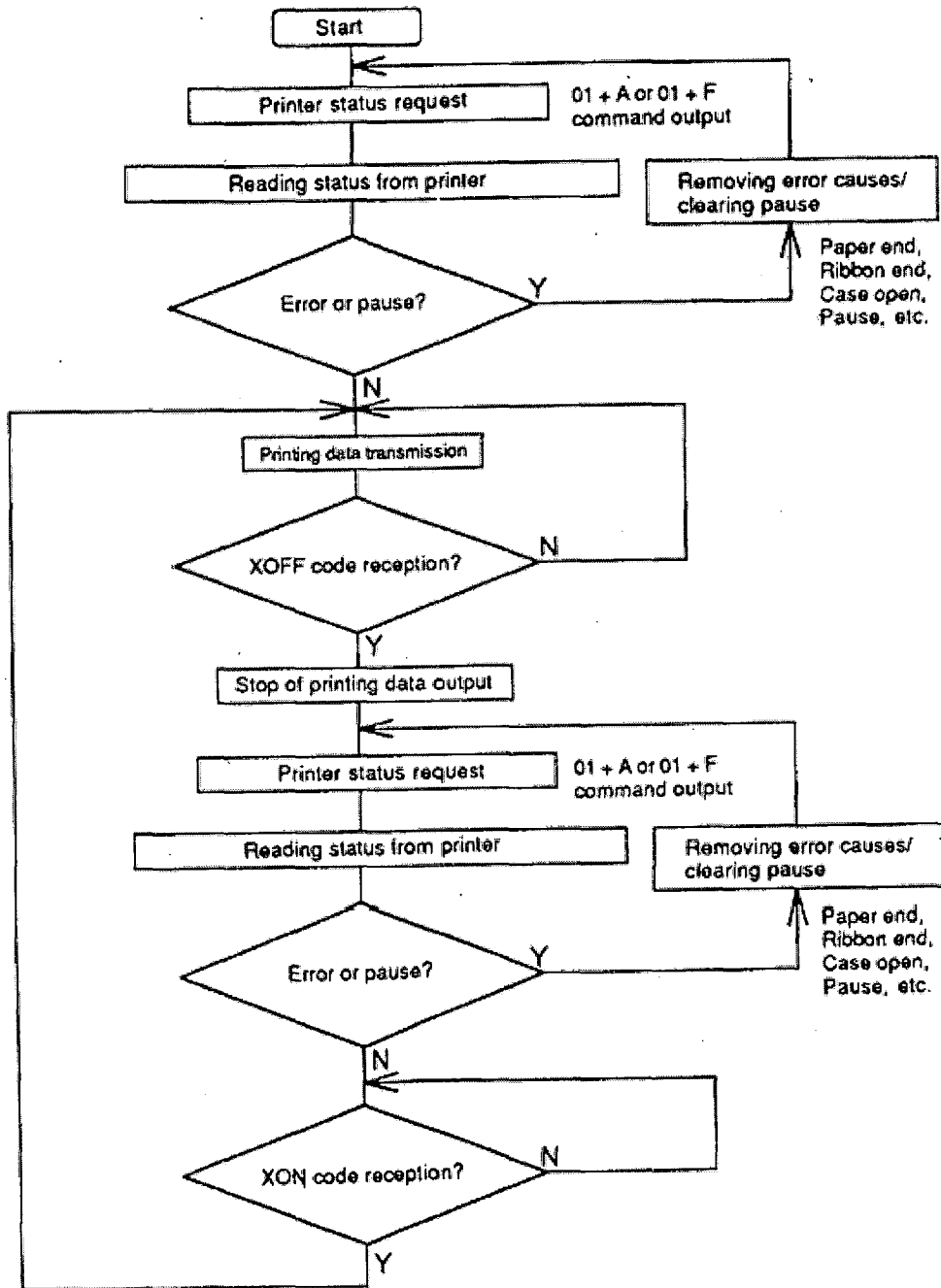


Figure 4.3 Showing Communication control flowchart

5. DataBase Handling

5.1 ADO[9]

Overview

ActiveX[®] Data Objects (ADO) provides a common programming model for any OLE DB data source; it is essentially a collection of objects that expose the attributes and methods used to communicate with a data source. ADO uses general OLE DB providers to access unique features of specific data sources; it also uses native OLE DB providers, including a specific OLE DB provider that provides access to Open Database Connectivity (ODBC) drivers. Designed to replace the need for all other high-level data access methods, ADO can access relational, Indexed Sequential Access Method (ISAM), or hierarchical databases, or any type of data source—as long as there is an ODBC-compliant driver.

ADO's ease of use, speed, and low memory overhead make it ideal for server-side scripting. In fact, ADO is the recommended technology for data access for ASP applications. ADO can be called directly from server-side scripts or from business components.

Unlike earlier data access methods, ADO does not require navigation through a hierarchy to create objects; most ADO objects can be created independently, which allows greater flexibility in reusing objects in different contexts and reduces memory consumption. ADO also takes advantage of ODBC 3.0 connection pooling for ODBC data sources, and session pooling for OLE DB providers. This eliminates the need to continuously create new **Connection** objects for each user, which is very resource intensive.

What ADO cannot do, however, is provide remote data to the client. Once the data has been retrieved and sent to the browser, the user cannot easily manipulate it or make changes to it within the client application. Data operations—including filtering and record modifications—must take place on the server, where the actual data manipulation objects reside.

OLE DB, the foundation of Microsoft's Universal Data Access model, is a set of COM interfaces that provides a standard way for programs to access data. The way your application uses ADO functionality will be partially determined by whether or not there is an OLE DB provider for the data. ADO is designed to work with OLE DB, and in most instances your ADO components will communicate with databases through OLE DB; you can also use ADO to communicate directly with the ODBC driver, if no OLE DB provider is available. Using ADO through an OLE DB provider has an impact on the following areas:

- Using Stored Procedures with ADO: Describes several key issues you should consider when using stored procedures.
- Choosing the Client Networking Library for ADO: Explains the issues surrounding the selection of a networking library and its impact on database access speed.

ActiveX Data Objects are a language-neutral object model that expose data raised by an underlying OLE DB Provider. The most commonly used OLE DB Provider is the OLE DB Provider for ODBC Drivers, which exposes ODBC Data sources to ADO.

The ADODB Library contains additional server side objects (Connection, Command, Error, Parameters, and so forth). These are best used within server side components to communicate with the database.

ADO is dependent upon the data provider it uses. The easiest/quickest way is to check the Supports property to confirm that the connection or recordset you opened supports the functionality you actually need. You should also confirm that the CursorType, and LockType match what you expected. If the underlying provider cannot support the cursor you requested, ADO downgrades these properties to get the cursor that is closest to what you requested.

A more in-depth approach involves analyzing the underlying data provider you are using underneath ADO. The most common provider is the OLE DB Provider for ODBC Drivers that exposes ODBC Data sources to ADO. You can use the Properties collection(s) for the Connection, Command, Recordset, and Field objects and compare the results displayed there to the OLE DB Specification and the OLE DB Leveling Document, both included with the OLE DB SDK.

ADO will not automatically define each value for each property exposed in the collections. The property will only be set when the operation you are performing with ADO actually needs to use that property exposed by the underlying provider. This is for performance, and varies from other object models such as DAO which initialize any and all properties whether the operation you are performing utilizes them or not.

Using the Recordset.Supports property is the easiest way to verify that what you expect is actually what the provider exposed to you, given the specific type of connection and recordset you have opened.

5.2 The ADO objects[9]

There are three major objects in the ADO environment: connection, command, and recordset. Unlike the DAO recordset, the ADO recordset can be created from scratch without accessing any prior objects. Here, for instance, is some sample code that creates a recordset:

```
Dim rec As Recordset
Set rec = New Recordset
rec.Source = "Select * From Table1;"
rec.ActiveConnection = "DSN=MyDSN;"
rec.Open
```

The second line of code really brings home the difference between ADO and DAO: A recordset is created using the New command rather than through the CreateRecordset method of some other DAO object. These are "empty" recordsets because they contain no records or, indeed, any association with a datasource from which records could be retrieved. In DAO, when a recordset is created, it draws information from

the object from which it's created (either a Database or a QueryDef object) and thus "knows" which database it's to draw its records from.

Since this recordset doesn't have any place from which to get this information, it must have it passed to it through its properties. The Source property allows you to specify the command or parameters that are used to specify the data to be retrieved. In this case, that's a SQL statement.

The ActiveConnection property allows you to specify where the data for the recordset is to come from. Like an ODBC connect string, this can be quite a long list of parameters, including the name of the database, the userid and password to log on with, and much more. One of the nice things about converting to ADO is that you can continue to work with your existing ODBC databases.

The recordset's Open method uses the information specified in the properties to retrieve the specified records into the recordset. From there on, the DAO methods can be used to: MoveFirst, MoveNext, and MoveLast.

5.2.1 Recordset Object Basics

To work with records in a database by using Visual Basic or VBScript code, you use ADO **Recordset[9]** objects. A **Recordset** object represents the records from a single table or the set of records returned by executing a command, such as an SQL string, an Access query, or a SQL Server stored procedure.

User can open a **Recordset** object in ADO by using any of these three methods:

- The **Execute** method of the **Connection** object
- The **Execute** method of the **Command** object
- The **Open** method of the **Recordset** object

The syntax for each of these methods is as follows:

```
Set recordset = connection.Execute ([CommandText, [RecordsAffected, [Options]])
```

```
Set recordset = command.Execute ([RecordsAffected, [Parameters, [Options]])
```

```
recordset.Open [Source, [ActiveConnection, [CursorType, [LockType, [Options]]]]
```

Although using the Execute method of the Connection or Command object returns a Recordset object, these methods are primarily intended for executing commands (typically SQL strings) that don't return records; queries that are called *action queries* in Access. When they are used to return a set of records, they create only Recordset objects of the Forward-only, Read-only cursor type, and there is no way to specify any other cursor type. Because of this limitation, we will not discuss using the Execute method of the Connection or Command object for opening Recordset objects for the purpose of creating a client-side set of records. For information about cursor types, see "Specifying Cursor Types" later in this chapter.

However, a Command object can be passed to the Open method of the Recordset object as the Source argument. This can be useful in two ways:

- You can use the Prepared property of the Command object to optimize and precompile the command. If you will be using the Command object more than once within the scope of your procedure, this will optimize the performance of your procedure.
- A Command object is required if you want to supply parameters to a query that can be reused efficiently.

These applications of the Command object are discussed later in this chapter. The following sections discuss how to use the Open method of the Recordset object, which provides more options and greater flexibility when you open Recordset objects.

5.2.2 Using the Open Method of a Recordset Object

The syntax for the **Open** method of an ADO Recordset object is as follows:

recordset.Open Source, ActiveConnection, CursorType, LockType, Options

The following table briefly describes each of the arguments of the **Open** method.

Argument	Description
<i>Source</i>	Optional. A valid Command object variable name, an SQL statement, a table name, a query name (Access), a stored procedure name (SQL Server), or the file name of a Recordset object previously saved by using the Save method of the Recordset object.
<i>ActiveConnection</i>	Optional. A valid Connection object variable name, a String containing connection string parameters, or if you are creating a Recordset object associated with the current Access database, you can pass CurrentProject. Connection for this argument.
<i>CursorType</i>	Optional. A constant that determines the type of cursor that the provider should use when it opens the Recordset object.
<i>LockType</i>	Optional. A constant that determines what type of locking (concurrency) the provider should use when it opens the Recordset object.
<i>Options</i>	Optional. A constant that indicates how the provider should evaluate the <i>Source</i> argument if it represents something other than a Command object, or that the Recordset object should be restored from a file where it was previously saved.

5.2.3 Connecting a Recordset Object Variable

When you use some OLE DB providers (such as the Microsoft Jet 4.0 and SQL Server OLE DB providers), you can create objects independently of a previously defined object by using the argument of the method to pass a connection string. ADO still creates a object, but it doesn't assign that object to an object variable. For example, the following code fragment opens a object by passing a connection string to the **Open[9]** method.

```
Dim cnn1 As ADODB.Connection
Dim rst As ADODB.Recordset
Dim strConnect As String
Dim strSQL As String

strConnect = "Provider= sqloledb ;" & _
  "Data Source= quantam ;Initial Catalog=testing;" & _
  "User Id=shaky;Password=shaky;"
cnn1.Open strConnect
strSQL = "SELECT * FROM Customers WHERE Region = 'WA'"
rst.Open strSQL, cnn1, 3, 1
(recordset.Open Source, ActiveConnection, CursorType, LockType, Options)
```

5.2.4 Specifying Cursor Types[9]

At any given time, a **Recordset** object can only refer to one record within the set as the *current record*. The software functionality that lets you work programmatically with a set of records is referred to as a *cursor*. You can think of a cursor as a device that you can use to scroll through a set of records in a database to read, add, delete, or update records. There are four types of cursors for **Recordset** objects in ADO : Dynamic, Keyset, Static, and Forward-only.

To specify the cursor type, set the **CursorType** property of a **Recordset** object before you open it.

The following table describes the features of each cursor type and lists the constants you can use to set or read the **CursorType** property.

Cursor type	Constant	Description
Dynamic	ADOpenDynamic	Reflects any new additions, changes, and deletions made by other users, and allows all types of movement through the Recordset object that don't rely on bookmarks; allows bookmarks if the provider supports them. This cursor type isn't supported by the Microsoft Jet 4.0 OLE DB Provider.
Keyset	ADOpenKeyset	Behaves like a Dynamic cursor except that

		it doesn't contain any new or deleted records added by other users. Any data changes made by other users to the records available when the Recordset object was opened will still be visible. A Keyset cursor always supports bookmarks and therefore allows all types of movement through the Recordset object.
Static	ADOpenStatic	Provides a static, but updatable, copy of a set of records. Always allows bookmarks and therefore allows all types of movement through the Recordset object. Any additions, changes, or deletions by other users will not be visible until the Resync method is called. This is the only type of cursor allowed when you open a client-side (ADOR) Recordset object.
Forward-only	ADOpenForwardOnly	Behaves identically to a Static cursor except that it only allows you to scroll forward through records. This improves performance in situations where you need to make only a single pass through a Recordset object. (Default) Note The RecordCount property for a forward-only Recordset object always returns - 1 because ADO can't determine the number of records in a forward-only Recordset object. To get a valid count of records when using a Recordset object to work with an Access database, you must use either a Keyset cursor or a Static cursor.

5.2.5 Specifying Locking[9]

The LockType property or argument specifies what kind of locking is used while you edit records in the Recordset object. *Locking* is used to regulate what other users in a multiuser database can do with a record while it is being edited. If you don't set the LockType property, read-only locks will be used by default, which will prevent you from editing records. To edit the data in a Recordset object, you must set the LockType property before you open it, or else you must pass the LockType argument to the Open method. The LockType property is read/write before a Recordset object is opened (or after it is closed), and read-only while it is open. The following table lists the constants you can use to set or read the LockType property and describes how each lock functions when you are editing records.

Constant	Description
adLockReADOnly	Read-only — you cannot edit the data. (Default)
adLockPessimistic	Pessimistic locking, record by record — the provider does what is necessary to ensure successful editing of the records, usually by locking records at the data source as soon as you start editing records. No other users can read or edit the data until you either save changes with the Update method or cancel them with the CancelUpdate method.
adLockOptimistic	Optimistic locking, record by record — the provider locks records only when you call the Update method. Other users can read, edit, and save changes to the same record while you have it open.
adLockBatchOptimistic	Optimistic batch updates — required for batch update mode as opposed to immediate update mode.

5.3 Data Access Using ActiveX Data Objects (ADO)

ActiveX Data Objects (ADO) is designed to be an easy-to-use application-level interface to any OLE DB data provider, including relational and non-relational databases, e-mail and file systems, text and graphics, and custom business objects, as well as existing ODBC data sources. Virtually all of the data available throughout the enterprise is available using the ADO data access technology.

ADO is easy to use, language-independent, implemented with a small footprint, uses minimal network traffic, and has few layers between the client application and the data source — all to provide lightweight, high-performance data access.

The general characteristics of ADO are:

- Ease of use.
- High performance.
- Programmatic control of cursors.
- Complex cursor types, including batch and server- and client-side cursors.
- Ability to return multiple result sets from a single query.
- Synchronous, asynchronous, or event-driven query execution.
- Reusable, property-changeable objects.
- Advanced recordset cache management.
- Flexibility — it works with existing database technologies and all OLE DB providers.
- Excellent error trapping.

The simple semantics of ADO and universal application mean minimal developer training, rapid application development, and inexpensive maintenance.

6. Standards And Guidelines

6.1 Introduction

S&G is used to name different work products in the projects. It also defines the way to name various program components. The convention described here applies to Visual Basic, ASP and VB Script code.

6.2 File Naming Conventions[6]

Physical File Naming Convention

In order to identify a file a unique name has to be assigned to every file. The Name given to a file should be able to uniquely identify some features to the file.

1	2	3	4.....
M	FFFF	Y	.EXT

Where

M – Module

I	<u>I</u> Inventory Module
S	<u>S</u> ales Module
R	<u>R</u> eturns Module
L	Search / <u>L</u> ookup Module
D	Reports and <u>D</u> ata Analysis Module
O	<u>O</u> rders Module
C	Customer Service
A	<u>A</u> ministration Module

FFFF... – Descriptive File Name

Y – Document Type

F	Functional Specs
P	Program specs
M	User Manual
C	Code Files (Main ASP,HTM Code etc)
H	Help Files
D	Defect Log
U	Unit Test reports
I	Integration Test report
Q	QAR
A	Release Advise
S	Style Sheets
J	JavaScript Files
E	Back End Processing Files / scripts File

EXT – File Extension

DOC	For Document Files
ASP	For ASP scripts files
HTM	For HTML files
JS	For JavaScript Files
CSS	For Cascading Style Sheets
VBP	Visual Basic Project
FRX	Visual Basic Form Binary File
DLL	Dynamic Linked Library

DLL Naming Convention[6]

The DLLs, which are made for ISMA projects are a sort of wrapper around a set of related tables in the database. The name of the DLL must specify some of the basic purpose / feature of the DLL. They must be named as per the following convention.

<Logical Name>.DLL

For example product.dll – this specifies that this dll consist of those functions, which are related to product or its related tables.

The name of the class file related to the DLL should be prefixed with cls that is

cls<Logical Name>.DLL

For example clsProduct.cls

Similarly, the related interface, if any should be prefixed with “i” that is

i<Logical Name>

6.3 Coding Standards And Guidelines

There are coding conventions for programs. Coding conventions are programming guidelines that focus not on the logic of the program but on its physical structure and appearance. They make the code easier to read understand, and maintain. Coding convention can include.

- **Naming Convention for objects, Classes, variables and procedures**
- **Standardized formats for labelling and commenting code.**
- **Guidelines for spacing, formatting and indenting.**

Variables & Constants Naming Convention

This topic lists recommended conventions for constants and variables supported by ASP files. It also discusses the issues of identifying data type and scope. Variables should always be defined with the smallest scope possible. Global variables can create

enormously complex state machines and make the logic of an application extremely difficult to understand.

In a Visual Basic application, global variables should be used only when there is no other convenient way to share data between forms. When global variables must be used, it is good practice to declare them all in a single module, grouped by function. Give the module a meaningful name that indicates its purpose, such as Public.bas.

It is good coding practice to write modular code whenever possible. For example, if application displays a dialog box, put all the controls and code required to perform the dialog's task in a single form. This helps to keep the application's code organized into useful components and minimizes its run-time overhead.

With the exception of global variables (which should not be passed), procedures and functions should operate only on objects passed to them. Global variables that are used in procedures should be identified in the declaration section at the beginning of the procedure. In addition, arguments passed to subs and functions using ByVal, unless explicitly need to change the value of the passed argument.

Scope	Declaration	Visible in
Procedure-level	'Private' in procedure, sub, or function	The procedure in which it is declared
Module-level	'Private' in the declarations section of a form or code module (.frm, .bas)	Every procedure in the form or code module
Global	'Public' in the declarations section of a code module (.bas)	Everywhere in the application

The name of the variables should be as per the following naming convention:

<X><YYY><ZZZZZZZZ...>

where X is as follow

- f Local Variables local to function(Local with respect to the particular function)
- i Input parameter to a sub routine (to function or sub procedures)
- p Page level Variables(Local respective to a page)
- q Query string Variable(Variables to be passed from query string)
- h Hidden Fields Variables
- o Output parameter to a sub routine (to function or sub procedures)

Where Y is the datatype of the variable. List of datatypes is as follows:

For purpose of readability and consistency, use the following prefixes with descriptive names for variables in code.

Data type	Prefix	Example
Boolean	bin	l binFound
Byte	byt	g bytRasterData
Date(Time)	dtm	a dtmStart
Double	dbl	s dblTolerance
Error	err	l errOrder
Integer	int	l intQuantity
Long	lng	l lngDistance
Single	sng	l sngAverage
String	str	l strFirstName
User-defined type	udt	l udtCustomer
Const	con	l conTax
Collection	cll	l cllCustomer
Variant	vnt	m vntChecksum
Objects	obj	l objCurrent

Note : - For arrays ,variables should be names as <x><yyy>Arr<Logical Name>

Where x is the scope level alphabet i.e. either l / m / g / a / s only
Where yyy is the prefix for the data type like str / int / vnt

For Example

lstrArrName for array of sting datatype (local)
mintArrName for array of integer data type (module level)
avntArrName for array of variant datatype (application scope)

For input /output parameter , use the following convention

<x><yyy><Logical Name>

where x is "i" or "o"

where yyy is the data type of the value

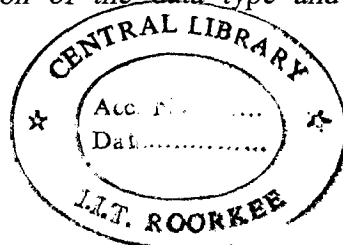
Input parameter for example – istrName , iarrListOfNames etc.

Output Parameter for example – ostrName , oarrListOfName , etc

Note : 1. The variable name should not exceed 30 character as a whole .

2. though in VBSCRIPT the variables declared are all of type VARIANT but still use the above naming convention as per the data type of the data value of the variables.

3. Use Hungarian Notation of the data type and logical name part of the variables.



For Sub Programs following scheme is used :-

<YYYY><ZZZZZZ...>

where y is func – function , proc - Procedure

and Z is descriptive name of the sub program e.g fnGetDeptList , sbCheckPrice .
The sub program name should not exceed 30 as a whole.

Descriptive Variables and Procedure Names

The body of a variable or procedure name should use mixed case and should be as long as necessary to describe its purpose . In addition,function names should begin with a verb , such as sblnitdeptArray or fnCloseConnection.

For frequent used or long term,standard abbreviations are recommended to help keep name length reasonable .In general variable names greater than 32 character can be difficult to read on VGA displays.

When using abbreviation, make sure they are consistence throughout the entire application. Randomly switching between Cnt and Count within a project will lead to unnecessary confusion.

Structured Coding convention

In addition to naming convention ,structured coding convention ,such as code commenting and consistent indenting, can greatly improve readability. This topic discusses standards for these areas.

Code commenting convention

All Procedures and functions should begin with a brief comment describing the functional characteristics of the procedures (what it does) . This description should not describe the implementation details (how it does) because these often change over time ,resulting in unnecessary comment maintenance work .

Section Heading	Comment Contents
Project Name	Name of Project
Purpose	What the Procedure does (not how)
Author	The name of the person who has developed the procedure
Date	The Date on which development of the procedure started
Date Modified	Date on which modification have been done after review

7. Software testing

Testing for “*Inventory-Sales Management And Analysis System*” is done with the intent of checking the application for errors or defects to rectify them before the implementation. No matter how good the programming is, bugs usually creep in. There could be logical errors, erroneous assumptions made while coding, failure in understanding the exact requirements, etc. Testing of this software is done to find and rectify all those errors.

7.1 Testing Objectives

There is a common misconception that software testing is an activity to prove the correctness of software. The reality however is that testing should be viewed more as a destructive process than a constructive process of software development.

“The objective of the testing is not to show the absence of defects, but to show their presence”. Hence, any structured testing activity has to comply this underlying objective.

Testing for “*Inventory-Sales Management And Analysis System*” is done with the following objectives in mind:

- To discover yet undiscovered errors
- To determine whether all user requirements are being met
- To find out whether all design specifications are being met
- To ensure that code compiles with no error
- To ensure that there is no memory leaks
- To find out that there are no ‘dead code’ areas i.e. code which never gets executed
- To ensure that maximum errors are discovered in this stage to minimize the debugging effort and cost
- To ensure that the software produced is of high quality.

7.2 Types of Testing

Testing for “*Inventory-Sales Management And Analysis System*” is done in the following order:

1. Unit Testing
2. Integration Testing
3. Stress Testing
4. Performance Testing

7.2.1 Unit Testing

Unit Testing is performed after the coding steps in “*Inventory-Sales Management And Analysis System*”. To perform Unit Testing, unit test cases are designed to uncover maximum errors.

Different test cases are used to uncover different kinds of errors and bugs. Each test had its own significance in Unit Testing. These Inputs are mainly as follows:

1. Interface Checkpoints
2. Local Data Structure
3. Independent Paths
4. Boundary Conditions
5. Error Handling Paths

All these tests are very useful to understand and uncover the errors.

1. Interface Checkpoints – It is performed to check the information flow across the input and output of the module to be tested. Various tests performed are: -

1. The number of input parameters should be equal to output parameters
2. Correct number and type of arguments should be passed to the stored procedures
3. Global Variable definitions are consistent across modules

It is tested that data that is entered should be retrieved properly. Records were entered and saved in the database and then tests are carried out to ensure that proper records are retrieved and at proper places.

Errors Generated – The Errors are faced during the retrieval of some records.

7. Software testing

Testing for “*Inventory-Sales Management And Analysis System*” is done with the intent of checking the application for errors or defects to rectify them before the implementation. No matter how good the programming is, bugs usually creep in. There could be logical errors, erroneous assumptions made while coding, failure in understanding the exact requirements, etc. Testing of this software is done to find and rectify all those errors.

7.1 Testing Objectives

There is a common misconception that software testing is an activity to prove the correctness of software. The reality however is that testing should be viewed more as a destructive process than a constructive process of software development.

“The objective of the testing is not to show the absence of defects, but to show their presence”. Hence, any structured testing activity has to comply this underlying objective.

Testing for “*Inventory-Sales Management And Analysis System*” is done with the following objectives in mind:

- To discover yet undiscovered errors
- To determine whether all user requirements are being met
- To find out whether all design specifications are being met
- To ensure that code compiles with no error
- To ensure that there is no memory leaks
- To find out that there are no ‘dead code’ areas i.e. code which never gets executed
- To ensure that maximum errors are discovered in this stage to minimize the debugging effort and cost
- To ensure that the software produced is of high quality.

7.2 Types of Testing

Testing for “*Inventory-Sales Management And Analysis System*” is done in the following order:

1. Unit Testing
2. Integration Testing
3. Stress Testing
4. Performance Testing

7.2.1 Unit Testing

Unit Testing is performed after the coding steps in “*Inventory-Sales Management And Analysis System*”. To perform Unit Testing, unit test cases are designed to uncover maximum errors.

Different test cases are used to uncover different kinds of errors and bugs. Each test had its own significance in Unit Testing. These Inputs are mainly as follows:

1. Interface Checkpoints
2. Local Data Structure
3. Independent Paths
4. Boundary Conditions
5. Error Handling Paths

All these tests are very useful to understand and uncover the errors.

1. Interface Checkpoints – It is performed to check the information flow across the input and output of the module to be tested. Various tests performed are: -

1. The number of input parameters should be equal to output parameters
2. Correct number and type of arguments should be passed to the stored procedures
3. Global Variable definitions are consistent across modules

It is tested that data that is entered should be retrieved properly. Records were entered and saved in the database and then tests are carried out to ensure that proper records are retrieved and at proper places.

Errors Generated – The Errors are faced during the retrieval of some records.

2. *Local Data Structure* – It is a common source by which large number of errors can be generated. It may be: -

1. Incorrect database name
2. Improper or Inconsistent typing
3. Incorrect variable names (names not according to coding standards)
4. Invalid table name or default value – a number value is given in place of character value. Therefore, the data structure cannot directly recognize the table name.
5. Inconsistent data type for table field in MS –SQL 7
6. Impact of global data on the module is also tested.

Errors Generated – Various incorrect variable names are detected that are not following the coding standards. The accuracy of the data must be checked with checking points so that if user inputs incorrect data then the user must be informed. It is similar to working with a virtual operating system where each incorrect step taken by the user results in a message box informing the user about the fault.

3. *Independent Paths* – Selective testing of execution paths is very important in unit testing. Various test cases are developed to test the control flow, computations and comparisons. Basis Path Testing and loop testing is done to uncover errors in which various values are selected to determine the control flow.

Errors Generated – No errors are generated in testing the control flow.

4. *Boundary Conditions* – To see behaviour of module at minimum and maximum values this test is very helpful. Various tests performed are:

1. By providing input more than the upper limit. For example, by inputting a large number of characters in the Text field.
2. Another way to uncover the errors is by providing the blank spaces in the field.

Errors Generated – By providing the values that are more than the upper limit, overflow occurs where client side validations are not used. Otherwise, validations are used to validate the input.

5. *Error Handling Paths* – During the test case, potential errors must be handled. These errors should be carefully managed because they do not tell much about themselves. Error description does not provide enough amount of information to assist errors. The most common errors that usually creep in are in form of database errors. Appropriate messages are provided as and when errors occur.

Errors Generated - Error description is sometimes unintelligible i.e., they do not speak about what they are. It is very difficult to understand them. Otherwise, good description is provided about the nature of errors.

7.2.2 Integration Testing

The “*Inventory-Sales Management And Analysis System*” consists of many modules. All these modules are tested unit – wise before conducting Integration Testing. Each module takes some input from other module and provides the output to another module.

To conduct the Integration Testing, it is necessary to realize that each of the above modules consists of series of separate screens. Portfolio module has many sub – modules also containing different screens. Each screen provides some services to its adjacent screens.

7.2.3 Stress Testing

Stress Tests are designed to confront programs with abnormal situations. For Stress Testing of “*Inventory-Sales Management And Analysis System*” Stress Test cases are designed through which abnormal conditions are applied for software’s execution.

Some test cases on which “*Inventory-Sales Management And Analysis System*” is tested are: -

1. A large number of users (about 50) are asked to work on single page entering data at a very high rate to test how input functions respond
2. Retrieving data from a database by many users to test how database will respond
3. Seizing very large amount of available memory by enlarging the number of customers in the database

7.2.4 Performance Testing

The performance testing is done to judge the performance of the product under specific conditions. To conduct performance testing firstly the nature of software should be known. The nature of “*Inventory-Sales Management And Analysis System*” is a real time and time bounded application program.

Another essential need for the Performance Testing is to consider the whereabouts of both the hardware and the software instrumentation. In the developed product, resource utilization is measured to know this fact.

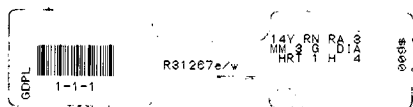
To increase the performance of the system that is the speed of retrieval of data from the database, some technologies are used. To retrieve the data from the database, which is MS – SQL Server 7.

8. Conclusion

This has been an altogether different experience working on a live project for about five months in CITEs. The application developed over there was concerning Inventory management and BarCode development of the items in the Inventory. Inventory management mainly concerned with new inventory at the HOME Office and how they are being dealt there. It involved entering item details in the database through input Screen or through importing file, which contains details of items. While importing item details from a file, it is checked for errors and if any they are corrected before being saved into the database. For searching a particular item in the Inventory, screen is developed in which any item information is provided and item is search from the database. Whenever a new item arrives in Inventory BarCode is generated for easy identification of the item for further use or reference. During Software testing Defects Reports are made with the help of which Errors are removed from the application.

Result

The Screens developed during application development are shown in Appendix-A, Appendix-B and Appendix-C. Appendix-A consists of Screens for Inventory Management. Figure A.1 Shows Screen through which item details can be entered for new items coming to the inventory. Figure A.2 Shows Screen through which Item details can be entered into the database by importing file but before correcting all errors present in the file. Figure A.3 Shows errors encounter while importing Inventory from the file and these errors can also be corrected in this screen. Figure A.4 Shows screen consisting of search Criteria through which any item can be searched in the database. Appendix-B consists of defect Report that is made during testing phase of the SDLC. This defect report is helpful in correcting errors in a program. Defect report also consists of Screen shot of error obtained during testing. Appendix-C consists of screen, which is used for printing tags. These tags can be printed one by one or in batches. Some of the tags printed are shown below which consists of item details like Inventory No., Price, Vendor of Item, Style of item. BarCodes are printed for Inventory Number only other details are printed as it is.



Scope for Further Development

Scanned picture of Item can also be kept in database so that item can be viewed how it looks like. Providing facility to enter Inventory details on the Internet. Generating BarCode for different types of Labels other than shown above.

Appendix – A

Screen Shots for Inventory Management

Richa Creations. - [New Inventory (frmNewInventory02C)]

Inventory Sales Return Orders Search Reports Data Analysis Admin Customer Care Tools Exit

Inventory

Major Class Minor Class

Style Category Inventory Number

Primary Material

CASE HOME Program ID

Item Weight Units

Ring Width Quantity 1

Stone Definition Stone

Per Quantity Cost

Retail Wholesale

% Markup

List Price

Vendor ID

Vendor Invoice Date

Vendor Name

Vendor Invoice

Vendor SKU

Style Number

Cost Sheet History Add Duplicate Edit End/Lookup Print Tag Save Cancel Exit

sa SCRE CAPS NUM 20/05/02 02:57

Figure A.1 New Inventory(Form to Enter New Inventory)

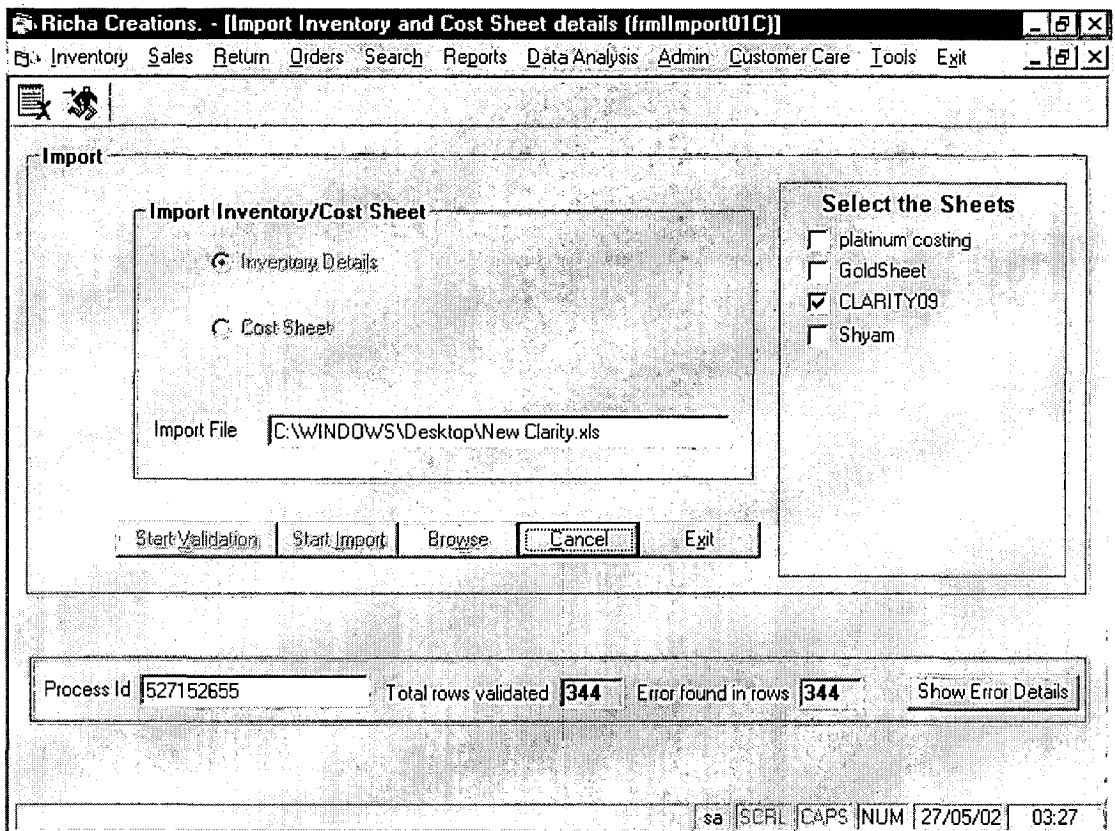


Figure A.2 Import Inventory(Import New Inventory from a file)

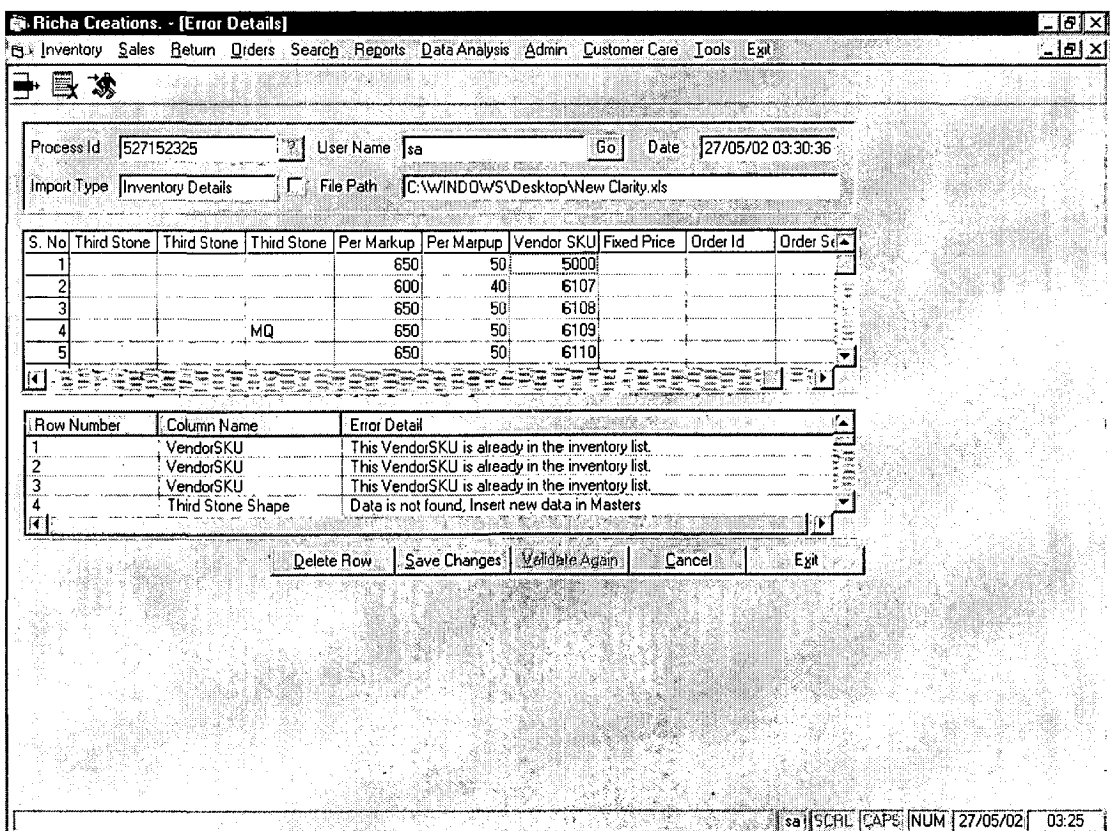


Figure A.3 Error Details(Error Found During Importing Inventory File and correcting them)

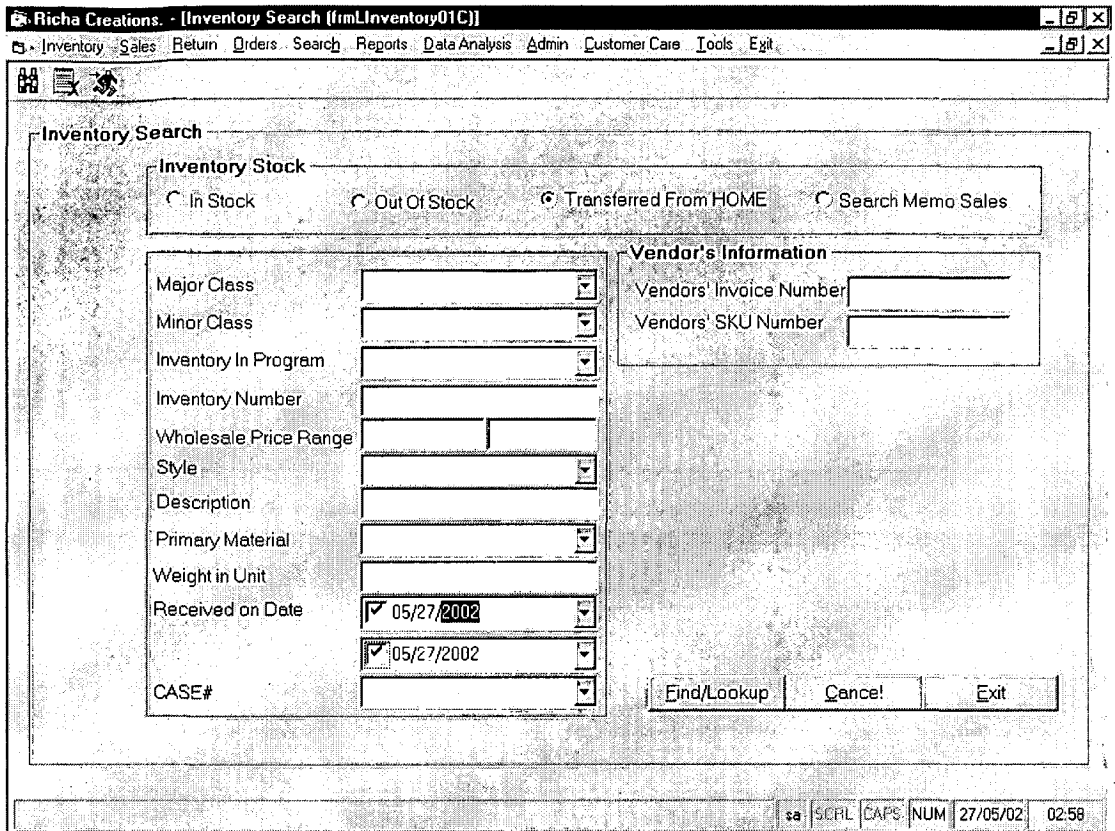


Figure A.4 Search Inventory(Searching Inventory)

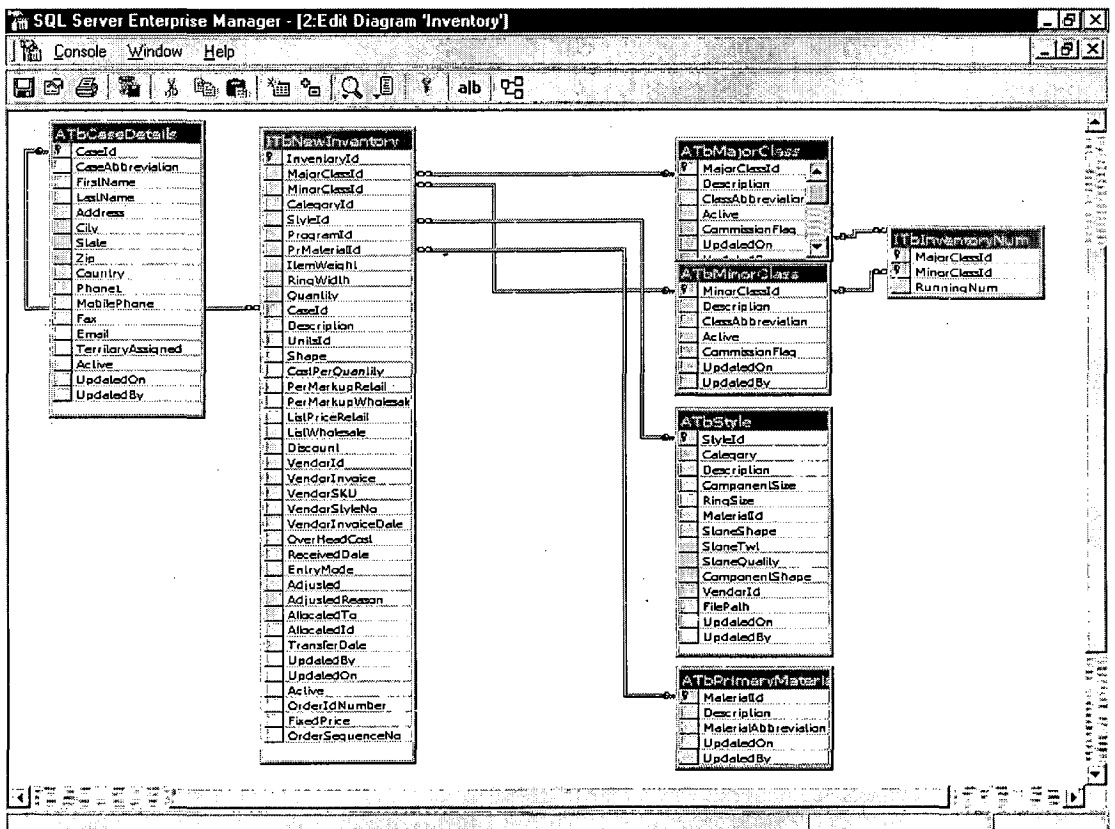


Figure A.5 Database Relationship Diagram

Appendix B

Defect Report

Project Name :

Item Reviewed :

Ver.no :

Defects :

Sl No	Location of Defect	Description of defect	Cat*	Corrective Action	Sr c **	Closed by / on	Verified by / on
1	FrmITransfer04 C	When Delete Row button is clicked , it shows alert that No Data Available for Removal . This alert message comes with the symbol X in red color. This alert message should not be come with X symbol.	R	Corrected		Tara Chand, June 11, 2002	
2		Label below the label Total Cost is missing.	C	Corrected		Tara Chand, June 11, 2002	
3		Textbox below the textbox of Total Price is editable and it allows characters to be added to it.	C	Corrected		Tara Chand, June 11, 2002	
4		Fedex Charges text box does not allow me to enter 9.5 as the charge. It does not allow me to enter anything after decimal.	C	Corrected		Tara Chand, June 11, 2002	
5		It saves data successfully after entering data as 23 . in Fedex Charge text box.	C	It is Ok. Database treat this as 23.0		Tara Chand, June 11, 2002	
6		On clicking Preview button after this ERROR shown in figure B.1 was encountered.	B	Corrected			
7		Text box Style which is disabled is not being used anywhere. Remove it.	R	It displays information for a fraction of second and cannot be viewed when making an entry in the form.		Tara Chand, June 11, 2002	

8		Text box Weight which is disabled is not being used anywhere. Remove it.	R	It displays information for a fraction of second and cannot be viewed when making an entry in the form.	Tara Chand, June 11, 2002
9		Total Price is the total of the whole sale price of the inventories in the list. This is currently being rounded. Whole sale prices are 307.5 and 318 where as total is 326. Calculate exact value.	R	Corrected	Tara Chand, June 11, 2002
10		On clicking Print button after this ERROR shown in figure B.2 was encountered.	B	Corrected	Tara Chand, June 11, 2002
11		On clicking Cancel button, the text box below Total Price is not being re-setted	C	Corrected	Tara Chand, June 11, 2002
12	frmITranserList C	Both the dates should be today's date.	R	It is for purpose of displaying few records by default.	Tara Chand, June 11, 2002
13		Exit button in the toolbar should be the right most button	R	Corrected	Tara Chand, June 11, 2002
14		On selecting a row the first column in the selected row you get a yellow colored border. Remove it	R	Since this column is having focus in it that is why it shows a yellow border.	Tara Chand, June 11, 2002

15	FrmIPhysicalInventory03 C	It saves data with arbitrary inventory with zero quantity. What is the use. You should not allow this.	R	It is ok. Because this quantity is 1 for existing inventory and 0 for inventory which is not in the database.	Tara Chand, June 11, 2002
16		On clicking Accept List , it becomes disabled but the save icon in tool bar remains enabled.	C	Corrected	Tara Chand, June 11, 2002
17		The tool tip of save icon in toolbar is wrong. It should be Accept List	R	Corrected	Tara Chand, June 11, 2002
18		I entered one inventory and then clicked Accept List . It disabled the grid. I again entered an inventory using Add to List . It adds the inventory in the grid. But I cannot accept it because this button is disabled.	B	Corrected	Tara Chand, June 11, 2002

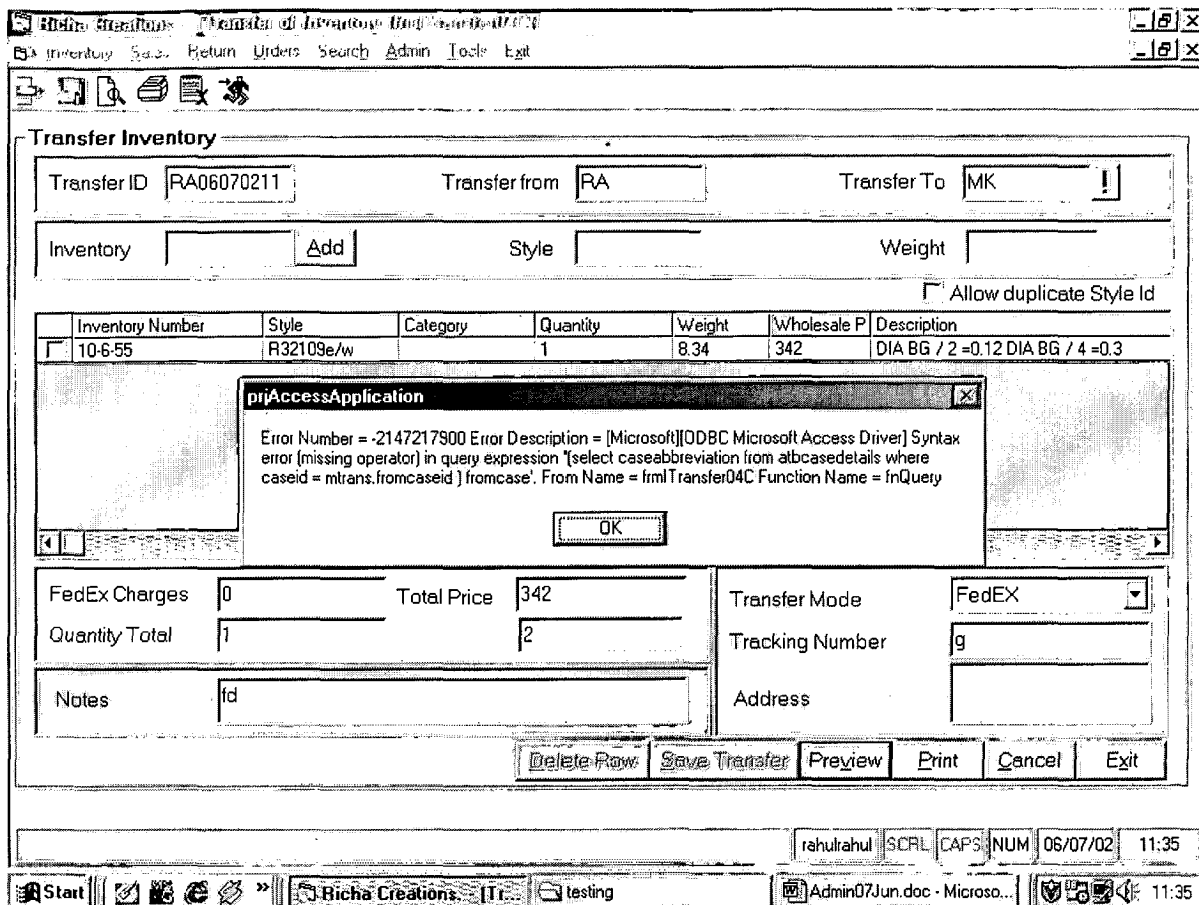


Figure B.1 Screen Shot of defect generated during Testing phase

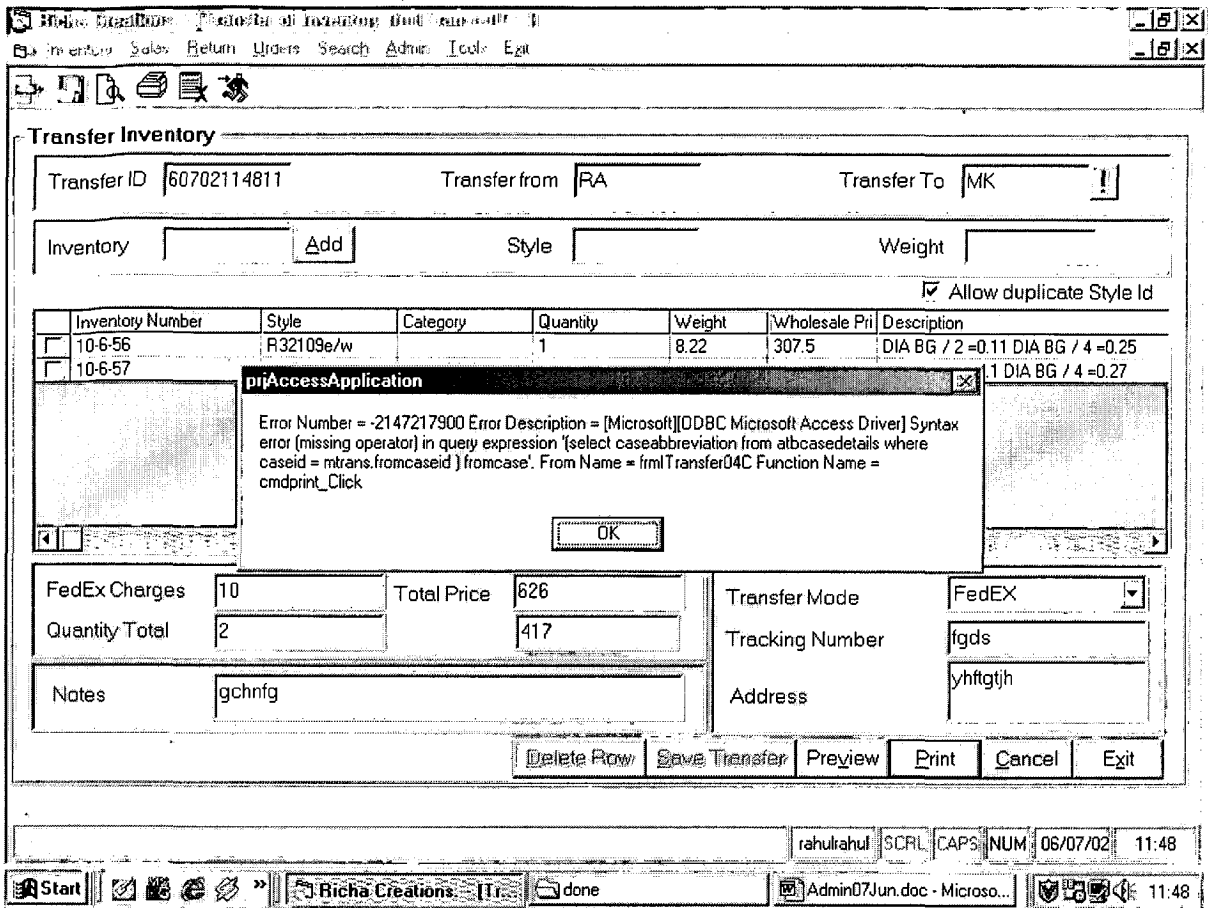


Figure B.1 Screen Shot of defect generated during Testing phase

Appendix C

Screen Shot for Printing BarCode

Alcha Creations. - (Print Tag (fimiPrintTag06C))

Inventory Sales Return Orders Search Reports Data Analysis Admin Customer Care Tools Exit

Inventory Tag Print

REP: HOME Reason: Tag Label: Label 1

Select Price To Print: Retail Price Wholesale Price

Inventory #	Style	Description	Retail Price	Wholesale Price	Vendor	Qty To
1-10-7	R31458	DIA BG 12 0.53 DIA RD 18 0.41	2422.5	484.5	GDPL	
1-10-8	R31458	DIA BG 13 0.52 DIA RD 18 0.41	2400	480	GDPL	
1-10-9	R32150	DIA BG 9 0.77 DIA RD 12 0.321	3720	744	GDPL	
1-13-1	R31088	DIA RD 16 0.48 DIA PRI 6 0.58	4809	961.8	GDPL	
1-13-2	R31088	DIA RD 16 0.48 DIA PRI 6 0.58	4809	961.8	GDPL	
1-18-2	R31624	DIA BG 8 0.27 DIA PRI 20 0.45	3345	669	GDPL	
1-18-3	R31624	DIA BG 8 0.27 DIA PRI 20 0.45	3345	669	GDPL	
1-2-1	RN6684	DIA RD 12 0.41	1605	321	GDPL	
1-2-2	RN6684	DIA RD 12 0.41	1605	321	GDPL	
1-21-11	R32142	DIA PRI 3 0.26	2145	429	GDPL	
1-21-23	R31694	DIA PRI 7 0.49	2752.5	550.5	GDPL	
1-21-24	R31694	DIA PRI 7 0.48	2737.5	547.5	GDPL	
1-21-25	RN7148	DIA PRI 6 0.15	832.5	166.5	GDPL	
1-21-26	RN6629E	DIA PRI 10 0.17	840	168	GDPL	

Add Row Edit Row Delete Row Total 486

Reset List Update Reason Purge Printed Pending Prints Search Print Print All Cancel Exit

SR | SCRL | CAPS | NUM | 20/06/02 | 01:00

Figure C.1 Screen for Printing BarCode

References

1. Evangelos Petroustos, **Mastering Visual Basic 6.0**, BPB Publication
2. Command Reference Manual C.ITOH Printer
3. Richa Creation Product CD, Richa Creation, Colorado
4. Clarity Software, Richa Creation, Colorado
5. SRS Document Of Inventory Sales Management And Analysis System
6. http://www.taltech.com/TALtech_web/resources/intro_to_bc/introbc.htm
7. <http://www.makebarcode.com/barcode.html>
8. http://www.taltech.com/TALtech_web/resources/intro_to_bc/bcsymbol.htm
9. <http://www.msdn.microsoft.com>