

13 JUL 2000



**INFORMATION SYSTEM FOR GOVERNMENT INVENTORY
MANAGEMENT : A CASE STUDY OF
LAND DEVELOPMENT DEPARTMENT**

USA JUGTRIMONGKOL

อภินันท์นาการ

จาก

สีรพัชรทิพย์ ม.มณีภาค

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE
(TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT)**

FACULTY OF GRADUTE STUDIES

MAHIDOL UNIVERSITY

2000

ISBN 974-663-748-7

COPYRIGHT OF MAHIDOL UNIVERSITY

TH
U841
2000
C.2

44739 e.2

Copyright by Mahidol University

Thesis
entitled

**INFORMATION SYSTEM FOR GOVERNMENT INVENTORY
MANAGEMENT : A CASE STUDY OF
LAND DEVELOPMENT DEPARTMENT**

Usa Jugtrimongkol
.....
Miss Usa Jugtrimongkol
Candidate

Thanakorn Uan-On
.....
Thanakorn Uan-On, D.Engr.
Major-advisor

Suttinant Nantachit
.....
Suttinant Nantachit, M.S.
Co-advisor

Piyada Chitchumnong
.....
Piyada Chitchumnong, M.Sc.
Co-advisor

Liangchai Limlomwongse
.....
Prof.Liangchai Limlomwongse,
Ph.D.
Dean
Faculty of Graduate Studies

Thanakorn Uao-On
.....
Thanakorn Uao-On, D.Engr.
Chairman
Master of Science Programme in
Technology of Information System
Management
Faculty of Engineering

Thesis
entitled

**INFORMATION SYSTEM FOR GOVERNMENT INVENTORY
MANAGEMENT : A CASE STUDY OF
LAND DEVELOPMENT DEPARTMENT**

was submitted to the Faculty of Graduate Studies, Mahidol University for the degree of
Master of Science (Technology of Information System Management)

on

March 17, 2000

Usa Jugtrimongkol
.....
Miss Usa Jugtrimongkol
Candidate

Thanakorn Uan-On
.....
Thanakorn Uan-On, D.Engr.
Chairman

Suttinant Nantachit
.....
Suttinant Nantachit, M.S.
Member

Nitima Jitjumng
.....
Nitima Jitjumng, M.A.
Member

Piyada Chitchumnong
.....
Piyada Chitchumnong, M.Sc.
Member

Liyd. Limlomwongse
.....
Prof.Liangchai Limlomwongse,
Ph.D.
Dean
Faculty of Graduate Studies
Mahidol University

Thanakorn Uao-On
.....
Thanakorn Uao-On, D.Engr.
Dean
Faculty of Engineering
Mahidol University

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and deep appreciation to Dr.Thanakorn Uan-on my advisor for his valuable advice and suggestion. I would particularly like to thanks Mr. Suttinant Nantachit and Mrs. Piyada Chitchumnong my committee members for their extremely support and efforts in valuable data of Land Development Department.

I would like to thank Mr. Soopsawat Arkuttananta for his advice in Powersoft Powerbuilder programming and Mr. Sawatchai Sirintawat for improving English document. Thanks are also extended to all my friends for their support and wonderful friendship.

In addition, I am indebted to my family for their love, everlasting supports and enthusiasm throughout the whole life.

Usa Jugtrimongkol

40377717 EGT/M : MAJOR : TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT; M.Sc. (TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT)

KEY WORDS : SYSTEM ANALYSIS / SYSTEM DESIGN
INFORMATION SYSTEM DEVELOPMENT
INVENTORY SYSTEM / CLIENT/SERVER
DATABASE

USA JUGTRIMONGKOL: INFORMATION SYSTEM FOR GOVERNMENT INVENTORY MANAGEMENT: A CASE STUDY OF LAND DEVELOPMENT DEPARTMENT. THESIS ADVISORS: THANAKORN UAN-ON, D. Engr., SUTTINANT NANTTACHIT, M.S. , PIYADA CHITCHUMNONG, M.Sc. 167 p. ISBN 974-663-748-7

Inventory management is an important factor of success or failure in organization. As the amount of inventory increases and as the rate of hardware flow into and out of stock increases, it becomes more and more difficult for a human to remember even approximate inventory balances. An information system is needed to supplement the human mind. This research introduces an information system for government inventory management: a case study of Land Development Department, which reduces access time, increases accuracy and consistent data. The information system was designed and developed using data flow analysis, relational database (Sybase SQL anywhere 5.0 as DBMS) and Object-Oriented programming (Powersoft Powerbuilder Enterprise/32 Version 6.0).

The information system consists of three major processes: 1) Data manipulation: Manipulate relative data; 2) Transaction processing: Support operation management, which are registration, distribution, borrowing, return, repair and disposal; 3) Query and report: Process pre-defined outputs and preplanned printed reports. Since the information system works on client/server network, users are divided into 3 levels: database administrator, operator and general user. Moreover, the information system used in many places, may not support an online system. For this reason, it has an extra function to make the database up to date and consistent.

The information system solves and improves the inefficiency of government inventory management. Furthermore, the information system can develop functions of privileged operation to discard requested documents, applied to Internet/Intranet or OLAP application. In addition the information system can be linked to another relative system such as purchasing, procurement, budget planning and so on for a complete inventory management and control system, which can be applied to DSS or EIS.

4037717 EGT/M : สาขาวิชา : เทคโนโลยีการจัดการระบบสารสนเทศ;

วท.ม.(เทคโนโลยีการจัดการระบบสารสนเทศ)

อุษา จักร์ตรีมงคล : ระบบสารสนเทศเพื่อการบริหารครุภัณฑ์ของส่วนราชการ กรณีศึกษาของกรมพัฒนาที่ดิน (INFORMATION SYSTEM FOR GOVERNMENT INVENTORY MANAGEMENT: A CASE STUDY OF LAND DEVELOPMENT DEPARTMENT).

คณะกรรมการควบคุมวิทยานิพนธ์:ธนากร อ้วนอ่อน,D.Engr., สุทธิพันธ์ นันทจิต M.S., ปิยะดา จิตต์จำนงค์, M.Sc. 167 หน้า. ISBN 974-663-748-7

การบริหารครุภัณฑ์เป็นปัจจัยสำคัญต่อความสำเร็จในองค์กร ระบบสารสนเทศเป็นสิ่งจำเป็นเพื่อช่วยการทำงานแบบเดิม งานวิจัยนี้เสนอระบบสารสนเทศเพื่อการบริหารครุภัณฑ์ของส่วนราชการ กรณีศึกษาของกรมพัฒนาที่ดิน ที่ช่วยลดเวลาการเข้าถึงข้อมูล เพิ่มความถูกต้องและสอดคล้องของข้อมูล ระบบสารสนเทศออกแบบและพัฒนาด้วยวิธีวิเคราะห์กระแสข้อมูล (Data flow analysis), ฐานข้อมูลเชิงสัมพันธ์ (relational database ใช้ Sybase SQL anywhere 5.0 เป็น DBMS) และการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented programming ใช้ Powersoft Powerbuilder Enterprise/32 Version 6.0).

ระบบบริหารครุภัณฑ์ประกอบด้วย 3 ส่วนคือ 1) การจัดการฐานข้อมูล: จัดการข้อมูลพื้นฐานที่เกี่ยวข้องในระบบ 2) การประมวลผลรายการเปลี่ยนแปลง: สนับสนุนงานบริหารครุภัณฑ์ ได้แก่ ลงทะเบียน, เบิก-จ่าย, ยืม, คืน, ช่อม และ จำหน่าย 3) การสอบถามและรายงาน: ประมวลการสอบถามและรายงาน เนื่องจากระบบบริหารครุภัณฑ์นี้ทำงานบนระบบไคลเอนต์เซิร์ฟเวอร์ จึงแบ่งผู้ใช้ออกเป็น 3 ระดับคือ ผู้บริหารฐานข้อมูล, เจ้าหน้าที่ปฏิบัติงาน และ ผู้ใช้ทั่วไป นอกจากนี้ระบบบริหารครุภัณฑ์ใช้ในหลายแห่ง ซึ่งบางแห่งไม่รองรับการทำงานแบบออนไลน์ทำให้มีส่วนเพิ่มขึ้นมาสำหรับปรับปรุงฐานข้อมูลให้ถูกต้องและสอดคล้องกัน ระบบบริหารครุภัณฑ์ช่วยแก้ปัญหาและปรับปรุงการดำเนินงานที่ไม่มีประสิทธิภาพของการบริหารครุภัณฑ์ในส่วนราชการ และสามารถนำไปพัฒนาฟังก์ชันควบคุมสิทธิอำนาจในการปฏิบัติงานเพื่อลดเอกสารคำร้อง หรือประยุกต์เป็นโปรแกรมประยุกต์บนอินเทอร์เน็ตหรืออินทราเน็ต หรือนำไปเชื่อมต่อกับระบบที่เกี่ยวข้อง เช่น การจัดซื้อ, การจัดหา, การวางแผนงบประมาณ เป็นต้น เพื่อให้ได้ระบบควบคุมและจัดการครุภัณฑ์ที่สมบูรณ์ ซึ่งระบบใหม่ที่รวมขึ้นยังสามารถพัฒนาต่อเป็นระบบสนับสนุนการตัดสินใจ(DSS) หรือ ระบบสนับสนุนสารสนเทศสำหรับผู้บริหาร(EIS)

CONTENTS

	Page
ACKNOWLEDGEMENT	iii
ABSTRACT (ENGLISH)	iv
ABSTRACT (THAI)	v
CONTENTS	vi
LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTER I INTRODUCTION	1
Background and Statement of Problems	1
Objectives	3
Scope of Work	4
Expected Results	5
CHAPTER II LITERATURE SERVAY	6
Material Management and Inventory Control	6
Material Classification	10
Management Information Systems	12
The Client/Server Model	14
Relational Data Model	18
Data Flow Analysis	21
Testing	23
Software Development Tools	24
Related Research	27

CONTENTS (Continued)

	Page
CHAPTER III MATERIALS AND METHODS	28
Materials	28
Methods	29
CHAPTER IV RESULT	33
Current System	33
Results from System Analysis and Design	36
Data Flow Diagram	36
Data Dictionary	44
Input and Output Design	48
Database Design	52
Procedure Design	58
Information System for Government Inventory Management	61
CHAPTER V DISCUSSION	64
CHAPTER VI CONCLUSION	66
REFERENCES	69
APPENDIX A EXISTING REPORT	72
APPENDIX B DESIGNED REPORT	82
APPENDIX C DESIGNED SCREEN	97
APPENDIX D PROGRAM SPECIFICATION	113
BIOGRAPHY	167

LIST OF TABLES

	Page
Table 4.1 Data dictionary for data structure	48
Table 4.2 Characteristic of process	59
Table 4.3 Ancestor window objects	60
Table 4.4 Shared window object	61
Table 4.5 Capability of process	61
Table 4.6 Privilege of user	63

LIST OF FIGURES

	Page
Figure 2.1 Government inventory management function	10
Figure 2.2 FSN material classification	11
Figure 2.3 The three levels of management	12
Figure 2.4 MIS subsystems	13
Figure 2.5 Client/server computing	15
Figure 2.6 The Client/server database model	16
Figure 2.7 Relation data model	18
Figure 2.8 Table in 1NF	19
Figure 2.9 Table in 2NF	20
Figure 2.10 Table in 3NF	21
Figure 2.11 Data Flow Diagram using Yourdon notation	22
Figure 2.12 ODBC	25
Figure 3.1 Steps and research methodology	32
Figure 4.1 Graphic presentation of current storekeeping process	33
Figure 4.2 Graphic presentation of current distribution process	34
Figure 4.3 Graphic presentation of current borrow process	34
Figure 4.4 Graphic presentation of current repair process	35
Figure 4.5 Graphic presentation of current annual audit process	35
Figure 4.6 Graphic presentation of current disposal process	36
Figure 4.7 Context diagram of government inventory management system	37

LIST OF FIGURES (Continued)

	Page
Figure 4.8 DFD Level 1: Government inventory management system	37
Figure 4.9 DFD Level 2: 1.Data manipulation	38
Figure 4.10 DFD Level 3: Data manipulation	39
Figure 4.11 DFD Level 2: 2.Transaction process	40
Figure 4.12 DFD Level 3: 2.1 Register hardware	41
Figure 4.13 DFD Level 3: 2.2 Distribute hardware	41
Figure 4.14 DFD Level 3: Transaction process	42
Figure 4.15 DFD Level 2: 3. Query process	43
Figure 4.16 DFD Level 2: 4. Report generation	43
Figure 4.17 First level process descriptions	44
Figure 4.18 Third level process descriptions	45
Figure 4.19 Third level process descriptions (continued)	46
Figure 4.20 Logic summary of procedure	47
Figure 4.21 Data dictionary entries for data flows	49
Figure 4.22 Data dictionary entries for data flows (continued)	50
Figure 4.23 Data dictionary entries for data flows (continued)	51
Figure 4.24 Data structure diagram	53
Figure 4.25 Structure chart	59
Figure C.1 w_db_hw_group	98
Figure C.2 w_db_hw_class	98

LIST OF FIGURES (Continued)

	Page
Figure C.3 w_db_hw_type	99
Figure C.4 w_db_hw	99
Figure C.5 w_db_supplier	100
Figure C.6 w_db_division	100
Figure C.7 w_db_section	101
Figure C.8 w_db_staff	101
Figure C.9 w_db_security	102
Figure C.10 w_db_project	102
Figure C.11 w_trans_control	103
Figure C.12 w_regi_add	103
Figure C.13 w_trans_dist	104
Figure C.14 w_trans_retu	104
Figure C.15 w_trans_repa	105
Figure C.16 w_trans_disp	105
Figure C.17 w_check_repa	106
Figure C.18 w_trans_sql	106
Figure C.19 w_q_hw_spec_show	107
Figure C.20 w_q_hw_status	107
Figure C.21 w_q_regi_desc	108
Figure C.22 w_g_sum_quantity	108

LIST OF FIGURES (Continued)

	Page
Figure C.23 w_g_drilldown	109
Figure C.24 w_graph_spacing	109
Figure C.25 w_graph_type	110
Figure C.26 w_preview	110
Figure C.27 w_set_arg	111
Figure C.28 w_pb_calendar	111
Figure C.29 w_hw_code_search	111
Figure C.30 w_hw_id_search	112
Figure C.31 w_login	112
Figure C.32 w_about	112

CHAPTER I

INTRODUCTION

1.1 Background and Statement of Problems

Inventory management is an important factor of success or failure in organization. Inventory management involves almost all section in organization and is responsible for controlling and auditing inventory. The main functions of inventory management are registration, distribution, borrowing, repair, disposal, annual audit, and reporting. As the amount of inventory increases and as the rate of hardware flow into and out of stock increases, it becomes more and more difficult for a human to remember even approximate inventory balances. Some form of record keeping is needed to supplement the human mind, Regardless of whether the records are operated by information system. The problems of inefficient operation are as follows:

1. Nonstandard data store.

Each section differently keeps data of hardware; consequently hardware management cannot retrieve entire department data.

2. Incorrect and inconsistent data.

Manipulation data is analog operation; officer must manipulate each file by himself so rate of incorrect and inconsistent data is higher than electronic operation. Updating data is the biggest factor of inconsistent data because officer may forget to update some files.

3. **Difficult and slow data searching.**

In case of a lot of data; officer takes his time with manual searching and may not discover data in time.

4. **Difficult data checking.**

Checking data of hardware (such as status, location), officer must use hardware number (or registration number) to check the data. If he does not know the number, this operation will spend a lot of time. Comparison with information system, he can use drill-down technique or entity relationship to discover the hardware material.

5. **Incorrect result.**

There are many queries in inventory control, some queries use arithmetic, searching and checking to calculate and summarize the result. So, it is possible to get incorrect result.

6. **Inconsistent report.**

It's the same query process, the process may produce inconsistent and incorrect reports.

7. **Difficult operation.**

Inventory control operation has many constraints and steps. Studying these steps is hard for new officer.

The problems of inefficiency operation can be solved and improved by information system. The information system changes operation from analog to electronics, search from manual to digital and database from analog to digital. The

information system introduces online operation, which reduces access time, increases accuracy and consistent data.

The Land Development Department was established on 23 May 1963 within the Ministry of National Development. Some years after their establishment, the government agencies were restructured, and as from 29 September 1972 the Land Development Department was transferred to the Ministry of Agriculture and Cooperatives. Land Development Department is responsible for:

- Soil survey, classification, and analysis and land use planning.
- Conduct experiments and carry out research on various aspects of land development.
- Assist farmers in soil and water conservation and soil improvement, the production of seeds for cover crops and the production of soil improvement materials.
- Transfer knowledge of soil development and soil science for multiple purpose use (1).

From its responsibility, the Land Development Department has a lot of hardware's data. At present, an inventory management operation of Land Development Department has not information system to manage these data.

1.2 Objectives

To analyze and design information system for government inventory management system.

To create database and implementing information system for government inventory management for a case study of Land Development Department.

1.3 Scope of Work

The information system for government inventory management of Land Development Department is developed by Powersoft PowerBuilder Enterprise/32 version 6.0. The information system works on client/server network, which supports operation management and has functions as follows:

- **Hardware registration:** It records data of receipt hardware for example hardware description, procurement method.
- **Hardware distribution:** This is a distribution control, updating status and hardware used place.
- **Hardware borrowing:** It's the same as distribution but it has return date data.
- **Hardware repair:** It records data of repair.
- **Hardware disposal:** After unused hardware is disposed, officer will use this function to update data of disposed hardware from hardware registration file.
- **Inquiry and Report:** The outputs of this function are report and result for operational manger (head of material section) such as status and used place of hardware, quantity of hardware, hardware detail.

This information system has central relational database model that is created by Sybase SQL anywhere 5.0. The database consists of 3 parts as follows:

- Historical record of standard hardware detail such as type, characteristic.
- Historical record of hardware purchasing has detail about receptive date, supplier and purchasing detail.
- Historical records of hardware operation (e.g. distribute, borrow, and repair) keeps data that can tell us where is the hardware.

1.4 Expected Results

The work should results information system for government inventory management, which consists of

1. Hardware registration module;
2. Hardware distribution module;
3. Hardware borrowing module;
4. Hardware repair module;
5. Hardware disposal module; and
6. Inquiry and report module.

Which have the capability of tracking hardware management operation and improving inefficient operation.

CHAPTER II

LITERATURE SURVEY

In this chapter, “Material” and “Hardware” are the same meaning.

2.1 Material Management and Inventory Control

Materials management is a coordinate function responsible for planning and controlling materials flow. Its objectives are as follows(2):

- Maximize the use of the firm’s resources.
- Provide the required level of customer service.

Inventory management is responsible for planning and controlling inventory from the raw material stage to the customer. Inventory must be considered at each of the planning levels and is thus part of production planning, master production scheduling, and material requirements planning(2).

The term *Inventory* can be used to mean several different things, such as (3);

1. The stock on hand of materials at a given time (a tangible asset which can be seen, measured, and counted);
2. an itemized list of all physical assets;
3. (as a verb) to determine the quantity of items on hand;
4. (for financial and accounting records) the value of the stock of goods owned by an organization at particular time.

2.1.1 Material Management Functions and Responsibility

1. Procurement and Purchasing

Purchasing is the “process of buying”, Purchasing is solely the responsibility of the purchasing department. Obtaining the right material, in the right quantities, with the right delivery (time and place), from the right resource, and at the right price are all purchasing functions(2).

2. Receiving

Responsibilities of material receiving are checking quantity and quality of receipt material, which justify purchasing order, storage area selection and providing tools and people for transfer. The main functions are (4).

After Materials have been received, receiving report will generate. The report has an important supplier data for example, transport date, decayed materials, percentage of returned materials, and so on. In addition, this report is useful for summarizing purchase order and negotiating of supply agreement in future(5,6).

3. Supervision and protection

This section has to plan appropriate storage location and method, provide security, protect materials from damage and lost. These tasks are convenient for easy and quick material identification (4).

Closed stores system stores materials in closed or controlled area, which does not allow anyone to go into there except officer. Materials are distributed and kept when there are requisition or receipt report. The objective of this system is best material safety. It has strict accounting control, furthermore receiving record and inventory identification are important activities (5,6).

4. Distribution

Main responsibilities of distribution are authorization verification; distributing materials follow requisition, checking characteristic of distributed materials (4).

5. Inventory control

It sets inventory control system, which suits for each type of materials and controls material quantity (4).

Since recording and keeping are separated, sometimes the quantity of on hand material may not equal to the quantity of material balance on report. So, it should check the quantity of material in period, which has 3 methods as follows; Fixed Annual Inventory, Continuous Inventory and Low-Point Inventory (5,6).

6. Record and report.

It has to record of receipt, distribution, control and quantity on hand of materials, update data, collect purchasing data and statistical data, create report such as receipt report, annual reports and stock checking(4).

7. Unused Disposal

This section has to find unused materials, report list of unused materials to every section, distinguish unused materials from stock and propose effective unused disposal method (4).

8. Transportation

Transportation has to prepare and pack materials, identify transportation method and send them to destination on time. The destination must get good material in the right quantity (4).

9. Cooperation to other departments (4)

2.1.2 Material Management in the Government

Material Management in the government need to have predefine project plan.

Planning of material need historical data from one of these:

1. Statistical of material withdrawing ; This data collect form “requisition”.
2. Statistical of material distributing ; The quantity distributed may not equal to the quantity material withdrawn.
3. Statistical of material use of sections; It is calculated by subtracting the quantity on hand from the receipt quantity (4).

Definition of hardware

Hardware expense is expense of purchasing or exchange thing, which is permanent and long life. It includes expense from modification and direct abroad purchasing. Hardware is classified into 2 types as follows :

1. Limited life hardware is anything that is permanent but it is not forever such as a car with a life of 10 years, an air-condition with a life of 5 years.
2. Unlimited life hardware is anything that is permanent use such as iron cabinet, table, chair (4).

Government inventory management

Government inventory management has 4 main functions; control, borrowing, repair and disposal. This shows in Figure 2.1.

Storekeeping has receiving, hardware registration. When distribution occurs, officer has to update status and used place. Borrowing and repair are the same distribution. These are routine function. Before the end of September, head of material inventory section will appoint committee to audit past operation from October 1st to

September 30th. After annual audit has finished, officer will create report of unused hardware to dispose. Unused hardware is sold, exchanged, given to other sections or destroyed. If unused hardware cannot dispose by these methods, officer will dispose the hardware as lose hardware. Then officer deletes data of disposed hardware from hardware registration (5,7,8).

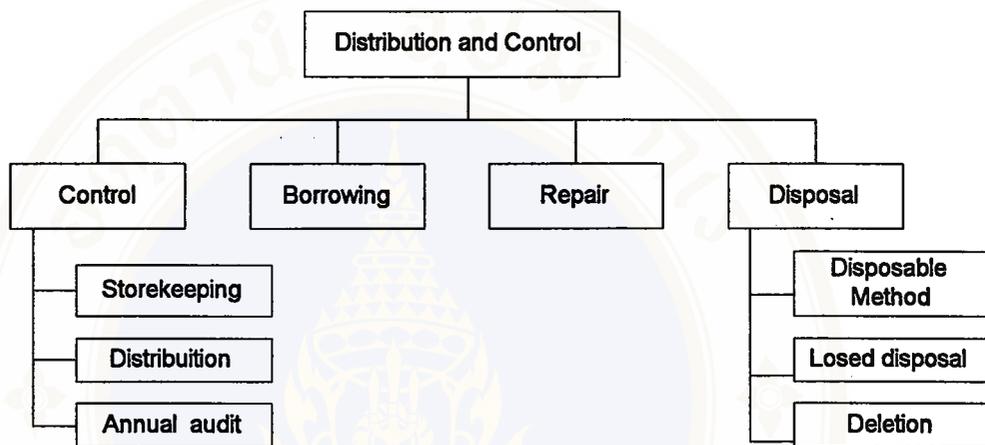


Figure 2.1 Government inventory management function

2.2 Material Classification

Catalogue of product is a tools for material classification because it has exactly cost and specification detail ,which help in classify. Hence, some suppliers precisely classify component or material number. However organization purchase material from many suppliers, there are problem in confusing or too many number. So , development of material classification is thing, that should do and supplier's material number should not be thrown away but use it for reference (5,6).

2.2.1 Material Classification Principle

1. Classification system bases on pre-defined plans of organization, which is stable and standard concept.

2. The system concludes all lists of material classification and has space for materials in the future.
3. Each of material lists must have especially characteristic, which differ from another. User can find material from main class to sub class and finally discover the material.
4. It is simple system, general officers and non-expert can use it. It is convenient for internal communication.
5. It has a complicate order such as order by size (big size to small size), order by cost(cheap cost to expensive cost) (4).

2.2.2 FSN Material Classification

Material Classification of Thailand uses FSN system (Federal Stock Number).

This system has 11 digits as following:

1. First part has 4 digits; it means group class.
2. Second part has 3 digits; it means type.
3. Third part has 4 digits; it means description.

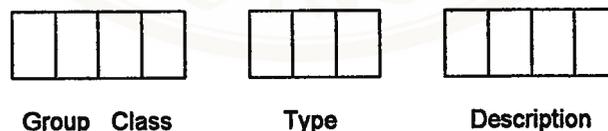


Figure 2.2 FSN material classification

Example: 7110-002-0001 is a chest of 4 drawers

1. 7110: '71' is group of furniture and '10' is class of office decoration.
2. '002' is type of chest of drawers.
3. '0001' is description or specification of material. The third part has not certain number but it should order by material's specification. The numbers are unique and

start from 0001 to 9999. For example 7110-002-0001 is a chest of 4 drawers, 7110-002-0002 is a chest of 5 drawers (9).

2.3 Management Information Systems

Management positions in a firm are often broken into three levels: upper management (the president and vice-presidents), middle management (the people responsible for anything between upper and lower management), and lower management (the people directly responsible for managing those who produce the firm's outputs). The roles of these levels of management are summarized in the pyramid in Figure 2.3



Figure 2.3 The three levels of management

Management information system (MIS) is any system that provides people with either data or information relating to an organization's operation. The systems provide routine information to managers and decision-maker. The focus of a MIS is on operation efficiency (10,11).

The MIS effort in an organization as composed of the following four subsystems. (See Figure 2.4) Knowledge-based systems such as artificial intelligence (AI) and expert systems (ES) are also information system, but are not depicted as separated as subsystems because they may be used in conjunction with a TPS, MRS, DSS and OIS.

- *Transaction processing systems (TPS)* comprise the routine, day-to-day accounting operations that have been an important part of most firms' computer processing. Transaction processing cycle consists of five major segments: data entry, transaction processing, file/database updating, report/document generation, and inquiry/response processing.

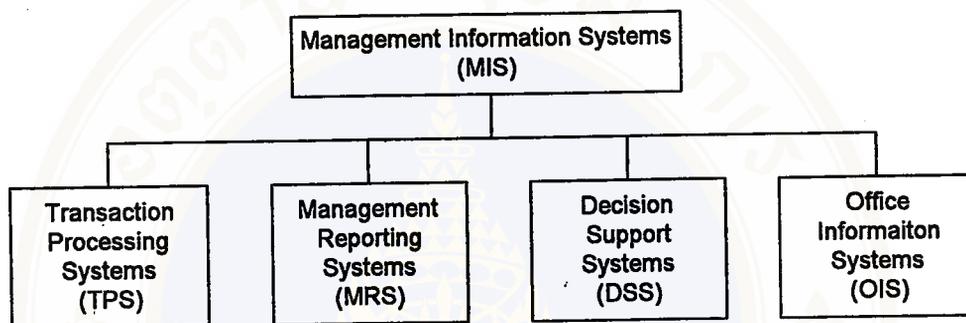


Figure 2.4 MIS subsystems

- *Management reporting systems (MRS)* generates the preplanned printed reports for decision-making purposes. MRP are used for both management planning and management control functions. Properties of management reporting systems:
 - Support structure and semi-structured decisions, primarily at the middle- and lower- management levels.
 - Provide fixed types of information, in an established format; the information requirements of users are normally known and stable
 - Often implemented with voluminous, hardcopy reports, requiring each user to search specifically for key information.
 - Usually require a formal request to be submitted; formal systems development may be required to approve
 - Often require a formal run schedule

- Usually consist of internal operational data, rather than data about the external environment
- Usually more concerned with data about the past than relating to the future
- Often oriented to summary and exception reporting.
- *Decision support systems (DSS)* provide a set of easy-to-use modeling, retrieving, and reporting capabilities so that people can generate the information they feel will be useful to them when making decisions.
- *Office information systems (OIS)* include the use of such computer-based, office-oriented technologies as word processing, desktop publishing, electronic-mail, video teleconferencing, and so on (10).

Form these definitions, The Information System for Government Inventory System combines between TPS and MRS.

2.4 The Client/Server Model

The client/server model is based on the concept that each application consists of two functional parts: one that initiates communication and another that responds to it. The server waits for incoming communication requests from a client, performs the requested actions for the client, and returns the result to the client. The client then retrieves data from the server (12).

The fundamental client/server application structure consists of three physical components:

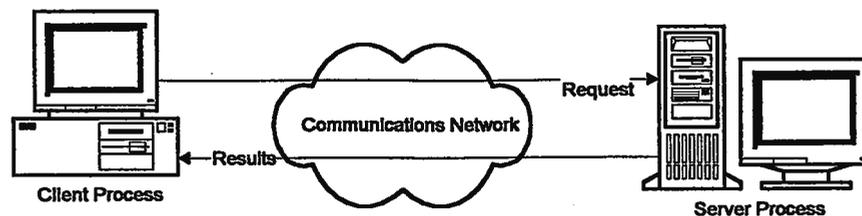


Figure 2.5 Client/server computing Client/server computing

- *Client Machine.* This is the computing system that runs the client component of the application . It makes one or more requests for services.
- *Server Machine.* This is the computing system that runs the server component of the application. It satisfies a client's request for a service and returns results to the client component.
- *Communication Network.* These are the communication facilities that allow one or more message making up a request to be passed from client to the server and that allow one or more messages containing results to be passed from the server back to the client.(13)

2.4.1 Client/server Database Model

In the true client/server database model, shown in Figure 2.6, the database again resides on a machine other than those that run the application processing components. But the database software is split between the client system that runs the application program and the server system that stores the database.

In this model, the application processing components on the client system make requests of the local database software. The local database software component in the client machine then communicates with the complementary database software running

on the server. The server database software makes requests for accesses to the database and passes results back to the client machine.

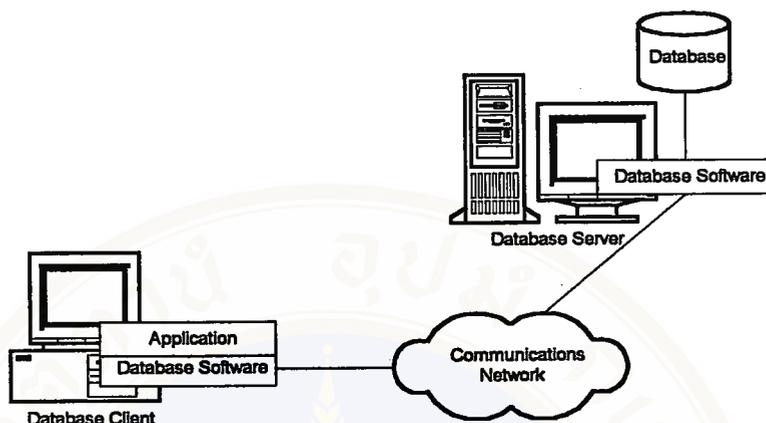


Figure 2.6 The Client/server database model

In the client/server database model, it is common to refer front-end software and back-end software. *Front-end software* typically runs on a personal computer or desktop workstation and serves the computing needs of a single individual. The front-end software typically plays the role of the client in a client/server database application and performs functions that are oriented to the needs of the end user.

Back-end software consists of the client/server database software and network software that runs on a machine playing the role of database server (13).

2.4.2 Distributed Database

Distributed database – a database in which the actual data may be spread across several smaller databases connected via telecommunications devices. Distributed databases give corporations more flexibility in how databases are organized and used.

Distributed databases allow more users direct access at different user site. To reduce the demand on telecommunications lines, some organizations will build a replicate database. A replicated database is a database that holds a duplicate set of

frequently used data. At the beginning of the data, a company will send a copy of important data to each distributed processing location. At the end of the day, the different sites will send the changed data back to be stored in the main database (10,11).

Database Extraction and Replication

With extraction and replication, certain data element values from the database are stored in multiple location, possibly to make the database more available and resistant to failures or to enhance performance.

- *Extraction.* A database copy is called an *extract* when the copy is intended to be used on a read-only basis. Data element values in a extract are not intended to be updated.
- *Replication.* A database copy is called a *replica* when data element values in the copy can be updated (13).

Lock Types

The two types of locks that are the most commonly implemented are:

- *Shared-Access Locks.* A shared-access lock owner to read the locked data but not to change it. A user can acquire a shared-access lock on data after one or more other users have also acquired a shared access lock on it.
- *Exclusive-Access Locks.* An exclusive-access lock allows the lock owner to read and change the locked data in any desired way. No other user can access the locked data or acquire a lock on it while an exclusive-access lock is in effect, On the other hand, no user can acquire an exclusive-access lock on data that already has a shared-access lock on it (13).

2.5 Relational Data Model

Relational databases have been extensively used for well over a decade. The relational data model is based on a relation, a two-dimensional table. Each row of a table, called a tuple represents a record or collection of related facts. The columns of the table are called attribute. Each attribute can take on only certain values. Creation of these tables and their columns is done using SQL. Similarly storage and retrieval of data is also done using SQL (Structured Query Language). (14,11) As such SQL is one of the primary factors in the explosion of client/server development activity (15).

Dept. number	Dept. name	Manager SSN
257	Accounting	421-55-9993
632	Manufacturing	765-00-3192
598	Marketing	098-40-1370

Figure2.7 Relational data model

2.5.1 Normalization

Normalization is a process of simplifying the relationship between data elements in a record. Through normalization a collection of data in a record structure is replaced by successive record structures that are simpler and more predictable and therefore more manageable. Normalization is carried out of four reasons:

- To structure the data so that any pertinent relationships between entities can be represented.
- To permit simple retrieval of data in response to query and report requests.
- To simplify the maintenance of the data through updates, insertions, and deletions
- To reduce the need to restructure or reorganize data when new application requirements arise (14).

First normal form:

A relational table, by definition, is in first normal form. All values of the columns are atomic. That is, they contain no repeating values. Figure 2.8 shows the table FIRST in 1NF.

FIRST

s#	status	city	p#	qty
s1	20	London	p1	300
s1	20	London	p2	200
s1	20	London	p3	400
s1	20	London	p4	200
s1	20	London	p5	100
s1	20	London	p6	100
s2	10	Paris	p1	300
s2	10	Paris	p2	400
s3	10	Paris	p2	200
s4	20	London	p2	200
s4	20	London	p4	300
s4	20	London	p5	400

Figure 2.8 Table in 1NF

Although the table FIRST is in 1NF it contains redundant data. For example, information about the supplier’s location and the location’s status has to be repeated for every part supplied. Redundancy causes what are called update

Second normal form:

The definition of second normal form states that only tables with composite primary keys can be in 1NF but not in 2NF.

A relational table is in second normal form 2NF if it is in 1NF and every non-key column is fully dependent upon the primary key. That is, every non-key column must be dependent upon the entire primary key.

The process for transforming a 1NF table to 2NF is:

1. Identify any determinants, other the composite key, and the columns they determine.

2. Create and name a new table for each determinant and the unique columns it determines.
3. Move the determined columns from the original table to the new table. The determinate becomes the primary key of the new table.
4. Delete the columns you just moved from the original table except for the determinate, which will serve as a foreign key.
5. The original table may be renamed to maintain semantic meaning.

SECOND			PARTS		
s#	status	city	s#	p#	qty
s1	20	London	s1	p1	300
s2	10	Paris	s1	p2	200
s3	10	Paris	s1	p3	400
s4	20	London	s1	p4	200
s5	30	Athens	s1	p5	100
			s1	p6	100
			s2	p1	300
			s2	p2	400
			s3	p2	200
			s4	p2	200
			s4	p4	300
			s4	p5	400

Figure 2.9 Table in 2NF

Third normal form

The third normal form requires that all columns in a relational table are dependent only upon the primary key. A more formal definition is:

A relational table is in third normal form (3NF) if it is already in 2NF and every non-key column is non-transitively dependent upon its primary key. In other words, all non-key attributes are functionally dependent only upon the primary key.

The process of transforming a table into 3NF is:

1. Identify any determinants, other the primary key, and the columns they determine.

2. Create and name a new table for each determinant and the unique columns it determines.
3. Move the determined columns from the original table to the new table. The determinate becomes the primary key of the new table.
4. Delete the columns you just moved from the original table except for the determinate, which will serve as a foreign key.
5. The original table may be renamed to maintain semantic meaning (16).

SUPPLIER_CITY	
s#	city
s1	London
s2	Paris
s3	Paris
s4	London
s5	Athens

CITY_STATUS	
city	status
London	20
Paris	10
Athens	30
Rome	50

Figure 2.10 Table in 3NF

2.6 Data Flow Analysis

Data flow analysis examines the use of data to carry out specific business processes within the scope of a systems investigation. The components of data flow strategy span both requirements determination and system design. A prescribed notation facilitates the documentation of the current system and its analysis by all participants in the process of determining system requirement

Tools of Data Flow strategy.

1. Data Flow Diagram.

The graphic tool used to describe and analyze the movement of data through a system – manual or automated – including the processes, stores of data, and delays in the system.

2. Data Dictionary.

The logical characteristics of current systems data stores, including name, description, aliases, contents, and organization. Identifies processes where the data used and where immediate access to information is needed. Serves as the basic for identifying database requirements during system design.

3. Data Structure Diagram.

A pictorial description of the relation between entities (people, places, events, and things) in a system and the set of information about the entity. Does not deal with physical data store.

4. Structure Chart.

A design tool that pictorially shows the relation between processing modules in computer software. Describes the hierarchy of component modules and the data that are transmitted between them. Includes analysis of input-to-output transformations and analysis of transactions.

The first two are developed during requirements determination while the later two most useful during systems design (14).

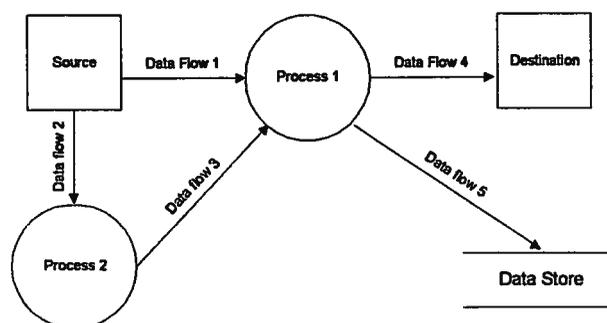


Figure 2.11 Data Flow Diagram using Yourdon notation

2.7 Testing

Back-box testing attempts to find errors in the following categories: 1) incorrect or missing functions, 2) interface errors, 3) errors in data structures or external database access, 4) performance errors, and 5) initialization and termination errors.

Back-box testing are designed to uncover errors in functional requirements without regard to the internal working of program.

Boundary value analysis (BVA) has been developed as a technique Boundary value analysis leads to a selection of test cases that exercise bounding values.

Guidelines for BVA are as following

1. If an input condition specifies a range bounded by value a and b, test cases should be designed with values a and b, just above and just below a and b respectively.
2. If an input condition specifies a number of values, test cases should be developed that exercise the minimum and maximum numbers. Values just above and below minimum and maximum are also tested.
3. Guidelines 1 and 2 are applied to output conditions. For example, assume that a temperature vs. pressure table is required as output from an engineering analysis program. Test cased should be designed to create and output report that produces the maximum (and minimum) allowable number of table entries.
4. If internal program data structures have prescribed boundaries (e.g., an array has a defined limit of 100 entries), be certain to design a test case to exercise the data structure at its boundary (17).

2.8 Software Development Tools

The 4GLs provide users with the ability to develop GUI applications quickly and painlessly. To be productive, an application programmer must be able to build GUI applications in a very short amount of time, even if some flexibility and efficiency must be given up. Powersoft's Powerbuilder has become a very popular database front end building tool. It has developed a reputation for ease of use and robustness (12).

PowerBuilder's Technical Strengths

DataWindow: It is the primary means by which a PowerBuilder application talks to the database. It has built-in features to format data for display, allow different edit-styles, validate data entered by user, generate appropriate SQL based on the changes made by a user and also the RDBMS it is talking to and scores of other such invaluable features.

Object-Oriented (OO): PowerBuilder is an object-oriented language. Though it is not a pure object-oriented language, it supports inheritance in most of the areas, permits encapsulation and enables polymorphism. Because of these reasons, it is possible to architect your applications in such a way as to reuse code within and across applications. If you make use of OO features, it also makes it simpler to maintain that application.

Native Drivers: Though ODBC (Open Database Connectivity) is good for accessing multiple databases through a common gateway, it covers only the common minimum features of these databases. PowerBuilder provides native drivers for all the major RDBMSs, such as Oracle, Sybase, Informix, DB2, MS SQLServer, etc., so that you can take advantage of the power of these.

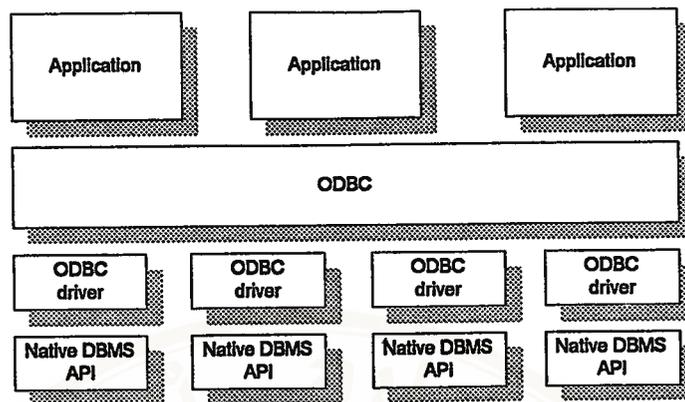


Figure 2.12 ODBC

Cross Platform: You can write code once and run that application on all the flavors of Windows, namely Windows 3.1, Windows for workgroups, Win95 and Windows NT. You can also use the same code to run the application on Mac and Sun Solaris UNIX.

Web-enabled: With PowerBuilder 5.0 and the Internet add-ons, you can build an application which can access data in an RDBMS through a browser, whether it is on the corporate intranet or on the internet (12,18).

Sybase

Sybase is an increasingly popular database. The databases can execute miscellaneous transaction processing logic. Data consistency and integrity checks don't have to be coded into each different client application. Instead it can be enforced centrally in the server. The feature of Sybase is follow:

- Run on a wide range of computer architectures such as Sun's Solaris, DEC's VMS, IBM's AIX, HP's HPUX, IBM's OS2, Novell's NLM and Windows NT.
- Support variety of networking protocols. These networking protocols such as TCP/IP, Decnet, Novell IPX, and LAN Manager.

- Programs supplied by independent software vendors, or user written programs.
- Support Client/Server Architecture.
- Other Sybase produces called gateways to give clients and servers on line access to non-Sybase data.
- Transact-SQL is Sybase's extension to SQL. Transact-SQL permits procedural processing with control statements.
- Sybase servers also include the ability to communicate directly between each other without going through a client. The inter-server communication is implemented as an easy to write procedure call. RPC's can be called in stored procedure or triggers so that the client program doesn't have to know or care about other sources of data (19,20).

Windows NT

Windows NT optimizations include:

- Implementation of virtual processors and client requests as Windows/NT threads to take advantage of DSA's inherent multithreading capability;
- The effective handling of disk I/O and network I/O through the use of the Win32 API.
- Using Microsoft's advanced operating system technologies, such as preemptive multitasking, to optimize resource utilization.
- Implementation as a Windows NT Service for integrated control with the operating system.
- Integration with, and full support for, the Windows 95/NT Registry (19).

2.9 Related Research

Wijada Rattamanee (8) analyzed and designed inventory control system of KMITL, in particular controlling and auditing models. The design was intended to be easy and speed up the operation procedure. The system was analyzed and designed by data flow diagram using Gane and Sarson notation. The new systems included database design, prototype design and report design.

Nathavipa Phasuk, Voramom Uronkarn (21) did computer program that was able to collect equipment data, academic equipment and office equipment in Industrial Engineering Department factory of Engineering Mahidol University. The program was designed and developed using Microsoft Access version 2.0, which can run on Windows 3.11 and Windows 95 platform. It can be added data, edited data, listing and reported in formal forms.

Wilert Lertbudikul (22) designed and developed medical supply inventory information system for a large hospital. The research followed these steps; current system feasibility study, system analysis, system design and system development. The system was designed and developed into 2 types, which were online interactive using CICS/VS communication control system and batch using OS/VS1 operating system. The system focused on high data quantity, high-speed data searching, collecting and summarizing the important information for management. The system consisted of 4 subsystems; initial and security subsystem, basic database recording subsystem, daily transaction subsystem and query and statistical subsystem. This developed system was suitable for a large hospital.

CHAPTER III

MATERIALS AND METHODS

3.1 Materials

- **Hardware**

- **Computer (Minimum)**

Server: Intel Pentium 166 MHz

RAM 32 MB

Hard disk 1.2 GB

- **Client: Intel 486DX.**

RAM 16 MB

Hard disk 1.2 GB

- **Communication equipment: LAN card, Wire, Hub**

- **Printer.**

- **Software**

Database : Sybase SQL anywhere 5.0

Application development tools: Powersoft PowerBuilder Enterprise/32 Ver. 6.0

Operating System : Windows NT 4.0, Window95

Help authoring tools : Oasis SE Version 1.3.6.0

3.2 Methods

1. Study operation of inventory management system

1.1. Review data of government inventory management system from books and researches such as steps of operation, function and material classification system.

1.2. Identify problems of inventory management system.

1.3. Scope problem solving by information technology.

2. Collect data and identify system requirement of Land Development Department

2.1. Review existing system documents. General document should be reviewed first. General documentation includes management overviews, description of the organization's structure, and functional task. List of the existing system documents are as follows:

- Organization chart
- Registration document
- Registration document of each hardware
- Summary of hardware quantity in department
- Summary of hardware quantity in requested budget year
- Summary of hardware registration
- Summary of hardware quantity
- Summary of hardware quantity in requested budget year and section

2.2. Observe in operating environment, it can indicate how individuals participate in the operation of system. Interview user and related officer, whom tell requirement and operation problem.

3. Analyze and design system

Use data flow analysis to analyze the system. The components of data flow strategy span both requirement determination and system design. Following the flow of data through processes tells how data are input, processed, stored, retrieved, used, changed and output.

3.1 Design of input and output

Design input/output screens. These screens are user-friendly with GUI (graphic user interface). The input screens must have input controls, which provides way to ensure that 1) only authorize users can access the system, 2) guarantee the transactions are acceptable, 3) the data are validated for accuracy, and 4) There is the determination whether any necessary data have been omitted.

Design output reports, which support the activity of the operation and management. The outputs are generated in tubular form.

3.2 Design of procedure

Distinguish between manual and computer processes. The computer processes are to identify constraints such as authorization, volume, process timing, input method, output method and so on.

Design procedure and process specification. Formal statements using techniques and languages that enable analysts to describe important activities that make up the system

3.3 Design of database

Identify data used and information produced.

Select and identify data from collected data and system requirement, it gets related entities, attributes and constraints.

Identify relationships between those entities and design database with relational database concept.

4. Develop system

4.1. Create database. Transform designed data structure diagram to database.

4.2. Develop application to connect database and support operations. This step has unit testing.

4.3. Create help file.

5. Test software

Verify and validate software.

Check if the software conforms to its specification and if the software meets the expectation of the software requirement by black-box testing. Uncover errors and ensure that the software performs properly. Test cases are designed by boundary value analysis method.

6. Summarize and create document

This document includes detail, methodology, conclusion and recommendation of the research.

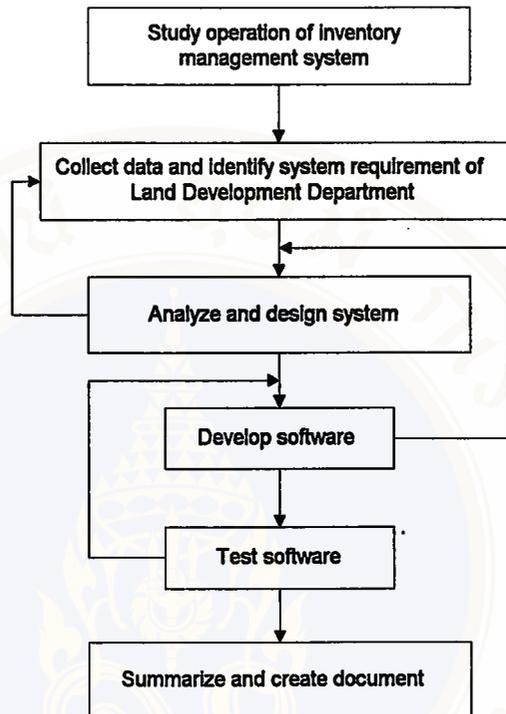


Figure 3.1 Steps and research methodology

CHAPTER IV

RESULTS



The result of study for this research is the information system for government inventory management: a case study of Land Development Department. The information system is designed and developed using data flow analysis, relational database and Object-Oriented programming.

4.1 Current System

The combination of step 1 and 2 in chapter 3 produces current government inventory system in graph presentation style, which may be more effective to describing the system since the standard symbols, may limit communication.

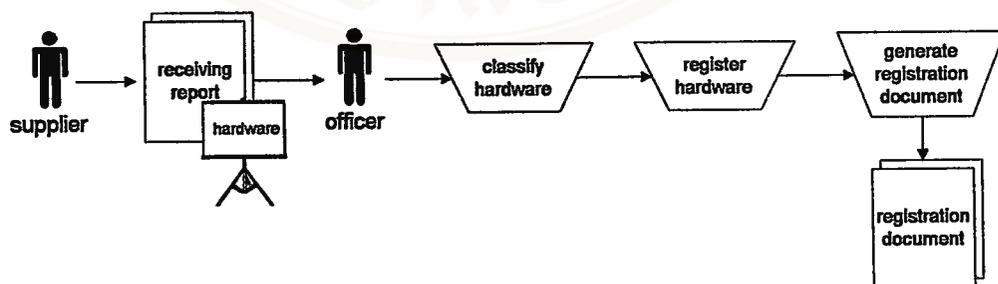


Figure 4.1 Graphic presentation of current storekeeping process

Figure 4.1 describes storekeeping process, supplier submits hardware and receiving report to officer The receiving report consists of supplier detail (e.g. name, address, and telephone number), transport date, specification, quantity and cost per unit

of hardware. Next, officer will classify hardware by group, class, type and specification with precedence. After hardware has been classified, officer will generate hardware ID and register the hardware. Registration data is kept in registration document, which is shown in appendix A.

Figure 4.2 illustrates distribution process. Then approval requisition form is sent to officer, officer will update distributor name and recipient name in registration document. Borrow process (Figure 4.3) is the same as distribution process but it adds return date in registration document.

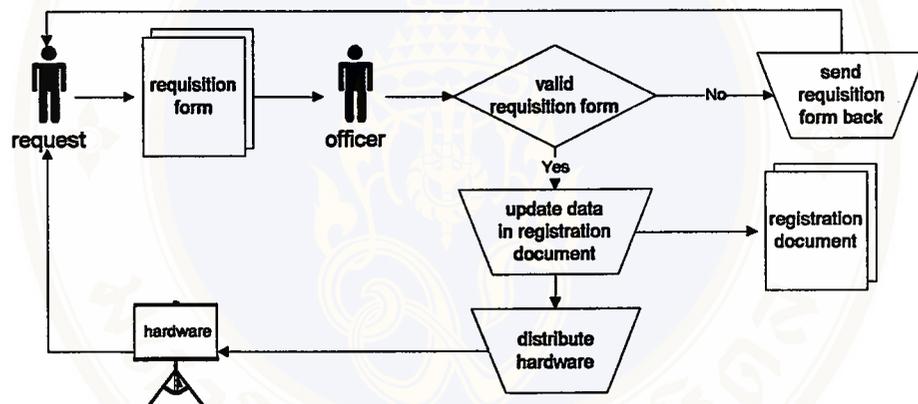


Figure 4.2 Graphic presentation of current distribution process

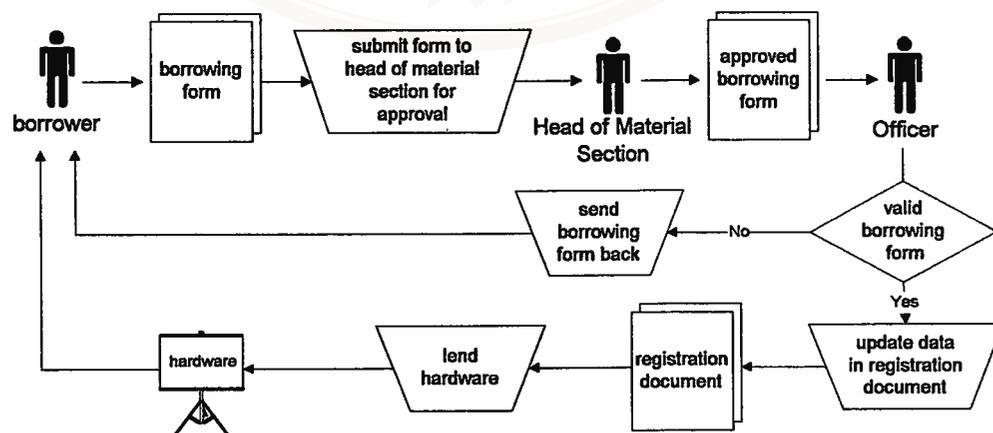


Figure 4.3 Graphic presentation of current borrow process

In case of there is out of order hardware, user will give its ID, name for validation. Valid hardware is collected and generated to repair report. This report is sent to purchasing department, who contacts to supplier and tracks repair as shown in Figure 4.4.

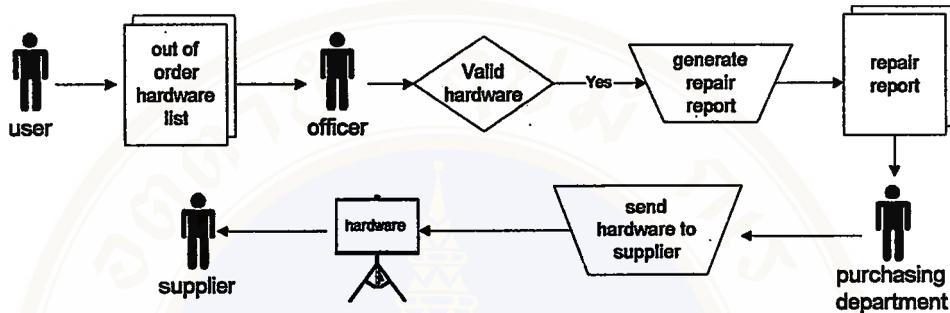


Figure 4.4 Graphic presentation of current repair process

Before the end of September head of material inventory section will appoint committee to audit past operation from October 1st to September 30th. After annual audit has finished, officer will create report of unused hardware to dispose as shown in Figure 4.5.

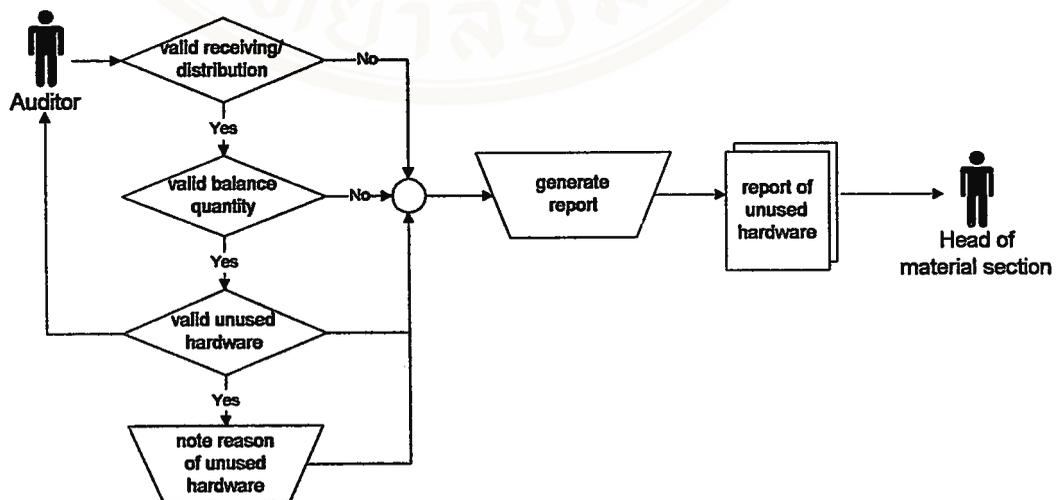


Figure 4.5 Graphic presentation of current annual audit process

Figure 4.6 shows disposal process, unused hardware is sold, exchanged, given to other sections or destroyed. If unused hardware cannot dispose by these methods, officer will dispose the hardware as lose hardware. Then officer removes data of disposed hardware from registration document

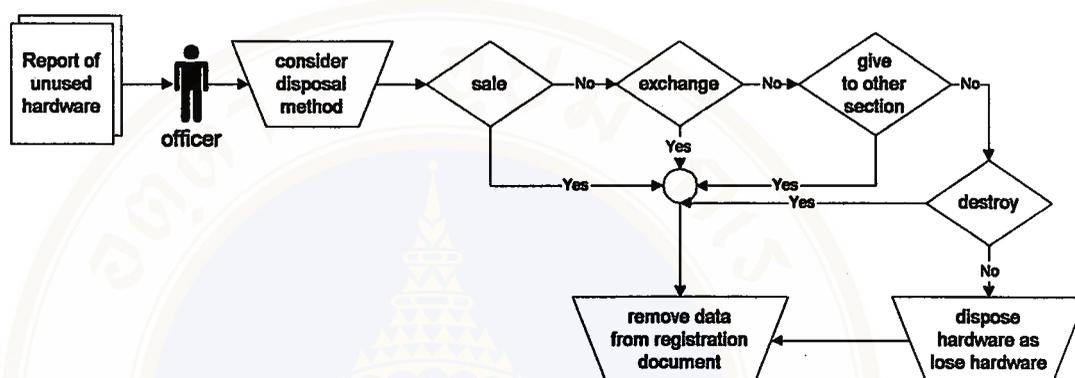


Figure 4.6 Graphic presentation of current disposal process

4.2 Results from System Analysis and Design

The results from system analysis and design consist of data flow diagram (describes and analyzes the movement of data through system), data dictionary (defines systems elements), data structure diagram (describes relation between entities), structure charts, input, output and program specification.

4.2.1 Data Flow Diagram

In Figure 4.7, The presentation graphs of current system are converted to new system, which consists of the following entities: section, head of material section, officer, purchasing department

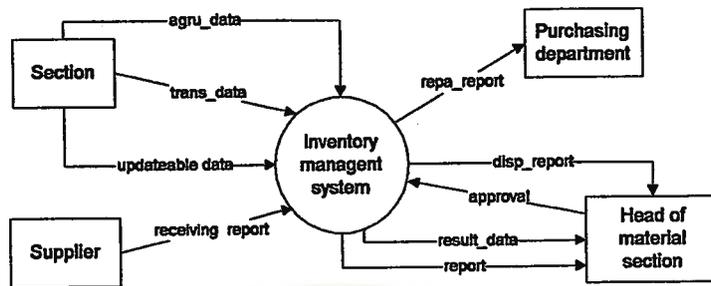


Figure 4.7 Context diagram of government

The new system revealed 4 major processes: data manipulation, transaction process, queries process and report generation. The overall system, consisting of these processes, is shown in Figure 4.8

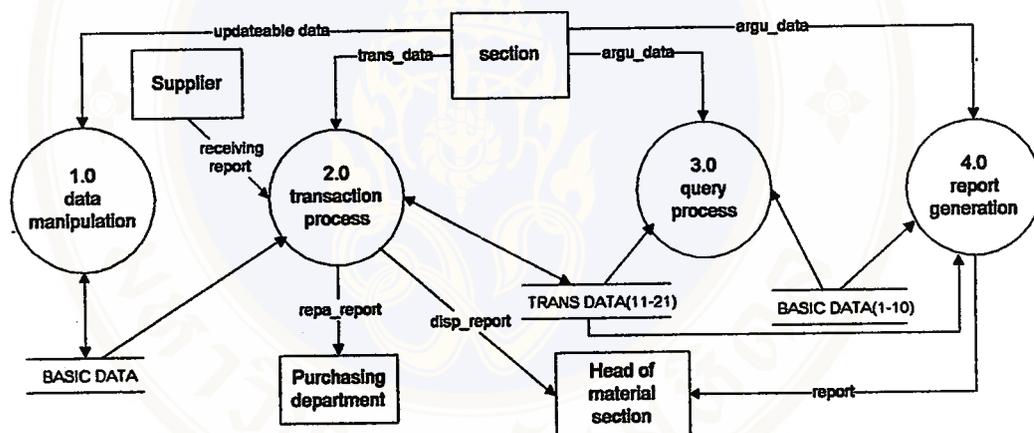


Figure 4.8 DFD Level 1: Government inventory management system

This data flow analysis also shows that the system use 21 data stores, which are divided into 2 categories:

1. "Basic data": It consists of 10 data stores, which are HW, HW_GROUP, HW_CLASS, HW_TYPE, SUPPLIER, DIVISION, SECTION, SECURITY, STAFF and PROJECT. These data stores are not usually modified. The system fetch read-only data to display and reference.

2. “Trans data”: It consists of 11 data stores, which are CONTROL, CONT_DETAIL, HW_REGI, DISTRIBUTE, DIST_DETAIL, RETURN, RETU_DETAIL, DISTPOSE, DISP_DETAIL, REPAIR and REPA_DETAIL. These data stores are modified in each transaction process (registration, distribution, return, repair and disposal process)

The details of “basic data” and “transaction data” data stores are described in section 4.2.4. Database Design. The system is explained in detail through the lower-level subsystem diagram, that developed to understand the activities associated with the system.

Level 2: 1. Data Manipulation

Data manipulation consists of the activities of adding, deleting and editing data. These activities manipulate data in “Basic data” data store. Some data store has one-to-many relationship with another one consequently; its activities have a little different detail.

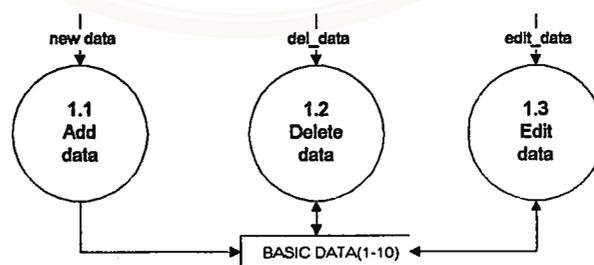


Figure 4.9 DFD Level 2: 1 Data manipulation

The downward explosion of the data manipulation process maintains the numbering scheme introduced at the system level 3 in Figure 4.10.

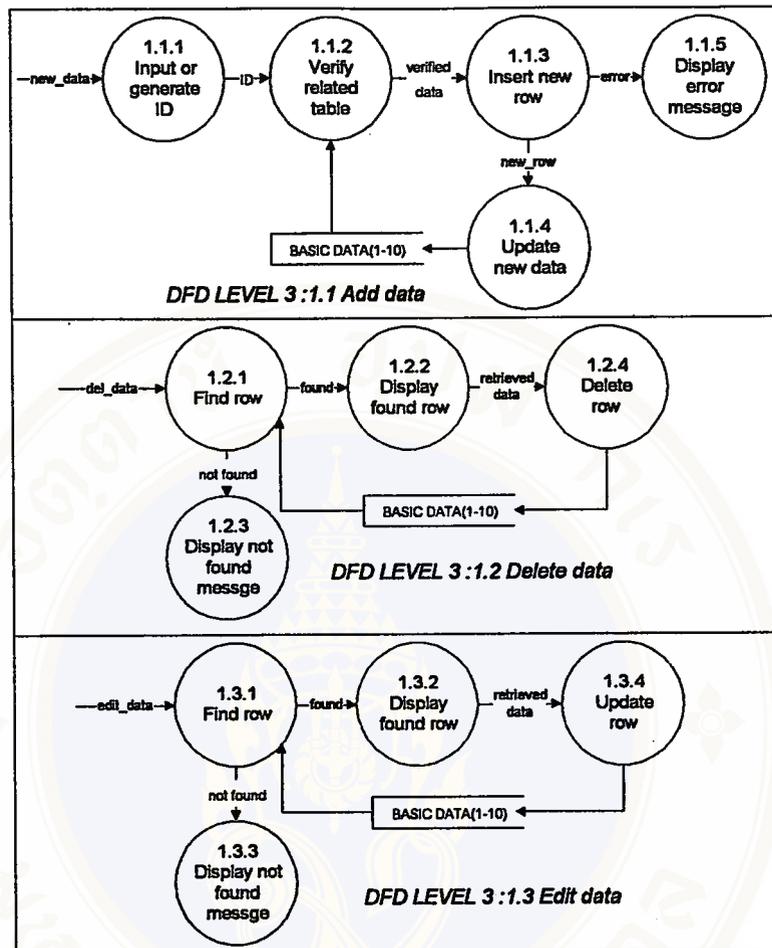


Figure 4.10 DFD Level 3: Data manipulation

Level 2: 2. Transaction Process

Transaction process consists of 6 activities: “register hardware”, “distribute hardware”, “return hardware”, “repair hardware”, “receive repaired hardware” and “dispose hardware”, which are shown in Figure 4.11. In Figure 4.8 DFD Level 1 mentions that the activities use “Basic data” and “Trans data”, but it does not explain. This level explains details of related data stores in each activity. In addition, DFD level 3 are show in Figure 4.12 -4.14 to clarify the activities.

subactivity stores registration data into 3 data stores: CONTROL, CONT_DETAIL and HW_REGI.

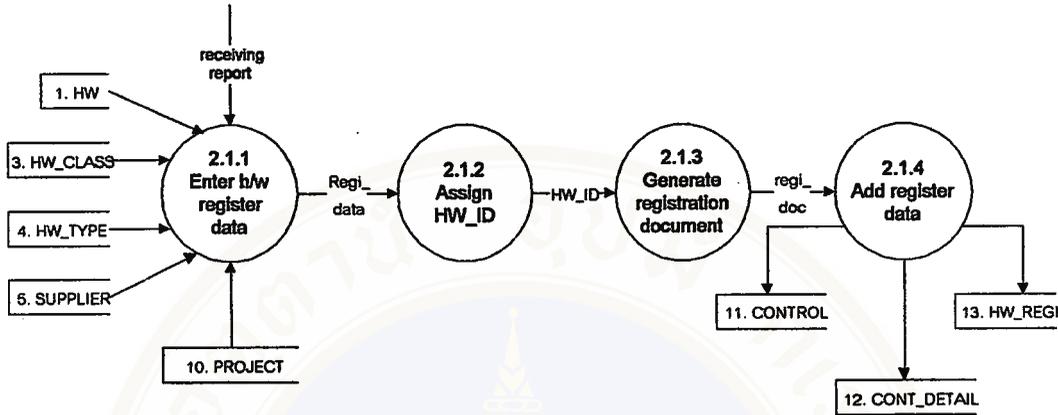


Figure 4.12 DFD Level 3: 2.1 Register hardware

“Distribute hardware” activity involves 6 subactivities. “Enter distribution data” subactivity gets one of important data is HW_ID, which is verified in 2 subactivities, “verify HW_ID”: to ensure that there is the HW_ID in data store and “verify h/w status”: to ensure that the hardware can be distributed (status = WAIT).

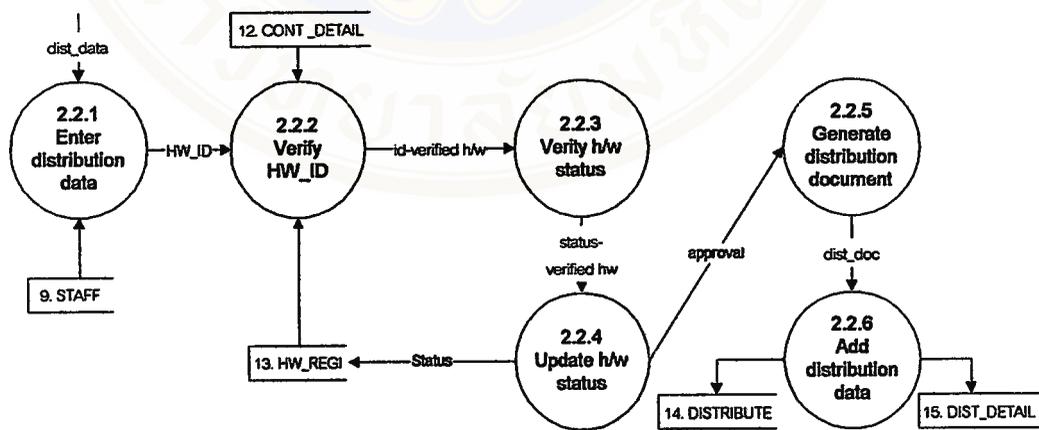


Figure 4.13 DFD Level 3: 2.2 Distribute hardware

When the HW_ID pass the verification “update h/w status” subactivity changes status value in HW_REGI to distributed or borrowed status (status = USE or status = BORROW). “Generate distribution document” subactivity transform data into printed report. Finally, “add distribution data” subactivity stores distribution data into

DISTRIBUTE and DIST_DETAIL data store. Other activities in transaction process are similar to “distribute hardware” activity, hence their description are skipped

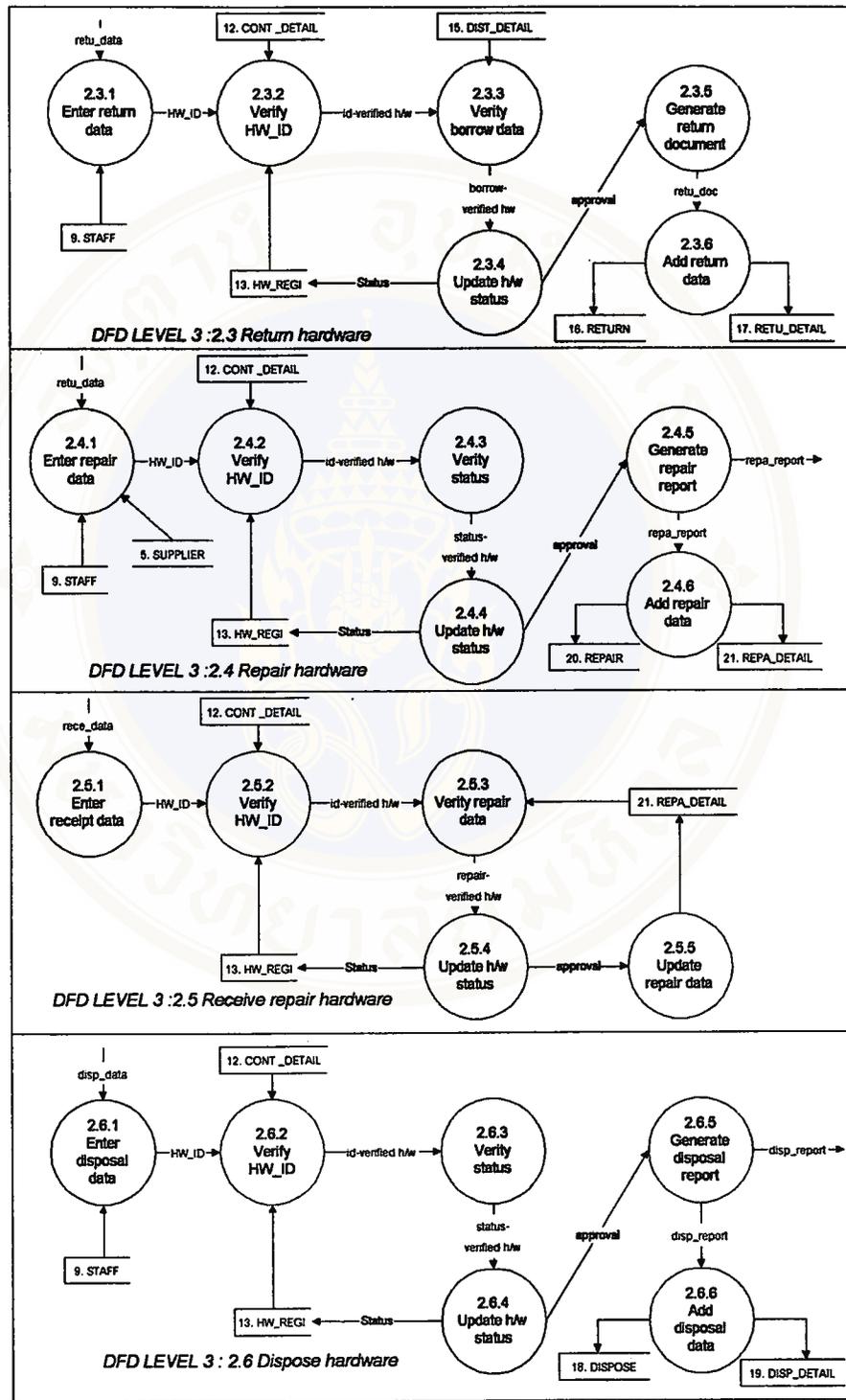


Figure 4.14 DFD Level 3: Transaction process

Level 2: 3. Query process

Query process consists of 4 activities, which use all data stores to process data. Result data has 2 types: tabular and graph, which have individual processing. See list of result data in section 4.2.2 data dictionary. “Process data” activity is calculation, summary and comparative analysis.

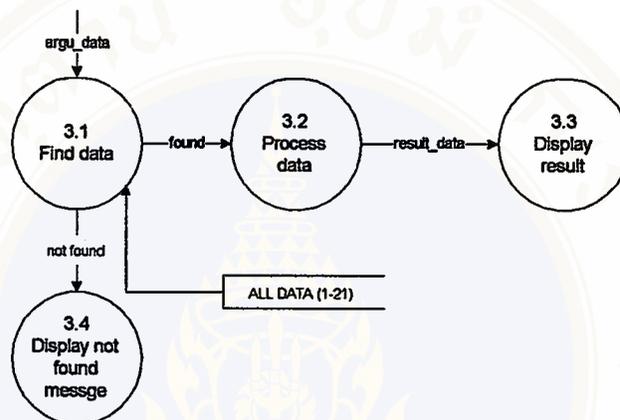


Figure 4.15 DFD Level 2: 3. Query Process

Level 2: 4. Report generation

Report generation consists of 4 activities, which use all data stores to process printable reports. See list of printable reports in 4.2.2 data dictionary. “Process data” activity is calculation and summary data.

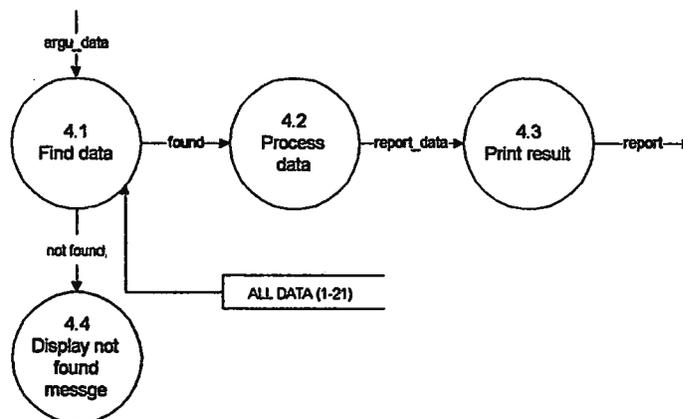


Figure 4.16 DFD Level 2: 4. Report generation

4.2.2 Data Dictionary

The data dictionary elements developed for government inventory management system consists of process, procedure, data flow and data structure. The process of developing the data dictionary itself forces analysis to clarify their understanding of the data in the system

Figure 4.17 identifies each of the processes included in the DFD level 1. Each process is briefly described to clarify its purpose. The inbound and outbound data flows for each of the five processes are also shown.

<p>PROCESS NAME: 1. Data manipulation DESCRIPTION: Data of "BASIC DATA" is added, edited or deleted and stored in database. INBOUND DATA FLOWS: updateable_data OUTBOUND DATA FLOWS: data of "BASIC DATA" data store</p>
<p>PROCESS NAME: 2. Transaction process DESCRIPTION: Receiving report is received and generated to registration data. The registration data is stored in data store. After that every activities with registered hardware will be recorded in "TRANS DATA" data store. INBOUND DATA FLOWS: In: trans_data OUTBOUND DATA FLOWS: data of "TRANS DATA" data store.</p>
<p>PROCESS NAME: 3. Query process DESCRIPTION: User can select question to query by identifying argument, system will process result data in graph or tabular format of display screen. INBOUND DATA FLOWS: argu_data data of "BASIC DATA" data store data of "TRANS DATA" data store OUTBOUND DATA FLOWS:</p>
<p>PROCESS NAME: 4. Report generation DESCRIPTION: User can select question to generate report by identifying argument, system will process printed report. INBOUND DATA FLOWS: argu_data data of "BASIC DATA" data store data of "TRANS DATA" data store OUTBOUND DATA FLOWS: report</p>

Figure 4.17 First level process descriptions.

Figure 4.18 through 4.19 identify processes that take place in the DFD level 3 of transaction process. The processing logic is summarized. Some process has alias name and procedure. The logic summary of procedure is shown in Figure 4.20.

<p>PROCESS NAME: 2.1.1 Enter h/w registration LOGIC SUMMARY: SELECT MAX(convert (numeric, cont_no)) INTO: ls_main_no FROM CONTROL WHERE (control.div_id = div_id) and (control.sect_id = sect_id) and (control.year = year)</p> <p>Cont_no = string (ls_main_no +1) READ budget_year FROM edit box (0<budget_year<=This Year) READ project_id , method FROM list box READ receipt_date FROM edit box (0< receipt_date<= NOW) READ supplier_id FROM list box READ total_cost , recipient_name and comment FROM edit box REPEAT READ class_id, type_id, spec_id FROM list box OR CALL search_hw_code READ desc, cost, quantity and comment FROM edit box. CALL Assign_hw_id UNTIL END OF REGISTRATION</p>	<p>PROCESS NAME: 2.1.2 Assign hw_id LOGIC SUMMARY: Read hw_id_temp FROM edit box</p> <p>FOR i=1 to quantity SELECT count(hw_id) INTO :no FROM HW_REGI, CONT_DETAIL WHERE (cont_detail.class_id = class_id) and (cont_detail.type_id =type_id) and (cont_detail.spec_id = spec_id)</p> <p> HW_ID = hw_temp + ' ' + string (no + i) READ machine_code FROM edit box SET status = 1 // wait SET use_place = NULL END FOR</p>
<p>PROCESS NAME: 2.1.3 Generate registration document LOGIC SUMMARY: SELECT * FROM CONTROL, CONT_DETAIL, HW_REGI WHERE (control.div_id = div_id) and (control.sect_id = sect_id) and (control.year = year) and (control.cont_no = no) Arrange data to registration Form Print</p>	<p>PROCESS NAME: 2.1.4 Add register data LOGIC SUMMARY: ADD NEW RECORD IN CONTROL, CONT_DETAIL, HW_REGI</p>
<p>PROCESS NAME: 2.2.2 Verify_HW_ID ALIAS: 2.3.2, 2.4.2, 2.5.2, 2.6.2 DESCRIPTION: To ensure that there is HW_ID in system LOGIC SUMMARY: IF (SELECT * FROM HW_REGI WHERE hw_regi.hw_id=hw_id) THEN RETURN pass=TRUE END IF</p>	<p>PROCESS NAME: 2.2.1 Enter distribution data LOGIC SUMMARY: SELECT MAX(convert (numeric, dist_no)) INTO :ls_main_no FROM DISTRIBUTE WHERE (distribute.div_id = div_id) and (distribute.sect_id = sect_id) and (distribute.year = year)</p> <p>dist_no = string (ls_main_no +1) READ date FROM edit box (0< receipt_date<= NOW) READ recipient_id and distributor_id FROM list box</p> <p>REPEAT READ HW_ID FROM list box OR CALL Search_HW_ID DISPLAY hardware_detail FROM file IF proc=distribute THEN Status=1 ELSE IF proc=borrow THEN Status=2 READ return_date FROM edit box. END IF UNTIL END OF DISTRIBUTION</p>
<p>PROCESS NAME: 2.2.3 Verify h/w status ALIAS: 2.4.3, 2.6.3 DESCRIPTION: To ensure that the hardware is in correct status LOGIC SUMMARY: CASE proc OF Distribute: find_status = '1' // wait Return: find_status = '3' // borrow Repair: find_status = '4' // bad Dispose: find_status = '4' // bad END CASE IF status = find_status THEN pass=TRUE END IF</p>	<p>PROCESS NAME: 2.2.4 Update h/w status ALIAS: 2.3.4, 2.4.4, 2.5.4, 2.6.4 LOGIC SUMMARY: CASE proc of Distribute:HW_REGI.status=2 Borrow: HW_REGI.status=3 Return: READ FROM ratio button IF ratio_button.click = 'ดี' THEN // good HW_REGI.status= 1 ELSE IF ratio_button.click = 'ไม่ดี' THEN // bad HW_REGI.status= 4 END IF Repair: HW_REGI.status=5 Dispose: HW_REGI.status=6 END CASE</p>
<p>PROCESS NAME: 2.2.5 Generate distribution document LOGIC SUMMARY: SELECT * FROM DISTRIBUTE, DIST_DETAIL WHERE (distribute.div_id = div_id) and (distribute.sect_id = sect_id) and (distribute.year = year) and (distribute.dist_no = no) Arrange data to distribution Form Print</p>	<p>PROCESS NAME: 2.2.6 Add distribution data LOGIC SUMMARY: ADD NEW RECORD IN DISTRIBUTE AND DIST_DETAIL</p>

Figure 4.18 Third level process descriptions

<p>PROCESS NAME: 2.3.1 Enter return data LOGIC SUMMARY: SELECT MAX(convert (numeric, retu_no)) INTO :ls_main_no FROM RETURN WHERE (return.div_id = div_id) and (return.sect_id = sect_id) and (return.year = year)</p> <p>retu_no= string (ls_main_no +1) READ date FROM edit box (0< receipt_date<= NOW) READ return_id and recipient_id FROM list box REPEAT READ HW_ID from list box OR CALL Search_HW_ID DISPLAY hardware_detail FROM file UNTIL END OF RETURN</p>	<p>PROCESS NAME: 2.3.3 Verity_borrow LOGIC SUMMARY: SELECT status FROM HW_REGI WHERE hw_relg.hw_id= hw_id IF (status=3) THEN SELECT dis_detail.dist_no, distribute.year, dist_detail.return_date FROM DIST_DETAIL, DISTRIBUTE WHERE (dist_detail.hw_id =hw_id) and (dist_detail.div_id =div_id) and (dist_detail.sect_id=sect_id) and (dist_detail.status=2) ORDER BY distribute.date DESC;</p> <p> DISPLAY div_id+sect_id +'/'+dist_year+'/' +dist_no IF return_date< NOW THEN DISPLAY message "เกินกำหนดคืน" ELSE DISPLAY "ทรัพย์สินถูกขโมยไม่ได้ถูกขืม" END IF END IF</p>
<p>PROCESS NAME: 2.3.5 Generate return document LOGIC SUMMARY: SELECT * FROM RETURN, RETU_DETAIL WHERE (return.div_id = div_id) and (return.sect_id = sect_id) and (return.year = year) and (return.retu_no = no) Arrange data to return Form Print</p>	<p>PROCESS NAME: 2.3.6 Add_return data LOGIC SUMMARY: ADD NEW RECORD IN RETURN AND RETU_DETAIL</p>
<p>PROCESS NAME: 2.4.1 Enter repair data LOGIC SUMMARY: SELECT MAX(convert (numeric, repa_no)) INTO :ls_main_no FROM REPAIR WHERE (repair.div_id = div_id) and (repair.sect_id = sect_id) and (repair.year = year)</p> <p>repa_no= string (ls_main_no +1) READ date FROM edit box (0< receipt_date<= NOW) READ supp_id FROM list box REPEAT READ HW_ID from list box OR CALL Search_HW_ID DISPLAY hardware_detail FROM file READ cause,cost,repa_date,retu_date FROM edit box UNTIL END OF REPAIR</p>	<p>PROCESS NAME: 2.4.5 Generate repair report LOGIC SUMMARY: SELECT * FROM REPAIR, REPA_DETAIL WHERE (repair.div_id = div_id) and (repair.sect_id = sect_id) and (repair.year = year) and (repair.repa_no = no) Arrange data to repair Form Print</p>
<p>PROCESS NAME: 2.5.1 Enter receipt data LOGIC SUMMARY: REPEAT READ HW_ID FROM edit box OR CALL Search_HW_ID CALL Verify repair data DISPLAY repair_detail from file READ comment FROM listbox UNTIL END OF RECEIPT DATA</p>	<p>PROCESS NAME: 2.4.6 Add repair data LOGIC SUMMARY: ADD NEW RECORD IN REPAIR AND REPA_DETAIL</p>
<p>PROCESS NAME: 2.5.5 Update repair data LOGIC SUMMARY: Update comment = ""+ comment To REPA_DETAIL</p>	<p>PROCESS NAME: 2.5.3 Verify repair data LOGIC SUMMARY: SELECT status FROM HW_REGI WHERE hw_regi.hw_id= hw_id IF (status=5) THEN SELECT * FROM REPA_DETAIL, REPAIR WHERE (repa_detail.hw_id =hw_id) and(repa_detail.div_id=div_id) and (repa_detail.sect_id=sect_id) ORDER BY repa.date DESC;</p> <p> DISPLAY repair_detail Pass = TRUE END IF</p>
<p>PROCESS NAME: 2.6.5 Generate disposal report LOGIC SUMMARY: SELECT * FROM DISPOSE, DISP_DETAIL WHERE (dispose.div_id = div_id) and (dispose.sect_id = sect_id) and (dispose .year = year) and (dispose .disp_no = no) Arrange data to disposal Form Print</p>	<p>PROCESS NAME: 2.6.1 Enter disposal data LOGIC SUMMARY: SELECT MAX(convert (numeric, dist_no)) INTO :ls_main_no FROM DISPOSE WHERE (dispose.div_id = div_id) and (dispose.sect_id = sect_id) and (dispose.year = year)</p> <p>disp_no= string (ls_main_no +1) READ date FROM edit box (0< receipt_date<= NOW) REPEAT READ HW_ID from list box OR CALL Search_HW_ID DISPLAY hardware_detail FROM file READ method FROM listbox READ cause,cost,comment FROM editbox UNTIL END OF DISPOSE</p>
<p>PROCESS NAME: 2.6.6 Add disposal data LOGIC SUMMARY: ADD NEW RECORD IN DISPOSE AND DISP_DETAIL</p>	

Figure 4.19 Third level process description (continued)

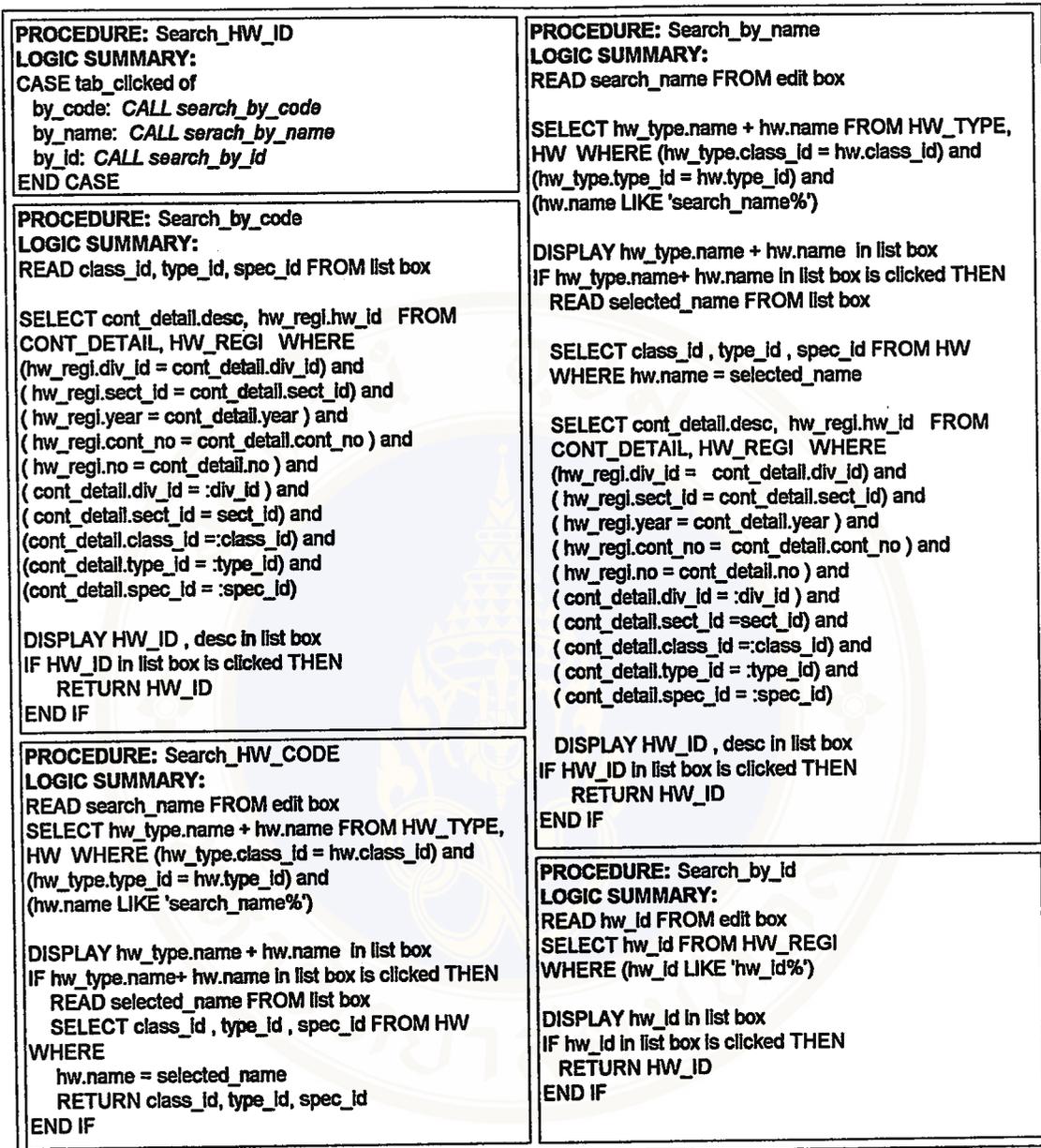


Figure 4.20 Logic summary of procedure

The other processes in data manipulation, query process and report generation are not included in process description because the low-level of its DFD are sufficient explanation.

Figure 4.21 through 4.22 define 35 data flows that are included in the system. The processes that data flows originate from and into are also identified by number and name. Each data flow is also described by the data structures it contains. The data

structures that are part of this system are describe in Table 4.1. Unnamed data flows mean all data of data stores, which see detail in section 4.2.4 Database design.

Table 4.1 Data dictionary for data structure

Data Structure	Description
HW CODE	Class id + type Id + spec id
HW data	*see 1. HW data store*
Hw disp list	HW ID + disposal method + disposal cause + (sale cost)
Hw dist list	(HW CODE) + HW ID + (return date)
Hw_list	Hw_name + (hw_specification) + cost per unit + quantity + (comment)
Hw repa list	HW ID + cause + cost + repair date + receive date
Hw retu list	(HW CODE) + HW ID + (dist no)
Supplier data	*see 7.SUPPLIER data store*

4.2.3 Input and Output Design

Output reports, which support the activity of the operation and management. The outputs are generated in tubular form. See designed output reports in appendix B.

Input and output screens are user-friendly with GUI (graphic user interface, they are MDI window. An MDI window is a frame window in which you can open multiple document windows (sheets) and move among the sheets. The input screens have input validation. Some output screens, especially in the query process have a characteristic of drill down. A user can choose to access between a particular data, which is deep in detail, or just overall data, which is convenient for comparative analysis. See designed screens in Appendix C.

<p>DATA FLOW NAME: approval DESCRIPTION: result that there is no error after system has updated hardware status in "HW_REGI" data store</p> <p>FROM PROCESS: 2.2.4, 2.4.4, 2.6.4 update h/w status</p> <p>TO PROCESS: 2.2.5 Generate distribution document 2.3.5 Generate return document 2.4.5 Generate repair report 2.5.5 Update repair data 2.6.5 Generate disposal report</p> <p>DATA STRUCTURE: [dist_data, retu_data, repa_data, disp_data] + id_verified h/w</p>	<p>DATA FLOW NAME: argu_data DESCRIPTION: Retrieved argument of query process and report generation that is identified by user</p> <p>FROM PROCESS:</p> <p>TO PROCESS: 3.0 query process 3.1 find data 4.0 report generation 4.1 find data</p> <p>DATA STRUCTURE: (div_id + sect_id) + ((year, budget year)) + (HW_CODE) + ((today, start date + stop date))</p>
<p>DATA FLOW NAME: borrow_verified h/w DESCRIPTION: system verifies that hardware was borrowed</p> <p>FROM PROCESS: 2.3.3 Verify borrow data TO PROCESS: 2.3.4 update h/w status</p> <p>DATA STRUCTURE: dist_no + return_date</p>	<p>DATA FLOW NAME: del_data DESCRIPTION: ID or name that can address row</p> <p>FROM PROCESS:</p> <p>TO PROCESS: 1.2 delete data 1.2.1 find row</p> <p>DATA STRUCTURE: [ID, name] *depend on each data store*</p>
<p>DATA FLOW NAME: disp_data DESCRIPTION: data of disposed hardware that is given from auditor in annual audit</p> <p>FROM PROCESS: 2.6 dispose hardware 2.6.1 enter disposal data</p> <p>DATA STRUCTURE: div_id + sect_id + year + disp_no + disp_date + {disp_hw_list}</p>	<p>DATA FLOW NAME: dist_data DESCRIPTION: data in approval requisition form</p> <p>FROM PROCESS:</p> <p>TO PROCESS: 2.2 distribute hardware 2.2.1 enter distribution data</p> <p>DATA STRUCTURE: Div_id + sect_id + year + dist_no + dist_date + [recipient ID, recipient name] + [distributor ID, distributor name] + {hw_dist_list}</p>
<p>DATA FLOW NAME: disp_report DESCRIPTION: Report of unused hardware that has been considered disposal method. This report has 2 copies, one is sent to head of material section and other keeps in system</p> <p>FROM PROCESS: 2.0 transaction process 2.6 dispose hardware 2.6.5 generate disposal report</p> <p>TO PROCESS: 2.6.6 add disposal data</p> <p>DATA STRUCTURE: Div_id + sect_id + year + disp_no + disposal date + {hw_disp_list}</p>	<p>DATA FLOW NAME: dist_doc DESCRIPTION: distribution document for checking</p> <p>FROM PROCESS: 2.2.5 Generate distribution document</p> <p>TO PROCESS: 2.2.6 Add distribution data</p> <p>DATA STRUCTURE: *see report in appendix B*</p>
<p>DATA FLOW NAME: edit_data DESCRIPTION: ID or name that can address row and new data for updates the row</p> <p>FROM PROCESS: 1.3 edit data 1.3.1 find row</p> <p>DATA STRUCTURE: [ID, name]+ new_data</p>	<p>DATA FLOW NAME: error DESCRIPTION: Error message from system that respond to user</p> <p>FROM PROCESS: 1.1.3 insert new row TO PROCESS: 1.1.5 display error message</p> <p>DATA STRUCTURE: *string of error statement*</p>
<p>DATA FLOW NAME: found DESCRIPTION: result that system can find requested row</p> <p>FROM PROCESS: 1.2.1 find row 1.3.1 find row</p> <p>TO PROCESS: 1.2.1 display found row 1.3.2 display found row</p> <p>DATA STRUCTURE: *row number of found row*</p>	<p>DATA FLOW NAME: HW_ID DESCRIPTION: Unique ID that represents registered hardware, last digit is generated by system to protect redundant data</p> <p>FROM PROCESS: 2.1.1 Enter h/w register data 2.2.1 Enter distribution data 2.3.1 Enter return data 2.4.1 Enter repair data 2.5.1 Enter receipt data 2.6.1 Enter disposal data</p> <p>TO PROCESS: 2.1.3 Generate registration documentation 2.2.2, 2.3.2, 2.4.2, 2.5.2, 2.6.2 Verify HW_ID</p> <p>DATA STRUCTURE: *string that usually consists of section_code + class_id + type_id + spec + year*</p>

Figure 4.21 Data dictionary entries for data flows

<p>DATA FLOW NAME: ID DESCRIPTION: Unique code that is primary key in data store and is generated by system or user input FROM PROCESS: 1.1.1 input or generate ID TO PROCESS: 1.1.2 verify related table DATA STRUCTURE: *primary key in data store*</p>	<p>DATA FLOW NAME: id_verified h/w DESCRIPTION: system verifies that hardware was registered FROM PROCESS: 2.2.2, 2.3.2, 2.4.2, 2.5.2, 2.6.2 Verify HW_ID TO PROCESS: 2.2.3, 2.4.3, 2.6.3 verify status 2.3.3 verify borrow data 2.5.3 verify repair data DATA STRUCTURE: *data in HW_REGI and CONT_DETAIL data store*</p>
<p>DATA FLOW NAME: new_data DESCRIPTION: Data of 10 data stores ("BASIC DATA") that pass validation rule FROM PROCESS: TO PROCESS: 1.1 add data 1.1.1 input or generate ID DATA STRUCTURE: *data of data store*</p>	<p>DATA FLOW NAME: new_row DESCRIPTION: approval data that is inserted into data store FROM PROCESS: 1.1.3 Insert new row TO PROCESS: 1.1.4 update new data DATA STRUCTURE: *alias: verified data*</p>
<p>DATA FLOW NAME: not found DESCRIPTION: result that system cannot find requested row FROM PROCESS: 1.2.1 find row 1.3.1 find row TO PROCESS: 1.2.3 display not found message 1.3.3 display not found message DATA STRUCTURE: *0*</p>	<p>DATA FLOW NAME: rece_data DESCRIPTION: data of repaired hardware FROM PROCESS: TO PROCESS: 2.5 receive repaired hardware 2.5.1 enter receipt data DATA STRUCTURE: [supplier ID, supplier name] + [hardware id, hardware name] + (cost) + (comment)</p>
<p>DATA FLOW NAME: receiving report DESCRIPTION: receipt hardware list and supplier data from supplier FROM PROCESS: TO PROCESS: 2.0 Transaction process 2.1 Registration hardware 2.1.1 Enter h/w register data DATA STRUCTURE: Ship_id + supplier_data + receive_data + {hw_list}</p>	<p>DATA FLOW NAME: regi_data DESCRIPTION: data in approval receiving report and classified hardware list FROM PROCESS: TO PROCESS: 2.1.1 enter h/w register data 2.1.2 assign HW_ID DATA STRUCTURE: Receiving report + supplier_data + hw_data</p>
<p>DATA FLOW NAME: regi_doc DESCRIPTION: new registration document for checking FROM PROCESS: 2.1.3 Generate registration document TO PROCESS: 2.1.4 Add register data DATA STRUCTURE: * see report in appendix B*</p>	<p>DATA FLOW NAME: repa_data DESCRIPTION: out of order hardware data FROM PROCESS: TO PROCESS: 2.4 repair hardware 2.4.1 enter repair data DATA STRUCTURE: div_id + sect_id + year + repa_no + repa_date + [supplier ID, supplier name] + {hw_repa_list}</p>
<p>DATA FLOW NAME: repa_report DESCRIPTION: Report of repair hardware that has been verified for adjustment of the supplier data and hardware data. This report has 2 copies, one is sent to purchasing department and other keeps in system. FROM PROCESS: 2.0 transaction process 2.4 repair hardware 2.4.5 generate repair report TO PROCESS: 2.4.6 add repair report DATA STRUCTURE: div_id + sect_id + year + Repa_no + repair date + supplier_data + {hw_repa_list}</p>	<p>DATA FLOW NAME: repair_verified h/w DESCRIPTION: system verified that hardware was repaired FROM PROCESS: 2.5.3 Verify repair data TO PROCESS: 2.5.4 Update h/w status DATA STRUCTURE: * alias: id_verified h/w*</p> <p>DATA FLOW NAME: report_data DESCRIPTION: data from report generation FROM PROCESS: 4.2 process data TO PROCESS: 4.3 print result DATA STRUCTURE: *same as the "report" but it is not printed output *</p>

Figure 4.22 Data dictionary entries for data flows (continued)

<p>DATA FLOW NAME: report DESCRIPTION: report is generated by system that has varies style FROM PROCESS: 4.0 report generation 4.3 print result TO PROCESS: DATA STRUCTURE: [hardware report, supplier report, division report, section report, summary of hardware registration in requested budget year, summary of repair hardware, summary of disposal hardware, summary of hardware quantity in department, summary of hardware quantity in requested budget year summary of hardware quantity in section]</p>	<p>DATA FLOW NAME: result_data DESCRIPTION: data from query process FROM PROCESS: 3.2 process data TO PROCESS: 3.3 display result DATA STRUCTURE: [hardware data and status of requested HW_ID hardware data of requested specification and section Hardward list in requested status, section and specification, List of all/specific borrowed hardware in requested return date, List of all/ specific repaired hardware in requested return date, summary graph of hardware quantity in department or division, summary graph of hardware quantity that is distinguished by status, drill down graph of hardware quantity in section]</p>
<p>DATA FLOW NAME: retu_data DESCRIPTION: borrowed hardware data from borrower that is about hardware, status and so on FROM PROCESS: TO PROCESS: 2.3 return hardware 2.3.1 enter return data DATA STRUCTURE: div_id + sect_id + year + retu_no + retu_date + [return ID, return name] + [recipient ID, recipient name] + {hw_retu_list}</p>	<p>DATA FLOW NAME: retu_doc DESCRIPTION: return document for checking FROM PROCESS: 2.3.5 Generate return document TO PROCESS: 2.3.6 Add return data DATA STRUCTURE: *see report in appendix B*</p>
<p>DATA FLOW NAME: retrieved data DESCRIPTION: data that is retrieved from data store when system get row number FROM PROCESS: 1.2.2 display found row 1.3.2 display found row TO PROCESS: DATA STRUCTURE: *data in data store*</p>	<p>DATA FLOW NAME: status DESCRIPTION: status of hardware that store in HW_REGI data store FROM PROCESS: 2.2.4, 2.3.4, 2.4.4,2.5.4, 2.6.4 update h/w status TO PROCESS: DATA STRUCTURE: [wait, use, borrow, bad, repair, dispose]</p>
<p>DATA FLOW NAME: status_verified h/w DESCRIPTION: system verifies that hardware is in valid status FROM PROCESS: 2.2.3, 2.4.3, 2.6.3 verify status TO PROCESS: 2.2.4, 2.4.4, 2.6.4 update h/w status DATA STRUCTURE: *allas: id_verified h/w*</p>	<p>DATA FLOW NAME: trans_data DESCRIPTION: Transaction data from section, that would like to distribute, borrow, return, repair or dispose hardware. FROM PROCESS: TO PROCESS: 2.0 transaction process DATA STRUCTURE: [dist_data, retu_data, repa_data, rece_data, disp_data]</p>
<p>DATA FLOW NAME: Updateable data DESCRIPTION: Data of 10 data stores, which can add, edit or delete. The data must pass validation rule. FROM PROCESS: TO PROCESS: 1.0 data manipulation DATA STRUCTURE: [new_data, del_data, edit_data]</p>	<p>DATA FLOW NAME: Verified data DESCRIPTION: Data of basic data is verified with validation rule FROM PROCESS: 1.1.2 verity related table TO PROCESS: 1.1.3 insert new row DATA STRUCTURE: (ID) + new_data *verified data*</p>

Figure 4.23 Data dictionary entries for data flows (continued)

4.2.4 Database Design

Selecting and identifying data from collected data and system requirement, it gets related entities, attributes and constraints. These entities are identified relationships and designed by relational database concept. Through normalization a collection of data in a record structure is refined and shown in Figure 4.24 data structure diagram. The designed database is central database, which works as back office database. In addition, the information system uses in many places, somewhere may not support online system. For this reason, it has an extra data store to store SQL commands of transaction process. The data store is sent to database center and executed, then "Trans data" has consistent data. Descriptions of data stores are shown as following.

1. Table: HW

PK	FK	Field Name	Data Type	Description
✓	✓	Class_ID	Char(4)	Hardware class ID
✓	✓	Type_ID	Char(3)	Hardware type ID
✓		Spec_ID	Char(4)	Hardware specification ID
		Name	Char(40)	Hardware specification name
		Unit	Char(10)	Hardware unit e.g. box, dozen
		Cost	Real	Average cost that is set by Bureau Of Budget
		Comment	Char(250)	Comment

2. Table: HW GROUP

PK	FK	Field Name	Data Type	Description
✓		Group_ID	Char(2)	Hardware group ID
		Name	Char(40)	Hardware group name
		Desc	Char(68)	Description

3. Table: HW CLASS

PK	FK	Field Name	Data Type	Description
✓		Class_ID	Char(4)	Hardware class ID
		Name1	Char(68)	Hardware class name
		Name2	Char(40)	Short name
	✓	Group_ID	Char(2)	Hardware group ID

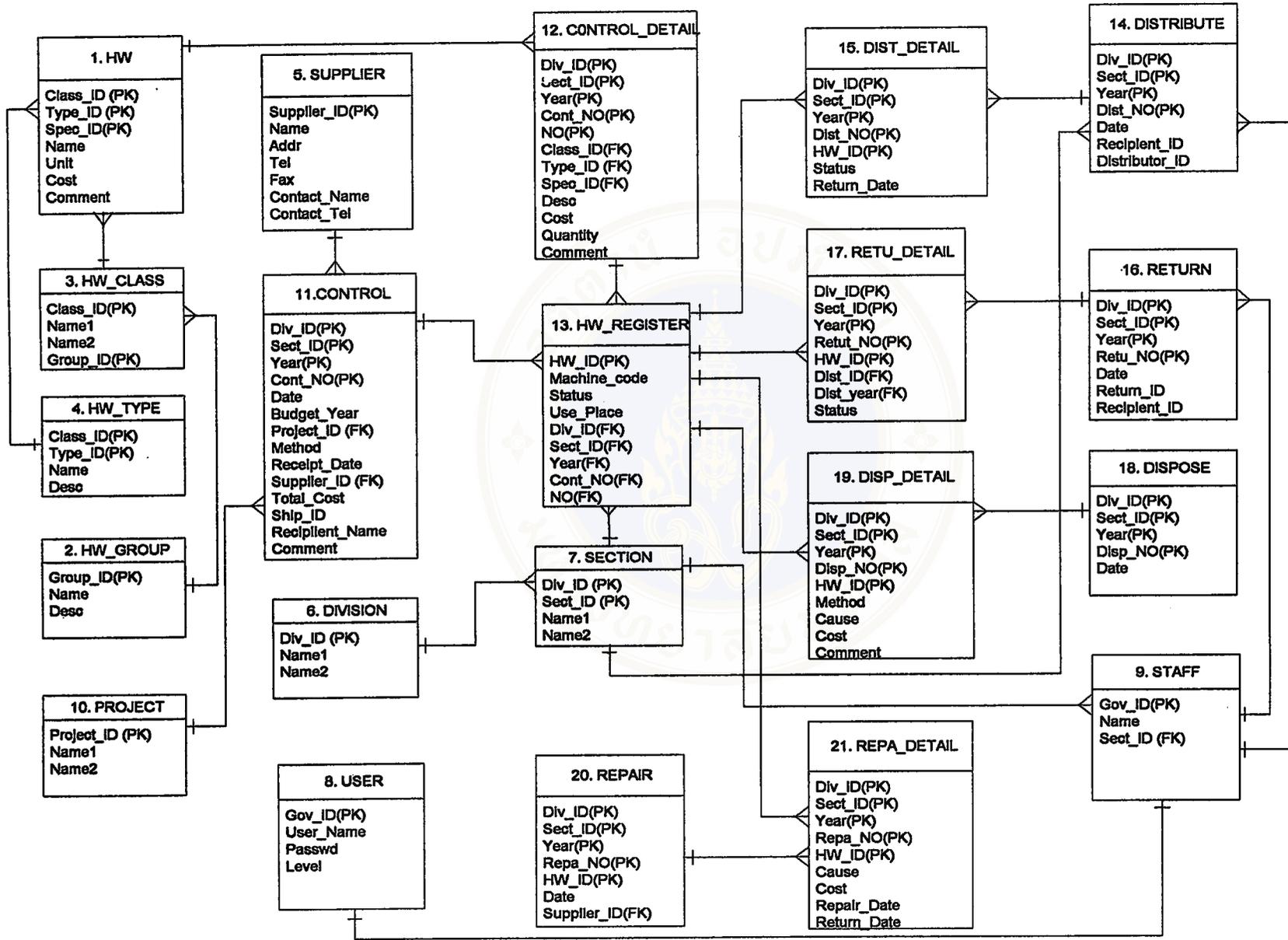


Figure 4.24 Data structure diagram

4. Table: HW TYPE

PK	FK	Field Name	Data Type	Description
✓		Class ID	Char(4)	Hardware class ID
✓		Type ID	Char(3)	Hardware type ID
		Name	Char(40)	Hardware type name
		Desc	Char(68)	Hardware type description

5. Table: SUPPLIER

PK	FK	Field Name	Data Type	Description
✓		Supplier ID	Char(4)	Supplier ID
		Name	Char(68)	Supplier name
		Addr	Char(100)	Address
		Tel	Char(20)	Telephone number
		Fax	Char(20)	Facsimile number
		Contact Name	Char(68)	Contact name
		Contact Tel	Char(20)	Contact telephone number

6. Table: DIVISION

PK	FK	Field Name	Data Type	Description
✓		Div ID	Char(3)	Division ID
		Name1	Char(68)	Division Name
		Name2	Char(40)	Division short name

7. Table: SECTION

PK	FK	Field Name	Data Type	Description
✓	✓	Div ID	Char(3)	Division ID
✓		Sect ID	Char(3)	Section ID
		Name1	Char(68)	Section name
		Name2	Char(40)	Section short name

8. Table: SECURITY

PK	FK	Field Name	Data Type	Description
✓		Gov ID	Char(6)	Government official ID
		User ID	Char(10)	User ID for enter system
		Passwd	Char(10)	Password
		Level	Char(1)	Authorization(1= DBA/ system administrator, 2= operator, 3=general user)

9. Table: STAFF

PK	FK	Field Name	Data Type	Description
✓		Gov ID	Char(6)	Government official ID
		Name	Char(68)	Official name
	✓	Div ID	Char(3)	Division ID
	✓	Sect ID	Char(3)	Section ID

10. Table: PROJECT

PK	FK	Field Name	Data Type	Description
✓		Project ID	Char(6)	Project ID
		Name1	Char(100)	Project name
		Name2	Char(80)	Project short name

11. Table: CONTROL

PK	FK	Field Name	Data Type	Description
✓	✓	Div ID	Char(3)	Division ID
✓	✓	Sect ID	Char(3)	Section ID
✓		Year	Char(2)	Record year
✓		Cont NO	Char(4)	Running number of control table
		Date	Date	Registration date
		Budget Year	Char(4)	Budget year
	✓	Project ID	Char(6)	Project ID
		Method	Char(1)	Receiving method (buy=1, donate=2, receive from other section=3)
		Receipt Date	Date	Receipt date
	✓	Supplier ID	Char(4)	Supplier ID
		Total Cost	Real	Total cost of receiving report
		Ship ID	Char(10)	Ship ID
		Recipient Name	Char(68)	Recipient name
		Comment	Char(150)	Comment e.g. donor name

12. Table: CONT DETAIL

PK	FK	Field Name	Data Type	Description
✓	✓	Div ID	Char(3)	Division ID
✓	✓	Sect ID	Char(3)	Section ID
✓	✓	Year	Char(2)	Record year
✓	✓	Cont NO	Char(4)	Running number of CONTROL table
✓		NO(PK)	Char(4)	Running number of hardware list
	✓	Class ID	Char(4)	Hardware class ID
	✓	Type ID	Char(3)	Hardware type ID
	✓	Spec ID	Char(4)	Hardware specification ID
		Desc	Char(40)	Hardware specification description , trademark , size and the like
		Cost	Real	Cost per unit
		Quantity	Number	Quantity
		Comment	Char(68)	Comment

13. Table: HW REGI

PK	FK	Field Name	Data Type	Description
✓		HW ID	Char(30)	Hardware ID
		Machine code	Char(20)	Machine code that screen on hardware
		Status	Char(1)	Hardware status(wait=1, use=2, borrow = 3, bad =4, repair=5,dispose=6)
		Use Place	Char(68)	Use place e.g. section /building name
	✓	Div ID	Char(3)	Division ID
	✓	Sect ID	Char(3)	Section ID
	✓	Year	Char(2)	Record year
	✓	Cont_NO	Char(4)	Running number of CONTROL table
	✓	NO	Char(4)	Running number of hardware list

14. Table: DISTRIBUTE

PK	FK	Field Name	Data Type	Description
✓		Div ID	Char(3)	Division ID
✓		Sect ID	Char(3)	Section ID
✓		Year	Char(2)	Record year
✓		Dist_NO	Char(4)	Running number of DISTRIBUTE table
		Date	Date	Distribution date
	✓	Recipient ID	Char(6)	Recipient ID (Gov ID)
	✓	Distributor ID	Char(6)	Distributor ID (Gov ID)

15. Table: DIST DETAIL

PK	FK	Field Name	Data Type	Description
✓	✓	Div ID	Char(3)	Division ID
✓	✓	Sect ID	Char(3)	Section ID
✓	✓	Year	Char(2)	Record year
✓	✓	Dist_NO	Char(4)	Running number of DISTRIBUTE table
✓		HW ID	Char(20)	Hardware ID
		Status	Char(1)	Distribution status (distribute=1, borrow =2)
		Return_Date	Date	Return date, In case of distribution return date can be none

16. Table: RETURN

PK	FK	Field Name	Data Type	Description
✓		Div ID	Char(3)	Division ID
✓		Sect ID	Char(3)	Section ID
✓		Year	Char(2)	Record year
✓		Retu_NO	Char(4)	Running number of RETURN table
		Date	Date	Return date
	✓	Return ID	Char(6)	Return ID (Gov ID)
	✓	Recipient ID	Char(6)	Recipient ID (Gov ID)

17. Table: RETU DETAIL

PK	FK	Field Name	Data Type	Description
✓	✓	Div ID	Char(3)	Division ID
✓	✓	Sect ID	Char(3)	Section ID
✓	✓	Year	Char(2)	Record year
✓	✓	Retu NO	Char(4)	Running number of RETURN table
✓		HW ID	Char(20)	Hardware ID
	✓	Dist ID	Char(4)	Distribute ID
	✓	Dist Year	Char(2)	Distribute record year
		Status	Char(1)	returned hardware status(good=1, bad =2)

18. Table: DISPOSE

PK	FK	Field Name	Data Type	Description
✓		Div ID	Char(3)	Division ID
✓		Sect ID	Char(3)	Section ID
✓		Year	Char(2)	Record year
✓		Disp_NO	Char(4)	Running number of DISPOSE table
		Date	Date	Disposal date

19. Table: DISP DETAIL

PK	FK	Field Name	Data Type	Description
✓	✓	Div ID	Char(3)	Division ID
✓	✓	Sect ID	Char(3)	Section ID
✓	✓	Year	Char(2)	Record year
✓	✓	Disp NO	Char(4)	Running number of DISPOSE table
✓		HW ID	Char(20)	Hardware ID
		Method	Char(1)	Disposal method (sell=1, exchange =2, give to other section=3, destroy=4, lost =5)
		Cause	Char(68)	Disposal cost
		Cost	Real	Sell cost (if disposal method is "sell")
		Comment	Char(120)	Comment

20. Table: REPAIR

PK	FK	Field Name	Data Type	Description
✓		Div ID	Char(3)	Division ID
✓		Sect ID	Char(3)	Section ID
✓		Year	Char(2)	Record year
✓		Repa NO	Char(4)	Running number of REPAIR table
		HW ID	Char(20)	Hardware ID
		Date	Date	Record date
	✓	Supplier ID	Char(4)	Supplier ID that repair hardware

21. Table: REPA DETAIL

PK	FK	Field Name	Data Type	Description
✓	✓	Div ID	Char(3)	Division ID
✓	✓	Sect ID	Char(3)	Section ID
✓	✓	Year	Char(2)	Record year
✓	✓	Repa NO	Char(4)	Running number of REPAIR table
✓		HW ID	Char(20)	Hardware ID
		Cause	Char(68)	Repair cause
		Cost	Real	Repair cost
		Repair Date	Date	Repair date
		Return Date	Date	Return repaired hardware date

22. Table: SQL TRANS

PK	FK	Field Name	Data Type	Description
✓		NO	Double	Auto running key
		Sql text	Char(32767)	SQL statement
		Date time	TimeStamp	Date & time of recording
		Done	Char(1)	If 'Y' this record was sent to update.

4.2.5 Procedure Design

PowerBuilder is object-oriented programming, which each created menu or window with PowerBuilder is a self-contained module called an object. The basic building blocks of a PowerBuilder application are the created objects. Each object contains the particular characteristics and behaviors (properties, events, and functions) that are appropriate to it. By taking advantage of object-oriented programming techniques such as encapsulation, inheritance, and polymorphism, the processes in DFD are refined and adjusted as following:

1. Transform the processes in DFD into structure chart, which is easier to maintain and reduce complexity.

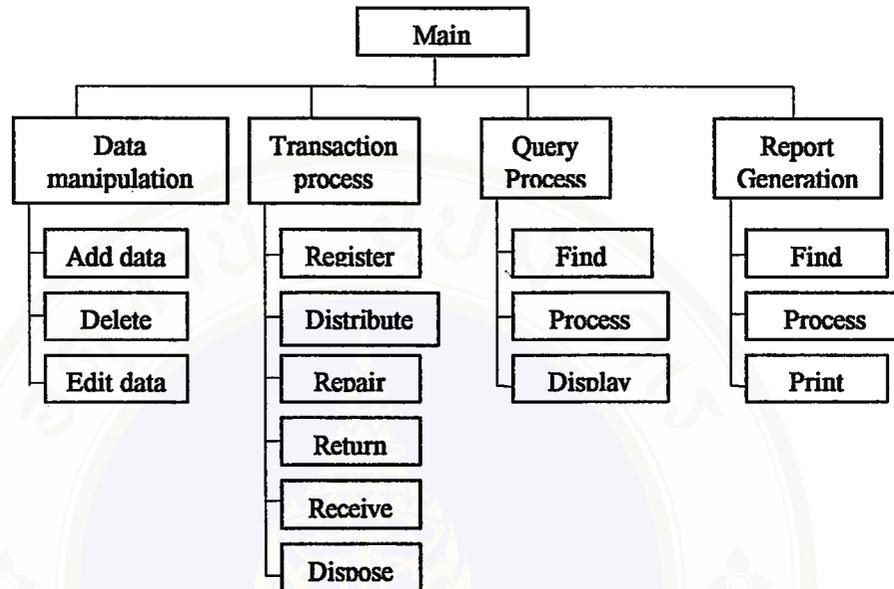


Figure 4.25 Structure Chart

2. Boxes in structure chart are grouped and designed to window objects. These window objects have some common functions and events, which can design to ancestor window objects for inheritance and reuse.

Table 4.2 Characteristic of process

Process	Characteristic
Data manipulation	- All windows can inherit
Transaction process	- Receive repaired hardware window can not inherit - Registration has a additional window for adding hardware registration ID
Query process	- Distinguish tabular and graph output window - Tabular window can not inherit because they have too different function. - Graph window can inherit
Report generation	- All windows use the same window but use different display data (data window object)

3. Design additional events to ease of use in window operations style and convenient operation such as open, close and so on.

Table 4.3 Ancestor window objects

Process	Event group	Event
Data manipulation	Window	Open Close
	Navigation	First Previous Next Last Search
	Data	Add Delete Update Undo Undelete Cancel
Transaction process	Window	Open Close
	Data	Update master Insert detail Update detail Search Print
Query process (graph)	Window	Open Close
	Data	Set argument Process data
	Convenient	Set graph type Set graph spacing
Report generation	Window	Open Close
	Data	Set argument Process data
	Convenient	Zoom Page Save as Printer setup Print

4. Consider shared functions to distribute and create new window object.

Table 4.4 Shared window object

Process	Function	Description
Data manipulation	Search	Search data by name
Transaction process	Search	Search hw_code Search hw_id by name Search hw_id by hw_code Search hw_id by hw_id
	Print	Print output document
Query process	Display Graph	Set graph type Set graph spacing
Query process and report generation	Set argument	Set section Set hardware specification Set date Set year
Share	Calendar	Calendar
	About	About the system

5. Design program specification of each window that is shown in Appendix D.

4.3 Information System for Government Inventory Management

Using the Database painter of Sybase SQL anywhere 5.0 to create database. Powersoft PowerBuilder Enterprise/32 Version 6.0 is graphical application development tools and Oasis SE Version 1.3.6.0 is help authoring tools for government inventory management system, which implements under microcomputer Pentium II 350 MHz CPU, Windows98 operating system. The information system consists of 3 major processes: 1) Data Manipulation 2)Transaction Processing, 3) Query and Report. The capability of processes is show in Table 4.5

Table 4.5 Capability of process

Process	Capability
Data manipulation	Manipulate data in these data store 1. Hardware group 2. Hardware class 3. Hardware type 4. Hardware (specification)

Process	Capability
	5. Supplier 6. Division 7. Section 8. Staff 9. Security 10. Project
Transaction process	1. Register hardware 2. Distribute / borrow hardware 3. Return hardware 4. Repair hardware 5. Receive repaired hardware 6. Dispose hardware
Query and report	A. <i>Tabular result</i> <ol style="list-style-type: none"> 1. Hardware data and status of requested HW_ID 2. Hardware data of requested specification and section 3. Hardware list in requested status , section and specification 4. List of all/specific borrowed hardware in requested return date 5. List of all/specific repaired hardware in requested return date B. <i>Graph result</i> <ol style="list-style-type: none"> 1. Summary graph of hardware quantity in department or division 2. Summary graph of hardware quantity that is distinguished by status 3. Drill down graph of hardware quantity in section C. <i>Report</i> <ol style="list-style-type: none"> 1. Hardware report (group, class, type, specification) 2. Supplier report 3. Division and section report 4. Summary of hardware registration in requested budget year 5. Summary of repair hardware 6. Summary of disposal hardware 7. Summary of hardware quantity in section 8. Summary of hardware quantity in department 9. Summary of hardware quantity in requested budget year 10. Summary of hardware quantity in requested budget year and section

Since the information system works on client/server network, many users can use it. The users are divided into 3 levels: Database Administrator, operator and general user. The privilege of each user group is shown in Table 4.6.

Table 4.6 Privilege of users

Process	Data Manipulation	Transaction Processing	Query And report
User			
DBA	✓		✓
Operator		✓	✓
General user			✓



Users access the information system through login window, which verifies user ID, password and accessible level. In addition, operator can access data of transaction processing in division and section that user are under.

By the way, the information system has an additional process for data access of branch office, which cannot communicate database server across network. The process keeps SQL commands of each transaction process, then user (operator) uses the information system to gets the SQL command into text file, which is sent to database center by electronic mail or file transfer process. DBA uses Interactive SQL to execute the text file, then "Trans data" will be updated and database has consistent data.

CHAPTER V

DISCUSSION

The information system for government inventory management for a case study of Land Development Department works on client/server network. The system consists of three major processes as follows:

- **Data manipulation:** Manipulate relative data such as supplier, hardware, staff, section and so on.
- **Transaction processing:** Support operation management, which are registration, distribution, borrowing, repair and disposal.
- **Query and report:** Process pre-defined outputs and preplanned printed reports. A user can choose to access between a particular data, which is deep in detail, or just overall data, which is convenient for comparative analysis.

Users of the information system are divided into 3 levels: Database administrator, operator and general user. All levels can access query and report menu, data manipulation menu can be accessed by database administrator level and transaction processing menu can be accessed by operator level. They access the information system through login window, which verifies user ID, password and accessible level. In addition, operator can access data of transaction processing in division and section in which user is under.

Moreover, the information system uses in many places, somewhere may not support online system. For this reason, it has an extra function to store SQL commands of transaction process into file. The file is sent and executed at database center, then the database center has up to date and consistent data.

The information system is designed and developed using data flow analysis, relational database and Object-Oriented programming. It uses the Database painter of Sybase SQL anywhere 5.0 to create database, Powersoft PowerBuilder Enterprise/32 Version 6.0 to develop application and Oasis SE Version 1.3.6.0 to create help file.

The advantages of the information system for government inventory management for a case study of Land Development Department are as follows:

- The problems of inefficiency government inventory management can be solved and improved. The information system changes operation from analog to electronics, searching from manual to digital and database from analog to digital. The information system introduces online operation and standard database, which reduces access time, increases accuracy and provides consistent data.
- There is database and information system for government inventory management.
- Material section of Land Development Department and other government organizations have information system for government inventory management to apply their inventory management system

CHAPTER VI

CONCLUSION

Increasing important role of information system in most organizations makes information system is critical to the success of every business. Land Development Department has an advantage to develop information system for government inventory management because of its computer network system. Not only Land Development Department but also other government organizations can use the information system to improve their ineffective operation of government inventory management.

In addition, the possibility of a new software development style, in which programmers stop coding everything from scratch and begin assembling applications from well-stocked catalogs of reusable software components. Object-oriented programming has clearly been accepted by the new software development style because of the benefits as follows:

- Object-oriented programs can be more easily maintained and adapted than procedural languages.
- More code can be reused from project to project when the code is object-oriented, rather than procedural.
- Object-oriented programs scale well in regard to the amount of complexity that the developer must face compared to procedural programs.

Consequently, the information system for government inventory management for a case study of Land Development Department was developed by PowerBuilder that is object-oriented programming tool. This research shows that the information system supports all above benefits especially in inheritance and encapsulation.

Data flow analysis was used in this research, it was a valuable for performing process analysis although it had to be adjusted and refined in design process. Beside, as data flow analysis have been used for a long time, this research may be guideline for developer who would like to move his old data flow diagram and structure programming to object-oriented programming. But objects in this research focus on replacing process in data flow diagram with window object.

By the way, data in the information system was designed by relational database model. Relational database is adequate for database model in this research because information needs are two-dimensional and not complex data, which can be normalized. Additional SQL-based relational database is more important characteristics that should be considered.

Recommendation

1. Develop functions of privileged operation to discard requested document, which are requisition form, borrowing form, out of order hardware list. The functions can control authorization to let the information system know who can distribute, borrow, or approve. In addition, the functions have to support electronic signature.

2. Develop this information system to OLAP (Online Analytical Processing) system that is designed for live data access and analysis. In this information system, most results in query process are summary and comparative analysis model. Developer can add statistical analysis model for viewing trends and required historical data. The OLAP system is used by analysts and managers who frequently want a higher-level aggregated view of the data.
3. Develop and link this information system to another relative system such purchasing, procurement, budget planning and so on. The combinations of systems make complete inventory management and control system, which can be developed to DSS or EIS.
4. Apply this information system to Internet/ Intranet application. PowerBuilder version 7.0 offers significant productivity enhancements and broad support for Web-based component standards, web.pb access Distributed PowerBuilder DPB objects as part of an HTTP application.

REFERENCES

1. Land Development Department (1996). Introduction to Land Development Department. Available from: <http://www.ddd.go.th>. [Accessed 1999 Apr 24].
2. Arnold J.R. Introduction to materials management. 3rd ed. USA: Prentice Hall PTR; 1998.
3. Tersine Richard J. Principles of inventory and materials management. 4th ed. USA: Prentice Hall PTR; 1994.
4. มหาวิทยาลัยสุโขทัยธรรมราช. การบริหารวัสดุและการจัดซื้อ. พิมพ์ครั้งที่ 5. กรุงเทพฯ: สำนักพิมพ์มหาวิทยาลัยสุโขทัยธรรมราช; 2533.
5. สุนา อยู่โพธิ์. การจัดซื้อและบริหารพัสดุ. กรุงเทพฯ : โรงพิมพ์ชวนพิมพ์; 2539.
6. อุดลย์ จาตุรงค์กุล. การจัดซื้อ. พิมพ์ครั้งที่ 2. กรุงเทพฯ: โรงพิมพ์มหาวิทยาลัยธรรมศาสตร์; 2538.
7. ศิริจันทร์ ทองประเสริฐ. ระบบพัสดุคงคลัง. กรุงเทพฯ: ภาควิชาวิศวกรรมอุตสาหกรรม จุฬาลงกรณ์มหาวิทยาลัย; 2538.
8. วิภา รัตตะมณี. โครงการวิเคราะห์และออกแบบระบบงานควบคุมและตรวจสอบพัสดุ. กรุงเทพฯ: ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง; 2540.
9. สำนักงบประมาณ. คู่มือการกำหนดหมายเลขพัสดุ. กรุงเทพฯ; 2537.
10. Charles Parker. Management information systems: strategy and action. 2nd ed. Singapore: McGraw-Hill; 1993.

11. Ralph M. Stair. Principles of information systems: a managerial approach. 2nd ed.
USA: boyd&fraser publishing company; 1996.
12. Bakman Alex. How to deliver client/server applications that work. USA: Prentice Hall PTR; 1995.
13. Martin, James. Client/server database : enterprise computing / James Martin, Joe Leben. USA: Prentice Hall PTR; 1995.
14. James A. Senn. Analysis and design of information systems. 2nd ed., Singapore: McGraw-Hill; 1989.
15. Larry T. Vaughn. Client/server system design and implementation. Singapore: McGraw-Hill; 1994.
16. The University of Texas at Austin(1996). Normalization. Available from:
<http://www.utexas.edu/cc/dbms/utinfo/reemode/normal1.html>.
[Accessed 1999 Apr 22].
17. Roger S. Pressman. Software engineering a practitioner's approach. 4th ed.
Singapore : McGraw-Hill; 1997.
18. Ashok Ramachandran (1998). PowerBuilder Primers. Available from:
<http://www.ashok.pair.com/pbprimer.html>. [Accessed 1999 Apr 22].
19. Informix Software Inc.(1999). Informix. Available from: <http://www.informix.com>.
[Accessed 1999, Apr, 23].
20. Sybase Inc.(1999). Sybase Available from: <http://www.sybase.com>.
[Accessed 1999, Apr, 23].

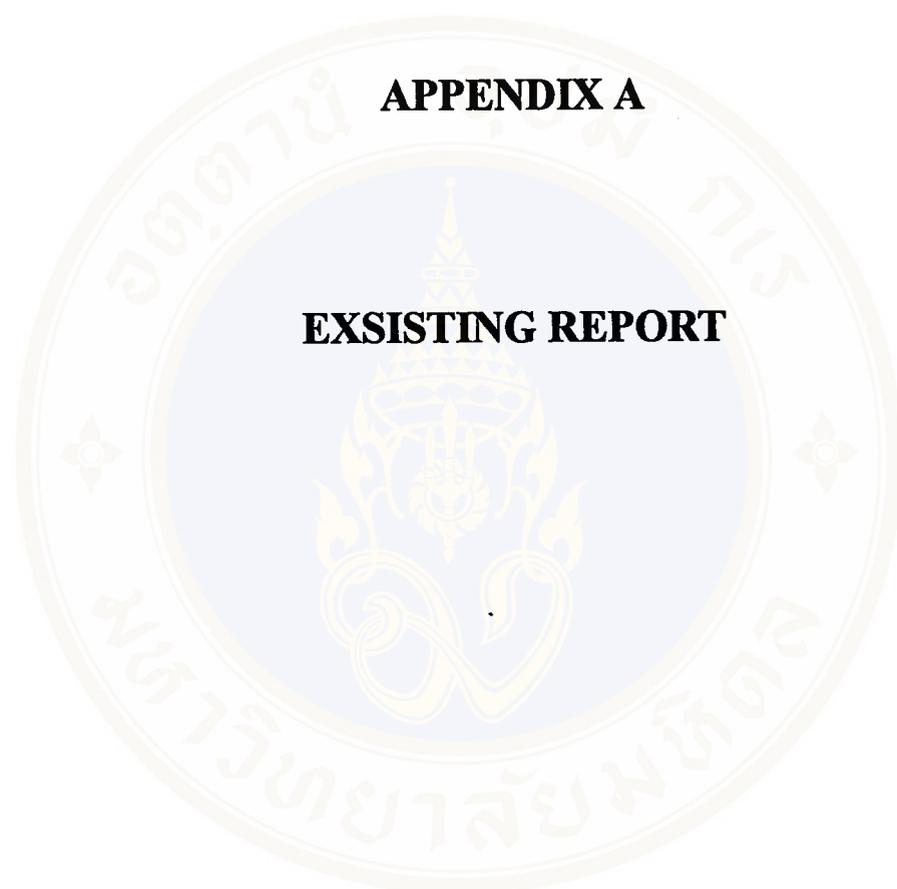
21. Nathavipa Phasuk, Voramon Uronkarn. Management of equipment data of Industrial Engineering Department by Microsoft Access.

Bangkok: Faculty of Engineering, Mahidol University; 1996.

22. วิเลิศ เลิศบัณฑิตกุล. ระบบสารสนเทศสำหรับคลังยาและเวชภัณฑ์ของโรงพยาบาลขนาดใหญ่

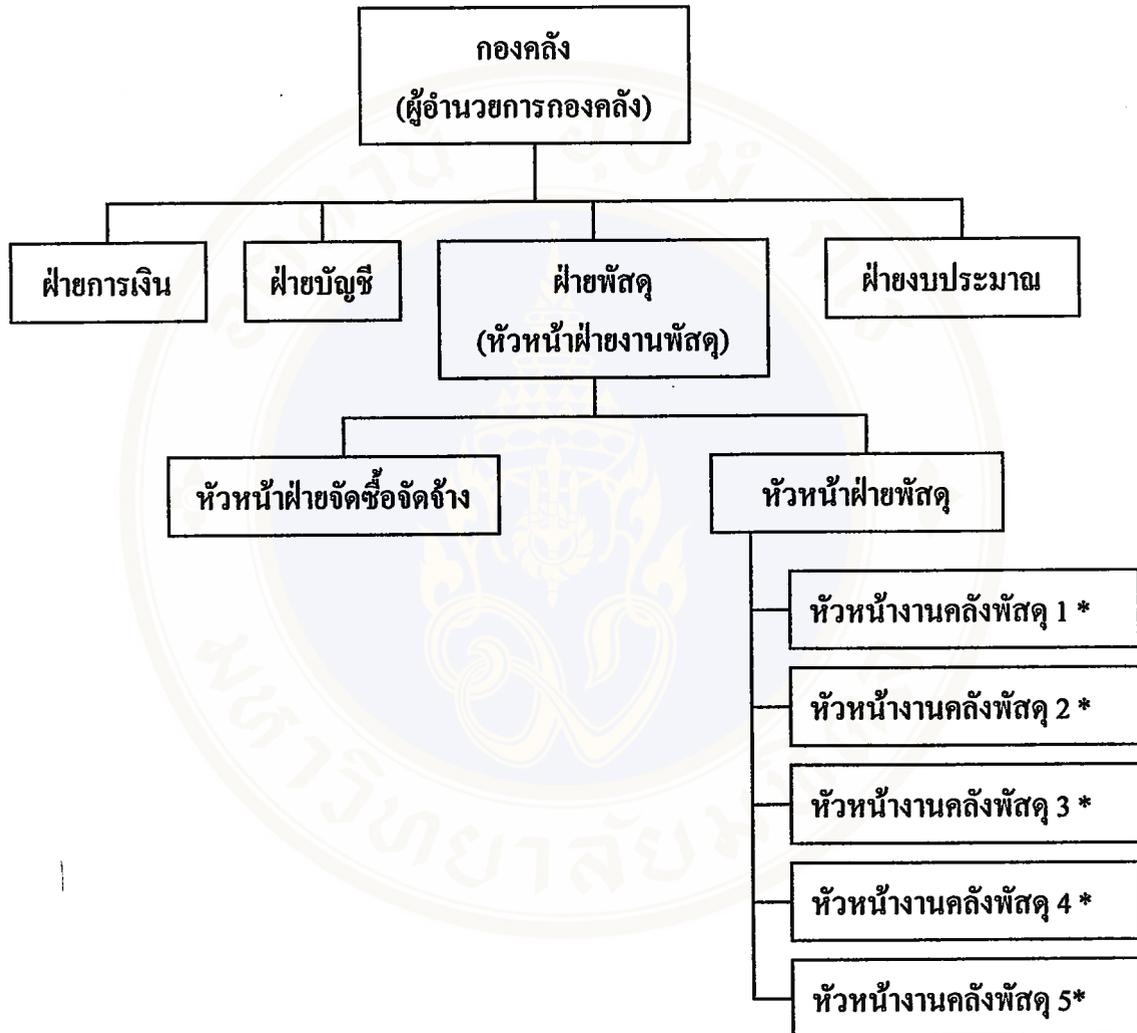
[วิทยานิพนธ์ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์].





ฝ่ายพัสดุของกรมพัฒนาที่ดินแบ่งออกเป็น 2 ประเภทคือ

1. ส่วนกลาง แสดงดังแผนผังด้านล่าง
2. ส่วนต่างจังหวัด จะเป็นส่วนหนึ่งของฝ่ายบริหารทั่วไป



* มีการแบ่งส่วนรับผิดชอบกองที่ดูแลคนละ 1-2 กอง

แผนผังกองคลังของกรมพัฒนาที่ดิน (ส่วนกลาง)

ทะเบียนครุภัณฑ์

ส่วนราชการ.....

หน่วยงาน.....

แผนที่.....

ประเภท.....ชื่อหรือชนิดครุภัณฑ์.....

วัน เดือน ปี	เลขที่ หรือรหัส	ชื่อ ชนิด แบบ ขนาด และ ลักษณะ	หมายเลข และ ทะเบียน	ราคา ต่อ หน่วย (บาท)	วิธี การได้ มา	เลขที่ เอกสาร	ใช้ ประจำ ที่	หัก ฐาน การ จ่าย	รายการ เปลี่ยน แปลง	เลขที่ เอกสาร	หมายเหตุ

Registration document

ครุภัณฑ์

รหัส.....

ชื่อสิ่งของ.....

ขนาดลักษณะ.....

บริษัทที่ขาย.....

หมายเลขประจำครุภัณฑ์

ลำดับ ที่	รายการยืม				รายการส่งคืน			หมายเหตุ
	วัน เดือน ปี	ชื่อผู้ยืม	ชื่อผู้จ่าย	ชื่อผู้รับของ	วัน เดือน ปี	ชื่อผู้ส่งคืน	ชื่อผู้รับคืน	

Registration document of each hardware

สรุปบัญชีหมวดค่าครุภัณฑ์ จำแนกตามหน่วยงาน
กรมพัฒนาที่ดิน

หน้าที 1

กอง สำนักงานพัฒนาที่ดินเขต 1

พิมพ์วันที่ 12/10/41

ลำดับ	ฝ่าย	รายการ	จำนวนที่ควร จะมี	จำนวนที่ ได้รับทั้งหมด	ชั่งขาด	หมายเหตุ
1	สำนักงานเขต	รถบรรทุก น้ำหนักบรรทุกไม่ต่ำกว่า 1 ตัน เครื่องยนต์ดีเซลช่วงยาว	0	4	-4	
2	ฝ่ายวิชาการ	ตู้เหล็กเก็บเอกสาร 2 บาน	0	8	-8	
		ตู้เหล็กเก็บเอกสาร 4 ลินชัก	0	4	-4	
		เครื่องพ่นยาแบบใช้แรงลม ขนาดเครื่องยนต์ไม่ต่ำกว่า 3 แรงม้า	0	4	-4	
		เครื่องสูบน้ำใช้เครื่องเบนซิน สูบน้ำไม่น้อยกว่า 1,100 ลิตร/นาที	0	2	-2	
		เทปวัดระยะขนาดกว้าง 13 มม. ความยาวไม่น้อยกว่า 50 เมตร	0	4	-4	
3	สถานีพัฒนาที่ดินปทุมธานี	กล้องระดับ ชนิดอัตโนมัติพร้อมขาตั้ง	0	4	-4	
		กล้องวัดมุมชนิดธรรมดา	0	4	-4	
		ตู้เหล็กเก็บเอกสาร 2 บาน	0	8	-8	
		
		รวม		332	-332	

Summary of hardware quantity in department

สรุปบัญชีหมวดค่าครุภัณฑ์ ที่ดินและสิ่งก่อสร้าง จำแนกตามปีงบประมาณ
กรมพัฒนาที่ดิน

ประจำปีงบประมาณ 2540

หน้าที่ 1
พิมพ์วันที่ 12/10/41

หน่วยงาน(กอง)	ฝ่าย	หมวด	รายการ	จำนวน
สำนักงานพัฒนาที่ดินเขต 1	สถานีพัฒนาที่ดินจังหวัดนครปฐม	หมวดค่าครุภัณฑ์	ครุภัณฑ์สำนักงาน	
			ตู้เหล็กเก็บเอกสาร 2 บาน	2
			ตู้เหล็กเก็บเอกสาร 4 ลินชัก	4
			พัดลมแบบตั้งพื้น ขนาด 16 นิ้ว(400 มม.)	3
โต๊ะทำงานพร้อมเก้าอี้ ระดับ 3-6	3			
สำนักงานพัฒนาที่ดินเขต 1	สถานีพัฒนาที่ดินนครนายก	หมวดค่าครุภัณฑ์	ครุภัณฑ์สำนักงาน	
			ตู้เหล็กเก็บเอกสาร 4 ลินชัก	1
			โต๊ะทำงานพร้อมเก้าอี้ ระดับ 3 -6	2
สำนักงานพัฒนาที่ดินเขต 1	สถานีพัฒนาที่ดินปทุมธานี	หมวดค่าครุภัณฑ์	ครุภัณฑ์ยานพาหนะและขนส่ง	
			รถบรรทุก น้ำหนักบรรทุกไม่ต่ำกว่า 1 ตัน เครื่องยนต์ดีเซลช่วงยาว	1
สำนักงานพัฒนาที่ดินเขต 1	สถานีพัฒนาที่ดินปทุมธานี	หมวดค่าครุภัณฑ์	ครุภัณฑ์สำนักงาน	
			ตู้เหล็กเก็บเอกสาร 2 บาน	4
			ตู้เหล็กเก็บเอกสาร 4 ลินชัก	3
			โต๊ะทำงานพร้อมเก้าอี้ ระดับ 1-2	2
			โต๊ะทำงานพร้อมเก้าอี้ ระดับ 3-6	2

Summary of hardware quantity in requested budget year

รายงานตรวจนับครุภัณฑ์คงเหลือประจำปี 2541
เพียงวันที่ 30 กันยายน 2541 สำนักงานพัฒนาที่ดินเขต 11

หน้าที 1

ฝ่ายบริหารทั่วไป

พิมพ์วันที่ 12/10/41

ลำดับ ที่	ประเภทครุภัณฑ์	ยี่ห้อ	แบบ ชนิด ขนาด	หมายเลข เครื่อง	หมายเลขทะเบียน	วัน เดือน ปี	ยอดคง เหลือ	บันทึกคณะกรรมการ			หมายเหตุ	รหัส กอง	รหัส ครุภัณฑ์
								นับได้	เกิน	ขาด			
	ครุภัณฑ์ยานพาหนะและขนส่ง												
	(รถยนต์นั่ง)												
1	รถยนต์นั่ง	TOYOTA	โฟร์วิล	8ย.3303		18 กย.41	1 คัน	1 คัน	-	-			
2	รถยนต์นั่ง		แลนค์โรเวอร์	6ม-4116		18 กย.41	1 คัน	1 คัน	-	-			
3	รถยนต์นั่ง		แลนค์โรเวอร์	9ม-0871		18 กย.41	1 คัน	1 คัน	-	-			
	ครุภัณฑ์สำนักงาน												
1	โต๊ะทำงานพร้อมเก้าอี้ ระดับ1-2	-	60x1.20x75	-		28 กค.27	1 ชุด	1 ชุด	-	-	ชำรุดใช้การไม่ได้		
2	โต๊ะทำงานพร้อมเก้าอี้ ระดับ1-2	-	60x1.20x75	-		28 กค.27	1 ชุด	1 ชุด	-	-	ชำรุดใช้การไม่ได้		
3	โต๊ะทำงานพร้อมเก้าอี้ ระดับ1-2	-	1.5x2	-		4 เมอ.32	1 ชุด	1 ชุด	-	-			
4	โต๊ะตั้งวิทยุ	-	-	-		23 ธค.31	1 ชุด	1 ชุด	-	-			
5	โต๊ะพิมพ์ดีด พร้อมเก้าอี้	-	.05x1.20x60	-		23 ธค.31	1 ชุด	1 ชุด	-	-	ชำรุดใช้การไม่ได้		
6	โต๊ะหมู่บูชา		ชนิดหมู่ 9 หน้า 8"	-		30 ธค.30	1 ชุด	1 ชุด	-	-			
7	โต๊ะพิมพ์ดีด พร้อมเก้าอี้	-	-	-		5 มค.31	1 ชุด	1 ชุด	-	-			

Summary of hardware registration

รายงานตรวจนับครุภัณฑ์คงเหลือประจำปี 2539
เพียงวันที่ 30 กันยายน 2539 สถานีพัฒนาที่ดินเชียงราย สำนักงานพัฒนาที่ดินเขต 7

หน้าที่ 1

พิมพ์วันที่ 12/10/41

ครุภัณฑ์โฆษณาและเผยแพร่

ลำดับ ที่	ประเภทครุภัณฑ์	ยี่ห้อ	แบบ ชนิด ขนาด	หมายเลข เครื่อง	หมายเลขทะเบียน	วัน เดือน ปี	ยอดคง เหลือ	บันทึกคณะกรรมการ			หมายเหตุ	รหัส กอง	รหัส ครุภัณฑ์
								นับได้	เกิน	ขาด			
1	เครื่องฉายภาพยนตร์	โอคูชิน	รุ่น 3C 10 เอ็นเอช 16 มม.	54579	6730-003-0026ศพด.ชร.01	15 ตค.26	1 เครื่อง	1	-	-			
2	เครื่องฉายข้ามศีรษะ	เคบิน	แบบใช้แทนฉายขนาด 10x10 ไร้ไฟ AC	0353371 2	6730-001-0026ศพด.ชร.01	15 ตค.26	1 ชุด	1	-	-			
3	กล้องถ่ายรูป	ยาสึก้า	ขนาดเลนส์ 1:1.8FX	1075582	5720-005-0028ศพด.ชร.01	24 กย.28	1 ชุด	1	-	-			
4	กล้องถ่ายรูป	แคนนอน	AF MII	4754439	6720-005-0029ศพด.ชร.01	29 เมย.29	1 ชุด	1	-	-			
5	เครื่องบันทึกเสียง	มิตซูบิชิ	แบบตลับตั้งโต๊ะระบบสเตอริโอเล่นเทปได้	8931	7450-001-0026ศพด.ชร.02	15 ตค.26	1 เครื่อง	1	-	-			
6	ไมโครโฟน	NEC	-		7450-008-0026ศพด.ชร.01-06	15 ตค.26	6 ชุด	6	-	-			
7	ไมโครโฟน	ไดนามิกส์	ชนิดตั้งโต๊ะ		7450-007-0026ศพด.ชร.07-09	15 ตค.26	3 ชุด	3	-	-			
8	เครื่องขยายเสียง	รอยแอล	ขนาดกำลังส่ง 50 W	6615	7450-007-0026ศพด.ชร.01-04	15 ตค.26	1 เครื่อง	1	-	-			
9	โทรโข่ง		แบบสะพานไหล มีไมโครโฟนในตัว		7450-007-0035ศพด.ชร.07-10	29 กย.35	4 ชุด	4	-	-			

Summary of hardware registration(other section)

สรุปบัญชีครุภัณฑ์ ที่ดินและสิ่งก่อสร้าง

กรมพัฒนาที่ดิน

หน้าที่ 1

พิมพ์วันที่ 12/10/41

ลำดับ	รายการ	จำนวนที่มีอยู่	หมายเหตุ
1	รถฟาร์มแทรกเตอร์ขนาดเครื่องยนต์ไม่ต่ำกว่า 75 แรงม้า	3	
2	รถฟาร์มฯ เครื่องยนต์ไม่ต่ำกว่า 85 แรงม้า ชนิดขับเคลื่อน 2 ล้อ	3	
3	รถฟาร์มฯ เครื่องยนต์ไม่ต่ำกว่า 85 แรงม้า ชนิดขับเคลื่อน 4 ล้อ	3	
4	รถไถ	3	
5	รถไถชนิดเดินตาม ขนาดเครื่องยนต์ไม่ต่ำกว่า 10 แรงม้า	3	
6	เครื่องพ่นยา	1	
7	เครื่องพ่นยาแบบสูบโยก ขนาดถังจุน้ำยาได้ไม่น้อยกว่า 16	4	
8	เครื่องพ่นยาแบบใช้แรงลม ขนาดเครื่องยนต์ไม่ต่ำกว่า 3 แรงม้า	19	
9	เครื่องพ่นยาแบบใช้แรงดันของเหลวขนาดไม่ต่ำกว่า 1.50 แรงม้า	9	
10	เครื่องขัง	2	
11	เครื่องขังชนิดที่ 4 แบบมีตุ้มถ่วงชนิดเสากลมขนาดพิกัด 200 กก.	1	
12	เครื่องขังชนิดที่ 4 แบบมีตุ้มถ่วงชนิดเสาเหลี่ยมขนาดพิกัด 200 กก.	1	
13	เครื่องสูบน้ำ	3	
14	เครื่องสูบน้ำชนิดเครื่องเบนซิน สูบน้ำได้ไม่น้อยกว่า 450 ลิตร/นาที	3	
15	เครื่องสูบน้ำชนิดเครื่องเบนซินสูบน้ำได้ไม่น้อยกว่า 1,100 ลิตร/นาที	5	
16	เครื่องสูบน้ำใช้เครื่องยนต์ดีเซลสูบน้ำได้ 1,750 ลิตร/นาที	6	
17	เครื่องสูบน้ำใช้มอเตอร์ไฟฟ้าสูบน้ำได้ 450 ลิตร/นาที	1	
18	เครื่องสูบน้ำใช้มอเตอร์ไฟฟ้าสูบน้ำได้ 1,130 ลิตร/นาที	6	
19	ใบมีดหน้า	2	
20	จอร์รับภาพ	2	
21	จอร์รับภาพ ขนาด 125*125 นิ้ว	14	
22	จอร์รับภาพ ขนาด 175*175 นิ้ว	6	
23	เครื่องฉายสไลด์แบบถาดกลมแนวนอน	8	
24	เครื่องฉายภาพข้ามศีรษะ	30	
	
	รวม	XXX	

Summary of hardware quantity

สรุปบัญชีหมวดค่าครุภัณฑ์ จำแนกตามหน่วยงาน/รายการ

กรมพัฒนาที่ดิน

หน้าที่ 1

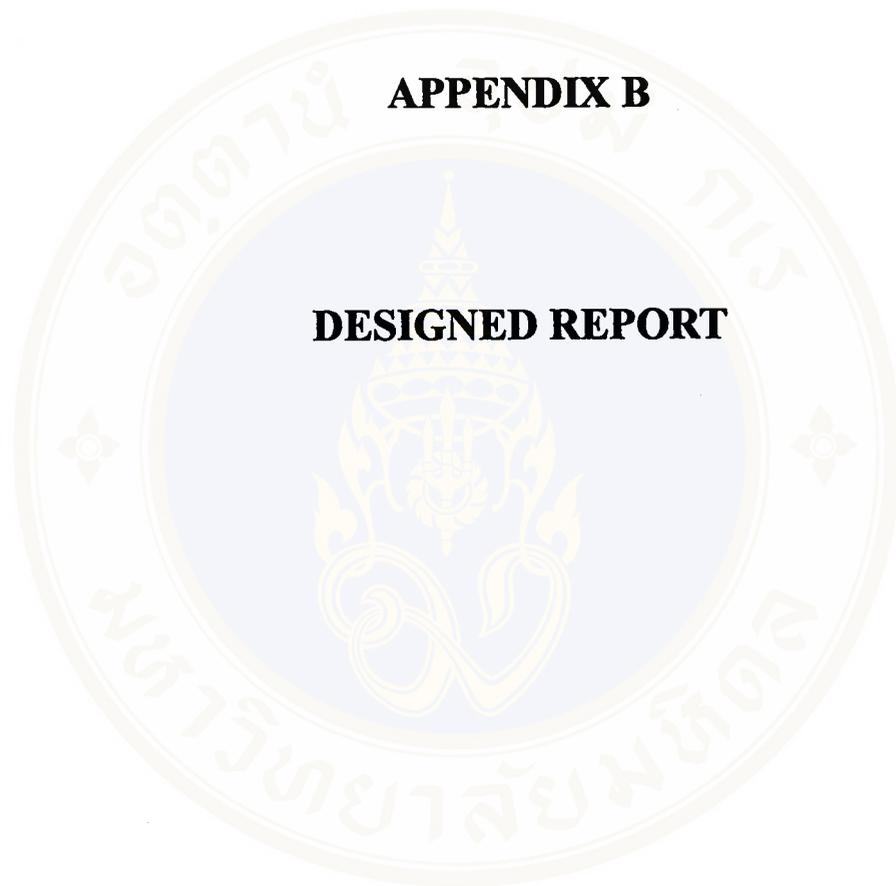
พิมพ์วันที่ 12/10/41

กอง สำนักงานพัฒนาที่ดินเขต 1

ฝ่าย สถานีพัฒนาที่ดินปทุมธานี

ลำดับ	รายการ	ปีที่ได้รับ	จำนวน	รวม	หมายเหตุ
1	รถบรรทุก	2537	1	1	
	รวม			1	
2	รถบรรทุกน้ำหนักไม่ต่ำกว่า 1 ตัน เครื่องยนต์ดีเซลช่วงยาว	2537	1		
		2538	1		
	รวม			2	
	รวมครุภัณฑ์ยานพาหนะและขนส่งทั้งสิ้น			3	
3	โต๊ะทำงานพร้อมเก้าอี้ ระดับ 1-2	2538	2	2	
4	โต๊ะทำงานพร้อมเก้าอี้ระดับ 3-6	2538	2		
		2540	4		
				6	
5	ตู้เหล็กเก็บเอกสาร 2 บาน	2538	4	4	
6	ตู้เหล็กเก็บเอกสาร 4 ลิ้นชัก	2538	3	3	
...

Summary of hardware quantity in requested budget year and section



วันที่พิมพ์ 07/02/2543		
รายงานกลุ่มครุภัณฑ์		
รหัสกลุ่มครุภัณฑ์	ชื่อกลุ่มครุภัณฑ์	คำอธิบายเพิ่มเติม
10	ครุภัณฑ์อาวุธ	
23	ครุภัณฑ์ยานพาหนะและขนส่ง	ยานพาหนะพื้นดิน ยานยนต์ รถพ่วง และ จักรยาน
38	ครุภัณฑ์ก่อสร้าง	อุปกรณ์ก่อสร้าง การถมดิน การขุดดินและการซ่อมบ
65	ครุภัณฑ์การแพทย์	
66	ครุภัณฑ์สำรวจ	
87	ครุภัณฑ์โฆษณาและเผยแพร่	
71	ครุภัณฑ์สำนักงาน	
73	ครุภัณฑ์งานบ้านงานครัว	

Hardware group report

วันที่พิมพ์ 07/02/2543

รายงานประเภทครุภัณฑ์แยกตามกลุ่ม

รหัสประเภทครุภัณฑ์	ชื่อ	ชื่อที่ใช้เรียกแทน
รหัสกลุ่มครุภัณฑ์ 10 ครุภัณฑ์อาจารย์		
1010	เป็นที่นิยมนานเกินกว่า 30 มม ถึง 75 มม	เป็น 30-75 มม
1005	เป็นที่นิยมนานถึง 30 มม	เป็น 30 มม
รหัสกลุ่มครุภัณฑ์ 23 ครุภัณฑ์ยานพาหนะและขนส่ง		
2330	รถพ่วง	รถพ่วงเครื่องพร้อมและแคว่ก
2320	รถบรรทุกและรถลากจูงใช้ล้อ	
2305	ยานพาหนะใช้บนพื้นดิน	
2340	รถจักรยานยนต์ รถจักรยาน	

Hardware class report

รายงานชนิดครุภัณฑ์แยกตามประเภท		วันที่พิมพ์ 07/02/2543
รหัสชนิดครุภัณฑ์	ชื่อชนิดครุภัณฑ์	คำอธิบายเพิ่มเติม
กลุ่มครุภัณฑ์ 10 ครุภัณฑ์อาวุธ		
ประเภทครุภัณฑ์ 1006	ปืนที่มีขนาดถึง 30 มม	
002	ปืนพกกลไก	
001	ปืนพกอัตโนมัติ	
กลุ่มครุภัณฑ์ 23 ครุภัณฑ์ยานพาหนะและขนส่ง		
ประเภทครุภัณฑ์ 2305	ยานพาหนะใช้บนพื้นดิน	
...	...	

Hardware type report

รายงานครุภัณฑ์แยกตามชนิด				วันที่พิมพ์ 07/02/2543
ชนิดครุภัณฑ์	ชื่อ/ชนิด รหัสครุภัณฑ์	หน่วยนับ	ราคา	หมายเหตุ
กลุ่มครุภัณฑ์ 23 ครุภัณฑ์ยานพาหนะและขนส่ง				
<u>ยานยนต์โดยสาร</u>				
รถโดยสารตู้				
	ขนาด 12 ที่นั่ง (ดีเซล) 2310-004-0001	คัน	700,000.00	
รถยนต์นั่งแก๊ส				
	รถประจำตำแหน่งอธิบดี 2310-002-0002	คัน	1,200,000.00	ของปลัดกระทรวง, เอกอัครราชทูตประจำกรม รองเลขาธิการนายกรัฐมนตรีฝ่ายการเมือง, เ
	...			- - -

Hardware specification report

วันที่พิมพ์ 07/02/2543

รายงานกอง/สำนักงาน

รหัส	ชื่อ	ชื่อย่อ
000	ส่วนกลาง	ส่วนกลาง
101	สำนักงานเลขานุการกรม	สสท.
102	กองคลัง	กค.
103	กองการเจ้าหน้าที่	กคจ.
104	กองช่าง	กช.
105	กองแผนงาน	กผง.
106	กองแผนที่และการพิมพ์	กผพ.
107	กองวางแผนการใช้ที่ดิน	กวม.

Division report

วันที่พิมพ์ 07/02/2543

รายงานหน่วยงานแบ่งตามกอง/สำนักงาน

รหัสหน่วยงาน	ชื่อหน่วยงาน	ชื่อย่อ
รองสำนักงาน 000	ส่วนกลาง	
000	ส่วนกลาง	ส่วนกลาง
001	หน่วยงานตรวจสอบภายใน	หน่วยงานตรวจสอบภายใน

รองสำนักงาน 101 สำนักงานเลขานุการกรม

000	สำนักงานเลขานุการกรม	สสท.
001	ฝ่ายสารบรรณ	ฝ่ายสารบรรณ
002	ฝ่ายนิติการ	ฝ่ายนิติการ
003	ฝ่ายช่วยส่วนราชการและประสานราชการ	ฝ่ายช่วยส่วนราชการ
004	ฝ่ายเผยแพร่และประชาสัมพันธ์	ฝ่ายเผยแพร่ฯ

Section report

รายชื่อผู้ประกอบการบริษัทผู้จำหน่าย		พิมพ์วันที่ 07/02/2543	
รหัส	ชื่อ	ที่อยู่	โทรศัพท์ โทรสาร
0001	คลองเตย	3658 ซาดชา 7 ซ. อ.หะซาม 4 แขวงคลองเตย เขต ห้วยขวาง กรุงเทพฯ 10110	2495060 1235444
0002	งามวงศ์วาน	อาคารชุมชนสายโทรศัพท์ที่งามวงศ์วาน อ.งามวงศ์วาน จ. .เมือง นนทบุรี กทม 11000	โทรสาร 1234567
0003	แจ้งวัฒนะ	688/23 ทดถึสี่แควฯ 3 อ.แจ้งวัฒนะ คลองถนน บางเขน กทม 10220	โทรสาร 1111111
0004	บางแค	36551-63 จากซาดสี่คูโบ อ.บางแคเขต แขวงท่าชะ เขมาบางกอกใหญ่ กทม 10600	โทรสาร 11111
0005	บางนา	1195 หมู่ 8 จากตราบางนาธานี อ.บางนาเขต แขวง นา อ.บางนา-ตลาด กม 3 กทม 10260	โทรสาร 11111111

Supplier report

วันที่พิมพ์ 07/02/26

รายการทะเบียนครุภัณฑ์ประจำปีงบประมาณ 2541

กอง/สำนักงาน กองคลัง		ฝ่าย/สำนักงาน งานสุภาพ					
ประเภทครุภัณฑ์	คุณลักษณะ/ชื่อหรือแบบ/ขนาด	หน่วยนับ	เลขทะเบียนครุภัณฑ์ หมายเลขเครื่อง	วันที่มีทะเบียน	ราคาหน่วย	วิธีการที่ได้มา	ใช้ประจำที่
กลุ่มครุภัณฑ์ เครื่องมือสำนักงาน							
<u>เครื่องพิมพ์สำนักงาน</u>							
เก้าอี้ชนิดต่างๆ	เก้าอี้ผู้มาติดต่อ เก้าอี้ไม้ขาเหล็ก	ตัว	7110-006-41-สก-2/24 ไม่มี	29/10/2542	250.00	ซื้อ	ห้องสุภาพ
		ตัว	7110-006-41-สก-2/25 ไม่มี	29/10/2542	250.00	ซื้อ	
		ตัว	7110-006-41-สก-2/26 ไม่มี	29/10/2542	250.00	ซื้อ	ฟฟฟ

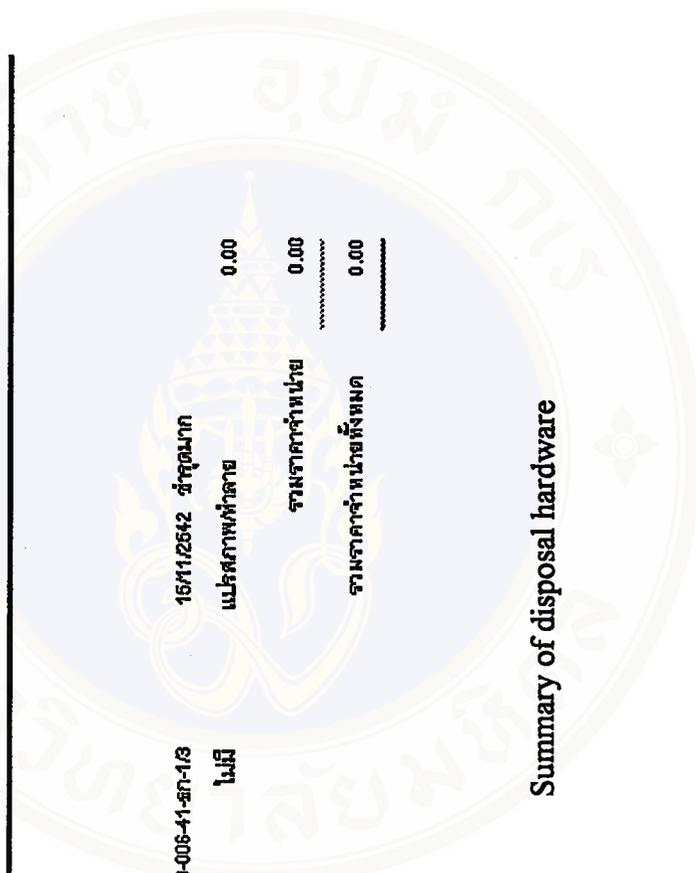
Summary of hardware registration in request budget year

กองช่าง ไม้ทำงาน กองคลัง		รายการครุภัณฑ์ส่งซ่อมแยกตามหน่วยงานประจำปี 42		ฝ่ายสถาปนา งานธุรการ		วันที่พิมพ์ 07/02/2542	
ประเภทครุภัณฑ์	คุณสมบัติหรือชื่อแบบขนาด	หมายเลขครุภัณฑ์	วันที่ออกใบส่งซ่อม	บริษัทหน่วยงานที่ซ่อม	ราคาซ่อม	วันที่ส่งซ่อม	กำหนดวงเงิน
		หมายเลขเครื่อง	สาเหตุการซ่อม				
กลุ่มครุภัณฑ์ ครุภัณฑ์สำนักงาน							
<u>เครื่องพิมพ์สำนักงาน</u>							
เครื่องพิมพ์ต่าง ๆ	เครื่องพิมพ์ดีด	7110-008-41-สก-1/4	21/12/42	คลองเตย	20.00	21/12/2542	
	เครื่องพิมพ์ดีด	ไม่มี		ช่างสุต		29/12/2542	
เครื่องพิมพ์ต่าง ๆ	เครื่องพิมพ์ดีด	7110-008-41-สก-2/21		คลองเตย	20.00	20/12/2542	
	เครื่องพิมพ์ดีด	ไม่มี		ช่างสุต		23/12/2542	
เครื่องพิมพ์ต่าง ๆ	เครื่องพิมพ์ดีด	7110-008-41-สก-2/24		คลองเตย	20.00	20/12/2542	
	เครื่องพิมพ์ดีด	ไม่มี		ช่างสุต		23/12/2542	

Summary of repair hardware

รายงานสรุปภัณฑ์ที่จำหน่ายประจำปี 42		ฝ่าย/สถาบันงาน งานธุรการ	
กอง/สำนักงาน กองคลัง	คุณลักษณะ/ชื่อแบบ/ขนาด	วันที่จำหน่าย วิธีการจำหน่าย	ส่วนราชการจำหน่าย ราคาจำหน่าย
ประเภทครุภัณฑ์	เลขทะเบียนครุภัณฑ์ หมายเลขเครื่อง		
กลุ่มครุภัณฑ์ ครุภัณฑ์สำนักงาน			
เครื่องขยายเสียงสำนักงาน			
เก้าอี้ชนิดต่างๆ	เก้าอี้ผู้มาติดต่อ เก้าอี้งานสำหรับประชาชนสีเขียว	15/11/2542 ชำรุดมาก แบรคสภาพทำลาย	0.00
		ไม่มี	0.00
		7110-006-41-สก-1/3	0.00
			รวมราคาจำหน่าย
			รวมราคาจำหน่ายทั้งหมด
			0.00

Summary of disposal hardware





สรุปบัญชีครุภัณฑ์		วันที่พิมพ์ 07/02/2543
ประเภทครุภัณฑ์	แบบ ชนิด ขนาด	จำนวนที่มีอยู่ หน่วยนับ
ครุภัณฑ์ยานพาหนะและขนส่ง		12
รถจักรยาน 2 ล้อ	ขนาดวงล้อ 26 นิ้ว	6 คัน
รถจักรยานยนต์	ขนาด 90 ซีซี	3 คัน
รถจักรยานยนต์	ขนาด 110 ซีซี	3 คัน
ครุภัณฑ์งานบ้านงานครัว		2
ตู้เย็น	ขนาด 5 คิวบิกฟุต	1 ตู้
เครื่องกรองน้ำแบบต่างๆ	ใส่กรองเซรามิค	1 เครื่อง
ครุภัณฑ์โฆษณาและเผยแพร่		2
กล้องถ่ายภาพ	ธรรมดาขนาดเลนส์ 1:1.8	1 กล้อง
จอรับภาพ	ขนาด 125*126 ซม	1 จอ
โทรทัศน์สี	ขนาด 20 นิ้ว	1 เครื่อง
รวม		81

Summary of hardware quantity in department

สรุปบัญชีครุภัณฑ์จำแนกตามหน่วยงาน

วันที่พิมพ์ 07/02/2543

กองสำนักงาน กองคลัง

ฝ่ายประเภทครุภัณฑ์	แบบ ชนิด ขนาด	จำนวนที่มีอยู่ หน่วยนับ
งานธุรการ		
ครุภัณฑ์สำนักงาน		49
ผู้เก็บแม่พิมพ์	ผู้เก็บแม่พิมพ์	1 ผู้
โต๊ะชนิดต่างๆ	โต๊ะทำงานระดับ 1-2	3 ชุด
เก้าอี้ชนิดต่างๆ	เก้าอี้ผู้นั่งติดต่อ	45 ตัว
ครุภัณฑ์สำนักงาน		4
ตู้เสื้อผ้า	ขนาด 2 บาน	2 ตู้
ตู้เหล็ก	ขนาด 2 บาน	2 ตู้
ครุภัณฑ์โฆษณาและเผยแพร่		1
โทรทัศน์สี	ขนาด 20 นิ้ว	1 เครื่อง
ครุภัณฑ์งานบ้านงานครัว		2
ตู้เย็น	ขนาด 6 คิวบิกฟุต	2 ตู้

Summary of hardware quantity in section

สรุปบัญชีครุภัณฑ์จำแนกตามปีงบประมาณ		วันที่พิมพ์ 07/02/2543
ปี	หมวดย่อยรายการ	จำนวนที่มีอยู่ หน่วยนับ
กอง กองคลัง งานธุรการ		14
	<u>ครุภัณฑ์สำนักงาน</u>	3
	ผู้เก็บแผนที่ ผู้เก็บแผนที่	1 ผู้
	โต๊ะชนิดต่างๆ โต๊ะทำงานระดับ 1-2	2 ชุด
	<u>ครุภัณฑ์ยานพาหนะและขนส่ง</u>	3
	รถจักรยานยนต์ ขนาด 110 ซีซี	3 คัน
	<u>ครุภัณฑ์สำนักงาน</u>	2
	ผู้เหล็ก ขนาด 2 มาณ	2 ผู้
	<u>ครุภัณฑ์ยานพาหนะและขนส่ง</u>	1
	รถจักรยานยนต์ ขนาด 90 ซีซี	1 คัน
	<u>ครุภัณฑ์สำนักงาน</u>	5
	เครื่องโทรศัพท์ แบบ single line	5 เครื่อง
	รวม	14

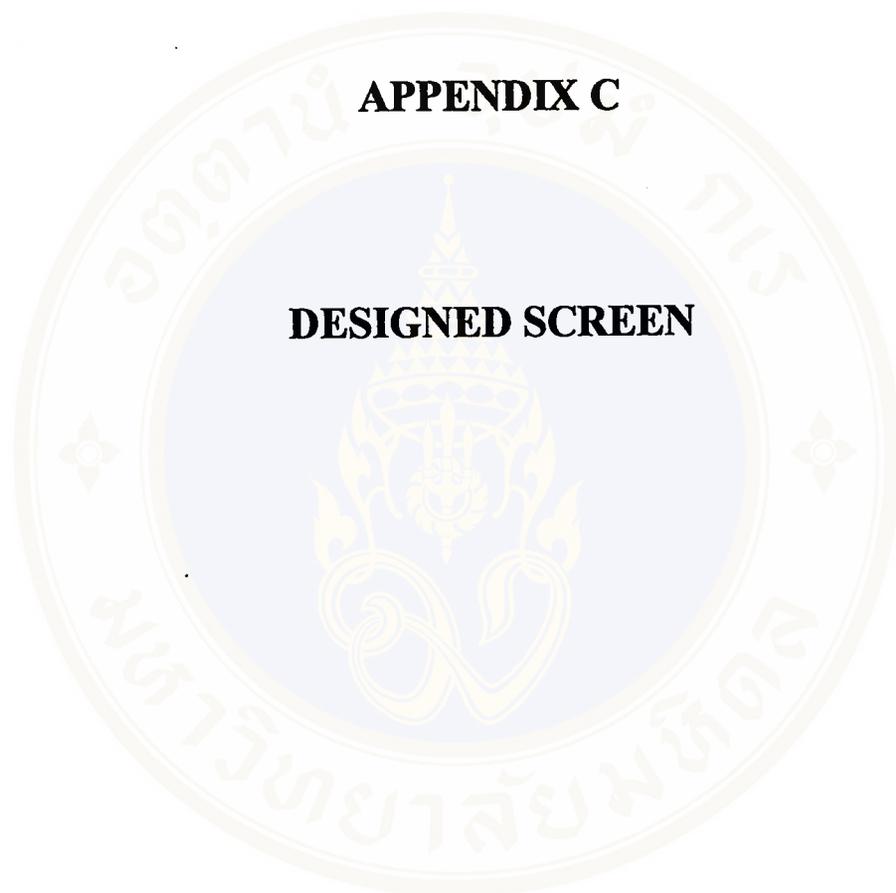
Summary of hardware quantity in requested budget year

สรุปบัญชีครุภัณฑ์จำแนกตามหน่วยงาน		วันที่พิมพ์ 07/02/2543	
กองสำนักงาน กองคลัง		ฝ่าย งานธุรการ	
ประเภทครุภัณฑ์	แบบ ชนิด ขนาด	ปีที่ได้รับ	จำนวนที่มีอยู่ หน่วยนับ
ครุภัณฑ์สำนักงาน			49
ตู้เก็บแม่พิมพ์	ตู้เก็บแม่พิมพ์	2543	1 ตู้
โต๊ะชนิดต่างๆ	โต๊ะทำงานระดับ 1-2	2542	3 ชุด
		2543	1
		2543	2
เก้าอี้ชนิดต่างๆ	เก้าอี้ผู้มาติดต่อ	2542	45 ตัว
			45
ครุภัณฑ์ยานพาหนะและขนส่ง			3
รถจักรยานยนต์	ขนาด 110 ซีซี	2543	3 คัน
			3
ครุภัณฑ์โฆษณาและเผยแพร่			1
จอรับภาพ	ขนาด 125*125 ซม	2542	1 จอ
			1

Summary of hardware quantity in requested budget year and section

APPENDIX C

DESIGNED SCREEN



1.Data manipulation

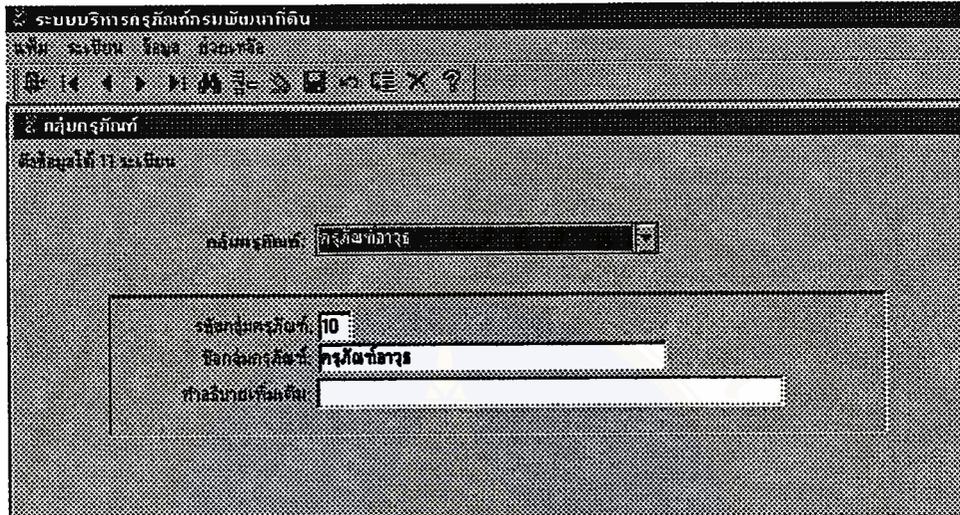


Figure C.1 w_db_hw_group

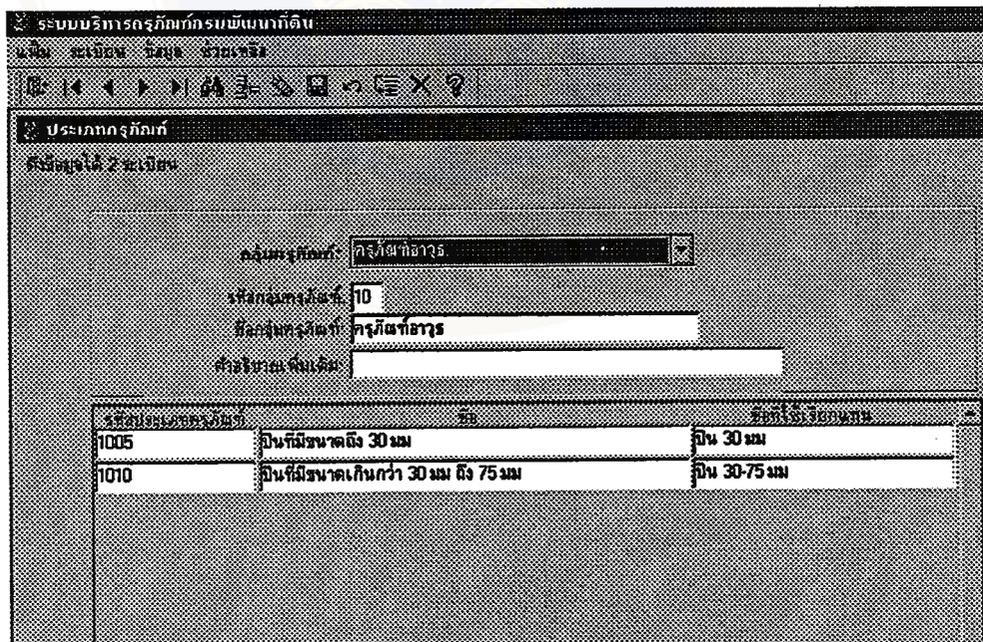


Figure C.2 w_db_hw_class

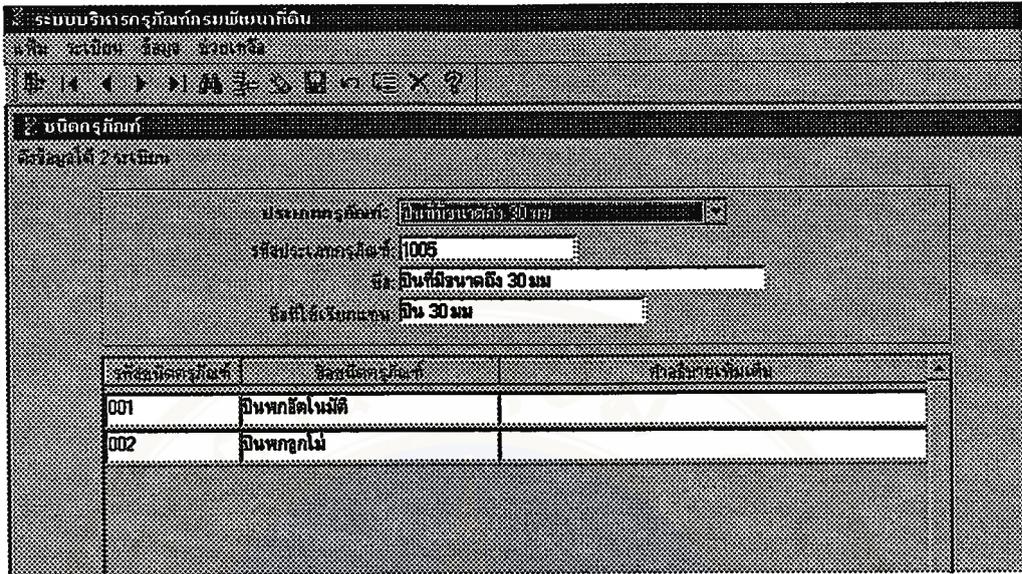


Figure C.3 w_db_hw_type

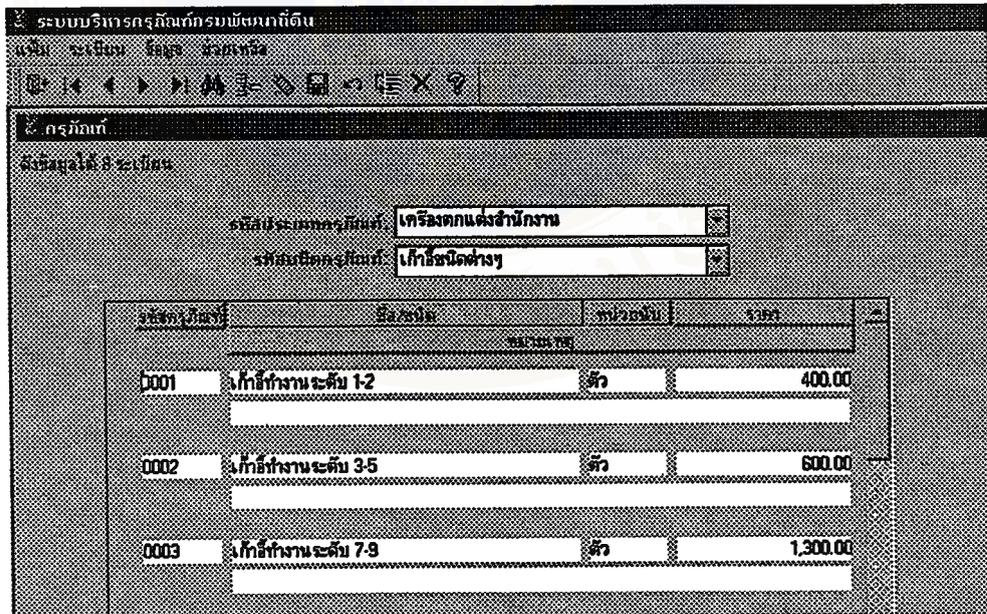


Figure C.4 w_db_hw

Figure C.5 w_db_supplier

รหัสกอง/สำนักงาน	ชื่อกอง/สำนักงาน	ชื่อยศกอง/สำนักงาน
000	สำนักงาน	สำนักงาน
101	สำนักงานเขตแผนการกรม	สสท.
102	กองคลัง	กท.
103	กองการเจ้าหน้าที่	กทจ.
104	กองช่าง	กทช.
105	กองแผนงาน	กทผ.
106	กองแผนงานและการพิมพ์	กทพ.
107	กองวางแผนการใช้ที่ดิน	กทผ.
108	กองวิเคราะห์ดิน	กทค.
109	กองสำรวจและจำแนกดิน	กทค.
110	กองอนุรักษ์ดินและน้ำ	กทอ.
111	สำนักงานพัฒนาที่ดินเขต 1	สทช. 1

Figure C.6 w_db_division

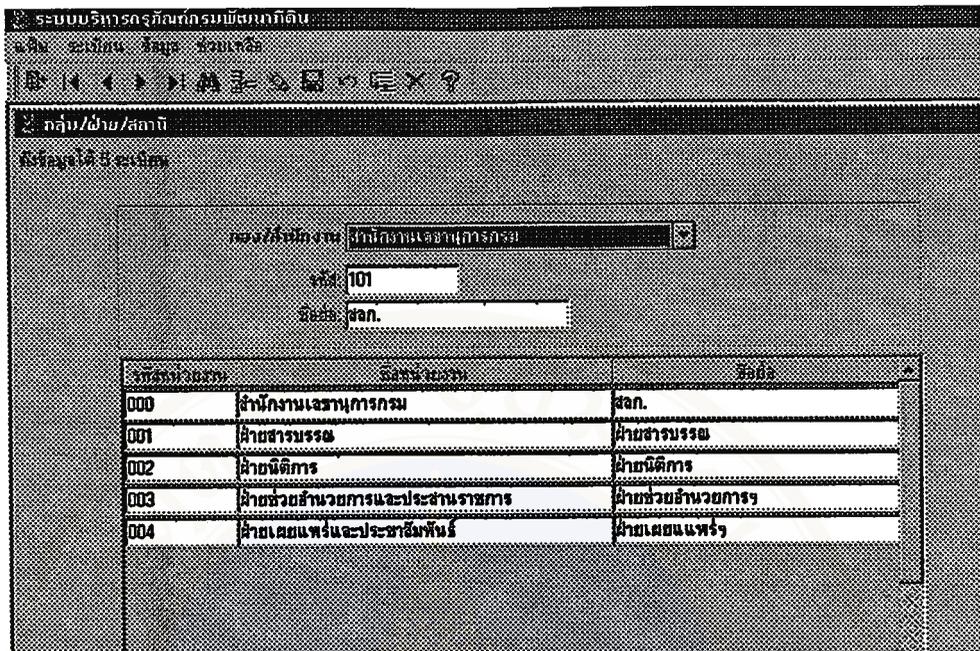


Figure C.7 w_db_section

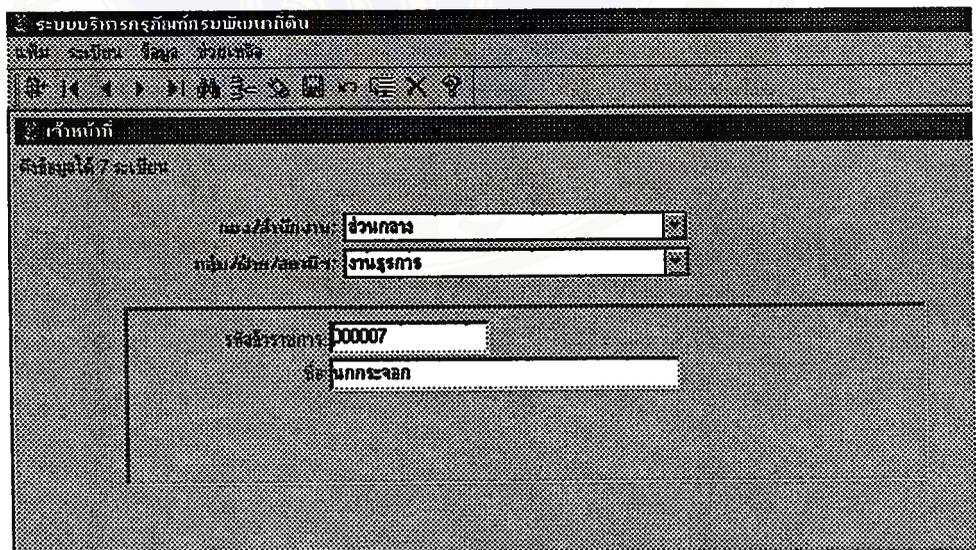


Figure C.8 w_db_staff

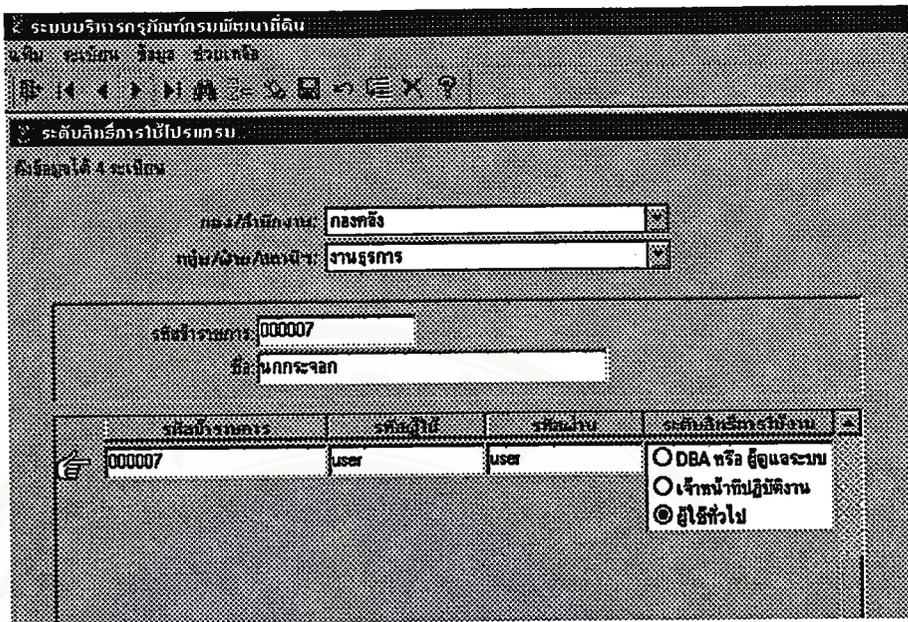


Figure C.9 w_db_security

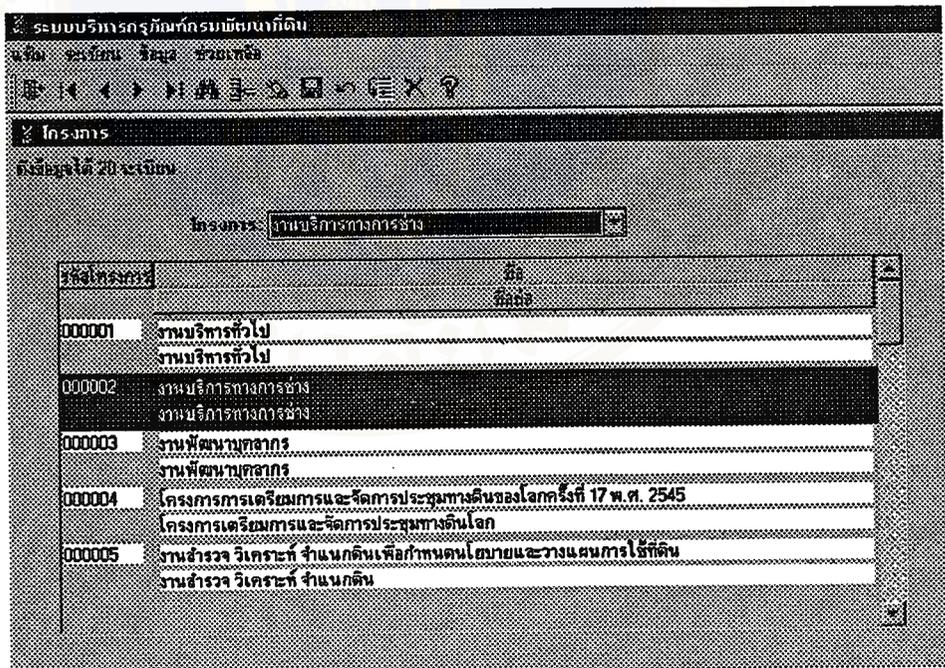


Figure C.10 w_db_project

3. Transaction

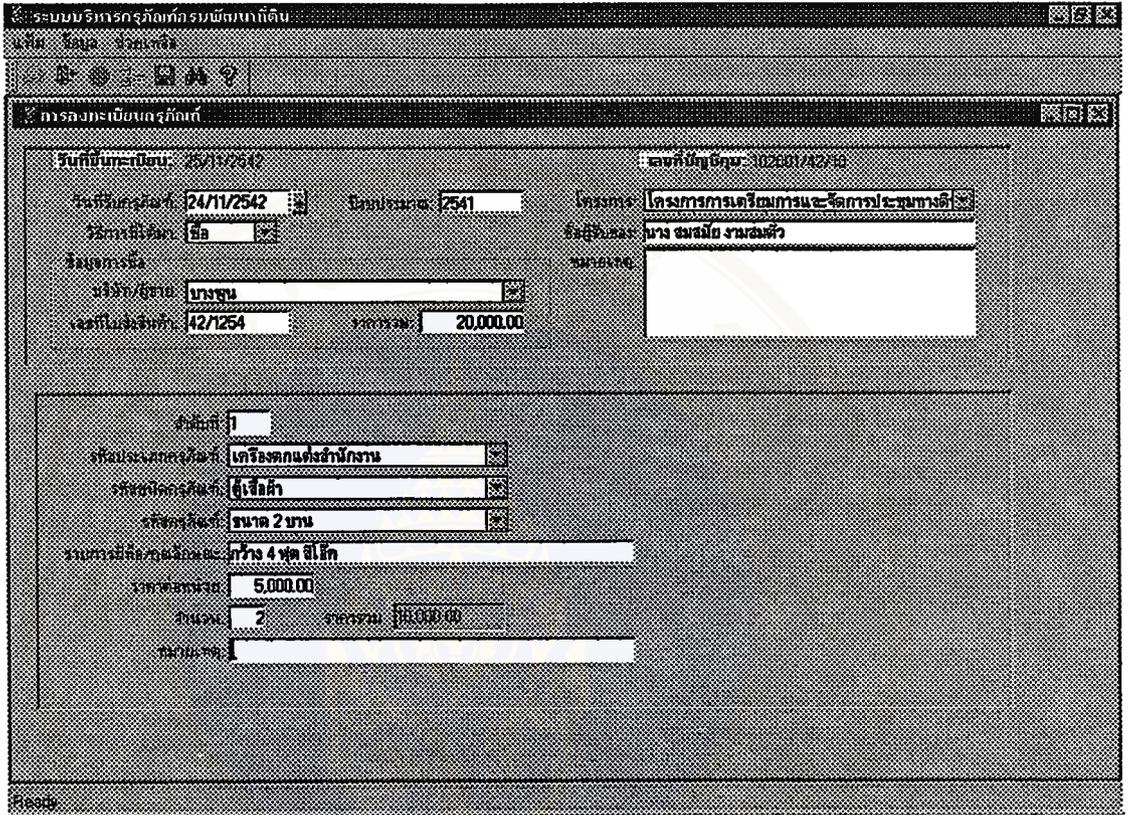


Figure C.11 w_trans_control

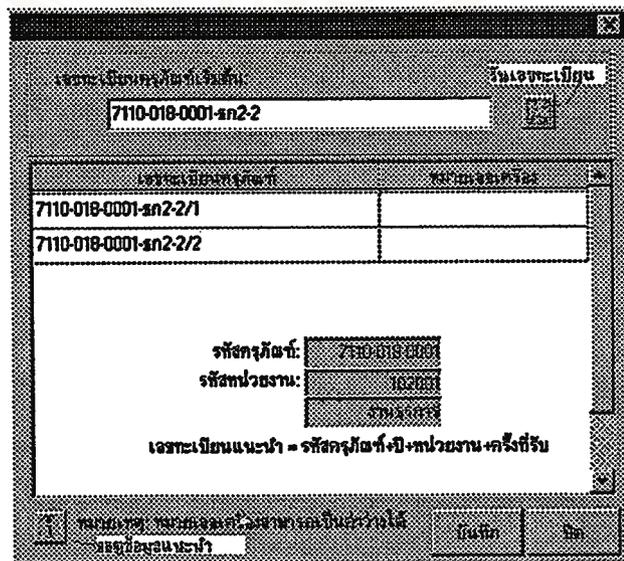


Figure C.12 w_regi_add

ระบบบริหารธุรกิจกรมบัญชีกลาง
เมนู: โฉม, ขยาย, ปิด
การเบิกจ่าย/เงินอุดหนุน

เลขที่ใบเบิกจ่าย: 102001/42/7

วันที่จ่าย: 17/11/2542
 ใ้เงิน: นกกะเจดก
 ผู้รับเงิน: สุมา จักปัดเงินทอง

เลขที่บัญชีเงินฝาก	สถานะที่ใช้	วันที่สิ้น
		00/00/0000
เลขที่บัญชีเงินฝาก		
7110-006-41-สก-1/10	แก้ไขชนิดต่างๆ แก้ไขผู้มาคิด	
7110-006-41-สก-2/24	แก้ไขชนิดต่างๆ แก้ไขผู้มาคิด	
7110-006-41-สก-2/25	แก้ไขชนิดต่างๆ แก้ไขผู้มาคิด	

Figure C.13 w_trans_dist

ระบบบริหารธุรกิจกรมบัญชีกลาง
เมนู: โฉม, ขยาย, ปิด
การคืนเงินอุดหนุน

เลขที่ใบคืน: 102001/42/3

วันที่คืน: 25/11/2542
 ผู้คืน: ป่าเงิน ป่าทอง
 ผู้รับคืน: สุมา จักปัดเงินทอง

เลขที่บัญชีเงินฝาก	เลขที่ใบเบิกจ่าย	สถานะ
		<input type="radio"/> ใช้รวมได้ <input type="radio"/> ชำระ

Figure C.14 w_trans_retu

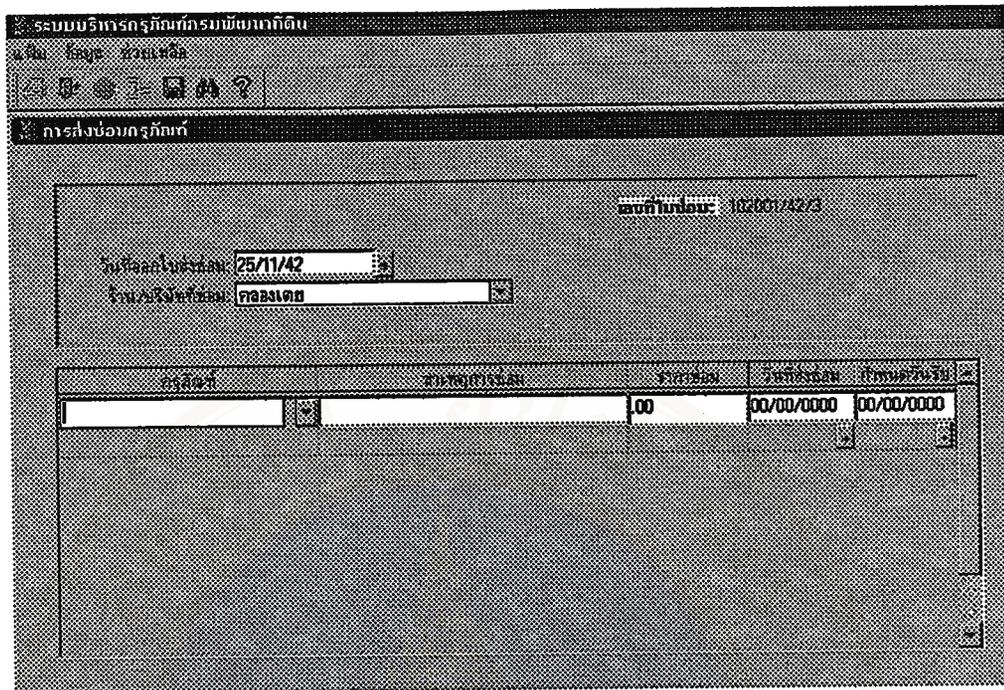


Figure C.15 w_trans_repa

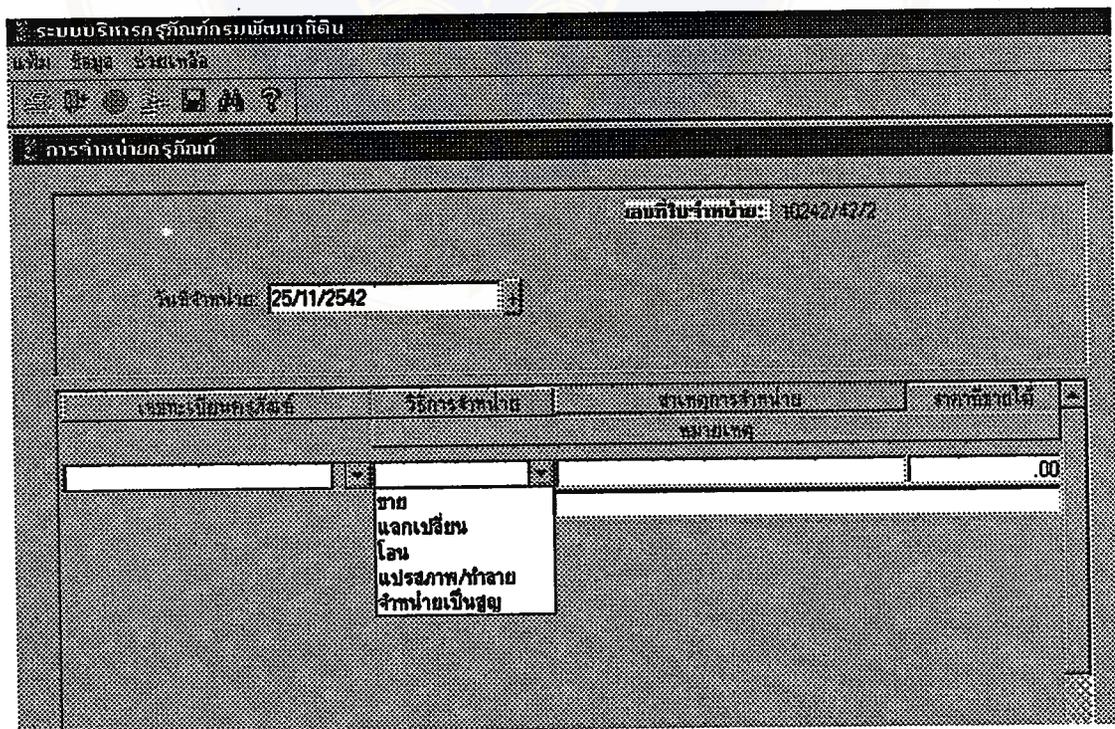


Figure C.16 w_trans_disp

Figure C.17 w_check_repair

Figure C.18 w_trans_sql

3. Query and Report

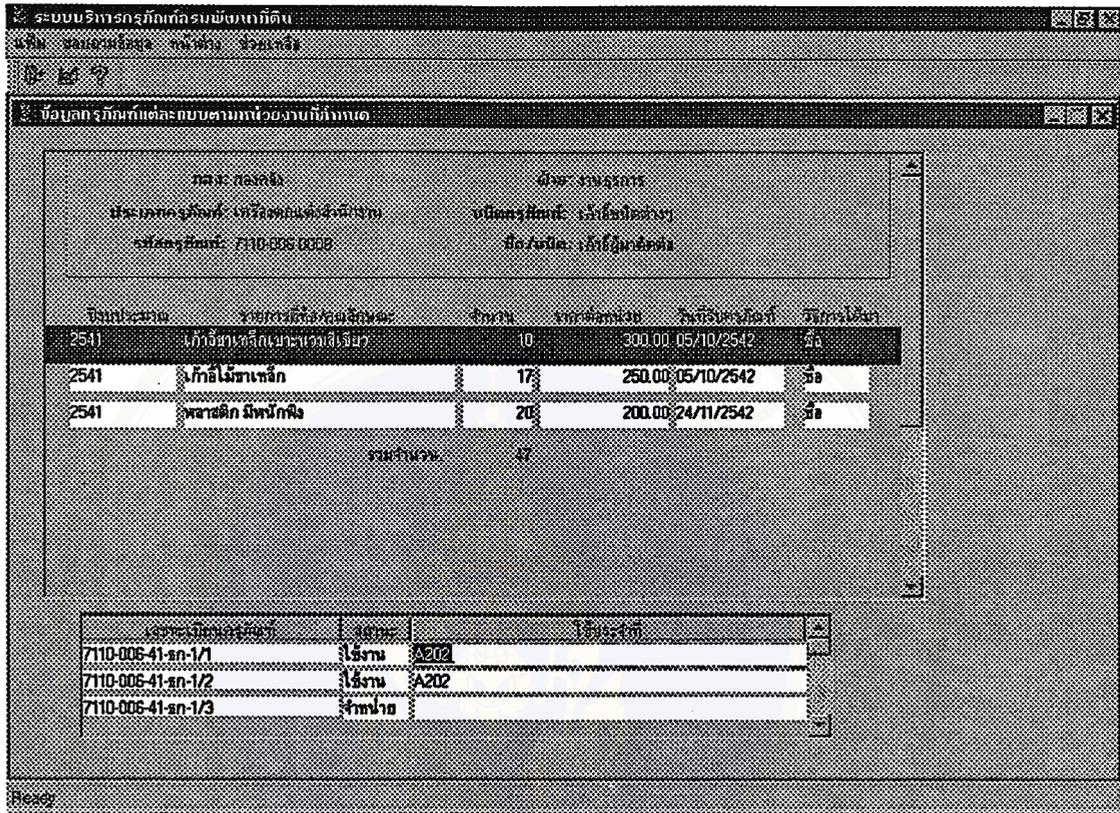


Figure C.19 w_q_hw_spec_show

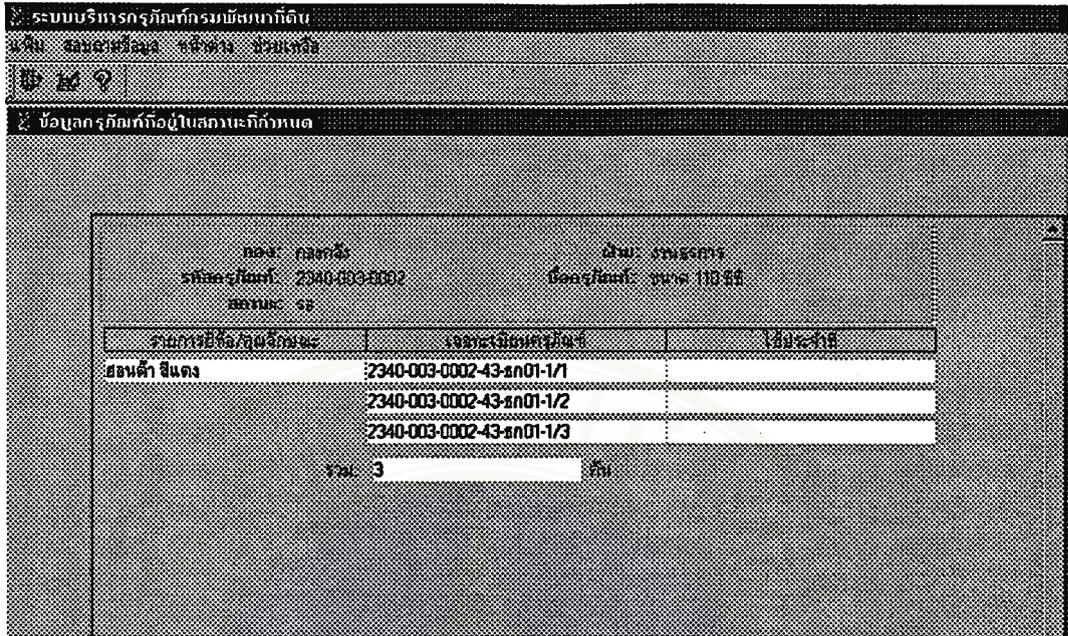


Figure C.20 w_q_hw_status

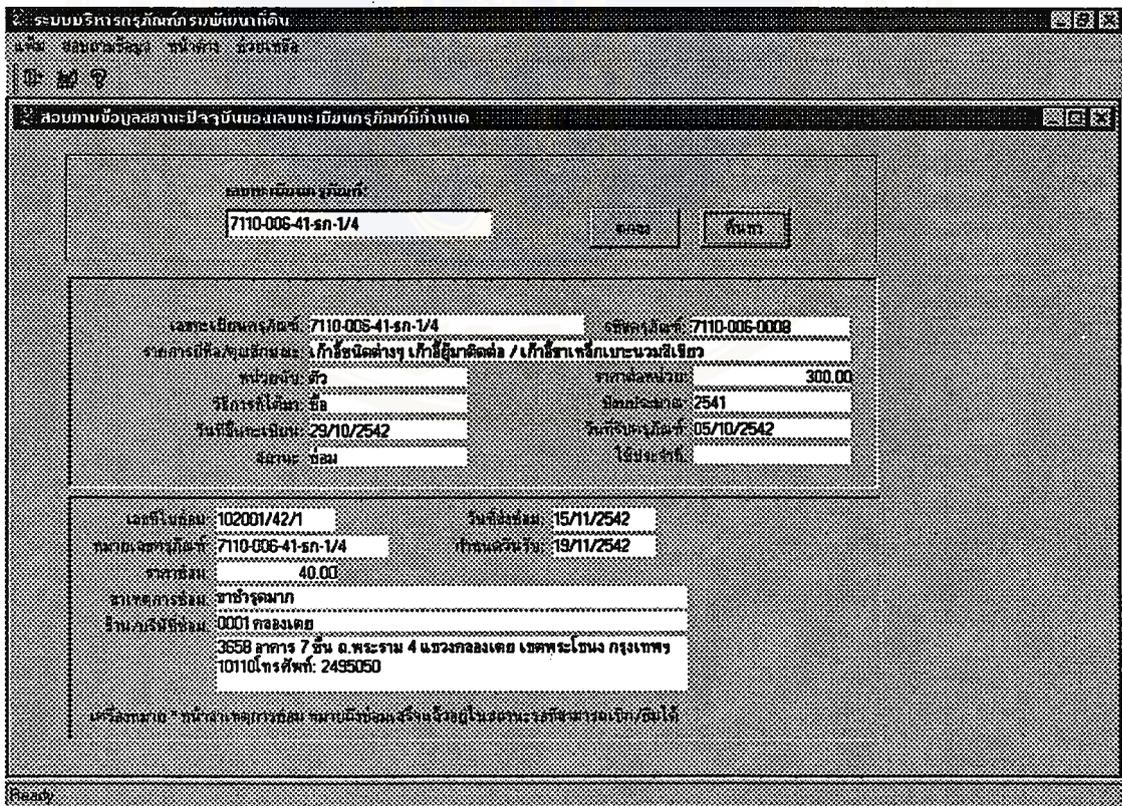


Figure C.21 w_q_regi_desc

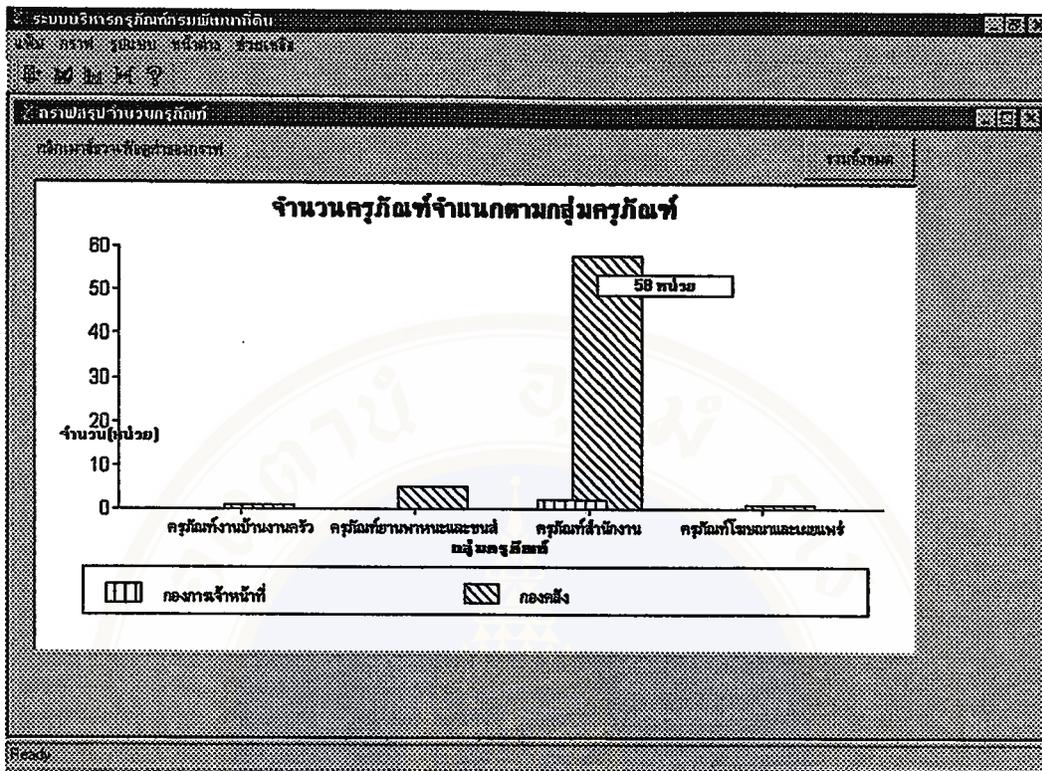


Figure C.22 w_g_sum_quantity

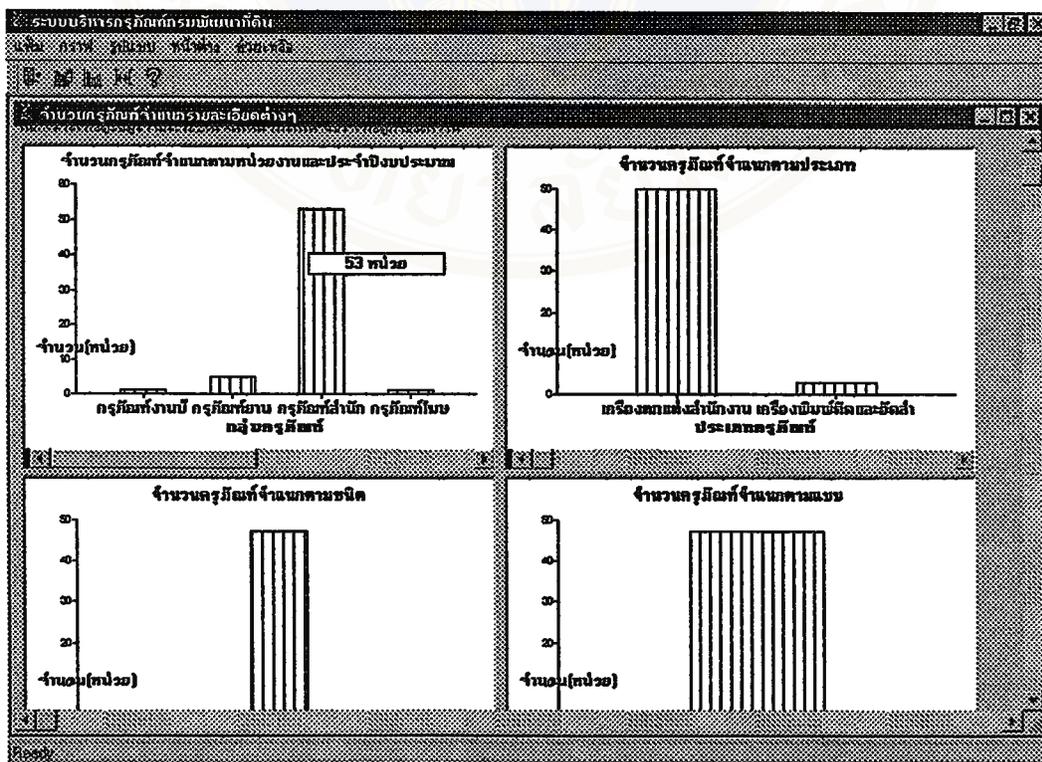


Figure C.23 w_g_drilldown

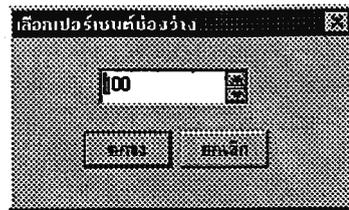


Figure C.24 w_graph_spacing

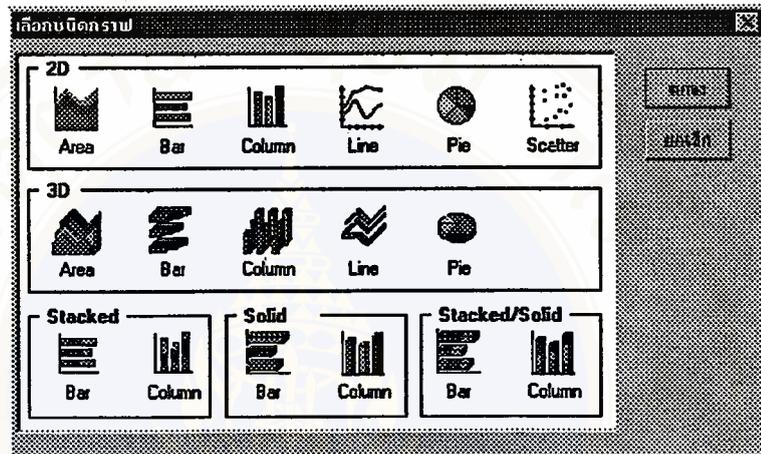


Figure C.25 w_graph_type

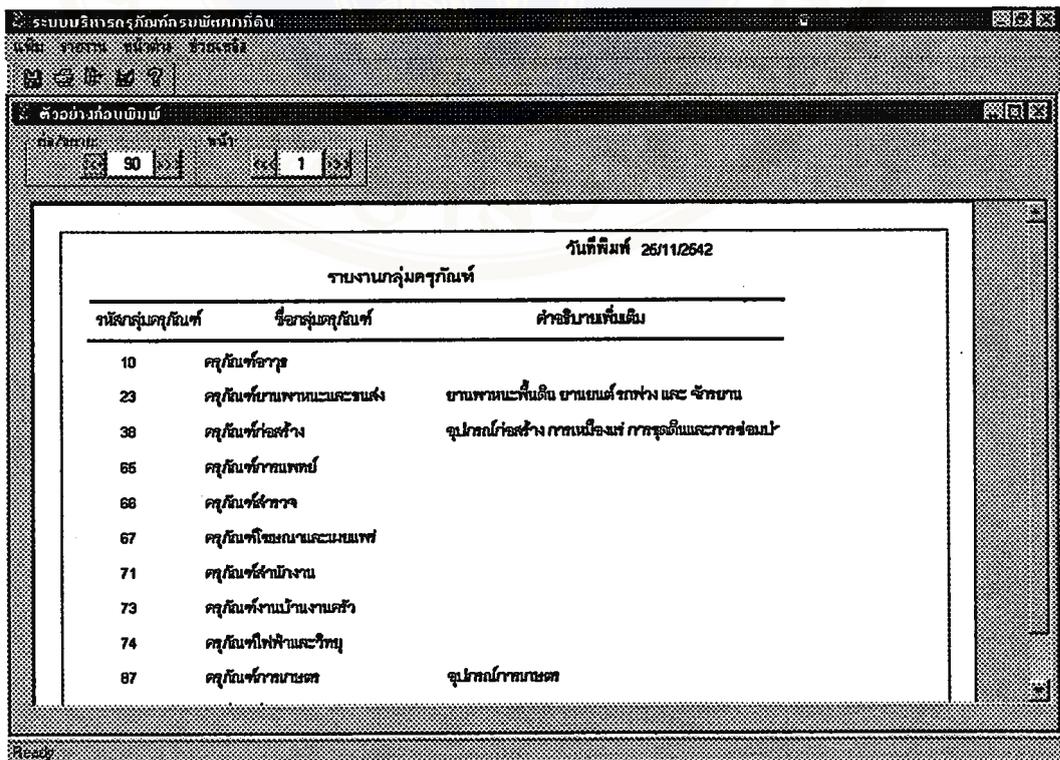


Figure C.26 w_preview

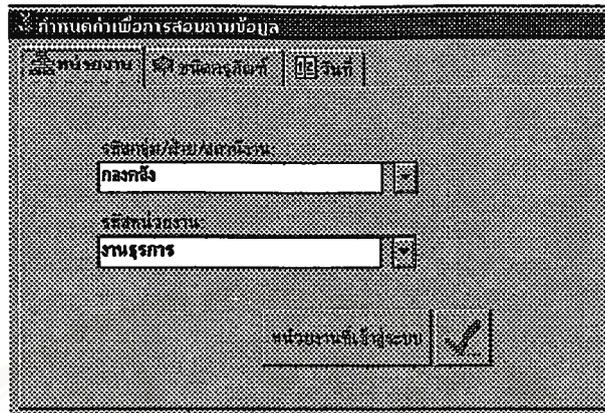


Figure C.27 w_set_arg

4. Shared Window

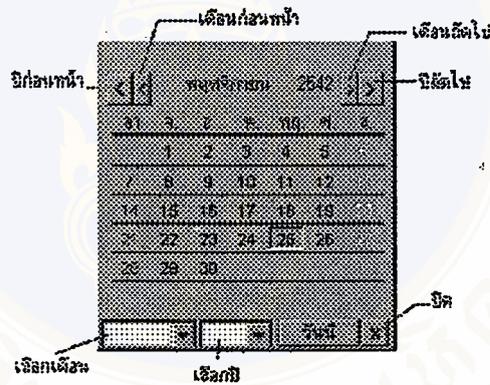


Figure C.28 w_pb_calendar

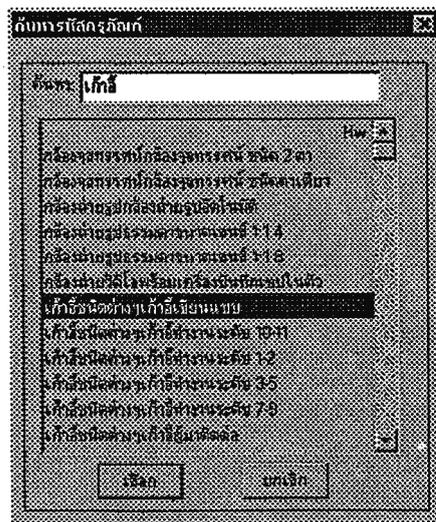


Figure C.29 w_hw_code-search

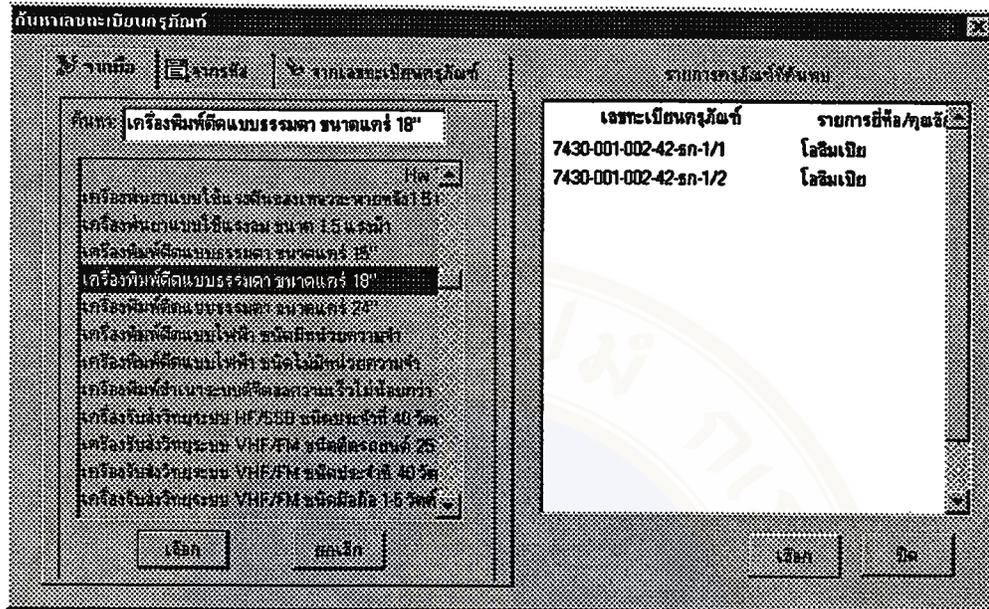


Figure C.30 w_hw_id_search

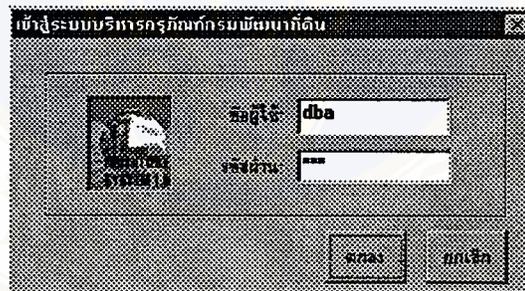


Figure C.31 w_login

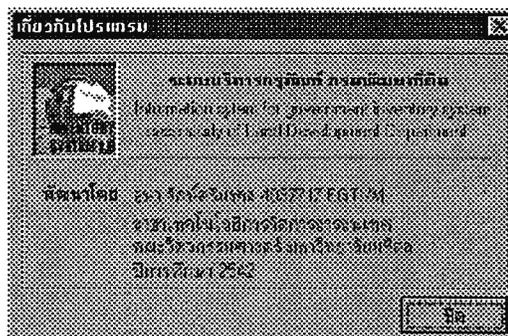
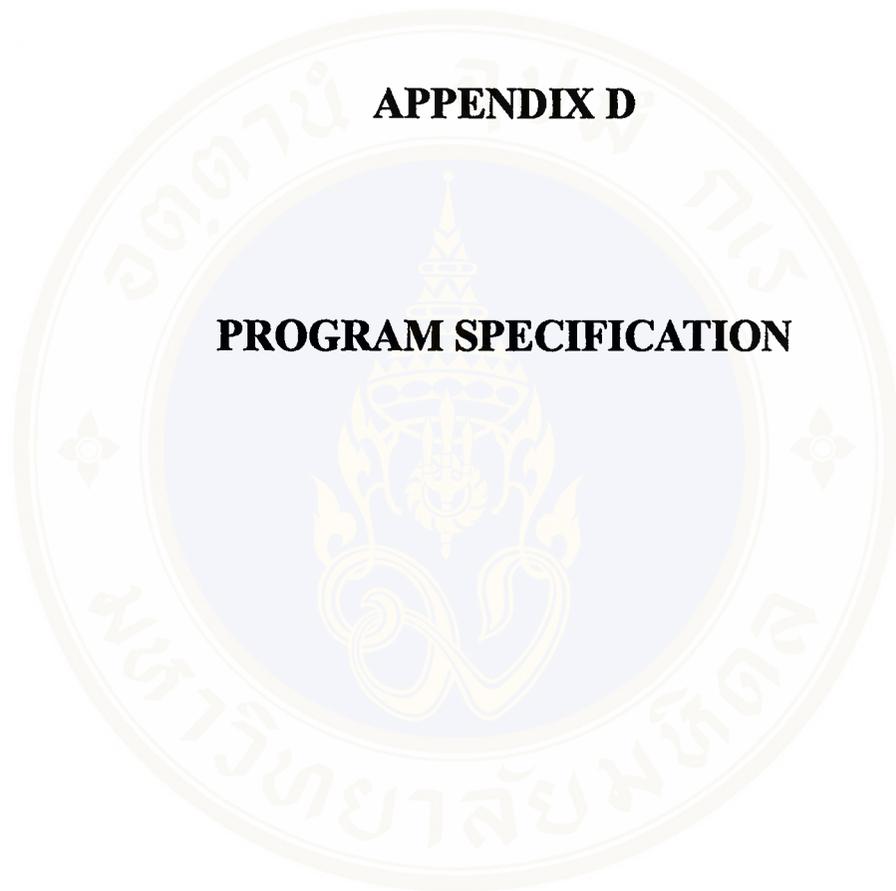


Figure C.32 w_about

APPENDIX D

PROGRAM SPECIFICATION



1. Data manipulation

These windows manipulate data in “BASIC DATA” data store. It has an ancestor window, 10 descendent windows and a shared window.

[Window] w_db_base from Window
[Description] ancestor window for data manipulation

Controls	Control Name	Remarks
Window	<w db base>	Menu="m db maint";
Datawindow	dw 1	Tab=20; Disable;
Datawindow	dw 2	Tab=20;
Datawindow	dw list	Tab=10;
Statictext	st status	

Declaration	
Instance variable	boolean ib_needSave = false long il_row, il_row_deleted datawindow idw_x

Script	Type	Name
w_db_base	Event	Close Closequery Open ue_add () ue_cancel () ue_delete () ue_first () ue_init () ue_last () ue_next () ue_prev () ue_show_status (string as_str) ue_undelete () ue_undo () ue update ()
dw_2	ControlEvent	Dberror Itemchanged Retrieveend

<i>Event: w db base.Close</i>
Close(This)
<i>Event: w db base.Open</i>
This.Post Event ue_init()
<i>Event: w db base.ue init ()</i>
Disable undelete button

<i>Event: w db base. Closequery</i>
IF dw_2.DeletedCount()+dw_2.ModifiedCount() > 0 OR ib_needSave THEN MessageBox (Ask for save ,Question!, YesNoCancel,1) IF update THEN PostEvent("ue_update") and Close Window ELSEIF Cancel THEN Don't allow to close window Else Don't save and Close window END IF ELSE Close Window END IF
<i>Event: w db base.ue add ()</i>
il_row = dw_2.insertrow(dw_2.getrow()) IF Error then dw_2.setrow(il_row) and disable dw_list ib_needsave = TRUE ELSE MessageBox(Cannot insert row) END IF
<i>Event: w db base.ue cancel ()</i>
IF MessageBox(Confirm to cancel) = Yes THEN This.Post Event ue_init() and enable dw_list ib_needsave = False END IF
<i>Event: w db base.ue delete ()</i>
IF MessageBox(Confirm to delete)=Yes THEN il_row_deleted = dw_2.GetRow() dw_2.deleterow(il_row_deleted) ib_needSave =True and enable undelete button END IF
<i>Event: w db base.ue first ()</i>
if not ib_needSave then idw_x.ScrollToRow(1) and get focus end if
<i>Event: w db base. ue last ()</i>
if not ib_needSave then idw_x.ScrollToRow(idw_x.rowCount()) and get focus end if
<i>Event: w db base. ue next ()</i>
if not ib_needSave then idw_x.ScrollNextRow() and get focus end if
<i>Event: w db base. ue prev ()</i>
if not ib_needSave then idw_x.ScrollPriorRow() and get focus end if
<i>Event: w db base. ue show status (string as str)</i>
st_status.text = as_str
<i>Event: w db base. Ue undelete ()</i>
Get the last row in the deleted buffer Move the last row in the deleted buffer to the primary buffer set focus to the row that was "restored"
<i>Event: w db base. Ue undo ()</i>
Get the original value of this row/column Reset it to the original value Reset the modified flag for this row

<i>Event: w db base. Ue update ()</i>
if dw_2.Update() > 0 then commit; and enable dw_list ib_needsave = False else RollBack; end if
<i>ControlEvent: dw 2. Dberror</i>
set sqldbcode, sqlerrtext, sqlsyntax, row, dw_name to error_data OpenWithParm(w_db_error,error_data) Return 1
<i>ControlEvent: dw 2. Itemchanged</i>
ib_needsave = True
<i>ControlEvent: dw 2. Retrieveend</i>
Parent.Post Event ue_show_status("retrieved data "+string(This.RowCount())+" rows ")

[Window] w_db_division from w_db_base	Normal
[Description] manipulate data in division data store	

Controls	Control Name	Remarks
w db base	<w db division>	Title="Division";
w db base`dw 1	dw 1	Tab=30; Invisible;
w db base`dw 2	dw 2	DataObject="d division";
w db base`dw list	dw list	DataObject="d ddw division";

Script	Type	Name
w_db_division	Event	ue_init ue_search
dw 2	ControlEvent	Rowfocuschanged
dw list	ControlEvent	Itemchanged

<i>Event: w db division. ue init</i>
Call super::ue_init dw_list.GetChild("div_id",dwc_1) Retrive dwc 1, dw list and dw 2
<i>Event: w db division. ue search</i>
string ls_sql, ls_search_name long ll_found ls_sql = 'select name1 from division' ls_search_name = OpenWithParm(w_search,ls_sql) If ls_search_name <> "" then ll_found = dw_2.Find("name1 = '"+ls_search_name+"", 1, dw_2.RowCount()) IF ll_found > 0 THEN dw_2.ScrollToRow(ll_found) ELSE MessageBox(Not found message) End if END IF
<i>ControlEvent: dw list.itemchanged</i>
il_row = dw_2.Find("div_id = '"+ This.GetText() +"'", 1, dw_2.RowCount()) dw_2.ScrollToRow(il_row)

```

ControlEvent: dw 2.Rowfocuschanged
long ll_found string ls_search
IF ib_needSave then
    This.ScrollToRow(il_row) // move to edit row
Else
    ls_search = This.GetItemString(this.getRow(),"div_id")
    dw_2.ScrollToRow(currentRow)
    ll_found = dw_list.Find("div_id = '"+ls_search+"'", 1, dw_list.RowCount())
    IF ll_found > 0 THEN dw_List.ScrollToRow(ll_found) end if
end if
    
```

[Window] w_db_hw from w_db_base Master-detail
 [Description] Manipulate data in HW data store

Controls	Control Name	Remarks
w_db base	<w_db_hw>	Title="hardware";
w_db base'dw 1	Dw 1	Tab=30; Invisible;
w_db base'dw 2	Dw 2	DataObject="d tb_hw";
w_db base'dw list	Dw list	DataObject="d ddw_hw_type";

Declaration	
Instance Variable	String is_class_id, is_type_id, is_id Datawindowchild dwc_1,dwc_2 Integer ii_id // integer of primary key

Script	Type	Name
w_db_hw	Event	ue_add ue_cal_id ue_init ue_retrieve ue_search
dw 2	ControlEvent	Rowfocuschanged
Dw_list	ControlEvent	Itemchanged Itemfocuschanged Rowfocuschanged

```

Event: w_db_hw.ue add
Call super::ue_add
if il_row > 0 then // no error
    dw_2.SetItem of class_id and type_id on il_row
    convert ii_id++ to string[4] and set item to spec_id on il_row
end if
    
```

```

Event: w_db_hw.Ue cal id
Select max(convert(numeric,spec_id)) Into :is_id from hw
where class_id = :is_class_id and type_id = :is_type_id;
IF IsNull(is_id) then
    ii_id = 0
Else ii_id =integer(is_id)
End if
    
```

```

Event: w_db_hw.Ue retrieve
Dw_2.Retrieve(is_class_id,is_type_id)
This.Post Event ue_cal_id()
    
```

<i>Event: w db hw. ue init</i>
Call super::ue_init Set DatawindowChild of class_id and type_id Retrive datawindowChild and dw_list Dw 2.Modify("spec id.EditMask.ReadOnly = yes") and Idw_x = dw 2
<i>Event: w db hw. ue search</i>
String ls_sql, ls_search_name ls_sql = 'select name from hw_type' ls_search_name = OpenWithParm(w_search,ls_sql) if ls_search_name <> "" then Select class_id,type_id into :is_class_id, :is_type_id From hw_type Where name = :ls_search_name; dw_list.Find("class_id = '"+is_class_id+"' and type_id = '"+is_type_id+ "'", 1, dw_list.RowCount()) dw_list.ScrollToRow(found row) END IF
<i>ControlEvent: dw 2.RowFocusChange</i>
IF ib_needSave then This.ScrollToRow(il_row) Else dw_2.SetRowFocusIndicator(Handle, 0) end if
<i>ControlEvent: dw list. Itemchanged</i>
IF dwo.name = 'type_id' THEN This.Post Event rowFocuschanged(This.getrow()) END IF
<i>ControlEvent: dw list. Itemfocuschanged</i>
IF dwo.name = 'type_id' THEN is_class_id = This.GetItemString(This.GetRow(),'class_id') IF isNull (is_class_id) then MessageBox(Warn to choose data in class_id ') and Set Column 'class_id' Return end if dwc_2.SetFilter("class_id = '"+is_class_id+ "'") and Filter END IF
<i>ControlEvent: dw list. Rowfocuschanged</i>
is_type_id = This.GetItemString(This.GetRow(),'type_id') Parent.Post event ue_retrieve()

[Window] w_db_hw_class from w_db_base	Master-detail
[Description] manipulate data in hw class data store	

Controls	Control Name	Remarks
w db base	<w db hw class>	Title="HW CLASS";
w db base'dw 1	dw 1	Tab=30; DataObject="d hw group";
w db base'dw 2	dw 2	DataObject="d hw class";
w db base'dw list	dw_list	DataObject="d ddw hw group";

Declaration	
Instance Variable	string is_group_id

Script	Type	Name
w_db_hw_class	Event	ue_init ue_retrieve ue_search

Script	Type	Name
dw 1	ControlEvent	Rowfocuschanged
dw_list	ControlEvent	Itemchanged

<i>Event w db hw class.ue init</i>
<pre> Call super::ue_init dw_list.GetChild("group_id",dwc_1) retrieve dwc_1, dw_list and dw_2 idw x = dw 1 // set for record navigator </pre>
<i>Event w db hw class.ue retrieve</i>
<pre> dw_2.retrieve(is_group_id) </pre>
<i>Event w db hw class.ue search</i>
<pre> string ls_sql, ls_search_name long ll_found ls_sql = 'select name from hw_group' ls_search_name = OpenWithParm(w_search,ls_sql) if ls_search_name <> "" then ll_found = dw_1.Find("name = '"+ls_search_name+"' ", 1, dw_1.RowCount()) dw_1.ScrollToRow(ll_found) END IF </pre>
<i>ControlEvent.dw 1.rowfocuschanged</i>
<pre> long ll_row if not ib_needSave then is_group_id = this.GetItemString(This.GetRow(),"group_id") ll_row = dw_list.Find("group_id = '"+is_group_id+"' ",1,dw_list.RowCount()) dw_list.ScrollToRow(ll_row) Parent.Post EVENT ue_retrieve() end if </pre>
<i>ControlEvent.dw list.itemchanged</i>
<pre> ll_row = dw_1.Find("group_id = '"+ This.GetText()+"' ", 1, dw_1.RowCount()) dw 1.ScrollToRow(ll_row) </pre>

[Window] w_db_hw_group from w_db_base	Normal
[Description] manipulate data in HW_GROUP data store	

Controls	Control Name	Remarks
w_db base	<w_db_hw_group>	Title="HW_GROUP";
w_db base'dw 1	Dw 1	Tab=30; Invisible;
w_db base'dw 2	Dw 2	DataObject="d_hw_group";
w_db base'dw_list	Dw_list	DataObject="d_ddw_hw_group";

Script	Type	Name
w_db_hw_group	Event	ue_init ue_search
dw 2	ControlEvent	Rowfocuschanged
dw_list	ControlEvent	Itemchanged

<i>Event: w dw hw_group.ue init</i>
<pre> Call super::ue_init dw_list.GetChild("group_id",dwc_1) retrieve dwc_1,dw_list and dw_2 idw x = dw 2 and dw 2.Modify("group_id.EditMask.ReadOnly = yes") </pre>

<pre> Event: w dw hw group.ue search string ls_sql, ls_search_name long ll_found ls_sql = 'select name from hw_group' ls_search_name = OpenWithParm(w_search,ls_sql) if ls_search_name <> "" then ll_found = dw_2.Find("name = '"+ls_search_name+'"', 1, dw_2.RowCount()) dw_2.ScrollToRow(ll_found) END IF </pre>
<pre> ControlEvent:dw_2.rowfocuschanged long ll_found string ls_search IF ib_needSave then This.ScrollToRow(il_row) Else ls_search = This.GetItemString(this.getRow(),"group_id") ll_found = dw_list.Find("group_id = '"+ls_search+'"', 1, dw_list.RowCount()) dw_List.ScrollToRow(ll_found) end if </pre>
<pre> ControlEvent:dw_list.itemchanged il_row = dw_2.Find("group_id = '"+ This.GetText()+"'", 1, dw_2.RowCount()) dw_2.ScrollToRow(il_row) </pre>

[Window] w_db_hw_type from w_db_base	Master-detail
[Description] manipulate data in HW TYPE data store	

Controls	Control Name	Remarks
w_db_base	<w_db_hw_type>	Title="hardware type";
w_db_base`dw_1	Dw_1	Tab=30; DataObject="d_hw_class ff";
w_db_base`dw_2	Dw_2	DataObject="d_hw_type_grid";
w_db_base`dw_list	Dw_list	DataObject="d_ddw_hw_class";

Declaration	
Instance Variable	String is_class_id

Script	Type	Name
w_db_hw_type	Event	ue_add ue_init ue_retrieve ue_search
dw_1	ControlEvent	Rowfocuschanged
dw_list	ControlEvent	Itemchanged

<pre> Event: w_db_hw_type.ue add Call super::ue_add if no error then dw_2.setItem(il_row,"class_id",is_class_id) dw_2.setrow(il_row) and set column 'type_id' dw_2.Modify("type_id.EditMask.ReadOnly = no") end if </pre>
<pre> Event: w_db_hw_type.ue retrieve dw_2.retrieve(is_class_id) </pre>

<i>Event: w db hw type.ue init</i>
Call super::ue_init dw_list.GetChild("class_id",dwc_1) retrieve dwc_1,dw_list and dw_2 idw_x = dw_1 dw_2.Modify("type_id.EditMask.ReadOnly = yes")
<i>Event: w db hw type.ue search</i>
string ls_sql, ls_search_name long ll_found ls_sql = 'select name1 from hw_class' ls_search_name = OpenWithParm(w_search,ls_sql) IF ls_search_name <> "" then ll_found = dw_1.Find("name1 = '"+ls_search_name+ "'", 1, dw_1.RowCount()) dw_1.ScrollToRow(ll_found) END IF
<i>ControlEvent: dw 1.Rowfocuschanged</i>
Long ll_row If not ib_needSave then is_class_id = this.GetItemString(This.GetRow(),"class_id") ll_row = dw_list.Find("class_id = '"+is_class_id+ "'",1,dw_list.RowCount()) dw_list.ScrollToRow(ll_row) Parent.Post EVENT ue_retrieve() End if
<i>ControlEvent: dw list.itemchange</i>
Ll_row = dw_1.Find("class_id = '"+ This.gettext() + "'", 1, dw_1.RowCount()) dw 1.ScrollToRow(ll_row)

[Window] w_db_project from w_db_base	Normal
[Description] manipulate data in PROJECT data store	

Controls	Control Name	Remarks
w_db base	<w_db project>	Title="Project";
w_db base'dw 1	dw 1	Tab=30; Invisible;
w_db base'dw 2	dw 2	DataObject="d tb project";
w_db base'dw list	dw_list	DataObject="d ddw project";

Script	Type	Name
w_db_project	Event	ue_init ue search
dw 2	ControlEvent	Rowfocuschanged
dw list	ControlEvent	Itemchanged

<i>Event: w db project.ue init</i>
Call super::ue_init dw_list.GetChild("project_id",dwc_1) and retrieve dwc_1, dw_list
<i>Event: w db project.ue search</i>
string ls_sql, ls_search_name long ll_found ls_sql = 'select name1 from project' ls_search_name = OpenWithParm(w_search,ls_sql) if ls_search_name <> "" then ll_found = dw_2.Find("name1 = '"+ls_search_name+ "'", 1, dw_2.RowCount()) dw_2.ScrollToRow(ll_found) END IF

```

ControlEvent: dw 2.Rowfocuschanged
long ll_found string ls_search
IF ib_needSave then This.ScrollToRow(il_row)
Else
    ls_search = This.GetItemString(this.getRow(),"project_id")
    ll_found = dw_list.Find("project_id = '"+ls_search+"'", 1, dw_list.RowCount())
    dw_List.ScrollToRow(ll_found)
end if

ControlEvent: dw list.Itemchanged
il_row = dw 2.Find("project_id = '"+ This.gettext() +"'", 1, dw 2.RowCount())
dw 2.ScrollToRow(il_row)
    
```

[Window] w_db_section from w_db_base Normal
[Description] manipulate data in SECTION data store

Controls	Control Name	Remarks
w_db base	<w_db_section>	Title="Section";
w_db base'dw 1	dw 1	Tab=30; DataObject="d division ct";
w_db base'dw 2	dw 2	DataObject="d section";
w_db base'dw list	dw_list	DataObject="d ddw division";

Declaration	
Instance Variable	String is_div_id, is_id Int ii_id

Script	Type	Name
w_db_section	Event	ue_add ue_cal_id ue_init ue_retrieve ue_search
dw 1	ControlEvent	rowfocuschanged
dw_list	ControlEvent	itemchanged

```

Event: w db_section.ue add
Call super::ue_add
if il_row > 0 then
    ii_id++
    convert ii_id to string[3]
    Set item "sect_id" and div_div to dw_2 on il_row
    dw_2.setrow(il_row) and set column 'name1'
end if
    
```

```

Event: w db_section.ue cal id
select MAX(sect_id) Into :is_id from section where div_id = :is_div_id;
ii_id = Integer(is_id)
li_id = ii_id+1
    
```

```

Event: w db_section.ue init
Call super::ue_init
dw_list.GetChild("div_id",dwc_1)
Retrieve dwc_1, dw_list, dw_1
idw_x = dw_1
dw 2.Modify("sect_id.EditMask.ReadOnly = yes")
    
```

```

Event: w_db_section.ue_retrieve
dw_2.retrieve(is_div_id)
This.Post Event ue_cal_id()
Event: w_db_section.search
string ls_sql, ls_search_name long ll_found
ls_sql = 'select name1 from division'
ls_search_name = OpenWithParm(w_search,ls_sql)
if ls_search_name <> "" then
    ll_found = dw_1.Find("name1 = "+ls_search_name+"", 1, dw_1.RowCount())
    dw_1.ScrollToRow(ll_found)
END IF
ControlEvent: dw_1.rowfocuschanged
long ll_row
if not ib_needSave then
    is_div_id = this.GetItemString(This.GetRow(),"div_id")
    ll_row = dw_list.Find("div_id = "+is_div_id+"",1,dw_list.RowCount())
    dw_list.ScrollToRow(ll_row)
    Parent.Post EVENT ue_retrieve()
end if
ControlEvent: dw_list.itemchanged
ll_row = dw_1.Find("div_id = "+This.gettext() +""", 1, dw_1.RowCount())
dw_1.ScrollToRow(ll_row)
    
```

[Window] w_db_security from w_db_base One to one relationship
 [Description] manipulate data in SECURITY data store

Controls	Control Name	Remarks
w_db_base	<w_db_security>	Title="Privilege of users ";
w_db_base'dw_1	Dw_1	Tab=30; DataObject="d_staff ff";
w_db_base'dw_2	Dw_2	DataObject="d_tb_security";
w_db_base'dw_list	Dw_list	DataObject="d_ddw_section";

Declaration	
Instance Variable	String is_div_id, is_sect_id Datawindowchild dwc_1,dwc_2 Boolean ib_found = false

Script	Type	Name
w_db_security	Event	ue_add ue_init ue_search
dw_1	Event	ue_retrieve ()
	ControlEvent	Rowfocuschanged
dw_2	ControlEvent	Rowfocuschanged
dw_list	ControlEvent	Itemchanged
		Itemfocuschanged
		Rowfocuschanged

```

Event: dw_1.ue_retrieve
This.Retrieve(is_div_id,is_sect_id)
if This.RowCount() > 0 then dw_2.visible = TRUE end if
    
```

<p><i>Event: w db security.ue add</i></p> <p>Call super::ue_add string ls_id if il_row > 0 and (not ib_found) then ls_id = dw_1.GetItemString(dw_1.GetRow(),"gov_id") dw_2.SetItem (il_row, "gov_id", ls_id) dw_2.scrollToRow(il_row) and set column 'user_id' end if</p>
<p><i>Event: w db security.ue init</i></p> <p>Call super::ue_init dw_list.getChild('div_id',dwc_1) dw_list.getChild('sect_id',dwc_2) retrieve dwc_1, dwc_2, dw_list, dw_1, dw_2 idw_x = dw_1 and disable insert button</p>
<p><i>Event: w db security.search</i></p> <p>String ls_sql, ls_search_name long ll_found ls_sql = 'select name from staff' ls_search_name = OpenWithParm(w_search,ls_sql) if ls_search_name <> "" then Select div_id,sect_id into :is_div_id, :is_sect_id From staff Where name = :ls_search_name; ll_found = dw_list.Find("div_id = '"+is_div_id+"' and sect_id = '"+is_sect_id+"", 1, dw_list.RowCount()) dw_list.ScrollToRow(ll_found) dw_1.Post Event ue_retrieve() ll_found = dw_1.Find("gov_id = '"+ls_search_name+"",1,dw_1.RowCount()) dw_1.ScrollToRow(ll_found) END IF</p>
<p><i>ControlEvent: dw 1.Rowfocuschanged</i></p> <p>String ls_gov_id long ll_found ll_found = This.GetRow() IF ll_found > 0 then ls_gov_id = This.GetItemstring(ll_found,"gov_id") ll_found = dw_2.Find("gov_id = '"+ls_gov_id+"", 1, dw_2.RowCount()) If ll_found > 0 then ib_found = True dw_2.scrollToRow(ll_found) and disable insert button else ib_found = false and enable insert button end if end if</p>
<p><i>ControlEvent: dw 2. Rowfocuschanged</i></p> <p>IF ib_needSave then This.ScrollToRow(il_row) Else dw_2.SetRowFocusIndicator(Handle, 0) End if</p>
<p><i>ControlEvent: dw list .Itemchanged</i></p> <p>IF dwo.name = 'sect_id' THEN This.Post Event rowFocuschanged(This.getrow()) END IF</p>
<p><i>ControlEvent: dw list. Rowfocuschanged</i></p> <p>is_sect_id = This.GetItemString(This.GetRow(),'sect_id') dw_1.Post Event ue_retrieve()</p>

```

ControlEvent: dw list. Itemfocuschanged
IF dwo.name = 'sect_id' THEN
    is_div_id = This.GetItemString(This.GetRow(), 'div_id')
    IF isNull(is_div_id) then
        MessageBox( Warn to choose division id) and Set Column 'div_id'
    Return
End if
dwc_2.SetFilter("div_id = '"+is_div_id+"'") and filter
END IF
    
```

[Window] w_db_staff from w_db_base Normal
 [Description] manipulate data in STAFF

Controls	Control Name	Remarks
w db base	<w db staff>	Title="Staff";
w db base'dw 1	dw 1	Tab=30; Invisible;
w db base'dw 2	dw 2	DataObject="d staff ff";
w db base'dw list	dw list	DataObject="d ddw section";

Declaration	
Instance Variable	string is_div_id, is_sect_id datawindowchild dwc_1, dwc_2

Script	Type	Name
w_db_staff	Event	ue_add ue_init ue_retrieve ue_search
dw 2	Event	ue_dropdown pbm_dwndropdown
dw_list	ControlEvent	Itemchanged Itemfocuschanged Rowfocuschanged

```

Event: w db staff. ue add
Call super::ue_add
If no error then
    Dw_2.setItem div_id and sect_id
    Dw_2.setrow(il_row) and set column 'gov_id'
end if

Event: w db staff. ue init
Call super::ue_init
dw_list.getChild('div_id', dwc_1)
dw_list.getChild('sect_id', dwc_2)
retrieve dwc_1, dwc_2, dw_list and idw_x = dw_2

Event: w db staff. ue retrieve
dw_2.retrieve(is_div_id, is_sect_id)

Event: dw 2. ue dropdown pbm_dwndropdown
if IsNull(is_div_id) then
    MessageBox("Warn to choose division id)
else
    dwc_2.setFilter("div_id = '"+is_div_id+"'") and filter
end if
    
```

<i>Event: w_db_staff.ue_search</i>
String ls_sql, ls_search_name long ll_found ls_sql = 'select name1 from division' ls_search_name = OpenWithParm(w_search,ls_sql) if ls_search_name <> "" then ll_found = dw_1.Find("name1 = '"+ls_search_name+ "'", 1, dw_1.RowCount()) dw_1.ScrollToRow(ll_found) END IF
<i>ControlEvent: dw_list.Itemchanged</i>
IF dwo.name = 'sect_id' THEN This.Post Event rowFocuschanged(This.getrow()) END IF
<i>ControlEvent: dw_list.Itemfocuschanged</i>
IF dwo.name = 'div_id' THEN is_div_id = This.GetItemString(This.GetRow(),'div_id') IF isNull(is_div_id) then MessageBox(Warn to choose division id) and Return end if dwc_2.SetFilter("div_id = '"+is_div_id+ "'") and filter END IF
<i>ControlEvent: dw_list.Rowfocuschanged</i>
is_sect_id = This.GetItemString(This.GetRow(),'sect_id') Parent.Post event ue_retrieve()

[Window] w_db_supplier from w_db_base	Normal
[Description] manipulate data in SUPPLIER data store	

Controls	Control Name	Remarks
w_db_base	<w_db_supplier>	Title="Supplier/company";
w_db_base'dw_1	dw_1	Tab=30; Invisible;
w_db_base'dw_2	dw_2	DataObject="d_supplier";
w_db_base'dw_list	dw_list	DataObject="d_ddw_supplier";

Declaration	
Instance Variable	integer ii_supplier_id string is_id

Script	Type	Name
w_db_supplier	Event	ue_add ue_cal_id ue_init ue_search
dw_2	ControlEvent	Rowfocuschanged
dw_list	ControlEvent	Itemchanged

<i>Event: w_db_supplier.ue_add</i>
Call super:: ue_add Convert ii_supplier_id++ to string[4] dw_2.setItem(il_row,"supplier_id",ls_supplier_id) and set column 'name'
<i>Event: w_db_supplier.ue_cal_id</i>
Select max(convert(numeric,supplier_id)) Into :is_id from supplier; ii_supplier_id = Integer(is_id)

<i>Event: w db supplier.ue init</i>
Call super:ue_init dw_list.GetChild("supplier_id",dwc_1) retrieve dwc_1, dw_list, dw_2 This.Post Event ue_cal_id() idw x = dw 2 and dw 2.Modify("supplier_id.EditMask.ReadOnly = yes")
<i>Event: w db supplier.ue search</i>
string ls_sql, ls_search_name long ll_found ls_sql = 'select name from supplier' ls_search_name = OpenWithParm(w_search,ls_sql) if ls_search_name <> "" then ll_found = dw_2.Find("name = '"+ls_search_name+ "'", 1, dw_2.RowCount()) dw_2.ScrollToRow(ll_found) END IF
<i>ControlEvent: dw 2.rowfocuschanged</i>
long ll_found string ls_search IF ib_needSave then This.ScrollToRow(il_row) Else ls_search = This.GetItemString(this.getRow(),"supplier_id") ll_found = dw_list.Find("supplier_id = '"+ls_search+ "'", 1, dw_list.RowCount()) dw_List.ScrollToRow(ll_found) end if
<i>ControlEvent: dw list.itemchanged</i>
il_row = dw_2.Find("supplier_id = '"+ This.GetText()+ "'", 1, dw_2.RowCount()) dw_2.ScrollToRow(il_row)

[Window] w_search from Window	Shared window
[Description] receive SQL statement to create datawindow and return selected name	

Controls	Control Name	Remarks
Window	<w_search>	Type=response!, Title="Search data";
u selection list	uo 1	

Declaration	
Instance Variable	string is_name.

Script	Type	Name
w_search	Event	Close Open
uo 1	ControlEvent	ue_entry_chosen

<i>Event: w_search.close</i>
CloseWithReturn(This, is_name)
<i>Event: w_search.open</i>
string ls_sql ls_sql = Message.StringParm uo 1.create datawindow(sqlca, ls sql)
<i>ControlEvent: uo 1.ue entry chosen</i>
is_name = return_selected () Close(Parent)

2. Transaction process

These windows manipulate data in “TRANS DATA” data store. It divides into 2 major processes: adding and editing. Adding process has an ancestor window, 5 descendent windows, a shared window and 2 additional windows. Editing process has an ancestor window, 5 descendent windows and a shared window.

Adding process

[Window] w_trans_base from Window
[Description] window ancestor for transaction adding process

Controls	Control Name	Remarks
Window	<w_trans_base>	Menu="m_trans_add"; Title="Transaction";
Datawindow	dw_1	Tab=20;
Datawindow	dw_2	Tab=10;

Declaration	
Shared Variable	string ss_list
Instance Variable	boolean ib_needSave = false, ib_pass = false long il_row_1, il_row_2, il_row, il_main_no datawindowchild dwc_hw_id string is_hw_id, is_dataobject datastore ids_sql_trans

Script	Type	Name
w_trans_base	Event	Close Closequery Open ue_add_dw_1() ue_add_dw_2() ue_hw_search() ue_init() ue_print() ue_update_dw_1() ue_update_dw_2()
dw_1	ControlEvent	Rowfocuschanged Sqlpreview Dberror
dw_2	Event	ue_filter (string as_hw_id)
	ControlEvent	Dberror Sqlpreview

<i>Event: w_trans_base. Close</i>
IF ib_pass Then This.Trigger event ue_check_error() andCommit; Else Rollback; end if Destroy ids_sql_trans

<p><i>Event: w trans base. Closequery</i></p> <pre> IF (dw_1.ModifiedCount() > 0 OR dw_2.ModifiedCount() > 0) and (ib_needsave) THEN MessageBox (Ask for save ,Question!, YesNoCancel!,1) IF update THEN This.Trigger Event ue_update_dw_2() IF ib_pass then close window Else don't allow to close window End if ELSEIF Cancel THEN Don't allow to close the window Else ib_pass = false and Don't save and Close window END IF ELSE Close Window END IF </pre>
<p><i>Event: w trans base. Open</i></p> <pre> This.Post Event ue_init() </pre>
<p><i>Event: w trans base. ue add dw 1 ()</i></p> <pre> il_row_1 = dw_1.insertrow(0) if il_row_1 <= 0 then MessageBox("Cannot insert row)and disable update_dw_1 button Return elseif il_row_1 > 0 then dw_1.ScrollToRow(il_row_1) set item gs_div_id, gs_sect_id, gs_year to dw_1 end if </pre>
<p><i>Event: w trans base. ue add dw 2 ()</i></p> <pre> il_row_2 = dw_2.insertrow(0) if il_row_2 <= 0 then MessageBox(Cannot insert row) Return elseif il_row_2 > 0 then dw_2.scrollToRow(il_row_2) ib_needsave = TRUE Enable search, update_dw_2 button and disable insert_dw_2 button. end if </pre>
<p><i>Event: w trans base. ue_hw_search ()</i></p> <pre> IF gs_proc <> gs_cont then is_hw_id = OpenWithParm(w_hw_id_search,1) if is_hw_id = " " then return end if end if </pre>
<p><i>Event: w trans base ue init ()</i></p> <pre> Disable insert_dw_2, update_dw_2, search, m_print button and set invisible dw_2 dw_1.Retrieve(gs_div_id,gs_sect_id,gs_year) ids_sql_trans = Create DataStore "d_ds_sql_trans" </pre>
<p><i>Event: w trans base. ue print ()</i></p> <pre> str_trans_print lstr_data w_trans_print l_w_instance lstr_data.s_dataobject = is_dataobject lstr_data.s_no = string(il_main_no) OpenSheetWithParm(l_w_instance,lstr_data,g_w_mdi,0,Layered!) </pre>

<i>Event: w trans base. ue update dw 1 ()</i>
dw_1.TriggerEvent ("ue_check_error") If ib_pass then IF dw_1.Update() = 1 Then This.Post event ue_retrieve_dw_2() This.Post Event ue_add_dw_2() Disable update_dw_1, dw_1 button and visible dw_2 Else Ib_pass =false and set focus dw_1 end if End IF
<i>Event: w trans base. ue update dw 2 ().</i>
dw_2.TriggerEvent ("ue_check_error") enable print button ib_needSave = False
<i>ControlEvent: dw 1. Rowfocuschanged</i>
this.ScrollToRow(il_row_1)
<i>ControlEvent: dw 1. Sqlpreview</i>
string ls_sql long ll_row IF (sqltype = PreviewInsert!) or (sqltype = PreviewUpdate!) then ls_sql = sqlsyntax + ';' ll_row = ids_sql_trans.InsertRow(0) ids_sql_trans.object.sql_text[ll_row] = ls_sql ids_sql_trans.object.done[ll_row] = 'N' IF ids_sql_trans.Update() <> 1 then MessageBox(Cannot update please close window without save) ib_pass = false END IF END IF
<i>ControlEvent: dw 1. Dberror</i>
See in w_db_base.dberror
<i>Event: dw 2. ue filter (string as hw id)</i>
dwc_hw_id.SetFilter("hw_regi_hw_id <> '"+as_hw_id+"'") and Filter
<i>ControlEvent: dw 2. Dberror</i>
See in w_db_base.dberror
<i>ControlEvent: dw 2. Sqlpreview</i>
See in dw_1.sqlpreview

[Window] w_trans_control from w_trans_base
[Description] hardware control and registration

Controls	Control Name	Remarks
w trans base	<w trans control>	Title="Register";
w trans base'dw_1	dw_1	DataObject="d control add";
w trans base'dw_2	dw_2	DataObject="d cont detail ff";

Declaration	
Instance Variable	datawindowchild dwc_1, dwc_2 il_detail_no ir_total=0

Script	Type	Name
w_trans_control	Event	ue_add_dw_1 ue_add_dw_2 ue_cal_id ue_check_error ue_hw_search ue_init ue_retrieve_dw_2 ue_update_dw_2
Dw_1	Event	type integer ue_change_total()
	ControlEvent	Buttonclicked ue_check_error
Dw_2	ControlEvent	ue_check_error ue_dropdown

<i>Event: w_trans_control. ue_add_dw_1</i>
Call super::ue_add_dw_1
Set string(il_main_no), gd_today to dw_1
<i>Event: w_trans_control. ue_add_dw_2</i>
Call super::ue_add_dw_2
Set gs_div_id, gs_sect_id, gs_year, string(il_main_no), string(il_detail_no) to dw_2
<i>Event: w_trans_control. ue_cal_id</i>
select max(convert(numeric,cont_no)) Into :ls_main_no from control where div_id = :gs_div_id and sect_id = :gs_sect_id and year = :gs_year; il_main_no = Integer(ls_main_no)+1
<i>Event: w_trans_control. ue_check_error</i>
IF real(dw_1.Object.total_cost[il_row_2]) <> ir_total then MessageBox (Total cost is incorrect, Ask to change) IF Change THEN Dw_1.Event ue_change_total() END IF end if
<i>Event: w_trans_control. ue_hw_search</i>
Call super::ue_hw_search str_hw_code lstr_hw_code lstr_hw_code = Open(w_hw_code_search,This) if lstr_hw_code.s_class_id <> " " then ll_row = dw_2.getrow() set class_id, type_id, spec_id to dw_2 end if
<i>Event: w_trans_control. ue_init</i>
Call super::ue_init is_dataobject = 'd_r_control' This.Post Event ue_add_dw_1()
<i>Event: dw_1.ue_change_total</i>
This.object.total_cost[il_row_1] = ir_total if This.Update() = 1 then //success return 1 end if

<pre> Event: w trans control. ue retrieve dw 2 Call super:: ue_retrieve_dw_2 dw_2.GetChild('type_id',dwc_1) dw_2.GetChild('spec_id',dwc_2) retrieve dwc_1, dwc_2 and dw_2.Retrieve(gs_div_id,gs_sect_id,gs_year,string(il_main_no)) if dw_2.RowCount() > 0 then il_detail_no = dw_2.Rowcount() + 1 else il_detail_no = 1 end if </pre>
<pre> Event: w trans control. ue update dw 2 Call super::ue_update_dw_2 if ib_pass = false then return end if str_cont_d_pk str_parms int li_pass, i_rc IF dw_2.update()<=0 THEN Ib_pass = False Else Set gs_div_id, gs_sect_id, gs_year, String(il_main_no), String(il_detail_no), dw_2.object.class_id[il_row_2],dw_2.object.type_id[il_row_2], dw_2.object.spec_id[il_row_2], dw_2.object.quantity[il_row_2] to str_parms li_pass = OpenWithParm(w_regi_add,str_parms) IF li_pass = 0 Then // not pass, there're error while system had updated regi data MessageBox (Ask to register again) IF register THEN This.TriggerEvent("ue_update_dw_2") ELSEIF cancel THEN Ib_pass = false Disable insert_dw_2 button and set focus dw_2 END IF ELSE Enable insert_dw_2 button and disable update_dw_2 button ir_total= ir_total+ dw_2.Object.cal_cost[il_row_2] ib_pass= true and il_detail_no++ END IF END IF </pre>
<pre> ControlEvent: dw 2.ue dropdown ll_row = This.GetRow() ls_column = This.GetColumnName() IF ls_column = 'type_id' THEN ls_class_id = dw_2.object.class_id[ll_row] IF isNull (ls_class_id) then MessageBox(Want to choose class_id) and Set Column 'class_id' Return End if dwc_1.SetFilter("class_id = '"+ls_class_id+"'") and filter elseif ls_column = 'spec_id' THEN ls_class_id = dw_2.object.class_id[ll_row] , ls_type_id = dw_2.object.type_id[ll_row] dwc_2.setFilter("class_id = '"+ls_class_id+"'and type_id = '"+ls_type_id+"'") and filter END IF </pre>
<pre> ControlEvent: dw 1. Buttonclicked ls_date = Open(w_pb_calendar) if ls_date <> '00/00/0000' then This.SetItem(il_row_1,'receipt_date',date(ls_date)) end if </pre>



ControlEvent: dw 1.ue check error
Check Today >= receipt_date <> null Check budget_year, project_id <> null If method = 1 (buy) then Supplier_id, ship_id <> null end if
ControlEvent: dw 2.ue check error
Check class_id,type_id,spec_id and desc <> null

[Window] w_trans_dist from w_trans_base
[Description] Distribution or borrowing

Controls	Control Name	Remarks
w_trans_base	<w_trans_dist>	Title="Distribute or borrow";
w_trans_base`dw 1	dw 1	DataObject="d_distribute_add";
w_trans_base`dw 2	dw 2	DataObject="d_dist_detail_grid";

Declaration	
Instance Variable	datastore ids_dist_detail, ids_hw_regi

Script	Type	Name
w_trans_dist	Even	Close ue_add_dw_1 ue_add_dw_2 ue_cal_id ue_hw_search ue_init ue_retrieve_dw_2 ue_update_dw_2
dw_1	ControlEvent	Buttonclicked ue_check_error
dw_2	ControlEvent	Buttonclicked ue_check_error

<i>Event: w_trans_dist.Close</i>
Call super::Close Destroy ids_hw_regi Destroy ids_dist_detail
<i>Event: w_trans_dist.ue_add_dw_1</i>
Call super:ue_add_dw_1 Set string(il_main_no), gd_today to dw_1
<i>Event: w_trans_dist.ue_add_dw_2</i>
Call super:: ue_add_dw_2 Set gs_sect_id, gs_sect_id, gs_year, string(il_main_no) to dw_2 set column "dist_detail_hw_id"
<i>Event: w_trans_dist.ue_cal_id</i>
string ls_main_no select max(convert(numeric,dist_no)) Into :ls_main_no from distribute where div_id = :gs_div_id and sect_id = :gs_sect_id and year = :gs_year, il_main_no = Integer(ls_main_no)+1

<i>Event: w trans dist. ue hw search</i>
Call super::ue_hw_search dw_2.object.dist_detail_hw_id[il_row_2] =is_hw_id Set column "hw_regi_use_place"
<i>Event: w trans dist. ue init</i>
Call super::ue_init is_dataobject = 'd_r_dist' This.Post Event ue_add_dw_1()
<i>Event: w trans dist. ue retrieve dw 2</i>
dw_2.getChild("dist_detail_hw_id",dwc_hw_id) ids_dist_detail = Create DataStore "d_ds_dist_detail" ids_hw_regi = Create DataStore "d_ds_hw_regi" retrieve dwc_hw_id, dw_2, ids_dist_detail, ids_hw_regi IF gs_proc = gs_distribute then dw_2.modify("dist_detail_return_date.visible = '0'") invisible dw_2.object.btn_date END IF
<i>Event: w trans dist. ue update dw 2</i>
Call super:: ue_update_dw_2 string ls_use_place,ls_status, ls_hw_id ,ls_dist_no date ld_return_date if ib_pass = false then return end if if gs_proc = gs_distribute then ls_status = "2" and ld_return_date = Date("01/01/0001") elseif gs_proc = gs_borrow then ls_status = "3" and ld_return_date = dw_2.object.dist_detail_return_date[il_row_2] end if Get ls_use_place, ls_hw_id from dw_2 Set all items to ids_dist_detail Update ls_status and ls_use_place to ids_hw_regi Insert SQL statement of ids_dist_detail and ids_hw_regi to ids_sql_trans Invisible update_dw_2 and enable insert_dw_2 dw_2.Post Event ue_Filter(ls_hw_id)
<i>ControlEvent: dw 1.ButtonClicked</i>
ls_date = Open(w_pb_calendar) if ls_date <> '00/00/0000' then This.SetItem(il_row_1,'date',date(ls_date)) end if
<i>ControlEvent: dw 1.ue check error</i>
Check Today >= date <> null Check recipient_id,distributor_id <>null
<i>ControlEvent: dw 2.ButtonClicked</i>
ls_date = Open(w_pb_calendar) if ls_date <> '00/00/0000' then This.SetItem(il_row_2,'dist_detail_return_date',date(ls_date)) end if
<i>ControlEvent: dw 2.ue check error</i>
Check hw_id<>null If gs_proc = borrow then Check today <=return_date<>null End if

[Window] w_trans_disp from w_trans_base
 [Description] Dispose hardware

Controls	Control Name	Remarks
w_trans_base	<w_trans_disp>	Title="Dispose";
w_trans_base'dw_1	dw_1	DataObject="d_dispose_add";
w_trans_base'dw_2	dw_2	DataObject="d_disp_detail_tb";

Declaration	
Instance Variable	datastore ids_hw_regi

Script	Type	Name
w_trans_disp	Event	Close ue_add_dw_1 ue_add_dw_2 ue_cal_id ue_hw_search ue_init ue_retrieve_dw_2 ue_update_dw_2
dw_1	ControlEvent	ButtonClicked ue_check_error
dw_2	ControlEvent	ue_check_error

```

Event: w_trans_disp.Close
Call super:: Close
Destroy ids_hw_regi
Event: w_trans_disp.ue_add_dw_1
Call super:: ue_add_dw_1
Set string(il_main_no), gd_today to dw_1
Event: w_trans_disp.ue_add_dw_2
Call super:: ue_add_dw_2
Set gs_div_id, gs_sect_id, gs_year, string(il_main_no) to dw_2
Set column "hw_id"
Event: w_trans_disp.ue_cal_id
Select max(convert(numeric,disp_no)) Into :ls_main_no from dispose
Where div_id = :gs_div_id and sect_id = :gs_sect_id and year = :gs_year,
il_main_no = Integer(ls_main_no)+1
Event: w_trans_disp.ue_hw_search
Call super:: ue_hw_search
dw_2.object.hw_id[il_row_2] =is_hw_id
Set column "method"
Event: w_trans_disp.ue_init
Call super:: ue_init
is_dataobject = 'd_r_disp'
This.Post Event ue_add_dw_1()
Event: w_trans_disp.ue_retrieve_dw_2
dw_2.getChild("hw_id",dwc_hw_id)
ids_hw_regi = Create DataStore "d_ds_hw_regi"
Retrieve dwc_hw_id, dw_2, ids_hw_regi
ControlEvent: dw_1.ue_check_error
Check Today>=date <null
    
```

<i>Event: w trans disp. ue update dw 2</i>
Call super:: ue_update_dw_2 if ib_pass = false then return end if IF dw_2.update()<=0 THEN ib_pass = False ELSE Update ids_hw_regi.status = "6" // dispose Insert SQL statement of ids_hw_regi to ids_sql_trans dw_2.Post Event ue_Filter(ls_hw_id) END IF
<i>ControlEvent: dw 1.ButtonClicked</i>
ls_date = Open(w_pb_calendar) if ls_date <> '00/00/0000' then This.SetItem(il_row_1,'date',date(ls_date)) end if
<i>ControlEvent: dw 2.ue check error</i>
Check hw_id,cause,method <.>null If gs_proc = borrow then Check today <=return_date<>null End if

[Window] w_trans_repa from w_trans_base
 [Description] Repair

Controls	Control Name	Remarks
w trans base	<w trans_repa>	Title="Repair";
w_trans_base`dw 1	dw 1	DataObject="d repair add";
w_trans_base`dw 2	dw 2	DataObject="d_repa_detail_grid";

Declaration	
Instance Variable	Datastore ids_hw_regi

Script	Type	Name
w_trans_repa	Event	Close ue_add_dw_1 ue_add_dw_2 ue_cal_id ue_hw_search ue_init ue_retrieve_dw_2 ue_update_dw_2
dw_1	ControlEvent	Buttonclicked ue_check_error
dw_2	ControlEvent	Buttonclicked ue_check_error

<i>Event: w trans_repa.Close</i>
Call super::Close Destroy ids_hw_regi

<i>Event: w trans repa. ue add dw 1</i>
Call super::ue_add_dw_1 Set string(il_main_no), gd_today to dw_1 Set column "supplier id"
<i>Event: w trans repa. Ue add dw 2</i>
Call super::ue_add_dw_2 Set gs_div_id, gs_sect_id, gs_year, string(il_main_no) to dw_2 set column "hw id"
<i>Event: w trans repa. Ue cal id</i>
select max(convert(numeric, repa_no)) Into :ls_main_no from repair where div_id = :gs_div_id and sect_id = :gs_sect_id and year = :gs_year, il_main_no = Integer(ls_main_no)+1
<i>Event: w trans repa. ue hw search</i>
Call super::ue_hw_search dw_2.object.hw_id[il_row_2] = is_hw_id Set column "cause"
<i>Event: w trans dist. ue init</i>
Call super::ue_init is_dataobject = 'd_r_repa' This.Post Event ue_add_dw_1()
<i>Event: w trans repa. Ue retrieve dw 2</i>
dw_2.getChild("hw_id", dwc_hw_id) ids_hw_regi = Create DataStore "d_ds_hw_regi" Retrieve dww_hw_id, ids_hw_regi, and dw_2
<i>Event: w trans repa. ue update dw 2</i>
Call super::ue_update_dw_2 If ib_pass = false then Return end if IF dw_2.update() <= 0 THEN ib_pass = False ELSE Update ids_hw_regi.status = "5" // repair Insert SQL statement of ids_hw_regi to ids_sql_trans dw_2.Post Event ue_Filter(ls_hw_id) END IF
<i>ControlEvent: dw 1.ButtonClicked</i>
ls_date = Open(w_pb_calendar) If ls_date < '00/00/0000' then This.SetItem(il_row_1, 'date', date(ls_date)) End if
<i>ControlEvent: dw 1.ue check error</i>
Check Today >= date < null Check supplier id < null
<i>ControlEvent: dw 2.ButtonClicked</i>
ls_date = Open(w_pb_calendar) if ls_date < '00/00/0000' then IF dwo.name = "cb_repair" then This.SetItem(il_row_2, 'repair_date', date(ls_date)) elseif dwo.name = "cb_return" then This.SetItem(il_row_2, 'return_date', date(ls_date)) end if end if
<i>ControlEvent: dw 2.ue check error</i>
Check hw_id, cause < null

[Window] w_trans_retu from w_trans_base
 [Description] Return hardware from borrower

Controls	Control Name	Remarks
w_trans_base	<w_trans_retu>	Title="Return";
w_trans_base'dw_1	dw_1	DataObject="d_return_add";
w_trans_base'dw_2	dw_2	DataObject="d_retu_detail_grid";

Declaration	
Instance Variable	Datastore ids_dist_detail, ids_hw_regi long il_dist

Script	Type	Name
w_trans_retu	Event	Close ue_add_dw_1 ue_add_dw_2 ue_cal_id ue_hw_search ue_init ue_retrieve_dw_2 ue_update_dw_2
dw_1	ControlEvent	Buttonclicked ue_check_error
dw_2	Event	type integer ue_search_dist no (string as hw_id)
	ControlEvent	Itemfocuschanged ue_check_error

<i>Event: w_trans_retu.Close</i>
Call super::ue_close Destroy ids_hw_regi and ids_dist_detail
<i>Event: w_trans_retu.ue_add_dw_1</i>
Call super::ue_add_dw_1 Set string(il_main_no), gd_today to dw_1
<i>Event: w_trans_retu.ue_add_dw_2</i>
Call super::ue_add_dw_2 Set gs_div_id, gs_sect_id, gs_year, string(il_main_no) to dw_2 Set column "hw_id"
<i>Event: w_trans_retu.ue_cal_id</i>
select max(convert(numeric,retu_no)) Into :ls_main_no from "return" where div_id = :gs_div_id and sect_id = :gs_sect_id and year = :gs_year; il_main_no = Integer(ls_main_no)+1
<i>Event: w_trans_retu.ue_hw_search</i>
Call super::ue_hw_search dw_2.object.hw_id[il_row_2] = is_hw_id Set column "status"
<i>Event: w_trans_retu.ue_init</i>
Call super::ue_init is_dataobject = 'd_r_retu' This.Post Event ue_add_dw_1()
<i>Event: w_trans_retu.ue_retrieve_dw_2</i>
dw_2.getChild("hw_id",dwc_hw_id) ids_hw_regi = Create DataStore "d_ds_hw_regi" ids_dist_detail = Create DataStore "d_ds_dist_detail_search" Retrieve dwc_hw_id, ids_hw_regi, ids_dist_detail and dw_2

<p><i>Event: w trans retu. ue update dw 2</i></p> <p>Call super::ue_update_dw_2 if ib_pass = false then return end if IF dw_2.update()<=0 THEN ib_pass = False ELSE If dw_2.status = 1 then Update ids_hw_regi.status = '1' Else Update ids_hw_regi.status = '4' End if Update status of ids_dist_detail = "0" Insert SQL statement of ids_hw_regi and ids_dist_detail to ids_sql_trans dw_2.Post Event ue_Filter(ls_hw_id) END IF</p>
<p><i>ControlEvent: dw 1.ButtonClicked</i></p> <p>ls_date =Open(w_pb_calendar) if ls_date <> '00/00/0000' then This.SetItem(il_row_1,'date',date(ls_date)) end if</p>
<p><i>ControlEvent: dw 1.ue check error</i></p> <p>Check Today>=date <>null Check return_id, recipient_id <> null</p>
<p><i>Event: dw 2.ue search dist no.</i></p> <p>il_dist = ids_dist_detail.Find("hw_id = '"+as_hw_id+"'",1,ids_dist_detail.RowCount()) This.object.Dist_no[il_row_2] = ids_dist_detail.object.dist_no[il_dist] This.object.Dist_year[il_row_2] = ids_dist_detail.object.year[il_dist] Return 1</p>
<p><i>ControlEvent: dw 2.itemfocuschanged</i></p> <p>if dwo.name = "status" then ls_findString = " hw_id = '"+ This.object.hw_id[il_row_2]+'" ll_Found = dwc_hw_id.Find(ls_FindString,1,dwc_hw_id.RowCount()) i_rc = This.Event ue_search_dist_no(ls_search) if i_rc = 0 then //fail This.SetColumn("hw_id") end if end if</p>
<p><i>ControlEvent: dw 2.ue check error</i></p> <p>Check hw_id, status <> null</p>

[Window] w_check_repair from Window
[Description] Update repaired hardware by changing status to wait and adding '**' to cause

Controls	Control Name	Remarks
Window	<w_check_repair>	Menu="m_check_repair"; Title="receive repaired hardware";
Datavindow	dw_1	Tab=40; Disable; DataObject="d ff repa detail";

Declaration	
Shared Variable	string ss_list
Instance Variable	boolean ib_pass = false datastore ids_hw_regi, ids_sql_trans

Script	Type	Name
w_check_repair	Event	Close Closequery Open ue_first () ue_init () ue_last () ue_next () ue_prev () ue_search_hw_id () ue_update_repair ()
dw_1	Event	ue_find (string as hw_id)
	ControlEvent	Dberror Sqlpreview

<i>Event: w check repair.Close</i>
IF ib_pass = True Then Commit; Else rollback; end if Destroy ids_hw_regi and ids_sql_trans
<i>Event: w check repair.Closequery</i>
See in w_db_base but using dw_1 instead of dw_2
<i>Event: w check repair.Open</i>
This.Post Event ue_init()
<i>Event: w check repair.ue_first ()</i>
dw_1.ScrollToRow(1)
<i>Event: w check repair.ue_init ()</i>
ids_sql_trans = Create DataStore "d_ds_sql_trans" ids_hw_regi = Create DataStore "d_ds_hw_regi" Retrieve dw_1, ids_sql_trans and ids_hw_regi dw_1.SetFilter("cause not like '*%'") and Filter()
<i>Event: w check repair.ue_last ()</i>
dw_1.ScrollToRow(dw_1.RowCount())
<i>Event: w check repair.ue_next ()</i>
dw_1.ScrollNextRow()
<i>Event: w check repair.ue_prev ()</i>
dw_1.ScrollPriorRow()
<i>Event: w check repair.ue search hw id ()</i>
ls_hw_id = OpenWithParm(w_hw_id_search,1) dw_1.Post Event ue_find(ls_hw_id)
<i>Event: dw 1. ue find (string as hw id)</i>
if as_hw_id <> " " then dw_1.Find("hw_id = '"+as_hw_id+"'", 1, dw_1.RowCount()) dw_1.ScrollToRow(ll_found) end if

<i>Event: w check repair. ue update repair ()</i>
dw_1.getrow() Insert '*' in dw_1.cause(currow) IF dw_1.update()<=0 THEN ib_pass = False else ids_hw_regi.Find("hw_id = '"+ls_hw_id+ "'",1, ids_hw_regi.RowCount()) Update ids_hw_regi.Object.status[ll_row] = "1" //wait END IF
<i>ControlEvent: dw_1.dberror</i>
See in w_db_base.dw_2.dberror
<i>ControlEvent: dw_1.sqlpreview</i>
See in w_trans_base.dw_1.Sqlpreview

[Window] w_trans_sql from Window
[Description] Prepare SQL statement data

Controls	Control Name	Remarks
Window	<w_trans_sql>	Title="Prepare SQL data";
datawindow	dw_2	Tab=10; DataObject="d_sql_trans";
picturebutton	pb_ok	Tab=20;
statictext	st_status	

Declaration	
Shared Variable	string ss_list
Instance Variable	datastore ids_sql_trans

Script	Type	Name
w_trans_sql	Event	Open ue_init () ue_update ()
dw_2	ControlEvent	dberror
pb_ok	ControlEvent	clicked

<i>Event: w_trans_sql.open</i>
This.Post Event ue_init()
<i>Event: w_trans_sql.ue init()</i>
ids_sql_trans = Create DataStore "d_ds_edit_sql" Retrieve ids_sql_trans and dw_2
<i>Event: w_trans_sql.ue update()</i>
ll_row = ids_sql_trans.RowCount() FOR ll_count=1 TO ll_row ids_sql_trans.object.done[ll_count] = 'Y' NEXT if ids_sql_trans.Update() > 0 then commit; else RollBack; end if

<i>ControlEvent: dw 2.dberror</i>
See in w_db_base.dw 2.dberror
<i>ControlEvent: pb ok.clicked</i>
<pre>i_rc = dw_2.saveas("",text!,False) if i_rc = 1 then Parent.Trigger event ue_update() end if</pre>

[Window] w_trans_print from w_print_base
[Description] Print transaction report

Controls	Control Name	Remarks
W_print_base	<w_trans_print>	Menu="m_trans_print";

Script	Type	Name
w_trans_print	Event	ue_init

<i>Event w_trans_print.ue_init()</i>
<pre>str_trans_print str_x string ls_rc str_x = Message.PowerObjectParm dw_1.dataobject = str_x.s_dataobject dw_1.SetTransObject(SQLCA) dw_1.Retrieve(gs_div_id,gs_sect_id,gs_year,str_x.s_no) This.Post event Ue_set_page() dw_1.modify("DataWindow.Print.Preview = yes")</pre>

Editing process

[Window] w_trans_edit_base from Window
[Description] ancestor window of transaction editing process

Controls	Control Name	Remarks
Window	<w_trans_edit_base>	Menu="m_trans_edit";
datawindow	dw_1	Tab=20; Disable;
datawindow	dw_2	Tab=10;

Declaration	
Instance Variable	<pre>boolean ib_needsave = false, ib_pass = false long il_row_1, il_row_2, il_row string is_hw_id, is_no datastore ids_sql_trans</pre>

Script	Type	Name
w_trans_edit_base	Event	<pre>Close Closequery Open ue_cancel () ue_init () ue_retrieve ()</pre>

Script	Type	Name
		ue_search () ue_undo () ue_update () ue update dw 2 ()
	PrivateFunc	wf_undo (datawindow adw_x) return integer
Dw_1	ControlEvent	Dberror Itemchanged Sqlpreview
Dw_2	Event	ue_check_error ()
	ControlEvent	dberror itemchanged sqlpreview

<i>Event: w trans edit base.Close</i>
IF ib_pass Then Commit; Else rollback; end if Destroy ids_sql_trans and close(This)
<i>Event: w trans edit base.Closequery</i>
See in w_db_base.Closequery but they check dw_1 and dw_2
<i>Event: w trans edit base.Open</i>
This.Post Event ue_init()
<i>Event: w trans edit base.ue cancel()</i>
See in w_db_base.ue_cancel
<i>Event: w trans edit base.ue init()</i>
ids_sql_trans = Create DataStore "d_ds_sql_trans" and invisible dw_1 and dw_2 This.Trigger Event ue_search()
<i>Event: w trans edit base.ue retrieve()</i>
if is_no <> 'n' then dw_1.Retrieve(gs_div_id,gs_sect_id,gs_year,is_no) and visible dw_1 il_row_1 = dw_1.GetRow() This.Trigger event ue_retrieve_dw_2() end if
<i>Event:w trans edit base.ue search</i>
is_no = open(w_trans_q_no,This) This.Trigger Event ue_retrieve()
<i>Event:w trans edit base.ue undo</i>
if wf_undo(dw_1)= 0 then wf_undo(dw_2) end if
<i>Event: w trans edit base.ue update()</i>
dw_1.TriggerEvent ("ue_check_error") If ib_pass then IF dw_1.Update() = 1 Then This.Trigger Event ue_update_dw_2() Else ib_pass =false and set focus dw_1 end if End IF
<i>Event: w trans edit base.ue update dw 2</i>
dw_2.TriggerEvent ("ue check error")

<i>PrivateFunction:w trans edit base.wf undo (datawindow adw x) return integer</i>
Get the original value of adw_x row/column Reset it to the original value Reset the modified flag for adw_x row
<i>ControlEvent: dw 1.dberror</i>
See in w_db_base.dw_2.dberror
<i>ControlEvent: dw 1.Itemchanged</i>
ib_needsave = True
<i>ControlEvent: dw 1.sqlpreview</i>
See in w_trans_base.dw_1.sqlpreview
<i>Event: dw 2.ue check error()</i>
long ll_row ll_row = This.RowCount() FOR il_row 2 = 1 TO ll_row This.Trigger Event ue_check_fn() if not(ib_pass) then return end if NEXT
<i>ControlEvent: dw 2.dberror</i>
See in w_db_base.dw_2.dberror
<i>ControlEvent: dw 2.itemchanged</i>
ib_needsave = True
<i>ControlEvent: dw 2.sqlpreview</i>
See in w_trans_base.dw_2.sqlpreview

[Window] w_trans_edit_cont from w_trans_edit_base
[Description] Edit registraton data

Controls	Control Name	Remarks
w_trans_edit_base	<w_trans_edit_cont>	Title="Edit registration";
w_trans_edit_base'dw 1	dw 1	DataObject="d control edit";
w_trans_edit_base'dw 2	dw 2	DataObject="d cont_detail edit";
datawindow	dw 3	Tab=20; DataObject="d_hw_regi_edit";

Declaration	
Instance Variable	datawindowchild dwc 1, dwc 2

Script	Type	Name
w_trans_edit_cont	Event	ue_retrieve_dw_2 ue_update_dw_2
dw_1	ControlEvent	Buttonclicked ue_check_error
dw_2	Event	ue_dropdown_pbm_dwndropdown
	ControlEvent	Rowfocuschanged ue_check_fn
dw_3	Event	ue_filter (string as no)
	ControlEvent	sqlpreview

<i>Event: w trans edit cont.ue retrieve dw 2</i>
dw_2.GetChild('type_id',dwc_1) dw_2.GetChild('spec_id',dwc_2) retrieve dwc_1, dwc_2, dw_2 ,dw_3 and visible dw_2 and dw_3 ls_no = dw_2.Object.no[1] dw_3.Post Event ue_filter(ls_no)
<i>Event: w trans edit cont.ue update dw 2</i>
Call super:: ue_update_dw_2 If ib_pass then IF dw_2.Update() <>1 and dw_3. Update<> 1Then ib_pass =false end if End IF
<i>ControlEvent: dw 2.Rowfocuschanged</i>
ls_no = This.GetItemString(this.getRow(),"no") dw_3.Post Event ue_filter(ls_no)
<i>Event: dw 2.ue filter(string as no)</i>
clear old filter This.SetFilter("no = "+ as_no + " ") and Filter
<i>ControlEvent: dw 3.sqlpreview</i>
See in w_trans_control dw_1.sqlpreview ** dw_1.ButtonClicked, dw_1.ue_check_error and dw_2.ue_dropdown see in w_trans_control dw_2. ue check fn see in w_trans_control.dw 2.ue check error

[Window] w_trans_edit_disp from w_trans_edit_base
[Description] Edit disposal data

Controls	Control Name	Remarks
w trans edit base	<w trans edit disp>	Title="Edit disposal";
w trans edit base`dw 1	dw 1	DataObject="d dispose edit";
w trans edit base`dw 2	dw 2	DataObject="d disp detail";

Script	Type	Name
w_trans_edit_disp	Event	ue_retrieve_dw_2 ue_update_dw_2
dw_1	ControlEvent	Buttonclicked ue_check_error
dw_2	ControlEvent	ue_check_fn

<i>Event: w trans edit disp.ue retrieve dw 2</i>
If retrieve dw_2 is no error then visible dw_2 end if
<i>Event: w trans edit disp.ue update dw2</i>
Call super:: ue_update_dw_2 if ib_pass = false then return end if IF dw_2.update()<=0 THEN ib_pass = False and dw_2.SetFocus() end if

**** dw_1.ButtonClicked and dw_1.ue_check_error see in w_trans_disp
dw_2.ue_check_fn see in w_trans_disp.dw_2.ue_check_error**

**[Window] w_trans_edit_dist from w_trans_edit_base
[Description] Edit distribution data**

Controls	Control Name	Remarks
w_trans_edit_base	<w_trans_edit_dist>	Title="Edit distribution";
w_trans_edit_base`dw_1	Dw_1	DataObject="d_distribute_edit";
w_trans_edit_base`dw_2	Dw_2	DataObject="d_dist_detail_edit";

Declaration	
Instance Variable	Datastore ids_dist_detail, ids_hw_regi

Script	Type	Name
w_trans_edit_dist	Event	ue_retrieve_dw_2 ue_update_dw_2 ue_update_fn()
dw_1	ControlEvent	ue_check_error

Event: w_trans_edit_dist.ue_retrieve_dw_2
ids_dist_detail = Create DataStore "d_ds_dist_detail"
ids_hw_regi = Create DataStore "d_ds_hw_regi"
retrieve dw_2, ids_dist_detail, ids_hw_regi and visible dw_2

Event: w_trans_edit_dist.ue_update_dw_2
Call super:: ue_update_dw_2
if ib_pass = false then
 return
end if
ll_row = dw_2.RowCount()
For il_row_2 = 1 to ll_row
 This.Trigger event ue_update_fn()
Next

Event: w_trans_edit_dist.ue_update_fn
Get ld_return_date, ls_use_place, ls_hw_id from dw_2
Find hw_id in ids_dist_detail
Update ld_return_date to ids_dist_detail
update use place in hw_regi table
insert sql statement of ids_dist_detail and hw_regi in ids_sql_trans

ControlEvent: dw_1.ue_check_error
See in w_trans_dist.dw_1.ue_check_error

**[Window] w_trans_edit_repa from w_trans_edit_base
[Description] Edit repair data**

Controls	Control Name	Remarks
w_trans_edit_base	<w_trans_edit_repa>	Title="Edit repair";
w_trans_edit_base`dw_1	dw_1	DataObject="d_repair_edit";
w_trans_edit_base`dw_2	dw_2	DataObject="d_repa_detail_edit";

Script	Type	Name
w_trans_edit_repa	Event	ue retrieve_dw_2 ue update_dw_2
dw_1	ControlEvent	ue check error
dw_2	ControlEvent	Buttonclicked ue check fn

<i>Event: w_trans_edit_repa.ue retrieve_dw_2</i>
ids_dist_detail = Create DataStore "d_ds_dist_detail" ids_hw_regi = Create DataStore "d_ds_hw_regi" retrieve dw_2, ids_dist_detail, ids_hw_regi and visible dw_2
<i>Event: w_trans_edit_repa.ue update_dw_2</i>
Call super:: ue_update_dw_2 if ib_pass = false then return end if IF dw_2.update()<=0 THEN ib_pass = False and dw_2.SetFocus() end if
<i>** dw_1.ue_check_error and dw_2.buttonclicked see in w_trans_repa dw_2.ue_check_fn see in w_trans_repa.ue_check_error</i>

[Window] w_trans_edit_retu from w_trans_edit_base
[Description] Edit return data

Controls	Control Name	Remarks
w_trans_edit_base	<w_trans_edit_retu>	Title="Edit return";
w_trans_edit_base`dw_1	dw_1	DataObject="d_return_edit";
w_trans_edit_base`dw_2	dw_2	DataObject="d_retu_detail_edit";

Declaration	
Instance Variable	Datastore ids_hw_regi

Script	Type	Name
w_trans_edit_retu	Event	ue retrieve_dw_2 ue update_dw_2 ue update_fn()
dw_1	ControlEvent	Buttonclicked ue check error

<i>Event: w_trans_edit_retu.ue retrieve_dw_2</i>
ids_hw_regi = Create DataStore "d_ds_hw_regi" Retrieve dw_2, ids_hw_regi and visible dw_2
<i>Event: w_trans_edit_retu.ue update_dw_2</i>
Call super:: ue_update_dw_2 if ib_pass = false then return end if For il_row_2 = 1 to dw_2.RowCount() This.Trigger event ue_update_fn() Next

<i>Event: w trans edit retu.ue update fn</i>
<pre> long ll_row string ls_hw_id, ls_status IF dw_2.update()<=0 THEN ib_pass = False else get ls_hw_id from dw_2 if dw_2.object.status[il_row_2] = "1" then //good ls_status = "1" elseif dw_2.object.status[il_row_2] = "2" then //bad ls_status = "4" end if Find ls_hw_if in ids_hw_regi and update status = ls_status Insert sql statement of ids_hw_regi to ids_sql_trans End if </pre>
<i>** dw 1.ue check error and dw 2.buttonclicked see in w trans retu</i>

[Window] w_trans_q_no from Window

[Description] Get editing number of transaction and display the data to edit

Controls	Control Name	Remarks
Window	<w trans q no>	Type=response!; Title="Get number";
Commandbutton	cb_cancel	Tab=40; Text="Cancel";
commandbutton	cb_ok	Tab=30; Text="OK";
editmask	em_no	Tab=10;
groupbox	gb_1	Tab=50; Text="Enter requested number";

Declaration	
Instance Variable	string is_no = 'n'

Script	Type	Name
w_trans_q_no	Event	Close
cb_cancel	ControlEvent	Clicked
cb_ok	ControlEvent	Clicked

<i>Event: w_trans_q_no.Close</i>
CloseWithReturn(This,is_no)
<i>ControlEvent:cb_ok.clicked</i>
<pre> IF isNull(em_no.text) then MessageBox(Warn to enter number) Else is_no = em_no.text CloseWithReturn(Parent,is_no) end if </pre>
<i>ControlEvent: cb_cancel.clicked</i>
<pre> is_no = 'n' CloseWithReturn(Parent,is_no) </pre>

3. Query and Report

These windows create pre-define results and reports. These outputs divide into 3 style: tabular, graph and report. Tabular has 3 windows, which display overall or deep in detail data. Graph has an ancestor window, 2 descendent windows and 2 shared window. Report has a window to preview and print. All 3 styles have to call w_set_agr window that is described in section 4. Shared window.

Tabular result

[Window] w_q_hw_regi_desc from Window
 [Description] Hardware data and status of requested HW ID

Controls	Control Name	Remarks
Window	<w_q_hw_regi_desc>	Menu="m_queries_main"; Title="Current status of hardware";
Commandbutton	cb_ok	Tab=20; Text="OK";
Commandbutton	cb_search	Tab=30; Text="SEARCH";
Datawindow	dw_1	Tab=50;Disable; DataObject="d ff_hw_detail_desc";
datawindow	dw_2	Tab=40;
groupbox	gb_hw_id	
singlelineedit	sle_hw_id	Tab=10;
statictext	st_1	Text="HW ID:";

Declaration	
Instance Variable	boolean ib_retu = false datastore ids_dist, ids_retu, ids_repa, ids_disp string is status='0', is expression, is hw_id

Script	Type	Name
w_q_hw_regi_desc	Event	Close open ue_init ()
cb_ok	ControlEvent	clicked
cb_search	ControlEvent	clicked
dw_1	Event	ue_find () ue_search (string as hw_id)
dw_2	Event	ue_find ()

<i>Event: w q hw regi desc.close</i>
Destroy ids_dist, ids_disp, ids_repa and ids_retu
<i>Event: w q hw regi desc.open</i>
Post Event ue_init()
<i>ControlEvent:cb_ok.clicked</i>
if sle_hw_id.text <> " " then dw_1.Post Event ue_search(sle_hw_id.text) end if

<i>Event: w q hw regi desc.ue init()</i>
invisible dw_1 and dw_2 ids_dist = Create DataStore "d_ff_dist_desc" ids_disp = Create DataStore "d_ff_disp_desc" ids_retu = Create DataStore "d_ff_retu_desc" ids_repa = Create DataStore "d_ff_repa_desc" retrieve dw_1 and all datastore
<i>ControlEvent:cb_search.Clicked</i>
ls_hw_id = OpenWithParm(w_hw_id_search,2) if ls_hw_id <> " " then sle_hw_id.text = ls_hw_id dw_1.Post Event ue_search(ls_hw_id) end if
<i>Event:dw_1.ue find()</i>
ll_row = This.GetRow() is_hw_id = This.object.hw_regi_hw_id[ll_row] is_status = This.Object.hw_regi_status[ll_row] dw_2.Post Event ue_find()
<i>Event:dw_1.ue search(string as hw id)</i>
ll_found = dw_1.Find("hw_regi_hw_id = '"+as_hw_id+"'", 1, dw_1.RowCount()) IF ll_found > 0 THEN dw_1.ScrollToRow(ll_found) visible dw_1 and invisible dw_2 dw_1.Post Event ue_find() ELSE Invisible dw_1 and dw_2 MessageBox(Not found message) END IF
<i>Event: dw_2.ue find()</i>
this.SetFilter("") and Filter // clear old filter CHOOSE CASE is_status CASE "1" // wait if not ib_retu then ids_retu.SetFilter("retu_detail_status = '1' ") and Filter is_expression = "retu_detail_hw_id = '"+is_hw_id+"" This.DataObject = ids_retu.DataObject ids_retu.ShareData(dw_2) and ib_retu = True end if CASE "2" // use ids_dist.SetFilter("dist_detail_status = '1' ") and Filter is_expression = "dist_detail_hw_id = '"+is_hw_id+"" This.DataObject = ids_dist.DataObject and ids_dist.ShareData(dw_2) CASE "3" //borrow ids_dist.SetFilter("dist_detail_status = '2' ") and Filter is_expression = "dist_detail_hw_id = '"+is_hw_id+"" This.DataObject = ids_dist.DataObject and ids_dist.ShareData(dw_2) CASE "4" //bad ids_retu.SetFilter("retu_detail_status = '2' ") and Filter() is_expression = "retu_detail_hw_id = '"+is_hw_id+"" This.DataObject = ids_retu.DataObject and ids_retu.ShareData(dw_2) CASE "5" //repair ids_repa.SetFilter("repa_detail_cause not like '*%'") and Filter() is_expression = "repa_detail_hw_id = '"+is_hw_id+"" This.DataObject = ids_repa.DataObject and ids_repa.ShareData(dw_2)

```

CASE "6" // dispose
    is_expression = "disp_detail_hw_id = '"+is_hw_id+"'"
    This.DataObject = ids_disp.DataObject and ids_disp.ShareData(dw_2)
END CHOOSE

ll_found = This.Find(is_expression,1,This.RowCount())
if ll_found > 0 then
    this.ScrollToRow(ll_found) and visible dw_2
elseif is_status = "1" then // wait may be in "wait" status coz it repaired
    This.SetFilter("") and Filter // clear old filter
    ids_repa.SetFilter("repa_detail_cause like '*%'" ) and Filter
    is_expression = "repa_detail_hw_id = '"+is_hw_id+"'"
    This.DataObject = ids_repa.DataObject and ids_repa.ShareData(dw_2)
    ib_retu = false
    ll_found = This.Find(is_expression,1,This.RowCount())
    if ll_found > 0 then
        This.ScrollToRow(ll_found) and visible dw_2
    end if
end if
end if
    
```

[Window] w_q_hw_spec_show from Window
 [Description] Display hardware data of requested specification and section

Controls	Control Name	Remarks
Window	<w_q_hw_spec_show>	Menu="m_queries_main"; Title="Hardware data in identified specifitycaiton and section";
datawindow	dw_1	Tab=10; DataObject="d tb hw spec list";
datawindow	dw_2	Tab=20; DataObject="d tb hw id spec list";

Script	Type	Name
w_q_hw_spec_show	Event	Open ue_init()
dw_1	ControlEvent	Doubleclicked Rowfocuschanged

```

Event:w q hw spec show.Open
Post Event ue_init()

Event:w q hw spec show.ue_init()
str_q_hw str_x
str_x = Message.PowerObjectParm
dw_1.Retrieve(str_x.s div id,str_x.s sect id,str_x.s class id,str_x.s type id,str_x.s spec id)

ControlEvent:dw_1.Doubleclicked
str_cont_d_pk str_data
ll_row = dw_1.getRow()
get div_id, sect_id, year, cont_no, no from dw_1
set these items to str_data
dw_2.Retrieve(str_data.s div id,str_data.s sect id,str_data.s year,str_data.s cont no,str_data.s no)

ControlEvent:dw_1.rowfocuschanged
SelectRow(0,false)
SelectRow(currentrow,True)
ScrollToRow(currentrow)
    
```

[Window] w_q_hw_status_show from Window
[Description] Display hardware list in requested status , section and specification

Controls	Control Name	Remarks
Window	<w_q_hw_status_show>	Menu="m_queries_main"; Title="Hardware list of identified status";
datawindow	dw_1	Tab=10;

Declaration	
Instance Variable	str_q_hw_istr_x

Script	Type	Name
w_q_hw_status_show	Event	Open ue_init()

Event: w_q_hw_status_show.open

Post Event ue_init()

Event: w_q_hw_status_show.ue_init()

istr_x = Message.PowerObjectParm

CHOOSE CASE istr_x.s_status

CASE '10'

dw_1.dataobject = 'd_tb_retu_borr_date'

dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id,istr_x.d_startdate,istr_x.d_stopdate)

CASE '11'

dw_1.dataobject = 'd_tb_retu_borr_date_spec'

dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id,istr_x.d_startdate,istr_x.d_stopdate,
istr_x.s_class_id,istr_x.s_type_id,istr_x.s_spec_id)

CASE '20'

dw_1.dataobject = 'd_tb_retu_repa_date'

dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id,istr_x.d_startdate,istr_x.d_stopdate)

CASE '21'

dw_1.dataobject = 'd_tb_retu_repa_date_spec'

dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id,istr_x.d_startdate,istr_x.d_stopdate,
istr_x.s_class_id,istr_x.s_type_id,istr_x.s_spec_id)

CASE ELSE

dw_1.dataobject = 'd_tb_hw_status_list'

dw_1.Retrieve(istr_x.s_div_id,istr_x.s_sect_id,istr_x.s_class_id,istr_x.s_type_id,
istr_x.s_spec_id,istr_x.s_status)

END CHOOSE

Graph result

[Window] w_graph_base from Window
[Description] ancestor window for graph result in query process

Controls	Control Name	Remarks
Window	<w_graph_base>	Menu="m_graph_main";
datawindow	dw_1	Tab=10;
statictext	st_1	Text="click right button to display graph value";
statictext	st_popup	Invisible; Text="Popup";

Script	Type	Name
w_graph_base	Event	Close Open ue_graph_type ue_spacing
dw_1	Event	ue_rbuttonup pbm_rbuttonup
	ControlEvent	Rbuttondown

<i>Event: w_graph_base.Close</i>
Close(This)
<i>Event: w_graph_base.Open</i>
Post event ue_init()
<i>Event: w_graph_base.ue_graph_type</i>
OpenWithParm (w_graph_type, dw_1)
<i>Event: w_graph_base.ue_spacing</i>
OpenWithParm (w_graph_spacing, dw_1)
<i>Event: dw_1.ue_rbuttonup pbm_rbuttonup</i>
st_popup.visible = false
<i>ControlEvent: dw_1.Rbuttondown</i>
grObjectType ClickedObject string ls_grgraphname="gr_1" int li_series, li_category ClickedObject = this.ObjectAtPointer (ls_grgraphname, li_series, li_category) If ClickedObject = TypeData! Then st_popup.text = string(this.GetData(ls_grgraphname, li_series, li_category)) + " units" st_popup.x = parent.PointerX() st_popup.y = parent.PointerY() - 65 st_popup.visible = true End If

[Window] w_g_sum_quantity from w_graph_base
[Description] Display summary quantity graph that have series property

Controls	Control Name	Remarks
w_graph_base	<w_g_sum_quantity>	Title="Summary hardware quantity graph";
commandbutton	cb_ok	Tab=20; Text="Total";
w_graph_base'dw_1	dw_1	DataObject="d_gh_sum_quan_series";

Declaration	
Instance Variable	string is_series, is_text

Script	Type	Name
w_g_sum_quantity	Event	ue_init
cb_ok	ControlEvent	clicked

```

Event: w_g_sum_quantity.ue_init
str_report str_x string ls_rc
str_x = Message.PowerObjectParm
dw_1.dataobject = str_x.s_dataobject
CHOOSE CASE str_x.i_style
    CASE 1 dw_1.Retrieve()
    CASE 2 dw_1.Retrieve(gstr_data.s_div_id,gstr_data.s_sect_id, gstr_data.s_year)
    CASE 3 dw_1.Retrieve(gstr_data.s_div_id,gstr_data.s_sect_id, gstr_data.s_budget_year)
    CASE 4 dw_1.Retrieve(gstr_data.s_div_id)
    CASE 5 dw_1.Retrieve(gstr_data.s_budget_year)
END CHOOSE

is_series = dw_1.Object.gr_1.series
CHOOSE CASE is_series
    CASE 'division_name!'
        is_text = 'Division Series'
    CASE 'hw_regi_status'
        is_text = 'Status Series'
END CHOOSE

ControlEvent.cb_ok.Clicked
IF This.Text = "Over All" then
    dw_1.Modify("gr_1.series = '~"noseries~" ' ") and This.Text = is_text
Else
    dw_1.Modify("gr_1.series = '"+is_series+"' ") and This.Text = "Over all"
End If
    
```

[Window] w_g_drilldown from w_graph_base
[Description] Drilldown of hardware quantity in group, class, type or specification

Controls	Control Name	Remarks
w_graph base	<w_g_drilldown>	Title="Drill down graph of hardware quantity";
w_graph base\dw_1	dw_1	DataObject="d_gh_hw_group_div_year";
datawindow	dw_2	Tab=20; DataObject="d_gh_hw_class_div_year";
datawindow	dw_3	Tab=30; DataObject="d_gh_hw_type_div_year";
datawindow	dw_4	Tab=40; DataObject="d_gh_hw_spec_div_year";

Declaration	
Instance Variable	string is_series, is_text

Script	Type	Name
w_g_drilldown	Event	ue_graph_type ue_init ue_spacing
dw_1	ControlEvent	Clicked
dw_2	Event	ue_rbuttonup pbm_rbuttonup
	ControlEvent	Clicked Rbuttondown
dw_3	Event	ue_rbuttonup pbm_rbuttonup
	ControlEvent	Clicked Rbuttondown
dw_4	Event	ue_rbuttonup pbm_rbuttonup
	ControlEvent	Rbuttondown

<i>Event: w g drilldown.ue graph type</i>
Call super:: ue_graph_type ls_style = dw_1.object.gr_1.graphtype dw_2.object.gr_1.graphtype = ls_style dw_3.object.gr_1.graphtype = ls_style dw_4.object.gr_1.graphtype = ls_style
<i>Event: w g drilldown.ue init</i>
dw_1.Retrieve(gstr_data.s_div_id,gstr_data.s_sect_id, gstr_data.s_budget_year)
<i>Event: w g drilldown.ue spacing</i>
Call super::ue_spacing ls_space = dw_1.object.gr_1.spacing dw_2.object.gr_1.spacing = ls_space dw_3.object.gr_1.spacing = ls_space dw_4.object.gr_1.spacing = ls_space
<i>ControlEvent: dw 1.clicked</i>
GrObjectType ClickedObject string ls_group, ls_grgraphname="gr_1" int li_series, li_category ClickedObject = this.ObjectAtPointer (ls_grgraphname, li_series,li_category) If ClickedObject = TypeData! or ClickedObject = TypeCategory! then ls_group = this.CategoryName (ls_grgraphname, li_category) dw_2.Modify (ls_grgraphname + ".title=" + "" hardware class in " + ls_group + "") dw_2.Retrieve(gstr_data.s_div_id,gstr_data.s_sect_id,gstr_data.s_budget_year, ls_group) Else MessageBox (Parent.Title, "Click on requested hardware group") End If
<i>** dw_2.Clicked see in dw_1.Click but change class to type and dw_2 to dw_3 dw_3.Clicked see in dw_1.Click but change class to spec and dw_2 to dw_4 Other Event and ControlEvent of dw_2, dw_3 and dw_2 see in w_graph_base. They are the same algorithms but have some different detail.</i>

[Window] w_graph_type from Window
[Description] General response window to modify a graph type

Controls	Control Name	Remarks
Window	<w_graph_type>	Type=response!; Title="Graph Type";
u_graph_gallery	uo_1	

Declaration	
Instance Variable	graph igr_parm datawindow idw_parm object io_passed

Script	Type	Name
w_graph_type	Event	open
uo_1	ControlEvent	gallery_cancel gallery_ok

```

Event: w_graph_type.open
// Receive and remember in the igr_parm or idw_parm instance variable
Graphicobject lgro_hold
Lgro_hold = message.powerobjectparm
If lgro_hold.TypeOf() = Graph! Then
    io_passed = Graph! And igr_parm = message.powerobjectparm
Elseif lgro_hold.TypeOf() = Datawindow! Then
    io_passed = Datawindow and idw_parm = message.powerobjectparm
End If

ControlEvent:uo 1.gallery cancel gallery ok
Get the graph type from the graph gallery user object.
Set the type in the passed graph object.
Close (parent)
    
```

[Window] w_graph_spacing from Window
[Description] General response window to set graph spacing

Controls	Control Name	Remarks
Window	<w_graph_spacing>	Type=response!; Title=" graph spacing";
commandbutton	cb_cancel	Tab=20; Text="Cancel";
commandbutton	cb_ok	Tab=30; Text="OK";
editmask	em_spacing	Tab=10;

Declaration	
Instance Variable	object io_passed graph igr_parm datawindow idw_parm int ii_original_spacing

Script	Type	Name
w_graph_spacing	Event	open
cb_cancel	ControlEvent	Clicked
cb_ok	ControlEvent	Clicked
em_spacing	Event	ue_exchange pbm_exchange

```

Event: w_graph_spacing.open
Graphicobject lgro_hold
lgro_hold = Message.PowerObjectParm
If lgro_hold.TypeOf() = Graph! Then
    io_passed = Graph! And igr_parm = Message.PowerObjectParm
    em_spacing.text = string(igr_parm.spacing) and ii_original_spacing = igr_parm.spacing
Elseif lgro_hold.TypeOf() = Datawindow! Then
    io_passed = Datawindow! And idw_parm = Message.PowerObjectParm
    em_spacing.text = idw_parm.Object.gr_1.spacing
    ii_original_spacing = Integer(em_spacing.text)
End If

Event: em_spacing.ue_exchange pbm_exchange
If io_passed = Graph! Then
    igr_parm.spacing = integer (em_spacing.text)
Elseif io_passed = Datawindow! Then
    idw_parm.Object.gr_1.spacing = em_spacing.text
End If
    
```

```

ControlEvent: cb_cancel.Clicked
If io_passed = Graph! Then
    igr_parm.spacing = ii_original_spacing
Elseif io_passed = Datawindow! Then
    idw_parm.Object.gr_1.spacing= string(ii_original_spacing)
End If
Close (parent)
ControlEvent: cb_ok.Clicked
Close (parent)
    
```

4. Shared windows

These windows derive from shared functions in above-mentioned section. Shared windows are w_db_error, w_login, w_pb_calendar, w_print_base, w_hw_code_search, w_hw_id_search, w_set_arg and w_about.

[Window] w_db_error from Window
 [Description] Display error of data window

Controls	Control Name	Remarks
Window	<w_db_error>	Type=response!; Title="Error";
commandbutton	cb_1	Tab=30; Text="Close";
multilineedit	mle_error_detail	Tab=10;
multilineedit	mle_fix_detail	Tab=20;
statictext	st_1	Text="Table";
statictext	st_2	Text="Row";
statictext	st_3	Text="Error detail";
Statictext	st_5	Text="Fix detail";
Statictext	st_row	
Statictext	st_table	

Script	Type	Name
w_db_error	Event	Open
cb_1	ControlEvent	Clicked

```

Event: w_db_error.open
str_db_error error_FromDW
error_FromDW = Message.PowerObjectParm
Convert error_FromDW to thai string
Display thai string
ControlEvent: cb_1.Clicked
Close(Parent)
    
```

[Window] w_login from Window
[Description] Validate access level

Controls	Control Name	Remarks
Window	<w_login>	Type=response!; Title="Login to inventory management system";
commandbutton	cb_close	Tab=40; Text="Cancel";
commandbutton	cb_ok	Tab=30; Text="OK";
singlelineedit	sle_passwd	Tab=20;
singlelineedit	sle_user_id	Tab=10;
statictext	st_1	Text="User name:";
statictext	st_2	Text="Password:";

Declaration	
Instance Variable	string is_login_state = "NO"

Script	Type	Name
w_login	Event	close
cb_close	ControlEvent	Clicked
cb_ok	ControlEvent	clicked

```

Event: w_login.Close
CloseWithReturn(This, is_login_state)

ControlEvent.cb_close.Clicked
Close(Parent)

ControlEvent.cb_ok.clicked
string ls_passwd, ls_gov_id, ls_level
Select passwd,gov_id,level into :ls_passwd, :ls_gov_id, :ls_level From security
    Where user_id = :sle_user_id.text;
if ls_passwd = sle_passwd.text then
    is_login_state = "YES"
    select div_id, sect_id into :gs_div_id, :gs_sect_id From staff where gov_id = :ls_gov_id;
    if isNumber(gs_div_id) and Len(gs_div_id) = 3 and &
        isNumber(gs_sect_id) and Len(gs_sect_id) = 3 then
        gs_level = ls_level
        parent.triggerevent("Close")
        OPEN ( g_w_mdi, g_s_mdi )
    END IF
Else    MessageBox(Invalid access)
END if
    
```

[Window] w_pb_calendar from Window
[Description] pop-up calendar window; all powerbuilder (not OLE)

Controls	Control Name	Remarks
Window	<w_pb_calendar>	Type=response!;
line	Ln 1	Disable;
u_pb_calendar	uo 1	Tab=1;

Structure	Variable type	Variable name
mousepos	long	Xpos
mousepos	long	ypos

Declaration	
External Function	FUNCTION boolean GetCursorPos(ref structure mousepos) LIBRARY "user32.dll"
Instance Variable	Private mousepos i_mousepos

Script	Type	Name
w_pb_calendar	Event	open

Event: w_pb_calendar.Open

Get the pointer position using the LOCAL External Function...

Set pop-up window in x, y position

The UO will CloseWithReturn(i_window,dateparm).

[Window] w_set_arg from Window

[Description] Setting section, hardware specification, date, year argument for query and report

Controls	Control Name	Remarks
Window	<w set_arg>	Title=" Set argument value";
commandbutton	cb_log sect	Tab=70; Text=" login section";
commandbutton	cb_startdate	Tab=60; Disable; Text="+";
commandbutton	cb_stopdate	Tab=50; Disable; Text="+";
datawindow	dw_hw	Tab=40; DataObject="d_hw code";
datawindow	dw_sect	Tab=20; DataObject="d_ff_sect select";
editmask	em_startdate	Tab=70; Disable;
editmask	em_stopdate	Tab=60; Disable
editmask	em_year	Tab=60;
Picturebutton	pb_date ok	Tab=60;
Picturebutton	pb_hw ok	Tab=50;
Picturebutton	pb_sect ok	Tab=50;
Picturebutton	pb_year ok	Tab=70;
radiobutton	rb_range	Text="Date";
radiobutton	rb_today	Text="Today";
statictext	st_date	Text="to";
tab	tab_query	Tab=10;
userobject	tabpage_date	Text="Date";
userobject	tabpage_hw	Text="Hardware specification";
userobject	tabpage_sect	Text="section";
userobject	tabpage_year	Text="year/Budget year";

Declaration	
Instance Variable	datawindowchild dwc, dwc_1,dwc_2 boolean ib_today

Script	Type	Name
w_set_arg	Event	Open ue_init ()

Script	Type	Name
cb_log_sect	ControlEvent	Clicked
cb_startdate	ControlEvent	Clicked
cb_stopdate	ControlEvent	Clicked
dw_hw	Event	ue_get_hw_code()
	ControlEvent	Itemfocuschange
ddw_sect	Event	ue_find (string as_div_id, string as_sect_id) ue_get_sect_id ()
	ControlEvent	Itemfocuschanged
em_stopdate	ControlEvent	Losefocus
pb_date_ok	ControlEvent	clicked
pb_hw_ok	ControlEvent	Clicked
pb_sect_ok	ControlEvent	Clicked
pb_year_ok	ControlEvent	Clicked
rb_range	ControlEvent	Clicked
rb_today	ControlEvent	Clicked
tab_query	Event	ue_date_enable (boolean ab_flag)

<i>Event:w set arg.open</i>
This.Post Event ue_init()
<i>Event:w set arg.ue init()</i>
<pre> ls_parm = message.stringParm if ls_parm = 'query' then This.ChangeMenu(m_queries_main) Invisible tabpage_year tab_query.tabpage_hw.dw_hw.getChild('type_id',dwc_1) tab_query.tabpage_hw.dw_hw.getChild('spec_id',dwc_2) retrieve dwc_1, dwc_2 and tab_query.tabpage_hw elseif ls_parm = 'report' then This.ChangeMenu(m_reports_main) Invisible tabpage_date and tabpage_hw tab_query.tabpage_year.em_year.text = gstr_data.s_year elseif ls_parm = 'graph' then This.ChangeMenu(m_graph_main) Invisible tabpage_date and tabpage_hw tab_query.tabpage_year.em_year.text = gstr_data.s_year end if gstr_data.s_status = '0' tab_query.tabpage_sect.dw_sect.getChild('sect_id',dwc) Retrieve dwc and tab_query.tabpage_sect.dw_sect tab_query.tabpage_sect.dw_sect.Post event ue_find(gstr_data.s_div_id,gstr_data.s_sect_id) </pre>
<i>ControlEvent: cb_log_sect.clicked</i>
<pre> gstr_data.s_div_id = gs_div_id gstr_data.s_sect_id = gs_sect_id dw_sect.Trigger Event ue_find(gs_div_id, gs_sect_id) </pre>
<i>ControlEvent: cb_startdate.clicked</i>
<pre> ls_date = Open(w_pb_calendar) if ls_date <> '00/00/0000' then em_startdate.Text = ls_date end if </pre>

<i>ControlEvent: cb_stopdate.clicked</i>
<pre>ls_date = Open(w_pb_calendar) If ls_date <> '00/00/0000' then em_Stopdate.Text = ls_date em_stopDate.TriggerEvent(loseFocus!) end if</pre>
<i>Event: dw_hw.ue_get_hw_code()</i>
<pre>ll_row = This.getRow() gstr_data.s_class_id = This.GetItemString(ll_row, "class_id") gstr_data.s_type_id = This.GetItemString(ll_row, "type_id") gstr_data.s_spec_id = This.GetItemString(ll_row, "spec_id")</pre>
<i>ControlEvent: dw_hw.Itemfocuschanged</i>
<pre>ll_row = This.getRow() ls_column = This.getColumnname() IF ls_column = 'type_id' THEN ls_class_id = This.object.class_id[ll_row] dwc_1.SetFilter("class_id = '"+ls_class_id+"'") and filter elseif ls_column = 'spec_id' THEN ls_class_id = This.object.class_id[ll_row] ls_type_id = This.object.type_id[ll_row] ls_expression = dwc_2.setFilter("class_id = '"+ls_class_id+"'and type_id = '"+ls_type_id+"'") and filter END IF</pre>
<i>Event: ddw_sect.ue_find (string as div, string as sect id)</i>
<pre>ll_row = This.Find("div_id = '"+as_div_id+"'and sect_id = '"+as_sect_id+"'",1,This.RowCount()) if ll_row > 0 then set as_div_id , as_sect_id to ddw_sect This.ScrollToRow(ll_row) end if</pre>
<i>Event: ddw_sect.ue_get_sect_id()</i>
<pre>ll_row = This.getRow() gstr_data.s_div_id = This.GetItemString(ll_row, "div_id") gstr_data.s_sect_id = This.GetItemString(ll_row, "sect_id")</pre>
<i>ControlEvent: ddw_sect.Itemfocuschanged</i>
<pre>ll_row = This.getRow() IF dwo.name = 'sect_id' THEN ls_div_id = This.object.div_id[ll_row] dwc.SetFilter("div_id = '"+ls_div_id+"'") and filter end if</pre>
<i>ControlEvent: em_stopdate. Losefocus</i>
<pre>if Date(em_stopdate.text) < date(em_startdate.text) then ls_temp = em_stopdate.text em_stopdate.text = em_startDate.text em_startDate.text = ls_temp end if</pre>
<i>ControlEvent: pb_date ok. Clicked</i>
<pre>if ib_today then gstr_data.d_startdate = gd_today gstr_data.d_stopdate = gd_today else gstr_data.d_startdate = Date(em_startdate.text) gstr_data.d_stopdate = Date(em_stopdate.text) end if</pre>

<i>ControlEvent: pb_hw.ok.Clicked</i>
dw_hw.Post Event ue_get_hw_code()
<i>ControlEvent: pb_sect.ok.Clicked</i>
dw_sect.Post Event ue_get_sect_id()
<i>ControlEvent: pb_year.ok.clicked</i>
ls_this_year = string(year(today()+543) ls_year = trim(em_year.text) if Integer(ls_year) <= Integer(ls_this_year) then gstr_data.s_budget_year = ls_year gstr_data.s_year = Right(ls_year,2) end if
<i>ControlEvent: rb_range.Clicked</i>
ib_today = false tab_query.Post event ue_date_enable(True) em_startdate.SetFocus()
<i>ControlEvent: rb_today.Clicked</i>
ib_today = true tab_query.Post event ue_date_enable(false)
<i>Event: tab_query.ue_date_enable(boolean ab_flag)</i>
tab_query.tabpage_date.em_startdate.enabled =ab_flag tab_query.tabpage_date.em_stopdate.enabled =ab_flag tab_query.tabpage_date.cb_startdate.enabled = ab_flag tab_query.tabpage_date.cb_stopdate.enabled = ab_flag

[Window] w_print_base from Window
[Description] ancestor window for print output from dw_1

Controls	Control Name	Remarks
Window	<w_print_base>	Menu="m_reports main"; Title="Preview;
commandbutton	cb_1	Tab=60; Text="<<<";
commandbutton	cb_2	Tab=50; Text=">>>";
commandbutton	cb_last	Tab=40; Text="<<<";
commandbutton	cb_next	Tab=70; Text=">>>";
datawindow	dw_1	Tab=90;
editmask	em_1	Tab=80; Text="100";
editmask	em_page	Tab=30;
groupbox	gb_1	Tab=20; Text="Zoom:";
groupbox	gb_rows	Tab=10; Text="Page:";

Declaration	
Instance Variable	int ii_pagecount

Script	Type	Name
w_print_base	Event	Open Resize ue_print() ue_save_as() ue_set_page() ue_zoom(integer ai_size)
cb_1	ControlEvent	Clicked

Script	Type	Name
cb 2	ControlEvent	Clicked
cb_last	Event	clicked pbm_bnclicked
cb_next	Event	clicked pbm_bnclicked
em 1	ControlEvent	Modified

<i>Event: w_print base.open</i>
This.Post Event ue_init()
<i>Event: w_print base.resize</i>
dw_1.width = this.width - 100 dw_1.height = this.height - 400
<i>Event: w_print base.ue print()</i>
dw_1.print()
<i>Event: w_print base.ue save as().</i>
dw_1.saveas()
<i>Event: w_print base.ue set page()</i>
em_page.text = string(1) ii_pagecount = integer(dw_1.describe("evaluate('pagecount()',1)")) em_page.minmax = "1-"+string(ii_pagecount)
<i>Event: w_print base.ue zoom(integer as size)</i>
dw_1.modify('datawindow.print.preview.zoom = ' + String(ai_size))
<i>ControlEvent:cb 1.Clicked</i>
if integer(em_1.text) > 10 THEN li_size = integer(em_1.text) -10 parent.event trigger ue_zoom(li_size) em_1.text = string(li_size) end if
<i>ControlEvent:cb 2.Clicked</i>
li_size = integer(em_1.text) +10 parent.event trigger ue_zoom(li_size) em_1.text = string(li_size)
<i>Event:cb last.Clicked pbm_bnclicked</i>
if integer(em_page.text) >1 THEN em_page.text = string(integer(em_page.text) - 1) dw_1.scrollpriorpage() end if
<i>Event:cb next.Clicked pbm_bnclicked</i>
if integer(em_page.text) < ii_pagecount THEN em_page.text = string(integer(em_page.text) + 1) dw_1.scrollnextpage() end if
<i>ControlEvent: em 1.Modified</i>
parent.event trigger ue_zoom(integer(this.text))

[Window] w_hw_code_search from Window
[Description] hardware code search by name

Controls	Control Name	Remarks
Window	<w_hw_code_search>	Type=response!; Title="hardware code search";
u_selection_list	uo_1	

Declaration	
Instance Variable	string is_name str_hw_code istr_hw_code

Script	Type	Name
w_hw_code_search	Event	Close open
uo_1	ControlEvent	ue_entry_chosen

Event:w_hw_code_search.Close

```
if is_name <> '' then
    SELECT "hw"."class_id", "hw"."type_id", "hw"."spec_id" INTO :istr_hw_code.s_class_id,
        :istr_hw_code.s_type_id, :istr_hw_code.s_spec_id FROM "hw", "hw_type" WHERE
        ( "hw_type"."class_id" = "hw"."class_id" ) and ( "hw_type"."type_id" = "hw"."type_id" )
        and ( "hw_type"."name"+"hw"."name" = :is_name);
end if
CloseWithReturn(This,istr_hw_code)
```

Event:w_hw_code_search.open

```
istr_hw_code.s_class_id = " "
ls_sql = 'select "hw_type"."name" ' + '+' + "hw"."name" FROM "hw", "hw_type" WHERE'
ls_sql += '( "hw_type"."class_id" = "hw"."class_id")and ( "hw_type"."type_id" = "hw"."type_id" ) '
uo_1.create_datawindow(sqlca, ls_sql)
```

ControlEvent: uo_1.ue_entry.chosen

```
is_name = return_selected ()
Close(Parent)
```

[Window] w_hw_id_search from Window

[Description] search hw_id from name or class, type, specification

Controls	Control Name	Remarks
Window	<w_hw_id_search>	Type=response!; Title="Hardware ID search";
commandbutton	cb_1	Tab=70; Text="OK";
commandbutton	cb_close	Tab=20; Text="Close";
commandbutton	cb_ok	Tab=30; Text="OK";
datawindow	dw_2	Tab=40; DataObject="d_hw_code";
datawindow	dw_3	Tab=40; DataObject="d_hw_id_search";
statictext	st_1	Text="Found Hardware List";
tab	tab_search	Tab=10;
userobject	tabpage_hw_id	Text="ID";
userobject	tabpage_id	Text="Code";
userobject	tabpage_name	Text="Name";
u_selection_list	uo_1	Tab=50;
u_selection_list	uo_2	Tab=50;

Declaration	
Instance Variable	string is_hw_id, is_status long il_style datawindowChild dwc_1, dwc_2

Script	Type	Name
w_hw_id_search	Event	Close Open ue_init()
cb_1	ControlEvent	Clicked
cb_close	ControlEvent	clicked
cb_ok	ControlEvent	clicked
dw_2	Event	ue_get_hw_id()
	ControlEvent	itemfocuschanged
dw_3	Event	type integer ue_retrieve (string as_class_id, string as_type_id, string as_spec_id)
	ControlEvent	Doubleclicked
uo_1	ControlEvent	ue_entry_chosen
uo_2	ControlEvent	ue_entry_chosen

<i>Event:w_hw_id_search.Close</i>
CloseWithReturn(This,is_hw_id)
<i>Event:w_hw_id_search.Open</i>
il_style = Message.DoubleParm This.Post Event ue_init()
<i>Event:w_hw_id_search.ue_init()</i>
ls_sql = 'select "hw_type"."name" + '+' + "hw"."name" FROM "hw","hw_type" WHERE' ls_sql += '("hw_type"."class_id" = "hw"."class_id") and ("hw_type"."type_id" = "hw"."type_id")' tab_search.tabpage_name.uo_1.create_datawindow(sqlca, ls_sql) ls_sql = "select hw_id from hw_regi" tab_search.tabpage_hw_id.uo_2.create_datawindow(sqlca, ls_sql)
tab_search.tabpage_id.dw_2.getChild('type_id',dwc_1) tab_search.tabpage_id.dw_2.getChild('spec_id',dwc_2) // successful = 1 retrieve dwc_1, dwc_2, dw_2, dw_3 and invisible dw_3 if il_style = 1 then // transaction process, it must use status if (gs_proc = gs_distribute) or (gs_proc = gs_borrow) then is_status = "1" // wait elseif gs_proc = gs_return then is_status = "3" //borrow elseif gs_proc = gs_repair then is_status = "4" //bad elseif gs_proc =gs_dispose then is_status = "4" //bad elseif gs_proc = gs_check_repa then is_status = "5" //repair end if end if
<i>ControlEvent: cb_1.clicked</i>
dw_2.PostEvent ("ue_get_hw_id")

<i>ControlEvent: cb_close.clicked</i>
CloseWithReturn(Parent,is_hw_id)
<i>ControlEvent: cb_ok.clicked</i>
ll_row = dw_3.getrow() if ll_row > 0 then is_hw_id = dw_3.GetItemString(ll_row,"hw_regi_hw_id") CloseWithReturn(Parent,is_hw_id) end if
<i>Event: dw_2.ue_get_hw_id()</i>
ll_row = This.getRow() ls_class_id = This.GetItemString(ll_row, "class_id") ls_type_id = This.GetItemString(ll_row, "type_id") ls_spec_id = This.GetItemString(ll_row, "spec_id") dw_3.Event ue_retrieve(ls_class_id,ls_type_id,ls_spec_id)
<i>ControlEvent: dw_2.itemfocuschanged</i>
ll_row = This.GetRow() ls_column = This.getColumnname() IF ls_column = 'type_id' THEN ls_class_id = dw_2.object.class_id[ll_row] dwc_1.SetFilter("class_id = '"+ls_class_id+"'") and filter elseif ls_column = 'spec_id' THEN ls_class_id = dw_2.object.class_id[ll_row] ls_type_id = dw_2.object.type_id[ll_row] dwc_2.setFilter("class_id = '"+ls_class_id+"'and type_id = '"+ls_type_id+"'") and Filter END IF
<i>Event: dw_3.ue_retrieve</i>
if il_style = 1 then ls_expression = "cont_detail_class_id = '"+as_class_id+"'and & cont_detail_type_id = '"+as_type_id+"' and cont_detail_spec_id = '"+as_spec_id+"' and & hw_regi_status = '"+is_status+"'" else ls_expression = "cont_detail_class_id = '"+as_class_id+"' and cont_detail_type_id = '"+as_type_id+"' and cont_detail_spec_id = '"+as_spec_id+"'" end if dw_3.SetFilter(ls_expression) and filter
<i>ControlEvent: dw_3.Doubleclicked</i>
cb_ok.TriggerEvent(Clicked!)
<i>ControlEvent: uo_1.ue_entry_chosen</i>
str_hw_code lstr_hw_code ls_name = return_selected () if ls_name <> '' then SELECT "hw"."class_id", "hw"."type_id", "hw"."spec_id" INTO :lstr_hw_code.s_class_id, :lstr_hw_code.s_type_id, :lstr_hw_code.s_spec_id FROM "hw", "hw_type" WHERE ("hw_type"."class_id" = "hw"."class_id") and ("hw_type"."type_id" = "hw"."type_id") and ("hw_type"."name"+"hw"."name" = :ls_name); dw_3.Event ue_retrieve(lstr_hw_code.s_class_id,lstr_hw_code.s_type_id,lstr_hw_code.s_spec_id) end if
<i>ControlEvent: ue_2.ue_entry_chose</i>
is_hw_id = return_selected () cb_close.TriggerEvent(Clicked!)

BIOGRAPHY



NAME Miss Usa Jugtrimongkol

DATE OF BIRTH 8 June 1975

PLACE OF BIRTH Chachoengsao, Thailand

INSTITUTIONS ATTENDED Burapha University, 1993-1997:
Bachelor of Science (Computer Science)

Mahidol University, 1997-2000:
Master of Science (Technology of
Information system management)