1985

Budget / Account system

Chun Shan Cheung

University of Wollongong, uow_cheungs@uow.edu.au

Recommended Citation

http://ro.uow.edu.au/compsciwp/64
BUDGET/ACCOUNT SYSTEM

Chun Shan Cheung

Department of Computing Science
University of Wollongong

Preprint No. 85-7 January 1985

P.O. Box 1144, WOLLONGONG N.S.W. 2500, AUSTRALIA
tel (042)-270-859
telex AA29022
A report on the specification, design and implementation of the Budget/Account system
Content

Volume I

[1] Introduction

[2] Structured Specifications

[3] How structured specification is applied to the development of the Budget/Account system

[4] A Field Oriented Editor

References

Volume II - User Manual

[1] Introduction


[4] Entering the BAS


Volume III - Structured Specifications

[1] The Budget/Account System Development Menu


[5] Structure Charts


Volume IV - Program Listings
Acknowledgement

I would like to express my deepest gratitude to my supervisor Prof. Juris Reinfelds, Chairman of the Department of Computing Science at the University of Wollongong. Throughout the project he has been giving me advice and encouragement which are of great value to the successful completion of this project. I also wish to thank Dr. Dromey for his comments and suggestions about the structured method employed in this project. I would like to thank specially Dr. N. Gray for directing me to discuss this project with Prof. Reinfelds. Finally, I wish to thank Ms. Josephine Batty for polishing my accounting principles.
"The Budget/Account System" is a project in the Computing Science Honours Seminar. This project is a partial fulfillment of the Honours Master of Science in the Department of Computing at the University of Wollongong. The accounting package here is designed and implemented for the Department of Computing Science. A new method of system specification and design is also employed in this project. It is the structured method suggested by T. Demarco, E. Yourdon and L. Constantine. Hopefully, this structured method can help to produce a more precise specification for the system than the traditional method and therefore lead to better designs and implementations which can meet users' requirements.
Introduction

Traditionally system analysis and system specifications are relying mainly on the creativity and intuition of the analyst(s) involved. The functional specifications produced are characterised by the use of huge volume of English narrative text. Demarco suggested the use of English narrative text in specifying computer systems as "inefficient" and "inappropriate". This method of system analysis and specification makes partitioning of the system difficult. The ability to partition the system at this early stage of system development is vital to the successful partitioning of the system at the later stages.

An alternative way has been developed for system specifications and design. (Demarco, Yourdon and Constantine) It is suggested that this structured method can help the analyst(s) to have an early and precise partitioning of the system. This method helps the analyst to understand the system and to communicate his/her understanding to the system user.

This structured method is applied in the specification and the design of the Budget/Account System. In this document there will not be a lengthy discussion, as we traditionally do, about the functions and design of the Budget/Account System. Rather they are specified in the form
of structured specifications in Volume III of this document.
Structured Specifications

Structured specifications is a form of system specifications which is different from the traditional one. In structured specifications, diagrams are used wherever appropriate. It is not a single specification. Rather it is a network of partitioned and small process specifications. It is not presented in a linear manner. It is presented in a hierarchical manner. It goes from the abstract level to the bottom detail level. A structured specification is made up of three components. They are the Data Flow Diagram, Data Dictionary and Process Specifications.

[2.1]

Data Flow Diagram

It is a network representation of a system. It presents the system in terms of its component processes. All the interfaces among these processes are declared. It starts with the abstract upper level and proceeds to the more detail lower level diagrams. As we go from the abstract level to the detail level the system is partitioned.

Different levels of data flow diagrams must also be balanced. It means that the interface data of a process at the upper level must equal to total data interfaces of its
sub-processes at the lower level.

[2.2]

Data Dictionary

It is a collection of the definitions of the interfaces declared on the Data Flow Diagrams. The Data Flow Diagrams declare the interfaces between processes. All these interfaces must be defined in the Data Dictionary.

[2.3]

Process Specifications

There should be one process specification for each of the processes on the bottom level Data Flow Diagrams. It describes the functions of the process. It should be restricted to describe "what" the process does but not how the "what" is to be achieved. The process specifications can be written in a number of ways. They are the Structured English, Decision Table and even narrative English.

[2.3.1]

Structured English

It is a specification language that makes use of limited vocabulary and a limited syntax. The Vocabulary of it
should consist of a) imperative English verb, b) terms defined in the Data Dictionary, c) certain reserved words for logic formulation. The syntax of its statement should be limited to the following, a) simple declarative sentence, b) closed-end decision construct, c) closed-end repetition construct.

[2.3.2]

Decision Table.

Decision tables are best for defining complicated policies formulation. As there are lots of book on decision tables, we shall not include the discussion of decision tables in this document.

[2.3.3]

Narrative English

If the processes are carefully partitioned and all the inter-processes interfaces are declared and defined then the readability of narrative specifications are greatly improved. But still it is not recommended to write narrative specifications as better ways are available.
HOW STRUCTURED SPECIFICATION IS APPLIED TO THE DEVELOPMENT OF THE BUDGET/ACCOUNT SYSTEM

The actual structured specifications are included in volume III of this document. The following paragraphs is a discussion of how they are created and used. The structured specifications of the Budget/Account system are divided into two levels. They are the functional specifications and the design specifications.

[3.1]

The Functional Specifications

The functional specifications define the functions of the Budget/Account System. It consists of a set of leveled Data Flow diagrams, a functional specification dictionary and a set of process specifications for each of the processes on the bottom level of the Data Flow Diagrams.

[3.2]

The Design Specifications

The design specifications define the algorithms to achieve the functions defined by the functional specifications. It consists on a set of leveled structure charts, a
design specification dictionary and a set of module specifications. Each module in the design specifications corresponds to an actual program module during implementation. The structure charts serve to describe the hierarchical relations between the modules. The data interfaces between modules are also declared on the charts. The data dictionary serves to define these interfaces. The module specifications are the descriptions of the actual algorithms of each of the modules. It should be noted that the module specifications should be as programming language independent as possible. As a result the whole specification up to this point is still independent of the implementation language.
A Field Oriented Editor

A field oriented editor has been specially designed and implemented for the Budget/Account System. It is designed in such a way that it is independent of the application system that it serves. This editor can be separated from the current system and used in some other application systems.

[4.1]

Aim

The Budget/Account system requires several data entry screens. One possible way is to have the screen handling algorithms embedded in the several modules which accepts input from screen. Another way is to have a process serve as a front-end processor so that the other processes can be relieved from the screen handling algorithms. The latter solution is chosen for this system. A field oriented editor is designed in place of the front-end processor. Every process which requires a full screen data input can simply call on the editor by supplying a data structure which describes the screen layout. The editor will handle all the necessary processing and return the necessary data to the caller.
The Screen Data Structure

The input to the editor is a data structure called "screen". It is a structure in the language "C". The format of "screen" is as follows:

```c
struct screen {
    int fldcnt; /* number of fields on the screen */
    struct fldstruct field[MAX_FLD]; /* the fields */
}
```

```c
struct fldstruct {
    int fldnamy;
    int fldnamx;
    char *fldname;
    int atr;
    int fldy;
    int fldx;
    int fldlng;
    int fldendptr;
    char *fld;
}
```

The "screen" consists of a count of number of fields on the current screen and an array of another structure "fldstruct". The structure "fldstruct" describes a field. The character pointer "fldname" points to the title of the field that is to be displayed on the screen. "fldnamy" and "fldnamx" are the x and y coordinates of this title. The integer "atr" describes the attribute of the data field. It tells the editor that whether the expected input to this field is numeric, or alphabetic etc. The editor will perform some checking of the input character so that invalid character can be rejected immediately. "fldx" and "fldy" are the x
and y coordinates of the actual data field. The character pointer "fd" points to the buffer in which the input data is to be stored. The above information already describe the layout of the screen and the characteristics of each field. We have not yet discussed the integer "fldendptr". It is up to discussion that whether this field worth its existence. It points to the end of the data string. It is there to save the trouble of calling the system routine "strlen" to find out the end of the data string. It may be a save of time since it is quite often that the editor needs to know the end of the data string.

[4.3]

Defining a Screen

The above data structure describes the general framework of a screen. The actual layout of a particular screen can then be defined as the following:
The definition of the Account Screen

```c
struct screen nwacscrn =
{
  8, /*no of field*/
  0,27,"[ADD NEW ACCOUNT]",HDRFLD,0,0,0,0,0,
  2,0,"main account number ":",
    MANDFLD,2,22,MAACLNG,0,&(nwacbuf[0]), /*main ac no.*/
  4,0,"sub account number ":",
    0,4,22,SUBACLNG,0,&(nwacbuf[10]), /*sub-ac no.*/
  6,0,"account category ": (a=assets,b=budget,c=commitment,e=equity)",
    ACCATFLD+MANDFLD,6,22,1,0,&(nwacbuf[30]),
  7,0,"account name ":",
    HDRFLD,0,0,0,0,0,
  8,0,"account name ":",
    MANDFLD,8,22,ACNMLNG,0,&(nwacbuf[40]),
  10,0,"normal balance ": (c=credit, d=debit)",
    ACTYFLD+MANDFLD,10,22,1,0,&(nwacbuf[80]),
  12,0,"account alias ":",
    0,12,22,80,0,&(nwacbuf[100]), /*ac alias*/
};
```

This provides a fast and convenient way to define or change the layout of a screen. The screen handling algorithms are independent of the actual screen layout. They only assume a general structure. To change the physical screen layout simply means to change the actual definition of the screen. No modifications of any algorithms are need

[4.4]

Possible Enhancement

One of the possible criticism of such a mechanism may be on the procedure to define the actual layout of the screen. The position of each parameter is important. One possible solution to alleviate this trouble is to have a
screen definition processor. The screen designer can simply make up the actual layout of a screen in a file or even on the actual screen. The screen definition processor is then invoked to process the screen layout made up by the designer and produces the corresponding data structures. This may be justified if large number of screens are to be defined.
VOLUME II

Budget/Account System
User manual
References


Introduction

The Budget/Account System (BAS) is designed for the Department of Computer Science of the University of Wollongong. It is an interactive system which enables the user to get the most up-to-date information of the accounting system. All input to the BAS are done through the terminal under the administration of a user-friendly mechanism, a field oriented editor. Under this mechanism user can formulate the input at his/her own pace and at the same time under the guidance of the system.

It has been decided that the BAS has to be able to reconcile with the accounting system of the University Administration. Therefore the BAS does possess some unique features which may not be provided by other commercially available accounting package.
The BAS is basically a general ledger posting system. The accounting principle at work in it is the simple "double-entry system". (Hermanson, p63) This system of accounting "requires that the debits must equal the credits in every entry to record a transaction". (Hermanson, p63)

In other words, a transaction is consisting of two or more journal entries. The total amount debited must balance the total amount credited by these journal entries. In order to make use of this concept of transaction, the journal entries of the BAS are not separately posted, but are posted in terms of transactions. The journal entries of a transaction will only be posted when the total debits equal total credit.

Since the Department of Computing Science is not a separate accounting body and therefore it is important that the BAS must be able to reflect the same accounting picture as that provided by the University Administration. In order to achieve this, the structure of accounts and journal posting practice are designed to be as similar to that of the University Administration as possible

[2.1]

Account Category
The usual account categories are used in the BAS, namely the assets, expenses, liability, equity, revenue (or income). Besides these, two not so common categories are introduced in the BAS as they are used by the University Administration. They are the "commitment" and "budget".

The "budget" account, though called differently, is in fact viewed as the equity account in the BAS. That is to say the annual budget is viewed as an investment and therefore the equity of the Department. Further budget adjustment can then be treated as increase or decrease of the equity.

The meaning of "commitment" account may be different from the usual accounting context. It is treated differently in order to comply with the system of the University Administration. "A commitment occurs when approval is given to place an order or enter into a contract for provision of goods and services." (University accounting document, p23) In order that we can distinguish the commitment from the actual reception of services or goods, we treat the commitment as a form of pre-paid services. For a commitment transaction, the corresponding journal entries should therefore be posted to the commitment account as an increase in pre-paid services, (i.e. debit) and the liability account as an increase in liability (i.e. credit).

Another problem occurs when the actual amount paid is different from that of the commitment. In order to save the trouble of readjusting the journal entries, the following
practice is assumed. When the actual goods or services are paid, a transaction should be posted in order to reverse the earlier posted commitment transaction. The amount of this reversal transaction should be of the same value of the commitment transaction, but not the actual amount paid. This reversal transaction should credit the corresponding commitment account (i.e. decrement the pre-paid goods or services) and debit the liability account (i.e. decrement the liability). This serves to indicate that the commitment is no longer a form of pre-paid goods or services. The BAS provides a special function to enter this reversal transaction. The user needs to specify the corresponding document and the reversal entries will automatically be posted. User can then proceed to post the transaction for the actual reception of the goods or services in the usual manner.

[2.2]

The Accounting Equation

In usual accounting practice, the balance sheet "is made of three essential elements - assets, liabilities, and owner's equity". (Hermanson, p16) The relation of this three elements is called the "accounting equation". The basic accounting equation is:

Assets + Expenses = Liabilities + Owner's Equity + Income

This equation must always be in balance.
This equation also applies in the BAS, but with some slight modifications. With the introduction of the "commitment" and "budget" account, the accounting equation in BAS is:

\[ \text{Assets} + \text{Expense} + \text{Commitment} = \text{Liabilities} + \text{Budget} + \text{Income} \]

This accounting equation is applied when preparing the balance sheet of the BAS.
Account Structures

The accounts in the BAS are assumed to consist of main accounts and sub-accounts. The purpose of a main account is to group together a number of related accounts, so that these related accounts are treated as the sub-accounts of it. However the BAS imposes no limitation on the user to design the account structures in this manner. User can still treat each account as a separate main account without any sub-account. This structure of main account and sub-account is recommended as it is similar to that of the University Administration.

A recommended account structure is the use of main account and sub-account. The classification of main account can be the classification of ledgers of the University Administration. While the sub-account can then be the cost centre of the University Administration. Keeping the account structures in line with that of the University Administration facilitates the posting procedure of the Department from the statements provided by the University Administration. However, it should be noted that, the BAS imposes no restrictions on such account structures.

Each account under the BAS is identified by a unique account number. It is a string of maximum 19 characters. The structure of the account number is as follows:
<table>
<thead>
<tr>
<th>character</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 3</td>
<td>main account number</td>
</tr>
<tr>
<td>4</td>
<td>account category</td>
</tr>
<tr>
<td></td>
<td>( a = assets; b = budget; c = commitment;</td>
</tr>
<tr>
<td></td>
<td>e = equity; i = income; l = liabilities;</td>
</tr>
<tr>
<td></td>
<td>x = expenses )</td>
</tr>
<tr>
<td>5 - 7</td>
<td>first level sub-account number</td>
</tr>
<tr>
<td>8 - 19</td>
<td>other level sub-account number</td>
</tr>
<tr>
<td></td>
<td>( in groups of three digits )</td>
</tr>
</tbody>
</table>

Assumptions on Account Structures

It has been decided with the Department that the number of accounts should be limited in the sense that we do not introduce new account when it is not absolutely necessary. This limitation reduces the chances of error posting and facilitates control.

[2.4.1]

Budget and Commitment Accounts

For each assets and expenses account, there should be a corresponding commitment account if commitment will be made against the account. If there is a budget for an assets or expenses account then a budget account should be created for it. The creation of the commitment and budget account can automatically be created by the BAS as instructed by the user.
Cash and Payable Accounts

In order to reduce the chances of error posting and limit the number of accounts, the BAS assumes that there is only one cash account and one payable account for the Department. They will automatically be created during the installation of the BAS. They have predefined account numbers. The account number of the "System Cash Account" is "900a" and that of the "System Payable Account" is "9011". User may add alias (see section 5.2).
Transaction Posting

The usual meaning of "debit" and "credit" also applies in the BAS. A summary of their meanings to different accounts as follows:

<table>
<thead>
<tr>
<th>assets account or commitment account or expenses account</th>
<th>debit</th>
<th>credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>increase</td>
<td></td>
<td>decrease</td>
</tr>
<tr>
<td>liabilities accounts or budget or equity accounts or income accounts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>decrease</td>
<td></td>
<td>increase</td>
</tr>
<tr>
<td>increase</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We try to illustrate the posting of transaction by an example. Assume that we have the following transactions:

(1) annual budget = $500,000-
(2) terminal purchase = $1,000-
(3) commitment of plotter = $2,000-
(4) actual payment for plotter = $1,500-
(5) budget adjustment (increase') = $20,000-

The transactions are posted to the appropriate account as shown by the number.

<table>
<thead>
<tr>
<th>Description</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>annual budget</td>
<td></td>
<td>$500,000</td>
</tr>
<tr>
<td>terminal purchase</td>
<td></td>
<td>$1,000</td>
</tr>
<tr>
<td>commitment of plotter</td>
<td></td>
<td>$2,000</td>
</tr>
<tr>
<td>actual payment for plotter</td>
<td></td>
<td>$1,500</td>
</tr>
<tr>
<td>budget adjustment (increase')</td>
<td></td>
<td>$20,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Account</th>
<th>Dr</th>
<th>Cr</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer equipment (assets)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(2)</td>
<td>1,000</td>
</tr>
<tr>
<td></td>
<td>(4)</td>
<td>1,500</td>
</tr>
<tr>
<td>budget of computer equipment (budget)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td>500,000</td>
</tr>
<tr>
<td></td>
<td>(5)</td>
<td>20,000</td>
</tr>
<tr>
<td>system cash account (assets)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td>500,000</td>
</tr>
<tr>
<td></td>
<td>(2)</td>
<td>1,000</td>
</tr>
<tr>
<td></td>
<td>(5)</td>
<td>20,000</td>
</tr>
<tr>
<td>system payable account (liabilities)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(4)</td>
<td>2,000</td>
</tr>
<tr>
<td></td>
<td>(3)</td>
<td>2,000</td>
</tr>
</tbody>
</table>

Commitment of computer equipment (commitment)

<table>
<thead>
<tr>
<th>Description</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>commitment of computer equipment (commitment)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(3)</td>
<td>2,000</td>
</tr>
<tr>
<td></td>
<td>(4)</td>
<td>2,000</td>
</tr>
</tbody>
</table>
Entering the BAS

The BAS can be invoked by typing "BAS" and hit RETURN.

User will then be presented with a main menu as follow:

```
University of Wollongong
Department of Computing Science
Budget/Account System
main menu

0) exit
Account Commands
1) add new account
2) add account alias
Journal Commands
3) post transaction
4) reverse commitment
Report Commands
5) balance sheet
6) account report
7) supplier report
8) summary report
9) detail report
10) keyword trace report
11) budget balance
Supplier Commands
12) add supplier
13) update supplier
14) delete supplier

please enter function:
```

To select the appropriate "Command" or function, the user can type in the correct number and hit RETURN.
Account Commands

When presented with the main menu, user can type in "1" to add new account or "2" to add account alias. The BAS will then display an account screen as follow:

<table>
<thead>
<tr>
<th>[title]</th>
</tr>
</thead>
<tbody>
<tr>
<td>main account number :</td>
</tr>
<tr>
<td>sub-account number :</td>
</tr>
<tr>
<td>account category : (a=assets, b=budget, c=commitment, e=equity, i=income, l=liability, x=expenses)</td>
</tr>
<tr>
<td>account name :</td>
</tr>
<tr>
<td>normal balance : ( c=credit, d=debit )</td>
</tr>
<tr>
<td>account alias :</td>
</tr>
</tbody>
</table>

Fig. 5.1

The "title" of the screen will either be "ADD NEW ACCOUNT" or "UPDATE ACCOUNT", depending on the command selected by the user.

Add New Account

Besides being presented with the screen in Fig. 5.1, the user will then be informed to "enter data or type ctrl-d to terminate" by a line of message at the bottom of the screen. The user can then proceed to enter information about
the new account. For the procedure entering data on the screen, please refer to section 9 of this appendix. After all the data have been entered, user can type "control d" to terminate the data entry session. The BAS will then proceed to validate the data of the new account. If any error is detected, the BAS will display the appropriate error messages. User can then correct the error and the new account will be added.

[5.2]

Add Account Alias

The BAS is designed such that the accounts can be selected by the account number, name or abbreviations or any other alias as decided by the user. The account alias is the group of words the user choose to identify the account. The account alias need not be unique. When selecting account by alias the user will be presented with all the accounts which contain the alias and the user can then select the correct one. This add account alias command is for the user to put in additional alias to the account. The same screen in Fig 5.1 will be displayed and the user can follow the data entry procedure to add in account alias.
Journal Commands

The "journal commands" are selected when the user type in "3" or "4" in the main menu.

[6.1]

Post Transaction

If the user selects "3" in the main menu then the following transaction posting menu will be displayed.

<table>
<thead>
<tr>
<th>University of Wollongong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Department of Computing Science</td>
</tr>
<tr>
<td>Budget/Account System</td>
</tr>
<tr>
<td>Transaction Posting Menu</td>
</tr>
<tr>
<td>0) exit</td>
</tr>
<tr>
<td>1) add journal entry</td>
</tr>
<tr>
<td>2) update journal entry ( current transaction )</td>
</tr>
<tr>
<td>3) list journal entry ( current transaction )</td>
</tr>
<tr>
<td>4) delete current transaction</td>
</tr>
<tr>
<td>5) file transaction</td>
</tr>
</tbody>
</table>

enter selection :

Fig. 6.1

Functions can then be selected by typing in the corresponding number and hit RETURN. All the functions on this menu are concerned with the journal entries of the current tran-
saction. These journal entries are stored in a transaction file and will only be posted to the journal file when instructed by the user.
Add Journal Entry

User will firstly be asked to select the account the journal entry is to be posted. The message "please enter account identification" will be displayed at the bottom of the screen. User can then select the account by entering an identification of the account. Please refer to section 9 of this appendix for procedure to select an account. After the account is selected a journal screen (Fig. 6.2) will be displayed with date, account number and name present. Information of this journal entry can be entered by the normal data entry procedure. After one journal entry is added, user will then be asked to select another account to which the current transaction should be posted. The posting of other journal entries of a transaction is made easy as the information of the previous entry is retained on the screen. User can simply select the account and instruct the BAS to add the entry to the transaction file if no changes is needed.

The date field of the journal entry will contain the current date. It is the date of the posting of the entry and can not be altered by the user. The keyword field is any additional words that the user wants to put it to identify the journal entry in order that it can be included in later keyword trace report.
As for the journal type, i.e. whether the entry is a debit entry or credit entry, the BAS assumes that the first entry is the credit entry, and a character 'c' will be displayed in the journal type field. However user can change it to debit if it is in fact a debit entry. The BAS will then assumes that the next entry will be the corresponding offset entry by filling in the journal type field for the user.

To quit from this journal posting session, user can simply hit RETURN without typing in any other key when asked to select an account. Then the user will go back to the transaction posting menu.

---

[POST JOURNAL]

date : 841116
account number : 999a
account name : departmental cash account
amount : [ ]

cr/dr : ( c=credit,d=debit )
document number :
voucher number :
requisition number :
purchase order number :
supplier code :
description :
keywords :

Fig. 6.2
[6.1.2]

Update Journal Entry

This function enable the user to change the information on the journal entries currently stored in the transaction file. Each journal entry will be displayed and the user will be asked whether he/she wants to update the entry. User can choose to update the entry, go to next entry if there is any, or quit to the transaction posting menu. These selection can be made by typing in 'y', 'n' or 'q' when a journal entry is displayed and the message "update this entry <y/n/q>" is displayed at the bottom of the screen.

[6.1.3]

List Journal Entry

User can simply inspect all the journal entries in the transaction file by selecting this function. Each journal entry will then be displayed. The message "hit RETURN to continue or 'q' to quit" will be displayed at the bottom of the screen. User can go on to the next journal entry by simply hitting RETURN or go back to the menu by type in 'q' and hit RETURN. This function will not alter any information of the journal entries.
[6.1.4]  
Delete Current Transaction

This function should be used with case as it will erase all the journal entries in the transaction file. After this function is selected, the message "delete all entries of current transaction <y/n>" will be displayed. If the user responds by typing in 'y' and hit RETURN then all journal entries in the transaction file will be erased and user can proceed to start a new transaction. If 'n' is responded by the user, then the BAS will perform no action and return to the transaction posting menu.

[6.1.5]  
File Current Transaction

After all journal entries related to the current transaction have been posted, user can select this function to post the current transaction to the journal file. Before the actual posting to the journal file, the BAS will check if the total amount debited balances the total amount credited by this transaction. If they are not equal then an error message will be displayed. The user will be informed of which amount is greater than the other amount. The transaction will be unchanged and the user can proceed to correct the journal entries by the update function or even choose to delete the transaction. If the balances are correct then the
transaction will be posted to the journal file and the transaction file will be cleaned. User will then be presented with the transaction posting menu so that he/she can go on to post other transactions.

[6.1.6]

Exit from Transaction Posting Menu

This function enables the user to return to the main menu of the BAS. Before return to the main menu the BAS will check if the transaction file still contains any unposted transaction. If transaction still exits in the transaction file, the BAS will ask the user whether the transaction should be posted or simply discarded before exit. If the transaction is to be posted, the normal procedure and checking will be performed by BAS for posting the transaction. After the transaction is posted, the BAS will return to the main menu. User will be informed of any error in the transaction as if the function file current transaction is selected. If the user instructs the BAS not to file the transaction before exit, then the transaction file will be erased before the BAS return to the main menu.
Reverse Commitment

This function enables the user to reverse earlier commitment. User will be asked to enter the requisition number of the commitment made. If commitment entry is found, the journal entry will be displayed. The BAS will then ask the user to confirm the reversal of the commitment. The BAS will also check if the reversal has already been made so the no double reversal can be made. The reversal entries will then be automatically created and posted to offset the corresponding commitment account and the department payable account.
Report Commands

These commands enable the user to select the various reports provided by the BAS. Basically the user will be presented with the report request screen as in Fig. 7.1 so that he/she can specify the parameters for generating reports. Before the user can go ahead to enter these parameters to the request screen, he/she may be asked to select the appropriate account if the reports concern a specific account. This is the normal account selection procedure.

[REPORT REQUEST]

| report type : |
| start date : | ( yymmdd or blank for start of accounting period ) |
| end date : | ( yymmdd or blank for current date ) |
| account number : | ( blank for all accounts ) |
| account name : |
| supplier id : | ( blank for all suppliers ) |
| keyword : |
| destination : | ( p=printer ; t=terminal ) |

Fig. 7.1

The report type field will contain the name of report as selected by the user in the main menu. The start and end date, if specified, should be in the format of year-month-day with two digit in each sub-field. The account number
and name fields will be that of the account selected by the user. The BAS will assist the user to select the account in the normal account selection procedure. The supplier id field should contain the appropriate supplier code or blank for all suppliers. The user can specify whether the report is to be displayed to the terminal or sent to the printer by putting in 't' or 'p' in the destination field. The default destination is the terminal. It is obvious that not all the fields on the screen in Fig. 7.1 is relevant to a report, e.g. the keyword field is irrelevant to the balance sheet. The information needed for each type of report will be discussed in the following section. However the user need not remember them or even need not refer to this manual when generating reports. It is because all irrelevant field will be protected by the BAS in such a way that the user can not enter data onto these fields.

[7.1]

Balance Sheet

Start date and end date can specified for the balance sheet so the the balance sheet reflects the accounting situation of the specified period. Please refer to Fig. 7.2 for format of the balance sheet.

[7.2]

Account Report
For this report only the account number and name are needed. If all accounts are to be reported then these fields can be simply left blank. If a certain account is selected, the BAS will automatically include all the sub-accounts and the corresponding budget and commitment accounts. This report gives the static information of an account. Please refer to Fig. 7.3 for the format of the account report.

[7.3]

Supplier Report

Obviously only the supplier id is needed for this report. In order to generate report of all suppliers the supplier id field can simply be left blank. Please refer to Fig. 7.4 for the format of this report.

[7.4]

Summary Report

This report gives the total amount debited and credited to the account. It also gives the net balance of the account. If the account selected contains sub-account then the balances of its sub-account will also be included. In this case the balances of this account will include that of its sub-account. If corresponding budget or commitment accounts are present they will also be included. The start and end date
can also specified in order to restrict the balances to a certain period of time. Please refer to Fig. 7.5 for the format of this report.

[7.5]

Detail Report

This report gives a list of the journal entries that has been posted to an account. Again the corresponding budget and commitment account and all its sub-account will also be included in the report. Start date and end date can also be specified. Please refer to Fig. 7.6 for the format of this report.

[7.6]

Keyword Trace Report

This report gives a list of journal entries that contains a certain keyword specified by the user. Start date and end date can be specified to restrict the search for journal entries. Please refer to Fig. 7.7 for the format of this report.

[7.7]

Budget Balance
This report provides the information of the amount budgeted, the amount already spent and the net balance of a certain account. Please refer to Fig. 7.8 for the format of this report.
## Fig. 7.2 Balance Sheet

<table>
<thead>
<tr>
<th>Category</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assets</td>
<td>500000.00</td>
</tr>
<tr>
<td>Commitment</td>
<td>2000.00</td>
</tr>
<tr>
<td>Expenses</td>
<td>0.00</td>
</tr>
<tr>
<td><strong>Total Assets</strong></td>
<td><strong>502000.00</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equity</td>
<td>500000.00</td>
</tr>
<tr>
<td>Liability</td>
<td>2000.00</td>
</tr>
<tr>
<td>Income</td>
<td>0.00</td>
</tr>
<tr>
<td><strong>Total Equity</strong></td>
<td><strong>502000.00</strong></td>
</tr>
</tbody>
</table>
Main account number: 200
Sub-account number: 201
Account category: a
Account name: equipment computer
Normal balance: c
Account alias: computer
Fig. 7.4 Supplier Report

Supplier code : apple
Supplier name : apple computers
Description : selling apples
Address : sydney
Phone : 6053853
### Summary Report

<table>
<thead>
<tr>
<th>Account Number</th>
<th>Account Name</th>
<th>Debit</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>200b201</td>
<td>Budget Account</td>
<td>0.00</td>
<td>500,000.00</td>
</tr>
<tr>
<td>200c201</td>
<td>Commitment Account</td>
<td>2000.00</td>
<td>0.00</td>
</tr>
<tr>
<td>200a201</td>
<td>Equipment Computer</td>
<td>1100.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
## Fig. 7.6 Detail Report

**THE UNIVERSITY OF WOLLONGONG**  
**DEPARTMENT OF COMPUTING SCIENCE**  
**ACCOUNT DETAIL REPORT**  
**8400101 - 841220**  
**ACCOUNT DETAIL REPORT**  
**841125**  
**page 1**

<table>
<thead>
<tr>
<th>Account Name</th>
<th>Account Number</th>
<th>Debit</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>equipment computer</td>
<td>200a201</td>
<td></td>
<td></td>
</tr>
<tr>
<td>purchase terminal</td>
<td>841124</td>
<td>1000.00</td>
<td></td>
</tr>
<tr>
<td>printer paper</td>
<td>841125</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>vou nr : 9876</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

| Total                  | 1100.00 | 0.00  |
| Balance                | 1100.00 | 0.00  |
Fig. 7.7 Keyword Trace Report

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Debit</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>plotter</td>
<td>2000.00</td>
<td>2000.00</td>
</tr>
<tr>
<td>date</td>
<td>841124</td>
<td></td>
</tr>
<tr>
<td>account</td>
<td>200c201</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>req nr</td>
<td>12345</td>
<td></td>
</tr>
</tbody>
</table>

| date       | 841124 |        |
| account    | 9981   |        |
| name       |        |        |
| description|        |        |
| req nr     | 12345  |        |

Total: 2000.00 2000.00
The University of Wollongong
Department of Computing Science
Budget Balance

<table>
<thead>
<tr>
<th>Account Number</th>
<th>Account Name</th>
<th>Debit</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>200b201</td>
<td>Budget Account</td>
<td>3100.00</td>
<td>500000.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>balance:</td>
<td>496900.00</td>
</tr>
</tbody>
</table>
Supplier Commands

These commands enable the user to manipulate records in the supplier file. User can choose to add new supplier, delete supplier and update supplier record. Each supplier in the supplier file is identified by a unique supplier code of maximum 5 characters. The supplier code is assigned by the user during the addition of a new supplier. Other information of the supplier stored in the supplier is the actual name, address, description and phone number.

Add New Supplier

This function enable the user to add the record of a new supplier to the BAS. The User will be presented with a screen as in Fig 8.1. Data can then be entered following the normal data entry procedure. After all the data are entered, the BAS will check to see if the supplier code assigned by the user is already present in the supplier file. If it is found, the BAS will inform the user to change it.
Delete Supplier

User will firstly be asked to specify the supplier code of the supplier that he/she wants to delete. If the supplier code is found in the supplier file then the information of the supplier will be displayed to the user. The BAS will then ask user to confirm the deletion of the supplier. If the BAS is instructed to delete the supplier, then the supplier record will be removed from the supplier. Otherwise the BAS will do nothing to the supplier file.
Update Supplier

This function enables the user to change the information about a supplier that is already in the supplier file. User will be asked to specify the supplier code. The information of the supplier will then be displayed as the screen in Fig. 8.1. User can then proceed to change the information that is displayed on the screen. It should be noted that the supplier code cannot be modified by this update function. If the supplier code has to be changed, the way to do it is to delete the supplier and add in a new record of the supplier using desired supplier code.
Account Selection Procedure

The BAS provides a convenient way for user to select an account wherever an active references an account. Through this there is no need to remember or to find out the actual account number or name in order to tell the BAS which account is to be referenced.

Whenever an account is referenced, for example posting of a transaction or generating of a report, the BAS will firstly ask for the identification of an account. This identification can be the account number, name or any other identification that the user had assigned to identify an account. Recall that there is a field called the "Account alias" when an account is created. It is where the user can put in additional information to identify the account. When selecting the account the identification supplied by the user need not be the whole word that had been assigned. Rather, it can be part of the identification. All the account that contain similar identification will be collected by the BAS and be displayed to the user one at a time. The BAS will then ask the user to select the correct account. The information of an account will be displayed as the screen shown in Fig. 5.1. The message "is this the correct account <y/n/q>" will also be displayed at the bottom of the screen. If the account shown is the correct one, user can simply type in "y" and hit RETURN. Otherwise, user can type in "n" and hit RETURN, then the BAS will display
the information of the next account that matched the identifica-
tion. If the user do not want to select an account with
the identification that he/she has just specified, he/she
can type in "q" and hit RETURN. The BAS will then ask the
user to supply a new piece of identification to select an
account. In order to terminate this session with the BAS
to select an account, user can simply hit RETURN when asked
to enter an identification. In this case the BAS will return
to the previous menu or screen.
Data Entry Procedure

The BAS is designed such that the procedure of entering data is the same throughout the system. In general there are two situations in which user is required to enter data to the BAS. The first case is to supply instructions to the BAS, such as selecting functions, selecting accounts. The second case is to enter the actual data to the BAS, such as entering a new journal entry, or a new account.

Instructing the BAS

The BAS will ask user to supply instructional data. This is done through a communication message line at the bottom part of the screen. The maximum number of characters to be entered will be shown by a pair of square brackets as follows:

please enter account identification [ ]

User can then proceed to type in the characters. After all necessary data are typed, user can simply hit RETURN to inform the BAS that the data are ready.
Actual Data Input

In this case user is usually presented with a full screen. The screen is consisting of several fields. The cursor will initially be positioned at the first field. A pair of square brackets will be displayed around the area in which the data for the field is to be entered. This pair of brackets also serve to indicate the maximum length of the field. The input of data to these fields are not necessary in the physical order of the fields. User can go freely from one field to the next or to the previous. Hitting the TAB or the RETURN key will position the cursor to the start of the next field, while hitting the 'upward arrow' will position the cursor to the start of the previous field. The 'left arrow' and the 'right arrow' keys will move the cursor one position to the left or to the right respectively with a field. When the end of the field is reached, the cursor will automatically be moved to the start of next field. When the cursor is moved past the start of a field by the 'left arrow' it will be moved to the start of the previous field.

When all the data are entered, the user can inform the BAS by holding the 'control' key (ctrl) down and hit letter 'd' (ctrl-d). The BAS will then ask the user whether he/she thinks that the data is correctly entered or not with the following message:
"is the data correct <y/n/other key to continue edit> [ ]"

Both letters 'y' and 'n' signify the end of this data entry session with the BAS. The letter 'y' indicates to the BAS that the data is correct and it can proceed to process it. The letter 'n' means that the user does not want the BAS to proceed to process the data but rather to ignore the data entered on the screen and return to the previous screen.

User may discover that some more changes is needed after the key ctrl-d is pressed, then he/she can respond with any character, other than 'y' or 'n', in order to continue the data entry session.
VOLUME III

Budget/Account System
Structured Specifications
Errata

The following pages have been deleted from this volume:

p.31 - p.33

p.87 - p.88

p.92

p.100
The Budget/Account System Development Menu

One of the purposes of the use of structured specifications is to produce a form of system specification that is maintainable. System documentation which is difficult to maintain is destined to be a failure. Such documentation is useless once there is any change or enhancement to the system. Such documentation can no longer reflect the current state of the system. Therefore in order to have a maintainable system it is also important that the related documentation can easily be maintained. A development menu has been developed for maintaining the documentation of the Budget/Account System. The following sections will be a discussion of this menu and some guidelines for its user.

1.1

Design of the Development Menu

The programs in this menu are all shell scripts. They are written in the early stages of the development of the Budget/Account System. It is decided not to spend too much time on the development of this menu as it serves only to provide a convenient method to access the documentation. But obviously it is not the only way to gain access to the documentation. Therefore it has to be stressed that these
scripts are by no means good scripts. They are written to serve the basic functions. If this menu is to be used intensively or used for some other applications, then further modifications should be made. For example, the field oriented editor, which is developed in the later stage of the Budget/Account System, may be used to implement the updating of data item in the dictionary. Other modifications may be necessary to improve the response time of this menu.
Some User guidelines

In order to gain access to the documentation of the Budget/Account, user can use the command
/cs/900/9csc/411/aids/menu after the normal sign-on procedure on system B. A menu is then presented as follows:

+--------------------------------------------------------------------+
| 0 - EXIT                                                           |
| 1 - VIEW DICTIONARY                                                |
| 2 - EDIT DICTIONARY                                                |
| 3 - SORT DICTIONARY                                                |
| 4 - PRINT DICTIONARY                                               |
| 5 - EDIT DESIGN SPECIFICATION                                      |
| 6 - FORMAT DESIGN SPECIFICATION                                    |
| 7 - EDIT FUNCTIONAL SPECIFICATION                                  |
| 8 - FORMAT FUNCTIONAL SPECIFICATION                                |
| 9 - PRINT DESIGN SPECIFICATION                                    |
| 10 - PRINT FUNCTIONAL SPECIFICATION                                |
+--------------------------------------------------------------------+
please enter function:

Items on the menu can then be selected by typing the corresponding number and then press the key "return". When working within any selected functions, one can always return to the main menu by typing in nothing but just the return key. Functions of each item are as follows:

(0) EXIT - exit from the menu.
(1) **VIEW DICTIONARY** - read data word from a certain data dictionary. After item "1" is selected, user is then asked to enter the name of the data dictionary. A default data dictionary name is provided and if the default name is the one desired, just press the key "return". After selecting the dictionary "word=" will be displayed to solicit the desired data name. If the data name is found in the dictionary its comment and definition will be displayed, otherwise a not found message will be displayed. Press "return" key to return to main menu.

(2) **EDIT DICTIONARY** - add, delete or update data word in dictionary. Dictionary selection is same in "1". The screen based editor is used to implement this updating process. After data name is entered the screen based editor is invoked for the user to edit a file with the following content:

WORD = data name entered by user

COMMENT = comment of the data if the data already exists otherwi

DEFINITION = definition of the data if it already exists otherwi

Comment and definition of the data can then be entered in the usual way of an editor except that care should be taken as not to change the heading "WORD =", "COMMENT =" and "DEFINITION =". After all information are
entered type ctrl "d" and answer "y" when prompted to exit the editor. User will then be prompted for the following option: fi, fd, om. Type "fi" to file the changes and "om" to omit the changes. "fd" is for deleting the data name from the dictionary.

The following conventions are adopted in the definition of the items in the data dictionary.

a) \( a = n\{ \ b \} m \)

The pair of braces stands for the iterations of the item \( \ b \). If the number \( \ n \) and \( \ m \) are present then it means that there are at least \( \ n \) occurrences of \( \ b \) and at most \( \ m \) occurrences of \( \ b \).

b) \( d = [ a / b / c / \ldots ] \)

This pair of square bracket serves to include the possible alternative of the content of an item \( d \). That is to say the possible content may be \( a \) or \( b \) or \( c \) or others.

c) \( a = b + c \)

This means that the item \( a \) is made up of two other items \( b \) and \( c \).

d) \( a = b + ( c ) \)

It indicates that the item \( c \) is optional. It may or may not be present in \( a \).
(3) SORT DICTIONARY - sort data names in dictionary according to alphabetical order. Dictionary selection as above.

(4) PRINT DICTIONARY - print the whole dictionary to printer. Dictionary selection as above.

(5) EDIT DESIGN SPECIFICATION - update or add module description for design specification. The screen based editor is used to enter the description. The constructs used in the modules should be limited to the following:

if condition
then statements
else statements
fi

or

repeat until condition
statements
endrepeat

or

for ...
statements
endfor
select case ...
case 1: ...
    statements
    .
    .
case n: ...
    statements
endcase

or

while condition
statements
endwhile

or

begin
statements
end

It is not necessary to indent in order to show the above construct when enter the module description because function is available to format the module (see FORMAT below).
(6) FORMAT DESIGN SPECIFICATION - format the module so that the control structures are clearly shown with indentation. Data names used in the module will also be looked up in the design specification dictionary. If it is present in the dictionary it will be converted to upper case otherwise remain lower case. Thus after formatting a module, if a data name becomes upper case then it is defined in the dictionary.

(7) EDIT FUNCTIONAL SPECIFICATION - same as [5] except that it works on functional specifications.

(8) FORMAT FUNCTIONAL SPECIFICATION - same as [6] except that it works on functional specifications and references the functional specification dictionary.

(9) PRINT DESIGN SPECIFICATION - print a design specification module to printer.

(10) PRINT FUNCTIONAL SPECIFICATION - print a functional specification module to printer.
Data Flow Diagrams

This section is a collection of the data flow diagrams of the Budget/Account System. It starts with Diagram 0 which is the overview of the system. Each process of Diagram 0 is further divided into sub-processes. These sub-processes are depicted on separate data flow diagrams labeled as Diagram 2, Diagram 3 etc. The convention used in labeling the data flow diagrams is that Diagram n describes the sub-processes of process n.

As mentioned above that each process is partitioned into sub-processes. These sub-processes are then further partitioned into sub-processes. The criteria of the partitioning is that the bottom level sub-processes are small enough such that the process specifications are reasonably small, e.g. can be described in a single page.
Diagram 2

1. ADD SUPPLIER-SCREEN
   - ADD SUPPLIER-SEL
   - NEW SUPPLIER INFO

2. ADD NEW SUPPLIER
   - SUPPLIER RECORD

3. DELETE SUPPLIER
   - SUPPLIER RECORD

4. SUPPLIER FILE
   - SUPPLIER INFO

5. UPDATE SUPPLIER
   - UPDATE SUPPLIER-INFO
   - UPDATE SUPPLIER-SEL
   - SUPPLIER RECORD
Diagram 5.1
The Functional Specifications

In this section the functions of each of the bottom level processes of the data flow diagrams are specified. There is one module for each of these processes.
MODULE NAME : 1
DESCRIPTION: accept user selection

Procedure:
display MAIN-MENU
get MAIN-MENU-SELECTION
if valid MAIN-MENU-SELECTION
  then pass on to corresponding process
  else reject selection
fi
MODULE NAME : 2.1
DESCRIPTION : add new supplier

Procedure :
display ADD-SUPPLIER-SCREEN
get NEW-SUPPLIER-INFO
if SUPPLIER-CODE already exists on SUPPLIER-FILE
   then reject NEW-SUPPLIER-INFO
   else make up SUPPLIER-RECORD from NEW-SUPPLIER-INFO
       write SUPPLIER-RECORD to SUPPLIER-FILE
MODULE NAME : 2.2
DESCRIPTION : delete supplier

Procedure :
get DELETE-SUPPLIER-INFO
if SUPPLIER-CODE does not exist on SUPPLIER-FILE
   then reject DELETE-SUPPLIER-INFO
else get corresponding SUPPLIER-RECORD from SUPPLIER-FILE
   display DELETE-SUPPLIER-SCREEN
   ask user to confirm the deletion
   if positive confirmation
      then delete the SUPPLIER-RECORD from SUPPLIER-FILE
fi
fi
MODULE NAME: 2.3
DESCRIPTION: update supplier

Procedure:
get SUPPLIER-CODE from user
if SUPPLIER-CODE does not exist on SUPPLIER-FILE
  then reject SUPPLIER-CODE
else
  get corresponding SUPPLIER-RECORD from SUPPLIER-FILE
  display UPDATE-SUPPLIER-SCREEN
  get UPDATE-SUPPLIER-INFO
  make up SUPPLIER-RECORD from UPDATE-SUPPLIER-INFO
  rewrite SUPPLIER-RECORD to SUPPLIER-FILE
fi
MODULE NAME: 3.1
DESCRIPTION: generate balance sheet

Procedure:
display REPORT-SCREEN
get BAL-SHEET-INFO from user
for all JOURNAL-RECORD which within the start date and end-date
select case according to AC-CATEGORY of ACCOUNT-NUMBER of JOURNAL-RECORD
  case 1: assets-ac
    if JOURNAL-TYPE = debit
      then add JOURNAL-AMOUNT to ASSETS-TOTAL
    else subtract JOURNAL-AMOUNT from assts-total
  fi
  case 2: budget-ac
    if JOURNAL-TYPE = debit
      then subtract JOURNAL-AMOUNT from EQUITY-TOTAL
    else add JOURNAL-AMOUNT to EQUITY-TOTAL
  fi
  case 3: commitment-ac
    if JOURNAL-TYPE = debit
      then add JOURNAL-AMOUNT to COMMITMENT-TOTAL
    else subtract JOURNAL-AMOUNT from COMMITMENT-TOTAL
  fi
  case 4: expense-ac
    if JOURNAL-TYPE = debit
      then add JOURNAL-AMOUNT to EXPENSE-TOTAL
    else subtract JOURNAL-AMOUNT from EXPENSE-TOTAL
  fi
  case 5: liability-ac
    if JOURNAL-TYPE = debit
      then subtract JOURNAL-AMOUNT from LIABILITY-TOTAL
    else add JOURNAL-AMOUNT to LIABILITY-TOTAL
  fi
  case 6: income-ac
    if JOURNAL-TYPE = debit
      then subtract JOURNAL-AMOUNT from INCOME-TOTAL
    else add JOURNAL-AMOUNT to INCOME-TOTAL
  fi
endcase
BAL-SHEET-DR-TOTAL = ASSETS-TOTAL + COMMITMENT-TOTAL + EXPENSE-TOTAL
BAL-SHEET-CR-TOTAL = LIABILITY-TOTAL + EQUITY-TOTAL + INCOME-TOTAL
display BALANCE-SHEET
MODULE NAME : 3.2
DESCRIPTION : generating account report

Procedure :
1. get AC-ID from user
2. if AC-ID = all
    then goto step 5
   fi
3. get all ACCOUNT-RECORD which match the AC-ID
4. for each of the ACCOUNT-RECORD got
   4.1 make up DISPLAY-AC-SCREEN
   4.2 display the DISPLAY-AC-SCREEN
   4.3 ask user to confirm the correct account
   4.4 if positive confirmation
       then get all sub-account, budget-ac and commitment-ac of this account
       goto step 6
   fi
endfor
5. get all ACCOUNT-RECORD from ACCOUNT-FILE
6. make up ACCOUNT-REPORT from the ACCOUNT-RECORD got
MODULE NAME : 3.3
DESCRIPTION : generating supplier report

Procedure :
display REPORT-SCREEN
get SUP-RPT-INFO
get corresponding SUPPLIER-RECORD as indicated by SUPPLIER-CODE
    from SUPPLIER-FILE
make up SUPPLIER-REPORT from the SUPPLIER-RECORD got
MODULE NAME : 3.4
DESCRIPTION : generating summary report

Procedure :
1. get AC-ID from user
2. if AC-ID = all
   then goto step 5
fi
3. get all ACCOUNT-RECORD which match the AC-ID
4. for each of the ACCOUNT-RECORD got
   4.1 make up DISPLAY-AC-SCREEN
   4.2 display the DISPLAY-AC-SCREEN
   4.3 ask user to confirm the correct account
   4.4 if positive confirmation
      then get all sub-account, budget-ac and commitment-ac of this account
      goto step 6
   fi
endfor
5. get all ACCOUNT-RECORD from ACCOUNT-FILE which is within the
   start and end-date
6. make up SUMMARY-REPORT as following steps
7. for each of the ACCOUNT-RECORD got
   7.1 get all JOURNAL-RECORD of this account and its sub-account
   7.2 for each JOURNAL-RECORD got
      7.2.1 if JOURNAL-TYPE = debit
         then add JOURNAL-AMOUNT to debit-total
         else subtract JOURNAL-AMOUNT to credit-total
      fi
   endfor
   7.3 if AC-CATEGORY = assets-ac or commitment-ac or expense-ac
      then total-balance = debit-total - credit-total
      else total-balance = credit-total - debit-total
   fi
MODULE NAME : 3.5
DESCRIPTION : generating detail report

Procedure :
1. get AC-ID from user
2. if AC-ID = nil
   then goto step 5
fi
3. get all ACCOUNT-RECORD which match the AC-ID
4. for each of the ACCOUNT-RECORD got
   4.1 make up DISPLAY-AC-SCREEN
   4.2 display the DISPLAY-AC-SCREEN
   4.3 ask user to confirm the correct account
   4.4 if positive confirmation
       then get all sub-account, budget-ac and commitment-ac of this account
       goto step 6
   endfor
5. get all ACCOUNT-RECORD from ACCOUNT-FILE, which is within the start and end-date
6. make up DETAIL-REPORT as following steps
7. for each of the ACCOUNT-RECORD got
   7.1 get all JOURNAL-RECORD of this account and its sub-account
   7.2 for each JOURNAL-RECORD got
       7.2.1 put the JOURNAL-RECORD to the DETAIL-REPORT
       7.2.2 if JOURNAL-TYPE = debit
           then add JOURNAL-AMOUNT to debit-total
           else subtract journal-amount to credit-total
       endfor
   7.3 if AC-CATEGORY = assets-ac or commitment-ac or expense-ac
       then total-balance = debit-total - credit-total
       else total-balance = credit-total - debit-total
   endfor
MODULE NAME : 3.6
DESCRIPTION : generating keyword trace report.

Procedure :
display REPORT-SCREEN
get KEYW-TRACE-INFO
get all JOURNAL-RECORD which contain the keyword and
are within the start date and end-date
for each of the JOURNAL-RECORD got
    put JOURNAL-RECORD to KEYWORD-TRACE-REPORT
    if JOURNAL-TYPE = debit
        then add JOURNAL-AMOUNT to total-debit
    else add JOURNAL-AMOUNT to total-credit
    fi
endfor
MODULE NAME: 3.7
DESCRIPTION: generating budget balance

Procedure:
1. get AC-ID from user
2. if AC-ID = all
   then goto step 5
   fi
3. get all ACCOUNT-RECORD which match the AC-ID
4. for each of the ACCOUNT-RECORD got
   4.1 make up DISPLAY-AC-SCREEN
   4.2 display the DISPLAY-AC-SCREEN
   4.3 ask user to confirm the correct account
   4.4 if positive confirmation
      then get all sub-account, budget-ac and commitment-ac of this account
      if budget-account not found
         then reject this account
      else goto step 6
   fi
endfor
5. get all ACCOUNT-RECORD from ACCOUNT-FILE which is within the start and end-date
6. make up BUDGET-BALANCE-REPORT as following steps
7. for each of the ACCOUNT-RECORD got
   7.1 get all JOURNAL-RECORD of this account and its sub-account
   7.2 for each JOURNAL-RECORD got calculate the budget balance according to the following decision table

<table>
<thead>
<tr>
<th></th>
<th>assets-ac</th>
<th>budget-ac</th>
<th>commitment-ac</th>
<th>expense-ac</th>
<th>income-ac</th>
<th>credit</th>
<th>debit</th>
<th>add to balance</th>
<th>subtract from balance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

endif
endif


MODULE NAME : 4.1
DESCRIPTION : add new account

Procedure:
display NEW-AC-SCREEN
get NEW-ACCOUNT-INFO
if it is a sub-account
    then check whether its main-account and upper level account exist
        if not found
            then reject NEW-ACCOUNT-INFO
        fi
fi
check whether ACCOUNT-NUMBER already exist on ACCOUNT-FILE
if found
    then reject NEW-ACCOUNT-INFO
else add NEW-ACCOUNT-INFO to ACCOUNT-FILE
fi
MODULE NAME: 4.2
DESCRIPTION: add account alias

Procedure:
1. get AC-ID from user
2. get all ACCOUNT-RECORD which match AC-ID from ACCOUNT-FILE
3. for each of the ACCOUNT-RECORD got
   3.1 make up DISPLAY-AC-SCREEN
   3.2 display the DISPLAY-AC-SCREEN
   3.3 ask user to confirm the correct account
   3.4 if positive confirmation
       then goto
   fi
endfor
4. if ACCOUNT-RECORD got
   then make up UPDATE-AC-SCREEN
   display UPDATE-AC-SCREEN
   get UPDATE-ACCOUNT-INFO
   rewrite ACCOUNT-RECORD to ACCOUNT-FILE
fi
MODULE NAME : 5.1.1
DESCRIPTION : accept transaction menu selection

Procedure:
display TRANSACTION-MENU
get TRANS-MENU-SEL
if invalid selection
    then reject TRANS-MENU-SEL
    else pass on TRANS-MENU-SEL to corresponding process
fi
MODULE NAME: 5.1.2
DESCRIPTION: add new journal entry

Procedure:
1. get AC-ID from user
2. get all ACCOUNT-RECORD which match the AC-ID from ACCOUNT-FILE
3. for each of the ACCOUNT-RECORD got
   3.1 make up DISPLAY-AC-SCREEN
   3.2 display the DISPLAY-AC-SCREEN
   3.3 ask user to confirm the correct account
   3.4 if positive confirmation
       then goto 4
endfor
4. if ACCOUNT-RECORD got
   then put ACCOUNT-NUMBER and account-name to ENTER-JOURNAL-SCREEN;
       put current date to ENTER-JOURNAL-SCREEN
       display ENTER-JOURNAL-SCREEN;
       get NEW-JOURNAL-INFO;
       put JOURNAL-RECORD to TRANSACTION-FILE;
endfor
MODULE NAME : 5.1.3
DESCRIPTION : update journal entry

Procedure :
for each JOURNAL-RECORD in TRANSACTION-FILE
   make up update-journal-screen
   ask user to confirm update
   if positive confirmation
      then get NEW-JOURNAL-INFO from user
         rewrite the JOURNAL-RECORD to TRANSACTION-FILE
   fi
endfor
MODULE NAME : 5.1.4
DESCRIPTION : delete transaction

Procedure :
discard all JOURNAL-RECORD on TRANSACTION-FILE
MODULE NAME : 5.1.5
DESCRIPTION : list journal entry

Procedure :
for each JOURNAL-RECORD in TRANSACTION-FILE
    make up DISPLAY-JOURNAL-SCREEN from the JOURNAL-RECORD
    display the DISPLAY-JOURNAL-SCREEN
endfor
MODULE NAME : 5.1.6
DESCRIPTION : file transaction

Procedure :
if TRANSACTION-FILE not empty
   then
      for each JOURNAL-RECORD in TRANSACTION-FILE
         if JOURNAL-TYPE = credit
            then add JOURNAL-AMOUNT to total-credit;
            else add JOURNAL-AMOUNT to total-debit;
         fi
      endfor
      if total-credit = total-debit
         then add TRANSACTION-FILE to JOURNAL-FILE;
         else display error message
      fi
   fi
MODULE NAME : 5.2
DESCRIPTION : reverse commitment

Procedure :
get REVERSE-COMMITMENT-INFO from user
get all JOURNAL-RECORD which contain the requisition-number and
are posted to a commitment-ac from JOURNAL-FILE
for each of the JOURNAL-RECORD got
  make up DISPLAY-JOURNAL-SCREEN
  display the DISPLAY-JOURNAL-SCREEN
  ask user to confirm the reversal
  if positive confirmation
    then check whether this commitment has be reversed
    if not yet reversed
      then make up the reversal JOURNAL-RECORD
        post it to the corresponding commitment-ac
        and payable-ac
  fi
fi
endfor
Functional Specification Dictionary
ac-category
/* */
= [ assets-ac / budget-ac / commitment-ac / expense-ac / liability-ac / income-ac ]

ac-id
/* an identification for selecting an account. */
=

ac-rpt-info
/* parameter for generating account report */
= ( account-number + account-name ) + report-destn

ac-screen-hdrrs
/* Headings present on an account screen. Refer to screen layout for exact format. */
=

account-alias
/* any identification that the user assigned to identify the account */
=

account-file
/* file containing information of all the accounts. */
= { account-record }

account-info
/* */
= [ new-account-info / update-account-info ]

account-number
/* a unique number assigned to an account */
= main-ac-nr + ac-category + 0{ sub-ac-nr }4

account-record
/* */
= account-number + account-name + account-balance + account-alias

account-report
/* a report of the information about an account */
=

account-screen
/* It is a data entry or display screen of an account record */
= [ new-ac-screen / display-ac-screen / update-ac-screen ]

account-selection
/* */
= [ new-account-sel / update-account-sel ]

add-supplier-screen
/* a data entry screen for entering new supplier information */
assets-total
/* sum of balances of all assets account*/

bac-input-information
/* */
= [ account-info / journal-info / supplier-info / report-info ]

bal-sheet-cr-total
/* sum of balances of all liability, equity and revenue accounts*/

bal-sheet-dr-total
/* sum of balances of all assets, commitment and expense account*/

bal-sheet-info
/* parameter for generating the balance sheet*/
= ( start date ) + ( end date ) + report-destn

balance-sheet
/* */
= assets-total + commitment-total + expense-total + 
  liability-total + income-total + equity-total + bal-sheet-cr-total + 
  bal-sheet-dr-total

budget-bal-info
/* parameter for generating budget balance report*/
= ( account-number ) + ( account-name ) + report-destn

budget-balance-report
/* a report of the balance of the budget account*/
= { account-number + account-name + total-debit + 
   total-balance }

commitment-total
/* sum of balances of all commitment account*/

delete-supplier-info
/* */
= supplier-code

delete-supplier-screen
/* a data entry screen for entering information for deleting supplier*/

detail-report
/* a report of journal entry of selected accounts*/
= { journal-record + total-balance }

detail-rpt-info
/* parameter for generating detail report*/
= ( start date ) + ( end date ) + ( account-number ) +
( account-name ) + report-destn

display-ac-screen
/* a screen for displaying of an account record*/
= ac-screen-hdrs + account-record

equity-total
/* sum of balances of all budget and equity accounts*/
=

expense-total
/* sum of balances of all expense account*/
=

income-total
/* sum of all balances of all revenue account*/
=

journal-amount
/* amount of money posted by a journal entry*/
=

journal-file
/* file containing all the journal entries*/
= { journal-record }

journal-info
/* */
= [ new-journal-info / reverse-commit-info ]

journal-keywords
/* some user assigned keywords to identify the journal entry*/
=

journal-record
/* */
= account-number + account-name + date + journal-type +
  journal-amount + ( description ) + ( journal-keywords ) + ( supplier-code )
  + ( voucher-number ) + ( requisition-number ) + ( purchase-order-number ) +
  ( journal-doc-nr )

journal-screen
/* a screen for data entry or display of a journal entry. Refer to
  screen layout for exact format.*/
= [ enter-journal-screen / update-journal-screen /
  display-journal-screen ]

journal-selection
/* */
= [ post-trans-sel / reverse-commitment-sel ]

journal-type
/* */
= [ credit / debit ]

**keyw-trace-info**
/* parameter for generating keyword trace report*/
= ( start date ) + ( end date ) + keyword + report-destn

**keyword-trace-report**
/* a report of all journal entries which contains a certain keyword*/
= { journal-record } + total-debit + total-credit

**liability-total**
/* sum of balances of all liability account*/
=

**main-ac-nr**
/* main account number. It is the first 3 digit of an account number*/
=

**main-menu-selection**
/* selections on the main menu*/
= [ account-selection / journal-selection / supplier-selection / report-selection ]

**main-menu**
/* It is the main menu for the budget/account system. Refer screen layout for the exact screen layout*/
= { main-menu-selection }

**new-ac-screen**
/* a data entry screen for entering new account information*/
= account-record

**new-account-info**
/* information of a new account to be added*/
= account-record

**new-journal-info**
/* information of a new journal entry*/
= journal-record

**new-supplier-info**
/* */
= supplier-code + supplier-name + address + phone + description

**report-destn**
/* it indicates where the report is to be sent*/
= [ printer / terminal ]

**report-info**
/* */
= [ bal-sheet-info / ac-rpt-info / sup-rpt-info /
   summary-rpt-info / detail-rpt-info / keyw-trace-info / budget-bal-info ]
report-screen
/* a data entry screen for entering report parameter*/
= ( start date ) + ( end date ) + ( account-number ) +
   ( account-name ) + ( keyword ) + ( supplier-code ) + report-destn

report-selection
/* */
= [ balance-sheet-sel / account-rpt-sel / supplier-rpt-sel /
   summary-rpt-sel / detail-rpt-sel / keyword-trace-rpt-sel / budget-bal-sel ]

report-type
/* types of report*/
= [ balance-sheet / account-report / supplier-report /
   summary-report / detail-report / keyword-trace-report /
   budget-balance-report ]

sub-ac-nr
/* sub account number. It is in unit of 3 digits. It starts from
   the 5th digit of an account number*/
=

summary-report
/* a report of the balances of selected accounts*/
= { account-number + account-name + debit-total + credit-total +
   total-balance }

summary-rpt-info
/* parameter for generating account summary report*/
= ( start date ) + ( end date ) + ( account-number ) +
   ( account-name ) + report-destn

sup-rpt-info
/* parameter for generating supplier report*/
= ( supplier-code ) + report-destn

supplier-code
/* a unique identification for each supplier*/
= 1{ character }5

gensup-supplier-file
/* */
= { supplier-record }

gensup-supplier-record
/* */
= supplier-code + supplier-name + description +
  address + phone

gensup-supplier-report
/* a report of the information of suppliers*/
= { supplier-record }

gensup-supplier-screen
```c
/* */
= [ add-supplier-screen / delete-supplier-screen /
    update-supplier-screen ]

supplier-selection
/* */
= [ add-supplier-sel / update-supplier-sel / delete-supplier-sel ]

trans-menu-sel
/* functions available for updating current transaction.*/
= [ add-journal-sel / update-journal-sel / list-journal-sel /
    delete-trans-sel / file-trans-sel / exit-trans-sel ]

transaction-menu
/* a menu for user to select function related to the posting of
   a transaction.*/
= { trans-menu-sel }

update-supplier-info
/* */
= supplier-code + supplier-name + description + address + phone

update-supplier-screen
/* a data entry screen for updating supplier information*/
= supplier-record
```
Structure Charts

The Data Flow Diagrams, Data Dictionary and the Process Specifications form the functional specification of the Budget/Account System. The functional specification specifies what functions are to be provided by the system. It does not specify how to achieve the provision of the functions. It is the purpose of the design specifications. The design specifications describe the algorithms involved.

Based on the partitioning of the functional specifications, the system is then further partitioned in modules for implementation. This section is a collection of the structure charts of these modules. It describes the hierarchical relations between these modules. The interfaces between these modules are also defined on these charts.

The first chart is the top level module of the system. All the other charts are arranged in alphabetical order of the top level module on it.
Design Specifications
module name: acrpt
description: generating account report
input parameter:
update parameter:
output parameter:

procedure:
call gvrptrqt( account-rpt-sel : REPORT-SCREEN : status )
if status == true
    then
        if ACCOUNT-NUMBER in REPORT-SCREEN == blank
            then copy ACCOUNT-FILE to temp-file
            else call gallsubac( ACCOUNT-NUMBER : temp-file )
        fi
        for each of the ACCOUNT-RECORD in temp-file
            make up the ACCOUNT-REPORT
        endfor
        if REPORT-DESTN == printer
            then put the ACCOUNT-REPORT to printer
            else put the ACCOUNT-REPORT to terminal
        fi
    fi
module name : addacal
description : add account alias
input parameter :
update parameter :
output parameter :

procedure :
call getvalac(: UPDATE-AC-SCREEN : status)
while ( status == true )
    call edit(: UPDATE-AC-SCREEN : status)
    if status == true
        then call updac( UPDATE-AC-SCREEN )
    fi
    call getvalac(: UPDATE-AC-SCREEN : status)
endwhile
module name: addjnl

description: add journal entry to TRANSACTION-FILE

input parameter:
update parameter: TRANSACTION-FILE
output parameter:

procedure:
while
    call setup( : ENTER-JOURNAL-SCREEN : status )
    if status == quit
        then return
    fi
    call edit( : ENTER-JOURNAL-SCREEN : status )
    while status == true
        if valjnl( ENTER-JOURNAL-SCREEN :: status )
            then get JOURNAL-RECORD from ENTER-JOURNAL-SCREEN
                add JOURNAL-RECORD to TRANSACTION-FILE
                break
        fi
        call edit( : ENTER-JOURNAL-SCREEN : status )
    endwhile
endwhile
module name : addnewac
description : add new account
input parameter :
update parameter :
output parameter :

procedure :
eof = false
while ( eof == false )
call edit( new-ac-screen :status)
if status == true
then call valnwac( new-ac-screen :status)
if status == true
then extract ACCOUNT-RECORD from NEW-AC-SCREEN
add ACCOUNT-RECORD to ACCOUNT-FILE
call bldrelac( new-ac-scrn )
sort ACCOUNT-FILE
eof = true
fi
else eof = true
fi
endwhile
module name : addnewsup
description : add new supplier
input parameter :
update parameter :
output parameter :

procedure :
eof = false
while ( eof == false )
    call edit(: ADD-SUPPLIER-SCREEN : status )
    if status == true
        then call valnewsp( ADD-SUPPLIER-SCREEN :: status )
           if status == true
               then get SUPPLIER-RECORD from ADD-SUPPLIER-SCREEN
                   add SUPPLIER-RECORD to SUPPLIER-FILE
                   eof = true
               fi
            fi
        else eof = true
    fi
endwhile
module name: balsheet

description: generating balance sheet

input parameter:

update parameter:

output parameter:

procedure:
call gvrptrqt( balance-sheet-sel : REPORT-SCREEN : status )
if status == true
    then get all JOURNAL-RECORD which are within the start-date and end-date
        for each of the JOURNAL-RECORD got
            if JOURNAL-TYPE == debit
                then select case according to AC-CATEGORY of ACCOUNT-NUMBER of JOURNAL-RECORD
                    case 1: assets-ac
                        add JOURNAL-AMOUNT to dr-assets
                        break
                    case 2: budget-ac or equity-ac
                        add JOURNAL-AMOUNT to dr-equity
                        break
                    case 3: liability-ac
                        add JOURNAL-AMOUNT to dr-liability
                        break
                    case 4: expense-ac
                        add JOURNAL-AMOUNT to dr-expense
                        break
                    case 5: income-ac
                        add JOURNAL-AMOUNT to dr-income
                        break
                    case 6: commitment-ac
                        add JOURNAL-AMOUNT to dr-commitment
                        break
                endcase
            else
                select case according to AC-CATEGORY of ACCOUNT-NUMBER of JOURNAL-RECORD
                    case 1: assets-ac
                        add JOURNAL-AMOUNT to cr-assets
                        break
                    case 2: budget-ac or equity-ac
                        add JOURNAL-AMOUNT to cr-equity
                        break
                    case 3: liability-ac
                        add JOURNAL-AMOUNT to cr-liability
                        break
                    case 4: expense-ac
                        add JOURNAL-AMOUNT to cr-expense
                        break
                    case 5: income-ac
                        add JOURNAL-AMOUNT to cr-income
                        break
                    case 6: commitment-ac
                        add JOURNAL-AMOUNT to cr-commitment
                        break
                endcase
endcase
fi

ASSETS-TOTAL = dr-assets - cr-assets
COMMITMENT-TOTAL = dr-commitment - cr-commitment
EXPENSE-TOTAL = dr-expense - cr-expense
LIABILITY-TOTAL = cr-liability - dr-liability
INCOME-TOTAL = cr-income - dr-income
EQUITY-TOTAL = cr-equity - dr-equity

BAL-SHEET-CR-TOTAL = ASSETS-TOTAL + COMMITMENT-TOTAL + EXPENSE-TOTAL
BAL-SHEET-DR-TOTAL = LIABILITY-TOTAL + INCOME-TOTAL + EQUITY-TOTAL

if REPORT-DESTN = printer
    then put BALANCE-SHEET to printer
else display BALANCE-SHEET to terminal
fi
module name : bgtbal
description : budget balance
input parameter :
update parameter :
output parameter :

procedure :
call gvrptrqt( budget-bal-sel : REPORT-SCREEN : status )
if status == true
then get ACCOUNT-NUMBER from REPORT-SCREEN
   if ACCOUNT-NUMBER not == blank and AC-CATEGORY not == budget-ac
      then check whether corresponding budget-ac exists
         if not found
            then display error message
            return
         fi
   fi
if ACCOUNT-NUMBER == blank
   then get all ACCOUNT-RECORD of budget-ac from ACCOUNT-FILE to temp-ac-file
else get the ACCOUNT-RECORD of the corresponding budget-ac from ACCOUNT-FILE to temp-ac-file
fi
while ( not end-of temp-ac-file )
   get ACCOUNT-RECORD from temp-ac-file
   create temp-jnl-file
   call gallaejnl( ACCOUNT-NUMBER : temp-jnl-file )
   if ( start-date not == blank or end-date not == blank )
      then call filter(start-date, end-date : temp-jnl-file )
   fi
   call pbgtbal( account-number, account-name : temp-jnl-file, BUDGET-BALANCE-REPORT )
endwhile
if REPORT-DESTN == printer
   then put BUDGET-BALANCE-REPORT to printer
else display BUDGET-BALANCE-REPORT to terminal
fi
fi
module name : bldrelac
description : build related account
input parameter :
update parameter : NEW-AC-SCREEN
output parameter :
procedure :
get ACCOUNT-NUMBER from NEW-AC-SCREEN

if it is a main-account number or a first level sub-account number then
  if corresponding commitment-ac does not exists
    then ask user to confirm creation of commitment-ac
      if positive confirmation
        then make up ACCOUNT-RECORD of the corresponding commitment-ac
        put ACCOUNT-RECORD to NEW-AC-SCREEN
        call edit(: NEW-AC-SCREEN : status)
        if status == true
          then get ACCOUNT-RECORD from NEW-AC-SCREEN
          add ACCOUNT-RECORD to ACCOUNT-FILE
        fi
    fi
  fi
if corresponding budget-ac does not exists
  then ask user to confirm creation of budget-ac
  if positive confirmation
    then make up ACCOUNT-RECORD of the corresponding budget-ac
    put ACCOUNT-RECORD to NEW-AC-SCREEN
    call edit(: NEW-AC-SCREEN : status)
    if status == true
      then get ACCOUNT-RECORD from NEW-AC-SCREEN
      add ACCOUNT-RECORD to ACCOUNT-FILE
  fi
fi
fi
module name : delsup

description : delete SUPPLIER-RECORD

input parameter :

update parameter :

output parameter :

procedure :
call getsupcde(: SUPPLIER-CODE : status)
while ( status not = quit )
    dspsup( SUPPLIER-CODE : DELETE-SUPPLIER-SCREEN )
    ask user to confirm deletion
    if positive confirmation
        then remove corresponding SUPPLIER-RECORD from SUPPLIER-FILE
    fi
    call getsupcde(: UPDATE-SUPPLIER-SCREEN :status)
endwhile
module name: deltran

description: delete current transaction

input parameter:

update parameter: TRANSACTION-FILE

output parameter:

procedure:

discard all JOURNAL-RECORD in TRANSACTION-FILE
module name: dspsup

description: display SUPPLIER-RECORD to DELETE-SUPPLIER-SCREEN

input parameter: SUPPLIER-CODE

update parameter:

output parameter:

procedure:
get SUPPLIER-RECORD as indicated by SUPPLIER-CODE from SUPPLIER-RECORD
put SUPPLIER-RECORD to DELETE-SUPPLIER-SCREEN
display DELETE-SUPPLIER-SCREEN
module name : dtlrpt

descrition : dtlrpt report

input parameter :
update parameter :
output parameter :

procedure :
    call gvrprqt( detail-rpt-sel : REPORT-SCREEN : status )
    if status == true
        then create temp-ac-file
        if ACCOUNT-NUMBER of REPORT-SCREEN == blank
            then copy ACCOUNT-FILE to temp-ac-file
            else call gallsubac( ACCOUNT-NUMBER : temp-ac-file )
        fi
        while ( not end-of temp-ac-file )
            get ACCOUNT-RECORD from temp-ac-file
            create temp-jnl-file
            call gacjnl( ACCOUNT-NUMBER : temp-jnl-file )
            if ( start-date of REPORT-SCREEN not == blank or
                end-date of REPORT-SCREEN not == blank )
                then call filter( start-date, end-date : temp-jnl-file )
            fi
            call pdtlrpt( ACCOUNT-NUMBER , account-name : temp-jnl-file,
                          SUMMARY-REPORT )
        endwhile
    fi
    if REPORT-DESTN == printer
        then put SUMMARY-REPORT to printer
    else put SUMMARY-REPORT to terminal
fi
module name : filechk
description : checking whether transaction amount balance
input parameter : TRANSACTION-FILE
update parameter :
output parameter : status

procedure :
if TRANSACTION-FILE not empty then
   for each JOURNAL-RECORD in TRANSACTION-FILE
      if JOURNAL-TYPE = credit
         then add JOURNAL-AMOUNT to total-credit ;
      else add JOURNAL-AMOUNT to total-debit ;
      fi
   endfor
   if total-credit = total-debit then status = true
      else display error message
         status = false
   fi
else status = true
fi
module name: filter

description: exclude all journal entry in input file which in out of range

input parameter: start-date, end-date

update parameter: temp-jnl-file

output parameter:

procedure:
create temp-file
while (not end-of temp-jnl-file)
    get JOURNAL-RECORD from temp-jnl-file
    if start-date <= date of JOURNAL-RECORD is <= end-date
        then write JOURNAL-RECORD to temp-file
    fi
endwhile

copy temp-file to temp-jnl-file
module name: gacjnl

description: get all JOURNAL-RECORD of an account

input parameter: ACCOUNT-NUMBER

update parameter: temp-jnl-file

output parameter:

procedure:

get all JOURNAL-RECORD of the ACCOUNT-NUMBER and its sub-account from JOURNAL-FILE

put all JOURNAL-RECORD got to temp-jnl-file
module name : gallacjnl
description : get relevant JOURNAL-RECORD of an account
input parameter : ACCOUNT-NUMBER
update parameter : temp-jnl-file
output parameter : 

procedure :
get JOURNAL-RECORD of the ACCOUNT-NUMBER and its corresponding
budget-ac, commitment-account and sub-account from JOURNAL-FILE
put all JOURNAL-RECORD got to temp-jnl-file
module name: gallsubac

description: get the ACCOUNT-RECORD of an account and all its sub-account

input parameter: ACCOUNT-NUMBER

update parameter: temp-file

output parameter:

procedure:
get the ACCOUNT-RECORD as indicated by ACCOUNT-NUMBER from ACCOUNT-FILE
get also the ACCOUNT-RECORD of all its sub-account, corresponding commitment-ac and budget-ac from ACCOUNT-FILE.
put all ACCOUNT-RECORD got to temp-file
module name : genrpt
description : despatch the appropriate report generation routine
input parameter : REPORT-SELECTION
output parameter : 

procedure :
select case according to REPORT-SELECTION
  case 1 : balance-sheet-sel
    call balsheet()
    break
  case 2 : account-rpt-sel
    call acrpt()
    break
  case 3 : supplier-rpt-sel
    call suprpt()
    break
  case 4 : summary-rpt-sel
    call sumrpt()
    break
  case 5 : detail-rpt-sel
    call dtlrpt()
    break
  case 6 : keyword-trace-rpt-sel
    call keywrpt()
    break
  case 7 : budget-bal-sel
    call bgtbal()
    break
endcase
module name : getcmt

description : get commitment entry to be reversed

input parameter :
update parameter : DISPLAY-JOURNAL-SCREEN
output parameter : status

procedure :
get REVERSE-COMMITMENT-INFO from user
get all JOURNAL-RECORD which contain the requisition-number and
are posted to a commitment-ac from JOURNAL-FILE
for each of the JOURNAL-RECORD got
  make up DISPLAY-JOURNAL-SCREEN
  display the DISPLAY-JOURNAL-SCREEN
  ask user to confirm the reversal
  if positive confirmation
    then check whether this commitment has be reversed
      if not yet reversed
        then status = true
        return status
  fi
fi
endfor
module name : getrptac
description : get ACCOUNT-RECORD for report generation
input parameter :
Update parameter : DISPLAY-AC-SCREEN
output parameter : status

procedure :
while
  get AC-ID from user
  if AC-ID == blank
    then status = true
        clear ACCOUNT-NUMBER and account-name of DISPLAY-AC-SCREEN
        return status
    else get all ACCOUNT-RECORD which match AC-ID from ACCOUNT-FILE
        write ACCOUNT-RECORD got to temp-file
  fi
while ( not end-of temp-file )
  get an ACCOUNT-RECORD from temp-file
  display the ACCOUNT-RECORD
  ask user to confirm the correct account
  if positive confirmation
    then put the ACCOUNT-RECORD to UPDATE-AC-SCREEN
    status = true
    return status
  fi
endwhile
endwhile
module name: getsupcde

description: get SUPPLIER-CODE to be updated

input parameter:
update parameter: SUPPLIER-CODE

output parameter: status

procedure:
status = false
while (status == false)
get SUPPLIER-CODE from user
if SUPPLIER-CODE = blank
    then status = quit
else check whether SUPPLIER-CODE exists on SUPPLIER-FILE
    if found
        then status = true
        else status = false
            display error message
    fi
fi
endwhile
return status
module name : getvalac

description : get valid account

input parameter :

update parameter : UPDATE-AC-SCREEN

output parameter : status

procedure :

while

get AC-ID from user

if AC-ID == blank
then status = false
return status

else get all ACCOUNT-RECORD which match AC-ID from ACCOUNT-FILE
write ACCOUNT-RECORD got to temp-file

fi

while ( not end-of temp-file )

get an ACCOUNT-RECORD from temp-file

display the ACCOUNT-RECORD

ask user to confirm the correct account

if positive confirmation
then put the ACCOUNT-RECORD to UPDATE-AC-SCREEN
status = true
return status

fi

endwhile

endwhile
module name : getvaltran
description : get valid transaction
input parameter :
update parameter : TRANSACTION-FILE
output parameter :

procedure :
while
  display TRANSACTION-MENU
  get TRANS-MENU-SEL from user
  select case according to TRANS-MENU-SEL
    case 1 : add-journal-sel
      call addjnl( TRANSACTION-FILE )
      break
    case 2 : update-journal-sel
      call updjnl( TRANSACTION-FILE )
      break
    case 3 : list-journal-sel
      call listjnl( TRANSACTION-FILE )
      break
    case 4 : delete-trans-sel
      call deltran( TRANSACTION-FILE )
      break
    case 5 : file-trans-sel
      call filechk( TRANSACTION-FILE :: status)
      if status == true
        then return status
      fi
      break
    case 6 : exit-trans-sel
      call quitchk( TRANSACTION-FILE :: status)
      if status == quit
        then return status
      else
        if status == true
          then return status
        else break
      fi
      fi
  endwhile
module name : gvldnwac

description : get valid new account information

input parameter :

output parameter : NEW-AC-INFO

procedure :
initialise the AC-SCREEN as follows :
begin
for i := 0 to (sc-fld-cnt - 1) do
    sc-field[i].sc-fld-endptr := 0
endfor
sc-field[0].sc-fld-atr := num-fld
sc-field[1].sc-fld-atr := alpnum-fld
sc-field[2].sc-fld-atr := alpha-fld
sc-field[3].sc-fld-atr := alpnum-fld
end
ok := false
while ok = false do
    if ( edit( AC-SCREEN ) = true )
        then
            if ( vldnwac( AC-SCREEN ) = true )
                then status := true
                    ok := true
            fi
        else status := false
            ok := true
    fi
endwhile
return status
module name : gvrptrqt

description : get valid report request

input parameter : REPORT-SELECTION

update parameter : REPORT-SCREEN

output parameter : status

procedure :

getac-flg = true

select case according to REPORT-SELECTION

    case 1 : balance-sheet-sel
        set FLD-ATR of SUPPLIER-CODE and keyword of REPORT-SCREEN to protect-fld
        getac-flg = false
        break

    case 2 : account-rpt-sel
        set FLD-ATR of start-date, end-date, SUPPLIER-CODE and keyword of REPORT-SCREEN to protect-fld
        break

    case 3 : supplier-rpt-sel
        set FLD-ATR of start-date, end-date and keyword of REPORT-SCREEN to protect-fld
        getac-flg = false
        break

    case 4 : summary-rpt-sel
        set FLD-ATR of supplier-code, keyword of REPORT-SCREEN to protect-fld
        break

    case 5 : detail-rpt-sel
        set FLD-ATR of supplier-code, keyword of REPORT-SCREEN to protect-fld
        break

    case 6 : keyword-trace-rpt-sel
        set FLD-ATR of SUPPLIER-CODE to protect-fld
        break

    case 7 : budget-bal-sel
        set FLD-ATR of start-date, end-date, SUPPLIER-CODE and keyword of REPORT-SCREEN to protect-fld
        getac-flg = false
        break

endcase

set FLD-ATR of ACCOUNT-NUMBER and account-name of REPORT-SCREEN to protect-fld

if getac-flg == true
    then call getrptac(: DISPLAY-AC-SCREEN : status)
        copy ACCOUNT-NUMBER and account-name from DISPLAY-AC-SCREEN to REPORT-SCREEN
fi

call edit(: REPORT-SCREEN : status)

while ( status == true )
    call valrptrqt( REPORT-SCREEN :: status )
    if status == true
        then return status

fi

call edit(: REPORT-SCREEN : status)
endwhile
module name: keywrpt

description: keyword trace report

input parameter:

update parameter:

output parameter:

procedure:

call gvrptrqt( keyword-trace-rpt-sel : REPORT-SCREEN : status )

if status == true

then get all JOURNAL-RECORD which contain keyword in REPORT-SCREEN from JOURNAL-FILE

put all JOURNAL-RECORD to temp-jnl-file

if start-date of REPORT-SCREEN not == blank &&

end-date of REPORT-SCREEN not == blank

then call filter(start-date, end-date : temp-jnl-file )
fi

for each JOURNAL-RECORD in temp-jnl-file

put JOURNAL-RECORD to KEYWORD-TRACE-REPORT

if ( JOURNAL-TYPE == debit )

then add JOURNAL-AMOUNT to debit-total

else add JOURNAL-AMOUNT to credit-total
fi
endfor

if ( REPORT-DESTN == printer )

then put KEYWORD-TRACE-REPORT to printer

else put KEYWORD-TRACE-REPORT to terminal
fi

fi
module name : listjnl

description : list JOURNAL-RECORD in TRANSACTION-FILE

input parameter : TRANSACTION-FILE

update parameter :

output parameter :

procedure :

for each JOURNAL-RECORD in TRANSACTION-FILE
    make up DISPLAY-JOURNAL-SCREEN from the JOURNAL-RECORD
    display the DISPLAY-JOURNAL-SCREEN
endfor
module name : MAIN.MENU
description : get user selection and despatch the corresponding routine
input parameter :
update parameter :
output parameter :

procedure :
display MAIN-MENU
get MAIN-MENU-SELECTION
while ( MAIN-MENU-SELECTION not = quit )
    select case according to MAIN-MENU-SELECTION
    case 1 : new-account-sel
        call addnewac()
        break
    case 2 : update-account-sel
        call addacal()
        break
    case 3 : post-trans-sel
        call posttrans()
        break
    case 4 : reverse-commitment-sel
        call revcmt()
        break
    case 5 : REPORT-SELECTION
        call genrpt(report-selection)
        break
    case 6 : add-supplier-sel
        call addnewsup()
        break
    case 7 : update-supplier-sel
        call updsup()
        break
    case 8 : delete-supplier-sel
        call delsup()
        break
    case 9 : none of the above
        display error message
endcase
get MAIN-MENU-SELECTION
endwhile
module name: pbgtbal

description: make up budget balance of an account

input parameter: ACCOUNT-NUMBER, account-name, temp-jnl-file

update parameter: BUDGET-BALANCE-REPORT

output parameter:

output parameter:

procedure:
while ( not end-of temp-jnl-file )
    get JOURNAL-RECORD from temp-jnl-file
    calculate budget balance according to the following decision table

| assets-ac  | x | x |   |   |   |   |   |
| budget-ac  |   | x | x |   | x | x | x |
| commitment-ac |   | x |   | x |   |   |   |
| expense-ac |   |   |   | x | x | x | x |
| income-ac  |   |   |   |   |   | x | x |
| credit     | x | x | x | x | x | x | x |
| debit      | x | x | x | x | x | x | x |
| add to balance |   | x |   | x |   |   |   |
| subtract from balance | x | x | x |   |   | x | x |

endwhile
module name : pdtlrpt

description : make up DETAIL-REPORT of an account

input parameter : ACCOUNT-NUMBER, account-name, temp-jnl-file

update parameter : SUMMARY-REPORT

output parameter :

procedure :

for each JOURNAL-RECORD in temp-jnl-file

if JOURNAL-TYPE == debit

then add JOURNAL-AMOUNT to debit-total

else add JOURNAL-AMOUNT to credit-total

fi

put JOURNAL-RECORD to DETAIL-REPORT

endfor

if debit-total >= credit-total

then total-balance = debit-total - credit-total

else total-balance = credit-total - debit-total

fi
module name: posttrans

description: handle post transaction selection

input parameter:

update parameter:

output parameter:

procedure:
call getvaltran(: TRANSACTION-FILE :status)
while ( status not = quit )
    add TRANSACTION-FILE to JOURNAL-FILE
    clear TRANSACTION-FILE
    call getvaltran(: TRANSACTION-FILE :status)
endwhile
module name : psumrpt
description : make up SUMMARY-REPORT of an account
input parameter : account-number, account-name, temp-jnl-file
update parameter : SUMMARY-REPORT
output parameter :

procedure :
for each JOURNAL-RECORD in temp-jnl-file
    if JOURNAL-TYPE == debit
        then add JOURNAL-AMOUNT to debit-total
        else add JOURNAL-AMOUNT to credit-total
    fi
endfor
if debit-total >= credit-total
    then total-balance = debit-total - credit-total
    else total-balance = credit-total - debit-total
fi
module name : quitchk
description : check transaction balance before exit
input parameter : TRANSACTION-FILE
update parameter :
output parameter : status

procedure :
ask user to confirm filing TRANSACTION-FILE before exit
if positive confirmation
   then call filechk( TRANSACTION-FILE :: status )
else status = quit
fi
return status
module name: revcmt

description: reverse commitment

input parameter:

update parameter:

output parameter:

procedure:

call getcmt(: DISPLAY-JOURNAL-SCREEN :status)

if status = true

then get JOURNAL-RECORD from DISPLAY-JOURNAL-SCREEN

make up two reversal JOURNAL-RECORD

add the two JOURNAL-RECORD to JOURNAL-FILE

fi
module name: setSUP

description: set up UPDATE-SUPPLIER-SCREEN

input parameter: SUPPLIER-SCREEN

update parameter: UPDATE-SUPPLIER-SCREEN

output parameter:

procedure:
get SUPPLIER-RECORD as indicated by SUPPLIER-RECORD from SUPPLIER-FILE
put SUPPLIER-RECORD to UPDATE-SUPPLIER-SCREEN
module name : setup

description : set up ENTER-JOURNAL-SCREEN

input parameter :
update parameter : ENTER-JOURNAL-SCREEN
output parameter :

procedure :

)
module name : sumrpt
description : summary report
input parameter : update parameter : output parameter :

procedure :
call gvrptrqt( summary-rpt-sel : REPORT-SCREEN : status )
if status == true
  then create temp-ac-file
    if ACCOUNT-NUMBER of REPORT-SCREEN == blank
      then copy ACCOUNT-FILE to temp-ac-file
    else call gallsubac( ACCOUNT-NUMBER : temp-ac-file )
  fi
while ( not end-of temp-ac-file )
  get ACCOUNT-RECORD from temp-ac-file
  create temp-jnl-file
  call gacjn1( ACCOUNT-NUMBER : temp-jnl-file )
  if ( start-date of REPORT-SCREEN not == blank or
      end-date of REPORT-SCREEN not == blank )
    then call filter( start-date, end-date : temp-jnl-file )
  fi
  call psumrpt( account-number, account-name : temp-jnl-file, SUMMARY-REPORT )
endwhile
if REPORT-DESTN == printer
  then put SUMMARY-REPORT to printer
else put SUMMARY-REPORT to terminal
fi
module name : suprpt

description : supplier report

input parameter : 
update parameter : 
output parameter : 

procedure :
call gvrprtrqt( supplier-rpt-sel : REPORT-SCREEN : status )
if status == true
  then
    if SUPPLIER-CODE of REPORT-SCREEN == blank
      then copy SUPPLIER-FILE to temp-sup-file
      else get all SUPPLIER-RECORD which match the SUPPLIER-CODE
           from SUPPLIER-FILE
           put SUPPLIER-RECORD got to temp-sup-file
    fi
    for each SUPPLIER-RECORD in temp-sup-file
      put SUPPLIER-RECORD to SUPPLIER-REPORT
    endfor
    if ( REPORT-DESTN == printer )
      then put SUPPLIER-REPORT to printer
      else put SUPPLIER-REPORT to terminal
    fi
  fi
module name: updac

description: update account file

input parameter: UPDATE-AC-SCREEN

update parameter:

output parameter:

procedure:

get ACCOUNT-RECORD from UPDATE-AC-SCREEN

rewrite ACCOUNT-RECORD to ACCOUNT-FILE
module name : updjnl

description : update JOURNAL-RECORD in TRANSACTION-FILE

input parameter : update parameter : TRANSACTION-FILE

output parameter :

procedure :
for each JOURNAL-RECORD in TRANSACTION-FILE
make up update-journal-screen
ask user to confirm update
if positive confirmation
then get NEW-JOURNAL-INFO from user
   rewrite the JOURNAL-RECORD to TRANSACTION-FILE
fi
endfor
module name : updsup

description : update SUPPLIER-RECORD

input parameter :
update parameter :
output parameter :

procedure :

call getsupcde(: SUPPLIER-CODE : status)
while ( status not = quit )
    setup( SUPPLIER-CODE : UPDATE-SUPPLIER-SCREEN )
    call edit(: UPDATE-SUPPLIER-SCREEN :status)
    if status == true
        then get SUPPLIER-RECORD from UPDATE-SUPPLIER-SCREEN
            rewrite SUPPLIER-RECORD to SUPPLIER-FILE
    fi
    call getsupcde(: UPDATE-SUPPLIER-SCREEN :status)
endwhile
module name: valjnl
description: validate journal entry
input parameter: ENTER-JOURNAL-SCREEN
update parameter:
output parameter: status

procedure:
status = true
validate the JOURNAL AMOUNT
if not valid
   then status = false
fi
check whether SUPPLIER-CODE exists in SUPPLIER-FILE
if not found
   then status = false
fi
return status
module name : valnewsup
description : validate new SUPPLIER-RECORD
input parameter : ADD-SUPPLIER-SCREEN
update parameter :
output parameter : status

procedure :
get SUPPLIER-CODE from ADD-SUPPLIER-SCREEN
if SUPPLIER-CODE already exists on SUPPLIER-FILE
    then status = false
        display error message
    else status = true
fi
return status
module name : valnwac
description : validate new account
input parameter : NEW-AC-SCREEN
update parameter :
output parameter : status

procedure :
get ACCOUNT-NUMBER and account-name from NEW-AC-SCREEN
select case according to ACCOUNT-NUMBER
  case 1 : it is a main-account number
    if ACCOUNT-NUMBER exist on ACCOUNT-FILE
      then display error message
      status = false
    fi
  break
  case 2 : it is a sub-account number
    if its MAIN-AC-NR and upper level SUB-AC-NR exist on ACCOUNT-FILE
      then if ACCOUNT-NUMBER does exist on ACCOUNT-FILE
        then status = true
        else status = false
        display error message
      fi
    else status = false
    display error message
  fi
  break
endcase
if account-name not present
then status = false
  display error
fi
return status
module name: va1rptrqt

description: validate report request

input parameter: REPORT-SCREEN

update parameter:

output parameter: status

procedure:

status = true

if start-date of REPORT-SCREEN is invalid
   then status = false
   display error message
fi

if end-date of REPORT-SCREEN is invalid
   then status = false
   display error message
fi

if start-date > end-date
   then status = false
   display error message
fi

return status
Design Specification Dictionary
**ac-category**

/* */

= [ assets-ac / budget-ac / commitment-ac / expense-ac / liability-ac / income-ac ]

**ac-id**

/* an identification for selecting an account */

= 

**ac-rpt-info**

/* parameter for generating account report */

= ( account-number + account-name ) + report-destn

**ac-screen-hdrs**

/* Headings present on an account screen. Refer to screen layout for exact format */

= 

**account-alias**

/* any identification that the user assigned to identify the account */

= 

**account-file**

/* file containing information of all the accounts. A sequential file arranged in ascending order of the main account number */

= { account-record }

**account-info**

/* */

= [ new-account-info / update-account-info ]

**account-number**

/* main-ac-nr + ac-category + 0{ sub-ac-nr }4

= account-record + account-number + account-name + account-balance + account-alias + newline

**account-report**

/* a report of the information about an account */

= 

**account-screen**

/* It is a data entry or display screen of an account record */

= [ new-acc-screen / display-acc-screen / update-acc-screen ]

**account-selection**

/* */

= [ new-account-sel / update-account-sel ]
add-supplier-screen
/* a data entry screen for entering new supplier information.
   Refer to screen-struct for its physical structure. */
= supplier-code + supplier-name + description + address + phone

assets-total
/* sum of balances of all assets account */
=

bac-input-information
/* */
= [ account-info / journal-info / supplier-info / report-info ]

bal-sheet-cr-total
/* sum of balances of all liability, equity and revenue accounts */
=

bal-sheet-dr-total
/* sum of balances of all assets, commitment and expense account */
=

bal-sheet-info
/* parameter for generating the balance sheet */
= ( start date ) + ( end date ) + report-destn

balance-sheet
/* */
= assets-total + commitment-total + expense-total + liability-total + income-total + equity-total + bal-sheet-cr-total + bal-sheet-dr-total

budget-bal-info
/* parameter for generating budget balance report */
= ( account-number ) + ( account-name ) + report-destn

budget-balance-report
/* a report of the balance of the budget account */
= { account-number + account-name + total-debit + total-debit + total-balance }

commitment-total
/* sum of balances of all commitment account */
=

delete-supplier-info
/* */
= supplier-code

delete-supplier-screen
/* a data entry screen for entering information for deleting supplier */
= supplier-code + supplier-name + description + address + phone + screen-mask
detail-report
/* a report of journal entry of selected accounts*/
= { journal-record + total-balance }

detail-rpt-info
/* parameter for generating detail report*/
= ( start date ) + ( end date ) + ( account-number ) +
( account-name ) + report-destn

display-ac-screen
/* a screen for displaying of an account record. Refer to screen-struct for the physical structure.*/
= account-number + account-name + account-balance + account-alias

display-journal-screen
/* a display screen for display a journal entry. Refer to screen-struct for the physical structure.*/
= journal-record

enter-journal-screen
/* a data entry screen for entering new journal entry. Refer to screen-struct for its physical structure.*/
= journal-record

equity-total
/* sum of balances of all budget and equity accounts*/
=

detail-report
expense-total
/* sum of balances of all expense account*/
=
field-mask
/* information describing a field on a screen*/
= fld-hdr-xy + fld-hdr + fld-data-xy + fld-data-lng + fld-atr

fld-atr
/* attribute of a field*/

fld-data
/* a buffer which contains the data of a field on the screen*/
=

fld-hdr
/* the actual text of the header of a field*/
=

income-total
/* sum of all balances of all revenue account*/
=
journal-amount
/* amount of money posted by a journal entry*/

journal-file
/* file containing all the journal entries. It is a serial
 file in which the journal-records are arranged in the order of posting.*/
= { journal-record }

journal-info
/* */
= [ new-journal-info / reverse-commit-info ]

journal-keywords
/* some user assigned keywords to identify the journal entry*/

journal-record
/* a record in the journal file. Each field is separated by the
 character '|' and each record is separated by a newline character.*/
= account-number + account-name + date + journal-type +
 journal-amount + ( description ) + ( journal-keywords ) + ( supplier-code ) +
 ( voucher-number ) + ( requisition-number ) + ( purchase-order-number ) +
 ( journal-doc-nr )

journal-screen
/* a screen for data entry or display of a journal entry. Refer to
 screen layout for exact format.*/
= [ enter-journal-screen / update-journal-screen /
 display-journal-screen ]

journal-selection
/* */
= [ post-trans-sel / reverse-commitment-sel ]

journal-type
/* */
= [ credit / debit ]

keyw-trace-info
/* parameter for generating keyword trace report*/
= ( start date ) + ( end date ) + keyword + report-destn

keyword-trace-report
/* a report of all journal entries which contains a certain keyword*/
= { journal-record } + total-debit + total-credit

liability-total
/* sum of balances of all liability account*/

main-ac-nr
/* main account number. It is the first 3 digit of an account number.*/
main-menu-selection
/* selections on the main menu*/
= [ account-selection / journal-selection / supplier-selection /
   report-selection ]

main-menu
/* It is the main menu for the budget/account system. Refer
   screen layout for the exact screen layout.*/
= { main-menu-selection }

new-ac-screen
/* a data entry screen for entering new account information*/
= account-record

new-account-info
/* information of a new account to be added*/
= account-record

new-journal-info
/* information of a new journal entry*/
= journal-record

new-supplier-info
/* */
= supplier-code + supplier-name + address + phone + description

report-destn
/* it indicates where the report is to be sent*/
= [ printer / terminal ]

report-info
/* */
= [ bal-sheet-info / ac-rpt-info / sup-rpt-info /
   summary-rpt-info / detail-rpt-info / keyw-trace-info / budget-bal-info ]

report-screen
/* a data entry screen for entering report parameter. Refer
   to screen-struct for the physical structure.*/
= ( start date ) + ( end date ) + ( account-number ) +
  ( account-name ) + ( keyword ) + ( supplier-code ) + report-destn

report-selection
/* */
= [ balance-sheet-sel / account-rpt-sel / supplier-rpt-sel /
   summary-rpt-sel / detail-rpt-sel / keyword-trace-rpt-sel / budget-bal-sel ]

report-type
/* types of report*/
= [ balance-sheet / account-report / supplier-report /
   summary-report / detail-report / keyword-trace-report /
budget-balance-report ]

reverse-commitment-info
/* */
= requisition-number

screen-mask
/* a data struct which together with the corresponding data to be
displayed form a display screen*/
= { field-mask }

screen-struct
/* a data structure which describes the layout of a screen*/
= { fld-data } + screen-mask

sub-ac-nr
/* sub account number. It is in unit of 3 digits. It starts from
the 5th digit of an account number*/
=

summary-report
/* a report of the balances of selected accounts*/
= { account-number + account-name + debit-total + credit-total +
total-balance }

summary-rpt-info
/* parameter for generating account summary report*/
= ( start date ) + ( end date ) + ( account-number ) +
( account-name ) + report-destn

sup-rpt-info
/* parameter for generating supplier report*/
= ( supplier-code ) + report-destn

supplier-code
/* a unique identification for each supplier*/
= 1{ character }5

supplier-file
/* a sequential file arranged in ascending order of the
supplier code.*/
= { supplier-record }

supplier-record
/* a record in the supplier. Each field is separated by the character
' | ' and each record is separated by a newline character.*/
= supplier-code + ' | ' + supplier-name + ' | ' + description +
' | ' + address + ' | ' + phone + ' | ' + newline

supplier-report
/* a report of the information of suppliers*/
= { supplier-record }

supplier-screen
/* */
= [ add-supplier-screen / delete-supplier-screen / 
  update-supplier-screen ]

supplier-selection
/* */
= [ add-supplier-sel / update-supplier-sel / delete-supplier-sel ]

trans-menu-sel
/* functions available for updating current transaction */
= [ add-journal-sel / update-journal-sel / list-journal-sel / 
  delete-trans-sel / file-trans-sel / exit-trans-sel ]

transaction-file
/* a temporary file which contains the journal-record of a transaction */
= { journal-record }

transaction-menu
/* a menu for user to select function related to the posting of 
  a transaction */
= { trans-menu-sel }

update-ac-screen
/* a data entry screen for entering new account alias. Refer to 
  screen-struct for the physical structure */
= account-record

update-account-info
/* */
= account-record

update-supplier-info
/* */
= supplier-code + supplier-name + description + address + phone

update-supplier-screen
/* a data entry screen for updating supplier information */
= supplier-record