# How to Configure the QConnect DYNAKEY File

Purpose of Document
Human Factors Considerations in Designing DYNAKEY screens
What commands cause the display and key definitions to change?
How do I determine the order in which the keys will be displayed on the screen?
How do I change the description posted alongside a key?
How do I set up a dynakey to act like a regular application key?
How can I generate a string of keys to be sent to the application?
How can I use DYNAKEY to send keystrokes in addition to numbers I have keyed?
How can I define a DYNAKEY to enter an exact amount from a balance due?
How can I define a DYNAKEY to round up a balance due?
How can I define a DYNAKEY to add a specific amount to a balance due?
How do I define a DYNAKEY to cause a new screen to be displayed?
Can I display different DYNAKEY screens depending on whether data is keyed?
How can I prevent the display of sensitive information on the customer screen?
Can I define more than 8 keys to be displayed when a DYNAKEY is pressed or a state change occurs?
Can I cause display of lists of information on the dynakey?
Is there a way to easily define a series of keys that may be referenced from many different sources?
Do the DYNAKEY commands have to be defined in any particular order?
Can I prevent unnecessary reset of the DYNAKEY screen?
How can I make Dynakey behave differently on different terminals?
Sample DYNAKEY file

## Purpose of Document

This document provides basic concepts for programming commands used to configure the DynaKey peripheral on any POS terminal with the IBM Supermarket Application. The configuration (text) file created through the use of these commands is saved under the naming convention of DYNAKEY.XXX, where XXX is the terminal number. The default application name is DYNAKEY.000. Any terminal that does not have a specific DYNAKEY.XXX file assigned to it will use this default application. All DYNAKEY.XXX files will reside in the "ADX_UDT1" file directory on the IBM store controller which is configured as the Master File Server.

The DYNAKEY.XXX file allows the customer to dynamically change the definition of a series of dynamic keys located adjacent to the DYNAKEY screen. Depending on the state of the application at any point in time, the key definitions could be different.

## Human Factors Considerations in Designing DYNAKEY screens

The following suggestions are made in the interest of increasing screen useability and consistency:

- When the user is supposed to enter data prior to hitting a DYNAKEY, the following format should be used:

DESC="->Key Name"     (Ex: DESC="->Quantity")

- If the DYNAKEY is designed to display a new menu of DYNAKEY keys:

DESC="Menu >>"  (Example:  DESC="Departments >>")

- If a DYNAKEY can either accept keyed date from the operator or call for a menu display if no data is keyed, use the following format:

DESC="->FunctionKey >>"   (Example: ->Quantity >> "

- The use of  lower case in DYNAKEY descriptions is encouraged since the text becomes more readable
- Provide menu-driven DYNAKEY screens for complex operator inputs

**What commands cause the display and key definitions to change?**

This change can be intiated from several different sources:

1. The STATE command and associated INPUTS parameters
2. The GROUP command and associated INPUTS parameters
3. Call of a GROUP by a KEYMAP-defined motor key
4. Call of a GROUP via NEXT parameter

When an application begins, the STATE command always puts up the first screen. The STATE command is associated with a unique state value and is executed anytime that particular state value is opened. A new state is generated by the application. In a standard IBM application, different states may make certain keys invalid or require the use of unique key sequences. DYNAKEY layouts associated with a particular state are displayed each time that specific state value is opened (see the section " Can I prevent a reset of the DYNAKEY display " for the one exception to this rule).  The INPUTS parameter specifies the DYNAKEYS to be displayed whenever the specified STATE is entered.  The DESC parm is also displayed in the upper left corner of the DYNAKEY display.

Example:

STATE=10 DESC="**** ITEM SALES ****" INPUTS=G1,G2,G3,G4,G5,G6,G7,G8 In this case, when state 10 is entered, keys associated with groups 1-8 are displayed on the DYNAKEY screen. In addition, the descriptor field, "**** ITEM SALES ****", is displayed in the upper left corner of the DYNAKEY screen.

A GROUP command is generally called by a STATE or another GROUP command.  The example above shows a series of groups being called by a STATE command.  Similarly, a GROUP command can call other GROUP commands.

Example of a GROUP calling another GROUP:

GROUP=100 DESC="SUPERVISOR MENU" INPUTS=G101,G102,G103,G104

In this case, GROUP=100 (G100) represents a DYNAKEY which has the SUPERVISOR MENU descriptor. When the operator presses this DYNAKEY ("SUPERVISOR MENU"), descriptors represented by G101, G102, G103, and G104 are displayed as the new DYNAKEYS. In addition, the phrase, "SUPERVISOR MENU" is displayed in the upper left area of the DYNAKEY screen.

A GROUP command is actually executed (not simply displayed) under three conditions:

- The DYNAKEY associated with the GROUP is pressed by the operator
- A motor key associated with the GROUP number is specified in the KEYMAP file and that key is pressed by the operator

Example:

In the KEYMAP.xxx file, the following pertinent key definition exists: S-F3 = GROUP7 /* CASHIER MENU */ In the DYNAKEY.xxx file, the following pertinent key definition exists: GROUP=7 DESC="CASHIER MENU" INPUTS=G201,G202,G203,G204,G205,F75,G207,G211In this example, pressing the motor key S-F3 causes display of the key groups and function keys which are defined by Group #7. In addition, the phrase, "CASHIER MENU", is displayed in the upper left corner of the screen.

- A GROUP is specified by the NEXT parameter on a GROUP statement

Example:

```
 GROUP=3  DESC="TOTAL" DATA=81 INPUTS=G11,G12,G13,G14
GROUP=500 DESC="ENTER COUPON AMT" DATA=%,80 INPUTS=G501,G502 NEXT=G3
```

In this example, if the operator enters data prior to pressing the "ENTER COUPON AMT" DYNAKEY, the GROUP3 command will be executed. See the section, Can I display different DYNAKEY screens depending on whether data is keyed for more information on the use of the NEXT parameter.

**How do I determine the order in which the keys will be displayed on the screen?**

They will always be displayed in the order in which they appear in the INPUTS or LIST parameters.

**How do I change the Description posted alongside a key?**

Use to the DESC parameter associated with the display of a particular key and change that descriptor. Note that the descriptor length is limited to a total of 20 characters. When this key is pressed, the DESC string will be written to the upper left area of the DYNAKEY screen unless a state change takes place simultaneously. In this case, the DESC field of the STATE command will take precedence.

**How do I set up a dynakey to act like a regular application key?**

Use the FCODE command. The FCODE must be defined to be the function code that is given to the application by the standard IBM application. These codes are normally shown in the Planning Guide for any given application. These codes would be listed in a section dealing with the keyboard layouts used by the application. This FCODE would be referred to by other statements as Fxx where xx is the function number .

Example:

FCODE=61 DESC="SIGN ON/OFF"

For this example, the text "SIGN ON/OFF" would be displayed when that key became posted on the DYNAKEY screen (the result of a STATE command or an INPUTS parameter using F61). The code "61" would be sent to the application if that particular key were pressed by the operator. Note that any data entered by the operator prior to pressing the DYNAKEY associated with the function code would also be sent to the application. Since no INPUTS parms are specified in this case, any change in the screen would be caused by the expected STATE change. Entry of the Sign On code would normally cause a state change and display of a new screen associated with the new STATE.

Note: There is an exception to this operation. If the FCODE statement contains an INPUTS parameter AND no data is entered by the operator, the specified function code is NOT generated. In this case, the statement operates similar to a GROUP statement.

Example:

FCODE=75 DESC="QUANTITY" INPUTS=G301,G302,G303,G304,G305,G306 In this case, if the operator keys in a quantity then hits this DYNAKEY, the quantity entered by the operator followed by the "75" function code will be sent to the application and the INPUTS parms will be ignored. If no data is entered by the operator, nothing is sent to the application, but the DYNAKEY screen will now display the DYNAKEY groups represented, in sequence, by G301, G302, etc.

**How can I generate a string of keys to be sent to the application?**

Use the GROUP command and the DATA parameter to do this. The group number is used to make the group definition unique. The DATA parameter defines what function codes are sent to the application. Successive key codes are separated by commas. These codes are normally shown in the Planning Guide for any given application. These codes would be listed in a section dealing with the keyboard layouts used by the application. This GROUP command would be referred to by other statements as Gxx where xx is the group number . Any keyboard data would precede this data stream.

Example:

GROUP=2 DESC="GIFT CERT $ 50.00 " DATA=5,0,0,0,94

In this example, depression of this key would cause a series of keystrokes to appear as if the operator had entered a number, 5000, followed by the Miscellaneous Tender key.

**How can I use DYNAKEY to send keystrokes in addition to numbers I have keyed?**

Normally, any numeric data entered from the keyboard prior to a DYNAKEY entry is followed by the DATA codes specified. If the keyed data should go elsewhere in a DATA sequence, a "%" character should be used to specify where the keyed numeric data will be inserted.

Example #1:

GROUP=44 DESC="KEY CUPN AMT HERE" DATA=80 In this example, when this key is hit, data from the keyboard is sent to the application followed by an "80" function key (ENTER). If no data is keyed, only an "80" function key is sent to the application. In this example, no NEXT or INPUTS statement is specified. This example implies that a state change will cause the next display to occur. Example #2: GROUP=65 DESC="MFR COUPON $" DATA=65,1,9,1,85,%,80 INPUTS=G20,G21 In this example, when the DYNAKEY is hit, the % symbol specifies that data from the keyboard is sent to the application after the "85" key and before the "80" (ENTER) key. If no numeric data is keyed by the user, only the data up to the "%" location (65,1,9,1,85) is sent to the application and the keys specified by the INPUTS statement are displayed..

**How can I define a DYNAKEY to enter an exact amount from a balance due?**

Use the +R0 parameter with the DATA entry as below. Note that the balance due amount due will be displayed immediately to the right of the DESC field on the DYNAKEY screen. Note that the balance due information can be obtained two ways. A special Bal Due driver can provide the exact balance due to DYNAKEY. This requires a special User Exit to implement, but is more accurate. If the driver is not available, the Bal Due comes from the last amount displayed in the 2x20 area of the display.

Example:

GROUP=11 DESC="EXACT AMT $ " DATA=+R0,91

In this example, the "+R0" data parameter substitutes the balance due from the application as if an operator had keyed it. This number is then followed by the "91" key (Cash).

**How can I define a DYNAKEY to round up a balance due?**

Use the +Rxx parameter with the DATA entry as below. Note that the balance due will be displayed immediately to the right of the DESC field on the DYNAKEY screen. Note that the balance due information can be obtained two ways. A special Bal Due driver can provide the exact balance due to DYNAKEY. This requires a special User Exit to implement, but is more accurate. If the driver is not available, the Bal Due comes from the last amount displayed in the 2x20 area of the display. This is useful in providing common payment amounts offered by the customer.

Example:

GROUP=9311 DESC="$ " DATA=+R100,91

In this example, the "+R100" data parameter substitutes the balance due from the application and rounds it to the next dollar as if an operator had keyed it. This number is then followed by the CASH key(code 91).

**How can I define a DYNAKEY to add a specific amount to a balance due?**

Use the +xx parameter with the DATA entry as below. Note that the balance due plus the "xx" amount will be displayed immediately to the right of the DESC field on the DYNAKEY screen. Note that the balance due information can be obtained two ways. A special Bal Due driver can provide the exact balance due to DYNAKEY. This requires a special User Exit to implement, but is more accurate. If the driver is not available, the Bal Due comes from the last amount displayed in the 2x20 area of the display.

Example:

```
GROUP=9311 DESC="$ " DATA=+1000,92
```

In this example, the "+1000" data parameter substitutes the balance due from the application and adds ten dollars as if an operator had keyed it. This number is then followed by the CHECK key(code 92).

**How do I define a DYNAKEY to cause a new screen to be displayed?**

This is done using the INPUTS parameter on a GROUP or FCODE statement. The items in the INPUTS list are displayed on the DYNAKEY screen in sequence in which they appear in the list.

Example:

```
GROUP=3 DESC="FOOD STAMPS >" DATA=93,81 INPUTS=G41,G42
```

In this example, pressing this key will send data to the application (codes 93 and 81), then display GROUP 41 and GROUP 42 on the dynakey screen.

**Can I display different DYNAKEY screens depending on whether data is keyed?**

Yes, you use the NEXT command. The NEXT command must be used in conjunction with the % sign. The % sign specifies where data, if keyed by the operator, is placed in the sequence of information. If data is keyed by the operator, the command sp ecified by the NEXT instruction is actually EXECUTED, not simply displayed. If no data is keyed, the statement behaves as it normally would without a NEXT statement. If no operator data is keyed, all data from the % sign onward is truncated from the string and the INPUTS command is executed.

Two different results can be had using the NEXT command:

Case 1: If no data is entered by the operator, all data in the string from the % statement on is truncated and sent to the application. In addition, the INPUTS are displayed.

Example:

```
GROUP=20 "->Enter Chk Acct #" DATA=100,% INPUTS=G21 NEXT=G21
GROUP=21 "->Enter Chk Acct#" DATA=90,1,78 INPUTS=G25
GROUP=25 "->Enter Check $" DATA=92
```

In this example, if the operator keys no data, group 20 sends the data string: "100" to the appl (no state change occurs). The INPUTS are then displayed which cause Group 21 to be displayed, prompting the operator for the check account number. Note that, since no % is used in the group 21 sequence, the keyed data will preceed the "90" command. The sequence of operation in this example is: Group 20 is displayed on dynakey screen, dynakey 20 is pressed, 100 is sent to the application (no state change occurs), group 21 is displayed, operator keys data, presses DYNAKEY 21 key: keyed data, 100,1,78 is sent to the application (no state change occurs), group 25 is displayed with "enter check $" prompt, keyed data, group 25 key is pressed: keyed data, 92 is sent to the application (state change occurs).

Case 2: If the NEXT command is used with the % statement and data is entered by the operator, the full string of specified DATA statements and keyed data is sent to the application. In addition, the DYNAKEY specified by the NEXT statement is executed and any associated DATA statements are also sent to the application.

Example:
GROUP=20 "->Enter Chk Acct #" DATA=100,% INPUTS=G21 NEXT=G21
GROUP=21 "->Enter Chk Acct#" DATA=90,1,78 INPUTS=G25
GROUP=25 "->Enter Check $" DATA=92

In this example, Group 20 sends the data string: 100,xxxxx to the application, where xxxxx represents operator-keyed data. The NEXT command executes GROUP 21, sending the data string "90,1,78" to the application. Since this statement is executed, Group 25 is displayed, prompting the operator for the amount of the check.

Note: The user should be aware that in any of the above situations, the data sent to the application could result in a STATE change that would override either the expected display of the INPUTS or the execution of a NEXT statement.

**How can I prevent the display of sensitive information on the customer screen?**

Normally, data written to the printer will also be displayed on the simulated customer receipt tape on the DYNAKEY display or, alternatively, on the Customer Information Display (VGA). The DYNAKEY file offers a way to prevent display of certain print lines. Suppression of display of CID data is toggled on by entries in the DYNAKEY.xxx file. For the case in which a particular dynakey initiates suppression, a ",10" is put at the end of the DATA string. Two examples follow: GROUP=32021 DESC="TENDER TILL REPORT" DATA=100,91,10

GROUP=9504 DESC="GIVE MONEY BACK" DATA=0,94,10

Examples of dynakey definitions that do NOT trigger CID data display suppression follow:
GROUP=32021 DESC="TENDER TILL REPORT" DATA=100,91

GROUP=9504 DESC="GIVE MONEY BACK" DATA=0,94

The ",10" string is not sent to the application, but is used internally to signal QCONNECT to suppress future writes to the CID. This suppression will be in effect until a trigger to redisplay is seen by the QCONNECT emulator code. CID redisplay is triggered by either the scan of an item or a DYNAKEY is hit that does not suppress CID display (has no ",10" at the end of its definition).

The customer should be aware that, the CDIFILE (or QCDIFILE) offers another way to suppress the display of certain print lines. This alternative method may be simpler to implement. Please refer to the CDIFILE sample available on web site, WWW.QVSSOFTWARE.COM. Look under HELP files to find this sample. In short, a parameter is available to be entered into the CDIFILE which will allow an entire print line to be suppressed from display on either the DYNAKEY or Customer Information Display (CID) if a particular phrase is contained within that print line.

**Can I define more than 8 keys to be displayed when a DYNAKEY is pressed or a state change occurs?**

Yes. In this event, DYNAKEY displays the first 7 keys, then automatically creates an 8th entry, the MORE key. When the MORE key is pressed, up to 7 more entries are displayed, along with the MORE key. If there are more DYNAKEYS to be displayed, they continue to be displayed in a similar fashion each time the MORE key is hit. On the last page of DYNAKEYS, pressing the MORE key causes display of the first page of DYNAKEYs and so on.

**Can I cause display of lists of information on the dynakey?**

Display of lists is done using the LIST parameter. This feature is often used to show the cashier the item or department code for a variety of produce. The list to be displayed is contained in a file kept on the Master File Server store controller in directory, ADX_UDT1:. Each line of text within the list file can contain up to 36 characters. The DYNAKEY screen is capable of displaying up to 24 lines of text. The text can be displayed in either upper or lower case. If the list file contains more than 24 lines of text, the eighth dynakey will automatically become a <<MORE>> key. Depression of this key will display a second, third, fourth, etc. screen of text. On the last screen of text, a MORE key will cause re-display of the first screen of text, etc. To exit the list file screen, you must press a keyboard key, not a DYNAKEY. Normally, the CLEAR key (defined on the keyboard) would cause a STATE command to update the DYNAKEY definitions. Example:

```
  LIST=51 DESC="MENS TOILETRIES" FILE=DEPT50.LST
```

The LIST command identifies the LIST number, the description to be used when they DYNAKEY is displayed on screen, and the name of the file to be displayed. The LIST command would normally be called as part of a STATE command or an INPUTS or NEXT statement on another DYNAKEY definition.

Example:

```
GROUP=50 DESC="NON-FOOD" INPUTS=L51,L52,L53,L54,L55,L56,L57,L58
```

The GROUP command calls for the display of 8 LIST dynakeys (L51-L58).

Contents of a sample LIST file, (DEPT51.LST):

```
510 - Toothpaste

511- Shaving Cream

512 - Toothbrushes

513 - Razor Blades

514 - Anti-Fungal Cream
```

**Is there a way to easily define a series of keys that may be referenced from many different sources?**

Use the INLIST command. This simply provides an easy way to reference a commonly-used DYNAKEY display. The INLIST, once defined, can be easily referenced.

Example:

```
INLIST=10 INPUTS=G1,G2,G3,L1,L2,L3,F1
```

It may be referenced as follows:

GROUP=55 DESC="MAIN MENU" INPUTS=I10

In this case, when the Group 55 DYNAKEY is pressed, the INPUTS defined by the I10 INLIST will be displayed.

**Do the DYNAKEY commands have to be defined in any particular order?**

Only in one case: The LIST commands should come at the end of the DYNAKEY.XXX file preceded by the STATE commands.

**Can I prevent unnecessary reset of the DYNAKEY screen?**

Only in certain limited conditions. Customers have been annoyed when a scan of a check causes the application to re-open an already open state. This causes redisplay of the "home" screen, something the customer did not want to have happen. For this limited circumstance, this redisplay can be prevented. The command to prevent the re-display of the DYNAKEY screen when the CURRENT state is re-opened is set when any DYNAKEY with the HOLD_PROMPT parm is hit by the customer. The parm is part of the string of data that calls a DYNAKEY screen which you do not want to be unnecessarily reset. The parm is invoked as follows:

HOLD_PROMPT=YES

Example:

GROUP=9206 DESC="->ENTER CHECK AMT" DATA=92 HOLD_PROMPT=YES

The need for this option was found in the Supermarket application when a check was scanned by an external pin pad. For some reason, this act generated an unlock/lock command, reopening IOPROC to the same state. To prevent the unwarranted screen change, this parm was added.

It must be used as a parm for any DYNAKEY that can take you to a screen that you do not want to be reset if such a lock/unlock were to occur.

This state is reset under either of the following conditions:

1. Any motor key is hit from the main (non-DYNAKEY) keyboard
2. Any DYNAKEY is hit which does not have the HOLD_PROMPT parameter.

**How can I make Dynakey behave differently on different terminals?**

The standard way to implement different dynakey displays on different terminals within a store is to have dynakey files on the controller which are unique for a particular terminal ID. The default DYNAKEY file is named DYNAKEY.000. If you have a terminal which will be designed to behave differently based on its terminal address (ex: Pharmacy terminal), a unique dynakey file can be created with the name, DYNAKEY.xxx (where xxx represents the terminal ID). Normally, just a few changes in dynakey behavior are called for on some of these specialty terminals so the rest of the dynakey file for that terminal would be the same as the default. The store has the burden of creating multiple dynakey files for the different terminal tasks within the store.

There is an alternative method that you may prefer. Using this method, only one DYNAKEY.000 file is created. Within that file are conditional statements that implement different dynakey screens based on variables that have been DEFINED when the CDIFILE was processed for that terminal. Please refer to the QVS web site for documentation (sample CDIFILE). In general, the variables are DEFINED based on either the terminal number (TERM_NUM) or the store number (STORE_NUM). Once those defines have been created, DYNAKEY statements within the DYNAKEY.000 file can be activated based on if, then, else logic. An example follows. This example assumes that the variables used within this example have been set when the CDIFILE was executed by the terminal during terminal initiation.

EXAMPLE:

```
 #if ((MANHATTAN) || (LONG_ISLAND)) then
INLIST=5 INPUTS=G3000,G3001,G3002,G3003,G3004,G3005
#else
INLIST=5 INPUTS=G4000,G4001,G4002,G4003,G4004,G4005
#endif
```

In addition to the variables defined during the CDIFILE execution, the global variables TERM_NUM and STORE_NUM are available for use. The #ifelse statement is also available for use. "If" clauses can be nested.

This feature provides the ability to define DYNAKEY behavior chain-wide for all stores in one DYNAKEY.000 file.

==============================================================================

**SAMPLE DYNAKEY FILE**

This sample file is in use in the field and apparently has been tested with a specific application. There may be some errors and some definitions that are never referenced, but it is as good a place to start as any.

```
FCODE=62 DESC="TAX EXEMPT ORDER"
FCODE=65 DESC="HIT 1ST, VENDOR CPN"
FCODE=66 DESC="HIT 1ST, CMPTR CPN"
FCODE=67 DESC="MDSE RETURN"
FCODE=70 DESC="VOID"
FCODE=71 DESC="TARE #"
FCODE=72 DESC="WEIGHT"
FCODE=73 DESC="CLEAR"
FCODE=74 DESC="PRICE"
FCODE=75 DESC="QTY #"
FCODE=77 DESC="TAX/NON TAX"
FCODE=79 DESC="OVERRIDE"
```

```
FCODE=80 DESC="ENTER"
FCODE=81 DESC="TOTAL"
FCODE=83 DESC="FOOD STAMP ELIGIBLE"
FCODE=84 DESC="REFUND"
FCODE=85 DESC="DEPARTMENT"
FCODE=86 DESC="LOW TAX"
FCODE=87 DESC="HIGH TAX"
FCODE=88 DESC="NON TAX"
FCODE=89 DESC="PREFERRED"
FCODE=90 DESC="CHECK ID# "
FCODE=91 DESC="CASH $"
FCODE=92 DESC="CHECK $"
FCODE=93 DESC="FOOD STAMPS $"
FCODE=94 DESC="DEBIT CARD $"
FCODE=95 DESC="GIFT CERTIFICATE $"
FCODE=96 DESC="CREDIT CARD $"
FCODE=99 DESC="TRAVELERS CHECKS"
FCODE=100 DESC="NO SALE"
FCODE=251 DESC=" "
INLIST=10 INPUTS=G85,G89,G991,G992,G994,G995,G996,G997,G70,G1,G61,G14,G1001,G2
INLIST=11 INPUTS=G75,G52,G83,G87,G86,G88,G5
GROUP=85 DESC="DEPT CODE" DATA=%,85 INPUTS=G851 NEXT=G854
GROUP=851 DESC="KEY DEPT CODE HERE" DATA=85 INPUTS=F251,G853
GROUP=852 DESC="KEY PRICE HERE" DATA=80 NEXT=I10
GROUP=853 DESC="KEY PRICE HERE" DATA=%,80 NEXT=I10
GROUP=854 DESC=" " INPUTS=F251,G853
GROUP=991 DESC="FILM PROCESS(310)" DATA=3,1,0,85,%,80 INPUTS=F251,G852 NEXT=I10
GROUP=992 DESC="1 HOUR PHOTO(319)" DATA=3,1,9,85,%,80 INPUTS=F251,G852
GROUP=993 DESC="MAGAZINES (070)" DATA=7,0,85,%,80 INPUTS=F251,G852
GROUP=994 DESC="NEWSPAPERS (071)" DATA=7,1,85,%,80 INPUTS=F251,G852
GROUP=995 DESC="REG PRESCRIP(200)" DATA=2,0,0,85,%,80 INPUTS=F251,G852
GROUP=996 DESC="STAMPS (PLU 225)" DATA=2,2,5,80
GROUP=997 DESC="MONEY ORDERS>" INPUTS=G9971,G9972
GROUP=9971 DESC="M O SALES (PLU 112)" DATA=1,1,2,80
GROUP=9972 DESC="M O COMM. (PLU 113)" DATA=1,1,3,80
GROUP=1 DESC="LOCK REGISTER" DATA=100,61
GROUP=2 DESC="MODIFER MENU" INPUTS=I11
GROUP=3 DESC="TOTAL" DATA=81 INPUTS=G91,G921,G96,G94,G93,G65,G66,G937,G922,G95,G62
GROUP=4 DESC="OTHER TENDERS>" INPUTS=G937,G922,G95
GROUP=5 DESC="DUP. RECEIPT" DATA=100,9,9,61
GROUP=6 DESC="VOID PREVIOUS ENTRY" DATA=70,80
GROUP=7 DESC="COUPONS>" INPUTS=G65,G66
GROUP=12 DESC="KEY REQ'D INFO HERE" DATA=80
GROUP=13 DESC="ENTER" DATA=80
```

```
GROUP=14 DESC="VERIFY PRICE" DATA=6,61
GROUP=17 DESC="OPERATIONS MANUAL>>" INPUTS=L95,L96,L97,L98
GROUP=70 DESC="NO SALE" DATA=100,80
GROUP=52 DESC="VOID " DATA=70
GROUP=55 DESC="ENTER" DATA=80 INPUTS=I10
GROUP=57 DESC="ENTER AMT/PRESS HERE" DATA=74,9,1,0,2,80
GROUP=61 DESC="SIGN OFF" DATA=61,61
GROUP=62 DESC="TAX EXEMPT ORDER" DATA=%,62
INPUTS=F251,F251,F251,F251,F251,F251,F251,G621 NEXT=G3
GROUP=621 DESC="TAX EXEMPT #" DATA=%,62 NEXT=G3
GROUP=65 DESC="MFR COUPON $" DATA=65,1,9,1,85,%,80 INPUTS=G651,G652,G653,F251,G654
NEXT=G655
GROUP=651 DESC="$.25 COUPON" DATA=65,1,9,1,85,2,5,80 INPUTS=G655
GROUP=652 DESC="$.50 COUPON" DATA=65,1,9,1,85,5,0,80 INPUTS=G655
GROUP=653 DESC="$.75 COUPON" DATA=65,1,9,1,85,7,5,80 INPUTS=G655
GROUP=654 DESC="OTHER COUPON AMOUNT" DATA=65,1,9,1,85,%,80 NEXT=G3
GROUP=655 DESC="TOTAL" DATA=81 INPUTS=G91,G921,G96,G94,G93,G65,G66,G937,G922,G95,G62
GROUP=66 DESC="STORE COUPON" DATA=66 INPUTS=G661
GROUP=661 DESC="KEY CUPN DEPT HERE" DATA=85 INPUTS=F251,G663
GROUP=663 DESC="KEY CUPN AMT HERE" DATA=%,80 NEXT=G655
GROUP=73 DESC="CLEAR" DATA=73
GROUP=75 DESC="QTY#" DATA=%,75 INPUTS=G751,G752,G753,G754 NEXT=G755
GROUP=751 DESC="QTY 24 " DATA=2,4,75 INPUTS=I10
GROUP=752 DESC="QTY 48 " DATA=4,8,75 INPUTS=I10
GROUP=753 DESC="QTY 50 " DATA=5,0,75 INPUTS=I10
GROUP=754 DESC="KEY QTY & PRESS HERE" DATA=75 INPUTS=I10
GROUP=755 DESC="QTY#" INPUTS=I10
GROUP=83 DESC="FOOD STAMP/NON FS" DATA=83
GROUP=84 DESC="REFUND" DATA=84
GROUP=86 DESC="LOW TAX" DATA=86
GROUP=87 DESC="HIGH TAX" DATA=87
GROUP=88 DESC="NON TAX" DATA=88
GROUP=89 DESC="PREFERRED CUST" DATA=89
GROUP=50 DESC="CATAGORY LIST" INPUTS=G500,G501,G502,G503,G504,G505,G506,G507,G508,G509
GROUP=500 DESC="0 LOBBY... " INPUTS=L500,L501,L502,L503,L504,L505,L506,L507,L508,L509
GROUP=501 DESC="1 COSMETICS... " INPUTS=L510,L511,L512,L513,L514,L515,L516,L517,L518,L519
GROUP=502 DESC="2 DRUGS & BABY..." INPUTS=L520,L521,L522,L523,L524,L525,L526,L527,L528,L529
GROUP=503 DESC="3 CAMERAS... " INPUTS=L530,L531,L532,L533,L534,L535,L536,L537,L538,L539
GROUP=504 DESC="4 HOUSEWARES... " INPUTS=L540,L541,L542,L543,L544,L545,L546,L547,L548,L549
GROUP=505 DESC="5 HARDGOODS... " INPUTS=L550,L551,L552,L553,L554,L555,L556,L557,L558,L559
GROUP=506 DESC="6 STATIONERY... " INPUTS=L560,L561,L562,L563,L564,L565,L56 6,L567,L568,L569
GROUP=507 DESC="7 TOYS & SPORT..." INPUTS=L570,L571,L572,L573,L574,L575,L576,L577,L578,L579
GROUP=508 DESC="8 SOFTGOODS... " INPUTS=L580,L581,L582,L583,L584,L585,L586,L587,L588,L589
GROUP=509 DESC="9 SOFTGOODS... " INPUTS=L590,L591,L592,L593,L594,L595,L596,L597,L598,L599
```

```
GROUP=91 DESC="CASH $" DATA=%,91 INPUTS=G911,G912,G913,G914,G915,G916,F251,G917
GROUP=911 DESC="EXACT AMT $ " DATA=+R0,91
GROUP=912 DESC="$ " DATA=+R100,91
GROUP=913 DESC="$ " DATA=+R500,91
GROUP=914 DESC="$ " DATA=+R1000,91
GROUP=915 DESC="$ " DATA=+R2000,91
GROUP=916 DESC="KEY CASH AMT HERE" DATA=91
GROUP=917 DESC="FINALIZE (-) SALE" DATA=0,91
GROUP=92 DESC="CHECKS >" INPUTS=G921,G922
GROUP=921 DESC="CHECKS $" DATA=%,92 INPUTS=F251,G9211,G9212,G9213,G9214
GROUP=9211 DESC="$ " DATA=+0,92
GROUP=9212 DESC="BAL + $10 $ " DATA=+1000,92
GROUP=9213 DESC="BAL + $20 $ " DATA=+2000,92
GROUP=9214 DESC="KEY CHECK AMT HERE" DATA=%,92
GROUP=922 DESC="TRAVELERS CHECK $" DATA=%,99 INPUTS=G9221,G9222,G9223,G9224,G9225
GROUP=9221 DESC="$ 10.00" DATA=1,0,0,0,99
GROUP=9222 DESC="$ 20.00" DATA=2,0,0,0,99
GROUP=9223 DESC="$ 50.00" DATA=5,0,0,0,99
GROUP=9224 DESC="$ 100.00" DATA=1,0,0,0,0,99
GROUP=9225 DESC="KEY CHECK AMT HERE" DATA=99
GROUP=93 DESC="FOOD STAMPS >" DATA=93,81 INPUTS=G938,G936
GROUP=931 DESC="FOOD STMPS $ " DATA=+R100,93
GROUP=9311 DESC="EXACT AMT $" DATA=+0,97
GROUP=9312 DESC="EBT FD STMP $" DATA=+0,98
GROUP=932 DESC="FOOD STMPS $ " DATA=+R500,93
GROUP=9321 DESC="BAL + 5.00 $" DATA=+500,97
GROUP=933 DESC="FOOD STMPS $ " DATA=+R1000,93
GROUP=9331 DESC="BAL + 10.00 $" DATA=+1000,97
GROUP=934 DESC="FOOD STMPS $ " DATA=+R2000,93
GROUP=935 DESC="KEY FDSTMP AMT HERE" DATA=93
GROUP=9351 DESC="KEY EBT CASH AMT " DATA=97
GROUP=9352 DESC="KEY EBT FD STMP AMT" DATA=98
GROUP=936 DESC="EBT FOOD STAMPS >" INPUTS=G9312,F251,F251,F251,G9352
GROUP=937 DESC="EBT CASH $" DATA=%,97 INPUTS=G9311,G9321,G9331,F251,G9351
GROUP=938 DESC="FOOD STAMPS >" INPUTS=G931,G932,G933,G934,G935
GROUP=94 DESC="DEBIT CARD $" DATA=%,94 INPUTS=F251,F251,F251,G941,G942,G943,G944
GROUP=941 DESC="EXACT AMT $ " DATA=+R0,94
GROUP=942 DESC="BAL + $10 $ " DATA=+1000,94
GROUP=943 DESC="BAL + $20 $ " DATA=+2000,94
GROUP=944 DESC="KEY DEBIT AMT HERE" DATA=94
GROUP=96 DESC="CREDIT CARD $" DATA=%,96 INPUTS=F251,F251,G961,G962,F251,G963
GROUP=961 DESC="EXACT AMT $ " DATA=+R0,96
GROUP=962 DESC="KEY CREDIT AMT HERE" DATA=96
GROUP=963 DESC="FINALIZE (-) SALE" DATA=0,96
```

```
GROUP=95 DESC="GIFT CERTIFICATE $" DATA=%,95 INPUTS=G951,G952,G953,G954,G955,G956,G959
GROUP=951 DESC="EXACT AMT $ " DATA=+R0,95
GROUP=952 DESC="GIFT CERT $ 10.00 " DATA=1,0,0,0,95
GROUP=953 DESC="GIFT CERT $ 15.00 " DATA=1,5,0,0,95
GROUP=954 DESC="GIFT CERT $ 20.00 " DATA=2,0,0,0,95
GROUP=955 DESC="GIFT CERT $ 25.00 " DATA=2,5,0,0,95
GROUP=956 DESC="GIFT CERT $ 50.00 " DATA=5,0,0,0,95
GROUP=959 DESC="KEY GFTCERT AMT HERE" DATA=95
GROUP=1001 DESC="SUPERVISOR MENU" INPUTS=G84,G1002,G1003,G1004,G1005,G1007,G1008
GROUP=1002 DESC="CASH PICKUP >>" DATA=4,61
GROUP=1003 DESC="EMPLOYEE DISCOUNT%" DATA=79,%,61 INPUTS=G10031,G10032 NEXT=G1003
GROUP=10031 DESC="13 %" DATA=1,3,61
GROUP=10032 DESC="18 %" DATA=1,8,61
GROUP=1004 DESC="FORCED SIGNOFF" DATA=61,73
GROUP=1005 DESC="RESET PINPAD" DATA=0,78,94 INPUTS=G73
GROUP=1007 DESC="TRANSACTION CANCEL" DATA=70,81
GROUP=1008 DESC="CHECK CASHING >>" DATA=1,61
GROUP=239 DESC="MISC. INCOME >" INPUTS=F251,F251,G2391,F251,G2392 NEXT=G239
GROUP=2391 DESC="CASH $" DATA=%,239,81,91,91 INPUTS=F251,F251,G23911 NEXT=2391
GROUP=23911 DESC="ENTER MISC. INC. AMT" DATA=239,81,91,91
GROUP=2392 DESC="CHECK $" DATA=%,239,81,92,92 INPUTS=F251,F251,F251,F251,G23921
NEXT=G2392
GROUP=23921 DESC="ENTER MISC. INC. AMT" DATA=239,81,92,92
GROUP=801 DESC="OPERATOR NUMBER" DATA=78 INPUTS=F251,G802
GROUP=802 DESC="PASSWORD" DATA=61
GROUP=803 DESC="OPERATOR OVERRIDE" DATA=79,80
GROUP=804 DESC="MANAGER OVERRIDE" DATA=79,%,80 INPUTS=F251,F251,F251,F251,G8041
NEXT=G803
GROUP=8041 DESC="OVERRIDE # REQ'D" DATA=%,80
STATE=1 DESC="****PRESS CLEAR****" INPUTS=G73
STATE=2 DESC="*****SIGNON/OFF*****" INPUTS=G801
STATE=3 DESC="****SIGNON/OFF2****"
STATE=4 DESC="**PASSWORD CHANGE**"
STATE=5 DESC="***SPECIAL SIGNON***" INPUTS=F251,G802
STATE=6 DESC="*OVERRIDE REQUIRED*" INPUTS=G73,F251,G803,F251,G804
STATE=7 DESC="***ACCOUNT NUMBER***" INPUTS=G73,F251,F80
STATE=8 DESC="***ACCOUNT NUMBER***" INPUTS=G73,F251,F80
STATE=9 DESC="DATA ENTRY 1"
STATE=10 DESC="*****ITEM MENU*****" INPUTS=I10
STATE=11 DESC="KEY REQ'D INFO" INPUTS=G73,F251,F251,F251,G12
STATE=13 DESC="DATA ENTRY 3"
STATE=20 DESC="***TENDER CASHING***"INPUTS=G92,F251,F251,F70,F251,F251,F81
STATE=21 DESC="CASHIER LOAN/PICKUP" INPUTS=F91,F251,F251,F70,F251,F251,F81
STATE=22 DESC="*LAST BATCH STATUS*"
```

```
STATE=24 DESC="****VERIFY PRICE****" INPUTS=F251,F251,F251,F80,F251,F251,F70,F81
STATE=33 DESC="ENTER ID" INPUTS=G73,F251,F80
STATE=34 DESC="GET ACCT#"
STATE=35 DESC="VERIFYID"
STATE=36 DESC="INTVERFY"
STATE=37 DESC="DISPMGRK"
STATE=38 DESC="DISPMGR2"
STATE=40 DESC="CRED ACCT"
STATE=41 DESC="READ MICR FROM CHECK"
STATE=42 DESC="CLRENTER" INPUTS=F73,F251,F80
LIST=500 DESC="00 CANDY... " FILE=DEPT00.LST
LIST=501 DESC="01 NUTS, SNACKS... " FILE=DEPT01.LST
LIST=502 DESC="02 OTHER EDIBLES..." FILE=DEPT02.LST
LIST=503 DESC="03 TOBACCO... " FILE=DEPT03.LST
LIST=504 DESC="04 LIQUOR... " FILE=DEPT04.LST
LIST=505 DESC="05 BEER & WINE... " FILE=DEPT05.LST
LIST=506 DESC="06 NON-ALCOHOL... " FILE=DEPT06.LST
LIST=507 DESC="07 READING... " FILE=DEPT07.LST
LIST=508 DESC="08 RECORDS/TAPES..." FILE=DEPT08.LST
LIST=509 DESC="09 SERVICES... " FILE=DEPT09.LST
LIST=510 DESC="10 EYE MAKEUP... " FILE=DEPT10.LST
LIST=511 DESC="11 FACIAL MAKEUP..." FILE=DEPT11.LST
LIST=512 DESC="12 LIP&NAIL CARE..." FILE=DEPT12.LST
LIST=513 DESC="13 WOMENS FRAGRENCE" FILE=DEPT13.LST
LIST=514 DESC="14 WOMENS HAIR PREP" FILE=DEPT14.LST
LIST=515 DESC="15 MENS TOILETRIES." FILE=DEPT15.LST
LIST=516 DESC="16 DEODORANT & ORAL" FILE=DEPT16.LST
LIST=517 DESC="16 COSMETIC ACCESS." FILE=DEPT17.LST
LIST=518 DESC="18 JEWELRY... " FILE=DEPT18.LST
LIST=519 DESC="19 LOTION-SKIN CARE" FILE=DEPT19.LST
LIST=520 DESC="20 PRESCRIPTIONS..." FILE=DEPT20.LST
LIST=521 DESC="21 VITAMINS & TONIC" FILE=DEPT21.LST
LIST=522 DESC="22 COUGH,COLD,EAR.." FILE=DEPT22.LST
LIST=523 DESC="23 ANALGESICS... " FILE=DEPT23.LST
LIST=524 DESC="24 FT CARE,ANTACIDS" FILE=DEPT24.LST
LIST=525 DESC="25 SICKROOM AIDS..." FILE=DEPT25.LST
LIST=526 DESC="26 1ST AID,SKINPREP" FILE=DEPT26.LST
LIST=527 DESC="27 DIET & HLTH FOOD" FILE=DEPT27.LST
LIST=528 DESC="28 BABY NEEDS... " FILE=DEPT28.LST
LIST=529 DESC="29 FEMININE HYGIENE" FILE=DEPT29.LST
LIST=530 DESC="30 CAMERA & EQUIP.." FILE=DEPT30.LST
LIST=531 DESC="31 OTHER CAMERA... " FILE=DEPT31.LST
LIST=532 DESC="32 MISC CAMERA MDSE" FILE=DEPT32.LST
LIST=533 DESC="33 CAMERA DEPT ELEC" FILE=DEPT33.LST
```

```
LIST=534 DESC="34 ELECTRONICS... " FILE=DEPT34.LST
LIST=535 DESC="35 APPLIANCES... " FILE=DEPT35.LST
LIST=536 DESC="36 BATTERIES... " FILE=DEPT36.LST
LIST=537 DESC="37 PERSON CARE APPL" FILE=DEPT37.LST
LIST=538 DESC="38 CLOCKS &FLR CARE" FILE=DEPT38.LST
LIST=539 DESC="39 SEASONAL APPLNCE" FILE=DEPT39.LST
LIST=540 DESC="40 PLASTIC HSWARES." FILE=DEPT40.LST
LIST=541 DESC="41 COOK & BAKE WARE" FILE=DEPT41.LST
LIST=542 DESC="42 DINNER&GLASSWARE" FILE=DEPT42.LST
LIST=543 DESC="43 HOUSEHLD GADGETS" FILE=DEPT43.LST
LIST=544 DESC="44 CLOSET,SHOE CARE" FILE=DEPT44.LST
LIST=545 DESC="45 CLEANING SUPPLYS" FILE=DEPT45.LST
LIST=546 DESC="46 HOUSEHLD CHEMICL" FILE=DEPT46.LST
LIST=547 DESC="47 THERMOS&LUNCH KT" FILE=DEPT47.LST
LIST=548 DESC="48 HSHLD P APER-PLAS" FILE=DEPT48.LST
LIST=549 DESC="49 RESERVED... " FILE=DEPT49.LST
LIST=550 DESC="50 HARDWARE... " FILE=DEPT50.LST
LIST=551 DESC="51 ELECTRICAL... " FILE=DEPT51.LST
LIST=552 DESC="52 PAINT... " FILE=DEPT52.LST
LIST=553 DESC="53 AUTOMOTIVE... " FILE=DEPT53.LST
LIST=554 DESC="54 GARDEN... " FILE=DEPT54.LST
LIST=555 DESC="55 PATIO & BBQ... " FILE=DEPT55.LST
LIST=556 DESC="56 TRIM-A-TREE... " FILE=DEPT56.LST
LIST=557 DESC="57 PET... " FILE=DEPT57.LST
LIST=558 DESC="58 HOME FURNISHINGS" FILE=DEPT58.LST
LIST=559 DESC="59 FURNITURE... " FILE=DEPT59.LST
LIST=560 DESC="60 SCHOOL&ART SUPPL" FILE=DEPT60.LST
LIST=561 DESC="61 STATIONRY&FRAMES" FILE=DEPT61.LST
LIST=562 DESC="62 HOME&OFFICE SUPP" FILE=DEPT62.LST
LIST=563 DESC="63 CARD SHOP HALLMR" FILE=DEPT63.LST
LIST=564 DESC="64 NON HALLMRK SEAS" FILE=DEPT64.LST
LIST=565 DESC="65 SUNGLASSES... " FILE=DEPT65.LST
LIST=566 DESC="66 LUGGAGE&LEATHER " FILE=DEPT66.LST
LIST=567 DESC="67 GIFTWARE... " FILE=DEPT67.LST
LIST=568 DESC="68 CARD SHOP HALLMA" FILE=DEPT68.LST
LIST=569 DESC="69 FUTURE USE... " FILE=DEPT69.LST
LIST=570 DESC="70 HUNTING... " FILE=DEPT70.LST
LIST=571 DESC="71 FISHING&CAMPING." FILE=DEPT71.LST
LIST=572 DESC="72 OTHER SPORTING.." FILE=DEPT72.LST
LIST=573 DESC="73 NEUTER TOYS... " FILE=DEPT73.LST
LIST=574 DESC="74 BOYS TOYS... " FILE=DEPT74.LST
LIST=575 DESC="75 OUTDOOR BLK TOYS" FILE=DEPT75.LST
LIST=576 DESC="76 GIRLS TOYS... " FILE=DEPT76.LST
LIST=577 DESC="77 CRAFTS... " FILE=DEPT77.LST
```

```
LIST=578 DESC="78 RESERVED... " FILE=DEPT78.LST
LIST=579 DESC="79 RESERVED... " FILE=DEPT79.LST
LIST=580 DESC="80 BEDDING&PILLOWS." FILE=DEPT80.LST
LIST=581 DESC="81 KITCHEN&BATH... " FILE=DEPT81.LST
LIST=582 DESC="82 SEWING&KNITTING." FILE=DEPT82.LST
LIST=583 DESC="83 RUGS,MATS & MISC" FILE=DEPT83.LST
LIST=584 DESC="84 SHOES & FOOTWEAR" FILE=DEPT84.LST
LIST=585 DESC="85 LADIES HOSIERY.." FILE=DEPT85.LST
LIST=586 DESC="86 CASUAL HOSIERY.." FILE=DEPT86.LST
LIST=587 DESC="87 LADIES LINGERIE." FILE=DEPT87.LST
LIST=588 DESC="88 BOTTOMS & TOPS.." FILE=DEPT88.LST
LIST=589 DESC="89 OTHER LADIES APP" FILE=DEPT89.LST
LIST=590 DESC="90 GIRL'S APPAREL.." FILE=DEPT90.LST
LIST=591 DESC="91 RESERVED... " FILE=DEPT91.LST
LIST=592 DESC="92 ACCESSORIES... " FILE=DEPT92.LST
LIST=593 DESC="93 BOY'S APPAREL..." FILE=DEPT93.LST
LIST=594 DESC="94 BOY'S UNDERWR..." FILE=DEPT94.LST
LIST=595 DESC="95 MEN'S APPAREL..." FILE=DEPT95.LST
LIST=596 DESC="96 MEN'S BASICS... " FILE=DEPT96.LST
LIST=597 DESC="97 TODDLERS... " FILE=DEPT97.LST
LIST=598 DESC="98 INFANT'S... " FILE=DEPT98.LST
LIST=599 DESC="99 RESERVED... " FILE=DEPT99.LST
```