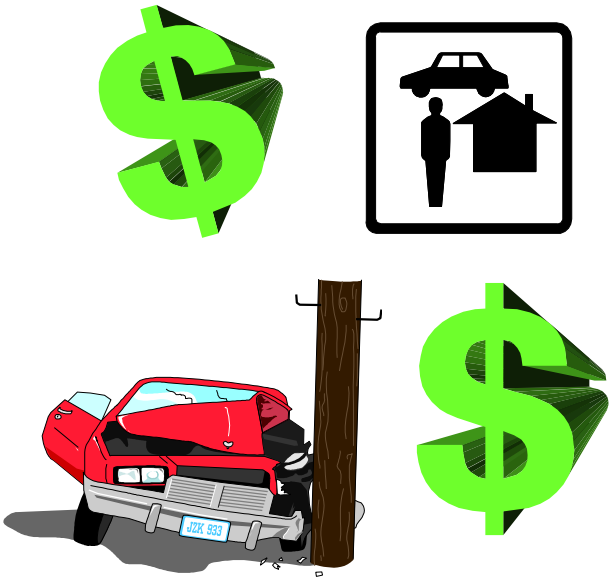
Registry of Motor Vehicles

UMS

Guest Customization Manual Release 6.0

November 2002



T21495-1002

Table of Contents

1 UMS Overview

Proscriptions

This UMS GUEST CUSTOMIZATION MANUAL is provided to assist those users who have a definite requirement to modify or extend the UMS application software. There are five important principles to guide such modifications and/or extensions:

1. While the RMV provides problem determination and correction for the UMS Guest Software, it **cannot** provide such support to user-modified or extended Guest Programs.

2. The UMS System Control and Support Programs named UGZxxxxP **may not** be modified. The only exception is for those modules involved in site definition whose modifications are specified in the Install Manual.

3. **All fields passed to the HOST for whatever purpose must successfully pass their LXTABLE edit** where such an edit is provided. The list on page 72 lists the edits that **must** be employed. The section beginning on page 57 discusses the LXTABLE in detail. Appendix A provides discussion and examples of the LXTABLE use in programs outside of the UMS Control Structure.

4. Prior to making contact with the HOST, the **UGTL transaction must** have executed successfully in the current execution of the CICS region concerned.

5. Contact with the HOST will be effected **only** by means of the UMS System Interface programs supplied by the RMV. Please refer to #2 above.

In addition, please read the convention section beginning on page 9 and the interface section beginning on page 17.

Overview

The Uninsured Motorist System (UMS) was developed to allow insurance companies to notify the RMV when a policy on a vehicle had been cancelled, creating a possible uninsured motorist. The system operates in both the batch and on-line mode. Over time, enhancements have been made to the system to cover far more than insurance policy changes. It has intimate connections with ALARS (Automated Licensing and Renewal

System) such that former distinctions are now blurred. To the same extent, the responsibilities for these systems have changed as well.

The UMS system is built on a client-server basis. The RMV computer is the server, or Host, for data on registrations, licensing, and related issues. Many of the clients or guests are insurance companies or "outside" agencies with their own data processing facilities and needs. UMS provides them with a complete interface and set of application programs to access the RMV data. They are free to make modifications to the UMS Guest programs. They are, however, required to use the UMS Structural and LXTABLE edit facilities so they can maintain a maximum posture of release independence and conform to data edit requirements. The system design places much of the workload on their machine. The UMS design makes simple requests of the Host database system so that the database run-units are short with as few I/Os as possible to give the very best available response for all users.

A client who either modifies the released software or creates its own **assumes full responsibility** for conforming to RMV data Standards.

A dozen CICS and IDMS regions provide the various classes of service required. In all cases, CICS is the teleprocessing and application front-end for the IDMS CVs. The database services are filtered through the Logical Record Facility (LRF) of IDMS. The General User CICS region is the LU6.2 contact point in the Host for "guest" users who have connect privileges during normal business hours.

Three general types of Guest-Host interface are in service. In Type 1, the UMS Guest software resides in the Guest's mainframe and uses LU6.2 services to reach the RMV computer. The RMV host software does some validations, security checking, and the IDMS database I/O. The Type 2 interface is used by several official agencies. In this, the Guest software runs in the RMV CPU and the LU6.2 facility is replaced by an interface program. The Type 3 interface is similar to Type 1, but either the machine is not an IBM or the user (guest) has heavily modified the "guest" software. This is a case of a "black box" communicating via LU6.2. We have no particular responsibility for problems with this type of "guest" software.

For a more **detailed description** of functions or for **error message information**, consult the UMS Technical Manual for this release level.

The overall design objectives are to use single function, independent modules and segregate the front-end (CICS or TP) and back-end (IDMS or database) processing. This has been achieved through several subordinate objectives or techniques. The first UMS design objective is to place as much of the application work as possible in the Guest software. The second objective is to drive the Guest and Host software systems with an application control structure. This structure, called Control/Dispatch (CD), supports the third objective, which is that development and maintenance time for applications be minimized. CICS provides all the needed control facilities, but they must be hard-coded into the programs. The programs become larger in size and complexity, and changes to

the processing structure cause changes to a number of modules. It is also difficult to run new and old versions of programs in parallel.

The UMS CD system is table driven so that control flow changes are only applied in one table. Common services, such as screen mapping, data movement, field editing, and error message handling are all directed by the CD system and occur in system modules outside of the application program. The chapters on Control Dispatch, UMS Program Control Tables, and LXTABLE Processing describe these control mechanisms in detail. Many utility routines are provided so that the application program can concentrate on the specific business function and just call for basic "services" such as date routines. Components are intentionally isolated so that debugging and error resolution is simplified and new functions can be added without changing or disturbing the existing ones.

The fourth objective is function-reuse. This applies particularly to the Host side functions that provide the database services. If a new requirement needs half of the data provided by Host-function "X" and additional data not provided anywhere else, a new Host function will be written to get the "additional data." In this way, if there is a problem with the function "X" data, there is still only one place to look for the problem. There would be many possibilities if each new data requirement was met with a new function providing only a specific group of fields. Such a proliferation would be expensive in terms of development and a nearly insupportable nightmare in terms of maintenance and problem resolution.

The fifth objective is to make the interface to the DBMS narrow and well defined as possible. This ensures that the impact of changing the DBMS would be minimal to the application systems. As a corollary, all programs must use the application "common type services," which operate through specific interfaces. This aids in development and, in case of error or needed change, there is only one place to apply the correction and then all programs will be using the new code. This principle is extended further in cases such as license plate number edits where the edit rules set is elaborate and subject to sudden change. The edit code is in a separate routine accessible to all applications but it is driven by a set of rules that is downloaded from the Host at Guest start-up time. In this way, the rules could be changed daily with no visibility to the users at all. This design eliminates the need to manually distribute zaps or complete new copies of the rules tables and ensures that all users are totally and painlessly up to date.

2 Control - Dispatch

System Overview

The UMS Guest System requires initialization once the CICS region's startup is complete. The UGTL transaction invokes UGZ0015P to perform the various operations. This process is discussed in more detail in a subsequent chapter. If this process is not completed, UMS will abend with a code of “UGTL” or “UGTH.”

The transaction “UG03” is the normal entry to the UMS Guest Processing. It invokes UGZ0001P, the Guest Side Control Program. This routine controls the processing thread for the transaction. Application programs that are invoked usually return to this routine unless they abend. The control program examines the control structures and passes the thread to the next program or function for the task. The focus of control is the system portion of the common area and the PCTEG entry in UGZ0002P for the transaction in flight. The PCTEG is discussed in detail later. It contains a list of programs to be invoked in turn to do the work of the transaction. If the automatic screen mapping, demapping, and editing feature is to be used, the name of the (LX) control table is given. If a special clean-up program is needed, its name is given. Flags show the ability to do updates, scrolling, and re-scrolling as well as the ability to communicate with the Host System. Some additional features are discussed in the detailed treatment. The Guest or Host system may also be invoked by external programs that start the transaction UG05 (that points to UGZ0008P). UGZ0008P can be called directly by link or xctl.

At transaction initialization time, the data from the corresponding PCTEG entry is placed into the system portion of the Guest Common Area. The structure of the Guest (and Host) Common Areas are briefly covered in the following section of this chapter. The control program and the application both refer to the Guest Common Area as the transaction progresses. The application only moves data elements. The change of a program name is transparent and only requires a change in the PCTEG entry for that transaction. The order of the program names in the PCTEG is function dependent and the application can vary the name moved to the 'next program' field depending on run-time conditions.

The first named program in the PCTEG list is the first one called by the control program once mapping is complete. A program can be first for only one transaction (and its test version, 1st character = 'T'). When a screen of data is to be processed, the designated LX map table is loaded and the LX map program completes its work before that first application program is invoked. If the map module detects errors it can cycle through error-message issue and re-try the mapping until the input is error free. The data from the

screen is placed into the dsect in the common area before the application gains control. The program can edit the data further and issue error messages of its own to the screen.

A number of service routines are available to do common functions such as upper-case conversion and date conversions. These are described in Appendix G of the *UMS Programmer's Manual*. Calls to the Host side for database services can be made for retrieval and/or update. Control can be returned to the caller or a designated program. Each time the thread returns control to CICS (except for logoff), the control program sets the next transid to UG04. This id shows a continuing transaction to insure initialization that does not negate work already done. If access to the Host System is indicated a Host Interface Area block of storage is acquired and initialized by the Control Program. When application work is complete, the Control Program invokes the LX-Table feature to move the data to the screen. As is described in the LX-Table processing chapter, exit routines may be summoned during screen processing (in or out) to perform special editing. These routines can generate error messages.

Error messages are designated by a numeric value in the Common Storage Area. This value is used in conjunction with UGZ0004P, the Error Message Dictionary, to retrieve the actual text and place it in the output map. When the thread comes back to the Control Program as completed, the designated clean-up program , normally UGZ0005P, issues the write to the terminal and makes any required final "adjustments."

Each UMS screen normally has an LX table, guest input program, one or more host programs, and a guest output program. When the user enters data onto the screen, the LX table processor uses the LX table for that screen and edits the data. Next, the guest input program receives control to format the guest-to-host blocks (refer to the Guest-to-Host Blocks section) and performs any additional editing. The guest input program passes control to the host side by using an internal function code to indicate which host programs to invoke (refer to the Internal Function Codes section). The host programs perform database access and formats the needed data into the host-to-guest blocks. The guest output program receives the host-to-guest blocks, uses this data to make decisions about screen highlighting, and performs other miscellaneous tasks. The last programs to receive control are the LX table processor and the Clean up Program, UGZ0005P. These programs use the screen's LX table to map the data from the General Storage Area (GSA) to the screens, and sends the screen back to the user.

General Overview of UMS-Guest Control/ Dispatch

Control/Dispatch (often abbreviated CD) is the name given to the various service level functions provided for the UMS guest application programs. There are two purposes to CD. The first is to provide a level of standardization to functions common to multiple application areas. The second is to perform functions deemed overly complex for the typical application module.

General areas of functionality:

A. Screen/CRT/CICS - Map management
B. Memory management
C. Function validation
D. Host interface
E. Table services

Each of these areas is a component of CD, and may exist as a unique module, or as a component of a multi-purpose module(s).

Main Control

All UMS transactions are "wired" to the same PCT entry. UMS requires at least 2 transactions to run. One (typically UG03) is referred to as the initialization transaction. Another (typically UG04) is referred to as the default run transaction. When the main control program gets control, it determines if this is an initialization call by checking the transaction name against the initialization transaction name. If the initialization transaction is found, the common area is cleared, a flag indicating initialization is set, the current map-name is set to the logon map, and control is transferred to output services. Note that when the user comes back through, the logon process is treated in a manner essentially the same as any other function, except that the user is required to complete logon before a function change is allowed.

UMS supports three mechanisms of saving COMMAREA (Application-high core, VSAM and CICS-high core). The main control module insures that the current COMMAREA image is in an area located below the line.

CD determines if the user has changed the function-code. If so, the new function code is validated. If it is valid and all required modules are present, the internal data is changed to cause the selected function to be dispatched. If an error is detected, output services are invoked to send the message to the user.

If no function is currently selected, output services are invoked to inform the user.

If a function key has been pressed, the meaning of the key is checked for validity in the current environment. If it is invalid, output services are invoked to send the error message to the user. Otherwise, the appropriate service is invoked.

The only remaining action is to dispatch the application. Most applications use table driven mapping/ demapping (LXTBL) services, but a few do not. Those that do not use LXTBL services are transferred directly. Those that use the LXTBL, require services to be dispatched before the application.

Host Interface

When an application determines that it must obtain data from the host, it builds the application portion of a host parameter block and then transfers to the host interface module defined. The interface module checks for the host being active, completes the control portion of the host parameter block and transmits the data to the host. When the response is received, the host interface module transfers back to the requesting application. Note that if the host is not active, or if a fatal error is detected on the host side, the host interface program directly invokes output services to post an error message.

Output Services

Output services has four activities to perform:

1. Format the common map header (date, time, etc)
2. Look up any message code in the message dictionary and place the text in the map
3. Setup for saving the COMMAREA according to the site option for COMMAREA location
4. Determine the next transaction code (specified for the executing function, or the default run transaction)

Once these activities are complete, the map is sent.

Mapping/Demapping Services

Most applications use these services for processing maps. These services provide table driven transfer and editing of fields between the COMMAREA and the map buffer. Some fairly sophisticated cross-field edits and Host Table edits are available. If errors are detected, they result in the direct invocation of output services. Any application which uses these services for demapping also uses them for mapping. Transferring to these services occurs immediately prior to and after application (before output services).

Secondary Session Services

This mechanism allows the user to temporarily leave the current function, perform another session, and return to the original. CD treats the secondary session as a toggled entity. If the second session is not active, the request must be made to activate; if it is active, the request must be made to terminate. Activation consists of saving the COMMAREA and current screen (via 3270- READBUF) in a temporary storage queue and going through normal dispatch. Termination consists of restoring the COMMAREA screen and going through normal output services.

Control/ Dispatch Conventions

This is a general set of guidelines, or rules, which must be followed in UMS applications in order to maintain the proper interface with the control/dispatch mechanism.

1. Module names are managed through the PCTE (G & H) entries in both the Guest and Host systems. All reference to specific modules in the applications is via cells in the common area loaded by control-dispatch. No application is to have any reference to specific module names.

2. Guest function names are managed through the PCTE entries. No reference to specific function names (except for internal-only names) is to be present in any set of application modules.

3. Host function (block-type) codes follow the same conventions as guest function names with respect to applications running on the host side. On the guest side, applications calling host functions must be cognizant of the appropriate host name, and seed it into the host interface area.

4. In cases where a guest function is responsible for the output side of a particular duplicate resolution, the parallel host function must be included in the appropriate guest PCTE entry. With this exception, there is no parallelism between host and guest PCTE entries.

5. All maps must be generated using the standard UMS map header macro. This results in all the UMS maps being identical in structure throughout the message area. The structure forces one map per mapset. Further, all transmission and receipt of terminal data is handled by the control-dispatch mechanism. This means that control-dispatch will be responsible for providing an application with an input-map when the application is invoked, and sending an output map when the application exits to the control functions. Control dispatch also provides an area in which the map variables may be placed by the application in concert with cells for map-name, cursor position, message-code, message-text and length of variable-data. This information allows proper map return.

Additionally, the LXTBL option provides a mechanism by which the control dispatch mechanism relates map-data, map-name and common-area fields automatically without specific application intervention. This feature provides for raw field editing, support of basic field types including character, numeric, date, internal zoned-decimal, internal packed-decimal, internal binary, and internal compressed-date. Further, operands are included on the definition macros to provide for a wide range of field content editing and cross-field editing.

6. Control-dispatch is responsible for memory management. Application programs may not contain GETMAIN/FREEMAIN (CICS or MVS) requests.

   On the guest side, Control dispatch saves and restores the common area across each pseudo-conversational interaction. Hooks exist on the guest side for the acquisition of additional work area for specific functions which are not retained across pseudo-conversational interactions. Such acquisition will be implemented when a definitive need arises.

7. Guest application programs for UMS are named in the PCTEG entries, and are expected to be concise units aimed at specific business functions. Broad based functionality is construed as being of a utility nature (such as the LXTBL mechanism, date-conversion, address-editing, common field-editing, etc) and being a function-dispatch service. Any services of this nature not currently meeting these general needs will be provided as requested.

   The general idea is that the application set defined for a given business function will consist of 1 or more modules. The modules will be viewed as being between the processing of terminal-input and a host dialogue, followed by 1 (or possibly more) modules viewed as being between the receipt of a host response and terminal-output. In practice, it is intended to be possible (when testing modifications to a guest function) to run both old and new functions in the same region by having one PCTE pointing at the old, and another at the new. This clearly demands a narrow focus of application, with modules serving one business function only.

   The functional-independence of the module structure is required not only to maintain these abilities, but also to insure that there is some reasonable potential for guest users to migrate the functionality of this code to their existing systems.

8. The same general set of criteria applies to the host. The host is viewed by the guest as a data-server that operates on a quick in and out basis. The host treats each guest interaction as a unique interaction, with no knowledge of any previous interaction. The host common-area is not retained across interactions with the guest.

There is no restriction against the guest going to the host more than once for a given transaction. The intent is to push all work except raw data-service onto the guest. Thus, all possible work is done at the guest side, and is not to be repeated at the host side.

9. The UMS control/dispatch contains mechanisms for interface with foreign (non-UMS) applications, such as ALAR, SYSM, etc as well as any others that may be running on the guest.

   The specification of a business function as being external (foreign) occurs in the guest PCTE entry, where the XFER= option must be specified with a value of YES or DATA. Control may be passed to an external program or to another Task. The LXTABLE discussion covers this feature in detail.

10. UMS control-dispatch is able to accept control from foreign applications in the same fashion as it can initiate them. The following are available to foreign applications:

    A. START the external initiation transaction (currently UG05), with or without the 23-byte data area mentioned above.
    B. XCTL to the external initiation module (currently UGZ0008P), with or without the 23-byte data area mentioned above.

    If no data area is passed, the effect is exactly the same as if a terminal level initiation of the guest occurred.

    When a data area is passed, control-dispatch attempts to sign the user on to UMS using the ID/PASSWORD fields in the data-area. If this fails, the effect is the same as if no data had been passed.

    If the signon succeeds, the passed entry-reason code is checked for being a code which could have been developed from keyboard entry. If it is not, it is forced to the code for ENTER.

    The entry is then sent through function-dispatch exactly as if it had come from a keyboard. Generally, all functionality which would have been present from the UMS function selection screen is supported. The only current exception to this is that if the supplied entry-reason code references F4 or F9, any rescroll data present for the device is lost.

11. Scroll applications are expected to use the specified scroll-area for their functionality, and thus not interfere with applications that might have exited to them for duplicate resolution.

12. Each guest business function is expected to maintain its own view of the structure of the guest common work-area. There is no supported concept of a global view

of this area, and no assurance is thus present that all (or any) other applications will follow the convention used by one particular application for this area. This approach is essential for functional independence as well as the ability to test functions on a modular basis. The same general concept applies to the host common work-area.

Note that this discussion applies to areas described as application work areas, and NOT to areas described as owned by control-dispatch.

13. Clearly, the intent is that guest applications which specify a host interface area consider this area to be an area provided in addition to the work-area at the end of guest-common. Applications are expected to move data directly to and from this area, without unnecessary buffering through common. It has clearly been stated that if required, an ability to reference LXTBL processing to the host interface area will be provided, as well as control-dispatch table-driven maintenance of the host interface area. When the guest initiates some action, the host may take the view that the guest's host-interface area is present in the specified section which is in the middle of the host common. The actual amount of data moving in each direction is driven by the detail length fields on both the host and guest. It is important for performance reasons that the various host interface area formats be constructed such that the minimum amount of space be used. Under no circumstances should the application take the arbitrary attitude of always sending the maximum length.

14. Host applications should be constructed such that the minimum data needed for a given guest activity is returned to the guest via the host interface area. The passing of data elements (or data records) not needed for the current guest activity is not to occur. In a similar manner, the guest sends only minimum data to the host. For a typical retrieval operation, this would be perhaps a key and key-type. For a typical update operation, this would be surrogate(s) and only the new (or changed) data elements. In some critical update operations, also a significant data element (which would be examined by the host for change, inhibiting the operation if an intervening change occurred) might also be passed.

15. There is clear intent to push activity to the guest, as well as to control activity on the host. Host functions are intended to be data activities limited in scope and with necessary relationships to each other. This serves as a natural choke on the extent to which the various guests can load the host. In terms of the current system, for example, the points at which RS, LP, and RN say “READ LIMIT EXCEEDED” (or equivalent language) require a host return to the guest. In a similar fashion, the design of the guest to host requests for code being written must be such that high host loads are avoided. Multiple guest to host interactions are preferable.

16. Message text for output should be through the message-code, message dictionary facility. In defining the messages, the alias feature should be used whenever possible.

17. The application work area in Guest Common Area is 2290 bytes in length. The application work area in Host Common Area is 512 bytes in length, it is preceded by the 2048 byte host interface area.

18. The general relationship between function-dispatch and associated applications is an XCTL relationship. In all cases, the Common Area (host or guest) is passed, using as a length the half-word length cell contained in the appropriate common-area. Applications MUST NOT have a hard-coded common length. The relationship between various modules described in a given PCTE entry may be XCTL or LINK (in either case passing the Common Area as above). The LINK relationship is not desirable, and if used should not go down but one level.

19. Modules (as opposed to functions) are not to be shared between UMS and ALAR. This is for several reasons including (but not limited to) a requirement for not having to test both systems in parallel during modification phases, differences in coding conventions/structures and significant differences in internal security structure.

20. UMS contains a call-by-address mechanism which maintains re-entrance on the called modules. This requires coding techniques which have not been applied to call-by-address modules within ALAR, and that allow for CICS releases which dispatch applications under multiple TCB's thus requiring true reentrancy (not quasi-reentrancy). As guests may be (and often are) running under CICS releases well in advance of those at RMV, these techniques (and control-dispatch owning of the involved modules) is a requirement for this kind of structure.

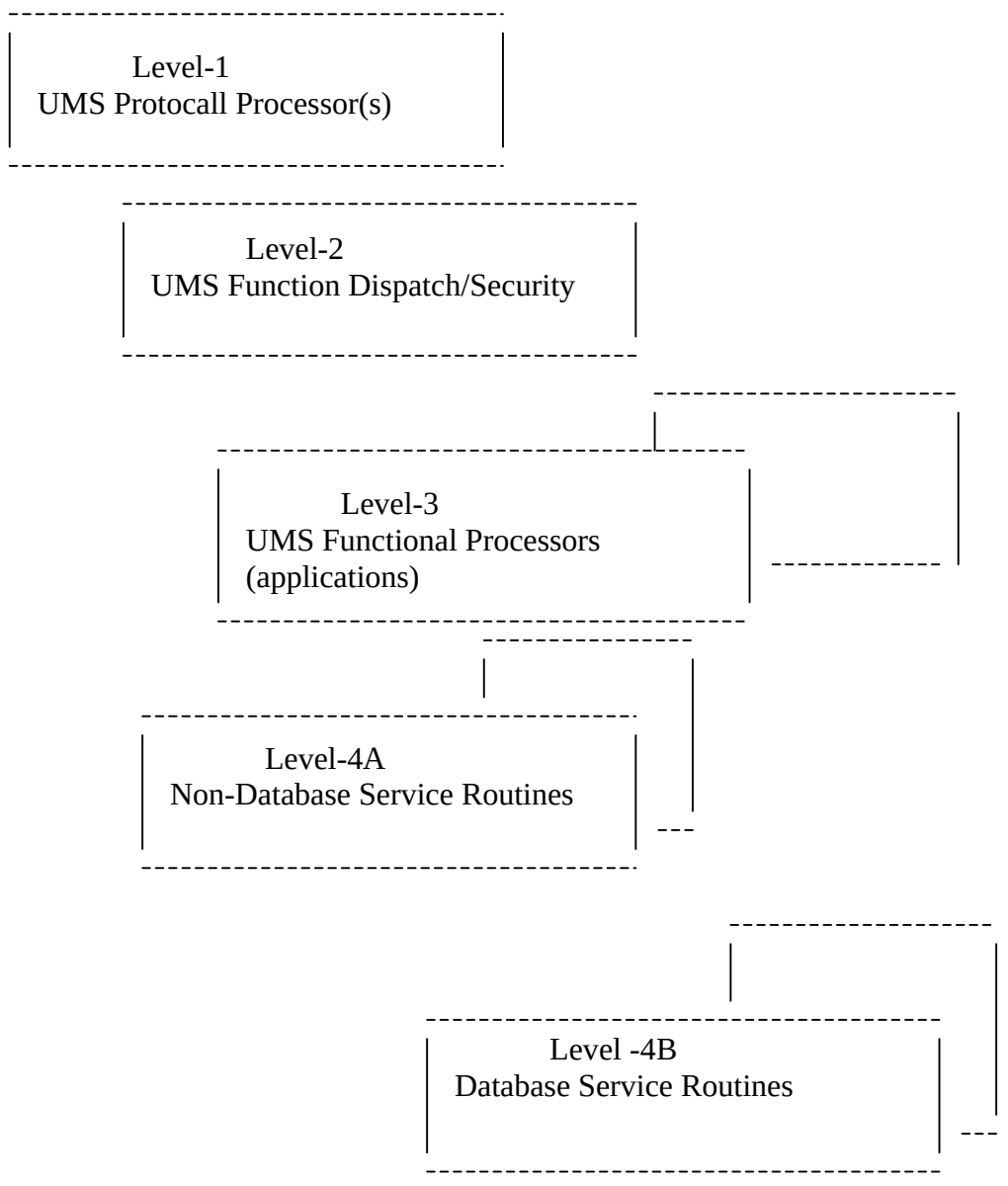
21. The use of most of the Program Function Keys is standardized throughout the system. Deviations from the standard must be approved in advance to avoid adverse impact on the rest of the system and its users. The uses are listed on page 19.

At this point it would be helpful to read through the COBOL field names and the assembler comments on the Guest Common Area field-layouts in Appendix F of the *UMS Programmer's Manual* and review the Guest Common Area Structure table beginning on page 23.

General Structure Specification

Host Side, "UMS" Interface

The host side of this product consists of a set of modules which can be viewed as existing in 5 logical levels, but in practice run in a 2 level structure. The following diagram demonstrates the logical level structure, as best as can be represented in two dimensions:



Note that in the logical view, each level communicates down one level in a bidirectional manner. Also, note that levels 4A and 4B should be viewed as being at the same logical hierarchy, but so defined as to maintain a strict isolation between database and non-database functionality.

This structure is logically correct, but introduces a series of relationships which do not provide for the best utilization of resource under a CICS monitor. For this reason, the logically separate levels (1, 2, & 3) are really at the same physical level in the actual implementation. Each passes off laterally to its successor, and the last level 3 sequence invoked passes off to the completion phase of level 2, which in turn passes off to the return phase of level 1.

Guest Side Conventions

The general structure of the guest-side processes for UMS is similar to the existing ALAR structure in that there is a control-supervisor structure which serves to provide terminal services, a memory (common) area, interpretation of screen function-code, etc. A set of service functions is also provided. The control-dispatch software is referred to as "CD."

The general flow of events is that CD will be entered as a result of terminal input and will ascertain if an active process is in progress for the transaction. If not, the active process will be set to the signon process. In either case, a common-area will be acquired (of the proper length for the process), and seeded with a value-set. The value-set will be composed of values saved from prior invocations and information concerning the current invocation.

CD will transfer to the specified application process via XCTL, passing a commarea. This is the general pattern of operations throughout. CD seeds the commarea with a number of module names including that of a cleanup module to which control must pass when the application is complete. This process will issue the required transmission to the terminal.

CD passes the first application process a set of flags concerning the reason for entry. The potential reasons include (1) entry due to new function-code, (2) reentry due to enter, (3) reentry due to F7/8, (4) entry due to F4, (5) reentry after host-interface, etc. It is important to note that there is a high emphasis on small modules which use XCTL to pass off for a next logical phase, as opposed to large single modules or many level deep communication via LINK. The host interface requires access via XCTL.

The guest processing pattern is driven from a process control table. This table relates external function-codes with a set of program-names, specifies function key functionality/restrictions, defines common-area length, etc.

Because the intent is to provide code to the insurance industry which has reasonable potential for customization and device migration, it is important that the front-end function of input-map field fetching be separated from field processing. The guest common area structure is given in detail in another chapter. The structure provides for a scroll (browse) surrogate table and page table. Two bytes are provided for indicating information relative to the format of this table. We intend that a relationship between the process-control-table and the current format of these two bytes indicate if a current F4 type transfer is valid. Further, the F4 target is to be provided with information about its source based on this input. We have no firm position on if the meaning of these two bytes should algorithmically define the table, or if they should be a code defining the table.

A number of cells are provided for GETMAIN areas, including map-in (provided by CD), map-out, host-interface. Due to the requirement of complete input processing first, the map-out area could overlay the map-in area. Further, if this were desirable, the maximum map-out length for the process could be included in the process-control-table entry for the function as obtained by CD. The CD save-commarea function is used during the host interface activity.

Host Side Conventions

A major component of this development effort is the isolation of the various components of the system from each other. The major components relative to the HOST side are:

1. Protocol
2. Functional Control
3. Primary Security
4. Secondary Security
5. Primary Application Processes
6. Applications Service Routines
7. Database Service Routines

In order to maximize performance, every effort should be made to devise application modules as short, functional processes that complete their designated activity and XCTL to the next required functional process (or back to the cleanup process specified by function dispatch if the application activity is complete). When choices must be made relative to size, emphasis should be first on reducing working-storage and secondly on reducing procedural sizes. Clearly, coding conventions that assist in this process should be utilized. For example, switches and flags should be 1 byte alpha-numeric items and compared to (or set to) literals as opposed to data-items.

The goal of component isolation is best achieved in our current environment in the area of database access. The structure of this product implies that the database is sufficiently removed from the applications that the access method might be changeable without

application change. In turn, this means that database functionality would be served by a set of service modules which would accept function-codes (and involved data), and perform actions (returning status information and/or data). Thus, in the current environment, all IDMS verbs and interactions exist at the 4B (database service) level only. The parameter and data-record passing with the higher level(s) is done with standard CICS and COBOL mechanisms, meaning that the higher level modules are not compiled with the IDMS preprocessor and that standard copy members exist for the passage of data between the 4B and other levels.

In a similar fashion, most of the security processing is completely removed from the applications. The only extent to which the applications are involved with security concerns cases where the security determination is based on field content. For these situations, the application needs to be aware of the existence of a security constraint concerning a given field within the application. The responsibility of the application (when retrieving the involved data) is to call the security assist routine passing the address of the field(s) and an identifying function-code. The security assist routine will pass back a return-code from a set such as the following:

00 No security constraint implied.
01 User cannot see field. Security has overlaid field data with "nonexistent" value for passing to user.
02 User cannot see involved record. Treat identical to record-not-found.
03 User process to be abended. Security has seeded return area with appropriate error-code. XCTL to cleanup process.

It is important to note that when the applications traverse down levels using LINK (or CALL if an assist routine is involved), the error-code field in the common-area text body needs to be evaluated for a non-zero content. If this circumstance occurs, the invoked level must immediately return up 1 level. Since each level will contain such checking (similar to a return-code test) the effect will be that as soon as the error-code becomes non-zero, the cleanup process will be invoked.

Inter-Process Communication, "UMS" Interface

In the traditional fashion, a common area will be used to maintain the memory items necessary for any invocation of these processes. This common area will have several differences from the one in use for the existing ALAR Supervisor. The most glaring of these differences is that the common area will NOT be preserved across invocations from the same end node.

In concept, the UMS Protocol Processor will acquire the common area, initialize certain cells and pass the area to the Function Dispatch Processor. In turn, the area will be passed from process to process until the cycle is complete. The detail structure of the common area is given later. Its general structure is as follows:

Area 1: Protocol Control Area:
This area is reserved for use by the protocol processor.

Area 2: Module Control Name Area:
This area is used for the storage of module-names that are required in the course of the various processes. These names are placed here at transaction start-up time by the UMS-GUEST Control Program. The control program extracts these names from the Guest Program Control Table (PCTEG) based on the transaction code entered. The intention is that no program manipulate hard-coded program name literals. The literals will only be in the table so that a change in the table will be completely effective throughout the entire run-time system.

Area 3: Assist Routine/Table Addresses:
This area is used for the storage of the entry-point addresses of functional assistance modules which are commonly used and whose residence is insured prior to invocation of any application process. Typical examples of entries here are date-conversion routine, address conversion routine, specialized move routine for structuring return messages, etc.

Area 4: Post-Function Security Data:
This area is used to hold data required for security functions which cannot be evaluated prior to function initiation. The design of the UMS processing is that the function-dispatch level is cognizant of any items which the existing ALAR software would classify as subfunctions and interrogates security prior to entering applications on this level. However, this information is used for the resolution of data dependent security items.

Area 5: Interchange Text Body:
This area holds the message from/to the guest. The input-side protocol process seeds this area with the input text, and clears all cells used to control output/error flow. The applications (using the provided assist routines) build the response (and error-codes) in this area for eventual return to the guest. The defined size of this area will limit the maximum length response possible for a given message.

Area 6: Data Services Work Area:
This area is a generalized area used for both communications with the service routines (database and non-database), as well as a work-area for the invoked application programs. By convention, the service routines will use the area from the top down, and the applications will use the area from the bottom up.

Program Function Key Standards

The UMS System contains capabilities programmed for the function keys. Throughout the UMS Subsystem, some function keys can be used to move back and forth between primary and secondary sessions and functions, while others facilitate simpler operations, such as scrolling the screen data.

The following is an explanation of each function key's use within the system. Please note that this is a general list for the entire system, and not all function keys are available to each function. The individual documentation for each function will indicate the function keys available.

F1 The F1 key will return the user to the signon screen. However, if the user is currently positioned at the signon screen, the use of the F1 key will return the user to CICS.

F2 The F2 key moves the user to the UMS Menu screen. Use of this key is equivalent to entering UMM in the function field and pressing the “Enter” key.

F3 The F3 key can be used only in a primary session with screens containing a built-in reference list. The user must request a function change and move the cursor to a selectable item on the reference list before invoking the limited secondary session via F3.

F4 The F4 key is used to select a record from a scroll screen and switch to a secondary function. The user must enter the desired function and move the cursor to the desired record before invoking this function.

F5 The F5 key is not currently used in UMS.

F6 The F6 key is used to invoke the “Screen Hop” facility, a method of carrying information to another inquiry screen after a change of function code.

NOTE: As this procedure is unique and fairly involved, the user should consult the separate documentation on the "Screen Hop" facility for a complete explanation.

F7 The F7 key is used to page backwards on a scroll screen. This key is also used on screens which process more records than can fit on one screen.

NOTE: Although UMS allows paging forward through an entire set of screen data, backward scrolling permits a maximum of eleven (11) pages.

F8 The F8 key is used to page forward on a scroll screen. This key is also used on screens which process more records than can fit on one screen.

F9 The F9 key is used after UMS has automatically invoked a scroll screen due to the specification of a duplicate key by the user. The cursor is moved to the desired line item and F9 is pressed to return to the original function.

F10 The F10 key is not currently used in UMS.

F11 The F11 key is used to reinvoke the last scroll function at the exact point the user left it to invoke the current function. The user must enter the appropriate function code prior to pressing F11.

F12 The F12 key is used in functions to update the database with the values shown on the screen.

Special Session Definitions

Scroll Screens

Some screens are classified as scroll screens. This implies that they present a list of records whose keys are in some way similar (or identical). The screens may be entered by direct user entry of the function code, or automatically by the programs if the programs determine that the user entry is insufficient to decide between similar records stored on the database. The idea is that the user will, by means of cursor positioning and function keys, make the final determination of the desired record.

Most Scroll screens support paging forward and backward, as well as paging by partial pages. The actual selection of the desired record is accomplished by an internal "Browse Table" mechanism.

Reference Lists

Some screens build a "Reference List" to provide additional information about records on the screen. This list is accessed and used in a manner similar to the Scroll Screen/Browse Table mechanism, but is intended for brief examination of supportive detail rather than as a general processing path.

Primary and Secondary Sessions

UMS allows the user to maintain both the normal (or Primary) session and a Secondary session. When the secondary session is invoked, the entire environment which describes the primary session is preserved and restored on return. The intent is that the secondary session can be used for detail reference which may be required to continue the activity being accomplished in the primary session, without the necessity to back out of the detail entered in the primary, and later reenter it. Generally, any inquiry function can be accomplished in either session, but update functions are restricted to the primary session.

At any time, the user can determine **which session is active** by examining the date in the top left corner of the screen.

- If the separators are the "/" character, the session is primary.
- If the separators are the "-" character, the session is secondary.

The user may directly invoke a secondary session which allows all inquiry functions, or may cursor select a limited secondary session which allows only one inquiry function.

Guest-Host Control Blocks

The guest/host (guest-to-host and host-to-guest) blocks are used as communication areas between the guest software at the insurance companies, and the host software at the RMV. Guest-to-host blocks contain data to be sent from the guest to the host side, and host-to-guest blocks contain data to be sent from the host to the guest side.

These blocks are defined in the host and guest programs, and the host-to-guest block for a screen is usually defined in the **LX** table. An **LX** table is a macro that performs editing and mapping for a screen. If changes are made to the host-to-guest block, then the SE must make the changes to the applicable LX table also.

NOTE: The UMS screen "UPH" will be used as an example throughout this description.

The guest-to-host blocks contain key data fields and information. The key data fields are moved from the applicable UMS screen, that are necessary to perform the database retrieval on the host side. The other data in this block consists of data retrieved from the database and data used for updating screens. The guest-to-host block for each UMS screen is different because all screens have unique key data fields and some are inquiry only while others are updatable. Guest-to-host blocks vary in length, but each has 62 bytes at the top that are "reserved" for specific functions.

- The first 40 bytes of the 62 contain routing data, security data and the four-byte internal function code for that screen.

- The next 22 bytes of the 62 contain error information, various indicators and other information, or may be used for duplicate key logic.

For “UPH,” these 22 bytes contain key data for the screen so that if duplicates are encountered during host data base retrieval, duplicate processing may be based on the contents of the 22 bytes.

Interface modules are invoked when transferring from the guest side to the host side and vice-versa. These interface modules use the first 62 bytes of the guest-to-host blocks for their processing (routing and duplicate key logic). The interface modules send the 62 bytes, along with the rest of the block, to the applicable host program in its common area.

Each guest input program moves the length of its guest-to-host block to the common area before transferring control to the host side. This length tells the interface modules the length of the applicable guest-to-host block being passed. The length is determined by subtracting 40 bytes from the total length of the guest-to-host block. The 40 bytes subtracted are part of the 62 reserved bytes at the beginning of the block.

The “UPH” screen's guest-to-host block consists of 40 bytes used for routing, security, and the internal function code, followed by 22 bytes used for duplicate key logic, followed by the rest of the block. The total length of this guest-to-host block is 214 bytes, so the response length is 174 bytes:

214 bytes - 40 bytes = 174 bytes.

When National Driver Registry (NDR) was implemented, the “UPH” screen's 22 bytes, reserved for duplicate key logic, were not enough to hold the 25 byte out-of-state license plus the 2 byte state code (27 bytes total). Five bytes were added to the block, immediately following the 22 bytes, to hold the full 27 bytes needed for duplicate key logic.

Once control has been sent to the host side, the applicable host program saves the guest-to-host block received by moving it from one area of the common area to another. The applicable host program is either the input that performs database retrieval, or the output that performs database updates. The area that the guest-to-host block had previously occupied is initialized to low-values and will be filled with the host-to-guest block to be sent back to the guest output program. The host-to-guest block will contain data moved from the guest-to-host block, in addition to applicable data obtained after the host program has performed the database retrieval/update. The majority of the data sent back to the guest output program, in the host-to-guest block, will be displayed on the screen.

Before transferring to the guest side, the host program moves the length of its host-to-guest block to the common area. This length will tell the interface modules the length of the applicable host-to-guest block being passed. Each host-to-guest block has a different length. For the “UPH” screen, the block consists of a fixed header (174 bytes) and up to ten details (each is 47 bytes). The length may vary based on how much data was

retrieved from the database (in contrast to the "UPH" screen's guest-to-host block, which is constant). The formula used to compute the length for the "UPH" screen's host-to-guest block is as follows:

174 + (47 * # of details) = length (up to 644 bytes)

When the guest output program receives control from the host program, it will process the data transmitted in the host-to-guest block. Some of this data is moved to the common area. When the LX table is executed, data from the common area is moved to the output map. When the guest input program receives control from CICS again, it will take the data from the common area (which the guest output program moved), plus any new data entered on the screen, and move it to the guest-to-host block. This block will again be sent to the host programs and the cycle repeats itself.

Guest Common Area Structure

(4074 bytes in length)

Note that each section of the common-area has been described in a separate definition. The length of each section is given, and the content is described as offsets relative to the section. COBOL layouts will be provided. Areas which are used by high-level functions (and are transparent to the applications) have not been detailed. Most areas preceding the application work-area must be treated as read-only by the applications. There is a small set of exceptions to this which are noted. The general organization of this structure is very similar to the host side.

Protocol Control Area (64 bytes in length)

Bytes	Len	Usage
0000-0007	08	general purpose double-word aligned temporarily
0008-0009	02	length of common-area. halfword aligned for XCTL, LINK.
0010-0011	02	current inquiry/response detail length. initialized to zero and maintained by appropriate assist routine(s).
0012-0015	04	current function-name
0016-0017	02	internal control flag bits
0018-0021	04	previous function-name
0022-0023	02	length of last receive-map
0024-0035	12	process control flags from PCTEG for current function
0036-0042	07	previous map-name
0043-0043	01	internal flag
0044-0047	04	next or current transid
0048-0048	01	internal flag
0049-0063	15	reserved for protocol dependency

Module Control Name Area (192 bytes in length)

Bytes	Len	Usage
0000-0007	08	guest-to-host protocol interface module
0008-0015	08	1st application service module for current invocation
0016-0023	08	2nd application service module for current invocation
0024-0031	08	3rd application service module for current invocation
0032-0039	08	4th application service module for current invocation (cleanup module) note that only 1 application module is required for a given function.
0040-0047	08	after-host return module. copied by application into this cell from one of the preceding 4.
0048-0055	08	optional edit/map/demap table (LXTBL) module
0056-0063	08	crt sendback module
0064-0071	08	internal dispatch routine
0072-0079	08	reference list manager
0080-0087	08	non-resident services module
0088-0191	120	room for 13 additional names

Assist Routine/Table Area (192 bytes in length)

Bytes	Len	Usage
0000-0003	04	address of date-conversion module
0004-0007	04	address of data-name to address conversion routine
0008-0011	04	address of text fetch routine for host output text
0012-0015	04	address of text fetch routine for host input text
0016-0019	04	address of guest control-table
0020-0023	04	address of translate/test table set
0024-0027	04	address of edit utility set
0028-0031	04	address of Shell Sort routine
0032-0191	164	room for 40 additional entries

Post Function Security Data (24 bytes in length)

Bytes	Len	Usage
0000-0023	24	reserved for security interface data (5 addresses used, 1 available)

Scroll Function Data (742 bytes in length)

Bytes	Len	Usage
0000-0003	04	last scroll-function name
0004-0005	02	reserved
0006-0017	12	scroll table descriptor. built by application program. 1 byte # entries per line, zoned decimal.

		1 byte # surrogates per entry, zoned decimal (1-8) 2 bytes first line used on screen, binary 8 bytes surrogate type table
0018-0084	67	scroll key. built by application program.
0085-0085	01	core status flag. set by application. must be 'Y' if scroll-area is considered valid.
0086-0181	96	page management table. Used by application for page forward and page back. Format is at application discretion. (12 8-byte entries)
0182-0741	560	scroll/browse table. set of up to 140 4-byte entries surrogate entries formatted as indicated by the scroll descriptor table. Built by application, used by function-dispatch.

WARNING: **Cells above this point are read-only unless otherwise noted.** Cells below this point are read-write unless otherwise noted.

Data Services Work Area (512 bytes in length)

Bytes	Len	Usage
0000-0001	02	flags for data-move routines
0002-0021	20	date conversion area. contains EIBDATE in binary format on initial function-entry if LXTBL is not used.
0022-0028	07	name of current map. set by application if not using LXTBL. Read-only if using LXTBL.
0029-0030	02	Entry reason-code for current invocation
0031-0031	01	type of surrogate if entry is F4 or F9. read-only.
0032-0035	04	surrogate value if entry is F4 or F9. read-only.
0036-0037	02	EIBDATE in binary (WAASDATE) format. read-only.
0038-0097	60	reserved surrogate list. application agreement is required on which significant surrogate is in which cell.
0098-0101	04	output map-length. read-only if LXTBL used.
0102-0105	04	address of input-map area. read-only.
0106-0109	04	address of output-map area. read-only.
0110-0113	04	address of host-interface area, zero if no area. read-only.
0114-0129	16	reserved for up to 4 additional addresses. read-only.
0130-0130	01	error-intercept flag. set to 'Y' by application if host errors are to be intercepted without return to the application.
0131-0131	01	internal dispatch type. application must set to '0' if using an internal dispatch service.
0132-0135	04	name of function requested for internal dispatch.
0136-0159	24	first 24 detail bytes from last host call. read-only.
0160-0511	352	reserved

Undefined Area (2 bytes)

Bytes	Len	Usage
0000-0001	02	reserved

Error/Message Area (54 bytes)

Bytes	Len	Usage
0000-0003	04	current error/message code
0004-0053	50	override text

Cursor Area (2 bytes)

Bytes	Len	Usage
0000-0001	02	cursor position data. read-only if LXTBL used.

Individual Application Work Area (2290 bytes)

Bytes	Len	Usage
0000-2289	2290	reserved for any use any application wants to make of this area. Preserved across psuedo-conversational interactions by control-dispatch. not guaranteed relative to format between applications.

UMS Software-Interface Hooks

The guest side of the UMS system has been designed to minimize points of interface with the external environment and to provide specific points where a user site can hook to the system with reasonable levels of effort and impact.

As a general design, UMS Guest:

- Interfaces with the host in only one module - UGZ011P
- Obtains normal 3270 input in only one module - UGZ0001P
- Outputs normal 3270 output in only one module - UGZ0005P

Normal 3270 input/output is used in the context of *CICS RECEIVE-MAP* and *SEND-MAP*. Additionally, the 3270 device is accessed in *READ-BUFFER* mode and/or *SEND* mode when the secondary session is entered or exited.

As a design objective, we have assumed that the user site might typically desire to log transmissions to and/or from the host and/or generate its own interactions with the host.

To this end, the host activity was centered in one module, UHZ0011P (not distributed). We have tried to make it relatively easy for the user to write these activities to a log or invoke a user program (exit) to process them. To that end, we wrote the code such that we believed the points at which this would be done to be relatively obvious.

Secondly, the data structure passed to UGZ0011P is such that it can be generated relatively easily without using all of the UMS Guest software. This allows invoking the host from some other program at the guest site without using the normal guest dispatch-control mechanism. As a matter of practice, this is the technique used by **UGTL/UGTH** (the guest initialization) transactions to invoke the host and can be seen in a limited form in UGZ0020P.

If a user site elects this approach, the site commits itself to using the RMV's guest host data block formats. These formats are subject to potential change, but have been designed to make the upgrade as painless as possible. In particular, the design includes a 4 byte block type field with a commitment on the part of the RMV to provide an overlap period where both the old and new block type codes function. Such an overlap period allows the unmodified guest an upgrade window and the modified guest a development window (if required). Because the RMV is committed to at least one guest which is totally a user development effort, it is clear that some degree of documentation will proceed any block type changes.

It would also be possible for a user site to hook into the UGZ0001P and UGZ0005P programs with the intent of simulating BMS map input/output functions. However, the following cautions are in order:

1. Across releases to date, UGZ0011P has been very stable and we expect this to be an ongoing trend. UGZ0001P has not been stable, and we do not expect it to be. UGZ0005P has been more stable than UGZ0001P and less than UGZ0011P. Any user hooking these programs commits themselves to perpetuating their code across all new releases in critical modules where the RMV has no design intent to support release exits or other assistance for this effort.

2. The RMV has every intention of modifying (as required for bugs or development) screen formats at release time. Such modifications would seriously impact any user who elected to hook in this method.

3. The current UMS Guest takes some effort to be reasonably efficient in its 3270 transmissions. Such efforts will certainly increase as the product matures. Knowledge of this position should indicate to anyone planning to hook our SEND-MAP images the amount of work required. Also, any such effort would have to be cognizant of the 3270 READ-BUFFER usage.

Host Common Area Structure

(3072 bytes in length)

Note that each section of the Common Area has been described in a separate definition. The length of each section is given, and the content is described as offsets relative to the beginning of the section. COBOL layouts will be provided. Areas which are used by high-level functions (and are transparent to the applications) have not been detailed.

Protocol Control Area (64 bytes in length)

Bytes	Len	Usage
0000-0007	08	general purpose double-word aligned temporarily
0008-0009	02	length of common-area. halfword aligned for XCTL, LINK.
0010-0011	02	current response detail length. initialized to zero and maintained by appropriate assist routine(s).
0012-0071	60	reserved for protocol dependency

Module Control Name Area (192 bytes in length)

Bytes	Len	Usage
0000-0007	08	output-side protocol processor name
0008-0015	08	cleanup processor name
0016-0023	08	1st application service module for current invocation
0016-0191	176	room for 22 additional names (13 in use)

Assist Routine/Table Area (192 bytes in length)

Bytes	Len	Usage
0000-0003	04	address of date-conversion module
0004-0007	04	address of data-name to address conversion routine
0008-0011	04	address of text insertion routine for output text
0012-0015	04	address of data-dependent security evaluation routine
0016-0191	176	room for 44 additional entries

Post Function Security Data (64 bytes in length)

Bytes	Len	Usage
0000-0063	64	reserved for data-dependent security function use

Interchange Text Body (2048 bytes in length)

Bytes	Len	Usage
0000-0023	24	Routing information. Used by Guest. Returned with response.
0024-0035	12	inbound security descriptor area. 4 bytes guest site name 4 bytes guest sub-site name (agent) 4 bytes guest sub-site password (note that this field is not valid when passed to application processes)
0024-0035	12	outbound timing data (provided by protocol processor) 6 bytes same as inbound security descriptor 2 bytes time of response 2 bytes date of response
0036-0039	04	Block (transaction) type code (same as inbound) 1 bytes subsystem code (reg, lic, ums, etc) 3 bytes subsystem function code
0040-2047	2008	This area constitutes the maximum length variable text for guest-to-host or host-to-guest transmissions. The treatment of this area is different depending on the direction of the transmission.

Host-to-Guest

Bytes	Len	Usage
0040-0043	04	Application dependent response status flags. Byte 0040 used for scroll response: '1' data presented ends scroll (end-of-set) '2' data presented does not end scroll
0044-0047	04	error-code. from defined list.
0048-0049	02	response count. 00=none, 01=one, etc.
0050-0061	12	error qualifier for output.
0061-2047	??	transaction dependent response for output.

Guest-to-Host

Bytes	Len	Usage
0040-0060	22	preserved header portion of input transaction
0061-2047	??	variable text body

The concept being suggested here is that logically the variable portion of the input text body consist of two areas. These areas are separate only in that the function dispatch process will isolate the first 22 bytes previous to invoking the application process, and place it as shown below. From an application perspective, this allows formatting the inbound text with critical data first, and so placed that the application need not have concern about overlaying this data with initial response data. From a control overhead

perspective, in the case of longer input text bodies, the time spent shifting data down (as was previously envisioned) is reduced.

Data Services Work Area (512 bytes in length)

Bytes	Len	Usage
0000-0511	512	reserved for data-services and application use. First 22 bytes are used to preserve the first 22 bytes of the input text body. Next 16 bytes are defined as standard parameter locations for assist routines. When not calling the routines using these cells, they may be overlaid.

3 UMS Naming and Version Conventions

All of the UMS application material (Subschemas, Copy Members, Programs, LX tables, and Maps) will use the following naming standard. The Module Name has the form of **'ABCDDDDE'** where:

A Constant of 'U' For UMS System

B Constant of 'G' For Guest Site
Constant of 'H' For Host Site
Constant of 'I' For Independent(used in both)
Constant of 'V' for Virtual Guests

C Application Function

B	Booking System (exam)
C	Cash
E	Emmissions
H	MAB
I	Inspection Maintenance
L	License
M	MRB
N	Non-Renew License
P	Policy Modules
R	Registration Modules
S	Suspension
U	Uninsured Motorists and Cross System
V	Motor Voter
W	Overweight Permits
Z	System Control Modules, General (edits, tables, etc.)

DDDD Number 1 through 9999

E Type of Program

P	Program
M	Map - (Guest side)
L	Subschema - (Host side)
T	LX table - (Guest side)
J	JCL
G	Copybook

Y Copybook

4 UMS Program Control Tables

One of the objectives of the UMS system design is the simplification of application design, coding, maintenance, and release-control. This is especially important where a part of the software (the Guest system) is distributed to outside users. To this end, the system control structure is maintained outside of the application programs. Applications are not to issue CICS Program Control Macros because these involve program-name literals. If one is to change program names or the flow of control then all affected programs must be changed and recompiled. In UMS, **UGZ0001P / UHZ0001P** (Guest/Host Control Program) in conjunction with the **UGZ0002P / UHZ0002P** (Guest/Host Control Table is used to control the execution of the host and guest programs that make up the UMS application online system. During initialization of the application, the control program will move the names of the programs designated in the Control Table to the common area for later use by the application programs. The Guest and Host tables, referred to as **PCTEG** and **PCTEH** after the name of their principal macro, vary slightly in structure while serving the same purpose.

Program Control Table Entry Guest (PCTEG):

The PCTEG controls execution of guest side programs and LXTABLE processing. When the user enters a UMS function code, the Guest Side Dispatcher (UGZ0001P) will access the PCTEG, based on the UMS screen function code. PCTEG passes control to the appropriate guest input program (the **first** in the program-name list). Following is an example of an entry in the PCTEG for UPOI:

PCTEG “UPOI,” PGM = (UGU1011P, UGU1012P), XLATE = UGU1010T

PCTEG The macro invocation for this entry.

UPOI Is the UMS screen function code for this entry.

PGM= Designates the guest program name-list.

XLATE Names the LX table for the UPOI function.

Warning: Be certain to review the detail discussion that follows, as there are more parameters than are shown above.

The table-driven control structure facilitates versioning of programs and functions. The following example illustrates the steps necessary to update the PCTEG to accommodate versioning of the UPH screen from release one to two.

Source Code Example - PCTEG

This example is the source for the assembled code that follows and its hexidecimal dump.

Note: These examples have been slightly modified by text-edit to allow them to fit in the page or to improve their readability. For brevity, some macros have a number of parameters to show the results of expansion. Some combinations, while assembled correctly may be illogical. Please consult the detailed write-up for parameter usage.

```
UGZ0002P  CSECT
NUMENT    DC        F'0'          NUMBER OF ENTRIES
          DC        A(FIRST)      ADDRESS OF FIRST ENTRY
NUMXFER   DC        F'0'          NUMBER OF XFER (DUPKEY) ENTRIES
          DC        A(XFER1)      ADDRESS OF FIRST XFER ENTRY
NUMALIAS  DC        F'0'          NUMBER OF ALIAS ENTRIES
          DC        A(ALIAS1)     ADDRESS OF FIRST ALIAS ENTRY
UGZ0002Q  CSECT
XFER1     EQU       *             FIRST XFER ENTRY
UGZ0002R  CSECT
ALIAS1    EQU       *             FIRST ALIAS ENTRY
UGZ0002P  CSECT
*
FIRST     DS        0F
          PCTEG     'ALAR',PGM=UGZ0009P,TASK=ALAR,XFER=YES
*
*
          PCTEG     'ALAS',PGM=(UGZ0009P,XFERTEST),XFER=DATA
*
*
          PCTEG     'LN  ',PGM=(UGL0020P,UGL0021P,UGL0022P),
                    RESCROLL=YES,DUPKEY=LI04,ALIAS='ULN ',
                    XLATE=UGL0020T,UPDATE=YES,
                    FLAGS=FF0000000000EE11,CLEAN=CLEANPGM,
                    HOSTA=YES,SCROLL=YES
*
*
          PCTEG     'LNO ',PGM=(UGL0240P,UGL0241P),HOSTA=YES,
                    RESCROLL=YES,DUPKEY=LD08,XLATE=UGL0240T,
                    SCROLL=YES,ALIAS='ULNO'
*
*
          PCTEG     'STAT',PGM=UGZ0010P,HOSTA=NO,
                    FLAGS=FF00000000000000
```

```
*
*
            PCTEG       'TPIC',PGM=(UGUI120P,UGUO120P),HOSTA=YES,
                        RESCROLL=YES,DUPKEY=LD18,
                        XLATE=UGUX120T,SCROLL=YES,
                        INTERNL=YES,UPDATE=YES,PF9=01
*
*
            PCTEG       'TRI ',PGM=(UGRI110P,UGRO110P),
                        XLATE=UGRX110T,HOSTA=YES,PF4=1024,PF9=1024
*
*
            PCTEG       'VER ',PGM=UGZ0032P,HOSTA=NO
*
*
LAST        DC          XL4'FFFFFFFF'
UGZ0002Q    CSECT
XFER2       EQU         *               FIRST XFER ENTRY
UGZ0002P    CSECT
            ORG         NUMENT
            DC          A((LAST-FIRST)/(SYM2-SYM1))
            ORG         NUMXFER
            DC          A((XFER2-XFER1)/8)
            CLEANUP
            ORG
            DC          CL8'&SYSDATE',CL1' ',CL5'&SYSTIME'
            DC          C'=VER 03.01='
            END
```

Note: Change in Use for UMS

Part of the original intent of UGZ0002P was to provide a reference to all application (3rd character of name not ‘Z’) modules required by the guest, and to require all references to these module names to be indirect through UGZ0002P.

A decision was made in a previous UMS Release to send all modules to the Insurance Companies. Part of the reason was based on **coding inconsistencies**, which referenced module names without indirect references through UGZ0002P. This resulted in releases not including adequate modules.

Therefore, there is no further need to run the program as **part of the UMS Tape Release procedure** to identify programs by the TYPE parameter. However, all other uses for the module are still very much intact with regard to PCTEG/PCTEH, which controls execution of the programs and LXTABLE processing.

The PCTEG in Detail

UGZ0002P is the module also known as PCTEG. This module controls the functions that can be entered at the guest site on the screen in the field labeled **FUNCTION**. This module needs to move up the "ladder" to the higher regions as new functions are moved up for further integrated testing.

When current functions are being replaced by "new" functions, additional table entries are needed to allow both the current and "new" function to co-exist in the same region (This assumes that module names have changed, but function has remained the same). For example, there is a "new" version of LI. The current entry would be used to create an entry called **TLI** (the entries have to be in ascending alphabetical order). The TLI entry would need to have the *alias* parameter removed if it existed. The "new" LI entry would look like the old, except for the program name changes and any other parameter changes.

After the table changes are made, the table must be reassembled and linked into the proper test load library.

```
PCTEG     <tskcode>,PGM=,TASK=,XLATE=,FLAGS=,CLEAN=,
          ALIAS=,SCROLL=NO,RESCROLL=NO,UPDATE=NO,
          HOSTA=NO,XFER=NO,PF4=,PF9=,PF7AND8=NO,
          DUPKEY=,INTERNAL=NO,TYPE=ALL
```

Field Definitions

<tskcode> A 4-byte task code within quotes. PCTE entries **must be** sequenced by this code. Assumed to be a test entry if the first byte is "T."

ALIAS=**<null>** | <altskcode> Optional alternate task code for this function. Same rules as name, except sequencing is not required/appropriate. It may not duplicate any NAME parm. If ALIAS is specified, INTERNAL must equal NO.

PGM=(<pgmlist>) 1 to 3 processing program names separated by commas. 1 name is required. A given application program may be the **first** program for one non-test entry and one test entry, only.

TASK=<optaskcd> A 4-byte optional dispatch task (transaction) code (or name) (4 ebcdic or 8 hex digits).

XLATE=**<null>** | <lxmodname> Name of optional map/demap translation table. 1 to 8 characters long.

FLAGS=**<null>** | <flags> An 8-byte optional flagset (expressed in hex, 16 digits) (not allowed if F4 or F9 strings present except as surrogate types). These are available to the programs in this function and may be used to direct processing based on the function code that invoked the program(s). The contents are strictly application dependent.

CLEAN=**<null>** | <clnpgm> Name of optional tail-end cleanup program, 1 to 8 characters. If omitted the system clean-up program UGZ0005P is used. It is normal to omit this parameter.

SCROLL=YES/**NO** If YES, program accepts F7/F8 and issues F9/F4.

RESCROLL=YES/**NO** If program has SCROLL=YES **and** can accept rescroll.

UPDATE=YES/**NO** Program accepts F12 for update

HOSTA=YES/**NO** Program requires host interface area

F4=<null> | <sgcdlist> String of 1 to 4 surrogate codes allowed for F4 entry. Not allowed if SCROLL=YES. No quotes, no commas.

F9=<null> | <sgcdlist> String of 1 to 4 surrogate codes allowed for F9 entry. Not allowed if SCROLL=YES. No quotes, no commas.

F7AND8=YES/**NO** If YES, pgm accepts F7/F8 but is not a scroll pgm.

DUPKEY=<null>| <hostcd> The 4-byte host duplicate-key function associated with this guest function. If used, HOSTA=YES is required.

INTERNL=YES/**NO** Is this function restricted to internal dispatch (as opposed to user or keyboard initiated dispatch)?

TYPE=**ALL** | RMV This specifies if this function is applicable to all sites or just RMV. It determines the relationship of this entry to the global assembly options in sysparm. See the following Sysparm Option discussion.

XFER=YES/**NO**/DATA This directs the immediate transfer of control to a task-code or program external to the UMS Guest system. The PGM parameter must have UGZ0009P as the first program and HOSTA=NO. If transfer is to a task, than TASK=<xfertask>, where <xfertask> is the transid to start, is required. Control passes by a CICS START. If control goes to a program (<xferpgm>) then its name will be the second value in the PGM= parameter; ie. PGM=(UGZ0009p,<xferpgm>). Control will pass by CICS XCTL. YES and DATA are synomonous except that the DATA option passes a 23 byte (comm) data area. When this option is specified, the PGM= option must specify the foreign interface module (UGZ0009P) as

the first program. A value must be specified for the TASK= option, and a value may be specified for a second program name. If XFER=DATA, a parameter-area of the following format is built by the UMS control-dispatch mechanism for the foreign application:

bytes	len	usage
00-03	4	ID used to sign-on to UMS
04-11	8	Password used to sign-on to UMS
12-15	4	UMS function-code entered
16-17	2	UMS entry-code
18-22	5	UMS surrogate-type and code if F4 or F9

total length: 23 bytes

If control-dispatch detects that no second program-name was specified, it starts a terminal attached transaction (taking the name from the TASK= operand) in behalf of the terminal. If XFER=DATA was specified, the 23-byte parameter-area is passed as start-data, otherwise there is none. If control-dispatch detects that a second program-name was specified, it XCTL's to the specified program. Only if XFER=DATA is specified, is the 23-byte parameter-area is passed as a common-area, otherwise no common-area is passed. The invoked transaction runs under the transaction-id UMS was running under.

The Guest common area is freed before the transfer.

SYSPARM Options

The use of the SYSPARM option allows the creation of tailored output from the assembly of the Program Control Table. The output can be:

1. Full load module with all entries.
2. Load module with RMV-peculiar entries.
3. Generate delete control cards only.

One enters the sysparm into the standard assembly proc as follows:

CPARM2=',SYSPARM=<umstblopt>'.

where <umstblopt> may be as follows:

<umstblopt>=**GENERATE_ALL**

The full PCTEG will be generated. This is the default if no sysparm is present.

<umstblopt>=GENERATE_NOT_RMV

A partial PCTEG will be generated. All entries with TYPE=RMV will be excluded from the generation.

<umstblopt>=PUNCH_DELETE_DSN=<data-set-name>

No PCTEG will be generated. Card images of the form:

DELETE <data-set-name>(<member-name>)

will be generated for all modules referenced on the PCTEG/TYPE= statement. These cards may be used as IDCAMS input. This option will also raise CONDITION-CODE=7, largely to inhibit execution of the LKED step in the standard assembly proc.

Example:

SYSPARM=PUNCH_DELETE_DSN=RMVMV.UMS.LOADLIB

might generate, among others:

DELETE RMVMV.UMS.LOADLIB(UGZ0002P)

Assembled Code Example - PCTEG

Note: These examples have been slightly modified by text-edit to allow them to fit in the page or to improve their readability. For brevity, some macros have a number of parameters to show the results of expansion. Some combinations, while assembled correctly may be illogical. Please consult the detailed write-up for parameter usage.

```
                                                EXTERNAL SYMBOL DICTIONARY                PAGE    1
SYMBOL   TYPE  ID  ADDR  LENGTH  LD ID FLAGS                          ASM H V 02 11.28 07/23/91

UGZ0002P  SD 0001 000000 000215         00
UGZ0002Q  SD 0002 000218 000018         00
UGZ0002R  SD 0003 000230 000010         00

000000                                  371 UGZ0002P CSECT                                    03810000
000000 00000000                         372 NUMENT   DC    F'0'       NUMBER OF ENTRIES      03820000
000004 00000018                         373          DC    A(FIRST)   ADDRESS OF FIRST ENTRY 03830000
000008 00000000                         374 NUMXFER  DC    F'0' NUMBER XFER (DUPKEY) ENTRIES 03840000
00000C 00000218                         375          DC    A(XFER1)   ADDR FIRST XFER ENTRY  03850000
000010 00000000                         376 NUMALIAS DC    F'0'       NUMBER ALIAS ENTRIES   03860000
000014 00000230                         377          DC    A(ALIAS1)  ADDR FIRST ALIAS ENTRY 03870000
000218                                  378 UGZ0002Q CSECT                                    03880000
                               00218    379 XFER1    EQU   *              FIRST XFER ENTRY  03890000
000230                                  380 UGZ0002R CSECT                                    03900000
                               00230    381 ALIAS1   EQU   *              FIRST ALIAS ENTRY 03910000
000018                                  382 UGZ0002P CSECT                                    03920000
                                        383 *                                                 03930000
000018                                  384 FIRST    DS    0F                                 03940000
                                        385          PCTEG 'ALAR',PGM=UGZ0009P,             X
                                                           TASK=ALAR,XFER=YES                 03950000
```

```
000018                          386+SYM1     DS    0CL1                                 01-00282
000018 C1D3C1D9C1D3C1D9         387+         DC    CL4'ALAR',CL4'ALAR'                  01-00292
000020 E4C7E9F0F0F0F9D7         388+GST0001  DC    CL8'UGZ0009P',CL8'       '           01-00293
000030 4040404040404040         389+         DC    CL8'        ',CL8'       '           01-00294
000040 4040404040404040         390+         DC    CL8'      '                          01-00295
000048 00000000                 391+         DC    AL1(0),AL1(0),AL2(0)                 01-00296
00004C 0000000000000000         392+         DC    XL8'0000000000000000'                01-00301
000054                          393+SYM2     DS    0CL1                                 01-00303
                                394 *                                                   03960001
                                395 *                                                   03970001
                                396          PCTEG 'ALAS',PGM=(UGZ0009P,XFERTEST),  X
                                                    XFER=DATA                           03980001
000054 C1D3C1E240404040         397+         DC    CL4'ALAS',CL4'    '                  01-00292
00005C E4C7E9F0F0F0F9D7         398+GST0008  DC    CL8'UGZ0009P',CL8'XFERTEST'          01-00293
00006C 4040404040404040         399+         DC    CL8'        ',CL8'       '           01-00294
00007C 4040404040404040         400+         DC    CL8'      '                          01-00295
000084 20000000                 401+         DC    AL1(32),AL1(0),AL2(0)                01-00296
000088 0000000000000000         402+         DC    XL8'0000000000000000'                01-00301
                                403 *                                                   03990001
                                404 *                                                   04000001
                                405          PCTEG 'LN  ',ALIAS='ULN ',             X
                                                    PGM=(UGL0020P,UGL0021P,UGL0022P),X
                                                    RESCROLL=YES,DUPKEY=LI04,          X
                                                    XLATE=UGL0020T,UPDATE=YES,         X04030003
                                                    FLAGS=FF0000000000EE11,            X
                                                    CLEAN=CLEANPGM,                    X04040002
                                                    HOSTA=YES,SCROLL=YES                04050002
000218                          406+UGZ0002Q CSECT                                      01-00267
000218 D3C9F0F400000090         407+         DC    CL4'LI04',AL4(SYM$LI04)              01-00268
000090                          408+UGZ0002P CSECT                                      01-00269
                         00090  409+ALI0015    EQU *                                  01-00290
000090 D3D5404040404040         410+SYM$LI04 DC    CL4'LN  ',CL4'    '                  01-00292
000098 E4C7D3F0F0F2F0D7         411+PUGL0020P DC   CL8'UGL0020P',CL8'UGL0021P'          01-00293
0000A8 E4C7D3F0F0F2F2D7         412+         DC    CL8'UGL0022P',CL8'CLEANPGM'          01-00294
```

```
0000B8 E4C7D3F0F0F2F0E3            413+         DC    CL8'UGL0020T'                        01-00295
0000C0 C7000000                    414+         DC    AL1(199),AL1(0),AL2(0)               01-00296
0000C4 FF0000000000EE11            415+         DC    XL8'FF0000000000EE11'                01-00301
                                   416 *                                                   04060003
                                   417 *                                                   04070003
                                   418          PCTEG 'LNO ',PGM=(UGL0240P,UGL0241P),  X
                                                      HOSTA=YES,XLATE=UGL0240T,          X
                                                      RESCROLL=YES,DUPKEY=LD08,          X
                                                      SCROLL=YES,ALIAS='ULNO'              04100003
000220                             419+UGZ0002Q CSECT                                      01-00267
000220 D3C4F0F8000000CC            420+         DC    CL4'LD08',AL4(SYM$LD08)              01-00268
0000CC                             421+UGZ0002P CSECT                                      01-00269
                          000CC    422+ALI0022    EQU *                                    01-00290
0000CC D3D5D64040404040            423+SYM$LD08 DC    CL4'LNO ',CL4'    '                  01-00292
0000D4 E4C7D3F0F2F4F0D7            424+PUGL0240P DC   CL8'UGL0240P',CL8'UGL0241P'         01-00293
0000E4 4040404040404040            425+         DC    CL8'        ',CL8'        '        01-00294
0000F4 E4C7D3F0F2F4F0E3            426+         DC    CL8'UGL0240T'                        01-00295
0000FC C3000000                    427+         DC    AL1(195),AL1(0),AL2(0)               01-00296
000100 0000000000000000            428+         DC    XL8'0000000000000000'                01-00301
                                   429 *                                                   04110001
                                   430 *                                                   04120001
                                   431          PCTEG 'STAT',PGM=UGZ0010P,HOSTA=NO,    X04130000
                                                      FLAGS=FF00000000000000               04140000
000108 E2E3C1E340404040            432+         DC    CL4'STAT',CL4'    '                  01-00292
000110 E4C7E9F0F0F1F0D7            433+GST0029  DC    CL8'UGZ0010P',CL8'        '          01-00293
000120 4040404040404040            434+         DC    CL8'        ',CL8'        '        01-00294
000130 4040404040404040            435+         DC    CL8'      '                          01-00295
000138 00000000                    436+         DC    AL1(0),AL1(0),AL2(0)                 01-00296
00013C FF00000000000000            437+         DC    XL8'FF00000000000000'                01-00301
                                   438 *                                                   04150004
                                   439 *                                                   04160004
                                   440          PCTEG 'TPIC',PGM=(UGUI120P,UGUO120P),  X
                                                      RESCROLL=YES,DUPKEY=LD18,          X
                                                      XLATE=UGUX120T,HOSTA=YES,PF9=01,   X
```

```
                                                    SCROLL=YES,INTERNL=YES,UPDATE=YES 04190004
000228                                441+UGZ0002Q CSECT                                01-00267
000228 D3C4F1F800000144               442+         DC    CL4'LD18',AL4(SYM$LD18)        01-00268
000144                                443+UGZ0002P CSECT                                01-00269
000144 E3D7C9C340404040               444+SYM$LD18 DC    CL4'TPIC',CL4'    '            01-00292
00014C E4C7E4C9F1F2F0D7               445+TUGUI120P DC   CL8'UGUI120P',CL8'UGUO120P'    01-00293
00015C 4040404040404040               446+         DC    CL8'        ',CL8'        '    01-00294
00016C E4C7E4E7F1F2F0E3               447+         DC    CL8'UGUX120T'                  01-00295
000174 D7010000                       448+         DC    AL1(215),AL1(1),AL2(0)         01-00296
000178 40404040F0F14040               449+         DC    CL4' ',CL4'01'                 01-00298
                                      450 *                                             04200001
                                      451 *                                             04210001
                                      452          PCTEG 'TRI ',PGM=(UGRI110P,UGRO110P),  X
                                                    XLATE=UGRX110T,                   X04220000
                                                    HOSTA=YES,PF4=1024,PF9=1024        04230000
000180 E3D9C94040404040               453+         DC    CL4'TRI ',CL4'    '            01-00292
000188 E4C7D9C9F1F1F0D7               454+TUGRI110P DC   CL8'UGRI110P',CL8'UGRO110P'    01-00293
000198 4040404040404040               455+         DC    CL8'        ',CL8'        '    01-00294
0001A8 E4C7D9E7F1F1F0E3               456+         DC    CL8'UGRX110T'                  01-00295
0001B0 58000000                       457+         DC    AL1(88),AL1(0),AL2(0)          01-00296
0001B4 F1F0F2F4F1F0F2F4               458+         DC    CL4'1024',CL4'1024'            01-00298
                                      459 *                                             04240001
                                      460 *                                             04250001
                                      461          PCTEG 'VER ',PGM=UGZ0032P,HOSTA=NO     04260000
0001BC E5C5D94040404040               462+         DC    CL4'VER ',CL4'    '            01-00292
0001C4 E4C7E9F0F0F3F2D7               463+GST0050  DC    CL8'UGZ0032P',CL8'        '    01-00293
0001D4 4040404040404040               464+         DC    CL8'        ',CL8'        '    01-00294
0001E4 4040404040404040               465+         DC    CL8'      '                    01-00295
0001EC 00000000                       466+         DC    AL1(0),AL1(0),AL2(0)           01-00296
0001F0 0000000000000000               467+         DC    XL8'0000000000000000'          01-00301
                                      468 *                                             04270001
                                      469 *                                             04280001
0001F8 FFFFFFFF                       470 LAST     DC    XL4'FFFFFFFF'                  04290000
000230                                471 UGZ0002Q CSECT                                04300000
```

```
                          00230   472 XFER2    EQU   *               FIRST XFER ENTRY  04310000
0001FC                            473 UGZ0002P CSECT                                   04320000
0001FC                    00000   474          ORG   NUMENT                           04330000
000000 00000008                   475          DC    A((LAST-FIRST)/(SYM2-SYM1))      04340000
000004                    00008   476          ORG   NUMXFER                          04350000
000008 00000003                   477          DC    A((XFER2-XFER1)/8)               04360000
                                  478          CLEANUP                                04370000
000230                            479+UGZ0002R CSECT                                   01-00350
000230 E4D3D54000000090           480+         DC    CL4'ULN ',AL4(ALI0015)           01-00351
000238 E4D3D5D6000000CC           481+         DC    CL4'ULNO',AL4(ALI0022)           01-00360
                          00240   482+ALIEND   EQU   *                                01-00363
00000C                            483+UGZ0002P CSECT                                   01-00364
00000C                    00010   484+         ORG   NUMALIAS                         01-00365
                          00002   485+ALICOUNT EQU   (ALIEND-ALIAS1)/8                01-00366
000010 00000002                   486+         DC    AL4(ALICOUNT)                    01-00367
000014                    001FC   487+         ORG                                    01-00368
0001FC                    001FC   488          ORG                                    04380000
                                  489          DC    CL8'&SYSDATE',CL1' ',CL5'&SYSTIME'04390000
0001FC F0F761F2F361F9F1             +          DC    CL8'07/23/91',CL1' ',CL5'11.28'  04390000
00020A 7EE5C5D940F0F34B           490          DC    C'=VER 03.01='                   04400000
                                  491          END                                    04410000

                                               RELOCATION DICTIONARY                PAGE   11
 POS.ID   REL.ID   FLAGS   ADDRESS                                ASM H V 02 11.28 07/23/91
  0001     0001     0C     000004
  0001     0002     0C     00000C
  0001     0003     0C     000014
  0002     0001     0C     00021C
  0002     0001     0C     000224
  0002     0001     0C     00022C
  0003     0001     0C     000234
  0003     0001     0C     00023C

                              DIAGNOSTIC CROSS REFERENCE AND ASSEMBLER SUMMA
```

```
    NO STATEMENTS FLAGGED IN THIS ASSEMBLY
  OVERRIDING PARAMETERS-  LIST,NODECK,LOAD,LINECOUNT(65)
  OPTIONS FOR THIS ASSEMBLY
   NODECK, OBJECT, LIST, XREF(FULL), NORENT, NOTEST, NOBATCH, ALIGN, ESD, RLD,NOTERM, NODBCS,
   LINECOUNT(65), FLAG(0), SYSPARM()
  NO OVERRIDING DD NAMES
    433 CARDS FROM SYSIN       0 CARDS FROM SYSLIB
    584 LINES OUTPUT          23 CARDS OUTPUT

 MVS/XA DFP VER 2 LINKAGE EDITOR                11:28:44  TUE  JUL 23, 1991
 JOB RMCJEBBE   STEP STEP010    PROCEDURE LKED
 INVOCATION PARAMETERS - LIST,XREF
 ACTUAL SIZE=(317440,79872)
 OUTPUT DATA SET IS ON VOLUME RELP02

                                                CROSS REFERENCE TABLE

  CONTROL SECTION                          ENTRY
    NAME    ORIGIN  LENGTH                   NAME   LOCATION     NAME   LOCATION
  UGZ0002P      00     215
  UGZ0002Q     218      18
  UGZ0002R     230      10


  LOCATION  REFERS TO SYMBOL  IN CONTROL SECTION     LOCATION  REFERS TO  SYMBOL  IN CONTROL
SECTION
       C            UGZ0002Q        UGZ0002Q             14             UGZ0002R        UGZ0002R
     21C            UGZ0002P        UGZ0002P            224             UGZ0002P        UGZ0002P
     22C            UGZ0002P        UGZ0002P            234             UGZ0002P        UGZ0002P
     23C            UGZ0002P        UGZ0002P

 ENTRY ADDRESS       00
 TOTAL LENGTH       240
```

```
** TST0002P REPLACED AND HAS AMODE 24
** LOAD MODULE HAS RMODE 24
** AUTHORIZATION CODE IS         0.
```

Hexidecimal Dump of PCTEG Example

```
AMASPZAP  INSPECTS, MODIFIES, AND DUMPS CSECTS OR SPECIFIC DATA RECORDS ON DIRECT ACCESS STORAGE.
  DUMPT TST0002P ALL                                                    00110005

**CCHHR- 0011000419   RECORD LENGTH- 000240        MEMBER NAME  TST0002P  CSECT NAME  UGZ0002P
000000   00000008 00000018 00000003 00000218   00000002 00000230 C1D3C1D9 C1D3C1D9 *.................*
                                                                                  *........ALARALAR*
000020   E4C7E9F0 F0F0F9D7 40404040 40404040   40404040 40404040 40404040 40404040 *UGZ0009P        *
                                                                                  *                *
000040   40404040 40404040 00000000 00000000   00000000 C1D3C1E2 40404040 E4C7E9F0 *        ........*
                                                                                  *....ALAS    UGZ0*
000060   F0F0F9D7 E7C6C5D9 E3C5E2E3 40404040   40404040 40404040 40404040 40404040 *009PXFERTEST    *
                                                                                  *                *
000080   40404040 20000000 00000000 00000000   D3D54040 40404040 E4C7D3F0 F0F2F0D7 *    ............*
                                                                                  *LN      UGL0020P*
0000A0   E4C7D3F0 F0F2F1D7 E4C7D3F0 F0F2F2D7   C3D3C5C1 D5D7C7D4 E4C7D3F0 F0F2F0E3 *UGL0021PUGL0022P*
                                                                                  *CLEANPGMUGL0020T*
0000C0   C7000000 FF000000 0000EE11 D3D5D640   40404040 E4C7D3F0 F2F4F0D7 E4C7D3F0 *G...........LNO *
                                                                                  *    UGL0240PUGL0*
0000E0   F2F4F1D7 40404040 40404040 40404040   40404040 E4C7D3F0 F2F4F0E3 C3000000 *241P            *
                                                                                  *    UGL0240TC...*
000100   00000000 00000000 E2E3C1E3 40404040   E4C7E9F0 F0F1F0D7 40404040 40404040 *........STAT    *
                                                                                  *UGZ0010P        *
000120   40404040 40404040 40404040 40404040   40404040 40404040 00000000 FF000000 *                *
                                                                                  *        ........*
000140   00000000 E3D7C9C3 40404040 E4C7E4C9   F1F2F0D7 E4C7E4D6 F1F2F0D7 40404040 *....TPIC    UGUI*
                                                                                  *120PUGUO120P    *
000160   40404040 40404040 40404040 E4C7E4E7   F1F2F0E3 D7010000 40404040 F0F14040 *            UGUX*
                                                                                  *120TP...    01  *
```

```
000180   E3D9C940 40404040 E4C7D9C9 F1F1F0D7   E4C7D9D6 F1F1F0D7 40404040 40404040 *TRI     UGRI110P*
                                                                                  *UGRO110P        *
0001A0   40404040 40404040 E4C7D9E7 F1F1F0E3   58000000 F1F0F2F4 F1F0F2F4 E5C5D940 *        UGRX110T*
                                                                                  *....10241024VER *
0001C0   40404040 E4C7E9F0 F0F3F2D7 40404040   40404040 40404040 40404040 40404040 *    UGZ0032P    *
                                                                                  *                *
0001E0   40404040 40404040 40404040 00000000   00000000 00000000 FFFFFFFF F0F761F2 *            ....*
                                                                                  *............07/2*
000200   F361F9F1 40F1F14B F2F87EE5 C5D940F0   F34BF0F1 7E                         *3/91 11.28=VER 0*
                                                                                  *3.01=           *

**CCHHR- 0011000419   RECORD LENGTH- 000240        MEMBER NAME  TST0002P  CSECT NAME  UGZ0002Q
000000   D3C9F0F4 00000090 D3C4F0F8 000000CC   D3C4F1F8 00000144                   *LI04....LD08....*
                                                                                  *LD18....        *

**CCHHR- 0011000419   RECORD LENGTH- 000240        MEMBER NAME  TST0002P  CSEC  NAME  UGZ0002R

 NAME  UGZ0002R
000000   E4D3D540 00000090 E4D3D5D6 000000CC                                     *ULN ....ULNO....*

AMA113I COMPLETED DUMP REQUIREMENTS

AMA100I AMASPZAP PROCESSING COMPLETED
  ****************************** BOTTOM OF DATA************************************************
```

Internal Function Codes

UMS internal function codes are used by the Host Side Dispatcher program (UHZ0001P) to indicate which host programs, service modules, and duplicate resolution modules to invoke. The guest input program is responsible for determining which internal function code is needed and moving it to the appropriate displacement in the guest-to-host block of the common area. When control passes to the host side, UHZ0001P matches the internal function code within the guest-to-host block of the common area to the internal function code in the Program Control Table Entry Host table (PCTEH; refer to the Program Control Table section for more detail) to determine which host program to execute.

Each online UMS screen has two sets of internal function codes. One set is used with the current version of the guest and host software and the other set is available for the next version or release of software (refer to the Versioning and Naming Standards sections for more detail). In other words, each time versioning is applied to an online UMS screen, the application programmer alternates between the function codes listed in the columns below.

For example, if UPIC is versioned and the current Guest Input program references "UU14" and "UU15," the new versioned Guest Input program would reference "UU06" and "UU07."

The UPH screen is another example of a screen that employs two function codes at once. On the UPH screen, policy history information may be requested using either an in-state or an out-of-state license number as a key. Currently, if a request is performed using an in-state license, the Guest Input program (UGU3041P) will move a "UU08" to the guest-to-host block of the common area. For an out-of-state license request, "UU09" will be moved to the guest-to-host block of the common area. The next time UPH is versioned, "UU16" will be moved to the guest-to-host block of the common area when the new version of UPH performs an in-state license request. Similarly, "UU17" will be moved when a request using an out-of-state license is performed.

NOTE: The names and function codes below are examples only. These things change with every release of the system to minimize confusion with earlier releases of the software.

Internal Function Codes List

Screen Name	Function Codes
COR	COR1 COR3 URSR
LH	LH10 LH20 LH30
LI	LI07 LI08 LI09 LI10 LI96 LXP1 PR05 LI12 LB10
LN	LI04
LNO	LD08
LNS	LD04
LTH	LI07/LB10 LI08/LB10 LI10/LI09
MRBS	MRBX
NRL	LI10 LI07 LI08 NR40
NRR	NR40 UR15
UP, UR, UL	None
R1C	R116

RA	RA01 RA02 RA03
RH	RH01
RN	RN01
RNF	UR09
SDH	SDH0 SDH1 SDH2 SDH3 SDH4 SDH5 SDH6 SDH7 SDH8 SDH9
ULP	LI03
UMA	MRB0 UMA1
UMC	MRB0 UMC1
UMI9	MRB0 UMIQ UMI3
UMIQ	UMI3
UMO	MRB0 UMO1
UMVH	UMV1
UMVS	UMVX
UMVI	UMV1
UPA	PA05

	PA06
	PA07
	PA08
	PA10
	PA11
	PA12
	PA13
	PA14
UPH	UU16
	UU17
	UU22
	UU23
	UU24
UPIC	UU14
	UU15
	UU20
	UU21
UPMV	PA09
UPOI	UU18
	UU25
UPTH	UU20
	UU26
URBS	BS02
URI	URIK
	URIL
	URIM
	URIN
	UR15
	UR16
	UR17
	UR18
URSN	UR04
	UR07
URSR	UR02
	UR05
URSV	UR03

	UR06
URVN	UR10
	UR16
USH	SH03
	SH04
	SH05
	SH06
	SH07
	SH08
UVH	VH03
	VH04
	VH07
	VH08
VT	VT03

5 LXTABLE Processing

Feature Summary - LXTABLE Processing

The UMS subsystem has incorporated an automatic screen data handler referred to as "LXTABLE Processing," which is located in its GUEST software. This processing mechanism is invoked transparently to the application program(s) by the Control/Dispatch (C/D) mechanism based on the XLATE=<lx-tbl-name> parameter in the System Control Table. The value <lx-tbl-name> is the linkedit name of the assembled LXTABLE. On input and output operations, it converts data from screen to internal format and vice versa. On input only, it will perform the value edits and/or specific content/cross field edits requested. If edit errors occur on input, control never passes to the application, but the screen is rewritten with error messages. The input-edit error-message cycle continues until all fields pass the edits. At that point, control is passed to the designated application program with the screen-data in the Common Area.

LXTABLE processing has two components, the table-driven 'LX-Program' (UGZ0006P) and a macro generated, application specific 'LX-table' linked as a load module. This mechanism's use is **required** in all UMS applications to standardize field editing and screen formatting functions. As the program UGZ0006P serves all applications, it does reduce programmer effort in these areas speeding application development and reducing maintenance time. Its basic design is rooted in a product designed and implemented for the ALAR RA function, but the UMS version is quite different.

The LX-table relates the data fields of the user's commarea with the defined screen-dsect for the application. This table is created by the programmer and defines all the involved data fields. The entries in the table for each field pair may specify certain standard edits, including cross-field checks and values. In this discussion, the term LXTABLE refers to the composite effect of the two components.

The LXTABLE mechanism depends in part, on the presumption that each element has two control fields associated with it. These fields are each 1 byte long and are named ZFLD and TFLD. The ZFLD byte is used to contain commands relative to the field passed between the application and the LXTABLE mechanism. The TFLD is seldom used and is a provision for passing an optional override attribute. The derivation of the names ZFLD and TFLD is rooted in the ALAR RA implementation, and the mnemonic implications of these names (if any) is not known.

General data elements may be located anywhere in the UMS common area, but the structure requires that each data element be immediately preceded by the ZFLD and

TFLD bytes. This is very dissimilar from the ALAR RA implementation which uses 3 lists of fields. The processing supports the use of output-only (as opposed to general) data elements. These elements are always located in the host interface block and are not preceded by the ZFLD/TFLD set.

LXTABLE provides automatic field conversion services for the application. This relieves the application of this coding and insures that they are done in a consistent manner. Complete instructions are available in the Detail Description portion of this chapter. The following are some of the types of field conversion supported. See the Edit Type Table on page 72 for a full listing:

1. Date: on input, dates are analyzed for proper syntactical format (mm/dd/yy, mm/dd/yyyy, mmddyy, mmddyyyy, yyyy/mm/dd) and converted to standard halfword binary (WAASDATE, # of days away from 1/1/41) format. The various components of the date are also checked for legitimacy. On output, the standard halfword binary dates are converted to mm/dd/yyyy format. The null-date (x'8000') is properly converted.

2. Month/Year dates: these dates are converted to and from the RMV standard internal format for month/year dates, with proper field editing. The most common use for this type of field is registration expiration dates.

3. Zipcodes: zipcodes are converted to and from standard RMV internal form.

4. Case: for input fields, LXTABLE automatically performs lower to upper case translations. This is particularly important in the UMS guest environment where it is not possible to require that the terminals be configured for upper-case only. Further, the same process strips nonstandard display characters which may be included by some non-IBM 3270 type devices.

5. Erase-EOF: automatically converted to the null value for the defined field type.

6. Numeric: on input, fields with a generic type of numeric are edited for numeric content (or omitted), and treated as though they had been entered with right justification and left zero fill. On output, they are formatted according to several possible standardized structures.

7. Numeric Conversion: fields on the screen with a generic type of numeric clearly must contain EBCDIC data. LXTABLE allows the programmer to have his internal data definitions in one of several supported formats. The programmer may define the internal numeric fields as packed, binary or zoned. The LXTABLE will automatically convert the screen fields to/from these formats.

LXTABLE provides a generalized VALUE editing scheme for screen fields. This means that the programmer, in the LXTABLE itself, may specify lists of values which are allowed (or disallowed) for each field. LXTABLE will force the conformation to these

specifications. This facility provides a generic error message for violations, or allows the option of specifying a specific text for each field. The EXIT facility allows the user to write specialized edits that may be used on one or more screen fields.

On output, LXTABLE accepts commands (in the ZFLD byte) relative to each field. The commands and their values ( ) are:

1. ERROR: (E) highlight the field, position the cursor to the field if it is the first such field on the screen.

2. TRANSMIT: (X) the field is overlaid on the screen with its current content in the commarea, after proper conversion.

3. ERROR/TRANSMIT: (H) a combination of 1 and 2.

4. RESET: (R) the field attribute is set to its default value from the map definition.

5. TRANSMIT/CURSOR: (K) the field is overlaid on the screen with its current content in the commarea, after proper conversion. The cursor is positioned to the field if it is the first such field on the screen.

After processing these commands, LXTABLE clears the ZFLD byte.

The ZFLD byte for unprotected input fields is used to indicate to the application program if the field in question has been logically changed (value = 'C'). Logically changed does not mean the MDT is set, but rather means that, after all involved conversions, it has a different value than the value currently stored in the commarea. In the ZFLD notification of change, LXTABLE uses a stepdown technique of indicating change in current iteration ('C') and change in previous iteration ('P'). All of these notifications are superseded by any output command for the field which is issued by the application.

The TFLD serves two functions. On input, if the option PASSMDT=YES is set, the presence of the MDT for the field is shown by B'00000100' in the TFLD. Absence of the MDT is shown as X'00'. On output, any value in the TFLD is sent to the screen as the attribute byte for this iteration.

LXTABLE contains an option (AUTORES) which allows the reset-to-default attribute option to be the default for all transmit operations.

LXTABLE incorporates a concept known as field classes. The classes are DATA, KEY, and KEY-SUPPORTIVE. The purpose of this concept is to allow a screen to be partially edited. In essence, if fields are put in the class of KEY -- AND ARE LOGICALLY CHANGED -- the editing of DATA fields which follow on the screen is bypassed. This technique is important for situations where the operator may get sufficiently wound-up in errors to decide that his only option is to alter a key field and essentially restart a transaction.

LXTABLE processing is declared for a function in its UMS System Process Control Table Entry, Guest (PCTEG). The exact parameter is:

XLATE=<lx-tbl-name>

Providing the 8-character LX-table name (<lx-tbl-name>), activates LXTABLE processing at the appropriate times. As part of the NDR implementation, a technique of invoking the UMS GUEST LXTABLE processor from an ALAR application was developed. This technique is not transparent to the application (as in the UMS GUEST version), but requires the addition of a few statements to the application program.

The LXTABLE exception EXIT processing facility is a mechanism which allows the programmer to augment the LXTABLE processing on a field by field basis in designated circumstances such as edit fail, edit pass, or every time. This EXIT facility allows the execution of user code within the LXTABLE input process. This code may be used to issue errors not normally recognized by LXTABLE or to bypass errors normally recognized. These conditions may be recognized with complete access to the commarea, as well as access to the map data (normally not available with LXTABLE processing). Note that this option is only to be used in exceptional circumstances. If the editing to be done will be required in other places in the system, then a standard edit will be produced to use with the EDIT parameter. This exit facility is further described in the second part of this chapter.

The field types defined below are supported.

type	minimum length	maximum length	GSA data-type	MAP data-type
char	1	79	C	C
num	1	15	Z	Z
packed	1	15	P	Z
binary	1	9	FL4 or F	Z
date	10	10	HL2 or H	C

The LXTABLE software processes fields in the order in which they are assembled in the LXTBL entries in the LX-table module. These entries should be defined left to right, top to bottom in relation to the screen. If they are not, processing will occur in a proper manner but an error message posted will refer to the first processed field, not necessarily the first on the screen with an error.

Fields determined to be changed have their flag-byte value first set to an intermediate value (which is C'*') until all error-resolution is complete, and are then set to the proper value of C'C' before the application program is invoked. In addition to being intensified, the error fields have their MDT forced on when the error screen is shown, to insure that they are reentered.

The LXTABLE software maintains two levels of field change detection. The basic level is finding the MDT that is turned on. Once this is detected, the software compares the value in the field (after the syntax checked) with the value currently in the GSA area. If they are equal, the field is treated as though it had not been changed. For these purposes, a numeric field which has been erase-EOF'd is treated as if a value of 0 (zero) had been entered. Note that when comparing numeric fields (zoned or packed), if the GSA data area does not contain a valid sign code, the comparison is treated as not-equal (field changed).

There is a potential case where a field may be flagged as changed but logically have the same value as prior to the change. This occurs if (1) a legitimate change is detected on the field, (2) in the same demap operation an error is detected in another field, and (3) during the correction of the error the user changes the first field back to its original value.

If the application program indicates a field with error, and does not provide an attribute byte, the mapping software will supply a default attribute. The value will be intensified, unprotected, MDT-on. If the field is numeric (in the LXTBL definition) this attribute will be set as well.

The fields corresponding to mapname, maplength and cursor-position in the guest common-area should not be changed. If the map/demap software detects a changed value in maplength, it will assume that the user has prepared the correct map for transmission and will perform no output gsa-map interactions. This is an override feature available to the application which must be exercised with EXTREME care as it is possible to create an unresolvable error-conflict situation after the next receive operation.

UMSLXTBL Quick Look

To implement LXTABLE processing for an application, one must add the name of the LX-Table to the Guest PCTEG in the form of the parameter

XLATE=<lx-tbl-name>

To create the LX-Table itself the CICS map and its dsect must be created along with the GSA Common storage definitions the application program will use. With these available the LX-Table may be assembled. It requires one TYPE=START macro and, for each screen-field / GSA Common field pair, one TYPE=DETAIL macro. Descriptions of the parameter values is in the following section, Detail Description. When items are enclosed in { }, choose one of them.

```
<name>  UMSLXTBL TYPE=START,MAPNAME=<mapname>,AUTORES=YES/NO,
                 PASSMDT=YES/NO,
                 LEVEL0=YES/NO/<null>


      UMSLXTBL TYPE=DETAIL,MAPFLD=<mfldname>,
               { GSAFLD=<gfldname>,| HOSTFLD=<hfldname>, }
               OPTION1=<null>|LZFILL|RTJUST,
               KEYFLD=<null>|0|1|3|4,
               VALUE=(<vtype>,<valuelst> | R<vtype>,<value-
               pairs>),
               VALEROR=<error-num>,
               EDIT=(<editype><edit fld lst>),
               REQUIRE=YES|NO,
               EXIT=(<exitparms>)
```

Detail Description

The LXTABLE Macro

This section discusses the parameters of the LX-Table macros in detail with some examples. The LXTABLE relates mapfields to storage fields and specifies certain kinds of automatic edits and value checking that may be required. The data generated in the table allows the MAP/DEMAP Module to move the data from the mapped-in screen to common storage or vice versa with the editing requested and certain data-type conversions (eg. edited date <--> internal half-word format date). The default value for a parameter will be in **BOLD** type. Appendix A of the *UMS Programmer's Manual* has a sample generation of a small LX table showing the various parameters and features in use. Part I of the appendix is the assembly listing to illustrate the macro expansions and code generation. Part II is a hex dump of the load module from the assembly (and link). These examples are marked to show where the generated code appears in the load module. Fragments of code are included as examples with this discussion.

The LX-Table load module will always have the csect <lx-tbl-name> as the first csect. If the VALUE or EDIT parameters are specified a second csect, always called CSECT2, will be generated in the load module.

In the various illustrations, <null> means the parameter was omitted (or specified as, for example, LEVEL0=, or LEVEL0=''). The vertical bar '|' separates the valid parameter values. The <null> represents the omitted parameter, or value, as mentioned previously. A bold value is the **default** parameter value.

The UMSLXTBL TYPE=START

The first invocation, or macro in the table, must be:

<name> UMSLXTBL TYPE=START,MAPNAME=<mapname>,AUTORES=YES/**NO**,
PASSMDT=YES/**NO**,
LEVEL0=**YES**/NO/<null>

<name> is the name of the LX-Table to be created, 8-characters long

TYPE=START generates the LX-Table header.

<mapname> is the name of the related map, 7-characters long

The above are required on TYPE=START and are **not** allowed for TYPE=DETAIL.

LEVEL0=**YES** | NO allows or suppresses an mnote showing the type of edit that was requested for the individual fields.

PASSMDT=YES | **NO** pass or not pass the modified data tag through to the application program

AUTORES=YES | **NO** reset or not reset screen attributes to the map default

Sample expansion, TYPE=START

The field generations for TYPE=START are as follows:

```
                                1227 *
                                1229 UGL0260T UMSLXTBL TYPE=START,
                                                    MAPNAME=UGL0260,
                                                    LEVEL0=NO
                                1230+UGL0260T START 0
                                1231+         PUSH  PRINT
                                1232+         PRINT OFF
                                1249+         POP   PRINT
                                1250+         USING MAPINP,R4
                                1251+         USING
000000 E4C7D3F0F2F6F0E3         1252+         DC    CL8'UGL0260T'
000008 E4C7D3F0F2F6F0           1253+         DC    CL7'UGL0260'
00000F F0F661F1F861F9F1         1254+         DC    CL8'06/18/91'
000017 40                       1255+         DC    CL1' '
000018 F2F34BF5F2               1256+         DC    CL5'23.52'
00001D 00000000                 1257+         DC    XL4'0'
000021 00                       1258+         DC    BL1'00000000'  FLAGS
000022 000000                   1259+         DC    XL3'0'    RESERVED
000025 FFFF                     1260+         DC    XL2'FFFF' TEMP TRLR
```

Note: this code generation is streamlined for ease of viewing with proper code output.

Line 1252 is the name of this LX-Table. Line 1253 is the related CICS map(set) name. Note that the date and time of assembly are assembled into the header for documentation purposes (lines 1254 and 1256). The blank in line 1255 is for date-time readability. In line 1258, the field of FLAGS, the bits are used as follows (left to right):

```
1.......            reset MDTs              (AUTORES=YES)
.1......            pass MDTs through       (PASSMDT=YES)
..000000            unused
```

The fields in lines 1257 and 1259 are unused. In line 1260, the XL2'FFFF' in this and the expansions of UMSLXTBL TYPE=DETAIL, exist only if 'this' invocation of the macro is the last in the whole table. If it is not the 'last,' the expansion of the succeeding macro overlays it. If it is the last, the MAP/DEMAP Module knows it just saw the last set of fields to process.

The UMSLXTBL TYPE=DETAIL

The fields and their edits are described below. The field moves and edits proceed in the order in which the field pairs are assembled into the LX-Table. You therefore control the processing order of the screen fields. When items are enclosed in { }, choose one of them.

UMSLXTBL TYPE=DETAIL,MAPFLD=<mfldname>,
{ GSAFLD=<gfldname>,| HOSTFLD=<hfldname>, }
OPTION1=**<null>**|LZFILL|RTJUST,
KEYFLD=**<null>**|0|1|3|4,

The 5 following parms are optional and related to edits for the field. They are discussed in groups after the parameters above.

VALUE=**<null>** | (<vtype>,<value> |
R<vtype>,<valuepair>)
VALEROR=**<null>** | <error-num>,
EDIT=**<null>** | (<editype>,<edit fld list>),
REQUIRE=**NO** | YES,
EXIT=**<null>** | (<excsect>,<exreas>,<exinvn>)

MAPFLD, GASFLD, HOSTFLD

The TYPE=DETAIL describes a CICS map field-name GSA Common data field-name pair between which data will be moved with needed data-conversion and optional edits. The CICS map field-name is <mfldname> and the MAPFLD parameter is required. The GSA Common field may be in either ‘program storage’ or in HOST related storage. If it is in ‘program storage’ use the GSAFLD parameter with <gfldname> as the field name. For HOST related storage, use the HOSTFLD parm with <hfldname>. If HOSTFLD = <hfldname> is specified, the field-pair is restricted for storage-to-screen use only (&OUTONL = '1'). The ‘program’ field name is placed into the internal macro variable, &INTFLD. Data-field types are checked.

Allowed for MAPFLD are T'= C | Z.

Allowed for 'program' fields are T'= C | Z | F | H | P | X | G.

If the type is 'G', and L'= 2, the type is set to 'H', or if L'= 4, then type is set to 'F'. Anything else is an error. The valid type-pairs for <mapfld><intfld> are 'ZZ', 'ZP', 'CC', 'ZF', 'CH', and 'CX'. The general edit rules and internal macro field settings are as follows:

```
'ZZ'  Both zoned numeric - max 15 digits long, field
     lengths to be equal

          &NUMFLD   '1'       &RTN      '000'
          &VALLOW   '1'       &VALHI    L'<mapfld>
          &IFVALOK  '#'       &VALHDR   'PL&WRKLEN'
          &WRKLEN   (L'<mapfld>/2)+1

'ZP'  Zoned numeric screen, packed in 'program' L'&INTFLD
     to be LE 8, L'&INTFLD+L'INTFLD-1 to be GE L'&MAPFLD

          &NUMFLD   '1'       &RTN      '004'
          &VALLOW   '1'       &VALHI    L'<mapfld>
          &IFVALOK  '#'       &VALHDR   'PL&WRKLEN'
          &WRKLEN   (L'<mapfld>/2)+1


'CC'  Both character fields with fields to be of equal
     length and LE 79 characters long.

          &NUMFLD   '1'       &RTN      '008'
          &VALLOW   '1'       &VALHI    L'<mapfld>
          &IFVALOK  'C'       &VALHDR   'CL&WRKLEN'
          &WRKLEN   L'<mapfld>


'ZF'  Zoned numeric screen field, length LE 9, and
     L'&INTFLD to be EQ 4

          &NUMFLD   '1'       &RTN      '012'
          &VALLOW   '1'       &VALHI    L'<mapfld>
          &IFVALOK  '#'       &VALHDR   'FL4'
          &WRKLEN   '4'


'CH'  Screen date to internal fmt (no value editing
     allowed), L'&MAPFLD to be 7 or 10, L'INTFLD to be 2

          &NUMFLD   <null>    &RTN      '016' (L'10),
                                        '020' (L'7)
          &VALLOW   '4'       &VALHI    '4'
          &IFVALOK  'X'       &VALHDR   'AL2(X'''
          &WRKLEN   '2'       &DATEFLG  '1'


'CX'  Screen zip-code to internal fmt, L'&MAPFLD to be 10,
     L'INTFLD to be 6
```

```
&ZIPFLAG  '1'       &RTN      '024'
```

RTJUST and LZFILL(for numeric only) are edited. These actions are applied upon output only. They have no effect at screen-input processing time. Type flags are checked and, if &WARN NE 'N', macro emits MNOTE,0 for edit specified, if any.

If &VALUE NE <null> &VALFLAG '1'
If &EDIT NE <null> &EDTFLAG '1'

&CHKMAP = 'M$&MAPFLD'

&XYZ is set to a concatenation of all flags

&OP1FLAG&ZIPFLAG&EDTFLAG&VALFLAG&OUTONL&EXPIFLG&DATEFLG &NUMFLG

Code generation now occurs for the basic part of the macro and, if there is value or edit clauses, the macro expansion is completed. A general layout follows with some actual examples in the following section.

Code generated is: in 'main' csect

```
1    &CHKGSA   DC AL1(<dns>),AL1(&RTN)
2    &CHKMAP   DC BL1'&XYZ'
3              DC BL1'&KEYF&OP1FLAH'
4              DC AL2(&INTFLD-&BASE-2)
               DC AL2(&MAPFLD-MAPINP-3)
5              DC AL1(L'&INTFLD-1),AL1(l'&MAPFLD-1)
6              DC AL4(<v-array>)    --> value array, if used
7              DC AL4(<e-array>)    --> edit array, if used
8              DC XL2'FFFF'
```

1. First is one byte of displacement (<dns>) to the next table section, either another field or EDIT/VALUE fields for this one. Points to the X"FFFF' only if this the end of the table. The X'FFFF' is overlaid by the continuing or the next macro expansion.
   Second is one byte of field-pair type.

2. One byte of flags as follows:

`1.......`	Left Zero Fill asked
`.1......`	Zip Code fields
`..1.....`	Edit control block present
`...1....`	Value control block present
`....1...`	Out only field-pair
`.....1..`	Expiration format date
`......1.`	Internal-External date fields
`.......1`	A numeric value pair

3. One byte of flags as follows:

`0000....`	Unused
`....111.`	KEYFLD value in binary
`.......1`	Right Justify asked

4. Two bytes for program-field displacement from either UGCOMMON or UHCITEXT. Then two bytes of map-field displacement from MAPINP.

5. One byte = length of 'program'-field -1.
One byte = length of map-field -1.

6. A pointer to the value array (if specified in a VALUE parameter) in CSECT2 of the load module.

7. A pointer to the edit array (if specified in an EDIT parameter) in CSECT2 of the load module.

8. Two bytes, X'FFFF', as an end of table flag. If there is another macro after this one, its expansion overlays the X'FFFF'.

Value Checking

The value checking mechanism enables one to allow or disallow a single value, a series of individual values, or a range of values through the VALUE= parameter in the UMSLXTBL macro. The 9-digit VALEROR= parameter value directs the issuance of a specific error message if the value edit fails. If the value edit fails and no VALEROR parameter is given, a generic error message is issued. All the code generated for value editing is assembled into the csect called CSECT2. The LX-Table load module will always have the csect <name> and optionally, CSECT2 if value checking and/or editing is specified. MNOTEs are issued for any edit failure in the macro. The parameter is coded as follows:

The presence of a 9 numeric digit <errornum> in the VALEROR= clause, causes &ECODE to be set to '1' (else it is '0').

If &IFVALOK is 'X', &SIGN is set to '1', else '0'.

```
VALUE=<null> |
 (<vtype>,<valuelst> | R<vtype>,<value-pairs>)

Where: <vtype> = ALLOW | DISALLOW
       <valuelst>    = <value>,<value> ... as required
       <value-pairs> = <value2>,<value2> ... as required
       <value2>      = <value>,<value>    a pair for a
                                          range
       <value> = any value the field may assume
```

The VALUE clause may not be used on output-only fields (&HOSTFLD that causes &OUTONL to be '1') or on fields where &IFVALOK is left as 'N'. The characters in the value string are edited for correctness as numerics or a hex string as the case may be. An MNOTE issues in case of error. Two internal macro values are set based on <vtype>. The VALUE clause may be specified with the EDIT clause in the same TYPE=DETAIL macro. Note that in such a case, the VALUE clause edits are applied before those of the EDIT clause.

WARNING: An input value of 0 (zero) or blanks are passed through to the application regardless of the value clause's sub-parameters.

```
                                  &COMPTYP  &VALTYPE  Value

 <vtype> = ALLOW                  '0'       '0'       single
           DISALLOW               '1'       '0'       single
R<vtype> = ALLOW                  '0'       '1'       range
           DISALLOW               '1'       '1        range
```

Code generated is: In 'main' csect

```
          DC  AL4(<v-array>)   --> to value list in csect2
```

Code generated is: In CSECT2

```
          DC  FL4'&VALEROR'  error msg number (if any)
<vlabel>  DC  BL1'&ECODE&SIGN&COMPTYP&VALTYP'  flags
          DC  AL1(<actual field length>)
          DC  AL2(<number of list elements>)
          DC  <elements>
     &SYSECT   CSECT       return to 'main' csect
```

```
     <elements> = sets of the data type, the true field
          length and a value that represents the value(s)
          in the list.

          For example    CL3'C&C'  CL5'RMV'  ZL4'789'
```

If &VALEROR is <null>, &ECODE is '0' and the FL4 field is not generated. If &VALEROR is given and a value edit fails, that error message will be displayed on the screen with out application program intervention. If VALEROR is omitted, a generic message is sent.

Edit Checking

The EDIT clause routines perform services such as check-digit validation, ensure numerics, town-code conversions, date checks and conversions, and non-database cross-field checking. One may designate the field as required, REQUIRE=YES, to produce an error message if the field is not supplied. Code generation builds control blocks in CSECT2 of the LXTABLE. The EDIT and VALUE clauses may be specified in the same TYPE=DETAIL macro. In such a case, the VALUE clause edits are applied those of the EDIT clause.

```
     The form of the parameters are:

     REQUIRE=NO | YES    if Yes and no data, error
                         if YES, &REQ = 1 ( else = 0 )


     EDIT=(<editype><edit fld lst>)

     <editype> a literal from the Edit Type Table on page
          5-18  that tells UGZ0006P(map/demap pgm) what to
          do

     <edit fld lst> = <fldn> | <fldn>,<fldn+1> ...

          <fldn> etc is the name(s) of the field(s) to be
               edited (the table shows the requirements)



     Code generated: in csect CSECT2


<E-label> DC  AL1(<e-label - E-label)  length of this array
          DC  AL1(&REQ&MODE)       &MODE unused, = 0
          DC  AL2(&TYPE)           edit to do, 0-19
          DC  AL4(<lxptr>)         --> block in main table
<e-label> DC  0AL1                 label for length calc
```

Edit Type Table

Note: The number of fields required is absolute, too few or too many errors out the macro with an MNOTE. If the # of fields required is 1, the edit applies to the current macro being expanded. If more than 1 field is required, they must be in the current macro or already expanded. If any of the fields come later, the macro errors out. The fields must have the type and length specified or the macro fails.

Edit Type	Literal	Description	# Fields Required
0	LICNO	MA lic #, 9-byte	1 CL9 license number
1	VIN	VIN, 17 byte, chkdgt	1 CL17 the VIN
2	TCODE	Town Code	1 PL2 internal town code
3	ICODE	Insurance Code	1 PL2 internal Co. id.
4	REGNO	Registration	3 CL3 plate type CL1 plate color CL7 registration #
5	MADDR	Optional Address	5 (see type 6)
6	RADDR	Required Address	5 CL20 address line 1 CL20 address line 2 CL15 city/town CL2 state XL6 zip code
7	MLICN	License Number, 9-byte	2 CL9 license number CL2 state
8	TITLE#	Title Number	1 CL8 the title number
9	PDATE	Policy Date Ranges	2 H effective date H expiration date
10	PNAME	Person Name	3 CL16 last name CL12 first name CL8 middle name
11	ILOCN	Insurance Co. Location	2 PL2 Co. code, internal PL2 town code, internal
12	MLINC2	License #, 25-byte	2 CL25 license #(non MA) CL2 state
13	SOCNO	Social Security #	1 CL9 SSAN (CL / ZL)
14	MONEY	Money edit	1 CL3 to CL21, money field
15	LICNO/S	Lic # w/'S' allowed	1 CL9 license #
16	MLICN2 /NOMA	License, 25-byte w/ no ST default	2 CL25 license # CL2 state
17	AGE1511	Age Edit, dob/permit	1 H dob, internal fmt
18	AGE1600	Age Edit, dob/license	1 H dob, internal fmt
19	MLICN2/XX	Lic # 25 byte w/ XXnolicense	2 CL25 license # CL2 state

Sample Map Source Fragment

Note the use of the UMSHEADR macro to form the standard heading on the map (screen). See the assembled sample to view the generation of FLD0001 through FLD0004 and the (standard) header literals. See the sample map dsect (following) for the commarea definition. You may relate the use of the field names in the LX-Table macros for the generation of the 'translate table.'

```
UGL0260  UMSHEADR LTH,'LICENSE TRANSACTION HISTORY'
         DFHMDF POS=(004,002),LENGTH=0001,INITIAL=' '
         DFHMDF POS=(005,001),LENGTH=0010,                     X
      INITIAL='LICENSE #:'
FLD0005  DFHMDF POS=(005,012),LENGTH=0025,                     X
               JUSTIFY=(LEFT,BLANK),                           X
               ATTRB=(UNPROT,IC)
         DFHMDF POS=(005,038),LENGTH=0006,                     X
               INITIAL='STATE:'
FLD0006  DFHMDF POS=(005,045),LENGTH=0002,                     X
               JUSTIFY=(LEFT,BLANK),                           X
               ATTRB=(UNPROT)
         DFHMDF POS=(005,048),LENGTH=0004,                     X
               INITIAL='SS#:'
FLD0007  DFHMDF POS=(005,053),LENGTH=0009,INITIAL=' ',         X
                ATTRB=UNPROT
         DFHMDF POS=(005,063),LENGTH=0001
         DFHMDF POS=(006,004),LENGTH=0007,                     X
               INITIAL='NAME L:'
FLD0010  DFHMDF POS=(006,012),LENGTH=0016,                     X
               JUSTIFY=(LEFT,BLANK),                           X
               ATTRB=(UNPROT)
```

Sample Assembler Map Dsect

```
                                 1111          COPY  UGL0260C
000FEA                    006F8  1112          ORG   UGCUSERA
                                 1113 * START OF GSA STORAGE
                                 1114 LHLIC#   GFLD CL25     LH-LIC NUMB
0006F8                           1115+LHLIC#_Z DS    CL1      ZFIELD
0006F9                           1116+LHLIC#_T DS    CL1      TFIELD
0006FA                           1117+LHLIC#   DS    CL25     DATA FIELD

                                 1118 LHSTATE  GFLD CL2       STATE
000713                           1119+LHSTATE_Z DS   CL1      ZFIELD
000714                           1120+LHSTATE_T DS   CL1      TFIELD
000715                           1121+LHSTATE  DS    CL2      DATA FIELD

                                 1122 LHSSN    GFLD CL9       SSN
000717                           1123+LHSSN_Z  DS    CL1      ZFIELD
000718                           1124+LHSSN_T  DS    CL1      TFIELD
000719                           1125+LHSSN    DS    CL9      DATA FIEL
                                 1126 LHLNAME  GFLD CL16      LAST-NAME
000722                           1127+LHLNAME_Z DS   CL1      ZFIELD
000723                           1128+LHLNAME_T DS   CL1      TFIELD
000724                           1129+LHLNAME  DS    CL16     DATA FIELD
                                 1130 LHFNAME  GFLD CL12      FIRST-NAME
000734                           1131+LHFNAME_Z DS   CL1      ZFIELD
000735                           1132+LHFNAME_T DS   CL1      TFIELD

          Data ommitted for brevity.

                                 1209 LHLINE44 GFLD CL79     DETAIL LINE
000B7B                           1210+LHLINE44_Z DS  CL1     ZFIELD
000B7C                           1211+LHLINE44_T DS  CL1     TFIELD
```

```
000B7D                  1212+LHLINE44 DS    CL79    DATA FIELD
                        1213 LHLINE45 GFLD CL79     DETAIL LINE
000BCC                  1214+LHLINE45_Z DS  CL1     ZFIELD
000BCD                  1215+LHLINE45_T DS  CL1     TFIELD
000BCE                  1216+LHLINE45 DS    CL79    DATA FIELD
000C1D                  1217 PHASE    DS    XL1     PHASE COU
000C20                  1218 PERSSURR DS    F       CURRENT P
000C24                  1219 NUMOBREF DS    H       TOTAL BRE
000C26                  1220 BREF1    DS    H       NUMBER OF
000C28                  1221 BREFX    DS    H       NUMBER OF
000C2A                  1222 PAGEC    DS    XL1     CURRENT P
000C2B                  1223 PAGELIST DS    256XL1  FIRST REC
000D2B                  1224 NOTFOUND DS    XL1     FLAG, NOT
                 006FA  1225 LHGSA    EQU  LHLIC#,*-LHLIC#,C'X'GSA
000D2C           00FEA  1226          ORG
```

LXTABLE Assembled Example

```
                                          EXTERNAL SYMBOL DICTIONARY
SYMBOL   TYPE  ID  ADDR  LENGTH  LD ID FLAGS
LXTABTST  SD 0001 000000 000117         00
CSECT2    SD 0002 000118 00006C         00
CSECT3    SD 0003 000188 00000D         00
UGLTEST1  SD 0004 000198 000056         00

        LXTABTST - UMS SAMPLE LXTBL DEFINITION
LOC  OBJECT CODE    ADDR1 ADDR2  STMT SOURCE STATEMENT                  ASM H V 02 15.47 07/09/91

                                     2 * C&C ASSOCIATES 1989                                    00100000
                                     3          COPY  ECELXMAC                                  00110000
                                     4          PUSH  PRINT                                     00010000
                                     5          PRINT OFF                                       00020000
                                   996          POP   PRINT                                     09700000
                                   998          UMSHEADR TYPE=DSECT                             00140000
000000                             999+MAPINP   DSECT                                           01-UMSHE
                                  1000+* HEADER OF INPUT-MAP/OUTPUT MAP AREA                    01-UMSHE
000000                            1001+         DS    CL12            HEADER                   01-UMSHE
00000C                            1002+         DS    CL3    ADDRESS/ATTRIBUTES FOR DATE       01-UMSHE
00000F                            1003+MIDATE   DS    CL10   DATE TEXT                         01-UMSHE
000019                            1004+         DS    CL3    ADDRESS/ATTRIBUTES FOR TIME       01-UMSHE
00001C                            1005+MITIME   DS    CL5    TIME TEXT                         01-UMSHE
000021                            1006+         DS    CL3    ADDRESS/ATTRIBUTES FOR FUNC       01-UMSHE
000024                            1007+MIFUNC   DS    CL4    FUNC TEXT                         01-UMSHE
000028                            1008+         DS    CL3    ADDRESS/ATTRIBUTES FOR TEXT       01-UMSHE
00002B                            1009+MITEXT   DS    CL50   MESG TEXT                         01-UMSHE
                           0005D  1010+MI$SLEN  EQU   *-MAPINP BASE LENGTH                     01-UMSHE
00005D                            1011          DS    CL3             BMS DATA                 00150000
000060                            1012 MCHAR    DS    CL17            CHARACTER FIELD          00160000
000071                            1013          DS    CL3             BMS DATA                 00170000
```

```
000074  1014 MCHAR2   DS   CL10          CHARACTER FIELD     00180000
00007E  1015          DS   CL3           BMS DATA            00190000
000081  1016 MLAST    DS   CL16          CHARACTER FIELD     00200000
000091  1017          DS   CL3           BMS DATA            00210000
000094  1018 MFIRST   DS   CL12          CHARACTER FIELD     00220000
0000A0  1019          DS   CL3           BMS DATA            00230000
0000A3  1020 MMIDDLE  DS   CL8           CHARACTER FIELD     00240000
0000AB  1021          DS   CL3           BMS DATA            00250000
0000AE  1022 MNUM1    DS   ZL5           NUMERIC FIELD       00260000
0000B3  1023          DS   CL3           BMS DATA            00270000
0000B6  1024 MNUM2    DS   ZL7           NUMERIC FIELD       00280000
0000BD  1025          DS   CL3           BMS DATA            00290000
0000C0  1026 MNUM3    DS   ZL4           NUMERIC FIELD       00300000
0000C4  1027          DS   CL3           BMS DATA            00310000
0000C7  1028 MDATE    DS   CL10          DATE FIELD          00320000
0000D1  1029          DS   CL3           BMS DATA            00330000
0000D4  1030 MDATE2   DS   CL07          DATE FIELD          00340000
0000DB  1031          DS   CL3           BMS DATA            00350000
0000DE  1032 MDATEA   DS   CL10          DATE FIELD          00360000
0000E8  1033          DS   CL3           BMS DATA            00370000
0000EB  1034 MDATEB   DS   CL10          DATE FIELD          00380000
0000F5  1035          DS   CL3           BMS DATA            00390000
0000F8  1036 MPREF    DS   CL03          PREFIX              00400000
0000FB  1037          DS   CL3           BMS DATA            00410000
0000FE  1038 MCOLOR   DS   CL01          COLOR               00420000
0000FF  1039          DS   CL3           BMS DATA            00430000
000102  1040 MREGNO   DS   CL07          REGNO               00440000
000109  1041          DS   CL3           BMS DATA            00450000
00010C  1042 MZIPC    DS   CL10          ZIPCODE             00460000
000116  1043          DS   CL3           BMS DATA            0470000
000119  1044 MTITLE#  DS   CL8           TITLE#              00480000
000121  1045          DS   CL3           BMS DATA            00490000
000124  1046 MLICN25  DS   CL25          LICNO               00500000
00013D  1047          DS   CL3           BMS DATA            00510000
```

```
000140                          1048 MSTATE25 DS    CL2             STATE                    00520000

                                1050          COPY  UGZCOMMA                                 00540000

                                1052 * UNINSURED MOTORIST. COMMON-AREA, GUEST SIDE.          00020000
000000                          1053 UGCOMMON DSECT                                          00030000
000000                          1054 UGCPROTO DS    XL64      SYSTEM PROTOCALL DATA          00040000
000040                 00000    1055          ORG   UGCPROTO                                 00050000
                                1056 * WITH TWO EXCEPTIONS, THE PROTOCALL AREA IS
                                     * RESERVED FOR THE CONTROL                              00060000
                                1057 * SOFTWARE. APPLICATIONS MAY USE THE
                                     * DUBLE-WORD TEMP AND SHOULD NOTE                       00070000
                                1058 * THE QUALIFICATION ABOUT PREVIOUS MAPNAME              00080000
000000                          1059 UGCDTEMP DS    D GENERAL PURPOSE DOUBLE-WORD TEMP       00090000
000008                          1060 UGCCLENG DS    H      LENGTH FOR USE WITH XCTL.         00100000
00000A                          1061 UGCRDLEN DS    H      INQUIRY/REPLY DETAIL LENGTH       00110000
00000C                          1062 UGCCURFN DS    CL4    CURRENT FUNCTION-NAME             00120000
000010                          1063 UGCALLON DS    XL1    FLAG BYTE ALL BITS INIT ON        00130000
                       00001    1064 UGCMAPFI EQU   B'00000001' MAP/DEMAP OPER. DEMAP        00140000
                       00002    1065 UGCMAPFO EQU   B'00000010' MAP/DEMAP OPER MAP           00150000
000011                          1066          DS    CL1                                      00160000
                                1271 *                                                       02030000
                       006F8    1272 UGCUSERA EQU   *        THE USER-PROGRAM AREA           02040000
0006F8                          1273          DS    2290XL1         PUSH TO PAGE             02050000
00FEA  1274 UGCOMTLN EQU   *-UGCOMMON      TOTAL LENGTH                  02060000
                                1275 * THE LENGTH OF THIS AREA IS TAKEN FROM
                                     * THE PROCESS-CONTROL TABLE                             02070000
                                1276 * BUT WILL NOT BE LESS THAN THE VALUE OF
                                     * THE ABOVE EQUATE AT THE LAST                          02080000
                                1277 *  ASSEMBLY OF UGZ0015P                                 02090000
000FEA                 006F8    1278          ORG   UGCUSERA                                 00550000
                                1279 * START OF GSA STORAGE                                 00560000
0006F8                          1280          DS    CL1,CL1         ZFLD, TFLD               00570000
0006FA                          1281 GTITLE#  DS    CL8             TITLE#                   00580000
```

```
000702                  1282           DS    CL1,CL1         ZFLD, TFLD            00590000
000704                  1283 GDATE     DS    HL2  BINARY-DATE(MUST BE H OR HL2)    00600000
000706                  1284           DS    CL1,CL1         ZFLD, TFLD            00610000
000708                  1285 GDATE2    DS    HL2  BINARY-DATE(MUST BE H OR HL2)    00620000
00070A                  1286           DS    CL1,CL1         ZFLD, TFLD            00630000
00070C                  1287 GDATEA    DS    HL2  BINARY-DATE(MUST BE H OR HL2)    00640000

                        1289 GDATEB    GFLD  HL2                                   00660007
00070E                  1290+GDATEB_Z  DS    CL1             ZFIELD                01-00873
00070F                  1291+GDATEB_T  DS    CL1             TFIELD                01-00874
000710                  1292+GDATEB    DS    HL2             DATA FIELD            01-00875
                        1294 GNUMB     GFLD  FL4                                   00680007
000712                  1295+GNUMB_Z   DS    CL1             ZFIELD                01-00873
000713                  1296+GNUMB_T   DS    CL1             TFIELD                01-00874
000714                  1297+GNUMB     DS    FL4             DATA FIELD            01-00875

                        1299 *         DS    CL1,CL1        ZFLD, TFLD             00700007
                        1300 *GDATEB   DS    HL2  BINARY-DATE(MUST BE H, HL2)     00710007
                        1301 *         DS    CL1,CL1         ZFLD, TFLD            00720007
                        1302 *GNUMB    DS    FL4  BINARY ITEM(MUST BE F, FL4)     00730007
000718                  1304           DS    CL1,CL1         ZFLD, TFLD            00750000
00071A                  1305 GNUMP     DS    PL5  PACKED FIELD (WATCH LENGTH)     00760000
00071F                  1306           DS    CL1,CL1         ZFLD, TFLD            00770000
000721                  1307 GNUMZ     DS    ZL5  NUMERIC (MUST = LEN OF SOURCE)  00780000
000726                  1308           DS    CL1,CL1         ZFLD, TFLD            00790000
000728                  1309 GCHAR     DS    CL17   CHAR.(MUST = LEN OF SOURCE)    0080000
000739                  1310           DS    CL1,CL1         ZFLD, TFLD            0081000
00073B                  1311 GLAST     DS    CL16   CHAR.(MUST = LEN OF SOURCE)    0082000
00074B                  1312           DS    CL1,CL1         ZFLD, TFLD            0083000
00074D                  1313 GFIRST    DS    CL12   CHAR.(MUST = LEN OF SOURCE)    0084000
000759                  1314           DS    CL1,CL1         ZFLD, TFLD            0085000
00075B                  1315 GMIDDLE   DS    CL8    CHAR.(MUST = LEN AS SOURCE)    0086000
000763                  1316           DS    CL1,CL1         ZFLD, TFLD            0087000
000765                  1317 GPREF     DS    CL3    CHAR.(MUST = LEN AS SOURCE)    0088000
```

```
000768                  1318           DS    CL1,CL1         ZFLD, TFLD              0089000
00076A                  1319 GCOLOR    DS    CL1    CHAR.(MUST = LEN AS SOURCE)      0090000
00076B                  1320           DS    CL1,CL1         ZFLD, TFLD              0091000
00076D                  1321 GREGNO    DS    CL7    CHAR.(MUST = LEN AS SOURCE)      0092000
000774                  1322           DS    CL1,CL1         ZFLD, TFLD              0093000
000776                  1323 GZIPC     DS    XL6    CHAR.(MUST = LEN AS SOURCE)      0094000
00077C                  1324           DS    CL1,CL1         ZFLD, TFLD              0095000
00077E                  1325 GICODE1   DS    CL30   CHAR.(MUST = LEN AS SOURCE)      0096000
00079C                  1326           DS    CL1,CL1         ZFLD, TFLD              0097000
00079E                  1327 GICODE2   DS    CL18   CHAR.(MUST = LEN AS SOURCE)      0098000
0007B0                  1328           DS    CL1,CL1         ZFLD, TFLD              0099000
0007B2                  1329 GXCODEA   DS    PL2    CHAR.(MUST = LEN AS SOURCE)      0100000
0007B4                  1330           DS    CL1,CL1         ZFLD, TFLD              0101000
0007B6                  1331 GSTATE25  DS    CL2    CHAR.(MUST = LEN AS SOURCE)      0102000
0007B8                  1332           DS    CL1,CL1         ZFLD, TFLD              0103000
0007BA                  1333 GLICN25   DS    CL25   CHAR.(MUST = LEN AS SOURCE)      0104000
                        1335           COPY  UHZCOMMA                               01060000

                        1337 ***** UMS COMMON-AREA, HOST SIDE. ** ECE ***            00020000
000000                  1338 UHCOMMON  DSECT                                         00030000
000000                  1339 UHCPROTO  DS    XL64        SYSTEM PROTOCALL DATA       00040000
000040           00000  1340           ORG   UHCPROTO                               00050000
000000                  1341 UHCDTEMP  DS    D   GENERAL PURPOSE DBL-WORD TEMP.     00060000
                 00000  1342 UHCPARM1  EQU   UHCDTEMP,4  SPECIAL-PURPOSE PARM-1     00070000
                 00004  1343 UHCPARM2  EQU   UHCDTEMP+4,4 SPECIAL-PURPOSE PARM-2    00080000
000008                  1344 UHCCLENG  DS    H      LENGTH FOR USE WITH XCTL.       00090000
00000A                  1345 UHCRDLEN  DS    H      RESPONSE DETAIL LENGTH          00100000
00000C                  1346 UHCEFLAG  DS    XL1    ENTRY-REASON.
                                                        X'FF'=DUPKEY RESOLUTION     00110000
00000D                  1347           DS    XL1            FILLER                  00120000
00000E                  1348 UHCDUPNM  DS    CL4  HOLD DUPKEY NAME BY DISPATCHER    00130000
000012                  1349 UFHPINS2  DS    10PL2  ADDED INSURNACE-CODE LIST       00140000
000026                  1350 UHPCTIND  DS    H    HOST PCT INDEX. X'FFFF' = NONE    00150000
000028                  1351 UHSTARTT  DS    D    R/T CLOCK AT UHZ0001P INITI.      00160000
```

```
000030                    00040  1352          ORG                                                 00170000
000040                           1353 UHCMODNA DS    XL192           MODULE NAME AREA             00180000
000100                    00040  1354          ORG   UHCMODNA                                      00190000
000040                           1355 UHCOUTPR DS    CL8   O/P-SIDE PROTOCALL PROCESSOR          00200000
000048                           1356 UHCMODX  DS    0CL(4*8)   MODULES FROM PCTE ENTRY          00210000
000048                           1357 UHCMOD1  DS    CL8   1ST APPLIC. PROCESSOR MODULE          00220000
000050                           1358 UHCMOD2  DS    CL8   2ND APPLIC. PROCESSOR MODULE          00230000
000A30                    00C00  1452          ORG                                                 01170000
                          00C00  1453 UHCOMLEN EQU   *-UHCOMMON      COMMON LENGTH                01180000
000C00                    0023E  1454          ORG   UHCIDATA                                     01070000
                                 1455 * OUTPUT-ONLY HOSTAREA FIELDS GO HERE.
                                      *    THERE ARE NO ZFLD OR TFLD BYTES.                       01080000
                                 1456 * THE AREAS ARE NOT SENT
                                      *    IF THEY ARE LOW-VALUES.                                01090000
00023E                           1457 HCHAR    DS    CL10                                        01100000

000248                           1458 HXCODEB  DS    PL2   CHAR.(MUST BE = LEN AS SOURCE)       01110000

                                 1461 * DEFINE THE TABLE. SPECIFY CSECT-NAME IN
                                      *   IDENTIFICATION FIELD                                   01140008
                                 1462 *  AND MAP-NAME                                           01150008
                                 1464 LXTABTST UMSLXTBL TYPE=START,MAPNAME=UGLTEST,            X01170009
                                                        LEVEL0=NO
000000                           1465+LXTABTST START 0                                          01-00090
                                 1466+         PUSH  PRINT                                      01-00091
                                 1467+         PRINT OFF                                        01-00092
                                 1484+         POP   PRINT                                      01-00110
                          00000  1485+         USING MAPINP,R4                                  01-00111
                          00000  1486+         USING UGCOMMON,R9                                01-00112
000000 D3E7E3C1C2E3E2E3          1487+         DC    CL8'LXTABTST'                              01-00113
000008 E4C7D3E3C5E2E3            1488+         DC    CL7'UGLTEST'                               01-00114
00000F F0F761F0F961F9F1          1489+         DC    CL8'07/09/91',CL1' '                       01-00115
000018 F1F54BF4F7                1490+         DC    CL5'15.47'                                 01-00116
00001D 00000000                  1491+         DC    XL4'0'                                     01-00117
```

```
000021 00                       1492+         DC    BL1'00000000'              FLAGS        01-00118
000022 000000                   1493+         DC    XL3'0'          RESERVED                01-00119
000025 FFFF                     1494+         DC    XL2'FFFF'       TEMP TRAILER            01-00120
                                1496 *                                                      01190008
                                1497 * EXAMPLE: CHARACTER TO CHARACTER, FIELDS MUST BE SAME LENGTH
                                1499     UMSLXTBL MAPFLD=MCHAR,GSAFLD=GCHAR,EDIT=VIN,KEYFLD=1
000027                  00025   1500+         ORG   *-2                                     01-00345
000025 0E08                     1501+G$GCHAR  DC    AL1(L0005-*),AL1(008)                   01-00346
000027 20                       1502+M$MCHAR  DC    BL1'00100000'                           01-00347
000028 02                       1503+         DC    BL1'00010'                              01-00348
000029 0726                     1504+         DC    AL2(GCHAR-UGCOMMON-2)                   01-00349
00002B 005D                     1505+         DC    AL2(MCHAR-MAPINP-3)                     01-00350
00002D 1010                     1506+         DC    AL1(L'GCHAR-1),AL1(L'MCHAR-1)           01-00351
00002F                          1507+E$GCHAR  DS    0AL1            DEFINE REFERENCE        02-00531
00002F 00000118                 1508+         DC    AL4(E0006)      POINT TO EDIT ARRAY     02-00535
000118                          1509+CSECT2   CSECT                                         02-00536
000118 08                       1510+E0006    DC    AL1(L0006-*)    LENGTH OF ARRAY         02-00537
000119 000001                   1511+         DC    AL1(00),AL2(1)  MODE FLAG BITS / TYPE   02-00755
00011C 00000025                 1512+         DC    AL4(G$GCHAR)    REFERENCE TO LXTBL BLOCK
000120                          1513+L0006    DS    0AL1            END OF ARRAY            02-00770
000033                          1514+LXTABTST CSECT                                         02-00771
000033 FFFF                     1515+L0005    DC    XL2'FFFF'                               01-00361
                                1517 *                                                      01240008
                                1518 * EXAMPLE: CHARACTER TO CHARACTER,
                                     *           FIELDS MUST BE SAME LENGTH, OUTPUT         01250008
                                1520          UMSLXTBL MAPFLD=MCHAR2,HOSTFLD=HCHAR          01270008
000035                  00033   1521+         ORG   *-2                                     01-00345
000033 0A08                     1522+         DC    AL1(L0007-*),AL1(008)                   01-00346
000035 08                       1523+M$MCHAR2 DC    BL1'00001000'                           01-00347
000036 00                       1524+         DC    BL1'00000'                              01-00348
000037 003C                     1525+         DC    AL2(HCHAR-UHCITEXT-2)                   01-00349
000039 0071                     1526+         DC    AL2(MCHAR2-MAPINP-3)                    01-00350
00003B 0909                     1527+         DC    AL1(L'HCHAR-1),AL1(L'MCHAR2-1)          01-00351
00003D FFFF                     1528+L0007    DC    XL2'FFFF'                               01-00361
```

```
                                 1530 *                                                 01290008
                                 1531 * EXAMPLE: NUMERIC TO NUMERIC, FIELDS MUST BE SAME LENGTH
                                 1533     UMSLXTBL MAPFLD=MNUM1,GSAFLD=GNUMZ,
                                                   VALUE=(RALLOW,1,2,3,45670)           01320008
00003F                     0003D 1534+         ORG   *-2                                01-00345
00003D 0E00                      1535+G$GNUMZ  DC    AL1(L0008-*),AL1(000)             01-00346
00003F 11                        1536+M$MNUM1  DC    BL1'00010001'                     01-00347
000040 00                        1537+         DC    BL1'00000'                        01-00348
000041 071F                      1538+         DC    AL2(GNUMZ-UGCOMMON-2)             01-00349
000043 00AB                      1539+         DC    AL2(MNUM1-MAPINP-3)               01-00350
000045 0404                      1540+         DC    AL1(L'GNUMZ-1),AL1(L'MNUM1-1)     01-00351
000047 00000120                  1541+         DC    AL4(V0009)      POINT TO VALUE LIST 02-00431
000120                           1542+CSECT2   CSECT                                    02-00432
000120 01                        1543+V0009    DC    BL1'0001'       FLAGS             02-00436
000121 03                        1544+         DC    AL1(3)          TRUE ENTRY LENGTH 02-00437
000122 0004                      1545+         DC    AL2(4)          NUMBER OF LIST ELEMENTS02-00438
000124 00001C                    1546+         DC    PL3'1'                            02-00473
000127 00002C                    1547+         DC    PL3'2'                            02-00473
00012A 00003C                    1548+         DC    PL3'3'                            02-00473
00012D 45670C                    1549+         DC    PL3'45670'                        02-00473
00004B                           1550+LXTABTST CSECT                                    02-00478
00004B FFFF                      1551+L0008    DC    XL2'FFFF'                         01-00361

                                 1553 *                                                 01340008
                                 1554 * EXAMPLE: NUMERIC TO PACKED, OUTPUT MUST BE LONG-ENOUGH
                                 1556          UMSLXTBL MAPFLD=MNUM2,GSAFLD=GNUMP,         X
                                                   OPTION1=LZFILL,EXIT=USER01,KEYFLD=4 01370008
000000                           1557+EXITDATA DSECT                                    02-00909
000000                           1558+EXITCODE DS    AL2      UNIQUE IDENTIFIER CODE SC1290 02-00910
000002                           1559+EXITWHYS DS    XL1      SUMMATION OF ENTRY REASONS   02-00911
000003                           1560+EXITRETN DS    AL4      RETURN ADDRESS         SC1290 02-00912
000007                           1561+EXITWHYX DS    XL1      REASON FOR THIS ENTRY  SC1290 02-00913
                           00001 1562+EXRVALOK EQU   B'00000001'     VALUE IS OK     SC1290 02-00914
                           00002 1563+EXRVALNG EQU   B'00000010'     VALUE IS NOT OK SC1290 02-00915
```

```
                          00004  1564+EXREDTOK EQU   B'00000100'     EDIT IS OK      SC1290 02-00916
                          00008  1565+EXREDTNG EQU   B'00001000'     EDIT IS NOT OK  SC1290 02-00917
                          00010  1566+EXRALWAY EQU   B'00010000' ALWAYS ENTER UNLESS FMT ERR02-00918
000008                           1567+EXITUSER DS    0XL5   USER PARMS               SC1290 02-00919
000008                           1568+EXITACTN DS    XL1    USER ACTION REQUEST      SC1290 02-00920
                          00000  1569+EXANOACT EQU   B'00000000' CONTINUE, NO ACTION SC1290 02-00921
                          00001  1570+EXANOERR EQU   B'00000001' CONTINUE, NO ERROR  SC1290 02-00922
                          00002  1571+EXAWIERR EQU   B'00000010' CONTINUE, POST ERR. SC1290 02-00923
000009                           1572+EXITEROR DS    XL4         USER ERROR CODE     SC1290 02-00924
00004D                           1573+LXTABTST CSECT                                        02-00925
                                 1574+UMSLXTBL/EXIT REASON DEFAULTED TO ALL (FF)            02-00933
                                 1575+UMSLXTBL/EXIT LOCATION-CODE DEFAULTED TO 65535        02-00965
00004D                    0004B  1576+         ORG   *-2                                    01-00345
00004B 1104                      1577+G$GNUMP  DC    AL1(L0010-*),AL1(004)                 01-00346
00004D 81                        1578+M$MNUM2  DC    BL1'10000001'                         01-00347
00004E 18                        1579+         DC    BL1'11000'                            01-00348
00004F 0718                      1580+         DC    AL2(GNUMP-UGCOMMON-2)                 01-00349
000051 00B3                      1581+         DC    AL2(MNUM2-MAPINP-3)                   01-00350
000053 0406                      1582+         DC    AL1(L'GNUMP-1),AL1(L'MNUM2-1)         01-00351
000055 00000198FFFFFF            1583+         DC    AL4(USER01),XL1'FF',AL2( 5535)       X01-00359
                                    +                              EXIT STRING VALUE
00005C FFFF                      1584+L0010    DC    XL2'FFFF'                             01-00361
                                 1586 *                                                     01390008
                                 1587 * EXAMPLE: NUMERIC TO BINARY, I/P MUST BE 1-9 DIGITS  01400008
                                 1588 *  OUTPUT MUST BE F OR FL4                            01410008
                                 1589          UMSLXTBL MAPFLD=MNUM3,GSAFLD=GNUMB,OPTION1=RTJUST,
                                                     KEYFLD=1
00005E                    0005C  1590+         ORG   *-2                                    01-00345
00005C 0A0C                      1591+G$GNUMB  DC    AL1(L0012-*),AL1(012)                 01-00346
00005E 01                        1592+M$MNUM3  DC    BL1'00000001'                         01-00347
00005F 03                        1593+         DC    BL1'00011'                            01-00348
000060 0712                      1594+         DC    AL2(GNUMB-UGCOMMON-2)                 01-00349
000062 00BD                      1595+         DC    AL2(MNUM3-MAPINP-3)                   01-00350
000064 0303                      1596+         DC    AL1(L'GNUMB-1),AL1(L'MNUM3-1)         01-00351
```

```
000066 FFFF                    1597+L0012    DC    XL2'FFFF'                              01-00361
                               1599 *                                                     01440008
                               1600 * EXAMPLE: DATE, INPUT MUST BE CL10, OUTPUT MUST BE H OR HL2
                               1602          UMSLXTBL MAPFLD=MDATE,GSAFLD=GDATE,EDIT=AGE1600,
                                                 VALUE=(ALLOW,8000,0000,FFFF,7FFF),
                                                 EXIT=(USER02,01,99)                    01480008
000068                  00066  1603+          ORG   *-2                                01-00345
000066 1910                    1604+G$GDATE  DC    AL1(L0013-*),AL1(016)                01-00346
000068 32                      1605+M$MDATE  DC    BL1'00110010'                        01-00347
000069 10                      1606+         DC    BL1'10000'                           01-00348
00006A 0702                    1607+         DC    AL2(GDATE-UGCOMMON-2)                01-00349
00006C 00C4                    1608+         DC    AL2(MDATE-MAPINP-3)                  01-00350
00006E 0109                    1609+         DC    AL1(L'GDATE-1),AL1(L'MDATE-1)        01-00351
000070 00000130                1610+         DC    AL4(V0015)      POINT TO VALUE LIST  02-00431
000130                         1611+CSECT2   CSECT                                      02-00432
000130 04                      1612+V0015    DC    BL1'0100'       FLAGS                02-00436
000131 02                      1613+         DC    AL1(2)          TRUE ENTRY LENGTH    02-00437
000132 0004                    1614+         DC    AL2(4)          NUMBER OF LIST ELEMENTS02-00438
000134 0000                    1615+         DC    AL2(X'8000'+32768)                   02-00473
000136 8000                    1616+         DC    AL2(X'0000'+32768)                   02-00473
000138 7FFF                    1617+         DC    AL2(X'FFFF'+32768)                   02-00473
00013A FFFF                    1618+         DC    AL2(X'7FFF'+32768)                   02-00473
000074                         1619+LXTABTST CSECT                                      02-00478
000074                         1620+E$GDATE  DS    0AL1            DEFINE REFERENCE     02-00531
000074 0000013C                1621+         DC    AL4(E0016)      POINT TO EDIT ARRAY  02-00535
00013C                         1622+CSECT2   CSECT                                      02-00536
00013C 08                      1623+E0016    DC    AL1(L0016-*)    LENGTH OF ARRAY      02-00537
00013D 000012                  1624+         DC    AL1(00),AL2(18) MODE FLAG BITS / TYPE 02-00755
000140 00000066                1625+         DC    AL4(G$GDATE)    REFERENCE TO LXTBL BLK 02-00766
000144                         1626+L0016    DS    0AL1            END OF ARRAY         02-00770
000078                         1627+LXTABTST CSECT
000078 000001B8010063          1628+         DC    AL4(USER02),XL1'01',AL2(99)          X01-00359
                                  +                                EXIT STRING VALUE
00007F FFFF                    1629+L0013    DC    XL2'FFFF'                            01-00361
```

```
                                 1631 * EXAMPLE: EXPDATE, INPUT MUST BE CL7,
                                      *                   OUTPUT MUST BE H OR HL2
                                 1633           UMSLXTBL MAPFLD=MDATE2,GSAFLD=GDATE2          01520008
000081                    0007F  1634+          ORG   *-2                                     01-00345
00007F 0A14                      1635+G$GDATE2 DC    AL1(L0017-*),AL1(020)                  01-00346
000081 04                        1636+M$MDATE2 DC    BL1'00000100'                          01-00347
000082 00                        1637+         DC    BL1'00000'                             01-00348
000083 0706                      1638+         DC    AL2(GDATE2-UGCOMMON-2)                 01-00349
000085 00D1                      1639+         DC    AL2(MDATE2-MAPINP-3)                   01-00350
000087 0106                      1640+         DC    AL1(L'GDATE2-1),AL1(L'MDATE2-1)        01-00351
000089 FFFF                      1641+L0017    DC    XL2'FFFF'                              01-00361

                                 1643 * EXAMPLE: COMPLETE SET FOR REGISTRATION               01540008
                                 1645           UMSLXTBL MAPFLD=MPREF,GSAFLD=GPREF            01560008
00008B                    00089  1646+          ORG   *-2                                     01-00345
000089 0A08                      1647+G$GPREF  DC    AL1(L0018-*),AL1(008)                  01-00346
00008B 00                        1648+M$MPREF  DC    BL1'00000000'                          01-00347
00008C 00                        1649+         DC    BL1'00000'                             01-00348
00008D 0763                      1650+         DC    AL2(GPREF-UGCOMMON-2)                  01-00349
00008F 00F5                      1651+         DC    AL2(MPREF-MAPINP-3)                    01-00350
000091 0202                      1652+         DC    AL1(L'GPREF-1),AL1(L'MPREF-1)          01-00351
000093 FFFF                      1653+L0018    DC    XL2'FFFF'                              01-00361

                                 1655           UMSLXTBL MAPFLD=MCOLOR,GSAFLD=GCOLOR          01580008
000095                    00093  1656+          ORG   *-2                                     01-00345
000093 0A08                      1657+G$GCOLOR DC    AL1(L0019-*),AL1(008)                  01-00346
000095 00                        1658+M$MCOLOR DC    BL1'00000000'                          01-00347
000096 00                        1658+         DC    BL1'00000'                             01-00348
000097 0768                      1660+         DC    AL2(GOLOR-UGCOMMON-2)                  01-00349
000099 00FB                      1661+         DC    AL2(MCOLOR-MAPINP-3)                   01-00350
00009B 0000                      1662+         DC    AL1(L'GCOLOR-1),AL1(L'MCOLOR-1)        01-00351
00009D FFFF                      1663+L0020    DC    XL2'FFFF'                              01-00361
```

```
                                    1665          UMSLXTBL MAPFLD=MDATEA,GSAFLD=GDATEA          01600008
00009F                        0009D 1666+         ORG   *-2                                    01-00345
00009D 0A10                         1667+G$GDATEA DC    AL1(L0020-*),AL1(016)                 01-00346
00009F 02                           1668+M$MDATEA DC    BL1'00000010'                         01-00347
0000A0 00                           1669+         DC    BL1'00000'                            01-00348
0000A1 070A                         1670+         DC    AL2(GDATEA-UGCOMMON-2)                01-00349
0000A3 00D8                         1671+         DC    AL2(MDATEA-MAPINP-3)                  01-00350
0000A5 0109                         1672+         DC    AL1(L'GDATEA-1),AL1(L'MDATEA-1)       01-00351
0000A7 FFFF                         1673+L0020    DC    XL2'FFFF'                             01-00361

                                    1675          UMSLXTBL MAPFLD=MDATEB,GSAFLD=GDATEB,       X01620008
                                                        EDIT=(PDATE,GDATEA,GDATEB)             01630008
0000A9                        000A7 1676+         ORG   *-2                                    01-00345
0000A7 0E10                         1677+G$GDATEB DC    AL1(L0021-*),AL1(016)                 01-00346
0000A9 22                           1678+M$MDATEB DC    BL1'00100010'                         01-00347
0000AA 00                           1679+         DC    BL1'00000'                            01-00348
0000AB 070E                         1680+         DC    AL2(GDATEB-UGCOMMON-2)                01-00349
0000AD 00E8                         1681+         DC    AL2(MDATEB-MAPINP-3)                  01-00350
0000AF 0109                         1682+         DC    AL1(L'GDATEB-1),AL1(L'MDATEB-1)       01-00351
0000B1                              1683+E$GDATEA DS    0AL1            DEFINE USAGE          02-00518
0000B1                              1684+E$GDATEB DS    0AL1            DEFINE USAGE          02-00518
0000B1 00000144                     1685+         DC    AL4(E0022)      POINT TO EDIT ARRAY   02-00535
000144                              1686+CSECT2   CSECT                                        02-00536
000144 0C                           1687+E0022    DC    AL1(L0022-*)    LENGTH OF ARRAY       02-00537
000145 000009                       1688+         DC    AL1(00),AL2(9)  MODE FLAG BITS / TYPE 02-00755
000148 0000009D                     1689+         DC    AL4(G$GDATEA)   REFERENCE TO LXTBL BLK 02-00762
00014C 000000A7                     1690+         DC    AL4(G$GDATEB)   REFERENCE TO LXTBL BLK 02-00762
000150                              1691+L0022    DS    0AL1            END OF ARRAY          02-00770
0000B5                              1692+LXTABTST CSECT                                        02-00771
0000B5 FFFF                         1693+L0021    DC    XL2'FFFF'                             01-00361

                                    1695          UMSLXTBL MAPFLD=MLAST,GSAFLD=GLAST          01650008
0000B7                        000B5 1696+         ORG   *-2                                    01-00345
0000B5 0A08                         1697+G$GLAST  DC    AL1(L0023-*),AL1(008)                 01-00346
```

```
0000B7 00                       1698+M$MLAST  DC    BL1'00000000'                          01-00347
0000B8 00                       1699+         DC    BL1'00000'                             01-00348
0000B9 0739                     1700+         DC    AL2(GLAST-UGCOMMON-2)                  01-00349
0000BB 007E                     1701+         DC    AL2(MLAST-MAPINP-3)                    01-00350
0000BD 0F0F                     1702+         DC    AL1(L'GLAST-1),AL1(L'MLAST-1)          01-00351
0000BF FFFF                     1703+L0023    DC    XL2'FFFF'                              01-00361

                                1705          UMSLXTBL MAPFLD=MLICN25,GSAFLD=GLICN25       01670008
0000C1                   000BF  1706+         ORG   *-2                                    01-00345
0000BF 0A08                     1707+G$GLICN25 DC   AL1(L0024-*),AL1(008)                  01-00346
0000C1 00                       1708+M$MLICN25 DC   BL1'00000000'                          01-00347
0000C2 00                       1709+         DC    BL1'00000'                             01-00348
0000C3 07B8                     1710+         DC    AL2(GLICN25-UGCOMMON-2)                01-00349
0000C5 0121                     1711+         DC    AL2(MLICN25-MAPINP-3)                  01-00350
0000C7 1818                     1712+         DC    AL1(L'GLICN25-1),AL1(L'MLICN25-1)      01-00351
0000C9 FFFF                     1713+L0024    DC    XL2'FFFF'                              01-00361

                                1715          UMSLXTBL MAPFLD=MSTATE25,GSAFLD=GSTATE25,   X01690008
                                              EDIT=(MLICN2,GLICN25,GSTATE25)               01700008
0000CB                   000C9  1716+         ORG   *-2                                    01-00345
0000C9 0E08                     1717+G$GSTATE25 DC  AL1(L0025-*),AL1(008)                  01-00346
0000CB 20                       1718+M$MSTATE25 DC  BL1'00100000'                          01-00347
0000CC 00                       1719+         DC    BL1'00000'                             01-00348
0000CD 07B4                     1720+         DC    AL2(GSTATE25-UGCOMMON-2)               01-00349
0000CF 013D                     1721+         DC    AL2(MSTATE25-MAPINP-3)                 01-00350
0000D1 0101                     1722+         DC    AL1(L'GSTATE25-1),AL1(L'MSTATE25-1)    01-00351
0000D3                          1723+E$GLICN25 DS   0AL1           DEFINE USAGE           02-00518
0000D3                          1724+E$GSTATE25 DS  0AL1           DEFINE USAGE           02-00518
0000D3 00000150                 1725+         DC    AL4(E0026)     POINT TO EDIT ARRAY    02-00535
000150                          1726+CSECT2   CSECT                                        02-00536
000150 0C                       1727+E0026    DC    AL1(L0026-*)   LENGTH OF ARRAY        02-00537
000151 00000C                   1728+         DC    AL1(00),AL2(12) MODE FLAG BITS / TYPE  02-00755
000154 000000BF                 1729+         DC    AL4(G$GLICN25)  REFERENCE TO LXTBL BLK 02-00762
000158 000000C9                 1730+         DC    AL4(G$GSTATE25) REFERENCE TO LXTBL BLK 02-00762
```

```
00015C                          1731+L0026    DS    0AL1            END OF ARRAY          02-00770
0000D7                          1732+LXTABTST CSECT                                       02-00771
0000D7 FFFF                     1733+L0025    DC    XL2'FFFF'                             01-00361

                                1735          UMSLXTBL MAPFLD=MFIRST,GSAFLD=GFIRST        01720008
0000D9                   000D7  1736+         ORG   *-2                                  01-00345
0000D7 0A08                     1737+G$GFIRST DC    AL1(L0027-*),AL1(008)                 01-00346
0000D9 00                       1738+M$MFIRST DC    BL1'00000000'                         01-00347
0000DA 00                       1739+         DC    BL1'00000'                            01-00348
0000DB 074B                     1740+         DC    AL2(GFIRST-UGCOMMON-2)                01-00349
0000DD 0091                     1741+         DC    AL2(MFIRST-MAPINP-3)                  01-00350
0000DF 0B0B                     1742+         DC    AL1(L'GFIRST-1),AL1(L'MFIRST-1)       01-00351
0000E1 FFFF                     1743+L0027    DC    XL2'FFFF'                             01-00361

                                1745          UMSLXTBL MAPFLD=MMIDDLE,GSAFLD=GMIDDLE,     X01740008
                                              EDIT=(PNAME,GLAST,GFIRST,GMIDDLE)           01750008
0000E3                   000E1  1746+         ORG   *-2                                  01-00345
0000E1 0E08                     1747+G$GMIDDLE DC   AL1(L0028-*),AL1(008)                 01-00346
0000E3 20                       1748+M$MMIDDLE DC   BL1'00100000'                         01-00347
0000E4 00                       1749+         DC    BL1'00000'                            01-00348
0000E5 0759                     1750+         DC    AL2(GMIDDLE-UGCOMMON-2)               01-00349
0000E7 00A0                     1751+         DC    AL2(MMIDDLE-MAPINP-3)                 01-00350
0000E9 0707                     1752+         DC    AL1(L'GMIDDLE-1),AL1(L'MMIDDLE-1)     01-00351
0000EB                          1753+E$GLAST  DS    0AL1            DEFINE USAGE          02-00518
0000EB                          1754+E$GFIRST DS    0AL1            DEFINE USAGE          02-00518
0000EB                          1755+E$GMIDDLE DS   0AL1            DEFINE USAGE          02-00518
0000EB 0000015C                 1756+         DC    AL4(E0029)      POINT TO EDIT ARRAY   02-00535
00015C                          1757+CSECT2   CSECT                                       02-00536
00015C 10                       1758+E0029    DC    AL1(L0029-*)    LENGTH OF ARRAY       02-00537
00015D 00000A                   1759+         DC    AL1(00),AL2(10) MODE FLAG BITS / TYPE 02-00755
000160 000000B5                 1760+         DC    AL4(G$GLAST)    REFERENCE TO LXTBL BLK 02-00762
000164 000000D7                 1761+         DC    AL4(G$GFIRST)   REFERENCE TO LXTBL BLK 02-00762
000168 000000E1                 1762+         DC    AL4(G$GMIDDLE)  REFERENCE TO LXTBL BLK 02-00762
00016C                          1763+L0029    DS    0AL1            END OF ARRAY          02-00770
```

```
0000EF                          1764+LXTABTST CSECT                                    02-00771
0000EF FFFF                     1765+L0028    DC    XL2'FFFF'                          01-00361

                                1767          UMSLXTBL MAPFLD=MREGNO,GSAFLD=GREGNO,       X01770008
                                                     EDIT=(REGNO,GPREF,GCOLOR,GREGNO)     01780008
0000F1                   000EF  1768+          ORG   *-2                                01-00345
0000EF 0E08                     1769+G$GREGNO DC    AL1(L0030-*),AL1(008)              01-00346
0000F1 20                       1770+M$MREGNO DC    BL1'00100000'                      01-00347
0000F2 00                       1771+         DC    BL1'00000'                         01-00348
0000F3 076B                     1772+         DC    AL2(GREGNO-UGCOMMON-2)             01-00349
0000F5 00FF                     1773+         DC    AL2(MREGNO-MAPINP-3)               01-00350
0000F7 0606                     1774+         DC    AL1(L'GREGNO-1),AL1(L'MREGNO-1)    01-00351
0000F9                          1775+E$GPREF  DS    0AL1            DEFINE USAGE       02-00518
0000F9                          1776+E$GCOLOR DS    0AL1            DEFINE USAGE       02-00518
0000F9                          1777+E$GREGNO DS    0AL1            DEFINE USAGE       02-00518
0000F9 0000016C                 1778+         DC    AL4(E0031)      POINT TO EDIT ARRAY 02-00535
00016C                          1779+CSECT2   CSECT                                    02-00536
00016C 10                       1780+E0031    DC    AL1(L0031-*)    LENGTH OF ARRAY    02-00537
00016D 000004                   1781+         DC    AL1(00),AL2(4)  MODE FLAG BITS / TYPE 02-00755
000170 00000089                 1782+         DC    AL4(G$GPREF)    REFERENCE TO LXTBL BLK 02-00762
000174 00000093                 1783+         DC    AL4(G$GCOLOR)   REFERENCE TO LXTBL BLK 02-00762
000178 000000EF                 1784+         DC    AL4(G$GREGNO)   REFERENCE TO LXTBL BLK 02-00762
00017C                          1785+L0031    DS    0AL1            END OF ARRAY       02-00770
0000FD                          1786+LXTABTST CSECT                                    02-00771
0000FD FFFF                     1787+L0030    DC    XL2'FFFF'                          01-00361
                                1789          UMSLXTBL MAPFLD=MZIPC,GSAFLD=GZIPC         01800008
0000FF                   000FD  1790+          ORG   *-2                                01-00345
0000FD 0A18                     1791+G$GZIPC  DC    AL1(L0032-*),AL1(024)              01-00346
0000FF 40                       1792+M$MZIPC  DC    BL1'01000000'                      01-00347
000100 00                       1793+         DC    BL1'00000'                         01-00348
000101 0774                     1794+         DC    AL2(GZIPC-UGCOMMON-2)              01-00349
000103 0109                     1795+         DC    AL2(MZIPC-MAPINP-3)                01-00350
000105 0509                     1796+         DC    AL1(L'GZIPC-1),AL1(L'MZIPC-1)      01-00351
000107 FFFF                     1797+L0032    DC    XL2'FFFF'                          01-00361
```

```
                                    1799          UMSLXCON GSAINP=GXCODEA,GSAOUT=GICODE1,TYPE=ICODE
000109                        0001D 1800+         ORG   LXTABTST+X'1D'                          01-00830
00001D 00000188                     1801+         DC    AL4(CONTABLE)   CONVERSION TABLE PTR   01-00831
000021                        00109 1802+         ORG                                          01-00832
000188                              1803+CSECT3   CSECT                                        01-00833
000188 01                           1804+CONTABLE DC    BL1'00000001'   SET END  LAG           01-00834
000109                              1805+LXTABTST CSECT                                        01-00835
000189                              1806+CSECT3   CSECT                                        01-00837
000189                        00188 1807+         ORG   *-1                                    01-00838
000188 0000                         1808+         DC    BL1'00',AL1(0)                         01-00839
00018A 07B0                         1809+S#GXCODEA  DC  AL2(GXCODEA-UGCOMMON-2)               01-00840
00018C 077C                         1810+         DC    AL2(GICODE1-UGCOMMON-2)                01-00841
00018E 01                           1811+T#GICODE1 DC   BL1'00000001'                          01-00842
000109                              1812+LXTABTST CSECT                                        01-00843

                                    1814          UMSLXCON HOSTINP=HXCODEB,GSAOUT=GICODE2,TYPE=ICODE
00018F                              1815+CSECT3   CSECT                                        01-00837
00018F                        0018E 1816+         ORG   *-1                                    01-00838
00018E 0201                         1817+         DC    BL1'10',AL1(1)                         01-00839
000190 0046                         1818+S#HXCODEB  DC  AL2(HXCODEB-UHCITEXT-2)               01-00840
000192 079C                         1819+         DC    AL2(GICODE2-UGCOMMON-2)                01-00841
000194 01                           1820+T#GICODE2 DC   BL1'00000001'                          01-00842
000109                              1821+LXTABTST CSECT                                        01-00843

                                    1823          UMSLXTBL MAPFLD=MTITLE#,GSAFLD=GTITLE#,EDIT=TITLE#
000109                        00107 1824+         ORG   *-2                                    01-00345
000107 0E08                         1825+G$GTITLE# DC   AL1(L0035-*),AL1(008)                  01-00346
000109 20                           1826+M$MTITLE# DC   BL1'00100000'                          01-00347
00010A 00                           1827+         DC    BL1'00000'                             01-00348
00010B 06F8                         1828+         DC    AL2(GTITLE#-UGCOMMON-2)                01-00349
00010D 0116                         1829+         DC    AL2(MTITLE#-MAPINP-3)                  01-00350
00010F 0707                         1830+         DC    AL1(L'GTITLE#-1),AL1(L'MTITLE#-1)      01-00351
000111                              1831+E$GTITLE# DS   0AL1           DEFINE REFERENCE       02-00531
```

```
000111 0000017C                    1832+         DC    AL4(E0036)      POINT TO EDIT ARRAY    02-00535
00017C                             1833+CSECT2   CSECT                                        02-00536
00017C 08                          1834+E0036    DC    AL1(L0036-*)    LENGTH OF ARRAY        02-00537
00017D 000008                      1835+         DC    AL1(00),AL2(8)  MODE FLAG BITS / TYPE  02-00755
000180 00000107                    1836+         DC    AL4(G$GTITLE#)  REFERENCE TO LXTBL BLK 02-00766
000184                             1837+L0036    DS    0AL1            END OF ARRAY           02-00770
000115                             1838+LXTABTST CSECT                                        02-00771
000115 FFFF                        1839+L0035    DC    XL2'FFFF'                              01-00361
                                   1841 ********************************************************  01880001
                                   1842 **                                                    **  01890001
                                   1843 **   USER EXITS:  ALL GO HERE,                        **
                                        **                AT THE END OF THE LXTABLE           **  01900001
                                   1844 **                                                    **  01910001
                                   1845 **   NOTE: LEADING 'DROP' IS VERY IMPORTANT           **  01920006
                                   1846 ********************************************************  01930001
                                   1848          DROP                                         01950006
000198                             1849 UGLTEST1 CSECT                                        01960000
                          00000    1850          USING UGCOMMON,R9     COMMON AREA            01970000
                          00000    1851          USING EXITDATA,R8     EXIT DATA              01980000
                                        ************************************************
                                   1853 * THIS EXAMPLE WILL FORCE NO-ERROR IF THERE        **
                                        *  IS AN EXISTING ERROR AND WE HAVE BEEN           **
                                        *  ENTERED FOR THE REASON OF AN ERROR. IN ANY      **
                                   1855 *  OTHER CASE, NORMAL FLOW WILL BE OBSERVED        **
                                        ************************************************
                          00198    1857          USING USER01,R12       ROUTINE BASE        02040000
000198                             1858 USER01   DS    0H                                     02050000
000198 D603 96C0 96C0 006C0 006C0  1859          OC    UGGERRCD,UGGERRCD    PREVIOUS ERROR?   02060000
00019E 4780 C016            001AE  1860          BZ    U19999                 NO              02070000
0001A2 910A 8007      00007        1861          TM    EXITWHYX,EXRVALNG+EXREDTNG  ERROR ?    02080000
0001A6 4780 C016            001AE  1862          BZ    U19999                 NO              02090000
0001AA 9201 8008      00008        1863          MVI   EXITACTN,EXANOERR     YES, FORCE ERROR 02100000
0001AE                             1865 U19999   DS    0H                                     02120000
0001AE BFEF 8003            00003  1866          ICM   R14,B'1111',EXITRETN  GET RTN ADDRESS  02130000
```

```
0001B2 07FE                          1867          BR    R14                              EXIT     02140000
0001B8                               1869          LTORG                                           02160000
                                     1871          DROP                                            02180000
                               00000 1872          USING UGCOMMON,R9        COMMON AREA            02190000
                               00000 1873          USING EXITDATA,R8        EXIT DATA              02200000

                                     ************************************************
                                     1875 * THIS EXAMPLE WILL FORCE AN ERROR IF             **
                                     * GDATE = X'7FFF' AND GCHAR = ALL '$'                 **  02220000
                                     1876 * IN ANY OTHER CASE, NORMAL FLOW WILL BE OBSERVED **  02230000
                                     ************************************************
                               001B8 1878          USING USER02,R12          ROUTINE BASE       02250000
0001B8                               1880 USER02    DS    0H                                    02270000
0001B8 D501 9704 C034 00704 001EC    1881          CLC   GDATE,=X'7FFF'   7FFF?                  02280000
0001BE 4770 C026            001DE    1882          BNE   U29999              NO                 02290000
0001C2 955B 9728      00728          1883          CLI   GCHAR,C'$'       1ST BYTE OF GCHAR=$?  02300000
0001C6 4770 C026            001DE    1884          BNE   U29999              NO                 02310000
0001CA D50F 9729 9728 00729 00728    1885          CLC   GCHAR+1(L'GCHAR-1),GCHAR     REST=$?   02320000
0001D0 4770 C026            001DE    1886          BNE   U29999                       NO        02330000
0001D4 D203 8009 C030 00009 001E8    1887          MVC   EXITEROR,=F'123456789' SET ERROR-CODE  02340000
0001DA 9202 8008      00008          1888          MVI   EXITACTN,EXAWIERR         FORCE ERROR  02350000
0001DE                               1890 U29999    DS    0H                                    02370000
0001DE BFEF 8003            00003    1891          ICM   R14,B'1111',EXITRETN   GET RTN ADDRESS 02380000

        LXTABTST - UMS SAMPLE LXTBL DEFINITION                                        PAGE   20
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                 ASM H V 02 15.47 07/09/91

0001E2 07FE                          1892          BR    R14             EXIT                  02390000
0001E8                               1894          LTORG                                       02410000
0001E8 075BCD15                      1895                =F'123456789'
0001EC 7FFF                          1896                =X'7FFF'
                                     1897          DROP                                        02420000
                                     1899          END                                         02440000
```

```
MVS/XA DFP VER 2 LINKAGE EDITOR                15:47:57  TUE  JUL 09, 1991
JOB RMCJEBBX   STEP STEP010    PROCEDURE LKED
INVOCATION PARAMETERS - LIST,XREF
ACTUAL SIZE=(317440,79872)
OUTPUT DATA SET RMVMV.UMS.RMCJEBB IS ON VOLUME RELP02

                                               CROSS REFERENCE TABLE

 CONTROL SECTION                      ENTRY

   NAME    ORIGIN  LENGTH
 LXTABTST      00     117
 CSECT2       118      6C
 CSECT3       188       D
 UGLTEST1     198      56


 LOCATION  REFERS TO SYMBOL  IN CONTROL SECTION     LOCATION  REFERS TO  SYMBOL  IN CONTROL SECTI
      2F            CSECT2          CSECT2              47             CSECT2          CSECT2
      70            CSECT2          CSECT2              74             CSECT2          CSECT2
      B1            CSECT2          CSECT2              D3             CSECT2          CSECT2
      EB            CSECT2          CSECT2              F9             CSECT2          CSECT2
     111            CSECT2          CSECT2              1D             CSECT3          CSECT3
      55            UGLTEST1        UGLTEST1            78             UGLTEST1        UGLTEST1
     11C            LXTABTST        LXTABTST           140             LXTABTST        LXTABTST
     148            LXTABTST        LXTABTST           14C             LXTABTST        LXTABTST
     154            LXTABTST        LXTABTST           158             LXTABTST        LXTABTST
     160            LXTABTST        LXTABTST           164             LXTABTST        LXTABTST
     168            LXTABTST        LXTABTST           170             LXTABTST        LXTABTST
     174            LXTABTST        LXTABTST           178             LXTABTST        LXTABTST
     180            LXTABTST        LXTABTST
```

```
ENTRY ADDRESS       00

TOTAL LENGTH       1F0
** LXTABTST REPLACED AND HAS AMODE 24
** LOAD MODULE HAS RMODE 24
** AUTHORIZATION CODE IS         0.
```

Hexidecimal Dump of LXTABLE Example

```
AMASPZAP  INSPECTS, MODIFIES, AND DUMPS CSECTS OR SPECIFIC DATA RECORDS ON DIRECT ACCESS STORAGE.
  DUMPT LXTABTST ALL                                                    00110004

**CCHHR- 0011000314   RECORD LENGTH- 0001F0        MEMBER NAME  LXTABTST  CSECT NAME  LXTABTST
000000  D3E7E3C1 C2E3E2E3 E4C7D3E3 C5E2E3F0   F761F0F9 61F9F140 F1F54BF4 57000001 *LXTABTSTUGLTEST0*
                                                                               *7/09/91 15.47...*
000020   88000000 000E0820 02072600 5D101000   0001180A 08080000 3C007109 090E0011 *.............)...*
                                                                               *................*
000040   00071F00 AB040400 00012011 04811007   1800B304 06000001 98FFFFFF 0A0C0101 *................*
                                                                               *................*
000060   071200BD 03031910 32100702 00C40109   00000130 0000013C 000001B8 0100630A *.............D..*
                                                                               *................*
000080   14040007 0600D101 060A0800 00076300   F502020A 08000007 6800FB00 000A1002 *......J.........*
                                                                               *5..............*
0000A0   00070A00 DB01090E 10220007 0E00E801   09000001 440A0800 00073900 7E0F0F0A *.............Y.*
                                                                               *............=...*
0000C0   08000007 B8012118 180E0820 0007B401   3D010100 0001500A 08000007 4B00910B *................*
                                                                               *......&.........*
0000E0   0B0E0820 00075900 A0070700 00015C0E   08200007 6B00FF06 06000001 6C0A1840 *.............*.*
                                                                               *....,.......%.. *
000100   00077401 0905090E 08200006 F8011607   07000001 7CFFFF                   *............8...*
                                                                               *....@..         *

**CCHHR- 0011000314   RECORD LENGTH- 0001F0        MEMBER NAME  LXTABTST  CSECT NAME  CSECT2
000000   08000001 00000025 01030004 00001C00   002C0000 3C45670C 04020004 00008000 *................*
                                                                               *................*
000020   7FFFFFFF 08000012 00000066 0C000009   0000009D 000000A7 0C00000C 000000BF *"................*
                                                                               *................*
000040   000000C9 1000000A 000000B5 000000D7   000000E1 10000004 00000089 00000093 *...I...........P*
*................*
```

```
000060   000000EF 08000008 00000107                                      *...........    *


**CCHHR- 0011000314   RECORD LENGTH- 0001F0         MEMBER NAME  LXTABTST  CSECT NAME  CSECT3
000000   000007B0 077C0201 0046079C 01                                   *.....@.......   *


**CCHHR- 0011000314   RECORD LENGTH- 0001F0         MEMBER NAME  LXTABTST  CSECT NAME  UGLTEST1
000000   D60396C0 96C04780 C016910A 80074780   C0169201 8008BFEF 800307FE 00000000 *O..............*
                                                                                    *..............*
000020   D5019704 C0344770 C026955B 97284770   C026D50F 97299728 4770C026 D2038009 *N.........$....*
                                                                                    *..N.........K...*
000040   C0309202 8008BFEF 800307FE 00000000   075BCD15 7FFF                       *..............*
                                                                                    *.$..".          *
AMA113I COMPLETED DUMP REQUIREMENTS

AMA100I AMASPZAP PROCESSING COMPLETED

****************************** BOTTOM OF DATA ********************************
```

6 System Utility Programs

Resident Utilities

The Guest System Utilities are divided between non-resident and resident. The non-resident routines require access to large tables, such as the one that converts Massachusetts place names to a number and vice versa. Their functionality will be discussed later. The resident utilities are sections in module UGZ0003P that are loaded into memory at system start-up. The addresses of the entry points for each of the routines in the module are placed in the Guest Common Area at transaction start-up by the UMS System Control Program, UGZ0001P.

This is another example of "soft linkage" between system and application. The utilities may be changed in virtually any way and the application will need no maintenance. In addition, where the utilities are not hard-linked to the application, the size of the application module is kept to a minimum. The linkage is achieved through a small routine called UICALLST. This routine **must** be hard-linked to the application program. Do not try any kind of direct calls. The coding is simple and is as follows for COBOL:

CALL 'UICALLST' USING <funcaddrfld> <fldlist>.

<funcaddrfld> parameter is the name of the Guest Common Area field designated for the service required. These begin at UGCOMMON-ASSIST-ADDRESS-AREA for COBOL and UGCAMOD1 in the Assembler version. The comments or the actual field name will designate the service provided.

<fldlist> is the name(s) of the field(s) pertaining to the service called, see the detailed descriptions below.

Before issuing the call, one must properly initialize the Guest Common fields used by the particular routine. Each will be mentioned, in turn.

Date Conversion Routine

The caller provides one of the three date formats, Gregorian, Julian, or internal (two-byte binary) and the routine provides the other two. The choice is whether the user wants the day of the week also. The Gregorian date is returned as mm/dd/yyyy. It may be given as the 2 or 4 digit year, with or without the slashes, and the year may come first (yymmdd).

> Note: The internal, binary or serial date is a two-byte binary value of the number of days since January 1, 1940.

The data areas of interest are:

1. The FORMAT byte: this designates the form of date given and whether or not to return day-of-week. Values allowed are 0-2 and 4-6.

Return Day-of-week		No	Yes
input form given	Binary	0	4
	Gregorian	1	5
	Julian	2	6

2. The return code or status byte has the following value:

```
0 = OK                          1 = Range error
2 = Non-numeric input           3 = Invalid month
4 = Invalid day of month        5 = Invalid year
6 = Invalid day of year         7 = Null-date input
8 = Format unknown
```

Example of a COBOL Invocation.

CALL 'UICALLST' USING UGCOMMON-DATE-ROUTINE ADDRESS DFHCOMMAREA.

The Date Data-Fields

COBOL

```
15  UGCOMMON-DATE-AREA.
  20  UGCOMMON-DATE-INPUT-FORMAT    PIC X.
  20  UGCOMMON-DATE-RETURN-STATUS   PIC X.
  20  UGCOMMON-DATE-BINARY-FORMAT   PIC XX.
  20  UGCOMMON-DATE-JULIAN-FORMAT   PIC 9(07).
  20  FILLER REDEFINES UGCOMMON-DATE-JULIAN-FORMAT.
    25  UGCOMMON-DATE-JULIAN-YEAR   PIC 9(04).
    25  UGCOMMON-DATE-JULIAN-DAY    PIC 9(03).
  20  UGCOMMON-DATE-GREG-FORMAT-X.
    25  UGCOMMON-DATE-GREG-FORMAT   PIC 9(08).
  20  FILLER REDEFINES UGCOMMON-DATE-GREG-FORMAT-X.
    25  UGCOMMON-DATE-GREG-YEAR     PIC 9(04).
    25  UGCOMMON-DATE-GREG-MONTH    PIC 9(02).
    25  UGCOMMON-DATE-GREG-DAY      PIC 9(02).
  20  UGCOMMON-DATE-DAY-OF-WEEK     PIC 9.
```

Assembler

```
UGCDCONA DS    0CL20          DATE CONVERSION AREA
UGCDCFMT DS    CL1            INPUT FORMAT
UGCDCSTA DS    CL1            RETURN STATUS
UGCDCBIN DS    XL2            BINARY FORMAT
UGCDCJUL DS    CL7            JULIAN YYYYDDD FORMAT
UGCDCGRE DS    CL8            GREGORIAN YYYYMMDD FORMAT
UGCDCDAY DS    CL1            DAY OF WEEK#
```

Data-Name Address Routine

This routine will provide full-word address (24-bit) address for a data field-name. The calling parameters are **pairs** of a field-name and (then) the name of the 4-byte field. The 4-byte-field is defined as S9(9) COMP, and receives the address. A sample COBOL invocation is:

```
05  FIELD-NAME-1         PIC X(20).
05  FIELD-NAME-2         PIC X.

10  FIELD-NAME-1-ADDR    PIC S9(9) COMP.
10  FIELD-NAME-2-ADDR    PIC S9(9) COMP.
```

```
CALL 'UICALLST' USING UGCOMMON-DATA-NAME-ADDRESS
                      FIELD-NAME-1  FIELD-NAME-1-ADDR
                      FIELD-NAME-2  FIELD-NAME-2-ADDR.
```

Upon return from the routine, the -ADDR locations hold the 24-bit address of their 'companion fields.' No return code is provided. If the field name is not entered first, before the full-word name in the call, the results are guaranteed to be unpredictable.

The third and fourth function fields in the list are currently unused and will, if used, give an immediate no-op return to your program. The fifth and sixth functions are tables for UMS system control program use only. If these functions are invoked, a program check will occur.

Miscellaneous Edit Services

The seventh function, UGCOMMON-MISC-TRANSLATE-TABLES, or UGCAMSTR in assembler provides some miscellaneous edits with Upper Case Translation. The COBOL calling sequence is:

CALL 'UICALLST' USING <comareaname><fldname>

<comareaname> is the name of the start of the guest common area.

<fldname> is the name of the field to be processed

Prior to issuing the call, one must prime UGEDCODE and UGEDLENG with the specific service code for the field and the field's length. In all of these services, the field's characters are first translated from Lower to Upper case. The service codes, the implied field-lengths and the nature of the service are as follows:

Code	Length implied	Service description
0	none	Lower to Upper Case translation only
1	9	Mass. Lic.# edit (len optional)
2	2	State-code edit (len optional)
3	27	Syntatic Lic#/State code edit (len optional)

Note: If the twos-compliment of the code is passed, the translate is skipped.

In-Core Online Sort

The eighth and last routine is the core sort. The limitation is that the table to be sorted size plus the **required** 'dummy entry' must not exceed 65,280 bytes. The routine is called with three parameters illustrated below for COBOL:

CALL 'UICALLST' USING <ugcaddr>, <sortarea>, <sortctrl>

<ugcaddr> is the name of the start of the Common Area

<sortarea> is the name of the sort area or table to be sorted as illustrated below

<sortctrl> is the name of a 12 byte sort-descriptor data structure illustrated below

```
01  SORT-TABLE.
     05  ST-ELEMENT OCCURS XX  TIMES  PIC X(YY).
     05  ST-DUMMY                     PIC X(YY).


01  SORT-DESCRIPTOR.
     05  SD-NUM-ENTRIES                PIC S9(9) COMP.
               ( not including the 'dummy entry' )
     05  SD-LEN-ENTRY                  PIC S9(4) COMP.
          88 SD-GOOD-LENGTH  VALUE 1 THRU 256.
     05  SD-LEN-KEY                    PIC S9(4) COMP.
     05  SD-POSITION-OF-KEY            PIC S9(4) COMP.
     05  SD-SORT-TYPE                  PIC X.
          88 SD-ASCENDING      VALUE 'A'.
          88 SD-DESCENDING     VALUE 'D'.
     05  SD-STATUS-BINARY              PIC X.

(NOTE:    88 SD-STATUS-OK                    VALUE 0.
  these   88 SD-STATUS-BAD-TYPE             VALUE 4.
  are     88 SD-STATUS-BAD-ENTRY-LEN      VALUE 8.
  to      88 SD-STATUS-BAD-KEY-LEN         VALUE 12.
  document 88 SD-STATUS-BAD-KEY-POSN       VALUE 16.
  values        88 SD-STATUS-TOO-MANY-ENTRIES    VALUE 20.
  only)         88 SD-STATUS-TABLE-TOO-LARGE     VALUE 24.
```

To inspect the status in COBOL try:

```
01  SORT-STATUS-WORK            PIC S9(4)  COMP.
01  FILLER REDEFINES SORT-STATUS-WORK.
     05  FILLER                  PIC X.
     05  SS-WORK                 PIC X.
```

```
MOVE ZERO              TO  SORT-STATUS-WORK.
MOVE SD-STATUS-BINARY  TO  SS-WORK.
IF SORT-STATUS-WORK  EQUAL  0
     PERFORM GOOD-SORT
       ELSE
     PERFORM BOMB-IT.
```

NOTE: 88-LEVELS (not these) WOULD WORK ON SORT-STATUS-BINARY.

7 UMS Screen Mapping Procedures

Because the UMS System is CICS based, the use of Basic Mapping Support (BMS) is a virtual necessity. Several components must be produced. First is the MAP itself, an object module used at run time by CICS routines. Next is the storage definition of the data after being processed by BMS (on input or output). In UMS, a third DSECT, or storage definition is needed. It is used for the program's working storage or communication area. There are some options on creating the first two items mentioned in this chapter, but the third DSECT must be created by hand.

In this third definition, the data field must be UMS-compatible with the screen map definition; that is of the same type and length or of one that matches one of the LXTABLE's automatic conversions such as PIC 9(5) into S9(4) comp. Each one of these fields must be preceded by two one-byte fields suffixed by "_Z" and "_T" respectively. These Z and T fields have functions in the LXTABLE processing cycle that are discussed in that chapter. If you do not plan to use the Z and T fields, these can be replaced by two filler bytes as PIC XX or DS CL2. For example, if you are operating with the field "DRIVER-NAME," the DSECT fields might look as follows:

```
05  DRIVER-NAME-G.
  10  DRIVER-NAME-Z              PIC X.
  10  DRIVER-NAME-T              PIC X.
  10  DRIVER-NAME                PIC X(25).
```

These T and Z field names are conveniently generated by the GFLD macro for assembler programs and the LX table assembly. The call <fldname> GFLD <asmpic> is the pattern. For example:

```
DRVRNM GFLD CL25          will produce

DRVRNM_Z DS  CL1
DRVRNM_T DS  CL1
DRVRNM   DS  CL25
```

The field DRVRNM is the one that will be mentioned in the LXTABLE entries. The LXTABLE processing assumes that the other two fields are represented. If they are not, the mapping results will be unpredictable.

The options are in the creation of the map itself and its DSECT. System development has access to the EDS Screen Generator program which, for licensing reasons, is not placed on the distribution tape. The generator is described later in this chapter.

The manual map generation process has been somewhat simplified. In place of coding the DFHMSD and DFHMDI macros, code the required UMSHEADER macro. This macro generates the DFHMSD and DFHMDI invocations along with a standard heading for all UMS screens. Parameters on the macro provide the necessary customization as follows:

```
<name> UMSHEADR <sfunc>,<scrntitle>,TYPE=<type>,SYSTEM=<sys>

     <name>  is a required 1 to 6 character map name for
          the assembly

     <sfunc>  is a required positional parameter, a 1 to
          4 character function code, served by this
          screen, that appears in the screen header

     <scrntitle>  is a required positional parameter, a
          1 to 80 character literal that will be a
          title appearing on the second line of the
          screen

     <type> = NORMAL | DSECT     the assembler output
          form, Normal generates map statements, Dsect
          generates the DS definitions

     <sys> = UMS | ALAR    this determines the screen
          date format:    UMS = mm/dd/yyyy, ALAR =
          mm/dd/yy
```

This macro generates 4 UMS-standard fields:

- The current date
- The time
- The function (from <sfunc>)
- A 50-character message line

These are all manipulated outside of the application program. To write messages, one only specifies a message number (see chapter on Error Messages) and the appropriate text is inserted by the system control programs.

Following this macro, code the standard DFHMDF macros to define each map field and literal for the screen. The standard requires the names of the fields be of the form:

```
     FLD<fnum>    where <fnum> is a 3 or 4 digit number,
          beginning with 5 that is zero filled, such
          as:

               FLD005, FLD0007, FLD012
```

They begin with 5 as the UMSHEADR generates the first 4 fields in the map. After the last macro, one must supply the standard (CICS) **DFHMSD TYPE=FINAL** and an assembler **END** statement. When all statements are assembled and linked, the map module is complete. The only concern with using this process is the names of the fields. The mapping macros are assembler orientated and the names must be 7 characters or less, with the macro expansion supplying the eighth. Even though the newer assembler supports 30-character data names, the UMSHEADR macro does not currently support a language parameter and defaults to "LANG=ASM" in the DFHMSD macro. This forces one to hard code the COBOL storage definition. There are only two "tricks" to this.

- First, provide 12 bytes of filler at the head of the map storage.
- Second, provide three (3) bytes of storage ahead of each data field.

Using LXTABLE, one does not reference these fields so a definition of "PIC X(3)." is sufficient. Properly, they are first, "PIC S9(4) COMP." followed by "PIC X." This non-automatic description generation works well but one must be **very** careful that screen map changes affecting the length, type, or location of fields be reflected in **two** additional places, the hand-coded map "DSECT" and the storage definition for the application program which is accompanied by the "Z-field," "T-field" components. The EDS map generator has the advantage of generating the map source (which must be modified) and its COBOL storage definition in the same pass. This will eliminate one source of potential errors. Remember, map/dsect misalignments produce "unpredictable" results. At the end of this chapter is the various "code" segments for a very simple map to illustrate the components required. The Map Generator discussion can be read to see these components from another perspective.

8 UMS Online Error Messages

The error-code passed with the outbound message text is structured as a 4 byte field, which is treated as a 1 word binary number. From the application perspective, it needs to be considered as a signed 9 digit numeric field.

Digits	Length	Usage
0-1	2	major process identification
2-5	2	field grouping code - characters 4 through 7 of originatory program. i.e., for UGB0710P, these four characters would be '0710.'
6-8	3	sequence number

The concept is that a given code will be issued by **one module under 1 set of conditions**, only. The first 3 digits will relate to the module. The field grouping code will, within the module, attempt to localize the field(s) being processed when the error was detected. The detail code will attempt to further qualify the problem. These sub-field usages are designed to make analysis simpler from all perspectives. However, the general concept will be that a unique 9 digit code yields a unique message (from the message dictionary on the guest side). The only time the guest is concerned with the sub-code breakdown is if its dictionary is out of sync and the text cannot be located.

The dictionary will contain a flag set for each message which indicate if the 8 byte error qualifier field is used for this code, and if so how. This structure is not defined at this time.

Note: If the same error-text is produced from more than 1 place, it is assigned multiple codes so that the source can be localized as required.

The Host Function/Dispatch/Security process reserves all Error Codes with digits 0-2 in the range of 100-199 for itself. Guest processes need to be aware of this value range.

UMS online programs move error numbers to the common area when an error is encountered. The Guest Side Cleanup module (UGZ0005P) matches the error number with an error number in the error message table and finds a corresponding error message. Error messages reside in a member named UGZ0004P in the UMS source libraries.

**** This member consists of list the following messages:

Type of messages	Range of numbers	Copybook
UMS control CICS programs messages	199-200	
UMS error messages	201-299	
EDS messages	300-599	$EDSMSGS
MRB messages	600-699	$MRBMSGS
Stopper/enabler messages	800-899	
Alars error messages and NDR responses	900-999	

The EDS and MRB message copy members reside in the UMS macro libraries.

****** To change a message in $EDSMSGS or $MRBMSGS, the SE should copy the error message member from RMVM.RMV.UMSMACR3 library to RMVMV.RMV.UMSMACRO library, make the needed changes, copy the error message table UGZ0004P from RMVM.RMV.UMSSRCE3 to RMVMV.RMV.UMSSRCE, reassemble this table with new changes, move modified copybook and error message table (UGZ0004P) to CV14 and CV11 library using automated UMS move method.

To change a message (UMS, Alars , or Stopper/enabler message), the programmer should copy the error table UGZ0004P from RMVMV.RMV.UMSSRCE library to RMVMV.RMV.UMSSRCE, make the needed changes, reassemble the error table, move modified UGZ0004P (error message table) to CV14 and CV11 library using automated UMS move method.

Adding an Error Message to a Program

The members $EDSMSGS, $MRBMSGS, and UGZ0004P (error message table) are arranged by error number and program name. To add an error number, the programmer should find the sequence of numbers that exists for the program, and choose the next available number. If an error message is being added that already exists for a different program, then an alias is used as a pointer to the previous message. An alias can only be used if the new error number is greater than the alias number.

The programmer should copy $EDSMSGS or $MRBMSLS to RMVMV.RMV.MACRO (test library). If EDS message or MRB message must be added, then copy the error message table UGZ0004P from RMVMV.RMV.MACR3 to RMVM.RMV.MACRO, add new message (if it is required) and reassemble the error table.

Example:

The program UGRO011P needs an error message for "F8 INVALID, IMPROPER CURSOR POSITIONING". Following is a sample of the member $EDSMSGS:

```
ERR   203001010,UGRO010P,,'F8 INVALID, IMPROPER CURSOR
      POSITIONING'
ERR   203001011,UGRO010P,,'F8 INVALID, END OF SET ENCOUNTERED'
ERR   204001010,UGRO011P,,'ALIAS=200001011'
```

In this case, the new entry would appear as follows:

```
ERR   20400101011,UGRO011P,,'ALIAS=203001010'
```

In this example the ALIAS points to the error message used for error 203001010.

UGZ0004P - The Message Module

UGZ0004P is a module also known as the message text module. This module is the central repository for all UMS system messages. The system programs use a message code passed in the commarea to the clean-up control module (UGZ0005P) which will translate the code into the text which is then put into the appropriate area on the screen.

Messages must be in ascending order by error-code.

ERROR DICTIONARY ELEMENT DEFINITION

&P0 ERR &P1,&P2,&P3,&P4

&P1 = Required 9 decimal digit error-code

&P2 = Required 1 to 8 Byte Program-name

&P3 = Optional 1 to 8 hex digit flagset

&P4 = Required 1 to 50 byte message

NOTE: The message can be replaced by text of the form of 'ALIAS=123456789' where 123456789 is the code number of a previously defined message to reuse the same text with a different error number.

Some Examples:

```
ERR      203001011,UGR0010P,,'F8 INVALID.  END OF SET
         ENCOUNTERED.'
ERR      203001012,UGR0011P,,'ALIAS=200001011'
ERR      203001013,UGR0011P,,'ALIAS=200001011'
ERR      203001014,UGR0011P,,'ALIAS=201001013'
```

Assembled Example of a Message Module

Note: These examples have been slightly modified by text-edit to allow them to fit in the page or to improve their readability. For brevity, some macros have a number of parameters to show the results of expansion. Some combinations, while assembled correctly may be illogical. Please consult the detailed write-up for parameter usage.

```
                    EXTERNAL SYMBOL DICTIONARY                 PAGE    1
SYMBOL   TYPE  ID  ADDR  LENGTH  LD ID FLAGS                       ASM H V 02 14.58 07/10/91

CSECT1    SD 0001 000000 000064         00
CSECT2    SD 0002 000068 00006B         00
```

```
        ECE0004P - TEST GUEST-SIDE ERROR MESSAGE DICTIONARY                     PAGE    2
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                ASM H V 02 14.58 07/10/91

                                     2          PUSH     PRINT                          00110000
                                     3          PRINT    OFF                            00120000
                                   136          POP   PRINT                             01450000
                                   137          GBLC  &OPTPR                           01460000
                                   138 &OPTPR   SETC  '&SYSPARM'                       01470000
                                   139          AIF   ('&OPTPR' EQ 'GEN').TP020        01480000
                                   140          AIF   ('&OPTPR' EQ '').TP020           01490000
                                   141          AIF   ('&OPTPR' EQ 'NOGEN').TP010      01500000
                                   142 .TP010   ANOP                                   01520000
                                   143          PRINT GEN                              01530012
                                   144 .TP020   ANOP                                   01540000
000000                             145 CSECT1   START 0                                01550000
```

```
                                    146 *   S T A R T   O F   T A B L E   P R E F I X     01560000
000000 00000000                     147 NUMENT   DC    F'0'                               01570000
000004 0000002C                     148          DC    A(FIRSTTAG)                        01580000
000008 00000068                     149 FIRSTDTL DC    A(CSECT2)                          01590000
00000C 00000010                     150          DC    A(RELEASE)                         01600000
                                    151 RELEASE  DC    AL2(030),CL9'&SYSDATE'
                                                 DC    CL5'&SYSTIME',C'=VER 03.01='       01610000
000010 001EF0F761F1F061                +RELEASE  DC    AL2(030),CL9'07/10/91'
                                       +         DC    CL5'14.58',C'=VER 03.01='          01610000
00002C                              152 FIRSTTAG DS    0F                                 01620000
                                    153 *   E N D   O F   T A B L E   P R E F I X         01630000



                                    154    ERR   100000001,UHZ0001P,,'HOST MODULE NOT FOUND'
00002C                              155+CSECT1   CSECT                                     01-00125
00002C 05F5E101                     156+D100000001 DC F'100000001'
000030 14000000                        +           DC AL1(L'TG30001-1+X'0'),AL3(TG20001-CSECT2)
000068                              157+CSECT2   CSECT                                     01-00127
                             00068  158+TG20001    EQU *                                  01-00128
000068 C8D6E2E340D4D6C4             159+TG30001    DC  C'HOST MODULE NOT FOUND'           01-00132
000034                              160+CSECT1   CSECT                                     01-00133



                                    161    ERR   100000002,UHZ0022P,,'UNKNOWN HOST FUNCTION'
000034                              162+CSECT1   CSECT                                     01-00125
000034 05F5E102                     163+D100000002 DC F'100000002'
000038 14000015                        +           DC AL1(L'TG30002-1+X'0'),AL3(TG20002-CSECT2)
00007D                              164+CSECT2   CSECT                                     01-00127
                             0007D  165+TG20002    EQU *                                  01-00128
00007D E4D5D2D5D6E6D540             166+TG30002    DC  C'UNKNOWN HOST FUNCTION'           01-00132
00003C                              167+CSECT1   CSECT                                     01-00133
```

```
                                    168    ERR   100000003,UHZ0333P,,'RECURSIVE DUP-KEY CALL'
00003C                              169+CSECT1   CSECT                                    01-00125
00003C 05F5E103                     170+D100000003 DC F'100000003'
000040 1500002A                        +           DC AL1(L'TG30003-1+X'0'),AL3(TG20003-CSECT2)
000092                              171+CSECT2   CSECT                                    01-00127
                              00092 172+TG20003    EQU *                                01-00128
000092 D9C5C3E4D9E2C9E5             173+TG30003    DC  C'RECURSIVE DUP-KEY CALL'         01-00132
000044                              174+CSECT1   CSECT                                    01-00133


                                    175    ERR   456789004,UHZ0001P,,'DUP-KEY NOT SPECIFIED'
000044                              176+CSECT1   CSECT                                    01-00125
000044 1B3A0C0C                     177+D456789004 DC F'456789004'
000048 14000040                        +           DC AL1(L'TG30004-1+X'0'),AL3(TG20004-CSECT2)
0000A8                              178+CSECT2   CSECT                                    01-00127
                              000A8 179+TG20004    EQU *                                01-00128
0000A8 C4E4D760D2C5E840             180+TG30004    DC  C'DUP-KEY NOT SPECIFIED'          01-00132
00004C                              181+CSECT1   CSECT                                    01-00133


                                    182    ERR   567891123,UHZ0001P,,'ALIAS=100000003'    01680012
00004C                              183+CSECT1   CSECT                                    01-00118
00004C 21D954B3                     184+A567891123 DC F'567891123'
000050 FF00003C                        +           DC XL1'FF',AL3(D100000003-CSECT1)     01-00119


                                    185    ERR   776721234,UHZ0003P,12345678,'ALIAS=456789004'
000054                              186+CSECT1   CSECT                                    01-00118
000054 2E4BD352                     187+A776721234 DC F'776721234'
000058 FF000044                        +           DC XL1'FF',AL3(D456789004-CSECT1)     01-00119


                                    188    ERR   880000102,UHZ0003P,FFEE1234,'INVALID MVE LENGTH'
00005C                              189+CSECT1   CSECT                                    01-00125
```

```
00005C 3473BC66                    190+D880000102 DC F'880000102'
000060 95000055                       +           DC AL1(L'TG3000 -1+X'84'),AL3(TG20007-CSECT2)
0000BD                             191+CSECT2   CSECT                                        01-00127
                          000BD    192+TG20007    EQU *                                      01-00128
0000BD FFEE1234                    193+         DC    XL4'FFEE1234'                         01-00130
0000C1 C9D5E5C1D3C9C440            194+TG30007    DC  C'INVALID MVE LENGTH'                 01-00132
000064                             195+CSECT1   CSECT                                        01-00133

000064                             197 CSECT1   CSECT                                        04509200
                          00064    198 LASTTAG  EQU   *                                     04509300
000064                    00000    199          ORG   NUMENT                               04509400
000000 00000007                    200          DC    A((LASTTAG-FIRSTTAG)/8)              04509500
000004                    00064    201          ORG                                        04509600
                                   202          GBLC  &WARN,&WTEMP                         04509700
                                   203          GBLA  &MSGDUP,&ALIDUP                      04509800
                                   204 &WTEMP   SETC  '&MSGDUP'                            04509900
                                   205     MNOTE &WARN,'&WTEMP AUTOMATIC ALIAS ENTRIES GENERATED'
   IEV254  *** MNOTE ***              +     ,0 AUTOMATIC ALIAS ENTRIES GENERATED          04510000
                                   206 &WTEMP   SETC  '&ALIDUP'                            04520000
                                   207     MNOTE &WARN,'&WTEMP INVALID ALAS REFERENCES CHANGED'
   IEV254  *** MNOTE ***              +     ,0 INVALID ALIAS REFERENCES CHANGED           04530000
                                   208          END                                        04540000

 MVS/XA DFP VER 2 LINKAGE EDITOR                14:58:28  WED  JUL 10, 1991
 JOB RMCJEBBP   STEP STEP010    PROCEDURE LKED
 INVOCATION PARAMETERS - LIST,XREF
 ACTUAL SIZE=(317440,79872)
 OUTPUT DATA SET RMVMV.UMS.RMCJEBB IS ON VOLUME RELP02

                                          CROSS REFERENCE TABLE

  CONTROL SECTION                    ENTRY
    NAME    ORIGIN  LENGTH             NAME   LOCATION     NAME   LOCATION     NAME   LOCATION
NAME   LOCATION
```

```
  CSECT1         00       64
  CSECT2         68       6B


  LOCATION  REFERS TO SYMBOL  IN CONTROL SECTION       LOCATION  REFERS TO SYMBOL  IN CONTROL
SECTION
        8            CSECT2          CSECT2
 ENTRY ADDRESS       00

 TOTAL LENGTH        D8
 ** ECE0004P REPLACED AND HAS AMODE 24
 ** LOAD MODULE HAS RMODE 24
 ** AUTHORIZATION CODE IS         0.
```

Hexidecimal dump of Message Module Example

```
AMASPZAP  INSPECTS, MODIFIES, AND DUMPS CSECTS OR SPECIFIC DATA RECORDS ON DIRECT ACCESS STORAGE.
  DUMPT ECE0004P ALL                                                    00110006

**CCHHR- 0011000336   RECORD LENGTH- 0000D8        MEMBER NAME  ECE0004P  CSECT NAME  CSECT1
000000   00000007 0000002C 00000068 00000010   001EF0F7 61F1F061 F9F140F1 F44BF5F8  *................*
                                                                                   *..07/10/91 14.58*
000020   7EE5C5D9 40F0F34B F0F17E00 05F5E101   14000000 05F5E102 14000015 05F5E103  *=VER 03.01=..5..*
                                                                                   *.....5.......5..*
000040   1500002A 1B3A0C0C 14000040 21D954B3   FF00003C 2E4BD352 FF000044 3473BC66  *.......... .R..*
                                                                                   *......L........*
000060   95000055                                                                  *....            *


**CCHHR- 0011000336   RECORD LENGTH- 0000D8        MEMBER NAME  ECE0004P  CSECT NAME  CSECT2
000000   C8D6E2E3 40D4D6C4 E4D3C540 D5D6E340   C6D6E4D5 C4E4D5D2 D5D6E6D5 40C8D6E2  *HOST MODULE NOT *
                                                                                   *FOUNDUNKNOWN HOS*
000020   E340C6E4 D5C3E3C9 D6D5D9C5 C3E4D9E2   C9E5C540 C4E4D760 D2C5E840 C3C1D3D3  *T FUNCTIONRECURS*
                                                                                   *IVE DUP-KEY CALL*
000040   C4E4D760 D2C5E840 D5D6E340 E2D7C5C3   C9C6C9C5 C4FFEE12 34C9D5E5 C1D3C9C4  *DUP-KEY NOT SPEC*
                                                                                   *IFIED....INVALID*
000060   40D4E5C5 40D3C5D5 C7E3C8                                                  * MVE LENGTH     *


AMA113I COMPLETED DUMP REQUIREMENTS

AMA100I AMASPZAP PROCESSING COMPLETED
 ****************************************** BOTTOM OF DATA ************************************
```

9 Special System Functions

This chapter is essentially a continuation of Chapter 2, Control Dispatch. The system functions described here are more specialized in their operation and were therefore separated from the "main line" Control/Dispatch discussion.

Limited Secondary Session (LSS)

This service allows cursor selection of functions from within other functions, which are **not** scroll functions. It also allows the execution of that function, and the return to the original function at the exact point of exit. This is similar to sub-routine execution. The service works in conjunction with the PA1 secondary-session facility. The LSS session differs from the PA1 facility in that, when invoked, a function-name and surrogate are passed to the session. In this mode, the user **cannot** change the function (prohibited by Control Dispatch). The only function keys allowed are 7, 8, 9, and 12. All other function keys are treated as a "return to ordinary session" request. The only function changes which can occur are those accomplished by internal dispatch. The user is allowed to change key fields for the invoked function.

The concept of "reference list" is introduced. This list is very similar to the scroll table but is located in other storage media. It is intended to provide a cursor selection entry to the LSS which is independent of the scroll facility and is a detailed reference mechanism to it.

As with F6 (screen hop), any function which can receive a F4 selection is eligible to receive this service and **cannot** differentiate it from a F4 entry. F3 is used for cursor selection from a reference-list, and can only be used from their primary session. Any program is eligible to build a reference list and no PCTEG entry is required for this eligibility. Whenever a function change occurs on the guest (even if it occurs via internal dispatch or host function swap), any existing reference-list is purged. A module was built to handle the reference-list. Its name is stored in guest common in the eight bytes following the name of the internal dispatch module.

The reference list itself is built in the end of the map-output area. This should require no special changes as LXTABLE manages these areas. If the program does not use the LXTABLE mechanism for mapping it will have to take care in the use of the storage. After a link to the reference list manager the 572 bytes of the output map area are cleared by the manager. Applications must not use this space unless they call the manager. If they do use this space, it may confuse LXTBL or SENDBACK. A DSECT for the area is

in Appendix A of the *UMS Programmer's Manual*. If the reference list manager is called with entries-per-line equal to low-values, the reference list will be purged. Each call to the manager sets a return code in the guest common error-code. If it is non-zero, the application should exit and allow the error to be posted. The manager is called by a CICS LINK passing guest common. Because a LINK is involved and the manager stores the reference list on external media, excess calls MUST be avoided.

UMS Screen Hop Facility

Screen hop is defined as a mechanism of transferring from one business function to another where the logical business key being accessed in the first function presents the external appearance of being automatically transferred to the second. In UMS, the terminal operator will indicate a desire to accomplish such a transfer by means of F6. In the context of such a transfer, the business function during which F6 is pressed is referred as the "hop-from" function, and the new function keyed at the top of the screen is referred to as the "hop-to" function.

From a technical perspective, the use of F6 will be transparent to the hop-to function. On entry to this function, the terminal operator's use of F6 is indistinguishable from use of F4. The entry-reason code will contain the same value, and the F4/F9 surrogate type/value is set up in the same fashion. The 60 bytes in guest common called UGSURLST, previously not referenced (but reserved for screen-hop usage) is defined as 15F. It changed in definition to 12(CL1,FL4). UGSURLST is still 60 bytes long, but provides for surrogate-types as well as surrogate values.

The hop-from function has some responsibility in the use of this 60 byte area, as does the control-dispatch software. In order to properly use the 60 byte area, all functions must be categorized relative to this area. The possible categorizations are:

A. primary
B. supportive
C. neutral
D. primary & supportive

A function defined as primary puts definitive information in the table. It typically operates from a definitive key. Examples of this type of function are ULI and URI. ULI places a PERS surrogate and type in the table. URI places PERS, VEHR, VEHT, VEHC, and VMOD surrogates and types in the table. Because these are primary functions, they have sole responsibility for the table. Therefore, **before** going to the host to read a record they **must** move low-values to the table. **After** a successful return from the host, they must place the appropriate surrogate types and values in the table. The order in which the entries are placed in the table is unimportant to the control - dispatch functions, but does represent the order in which the entries are evaluated during an F6 transfer, and thus can be viewed as a priority definition mechanism.

Example: ULI, before going to host:

```
MOVE LOW-VALUES TO UGCOMMON-RESERVED-SURROGATES.
```

Example: ULI after successful return from host:

```
MOVE '4'  TO UGCOMMON-RESERVED-SURR-TYPE(1).
MOVE RMV-PERS-SURROGATE
        TO UGCOMMON-RESERVED-SURR-VALUE(1).
```

A function defined as supportive adds information to the table. It is usually used to get more information about a primary key. The primary key would generally have been entered on a previous screen. An example of this in ALAR is SP. SP typically operates from an existing VEHR or PERS key, and gets parking offense data (INCD-OFNS & ACTN surrogates). If this function were to be implemented in UMS, **before** going to the host with an existing key, it would SEARCH the table and if it found any type entries which were INCD-OFNS or ACTN it would put low-values in the type entry. **After** a successful return from the host, it again searches the table, and uses the first entries with low-values in the surrogate-type for its INCD-OFNS and ACTN entries.

Example: SP, before going to host:

```
2222-SETUP-TABLE.
    SET UGCOMMON-RESERVED-SURR-INDEX TO 1.
2222-SEARCH-TABLE.
      SEARCH UGCOMMON-RESERVED-SURR-LIST
            WHEN
            UGCOMMON-RESERVED-SURR-TYPE(UGCOMMON-RESERVED-SURR-INDEX)
            = (incd-ofns-surr-type OR actn-surr-type)
            MOVE LOW-VALUES TO
            SET UGCOMMON-RESERVED-SURR-INDEX UP BY +1
            GO TO 2222-SEARCH-TABLE.
2222-EXIT.
    EXIT.
```

Example: SP after successful return from host:

```
3333-FIXUP-TABLE.
    SET UGCOMMON-RESERVED-SURR-INDEX TO 1.
    SEARCH UGCOMMON-RESERVED-SURR-LIST
    AT END GO TO 3333-EXIT
    WHEN
    UGCOMMON-RESERVED-SURR-TYPE(UGCOMMON-RESERVED-SURR-INDEX) =
      LOW-VALUES
            MOVE INCD-OFNS-SURR-TYPE TO
            UGCOMMON-RESERVED-SURR-TYPE(UGCOMMON-RESERVED-SURR-INDEX)
            MOVE INCD-OFNS-SURR-VALUE TO
            UGCOMMON-RESERVED-SURR-VALUE(UGCOMMON-RESERVED-SURR-INDE
            X)
            SET UGCOMMON-RESERVED-SURR-INDEX UP BY +1.
            SEARCH UGCOMMON-RESERVED-SURR-LIST
            AT END GO TO 3333-EXIT
```

```
                    WHEN
                    UGCOMMON-RESERVED-SURR-TYPE(UGCOMMON-RESERVED-SURR
                    -INDEX) = LOW-VALUES
                    MOVE ACTN-SURR-VALUE TO
                    UGCOMMON-RESERVED-SURR-TYPE(UGCOMMON-RESERVED-SURR
                    -INDEX)
                    MOVE ACTN-SURR-VALUE TO
                    UGCOMMON-RESERVED-SURR-VALUE(UGCOMMON-RESERVED-SUR
                    R-INDEX)
                    SET UGCOMMON-RESERVED-SURR-INDEX UP BY +1.
3333-EXIT.
      EXIT.
```

A function defined as neutral does not change the table.

A function defined as primary and supportive may need both types of logic mentioned above. SP might be an example of this if it is entered by other than (F4 or F6) and a license# or regno key directly keyed.

The expression of the above categorizations is in the code contained within the applications functional processors. No UMS tables are changed for this support. The decisions relative to the categorizations are a business type decision. For UMS versions of existing ALAR applications, these decisions are made by duplicating existing ALAR functionality. For new applications, a specification level decision is required.

UMS control-dispatch will clear the above mentioned 60 bytes whenever a function-change occurs without the pressing of F6.

10 Record Surrogates

An IDMS database is the primary database used for the RMV. It is described, with Bachman diagrams, in a separate document. These documents are maintained by DBAs. The most unusual feature is the use of the 'surrogate records.' There are six surrogate record types listed below. They are stored in single record-type areas with a key which is a full-word binary number through the CAL-S algorithm, rather than the usual IDMSCALC. The next surrogate number comes from an instance of one record-type, whose value is incremented and then replaced. The algorithm uses the surrogate number as the records position within the area. It can, with some information about the area, derive the page it should be stored on. The record goes in the calc chain for that page. If the area has only one record type, the suggested page is always suitable and there is no overflow at all.

The beauty of this arrangement is that the area may be page-expanded, have pages added or be unloaded and reloaded into new file extents with a new geometry. In any case, the CAL-S algorithm will function undisturbed as the nth record in the area. It will always be the nth, no matter what changes are made. This surrogate number, which is a unique key for each of its record types, can be (and is) used as a calc-key for records, in other areas, stored by IDMSCALC. The uniqueness and uniform size of these surrogate keys have provided very good space-management results in the other areas. The surrogate values are also stored as foreign keys in other record types. As with any foreign key, the target (surrogate) record type is implied by its use.

The conversion from various sorts of keys, such as SSAN or vehicle registration number, to a related surrogate is often made by the host software. In such a case, the host controller, using the transaction's PCTEH entry, invokes the first named module to convert the 'key value' to a surrogate. If none is found, an error is returned. If one is found, control goes to the second and succeeding module(s) named in the PCTEH entry to complete the business function processing. If two or more 'matches' are found, the duplicate transaction specified is invoked to switch function and return the first screen of duplicates.

On the applications side of the system, there are requirements to manipulate lists of surrogate keys for multiple record types. To differentiate them, a one-byte prefix is defined for the six types used and then prefixed to the 4-byte surrogate key. The six types and their values are listed below. The following few lines from the copy book UGZCOMMC show the structure in which they are placed.

```
10      UGCOMMON-RESERVED-SURROGATES.
        15    UGCOMMON-RESERVED-SURR-LIST
                    OCCURS 12 TIMES   INDEXED BY
                        UGCOMMON-RESERVED-SURR-INDEX.
              20  UGCOMMON-RESERVED-SURR-TYPE       PIC X.
              20  UGCOMMON-RESERVED-SURR-VALUE     PIC S9(9) COMP.
```

The order for the record types (which may only be for a single type) is dictated for F4 and F9 by the 1 to 4 type codes in the fields UGCOMMON-F4-STRING and UGCOMMON-F9-STRING. These values are set by the System Control Program during initialization from the values in the corresponding strings located in the PCTEG entry. The code segment for these fields follows:

```
10  UGCOMMON-CUR-FUNC-CTRL-FLAGS.
        15  UGCOMMON-PROCESS-BYTE-1      PIC X.
        15  UGCOMMON-PROCESS-BYTE-2      PIC X.
        15  UGCOMMON-PROCESS-BYTE-3      PIC X.
        15  UGCOMMON-PROCESS-BYTE-4      PIC X.
        15  UGCOMMON-USER-FLAGS.
              20  UGCOMMON-PF4-STRING    PIC X(04).
              20  UGCOMMON-PF9-STRING    PIC X(04).
```

For scrolling, there is a different table that will hold 140 surrogate keys only. The type identification in this case is made from the 1 to 8 type codes stored in UGCOMMON-SCROLL-SURR-TYPE-TBL. Whatever sequence is here is assumed to repeat enough times to cover the number of surrogate keys shown by the index. The fields involved are as follows:

```
05  UGCOMMON-SCROLL-BASE-DATA                   PIC X(742).
05  FILLER REDEFINES UGCOMMON-SCROLL-BASE-DATA.
        10  UGCOMMON-LAST-SCROLL-FUNC           PIC X(04).
        10  FILLER                              PIC X(02).
        10  UGCOMMON-SCROLL-FORMAT              PIC X(12).
        10  FILLER REDEFINES UGCOMMON-SCROLL-FORMAT.
              15  UGCOMMON-SCROLL-ENT-PER-LINE  PIC 9.
              15  UGCOMMON-SCROLL-SURR-PER-ENT  PIC 9.
              15  UGCOMMON-SCROLL-DATA-LINE1    PIC S9(4) COMP.
              15  UGCOMMON-SCROLL-SURR-TYPE-TBL PIC X(08).
```

NOTE: Intervening fields are omitted to save space.

```
10  UGCOMMON-SCROLL-TABLE                PIC X(560).
10  FILLER REDEFINES UGCOMMON-SCROLL-TABLE.
      15  UGCOMMON-SURR-NUMB OCCURS 140 TIMES
                              INDEXED BY UGC-SURR-INDEX
                                   PIC S9(09) COMP.
05  UGCOMMON-APPLICATION-WORK-AREA       PIC X(512).
```

The Six Types of 'Application' Surrogates defined are:

0 = VEHR [S] (registration) 1 = VEHC [S] (veh. claim)

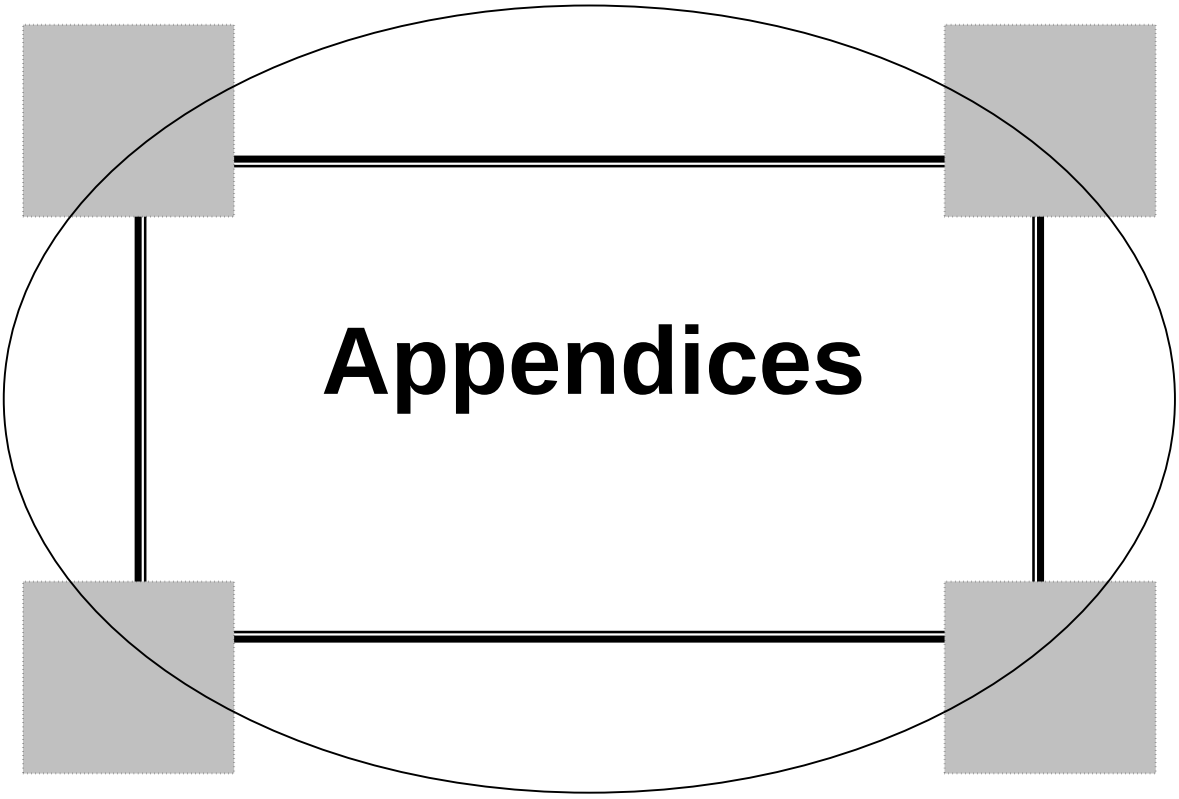
2 = VEHT [S] (veh. title) 3 = VMOD [I] (veh. model)

4 = PERS [S] (person record) 5 = NOWN [S] (corp. owner)

Low-Values or Spaces = None

NOTE: [S] surrogate is key for record stored CAL-S
[I] " " " " " IDMSCALC

NOTE: Additional mention of surrogates is found in the screen-hop function discussion on page 118.


Appendices

Appendix A:

Non-UMS LXTABLE Edit Example

NON-UMS LXTABLE DISCUSSION

In its native form, the UMS guest provides a front and back end service mechanism which accomplishes a number of common tasks. These tasks fall into three categories:

1. Conversion of data from external format to internal format.
2. Editing of common characteristics and enforcement of high level business rules upon input data.
3. Output conversion/formatting of internal data.

These services are grouped within a unit known as the LXTABLE. The name itself is relational to another service presented in a different system, and has no particular mnemonic relationship to the current system.

The intent of these services is to provide a table driven mechanism that performs common functionality, as well as insure that there is only one place to exit out of the system. The design is such that the component of the "rules" is resident on the HOST computer, and downloaded to each GUEST computer at startup time each day. This allows the RMV flexibility within the rules while being able to avoid frequent GUEST installations/maintenance.

The business rules are of primary importance to the RMV, as is the following of these rules by each GUEST. At the beginning of this document, the three tasks performed by LXTABLE were outlined. The first two tasks serve this purpose in any GUEST site that is not running the native RMV GUEST. Since these services ARE accessed within a CICS region by programs that are NOT part of the native RMV GUEST, the purpose of this package is to show how this is accomplished with an example program written in COBOL.

In a native UMS GUEST application, the LXTABLE services are invoked automatically by the dispatch/control services. The application program is not invoked if errors are detected. In a similar fashion, the conversion for output is automatic and transparent to the application. If the reader tries to compare the sample COBOL program with any UMS application, the conclusion is that there is no similarity. The reason for this is that in a NON UMS application, the application program is required to perform various functions which normally are the responsibility of the dispatch/control function.

In essence, the functionality of LXTABLE is simple. The programmer must construct an editing/conversion template (known as the LXTABLE). This load module is truly a table which relates input areas (raw data) with storage locations for processing (in a COMMAREA), describes how they are to be converted as well as describes what generic class of editing is to be applied.

When the editing function is invoked, the relations are processed based on the table. Conversions are performed, default values may be inserted and editing specifications are checked. If errors are detected, each is flagged and returned along with the error-code applicable to the first detected. First detected relates to the order in which the fields were specified in the LXTABLE. Normally, in the RMV usage, this relates to left-to-right, top-to-bottom construct of the input screen. This is a convention, not a technical requirement.

The process can handle all fields at once (as in the RMV usage), or process singular groups of fields. The minimum number of fields in a group is those needed to analyze all involved editing specifications. In the sample, which has two cases, one case (policy dates) requires two fields and the other (registration) requires three fields.

If several groups of fields are processed, it is important not merely to know if ALL fields passed the edits (as in the sample), but which DID NOT. In the sample, an item called EDIT-THIS-FIELD is moved to a 3 byte KEY field for each item to be processed. If the status returns non-zero, each KEY field which corresponds to an item which failed the edit will no longer have a value equal to EDIT-THIS-FIELD.

Sample Map, Non-UMS LXTABLE Invocation

```
= BMS map used by sample program. Provided for reference only.=
= it has NO applicability to the use of the technique.        =

USAMPLE  DFHMSD
            TYPE=MAP,
            CTRL=(FREEKB,FRSET),
            MODE=INOUT,
            STORAGE=AUTO,
            TERM=3270-2,
            TIOAPFX=YES
USAMPLE  DFHMDI
            SIZE=(24,80)
      DFHMDF POS=(001,023),LENGTH=0036,ATTRB=(ASKIP,BRT),
X00140000
            INITIAL='EXAMPLE OF USING UMS EDIT
      DFHMDF POS=(002,027),LENGTH=0028,ATTRB=(ASKIP,BRT),
            INITIAL='FROM A NON-UMS APPLICATION'
*
      DFHMDF POS=(005,001),LENGTH=13,INITIAL='POLICY EFFDT:',
            ATTRB=ASKIP
FLD005A  DFHMDF POS=(005,015),LENGTH=10,ATTRB=(UNPROT,IC,FSET)
      DFHMDF POS=(005,026),LENGTH=07,INITIAL=' EXPDT:',
            ATTRB=ASKIP
FLD005B  DFHMDF POS=(005,034),LENGTH=10,ATTRB=(UNPROT,FSET)
      DFHMDF POS=(005,045),LENGTH=1,INITIAL=' ',ATTRB=ASKIP
*
      DFHMDF POS=(007,001),LENGTH=09,INITIAL='REG TYPE:',
            ATTRB=ASKIP
FLD007A  DFHMDF POS=(007,011),LENGTH=3,ATTRB=(UNPROT,FSET)
      DFHMDF POS=(007,015),LENGTH=09,INITIAL=' NUMBER:',
            ATTRB=ASKIP
FLD007B  DFHMDF POS=(007,025),LENGTH=7,ATTRB=(UNPROT,FSET)
      DFHMDF POS=(007,033),LENGTH=08,INITIAL=' COLOR:',
            ATTRB=ASKIP
FLD007C  DFHMDF POS=(007,042),LENGTH=1,ATTRB=(UNPROT,FSET)
      DFHMDF POS=(007,044),LENGTH=1,INITIAL=' ',ATTRB=ASKIP
*
FLD014A  DFHMDF POS=(014,001),LENGTH=79,ATTRB=ASKIP,INITIAL=' '
FLD015A  DFHMDF POS=(015,001),LENGTH=79,ATTRB=ASKIP,INITIAL=' '
      DFHMSD TYPE=FINAL
      END
```

Program Sample, Non-UMS LXTABLE Invocation

```
       IDENTIFICATION DIVISION.                                 00110000
       PROGRAM-ID.    USAMPLEP.                                 00120000
       DATE-COMPILED.                                           00130000
       ENVIRONMENT DIVISION.                                    00140000
       DATA DIVISION.                                           00150000
       WORKING-STORAGE SECTION.                                 00160000
       01  DATA-WORK-ITEMS.                                     00170001
==============================
these items are relative to   =
this sample program only,     =
and not the general technique =
                     v
        04  TEMP PIC S9(9) COMP.                                00180001
        04  MAPAREA.
     00190005
             08  MA-TIOT-PREFIX PIC X(12).                      00200005
             08  MA-FLD005A.                                    00210005
              12  MA-FLD005A-LEN  PIC S9(4) COMP.               00220005
              12  MA-FLD005A-ATTR PIC X.                        00230005
              12  MA-FLD005A-DATA PIC X(10).                    00240005
             08  MA-FLD005B.                                    00250005
              12  MA-FLD005B-LEN  PIC S9(4) COMP.               00260005
              12  MA-FLD005B-ATTR PIC X.                        00270005
              12  MA-FLD005B-DATA PIC X(10).                    00280005
             08  MA-FLD007A.                                    00290005
              12  MA-FLD007A-LEN  PIC S9(4) COMP.               00300005
              12  MA-FLD007A-ATTR PIC X.                        00310005
              12  MA-FLD007A-DATA PIC X(3).                     00320005
             08  MA-FLD007B.                                    00330005
              12  MA-FLD007B-LEN  PIC S9(4) COMP.               00340005
              12  MA-FLD007B-ATTR PIC X.                        00350005
              12  MA-FLD007B-DATA PIC X(7).                     00360005
             08  MA-FLD007C.                                    00370005
              12  MA-FLD007C-LEN  PIC S9(4) COMP.               00380005
              12  MA-FLD007C-ATTR PIC X.                        00390005
              12  MA-FLD007C-DATA PIC X(1).                     00400005
             08  MA-FLD014A.                                    00410005
              12  MA-FLD014A-LEN  PIC S9(4) COMP.               00420005
              12  MA-FLD014A-ATTR PIC X.                        00430005
              12  MA-FLD014A-DATA PIC X(79).                    00440005
             08  MA-FLD015A.                                    00450005
              12  MA-FLD015A-LEN  PIC S9(4) COMP.               00460005
              12  MA-FLD015A-ATTR PIC X.                        00470005
```

```
             12  MA-FLD015A-DATA PIC X(79).              00480005
          04  END-MESSAGE PIC X(27) VALUE                00490001
              'END OF EDIT-SAMPLE SESSION'.             00500001
          04  MAPLENGTH PIC S9(4) COMP VALUE +222.       00510005
          04  CONSTANT-1 PIC S9(4) COMP VALUE +125.      00520001
              04  FILLER REDEFINES CONSTANT-1.
      00530001
              08  FILLER PIC X.                          00540001
              08  HEX-7D PIC X.                          00550001
                      ^
 ==============================

 ==============================
 these items are constants     =
 applicable to the RMV data    =
 structures                =
                      v
          04  CONSTANT-2 PIC S9(9) COMP VALUE +32768.    00560005
              04  FILLER REDEFINES CONSTANT-2.
      00570005
              08  FILLER PIC XX.                         00580005
              08  NULL-DATE PIC XX.                      00590005
          04  CONSTANT-3 PIC S9(9) COMP VALUE +256.      00600008
              04  FILLER REDEFINES CONSTANT-3.           00610008
              08  FILLER PIC X.
      00620008
              08  EDIT-THIS-FIELD PIC XXX.               00630008

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

 ==============================
 this is the structure which   =
 defines the area containing   =
 the raw data to edit and      =
 convert. the 93 byte area at  =
 the start is required by the  =
 technique. the other items    =
 must line up with the          =
 template (LXTABLE). the 3byte =
 field (....-key) preceding   =
 each data item is required    =
                      v
          04  DUMMY-AREA.                              00640006
              08  DA-HEADER       PIC X(93).
      00650007
              08  DA-FLD005A.                            00660006
              12  DA-FLD005A-KEY  PIC X(3).              00670006
```

```
             12  DA-FLD005A-DATA PIC X(10).            00680006
            08  DA-FLD005B.                           00690006
             12  DA-FLD005B-KEY  PIC X(3).             00700006
             12  DA-FLD005B-DATA PIC X(10).            00710006
            08  DA-FLD007A.                           00720006
             12  DA-FLD007A-KEY  PIC X(3).             00730006
             12  DA-FLD007A-DATA PIC X(3).             00740006
            08  DA-FLD007B.                           00750006
             12  DA-FLD007B-KEY  PIC X(3).             00760010
             12  DA-FLD007B-DATA PIC X(7).             00770006
            08  DA-FLD007C.                           00780006
             12  DA-FLD007C-KEY  PIC X(3).             00790010
             12  DA-FLD007C-DATA PIC X(1).             00800006
                    ^
==============================

==============================
these items are relative to   =
this sample program only,     =
and not the general technique =
                    v
        04  MSG1.                                    00810006
            08  FILLER PIC X(8) VALUE ' EFFDT:'.    00820006
            08  M1F1.                               00830006
             12  M1F1-MONTH PIC 99.                      00840009
             12  M1F1-DASH1 PIC X.                   00850006
             12  M1F1-DAY PIC 99.                    00860009
             12  M1F1-DASH2 PIC X.                   00870006
             12  M1F1-YEAR  PIC 9999.                00880006
            08  FILLER PIC X(8) VALUE ' EXPDT:'.    00890006
            08  M1F2.                               00900006
             12  M1F2-MONTH PIC 99.                      00910009
             12  M1F2-DASH1 PIC X.                   00920006
             12  M1F2-DAY PIC 99.                    00930009
             12  M1F2-DASH2 PIC X.                   00940006
             12  M1F2-YEAR  PIC 9999.                00950006
            08  FILLER PIC X VALUE ' '.             00960006
            08  M1F3.                               00970006
             12  M1F3-ERROR PIC X(6).                00980006
             12  M1F3-CODE  PIC 9(9).                00990006
        04  MSG2.                                   01000010
            08  FILLER PIC X(8) VALUE ' REGNO:'.    01010010
            08  M2F1-PREFIX PIC X(4).               01020010
            08  M2F1-REGNO  PIC X(8).               01030010
            08  M2F1-COLOR  PIC X(2).               01040010
            08  FILLER PIC X(14) VALUE SPACES.      01050010
            08  FILLER PIC X VALUE ' '.             01060010
```

```
           08  M2F3.                                  01070010
            12  M2F3-ERROR PIC X(6).                  01080010
            12  M2F3-CODE  PIC 9(9).                  01090010

==============================
       LINKAGE SECTION.                               01100000
==============================
these items are relative to   =
this sample program only,     =
and not the general technique =
NOTE that a BLL cell for a UMS=
GUEST common area is needed   =
but its position depends on   =
the actual program structure  =
                     v
       01  DFHCOMMAREA.                               01110000
        04 FILLER PIC X(4).                           01120000
       01  FILLER.                                    01130000
        05  FILLER       PIC S9(9) USAGE COMP.        01140000
        05  UMS-COMMAREA-ADDRESS   PIC S9(9) USAGE COMP.

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

==============================
this copybook (or an equiv-   =
alent) is required.       =

       01  UGCOMMON-COMMAREA.                                 01160001
        02  UGZCOMMA.                                        01170001
            COPY UGZCOMMC.                                  01180001
                     ^
==============================

==============================
this filler provides access to=
a field normally reserved for =
dispatch/control and also     =
positions for definitions of  =
the output fields from the    =
edit process.                 =
                     v
        02  FILLER REDEFINES UGZCOMMA.                      01190001
            04  FILLER PIC X(16).                          01200001
            04  UGCALLON PIC X.                            01210001
            04  FILLER PIC X(1767).
     01220001
                     ^
```

```
==============================
* OUTPUT DATA FIELDS COME HERE. THEY MUST BE AT THE SAME OFFSETS

*  WITHIN GUEST COMMON AS EXPRESSED IN THE ASSOCIATED LXTABLE.

*  EACH IS PRECEEDED BY ITS Z AND T FIELD BYTES.
==============================
these fields receive the      =
output from the edit process. =
the "...Z" and "...T" fields  =
are required by the technique.=
the "Z" fields may be useful  =
for the purposes of editing   =
from a non-UMS application,   =
but the "T" fields are not.   =
                     v
            04  SAEFFDT-GROUP.
     01260006
             08  SAEFFDT-Z PIC X.                        01270006
             08  SAEFFDT-T PIC X.                        01280006
             08  SAEFFDT  PIC XX.                        01290006
            04  SAEXPDT-GROUP.
     01300006
             08  SAEXPDT-Z PIC X.                        01310006
             08  SAEXPDT-T PIC X.                        01320006
             08  SAEXPDT  PIC XX.                        01330006
            04  SAPREFIX-GROUP.                         01340006
             08  SAPREFIX-Z PIC X.                       01350006
             08  SAPREFIX-T PIC X.                       01360006
             08  SAPREFIX   PIC X(3).                    01370006
            04  SAREGNO-GROUP.
     01380006
             08  SAREGNO-Z PIC X.                        01390006
             08  SAREGNO-T PIC X.                        01400006
             08  SAREGNO  PIC X(7).
     01410006
            04  SACOLOR-GROUP.
     01420006
             08  SACOLOR-Z PIC X.                        01430006
             08  SACOLOR-T PIC X.                        01440006
             08  SACOLOR  PIC X(1).
     01450010
                     ^
==============================
       PROCEDURE DIVISION.                               01460000
       0000-SETUP-PROGRAM.                               01470000
==============================
```

```
this code is just relative    =
to the sample program and has =
no relevance to the technique.=
                      v
**** TEST FOR FIRST-TIME THROUGH,
**** SEND FIRST EMPTY SCREEN IF SO:
     01480001
        IF EIBCALEN = ZERO                                01490001
            EXEC CICS SEND MAP('USAMPLE') MAPONLY ERASE
              FREEKB END-EXEC                             01510001
            EXEC CICS RETURN COMMAREA(TEMP) LENGTH(4)
              TRANSID(EIBTRNID) END-EXEC.                 01530001
******* INPUT MAP:                                        01540001
        EXEC CICS RECEIVE MAP('USAMPLE') INTO(MAPAREA)
                      NOHANDLE END-EXEC.                  01560001
******* TEST TERMINATION (ANY KEY EXCEPT ENTER):          01570001
        IF EIBAID NOT = HEX-7D
     01580001
            EXEC CICS SEND TEXT FROM(END-MESSAGE) CURSOR(1)
             LENGTH(27) ERASE FREEKB END-EXEC
     01600001
            EXEC CICS RETURN END-EXEC.                    01610001
                      ^
================================

================================
UIGETCOM returns a skeleton   =
UMS common area with a number =
of address constants plugged  =
                      v
******* GET A UMS-GUEST COMMON AREA:                      01620001
        EXEC CICS LINK    PROGRAM('UIGETCOM')
     01630001
                    COMMAREA(UMS-COMMAREA-ADDRESS)
     01640001
                    LENGTH(4) END-EXEC.                   01650001

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

================================
tell COBOL that a related BLL =
cell has changed          =
                      v
        SERVICE RELOAD UGCOMMON-COMMAREA.                 01660000
                      ^
================================
```

```
******* INITIALIZE CONSTANTS NORMALLY PROVIDED BY
DISPATCH-CONTROL
==============================
the edit functions require the=
current date in RMV format    =
get it from CICS in Julian    =
format and use the UMS service=
module to convert it to the   =
RMV format                    =

        MOVE EIBDATE TO UGCOMMON-DATE-JULIAN-FORMAT.   01680005
        MOVE '2' TO UGCOMMON-DATE-INPUT-FORMAT.        01690005
        CALL 'UICALLST' USING UGCOMMON-DATE-ROUTINE-ADDRESS
                     UGCOMMON-COMMAREA.                01710005
        MOVE UGCOMMON-DATE-BINARY-FORMAT TO            01720005
               UGCOMMON-CURRENT-BINARY-DATE.           01730005
                     ^
==============================

==============================
setup items required to show  =
that the next phase is edit   =
                     v
        MOVE +01 TO UGCOMMON-ENTRY-REASON.             01740005
        MOVE 'Y' TO UGCOMMON-MESSAGE-TEXT.             01750005
                     ^
==============================

==============================
this code is just relative    =
to the sample program and has =
no relevance to the technique.=
                     v
******* DO SOME PROCESSING RELATIVE TO
*******   THE TEST MAP WE JUST INPUT:
        MOVE SPACES TO MA-FLD014A-DATA, MA-FLD015A-DATA.
        IF MA-FLD005A-DATA = (LOW-VALUES OR SPACES) AND
           MA-FLD005B-DATA = (LOW-VALUES OR SPACES) AND
           MA-FLD007A-DATA = (LOW-VALUES OR SPACES) AND
           MA-FLD007B-DATA = (LOW-VALUES OR SPACES) AND
           MA-FLD007C-DATA = (LOW-VALUES OR SPACES)
           MOVE 'NO DATA WAS ENTERED' TO MA-FLD014A-DATA
           GO TO 9999-EXIT.                            01840005
                     ^
==============================

==============================
```

```
DUMMY-AREA will contain data  =
to be edited. in fact, the    =
edit services will treat it as=
if it were an input map       =
composite. a valid length is  =
required.           =
                    v
******* CONSTANT BELOW SHOULD REPRESENT THE TRUE LENGTH
DUMMY-AREA
        MOVE +58 TO UGCOMMON-RECEIVED-MAP-LENGTH.
                    ^
==============================

==============================
since DUMMY-AREA is equivalent=
to a map, its address is      =
required in both map address  =
cells in the common area. UMS =
services can be used to get   =
this value:                   =
                    v
******* ESTABLISH ADDRESSES OF DUMMY MAP AREA:01870006
        CALL 'UICALLST' USING UGCOMMON-DATA-NAME-ADDRESS
                    DUMMY-AREA, UGCOMMON-INPUT-MAP-ADDRESS.
        MOVE UGCOMMON-INPUT-MAP-ADDRESS TO
               UGCOMMON-OUTPUT-MAP-ADDRESS.
                    ^
==============================

==============================
the name of the user const-   =
ructed edit template (LXTABLE)=
must be placed in the common  =
area                          =
                    v
******* SET EDIT-TABLE NAME:                              01920006
        MOVE 'USAMPLET' TO UGCOMMON-MAP-TRANSLATION-TBL.
                    ^
==============================

==============================
this code is just relative    =
to the sample program and has =
no relevance to the technique.=
                    v
*******                                                  01940006
*******                                                  01950006
```

```
******* PROCESS EFFDT/EXPDT:                                  01960006
*******                                                      01970006
        IF MA-FLD005A-DATA = (LOW-VALUES OR SPACES) AND
           MA-FLD005B-DATA = (LOW-VALUES OR SPACES)
           MOVE 'NO POLICY DATE(S)' TO MA-FLD014A-DATA
           GO TO 1000-DO-REGNO.
                    ^
==============================

==============================
the edit services normally are=
used to process many fields at=
the same time. this technique =
edits all fields, but the ret-=
urned code is only for the 1st=
error, with the checking in   =
the order the definitions     =
occur in the template. this   =
example makes multiple passes =
to show all codes. the verbage=
documentation describes how   =
to tell which fields are in   =
error if multiple fields are  =
edited at once.                 =
                    v
******* THE NEXT 4 LINES OF CODE ARE IN ORDER FOR EACH
*******   EDITING PASS:
        MOVE LOW-VALUES TO UGCOMMON-PROGRAM-COMMAREA.
        MOVE HIGH-VALUES TO UGCALLON.
        MOVE ZERO TO UGCOMMON-CURRENT-ERROR-CODE,
               UGCOMMON-CURRENT-MAP-LENGTH.
                    ^
==============================

==============================
the conversion output fields  =
must be set to the default    =
value for the field mode.     =
                    v
******* INITIALIZE INVOLVED FIELDS TO DEFAULT VALUE(S)
     02070006
        MOVE NULL-DATE TO SAEFFDT, SAEXPDT.
     02080006
                    ^
==============================

==============================
```

```
only the fields to edit are   =
set in DUMMY-AREA. they must  =
be flagged as edit candidates.=
                      v
******* CLEAR DUMMY-AREA, PLUG INVOLVED FIELDS:
        MOVE LOW-VALUES TO DUMMY-AREA.
        MOVE MA-FLD005A-DATA TO DA-FLD005A-DATA.
        MOVE MA-FLD005B-DATA TO DA-FLD005B-DATA.
******* INDICATE WHICH FIELDS ARE TO BE PROCESSED:
        MOVE EDIT-THIS-FIELD TO DA-FLD005A-KEY
                                DA-FLD005B-KEY.
                      ^
================================

================================
this code is just relative    =
to the sample program and has =
no relevance to the technique.=
                      v
        MOVE SPACES TO MA-FLD005A-DATA, MA-FLD005B-DATA.
                      ^
================================

================================
invoke the edit process. note =
that a UGCA abend means that  =
the UMS commarea was not      =
obtained properly. a UGTL     =
abend means that there is not =
a running UMS guest in the    =
region. UGTL only occurs if   =
the particular edit requires  =
host download data.           =
                      v
******* INVOKE THE EDIT PROCESS:
        EXEC CICS LINK    PROGRAM('UGZ0006P')
                     COMMAREA(UGCOMMON-COMMAREA)
                     LENGTH(UGCOMMON-XCTL-LENGTH) END-EXEC.
                      ^
================================

================================
this code is just relative    =
to the sample program and has =
no relevance to the technique.=
it is worthy of observation  .=
because it demonstrates the   =
```

```
UMS field cascading technique,=
testing the error-code field  =
and converting binary dates   =
for output. note that date    =
conversion specifies the input=
format and all other formats  =
are always returned. in a     =
binary conversion the status  =
will always be valid.         =
                    v
******* THE REST OF THIS PARAGRAPH SIMPLY FORMATS FOR THE SAMPLE
*******   DISPLAY. THE ESSENCE OF THE LOGIC IS THAT
*******   IF UGCOMMON-CURRENT-ERROR-CODE = ZERO,
*******   THE EDIT WAS PASSED AND THE CONVERTED CORE-IMAGE
*******   WAS BUILT.
*******
******* ASSUME THAT EDIT/REFORMAT MIGHT HAVE CHANGED
*******   THE MAP IMAGE:
        MOVE DA-FLD005A-DATA TO M1F1.
        MOVE DA-FLD005B-DATA TO M1F2.
******* CONVERT INTERNAL FIELDS TO DISPLAY IF THEY WERE INPUT
OK:
        IF SAEFFDT NOT = NULL-DATE
            MOVE SAEFFDT TO UGCOMMON-DATE-BINARY-FORMAT
            MOVE '0' TO UGCOMMON-DATE-INPUT-FORMAT
            CALL 'UICALLST' USING UGCOMMON-DATE-ROUTINE-ADDRESS
                          UGCOMMON-COMMAREA
            MOVE UGCOMMON-DATE-GREG-DAY TO M1F1-DAY
            MOVE UGCOMMON-DATE-GREG-MONTH TO M1F1-MONTH
            MOVE UGCOMMON-DATE-GREG-YEAR TO M1F1-YEAR
            MOVE '/' TO M1F1-DASH1, M1F1-DASH2.
        IF SAEXPDT NOT = NULL-DATE
            MOVE SAEXPDT TO UGCOMMON-DATE-BINARY-FORMAT
            MOVE '0' TO UGCOMMON-DATE-INPUT-FORMAT
            CALL 'UICALLST' USING UGCOMMON-DATE-ROUTINE-ADDRESS
                          UGCOMMON-COMMAREA
            MOVE UGCOMMON-DATE-GREG-DAY TO M1F2-DAY
            MOVE UGCOMMON-DATE-GREG-MONTH TO M1F2-MONTH
            MOVE UGCOMMON-DATE-GREG-YEAR TO M1F2-YEAR
            MOVE '/' TO M1F2-DASH1, M1F2-DASH2.
******* DISPLAY "OK" OR THE ERROR-CODE:
        IF UGCOMMON-CURRENT-ERROR-CODE = ZERO
            MOVE 'OK' TO M1F3
            ELSE
            MOVE 'ERROR ' TO M1F3-ERROR
            MOVE UGCOMMON-CURRENT-ERROR-CODE TO M1F3-CODE.
******* PUT THE MESSAGE IN THE OUTPUT MAP
```

```
        MOVE MSG1 TO MA-FLD014A-DATA.
                     ^
==============================
       1000-DO-REGNO.
*******
*******
******* PROCESS REG PREFIX, NUMBER, COLOR:
*******
==============================
this code is just relative    =
to the sample program and has =
no relevance to the technique.=
                     v
        IF MA-FLD007A-DATA = (LOW-VALUES OR SPACES) AND
           MA-FLD007B-DATA = (LOW-VALUES OR SPACES) AND
           MA-FLD007C-DATA = (LOW-VALUES OR SPACES)
           MOVE 'NO REGISTRATION DATA' TO MA-FLD015A-DATA
           GO TO 9999-EXIT.
                     ^
==============================

==============================
the edit services normally are=
used to process many fields at=
the same time. this technique =
edits all fields, but the ret-=
urned code is only for the 1st=
error, with the checking in   =
the order the definitions     =
occur in the template. this   =
example makes multiple passes =
to show all codes. the verbage=
documentation describes how   =
to tell which fields are in   =
error if multiple fields are  =
edited at once.               =
                     v
******* THE NEXT 4 LINES OF CODE ARE IN ORDER FOR EACH EDITING
PASS:
        MOVE LOW-VALUES TO UGCOMMON-PROGRAM-COMMAREA.
        MOVE HIGH-VALUES TO UGCALLON.
        MOVE ZERO TO UGCOMMON-CURRENT-ERROR-CODE,
               UGCOMMON-CURRENT-MAP-LENGTH.
                     ^
==============================

==============================
```

```
the conversion output fields  =
must be set to the default    =
value for the field mode.     =
                      v
******* INITIALIZE INVOLVED FIELDS TO DEFAULT VALUE(S)
        MOVE SPACES TO SAPREFIX, SAREGNO, SACOLOR.
                      ^
================================

================================
only the fields to edit are   =
set in DUMMY-AREA. they must  =
be flagged as edit candidates.=
                      v
******* CLEAR DUMMY-AREA, PLUG INVOLVED FIELDS:
        MOVE LOW-VALUES TO DUMMY-AREA.
        MOVE MA-FLD007A-DATA TO DA-FLD007A-DATA.
        MOVE MA-FLD007B-DATA TO DA-FLD007B-DATA.
        MOVE MA-FLD007C-DATA TO DA-FLD007C-DATA.
******* INDICATE WHICH FIELDS ARE TO BE PROCESSED:
        MOVE EDIT-THIS-FIELD TO DA-FLD007A-KEY
                                DA-FLD007B-KEY
                                DA-FLD007C-KEY.
                      ^
================================

================================
this code is just relative    =
to the sample program and has =
no relevance to the technique.=
                      v
        MOVE SPACES TO MA-FLD007A-DATA, MA-FLD007B-DATA
                               MA-FLD007C-DATA.
                      ^
================================

================================
invoke the edit process. note =
that a UGCA abend means that  =
the UMS commarea was not      =
obtained properly. a UGTL     =
abend means that there is not =
a running UMS guest in the    =
region. UGTL only occurs if   =
the particular edit requires  =
host download data.           =
                      v
```

```
******* INVOKE THE EDIT PROCESS:
        EXEC CICS LINK    PROGRAM('UGZ0006P')
                  COMMAREA(UGCOMMON-COMMAREA)
                  LENGTH(UGCOMMON-XCTL-LENGTH) END-EXEC.
                   ^
==============================

==============================
this code is just relative    =
to the sample program and has =
no relevance to the technique.=
it is worthy of observation  .=
because it demonstrates the   =
UMS field cascading technique,=
and testing the error-code    =
field.                        =
                   v
******* THE REST OF THIS PARAGRAPH SIMPLY FORMATS FOR THE SAMPLE
*******   DISPLAY. THE ESSENCE OF THE LOGIC IS THAT
*******   IF UGCOMMON-CURRENT-ERROR-CODE = ZERO,
*******   THE EDIT WAS PASSED
*******   AND THE CONVERTED CORE-IMAGE WAS BUILT.
*******
******* ASSUME THAT EDIT/REFORMAT MIGHT HAVE CHANGED
*******   THE MAP IMAGE:
        MOVE DA-FLD007A-DATA TO M2F1-PREFIX.
        MOVE DA-FLD007B-DATA TO M2F1-REGNO.
        MOVE DA-FLD007C-DATA TO M2F1-COLOR.
******* CONVERT INTERNAL FIELDS TO DISPLAY IF THEY WERE INPUT
OK:
        IF SAPREFIX NOT = SPACES MOVE SAPREFIX   TO M2F1-PREFIX.
        IF SAREGNO  NOT = SPACES MOVE SAREGNO    TO M2F1-REGNO.
        IF SACOLOR  NOT = SPACES MOVE SACOLOR    TO M2F1-COLOR.
******* DISPLAY "OK" OR THE ERROR-CODE:
        IF UGCOMMON-CURRENT-ERROR-CODE = ZERO
            MOVE 'OK' TO M2F3
            ELSE
            MOVE 'ERROR ' TO M2F3-ERROR
            MOVE UGCOMMON-CURRENT-ERROR-CODE TO M2F3-CODE.
******* PUT THE MESSAGE IN THE OUTPUT MAP
        MOVE MSG2 TO MA-FLD015A-DATA.
                   ^
==============================
       9999-EXIT.
==============================
cleanup of the common area    =
is important, in particular if=
```

```
the logic is going in some    =
resident program which hangs  =
on data availability. the     =
UIGETCOM routine will return  =
new areas on successive calls.=
                      v
        EXEC CICS FREEMAIN DATA(UGCOMMON-COMMAREA) END-EXEC.
                      ^
==============================

==============================
this code is just relative    =
to the sample program and has =
no relevance to the technique.=
                      v
        MOVE 'A' TO MA-FLD005A-ATTR, MA-FLD005B-ATTR,
                 MA-FLD007A-ATTR, MA-FLD007B-ATTR,
                 MA-FLD007C-ATTR.
        EXEC CICS SEND MAP('USAMPLE') DATAONLY NOHANDLE
          FROM(MAPAREA) LENGTH(MAPLENGTH) FREEKB END-EXEC.
        EXEC CICS RETURN COMMAREA(TEMP) LENGTH(4)
          TRANSID(EIBTRNID) END-EXEC.
        GOBACK.
                      ^
==============================
```

Sample LXTABLE, Non-UMS Invocation

```
= editing TEMPLATE (LXTABLE) used by sample program =

============================
required:         =
                  v
      TITLE 'USAMPLET - UMS LX SCREEN '           00092000
      COPY  UMSLXMAC                              00093000
      PUSH  PRINT                                 00094000
      PRINT OFF                                        00095000
      COPY  UGZCOMMA                              00096000
      POP   PRINT                                 00097000
      ORG   UGCUSERA                              00098100
                  ^
============================

============================
conversion output fields.  =
this definition must match =
the definition (format,    =
length and offset) in the  =
invoking program.          =
(UMS commarea)             =
                  v
SAEFFDT  GFLD HL2        EFFECTIVE-DATE                00098300
SAEXPDT  GFLD HL2        EXPIRATION-DATE
     00098400
SAPREFIX GFLD CL3        PREFIX                        00098500
SAREGNO  GFLD CL7        REGNO
     00098600
SACOLOR  GFLD CL1        COLOR
     00098700
      ORG                                           00104400
*                                                   00104500
                  ^
============================

============================
required, name in col-1    =
must be load-module name.  =
NOMAPNA should be used for =
all non-UMS standard usage =
                  v
      EJECT                                         00104600
```

```
USAMPLET UMSLXTBL TYPE=START,MAPNAME=NOMAPNA,LEVEL0=NO 00105000
                  ^
============================

============================
date specifications. the   =
effective-date happens to  =
be first, and the expire   =
date second. the editing   =
specification MUST be on   =
the last entry.                =
                  v
      UMSLXTBL MAPFLD=FLD005A,GSAFLD=SAEFFDT      00110000
      UMSLXTBL MAPFLD=FLD005B,GSAFLD=SAEXPDT,          X00120000
          EDIT=(PDATE,SAEFFDT,SAEXPDT)            00130000
                  ^
============================

============================
reg specifications. the    =
order happens to be prefix,=
regno and color. the           =
editing specification MUST =
be on the last entry.      =
                  v
      UMSLXTBL MAPFLD=FLD007A,GSAFLD=SAPREFIX          00140000
      UMSLXTBL MAPFLD=FLD007B,GSAFLD=SAREGNO           00141000
      UMSLXTBL MAPFLD=FLD007C,GSAFLD=SACOLOR,          X00142000
            EDIT=(REGNO,SAPREFIX,SACOLOR,SAREGNO) 00160000
      EJECT                                       00490000
                  ^
============================

============================
required:         =
                  v
      UMSHEADR TYPE=DSECT                              00500000
                  ^
============================

============================
conversion input fields.   =
this definition must match =
the definition (format,    =
length and offset) in the  =
invoking program.      =
(DUMMY-AREA)           =
```

```
                  v
      DS CL3        BMS DATA                          00510000
FLD005A  DS CL10         EFFDT                             00520000
      DS CL3        BMS DATA                          00521000
FLD005B  DS CL10         EXPDT                             00522000
      DS CL3        BMS DATA                          00530000
FLD007A  DS CL3          PREFIX                        00540000
      DS CL3        BMS DATA                          00550000
FLD007B  DS CL7          REGNO                             00560000
      DS CL3        BMS DATA                          00570000
FLD007C  DS CL1          COLOR                             00580000
                  ^
============================

============================
required:         =
                  v
      END                                        01260300
                  ^
============================
```