ENHANCEMENT GUIDE
FOR
FICE + Release 3.01

Table of Contents

Section 1: Run-Time Environment

Logon - $F BGN 1	1-1
Menu Driver - $F MENU	1-2
Terminal Traits	1-3
Menu Tracking	1-3
Direct Menu Options	1-3
Set Time - $F STM 1	1-4
Program Loader - $F LOAD	1-4
Clock and Disk Polling Rates	1-5

Section 2: RMS - Miscellaneous

File Access Class Enforcement	2-1
Inter-terminal Record Protection	2-1
New Hashing Algorithm	2-2
Extended Platter and Sector Addressing	2-3
Working with Pseudo-Deleted Records	2-4
Sequential Access Buffers	2-5
Other Performance Improvements	2-7
Undocumented Code Removed from '246	2-7
Support of Fasstcom "T" CPU-Based Disks	2-7

Section 3: Keyboard Entry and Date Processing

Special Intercepts	3-1
Blank Range Checks	3-3
Cursor Positioning	3-3
Simplifications	3-4
Dates 8-4	
Miscellaneous Changes	3-5

Section 4: File Maintenance

File Maintenance Changes in the 3.x Releases	4-1
File Scanning	4-6
File Maintenance Phases	4-8

Section 5: Sorting, Merging, and Report Printing

Sorting	5-1
Merging	5-2
Printer Selection	5-2
Report Printing	5-2

Table (continue)

Section 6: General Inquiry

Alternate Key Support	6-1
Scan Key Support	6-1
$F '250 Emulation	6-1

Section 7: Miscellaneous Changes

Section 8: FICE+ Utilities Module Menus

Menu #1 -System Utilities	7-1
Menu #2 -Module	7-2
Menu #3 -Definition Files	7-3
Menu #4 -Record Files	7-4
Menu #5 -Programming Utilities	7-6
Menu #6 -Disk Utilities	7-6
Menu #7 -Security	7-7

Section 9: Definition Files

New Concepts	8-1
Definition File Enhancements	8-2
Definition File Parameters	8-4
Definition File Fields	8-5
Definition File Screens	8-6
Definition File Processing	8-6

Section 10: New Utilities

Display File Parameter 9-1
Close Files 9-1
Compress Record File 9-1 Record File
Editor 9-1 Disk Editor 9-2

Section 11: Other FICE+ Utilities Changes

Restructure Record File	10-1
Create Files by Module	10-2

Section 12: System Parameters

New and Modified Fields	11-1
Other System Parameter Changes	11-3
Device Table	11-4

Table (continue)

Section 13: The FICE+ Module ($F)

FICE+ Programs 12-1
FICE+ Program Descriptions 12-2
FICE+ Special Subroutines 12-12
Special Subroutine Descriptions 12-12

Section 14: FICE+ Subroutines

FICE+ Subroutine Files 13-1
FICE+ Subroutine Descriptions 13-4
'180 - Position Cursor 13-5
'181 - Numeric Input 13-6
'182 - Alphanumeric Input 13-9
'183 - Display Error Message 13-13
'186 - Add Days to Date 13-14
'187 - Pack Date 13-15
'188 - Unpack Date 13-16
'189 - Days Between Two Dates 13-17
'190 - Prepare Printer 13-18
'191 - Prepare Printer for Special Forms 13-19
'193 - Forms Alignment 13-20
'194 - Select CRT 13-21
'195 - Select Printer with Message 13-22
'196 - Select Printer 13-23
'197 - Standard Page Heading 13-24
'198 - Form Feed 13-25
'199 - Update Line Count 13-26
'200 - Record Access 13-27
'201 - Open FICE+ File (with Return Code) 13-28
'202 - ISAM File Operation 13-29
'203 - Write Bytes 13-31
'204 - Read Bytes 13-32
'205 - Change Key/Delete Status 13-33
‘206 - Get Key/Delete Status 13-35
'207- Change Record Protection 13-36
'208 - Get Modules in System 13-37
'209 - Exclusive Access 13-38
'210 - One-Level File Maintenance 13-39
'211 - Open FICE+ File (No Return Code) 13-40
'212 - Set Logical Record Pointer 13-41
'213 - Write Record 13-42
'214 - Read Record 13-43
'215 - Add Record 13-44
'216 - Delete Record 13-45
'217 - Add Key 13-46
'218 - Delete Key 13-47
'219 - Find Key 13-48

Table(continue)

'220 - Two-Level File Maintenance 13-49
'221 - Physical Sequential Access (Setup) 13-50

'222 - Physical Sequential Access (Read) 13-51
'223 - Key Sequential Access (Setup) 13-52
'224 - Key Sequential Access (Read) 13-53
‘225 - Read Next Second-Level Record 13-54
'226 - Store and Display Field 13-55
'227 - Skip Fields on Input 13--56
'230 - Sort 13-57
'231 - Open File for Sorting 13-58
'232 - Establish Sort Sequence 13-59
'233 - Edit Sort Sequence 13-60
'234 - Stone Additional Report Information 13-61
'235 - Select Print Options 13-62
'236 - Retrieve Additional Report Information 13-63
'237 - Open Sort Work File 13-64
'238 - Retrieve Sorted Record Numbers 13-65
'239 - Generalized Report Printing 13-66
'240 - Generalized Inquiry 13-67
'241 - Create Record File 13-69
'242 - Initialize Record File 13-70
'243 - Set File Status Bytes 13-71
'244 - Retrieve File Status Bytes 13-72
'245 - Get Next Sequential Numeric Key 13-73
'246 - Rebuild Path Files 13-74
'247 - Add File to File Index 13-75
'248 - Delete File from File Index 13-76
'249 - Search File Index for File 13-77
'250 - Center Message 13-78
‘400- File#, Field Name 13-78

Section 15: Technical Information

FICE+ 3.01 System File Layouts 14-1
FICE+ 3.01 Record File Layout 14-6
FICE+ 3.01 File Control Blocks 14-8
FICE+ 3.01 Definition Files 14-10

Appendices

Appendix E : FICE+ system Error Messages (for Release 3.01)

Section 1
Run-Time Environment

This section consists of discussions on topics relating to the $F run-time environment in

Release 3.01. Only those capabilities in programs and subroutines that relate to each discussion are covered. For information on all of the capabilities of a particular program or subroutine, consult additional sections of this Guide.

Logon - $F BGN 1

Password Entry

All releases of FICE+ have accepted 12-character passwords although only eight characters have been shown. Password entry has been updated to show 12 characters.

During password entry, the cursor shows the characters that have been entered. Backspace can be used to delete mis-typed characters. Both ERASE and $F '8 will erase the entire entry. If the password entry routine recognizes that no characters have been entered (the cursor is under the first character of the password) when RETURN is keyed, the password remains unchanged. This can happen in three ways:

1) No characters were entered. For instance, if password entry was accidentally selected, simply press RETURN. No change takes place.

2) Password entry is canceled by backspacing to the first character using the Backspace key, then pressing RETURN.

3) Password entry is canceled by erasing the current entry with ERASE or $F '8 and pressing RETURN.

Scanning Company IDs

The Scan Key is active when entering the Company ID. If pressed, a list of all on-line Company ID's is displayed.

Close Files

File closing is now done through e separate utility in the FICE+ Utilities module on the Record File Utilities menu. Refer to the New Utilities section for more information.

Menu Driver - $F MENU

Initialization

$F MENU re-initializes the following items from the SYSTEM file:

- Location of the SYSTEM file (U)
- Background status
- Percent complete status
- Help usage
- Subroutine load check
- Clock and disk polling rates
- Global partition program names
- Sort and find next buffer sizes
- Field attribute and save-for-default buffer sizes
- Date roll-over (U8)
- Terminal date (Q1$)
- Company S (Q3$,Q4$)
- Terminal and background Queue as

The following services are also performed:

- The device table is reinitialized.
- Terminal status is set to "menu".
- Exclusive write privileges for the current company (Q4$) are released.
- All record files are closed.
- Any attached SYSSRT files are released.
- All devices are unhogged.

$F MENU no longer adjusts the numeric variables used to dimension buffers in file maintenance. Buffer sizes remain constant unless specifically changed by editing the system parameters.

Access to Special Programs

Inter-terminal messages may be sent (using $F '0) and received only while at a menu. However, notification that a message is waiting can be received while a file maintenance task is executing. The background queue control program is now accessible only from a menu.
Options 92 and 93 have been added to print and clear a terminal print queue respectively. This prevents having to enter the Control Module in order to send a queue to the printer.

Terminal Traits

The terminal record in the SYSTEM file has been redesigned. The new format sets aside space for "terminal traits ", which are system parameters that can be set on a terminal basis. The new format is more standardized, recovers wasted space, and initializes reserved areas to known values for specific future uses. The user's Terminal Traits can be edited by selecting special option '94' from any menu.

In addition to such items as the report queue S selections, terminal traits also include information on the user's terminal, such as CRT row and column dimensions and its graphics capabilities. New system variables allow programmers to check the attributes of a particular terminal before attempting to use special displays. FICE+ checks these variables before attempting to draw boxes or generate graphic displays. Also included is information on the user's terminal printer.

Menu Tracking

FICE+ now tracks a terminal's menu status in two new system variables. This eliminates a disk update each time the user switched to a new menu.

Direct Menu Options

There are now two ways to call direct menus instead of one. A direct menu call allows the user to proceed "directly" to a menu from any other menu without having to go through the menu hierarchy.

Prior to 3.01, direct menus could only be called with the $F DMENU program. Briefly, $F DMENU examines Q8$ (the function code), extracts the module and menu number from it, then loads $F MENU. The information in Q8$ must be in the form "MM#" where "MM" is the module name, and "#" is the number of the menu in the module to be displayed. If $F DMENU is called with Q8$ set to "$$2", menu #2 in the Control Module is displayed. A value of "SMO" would cause the main system menu to come up.

Because it is a program, $F DMENU can be called via special exits on function keys, and by other programs.

The new method combines the standard method of switching menus with a value in the function code to add two new capacities. The standard method is to use an up-arrow instead of a program name (or module ID on the System Menu) in Program Menu Maintenance to indicate that a new menu is to be displayed. Now, based on the function code, one of three things can happen if an up-arrow is assigned to a menu item number:

1) If Q8$ = blank, the menu number corresponding to the item # is loaded. This is the current way that menu number selection works.

2) If Q8$ contains a single digit from "0" through "9", the menu number corresponding to this digit is loaded in the current module.

3) If Q8$ is of the format "MM#" (the same format used by $F DMENU), menu number "#" is loaded in module "MM".

Types 2 and 3 are much faster than $F DMENU when going directly from one menu to another. Additionally, Type 2 means that the item numbers on menu zero no longer have to agree with the menu numbers that are called. This is useful on module menus that call both menus and tasks.

The new method is entirely generic. The up-arrow designation can be specified on any menu, including task menus.

Set Time - $F STM 1

This utility can be used to set or check the time-of day clock. It was formerly in the FICE+ Utilities module as "$U STM 1" and has been moved to "$F". It is available es special option '96' from any menu.

The utility supports FICE. Because of incompatibilities between the two languages, operating system password protection is not supported. The password "SYSTEM" is used for all time setting operations.

Only the terminal connected to port 1 (#TERM = 1) at each CPU may set the clock. Other terminals may only inspect it.

The time-of-day is also available on menus.

Program Loader - $F LOAD

$F LOAD supports all program loading requirements on all supported operating systems (plus a few that are no longer supported). This reduces the number of program loaders from b to this single program.

Clock and Disk Polling Rates

FICE+ no longer uses arbitrary $BREAK values to give away CPU time. All delays are now based on the clock and disk polling rates which are adjustable in the system parameters.

The following programs now use one or more of these polling rates:

$F ‘194 Printer subs
Clock poll when waiting for background printer availability.

$F FSC 1	File status conflict
	Clock poll used between screen updates.
$F MSG 1	Inter-terminal messages
	Clock poll used between screen updates.
$F RSB 1	Control background printing
	Clock and/or disk polls between screen updates.
$F RSP 2	Print report queue
	Clock and/or disk polls when waiting for a printer.
$F RSW 1	Background disk polling
	Clock and/or disk polls when waiting for a disk or queue.
$F SSD $	Systems status display
	Clock poll used between screen updates.

For further information on polling rates, refer to the System Parameters section.

**

Section 2
RMS - Miscellaneous

This section consists of discussions on topics relating to Release 3.01's record management system. Only those capabilities in programs and subroutines that relate to each discussion are covered. Refer to other sections of this Guide for more information on a particular program or subroutine.

File Access Class Enforcement

File access classes can be optionally enforced. If the enforcement flag in the system parameters is set to 'Y', attempts to read from a closed file, or attempts to write to a file with only read access rights generate a FICE+ system error.

File and record access conflict processing is based on how a file is used by the programmer. This intention is defined when a file is opened using '201 or '211, as specified by the value of the access class parameter, such as ‘R' for read only.

For instance, if a file is opened in read only mode, FICE+ assumes that records will not be added or deleted, and, in the interests of execution time, bypasses record conflict processing. Alternatively, when a file is opened in 'W' mode (exclusive write), the system assumes that no other terminal will be writing to the file, and again much of the conflict processing is bypassed.

In releases prior to 3.0, once a file was open (even if opened with a 'C' to close it), any read or write operation could be performed on the file regardless of the access class. This led to mysterious data losses.

Access class enforcement is automatically enabled when the SYSTEM file is created or initialized. You may temporarily disable enforcement by changing the Held in the System Parameters to "N". This may be useful when upgrading older installations to 3.01 in order to allow the end user to continue operations while programs are checked for access class consistency. As soon as possible, re-enable access class enforcement to ensure that files and records are properly handled in a multi-user environment. Access classes continue to be ignored by the subroutines '203, '204, and '207 because of their special nature.

Inter-Terminal Record Protection

As in previous releases, FICE+ 3.01 supports record protection. The protect code used with file access subroutines, such as '200, '202, '213, '214, '222, '224, and '225, allows a special "in use" byte to be set or cleared. The contents of this byte can be checked by the programmer (by examining QO) when a record is accessed to determine if it is in use. One, and only one, record can be protected for each open file for each terminal.

Prior to Release 3.0, undocumented record protection situations could occur, and if they did, record protection and unprotection were unpredictable. In 3.01, record protection is defined for the following circumstances:

Records can be protected when read if:

- A protect code of 1 is specified.
- The file is opened in shared mode (with an access class of 'S').
- The record is not already protected by another terminal.

Records can be unprotected when read if:

- Never. It is assumed that a protect code of 0 means that protection is not required, not that a record should be unprotected. Records can be explicitly unprotected using '207.

Records can be protected when written if:

- A protect code of 1 is specified.
- The file is opened in any write mode ('S', 'W', or 'X').

Records can be unprotected when written if

- A protect code of 0 is specified.
- The file is opened in any write mode ('S', 'W', or 'X').

Note: The status of the "in use" flag on disk is not checked when a record is rewritten. It is assumed that records that a~ rewritten u~ere previously protected in another subroutine call.

The currently protected record is automatically unprotected if:

- An attempt is made to protect another record in the same file.
- The file is closed.
- The file number is reused for another file, or the same file is reopened under the same file number with a different access class.

Note: The currently protected record is also unprotected if it is rewritten with a protect code of 0.

New Hashing Algorithm
If a hash loop counter of zero is assigned to a data file, a new key hashing routine is used to store and locate keys in its hashed path file.
Compared to the standard routine with a hash loop of three, the new routine is approximately three times faster and generates much better distribution patterns with small keys. Keys larger than 12 bytes should continue to use a hash loop counter of three.

Extended Platter and Sector Addressing

FICE+ has been modified to take advantage of two recent FICE language enhancements: Fasstcom's extended device table and Niakwa's 3-byte disk addressing standard. This increases FICE's disk address space to match the capacities commonly in use on super-minis and the read/write optical storage units expected in the near future from several manufacturers.

Extended Device Table

The System Parameters have been modified to handle tables of up to 127 disk devices to support Fasstcom FICE's extended device table.

To invoke a larger device table, you must modify a statement in $F START. A SELECT

15/000 statement has been added at line 8110, and a scratch-and-save at $F '30 is included to make this change easier.

As a rule, you should not make custom modifications to FICE+.

FICE Release 3.1 only supports the larger table if the highest slot to be used is explicitly declared in a SELECT statement using a numeric literal. A variable cannot be used, i.e. SELECT #X /D11. Once declared, the table remains in memory until a CLEAR or LOAD RUN.

Modify the SELECT statement to reflect the highest device table slot you intend to declare in the System Parameters. The highest slot supported by 3.01 is 127. Each slot beyond # 15 adds a few bytes of overhead to your partition.

For example, to support 40 platters, the statement should read:

```
SELECT #40/000
```

This will set aside 41 slots. Slots #0 through #39 can be platters, slot #40 will be used by FICE as scratch space to open printers, etc. FICE no longer hard-codes #15 for this purpose. It uses #U7 instead.

Note: If the LIST COM/DIM statement is executed while using the extended device table, a bug in Release 3.1 of FICE causes the CPU to freeze up. The only remedy is to shut off the CPU. Fasstcom has fixed the problem in Release 3.3.

Extended Platter Addressing

This allows platters to range up to 4 gigabytes in size. FICE+has been modified to support 24-bit addresses in all programs and subroutines that access the disk index.

Note: D.AT.A 3500 Release 2.3 extensively stores sector addresses and offsets in 16 bits both in memory and in disk tables. It also makes extensive use of FICE's ability to specify 1&bit disk addressed in an alpha variable. Consequently, D.AT.A 3500 Releases 2.3 and earlier must not be used on platters that exceed 16 megabytes or unpredictable damage may result.

Theoretical Limits

In previous releases of FICE+, the maximum disk address space has been 15 16-megabyte platters or 240 megabytes. Although much more could be physically connected to the CPU, this was close to the 256 megabytes that could be addressed simultaneously by the FICE languages.

If either Fasstcom or Niakwa adopts the recent enhancement of the other, the theoretical disk address space of the FICE languages becomes 1,099,511,827,776 bytes, 1 terabyte. But other considerations, including those in FICE+, drastically reduce this number.

First of all, although FICE now supports a device table with 256 slots, it is substantially constrained by its disk addressing scheme to 96 platters. (The system parameters currently accept values up to 99.) Consequently, FICE+'s capacity using FICE is 96 16-megabyte platters or 1,636 megabytes.

Working with Pseudo-Deleted Records

Both '215 and '216 have been enhanced to handle special circumstances involving pseudo-deleted records (2nd-level records containing HEX(FFFE) in the first two bytes). In addition, both subroutines have been reworked to eliminate an unnecessary call to '219 (made possible partly by the inclusion of '219 into the $F '217 program file) resulting in significantly faster execution.

'215 (Add Record)

An undocumented capability in Release 2.3 allows '215 to add a record using a primary key that is not actually in the record. Some programmers have used this capability to "add" pseudo-deleted records to a second-level file and have built their applications around it. A good example is in the Homebuilder Estimating module.

Release 3.0 did not support this capability because the keyfield parameter in '215 was ignored. The key fields in QI$Q were used instead for both the primary and any alternates to ensure a proper match. To get around this, applications had to add a dummy record, then rewrite it in the HEX(FFFE) format. Obviously, if the record had alternate keys as well, problems would eventually surface.

To support the ability to add a pseudo-deleted record to a file, '215 supports a negative record file number parameter. (This is consistent with the negative record file number parameter used by ‘225 to include pseudo-deleted records.)

If the file number is negative, '215 assumes that a pseudo-deleted record is being added. The key to be used is the key field parameter in the call to '215. This is the only instance where '215 uses this parameter. '215 will automatically construct a record in the HEX(FFFE) format and add it. All alternate key processing will be skipped.

'216 (Delete Record)
A similar situation arose when applications programmers attempted to use standard methods to pseudo-delete records in a post or update program. In the standard method, a record in the HEX(FFFE) format is rewritten over the top of an existing record using '213. While this works well with files which contain only primary keys. it will leave orphan alternates in the path files pointing to deleted records if alternate keys are in use.

With the new method, a negative record file number can be passed as the first parameter in a call to '216. If the file number is negative, '216 will automatically pseudo-delete the record corresponding to the supplied key and delete any alternates. Return codes remain the same.

Sequential Access Buffers
Physical Sequential ($F '221)
$F '221 has the ability to buffer sectors during physical sequential access of a file, which substantially improves its performance. However, if the file is being modified at the same time as it is being read sequentially, the buffer will not necessarily reflect the true contents of the file, leading to unexpected problems, such as getting . a record from the buffer that no longer exists on disk.

In Releases 2.3 and 3.0, the solution was to buffer sectors only when the file was opened in a read mode and only if the terminal was not attempting to protect the records as they were being read. While this solved the problem when a single terminal was both reading and modifying the file, it did not prevent conflicting situations arising at two different terminals.

Release 3.01 deals with this situation by checking to see if any terminal has the file opened in a write mode. The check is performed in the setup subroutine, '221. This is another instance where it is important that file access classes be accurate when a file is opened.

Buffering, which was an option in Release 2.3 and earlier only if the "$F '221 >" subroutine was loaded, is automatically enabled in Release 3.01. Buffering can be disabled by opening the file in a write mode or by reading the records protected in '222.

Key Sequential ($F '223)

The key sequential access subroutines, which access records by key (or just keys) buffer information from the key paths while processing a file.

Because of the way this subroutine works, accesses to ISAM files would take twice as long, and key sequential access on hashed paths would be virtually impossible if the buffer was disabled. Therefore, it is strongly recommended that any file read with '223/'224 be opened with a "P" (protect) Access Class.

Another advantage of buffering is that the applications program can read other records in the file without disturbing the sequential key process (this was not possible in 3.0 with ISAM keys).

If you do need to modify a file while reading it sequentially by key, this is possible if the key is ISAM. Use a method similar to the following example:

```
8100   REM Find TX1 records for customer A0$
       :REM Primary ISAM Key is CUSTOMER#, APPLY TO, SEQ in NO$
8110   NO$ = A0$ & ALL(00): REM Setup equivalent of '223
8120  GOSUB '202(1,1,0,N0$,">"):REM Get Next using Greater Than ('224)
      :IF 41$()>HEX(FF) THEN 8200: REM End of file?
      :GOSUB '88("U"): REM Unpack record
      :IF A0$ <> STR(NO$ " 5) THEN 8200:REM Done if Customer #
      changes :GOSUB 8130: REM Add, delete or modify records in file
      :GOTO 8120: REM ">" read at 8120 will "get next" using NO$
```

Other Performance Improvements

'201 and '211 detect when a file is being reopened using the same file number and access class. This occurs, for example, when the same file is opened in successively chained programs, or the same program is recalled or restarted. If detected, the subroutine merely initializes Q, Q0$, Ql and Q2 and returns. Prior to 3.0, the file was first closed and then reopened.

'215, '216, '217, '218, and '219 have been reworked to eliminate redundant calls to '219 when adding and deleting records with '215 and '216.

Because of the extensive overhead required to $OPEN disks on Novell Networks, $F '200 subroutines check if the disk is already hogged before attempting a $OPEN.

Undocumented Code Removed from '246

'246, the Rebuild Path Files subroutine (formerly known as "Rehash"), initializes all path files and rebuilds them. Prior to Release 3.0, $F '246 automatically deleted any records containing a duplicate key. This function was undocumented and potentially dangerous, and has been removed.

Support of Fasstcom "T" CPU-Based Disks

Checks for Fasstcom "T" CPU-based 2230 and 2260 disk platters have been removed from all programs.

**

Section 3
Keyboard Entry and Date Processing

This section consists of discussions on topics relating to keyboard entry subroutines and date handling in Release 3.01. Only those capabilities in programs and subroutines that relate to each discussion are covered. For information on all of the capabilities of a particular program or subroutine, consult additional sections of this Guide.

Special Intercepts

Subroutines '181 and '182 now offer a new feature called special intercept keys. The term "special" is used to differentiate these keys from the standard intercept keys ($Fs'1, '2, and '3) that call a user-written subroutine ('192).

Special intercepts allow the programmer to designate that certain keystrokes cause an immediate exit from the keyboard input subroutines, but, unlike a standard RETURN, no error checking is performed.

To use special intercepts, the values produced by these keystrokes must be identified to '181 and '182. A new, common variable is designated for this purpose, WO$(2)32. The first element specifies special function key intercepts, the second specifies standard keystrokes.

To use special intercepts, set the elements of WO$() to the keystrokes you wish to trap before calling '181 or '182. For example:

```
W0$(1)=HEX(00)
W0$ (2)="X»
GOSUB ‘181(2,Q,1,0,0,9,0,3,»R» ,1,0)
```

In this example, '181 exits if special function '0 is hit, or a capital letter 'X', even though this is a numeric entry routine.

WO$0 is treated like the OK Input specification; trailing spaces are ignored.

Note: If automatic case translation is enabled (the required/optional flag is a lower case letter), translation to uppercase happens after a keystroke is checked for an intercept, but before checks are made for OK Input.

WO$0 is initialized to blanks when '181 or '182 exit, either from a carriage return or a special intercept. $F LOAD also sets WO$0 to blanks to ensure that WO$p is in its initialized state following a rapid exit.

There are two ways to identify the keystroke that causes a special exit. W$ always contains the last keystroke made in a '181/'182 routine. In standard returns, W$ is set to HEX(OD) -- the carriage return.

WO is also returned. If WO=0, then no special intercepts occurred. If a function key caused a special intercept, WO is set to a value of I through 32, based on the position in WO$(1) of the character causing the return. If a standard character caused a special intercept, WO is set to a value between -1 and -32.

In the previous example, if WO=0 then no special intercept was made and Q or Q$ is the value entered. IF WO=1, then special function key '0 caused the exit. IF WO=-I, then an 'X' caused the exit. When intercepted, Q$ and Q contain any partial values already entered.

Special intercepts are useful for performing special routines which must acquire data from the operator. They are much less complicated to use than standard intercepts.

For example, if special intercepts are disabled and 'I, '2, or '3 is pressed while '182 is accepting keyboard input, '182 calls '192 to execute a standard intercept routine. You must have written this subroutine and loaded it in at a line number lower than 3000.

If the routine you really want to execute is actually part of your main line program, '192 must call your main line routine as a subroutine. If your routine requires additional information from the user, you have to know how to cache a number of undocumented variables so that they can be restored when you return to '192 (or '181/'182) so that the user can continue.

If a user happens to hit a standard intercept key while entering keyboard data in your special routine, the chances are excellent that the logic in your program will get hopelessly tangled up. This is what happens when the search key is accidentally pressed twice in Distributor III file maintenance.

With special intercepts, '181/'182 simply exit. You can test if a special routine is called for, execute the special routine, then branch back to the call to '181/'182 using the partial value previously entered as a default.

Example:

```
8300   GOSUB '250("Enter Ageing Date ('Cancel' to Reselect Report)",8)
       :W0$(1) = HEX(F0): REM Edit(Cancel) key
       :GOSUB '182(2,Q,2,X1$," "," ",0,3,"R","L"," ",3)
       :X1$ = Q$
       :REM Reselect?: ON WO GOTO 8100
```

In this example, the Edit(Cancel) key has been defined as a special intercept. If pressed, '182 returns instead of exiting out to $F MENU. X1$ receives whatever part of the date has been entered by the operator, which is then used as the default when execution returns to line 8300.

Blank Range Checks

Blanks have a special meaning to the range checks in ‘182, the alpha keyboard entry subroutine. Blank range checks are automatically translated to high and low values.

A low range check of ‘blank' is often used in applications. The keyboard subroutines have always interpreted a blank low range literally. While this works in the English-speaking world, it excludes 16 characters commonly used by other languages which collate as "less than" blank. From now on, blank low ranges will be interpreted as the lowest values (ALL(00)).

Except for the Type 3 logic fields, programmers have always had to specify high ranges in hex where all characters are allowed. Typically, this has been HEX(80) or HEX(FF) for most fields, or HEX(999999) for dates. If blank, high ranges are now set to the highest values (ALL(FF)).

Cursor Positioning

'180 was originally intended to manage the complicated CRT control codes needed for cursor positioning when FICE+ was used with Fasstcom FICE. In the late '70s, the PRINT AT verb became available and was adopted in place of the more complicated '180 algorithm.

At one time, both 16x64 and 24x80 CRTs were being sold. Some programmers wrote their applications to run on the small CRTs to stay compatible with both. Unfortunately, the smaller layouts looked unbalanced on the 24x80s because the information was offset to the upper-left portion of the display.

FICE+ was enhanced to minimize the problem by allowing 16x64 screens to be centered automatically on 24x80s based on a system parameter. But this caused problems with programmers who designed their PRINT ATs on 16x64s because their screens were no longer where they thought they were on 24x80s. To solve this problem, '180 was changed from a trivial cursor positioned to a means of compensating for unexpected display shifts made by FICE. If '180 was used to position the cursor when working on a 16x64, it would appear in the right place on a 24x80.

But not all of the problems could be fixed this way. Programmers who typically worked with 24x80s could be surprised if a screen they designed was small enough to fit on a 16x64. FICE assumed it was a 16x64 and shifted it. To compensate, programmers had to stick a character in the lower-right hand corner of a 24x80 display to get FICE to leave it alone.

To simplify the situation, automatic shifting was removed in Release 3.0. The values passed to '180 are the coordinates that are used. The trend among applications is that older modules tend to use '180, while newer ones tend to use PRINT AT which is faster, more readable, and requires less program text. All references to '180 have been removed from FICE+ subroutines and the FICE Utilities.

Simplifications

'181 and '182 no longer give special meaning to $F '0 and shift/TAB (shift/FN on DE keyboards). Consequently these keys do not automatically execute the interterminal message utility and the background queue control program when they are executed unless the user is at a menu. This allows the keys to be redefined for other tasks (such as the Scan Key), and prevents the accidental loss of data when the keys are pressed inadvertently.

The keyboard entry subroutines are now generic. All situation-specific code has been removed. They now behave in a consistent manner regardless of the type of task being executed.

Dates

Blank Dates

Blank dates are now treated as null dates that contain ALL(00). The following subroutines are affected:

'182 The default date can be blank. Refer to the discussion on "Blank Range Checks" for additional information.

'186 Both blank and null dates are considered as illegal dates by this subroutine.

'187 Q$ will be set to ALL(00) if a blank date is sent to '187.

'188 Q$ will be set to blanks if a blank date is sent to ' 188.
'
189 Both blank and null dates are considered as illegal dates by this subroutine.

Date Subroutine Locations

Both '187 (pack date) and '188 (unpack date) are now included with the $F '180 program file. Either subroutine can still be loaded independently by putting $F '187 or $F '188 on the load line.

Programmer Specified Rollover Year

The rollover year in the "add days to date" and "days between dates" subroutines is specified using the common variable U8. The roll-over year determines the first year to be included in the current century. All year values less than U8 are assumed to be in the next century.

In previous releases, the rollover year was hardcoded at 50, which is the standard default for U8. The calendar used by these routines ran from the year 1950 through 2049.

You can change U8 for specific purposes. U8 is reinitialized to its default value when execution returns to a menu. The default value is specified in the SYSTEM file and can be changed by editing the system parameters.

In previous releases, U8 was a reserved variable used to regulate $CLOSE during file access subroutines. Applications programs that specifically change the value of U8 may result in hogged disks if used with previous releases.

Miscellaneous Changes

- Range checks in a Type 3 field are automatically set to high and low values. (In previous releases, the "low" was never adjusted.)

- Unprintable characters in defaults are translated to "?" instead of blanks, making it more apparent to the operator that something is wrong.

- The Required/Optional and Range Check fields are automatically defaulted using Type 6 fields.

- '188 ignores illegally packed dates.

**

Section 4
File Maintenance

This section consists of general discussions on file maintenance in Release 3.01. Only those capabilities in programs and subroutines that relate to the individual discussion are included. For information on all of the capabilities of a particular program or subroutine, consult additional sections of this Guide.

File Maintenance Changes in the 3.x Releases

The file maintenance subroutines were completely rewritten for Release 3.0 and several substantial changes were made for Release 3.01. In both releases, many enhancements were added, several simplifications took place, and allowances were made in several areas to clear the way for additional enhancements in future releases.

The following paragraphs discuss the more important changes. Several of these discussions also cover topics relating to upward compatibility.

Program File Reduction

The number of supported file maintenance programs has been reduced from 9 to 2, consisting of $F '220 and $F '226.

All "greater than" or "less than" versions of these subroutines have been eliminated. $F LOAD automatically traps any references to ' >' and ' <' versions of these subroutines on program load lines and substitutes the appropriate subroutine names.

The file $F '210 has also been eliminated. The subroutine '210 is now part of $F '220. $F LOAD automatically traps any references to $F '210 on program load lines and substitutes $F '220. Programs do not need to be changed. Calls to '210 result in one-level file maintenance, calls to '220 result in multi-level file maintenance.

Variable Standardization

In virtually all FICE+ Utilities programs and many FICE+ subroutines including file maintenance, a standardized and documented set of variables is used to reference all values in the definition file field and parameters records. A new subroutine, $F ' 10, is the pack/unpack for the definition file. It is automatically loaded by $F LOAD in all file maintenance programs. $F ' 10 does not have to be specified on the load line of file maintenance programs.

File Maintenance

Alternate Key Support

Release 3.01 uses an efficient and versatile algorithm to automatically manage changes to alternate keys in file maintenance.

Before a record is resaved in Change Mode, the original record is retrieved from disk and compared with the "changed" record. If no changes have been made, the routine exits. If changes have not been made to the primary or any alternate keys, a '213 is used to update the disk. If either the primary key or any of the alternate keys has changed, the original record is deleted (along with any alternate keys) and the "changed" record is added to the file.

This logic recognizes all changes made to the record, including those made by the application program directly to TI$0, prior to the record being saved.

If, when the record is re-added, a duplicate key or path file full error results, the "changed" record is removed and the original record is re-added to the file. An appropriate error message alerts the operator that a problem has occurred.

Whenever the user enters or changes an alternate key field, FICE+ checks for illegal duplicates immediately where appropriate. The procedure above is a fail-safe mechanism to handle situations where TI$() has been changed outside of FICE+'s control, or where, in multi-user environments, another operator adds a conflicting alternate key between the time that the field is entered by the operator and checked by FICE+, and when the record is actually saved.

Multiple-field Items

It is possible to assign multiple fields to the same item number. This simplifies file maintenance where a number of fields are handled as a logical group. The PZ Payroll module takes advantage of this capability extensively.

The fields must be assigned to contiguous sequence numbers. If the item number is selected by the operator, all fields with that item number are prompted for.

Inquiry Mode

A new mode has been added to file maintenance. Inquiry mode provides a way for the operator to examine records without making any changes. This eliminates accidental changes and the time consuming overhead necessary to process records in Change Mode by both FICE+ and the custom applications program.

Custom applications which preset Z3 to allow an automatic roll-in to a file maintenance mode may have to be changed. In 2.3, if Z3 was set to 4 through 6, second-level add, delete, or change mode was entered. '4' now stands for first-level inquiry, while '5' through '8' now represent second-level functions.

Reserved Keys

When multiple operators are adding records to the same file at the same time, it is important that the file maintenance subroutines ensure that each operator works on s record that has a unique primary key. The simplest method is to attempt to "reserve" the chosen key immediately after it is entered by the operator. If the key is already in use elsewhere, the operator can alter the key until a unique key is found.

Release 2.3 reserves keys by adding a standard default record to the file using the key supplied by the operator. When the rest of the fields are filled in, the record is "resaved" with a '213. If the operator decides not to complete entering the record, the default record is automatically deleted. If a situation arises where a default record accidentally remains in the file, it can be handled normally because it contains legitimate fields, although most are blanks and zeros. The record can be deleted in file maintenance, and sorted by the sort routines.

Since Release 3.01 adds alternate keys automatically whenever a record is added, default records cannot be added to a file because the default values in fields used for non-duplicate alternate key paths will cause conflicts if more than one person adds records to the file at the same time.

Release 3.01 solves this problem by adding only the key to the primary path file, and not the record. This is a much quicker operation. If the operator decides not to use the key, it is automatically deleted.

The terminal number reserving the key is "coded" in the record number used for the key, such that the record number equals 1 million minus the terminal number (Z0=1E6-U0). (This further means that the limitation on the number of records FICE+ can support in file maintenance is reduced from 999,999 to 999,744 for a 255-terminal system.)

What happens if a key is accidentally left in the file? The record number associated with the key will be so large that it will point to a non-existent record. The record access subroutines, such as '214, return Z0=0 and 41$()=ALL(FF) in this instance. Programs, including Change and Inquiry Modes in file maintenance, will interpret. this as "Record Not On File".

In Add Mode, processing depends on the terminal number that originally added the record, and the terminal number that is attempting to use the same key. If they are identical, the record can be added. If they are different, the message "Record in Use at Terminal xxx" will be displayed.

Calls to '215 will return Z0=0 (duplicate key), but '219 will return the coded terminal number reserving the key.

Reserved keys left in a file can be eliminated either by reusing the keys in file maintenance, or by rebuilding the path files with '246 or by running the Rebuild Path Files option on the "Record File Utilities" menu in the FICE+ Utilities Module.

Reusing Deleted Second-Level Record Keys

In 2.x releases, if a second-level record was deleted, its key could be reused later for a new record. This ability was intimately tied to the fact that a record always existed for a key (even if just a default record), which is not the case with reserved keys in 3.01 file maintenance. In 3.01, newly added second-level records must always use a sequence number 1 greater than the highest already used.

It is TOM's intention to restore this ability in a future release, or to make it inconsequential by allowing increments other than 1. We hope the current method does not cause any inconvenience.

Use of Special Intercepts

File maintenance takes advantage of special intercepts when special function keys '1, '2, and '3 are pressed (such as a search key) and calls '192 directly. This allows the application programmer to make calls to the input subroutines ('181 and '182) during the standard intercept without fear of disturbing the 'W' variables used in field editing.

Field Attribute Buffering

Field attribute buffering is now standard; however, not all fields are buffered. Previous buffering techniques made allowances for variable length attributes, such as OK Input and Range Checks, to save memory. Experiments show that the overhead required to extract the attributes with this technique results in execution slower than a RAM disk retrieval. Consequently, file maintenance buffers the attributes of only those fields without range checks or OK input characters, where buffering can be done in a simple and straightforward manner.

Screen Buffers

File maintenance screens are no longer buffered. Screens are saved and loaded "DA" and can be accessed with a single FICE statement, thus taking full advantage of any buffering and caching inherent in the disk drive or host operating system.

New Screen Variable

At any point in file maintenance, T9$() contains a copy of the file maintenance screen (CRT lines S through 23) as seen by FICE+. This includes any fields automatically referenced from other files, but does NOT include any printing to the CRT done by the applications program. T9$() is meant to be used by FICE+ or the applications programmer to repaint a file maintenance screen that has been disturbed by another function, such as "help" or the Scan Key.

Automatic References by Alternate Key

Automatic file references can be made by alternate as well as primary key. The definition file field definitions have been enhanced to accept alternate path file references in addition to primary path references.

When precessing definition files, a warning is generated if the alternate key path allows duplicates. In this case, there is no guarantee that a unique record in the other file will be retrieved and used by the file maintenance subroutines.

Recursive References to Other Files

Fields automatically retrieved from another file can be used as keys to retrieve records from other files automatically. See the discussion on File Maintenance Phases later in this section for more information.

Prompt Subroutines Incorporated into File Maintenance

$F '250-type capabilities, such as the centering of prompts and messages, are now available in '226. If a blank field name is used, the alpha and numeric values are interpreted as the Message and Length parameters that would normally be passed in a call to '250. '226 does not support the -3 parameter (clear help display). For more information, read the discussion on '250 in the FICE+ Module ($F) section.

File Opens

The names of record files are shown as they are opened. This eliminates seemingly long "program load times" by showing the user that something is happening.

File Scanning

The Scan Key is automatically implemented in all file maintenance and general inquiry programs as Special Function ‘0. The only requirement is that the file being scanned must have at least one ISAM key path.

Scanning in File Maintenance

To Scan, the cursor must be positioned on a key field, either to the "local" file (the file you are working on) or a "referenced" file ("other files to open" in the definition file parameters). The appropriate file will be selected for scanning. In the case of local primary keys which are also keys to another file, pressing the Scan Key toggles scanning back and forth between the local and referenced file.

The error message "Scanning is Not Possible Here" will appear under certain circumstances:

1) The cursor is not positioned at a key field.
2) The file to be scanned is not open.
3) The file to be scanned does not have an ISAM path.
4) A referenced file's definition file cannot be opened. The Scan Key attempts to open definition files as file #V0, the highest file number that can be opened (a system parameter). If a file is already open in slot #V0, and that file is not a definition file, it is assumed that the file maintenance program requires the use of that file.

After opening the definition file, the scanning logic checks the definition file parameters for ISAM key paths. If there is more than one, the operator is allowed to choose. The first available (lowest numbered) path is defaulted. If only one path is available, it is automatically selected.

Following path selection, the operator is prompted to choose a starting value. An appropriate default is supplied based on the current contents of the local record. If blank, scanning will start from the beginning of the file.

Full screen scanning displays multiple records per screen. Records can be displayed from the beginning of the file to the end of the file from any starting point. Files can be scanned both backwards and forwards.

The format of the display is automatic. Up to three fields from the record are extracted and

Shows. These consist of the path key, the primary key, and the ID Field. The display is automatically truncated at 80 columns.

During scanning, the operator may select a record, move the selection cursor, call for another page of records, reselect the file, path, and starting value, or exit with no changes.

In this table, the key cap descriptions correspond to those found on a Fasstcom DW keyboard. Refer to your FICE language and terminal reference manuals to determine the equivalents on other keyboards. The following Scan functions are defined:

Function	Key	Comments
Select Record	RETURN	The appropriate value is defaulted into the field where the cursor was when the Scan Key was pressed.
Cancel	CANCEL/EDIT	Nothing is changed.
Next Record	Space Bar Cursor Down $F '5	Next Screen is executed at the last record.
Prev Record	Backspace Cursor Up $F '6	Prev Screen is executed at first record.
Next Screen	NEXT SCRN Shift Cursor Down $F '21	Ignored at end of file.
Prev Screen	PREV SCRN Shift Cursor Up $F ' 22	Ignored at beginning of file.
Scan Key	$F '0	Used to restart the Scan. If both a local and a referenced file are involved, scanning toggles between the two.

$$ SER $, the current "file scan" subroutine, is still supported, but is no longer required. The Scan Key uses SF '0 to avoid $$ SER $ key conflicts.

File Maintenance Phases

Documentation on the behavior of the file maintenance subroutines has been limited in the past when describing what actions are performed during what are known as Entry Phase, Edit Phase and Display Phase. The phase determines how and when certain aspects of file maintenance, such as special case exits, are handled when a field is processed.

Changes, documented and undocumented, have been made to the phases in past releases of FICE+, including 3.0, and in Bug-Bytes. Our goal is to make Edit Phase perform consistently with Entry Phase. This reduces the amount of code that must be written in a custom file maintenance program and makes the FICE+ file maintenance subroutines more predictable.

The following is a general description of each phase:

Entry Phase FICE+ is in this phase whenever a field is entered for the first time. Examples are the initial entry of fields in Add Mode, or entering the key to a record in the other modes.

Edit Phase FICE+ drops into Edit Phase whenever the "Enter Item # to Change" prompt is issued in Add and Change Modes.

Display Phase Display Phase is used to display fields on the screen. No keyboard entry takes place. For instance, after a record has been retrieved in Change Mode, Display Phase is used to paint the field contents on the screen.

The phases are used in the following manner for the four file maintenance modes:

Add Mode Execution begins in Entry Phase when a blank input form is presented to the operator. After all screen sequences have been processed, file maintenance displays the "Enter Item # to Change" prompt and goes into Edit Phase. The two phases alternate with each new screen.

Delete Mode Execution starts in Entry Phase while the key to the record is entered. After a valid key is entered, Display Phase allows all fields on screen #1 to be displayed. After the operator chooses whether or not to delete the record, the process starts over.

Change Mode Execution starts in Entry Phase until a valid key is entered on screen #1. After the record is retrieved, Display Phase is used to fill in the remaining fields on the screen. Edit Phase takes over when the operator is prompted to "Enter Item # to Change". Display Phase and Edit Phase alternate on any additional screens.

Inquiry Mode The operator is prompted for a key in Entry Phase. Display Phase shows the record to the operator for all screens.

Since Release 3.01 allows a field to be retrieved from a data file and, at the same time, be used as the key to another data file, certain changes were made to the methodology used in the 2.x releases of FICE+. For example:

1) Transaction	----	
2) Customer #	-----	----Customer’s Name--------
3) Salesperson #	---	----Salesperson Name------
4) Sales Account	--------	------Account Name----------
5) Product #	------------	-------Product Name----------

In this hypothetical case, when the operator enters the Customer Number, the key is validated and the customer's record is retrieved from the Customer master file. In addition the Customer's Name is displayed and the Salesperson Number is also retrieved and saved.

This is as far as Release 2.3 and all prior releases could go. Although it is a key to a file, the Salesperson Number could never be automatically validated against the Salesperson master file, nor could it be used to automatically display the Salesperson Name. This is because a field cannot be both retrieved from one file and still be used as the key to another in 2.3.
This restriction is eliminated in 3.01. Not only can the Salesperson Number be validated as a legitimate key and the Salesperson Name displayed, the Sales Account can be retrieved from the Salesperson master file, validated against the Chart of Accounts master file, and the Account Name displayed -- all automatically.

Automatic Field Retrieval

If, in Edit Phase, the operator selects an item number assigned to a field that is not a key to another file, FICE+returns to the "Enter Item # to Change" prompt immediately after the field is edited.

If the field is the key to another file, screen sequences continue to be processed until one of the following occurs:

1) There are no more sequences to be precessed.
2) A field is encountered that is not retrieved from another file.

In other words, automatic retrieval continues to cascade through files and fields until the end of the "logical group" is encountered. In the example on the previous page, if the Customer Number was modified, processing would stop after the G/L Account Name was updated. The Product Number, which is not retrieved from the Chart of Accounts master file, is considered part of a separate group.

Pre-Input Special Case Exits

Although efforts will continue to be made to make Entry and Edit Phases execute consistently, one exception will always persist: pre-input special case exits ('184) will always continue to operate only in Entry Phase.

Post-Input Special Case Exits ('185)

In older, I.x releases of FICE+, '185 was called in Entry Phase only if the "Enter?" flag (a parameter in the definition file field record) was set to "Y". The reason for this was that '186 was only called from the keyboard subroutines ('181 and '182) and the keyboard subroutines were only used by file maintenance if "Enter?" was "Y". Consequently, programmers had to choose between streamlining the entry process by strategically skipping field entries, and being able to trap the value of that field in a '185 for later use or validation.

In later I.x and 2.x releases, '185 was still called only in Entry Phase, but non-entered fields were included by using a special call to '185 directly from the file maintenance subroutines. But this only solved part of the problem. Edit Phase, encountered in Add and Change Modes, still operated the same as the old Entry Phase --'185 didn't get called unless the user specifically edited the field.

For instance, in the previous screen example, if the Salesman Number field had a special case exit number and was edited by the operator in response to the "Enter Item ..." prompt, '185 was called when the operator pressed RETURN. But if the Customer Number field was changed by the operator, which resulted in the Salesman Number being automatically changed (because it was automatically extracted from the Customer master file), '185 was not called.

To summarize, in Edit Phase, '185 was called for the field being edited, but not for fields that were, afterwards, automatically updated by FICE+as a consequence. This resulted in programmers having to implement redundant processing (to ensure that something got done in both Entry Phase and Edit Phase in Add Mode), or use the shotgun approach (update everything in '264 in case something changed).

Release 3.01 solves this problem. If a field has a special case exit number, '185 will be called whenever the field is processed during normal file maintenance sequencing, whether or not the operator actually edits the field. (The '226 subroutine does not generate a cell to '185, even though it is used to modify a field. It is assumed that the programmer is well aware of the change that is taking place.)

This may be an incompatibility with previous releases, depending on the circumstances, but it probably is not. The most likely situation is that older programs will still be executing code to bypass a shortcoming that no longer exists. In general, no harm is done. These programs would already have had to deal with the circumstances where an automatically updated field was also edited directly by its item number. The new approach can be thought of as FICE+ calling the item number automatically for the operator.

The 'Enter?" Flag

IN 2.3 and previous releases, the "Enter?" flag has designated whether or not the cursor stopped at a particular field location only during Entry Phase of Add Mode. In Edit Phase, the flag has been ignored. When the operator chose an item number to edit, the cursor would stop at the associated field. If it was a key to another file, any fields retrieved were processed automatically without operator intervention. In fact, intervention was impossible.

This philosophy is changed somewhat in 3.01. In Edit Phase, automatic field processing will pay attention to the "Enter?" flag. If the flag is set to "N" on a retrieved field, the field will be updated from the other file, '185 will be called, and the field will be skipped. If Enter? is "1"', the field will be updated, the operator will be given a chance to edit the field, and '185 will be processed.

In other words, '185 processing and externally retrieved field handling now behave the same way in Edit Phase as they do in Entry Phase. This also includes '227 processing that designates that a certain number of fields are automatically skipped! The pre-input special case exit, '184, will continue to be called only in Entry Phase of Add Mode, and then only for fields that have not been previously processed.

Field Chaining

It should be noted that, although file maintenance supports retrieved fields being used as keys to other files, it currently only keeps track of one externally retrieved record at a time. This means that, in the previous example, additional fields could not be retrieved from the Customer master file after the Salesperson Number was validated. Once the salesperson's record has been accessed, fields can only be extracted from that record. A more universal approach will be addressed in a future release.

File Maintenance - Access to Records in '252

The 3.x aeries of releases redefined the contents of T1$() in those instances where, when the operator is adding records directly to the second level, the first-level record is retrieved for inspection by the application program in '252. Programs which have referenced the record in T1$() should use Q1$() instead.

This change corrects an inconsistency in the '250 series of subroutines and adds a needed capability as well.

When adding line items directly to a second-level file (instead of flowing through from the first level), the '252 subroutine is called after the first-level key has been entered. The purpose of the call is to present the applications programmer with the first-level record in order to update it, check it, or extract information to be included in the second-level record.

The problem with previous releases is that '252 is defined as presenting the first-level record in both T1$() and Q1$() even though T1$() is otherwise defined as containing the second-level record in all other instances of working with a second-level record. In addition to being redundant and adding to the general file maintenance confusion, it makes it very difficult to work with both first- and second-level records at the same time, and it complicates the process of setting up a second-level default record in T1$()

Consequently, the variables T1$() and Q1$() are defined in the '250 series of file maintenance subroutines to be the following:

T1$()	Will always contain the record that the operator is working on. T1$() is undefined if this record is undefined. For example, the second-level record is undefined in the Entry Phase of Change Mode when only the first-level part of the key has been entered.
Q1$()	Will always contain the record to the other file. If another file is not involved or undefined, Q1$() will contain T1$(). In essence, Q1$() will always contain the record that is most appropriate to the '25x subroutine call.

We emphasize that this redefinition affects only the above instance in '252. Everything else is the same.

In summary, the following chart details the contents of TI$0 and Q1$() during various modes of file maintenance. "Current" refers to the record that the operator is trying to add, delete or change.

Mode	T1$()	Q1$()	
'250	All	Current	Other file
'251	Add	Current	Current
'252	Delete(2)	Current	Current
	Change(3)	Current	Current
	Add(5): 1st-level key	Current	1st level
	Delete(6): 1st-level key	Undefined	1st level
	Delete(6): 2nd-level key	Current	Current
	Change(7): 1st-level key	Undefined	1st level
	Change(7): 2nd-level key	Current	Current
'253	All	Current	Current
‘254	All	Current	Current
'255	All	Current	Current

Again, the record that is the focus of the call to a '25x subroutine will be in Q1$(). T1$Q need only be used when it is necessary to refer to or update the current record (the record the operator is working on).

As a side note, during special case exits ('184 and '185), T1$() will contain the current record; Q1$() is undefined.

**

Sorting, Merging, and Report Printing

This section consist of discussions on topics relating to sorting and report printing in Release 3.01. Only those capabilities in programs and subroutines that relate to each discussion are covered. For information on all of the capabilities of a particular program or subroutine, consult additional sections of this Guide.

Sorting

Several changes have been made to the sort subroutines and programs.
$F '230

$F '230, the sort subroutine has been rewritten. It is slightly faster than in previous releases.

Release 3.01 has also introduced a change in the way sort parameters are stored in the first sector, U2(3), of the SYSSRT file. Applications which accessed this information directly will have to be changed.

The original parameters, stored DA, were:

Description	Stored
Sort key length + 6	Numeric
Size of sort buffer (bytes)	Alpha (3-byte binary number)
Size of sort key	Alpha (3-byte binary number)
Sort buffers used	Numeric
Total sort keys + 1	Numeric
Last SYSSRT sector used	Numeric
Sort keys per sort buffer	Numeric

The new parameters, stored DA, are:

Description	Stored
Key length (passed to '230)	Numeric
Sort buffers used	Numeric
Sort keys per sort buffer	Numeric
Sort keys in last buffer	Numeric
Total sort keys	Numeric
Last SYSSRT sector used	Numeric

Sorting, Merging, and Report Printing

Using 'Redefined' Fields

'Redefined' fields, which treat any string of bytes in a record as a separate field, can be used as field names in calls to '232 (establish sort sequence). For further information on field redefinitions, refer to the section on Definition Files.

Use of ISAM Keys

$F SRT 1, the key file sort utility, uses range checks to reduce the number of keys that must be examined when sorting a file by its primary ISAM key. This can result in significant reductions in sort time.

$F SRT 1 is only called when a single primary key field is sorted on.

Merging

Two new programs have been added, "$F MRG 1" and "$F MRG 2". $F MERGE is now a setup program which decides which of the two new programs is most suitable for the merging based on the results of the sort.

$F MRG 1 incorporates several refinements and is much faster than any merging operation previously available in FICE+. Typically, a merge operation that used to take over a minute is now done in b seconds. A one-hour merge is now done in less than four minutes. $F MRG 1 is flexible enough so that it will be called by $F MERGE in almost every situation.

$F MRG 2 is used when large numbers of records with large sort keys have been sorted. It is called automatically by $F MERGE where appropriate, typically with sorts including more than 100,000 records. $F MRG 2 is approximately 5 percent faster than previous merge programs.

Printer Selection

The "Terminal 'Traits" are checked prior to allowing the user to select a terminal printer. This prevents a partition from locking up if a terminal printer does not exist. A terminal printer cannot be selected if the Terminal Traits do not reflect that a terminal printer exists.

The Subheading field on the printer selection screen has been lengthened to 48 bytes (from 47), some fields have been realigned, and the minimum printer number is now 1 (it was 0).

Report Printing

Subtotals and grand totals are buffered in memory, reducing the number of disk accesses required when $F '239 processes control breaks.

Alternate Key Support

$F '240 contains two parameters, the "Alternate Key Letter" and "Prompt", that are meant to allow '240 to work with external alternate key files (files with a letter in the fifth character of the file name, such as ARCUA). '240's capabilities have been expanded to include support of internal alternate keys and special prompts for primary keys as well.

The "Alternate Key Letter" has been renamed to "Key File". Special processing is performed based on the contents of this field. If Key File is blank, '240 prompts for the primary key using its field name. For multipart keys, each field is prompted for in turn. This is the method used in previous FICE+ releases.

If Key File contains the letter "A" through "Z", the letter is assumed to identify an external alternate key file. Prompt is used to prompt the user for key input, and the record is retrieved by means of the external alternate key file.

If Key File contains a "0", '240 will ask the user to enter the primary key to the file. If the primary key is a multi-part key, the user will be asked to enter the key value as a single field. Prompt will be used to prompt the user. If Prompt is blank, the Primary Key Name field in the definition file parameters will be used.

If Key File contains the digits "1" through "8", the user will be prompted to enter the value for an internal alternate key corresponding to that number. The Prompt parameter will be used to request a key from the operator. If Prompt is blank, the alternate key's field name will be used.

Scan Key Support

'240 support the Scan Key. However, accesses cannot be made to "referenced" files. For more information, refer to the discussion on the Scan Key in the File Maintenance section of this Guide.

$F ‘250 Emulation

$F '250-like capabilities are available when $F '240 is in memory by referencing subroutine '226 instead of '250. Pass the same parameters to '226 as you would to $F '250.

**

The following miscellaneous changes were made in the 3.x releases of FICE+.

- The Edit Company Name program, $F ECN 1, has been renamed to $$ ECN 1.
- Many system error messages have been rewritten. All can now be called using a mnemonic rather than a number in Q8$.
- Password editing is now available from the security menu in the FICE+Utilities module.
- $F WDS 1, the Woodstock program, has been eliminated.
- The ' >' and ' <' versions of $F '221 have been eliminated. The "enhanced" version is now standard. $F LOAD automatically translates references to $F '221 > and $F '221 < to $F '221.
- The file index maintenance subroutines ('247, '248, and '249) have been modified to work with file indexes containing 128 file names.

**

Section 8
FICE+Utility Module Menus

The FICE+Utilities Module menus were extensively modified by Releases 3.0 and 3.01. The module menu now contains references to 7 task menus.

Menu #1 - System Utilities

The purpose of this menu is to provide utilities for installing software and managing the system. In recent years, this menu was used as a catchall for any new utility. New menus have been created for this purpose in 3.0.

This menu contains the following items:

Edit System Parameters

Many of the System Parameters have changed and the utility now accepts up to 127 disk addresses in the device table. For further information, read the section on System Parameters in this Guide.

Edit System Configuration

In addition to file index maintenance for SYSSRT and SYSRPT files, and virtual terminal table maintenance, this utility contains the means of turning virtual terminals on and off,

Edit Printer Attributes

A separate utility is now available for listing the printer attributes.

Change System File Sizes

This utility is used to change the sizes of the SYSTEM, SYSPAT, SYSSRT, and SYSRPT files.

List System Parameters

Starting in Release 3.0, the system parameters can be listed without going through all the screens with the Edit Systems Parameters utility.

List System Configuration

The virtual terminal table and the SYSSRT and SYSRPT file indexes are listed.

List Printer Attributes
The printer attributes list has been broken out of the Edit Printer Attributes utility.

Create System Files
This program creates SYSCO, SYSID SYSSRT, and SYSRPT files.

Verify Load Lines
This utility is used during applications installation to ensure that all needed subroutines have been transferred to the system disk.

Enter Company Name
This program allows the installer to enter or edit the contents of SYSCO files.

Convert File Indexes
This utility converts Release 2.3 and prior file indexes to 3.x format. For further information, read the section on upgrading 2.3 sites to Release 3.01.

Convert Definition Files
This utility converts Release 2.3 and prior definition files to 3.x format. For further information, read the section on upgrading 2.3 sites to Release 3.01.

Convert Record Files
This utility converts Release 2.3 and prior record files to 3.x format. For further information, read the section on upgrading 2.3 sites to Release 3.01.

Menu #2 - Module
The purpose of this menu remains the same in 3.01 as it was in prior releases. It is defined as providing for the creation and maintenance of module-oriented files. This menu contains the following items:

Create File Indexes
File indexes are created in the 128-file format.

Modify File Indexes
Using $F '1, all references to a particular disk can be changed to another.

Print File Indexes
The company ID can be specified.

Create Program Menus
The format of program menu files did not change with Release 3.0.

Modify Program Menus
A new form of "direct menus" is now supported. Read the Run-Time Environment section for more information.

Print Program Menus
The Company ID can be specified.

Copy Program Menus
This program can now copy, rename, and change the size of any menu file to any disk platter in any combination.

Menu #3 - Definition Files
The purpose of this menu remains the same as it was prior to Release 3.0. For further information, read the Definition Files section. This menu contains the following items:

Create File
Definition Files are created in 3.x format.

Edit Parameters
The main parameters screen was completely redesigned for Release 3.0.

Edit Fields
The field entry screen has a new format. Fields can be moved and copied.

Edit Screens
Screens are now loaded and saved in DA mode.

Process Files
This utility has been completely redesigned.

Print Documentation
Documentation for the entire system can be printed in one pass.

Move to End of Disk
This program supports Niakwa's 24-bit sector addressing.

<u>Change File Size</u>
Fields and screens can be added and subtracted from a definition file.

<u>Lock Files</u>
this utility prepares definition files for future record file restructures.

Menu #4 · Record Files
This menu continues to contain items oriented toward record files. However, an extensive list of new utilities are planned which will not fit conveniently on this menu. Consequently, this menu will split in the future. Four new utilities have been added since Release 2.3, and the entire File Create module has been incorporated. For further information on the new utilities, read the Record Files section. The menu items are:

<u>Create Record Files</u>
Files are created in 3.x format.

<u>Create External Alternate Kev File</u>
External key files are still supported but they cannot be used for ISAM.

<u>Initialize File</u>
This program uses the $F '242 initialize file subroutine which initializes the records, all path files, and all header records.

<u>Reset Default Values</u>
This program was modifed to work with the new definition file structure.

<u>Clear Record Protection</u>
This program works with the 3.x record file structure.

<u>Compress Record File</u>
This is a new utility. It makes a record file more efficient for sequential access and sorting. For further information, read the New Utilities section.

<u>Rebuild Path Files</u>
This program was formerly called Rehash Record File. Because both hashed and ISAM key paths are now supported, the old name was misleading. This program USES the '246 subroutine to rebuild the paths.

if a duplicate key is encountered where none are allowed, it is not added.

Move to End of Disk
This is the same program as found on the Definition Files menu.

Restructure File
This program was formerly known ss Reorganize Records. The program has been extensively modified to work with the new record file structure. For further information, consult the Other FICE+Utilities Changes section.

Change File Size
Formerly Expand/Contract, this program now adjusts the capacities of the alternate key path files to conform to the number of new records in the file. The path file parameters used are those in effect when the record file was created or restructured.

Display File Parameters
This is a new utility. It displays all information in the file and path headers plus the trailer. Both record files and definition files can be examined. For further information, read the New Utilities section.

Record File Editor
This is a powerful record file diagnostic and repair tool. It is introduced in Release 3.01. Directions for its use are in the New Utilities section.

Close Files
This is a new utility. It replaces the "CLOSE FILES" password that was used on the logon screen in Release 2.3 and prior.

The following items were incorporated from the File Create module which is now obsolete. These programs have been modified to work with the 3.x file structure and disk platters that exceed 16 megabytes. For further information, consult the Other FICE+Utilities Changes section.

Enter File Sizes
Edit Files Sizes
Print File Sizes
Calculate Disk Usage
Create Files

Menu #5 - Programming Utilities

This is a new menu. Its purpose is to contain those utilities that are oriented toward FICE programming and program files. It contains the following items:

<u>Temporary Program</u>
This item loads $U TEMP. .

<u>Variable X-Reference</u>
This utility now supports the 3.x definition file format.

<u>FICE+System Update</u>
This program now "blinks" the diskette drive only once.

Menu #6 - Disk Utilities

This is a new menu. Utilities that treat the disk or disk files generically are located here. It contains the following items:

<u>File Copy</u>
This utility has been modified to work with platters larger than 16 megabytes.

<u>Print Disk Index</u>
This utility allows a disk index to be sorted three different ways and printed in three different formats. It works with 24-bit disk addressing.

<u>Edit SYSID File</u>
This utility edits the portion of the disk label checked by the disk backup utilities.

<u>Edit Index Label</u>
The index label is an optional label that appears when the disk index is listed with a LIST DCT statement.

<u>Compress Platter</u>
This utility performs a "MOVE" in place. Disks larger than 16 megabytes are not supported.

<u>Disk Editor</u>
This is s powerful new diagnostic and repair tool. Directions for its use are in the New Utilities section.

Menu #7 - Security

This menu contains two items, password editing and password access, which were available from Woodstock in the Control Module in Release 2.3 and prior.

**

The structure of the definition files was extensively modifed and all FICE+Utilities programs which deal directly with definition files were rewritten with Releases 3.0 and 3.01. Only those programs that exhibit major changes to the programmer are detailed in this section.

New Concepts

There are several new concepts introduced in this release, including "redefined fields" and definition file "locks'.

Locking and Unlocking Definition Files

A new flag, called process status, is kept in the definition file. It determines whether the definition file is ready (locked) or in the process of being modified (unlocked).

A locked definition file is considered "ready». This implies that the definition file and all of the record files that it describes are ready to be used by the end user. The programmer is prevented from making any significant changes to the definition file while it is in this state.

If unlocked, a warning is issued to the user if attempts are made to use the definition file in file maintenance.

A definition file can be unlocked by pressing $F '0 when using any of the definition file editing utilities. It can be locked only by running the Lock Files utility. In order to be eligible for locking, a definition file must have been processed without errors. Otherwise the file remains unlocked.

Locking also updates all information used to restructure record files. While unlocked, any number of record files can be restructured (the Reorganization Generation Number has been withdrawn). However, once a lock is re-established, needed restructure data is cleared from the definition file; preparing it for the next set of changes.

Note: Do not restructure a record file twice with the same set of definition file changes; damaged data will result.

Field Redefinition

Record file fields can now be specified as redefining one or more other fields or portions of fields in the same file. Redefined fields are not actually saved in the record, but provide a means of accessing saved portions or groups of other saved fields as if they constituted a separate field.

For example, if a field contains a customer's phone number, the last four digits of the phone number could be redefined as a new field. This field can then be used in sort options, by the report generator, and as an alternate key, just like any other field, without having to specify the substring of the phone number wherever it must be used.

A redefined field must be a contiguous set of bytes in the record. To define it, a starting field name and byte, and an ending field name and byte are specified by the programmer. The start and ending fields can be the same.

An example of its usage is to represent multiple fields as a single field in order for it to be used as an alternate key. Consider the following portion of a record:

Field	Start Byte	Length
...		
PRODUCT #	100	12
CUSTOMER #	112	05
PROD/CUST	100	17

PROD/CUST is a redefined field. It starts at the first byte of PRODUCT # and continues through the fifth byte of CUSTOMER # making a 17-byte field. PROD/CUST's start byte is the same as that of PRODUCT #, but it is not saved. It merely represents information in other fields that is saved. PROD/CUST can be used as an alternate key, can be sorted on, and can be used by the report generator. If it is an ISAM key, it can be used to quickly find all records containing a particular product, sorted in customer number sequence.

In addition, you can redefine a multi-part primary key so that it can also be an alternate key. Assuming that the primary key and its alternate counterpart used different access methods, the records in this file can be accessed by primary key using both hashed and ISAM methods.

Note. Single-field primary keys do not have to be redefined to be used as their own alternates.

Definition File Enhancements

Structure

The structure of the definition file field record has been redone:

Fields of the same size and format have been grouped together.

The field name, which is the key to a definition file field record, has been moved to the beginning of the record for consistency.

Fields have been expanded and added, and documented spares have been created for future use.

Standardized variables are now used for all definition file references in FICE+.

Definition file field and parameters records can now be packed and unpacked with a standard $PACK/$UNPACK subroutine, just like any other record.

Field Name Moved to Beginning of Record

Similar to a record file, definition files have a key file plus header and trailer records. Release 3.0 moved the key field to the beginning of the record to conform to record file standards. In this location, record file operations, such as rehashing. can be performed on the definition file. This reduces the complexity and overall amount of exception code that has to be maintained because of the non-standard format.

Definition File $PACK/$UNPACK

The definition file now has its own $PACK/$UNPACK subroutines and a standardized set of variables for the file parameter and each field attribute associated with a record file field. Definition files can be treated like record files by the applications programmer in most respects.

The pack/unpack subroutines are $F ' 10 and $F ' 11. Both subroutines are identical except that '10 uses 'S' variables and '11 uses the equivalent 'T' variables. Refer to Technical Information section for the definition file record layouts and the meanings of the new variables.

By having two pack/unpacks, two field attribute records can be worked on at the same time for comparisons or conversion. By having a standardized set of variables. FICE+ is easier to understand and maintain. The variables also provide a means to access definition files in a standardized, straightforward manner.

Use of Definition File Disk Names

In the FICE Utilities, definition files are now specified by their actual disk name rather than the data file that they define. Beginning with this release, efforts will be made to allow FICE subroutines and utility programs to work with both record files and definition files transparently wherever possible. Thus, it is important to name files specifically when working with these subroutines and utility programs. Specific file naming is now standardized throughout FICE.

Where it is unambiguous, FICE+ will automatically translate record file names into definition file names and vice versa.

Definition File Parameters

The definition file parameters screen has been redone and several new fields have been added or changed. The most important new fields are those dealing with ISAM and AMAK

To automatically maintain alternate keys, the record file must be in 3.x format. This format is impressed on the record file when it is created, restructured or expanded/contracted. The definition file parameters determine the format of the record file.

The parameters required for AMAK are the path number, the field name, and the access type. For ISAM access types, the bucket size and packing density must also be specified.
A path number is determined for a particular alternate key by associating a single field name with the path. The field name can be from a normally saved field or it can be a redefined field, which can consist of many other saved fields.

Once a path number and alternate key field are connected, it is difficult to change this number further downstream. Although alternate keys can be added, deleted and modified at any time, applications programs become dependent on the path number selected here to represent a particular key.

Path numbers can be skipped. If an alternate key is deleted, alternate keys occurring after it do not have to be moved up to fill the gap.

The access type for a primary key field can be either 'H' for hashed, or 'I' for ISAM. Alternate keys can also use the 'D' and 'S' parameters which designates ISAM with duplicates. The'S' parameter indicates that blank alternate keys are not saved.

The bucket size is only required for ISAM keys. Single-sector buckets are appropriate for most paths. In general, buckets should not exceed 2 sectors.

The Est ISAM Density field shows the estimated used capacity expected of an ISAM path file. Density is defined as the average amount of space used in a key bucket when the record file is full. For moat files, the default is 65 percent. Files which experience constant record additions and deletions without being initialized (TX1, DSI, etc.) should use 50 percent. For more information, read the Philosophy and Theory section.

The parameters, which are stored in the next to last sector of the file, can be unpacked into standard variables by calling $F ' 10 or $F ' 11 with a "UP" parameter. The parameters record can be packed using a "PP" parameter. Either subroutine expects the parameters to reside in the Q$() sector buffer as the parameters must always be accessed and saved in BA mode.

The Open as Needed flag on the second screen refers to the capability of opening other files one at a time as each file is needed. It is pointless to use this capability in custom file maintenance programs that expect files to be opened before their keys are accessed in file maintenance, or with one-screen files or other files in which all referenced files open on the first screen no matter what the operator does. It is emphasized that there are very specific instances where setting this flag to 'Y' is a benefit. This flag replaces the system-wide flag that existed in previous releases.

Definition file parameters which do not require a file restructure when changed are accessible to the programmer even when the definition file is locked.

Definition File Fields

The field definition screen has been redone and reflects the new information kept with each field.

The fact that a field is saved is now kept in the file. (Prior to Release 3.0, this was implicitly determined based on whether or not the field had a starting byte). The value of Save can be 'Y' or 'N'. In addition, the value can also be 'R', which stands for "redefines". For more information on redefinition, refer to the discussion at the beginning of this section.

Retrieve and Get portions of the editing screen represent the two phases of extracting information from another file; retrieving a record, and extracting information from the retrieved record. FICE+ file maintenance now recognizes these as two separate operations and requires a set of information for each purpose.

Despite the fact that several ambiguities are resolved, separation of retrieval and extraction has another benefit; automatically extracted fields can be used as keys to automatically retrieve other records, from which fields can be extracted which are keys to other records, and so on.

References to other files can now be made by alternate key as well as primary key. The Must Be On File? field now carries 'Y', 'N', '&', and blank parameters. The '*' parameter is no longer used.

If a field is redefined, enter the redefinition start and end fields along with the start and end bytes of each included field. The start and end fields can be the same.

An additional feature has been built into the field editing program. Programmers can now move, copy, or replicate fields.

When moved, a field is deleted from one location and reinserted in another. Fields can be moved from one location to any other location. Copies of fields can also be placed anywhere. The replicate function allows multiple copies to be made of any field, but the new fields must be placed just after the original. The field names of any new fields created using copy or replicate must be altered so that all field names are unique.

Definition file field information which does not require a file restructure when changed is accessible even when the definition file is locked.

Definition File Screens

The main change to definition file screen maintenance is that maintenance behaves differently based on whether or not the file is locked. When locked, those items that would cause the definition file to be processed in order to take effect are disabled. Consequently, cosmetic changes to the screens are allowed, but not much else. When unlocked, all options can be executed.

Definition File Processing

The Process Definition Files utility has been rewritten. In addition to support of ISAM, AMAK, field redefinition, retrieve/get fields, new screen and field formats, and new status fields, the utility also generates a new set of error messages based on much more extensive error detection and correction.

'ALL' parameters may be specified in place of a specific Company a, module or file name when specifying the file to process. Company ID is only appropriate in Company Type 4 installations. Otherwise, the definition file company is assumed to be company ‘blank'.

In addition, three types of messages are generated:

Errors:	Errors prevent the definition file from being used. They must be corrected before the definition file can be locked and before record files can be created using this definition file's structure.
Changes:	This indicates that the definition file has been modified to the extent that any existing record files associated with it must be restructured before they can be used. However, new data files can be created with the new changes.
Warnings:	Warnings are used to signify that the Process utility has encountered something that may not work correctly, but that is not critical to the proper use of the definition file as a whole. In some instances, the warning advises that the process program has made a correction to the definition file.

When a definition file is processed, it is automatically unlocked. If a definition file that was previously locked processes cleanly (no "changes" or "errors"), it is automatically relocked. If any "change" or "error" messages appear, or if the file was unlocked to begin with, it remains unlocked.

Display File Parameters

This program has been updated to show the contents of the path file headers for all files that are in 3.x format. Refer to the Technical Information and Philosophy and Theory sections for more information on these parameters.

The display of these parameters is particularly useful when monitoring the performance of ISAM path files.

Close Files

This program replaces the capability of the same name that was formerly available on the system startup screen in $F BGN 1. Individual record files can be closed. In addition, Company S, module, file name, and terminal options can be specific or 'ALL'.

'ALL' must be used with the module and file name fields to cancel any outstanding exclusive write status to a particular terminal. To cancel system exclusive privileges at a terminal, Company ID, module, and file name must be 'ALL'.

Compress Record File

This is a new program. It performs a reorganization on the data file by eliminating "holes" caused by deleted records. This makes the file more efficient for sequential accesses, sorting, and adding records.

This utility can be used to clean up a record file suspected of having damaged deleted record links by eliminating all deleted records. Damaged records can be inspected and fixed using the Record File Editor.

Record File Editor

The Record File Editor permits the programmer or responsible user to examine and modify any record in any FICE+ record file, regardless of its condition or the availability of file maintenance screens.

The program begins by prompting for a module, file and Company S. Notice that a modified version of the Scan Key can be used to help locate these items. When the record file has been successfully opened, you are asked to select an access method to locate records. The following methods are supported:

Key	In this mode you are prompted for a key. If alternate keys exist, you may select a key path to use.
ISAM	Using this method, records can be retrieved in key sequential order if an ISAM path is available. This is the most convenient way to inspect a series of records with a duplicate ISAM key.
Physical	This is the most convenient method of inspecting records in a small file. Deleted records are automatically bypassed.
Absolute	Records are retrieved by absolute record number. This includes deleted records and

records which have never been used. Terminal default records can be also be retrieved.

After a record is retrieved, one or more screens of fields are displayed which you may edit. At any time you may resave a record by pressing the TAB or FN key. Prior to resaving, you will be prompted to edit the record protection codes, which allow you to change or eliminate protections on individual records.

The Record File Editor does not automatically update key path files. If you make changes to a key field, you should rebuild the key paths.

Note: This utility can be dangerous in uneducated, undisciplined or unprincipled hands.

Disk Editor

The disk editor is a FICE-based disk utility which provides a convenient way for the programmer to inspect and modify a disk platter. It performs most of the standard functions common to the more popular stand-alone disk editors available.

Following the prompt for s platter address, sector zero of the selected platter is displayed in horizontal format in Display Mode. $F '1 toggles the editor between Display Mode and Edit Mode.

Display Mode

Display mode is used to inspect sectors. It is identified by a label at the bottom of the screen, and by a prompt for an eight-character field at the top. Briefly, the following functions are available in Display Mode:

Next	If the field is left blank, the editor loads the next sector.
Absolute	If a number is entered, it is assumed to be an absolute sector address. If found, the sector is displayed.

Relative If the first character is a "+" or a "-", the remainder of the field is assumed to be a numeric offset from the current sector.

File Name If none of the above is true, the entered value is assumed to be the name of a file. If the file can be located, Display mode jumps to the first sector of the file.

Redisplay The RUN/Execute key or SF '16 redisplays the current sector. This is useful if your purpose is to watch the sector as it is being updated by another program.

Cancel Cancel or EDIT prompts for a new disk address.

Edit Mode $F '1 puts the editor in Edit Mode.

Edit Mode

In Edit Mode, the contents of the sector can be changed. However, the change to the disk does not occur until you explicitly save the sector. The following functions are available:

ASCII/HEX $F '2 toggles between ASCII and HEX keyboards. The current keyboard is displayed at the bottom of the screen, and the cursor blinks on the appropriate display. The ASCII keyboard accepts only standard key strokes (function key values and certain control keys are ignored). The hex keyboard accepts only legitimate hex digits.

Move Cursor All cursor movement. keys work as you would expect.

Delete Deletes current byte -- a null is placed in byte 256.

Insert Inserts a null at the cursor position; byte 256 is lost.

Erase Prompts for an erase character to be used to fill the sector from the current byte position to the end of the sector. Enter either a character or two hex digits.

Recall ('15) Restores the sector to its state prior to any changes.

Save F '3 allows you to save an edited sector. You must respond "Y" to the "Are You Sure?" prompt. Following the save, the editor returns to Display Mode.

**

Restructure Record File

This program was formerly known as Reorganize Records. It has been extensively modified to work with the new record file structure.

Record files in the old format are converted to 3.x format when they are restructured. In addition, alternate key files are added or deleted based on the contents of the definition file.

In previous releases, Restructure has processed all fields in a record whether they needed it or not. This ensured that all fields contained valid data and that their justification (right or left) was correct.

The "Validate All Fields?" option permits FICE+ to process only those fields that specifically require it. This saves significant amounts of CPU time. In those instances where the purpose of the restructure is to add, delete, or change the packing density of an alternate key, this option allows the utility to bypass all record processing. In this instance, only the keys are processed with the resulting savings in CPU and disk overhead.

The "Change Packing Density?" option allows the user to tailor the packing density of each ISAM path file. If set to "Y", the current packing densities for all path files are displayed for editing prior to the restructure.

The Reorganization Number was withdrawn in Release 3.0. Instead, a new flag in the definition file keeps track of whether or not a definition file has already been used to reorganize a file since the last time it was unlocked. This is used to generate a warning in the restructure program.

In addition, this program displays the dates that the definition file was processed, and the date that a data file was last restructured. Based on this information, it is assumed that the programmer can make a decision on whether or not to restructure the data file.

Note: Data may be lost if a data file is restructured twice without locking the definition file in between each restructure.

Create Files by Module

This set of programs was formerly the File Create (FC) module. FC is now incorporated into FICE+, and the FC programs are described below.

Enter File Sizes ($U EFX 1)

Use this function to enter the disk location of record files and the maximum number of records allocated to each record file. The disk location is indicated by disk device number in the system parameters and displayed at the bottom of the screen. A disk device number of ‘0' indicates that the file is not to be created.

Data you enter is stored in the parameters sector of the corresponding definition file.

Note: Only one file size storage area exists in each definition file. A multicompany system may share the same definition files. File sizes used to create files for one company must be edited or re-entered for each additional company.

Edit File Sizes ($U EFX 1)

Use this function to edit the disk location of record files and the maximum number of records to allocate. Disk location is indicated by the disk device number from the system parameters, and displayed at the bottom of the screen. When you run this program, all files for the selected module are displayed. For each file, you can edit the disk device number and the maximum allowable number of records. Use a disk device number of zero (0) to indicate that no file is to be created.

Print File Sizes ($U PFX 1)

Use this function to print a list of the data entered with the Enter File Sizes function, including file sizes in sectors. This data does not necessarily represent the actual locations and sizes of record files existing on the data disk.

Calculate Disk Usage ($U CDX 1)

This function computes the disk space required for the proposed number of records specified for each file and shows if sufficient space is available on the disk platter. When you run this program, disk usage is computed for either all files in all modules or ell files in a selected module. The required number of sectors for each disk platter is compared to the number of available sectors on that platter. Program and system file space is not included in the calculation.

Create Files ($U CFX 1)

Use this function to create and initialize data files. Before using this function, prepare disks by formatting and scratching them. Also, specify the appropriate Company a code for the files.

You can create Teat Size files or Actual Size files. Enter the maximum number of records allocated for test files in the Test Records field using the Enter / Edit File Parameters function. The Create Files program requests a disk unit for use in creating all test files. To create Actual Size files, use the Enter File Sizes or Edit File Sizes function to specify file size and disk location. The program does not attempt to recreate existing files.

Automatic Flom-Through of File Names

The previously entered Module S, File Name, and Company ID entered in a FICE+ Utilities program automatically flow through to the next.

**

Edit System Parameters has been modified to reflect the changes to the SYSTEM file made with Releases 3.0 and 3.01.

New and Modified Fields

The following fields are new or have been changed.

Applications System ID

The Applications S is used to identify the applications system to the Control Module utilities. Examples are 'HB4' for Homebuilder, and 'D-4' for System IV.

The $$ SYS $ program file, consisting of a series of DATA statements, has been eliminated. One of the purposes of this file was to identify the name of the Applications System for use with Initial Setup utilities. The new Initial Setup utilities now refer to this field to determine the ID.

Century Rollover Year

This is the two-digit year used by '186 (add days to date) and '189 (days between dates) to determine the first year of the current century.

In release 2.3 and prior, this number was hardcoded at '50'. It is now adjustable and can be modified within applications programs to handle those items, such as birthdates in insurance applications, where old 20th century dates are important.

CRT Rows and Columns

CRT rows and columns can now be changed on a system-wide basis. Maximum value is 255. These values are used as defaults for U1 and U2 if values are not available in the terminal traits.

Max Device Table Slot Number

This is the highest numbered slot in the device table. It defaults to 15 (16 slots numbered from 0 to 15). If the Fasstcom extended device table is in use, this number should be changed to reflect number of the highest slot available.

Highest File Number to Open

This field was previously called Maximum Simultaneously Open Files, which was misleading. The number in this field represents the highest file number that can be used to open a FICE+ data file.

This number is used to dimension the number of elements in Z$0. Each element is used to track the status of an open record file. Prior to 3.0, each element of Z$() was 38 bytes. The element length has been extended to 54 bytes.

Maximum Path Files Open Simultaneously

This is the number of virtual key paths that can be maintained simultaneously by the system. It is used to dimension the number of elements in ZO$(), a new common variable. Each element is used to track the status of an open path file and is 40 bytes in length.

A path is a key file, either primary or alternate, hashed or ISAM. Any record file which is in 3.x format (the file has been created, restructured, expanded, or contracted using 3.01) uses an element for each path in the record file's key file.

Elements are automatically assigned to a data file when it is opened, and released when it is closed. Your only concern is the number of primary and alternate key paths that must be accessible at any one time. Consequently, it is possible for this number to be less than the number in the Highest File Number field above.

Maximum Bucket Size

ISAM keys are kept in variable length "buckets". Buckets are 1 to 8 sectors in size.

The bucket size required for an ISAM path file is defined by the programmer for the paths in a record file when the file is created. The number entered here defines the largest bucket specified for any ISAM path file. In that sense, it is similar to specifying the dimensions of Q1$ (), the record buffer.

The default is 2 sectors resulting in 512-byte buckets. This bucket size is large enough to allow efficient ISAM random record accesses on all but the largest files or files with very large ISAM key lengths.

The number entered here is used to dimension the two bucket buffers, Z1$() and Z2$(). The bucket buffers are dimensioned so that each has an additional 64 bytes (useful for bucket splitting). For further information, please refer to the discussion on the B' 'Tree in the 'Philosophy and Theory" section.

Enforce Access Classes?
The default value is 'Y' for yes. For further information, read the information on enforcement of access classes in the Record Management System - General section.

Maximum Fields to Buffer

The number of fields, rather than the buffer size, is now specified for dimensioning the file attribute cache in file maintenance. Each cached field requires 56 bytes.

For best performance, make the number of fields as large as possible. Assuming memory is available, this is the maximum of the total of combined fields used in any two-level file maintenance program, but not exceeding 255.

Disk Polling Rate

Background programs must check their queue files periodically to see if any work must be done. The parameter supplied here determines the number of seconds that must elapse between checks.

Poor system performance results if a disk is checked too often. It is recommended that you use about 30 seconds. The maximum is 255 seconds (4 minutes, 15 seconds). With more background partitions on any single system, specify greater lengths of time. This rate is also used to give away time in other situations, such as waiting for error conditions to clear.

Clock Polling Rate

This number is used with the Disk Polling Rate to determine how often the clock is checked. If the clock is checked too often, poor CPU performance results.

On those systems that do not have a clock (the TIME function returns'999999'), the disk polling rate above is assumed to be zero, and this parameter, the clock polling rate, is used to determine how often the disk is checked.

The clock polling rate can assume values between 0 and 255. The amount of actual clock time that elapses for any given value is hardware, operating system, and system load dependent.

The value in this field is translated into the same number of $BREAK commands raised to the third power ($BREAK rate^3). On a Fasstcom 2200 with a very small system load, the maximum value of 255 could translate into as little as 2 or 3 minutes. On a Xenix operating system, this same value could be several days!

A default value of 30 is supplied and is intended for Fasstcom FICE+ CPUs. Use it until you have a chance to determine a more appropriate value, if necessary.

Other System Parameter Changes

Merge Buffer

Merge buffers are now allowcated dynamically based on the number of sort buffers generated by '230 that must be merged.

Number of Terminals

The number of terminals is no longer displayed es it cannot be modified from this program. It must be modified by changing the size of the SYSTEM file in order to adjust the number of terminal information records.

Terminal Multiplexing

Whether or not terminal multiplexing is in effect has always been determined from a flag kept in the SYSCPU file. For consistency, multiplexing status is modified in the Edit System Configuration program.

Open Files During Input

The Open Files During Input system-wide flag has been withdrawn. This capability is now specified on a file-by-file basis in the definition file.

Minimum Partition Size

All enhanced versions of subroutines are standard in 3.01.

Global Partition Subroutine

$F '217 is the only FICE+ subroutine that can run in a global partition.

Input Form Buffer

File maintenance input forms are no longer buffered. They are now saved and loaded "DA," which is much more efficient than the previous "BA" method.

Number of $C Companies

Most of the capabilities of the Multicompany Utilities Module ($C) are now included in 3.01 and the System IV applications. The Number of Companies to Use $C Utilities value is no longer supported. The variable V8, which contained this number, will be redefined in a future release of FICE+.

Device Table

The system device table bas been expanded to hold 128 device addresses for disk platters, including an entry for the system disk. The alternate device table is accessible by pressing $F '15. The address for the system disk is used as a reference. The disk address assigned to device table slot 0 (SELECT DISK) is never altered by FICE+.

Section 13
The FICE+ Module ($F)

This section individually describes the programs and special subroutines in the FICE+ Module. (FICE+ subroutines are covered in Section 19). To eliminate confusion, every program and special subroutine is mentioned even if its function remains unchanged from Release 2.3.

FICE+ Programs

FICE+ programs create and manage a consistent environment for FICE+-based applications. The following programs are provided with Release 3.01:

File Name	Description
$F @PART	Initialize Global Partition
$F @PRT2	Initialize Global Partition [Non-standard]
$F BGN 1	Logon
$F COM 1	Declare Dynamic Common Variables
$F CTNUE	FICE+ Entry [Warm Start]
$F DMENU	Direct Menu
$F END 0	FICE+ Exit
$F END 1	Null Program
$F ERR 1	FICE+ System Error Messages
$F FSC 1	File Access Conflict Display
$F SD 1	Identify Disk Platter
$F LOAD	Program Loader
$F M/F 1	General One-Level File Maintenance
$F M/F 2	General Two-Level File Maintenance

$F MENU	Program Menu Driver
$F MERGE	Merge [Setup]
$F MRG 1	Merge [Fast]
$F MRG 2	Merge [Standard]
$F MSG 1	Interterminal Messages
$F QUEUE	FICE+ Entry [Background]
$F RSB 1	Control Background Queues
$F RSC 1	Clear Report Queue
$F RSP 1	Print Report Queue [Setup]
$F RSP 2	Print Report Queue [Control)
$F RSW 1	Poll Queues
$F SORT	Sort [Common Variables)
$F SRT 0	Sort [Setup]
$F SRT 1	Sort [Keys)
$F SRT 2	Sort (Records]
$F SSD 1	Display Detail System Status
$F START	FICE+ Entry [Cold Start]
$F STM 1	Set Time
$F TRT 1	Edit Terminal Ti~sits

FICE+ Program Descriptions

$F @PART - Initialize Global Partition

A small amount of program text can be shared by all users if a 5K universal global partition is used with Fasstcom's FICE language. The global partition must automatically load $F @PART at boot time to be set up properly.

$F @PART reads the common program list in the SYSTEM file to determine which program text is to be shared. Based on the contents of the list, one of three actions can be taken:

1) IF the list is blank, it is automatically updated to contain a single reference to the program $F ‘217. $F ‘217 is the only FICE+ subroutine currently supported in the universal global partition.
2) If the list contains a single reference to $F ‘217, it is loaded as an overlay to $F @PART.
3) In all other cases, the list is assumed to contain up to five programs that are to be loaded into the universal partition along with the $F @PRT2 program.

$F @PRT2 - Initialize Global Partition (Non-Standard]

This program is used with non-standard universal global subroutine lists to provide compatibility with previous releases of FICE+. The program defines the name of the global partition and executes a $BREAK!.

$F BGN 1 - Logon

$F BGN 1 allows the user to enter or edit the password, date, user ID, and company identification code for an application. This program is automatically loaded from a cold start ($F START) or from $F CTNUE or $F QUEUE after a logoff.

This program is typically assigned to rapid exit key '16. It is always available as special menu option 98.

Password Entry

Several changes were made to password entry in 3.01.

- The screen reflects that the password can be 12 characters in length.
- Backspace is supported when editing a password.
- If the password field is selected for editing and no changes are made, the original password remains intact.

Company ID Entry

The Scan Key ($F '0) is supported when editing or entering a Company ID. Up to 128 valid Company IDs can be displayed.

$F COM 1 - Declare Dynamic Common Variables [3.01]

$F COM 1 is loaded immediately after the registration verification and setup programs and is executed only once in any FICE+ session. Its purpose is to declare dynamic COMmon variables.

$F CTNUE - FICE+ Entry [Warm Start]

$F CTNUE instructs FICE+ to attempt a warm start. If successful, the password, date, user ID, and company S currently in use at the terminal are automatically set by the startup programs and $F MENU is loaded. If the terminal is not logged on, the warm start fails and a cold start is run (see $F START).

FICE+ allows non-FICE+ applications, such as word processing, to be executed from the System Menu. If these external applications load $F CTNUE at their termination, the user does not have go through the logon process. Instead, the System Menu reappears following the FICE+ registration check.

$F CTNUE sets $PSTAT to "$F CTNUE" then loads $F START. The program name in $PSTAT is the flag which instructs the Registration 5 startup programs to attempt the warm start.
$F START is loaded so that the SELECT statement defining the size of the extended device table is executed.

$F DMENU - Direct Menu

$F DMENU is one of two methods by which direct menus can be called. (See the discussion on $F MENU for a description of the other method.) Direct menus provide a convenient means of going directly to a specific program menu without going through intervening menus in the menu hierarchy.

$F DMENU examines Q8$ (the function code) to determine which menu to use, then loads $F MENU. The value in the function code must be of the form "MM#" where "MM" is a module name, and "#" is the number of the menu to be loaded. For example, a function code of "AR3" will cause the system to move immediately to the third menu in module "AR". A code of "SMO" will load the System Menu.

Because it is a program, $F DMENU can be used by applications to override the menu to which FICE+ returns following completion of an application. It can also be assigned to a rapid exit key, which allows a particular menu to be accessed from anywhere in the system.

$F END 0 - FICE+ Exit

$F END 0 provides a common exit point from FICE+. FICE+ loads this program whenever a user logs off or rolls out to a non-FICE+application, such as word processing.

$F END 0 calls the registration program that releases the user's terminal. Until this is done, the terminal is still considered to be in use by the registration system.

When a user is executing FICE+, they are actually logged on in two places, FICE+ and Registration 5. $F MENU logs a user off of FICE+ when the user requests it. $F END 0 is used to log the user off of Registration 5.

$F END 0 assumes that the name of a program to load following the registration logoff procedure is passed to $F END 0 in Q$. If Q$ is blank (or contains "$F END 0" because $F END 0 was loaded by $F LOAD), the $F END 1 program is assumed as a destination.

Consequently, $F END 0 should only be loaded directly if a program other than $F END 1 is the eventual destination.

For instance, to roll out to a program called "GRAPHICS", the application program should look similar to the following:

```
8000 % 'ROLL OUT' - GO TO GRAPHICS PACKAGE
8100 Q$ = "PLOTMENU" :REM NAME OF MASTER MENU PROGRAM
8110 LOAD T "$F END 0" :REM RELEASE TERMINAL
```

When this program is executed, the user will be logged off of Registration 5, but will still be logged on to FICE+. This means that warm start using "$F CTNUE" can be used when FICE+is re-run.

$F END 1 - Null Program

$F END 1 is a null program and consists of a single image statement at line number zero. It is loaded whenever FICE+ needs the use of a program that does nothing.

When a user logs off, FICE+ attempts to locate a program named "END". If found, it is specified as the program to be automatically run following registration logoff. If "END" cannot be found, then $F END 1 is used as a destination.

$F END 1 can also used as a way of executing the equivalent of a RUN statement (not a RUN command) that is compatible with all supported FICE languages. By loading $F END 1 as an overlay to an existing program in memory, all actions that normally occur with a simple RUN command (non-common variables are reinitialized and all stacks are flushed) are executed, but the program itself remains unaffected.

This is useful when a program must be "restart" under program control. This technique is used to execute the "restart" command during report printing. Therefore, it is especially important that $F END 1 never be modified. Use the "END" program described above for custom FICE+ terminations.

$F ERR 1 - FICE+ System Error Messages

$F ERR 1 displays FICE+ system error messages.

$F ERR 1 can be used to display an application error message in Q1$(). Because FICE+ uses QS$ to specify system error messages, the application should set Q8$ to blank before calling $F ERR 1 to prevent an accidental match with a FICE+ error code. $F ERR 1 must be called using $F LOAD. For example:

```
8500 ON X GOTO 8510
        :Q1$() = "Communications Port Not Working"
        :Q8$ =» «
        :Q$ = "$F ERR 1"
        :GOTO 19
```

As a rule, $F ERR 1 should not be called by applications programs in posting situations. The $$ ERR 1 program is usually more appropriate. For further information, consult the Control Module Technical Reference Manual.

$F FSC 1 - File Access Conflict Display

$F FSC 1 generates the "File Access Conflict" display. It is called by FICE+ when an attempt is made to open a data file using an access class that is incompatible with how other terminals are using the file.

The program displays the name of the file in conflict and how it is being used by the other terminals on the system. The screen is periodically regenerated, based on the clock poll rate in the system parameters. To exit, press RETURN.

This program will be replaced in a future release.

$F IDD 1 - Identify Disk Platter

$F IDD 1 is loaded whenever the user selects special option 99 from a menu. $F IDD 1 requests a disk unit number, and displays a system identification label and a 4-line, user-maintained disk ID label. Only the user at terminal number 1 is allowed to edit the disk ID.

The disk label presented is one of two labels in the SYSCO file on the designated platter. The label used by the posting and backup programs can only be edited using a similar utility in the $U Module.

$F LOAD - Program Loader
Most FICE+ and FICE+-based applications programs cannot be loaded directly with a LOAD statement because they require additional subroutines in memory in order to run. Consequently, the $F LOAD program is used to simultaneously load the main-line program and all required subroutines into memory.

The applications program to load is specified in Q$, a common system variable. $F LOAD examines this variable, locates the file containing the application program on disk, and extracts from it the required list of subroutines from the "load line", the last image statement in the program. $F LOAD then edits the list based on various parameters in the SYSTEM file and loads the edited list of subroutines and the application program as a unit.

$F LOAD performs the following conversions to the load line:

- If a trailing comma is omitted from the load line, it is added.
- The following program names are converted:

Program	Converted To
$F '210 <	$F '220
$F '210	$F '220
$F ‘210>	$F ‘220
$F '220 >	$F '220
$F '220 <	$F '220
$F '226 <	$F '226
$F '226 >	$F '226
$F '221 <	$F '221
$F '221 >	$F '221
$F '223 <	$F '223
$F '223 >	$F '223
$U HLP $	$U HLE $ (if the help parameter = E).

- The following programs are added to the load line if not already there:

  $F P%C $ if "% Complete" is turned on, $F ‘194 occurs, and either $F ‘221 or $F '237 occurs.
  $F ' 10 if $F '226, $F '240, or $F '241 occurs.
  $F ‘200 if Help is in use.
  $F '215 and $F '217 if the Help Editor is in use.
  $F '217 if $F '200 occurs (to bring in '219).

- The following programs are removed from the load line:

  $U HLP $ if Help is turned off in the system parameters.
  All global programs if the universal global partition is active.
  Programs that do not exist if the subroutine load check parameter is "Y".

$F LOAD also performs the following additional functions:

Sends a page eject to the currently selected printer if "bottom-of-form" is still outstanding;

Reinitializes the CRT;

Reroutes exection back to the original program when a rapid exit program completes;

Clears common variables declared by applications programs before returning to a menu; and

Supplies the program initialization sequence that appears at lines 10 through 20 in every program loaded by $F LOAD.

The release number of FICE+ and the most recent Bug-Byte update are specifed on line 10. These values may be inspected at any time by listing the current program in memory. This line also attaches the universal global partition if it exists.

Line 15 reselects the CRT for PRINT output, performs a couple of housekeeping chores, and transfers execution to Line 5000. This line also sets !a to the number of the release of FICE+ (times 100) for examination by the applications program. In Release 3.01, Q is set to 301.

Line 18 sets Q$ to "$F MENU". It can be used by applications programs to load a menu.

Line 19 contains the statement LOAD T "$F LOAD". This line is useful for chaining to a new program. The name of the new program must be in Q$.

Line 20 always contains a RETURN which may be used by ON GOTO statements in applications programs for "computed" RETURNs.

$F M/F 1 - General One-Level File Maintenance

$F M/F 1 is a generalized file maintenance program that can be used in place of an applications program to perform file maintenance on a specific file. It can be called from a menu but is not suitable for use with a rapid exit key.

$F M/F 1 makes a call to '210, the one-level file maintenance subroutine, using variables to identify the name of the record file to be opened. The module ID is assumed to be the same as that of the menu containing the reference to $F M/F 1 (Q2$). The three-character file name is assumed to be in the function code (Q8$). The file is opened in shared ("S") mode.

$F M/F 1 contains references to marked subroutine labels '251= 255, but does not contain labels for '184, '185 and '250. Consequently, $F M/F 1 cannot be used if any field on the file maintenance screen has a special case exit.

$F M/F 2 - General Two-Level File Maintenance

$F M/F 2 is the same as $F M/F 1 except that '220, the two-level file maintenance subroutine, is called instead.

$F MENU - Program Menu Driver

A FICE+ task consists of one or more programs that are chained together that perform a particular function. For instance, when printing a report, the typical program sequence is: sort setup program, $F SORT, $F SRT 0, $F SRT 2, $F MERGE, and $F MRG 1, report printing program. This entire sequence of programs is considered a task.

$F MENU is the entry point for all applications tasks run from a menu. It displays menus, manages the menu hierarchy, prompts the user for selections, enforces password security, and prevents other users from using the system while critical tasks such as Restructure are running.

$F MENU is also the exit point for all applications tasks. At the completion of a task, $F MENU should be loaded using $F LOAD. This is most easily done by branching to line 18 with a GOTO statement. Line 18 contains the appropriate FICE code necessary to load $F MENU.

Some FICE+ subroutines, such as '239, will return automatically to a menu. Others, such as '235, return automatically based on parameters supplied by the user. In all other cases, it is up to the application programmer to signal that the task is complete by loading $F MENU.

$F MENU performs several "clean up" functions each time it is loaded to prepare the terminal for the next task:

- All open FICE+ data files are automatically closed if $F MENU is loaded by the task that opened them. (To close files left open by a crash or rollout, use the Close Files utility in the FICE+ Utilities module.)
- The location of the SYSTEM file is checked and some common system variables are reinitialized from the System Parameters. Consult the Technical Guide in this manual for the list of affected variables.
- The device table is reinitialized.
- SYSSRT and SYSRPT files are freed.
- The Company ID and terminal date are reset to the values used at logon.

$F MERGE - Merge (Setup]

Sort buffers created in a SYSSRT file by '230 must be merged into a single buffer before sorted records can be accessed. $F MERGE is a setup program for $F MRG 1 and $F MRG 2 which actually perform the merge operation using different methods. $F MERGE load the proper program based on the results of the sort.

$F MERGE is normally automatically loaded by the FICE+ sort programs $F SRT 1 and $F SRT 2. Custom sorts written by the applications programmer using '230 must load $F MERGE directly.

$F MRG 1 - Merge [Fast]

$F MRG 1 is automatically called by $F MERGE in any situation that allows it. The merging algorithm used by $F MRG 1 is much faster than the method used in $F MRG 2, but is limited to a maximum of 255 sort buffers. Only in rare instances is this maximum ever exceeded.

$F MRG 2 - Merge [Standard]

$F MRG 2 uses a merging algorithm that can merge any number of sort buffers. It is automatically called by $F MERGE whenever the number of sort buffers generated by '230 exceeds 255.

$F MSG 1 - interterminal Messages

$F MSG 1 allows the user to send messages and monitor whether messages have been received. $F MSG 1 is loaded whenever the user presses SF '0 at a menu. For more information, refer to the discussion on Interterminal Messages in the Release 2.2 Development Guide.

$F QUEUE - FICE+Entry Point [Background]

$F QUEUE instructs the startup programs to attempt to run the background queue printing routines in the current partition. If the partition is currently in foreground (a terminal is attached to the partition), the startups will attempt to release the terminal before proceeding. If the terminal cannot be released; the partition will be brought up in foreground.

Background printing can also be initiated by loading $F START into a partition that is already in background. If, because of a crash or other cause, a terminal becomes attached to the partition, $F QUEUE must be used to get the partition running again in background.

$F RSB 1 - Control Background Queues

$F RSB 1 allows the user to set the printer device address for background printing, restart a report, or temporarily halt or cancel background printing. $F RSB 1 may be loaded by pressing Shift TAB (or Shift FN) from any menu, selecting special option 97 from any menu, or by assigning the program to a rapid exit key. For further information, refer to the System IV General Information Manual.

$F RSC 1 - Clear Report Queue

$F RSC 1 clears all reports from the queue currently selected by the terminal. Refer to the description of the $F TRT 1 program and the General Information Manual for queue selection information. This program is available as special option 93 from any menu and may be called from a rapid exit key.

$F RSP 1 - Print Report Queue [Setup]

$F RSP 1 prompts the user to select a printer and a queue then loads $F RSP 2 to begin print operations. $F RSP 1 is available as special menu option 92 and may be assigned to a rapid exit key.

$F RSP 2 - Print Report Queue [Control]

$F RSP 2 performs two main functions, initialization and queue control. It is automatically loaded by $F RSP 1 or $F RSW 1.

Many reports rely on the fact that $F MENU automatically performs certain reinitializations after each task. To maintain compatibility, $F RSP 2 reinitializes these same items between reports.

$F RSP 2 also manages the queue by setting up each report that is to be printed and loading $F SORT. When no more reports are to be printed, $F RSP 2 returns to $F MENU in foreground, or to $F RSW 1 in background.

$F RSW 1 - Poll Queues

$F RSW 1 only operates in a background partition. Its purpose is to poll the queue assigned to it on a regular basis and check for reports that need to be printed. If a report is found, $F RSP 2 is loaded.

$F RSW 1 also checks to see if D.AT.A. 3500 Word Processing (Release 2.0 or greater) documents have been submitted for background printing at the assigned terminal. If so, D.AT.A. 3500 is automatically loaded. When the documents have finished printing, $F RSW 1 is reloaded into the partition.

$F SORT - Sort ICommon Variables]

$F SORT is used to initiate the record sorting process for any sort procedure set up using '231 and '232. $F SORT initializes the common variables used during sort operations and loads $F SRT 0. $F SORT may be loaded directly. It does not require the use of $F LOAD.

$F SRT 0 - Sort [Setup]

$F SRT 0 opens the record File to be sorted, sets up sorting control parameters, and loads either $F SRT 1 or $F SRT 2.

$F SRT 1 - Sort [Keys]

$F SRT 1 performs a key sort based on the data in the primary key path file. This program executes somewhat faster than $F SRT 2, but it can only be used when all fields which are sorted and range checked belong to the primary key, and no "OR" checking is involved.

If the primary key is ISAM, any range checks applied to it are used to reduce the number of keys in the path file that must be checked. On completion, $F SAT 1 automatically loads $F MERGE.

$F SRT 2 - Sort (Records]

$F SRT 2 performs a record sort based on the data in each record in the file. This program is used whenever a field to be sorted or range checked falls outside the bounds of the primary key, or if any "OR" checking is performed. On completion, $F SRT 2 automatically loads $F MERGE.

$F SSD 1 - Display Detail System Status

$F SSD 1 displays detailed information about each terminal on the system, including the virtual terminal number, user a, foreground status, and background status. It is available as special menu option 95. Refer to the System N General Information manual for futher information.

$F START - FICE+ Entry Point [Cold Start]

$F START is the principal entry point to FICE+. $F START normally instructs the registration startup programs to execute a cold start. If $PSTAT contains "$F CTNUE" in the first eight bytes, a warm start is attempted. For further information, refer to the discussions on the FICE+ startup programs elsewhere in this Guide.

A SELECT #15 /000 statement is contained in $F START to signify that 16 device table slots (0 through 15) are to be used. To invoke a larger table (in those languages that support this feature) you must modify this statement. A scratch-and-save at SF '30 is included to make the change easier. For further information, read the section on extended device tables elsewhere in this Guide.

$F STM 1 - Set Time

This program allows the user at physical terminal number one to set the time of day. Other users may use it to inspect the time.

$F TRT 1 - Edit Terminal Traits

$F TRT 1 allows the user to edit parameters that describe the terminal, terminal printer, and queues to be used. Available options for many of the fields are displayed when the field is selected for modification. For further information, read the discussion on Terminal 'Traits in the Run-Time Environment section.

FICE+ Special Subroutines

The special subroutines are used principally by the $F programs but are available for use by application programs as well.

File Name	Description
$F %%% $	Display Percent Complete
$F ' 10	Pack/Unpack Definition File (S Variables)
$F ' 11	Pack/Unpack Definition File (T Variables)
$F CAL $	Display Calendar
$F CTR $	$F '250 Overlay
$F DVT $	Display Device Table
$F P%C $	Print Percent Complete
$F SRT $	Select a Sort File
$F SSD $	Display Summary System Status
$F TIME	Get Time

Special Subroutine Descriptions

$F %%% $ - Display Percent Complete

This is a new subroutine. It will eventually replace $F P%C $, which is unwieldy and relies on overlays to function properly. This subroutine is currently used internally. In a later release, its parameters and usage will be documented for use by applications programs.

$F ' 10 - Pack/Unpack Definition File (S Variables)

This subroutines is used to pack and unpack the definition file field records and the definition file parameters. It is constructed similarly to the pack/unpack subroutines used in TOM applications.

To unpack a field record, pass a "U" parameter; to pack, use a "P". The field record is assumed to be in QI$0 since field records can be accessed using '200 using the field name as a key.

Parameter records are located in the next to last sector of the definition file and must be accessed in BA mode. $F ' 10 assumes that the parameters record is in Q$U. Use "UP" to unpack and "PP" to pack the parameters.

$F '10 used "S" variables. Refer to the definition file technical documentation for more information on $F ' 10's variables.

$F ' 11 - Pack/Unpack Definition File (T Variables)

$F' 11 is identical to $F' 10 except that "T" variables are used. This allows two field records or two sets of parameters records to be examined or compared simulataneously using documented variables.

$F CAL $ - Display Calendar

$F CAL $ displays a calendar for the month containing a given date and highlights the date. The following parameters are passed to this subroutine ('21):

1st parameter:	A packed date in YYMMDD format.
2nd parameter:	Start CRT row. If negative, highlighting and boxes on the date are cleared
3rd parameter:	Start CRT column

$F CTR $ - $F '250 Overlay

$F CTR $ overlays the DEFFN '250 in $F ‘250 with a DEFFN '25, allowing $F ‘250 to be used in a program that already has a predefined DEFFN '250. For further information, see the Release 2.2 Technical Reference Manual.

In file maintenance and general inquiries, calls to '226 can invoke $F '250 capabilities which can be used by the applications programmer. See the discussions under $F '226 and $F '240 for further information.

You are encouraged to eliminate the use of $F CTR $ in your applications where possible. Because of its awkward implementation, it is hoped that this subroutine can be removed from the $F subroutine library at some future date.

$F DVT $ - Display Device Table

$F DVT $ displays the device table defined in the system parameters. This display appears on the CRT row indicated by a parameter passed to this routine. It has the following format:

```
DEFFN '18(row)
```

A maximum of 14 platters are displayed on two lines. The following values are returned:

Q = Number of devices in the device table
Q8$(64)4 = The first 64 addresses in the device table.

$F P%C $ - Print Percent Complete

$F P%C $ displays the percent complete on the screen during report printing and during a physical sequential read of a FICE+ file using $F ‘221/222. For more information, refer to the 2.2 Technical Reference Manual.

This subroutine will eventually be replaced by $F %%% $. It is recommended that you do not use it in your applications.

$F SRT $ - Select a Sort File

$F SRT $ is a program subroutine that allows the user to select a sort file and attach it to your terminal. It has the following format:

```
DEFFN '252(SYSSRT#,Sectors)
```

SYSSRT# is the sort file number to select. If this parameter is 0, the first available sort file is selected.

Sectors is the minimum sectors that the selected sort work file must have. (Total work space = size of the sort work file - 15) This parameter is only used if the first parameter is 0.

Returns:	Q	=	Sort file number selected
		=	0 if no sort work file was found
	Q0	=	Terminal number using the sort file
		=	0 if the sort work file is not attached to another terminal

If your terminal is already attached to another sort work file, this subroutine releases it before attaching the selected sort work file to your terminal.

To correct an anomaly in previous releases, $F SRT $ sets U1(2) = 1 corresponding to "print report now". This problem was originally brought up in a Bug-Byte. It will be addressed in a future release of FICE+.

$F SSD $ - Display System Status
$F SSD $ is a subroutine program residing within line numbers 5500 - 5549. It displays a grid of all possible 255 terminals that indicates the status of each. It has the following format:

```
DEFFN '22(Type, Message)
```

Type = 1, Display company usage status
Type = 0, Display system usage status

Message = 1, Display "Close Files" message
Message = 0, No display

$F TIME - Get Time
$F TIME Returns the time of day in "hh:mm:ss xx" format. The system variable U$(19) specifies whether the time is in 12- or 24-hour format. If it is in 12-hour format, xx contains AM or PM. Otherwise, it is blank.
To access the time routine, the application program must include the program $F TIME on its load line. The statement GOSUB 30 returns the formatted time-of-day in the variable Q$. This routine will format a custom time value by placing this value in Q$ and executing the statement GOSUB 31.

Section 14

FICE+Subroutines

This section individually describes the standard subroutines in the FICE+Module. To eliminate confusion, every subroutine is mentioned even if its function remains unchanged from Release 2.3. (For information on special subroutines, refer to Section 13)

FICE+Subroutine Files

The standard set of FICE+ subroutines is used extensively by FICE+ and FICE+-based applications to manage files and prepare reports. FICE+ subroutines that are frequently used together are stored in a single program file on the system disk and loaded as a group. This table shows the subroutines in each program file, along with a brief description of each:

Program	Subroutine	Description
$F '180	'180	Position cursor
'181	Numeric input	
'182	Alphanumeric input	
'183	Display error message	
'187	Pack date (see $F '187)	
'188	Unpack date (see $F '188)	
$F '186	'186	Add days to date $F '187
'187	Pack date (see $F '180)	
$F '188	'188	Unpack date (see $F '180)
$F '189	'189	Days between two dates
$F '190	'190	Prepare printer
'191	Prepare printer for special forms	
$F '193	'193	Forms alignment
$F '194	'194	Select CRT (see $F '196)
'195	Select printer with message (see $F '196)	
'196	Select printer (see $F '196)	
'197	Standard page heading	
'198	Form Feed	
'199	Update Line Count	
$F '196	'194	Select CRT(see $F '194)
	'195	Display message and select printer (see $F '194)
	'196	Select printer (see $F '194)

FICE+Subroutines

Program	Subroutine	Description
$F '200	'200	Record access
'201	Open FICE+ file (with return code)	
'202	ISAM record access	
'203	Write bytes	
'204	Read bytes	
'205	Change key/delete status	
'206	Read key/delete status	
'207	Change record protection	
'211	Open FICE+ file (no return code)	
'212	Set logical record pointer	
'213	Write record	
'214	Read record	
$F '208	'208	Get modules in system
$F '209	'209	Exclusive access
$F '215	'215	Add record
'216	Delete record	
$F '217	'217	Add key
'218	Delete key	
'219	Find key	
$F '220	'210	One-level file maintenance
'220	Two-level file maintenance	
$F '221	'221	Physical sequential access [Setup]
'222	Physical sequential access [Read]	
$F '223	'223	Key sequential access [Setup]
'224	Key sequential access (Read]	
$F '225	'225	Read next second-level record
$F '226	'226	Store and display field
'227	Skip fields on input	
$F '230	'230	Sort
$F '231	'231	Open file for sorting
'232	Establish sort sequence	
$F '233	'233	Edit sort sequence
$F '234	'234	Store additional report information
$F '235	'235	Select print options

Program	Subroutine	Description
$F '236	'236	Retrieve additional report information
$F '237	'237	Open sort work File
'238	Retrieve sorted record numbers	
$F '239	'239	Generalized report printing
$F '240	'240	Generalized inquiry
$F '241	'241	Create record file
$F '242	'242	Initialize file
$F '243	'243	Set file status bytes
$F '244	'244	Inspect file status bytes
$F '245	'245	Get next sequential numeric key
$F '246	'246	Rebuild path files
$F '247	'247	Add file to file index
$F '248	'248	Delete file from file index
$F '249	'249	Find file in file index
$F '250	'250	Display prompt

Dedicated Marked Subroutine Labels

Several marked subroutine labels do not appear in the list of FICE+ subroutines and are intended for special purposes or are reserved for future use. Reserved labels must never be used in an applications program.

Label	Description
'184	Pre-input special case exit
'185	Post-input special case exit
'192	Standard intercept subroutine
'228	Reserved
'229	Reserved

FICE+ Subroutine Descriptions

The following pages describe the characteristics of the standard set of FICE+ subroutines. Each description includes structured information on the following topics, optionally followed by discussion where appropriate.

Parameters: The name of each parameter required by the subroutine is given in the order in which it must be passed to the subroutine. Additional information includes: receiving variable type (numeric [N] or alpha [A]); its description; and its range of legal values if appropriate.

Example: The appearance of a typical subroutine call in an applications program.

Returns: These are values in variables supplied to the applications program by the subroutine.

File: The name of the disk file(s) that contain the subroutine. To use the subroutine, this file name must be included on the load line of the applications program.

External Subs:Some FICE+ subroutines need the use of subroutines in other program files in order to function. This is a list of the names of program files that must appear on the applications load line when this subroutine is used.

Assumptions: This is an optional topic that describes conditions that are necessary for the subroutine to operate properly, but which are not checked for by the subroutine.

Certain assumptions are always made:

- The CRT is SELECTed when using CRT oriented subroutines.
- Record files have been opened before attempts are made to access them.

The assumption of reasonableness is always made. In other words, FICE+ code assumes that the programmer wants the subroutine to work properly and isn't merely trying to find a way to break it. Therefore, blatantly illegal parameters are never checked, nor are they documented.

'180 - Position Cursor

This subroutine positions the cursor at an absolute location on the CRT.

Parameters:	Row	[N] 0 to CRT rows (U1 - 1)
	Column	[N] 0 to CRT columns (U2 - 1)
Example:	GOSUB '180(10,20)	
Returns:	None	
Program File:	$F ' 180	
External Subs:	None	

$F '180 consists of a simple PRINT AT statement. The cursor is turned off during positioning and turned back on before the subroutine returns.

'181 - Numeric Input

'181 accepts numeric keyboard input from a specific position on the CRT.

Parameters:	Row	[N] 0 to CRT rows (U1 - 1)
	Column	[[N] 0 to U2 - total digits - 3
	Type	[N] 0 or 1
	Default	[N]
	Minimum	[N] Low range check
	Maximum	[N] High range check
	Special Case	[N] 0 through 999
	Error Line	[N] 0 through U1-2
	Justification	[A] "R" (numerics are right justified)
	Decimals Left	[N] Digits left of decimal (0 to 13)
	Decimals Right	[N] Digits right of decimal (0 to 13)

Example: GOSUB '181(2,Q,1,X,0,10,0,3,"R",2,0)

Returns: Q = entered value
WO = special intercept code
W$ = last operator keystroke
W9 = 0 ('183 error status is cleared)

Program File: $F '180

External Subs: None

Assumptions: Maximum > = Minimum
Decimals Left + Decimals Right < = 13

Row and Column specify the location of the left-most digit in the number to be entered. An error message is generated on row Error Line if the entered value is outside the range of Minimum through Maximum.

Numeric input requires one position to the right of the last digit to display the sign of the number. Consequently, a digit must never be placed in the last column of the CRT.

Type 1 fields are signed numbers. Any legal value within the Minimum and Maximum may be entered. Type 0 fields are unsigned numbers. "Unsigned" describes their positive range and how they are stored on disk, not how they are entered. The operator may still enter a sign (+ or -), which appears to the right of the number.

'181 ensures that Minimum is > = zero with Type 0 fields.

Two entry modes are supported, fixed-point (the default) and floating-point.

In fixed-point, '181 mimics a traditional adding machine which has no decimal point key. The decimal point is assumed to be at a particular location and does not move. Thus, to key in 12.00, the operator must type "1200".

If the operator presses the decimal point key, '181 enters floating-point mode. All digits previously entered are assumed to be to the left of the decimal point while any additional digits are assumed to be on the right. The displayed decimal point "floats" to show the proper value of the number. If the ERASE key is pressed or the decimal point is driven off the display, fixed-point operation is resumed.

Excess digits supplied as a default in Default are truncated.

If Special Case > 0, '181 will make a call to '185 when the user ends field entry with the RETURN key. A DEFFN' 185 must appear in the application program to avoid a program error. '185 will be passed the value of Special Case and the value of the user's entry in Q. If the programmer changes the value of Q, the change will automatically be reflected on the CRT when '185 returns.

'181 supports only "right justification" (digits are entered from the right of the displayed number) and the value of Justification is ignored. Justification should be given a value of "R" to remain compatible with future enhancements.

'181 supports Rapid Exit, Standard Intercept, and Special Intercept keystrokes.

A Rapid Exit keystroke is a special function key labeled '16 to '31. Rapid Exit programs are defined in the System Parameters. If the Rapid Exit program for the key is blank or does not exist on the system disk, the keystroke is ignored. Otherwise, the identified program is immediately loaded with $F LOAD.

A Standard Intercept keystroke is a special function key labeled '1 to '3. If any of these three keys is pressed, '181 makes a call to '192 and passes a single argument to it containing the number of the function key (1, 2, or 3). For consistency with '182, the value currently entered by the operator is in Q$ when '192 is called, even though this is a numeric entry routine. The true value is available by CONVERTing Q$. The value in Q$ should be preserved during '192 processing or unpredictable results can occur.

$F '180 contains a null '192 subroutine at line 3000 to process standard intercept keystrokes by default. Any programmer-supplied '192 subroutine must occur prior to line 3000 to be effective.

It is ill-advised for Standard Intercept subroutines to make any calls to a FICE+keyboard entry subroutine. The values of critical and undocumented variables will be disturbed and will likely cause '181 to malfunction.

Special Intercept keystrokes override all other keystrokes, including Rapid Exits and Standard Intercepts. They are defined by the programmer by setting W0$() to their keyboard values prior to calling '181. W0$(1) is set to special function key values, W0$(2) to standard keystroke values.

If a Special Intercept key is pressed, '181 immediately exits. No range checking or validation is performed and Special Case is ignored. The following values are returned:

Q = the value of the number entered so far.
W$ = the value of the keystroke causing the Special Intercept.
W0 = s number showing the type of keystroke that caused '181 to exit.

> If W0>0, special function key STR(W0$(1),W0,1) caused a Special Intercept.
> If W0 < 0, standard key STR(W0$(2),ABS(W0),1) caused a Special Intercept.
> If zero, no Special Intercept was taken.

Trailing spaces in W0$(1) and W0$(1) are ignored. If W0$()=" ", all Special Intercept processing is disabled. '181 sets W0$()=" " whenever it exits.

'182 accepts alphanumeric keyboard input from a specific position on the CRT.

Parameters:	Row	[N] 0 to CRT rows (U1 - 1)
	Column	[N] 0 to U2 - Length
	Type	[N] 2, 3, 4, 6, or 6
	Default	[A] The default value
	Minimum	[A] Low range check
	Maximum	[A] High range check
	Special Case	[N] 0 through 999
	Error Line	[N] 0 through U1-2
	Required/Case	[A] R, r, O, or o
	Justification	[A] R or L
	OK Input	[A] Character list or blank
	Length	[N] 1 through 64

Example: GOSUB '182(2,Q,5,X$," "," ",0,3,"r","L"," ",3)

Returns: Q$ = entered value
W0 = special intercept code
W$ = last operator keystroke
W9 = 0 ('183 error message status is cleared)

Program File: $F '180

External Subs: None

Assumptions: Maximum > = Minimum
Default is valid for the Field Type

Row and Column specify the position of the first character of the field to be entered.

An error message is generated on row Error Line if the entered value is outside the range of Minimum through Maximum. Blank Minimum and Maximum are translated to ALL(00) and ALL(FF) respectively prior to keyboard entry.

If Special Case > 0, '182 will make a call to '185 when the user ends field entry with the RETURN key. A DEFFN' 185 must appear in the application program to avoid a fatal program error. '185 will be passed the value of Special Case and the value of the user's entry in Q$. If the programmer changes the value of Q$, the change will automatically be reflected on the CRT when '185 returns.

If Required/Case is set to "R" or "r" (required), an error message is generated on CRT row Error Line if the entered field is blank (or null for dates). If Required/Case is set to "O" or "o" (optional), fields may be left blank.

'182 (Cont'd)

If Required/Case is set to "R" or "O" with Type 5 fields, upper-and-lower case entries can be made. If Required/Case is set to "r" or "o", any lower-case alphabetic characters are immediately translated to uppercase. This translation occurs before OK Input characters are checked.

Justification designates whether '182 fields are left (L) or right (R) justified. Justification takes place before Minimum and Maximum are checked.

If OK Input is blank, all characters are accepted from the keyboard. Otherwise, only the keystroke values specified in OK Input are accepted. Any trailing spaces in OK Input are ignored.

The following Field Types are supported by '182:

Type 2 - Dates

'182 expects date arguments to be in packed decimal format. The entered date is returned in the first three bytes of Q$ in packed decimal format. Dates are automatically validated for authenticity. On optional date fields, a null date of HEX(000000) may be returned for non-entered fields.

Type 2 adjusts the following parameters:

Default	HEX(000000) if blank
Justification	Ignored
Required/Case	Case is ignored
OK Input	Digits plus space
Length	8

'182 uses '187 and '188 to convert packed dates to and from display format during editing.

Type 3 - Logic

Logic fields require that the operator enter "Y" or "N".

Type 3 adjusts the following parameters:

Minimum	HEX(00)
Maximum	HEX(FF)
Required/Case	Required with case translation ("r")
Justification	Meaningless
OK Input	NY
Length	1

Type 4 - Digits

Type 4 accepts digits (0-9) from the keyboard, justifies the field, and translates any blanks to

ASCII zeros.

Type 4 adjusts the following parameters:

OK Input	Digits plus space

Type 5-Alphanumeric

Type 5 accepts any keyboard input.

Type 6 - RETURN

Type 6 accepts only the RETURN key as a legal keyboard entry and adjusts the following parameters:

Minimum	Low values
Maximum	High values
Required/Case	O
Justification	Meaningless
OK Input	RETURN key
Length	1

'182 supports Rapid Exit, standard Intercept, and Special Intercept keystrokes.

A Rapid Exit keystroke is a special function key labeled '16 to '31. Rapid Exit programs are defined in the System Parameters. If the Rapid Exit program for the key is blank or does not exist on the system disk, the keystroke is ignored. Otherwise, the identified program is immediately loaded with $F LOAD.

A Standard Intercept keystroke is a special function key labeled '1 to '3. If any of these three keys is pressed, '182 makes a call to '192 and passes a single argument to it containing the number of the function key (1, 2, or 3). The value currently entered by the operator is in Q$ when '192 is called. The value in Q$ should be preserved during '192 processing or unpredictable results can occur.

$F ‘180 contains a null '192 subroutine at line 3000 to process standard intercept keystrokes by default. Any programmer-supplied '192 subroutine must occur prior to line 3000 to be effective.

It is ill-advised for Standard Intercept subroutines to make any calls to a FICE+keyboard entry subroutine. The values of critical and undocumented variables will be disturbed and likely cause '182 to malfunction.

Special Intercept keystrokes override all other keystrokes, including Rapid Exits and Standard Intercepts. They are defined by the programmer by setting W()$() to their keyboard values prior to calling '182. W()$(1) is set to special function key values, W()$(2) to standard keystroke values.

If a Special Intercept key is pressed, '182 immediately exits. No range checking or validation is performed and Special Case is ignored. The following values are returned:

Q$ = the value entered so far.
W$ = the keystroke causing the Special Intercept.
W0 = the type of keystroke that caused '182 to exit.

> If W0>0, special function key STR(W0$(1),W0,1) caused a Special Intercept.
> If W0<0, standard key STR(W0$(2),ABS(W0),1) caused a Special Intercept.
> If zero, no Special Intercept was taken.

Trailing spaces in W0$(1) and W0$(1) are ignored. If W0$()=" ", all Special Intercept processing is disabled. '182 sets W0$()=" " when the operator presses RETURN.

'183 - Display Error Message

'183 accepts an error message for display by ‘181 and '182.

Parameters:	Message	[A] Up to 76 characters in length.
Example:	GOSUB '183("Warehouse Not on File")	

Returns:	W9 = 1

Program File: $F ' 180

External Subs:	None

The error message is centered in a row of asterisks and displayed on row Error Line during '181 or '182 processing. Error messages greater than 76 characters are truncated.

'183 does not actually display the error message. It sets a flag (W9=1) that is checked by the keyboard input subroutines '181 and '182. If this flag is set and either '181 or '182 given control of the program (either by being called directly or returning from a special case exit), the error is displayed to the operator.

The keyboard subroutines always initialize W9 to zero when they return. Applications can take advantage of this fact to test for error conditions.

For example:

```
8400   GOSUB '182(2,Q,4,X5," ",»  ",0,3,"R»,"R»," ",5)
       :REM Get Vendor Number
8410   GOSUB '32
       :REM Check Vendor's Transactions
8420   ON W9 GOTO 8400
       :REM If '32 called '183, re-prompt and display error
```

If '183 is called more than once between keyboard input, the message displayed is the message contained in the last call to ' 183.

'186 - Add Days to Date

'186 computes a new date given a date and a number of days to add to (or subtract from) it.

Parameters:	Date	[A] Any legal packed date
	Days	[N] -36525 to +36525 (1 century)

Example:	GOSUB '186(Q1$,X*7)

Returns:	Q$ = New Date (packed format)
Program File:	$F ‘186
External Subs:	None

Assumptions: Date contains a valid packed date in YYMMDD format The magnitude of Days does not exceed one century

If Date is blank or Date contains an invalid month, New Date is set to HEX(000000) regardless of Days.

Days can be both positive and negative. If positive, a future date is calculated. If negative, a previous date is returned.

If the year in Date is less than the Century Rollover Year (U8), the date is assumed to be in the 21st Century. The Century Rollover Year is determined in the System Parameters. U8 may be changed by the applications programmer. U8 will be reinitialized to the value in the System Parameters by $F MENU.

'187 - Pack Date

'187 accepts a date in display format and converts it to packed decimal format.

Parameters:	Display Date	[A] Date in display format
Example:	GOSUB ' 187 (Q$)	
Returns:	Q$ = Packed Date	

Program File:	$F ‘180 $F '187
External Subs:	None
Assumptions:	Display Date is assumed to conform to the display format specified in the System Parameters.

If Display Date is blank, a null date of HEX(000000) is returned.

Display Date is converted two digits at a time. If valid digits are not found where expected, that portion of the date is converted to HEX(00).

'188 - Unpack Date

'188 converts a date from packed decimal format to display format.

Parameters:	Packed Date	[A)
Example:	GOSUB '188(Q$)	
Returns:	Q$ = Display Date	

Program File: $F '180
$F '188

External Suns: None

If Packed Date is blank or HEX(000000), Display Date is set to blank.

The hex digits in Packed Date are unpacked using the System Date Mask in U8$ and the System Date Format in U7$. Both US$ and U7$ are maintained in the System Parameters.

'189 - Days Between Two Dates

'189 calculates the number of days between two dates.

Parameters:	Date1	[A] Packed format
	Date2	[A] Packed format

Example: GOSUB '189(A1$,Q1$)

Returns: Q = Days

Program File: $F ‘189

External Subs: None

Assumptions: Date1 and Date2 contain packed decimal numbers
The difference between Date1 and Date2 < = one century

Days is computed by subtracting Date1 from Date2.

If either Date1 or Date2 does not conform to a valid date, Days is undefined.

If the year in either date is less than the Century Rollover Year (U8), the date is assumed to be in the 21st Century. The Century Rollover Year is determined in the System Parameters. U8 may be changed by the applications programmer. U8 will be reinitialized to the value in the system Parameters by $F MENU.

'190 - Prepare Printer

'190 displays the Printing Parameters screen and allows the operator to select an output device and edit certain attributes.

Parameters:	Title	[A] 48 characters
Form Feed Type		[A] blank, A, L, P
Rows		[N] -1, 0, 1-255
Columns		[N] -1, 0, 1-255

Example: GOSUB '190("Productivity Report"," ",-1,132)

Returns: R1 = Rows
R2 = Columns

R4 = Device number (-1, 0, 1-9)
R1$ = Device address

Program File: $F ‘190

External Subs: $F ' 180
$F ' 194 or $F ' 196

Form Feed Type determines how the paper or form in the printer is positioned at top-of form. There are four possible values:

A (automatic)	FICE+uses HEX(OC)
L (line feed)	FICE+calculates and sends the required number of HEX(0A)s, based on the line counter (R8) maintained by '199.
P (programmer)	FICE+does nothing. Form feeds are the responsibility of the programmer. Blank FICE+uses the system default.

Rows and Columns is used to specify how the default values are determined for these two fields, which the user may override. Each can have two possible values:

Value > 0	The value to be used as a default.
Value = -1	FICE+uses the system default.

For terminal printers, the Terminal Traits are checked first for system default values. If the Terminal Traits record contains a reference to the system default, or if the printer is a system printer, the Printer Attributes record is checked. If the Printer Attributes record contains a reference to the system default, the data is obtained from the system Parameters.

Just before exiting '190 or '191 displays a screen that should remain while the report is printing. Thus, all special prompts and entries should be completed before 190 or 191 is called.

'191 - Prepare Printer for Special Forms

'191 displays the Printing Parameters screen and allows the operator to select an output device and edit certain attributes. '191 calls '193 to prompt the user to align special forms.

Parameters:	Title	[A] 48 characters
Form Feed Type		[A] blank, A, L, P
Rows		[N] -1, 0, 1-255
Columns		[N] -1, 0, 1-255
Prompt		(11) 64 characters
Tab		[N]

Example: GOSUB '191("Invoice"," ",54,80,"Align at Top Left",0)

Returns: R1 = Rows
R2 = Columns

R4 = Device number (-1, 0, 1-9)
R1$ = Device address

Program File:	$F ' 190
External Subs:	$F ‘180 $F 193 $F ‘194 or $F ‘196

'191 is virtually identical to '190 except for the call to '193. For further information, refer to ‘190 and '193.

'193 - Forms Alignment

'193 prompts the user to mount special forms and prints an alignment mark.

Parameters:	Prompt	[A] 64 characters
	Tab [N]	
Example:	GOSUB '193("Align Invoice at Bottom Left Corner",0)	
Returns:	None	

Program File: $F '193

External Subs:	$F ‘180 $F '194 or $F '196

Assumptions: Printer to use is the last selected by the user.

After prompting the user to mount special forms, the user has the ability to print an alignment mark. The mark, "XXXXX", is printed at column Tab if the user chooses to print the mark. After printing a mark, the user has the option of realigning the form and reprinting the mark. This sequence can continue indefinitely.

The alignment mark is followed by a carriage return. This sets the printer at the beginning of the next line. If the user opts not to print an alignment mark, a carriage return is still issued to ensure that the form is always in the same place following the alignment prompt.

'193 will send an industry-standard VFU sequence to the printer if the VFU? field in the Printer Attributes record is set to "Y". The user must print an alignment mark in order to get a VFU sequence.

Many printers support industry-standard VFU sequences which change the forms length automatically handled by the printer. This allows Forms Type "A" and page eject characters (HEX(0C)) to be used for top-of-form. The forms length used is based on the number of rows selected by the operator in the previous call to '190 or '191.

'194 - Select CRT

'194 selects the CRT for printed output.

Parameters:	None
Example:	GOSUB ' 194
Returns:	None
Program File:	$F ‘194 or $F ‘196
External Subs:	None

'194 executes a SELECT PRINT 005(U2) statement where U2 is the number of CRT columns.

'195 - Select Printer with Message

'195 displays a message on the second line of the CRT and calls '196 to select the printer.

Parameters:	Message	[A] 64 characters

Example: GOSUB '195{"Printing Customer List")

Returns:
R1 = Rows
R2 = Columns
R3 = Expanded print (1=Y, 2=N)
R4 = Device number (-1, 0, 1-9)
R1$ = Device address

Q = Status
-1, illegal device address
0, printer not ready
1, printer ready
2, printer in use

Program File: $F '194 or $F '196

External Subs: None

Assumptions: The printer selected is the last printer accepted by '190 or '191.

For further information, refer to '196.

'196 - Select Printer

'196 selects the current printer for output.

Parameters: None

Example: GOSUB ' 196

Returns:	R1 = Rows
	R2 = Columns
	R3 = Expanded print (1=Y, 2=N)
	R4 = Device number (-1, 0, 1-9)
	R1$ = Device address
Q = Status	-1, illegal device address
	0, printer not ready
	1, printer ready
	2, printer in use

Program File: $F '194 or $F '196

External Subs: None

Assumptions: The printer selected is the Last printer accepted by '190 or '191.

Rows and Columns are the paper dimensions selected by the user in the last call to '190 or '191.

R3 is determined based on the parameters set in the Printer Attributes file. If the Expanded Print flag (R3), is set to 1 a HEX(0E) is assumed to turn on expanded print, and a carriage return is assumed to turn it off.

The Device Number is a quick reference to the type of device in use. It can have three values:

R4 = -1	CRT
R4 = 0	Terminal printer
R4 > 0	System printer (Device Number = last digit of the device address)

A Status of 0 can only be returned on Fasstcom system printers.

'197 - Standard Page Heading

'197 calls '198 to execute a form feed and reset the line counter. It then prints a standard page heading showing the company name, report title, date, and page number.

Parameters:	Left Margin	[N]
	Report Title	[A].64 characters
	Right Margin	[N]

Example: GOSUB '197(0,"Cash Receipts Journal",0)

Returns: None

Program File: $F ‘194

External Subs: None

Assumptions: The printer has been selected by ‘195 or '196.

The Left Margin and Right Margin affect the location of the Standard Page Heading. Margins other than zero are generally used only in special circumstances.

If expanded print is supported (R3=1), Report Title is printed in expanded print. Otherwise, it is double spaced. Neither expanded print or double spacing is used where the paper width is too narrow to contain it.

'198 - Form Feed

'198 executes a form feed, increments the page counter, and resets the line counter.

Parameters:	None
Example:	GOSUB '198
Returns:	None
Program File:	$F ‘194
External Subs:	None
Assumptions:	The printer has been selected using '195 or '196.

'198 checks the keyboard buffer for a "P" indicating a paper-out condition; resets the line counter (sets R8 = 0); and executes top-of-form commands according to the Form Feed Type specified in '190 or '191.

'199 - Update Line Count

'199 keeps track of the number of lines printed per page, and processes commands from the keyboard during report printing.

Parameters:	Count	[N] -999, < 0, 0, > 0
Example :	GOSUB ' 199 (1)	
Returns:	R8 = Current Line Count	
Program File:	$F '194	
External Subs:	None	

Count is interpreted as the intention to print the stated number lines as a group on the report. If there is not enough room to hold the number of lines stated by Count, '199 calls '261 for top-of-page processing, then assumes that the intended lines appear on the new page. Line Count is adjusted accordingly.

'199 also checks the keyboard for a command. The following keys are processed:

"C"	Continue	Required after a Halt, Restart, or Paper Change
"H"	Halt	Stops report

"R"	Restart	Sets up report io restart from the beginning
"0" to "9"	Set CRT FICE+	(0=fastest, 9=slowest)
"Cancel"	Cancel printing	

'199 supports the ability to process keyboard commands without updating Line Count, and to inhibit keyboard processing altogether. The type of processing is determined by the value of the parameter Count:

Count > 0	Line Count updated, keystrokes are processed
Count = 0	Line Count not updated, keystrokes are processed
Count < 0	Line Count updated by ABS(Count), keystrokes are not processed
Count =-999	Line Count not updated, keystrokes are not processed

It is good programming practice to regularly call '199 with a Count of zero on programs with significant amounts of processing between lines of print to reduce the response time if the user decides to cancel the report.

On certain reports, such as a transaction journal, it is sometimes necessary to prevent the report from being cancelled to ensure that the report is complete. A negative Count performs this function.

The Percent Complete display during a standard report ('239) or a physical sequential read of a file ('221) is activated by a call to '199,

'200 - Record Access

'200 accesses a record by key.

Parameters:	File Number	[N] > = 0
Path Number	[N]	
Protect Code	[N] 0 or 1	
	Key	[A]

Example: G0SUB '200(4,4,0,A05)

Returns: Q1$() = Record buffer

If STR(Q1$(),,1) < HEX(FF), record was found
If STR(Q1$(),,1) = HEX(FF), record not found or pseudo-deleted

If STR(Q1$(),,2)=HEX(FFFF), record not found
If STR(Q1$(),,2)=HEX(FFFE), record is pseudo-deleted

Q0 = Terminal protecting the record

If Q0 = 0, record in Q1$() is available
If Q0 > 0, terminal Q0 is using the record in Q1$()

Z0 = Record number of record in Q1$()

If Z0 = 0, record was not found.
If Z0 > 0, record was found, but may be pseudo-deleted.

Program File: $F '200

External Subs: $F ‘217 ('219)

The Protect Codes are:
0 = Unprotect
1 = Protect

In general, if a record will only be read, then the record should not be protected. If the record will be read, changed, and rewritten, and the file was opened in shared ("S") mode, then the record should be protected when it is accessed and unprotected when it is rewritten.

A record is returned in Q1$() regardless of whether or not it is protected. The application program must check the return code contained in Q0 where appropriate.

If a record is read with '200 using a Protect Code = 1, Q0 = 0 if the record was successfully protected. Otherwise Q0 = the number of the terminal using the record.
If File Number is zero, a '219 is performed on the key, but Q1$() is undefined.

'201 - Open FICE+ File (with Return Code)

'201 opens (or closes) a FICE+ record file and its associated path files and returns a code indicating whether or not the open was successful.

Parameters:	File Number	[N]
Module	[A)	
File Name	[A]	
Access Class	[A)	

Example: GOSUB '201(3,"IC","PR1","R")

Returns:	Q	= 1, file successfully opened
	= 0, open failed	
Q0$	= Name of file opened	
Q1	= Start sector of file	
Q2	= End sector of file	
U(File#)	= Disk Unit	
Z$(File#)	= File parameters	

Program File. $F ‘200

External Subs: None

The file access classes are:

X = Exclusive Access
W = Exclusive write (shared read)
S = Shared write
P = Write protect
R = Read
C = Close

'202 - ISAM File Operation

'202 accesses a record by ISAM key.

Parameters:	File Number	[N] > = 0
Path Number	[N]	
Protect Code	[N] 0 or 1	
Key	[A]	
Access Code	[A]	

Example: GOSUB '202(4,4.1,0,A$(1),"GE")

Returns: Q1$() = Record buffer

If STR(Q1$(),,1) < HEX(FF), record was found
If STR(Q1$(),,1) = HEX(FF), record not found or pseudo-deleted

If STR(Q1$(),,2)=HEX(FFFF), record was not found
If STR(Q1$(),,2)=HEX(FFFE), record is pseudo-deleted

Q0 = Terminal protecting the record

If Q0 = 0, record in Q1$() is available

If Q0 > 0, terminal Q0 is using record in Q1$()

Z0 = Record number of record in Q1$()

If Z0 = 0, record was not found.
If Z0 > 0, record was found, but may be pseudo-deleted.

Q$ = Key found by' 202

Program File: $F '200

External Subs: $F '217 ('219)

The Protect Codes are:

0 = Unprotect
1 = Protect

In general, if a record will only be read, then the record should not be protected. If the record will be read, changed, and rewritten, and the file is opened in shared ("S") mode, then the record should be protected when it is accessed and unprotected when it is rewritten.

A record is returned in Q1$() regardless of whether or not it is protected. The application program must check the return code in Q0 where appropriate.

If a record is read using a Protect Code = 1, Q0 = 0 if the record was successfully protected. Otherwise, Q0 = the number of the terminal using the record.

If File Number is zero, the ISAM operation is performed but Q1$0 is undefined. If Z0 > 0, then Q$ contains the key found by' 202.

Access Codes

A FICE-oriented and an English-oriented set of Access Codes are supported.

Pointer Positioning Access Codes

'EQ' or'= '	Read Equals
'GT' or ' > '	Read Greater Than
'GE' or ' > ='	Read Greater Than or Equal
'LT' or '< '	Read Less Than
'LE' or ' < ='	Read Less Than or Equal

Relative Pointer Access Codes

'RN' or 'R+'	Read Next
'RP' or 'R-'	Read Previous

Rey is only necessary for pointer positioning operations.

Attempts to access a non-ISAM key path generate a FICE+ system error.

For further information, read the Record Management System - ISAM section.

'203 - Write Bytes

'203 saves a portion of the record buffer QI$0 at a specified place on a specified disk.

Parameters: File Number [N]
Address [N]
Offset [N]
Length [N]
Protect Code [N] 0

Example: GOSUB '203(Z,Z(1)+Z(2)+1,(Z(4)-1)*ZO+1,Z(4),O)

Returns: None

Program File: $F '200

External Subs: None

Assumptions: U0(File Number) is a legal device table entry (a record file does not have to be open to use this subroutine). Referenced sector(s) can be written to. Length is between 1 and V*64 (the size of Q1$()).

This subroutine is primarily for internal use by FICE+. The number of bytes specified by Length is saved in sector Address starting at byte Offset. If Offset > 256, adjustments are made to Address. The bytes saved can span sectors.

The Protect Code is ignored.

'204 - Read Bytes

'204 loads date from a speed place on a specified disk into the record buffer Q1$Q.

Parameters:	File Number	[N)
	Address	[N]
	Offset [N]	
	Length	[N]
	Protect Code	[N] 0

Example: GOSUB '204(Z,Z(1)+Z(2)+1,(Z(4)-1)*ZO+1,Z(4),0)

Returns: None

Program File: $F ‘200

External Subs: None

Assumptions: U0(File Number) is a legal device table entry (a record file does not have to be open to use this subroutine).
Referenced sector(s) exist.
Length is between 1 and V*64 (the size of Q1$()).

This subroutine is primarily for internal use by FICE+: Length bytes are read from the disk starting at byte Offset in sector Address. If Offset > 256, adjustments ' are made to Address. The loaded byte string can span sectors.

The Protect Code is ignored.

'205 - Change Key/Delete Status

'205 sets or clears the key/delete record status byte in Z$(), the file control block.

Parameters:	File Number [N] Code (N] 0, 1, 2
Example:	GOSUB '205(1,0)
Returns:	None
Program File: $F '200	
External Subs:	None

Every record in a FICE+ file contains a key/delete status byte that can be used to query whether or not a record can be deleted and whether or not its primary key can be changed. In particular, the file maintenance subroutines inspect this status byte before permitting either of these operations.

This record status byte is not changed directly. When records are added or resaved, their key/delete status bytes are set equal to the value of the key/delete status byte in Z$(). '205 sets this value in Z$(). Records must be written to disk to be updated with this value.

Note: When a record is read /'rom disk, the key/delete status byte in Z$() is automatically updated just as if' '205 was called using the value of the status byte in the record. This ensures that records that are read, updated, and rewritten have the same key/delete status they started with.

If a specific key/delete value is required when a new record is added with a '215, '205 must be called immediately prior to the '215. Otherwise, the record's status byte will assume whatever value was left over in Z$().

To change the key/delete code in a record that already exists, '205 must be called after the record has been read, and before it is rewritten with a '213.

The Codes are:

0	= Record can be deleted, key can be changed
1	= Record cannot be deleted, key can be changed
2	= Record cannot be deleted, key cannot be changed

'206 - Get Key/Delete Status

'206 retrieves the key/delete status of the last record read or written.

Parameters:	File Number	[N]
Example:	GOSUB ' 206 (1)	
Returns:	Q = Key/Delete Status	
Program File:	$F '200	
External Subs:	None	

'206 retrieves the value of the key/delete status byte in Z$0. This byte is set by '205 or whenever a record is read with a Protect Code of 0 or 1.

The Key/Delete Codes are:

0 = Record can be deleted, key can be changed
1 = Record cannot be deleted, key can be changed
2 = Record cannot be deleted, key cannot be changed

For further information, read the discussion with '205.

'207 - Change Record Protection

'207 sets or clears the multiterminal record protection status of a record.

Parameters:	File Number	[N]
	Protect Code	[N] 0 or 1

Example: GOSUB '207(1,0)

Returns: If Q0 = 0, record was successfully protected or unprotected.
If Q0 > 0, record is currently protected by terminal Q0.

Program File: $F '200

External Subs: None

'207 performs a disk operation to set or clear the in-use flag for the Current Record. The Current Record is the last record read or written.

If the Current Record Pointer references a record that does not exist, '207 performs no operation.

'208 - Get Modules in System

'208 retrieves the first 128 unique Module Names that are part of the system.

Parameters:	None	
Example:	GOSUB ‘208	
Returns:	Q1$	= The first 32 Module Names
	Q$()	= The first 128 Module Names

Program File: $F '208

External Subs: None

Each two bytes of Q$ and Q$() contains a module name (for example, "ARAPGL"). The modules are retrieved in the order they occur in the system menu file, SMPM*. Duplicates are removed.

In Company Types 3 and 4, '208 will first attempt to locate the system menu file associated with the current Company a (Q3$) before using Company blank's system menu.

'209 - Exclusive Access

'209 changes system exclusive or company exclusive write status for a terminal.

Parameters:	Access Code	[A]
Example:	GOSUB ' 209 ("W")	

Returns:	Q$ = "Y", requested access granted.
	Q$ = "N", access denied.	

Program File: $F '209

External Subs: None

'209 allows a terminal to make critical updates to a company's files or to the system without interference from other terminals. The following four Access Codes are supported:

W = Company exclusive write. All files in the current company (Q3$) must be opened in a read-only mode or closed.

C = Clear company exclusive write.

WS = System exclusive. All other users must be logged off or at a menu.

CS = Clear system exclusive.

If an Access Code of W is passed for a company other than the log-on company (Q4$), that company remains write-protected until one of the following occurs:

- '209 is called with C from the same terminal in the same company
- The Close Files utility is run with Module and File Name set to "ALL»
- The terminal logs on under that company and returns to a menu

If WS is passed to '209, the system remains protected (no terminal, except terminal one, can leave a program menu) until one of the following events occurs:

- '209 is called and passed CS from the same terminal
- The Close Files utility is run with Company, Module, and File set to "ALL"

A WS parameter should always be passed before the system executes a critical function, such as a disk reorganization procedure. When a system is protected, terminals are not allowed to execute a task. The exception is terminal 1, which may proceed anyway, but which is warned that a critical operation is in progress. This allows terminal 1 to have access to utilities, such as the Close Files utility, in case of an inadvertent system lockout.

'210 - One-Level File Maintenance

'210 perform generalized file maintenance on a one-level file.

Parameters:	Module Name	[A]
	File Name	[A)
	Access Class	[A)

Example: GOSUB '210("GL","CHI","S")

Returns: None

Program File: $F '220

External Subs: $F '180
$F '200
$F '215
$F '217
$F '226

Refer to the sections on File Maintenance in this Guide and in the Release 2.2 Technical Reference Manual for detailed discussions on this subroutine.

'210 is designed to perform file maintenance on a single-level or a first-level file. One-level file maintenance on second-level files is not supported.

'211 - Open FICE+ File (No Return Code)

'211 opens (or closes) a FICE+ file and its associated path files. If the file does not exist, a FICE+ System Error is generated.

Parameters:	File Number	[N]
	Module	[A]
	File Name	[A]
	Access Class	[A]

Example: GOSUB '211(I,"AR","CU1","R")

Returns:	Q0$	= Name of file opened.
	Q1	= Start sector of file
	Q2	= End sector of file
	U(File#)	= Disk Unit
	Z$(File#)	= File parameters

Program File: $F '200

External Subs: None

The file Access Classes are:

X =	Exclusive
W =	Exclusive write (shared read)
S =	Shared write
P =	Write protect
R =	Read
C =	Close

'212 - Set Logical Record Pointer

'212 resets the logical record pointer to a specific record number.

Parameters:	File Number	[N]
	Record Number	[N]
Example:	GOSUB '212(1,Z0)	
Returns:	None	
Program File:	$F '200	
Requirments:	None	
Assumptions:	0 < = Record Number < = 999,999.	

'214 can be used to retrieve a record based on the logical record number set by '212.

'213 - Write Record

'213 saves the record buffer QI$0 on disk at the current record number.

Parameters:	File Number	[N]
	Protect Code	[N] 0 or 1

Example: GOSUB '213(1,0)

Returns: Z0 = 0 if current record number out of range
Z0 > 0 if record successfully resaved as record number Z0

Program File: $F ‘200

External Subs: None

'213 saves the record buffer at the current record pointer. In normal use, the record must first have been accessed using '200, '202, '214, '222, 224, or '239.

'213 should only be used to Resave records, never to add them. '213 does not automatically primary and alternate path files.

In most cases, a record should be saved with a Protect Code of zero to unprotect it.

'214 - Read Record

'214 loads the record buffer QI$Q from disk at the current record number.

Parameters:	File Number	[N]
	Protect Code	[N] 0 or 1

Example: GOSUB '214(1,0)

Returns: Z0 = 0 if current record number was out of range
Z0 > 0 if record Z0 successfully loaded

Q1$()= Record buffer

Q0 = Number of the terminal using the record
Q0 = 0 if no other terminal has protected the record

Program File: $F ‘200

External Subs: None

'200 is used to access a record by key. '214 is used only to retrieve the record corresponding to the current record pointer.

In general if a record will only be read, then the record should not be protected. If the record, will be read, changed, and rewritten, and the file is open in shared ("S») mode, then the record should be protected when it is accessed and unprotected when it is rewritten.

A record is returned in Q1$0 regardless of whether or not it is protected by another terminal. The application program must check the return code contained in Q0 where appropriate.

If a record is read with '214 using a Protect Code = 1, Q0 = 0 if the record was successfully protected. Otherwise Q0 = the number of the terminal using the record.

'215 - Add Record

'215 adds the record in the record buffer Q1$Q and updates the primary and alternate path files.

Parameters:	File Number	[N] < > 0
Key	[A]	
Protect Code	[N] 0 or 1	

Example: GOSUB '215(1,Q1$(),0)

Returns: If Z0 > 0, Z0 = the Record Number of the record just added.

If Z0 = 0, a duplicate key was encountered on a path file where duplicate keys are not allowed. The record was not added.

If Z1 = 0, the primary key caused the error.
If Z1 > 0, alternate key ZI caused the error.

If Z0 = -1, the record file or a key path file is full. The record was not added.

If Z1 = -1, the record file filled up.
If Z1 = 0, the primary key path file caused the error.
If Z1 > 0, alternate key ZI's path file caused the error.

Program File: $F '216

External Subs: $F '200
$F '217

If File Number is positive, the primary and alternate keys are extracted from the record buffer directly. Consequently, the Key parameter has no meaning other than documentation.

If File Number is negative, it instructs '215 to construct a pseudo-deleted record using Key and add it to ABS(File Number). All alternate key processing is ignored.

A full key path file situation arises when an ISAM key must be added to a key bucket that is full, and there are no more key buckets that can be allocated. However, the record that caused the overflow can never be added until the key file is rebuilt or the key density is reduced.

Z0 can also be set to -1 if a hashed path file fills up. However, this cannot occur under normal circumstances.

'216 - Delete Record

'216 deletes the record specified by the primary key and all alternate keys associated with it.

Parameters:	File Number	[N] < > 0
	Primary Key	[A]

Example: GOSUB '216(1,X$)

Returns: If Z0 > 0, Z0 = logical record number of the record deleted.
If Z0 = 0, the Primary Key was not found.

Program File: $F ‘216

External Subs: $F '200
$F '217

Assumptions: The file is open

If the primary key exists, the record is retrieved and alternate keys are deleted based on the contents of the fields in the record. If an alternate key cannot be found, deletion for that path is ignored.

The proper duplicate alternate key is deleted in an ISAM key path.

If File Number is negative, the record specified by Key is pseudo-deleted in file ABS(File Number). All alternate keys are deleted, but the primary key remains on file and the record is changed to HEX(FFFE) format.

'217 - Add Key

'217 adds a key to a path file.

Parameters: File Number [N]
Path Number [N]
Duplicate Code [N] 0 or 1
Key

Example: GOSUB '217(1,1,1,X5)

Returns: If Z0 > 0, the key was added pointing to logical record Z0.
If Z0 = 0, Key was a duplicate where duplicates are not allowed. If Z0 =-1, the path file is full.

Program File: $F '217

External Subs: None

If the Duplicates Code = 1, keys must be unique; if 0, duplicates are allowed.

Keys are added to Path Number along with the record number specified by the record pointer in STR(Z$(File Number)"3). If the add succeeds, Z0 equals the record number pointed to by File Number's record pointer.

Duplicate keys may be added to hashed path files and Type 'I' ISAM path files (no duplicates). This is not recommended as it is impossible to delete a specific duplicate except by initializing and rebuilding the entire file.

The "file full" return code (Z0 = -1) is supported for both hashed and ISAM path files.

'218 - Delete Key

'218 deletes a key from s path file.

Parameters:	Path Number	[N]
	Key	[A]

Example: GOSUB '218(1.2,X$)

Returns: Z0 = the logical record number associated with the deleted key
If Z0 = 0, the key was not found

Program File: $F '217

External Subs: None

In order to delete a duplicate ISAM key, the record number attached to the key in the path file must be the same as Path Number's current record pointer in STR(Z$(Path Number)"3). If not already set by another subroutine, it must be installed using '212 or by installing the value into the appropriate location in Z$(). The record number is crucial in determining which key in a set of duplicates is the proper one to delete.

'219 - Find Key

'219 searches a path file for a key.

Parameters:	Path Number	[N)
	Key	[A)

Example: GOSUB '219(1.3,X5)

Returns: Z0 = 0, the key was not found
Z0 > 0, then the key exists and Z0 is its associated record number

Program File: $F '217
($F '219 in a future release)

External Subs: None

When searching ISAM paths, an exact key match is required, otherwise Z0=0.

'220 · Two-Level File Maintenance

'220 performs generalized, two-level file maintenance.

Parameters:	Module Name [A] File Name [A] Access Class [A]
Example:	GOSUB '220("AR","1V1","S")
Returns:	None
Program File:	$F '220
External Subs:	$F ‘180 $F '200 $F '215 $F '217 $F '226

Refer to the sections on File Maintenance in this Guide and the Release 2.2 Technical Reference Manual for detailed discussions on this subroutine.

'221 · Physical Sequential Access [Setup]

'221 sets up a file for physical sequential access.

Parameters:	File Number	[N]
	Record Number	[N]

Example: GOSUB ' 221 (1 , 1)

Returns: None

Program File: $F ‘221

External Subs: $F '200

Assumptions: 0< = Record Number < = 999,999

Only one file at a time can be read physically sequentially.

A negative File Number forces all records to be read, even those that are deleted.

If Record Number = 0 or 1, the file is read from the beginning. Otherwise, the file is read beginning at the specified Record Number.

'222 - Physical Sequential Access [Read]

'222 accesses the next physically sequential record.

Parameters: File Number [N]
Protect Code [N] 0 or 1

Example: GOSUB '222 (1, 0)

Returns: Q1$() = Record, Z0 = Record Number
Q1$() = ALL(FF) if no more records exist

Q0 = Number of the terminal using the record
Q0 = 0 if no other terminal has protected the record

Program File: $F '221

External Subs: $F '200

Assumptions: '221 has been called to set up the file.

'222 performs two types of record reads. A fast buffered read is used if the file is opened in a read-only Access Class ® or P) and records are read unprotected. '214 is used if the file is opened in a write Access Class (S, W, or X) by any foreground or background terminal or if records are read with a Protect Code of 1.

If records are being added or deleted from the file while it is being sequentially° read, or ff records other than the current record are being changed, the buffer used by '222 may not reflect the true contents of the file. Consequently, buffering in these instances should not be used. (This situation can only occur if File Access Class Enforcement is set to "N".)

'223 · Key Sequential Access [Setup]

'223 sets up a file for sequential access by key using a range of key values.

Parameters:	Path Number	[N] < > 0
Low Range	[A]	
High Range	[A]	

Example:	GOSUB '223(1,1,GS," ")
Returns:	None
Program File:	$F ‘223
External Subs:	$F ‘200

Assumptions: If not blank, Low Range < = High Range

If a Low Range is specified, keys are returned that are > = Low Range. If a High Range is specified, keys are returned that are < = High Range.

If either Range is blank, special processing is invoked:
  If Low Range is blank, it is set to ALL(00) (low values).
  If both Ranges are blank, High Range is set to ALL(FF) (high values).
  If High Range is blank and Low Range is not blank, High Range is assumed to be the same as Low Range. Keys are retrieved that are > = Low Range and < = Low Range.

In summary, the four combinations are:

Low	High	Keys Accessed
Blank	Blank	Returns all keys.
Blank	Value	Returns all keys < = High.
Value	Blank	Returns all keys containing Low.
Value	Value	Returns all keys between Low and High.

If Path Number designates a bashed key path and Path Number is negative, hashed keys are not produced in key sequential order. For further information, refer to the discussion in the Release 2.3 Enhancement Guide.

Because keys from the disk are buffered in memory, '223 and '224 will not necessarily reflect any changes to the file made by your program or actions at another terminal. For this reason, it is recommended that programs which use '223 open the record file with a Access Class of "P".

If records are deleted from a file while this same file is being read key sequentially, a deleted record may be returned signifying the premature end of the sequence of records requested by ‘223.

'224 - Key Sequential Access (Read)

'224 accesses the next sequential record by key.

Parameters:	File Number [N] Protect Code [N] 0 or 1
Example:	GOSUB ' 224 (1,1)
Returns:	Q1$0 = Record Q1$(1,1) = HEX(FF) if no more records exist Q0 = Number of the terminal using the record Q0 = 0 if no other terminal has protected the record U2$ = Key Z0 = Record number associated with U2$
Program File:	$F '223
External Subs:	$F ‘200 $F ‘217 ('219)
Assumptions:	File has been set up with a call to '223.

If File Number is zero, only keys are returned in U2$.

For further information, refer to the discussion in the Release 2.3 Enhancements Guide.

'225 - Read Next Second-Level Record

'225 accesses the next sequential line item in a two-level file.

Parameters:	File 1	[N] File Number of first-level file
	File 2	[N] File Number of second-level file
	Protect Code	[N] 0 or 1
	Key	[A] key to File 1

Example: GOSUB '225(1,2,0,A$)

Returns:	Q1$() =	Record
	Q1$(1,1) =	HEX(FF) if no more records exist

Q0 = Number of the terminal using the record
Q0 = 0 if no other terminal has protected the record

Program File: $F '225

External Subs: $F ‘200
$F ‘217 ('219)

Line items can be read sequentially for only one pair of files at a time. All line items for a given first-level key must be read before they can be read for another first-level key. To interrupt reading for a given set of line items and then restart '225 for another (or the same) first-level key, U9(1) should be set to 0 before '225 is called for the new key.

When '225 has finished (STR(Q1$Q"2)=HEX(FFFF)), U9(1) is set to 0. Thus, if '225 is called again for the same first-level key after it has finished, the process is restarted as if it were running for the first time.

Second-level records should only be pseudo-deleted while the associated first-level record exists to prevent "orphaning" second-level records. '225 ignores orphaned second-level records. Only when the first-level record is deleted can second-level records be deleted.

The easiest way to pseudo-delete a record is by calling '216 with a negative File Number. Refer to the discussion on '216.

If a program deletes all second-level records associated with a given first-level record, the call to '225 must pass a negative number as the first parameter and use the key that begins in byte 3 of Q1$0. For further information, see the section on Sequential Access in the FICE+ 2.2 Development Guide.

See the description of ‘200 for an explanation of protection codes.

'226 - Store and Display Field

'226 is used in conjunction with file maintenance programs that call '210 or '220. During file maintenance, field values must occasionally be calculated based on input for other fields. '226 stores and displays the determined values for input fields.

'226 can also emulate most of the functions of the $F '250 subroutine.

Parameters:	Field Name	[A]
Numeric	[N)	
Alpha	[A)	

Examples: GOSUB '226("BALANCE",X*X0," ")
GOSUB '226("DUE DATE",0,X1$)

Returns: None

Program File: $F '226

External Subs: $F '220

Assumptions: '226 is called from a special case exit subroutine ('185).
Field Name exists on the currently displayed screen.

'226 is used in conjunction with '185 (post-input special-case exit). After a field is entered, the value(s) for certain other field(s) can be determined and displayed.

'226 writes this new value into the record buffer (T1$0) for the specified field. However, if this field is not defined as a "save as default" field in the definition file. the value is overwritten with its default value before user input.

If Field Name is blank, '226 performs similarly to the $F '250 subroutine. The Alpha and Numeric arguments become the two arguments normally passed to $F '250. The -3 parameter (erase Help) is not supported. Refer to '250 for further information.

'227 - Skip Fields on Input

'227 is used in conjunction with file maintenance programs that call '210 or '220. During file maintenance, '227 skips fields on initial input based on input for other fields.

Parameters:	Sequences	[N] > = 0
Example:	GOSUB ' 227 (1)	
Returns:	None	
Program File:	$F ‘226	
External Subs:	$F ‘220	

'227 is used with special case exits in file maintenance to conditionally skip field entry for a defined number of screen sequences.

If used with '184 (pre-input), Sequences must be zero. This instructs File maintenance to bypass the current field.

If used ,with '185, Sequences must be greater than 0, indicating the number of screen sequences to skip a/ter the sequence that has just been processed.

'227 cannot skip a field that is defined as a "required" field in the definition file and that has a blank default value.

'230 - Sort

'230 is a general-purpose sort subroutine.

Parameters:	Key Length	[N] 1 to 61
Example:	GOSUB '230(5)	
Returns:	None	

Program File: $F '230

External Subs: None

Assumptions: A SYSSRT work file has been opened.

'230 calls user-written subroutine '251. This subroutine must return the key to sort in Q$(1) and a 3-byte record locator in Q$. '251 returns each time it retrieves a record to sort. '251 sets Q1$(1,1) to HEX(FF) when no records remain to sort. For example:

```
8200   DEFFN '251
       : GOSUB '222(1,0)     : REM Load next record
       : IF Q1$(1,1)=HEX(FF) THEN RETURN
       : GOSUB ‘123("U")     : REM Unpack
       : Q$(1)=STR(A1$)&A2$          : REM Build sort key
       : PACK(######)Q$ FROM Z0           : REM Record position
       : RETURN
```

'230 is used when special tests must be made on records; otherwise, $F SORT should be used. '230 can also be used for non-FICE+ files.

A negative sort key length disables the Cancel key during the sort and merge.

A sort is cancelled by setting QI$(1,1)=HEX(F0) in '251.

When the sort is complete, the application program must load $F MERGE. The name of the program to load after the merge is passed in Q0$. For example:

```
8100    GOSUB '230(10)
       :Q0$="AR RPT 2"
       :LOAD T"$F MERGE"
```

'231 - Open File for Sorting

'231 opens a FICE+ file for sorting.

Parameters:	File Number	[N]
	Module	[A]
	File Name	[A]
	Access Class	[A]
	Program Name	[A] Program to load following the sort

Example: GOSUB '231(1,"AR","CU1","P","AR CUL 2")

Returns: None

Program File: $F '231

External Subs: $F '200
$F '217 ('219)

'231 opens both the record file and the definition file at the given file number. After '231 returns, the definition file (but not the record file) remains open at the given file number so that it can be accessed by '232. The record file is not opened again until $F SORT is executed. It remains open after the sort has finished.

If a negative file number is passed to '231, the percent complete display on the sort/merge/report screen is suppressed, and the Cancel key is disabled during the sort and merge.

'232 - Establish Sort Sequence

'232 establishes sort sequence and other sort parameters.

Parameters:	Sequence	[N] 0 through 7
	Alias	[A] Displayable version of Field Name
	Field Name	[A] Definition file field name
	Order	[A] + or -
	Top of Page	[A] Y, N, or*
	Minimum	[A] Low range check
	Maximum	[A] High range check
	Logic	[A] A or O

Example: GOSUB '232(1,"SALESMAN"," ","+","Y"," "," ","A")

Returns: None

Program File: $F '231

External Subs: $F '200
$F ‘217 ('219)

Assumptions: '231 has been called to open files for sorting.

If Field Name is blank, Alias is assumed to specify the name of the field.

Refer to the discussion in the Release 2.2 Technical Reference Guide for more information on this subroutine.

'233 · Edit Sort Sequence

'233 display a sort sequence and allow the user to change the sort sequence and range checks.

Parameters:	None
Example:	GOSUB ' 233
Returns:	None

Program File: $F '233

External Subs:	$F '180
	$F '231

Assumptions: Files have been opened for sorting by '231.

'234 - Store Additional Report Information

'234 stores up to 24 characters of additional information for report printing. This information is stored as part of the report queue when the report is added to the queue.

Parameters:	Data	[A] Up to 24 characters of information
Example:	G05U8 '234(Q$)	
Returns:	None	
Program File:	$F '234	
External Subs:	None	

'235 - Select Print Options

'235 allows the user to print a report immediately, select specific records by primary key, or add the report to a report queue.

Parameters:	Option Code	[N] -4, -3, -2, -1, 0, 1, 2, 3
Example:	GOSUB '235(0)	
Returns:	None	

Program File: $F '235

External Subs:$F '180 $F '200
$F ‘217 ('219)

The following option codes can be passed to ‘235:

0; All options permitted
1; Report cannot be printed now
2: Specific records cannot be selected
3: Report cannot be added to report queue -1: Suppress options; print report now
-2: Suppress options; print specific records -3: Suppress options; add report to terminal queue
-4: Suppress options; add report to background queue

'236 - Retrieve Additional Report Information

'236 retrieves the information stored by 234.

Parameters: None

Example: GOSUB '236

Returns: Q$ = 24 characters of information.

Program File: $F '236

External Subs: None

‘237 Open Sort Work File

‘237 sets up the current SYSSRT sort work file for access by ‘238.

Parameters : None

Example :	GOSUB ‘237
returns :	None
Program File :	$F ‘237
External Subs :	None

‘237 is used only for special situations involving ‘230. Normally, sorts are performed using the FICE+ program $F SORT and subroutine ‘239.

'238 - Retrieve Sorted Record Numbers

'238 retrieves the record numbers of sorted records.

Parameters:	Sequence	[N] Position in sort sequence or 0
Example:	GOSUB '238(0)	

Returns:	Q$ = 3-byte packed decimal record number Q1$(1,1) = HEX(FF) when end-of file is reached
Program File:	$F ‘237
External Subs:	None
Assumptions:	'237 was called to set up the SYSSRT file.

'238 can only be used to get the location of records that were sorted using '230. '238 is used during the call to '239.

If 0 is passed to '238, the location of the next sequential sorted record is retrieved.

'239 - Generalized Report Printing

'239 is used to print reports or perform specialized updates after a sort.

Parameters:	Report Name [A]
Example:	GOSUB '239("Customer List")

Returns:	Exits to $F MENU

Program File: $F '239

External Subs:	$F ‘194 or $F ‘196
	$F '200
	$F '237

'239 calls the following user-written subroutines to perform the indicated functions:

'250	Unpack record
'251	Print page heading
'252(Q1,Q2)	Print subheadings
'253	Print detail lines
'254(Q1,Q2)	Print subtotals
'255	Print grand totals

Q1 is a number from 1 to 7 that indicates the current level of the sort. It corresponds to the order of fields as seen on the left side of the sort select screen ('233). Thus, it is the order of the sort as modified by the operator.

Q2 is also a number from 1 to 7. It corresponds to the order of fields as seen on the right side of the sort selection screen. This corresponds to the order in which fields were identified in calls to '232.

'253 can pass up to 28 numbers to be totaled in R(). Subtotals and grand totals are passed to '254 and '255 in R(). All totaling is done in memory.

If no report name is passed to '239, the printer is not selected. This allows '239 to be used for purposes other than report printing.

'240 - Generalized Inquiry

'240 is used for writing standardized inquiry programs.

Parameters:	Module Name	[A]
	File Name	[A]
	Key File	[A]
	Prompt	[A]
	Screen Pages	[N]

Printer Pages	[N]
Form Feed Type	[A]
Rows	[N]
Columns	[N]

Example: GOSUB '240("AR","CU1","1","Alpha Sort",1,1," ",-1,80)

Returns: None

Program File: $F ‘240

External Subs: $F '180
$F '190
$F '194
$F '200
$F '217 ('219)

'240 calls the following user-written subroutines to perform the indicated functions:

'250	Unpack record
'251	Print page heading
'252(Q)	Printer output
'253(Q)	CRT output

Q, the parameter passed to '252 and '253, indicates the page number for multi-screen inquiry.

If Key File is blank, '240 prompts for the primary key by field name. For multipart keys, each field is prompted for in turn.

If Key File contains the letter "A" through "Z", the letter is assumed to identify an external alternate key file. Prompt is used to prompt the use for key input, and the record is retrieved via the external alternate key file.

If Key File contains a "0", '240 will ask the user to enter the primary key to the Tile. If the primary key is a multi-part key, the user will be asked to enter the key value as a single field. Prompt will be used to prompt the user. If prompt is blank, the Primary Key Name field in the definition file parameters will be used.

'240 (Cont'd)

If Key File contains the digits "1" through "8", the user will be prompted to enter an internal alternate key corresponding to that number. The Prompt parameter will be used to request a key from the operator. If Prompt is blank, the alternate key's field name will be used.

'240 supports the Scan Key. However, accesses cannot be made to "referenced" files. For more information, refer to the discussion on the Scan Key elsewhere in this document.

'240 uses file number 1 for the record file, file number 2 for external alternate key files, and file number 3 for the definition file.

If the number of hard copy pages is 0, the user is not given the option to print a hard copy. The last three parameters are passed to '190 when the hard copy print option is selected (see the description of '190).

$F '250 capabilities are available when using '240. Instead of calling '250, use '226 instead. For instance, the call GOSUB '226(Q$,0) centers the contents of Q$ on the current line of the CRT. The -3 Field Length parameter is not supported. For further information, read the discussion on $F '250.

'241 - Create Record File

'241 creates and initializes a record file and all associated path files on disk based on information in the definition file. This subroutine can also be used to calculate disk requirements for a record file without actually creating the file.

Parameters:	Disk Unit	[N] Disk on which to create the record file
	Module Name [A]	
	File Name	[A]
	File Size	[A] T,A,O,C
	Records	[N] Override if File Size = "O'

	Start Key	[N]
	Hash Loop	[N]

Example: GOSUB '241(1,"AR","DV1","0",10,1,0)

Returns: If File Size = "C", Q= Sectors required to create the file.
If File Size < > "C"

Q = 1 if the file is properly created
Q = 0 if the file is not created due to any of four causes indicated in the variable Q3

Q3 = 1, definition file not processed.
Q3 = 2, file index is full.
Q3 = 3, data file already exists
Q3 = 4, disk is full.

Program File: $F '241

External Subs: $F '10 (loaded automatically)
$F '180
$F '200

Files Sizes: A = Proposed record file size in definition file.
T = Test records in definition file
O = Override. Records is used.
C = Calculate file size.

The starting key is significant only if the file has been assigned a sequential, numeric key.

The hash loop counter indicates the number of times the key will cycle through the hashing algorithm. For short keys (12 bytes or less), a hash loop of zero is usually the most effective. For longer keys, use a hash loop of 3.

'242 - Initialize Record File

'242 initializes a record file and deletes all existing records in the file.

Parameters:	File Number	[N]
	Start Key	[N]

Example: GOSUB '242(1,1)

Returns: None

Program File: $F ‘242 .

External Subs: None

The starting key is significant only if the file uses assigned sequential, numeric keys. If-1 is passed to ‘242 as the starting key, the next key is not reset.

'243 - Set File Status Bytes

'243 sets a file status byte to a specified value.

Parameters:	File Number	[N]
	Status Byte	[N] 0 to 64
	Value	[N] 0 to 255

Example: GOSUB '243(1,1,1]

Returns: None

Program File: $F '243

External Subs: None

If Status Byte is 0, all 64 status bytes are set to Value. Value is translated to a one-byte hexadecimal value before it is stored.

'244 - Retrieve File Status Bytes

'244 retrieves file status bytes for inspection.

Parameters:	File Number [N)
Example:	GOSUB ' 244 (1)
Returns:	Q$ = 64 file status bytes

Program File: $F '244

External Subs: None

Q$ contains up to 64 hexadecimal values stored in the file by the file maintenance subroutines and '243.

'245 - Get Next Sequential Numeric Key

'245 retrieves the next available sequential numeric key. These keys are assigned to records added to system-maintained files in which the key consists entirely of a single numeric field (digits-only).

Parameters:	File Number [N]
Example:	GOSUB '245(1)
Returns:	Q$ = Key
Program File:	$F '245
External Subs:	None

The numeric key is incremented when a record is added to the file, except when the key length is greater than 12 bytes. In this case, the key cannot be incremented, and '245 cannot be used.

'246 - Rebuild Path Files

'246 rebuilds the path files for a record file or inspects the current hashing distribution.

Parameters:	File Number	[N]
	Hash Loop	[N]

Example: GOSUB ' 246 (1,1)

Returns: Q = Percentage of overflows (0-100)

Program File: $F '246

External Subs: $F '217

If Hash Loop > = 0, that hash loop is used when rebuilding any hashed paths in the file.

If Hash Loop = -1, the percentage of hashed bucket overflows is returned. If the file contains more than one hashed path, the results are combined.

If Hash Loop = -2, the hash loop value stored with the file is used.

If File Number < 0, '251 is called to display the percent complete, and '252 is called if a duplicate key is found with the relative record number passed as a parameter. The variable Q$ contains the key. On returning from '252, if Q$="Y", the duplicate key is allowed; if Q$="N", the duplicate key is deleted. '251 and '252 must be part of the application program if the file number is negative.

Prior to Release 3.0, $F '246 automatically deleted any records containing a duplicate key. This function was undocumented and potentially dangerous, and has been removed.

'247 - Add File to File Index

'247 adds a file and its platter location to a file index.

Parameters:	Module Name	[A]
	File Name	[A]
	Platter	[N] 1 though U7

Example: GOSUB '247("OE","001",7)

Returns: Q = 0 if the file index is full

Program File: $F '247

External Subs: None

Files created using '241 or $U are automatically added to the file index.

'248 - Delete File from File Index

'248 deletes a file from a file index.

Parameters:	Module Name [A] File Name [A]
Example:	GOSUB '248("AR","XX1")
Returns:	None
Program File:	$F '248
External Subs:	None

If a file is deleted from the disk index, the file should also be deleted from the file index.

'249 - Search File Index for File

'249 searches a file index and returns the platter location of the file.

Parameters:	Module Name	[A]
	File Name	[A]
Example:	GOSUB '249("IC","PR1")	
Returns:	Q = Disk platter location where the file resides	
	Q = 0 if the file is not in the file index	
Program File:	$F '249	
External Subs:	None	

'250 - Center Message

'250 centers text on the CRT.

Parameters:	Message	[A]
	Field Length	[N]

Examples:

```
GOSUB ‘250("Print Checks",-2)
GOSUB ‘250("Enter Check Number",6)
```

Returns: Q = Column position to be used for the first byte of field input in later calls to '181 or ' 182 (if Field Length is greater than 0)

If Field Length > 0, Message is centered on line 2 of the CRT with allowance for keyboard input of a field the size of Field Length.

If Field Length = 0, Message is centered on the current line (where the cursor is).

If Field Length = -1, Message is underscored before centering on the current line.

If Field Length = -2, the screen is cleared and a standard CRT heading is produced with

Message centered in asterisks on Line 0. If Message is blank, Line 0 contains all asterisks.

If Field Length = -3, any displayed "Help" message is cleared.

‘400 (File #, Field Name)

‘400 file# field name (variable z31, z32, z33, z34, z32$, z31$, z30$, z30$(), z31$(), z32$())
no de ligne 3498-3502

Description : Get value ‘of specified field name from record buffer of file # specified.

‘400 is included in $F ‘200 library

Parameter : GOSUB ‘400(file#, field name)
File# : a numeric value corresponding to the open file number to be read.
Field Name : The alpha-numeric name of the field to read.

Retour : Q= Numeric value if field is type 1
Q$= Alpha value if field type is <> 1 If field type is «DATE», ‘188 must be used an Q$ to unpack Date.

Exemple ; GOSUB ‘121 (3",AR»,»CUI»,»R»)
GOSUB ‘200 (3,3,0,»1234) : Rem get customer «1234"
GOSUB ‘400 (3,»CUST NAME») : Rem get customer name and put in Q$

400 suite

Note : Record must be last record read prior to call ‘400. ‘400 is mostly used to unpacked only those fields required by the application program, avoiding the use of the $P/$U routine, thus avoiding potential variable conflicts especially between modules.

Section 15

Technical Information

FICE+ 3.01 System File Layouts

System Parameters (SYSTEM)

Sector U (start of file + 13) has changed significantly in this release, but other portions of the file remain the same as release 2.3.

The chart on the following page shows the layout of sector U and the variables commonly used for each system parameter where appropriate. Fields marked with an asterisk are new or have changed since 2.3.

Sector U: System Parameters

Bvtes	Type	Pack	Variable	Description
001-001*	BIN	A001	U1	CRT rows (was BCD)
002-002*	BIN	A001	U2	CRT columns (was BCD)
003-003*	BIN	A001	U3	Company usage type (was BCD)
004-004	BIN	A001	U4	Maximum number of terminals
005-005	BCD	6001	V	Record buffer size = V * 64
006-006	BCD	6001	VO	Nbr of open files (Z$())
007-007	BCD	6001	U6	Disk unit for company files
008-008	BCD	6001	--	Nbr of disk units
009-O10	BCD	6002	V1	'223 buffer size = V1 * 256
011-012	BCD	6002	V2	'230 buffers; memory used = V2 * 160
013-014*	BCD	6002	V3	Bucket Size = V3 * 64 (was Merge Buff)
015-016	BCD	6002	--	Default printer rows
017-018	BCD	6002	--	Default printer columns
019-019	ALF	A001	U$(09)	Percent complete allowed (Y/N)
020-020	ALF	A001	--	Default form feed type (A/L/P)
021-021	ALF	A001	U$(11)	Subroutine load check (Y/N/E)
022-022*	ALF	A001	U$(16)	Open files as needed [not supported)
023-023	ALF	A001	U$(19)	12-hour time (Y/N) (was U9$)
024-024	ALF	A001	U$(10)	Help usage type
025-025	BIN	A001	V4	Nbr of save-as-default fields
026-026	ALF	A001	U$(O6)	Background allowed (Y/N)

027-027*	BIN	A001	U${18)	SPACEK for big prgms (not supported)
028-028*	BCD	6001	V5	Reserved (Was nbr of input forms)
029-030*	BCD	6002	V6	Field attributes buffer (Was BIN)
031-032	BCD	6202	U1(5)	Release ofFICE+(##.## or ####)
033-048	RSV	A010	--	Reserved for applications programmers
049-056	ALF	A008	U8$	Mask for null dates
057-059	ALF	A003	U7$	Date format map
060-061*	BCD	6001	--	Number of $C companies (not supported)
062-064*	ALF	A003	--	Last Post ID Executed
065-065*	BIN	A001	V7	Max Key files to Open
066-066*	B1N	A001	U7	Maximum device table slots
067-067*	BCD	6001	U8	Date Rol1 Over
068-070*	ALF	A003	--	Applications System ID
071-071*	ALF	A001	U$(12)	Y/N Enforce Accesses?
072-072*	BIN	A001	U$(24)	Disk Poll Rate [seconds)
073-073*	BIN	A001	U$(25)	Clock Poll Rate ($BREAK = value " 3)
074-192	RSY	0077	--	Reserved
193-232	ALF	A00B	U1$()	Common Program Names
233-256	RSV	0018	--	Reserved

3.01 Terminal Record

Sectors U+ 1 thru U+U4 (SOF+ 13+UO) - Loaded BA in Q$(4)64
(Default values are listed first.)

0S(1)	Bvtes	Typ	Variable	Description
	001-001	BIN	U1	Terminal CRT Rows
	002-002	BIN	U2	Terminal CRT Cols
	003-008	RSV		Reserved for terminal (nulls)
	009-009	ALF	U$(22)	Mosaic Graphics? (blank/1)
	010-010	ALF	US(23)	Box graphics [blank/1/2)
	011-016	RSV		Rerserved for terminal (blanks)
	017-017	ALF	U$(19)	12 Hour Time? (blank/Y/N)
	018-022	RSV		Miscellaneous attributes (nulls)
	023-023	BIN	US(20)	System Menu #
	024-024	BIN	U$(21)	Module Menu #
	025-026	ALF	Q2$	Module ID
	027-029	ALF	Q1$	Date
	030-032	ALF	Q3$,Q4$	COID
	033-048	ALF	U6$	User ID
	049-064	ALF		Password (or ALL(FF))
0$(2)	065-128	ALF		Pending Message
0$(3)	129-129	BIN		Default Terminal Printer Rows

	130-130	BIN		Default Terminal Printer Cols
	131-131	BIN		Last Printer Number
	132-132	BIN	U$(13)	Foreground Report queue
	133-133	BIN	U$(14)	Background Report Queue
	134-136	RSV		Reserved (nulls)
	137-137	ALF		Last Disposition
	138-138	ALF		Terminal Printer? (N/Y)
	139-144	RSV		Reserved (blanks)
	145-192	ALF		Subheading
0${4)	193-208	BIN		16 Security Classes (nulls)
	209-216	RSV		8 Spare (nulls)
	217-224	RSV		8 Spare (blanks)
	225-240	ALF		Apps area ($F MENU sets to nulls)
	241-256	ALF		Apps area (FICE+ never changes)

U$0

The system variable U$() has been expanded to 64 bytes.

U$(01) = Language
- M=FICE
- P=PCFICE (Unsupported)

U$(02) = Terminal Attachment
- A=Attached
- D=Detached (background)

U$(03) = 0/S Release (#.#)

U$(04) = Message Waiting
- HEX(00) = None
- HEX(O1) = Message
- HEX(02) = Message displayed

U$(05) = Background Print Control Flag
- HEX(01) = Abort (X or A)
- HEX(02) = Halt (H)
- HEX(03) = Continue [C)
- HEX(04) = Restart (R)
- HEX(05) = Pause at Top-of-Page (P)

U$(06) = B/G Allowed? (Y/N)

U$(07) = Sort File Already Selected & In Use? (Y,N,null)

U$(OB) · Reserved (was partition protect status)

U$(09) = % Complete Allowed? (Y/N)

U$(10) = Help Usage (Y/N/E)
U$(11) = Subroutine Load Check (Y/N/E)
U$(12) = Enforce Access Classes? (Y/N)
U$(13) = Foreground Queue
U$(14) = Background Queue
U$(15) = CPUs MUXed (Y/N)
U$(16) = Reserved (was 'Open Files during input?') (null)
U$(17) = Reserved (was 'Reselect DT') (null)
U$(18) = Reserved (was Minimum SPACEK for '>' programs) (null)
U$(19) = 12 Hour Time? (Y/N) (Replaces U9$)
U$(20) = Last System Menu # (initialized to null)
U$(21) = Last Module Menu ;Y (initialized to null)
U$(22) = Mosaic Character Set (1=Y, blank = no)
U$(23) = Box graphics (1=Fasstcom, 0=character, blank=none)
U$(24) = Disk poll rate
U$(25) = Clock poll rate
U$(26)
to
U$(fi4) = reserved (initialized to null)

Device Table (SYSCFG)

The device table file has been restructured to handle 128 devices in two configurations.

Each configuration consists of two sectors, each sector containing 64 4-byte elements.

Bvtes	Description	Pack
001-003	Address	A003
004-004	Device Type	A001

The first element is reserved for the SELECT DISK device, device #0. The second element is for device table slot #1, the next for slot #2, etc. The device type is reserved for future use and is initialized to blank.

File Indexes

File indexes now support 128 file names. This was done by redefining sector 2 of the file, which used to contain the access classes in use in Release 1.12. Sector 2 now has the same structure as sector 1. Each sector stores the name and disk unit. number of 64 files in 64 4-byte elements.

Sector 3 is initialized to ALL(00).

FICE+ 3.01 Record File Layout

Sector SOF (Start of File): File Header Record

The file header record ia used to atore parameters pertaining to the entire file.

The following fields are carried over from previous releases.

Bytes	Description	Pack	Variable
001-003	Next Available Record Number	6003	
004-006	Number of Records Used	6003	
007-009	Highest Record Number Ever Used	6003	
010-012	Total Key Sectors (Buckets + Headers)	6003	Z(2)
013-015	Maximum Number of Records	6003	Z(3)
016-018	Record Length	6003	Z(4)
019-021	Primary Key Length	6003	Z(5)
022-024	Hash Loop Counter	6003	Z(6)

The following fields sre new.

Bytes	Description	Pack	Variable
025-025	Type of Structure (HEX(30) = 3.0)	6001	
026-026	Highest Numbered Alternate Key Path	6001	
027-029	Primary Key Header Offset (Always I)	6003	
030-032	Alt Key #1 Header Offset	6003	
033-035	A1t Key #2 Header Offset	6003	
036-038	Alt Key #3 Header Offset	6003	
039-041	Alt Key #4 Header Offset	6003	

042-044	A1t Key #5 Header Offset	6003
045-047	Alt Key #6 Header Offset	6003
048-050	Alt Key #7 Header Offset	6003
051-053	Alt Key #8 Header Offset	6003
054-056	Date Created	A003
057-059	Date Restructured	A003
060-064	Reserved (nulls)	0005

The reorganization number at byte 64 has been eliminated.

The following fields are carried over from previous releases.

Bytes	Description	Pack	Variable
065-128	Next Key (ADD Mode)	A040	
129-192	Status Bytes	A040	
193-256	Last Key Added	A040	

Sector SOF + Header Offset: Path File Header Record

The path file header contains parameters that describe the path file. The path file was introduced in 3.0.

Bytes	Description	Pack	Variable
001-003	Next Available Bucket Number	6003	
004-006	Number of Buckets Used	6003	
007-009	Highest Bucket Number Ever Used	6003	
010-012	Path File Size (Sectors)	6003	Z0(02)
013-015	Maximum Number of Buckets	6003	Z0(03)
016-018	Bucket Size (Bytes)	6003	Z0(04)
019-021	Key Length	6003	Z0(05)
022-024	Hash Loop	6003	Z0(06)
025-027	Root Bucket Number	6003	Z0(07)
028-030	Key Start Byte in Record	6003	Z0(08)
031-033	Highest B' Tree Level in File	6003	Z0(09)
034-036	Maximum Keys per Bucket	6003	Z0(10)
037-039	Minimum Keys per Bucket	6003	Z0(11)
040-042	Key Density (~)	6003	Z0(12)
043-043	Mode (H=Hashed,I=ISAM,D=ISAM w/Dupes)	A001	
044-046	Date Created	A003	
047-049	Date Reorganized/Rehashed	A003	

Key Buckets

Key buckets are contained in the path files and are used to hold keys. Hashed kev buckets are always 1 sector in size. ISAM key buckets can be from 1 to 8 sectors in size.

ISAM Kev Bucket

Bytes	Description	Pack	Variable
001-nnn	Keys	Axxx	
EOB -11	Level	6003	Z4
EOB -8	Keys in This Bucket	6003	Z5
EOB -5	Previous Bucket Number	6003	Z2(1)
EOB -2	Next Bucket Number	6003	Z2(2)

Hashed Kev Bucket

Bytes	Description	Pack	Variable
001-255	Keys	AOFF	
256-256	Overflow Flag [HEx(FF/00)J	A001	

FICE+ 3.01 File Control Blocks

Record File Control Block

The elements of the record file control block, Z$(), contain the current parameters of each open file. Each element is 54 bytes in length. The defmition of the first 25 bytes is the same as in previous releases.

Byte	Var	Format	Description
001-003	Z0	6003	Current Record Number
004-006	Z(1)	6003	Start Sector
007-009	Z(2)	6003	Total Key Sectors (Key Buckets + Headers)
010-012	Z(3)	6003	Maximum Possible Records in the File
013-015	Z(4)	6003	Recard Length
016-018	Z(5)	6003	Primary Key Length
019-021	Z{6)	6003	Primary Key Hash Loop Counter
022-022		A001	Key Change/Delete Protection Code
023-025		6003	Last Record Number Protected
026-028		6003	Record File End Sector
029-031		6003	File Index Start Sector
032-034		6003	SYSCO Start Sector
035-037		A003	Filename
038-039		A002	Module Name
040-042		A003	Company ID
043-043		A001	W=Read/Write,R=Read Only, ' '=None
044-044		A001	Access status BIN(POS("PCRSWX"=Access))
045-045		A001	Primary Key Slot

046-046	A001	A1t Key #1 Slot
047-047	A001	A1t Key #2 Slot
048-048	A001	A1t Key #3 Slot
049-049	A001	Alt Key #4 Slot
050-050	A001	Alt Key #5 Slot
051-051	A001	Alt Key #6 Slot
052-052	A001	A1t Key #7 Slot
053-053	A001	Alt Key #8 Slot
054-054	6001	Version of FICE (HEX(00) or HEX(30))

Path File Control Block

The virtual path file control block, Z0$(), is new with this release. It stores the parameters for each open path file.

Byte	Var	Format	Description
001-003	Z0	6003	Current Bucket Number
004-006	Z0(01)	6003	Header Sector for This Key
007-009	Z0(02)	6003	Size of Key File (sectors)
010-012	Z0(03)	6003	Maximum Key Buckets
013-015	Z0(04)	6003	Bucket Length (bytes)
016-018	Z0(05)	6003	Key Length
019-021	Z0(06)	6003	Hashed: Hash Loop Counter ISAM: Bucket Key Number
022-024	Z0(07)	6003	ISAM Root Bucket Number
025-027	Z0(08)	6003	Key Start Byte
028-030	Z0(09)	6003	Highest ISAM Tree Level
031-033	Z0(10)	6003	Maximum Keys per Bucket
034-036	Z0(11)	6003	Minimum Keys per Bucket
037-039	Z0(12)	6003	ISAM Packing Density (%)
040-040		A001	Mode (H/I/D/S)

FICE+ 3.01 Definition Files

Parameters

The defmition file parameters, stored in sector EOF-1, have changed significantly from Release 2.3. The parameters may be unpacked from Q$Q into the specified variables by passing a "UP" parameter to $F ' 10. The parameters are only accessible in BA mode.

Byte	Descriotion	Variable	Field Spec
001-001	Version of FICE+	S4(01)	6001
002-002	Primary Key Fields	S4(02)	6001
003-003	Primary Key Fields Created	S4(03)	6001
004-004	Default Key Type	S4(04)	6001
005-005	Displays Part I	S4(05)	6001
006-006	Displays Part II	S4(06)	6001
007-007	Primary Key Length	S4(07)	6001
008-008	Last Primary Key Sequence number	S4(08)	6001
009-009	Create Disk Unit number	S4(09)	6001
010-010	Maximum Screens	S4(10) 6001	
011-011	Bucket Size - Primary	S4(11)	6001
012-012	Bucket Size - Alt Key #1	S4(12)	6001
013-013	Bucket Size - Alt Key #2	S4(13)	6001
014-014	Bucket Size - Alt Key #3	S4(14)	6001
015-015	Bucket Size - Alt Key #4	S4(15)	6001

016-016	Bucket Size - Alt Key #5	S4(16)	6001
017-017	Bucket Size - Alt Key #6	S4(17)	6001
018-018	Bucket Size - Alt Key #7	S4(18)	6001
019-019	Bucket Size - Alt Key #8	S4(19)	6001
020-020	Key Density	S4(20)	6001
021-021	ID Field Length	S4(21)	6001
022-022	spare	S4(22) 6001	
023-024	Record Length	S5(1)	6002
025-026	DATE ADDED Start Byte	S5(2)	6002
027-028	DATE CHANGED Start Byte	S5(3)	6002
029-030	Subroutine Number	S5(4)	6002

Defintion File Parameters (Cont'd)

Byte	Description	Variable	Field Spec
031-032	Record Length Created	S5(5)	6002
033-034	ID Field Start Byte	S5(6)	6002
035-037	Proposed Records	S6(1)	6003
038-040	Test Records	S6(2)	6003
041-041	Primary Key Mode	S4$(01)	A001
042-042	Primary Key Mode Created	S4$(02)	A001
043-043	Alt Key 1 Mode S4$(03)	A001	
044-044	Alt Key 2 Mode S4$(04)	A001	
045-045	Alt Key 3 Mode S4$(05)	A001	
046-046	Alt Key 4 Mode S45(06)	A001	
047-047	Alt Key 5 Mode S4$(07)	A001	
048-048	Alt Key 6 Mode S4$(08)	A001	
049-049	Alt Key 7 Mode 54$(09)	A001	
050-050	Alt Key 8 Mode S4$(10)	A001	
051-051	Open Files As Needed?	S4$(11)	A001
052-052	Process Status 54$(12)	A001	
053-053	Restructure Status (I/R/' ')	S4$(13)	A001
054-054	Minimum CRT Rows (binary)	S4$(14)	A001
055-055	Minimum CRT Cols (binary)	S4$(15)	A001
056-056	spare	54$(16)	A001
057-057	spare	S4$(17)	A001
058-060	Date Processed S1$	A003	
061-072	Alt Key 1 Field Name	S5$(1)	A00C
073-084	Alt Key 2 Field Name	S5$(2)	A00C

085-096	Alt Key 3 Field Name	S55(3)	A00C
097-108	Alt Key 4 Field Name	S5$(4)	A00C
109-120	Alt Key 5 Field Name	S5$(5)	A00C
121-132	Alt Key 6 Field Name	55$(6)	A00C
133-144	Alt Key 7 Field Name	S5$(7)	A00C
145-156	Alt Key 8 Field Name	S5$(8)	A00C
157-168	ID Field Name	S5$(9)	A00C
169-192	Key Name	S6$(1)	A018
193-216	Record Name	S6$(2)	A018
217-256	Descriptive File Name	S0$	A028

Field Definitions

A new set of variables has been defined for definition file field records. Field records can be unpacked from Q1$0 into the listed variables below by passing a 'U' parameter to $F '10.

After the definition file is processed, field records can be accessed using the standard FICE+ keyed access subroutines. The key to a definition file field record is the Field Name.

Example: GOSUB '200 (1, 1, 0, "DESCRIPTION")

Byte	Description	Variable	Field Spec
001-012	Field Name	S$(1)	A00C
013-024	'Key To' Field Name	S$(2)	A00C
025-036	'Get From' Field Name	S$(3)	A00C
037-048	Redefinition Start Field	S$(4)	A00C
049-06D	Redefinition End Field	S$(5)	A00C
061-061	Field Type (0-5)	S0(1)	5001
062-062	Field Type Created	S0(2)	5001
063-063	Alternate Key number	S0(3)	5001
064-064	Alternate Key number Created	S0(4)	5001
065-065	spare	S0(5)	5001
066-067	Length (bytes)	S1(1)	5002
068-069	Length Created	S1(2)	5002
070-071	Special Case Number	S1(3)	5002
072-073	'Key To' File number	S1(4)	5102
074-075	'Get From' File number	S1{5)	5102

076-076	Decimals Left/Rows	S2(1)	6001
077-D77	Decimals Right/Cols	S2(2)	6001
078-078	Redefinition Start Byte	S2(3)	6001
079-079	Redefinition End Byte	S2(4)	6001
080-080	spare	S2(5)	6001
081-081	spare	S2(6)	6001
082-083	Start Byte	S3(1)	6002
084-085	Start Byte Created	S3(2)	6002

Field Definitions (Cont'd)

Byte	Descriotion	Variable	Field Spec
86-087	'Key To' Start Byte	S3(3)	6002
088-089	'Get From' Start Byte	S3(4)	6002
090-091	spare	S3(5)	6002
092-092	Save?/Redefine (Y/N/R)	S0$(1)	A001
093-093	Enter? (Y/N)	S0$(2)	A001
094-094	Save as Next Default? (Y/N)	S0$(3)	A001
095-095	Required/Optional	S0f(4)	A001
096-096	Left/Right Justify	S0$[S)	A001
097-097	'Key To' Code (Y/N/8/' ')	S0$(6)	A001
098-098	Alternate Key Mode Created	S0$(7)	A001
099-099	Cache? (Y/N)	S0$(8)	A001
100-100	spare	S0$[9)	A001
101-102	Field Spec	S1$(1)	A002
103-104	Field Spec Created	S1$(2)	A002
105-120	OK Input Characters	S2$[1)	A010
121-136	Heading	S2$(2)	A010
137-154	Variable Name	S2$	A012
155-173	Low Range	S3$(1)	A013
174-192	High Range	S3$(2)	A013

193-254	Default	S$	A03E
255-256	System Record Protect Bytes	none	

Definition File Screens

A definition file may have from zero to 99 screen records. Each 12-sector record starts at sector Q where Q = Z(1) + Z(2) + Z(3) + (Screen number - 1)' 12 + 1

Sectors Q to Q+3: 64 16-bvte Field Elements (Proarammer Entered)

001-001	CRT Row	6001
002-002	CRT Column	6001
003-003	Item number	6001
004-004	reserved	0001
005-016	Field Name	A00C

Sector Q+4: 64 4-Bvte Field Elements (Created bv 'Process')

001-001	CRT Row	6001
002-002	CRT Column	6001
003-003	Item number	6001
004-004	Field number	A001Binary

Sectors Q+5 through Q+11: Screen

Each screen consists of three fields: an identification field, a screen parameters field, and the screen image. The identification field and most of the parameters fields are reserved for future use.

Each screen is loaded "DA" as:
```
DIM XS64, X0$25, X$(20)80
DATA LOAD DA T #U(X), (Q+5) X$, X0$, X$()
```

Identification Field:

001-064	Reserved

Screen Parameters Field:

001-001	Minimum Rows	A001 (Binary)
002-002	Minimum Cols	A001 [Binary)
003-025	Reserved	

Screen Field:

Each element represent one line of text on the CRT. Line 20 is currently not used.

System Error Messages

The following FICE+System Error Messages were introduced in Releases 3.0 and 3.01:

Access Claes Violation <File Name> - File Improperly Opened

This error occurs when Acceas Class enforcement is turned on in the System Parameters, snd an attempt has been made to use a file in a manmer contrary to the Access Class under which it was opened. The most common instance is the attempt to write to a file that has been opened in a read mode.

Path Descriptor Block Too Small

An attempt has been made to open more path files simultaneously than has been allowed for in the System Parameters. Increase the value of the "Max Key Files Open at a TSme" parameter.

Illegal ISAM Subroutine Call. Path File Contains Hashed Keys

A call was made to '202 specifying a hashed path file.

< File Name > File is Damaged - Processing Incomplete

A parameter in one of the headers in a record file references information outside the bounds of the file. The program has been shut down to prevent damage to data elsewhere on the disk. The data file is unusable and must be rebuilt.

Path File Error in <File Name>

An unexpected result was encountered in a path file. The error can usually be corrected by rebuilding the path file with the Rebuild Path Files utility in the FICE+Utilities module.

Undefined Path Reference in <File Name >

An attempt was made to reference a path file that does not exist. Correct the program.

Undefined Error

This error is called when $F ERR 1 does not have enough information to display a valid error message.