IBM TECHNICAL INFORMATION EXCHANGE

International Business Machines Corporation

TIE 5-0057
February 17, 1965
34 pages

IO

IBM 1401 TAPE IOCS ADVANCED NOTES

Tom Scharf
IBM Corp.
D. P. Customer Education
Gladengvn. 3B, Etterstad
Oslo, Norway

Distributed by
DPD Program Information Department
IBM Corporation
112 East Post Road
White Plains, New York

TITLE : IBM 1401 TAPE IOCS ADVANCED NOTES

AUTHOR : TOM SCHARF

DATE JANUARY 12., 1965.

DIRECT INQUIRIES TO : TOM SCHARF
IBM D. P. CUSTOMER EDUCATION
GLADENGVN. 3B, ETTERSTAD,
OSLO, NORWAY.

ABSTRACT : THIS PAPER IS THE RESULT OF EXPERIENCE WITH THIS PARTICULAR SYSTEM SINCE ITS ANNOUNCEMENT. THE APPROACH IS TO TAKE EACH DTF - example WLRADDR - , EACH DIOCS STATEMENT AND EACH MACRO INSTRUCTION AND EXPLAIN IN DETAIL ITS IMPORTANCE IN THE TOTAL PROGRAM STRUCTURE. AN EXPLANATION IS GIVEN OF HOW THE COMPILER HANDLES EACH PARAMETER AND WHICH RESULTS EACH PARAMETER HAS, FOR EXAMPLE ON THE DTF TABLE OR THE DTF ROUTINE. COMMON ERRORS AND MISUNDERSTANDINGS ARE TAKEN UP IN LIGHT OF THESE FACTS AND CERTAIN EVALUATIONS ARE MADE AS TO THE ADVISABILITY OF USING OR NOT USING CERTAIN PARAMETERS. THE PAPER WAS ORIGINALLY WRITTEN AS AN INSTRUCTOR BACKGROUND FOR ORDINARY IOCS COURSES AND FOR DIRECT USE ON AN ADVANCED IOCS COURSE.

INTRODUCTION

These notes are designed to help programmers and others who desire a deeper knowledge of this IOCS system. This can then be used as the basis for self-study or for a short course in advanced IOCS.

A sound understanding of the facts contained in this paper will help programmers to avoid common errors and misunderstandings as well as to better be able to analyze errors correctly and more quickly when they occur.

The notes are based on the author's own experience with several different non-overlap systems and purposely exclude areas which the author does not have extensive experience in such as overlap and printer DTF's.

It is the authors conviction that senior programmers, systems engineers and others who have responsibility for solving special problems with and for teaching this system will profit greatly by a detailed study of the IOCS - it is, after all, only one program with some minor variations but it is used in virtually all programs of many installations and problems come up every day which can much more quickly be solved through a detailed knowledge of the system.

This principle is valid for most programming systems to-day but since no programmer "has time" (read "makes time" or "takes time") then they lose time and money in the long run.

In preparation at present writing is an english translation of a table constructed (in Norwegian) giving an explanation of virtually every single symbolic label found in non-overlap IOCS, DIOCS and DTF routines. This is of course a very useful analysis tool and has been in wide use for several years here in Norway.

TABLE OF CONTENTS

cont.

THE DTF ENTRIES

1. DTF File name

Should preferably contain the word FILE (example: INFILE, FILEAB) to avoid confusion when OPEN, GET, PUT and other macros are used.

Is used by IOCS as the symbolic label on the first instruction in the corresponding DTF routine.

2. FILETYPE

For TAPE files. INPUT or OUTPUT must be specified since these parameters are decisive for correct compilation.
An error here ruins the whole DTF.
This information will also be reflected on the DTF table at FILE-NAME-2 by a one-position code which is tested by DIOCS routines.
R=tape input, W=tape output, 1= reader, 4= punch, 2=printer.

3. CHANDRIVE

Any drive number from 0 to 9 is valid.
Two files may have the same number if they are not OPEN at the same time.

The result of the entry here is that it is placed on the DTF table at FILENAME-3 (primarily for flip-flop use) and in the model I/O instruction on the four-line table at ¤1K00N.
These two locations can easily be patched to modify drive number.

4. ALTTAPE

Any digit 0-9 is valid.
Only result is that this entry is placed in the DTF table at location FILENAME-4 which is normally blank.
This location is tested at end of reel (1EOR on trailer) and swapped with the position at FILENAME-3 (which is used to determine drive number.

5. RECFORM

If neither VARIABLE or FIXED is specified then FIXED will be assumed.

If neither BLOCKED nor UNBLOCKED are specified then UN-BLOCKED is assumed.
i.e. This line is theoretically unnecessary for fixed unblocked files but is very strongly recommended as fully filled out as this strengthens documentation greatly and eliminates error possibilities. It costs virtually nothing!
This line is only used for tape and is extremely important - an error here is "fatal" - it ruins the whole DTF because major variations in the DTF routine as based on this line.

6. SIZEREC

Note the purpose of this entry is to give the IOCS routines information on the size of the data record - NB including possible record marks.
For fixed blocked records it is used for

a) increasing an internal accumulator (¤2Q00N) which starts at 000 for the first data record and increases by the record length as given in SIZEREC for each GET or PUT until the block is ended as signaled by a BCE test for a groupmark in the next potential data record. A two-address SBR instruction is used to increase the count.

b) Ensuring correct padding-record length.
For fixed unblocked files this parameter is used to check record length (if WLRADDR).
For other files the author is not aware of any use IOCS makes of this constant.
Errors during the first GET or during the CLOSE process are often caused when SIZEREC does not correspond to the DA used as IOAREA and the DA group mark is therefore not positioned "correctly" causing loops stopped only by core boundary.

7. BLOCKSIZE

Only for blocked files; total IOAREA DA length not including groupmark desired.
Used by IOCS:

a) For fixed blocked records is used to calculate the values of

Π2Q and Π4Q operands, these are not instructions. Π2Q is actually an internal accumulator used to keep track of which data record in the block is the current one and Π2Q is used to update a possible INDEXREG between each GET or PUT. Π4Q is a constant used to reset Π2Q after a WRL condition so that the next GET will read a new block. The operand value of the initial states of Π2Q and Π4Q are determined by setting them to (BLOCKSIZE - SIZEREC) and letting the autocoder assembler calculate this difference. The reason for this is that the first GET after OPEN will thus be forced to read in a new tape block when Π2Q is like this.
At EOF Π2Q will stand at "BLOCKSIZE" (so will INDEXREG if one is assigned) thus it is necessary to reset Π2Q with the aid of a RELSE if the file is to be re-opened!

b) For variable blocked output with VARBUILD this parameter is used to determine potential block size irrespective of actual DA size.

c) For all blocked files this parameter is used to check record length (WLRADDR must be specified for this check).

8. IOAREAS

This is the area which will serve as input or output directly from tape.
This parameter is a typical substitution type parameter which is placed directly into several instruction operands in the DTF routine any valid address form will do but any indexing used will be negated (+ X 0).
This is then usually a DA address as it is the first position of the input area which is needed. The DA must always have a group mark - on most file forms, this is a vital prerequisite to proper IOCS functioning.
The group mark is used to stop I/O operations but also to test for a full block (F/B) and stop move-record instructions when work areas are used (F/U).
It is possible for a careful programmer to patch - change this parameter if it is forgotten or in error.

9. TYPELABEL and
10. CHECKLABEL

The result of this entry is restricted to a one position code on the DTF table at FILENAME-5 and it is combined with the result of the CHECKLABEL entry to produce the code which is tested by the general label routine in DIOCS (moved then for DIOCS use to IOCPSV-4) to determine the extent of label reading, writing and checking necessary.

The codes are:

1	Standard - check all from input or check old output header.
A ($\overset{+}{1}$)	Standard - no checking of any part header or trailer.
J ($\overset{-}{1}$)	Standard-check IDENT and block count (on trailer).
Blank	No labels.
2	Non-standard.

A very important point to be noted here is that this code cannot be changed at will to any of the others unless the correct routines are present in the DIOCS.
For complete code flexibility it would be necessary to specify MIXED and CHECK in DIOCS LABELDEF. For example it is not possible to specify STANDARD in DIOCS and then change this code in the DTF table to blank in order to eliminate label reading. This does not work because all DTF's are assumed (STANDARD in LABELDEF) to have standard labels thus no test is made at all for blank in this position. Similarly it is impossible to specify complete checking by changing A to 1 if LABELDEF only has IDENT specified. The routines simply don't exist!
However it is possible to specify " downward" for example no checking (A) instead of full checking (1) or no labels (bl) instead of standard (1, A or J) if MIXED is specified simply by patching.

11. EOFADDR

The use IOCS compiler makes of this is simply to create a 3 position address constant on the corresponding DTF table (at FILENAME-7).

This area on the DTF table is normally a three position blank for output files.

Any correction or change may be made to the three-position constant by patching. No other changes are necessary. A missing EOFADDR is thus very easy to correct without recompilation.
Any symbolic or actual address may b e used subject to the usual rules for unsigned DCW address constants.
Thus an actual address must be specified in 3-position code (999 or N43) indexing is possible.
This address constant is moved up to the general DIOCS routines to a B000 instruction at ₰2K001 (not IOCRCL , which determines if it is EOF or EOR before branding to EOFADDR).
An EOFADDR accidentally specified for an output file is disregarded and no recompilation is necessary.

12. WLRADDR

This line is alvays optional however experience has shown that it is absolutely necessary on all input files (except variable unblocked where no check is possible) due to the possibility among many other things of electronic malfunction.

The check is carried out by a compare against a calculated constant as compared to the B-register after the RT operation (=GM+1, stored in IOCTBR).

Any group marks brought into the input area are automatically erased before branch to the WLRADDR.

The IOCS is conditioned to read a completely new block when the next GET order is given.
Aside from the obvious and very common WLR reason - incorrectly generated test data, WLR can come because of the following:

a) Very large "noise" records (over 12 pos.) generated during backspace rewrite and skip and blank tape routines because of imperfect positioning of R/W heads.

(cont.)

b) The 1401 internally can skip a character cycle or take an extra cycle during tape reading or writing thus extending or contracting the record.

c) Wrong tape mounted or wrong tape on wrong drive.

d) A previous program has inadvertantly written wrong length records due to a groupmark word mark accidentally appearing in the output area (esp. when using common I/O areas).

The standard WLR routine which is generated individually for each DTF routine and costs about 38 positions in core, tests ~~in principle~~ whether or not the record length is exactly correct or not ~~but in fact because (in the commonly used move mode) the groupmark with wordmark at the end of the IOAREA stops "too-long" records from being read in past the "correct length" only "too-short" records are normally tested for. However this has proved to be a very serious shortcoming.~~
~~It is possible to check for too-long tape records by a little additional programming.~~ An extra groupmark wordmark must be placed back each IOAREA DA groupmark:

```
IOAREA   DA      2x100,₰
GM       DA      ~~0x0,G~~ |x|, G-
         ~~and the GET coding modified.~~
~~01~~       ~~CW~~      ~~GM-1~~      ~~Allow too-long records~~
~~02~~       ~~GET~~     ~~INFILE~~
~~03~~       ~~MLCWA~~   ~~GM,GM-1~~   ~~Restore GM for end of~~
                                  ~~block test and move~~
                                  ~~record inst.~~
```

Since the IOCS will automatically clear the GM at GM when a too-long WLR occurs we must restore this in our WLR error routine:

```
MLCWA    @₹@, GM
MLCWA    @₹@, GM-1   (chaining & half-chaining is possible
                      here)
```

If the WLR routine is to be general then this GM restore could

(cont.)

be carried out before the CW GM-1 instruction above in a subroutine created for the GET. If this is even more generally desired then the address of the group marks can be found by using the fact that the 3-position field IOCTBR contains the address of the IRG-created groupmark plus one. If this is a too-long record then IOCTBR contains GM+1, however, if it is a too-short record then any other address in the IOAREA is possible, so be careful. A special symbol such as (12-4-8) placed after the GM could be used as a basis for establishing which case (too-long or too-short) had occurred. This information would enable the general routine to restore the GM if necessary and also give valuable information about the type of WLR which could be printed or typed or used to determine what type of recovery procedures are worth trying.

What kind of a WLR routine should be written? IBM leaves this up to the user.

a) A macro can be written for this purpose to standardize the routines.

b) An early version of IOCS branched to WLRADDR from the BU instruction thus it was possible to generalize for all WLR and return to the IOCS as for a convertional SBR subroutine. This is changed and SBR use will not work now.

c) A new GET will read in a new block; however a good WLR routine could try to backspace 10 or more times before giving up (the tape unit will be found in all cases in location IOCTRW+3.

```
Thus:    MLNS    10CTRW+3, * + 4
         BSP     0
```

can be used in a general routine.

d) In any case at least a halt and preferably a clear message giving tape unit (IOCTRW+3!) should be given before proceeding to a new GET.

e) Some attempt to identify the block should be made so that an investigation of the output can help determine if the record really was a "noise record". It will not be possible to get this information from a workarea (not filled) or an indexed IOAREA

(not correctly indexed at this point). The best bet is to, for example, take the first 80 absolute (non-indexed+x0) positions of the IOAREA and print or punch them. The IOAREA can be picked up for all cases at IOCTRW+6 for use in a general subroutine.

13. WORKAREA

The conception behind this entry is that one and only one workarea is used by an I/O FILE so why not let IOCS do the moving. The inclusion of this entry produces an MRCM (MCM) instruction in the DTF routine which simply moves a data record between IOAREA and WORKAREA. We must provide the recordmarks for this for blocked records.
INDEXREG cannot be used together with this for the same DTF because INDEXREG says "I want to process in the IOAREA without IOCS moving the record to a workarea." An area parameter in a GET for a file where WORKAREA is specified will not work for essentially the same reason. It will not be treated as an error - it will just simply not have any effect on data moving because no DTF routine coding has been included (as it normally will be when WORKAREA, INDEXREG, VARBUILD are absent.) to test the GET instruction to see if any record-moving is specified.

If desired, patching of the operand of the MCM instruction in the DTF routine can safely be carried out in order to correct the workarea specification.

14. INDEXREG

Is only available for fixed blocked records and is most commonly used on input.
An invalid specification here has far-reaching effects on the DTF routine and a recompile is the best thing here.
WORKAREA, VARBUILD or a GET-PUT area specification may not be used at the same time as INDEXREG for reasons noted under "WORKAREA", however records handled in this manner can be easily moved by the programmer down to a workarea. This is commonly

done to save processing time by avoiding the move for records not to be extensively processed.
Use of WORKAREA is the simplest alternative (and the safest), however we are forced to consider INDEXREG use because it saves processing time when total references to the indexed area are so few that the additional time needed for all indexing calculations (1401 hardware) pr. record does not exceed the time needed to move the record. This factor can easily result in for example $\frac{1}{2}$ hour saved daily for a file updating run which takes 3 or 4 hours a day.
Note especially that variable blocked records cannot be handled on the 1401 tape IOCS by INDEXREG they must be moved to work area. The reason for this might have something to do with the lack of MRCWM (move record with wordmarks) instruction as on 1410/7010.

If any reader has an application where substantial time savings will result from being able to index V/B records in input then I can console him that it is possible to "cheat" by specifying variable/unblocked to IOCS, letting each GET get a whole block and deblocking by relatively simple manual coding.
The author has constructed and sucessfully used a macro "NOGET" expressly for this purpose. Remember one detail for variable unblocked records IOCS will change the IRG-generated GM to a record mark during the GET, in case you try to test that to determine last data record.

15. VARBUILD

This is straightforwardly described in the manual. The usual error is using X1 which is not allowed as this is internally used simultaneously by IOCS during PUT (but is of course otherwise freely available to the user).
The VARBUILD rythm is

a) Give the size you want the processor to allocate for the next record to the varbuild operand.

b) PUT ,FILE. The IOCS now determines if the previously put records must be written out to make room for the new one or not. In any case the varbuild operand will then receive from IOCS the exact address (left-most) where the new record can be placed. The user can then move his record in using this information. For example


```
01    MLC   SIZE,X2          GIVE SIZE
02    PUT   ,UTFILE          ALLOCATE A PLACE
03    MRCM  WORKAR,0+X2      MOVE RECORD TO
                             ALLOCATED AREA
```

Use of an indexregister as VARBUILD operand saves one instruction in this example (but of course we assume the register is available).

Note that the varbuild operand (esp. an X-reg) may safely be used for any other purpose between these "put sequences".
Note that VARBUILD is in fact the only method of writing out variable/blocked records via this IOCS!

16. REWIND.

The only effect this entry has is on a one-position field on the DTF table just above the end of file address (at FILENAME-10).
There are three possible codes here:

Blank	= No rewind desired at beginning or end of reel
B	= Unload desired at end of reel, rewind at beginning
A	= Rewind desired at end and beginning of reel.

These codes are moved along with the whole DTF table to the joint DIOCS routines' area for DTF tables at IOCPSV each time this file has a beginning or end of reel situation. The rewind code resides at IOCPSV-9 which is EQU to IOCRWD and both labels are used in the DIOCS routines.
Blank and B are tested directly as needed while A is "assumed" if blank or B are not present, thus any other codes would give the "rewind" effect.
No codes will be tested at all if either UNLOAD or NORWD is not specified in DIOCS RWDOPTION. If at least one file has a NORWD entry then one of the two valid RWDOPTION entries must be entered otherwise no test will be made on this code and rewind will be assumed regardless of the DTF specification.

17. TOTALS

The result of one or two entries here is that for each count desired, a ten or 16 position accumulator will be provided at the upper end

of the DTF table immediately adjacent to the * which in fact is used as a signal to determine wehter or not there are more fields to be handled by the DIOCS routines after all other fields except hash and record fields are handled in the IOCENT routine which initializes a DTF table for use by the general DIOCS routines.

The field ¤7J00N is the field used for record counts. This can be referred to directly (f. ex. ¤7J002 for the first DTF) where knowledge of record count is desireable for example in order to limit file size on a reel to be sorted. No zone bits ever exist in this field.

¤7K00N is the label of the 16 position accumulator provided for hash totals although only the right hand 10 positions are used in the label and thus authomatic overflow is "lost".

In addition to the 10 position fields an A (add) instruction will be created at an appropriate place in the DTF routine to take these totals.

Note that a WM will be desireable to delimit hash fields less than ten positons. This field which must have a WM is in the workarea defined in the DTF if one is defined there; otherwise the wordmark must come in the IOAREA DA even if a workarea is named in the GET or PUT macro! The reason is logical; the DTF has no knowledge whatsoever of GET or PUT use.

It should be more obvious here why variable/blocked records must be read into a work area.

If the user neglects to make corresponding entries for HASH and RECORD in the DIOCS COUNTS fields then a specification in the DTF is useless for label procedures but can certainly be used for other purposes as mentioned above. The DIOCS has no knowledge of what HASH and RECORD totals are specified in the DTF's and vice-versa.

Note especially that hash totals in the DTF are not given by the word HASH as in COUNTS but are the relative position of the right-most position in the field to be hash-accumulated. This information is used as address adjustment in the add instruction only and can thus easily be corrected by patching.

18. HEADER

The only practical use of this entry is to give the 10 pos. header IDENT when an input file is only to be tested on IDENT and to give the IDENT and retention cycle desired for output files.

There is no practical sense in giving retention cycle and creation date for input files since these data are invariably provided via a RDLIN card when they are to be used.

Parameter 1 (the IDENT) will be placed on the DTF table immediately above the % constant which is the signal that label information follows. It is rather important that this parameter be always 10 positions since the constant will be generated incorrectly otherwise and ruin the IOCS processing.

The second parameter on output files will be placed in the three position field which is located three fields above the IDENT.

A picture is a good idea here.

Here is a typical DTF table:

```
LABEL USED
IN DIOCS    *          DTF TABLE FOR A TYPICAL FILE

                       DCW   @*@          END OF TABLE MARK
I0CHSH      ¤7K002     DCW   #16          FOR HASH TOTAL (only 10 lower
                                                 pos. used)
I0CRCT      ¤7J002     DCW   #10          FOR RECORD COUNT
I0CCRD                 DCW   #5           CREATION DATE
I0CRCY                 DCW   @075@        RETENTION CYCLE
I0CSEQ                 DCW   @001@        REEL NUMBER
I0CSER                 DCW   @20100@      FILE SERIAL NUMBER
I0CIDT                 DCW   @KUNDEFL20I@ FILE IDENT
                       DCW   @%@          BEGINNING-OF-LABEL-DATA MARK
            (Indication of which exits are used can come here)
I0CBLK                 DCW   &000000      BLOCK COUNT, LEFT HAND 5 POS.
I0CRWD                 DCW   @ B@         REWIND CODE (UNLOAD)
I0CE0F                 DCW   E0F          EOFADDR ADR CONSTANT
                       DCW   @  @         PADDING CHARACTER
I0CPSV-4               DCW   @ 1@         CHECK LABEL (ALL)
I0CPSV-3               DCW   @  @         POSSIBLE ALTERNATE TAPE UNIT
I0CPSV-2               DCW   @ 3@         MAIN TAPE UNIT
                       DCW   @ R@         INDICATES FILE TYPE R= tape input
                       DCW   @ M@         INDICATES MODE (MOVE OR LOAD)
```

The next position is the first in the DTF routine and it has the same label as the DTF FILENAME.

A study of this table is very important for an understanding of

IOCS. It should be obvious that the constants we give in the DTF go directly to this table in many cases and since the standard autocoder macro substitution system is used by IOCS also then an incorrect specification of the length of a field (for ex. 3 instead of 003 for retention cycle) would give a too-little constant and give undesired results at object time. To put it another way - if you get a funny looking label - check this table!
Parameter sequence is extremely important here - note especially the difference between input and output file specification!

19. REELSEQ

Generally this is rather useless - the author has never seen it used and cannot think of any good excuse to use it either.
The effect of course is an initial modification of the normally generated @001@ DCW on the label table.

20. SERIALNUM

Any five position field can be specified here but the user must determine his own system.
A useful system might be to always place the five position program -IDENT (col. 76-80 of autocoder cards) here so that tapes give an indication as to the program which produced them!

21. PADDING

The effect of the single position (usually "9") inserted here is simply to replace the normally blank position assigned as padding character on the DTF table.
This entry will most certainly not determine whether the padding routine will be carried out. Padding with whichever character is on the DTF table will always be carried out if necessary at CLOSE or FEORL on FIXED/BLOCKED OUTPUT FILES.
At that time this character is picked up from the table by the DTF padding routine at ¤ 3PooN.
The padding procedure is worth mentioning.
The padding character is moved by a loop in the padding routine from the next-to-last character in the record and one by one towards

the left end of the record then for all remaining records to be padded in the block an MCM (MRCM) is executed in order to "copy" this padding until the end of block is reached as indicated by the presence of a Groupmark (see routine ¤ 1N00N).

The most common failures at this point then are due to:

a) Error in SIZEREC or DA length specification.

b) No record mark at end of padding record; this usually occurs during testing of programs with so little test data that a full output block has not been created yet, this results in a process as the MCM destroys its own future A-field (esp. GM-WM) because the last data records lack record marks.

c) Missing Groupmark at end of DA.

Worthy of note is that present 1401 tape COBOL which uses IOCS will always pad with blanks unless the user via a patch to the DTF table (or ENTER AUTOCODER to change the table) changes the padding character.
Of course the user must always test for padding on input files which can contain padding.
A popular misconception is that to "save time" we can branch directly to the EOF routine. This must be forbidden as it is not compatible with IOCS because if we branch directly to the EOF routine after discovering a padding record:

a) Labels will never be checked (trailer),

b) Padding can come at the end of any block if FEORL or RELSE has been used in the program producing the tape - and if it does not use them now, it might at any future time be changed (for example to cut down reel size for sort input),

c) Padding will not necessarily come at all if the block is full!

The only correct programming is that every GET have an immediately following padding test which always branches to the GET again if we find padding. The use of a GET subroutine so that this is not "forgotten" is to be recommended.
Generally 9 should always be used for padding unless special circumstances dictate otherwise but remember - even if fixed/blocked files are used padding might not come at all.

Padding records should always be generated in test input for fixed blocked files!
Be careful to test a big enough field so that you can be absolutely sure that the record is padding. BCE testing is strongly discouraged. Remember that input data can be in error!

22. MODEPAR

Not normally used because MOVE mode which is most common is assumed when nothing is specified.
Result of this entry comes on the lowest position of the DTF table and in the model I/O instruction in the DTF routine under ¤1K00N table.

23. EXmADDR

Exits are at most installations never or hardly ever used although some will of course use them in every program because of special label procedures.
The author has very little practical experience with exits and would prefer therefore not to discuss them at length.
An interesting point here however can be mentioned.
If it is considered necessary to use exits then their effect can often be achieved by macro instructions or autocoder coding which ORG INTO IOCS routines and patch in exits symbolically. In one case the author constructed a macro to do just this so that labels were automatically displayed on a console typewriter for every open. Since a macro cannot refer to another macro's ¤ labels (and DIOCS is a macro) and no IOCXXX labels were where we needed them it was necessary to tack our own private label into an existing DIOCS instruction. This has succeeded very well and been in use for over two years but is certainly "dangerous" because we have no guarantee that the system will work after each new modification level! Of course the label must be added to existing model statements and we must never add or subtract from the model statements since each updating of IOCS is completely dependent on the expected sequence numbering of each model statement after the immediately preceding update.

The use of one or more of these statements in the DTF will produce eight different DCW constants, one for each potential exit, a * indicates the exit is not used for this file while lack of an * indicates that

a) the exit is to be activated when this file is handled,
b) instead of the *, the 3 position address which the exit is to branch to is included in its place.

These are included, if at all, just below the % sign on the DTF table.

Example: exit one (EX1ADDR) is specified as EXIT1.

```
DCW   @%@
DCW   EXIT1    address constant
DCW   @*@      indicates exit 2 not used
DCW   @*@
DCW   @*@
DCW   @*@
DCW   @*@
DCW   @*@
DCW   @*@      exit 8 is not used in this particular
               DTF.
```

No exit can be activated if it is not specified as being used in the DIOCS EXITS entry.

THE DIOCS ENTRIES

1. DIOCS — This word must come as the third (or 2nd if no JOB card) card of a compilation. That is, no intervening comments cards are allowed. Generally speaking no other cards except DIOCS, DTF, JOB and CTL should be used before the last DTF is finished.

2. DIOCSORG — The effect of this entry is simply to change the (333) normal ORG ahead of IOCS coding to the operand specified here. This feature is rarely used in actual practice.

3. IODEVICES — The point of this entry is to give information to the macro-processor that we expect to have DTF's in this program for the named devices (Tape, Printer etc.) and that therefore certain routines and instructions must be included in the DIOCS routines, especially to separate the tape files (with label handling) from unit record files.

   The extra coding generated by superfluous use of U/R parameter specification is minimal (a few BCE tests to avoid label routines). The use of the parameter TAPE is however extremely important as it is decisive for all tape model statements.

   It is a common misconception that we should specify U/R (Reader, Printer, Punch) when we have them on the 1401 or use them in the program but this is not at all the case. The deciding factor is - do we have DTF's for these devices - parameter sequence is quite irrelevant here.

4. TAPEUSE — The primary function of this entry is to reduce the DIOCS coding generated and to reduce the processing time for certain routines a little.

   There will be no adverse effects other than those mentioned above if this is not specified when only input or output files are present. However if this is incorrectly specified then a recompilation is demanded since this has far reaching effects on the routines generated.

   About 155 positions can be saved by using this entry.

5. FEATURES — The point here is that features such as overlap, release and print storage which the user desires used by I/O devices which are specified in one or more DTF's can be specified here. This will ensure that maximum use is made of these features.

5. cont. — It is never necessary to specify any of these features if the user does not want to. This can be desirable when no meaningful time savings can be achieved since I/O volume is low and the user prefers to save the core space which will be used by the additional routines.

   The overlap feature has a very great effect on the compilation of tape programs, being virtually a whole new IOCS, if this feature is never used (when overlap is not installed on a 1401) considerable compile time improvement can be made by cutting out all the model statements in the IOCS which are especially for overlap. This process is separately described in IOCS program library writeups and should normally be done.

   RELEASE should normally be specified for card reader or punch files of any volume (lets say several hundred cards or more) unless the slight additional core (over 100 pos. including a work area) required is more desperately needed elsewhere.

   Note that a card reader which is to use the release equipment will require that a work area be used. This can be recommended in any case as a great convenience since a DA with wordmarks cannot be written for positions 1 to 80. If release equipment is installed then, this can only be utilized effectively by IOCS specification.

   DIOCS routines are only affected slightly by these parameters in order to ensure "release" before tape I/O operations are performed. The main effect is that the respective DTF routines for U/R are increased to include coding. The "release" function will automatically be performed where necessary by all IOCS routines. However, user coding must also take this into account and before any lengthy operations are performed.

```
B     IOCRDX     (release the reader)
```

   and/or

```
B     IOCPNX     (release the punch)
```

   must be included. Failure to do this at the appropriate points will stop the machine and the card in question will fall in the normal pocket.

   The position of the I register will indicate the point at which the programmer must place the branch-to-release instruction (which, by the way) is an SBR-type subroutine).

6. LABELDEF — This important entry is one which new IOCS students have greatest difficulty understanding.

   It is only necessary if some tape files have any kind of labels.

   It determines which routines are to be included in the DIOCS for use as specified in the individual DTF or by the RDLIN macro.

6. cont. To simplify we can say that the following specification (we disregard TM as unimportant)

MIXED, CHECK, RDLIN

is the most general case. Any other specification limits the extent of the possible DTF specifications and is (like "Tapeuse") primarily an attempt to reduce the coding to be included in the DIOCS routine.

MIXED means that DTF files can have STANDARD, NON-STANDARD labels or none at all - and that the DTF table can at any time easily be patched to change that status.

If we instead of MIXED specify STANDARD then we have said that all tape files will always have standard labels and the user will find that it is very difficult indeed to patch IOCS so that a file may be temporarily read or written without labels.

The additional coding generated by MIXED is very modest being essentially some few BCE tests. So it is generally worthwhile to use this - it can for example simplify testing.

CHECK will allow all possible combinations of label checking as determined by the DTF table entry including no check at all.

Whereas IDENT eliminates the possibility that any file will be checked completely regardless of the DTF specification, whereas only about 50 positions would be saved.

Standard (or MIXED) and CHECK specification require about 1000 extra positions to give complete label handling, as opposed to not using lables at all or programming for them individually.

RDLIN specification will lead to the inclusion of a 174 position SBR-type subroutine called IOCRDL which is used exclusively by RDLIN macros. This fact is useful since the routine is easy to overlay after use to win core positions (ORG IOCRDL).

If CHECK is to be used in any DTF, then RDLIN is virtually manditory in order to change the creation date in the DTF table for correct comparison with the input header. Another consequence of RDLIN use is that RDLIN cards must be made for every file written with standard labels. It is most expedient to punch RDLIN cards automatically. At one installation the author constructed a macro "PURDL" which automatically patched IOCS and ensured RDLIN card production. Another device which is very simple is to utilize the fact that after OPEN on an input file, all the required information is in label form at the label I/O area named IOCSLB, which is accessable to the programmer.

For example:

```
OPEN    OUTFIL
SW      121
MLC     IOCSLB+39, 150
MLC     @RDLIN@          1/2 CHAIN
P
CS      180              OPTIONAL
SS      8                OPTIONAL
```

This is cheap and easy and a macro is readily constructed.

7. COUNTS These parameters will include the coding needed by the DIOCS routine to handle the count fields found in DTF's and to check and produce these counts on trailer labels. About 217 positions are required if both are specified and about 187 positions if only one is specified.

This entry does nothing but give the possibility for hash or record totals; the actual use they are put to depends entirely on the DTF specifications. If no DTF has counts then these routines only take up space but do no harm.

8. ALTDRIVE If included, this parameter includes 29 positions extra in the DIOCS routines which primarily test the alternate drive unit space on the DTF table (FILENAME-4) for a file at end of reel and shift the drive number (if it isn't blank) with the drive indicated in the main drive position (FILENAME-3), and ensure that the I/O instruction receives this new drive number.

If the altdrive space on the DTF table is blank, then this will have no effect. In other words, if altdrive has been specified in the DIOCS, then flip-flop unit changing can be entirely controlled by patching or programming changes to the single position on the DTF table (for example by control card or sense switch choice).

9. EXITS When specified, routines for testing the DTF table just below the % sign for indication of which exits are desired, are included as well as coding at the exit points in the DIOCS label routines in order to allow branch to the addresses given for each particular exit on the DTF table.

Core requirement for this specification is 70 positions (DTF table test routine) plus 9 extra positions for each exit named. These 9 are used for the exit itself. In addition the following number of positions for the exit as indicated must be included:

Exits	1-2-4-5-7-8	11 positions
	3	39 positions
	6	25 positions

for example exits 1 and 3 require altogether 138 positions.

10. RWDOPTION If UNLOAD is included, then extra coding will be included so that any DTF specifying UNLOAD on the DTF table will be tested and unload will be performed instead of the normal rewind. NORWD option will also be tested.

10. cont. If NORWD is specified, then tests will be made for NORWD option in the DTF table but otherwise, rewind will be assumed. UNLOAD is not taken into consideration by IOCS.

UNLOAD requires 40 additional positions while NORWIND (or NORWD it makes no difference since only the first three letters are tested) requires only 16 additional positions.

11. READERROR This entry describes what procedures are to be carried out when a "permanent" tape error is discovered. This means that the normal tape error routines have given up hope. If nothing is specified, BYPASS is assumed. That is, the record (the whole tape record, not just a data record) will be bypassed without any processing and a new record will be read before control is returned to the user.

Before this stage is reached, the IOCS HALT 3050 will appear and the operator can press start to continue trying. If after ten further attempts reading is not successful, then the read-error options (if any) will be executed.

If CLEAN is inserted as it should be if space allows (104 positions), then the tape will be packspaced 6 times and read forward ~~twice~~ before a new read is attempted. This brings the area of tape in question back to a vacuum cleaner on the tape station which will attempt to clean the tape.

If this however does not succeed, the other options will be tried. Clean is recommended in addition to all other possible parameters.

PROCESS, can be specified so that the tape record will be handled regardless. This option does not cost any core positions, but just changes a branch instruction.

Process is generally better than the bypass option because many errors are so insignificant that it is better to process the records after noting where the error occurs (if possible) to check the output later. If the controls built into the program are good enough, there is less danger here than there is when a record is simply hopped over !

A dump tape option is used by many customers but I think this is a rather expensive use of a tape unit since the actual error rate is very low. Let us say this situation under normal circumstances should not occur for more than a few records a year. If it does occur often enough to justify a dump tape, then something is radically wrong at your installation !

~~The SCAN option is in the authors opinion a ridiculous contraption. I have never ever met or heard of an operator who knew what to do in the case he got this option, much less knew how to correct the error. The best a very capable operator can do in this case can be done more quickly and safely by the PROCESS option. Don't ever specify this option is the best rule !~~

The SCAN option will halt the 1401 - forcing a recreation of the tape in error and re-run of the current job.

11. cont. In the authors opinion all options offered are rather poor. At one installation which had a 1407 console typewriter, the author constructed a macro instruction RDERR which all programmers use which automatically gives operator instructions to allow dump (the rotary and I/O switch must be changed so that a re-read will read in the record without correcting the parity, replacing error positions by)(* of the entire error record on the console with exact error indication. This is a great advantage because if the record is then processed (this can be determined by the programmer or the operator), then the results should be predictable since we know which symbols are in error.

This dump with parity error identification cannot be achieved on a printer because the printer has no indication for incorrect parity.

In addition no extra unit such as a dump tape is needed. Extra tapes are better used as flip-flop for example.

* The error positons are in core as even parity characters but is typed as)(.

THE MACROS

1. GET

GET provides us with a data record (logical record) from an input file. It does this by branching to the DTF routine for that file which actually provides the record for us and if necessary utilizes the joint routines in the DIOCS to read in new tape blocks to replenish the supply of data records which the DTF routine utilizes.

Let's look at a typical GET macro's generated coding:

01	GET	INFILE, WORK1	
02	SBR	IOCUXT+3,0J009	STORE ADDRESS OF NEXT INST AFTER THE GET MACRO
03	B	INFILE	BRANCH TO THE DTF FILE ROUTINE
04	DSA	WORK1	ADDRESS OF RECEIVING AREA
05	DCW	@NO@	DUMMY (CAN BE SKIP/SPACE INST)
06	DCW	@$@	DUMMY (CAN BE PARAMETER 4 SKIP/SPACE CONSTANT)
07,0J009	EQU	*+1	EQU TO ESTABLISH A LABEL WHICH CAN BE USED AS EXIT FROM THE MACRO

This sequence is so common and simple that it is well worth learning - not least to aid in changes and corrections by patching.

IOCUXT is an instruction which looks like this

IOCUXT B O

and is simply used as a general exit from GET, PUT and certain other routines (very useful in coreprint analysis), so the first instruction in the macro uses a two operand SBR to store an address constant in this branch instruction. One of several advantages of this technique is that it is independent of the DTF itself since the IOCUXT is in the DIOCS.

The next instruction 03 is a branch to whatever is written as the first parameter of the GET macro. Without any checking whatsoever this branch is attempted. Of course parameter 1 is supposed to be the DTF FILE NAME and that name is given as symbolic label for the first instruction in the DTF routine. In other words we tell the macro where to branch to. If we don't write the file name correctly in the macro we will still get a branch to our first parameter even if it is for example an input area or non-existent (undefined). The main point is that we can easily correct any error by a patch or an "alter" during a reassembly with alterations without IOCS regeneration (see autocoder operating procedures).

1. cont.

The GET (and all other IOCS macros) are in fact "ordinary" macros which have no other connection or knowledge of IOCS than the file name which, for the macro is simply a parameter which it uses as a label.

This lack of communication between the GET macro and the DTF and DIOCS routines is one of the main reasons that this particular IOCS system cannot do more "work" directly in the GET sequence as the 1410 and 1401 (DISK) IOCS do. These systems use a different type macro system which allows information to be stored from one macro to the next - for example from DIOCS to DTF to GET. The result is that the processing time in this particular IOCS system is relatively high. There is much more branching back and forth than is necessary in these other systems.

The fourth line (04) is normally a dummy entry, a $, but if an area parameter is specified (that is if parameter 2 which should be a receiving area) then a DSA line comes which produces an address constant. Note that since this is a DSA - actual addresses can be used such as 1 or 1234.

The DTF routine uses the information now stored in IOCUXT to find out where the macro, which referred to it, is. Then the routine finds this parameter, if the DTF has not specified INDEXREG or WORKAREA.

The parameter is placed in the B-operand of an MCM instruction which is not executed if the parameter is $, but which otherwise moves the I/O area data record to the area we have named as our second parameter - even if we have named a file by accident. This of course ruins the whole DTF routine. Anything other than $ which is named in the second parameter will cause an attempted move !

A full discussion of GET should actually include a detailed description of how the DTF file routine functions. However, that is much too great a problem to take up at this point. Suffice to say that any student of IOCS should be able to "read" some DTF routines and interpret their functions and methods at the autocoder level. Program flow charts for the DTF routines will of course be found in the IOCS writeups.

2. PUT

Now that we know the GET macro it can suffice to say that the principles are exactly the same as for the GET macro, the principal difference being that the order of the parameters 1 and 2 are reversed. Thus parameter 2 will be placed in the branch instruction and parameter one will be the DSA constant.

Now it is easy to see that

```
PUT  UTFILE
```

will not work correctly since parameter two is not present and parameter one is a file name instead of an area name.

```
PUT   , UTFILE
```

is required if parameter one is missing.

It would be simple to rewrite the PUT macro so that a one-parameter put would regard that one parameter as the file name and thus eliminate a very common error. However since this was not done by the designers of IOCS, we must assume that the intention was to force the programmer to document the fact that the other parameter was intentionally left out.

This error is in any case simple to correct without a full IOCS regeneration as we mentioned under GET.

3. OPEN

```
OPEN    INFILE      Macro
B       IOCOPN      Branch to open routine
NOP     INFILE-17   Label information is at position
                    INFILE-17
```

The above is the coding generated.

The parameters named in the macros are assumed to be valid file names.

All DTF's are built up in such a manner that the address of the filename (= first position in the DTF routine) minus 17 positions is the lower section of the DTF table. This is where open routines can find label information and information about exits.

The main reason the second line is generated with a NOP is that the SBR instruction at IOCOPN stores the return address (in IOCQUT - also a very useful point to examine coreprints).

If a plain address constant had been used, then the return to the instructions following would have to be accomplished by some form of address adjustment. The use of NOP then is a cheap way of eliminating this difficulty.

From this explanation the reader should now be able via patching and alter to add, take away and change OPEN macros in a very straightforward manner.

As the reader can confirm, the first instruction of the DTF routine is (for example)

```
INFILE  B  IOCUXT
```

which obviously means that a GET or PUT can get nothing done since any attempt to B INFILE just branches to the exit without getting/putting any records.

What this means is that the file is not open. One of the things done by the open routine is to change this B IOCUXT to

```
INFILE  NOP  IOCUXT
```

Thus the path is open to the rest of the DTF routine and we say the file is "OPEN". CLOSE of course restores the B.

From this discussion the reader should now be able to examine a core print or to test any file by a BCE instruction to see if it is OPENed or CLOSEd. This is very useful in error analysis and end-of-file routines. The author has constructed a pair of simple macros to make this easy to remember. One example should suffice,

```
BCLOS    END, INFILE          MACRO
BCE      END, INFILE, B       GENERATED CODING
BOPEN    CONTIN, OUTFIL       MACRO
BCE      CONTIN, OUTFIL, N    GEN
```

4. CLOSE CLOSE macro is constructed in the same manner as OPEN.

```
CLOSE   INFILE
B       IOCLOZ
NOP     INFILE-17
```

and so therefore easy to patch and change.

CLOSE (and OPEN) can be traced on a core print by examination of the branch instruction at IOCQUT which is a B 000 instruction similar to IOCUXT. The address at IOCQUT will give information about which OPEN or CLOSE was attempted last. IOCQUT is used by other routines internally in the DIOCS as well as the FEORL macro.

In particular IOCQUT is used by the end of reel routine, so if the program has failed after EOR but before close, then exact information on which DTFroutine is concerned will be in this branch instruction.

One important point about CLOSE which is commonly misunderstood is that close does not check the trailer label on a tape input file. When the tape mark is read the trailer is read and checked to the extent specified and then tested for EOR or EOF. If it is end of file (EOF) then the users end-of-file routine (EOFADDR) is executed. It is however the function of the CLOSE macro to rewind the tape at end-of-file.

Generally speaking CLOSE should be executed for input files immediately upon EOF. This gives a "free" switch as described under OPEN and prevents a variety of errors.

CLOSE for an output file will

a) write out last block, with padding if necessary

b) write tape mark, trailer label, tape mark if standard labels

c) rewind, unload the tape (as specified).

5. FEORL This is not very commonly used but one important use can be mentioned as an example. Output reels which are to be sorted must not be filled up over the length of normal work tapes even if the output reel happens to be extra long. This is known as exceeding maximum file size (MFS).

By counting records the programmer can force the end-of-reel at the correct point using FEORL (MFS can be calculated manually or by the sort program) thus ensuring a correct sort.

6. RELSE The only practical use the author has put this macro to has been re-initialization of internal DTF routine accumulators when a reel is to be re-read. This is especially important on fixed blocked files since the OPEN does not initialize the blocking accumulators which are left at a useless value at EOF.

A typical sequence might be

```
OP      OPEN    F
        GET     F
        CLOSE   F
        RELSE   F     RE-INITIALIZE TO BLOCK START POINT
        B       OP    RE-OPEN
```

The main effect of the relse is that Π2Q accumulator is reset to Π4Q constants value. This can of course be done by an MLC instruction.

The file does not have to be open in order to use RELSE since relse skips over the first two instructions in the DTF routine - especially the B/NOP switch.

7. RDLIN RDLIN coding generated is similar to OPEN and CLOSE.

```
RDLIN   FILE
B       IOCRDL
NOP     FILE-17
```

IOCRDL is a 174 position SBR-type subroutine which essentially

a) reads a card

b) confirms that it is RDLIN (in col. 16-20)

c) moves its data to corresponding areas in the DTF table for the file named

The important point is that no check is made to determine that the RDLIN card information goes to the correct file. So the sequence which the operator places the RDLIN cards in can be critical especially if the RDLIN cards are used for output files.

In addition note that the RDLIN cards do no label checking etc. whatsoever - they only change the entries in the DTF table - label writing and checking is carried out as if no RDLIN card was read.

A point which causes unexpected difficulties is that the IOCRDL subroutine will execute

```
CS  80
```

Thus the read area is cleared of any word marks set in a "housekeeping" routine in the read area. This is one of the many reasons the author recommends that programmers avoid using the read area directly.

RDLIN can of course be used at any time a change in label information is desired and the use of RDLIN can of course be controlled with a sense switch. The principal use however is to change the creation date specification before the DTF table is compared to an input header for full checking.

Of course since creation date must be changed then all other information must also be correctly contained in the RDLIN card - no blanks can be used to indicate "no-change".