U.S. DEPARTMENT OF COMMERCE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NATIONAL WEATHER SERVICE
NATIONAL METEOROLOGICAL CENTER

OFFICE NOTE 130

How to Transmit Data From the 360/195 to the 360/40

Peter Chase
Automation Division

SEPTEMBER 1976

This note will tell you how you can send data files from the IBM 360/195 computer complex to the 360/40 computer complex.

What We Have

There are two computer complexes which run almost all of NMC's programs. Both of them are located in Federal Office Building No. 4 in Suitland, Maryland.

The first is a complex of three IBM 360/195's, operated by NOAA's Office of Management and Computer Systems, and located on the first floor ("downstairs") of Wing 1.

The second is a complex of three IBM 360/40's and one 360/30, operated by the Systems Management Branch of the Automation Division, NMC, and located on the third floor ("upstairs") of the same wing.

The 195 complex is one of the world's largest and fastest systems. It was designed to run programs with lots of calculations on arrays of numbers, so it does best on big numerical weather prediction models. It operates mostly in batch processing mode, although it drives a time-sharing system in the Washington metropolitan area. Scattered around the country are many remote terminals through which you can enter jobs and receive printout.

How We Came To Have An Interface

Formerly we could exchange data between computer complexes only by exchanging reels of magnetic tape. To make this task easier, two "long line" tape drives attached to the 195 complex were installed upstairs across from the tape drives for the 40 complex.

As we got to sending more and more transmissions upstairs, the amount of tape swapping got all out of hand. But we had planned all along to install a link between the computer complexes so that data could be passed directly, just as we used to do when we had three CDC 6600's downstairs.

So we installed IBM Channel-to-Channel Adapters in System 2 and System 4 of the 40 complex. Both of these adapters run downstairs to a switch box in the 195 complex which can select either of them and switch it to any of the three 195 systems.

Now it remained to design two monitor programs, one for each complex. For the sake of simplicity and small program size, we decided to send only one file at a time and in only one direction at a time; that is, in half-duplex mode.

This design means that the user program is separated from the transmission process, because we didn't want to keep the user programs waiting for a chance to send their data. Instead we let the user programs tell the monitor programs when the data files are ready to go--the user program "posts" the monitor--and then the monitor worries about sending the file.

GMCS programmers of the Suitland Computer Support Project wrote the control portion of the 195 Monitor, with Automation Division people responsible for the input and output portions. The 40 Monitor was entirely programmed by Phil Brandis of Automation Division. Later on, Automation Division took over the whole project.

How It Works

The Systems Management Branch of Automation Division runs the 40 complex, so they decide whether to run the 40 Monitor on System 2 or System 4. They call down to the 195 operator who manually selects either the line from the adapter on System 2 or the line from the adapter on System 4, and switches it to one of the three 195 systems, SY1, SY2 or SY3.

There are two versions of the 195 Monitor, depending upon whether System 2 or System 4 is being used upstairs. The 195 operator loads the correct version into the 195 system to which the adapter has been switched. Meanwhile, the operator upstairs loads the 40 Monitor.

When both monitors are running, they "shake hands" over the channel-to-channel adapter. At this point we are open for business.

When a user job has a dataset ready to send upstairs, it directly or indirectly calls upon an NMC subroutine named W3AGØ2. This subroutine first constructs a control record which contains the following information about the dataset:

---the dataset name,
---the volume serial of the disk where it resides,
---a flag, set if the dataset is to be deleted after being sent,
---a keyword indicating what kind of data is here,
---the volume serial of a backup tape in case the adapter fails,
---a special jobname to use if it is necessary to create a backup tape copying job.

W3AGØ2 has two ways of contacting the 195 Monitor:

If the user job is running in the same 195 system as the Monitor, W3AGØ2 can call upon a special region-to-region posting subroutine to pass the control record to the Monitor.

If the user job is running in a different 195 system, W3AGØ2 writes the control record in a special queue. There are three such queues, one for each system. Every 2-1/2 minutes the 195 Monitor scans the queues for control records.

If the Monitor is down (that is, not working) and likely to stay down for a while, a flag is set in each control queue to prevent W3AGØ2 from writing in them. In this case, W3AGØ2 constructs a backup tape copy job and causes it to be entered into the job queue. The dataset will be copied to a tape independently of the Monitor in this case.

If the Monitor is not down, it stacks the control record in its own queue until the adapter is free for transmission. At that point the 195 Monitor sends a file heading record to the 40 Monitor which contains the keyword indicating what kind of data will follow. The 40 Monitor uses this keyword to enter a table containing the name of the overlay program to load in order to process the data.

Next the 195 Monitor sends the data up to the 40 Monitor block by block. As each block comes up it is passed to the overlay program for processing.

When the 195 Monitor reaches the end of data, it passes an indicator to the 40 Monitor. After a normal completion, if the delete flag was set in the control record, the 195 Monitor will scratch the dataset. (The delete flag will be set unless the dataset was catalogued and was located using the catalogue.)

Sending Files Downstairs

The 40 complex can also send data files in the other direction down the 195's. But we won't describe this now, as it is very specialized. Later on we hope to make it more generally useful.

Planning Your Checkout

You should plan to check your program out in three steps:

(1) First write your output on a temporary dataset. Note that if you can avoid writing on a tape at this stage, your turnaround time will be much improved. If you are creating coded files (strings of characters) you should list your output and check it over carefully. Don't hesitate to write special checkout listing programs, as the trouble of doing so will be well repaid. If you are planning on using W3AGØ2 (see below) don't activate the CALL W3AGØ2 statement until stage (3). Note that we have a subroutine W3AQØ9 that can be used to test if the job is being run in actual production.

(2) After the first stage of checkout has been completed, write your output on a magnetic tape. (Obtain a tape number from the OMCS librarian, 763-5823.) This will mean a change only in the DD statement for your output dataset. Now contact the appropriate person in Automation Division for a simulated transmission of your data.

(3) If the simulated transmission looks good, you can now set up the final production version of your program. There will probably be small changes needed to put the actual destination codes in your program.

If your program is to be run on a regularly-scheduled basis by Automation Division, gather together your source language into one library member for each programming language you used. Fill out a Job Implementation Form. If you need disk space, fill out the necessary Data Set Request Form. (Forms available from Automation Division.) We will then take over your program and set up the necessary job control cards to run it.

If the program is to be run on an irregular basis by the user, you will still need to clear the transmission with the appropriate person. You will still need to plan your disk utilization, and you should prepare a final load module which contains all necessary library routines so that only a GO-step is needed for execution.

Planning Your Data Format

Data can be sent to the 40 complex only if there is an overlay program in the 40 complex to process it. This means that you must either conform to the requirements of an existing format or write your own overlay program for the 360/40. (This can be done--see Appendix A, by Phil Brandis.)

The three most common types of data transmission to the 40 complex are:

FAX, or graphical data for facsimile or Varian transmission;
TRAN, or teletype transmission;
KCRT, or display-tube transmission.

The format of FAX data is determined by the subroutines developed by the Graphics Section, Services and Applications Branch, Automation Division. You have to see them in any case.

For the format of TRAN data, see NMC Office Note 100, also by Phil Brandis.

For the format of KCRT data, call the Presentation and Display Branch of the Communications Division, NWS.

There are overlay programs in the 40 complex to process each of the above types of data. As for other types of data--as mentioned above, you can write your own overlay program. If you do, note that the 195 Monitor is transparent to all kinds of data so long as it can find buffer space in its own region. The 40 Monitor is limited to block sizes no bigger than 3600 bytes. But if you are sending binary data and you want to use backup tape in case of CTC adapter failure, you may have a translation problem with seven-track tape. See below.

Production Disk Space

The day will come when your program will need disk space on the 195 for its transmissions. Understand that many hundreds of operational products are transmitted each day from the 195 complex. Further, each product must reside on a disk pack that is known to the 195 Monitor; that is, on a production disk pack. Therefore we have developed the following method of handling disk space:

(1) We don't catalog datasets. Instead we assign them to a specific production disk pack and give them unique and highly descriptive names.

(2) The following three steps will occur in the job that creastes the dataset:

---the first step deletes the dataset just in case it is still on the disk from previous usage,

---the second step (your program) creates the dataset with a disposition of NEW,PASS.

---the third step posts the 195 Monitor and gives the dataset a disposition of KEEP. (Steps two and three will be combined if you are using W3AGØ2.)

(3) The 195 Monitor in time sends the dataset upstairs and then deletes the dataset so that its space may be reused. (Even if the dataset goes to backup tape, the copy job will delete the dataset after copying it if it was not catalogued.)

Don't worry about the details for a regularly-scheduled production program since the people at Systems Management Branch of Automation Division will set it up. Otherwise, see Procedure WWUPSTRS below.

You can transmit a catalogued dataset upstairs, but in this case the 195 Monitor will not delete it afterwards, nor will the backup tape copy job.

For catalogued disk space, apply to the Services and Applications Branch of Automation Division for a Data Set Request Form. The Systems Management Branch controls non-catalogued ("kept") production space.

The Nitty Gritty - FAX, TRAN and KCRT

Documentation on TRAN format is complete--see Office Note 100, and also Appendix C, where we lead you by the hand through Checkout Jungle.

Documentation is somewhat scattered on FAX programming. We tell you what we know in Appendix B, but you must see Graphics Support Section, Services and Applications Branch, Automation Division for the details.

Documentation is not available for programming the KCRT system. There is a manual for operating the KCRT system, available at every KCRT display station. Appendix D sets forth some bits and pieces of knowledge. In any case, contact the Presentation and Display Branch of the Communications Division.

Procedure WWUPSTRS -- Short Programs

This is the way most jobs post their datasets to the 195 Monitor. Only very long jobs might consider using W3AGØ2.

In the production stage, your job might look like this:

```
//STEP1  EXEC  PGM=IEFBR14

//SCRATCH  DD  DSN=kept.data.set,UNIT=333Ø,VOL=SER=NWSxxx,

//  DISP=(SHR,DELETE)
```

```
//STEP2  EXEC  PGM=yourprog

//yourout  DD  DSN=kept.data.set,UNIT=333Ø,VOL=SER=NWSxxx,

//  SPACE=whatever,DCB=whatever,DISP=(NEW,PASS)


//STEP3  EXEC  WWUPSTRS,K=keyword,V=tapeno,S=suffix,UNIT=grpnam

//  D='kept.data.set',LOADSET=B
```

Here, kept.data.set is the name of the dataset to be transmitted,
NWSxxx is the volume serial of the disk assigned to it,
yourprog is your program which creates the dataset,
keyword identifies the data type to the 40 Monitor,
tapeno is the volume serial of the backup tape,
suffix is a letter or digit to create a unique backup jobname,
grpnam is the desired tape-group, NMCTP7 or TAPE7.

keyword consists of exactly 8 EBCDIC characters, left-justified with right blank fill. Surround it with apostrophes:

```
K='FAX   ' for graphics transmissions
K='TRAN  ' for teletype transmissions
K='KCRT  ' for display-tube transmissions
```

tapeno consists of exactly 6 EBCDIC characters, left-justified with right blank fill. Usually tapeno does have 6 non-blank characters, so you don't need apostrophes around it. But if you do give a blank tapeno, if the channel-to-channel adapter is down, then the transmission will not be made and the dataset will stay where it is. This may be a useful option:

```
V='      ' to skip any backup tape activity
```

Subroutine W3AGØ2 -- Long, Long Programs

Suppose you have a very long-running program, like a forecast model, and suppose that early in the run you create a dataset to be sent upstairs. You can post that dataset to the 195 Monitor right away, without waiting for the program to end, by calling W3AGØ2:

```
CALL W3AGØ2 ( ddname, dsname, keyword, tapeno, grpnam, + suffix, kret, isw )
```

Here, ddname is a given string of 8 EBCDIC characters, left-justified with right blank fill, naming a DD statement which specifies (at least) the unit type and volume serial of the disk containing the dataset. If dsname is blank or zero, the DD statement should give the dataset name. Disposition should be SHR.

dsname is either a given byte which is blank or zero (hexadecimal 40 or 00) or a given 44-byte string of EBCDIC characters, left-justified with right blank fill, containing the name of the dataset to be posted.

keyword, tapeno and suffix are as for WWUPSTRS.

grpnam is a given string of 8 EBCDIC characters, left-justified with right blank fill, giving the group name of the backup tape units--either 'TAPE7 ' or 'NMCTP7 '.

kret is a returned integer error code as follows:

0 - dataset successfully posted
1 - dataset went to backup tape
2 - dataset not sent

isw is an optional argument. If not used, a default value of 0 is assumed. If used, it is a given integer code as follows:

Ø - If this is a production job, W3AGØ2 will attempt to send the dataset via the 195 Monitor. If this is not a production job or if the 195 Monitor does not respond, W3AGØ2 will attempt to send the dataset via backup tape if a non-blank tapeno is given.

1 - Regardless of whether this is a production job, W3AGØ2 will attempt to send it via the 195 Monitor or not at all.

2 - If a non-blank tapeno is given, W3AGØ2 will send the dataset via backup tape or not at all.

W3AGØ2 needs the following DD statements:

```
//ddname  DD  UNIT=333Ø,VOL=SER=NWSxxx,DISP=SHR

//QUEUE  DD  UNIT=333Ø,VOL=SER=NWS2ØØ,DISP-SHR

//BACKUP  DD  (any sequential file, RECFM=F, BLKSIZE=8Ø, 1 track)

//IJPDAØ1  DD  DSN=*.BACKUP,VOL=REF=*.BACKUP,UNIT=333Ø,DISP=SHR

//IJPWTR  DD  SYSOUT=Z
```

Files BACKUP, IJPDAØ1 and IJPWTR are needed only for tape backup. For BACKUP, a 1-track catalogued or SYSDA dataset may be used.

File ddname is the same as in the W3AGØ2 calling sequence. NWSxxx is the same as in WWUPSTRS. If you don't need the flexibility of being able to specify the data set name from inside your program, you can add DSNAME=kept.data.set to the ddname DD statement and use Ø for dsname in the W3AGØ2 calling sequence. If the dataset is catalogued, remove the UNIT and VOL=SER parameters from the ddname DD statement so that the dataset will not be scratched after being sent.

How does W3AGØ2 know if the job is being run in actual production? It calls subroutine W3AGØ9, which examines the third and eighth characters of the jobname. If they are both numeric, W3AQ09 assumes that this is a production job.

Backup Tape

In production, the backup tape, if any, will be a tape assigned to you by the Systems Management Branch (not OMCS). As of this writing, this will be a seven-track tape, so there will be translation problems. The default translation will be TRTCH=ET, meaning translation from EBDIC to BCD with even parity. Since there are 256 possible EBCDIC characters but only 63 possible BCD characters, some characters will not get translated back again. This will not be a problem if you stick with the standard TRAN and KCRT formats.

If the W3AGØ2 subroutine recognizes the keyword 'FAX ', it will cause the TRTCH=C conversion to be used, since FAX data is binary.

If you have non-standard data to be transmitted and it is binary, you have a problem.

Good luck!

The 360-40 channel to channel (CTC) code comprises a main task and 2 subtasks. One subtask (XMITO195) transmits files (also called jobs or data sets) to the 195 while the other subtask (XMITO4Ø) receives files from the 195.

The CTC code occupies the F1 partition in the DOS system and is allowed a size of 64K. Currently the CTC code actually uses about 20K (including 2 3600 byte data buffers) with 4K set aside for expansion. This leaves about 40K available for a user code (explained below) with the user code also permitted use of the 2 3600 byte data buffers. The user code is always tacked on to the end of the CTC code. If, in the future, the CTC code should expand beyond 24K, the 40K designated for a user code would diminish accordingly.

The XMITO195 subtask not only links up with the 195 but also establishes communication with a 360-40 user code in order to send a file to the 195. Each user code requires a 40 character entry in a subtask table. These 40 bytes are as follows:

* The first 14 bytes contain information the 360-40 needs to know in
* order to load and execute the user code properly.
4 bytes - user code name (EBCDIC) - must be first 4 characters of user code (followed by first executable instruction)
8 bytes - user code DOS loading name (EBCDIC)
2 bytes - status information (HEX) - only 1 option currently used (This option decides whether or not to bring the code in fresh if its already in core).
* last 26 bytes are the first 26 bytes sent in a heading record to the 195 to
* handle the 195 needs.
2 bytes - status information (HEX) - irrelevant to user code
8 bytes - 195 job name (EBCDIC)
2 bytes - size of every user data record (HEX 1-3600 bytes)
8 bytes - 195 data set name (EBCDIC)
6 bytes - not currently used - set to blanks (EBCDIC)

EXAMPLE:

```
    DC      C'RAW1'              user code name
    DC      C'RAWDbbbbb'         loading name
    DC      X'1ØØØ'              load fresh
    DC      X'ØlØØ'              status information
    DC      C'WW1RAW1b'          195 job name
    DC      H'2Ø8Ø'              Record size (2Ø8Ø bytes)
    DC      C'DOWNDAT1'          195 data set name
    DC      6X'4Ø'               6 extra blanks
```

The XMITO195 subtask communicates with the user code via the following call sequence.

```
            BALR 14,15  (15 has address of 1st executable instruction in user code)
input to   (DC   AL1(input status)
 user code (DC   AL3(input buffer address)
```

```
output from (DC    AL1(output status)                )
 user code  (DC    AL3(output address or parameter)  )
            RETURN
```

The first 4 bytes are on a full word boundary and contain input information to the user code. The last 4 bytes are also on a full word boundary and contain certain information from the user code. Each time the subtask calls the user code it saves all registers (using register 11 to do so) and zeros the 4 output bytes. When the user code (stasher) returns B 8(14) , the subtask restores all registers (using register 11) and zeros the 4 input bytes.

The following defines the usage of the input status byte (only 1 bit is used at any one time)

X'Ø' - Subtask wants user code to stuff the next data record into the input buffer address
X'1' - Subtask wants user code to stuff 1st data record into the input buffer address
X'2' - Subtask asks whether the user code can send the file now
X'4' - Not used
X'8' - Subtask forced to terminate data file
X'1Ø' - Not used
X'2Ø' - " "
X'4Ø' - Subtask requests user code to resend the file from the beginning
X'8Ø' - Not used

The subtask flip-flops between 2 input data buffers. The data passed from the user code to the subtask must always be placed in the buffer address specified in the call sequence.

The following defines the usage of the output status byte sent by the user code.

X'Ø' - No problems-normal read (not 1st or last) - also normal response to'can you', terminate and resend.
X'1' - No problems - 1st data record
X'2' - " " - last data record
X'4' - Can't send file right now
X'8' - No data to send (not even 1 data record)
X'1Ø' - Close out job (no more data records)
X'2Ø' - Not used
X'4Ø' - " "
X'80' - Terminate this job

Normally the following sequence of events takes place. After loading the user code, the subtask asks the user code whether he can send the file. If the user code responds yes (X'Ø'), the subtask requests the first data record from the user code. The user code gets the first data record, stuffs it in the buffer address provided in the call sequence and makes a "no problems" return to the subtask (X'1').

Thereafter the subtask keeps requesting the next data record and the user code gets it, puts it into the appropriate input buffer and makes a "no problems" return to the subtask (X'Ø'). When the user code ships over the last data record, it makes a "no problems" return to the subtask (X'2'), whereupon communication twixt the subtask and the user code ceases.

Usually the user code returns only 1 bit in the output status but if we have a 1 Record file, the output status bit would reflect "no problems - 1st and last data records" and have 2 bits on (X'Ø3').

With the exception of registers 14 and 11, there are no register restrictions and the user code is free whatever registers it wishes. The output address in the call sequence is currently not used and were we to utilize it in the future for some purpose, register 1 would be involved.

It is up to the user code to handle its own I/O and I/O problems and to always return, regardless of circumstance to the subtask [B 8(14)].

If the user code can't send the file or has no data to send, the user code is considered done and a new user code is free to take its place. If the user code doesn't know which is the last data record and only finds out that there are no more data records when requested to send the next data record, it would return the "close out job" bit (X'1Ø') enabling the subtask to close out the job properly. If the user code encounters some problem and can't proceed, it would finalize its pointers and send the termination flag (X'8Ø') to the subtask. The job would then be terminated and dropped.

On the other side, if the subtask encounters a problem and can't continue or is requested to quit by the 195, the subtask will send a terminate flag (X'8') to the user code. The user code is expected to take whatever termination action he wishe (write checkpoints, reinitialize pointers etc.) and then return normally (X'0') to the subtask.

Currently we are not using the input 'resend' bit (X'4Ø'). If, in the middle of a file, the subtask or 195 should wish the file to be resent, the user code would be reloaded fresh and the subtask would reinitiate communication with the user code from the beginning ("can you...")

The following is the procedure employed with the 40 user code in the XMIT04Ø subtask.

Each user code requires a 20 byte entry in a subtask table. The 20 bytes defined as follows:

4 bytes - code name (EBCDIC) - must be 1st 4 characters of user code (followed by the 1st executable instruction).
8 bytes - user code loading name (EBCDIC)
2 bytes - status info (HEX) - 2 options currently available
(a) Whether or not to bring the user code in fresh if its already in core.
(b) Whether or not to translate EBCDIC data to Ascii.
6 bytes - 6 byte 195 name (EBCDIC)

EXAMPLE:

```
    DC    C'FAXX'              code name
    DC    C'FXPKbbbb'          loading name
    DC    X'1ØØØ'              load fresh and don't translate
    DC    C'FAXbbb'            195 name
```

When a file heading record comes up from the 195, the XMIT04Ø subtask gains control and loads the appropriate 40 user code at the end of the CTC program area (same area used by a user code when the XMIT0195 subtask is in the driver's seat).

```
          BALR 14,15 (Reg. 15 contains addr. of 1st executable instruction in
            user code)
input to (DC  AL1(input status)                          byte Ø
user code(DC  AL3(address of input data record)          byte 1,2,3

output   (DC  AL1(output status)                         byte 4
from user(DC  AL3(wait address etc)                      bytes 5,6,7
code      RETURN
```

The first 4 bytes are on a full word boundary and convey input information to the user code. The last 4 bytes are also on a full word boundary and convey output information from the user code. Each time the subtask goes to the user code, it saves all registers (using register 11) and zeros the 4 output bytes. When the user code (stasher) returns [always B 8(14)] the subtask restores the registers (using register 11) and zeros the 4 input bytes.

The following defines the usage of the input status byte
X'Ø' - Normal data record (not first or last)
X'1' - 1st data record
X'2' - Can user accept this file
X'4' - Not used
X'8' - forced to terminate job

X'1Ø' - Close out job - no data record (job may already be closed)
X'2Ø' - Not used
X'4Ø' - Not used
X'8Ø' - Last data record (go close out the job)

The subtask flip flops between 2 input data buffers. The address of the data buffer being released to the user code is found in bytes 1,2,3 of the call sequence. Except for the case of 1-record files where 2 bits (X'81') are on to indicate the 1st and last data record, only 1 bit should be on in the input status byte.

The following explains the usage of the output status byte sent by the user code to the subtask.

X'Ø' - 'no problems' return - normal response to any data record,
'can you', terminate and closeout.
X'1' - Not used
X'2' - Can't accept file now
X'4' - Not used
X'8' - Not used
X'1Ø'- Not used
X'2Ø'- Not used
X'4Ø'- Resend job (from beginning)
X'8Ø'- terminate the job

Normally the following sequence of events takes place. The 40 CTC code receives a file name from the 195 and passes control to the XMIT04Ø subtask. The subtask decodes the file name and loads in the appropriate user code. The subtask then asks the user code if he can accept this file. The user code says he can by returning normally (X'Ø') to the subtask. The subtask then signals the 195 to ship the first data record and as its received from the 195, stores it away in the 1st of the 2 input buffers. The 40 then signals the 195 for the next data record and as that is being received, passes over the 1st data record to the user code. When this next record has been received from the 195 and the user code is finished dealing with the 1st data record, the 40 requests another record from the 195. While this one is being received from the 195, the subtask ships the previous record to the user code. This goes on till the last record is received from the 195. The 40 finishes sending the next-to-last record to the user code and then turns on the last data record flag (X'8Ø') and sends this last record to the user code. At this point, the job is finished and communication ceases between the subtask and the user code.

With the exceptions of registers 14 and 11, there are no register restrictions and the user code is free to use all registers as it sees fit. The output (wait) address in the call sequence is currently not used and were we to use it in the future, register 1 would be needed.

As before, it is up to the user code to initiate its own I/O (disk etc.) and handle any I/O problems resulting from such initiation. Regardless of circumstances, the user code should always return to the subtask via B 8(14).

If the user code can't accept the file now, the user code is considered finished and a new user code free to take its place. If, for any reason, the user code can't proceed, it would send the termination flag (X'8Ø') to the subtask and the user code would be dropped. At any time during the passage of the file, if the user code requests a resend (X'4Ø'), the subtask will pass this request on to the 195 and then start over by shipping the user code the 1st data record etc.

On the other side, when the subtask sends over the 1st record (X'1'), the user code should initialize its pointers and when the subtask ships over the last data record (X'8Ø'), the user code should close out the file. If the subtask sends the "close out job-no data record' (X'1Ø') bit, the user code already received the last data record from the subtask and should now close out the file. If the subtask should be unable to proceed or be requested to terminate by the 195, the subtask will send the terminate flag (X'8') to the user code and expect the user code to take whatever termination action it requires before returning normally (X'Ø') to the subtask. Communication then ceases twixt the subtask and user code.

GRAPHICS TRANSMISSIONS - APPENDIX B

There are two writeups for the individual graphics subroutines but nothing that ties it all together. The best way to get started is to come in person to the Graphics Support Section, Services and Applications Branch, Automation Division, and obtain some kind of skeleton program which you can flesh out gradually.

The checkout procedure described under Planning Your Checkout must be modified for FAX since you can't look at your output until it has been actually plotted. Checkout step (2) should read like this:

> Use subset number 80 for checkout. (The subset number is the index to the disk space in the 40 complex where your graphics output will be stored.) Write your output on an E-tape downstairs. When your program has run successfully, use a separate IEBGENER copy job to copy the E-tape to an upstairs tape. (See Micki Farley, Computer Operators Section, 763-5813 for a tape number upstairs.) When the dataset has been copied to the upstairs tape, call the 360/40 operator at 763-5727 (if you are in the World Weather Building, use the direct line in the Varian room). Ask them to "V-file" your tape with indent zero. Output will appear on the Varian printer, Room 405, WWB, between 8 and 9:30 AM. Make arrangements to pick up your output. The Reproduction Section of the Forecast Division may be able to help.

For production you will need a subset number and one or more fax or Varian schedule sequence numbers, sometimes called slot numbers. Your program needs to know the subset number, which gets inserted in the dataset. The connection between the subset number and the schedule sequence number is made in a table in the 40 complex. This table controls the routing of your dataset.

Your program should also know the schedule sequence number in order to display it in the chart's legend.

Call Alexander Sadowski, Technical Procedures Branch, Meteorological Services Division, at 427-7713 for a fax schedule sequence number.

Call Dick Schnurr, Graphics Support Section, Automation Division, at 763-8115 for a subset number. He will arrange with Al Brinkley, Systems Management Branch, for a Varian schedule sequence number if needed.

When people talk to you about "map numbers" or "event numbers", ask them if they mean subset numbers or schedule sequence numbers or what. The 360/40 operators may confuse "ID" numbers with "indent" numbers. The indent number can be used to position the graphic output on the paper at plotting time. Normally tell the 360/40 operator to use zero indent.

TELETYPE TRANSMISSIONS - APPENDIX C

Familiarize yourself with NMC Office Note 100, by Mr. Phil Brandis. If you have questions, you can call Mr. Robert Allard of Automation Division at 763-8115.

Use the NWSTXPRT facility to list your output for checkout. In the example below, assume STEP1 creates a TRAN dataset. Then STEP2 will give you a listing of the dataset as it would appear on teletype and STEP3 will give you a blocked listing.

```
//STEP1  EXEC  NFORXCLG,...

...

//GO.FTmmFnnn  DD  DSN=&&TEMP,UNIT=SYSDA,SPACE=(TRK,(2,1)),

//  DCB=(RECFM=F,LRECL=128Ø),BLKSIZE=128Ø),DISP=(NEW,PASS)

//STEP2  EXEC  NWSTXPRT,PROG=PRTBULL

//TXPRINT.FTØ9FØØ1  DD  DSN=&&TEMP,DISP=(OLD,PASS)


//STEP3  EXEC  NWSTXPRT,PROG=TTYPRNT

//TXPRINT.FTØ9FØØ1  DD  DSN=&&TEMP,DISP=(OLD,PASS)
```

If these listings look perfect, change your output to magnetic tape. Your output DD statement will now look like this:

```
//GO.FTmmFnnn  DD  UNIT=TAPE7, VOL=SER=Etttttt,LABEL=(1,BLP,,OUT),

// DCB=(RECFM=F,LRECL=128Ø,BLKSIZE=128Ø),TRTCH=ET,DEN=2),

// DISP=(NEW,PASS)
```

Here, Etttttt is the volume serial of a 7-track unlabelled tape assigned to you by the tape librarian at OMCS. Call 763-5623.

If you have been faithfully following Office Note 100, you will at this stage be using the checkout catalog number, 100421. Now you can call Ms. Ann McDaniels at 763-5813 who can arrange for a simulated transmission of your teletype message. The 100421 catalog number will ensure that your bulletin won't go out "live".

If Ms. McDaniels reports that the bulletin looks good, call Mr. Phil Dales of Communications Division at 427-7741 for an operational bulletin heading and a catalog number. Insert these in your program. Notify Mr. Jim Lish at 763-5813 when you want to start production, since he must update the communications tables with your header and catalog number in order to route your bulletins properly.

DISPLAY TUBE TRANSMISSIONS - APPENDIX D

The KCRT system is run by the Presentation and Display Branch of the Communications Division in FOB#4, Room 2307. Call them at 763-5971.

The CRT's can display 32 rows of 84 columns each, but only 1004 characters can be sent to each CRT. This means that most of the display must be blank. Special control characters can be used to generate multiple blanks, to terminate a row with blanks, or to fill the remainder of the screen with blanks, so that the entire screen can be used. Three special characters are:

EBCDIC char	Effect
=	Fill the remainder of the row with blanks
$	Generate four blanks in place of the $
*	Fill the remainder of the screen with blanks, thus terminating the record. (This must be the last character of each record.)

Each block of 16 CRT data has the following format:

Bytes 1-4 are the bin name, 4 EBCDIC characters assigned by the P&D Branch and indicating a certain disk space in the 40 complex.

6-9 Byte 5 is a blank.
Bytes [redacted] are the count of characters in the block, including byte 5 [redacted] but excluding bytes 1-4. The count is given in EBCDIC digits with leading zeroes, not blanks.

Bytes 10-1012 are data characters. The block can be extended to a maximum of 1104 characters, but characters 1013-1104 may not be scanned. The last data character must be an asterisk *.

Geographical displays are much improved by using geographical backgrounds. These can be sketched in using periods, slants and reverse slants (0-2-8 punches).