NASA Technical Memorandum 81355

USER'S MANUAL FOR SYNC, A FORTRAN PROGRAM FOR MERGING AND TIME-SYNCHRONIZING DATA

Richard E. Maine

March 1981

NASA

NASA Technical Memorandum 81355

USER'S MANUAL FOR SYNC, A FORTRAN PROGRAM FOR MERGING AND TIME-SYNCHRONIZING DATA

Richard E. Maine
Dryden Flight Research Center
Edwards, California

National Aeronautics and
Space Administration

1981

CONTENTS

	Page
INTRODUCTION	1
SYNCHRONIZATION ALGORITHM	1
INPUT DESCRIPTION	5
Infile Definition	5
Skew Definition	7
Method Definition	7
Outfile Definition	8
Time Interval Definition	10
Examples	11
PRINTED OUTPUT	12
Normal Output	12
Error Messages	13
FILE HANDLING	14
PROGRAM TRANSMITTAL AND IMPLEMENTATION	15
APPENDIX - SUBROUTINES AND COMMON BLOCKS	17
Subroutines	17
Common Blocks	22
REFERENCES	23
SYNC (microfiche supplement)	Inside back cover

USER'S MANUAL FOR SYNC,

A FORTRAN PROGRAM FOR MERGING

AND TIME-SYNCHRONIZING DATA

Richard E. Maine
Dryden Flight Research Center

INTRODUCTION

This report is a user's manual for the FORTRAN 77 computer program SYNC. The basic function of SYNC is the merging and time-synchronization of data. The options available allow SYNC to be used for several other types of operations on files of time-dependent data; these operations include time-skewing data channels and thinning or interpolating data to sample rates not necessarily commensurable with the available sample rate.

The operation of the program is described in detail. The input cards required to specify the program options are described. The description is followed by a list of program messages and error conditions. The body of the report concludes with a discussion of the transmittal and implementation of the program. Individual subroutines and common blocks are described in the appendix. A complete listing of the SYNC program with reference maps produced by the CDC FTN 5.0 compiler (ref. 1) is included on microfiche as a supplement.

SYNCHRONIZATION ALGORITHM

In this report time-synchronization means the computation of data frames at uniform sampling intervals. Each data frame contains the values of all of the data channels at the time of the data frame. In general, the input file need not have a measurement of each data channel at exactly the frame time. In this event, some algorithm is required to compute the data values to be used between the times on the input file.

The SYNC program is intended to accept data which need not be sampled at uniform time intervals and may be in a compressed format. The time-

synchronization algorithm must, therefore, be fairly versatile. This section describes the general algorithm used and two special cases.

The general algorithm is designed to accommodate data which were sampled at a uniform sampling rate and may optionally have been compressed. The compression consists of omitting data samples which have not changed from the previous sample.

The synchronization algorithm uses two variables specified by the user, the channel time skew, CHSKEW, and the sample interval, CHDT. These variables can have different values for different channels.

The input file gives a time for each data value. The first step of the synchronization is to add the time skew to the time indicated by the input file. All subsequent computations use this skew-corrected time. If an input data value for some channel is available at exactly the frame time under consideration, that data value is used. Otherwise, the SYNC program computes the data value at the frame time based on the data values at the input times which bracket the frame time. These times are called TCLAST and TCNOW, and the corresponding data values are VCLAST and VCNOW.

The program computes a time called THOLD that is equal to the greater of TCNOW - CHDT or TCLAST, where CHDT is the sample interval for the channel. If THOLD lies between TCLAST and TCNOW (that is, 0 < CHDT < TCNOW - TCLAST), figure 1 represents the interpolation scheme used between TCLAST and TCNOW.

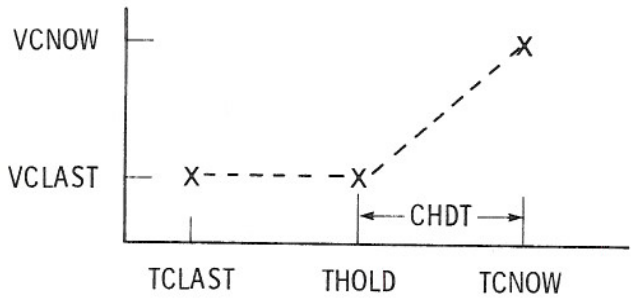

Figure 1. Illustration of hold and interpolate algorithm.

For frame times from TCLAST to THOLD, the value VCLAST is used. From THOLD to TCNOW, the value is obtained by linear interpolation.

This algorithm is appropriate for the interpolation of data compressed as described previously. It is presumed that the channel was sampled at time THOLD, but that the data value was not recorded because it was equal to the previous value. There may also have been other omitted samples between TCLAST and THOLD. The algorithm described thus decompresses the data and then interpolates. For this algorithm to work correctly, the original data must have been sampled at uniform time intervals and the correct value of the time interval

must be used for CHDT. Otherwise the program cannot correctly reconstruct the times of the original uncompressed data samples.

Figure 2 illustrates the interpolation between TCLAST and TCNOW when THOLD is equal to TCLAST (that is, CHDT ≧ TCNOW - TCLAST).

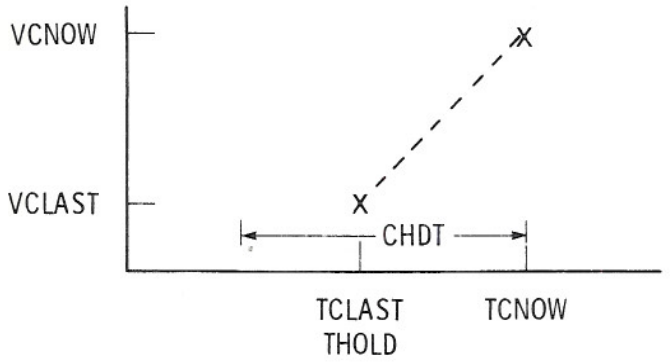

Figure 2. Illustration of linear interpolation algorithm.

The algorithm is identical to the previous case, except that there is no interval between TCLAST and THOLD. As long as the sample interval is less than or equal to CHDT, the program uses straight linear interpolation. This mode of operation is appropriate to input channels that are not compressed. The input sample intervals need not be uniform, but they should be less than or equal to CHDT. Sample intervals greater than CHDT are treated as in figure 1. The program does not include an option to always interpolate linearly (regardless of the sample interval) because such an option would entail significant computational burden.

Figure 3 illustrates the special case of CHDT equal to 0. In this case, THOLD is equal to TCNOW. The value VCLAST is used between TCLAST and

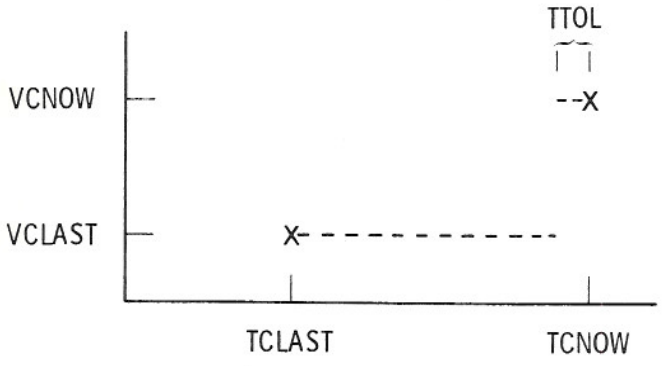

Figure 3. Illustration of hold-last-value algorithm.

TCNOW - TTOL. From TCNOW - TTOL to TCNOW, the value VCNOW is used. Since this situation has a discontinuity, it must be handled carefully to preclude significant errors. The basic intention is to hold the value VCLAST until time is exactly equal to TCNOW. Since time is used as a floating point variable in the SYNC program, the tolerance TTOL is required to insure that a frame which should be at a time exactly equal to TCNOW will use the value VCNOW. Without this tolerance, the roundoff error in floating point arithmetic could result in the use of the value VCLAST. The value of TTOL is the same for all data channels; it is nominally 0.0001 second. This mode of operation is appropriate to some forms of compressed data and to flags or channels which take discrete special values such that linear interpolation is meaningless.

The previous discussion neglects the possibility that the frame time might be before the first input time or after the last input time. If the frame time is after the last input time, the data value at the last input time is used. This is consistent with all of the above schemes, with TCNOW considered to be infinite. In order to define the output used before the first input time, each channel is initialized with a dummy value of 0 at time -10^{30}. The data value used will thus be 0 until within CHDT of the first input time. In the important special case where there are never any input values for a channel, the data values for that channel are always 0 .

The time-synchronized data for up to LCBUF (see the PROGRAM TRANSMITTAL AND IMPLEMENTATION section) output frame times are stored in a circular buffer called CIRBUF. The circular buffer is an array dimensioned MAXOUT by LCBUF; that is, a total of 12,800 words with the current values. It is by far the largest block of storage used by the program.

The circular buffer contains time-synchronized data for up to LCBUF output frame times. This buffer is necessary because relative time skews, differing synchronization algorithms, and differing input sample intervals may cause the synchronized values of some channels to be defined many frames ahead of the data available for other channels at a particular point of the processing. If any time tag read from an infile is such that the time tag plus the largest channel skew for that infile is more than LCBUF output frames ahead of the earliest unfinished frame, SYNC considers a significant time jump to have been encountered on the infile. The buffer is re-initialized to start after the time jump, and an appropriate error message is printed.

The program checks the size of LCBUF; it must be sufficiently large so that the buffer will not overflow in the above described manner during normal processing of data with no time dropouts. If LCBUF is too small, an error message is printed and the program stops. The exact criterion is that for each infile, (LCBUF - 1) times the outfile sample interval must be greater than the read-ahead time plus the maximum channel skew plus the default CHDT. The read-ahead time is 1.1 times TTOL plus the maximum of the difference CHDT minus the channel skew. Note that the file skew specified on the infile definition (see INFILE DEFINITION section) does not enter into this criterion. File skews are handled quite differently from channel skews; as a result, file skews can be arbitrarily large without affecting dimension limits.

INPUT DESCRIPTION

This section describes the card input required to run program SYNC. All input is free form; input need not be in any specific columns. Blanks may be freely used between keywords, values, and/or delimiters. In many cases, the program action depends on what delimiter terminates a field. If a field is terminated by one or more blanks, followed by a valid nonblank delimiter, the field is considered to be delimited by the nonblank delimiter. A field is considered to be blank-delimited only if there are no delimiters except for blanks separating it from the following field. Blanks must not be embedded within keywords or values. A keyword or value cannot be continued from one card to the next. Character values must not be enclosed in quotation marks.

The card input consists of a sequence of definitions. These definitions need not correspond to cards: there may be several definitions on a single card, and one definition can be spread over several cards. Each definition is composed of several fields. Fields longer than 10 characters are truncated without comment. The first field of each definition, which indicates the type of definition, is delimited by a blank or a comma. The various types of definitions and the fields within them are described in the following subsections. If the type of definition is not recognized, the program stops with an error message. The definitions can be given in any order consistent with the restrictions stated in the subsections. As a consequence of these restrictions, the first definition must always be an infile definition, and the time interval definition (if one is present) must be the last definition. An outfile definition and at least one infile definition are required. All other definitions are optional.

Infile Definition

If the first three characters of a definition type field are "INF", the definition is an infile definition. An infile definition defines the basic characteristics of an input data file. The first definition of a job must be an infile definition. Up to five infile definitions are allowed. (The section entitled PROGRAM TRANSMITTAL AND IMPLEMENTATION explains how to change this limitation.)

The second field of an infile definition must be a number between 1 and 5 inclusive, delimited by blanks, a comma, or a colon. This number, called the infile number, is used to refer to the file in subsequent definitions. The choice of the number is arbitrary within the range from 1 to 5. Infiles need not be defined in increasing order, nor need the infile numbers used be consecutive. The program will stop with an error message if two infile definitions use the same infile number.

The remaining fields of an infile definition consist of keywords, which are delimited by blanks or an equal sign and followed by values. The values are delimited by blanks, a comma, or a slash (/). A slash delimiter denotes the end of the definition. The keywords recognized are listed below, along with descrip-

tions of the following data fields. The keywords can be given in any order. Only the characters underlined below are checked for keyword recognition; the remaining characters of the keyword can be different or omitted. At least one keyword/value pair must be present.

NAME - name by which the system identifies the infile. Only the first 10 characters of the name are retained. The name cannot include slashes, commas, or embedded blanks. This name will be used in the file specifier of the FORTRAN OPEN statement. It is used nowhere else. The allowable names are processor dependent. The default value for NAME is 'FILE' concatenated with the infile number. This default name may not be allowable for all processors.

FORMAT - infile format. This is a 1- to 10-character string (characters after the tenth are ignored). This variable is made available to the routines which manipulate the input files, so that these routines can branch to appropriate code for manipulating different infile formats. The variable is not used by the routines published; it is provided to facilitate possible modifications. The default value for FORMAT is a blank string.

SKEW - infile time skew (seconds). This floating point quantity is added to all time values read from the infile. Any individual channel skews specified in a skew definition are in addition to the infile skew. The default value for SKEW is 0.

CHANNELS - number of data channels on the infile. Dimensions limit this integer variable to a maximum value of 100. The PROGRAM TRANSMITTAL AND IMPLEMENTATION section discusses how to change the dimension limit. Values larger than the limit will result in an error message. Any reference to a channel number greater than the number of channels for an infile will be diagnosed as an error. The default value of CHANNELS is 0.

UNIT - FORTRAN unit number for the infile. The default value of this integer variable is equal to the infile number.

METHOD - default synchronization method. This floating point variable is the default value of CHDT, which will be used to specify the synchronization method for any data channel of the infile. This default can be overridden for specific channels by a method definition. The dependence of the synchronization algorithm on the value of CHDT is described in the section entitled SYNCHRONIZATION ALGORITHM. The value of METHOD can alternatively be any character string beginning with an 'H' and containing no embedded blanks, commas, or slashes. In such a case, the default value of CHDT is 0, which results in a hold-last-value algorithm. The default value of METHOD is 0.

Examples of valid infile definitions are shown below.

```
INF 5 CHANS = 10/
INFILE 1 UNIT = 9, NAME = X15DATA SKEW = -.05 METHOD = HOLD-LAST-VALUE
CHANNELS = 15/ INFILE 3 CHANS = 20 METHOD = .025 /
```

Skew Definition

If the first two characters of a definition type field are SK, the definition is a skew definition. A skew definition defines the time skews of the data channels within an infile. A skew definition cannot reference an infile until after the corresponding infile definition. A skew definition is not required if all of the channel skews for an infile are 0.

The channel skews are added to the time values from the infile plus the infile skew specified in the infile definition to obtain the skew-corrected time values. The magnitude of the allowable skews is limited by program dimensions as discussed in the SYNCHRONIZATION ALGORITHM section. The default value for any skews not specified in the definition is 0.

The second field of a skew definition must be the blank-delimited keyword FILE. The third field is the infile number, delimited by blanks, a comma, or a colon. All of the information in the skew definition pertains only to this infile. The program stops with an error message if there is no previous infile definition for the infile number.

The remaining fields of the skew definition are skews or channel numbers. These fields are delimited by blanks or commas, except for the last field of the definition, which is delimited by a slash. The content of each field determines whether it is a skew or a channel number. If a decimal point occurs in the field, it is a skew; otherwise it is a channel number.

The processing of the skews and channel numbers is as follows. First, the current channel number is initialized to 1. Whenever a skew value is encountered, the skew for the current channel is set to that value and the current channel number is incremented by 1. Whenever a channel number is encountered, the current channel number is set to this value. Any attempt to assign a skew to a channel number larger than the number of channels on the infile will result in an error stop.

Examples of valid skew definitions are shown below. These examples assume that infiles 1 and 3 have been defined with at least 10 and 7 channels, respectively.

```
SKEWS, FILE 3: .01, .012, -.09, .01, .005, 0., .01 /
SKEWS FILE 1
10, .003, 7 .004 2 .001, -.003 .002 /
```

Method Definition

If the first four characters of a definition type field are METH, the definition is a method definition. A method definition defines the values of the CHDT variables that specify the synchronization methods for the various data channels within an infile. A method definition cannot reference an infile until after the corresponding infile definition. A method definition is not required if all of the channels for an infile use the default method for the infile.

The allowable values of CHDT and the dependence of the synchronization algorithm on the value of CHDT are discussed in the SYNCHRONIZATION ALGORITHM section. The default value of CHDT for any channel not referenced in the method definition is given by the METHOD keyword in the appropriate infile definition. The second field of a method definition must be the blank-delimited keyword FILE. The third field is the infile number, which is delimited by blanks, a comma, or a colon. All of the information in the method definition pertains only to this infile. The program stops with an error message if there is no previous infile definition for the infile number.

The remaining fields of the method definitions are methods or channel numbers. These fields are delimited by blanks or commas, except for the last field of the definition, which is delimited by a slash. The content of each field determines whether it is a method or a channel number. If the first character of a field is "H" (for hold-last-value), it is a method field and implies a CHDT value of 0. If the first character of a field is "D" (for default), it is a method field and the CHDT value used is the default given in the infile definition. If the first character of a field is neither "H" nor "D", and if the field contains a decimal point, then it is a method field giving the value of CHDT. If the field meets none of the above requirements, it must contain an integer channel number.

The procedure for the assignment of the method fields to data channels in the infile is identical to the procedure used to assign skews in a skew definition.

Examples of valid method definitions are shown below. These examples assume that infiles 2 and 4 have been defined with at least 10 and 4 channels, respectively.

```
METHODS, FILE 4: HOLD, DEFAULT, .05, HOLD /
METH FILE 2:
8 .02 0. .04 4 HOLD /
```

Outfile Definition

If the first four characters of a definition type field are OUTF, the definition is an outfile definition. An outfile definition defines the characteristics and data channels of the output data file. One and only one outfile definition must be present.

After the definition type, the remaining fields of an outfile definition consist of keywords delimited by blanks or an equal sign and followed by values. Except for the keyword FROM, each keyword is followed by a single value field delimited by blanks, a comma, or a slash. A slash delimiter denotes the end of the definition. The keywords recognized are listed below, along with descriptions of the following data fields. The keyword DT must be specified; all others are optional. The keywords can be given in any order, except that the keyword FROM must be last if it is present.

NAME - name by which the system identifies the outfile. Only the first 10 characters of the name are retained. The name cannot include slashes, commas, or embedded blanks. This name will be used in the file specifier of the FORTRAN OPEN statement. It is used nowhere else. The allowable names are processor dependent. The default value for NAME is 'DATA' (this default name may not be allowable on all processors).

UNIT - FORTRAN unit number for the outfile. The default value of this integer variable is 10.

DT - sample interval for the synchronized frames of the outfile (seconds). A positive nonzero value must be specified for this parameter. There is no valid default.

TTOL - time tolerance used by the synchronization algorithm (seconds). The use of this value is described in the SYNCHRONIZATION ALGORITHM section. The value must be positive and nonzero. The default is 0.0001 second.

FROM - infile numbers and data channels from which the outfile data channels are obtained. This keyword, if present, must be the last keyword of the outfile definition. If this keyword is omitted, the outfile data channels will be the concatenation of all the infile data channels in the order of the infile numbers. The program will stop with an error message if the number of outfile channels exceeds the dimension limit of 100, or if there are no outfile channels. The PROGRAM TRANSMITTAL AND IMPLEMENTATION section explains how to change the dimension limit.

Following a FROM keyword, the remaining fields in an outfile definition contain infile numbers and channel numbers delimited by blanks, commas, decimal points, dashes, or slashes. A slash delimiter denotes the last field of the definition.

A field contains an infile number if and only if it is delimited by a decimal point. All channel number fields refer to data channels of the infile specified by the most recent infile number field. Any channel number fields that precede the first infile number field refer to the lowest infile number defined.

A field delimited by a blank, a comma, a dash, or a slash is a channel number field. Each channel number field specifies the infile channel source for a single outfile channel, unless the delimiter between two channel number fields is a dash; in that event outfile channels are obtained from all infile channels between and including the two given. A dash separating a channel number field from a succeeding infile number field has no special effect. Any use of a channel number larger than the number of channels defined for the relevant infile results in an error stop. A channel number of 0 can be specified to indicate an unused outfile channel; the value of such an outfile channel will always be 0. The channel numbers can be specified in any order, except that when a range of channels is indicated by two channel numbers separated by a dash, the second number must be greater than or equal to the first. The data channels on the outfile are in the order of their definition. No infile channel

(except for the dummy channel number 0) can be used as a source for more than one outfile channel.

Examples of valid outfile definitions are shown below. Only one outfile definition can be in a single job.

```
OUTFILE, NAME = FILE1, DT = .05, FROM
     4.3-20, 1 2.6 0 4.21/
OUTF DT = .05, NAME = FILE2 FROM
     4.3 4 5-10-20 1 2.6-6 0 4.21/
OUTFILE DT = .02/
OUTFILE UNIT 11 NAME = OUT DT = .04 FROM 1-20/
```

Note that the definitions of the files named FILE1 and FILE2 have identical results except for the name.

Time Interval Definition

If the first two characters of a definition type field are TI, it is a time interval definition. A time interval definition specifies the time intervals during which available data will be written on the outfile. Data outside the requested time intervals will be ignored. If the time interval definition is omitted, the default is a single time interval from 0 to 24 hours. If a time interval definition is present, it must follow all other definitions. At most one time interval definition is allowed in a job.

Unlike other definitions, a time interval definition has specific requirements as to when a field must or must not be on a new card. This requirement allows the program to check for some obvious errors which might otherwise give unintended results.

A time interval definition consists of the definition type field, followed by an arbitrary number of time interval cards. Each time interval card must contain exactly eight fields of unsigned integer variables, delimited by blanks, dashes, colons, decimal points, or commas. The program stops with an error message if there are not exacty eight fields on a time interval card. The eight fields contain the hours, minutes, seconds, and milliseconds for the beginning and end of the interval, respectively. There are no software upper limits on the values; for instance, the number of seconds can be larger than 59.

The end time for each time interval must be greater than the corresponding beginning time, or the program will stop with an error message. The time intervals need not be in increasing order and can overlap; however, the program is more efficient if the intervals are non-overlapping and in increasing order.

A time interval definition is terminated only by the end-of-file on the input card file.

Examples of valid time interval definitions follow:

```
TIMES
7:30:0:0 - 7:39:59:999
TIME
10, 20 - 56:0 . 10 21 0 0
5 0 0 0 5 0 10 0
10 20 7.500 - 10 22 13.5
```

Note that the end of the last interval is 13 seconds and 5 milliseconds, not 13.5 seconds.

Examples

This section presents brief examples of input cards for two jobs.

Example 1. -

```
INFILE 1 CHANNELS = 20/
OUTFILE DT = .02/
```

This is the simplest possible input deck. All 20 channels of the infile are written to the outfile at 50 samples per second (0.02 second sample interval). All 20 channels use a hold-last-value algorithm. All portions of the infile from time 0 to 24 hours are used. By default, the file name and unit number for the infile are FILE1 and 1, respectively; for the outfile they are DATA and 10.

Example 2. -

```
INFILE 3 CHANNELS = 20/
INFILE 1 CHANS = 20, SKEW = .015, METHOD = .04/
SKEWS FILE 3 -.01/
OUTFILE DT = .02 FROM
     3.10-20 0 1.10-20 30 3.1/
SKEWS FILE 1: 15 .01 5 .01/
METHODS FILE 1: 15 HOLD 5 HOLD/
TIMES:
     7:30:0:0 - 7:30:10:0
     7:40:0:0 - 7:40:10:0
```

This example is somewhat more complicated. The outfile data come from channels 10 through 20 of infile 3, an unused channel, channels 10 through 20 and channel 30 of infile 1, and channel 1 of infile 3, for a total of 25 channels. The data from infile 3 are all synchronized with a hold-last-value algorithm. Channel 1 of infile 3 is skewed by -0.01 second; the remaining channels of infile 3 have no skews. Infile 1 is skewed by 0.015 second, decompressed to 0.04 second sample intervals, and then interpolated, except for channels 5 and 15, which are skewed by an additional 0.01 second and synchronized with a hold-last-value algorithm. (Channel 5 is not used for the outfile, so it is irrel evant.) The outfile contains only data in the two requested 10-second time intervals.

PRINTED OUTPUT

This section describes the printed output of the SYNC program. The first part of the discussion concerns the program's normal output; the second part concerns error messages.

Normal Output

The first page of the output begins with a header identifying it as output from the SYNC program. All of the card input information up to, but not including, the optional time interval definition is printed in the output listing. Any default values used are also listed. The program must gather all of this information and check it for completeness and consistency before the processing of the data files can begin.

The printing of a new page with the heading "SYNC. TIME INTERVALS" indicates that the input cards (except for the time interval cards) have been checked and that the processing of the data is about to begin.

Each requested time interval is printed in the listing when the processing of that interval begins. If there is no time interval definition, no message appears. The message "NO MORE TIME SEGMENTS REQUESTED" appears when it applies. The actual start time for each interval is printed. When the processing of a time interval is completed, the message "END OF REQUESTED TIME INTERVAL" is printed, along with the number of frames written to the outfile for that interval and the time of the last frame.

If an end-of-file or an error is encountered while the program is reading from any infile, the message "END-OF-FILE OR ERROR ON INFILE number" is printed and the program terminates. The termination may be normal and expected or indicative of an error, depending on circumstances.

If a time tag on an infile is much larger than the immediately preceding time tag on that infile, the program considers no data to be available between the two times. (The criterion for the allowable size of the time jump depends on program dimensions and is discussed in the section entitled SYNCHRONIZATION ALGORITHM.) In the event of a time jump, the program discontinues all output to the outfile for the unavailable time period. All outfile output is discontinued, even though data for some channels may be available on other infiles. The program prints a message to indicate the time jump. Compressed infiles can contain time tags with no associated data values in order to distinguish between time intervals where no data are available and intervals where all of the data are constant and have thus been compressed.

The message "FILES CLOSED. number TOTAL FRAMES ON OUTFILE. LAST IS time," is issued at the termination of the program. The frame count indicated is cumulative for all time intervals.

Error Messages

The error messages issued by the SYNC program are listed in alphabetical order and explained below. Lower case quantities in the message listings indicate variables.

"BUFFER SIZE INADEQUATE" (issued by OUTCHK). The value of LCBUF is too small by the criterion described in the SYNCHRONIZATION ALGORITHM section.

"CHANNEL NUMBER number IS ILLEGAL FOR INFILE number" (issued by MTHDEF, SKWDEF, or OUTDEF). A channel number is larger than the CHANS specified in the infile definition. This may indicate that CHANS was omitted from the infile definition (it defaults to 0).

"DIMENSION LIMIT DOES NOT ALLOW number CHANNELS" (issued by INFDEF). The number of channels specified for an infile exceeds the dimension limit of 100.

"DUPLICATE DEFINITION OF INFILE NUMBER" (issued by INFDEF). Two infile definitions have the same infile number.

"DUPLICATE OUTFILE DEFINITION" (issued by OUTDEF). There are two outfile definitions in the job.

"DUPLICATE USE OF INFILE CHANNEL" (issued by OUTDEF). Two different outfile channels are obtained from the same infile channel.

"END-OF-FILE OR ERROR ON INFILE number" (issued by ONEPNT). This message can indicate an expected end-of-file or an error condition.

"END OF RANGE PRECEDES START" (issued by OUTDEF). Two channel numbers separated by a dash (to indicate a range of channels) are not in increasing order.

"END TIME BEFORE START" (issued by TIMREQ). The requested end time of an interval precedes the requested start time.

"FIRST FIELD AFTER type SHOULD BE FILE; IT IS field" (issued by MTHDEF or SKWDEF). The second field of a method definition or skew definition is not FILE.

"ILLEGAL INFILE NUMBER" (issued by INFDEF). The infile number is less than 1 or greater than 5.

"ILLEGAL INPUT CHANNEL NUMBER number ON INFILE number" (issued by CHANS). The mentioned channel number returned from the infile is larger than the number of channels specified in the infile definition.

"NO INFILES DEFINED BEFORE OUTFILE WITH FROM" (issued by OUTDEF). An outfile definition contains a FROM keyword and no infiles have previously been defined.

"NO OUTPUT CHANNELS" (issued by OUTCHK). No output channels have been defined. This can occur only if no infiles were defined or if the number of channels for all of the infiles is 0 (which usually means it was omitted).

"NOT EXACTLY 8 FIELDS ON TIME CARD" (issued by TIMREQ). This message is self-explanatory.

"OUTFILE NOT DEFINED" (issued by OUTCHK). No outfile definition was in the job.

"OUTPUT DT IS ZERO OR NEGATIVE" (issued by OUTCHK). This usually indicates that DT was not specified in the outfile definition.

"PROGRAM ERROR - CIRCULAR BUFFER OVERFLOW" (issued by FILL). This is an internal program check which should not be encountered if the program is operating correctly.

"TIME JUMP ON INFILE number" (issued by TJUMP). This message indicates either an error or an expected time jump.

"TOO MANY OUTPUT CHANNELS" (issued by OUTDEF or OUTCHK). The number of output channels defined exceeds the dimension limit of 100.

"UNDEFINED INFILE NUMBER: number" (issued by MTHDEF, SKWDEF, or OUTDEF). The referenced infile number was not defined by a previous infile definition. This may mean that the infile definition is present, but that it follows the first reference to the infile.

"UNRECOGNIZED DEFINITION TYPE: field" (issued by DEFINE). The first field of a definition is not a recognized definition type.

"UNRECOGNIZED KEYWORD: field" (issued by INFDEF or OUTDEF). A keyword field does not contain a valid keyword.

In addition to the above, system error messages about the internal reads in IFMT and RFMT can result from illegally formed fields. An example would be an infile number field which contains a nonnumeric character.

FILE HANDLING

The outfile and the infiles are handled by a modular set of routines in order to facilitate modifications for handling different file structures.

The subroutine OPNINF, with entry points OPNINF, CLSINF, and REWINF, performs open, close, and rewind functions on an infile. Each of these entries has a single input argument, which is the infile number. Any other information required about the infile can be found in common blocks /INF/ and /INFC/. Subroutine RDINF reads data from an infile. RDINF has two arguments. The first is an input argument giving the infile number. The second is an output logical

variable which is set to .TRUE. by RDINF to indicate an end-of-file; otherwise, this argument must be set to .FALSE.. Unless an end-of-file is encountered, RDINF must define all of the variables in common block /INREC/ (see appendix). Each call to RDINF must return through this block a single time tag and all of the data associated with the tag. It is permitted for the number of data channels returned to be 0. Successive calls to RDINF must return strictly increasing values of the time tag, unless there is an intervening call to REWINF. The time tags need not be uniformly spaced.

The supplied versions of OPNINF, CLSINF, REWINF, and RDINF handle uncompressed, fixed length, FORTRAN unformatted files. Each record contains integer hours, minutes, seconds, and milliseconds, followed by all of the data channels.

The subroutine OPNOUT, with entry points OPNOUT and CLSOUT, opens or closes the outfile. There are no arguments. Pertinent information is found in common blocks /OUTREC/ and /NAMOUT/.

Subroutine WROUT writes a frame to the outfile; this routine also has no arguments. When WROUT is called, TIME contains a frame time in total seconds, and A contains the synchronized data for the frame. The supplied version of the subroutine WROUT writes the time and A on a standard unformatted FORTRAN file, with the time in integer hours, minutes, seconds, and milliseconds.

PROGRAM TRANSMITTAL AND IMPLEMENTATION

The following format will be used for the transmittal of the SYNC program unless explicit instructions request otherwise. The program will be sent on nine-track 800 BPI labeled tape (1600 BPI tape is also available on request). The label will be an ANSI standard label with name SYNCHR. The tape will contain ASCII coded card images (EBCDIC code is also available on request). Each card image will be a fixed length 80-character record. Records will be blocked in fixed length blocks 1200 characters in length. Each block will contain exactly 15 records with no padding; records will not span blocks.

The data on the tape are FORTRAN source code. CDC UPDATE source card images (refs. 2 and 3) can be provided instead on request. The program is complete as supplied; it contains no references to routines other than those supplied and those defined in the ANSI standard (ref. 4). The program is coded entirely in ANSI standard FORTRAN 77 full language (ref. 4). The only machine-dependent usages involve the restrictions on legitimate file names and the precision of floating point arithmetic. The ANSI standard does not specify these details.

The program consists of approximately 1300 source cards, including comments. With its current dimensions, the program executes in approximately 72,000 octal words on a CDC Cyber 70/73 with the NOS 1.4 level 518 operating system.

All array dimensions are defined with symbolic names. All dimension limit checks are performed with the same symbolic names. Therefore, in order to

change the array dimensions in a subroutine, it is only necessary to change the statements that specify the values of the symbols (the PARAMETER statements). This involves changing only a single value on a single card in the subroutine, even though several arrays might be dimensioned using the symbolic name for that value. If the program is maintained with CDC UPDATE (ref. 2) or a similar utility, dimension changes can be made by changing a single card in the entire program, rather than a card in each subroutine. The symbolic names used for program dimensions, the current values assigned to the symbols, and descriptions of the limitations involved are listed below.

MAXINF (value 5) - the maximum number of infiles.
MAXICH (value 100) - the maximum number of channels on an infile.
MAXOUT (value 100) - the maximum number of channels on the outfile.
LCBUF (value 128) - length of the circular buffer used for time-sychronization. Efficiency is improved on some computers if LCBUF is an integer power of 2, since numerous references to the buffer are made, all of them modulo LCBUF; the use of a power of 2 is not mandatory. The SYNCHRONIZATION ALGORITHM section discusses the program limitations affected by LCBUF.

The accuracy of single precision floating point variables on 32-bit computers may be inadequate for precise time-synchronization. With a typical 32-bit floating point word, the total time can be no greater than 5 hours if a time resolution of 1 millisecond is to be maintained. In some systems, the resolution may not be this good. It is therefore recommended that total time be stored in double precision on 32-bit computers. In order to use double precision for total time, every REAL statement in the program should be changed to a DOUBLE PRECISION statement. In addition, a function DFMT must be added, identical to RFMT, except that it is double precision and uses D format; the calls to RFMT to define FSKEW and DTOUT in routines INFDEF and OUTDEF should be altered to use DFMT, and DFMT should be declared double precision in these routines.

Dryden Flight Research Center
National Aeronautics and Space Administration
Edwards, California 93523
January 30, 1981

APPENDIX

SUBROUTINES AND COMMON BLOCKS

The first section of this appendix describes the purpose and general operation of the routines that constitute the SYNC program. The variables in the common blocks are described in the second section.

Subroutines

SYNC - SYNC is the main program. It first calls the subroutines that read the definitions and open the files. If the time segment number, NSEG, is 1 on return from subroutine DEFINE, there is no time interval definition; in this event, DEFINE has specified the default time interval and the call to TIMREQ must be skipped. If NSEG is not 1, TIMREQ is called to read the first time interval card.

The remainder of SYNC processes the requested time interval and then loops back to read a new time interval card. TIMREQ returns 0 for the time segment to indicate that no more time segments are requested.

DEFINE - Subroutine DEFINE reads the type fields of the definitions and calls the subroutines appropriate to the type fields to read and process the bodies of the definitions. The defaults for OUNIT and IUNIT are initialized to 0. A value of 0 for a file unit number is used as a flag to indicate that the file has not been defined. The loop from cards 33 to 50 calls appropriate routines to process the definitions. The loop is exited either on encountering an end-of-file (DELIM = 'NONE') or a time interval definition. If an end-of-file causes the exit, NSEG will be 1, to indicate that the default time interval is used; otherwise NSEG is set to 0 to indicate that the time interval requests must be read from the remaining input cards. Before returning, DEFINE calls INFCHK and OUTCHK to check the definitions for completeness and consistency.

INFDEF - Subroutine INFDEF processes an infile definition. It first reads and checks the infile number, and then sets defaults for all of the file parameters. The loop from cards 41 to 61 reads two fields containing a keyword followed by a value. The value is assigned to the appropriate variable, depending on the keyword. The loop repeats until a slash delimiter is encountered.

SKWDEF - Subroutine SKWDEF processes a skew definition. The subroutine first reads the infile number and verifies that it corresponds to a defined infile. The remainder of the subroutine processes channel numbers and skew values until a slash delimiter is encountered.

MTHDEF - Subroutine MTHDEF processes a method definition. The structure of the subroutine is the same as that of SKWDEF; it reads and verifies the infile number, and then processes the remaining fields in a loop. MTHDEF provides four possible treatments for the fields encountered in the loop. The choice of treatment depends on whether the field begins with "H" (for hold-last-value),

begins with "D" (for default), contains a decimal point (and thus a DT value), or fails all of the above criteria (and must be a channel number).

OUTDEF - Subroutine OUTDEF processes an outfile definition. The initial code checks for duplicate definitions and sets default values. The loop from cards 38 to 52 reads in the keywords and scalar values. This loop is terminated either when a slash delimiter is encountered after a value, or when the keyword 'FROM' is encountered. The rest of the subroutine is bypassed if the loop is terminated by a slash delimiter. The loop from cards 60 to 62 initializes the infile number to the smallest defined infile number. If no infile has been defined, an error message is printed. The loop from cards 64 to 101 processes the input fields, defining the channel sources until a slash delimiter is encountered. Each field is treated as an infile number if it is delimited by a decimal point, and as a channel number otherwise. The channel number fields are treated differently, depending on whether the previous field was delimited by a dash or not. If the previous delimiter was a dash, all of the channel numbers between the previous one and the new one are included. The previous channel number is stored in ICH, and the variable THRU indicates whether the previous delimiter was a dash.

INFCHK (IFILE) - Subroutine INFCHK computes the characteristics of an infile required for the processing. It computes the minimum and maximum channel skews and the read-ahead time. The read-ahead time defines how far beyond the output frame time the infile must be read to insure that all values required to complete the frame are available. The argument of INFCHK is the infile number of the infile to be processed.

OUTCHK - Subroutine OUTCHK verifies the validity of the outfile definition and completes the definition of the outfile characteristics. The subroutine first verifies that the outfile definition is present and that the sample interval is positive and nonzero. If no output channels are defined (NCHOUT is 0), the default output channels are obtained from all available input channels. If there are still no output channels, the program aborts with an error message. The loop from cards 46 to 55 obtains the outfile channel skews and synchronization algorithms from the data for the appropriate infile channel.

OPENS - Subroutine OPENS calls OPNINF and OPNOUT to open the infiles used and the outfile. It also initializes the cumulative output record count and the latest output frame time to 0.

TIMREQ - Subroutine TIMREQ reads a time interval definition card and interprets it. If an end-of-file is encountered, the time segment number, NSEG, is set to 0 to indicate that no more time intervals are requested. TIMREQ calls FIELDS to break the card image into eight fields; if the number of fields on the card is not exactly eight, an error message is printed. TIMREQ then computes the requested start and stop times in total seconds, verifying that the start time precedes the end time.

TINIT - Subroutine TINIT performs the initialization required to begin processing a new requested time interval. The record counts and time are initialized. The infiles are rewound if required, and the current-values-table is reset.

ONEPNT - Subroutine ONEPNT processes the data for one outfile frame time. For each infile, the loop from cards 32 to 52 calls RDINF to read the infile, CHANS to associate the infile data with the appropriate outfile channels, and FILL to fill the circular buffer with the synchronized data values. This process is repeated until the infile has been read far enough beyond the current outfile frame time to insure that all of the information pertinent to the current outfile frame is available. If an end-of-file is encountered, the buffer is flushed, all files are closed, and the program terminates. If a time jump is encountered, TJUMP is called to complete the processing of the current frame and to specify the time of the first frame after the jump; the subroutine then returns so that the calling routine can determine whether to continue processing on the new frame. If the infiles are all processed without encountering an end-of-file or a time jump, ONEPNT calls WRPNT to write an outfile record and increments the current time frame before returning.

CHANS(IFILE) - Subroutine CHANS determines the outfile channels to which a record of data from an infile is to be routed. The infile number is indicated by the argument. On entry to CHANS, the values in /INREC/ are all defined from the infile: TOD is the time tag; NSIGS is the number of data values at the time tag; and VALS and ICHANS contain the data values and corresponding infile channel numbers. Subroutine CHANS first adds the infile skew to the time tag, records the current skewed infile time in FTIME, and increments the infile record count. The loop from cards 30 to 41 replaces the infile channel numbers with the outfile channel numbers to which the data values are to be sent. Data values not used for the outfile are deleted from the list, and subsequent values are moved up. The length of the shortened list is put in NSIGS before the subroutine returns.

TJUMP (IFILE) - Subroutine TJUMP handles a time jump on the infile number indicated by the argument IFILE. TJUMP first calls FLUSH to write the current outfile frame, if available, from the circular buffer. The outfile frame time is then reset to a point after the time jump and a message is printed.

FILL - Subroutine FILL fills the circular buffer with synchronized data values. Data come into FILL in common block /INREC/. The loop from cards 26 to 56 processes the data values from /INREC/ one at a time. The first step in the processing is to place the data value and the associated time in VCNOW and TCNOW, moving the previous values to VCLAST and TCLAST. The channel skew is added to the time tag from /INREC/ during the move to TCNOW.

The routine then determines whether there are any outfile frames for which the synchronized data value for the channel can now be defined. TOUT is the time of the earliest outfile frame for which the synchronized data value of the channel has not been defined. If TCNOW is less than TOUT, the synchronized data value at TOUT cannot yet be defined, so the processing of the next channel begins. Otherwise, the synchronized data value is computed and placed in the circular buffer. The algorithm used to compute the synchronized values is discussed in the SYNCHRONIZATION ALGORITHM section. After placing the value in the circular buffer, the routine loops back to statement number 200 to check whether any further synchronized values can be defined for the channel.

WRPNT - Subroutine WRPNT moves a frame from the circular buffer to common block /OUTREC/ and calls WROUT to write the record. If any data value in the frame has not yet been defined, WROUT defines it using a hold-last-value algorithm. WRPNT also increments the interval record count and cumulative number of records.

FLUSH - Subroutine FLUSH flushes a record from the circular buffer if a sufficiently completed record is available. FLUSH is called after a time jump or end-of-file is detected. The criterion used in FLUSH to determine whether a completed record is available is less strict than the criterion in ONEPNT, which controls the normal processing. ONEPNT reads the infiles sufficiently far ahead so that no further information could affect the output frame being constructed. FLUSH requires only that all skewed time tags previous to the outfile frame time be read. The less stringent criterion used in FLUSH is designed to prevent the needless loss of the data in the last frame in an infile when the file is processed by SYNC.

CLOSES - Subroutine CLOSES closes the outfile and all defined infiles. It also prints the cumulative record count for the outfile.

NXTFLD (FIELD, DELIMS, DELIM) - Subroutine NXTFLD returns the next field from the card input file in the output argument FIELD. Fields longer than FIELD are truncated without comment. The input argument DELIMS is a list of delimiters used to terminate the field. The output argument DELIM is the delimiter which terminates the field. The end of a card is always considered a delimiter (fields cannot span cards). If the terminal delimiter of a field is the end of a card, DELIM is set to 'END'. If the card input file is exhausted, FIELD is set to blanks, and DELIM to 'NONE'. See subroutine FFLD for a description of the special handling of blank delimiters.

FIELDS (STRING, FLDS, DELIMS, NFMAX, NF, NEXT) - Subroutine FIELDS uses a specified list of delimiters to break a character string, STRING, into fields. The fields are returned in the output argument FLDS, which is a vector of character strings. Fields longer than the elements of FLDS are truncated without comment. The input argument DELIMS is a list of delimiters used to terminate fields; the end of the string is also used as a delimiter. Leading blanks are removed from the fields if one of the delimiters in DELIMS is a blank. The input argument, NFMAX, is the maximum number of fields to be returned, and the output argument, NF, is the actual number of fields returned; NF may be 0. The input/output argument NEXT is the character location in STRING at which the field search begins. On return, NEXT points to the start of the next nonblank field not returned in FLDS, or to LEN(STRING)+1 if no more fields are present.

FFLD (STRING, FIELD, DELIMS, DELIM, NEXT) - Subroutine FFLD returns in FIELD the first field found in the string STRING, starting at the location NEXT. Fields longer than FIELD are truncated without comment. FFLD is used by both NXTFLD and FIELDS. The input argument, DELIMS, is a list of delimiters used to terminate the field. The output argument DELIM is the delimiter which terminates the field. If the field is terminated by the end of the string, DELIM is set to 'END'. If no field is found in STRING, FIELD is set to blanks and DELIM is set to 'NONE'. The value of NEXT on return is as described for subroutine FIELDS.

Blanks receive special treatment if one of the characters in DELIMS is a blank. In this case, leading blanks are removed from the field; otherwise, the field starts at the NEXT character of STRING. If the terminal delimiter of a field is a blank (which is possible only if a blank is in DELIMS), FFLD searches for the first nonblank character after the field. If such a character exists and is one of the delimiters specified in DELIMS, this character (rather than the blank) is considered to terminate the field for purposes of defining DELIM and NEXT. The effect of this process is to allow blanks to be used freely anywhere except within a field.

IFMT(A) - Function IFMT returns the integer (I format) translation of an input character string, A. The format used is modified in execution time to reflect the length of the string A; this is necessary to handle arbitrary length strings correctly. The diagnostics of illegal strings for I format are left to the system's input/output routines.

RFMT(A) - Function RFMT returns the floating point (F format) translation of an input character string, A. The format used is modified in execution time to reflect the length of the string A; this is necessary to handle arbitrary length strings correctly. The diagnostics of illegal strings for F format are left to the system's input/output routines.

SECS(IT) - Function SECS returns the total floating point seconds translated from the integer hours, minutes, seconds, and milliseconds stored in the four-word input vector, IT.

HMSMS(T, IT) - Subroutine HMSMS converts time from total floating point seconds, T, into integer hours, minutes, seconds, and milliseconds and places the result in the four-word output vector, IT.

BOOBOO(MSG) - Subroutine BOOBOO prints an error message given as the input argument, MSG, and then stops.

OPNINF(IFILE), CLSINF(IFILE), REWINF(IFILE) - Subroutine OPNINF, with entry points OPNINF, CLSINF, and REWINF, opens, closes, and rewinds an infile. The input argument specifies the infile number. This routine may be altered to reflect different infile structures.

RDINF(IFILE,EOF) - Subroutine RDINF reads a data record from an infile. The infile number to be read is specified by the input argument, IFILE. The logical output argument, EOF, should be set to .TRUE. if an end-of-file or error is encountered; otherwise EOF should be set to .FALSE.. The information from the record read should be placed in /INREC/ unless EOF is set to .TRUE.. This routine may be altered to reflect different infile structures.

OPNOUT, CLSOUT - Subroutine OPNOUT, with entry points OPNOUT and CLSOUT, opens and closes the outfile. This routine may be altered to create different outfile structures.

WROUT - Subroutine WROUT writes a frame of data to the outfile. The data to be written are found in /OUTREC/. This routine may be altered to create different outfile structures.

Common Blocks

CHSKEW - Common block /CHSKEW/ contains the channel skews and interpolation methods for the outfile channels. These values are obtained from common block /CSKEW/ by subroutine OUTCHK.

CIRBUF - Common block /CIRBUF/ contains the circular output buffer CIRBUF and related variables. VCNOW and VCLAST are vectors of the current and previous data values from the infile for each outfile channel; TCNOW and TCLAST are the corresponding skew-corrected time tags. ITNEXT is the frame index of the outfile frame being processed. The actual index used in CIRBUF is the frame index modulo LCBUF. IDONE is a vector of frame indices that indicate the frame of the last value computed for each outfile channel.

CSKEW - Common block /CSKEW/ contains the channel skews and interpolation methods for each of the infile channels. This information is defined in routines INFDEF, SKWDEF, and MTHDEF.

INF, INFC - Common blocks /INF/ and /INFC/ contain various kinds of information about the infiles. The unit numbers (IUNIT), file skews (FSKEW), numbers of channels (NCHF), sample intervals (FDT), file names (FNAME), and file formats (FFMT) are specified by input cards. The computations of minimum and maximum channel skews (FSKWMN and FSKWMX) and read-ahead time (FRDAHD) are based on the input information. FTIME contains the latest time tag read from each file; the time tags are corrected for the file skew. NIRECS contains a count of the number of records read from each infile.

INREC - Common block /INREC/ contains the information from a single infile associated with one time tag. These data are defined by subroutine RDINF. Subroutine CHANS then deletes unused data and converts infile channel numbers to outfile channel numbers.

MAPCH - Common block /MAPCH/ contains a map for converting infile channel numbers to the corresponding outfile channel numbers. A zero value in this map means that the infile channel is not used. The map is defined in OUTDEF or OUTCHK.

OUTF, OUTFC - Common blocks /OUTF/ and /OUTFC/ contain information about the outfile. NRECS is the record count for the time interval, and NRTOT is the cumulative record count. The remaining variables are defined by subroutine OUTDEF.

OUTREC - Common block /OUTREC/ contains the data for one outfile frame. These data are picked from the circular output buffer by subroutine WRPNT.

REQTIM - Common block /REQTIM/ contains the description of a requested time interval. NSEG is the time segment number, which is set to 0 if no more time intervals are requested. STIME and ETIME are the requested start and stop times in total seconds. The information in /REQTIM/ is defined in DEFINE and TIMREQ.

TNEXT - Common block /TNEXT/ contains information that defines the time of the next outfile frame. TNEXT is the time of the frame in floating point seconds. TTOL is a time tolerance in floating point seconds. The use of TTOL is discussed in the SYNCHRONIZATION ALGORITHM section.

REFERENCES

1. FORTRAN Version 5 Reference Manual. Publication No. 60481300, Control Data Corp., c.1980.

2. UPDATE Version 1 Reference Manual. Publication No. 60449900, Control Data Corp., c.1980.

3. Maine, Richard E.: Programmer's Manual for MMLE3, A General FORTRAN Program for Maximum Likelihood Parameter Estimation. NASA TP-1690, 1981.

4. American National Standard Programming Language FORTRAN. ANSI X3.9-1978, American Nat. Stand. Inst. Inc., c.1978.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-81355		

4. Title and Subtitle	5. Report Date
USER'S MANUAL FOR SYNC, A FORTRAN PROGRAM FOR MERGING AND TIME-SYNCHRONIZING DATA	March 1981
	6. Performing Organization Code RTOP 505-43-14
7. Author(s) Richard E. Maine	8. Performing Organization Report No.
9. Performing Organization Name and Address Dryden Flight Research Center P.O. Box 273 Edwards, California 93523	10. Work Unit No.
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546	13. Type of Report and Period Covered Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

Microfiche supplement is attached.

16. Abstract

This report describes the FORTRAN 77 computer program SYNC for merging and time-synchronizing data. SYNC reads one or more input files which can contain either synchronous data frames or time-tagged data points, and which may be compressed. The program decompresses and time-synchronizes the data, correcting for any channel time skews. Interpolation and hold-last-value synchronization algorithms are available. The output from SYNC is a file of time-synchronized data frames at any requested sample rate.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Data merging Time synchronization FORTRAN	Unclassified—Unlimited STAR category: 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	26	

*For sale by the National Technical Information Service, Springfield, Virginia 22161