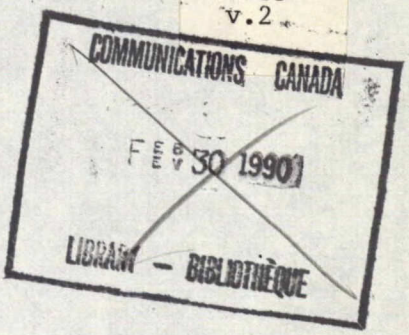
VALIDATION AND ENHANCEMENT OF TELETEXT SYSTEM SIMULATION

SOFTWARE UPDATE

MCS MILLER COMMUNICATIONS SYSTEMS LTD.



① Armstrong, R. M.

② VALIDATION AND ENHANCEMENT

OF

TELETEXT SYSTEM SIMULATION

SOFTWARE UPDATE

VERSION 0

==========

1. Introduction

1.1 Scope of Document

This document describes changes made to the TELIDON simulation software for Contract #8342, and is intended to serve as a supplement to [1]. Only those portions of the original software which have undergone modification are discussed; the user is strongly advised to consult [1] as a first resort in all cases.

Changes were made to allow full advantage to be taken of the FPS-5105 Array Processor (AP) and VAX/750 minicomputer acquired by MCS in 1984. Using this combination, the speed of the simulation was increased by a factor of fifty times, allowing many more simulations to be performed, and many more scenarios and features of the simulation to be explored. The deliverable software includes this AP version of the software, as well as a functionally identical VAX-only implementation. In all cases, code is written in accordance with FORTRAN-77 standards.

1.2 Array Processor Utilization

The Array Processor greatly increases the speed of repetitive

computations performed upon large vectors of values. For example, replacing the usual method of multiplying corresponding elements of matrices through use of a DO-loop construct with a call to the VMUL subroutine supplied in the AP math library typically cuts computation time by an order of magnitude or more. The execution time improvements associated with more complex operations, such as Fast Fourier Transforms, are even more dramatic.

The two disadvantages of the AP when compared to the VAX are (1) its inability to do disk I/O directly, and (2) the somewhat more primitive nature of the operations available on it. The first is a minor consideration; values can be transferred to and from the VAX host for output to disk, or for printing, with a minimum of difficulty. To keep execution times to a minimum, however, minimizing the number of such transfers performed is desirable. In aid of this, intermediate values calculated during the execution of the program which need not be output are left in the AP during program execution. This practice makes debugging the AP version of the program somewhat more difficult, but the increase in software development time was more than compensated for by the improved execution times noted above.

The second point forced a slight change in programming style. Vector values in the AP are referred to by the Advanced Math Library routines supplied by FPS by their base address and increment values. Throughout the AP version of the simulation these base addresses are given symbolic names (i.e. LSIGNAL refers to the location in AP memory of the base address of the SIGNAL vector). A brief summary of the Advanced Math Library routines primarily used in this simulation is provided in Table 1.

1.3 Summary of New Features

Three major functional changes, and several minor ones, have been made to the simulation. The first, and most important, is the implementation of the MK-V decoding scheme. A new bit-timing recovery scheme, and a new slicing level determination scheme, are included in this. The correlation method of bit-timing recovery has also been implemented in the new simulation, as has adaptive slicing level determination.

Among the less important alterations, a third filter file format has been included. Setting the filter file format specifier to 2 causes the filter values corresponding to each frequency to be read from the file as complex notation (i.e. the three real numbers in each record of the filter file are read as frequency in MHz and real and imaginary components of the complex gain at that frequency). Also, a separation of random number generation seeds in the main program body has been effected, so that the data sequence generated for a given random seed is unique, and unaffected by the selection of bit-timing recovery method. This change was made to allow direct comparison of simulation runs performed with different bit-timing recovery schemes.

==========

2. Changes Made to Main Program

Figure 1 shows an outline of the processing steps executed by the new version of the main program TELSIM. TELSIM is composed of three main sections, one performing
initialization, one generating and analyzing data, and a small third section to perform error counts. Of these, the second, which

generates and analyzes data, has been largely vectorized.

2.1 Initialization

In the initialization stage, which has been modified only slightly from the original program, simulation parameters are obtained from an input file specified by the user in response to a program prompt. In addition, the function QSAMP is called to initialize an array of values used in the subsequent calculation of the Q-function. A single call is made to the VAX random number generation program to supply a real random number which can be used by the AP as a seed for its own random number generator; this call is necessitated by the difference in the format of variables the two routines used as seeds. The input parameters are then echoed to the output listing file.

The next stage of initialization is the creation of the filters required by the main loop of the program. This generation is done in the VAX, since there is often a requirement to read some or all of these filters from files. A new routine, NORM, is used in this stage to normalize filter values, a step necessary to ensure the correctness of signal to noise ratio calculations (see [2]). Another new subroutine, BTRINIT, is also called at this stage. This routine generates the frequency domain representation of a bandpass filter to be used in the NK-V and Correlation approach to bit-timing recovery. The parameters for the impulse invariant approximation to a second-order Butterworth filter are also determined in this routine.

Calculation of gain and SNR parameters, as well as the creation of the invariant synchronization portion of the Telidon waveform, is done next. Once the gain and SNR values are known, the theoretical slicing level for the simulation parameters supplied is calculated; changes were made in this calculation to further generalize it.

After the error counter variables (KNT0 - KNT8) have been set to zero, constants required by calculations which are to be performed in the AP are stored in the arrays REALS and INTGS. While it is possible to move data from the VAX host to the Array Processor a single variable at a time, a considerable improvement in speed is obtained if a single DMA request is made, so these constants are stored in arrays which can then be moved in a single call. Data stored in arrays which are transferred to the AP using the APPUT subroutine are stored in the Array Processor sequentially in the same order as in the host. Thus, although the particular ordering of values in REALS and INTGS is unimportant, this order must correspond to the locations set in the memory table (see below). The preamble bit sequence is set in this section as well.

To improve the clarity of the program, memory locations in the AP are referred to by symbolic mnemonic names. The values associated with these mnemonics are set during this initialization stage, and are not changed during subsequent program execution. As AP memory is divided into pages, separate memory maps are required for each page (see the internal documentation of the code). Care must be taken when changing the size of any of the arrays, or the location or number of the variables, used in the AP.

Once these memory maps have been created, the new subroutine REVERS is called to rearrange the values stored in the filter arrays in the host. This is necessitated by the differing input formats expected by the VAX and AP FFT routines. Constants and filter values are moved into the AP, and filters combined to minimize subsequent calculations.

2.2 Signal Generation and Analysis

The main loop of the program generates, filters, adds noise to, and analyzes the Telidon signal. The number of times this loop is executed is set by the user in the input parameter file; each loop execution creates one burst of Telidon data. The Array Processor is used to perform all operations in this sequence of calculations except the generation of impulsive noise.

The first step is the generation of random data bits. This is performed by generated a sequence of random numbers on [0.,1.], subtracting 0.5 (so that the numbers lie in [-0.5,0.5]), and then applying a limiting function to create a sequence values which are either +1.0 or -1.0. Parity bit generation is done by successively multiplying the first, second, third, ..., seventh bits of each simulated byte of the data sequence. Since vector operations in the AP are done by specifying the initial address of the data to be operated on, and the increment separating successive vector elements, this parity generation is quite simply done by moving every seventh 'bit' (in actuality stored as a word in the AP) into the position corresponding to the eighth 'bit', and then successively multiplying the sixth, fifth, ..., first bits against this value to create the parity values.

Once the bit sequence has been generated, it is sampled and scaled to produce the Telidon line signal, then transformed into the frequency domain. Pulse shape filtering is applied by multiplying the frequency-domain pulse filter against the signal values, after which synchronization and colour burst information is added. Transmission filtering and multipath filtering are applied simultaneously, these two filters having previously been compressed into one (see above).

If the type of noise selected was not GAUSSIAN, impulsive noise is generated in the VAX host, and these values put in the Array Processor. Since the generation of impulse noise is not amenable to vectorization, it was felt that its generation would benefit only marginally from being done in the AP.

Gaussian noise is then generated, using the Array Processor. To do this, random numbers with a uniform distribution are first created, and then transformed to produce values with a Gaussian random distribution with specified mean and variance. These noise values are added to the impulse noise values (if any were introduced), and the result is added to the frequency-domain signal.

The signal is then passed through the receiver and receiver pulse shaping filters (which have previously been combined into one). If the bit-timing recovery type selected by the user is CORREL or MK-V, the operations involved in this bit-timing recovery are then performed. The signal is then transformed to the time domain and scaled, and the imaginary part of each signal sample is zeroed. These results are then returned to the host for further processing.

The final part of the main loop is executed in the VAX. In it, data estimation is performed, following which the data sequence is extracted from the received signal, decoded, and passed through a parity checking routine. Finally, the error count for this burst is obtained.

2.3 Final Output

Once the data generation and analysis loop has been executed the number of times specified, the routine OUTRES is called to place summary error counts in the output listing file. This routine has not been changed.

==========

3. Changes Made to Subroutines

The following subroutines have been altered in whole or in part from their original form.

3.1 BTR

The bit-timing recovery routine has been modified to allow MK-V decoder bit-timing recovery to be performed, while still allowing MK-IV bit-timing recovery to be simulated. If the 'MK-V' option was selected in the input parameter file, the 6th positive to negative zero crossing in the data sequence is located and used to characterize the position of the first bit.

3.2 MULGEN

The previous version of MULGEN performed calculations on values which were out of the band of frequencies of interest. The upper and lower bounds on the main loop of this routine have been altered so that only those calculations having an effect upon program execution are actually performed.

3.3 RECOVERSEQ

This routine takes the word/bit timing mark from the WORDSYNC routine (described below) and uses it to recover the transmitted sequence. This routine has been modified to perform adaptive slicing, if the user has so requested.

3.4 RDFILT

The RDFILT routine, which reads filter files from disk and performs interpolation to create a full-sized array of values representing that frequency-domain filter. The previous version of the routine allowed the user to specify as filter file formats either format 0, in which filter values were given as gain magnitude and phase, or format 1, in which filter values were given as gain magnitude and group delay. The present version also permits format 2, in which filter values are given as complex numbers.

3.5 RECEIV

The RECEIV routine is an interface between the main program and the

bit-timing recovery routine RECOVR. In keeping with the changes made to this latter (described below), the parameters passed into and out of RECEIV have been altered.

3.6 RECOVR

RECOVR has been altered to permit the user to select Correlation-type bit-timing recovery. In addition, the code to handle the case in which correlation-type bit-timing recovery is not selected has been altered to accommodate MK-V bit-timing recovery.

3.7 SLICER

This routine determines the slicing level for the received Telidon signal. This can be determined in the routine either through a calculation of the average signal level over a number of bits, or through use of an approximation to the method implemented in real terminals, which uses peak detection. This routine has been altered to accommodate the 'ADAPTIVE' and 'MK-V' slicing options, as well as the 'AVERAGING' and 'IDEAL' options available previously.

3.8 WORDSYNC

This routine determines the word synchronization. The bit-timing recovery routine supplies a timing reference, which should be in the center of the first bit in the preamble. If this is the case, the simplified 'add offset' method would work, and this option is provided for simplicity in testing. However, if the sequence has been shifted in time, the word synchronization byte must be used. This routine simulates the real decoder by comparing the word-sync sequence to the original sequence to determine the most probable synchronization point.

The new version of WORDSYNC also calculates the initial sample and hold values for adaptive slicing.

3.9 ZCROSS

This routine determines the position of the zero crossings during the preamble. These measurements are used by the BTR to determine the sampling isntants. The crossing point is approximated by a linear interpolation between points on opposite sides of the slicing level. These crossing points are stored in the array CROSS, with the direction of the crossing indicated by the sign of the timing mark. This routine has been modified to allow processing in accordance with MK-V decoder specifications.

==========

4. New Subroutines

The following new subroutines have been created, and added to existing software.

4.1 NORM

This routine takes as input a complex array of values and a scaling factor by which each value is to be multiplied. This routine is used to scale each filter array by the value of its center element (i.e., by its value at a frequency of 0 Hz), to facilitate calculation of signal to noise ratios.

4.2 BTRINIT

This subroutine generates the frequency domain representation of a bandpass filter used in the MK-V and Correlator approaches to bit-timing recovery. The parameters for the impulse invariant approximation to a second-order Butterworth filter are also determined in this routine. These IIR parameters are determinable for 'ADAPTIVE' and 'MK-V' slicing types.

4.3 CORBTR

This is a new bit-timing recovery routine which uses the correlation method of bit-timing recovery. The first 14 bits of the preamble are used to estimate clock phase; the routine must choose between one of 5 possible clock phases. Because these first bits of the preamble are alternating +1/-1, they will be correlated with a clock running at twice the bit rate. This routine assumes that there are 11 samples per bit, and that the possible clock phases are offsets of 0, 2, 4, 6, or 8 samples.

As input, this routine requires the windowed and amplitude limited portion of the synchronization signal that was passed through the bandpass filter. As output, it supplies the estimated sampling point of the first bit in the synchronization signal.

4.4 SLFILT

This function calculates the impulse invariant digital approximation to a 2nd order Butterworth filter, using the values passed in the input arrays FSTATE and SFILPAR.

4.5 REVERS

This simple routine is required because of the different input formats expected by the VAX FFT routine inherited from the previous software and the AP FFT routine supplied by Floating Point Systems. REVERS simply rearranges the frequency-domain terms of the array passed as input, so that terms are strictly ordered by increasing frequency value.

==========

5. Support Programs

The following programs are external to the main simulation program, and were written to aid in the analysis and display of results.

5.1 SUPERPLOT

5.1.1 Introduction

The SUPERPLOT routine is a utility allowing the user to interace with an HP 4740. Parameter are set interactively for :

- Number of ticks/subticks on axes
- Length of axes
- Position of axes relative to the edge of the paper
- Lower and higher boundaries for each axis
- Linear or Log scales
- Presence of a grid on the graph
- Title for the graph

This routine assumes default values for each option so that simple graphs can be created quickly.

Upon invoking the routine, the user is prompted for the name of a master file containing the data to be plotted. Once this has been obtained, the session is decomposed into two parts :

- selection of the first series of parameters :
  - axes lengths
  - axes position
  - type of axes
  - etc...
- 2 selection of the second series of parameters :
  - introduce title and position title
  - select sub-ticks numeration for LOG axes
  - etc...

One may optionally save the parameters chosen in a file, and use these parameters on a subsequent invocation of the plotting routine.

5.1.2 Using the Routine

Once the routine has been called, a screen appears and the routine prompts for the name of the data file. A new screen then appears on which is shown the current values of the parameters. Shaded areas are comments describing the purpose of each parameter, while white-on-black text shows the values of the parameters.

To change a parameter, the user must enter the number designating that parameter. This number appears in the shaded text associated with each parameter. Parameters relative to the X-axis are shown in all cases on the left side of the display, and those relevant to the Y-axis on the right side of the display.

Certain parameter values can only assume certain pre-selected values (eg. axis type can be either LINEAR or LOGARITHMIC), so selecting one of these parameters for alteration changes the parameter from its present value to the other option. When a labelling option is selected, a window is created, in which the user must enter the text which is to serve as the label.

When new values are to be entered, the lower right part of the screen may present some information regarding acceptable values, depending

upon which parameter is being changed. Help can be obtained while working with the first screen by typing 99 as an option, and on the second screen by typing 9. The screen can be refreshed by typing Ctrl-Z at any time.

Once the first-screen parameter values are acceptable, the screen may be exited by typing '0' (which brings the second screen up), or '-1', which aborts the program. The second screen can be exited by typing 'E' or 'B'.

5.1.3 General Options

Recovery of parameters (option 1)

If you type 1 as option you will be prompted for a name of a file. This file is supposed to contain the values of plotting parameters from a previous session (the method of saving parameters is described below). To cancel this option after it has been entered, simply type <Return> instead of a filename.

Add or Delete a set of data (Option 2)

This option allows the display of more than one set of data on the same set of axes. The routine will prompt for the name of a file, and will return an error message if the file named does not exist. Up to 7 sets of data may be drawn on a single set of axes, each in one of 7 different line types. To delete a set of data that has been added to the list of data to be plotted enter '*' instead of the name of a file; this will cancel the most recently entered data set.

Grid Lines (Option 3)

When active this option causes the program to draw a grid of lines on the graph, corresponding to intermediate values between the maximum and minimum values presented. The default is 'No Grid'.

Side and Top axes (Option 4)

When this option is 'ON' an enclosing box is drawn around the graph. If 'OFF', only the left and bottom axes will be drawn. The default option is 'ON'.

Position of the graph on the sheet (Option 5)

These values specify the starting location of the axes on the graph. Modifying them will move the graph on the page. Values are expressed in centimeters. The default starting position is (5.0,3.0).

Sizes of the axes (Option 6)

To modify the size of the axes, select option 6 and then modify the length for each axis (length is expressed in centimeters). Default values are 20 cm. for the X-axis and 12 cm. for the Y-axis.

Line type (Option 7)

Seven differnt line types are available with which to plot data. At any time, the line type indicated is the one being used for the present set of data. The default value is 7 (solid line).

Draw a symbol at each point (Option 8)

If this option is active a character symbol is used to mark each data point on the graph. This option is only valid when the number of data points is less than 50.

Pen (Option 9)

The HP plotter uses two pens, which can be of different colors. Using this option allows the user to specify which pen, and thus which color, is to be used for the graph. The default value is Pen #2.

Scaling the data (Option 50)

This option allows the user to specify a linear scaling and constant offset to be applied to the data (i.e., the points actually plotted are a linear function of the data points read from the file). This allows the user to plot data lying in the range 300.01 to 300.02 on axes marked from 0. to 10., if this is desired.

5.1.4 Options Relevant to Axes

Label (X : Option 20 ; Y : Option 40)

These options allow the user to enter text beside the axes. A string of up to 32 characters may be entered; this string is centered when output. Default values :

X : X
Y : Y

Character size (X : Option 21 ; Y : Option 41)

The character size of each label may be defined independantly. Upon entering this option, the user is prompted for a height and width. Values entered are taken to be in centimeters. A recommended width is 0.7*heigth). Default values :

Xwidth : 0.186 Xheigth : 0.280
Ywidth : 0.186 Yheigth : 0.280

Number of subticks (X : Option 22 ; Y : Option 42)

The number of marks on each axes can be set using these parameters. This option applies only to axes which are linear; logarithmic axes are marked automatically. Default values are :

X : 0
Y : 0

Ticks In / Out (X : Option 23 ; Y : Option 43)

This option determines whether ticks appear inside or outside the enclosing box drawn around the graph. Default values are :

X : Ticks OUT
Y : Ticks OUT

Change type of the axis (X : Option 24 ; Y : Option 44)

Selecting this option changes the axes type from LINEAR to LOG, and vice versa. If negative values exist in the data set and the user attempts to set that axis to LOG, an error message appears.
Default values :

Changes X into LOG
Changes Y into LOG

Clipping (X : Option 25 ; Y : Option 45)

If the user wants just a part of the data to be drawn, the boundaries can be restricted by choosing new values for the lowe and higher bounds. As well, the domain can be made larger without restriction. Default values are determined by the data.

Increment (X : Option 26 ; Y : Option 46)

This determines the spacing between values displayed on the axes. If no value is selected, the program automatically supplies a value resulting in approximately ten equally-spaced ticks.

5.1.5 Final Options

Introduce a title (Option I)

This option allows the user to enter a four-line long title to accompany the graph. While this is being done, the facilities described below may be used.

Title Down / Up (Option T)

If a title is entered, selecting this option results in the first line of the title being placed above the graph, and the final three (if any) being placed below the graph. Selecting 'DOWN' results in all text being placed below the graph.

X - LOG subticks numbered (Option X)

If the X Axis is logarithmic the user may choose to number the subticks from 1 to 9 on the graph. Default is no numeration

Y _ LOG subticks numbered (Option Y)

See above option X

S - Save the parameters

If you want to save the values defined during the first screen, selecting this option will result in a prompt for a file name into which the parameters will be placed.

5.1.6 Titling Options

While the title is being entered, the user may wish to put the name of a file or the data on the graph, or to put the names of all the files being plotted, along with the line types in which they are being plotted, on the graph. To do so, type :

$D or $d	: This sequence will be replaced by the date
$Tn or $tn	: This sequence will be replaced by the line type of the set number n
$Fn or $fn	: This sequence will be replaced by the name of the file which contains the set number n
$S or $s	: This sequence will be replaced by the equation of the scaling.
$n	: This sequence has for effect of storing the text from the position n on the line (If there is any

text between position 1 and n this text is kept)

$E or $e	: This sequence erases the whole text of a line
Control Z	: Marks the end of the introduction of text
Return	: Marks there is no modification to be done on the current line

---

5.2 CODING

This program is equivalent to the MCODING program created for the previous contract; all input and output, as well as all operations performed, are identical with the previous version.

5.2.1 ERRGEN

This program generates independent error sequences at a specified input bit error rate. The input parameter file is by default FOR004; a redefinition of this logical unit number is necessary if a different input parameter file is to be used.

The input parameter file must contain the following four parameters :

(1) Name of the output file (extension .PAR is supplied by program)
(2) Desired probability of error
(3) Seed for random number generator (integer)
(4) Number of packets to be generated

The file generated by this program can be analyzed directly by CODING. ERRGEN was developed to aid in the verification of the simulation's correctness. ERRGEN requires little execution time, and thus may be reasonably operated at low error rates.

5.3 COMPRESS

This program takes error sequences generated by the TELSIM program and places them in a compressed format for use with the CODING program. When prompted, the user must supply the name of an error sequence file to be processed. The program continues processing input files until the user specifies a null file (does not enter a name). The program prompts the user for the name of the output file into which the compressed sequences are to be written.

5.4 TAPEAREAD

This program reads error sequence files on tape generated at CRC and outputs the data stored in them in a compressed format suitable for use with the CODING program. ****

5.5 SUPPLY

*****

5.6 MC

This program separates the effects of word synchronization loss on the error rate expressed by a TELSIM error listing file. The program also averages error results from different simulation runs performed with the same data sequence to create 95% confidence intervals for the actual error rates. The number of packets lost is also calculated and output to the user.

The program first prompts the user for the portion of the data file name which all the files to be processed have in common; this is typically the first part of the prefix portion of the file name. The user is then prompted for the TV signal to noise ratio; this value is output to the output files created.

The program then reads all the files possessing the common portion of the file name supplied by the user and processes them. Three output files are generated : one containing the results of the interactive session, a second (PLOT1.DAT) and third (PLOT2.DAT) containing the SNR and error rate before separation as pairs (for use with the plotting program), and the SNR and error rate after separation in the same format, respectively.

The program repeats this cycle of processing until the user specifies a null file name (does not enter a string before typing <RETURN>).

5.7 EYETEL

EYETEL is a modified version of the TELSIM program used to create eye diagram data files. A single noiseless Telidon signal burst is generated in accordance with user-specified parameters, and this burst used to construct a file <runid>.SAV, where <runid> is the run identifier specified in the input parameters. This .SAV is then used by the MAKEYE program (below) to create a plottable eye diagram program.

The data from which the eye diagram is to be generated is taken from the single noiseless burst generated by EYETEL. A series of samples corresponding to two data bits is taken from a pre-selected point in the data stream and converted to plottable form. The samples corresponding to successive pairs of bits are taken and superimposed upon this, and these values written to the output file.

EYETEL reads its input parameters from a file identical to those used by the main program TELSIM, so that the user may use the same parameter file with either program.

5.8 MAKEYE

MAKEYE reads the files created by the EYETEL program and uses the data they contain to create output files which, when plotted, create an eye diagram. When prompted by the program, the user must enter the name of the data file to be used as input; this will be of the form <runid>.SAV, where <runid> is the run identifier that was specified by the user in the parameter file input to the EYETEL program. The output file of MAKEYE will then be titled <runid>.EYE, and may be submitted directly to the HP7470 plotter.

5.9 FILGEN

FILGEN is similar functionally to the $FILE-GENERATOR routine supplied for the previous contract. Written in DCL (Digital Command Language), this routine aids the user in creating simulation parameter files for series of simulations.

FILGEN is invoked by typing @FILGEN; optionally, a parameter name may be specified on the same line, in which case jobs are submitted to the batch queue named. FILGEN's user interface is identical with that of the original parameter file generation routine; the only difference in output is that digits particularizing the parameter files are not separated by underscore characters ('_') as were the digits in the filenames generated by the previous program, since the underscore character is not recognized by the VMS operating system as a legal character in a filename definition.

In conjunction with FILGEN's use, a second command file called SYNC.COM must be present. This parameter file is used to control synchronization of job execution, and is needed when multiple batch queues and multiple jobs are being used. Table 2 contains an example of such a command file.

TABLE 1 : FPS ADVANCED MATH LIBRARY SUBROUTINES USED

ROUTINE	RESULT
APGET	Gets a vector from the AP memory locations specified
APPSEL	Selects which page in AP memory subsequency operations will be performed on
APPUT	Puts a vector into the AP memory locations specified
APWD/APWR/ APWAIT	Timing routine required to ensure synchronization of processing between host and AP
EVMOV	Moves a vector from one page of memory to another
CFFT	Performs forward or inverse FFT on the complex vector specified
CFFTSC	Performs scaling on a complex vector after FFT operations
CVADD	Adds the two complex vectors specified
CVMUL	Multiplies the two complex vectors specified
RECT	Converts the complex vector values specified to rectangular form
VCLR	Fills the vector specified with zero values
VFIX	Converts the real vector specified to integer
VFLT	Converts the integer array specified to real values

VLIM	Clips a vector to specified upper and lower bounds
VLN	Performs the natural logarithm operation on the vector specified
VMOV	Moves a vector from one location within a page to another location in the same page
VMUL	Multiplies the two real vectors specified
VNEG	Negates the vector specified
VRAND	Generates a vector of random numbers uniformly distributed on [0.,1.]
VREAL	Extracts real parts of a complex vector
VSMSA	Multiplies vector elements by one scalar and adds a second scalar to each value (linear transformation).
VSMUL	Multiplies vector elements by a scalar
VSQRT	Performs the square root operation on the vector specified
VSUB	Subtracts one real vector from another

==========

TABLE 2 : Example of SYNC.COM

```
$ ! THIS JOB IS DONE TO INITIALIZE THE SUBMITTING OF FILGEN
$ WAIT 0:02
$ EOJ
```

ARMSTRONG, R.M.
--Validation and enhancement of teletext system simulation