# 752-ADC
# Analog to Digital Converter
## Operators Manual

TransEra

TABLE OF CONTENTS



* These Routines are not available on the 652-ADC.

# GENERAL DESCRIPTION

## Introduction

The 652/752-ADC is a 16 channel differential input A/D converter with 12 bits (4096 parts) of resolution over the input voltage range. There are 3 ranges which can be selected. One is with unity gain and provides a range of -10 volts to +10 volts. The second range is with a gain of 4 and goes from -2.5 volts to +2.5 volts. The third range has a gain of 32 and goes from -0.3 volts to +0.3 volts. Resolution with unity gain is about 5 millivolts. With a gain of 4 the resolution is about 1.2 millivolts. With a gain of 32 the resolution if about 0.16 millivolts.

The 16 differential channels are multiplexed through an instrumentation amplifier to a sample and hold circuit and then to the A/D converter. The converter requires about 48 microseconds to resolve 12 bits but can be short cycled at about 31 microseconds for 8 bits. The maximum sample rates for 12 bits and 8 bits are found on page 1-3.

The 652/752-ADC provides a direct interface with the 4050 microprocessor which allows direct memory storage as well as machine coded programs in ROM to control the converter circuitry. This greatly simplifies operation of the converter for the user. A BASIC CALL statement is all that is required to initiate any data acquisition sequence. There are a total of 18 routines that are programmed into the ROM for operating the converter or manipulating or displaying the data taken by the converter. Depending on the routine used there are a number of parameters that can be passed in the CALL statement for specifying the desired number of samples, sample rate, channel, trigger mode, input voltage range voltage level to trigger on, and the number of samples to save prior to the trigger. The converter can also perform autoranging and averaging.

The 652/752-ADC is thus a very versatile and powerful data acquisition system requiring a minimum of user programming. The various CALL routines will be explained in detail in Section 2.

## Installation Instructions

The power to the 4050 should be turned off before the 652/752-ADC is installed. After the power is shut off, the 652/752-ADC may be inserted into a slot in the firmware backpack or into a slot of a ROM Expander Unit. Press down gently until the edge card connector is seated in the receptacle connector.

An edge card connector plus a length of ribbon cable has been supplied with each unit for connecting the top edge card to the input voltages.

The 16 channel connector panel attaches to the 652/752-ADC ROM Pack via the edge card connector on the end of the ribbon cable. When putting the connector onto the 652/752-ADC ROM Pack, care must be taken to assure that the connector is positioned correctly. For proper positioning the colored tabs on the ROM Pack and the connector should be lined up as the connector is pressed onto the ROM Pack.

The pin assignments to the 16 channels on this connector can be found in Appendix B. Note that for single ended inputs (positive only inputs), the low and the ground lines should be tied together.

Specifications

| | |
|---|---|
| 652 Maximum Sample Rate (8 bits) | 32,050 sps |
| 652 Maximum Sample Rate (12 bits) | 20,833 sps |
| 752 Maximum Sample Rate (8 bits) | 28,100 sps |
| 752 Maximum Sample Rate (12 bits) | 21,000 sps |
| Sample Period Resolution | 8 microseconds |
| Voltage Range 1 (unity gain) | 10 volts |
| Voltage Range 2 (gain of 4) | 2.5 volts |
| Voltage Range 3 (gain of 32) | 0.3 volts |
| Maximum Number of Samples | Memory Available in 4050 |
| Number of Channels | 16 |
| Trigger Input | 1 TTL Load |
| Minimum Trigger Pulse Width | 20 microseconds |
| Resolution at unity gain | 5 millivolts |
| Resolution at gain of 4 | 1.2 millivolts |
| Resolution at gain of 32 | 0.16 millivolts |
| Linearity | .05% |
| Accuracy | .05% |
| Quantizing error | 1 bit |
| CMR DC-60 Hz | 70 dB |
| Input Impedance | 1M Ohms |

## String Data Format

Many of the 752-ADC routines use the following binary string data format. These include BURST, QSCAN, SCAN, UNPAK2, UNPACK, PACK, MIN$, MAX$, UNMESH, and PLOT$.

8 bit data is stored in the string one character per sample. All eight bits of the character are used. The first sample is put in the first position in the string, and so on.

12 bit data is stored in the string two characters per sample. The data is left justified. The first byte used for each sample contains the most significant 8 bits of the sample. The second byte used for each sample contains the least 4 bits of the sample in the character's most significant 4 bits. The least significant 4 bits of the second character are zero.

# A/D Converter Routines

## Introduction

There are 6 A/D converter routines in the 752-ADC and 5 in the 652-ADC. They are accessed through the CALL statement in BASIC. They are written in assembly language for the greatest possible speed of execution.

The 12BIT and 8BIT routines set up the mode of operation for the converter. This mode of operation is used by the BURST, QSCAN, and SCAN routines.

The simplest routine to use is the A/D routine which does autoranging and averaging before passing back one result from the specified channel or series of channels.

For taking multiple samples, three routines have been provided which allow a selectable sample rate, trigger mode range, and channel or channel sequence. These routines are called BURST, QSCAN, and SCAN. BURST is used for single channel sampling only so that the maximum sample rate can be achieved. Qscan is used for the fastest possible multi-channel sampling. QSCAN is available only on the 752ADC. SCAN allows any sequence of channels to be specified and allows triggering on the level of an input signal.

When performing data acquisition with BURST, QSCAN, or SCAN, any interrupts to the 4050 processor will cause inaccurate timing. This includes typing commands from the keyboard during data acquisition.

In this section, the A/D converter routines in the 652/752-ADC are explained. The format of the CALL statement is given, and each of the calling parameters is explained. Then, the general operation of the routine is explained.

In the listing of the calling parameters, the letters O and I following the parameter name indicate whether the parameter is used as an input to the routine, an output from the routine, or both. In general, input parameters may be variables, expressions, or constants. Output parameters must be variables. As far as possible, error checking is done by the routine to insure correct typing of the calling parameters.

12BIT and 8BIT

Format: CALL "12BIT"

CALL "8BIT"

12BIT sets the A/D converter to 12 bit mode. 8BIT sets the A/D converter to 8 bit mode. This mode is used in BURST, QSCAN, ~~FSCAN,~~ and SCAN. If neither 8BIT nor 12BIT is called, 12 bit is assumed. 8BIT should be called immediately prior to calling the sampling routine, because some other operations in the computer may cause the mode to be inadvertently set to 12 bit.

FSCAN Medical graphics

## A/D (Analog to Digital)

Format: CALL "A/D",A,C

A:O TARGET SCALAR
C:I CHANNEL NUMBER

CALL "A/D",A1

A1:O TARGET ARRAY (SIZE DETERMINES CHANNELS READ)

A/D reads either one channel or several, depending on the parameters passed. If a scalar expression and a scalar variable are passed, only one channel is read. If an array is passed, channels are read sequentially starting with channel 1 until the array is filled. For example, if A1 is dimensioned to 4 elements, channels 1 through 4 will be read. A/D performs autoranging and averaging on the converted values. 16 samples are taken and averaged for each conversion. The value returned is the actual voltage on the inputs.

Examples:

```
100 FOR C=1 TO 16
110 CALL "A/D",A,C
120 PRINT "CHANNEL ";C;": ";A;" V"
130 NEXT C
```

NOT STORED PRINTS VOLTAGE ONLY

```
100 DIM A(16)
110 A=0
120 CALL "A/D",A
130 FOR C=1 TO 16
140 PRINT "CHANNEL ";C;": ";A(C);" V"
150 NEXT C
```

STORED IN ARRAY

The first example reads each channel into the scalar A and prints the voltage. The second example reads all channels into the array A and prints the voltages.

SINGLE CHAN.

BURST

Format: CALL "BURST",A$,N,P,C,T,R

```
A$:O  TARGET STRING
 N:I  NUMBER OF SAMPLES TO TAKE
 P:I  PERIOD BETWEEN SAMPLES (SECONDS)
 C:I  CHANNEL NUMBER
 T:I  TRIGGER MODE
 R:I  RANGE (1, 4, or 32)
```

BURST samples data from a single channel. A$ is the target string. It must be dimensioned to at least N bytes in 8 bit mode and at least 2*N bytes in 12 bit mode. N is the number of samples to take. P is the period in seconds. Resolution at high sample rates is about 8 microseconds. A table is shown below of measured sample rates above 5000 sps available with BURST. C is the channel number to take data from.

T is the trigger mode. Legal trigger modes are 0, 1, 2, 3, -1, -2, and -3. Period P is not in force with trigger modes 3 and -3. The maximum rate is about 12 kHz in this mode.

| Trigger Mode | Behavior |
|---|---|
| 0 | Begins sampling immediately. |
| 1 | Waits for high TTL trigger. |
| -1 | Waits for low TTL trigger. |
| 2 | Waits for positive input voltage. |
| -2 | Waits for negative input voltage. |
| 3 | Waits for high TTL trigger each sample. |
| -3 | Waits for low TTL trigger each sample. |

R is the range specifier. R should have a value of 1, 4, or 32. This is the gain used by the instrumentation amplifier. If R equals 1, the input voltage range is 10 volts. If R equals 4, the input voltage range is 2.5 volts. If R equals 32, the input voltage range is 0.3 volts.

An alternative to using a string variable as the target for the data is the TransEra Auxilliary Memory. To use the Auxilliary Memory as the target, replace A$ in the CALL statement with a numeric expression. The data will be placed in the Auxilliary Memory in unformatted mode starting at this address.

AVAILABLE HIGH SPEED BURST PERIOD TABLE:

| 8-BIT | 12-BIT |
|---|---|
| 4.923 kHz | 4.923 kHz |
| 5.088 kHz | 5.078 kHz |
| 5.267 kHz | 5.258 kHz |
| 5.443 kHz | 5.438 kHz |
| 5.647 kHz | 5.647 kHz |
| 5.849 kHz | 5.845 kHz |
| 6.073 kHz | 6.067 kHz |
| 6.318 kHz | 6.295 kHz |
| 6.583 kHz | 6.575 kHz |
| 6.857 kHz | 6.857 kHz |
| 7.188 kHz | 7.168 kHz |
| 7.529 kHz | 7.521 kHz |
| 7.913 kHz | 7.899 kHz |
| 8.348 kHz | 8.322 kHz |
| 8.808 kHz | 8.727 kHz |
| 9.324 kHz | 9.290 kHz |
| 9.907 kHz | 9.891 kHz |
| 10.667 kHz | 10.613 kHz |
| 11.313 kHz | 11.294 kHz |
| 12.225 kHz | 12.208 kHz |
| 13.091 kHz | 13.049 kHz |
| 13.583 kHz | 13.714 kHz |
| 14.769 kHz | 14.933 kHz |
| 16.369 kHz | 16.552 kHz |
| 18.286 kHz | 18.495 kHz |
| 20.721 kHz | 21.000 kHz |
| 24.000 kHz | |
| 28.127 kHz | |

Example:

```
100 DIM A$(1000)
110 CALL "BURST",A$,500,1.0E-3,1,0,1
120 WINDOW 1,500,0,65535
130 CALL "PLOT$",A$,32,1,1,500,2
```

This example takes 500 samples from channel 1 at 1000 samples per second. The voltage range is 10 volts and sampling begins immediately. The samples are plotted to the screen.

QSCAN (Quick Scan)

Format: CALL "QSCAN",A$,N,C,T,R,L

A$:O TARGET STRING
N:I NUMBER OF SAMPLES TO TAKE
C:I NUMBER OF CHANNELS TO SCAN (1-C)
T:I TRIGGER MODE
R:I RANGE (1, 4, or 32)
L:I VOLTAGE TO USE FOR TRIGGER MODE 2

QSCAN samples multiple channels as quickly as possible. A$ receives the data. It must be dimensioned to at least N bytes in 8 bit mode and at least 2*N bytes in 12 bit mode. N is the total number of samples to put into A$. C is the number of channels to be scanned. Channels are scanned sequentially starting with channel 1.

QSCAN is available only on the 752-ADC.

T is the trigger mode. Legal trigger modes are 0, 1, 2, 3, -1, -2, and -3.

| Trigger Mode | Behavior |
|---|---|
| 0 | Begins sampling immediately. |
| 1 | Waits for high TTL trigger. |
| -1 | Waits for low TTL trigger. |
| 2 | Waits for voltage above L. |
| -2 | Waits for voltage below L. |
| 3 | Waits for high TTL trigger each channel scan. |
| -3 | Waits for low TTL trigger each channel scan. |

R is the range specifier. R should have a value of 1, 4, or 32. This is the gain used by the instrumentation amplifier. If R equals 1, the input voltage range is 10 volts. If R equals 4, the input voltage range is 2.5 volts. If R equals 32, the input voltage range is 0.3 volts.

L is the voltage level to use for triggering with trigger mode 2. It must still be present in the CALL statement even when using other trigger modes.

An alternative to using a string variable as the target for the data is the TransEra Auxilliary Memory. To use the Auxilliary Memory as the target, replace A$ in the CALL statement with a numeric expression. The data will be placed in the Auxilliary Memory in unformatted mode starting at this address.

Example:

```
100 DIM A$(1000)
110 CALL "QSCAN",A$,500,2,0,1,0
120 WINDOW 1,250,0,65536
130 FOR X=1 TO 2
140 CALL "PLOT$",A$,32,2,X,250,2
150 NEXT X
```

This example takes 500 samples from channels 1 and 2 (250 samples from each) into A$. The samples are then plotted to the screen.

SCAN

Format: CALL "SCAN",A$,N,P,C$,T,R,L,S

```
A$:O  TARGET STRING
 N:I  NUMBER OF SAMPLES TO TAKE AFTER TRIGGER
 P:I  PERIOD BETWEEN SCANS
C$:I  CHANNEL SCAN STRING
 T:I  TRIGGER MODE
 R:I  RANGE (1, 4, or 32)
 L:I  VOLTAGE TO USE FOR TRIGGER MODE 2
 S:I  NUMBER OF PRETRIGGER SAMPLES TO SAVE
```

SCAN allows sampling multiple channels in any order. It allows a voltage level trigger. It also allows samples from before the trigger to be saved. A$ receives the data. It must be dimensioned to at least N+S bytes in 8 bit mode and at least 2*(N+S) bytes in 12 bit mode. N is the number of posttrigger samples to put into A$.

P is the period between successive scans of the channel sequence. Because of the added features of SCAN, the maximum sample rate is about 3000 samples per second.

C$ is the channel sequence string. Each character in the C$ specifies a channel to read. This sequence is followed each scan. Channels 1 through 9 are denoted by the digits 1 through 9. Channels 10 through 16 are denoted by the letters A through G. Since C$ is modified by SCAN and restored when SCAN is done, if SCAN is terminated by pressing the BREAK key twice, the contents of C$ will be different from the original contents.

T is the trigger mode. Legal trigger modes are 0, 1, 2, -1, and -2.

| Trigger Mode | Behavior |
|---|---|
| 0 | Begins sampling immediately. |
| 1 | Waits for high TTL trigger. |
| -1 | Waits for low TTL trigger. |
| 2 | Waits for voltage above L. |
| -2 | Waits for voltage below L. |

R is the range specifier. R should have a value of 1, 4, or 32. This is the gain used by the instrumentation amplifier. If R equals 1, the input voltage range is 10 volts. If R equals 4, the input voltage range is 2.5 volts. If R equals 32, the input voltage range is 0.3 volts. R only selects the gain on channels 1 through 8. The gain of channels 9 through 12 (C) is fixed at 4 for SCAN. The gain of channels 13 (D) through 16 (G) is fixed at 32 for SCAN.

L is the voltage level to look for in trigger mode 2. L must be in the CALL statement even when other trigger modes are used.

S is the number of pretrigger samples to save. It is recommended that S be an even multiple of the length of the channel sequence string C$. SCAN takes samples while waiting for the trigger. When SCAN is finished, the pretrigger samples are justified within the target string. If S is very large, this may take a measurable time.

An alternative to using a string variable as the target for the data is the TransEra Auxilliary Memory. To use the Auxilliary Memory as the target, replace A$ in the CALL statement with a numeric expression. The data will be placed in the Auxilliary Memory in unformatted mode starting at this address. It is not possible to justify pretrigger samples in the Auxilliary Memory, so pretrigger samples should not be used in the Auxilliary Memory mode.

Example:

```
100 DIM A$(1000)
110 CALL "SCAN",A$,400,1.0E-3,"12",-1,1,0,100
120 WINDOW 1,250,0,65535
130 FOR X=1 TO 2
140 CALL "PLOT$",A$,32,2,X,250,2
150 NEXT X
```

This example takes 100 pretrigger samples and 400 posttrigger samples from channels 1 and 2 (50 and 200 from each) into A$. Triggering is on low TTL. The samples are then plotted to the screen.

# Data Conversion and Display Routines

## Introduction

There are 7 data conversion and display routines in the 752-ADC and 6 in the 652-ADC. They are accessed through the CALL statement in BASIC. They are written in assembly language for the greatest possible speed of execution. These routines are UNPAK2, UNPACK, PACK, MIN$, MAX$, UNMESH, and PLOT$. UNMESH is available only on the 752-ADC.

The UNPACK routine is used to convert the data in the string target variable used in BURST or SCAN into a floating point numeric format. The PACK routine performs the opposite of the UNPACK routine. The UNPAK2 routine allows the binary data to be scaled to the actual voltage while it is being unpacked.

The MIN$/MAX$ routines are used to find the value of the minimum or maximum sample in the binary string array.

The UNMESH routine allows binary string data from a multichannel scan to be split up into component channels.

The PLOT$ routine is for displaying data taken by the BURST and SCAN routines. Plots are made from data in the string variable without any scaling necessary. This means that with only two CALL statements, a sequence of data can be acquired and displayed.

In this section, the data conversion and display routines in the 652/752-ADC are explained. The format of the CALL statement is given, and each of the calling parameters is explained. Then, the general operation of the routine is explained.

In the listing of the calling parameters, the letters O and I following the parameter name indicate whether the parameter is used as an input to the routine, an output from the routine, or both. In general, input parameters may be variables, expressions, or constants. Output parameters must be variables. As far as possible, error checking is done by the routine to insure correct typing of the calling parameters.

UNPAK2 and UNPACK

Format: CALL "UNPAK2",A$,A,N,B,S,D
CALL "UNPACK",A$,A,N,B

| | | |
|---|---|---|
| A$:I | SOURCE STRING | |
| A:O | TARGET ARRAY | |
| N:I | NUMBER OF SAMPLES TO DO | |
| B:I | BYTES PER SAMPLE | |
| S:I | SUBTRACTOR VALUE | (UNPAK2 ONLY) |
| D:I | DIVISOR VALUE | (UNPAK2 ONLY) |

UNPAK2 and UNPACK is used for converting the data samples in the target strings from BURST, QSCAN, FSCAN, and SCAN to floating point numbers in an array. A$ is the string containing the data from the acquisition routine. It must contain at least B*N bytes.

A is the array into which the floating point data is put. It must be dimensioned to at least the number of samples to be unpacked.

N is the number of data samples to be transformed. B is the number of bytes in the data string to be used in each conversion. It should be 1 if the data was taken in 8 bit mode. It should be 2 if the data was taken in 12 bit mode.

UNPACK converts the binary data in the string directly into the corresponding integer value. This ranges from 0 to 255 for 8 bit data and 0 to 65535 for 12 bit data.

UNPAK2 uses S and D to scale the converted data to the actual voltage level read. In the 8 bit mode, S should be 128 and D should be 12.8*R. In the 12 bit mode, S should be 32768 and D should be 3276.8*R. Of course, if some value other than the actual voltage is desired, S and D may be specified accordingly.

Example:

```
100 DIM A$(1000),A(500)
110 CALL "BURST",A$,500,1.0E-3,1,0,1
120 CALL "UNPACK",A$,A,500,2
130 PRINT A
140 CALL "UNPAK2",A$,A,500,2,32768,3276.8
150 PRINT A
```

This example takes 500 samples from channel 1 into A$ at a sample rate of 1000 samples per second. Then, the samples are unpacked into their binary equivalent and printed. Lastly, the samples are unpacked into their voltage equivalent and printed.

PACK

Format: CALL "PACK",A$,A,N,B

A$:O TARGET STRING
A:I SOURCE ARRAY
N:I NUMBER OF SAMPLES TO DO
B:I BYTES PER SAMPLE

PACK transforms data from floating point format in an array to binary string data. This is the opposite function of UNPACK and UNPAK2. PACK may be profitabley used to prepare data for more economical tape and disc storage. A$ is the target string variable for the binary data. It must be dimensioned to at least B*N bytes.

A is the floating point array to be converted. It must contain at least N defined elements. Non integer values of A are rounded to the nearest integer. The maximum value of any element of A is 65535. If B is 1, The maximum value of an element of A is 255. B is the number of bytes in the target string to use for each conversion.

Example:

```
100 DIM A$(1000),A(500)
110 FOR X=1 TO 500
120 A(X)=65535*(0.5+0.5*SIN(X/500*2*PI))
130 NEXT X
140 CALL "PACK",A$,A,500,2
150 WINDOW 1,500,0,65535
160 CALL "PLOT$",A$,32,1,1,500,2
```

This example generates 500 samples of 12 bit data and packs it into A$. It then plots the data.

## MIN$ and MAX$ (Minimum in String and Maximum in String)

Format:
```
CALL "MIN$",A$,M,P,B
CALL "MAX$",A$,M,P,B

A$:I  SOURCE STRING
 M:O  TARGET FOR VALUE
 P:O  TARGET FOR POSITION
 B:I  BYTES PER SAMPLE
```

MIN$ and MAX$ find the minimum and maximum sample in a data string, respectively. A$ is the string containing the data to be searched. M is the target for the value of the sample found. P is the target for the position in the string of the sample found. B is the number of bytes per sample. B should be 1 for 8 bit data and 2 for 12 bit data.

Example:
```
100 DIM A$(1000)
110 CALL "BURST",A$,500,1.0E-3,1,0,1
120 CALL "MIN$",A$,M1,P1,2
130 CALL "MAX$",A$,M2,P2,2
140 PRINT "MIN: ";M1;" AT POSITION ";P1
150 PRINT "MAX: ";M2;" AT POSITION ";P2
```

This example takes data and finds the minimum and maximum points in the data.

UNMESH

Format: CALL "UNMESH",A$,B$,I,S,N,B

```
A$:I  SOURCE STRING (INTERLEAVED)
B$:O  TARGET STRING (ONLY ONE CHANNEL)
 I:I  NUMBER OF SAMPLES PER REPETITION
 S:I  STARTING POINT IN STRING
 N:I  NUMBER OF SAMPLES TO PUT IN B$
 B:I  BYTES PER SAMPLE
```

UNMESH extracts every I'th sample in A$ starting at sample number S. A$ is left unchanged. The extracted points are put in B$. The extraction process terminates after N samples have been extracted. B should be 1 for 8 bit data and 2 for 12 bit data. With 12 bit data S, N, and I refer to sample numbers, not byte numbers. UNMESH is useful in separating out one channel from a multichannel data string. It may also be useful for other purposes with single or multiple channel data.

UNMESH is available only on the 752-ADC.

Example:

```
100 DIM A$(1000),B$(500),C$(500)
110 CALL "SCAN",A$,500,1.0E-3,"12",0,1,0,0
120 WINDOW 1,250,0,65535
130 CALL "UNMESH",A$,B$,2,1,250,2
140 CALL "UNMESH",A$,C$,2,2,250,2
150 CALL "PLOT$",B$,32,1,1,250,2
160 CALL "PLOT$",C$,32,1,1,250,2
```

This example takes data from channels 1 and 2. It then unmeshes the data from channel 1 into B$ and the data from channel 2 into C$. B$ and C$ are plotted.

PLOT$ (Plot String)

Format: CALL "PLOT$",A$,D,I,S,N,B

A$:I SOURCE STRING TO PLOT
D:I DEVICE TO PLOT TO
I:I NUMBER OF SAMPLES PER REPETITION OF SCANNING SEQUENCE
S:I STARTING POINT IN STRING
N:I NUMBER OF SAMPLES PER CHANNEL TO PLOT
B:I BYTES PER SAMPLE

PLOT$ takes the data in the string variable A$ and does a continuous line plot to the screen or external plotter. The value of each sample is plotted in the Y direction and the X direction is automatically indexed to correspond to the sample being plotted.

A$ contains the data to be plotted. D is the number of the device to which the plotting should take place. It must be a legal value for a device address.

I is the interval in the data string at which samples are taken to be plotted. To make a plot of one channel from a multichannel data string, I should be equal to the number of channels in the scanning sequence. I may also be used to reduce the detail of the plot to increase drawing speed. S is the number of the first sample to be plotted. S may be used to specify the channel in the scanning sequence to be drawn.

N is the number of samples to plot. B is the number of bytes per sample. B should be 1 for 8 bit data and 2 for 12 bit data. A WINDOW statement should be performed prior to CALLING PLOT$. The WINDOW should be set at 1 to N in the X direction and the largest expected sample in the Y direction. This may be up to 255 for 8 bit data and 65535 for 12 bit data.

Example:

```
100 DIM A$(1000)
110 CALL "BURST",A$,500,1.0E-3,1,0,1
120 WINDOW 1,500,0,65535
130 CALL "PLOT$",A$,32,1,1,500,2
```

This example takes 500 samples and plots them to the screen.

# Signal Processing Routines

## Introduction

There are 4 signal processing routines in the 752-ADC. These are not available on the 652-ADC. They are accessed through the CALL statement in BASIC. They are written in assembly language for the greatest possible speed of execution. These routines are FFT$, IFT$, PACK$, and UNPAK$.

FFT$ computes the Fourier transform of string data. IFT$ computes the inverse Fourier transform of string data. PACK$ and UNPAK$ are similar to the PACK and UNPACK in section 2. PACK$ and UNPAK$ use the data format of FFT$ and IFT$ rather than the format of the ADC.

In this section, the signal processing routines in the 752-ADC are explained. The format of the CALL statement is given, and each of the calling parameters is explained. Then, the general operation of the routine is explained.

In the listing of the calling parameters, the letters O and I following the parameter name indicate whether the parameter is used as an input to the routine, an output from the routine, or both. In general, input parameters may be variables, expressions, or constants. Output parameters must be variables. As far as possible, error checking is done by the routine to insure correct typing of the calling parameters.

FFT$ and IFT$ (Fourier and inverse Fourier transforms)

Format: CALL "FFT$",A$,B$,N
CALL "IFT$",A$,B$,N

A$:IO REAL PART OF INPUT SIGNAL
B$:IO IMAGINARY PART OF INPUT SIGNAL
N:I NUMBER OF POINTS IN TRANSFORM

FFT$ computes the Fourier transform of the signal in A$ and B$ and places the result in A$ and B$. A$ and B$ are integer arrays packed in the format given in appendix A. Since the transform produces fixed-point results, the Fourier transform normalizes on the forward transform to produce numbers in the range of the fixed-point storage array. This is similar to the Fourier series but unlike most discrete Fourier transforms in use, which usually normalize on the inverse transform.

Since two bytes in each string are used for each point in the transform, each string must be dimensioned to at least 2*N. The current length of the strings need not be 2*N, however; if it is less than this figure, the string will be filled with zeros until it is 2*N bytes long. A null string is acceptable; it will be filled with zeros before the transform is computed. The easiest way to compute the transform of a real signal in A$ would be to use the following instrucions:

```
B$=""
CALL "FFT$",A$,B$,N
```

The real portion of the output of the Fourier Transform is contained in A$ and the imaginary portion in B$. The output points are in the order usual for complex discrete Fourier transforms, shown later in this section.

IFT$ computes the inverse Fourier transform of the signal in A$ and B$ and places the result in A$ and B$. No normalization is done in IFT$, because FFT$ normalizes. The output of IFT$ is in the same format as the input of FFT$ and vice versa.

Note that for FFT$ and IFT$ N must be an integral power of two between 4 and 8192.

PACK$ and UNPAK$

Format: CALL "PACK$",A$,A

A$:O STRING TARGET FOR PACKED DATA
A:I ARRAY CONTAINING DATA TO BE PACKED

CALL "UNPAK$",A$,A

A$:I STRING CONTAINING PACKED DATA
A:O TARGET ARRAY FOR UNPACKED DATA

PACK$ converts the floating point numeric data contained in the array A to fixed point format and places the output in A$. A$ must be dimensioned at least twice as large as A since two bytes are generated in A$ for each element in A. If any array element is less than -32768, -32768 is used in its place; if any element is larger than 32767, 32767 is used in its place. A$ need not be defined before PACK$ is called, but it must be dimensioned.

UNPAK$ converts the fixed point data contained in A$ to floating point data and places the output in A. A must be dimensioned at least one- half the current length of A$, since two bytes in A$ are used to com- pute each element in A.

Both PACK$ and UNPAK$ can accept multiple sets of parameters and will pack or unpack all of them. The parameters should be arranged in pairs of string,array. For example, CALL "PACK$",A$,A,B$,B,C$,C,D$,D is a valid command and will cause A to be packed in A$, B in B$, etc.

The fixed point and floating point formats are described in the following pages.

Fixed Point Format Used by Signal Processsing Routines

The integer representation of each floating number is equal to that number plus 32768. Thus, the range of integers that can be represented in the routines in the ROM pack is -32768 to +32767. Each numeric value occupies two adjacent positions in the string, the highest order byte being the first position. Some typical values and their equivalent representations are shown below:

| FLOATING POINT VALUE | INTEGER REPRESENTATION (HEXADECIMAL) 2 BYTES |
|---|---|
| +32767 | FF FF |
| 1000 | 83 E8 |
| 100 | 80 64 |
| 0 | 80 00 |
| -100 | 7F 9C |
| -1000 | 7C 18 |
| -32768 | 00 00 |

## Ordering of Frequency Data

The frequency-domain data output from FFT$ or input to INF$ is in the following order:

| ARRAY POSITION | STRING POSITIONS | FREQUENCY |
|---|---|---|
| 1 | 1,2 | 0 (constant or d.c. term) |
| 2 | 3,4 | fundamental |
| 3 | 5,6 | 2*fundamental |
| 4 | 7,8 | 3*fundamental |
| . | . | . |
| . | . | . |
| . | . | . |
| N/2 | N-1,N | (N/2-1)*fundamental |
| . | . | . |
| . | . | . |
| . | . | . |
| N/2+1 | N+1,N+2 | -(N/2)*fundamental |
| N/2+2 | N+3,N+4 | -(N/2-1)*fundamental |
| N/2+3 | N+5,N+6 | -(N/2-2)*fundamental |
| . | . | . |
| . | . | . |
| . | . | . |
| N-2 | 2N-5,2N-4 | -3*fundamental |
| N-1 | 2N-3,2N-2 | -2*fundamental |
| N | 2N-1,2N | -fundamental |

The fundamental is the frequency represented by N samples; thus fundamental=1/((time between samples)*N).

As in all complex Fourier transforms, if the input is a real signal, its power at each frequency except 0 is distributed evenly between the positive and negative components of the frequency, and thus is equal to the sum of the magnitudes of the power at the positive and negative components of the frequency. Note that the term corresponding to -(N/2)*fundamental has no positive counterpart, but, for real input functions, its counterpart is known to be its complex conjugate, as it true with all of the frequency components except zero.

SUMMARY OF ROUTINES

1. CALL "12BIT"
2. CALL "8BIT"
3. CALL "A/D",A,C
4. CALL "A/D",A1
5. CALL "BURST",A$,N,P,C,T,R
6. *CALL "QSCAN",A$,N,C,T,R,L
7. CALL "SCAN",A$,N,P,C$,T,R,L,S
8. CALL "UNPAK2",A$,A,N,B,S,D
9. CALL "UNPACK",A$,A,N,B
10. CALL "PACK",A$,A,N,B
11. CALL "MIN$",A$,M,P,B
12. CALL "MAX$",A$,M,P,B
13. *CALL "UNMESH",A$,B$,I,S,N,B
14. CALL "PLOT$"A$,D,I,S,N,B
15. *CALL "FFT$",A$,B$,N
16. *CALL "IFT$",A$,B$,N
17. *CALL "PACK$",A$,A
18. *CALL "UNPAK$",A$,A

4050 users using a second A/D Rom Pack in the same 4050 will utilize the following alternate call names:

1. CAll "A/D2",A,C
2. CALL "A/D2",A1
3. CALL "BURST2",A$,N,P,C,T,R
4. CALL "QSCAN2",A$,N,C,T,R,L
5. CALL "SCAN2",A$,N,P,C$,T,R,L,S

*These routines are available only on the 752-ADC.

5-1-84

4050 users using a second A/D Rom Pack in the same 4050 will utilize the following alternate call names:

1. CALL "A/D2",A,C
2. CALL "A/D2",A1
3. CALL "BURST2",A$,N,P,C,T,R
4. CALL "QSCAN2",A$,N,C,T,R,L
5. CALL "SCAN2",A$,N,P,C$,T,R,L,S

The call names have been changed in your 752's.
They are numbered 1, 2, 3, 4 ~~The~~
Use the information above, substituting the specific rom pack's number where the 2 is above.

9-272-