Errata

Title & Document Type: Programming the 3852 with Series 200/300 Controllers

Manual Part Number: 03852-90018

Revision Date: February 1, 1987

HP References in this Manual

This manual may contain references to HP or Hewlett-Packard. Please note that Hewlett-Packard's former test and measurement, semiconductor products and chemical analysis businesses are now part of Agilent Technologies. We have made no changes to this manual copy. The HP XXXX referred to in this document is now the Agilent XXXX. For example, model number HP8648A is now model number Agilent 8648A.

About this Manual

We've added this manual to the Agilent website in an effort to help you support your product. This manual provides the best information we could find. It may be incomplete or contain dated information, and the scan quality may not be ideal. If we find a better copy in the future, we will add it to the Agilent website.

Support for Your Product

Agilent no longer sells or supports this product. You will find any other available product information on the Agilent Test & Measurement website:

www.tm.agilent.com

Search for the model number of this product, and the resulting product page will guide you to any available information. Our service centers may be able to perform calibration if no repair parts are needed, but no other support from Agilent is available.

Agilent Technologies

Programming the HP3852 with Series 200/300 Controllers

Using the 44701 and 44702

PRELIMINARY

Programming the HP3852 with Series 200/300 Controllers

Using the 44701 and 44702

PRELIMINARY

This application paper is designed to demonstrate some common data acquisition applications using the Series 200/300 controller and the HP3852 Data Acquisition System and its two voltmeter accessories. The applications are presented in detail so that a user can gain a deeper understanding of the interactions of the HP3852, the 44701 and 44702, and the Series 200/300 controller. Hopefully, the Programming Trees for the voltmeter accessories will be very useful. They indicate a preferred programming sequence for both accessories that should prevent most coding errors and alleviate some of the dread of using the 44702 High Speed DVM.

The 44701 accessory is discussed first followed by four example programs demonstrating various capabilities of the 44701. The 44702 accessory is discussed next followed by ten example programs ranging in difficulty from simple examples to complex examples requiring a great deal of interaction between the accessory, the HP3852 mainframe and the Series 200/300 controller. Start with the discussions of the Programming Trees. These will provide a refresher for SE's who have attended the HP3852 class. For those that haven't, the Programming Trees will give a framework on which to build programming knowledge. This paper is structured to be used as an addition to the manuals. The manuals contain detailed reference data on the various commands but do not address their usage in great detail. These application examples were written to show some typical solutions and to give a basis from which to modify the programs to solve a particular application problem.

You may see this information in other documents. Where it is appropriate, this information may be given to customers. Comments and suggestions are both solicited and appreciated.

John M. da Cunha, Applications Engineer

Programming the HP44701/HP3852 with the Series 200/300

I. Introduction

The HP44701 Integrating Voltmeter uses a technique which allows a tradeoff between speed and resolution/noise rejection. Integrating voltmeters sample the input for a period of time and then digitize the result. By varying the sample period, the customer is able to optimize the voltmeter characteristics to suit the application. This allows a great deal of flexibility. If high resolution is desired, longer integration times, expressed as number of power line cycles, can be specified to gain noise rejection and resolution. If the application does not require full resolution, shorter integration times can be specified to gain greater speed. The HP44701 can measure DC volts, resistance and AC volts with variable integration time. It holds a single reading in its register.

II. High Level Programming Commands

The HP44701 Voltmeter is programmed by and sends its readings to the HP3852 Data Acquisition mainframe. The HP3852 has a high level command set that programs the HP44701 Voltmeter. The command set was designed to provide a means to control the voltmeter with a minimum of understanding of its workings as well as to provide for more advanced capabilities. A beginning user can take readings very simply, while a more advanced user can modify the default values of the voltmeter to customize it for a particular application.

To take readings in the easiest manner possible, use the CONFMEAS command. This command sets up the voltmeter in a default mode and advances the multiplexer channels while readings are taken. The mainframe (HP3852) takes the readings and puts them in an array in memory or in the HPIB output buffer. A typical command line looks like this:

```
100  OUTPUT 709;"CONFMEAS DCV 100-109 USE 0"
```

With this command, the mainframe and voltmeter work together to take ten DC voltage readings on channels 100 through 109 using the voltmeter in slot 0. The readings are placed in the HPIB output buffer where they can be accessed by the Series 200/300 controller. As long as the user is content with the default values as specified in the command reference manual for the CONF command, then CONFMEAS is sufficient. However, to use the full flexibility of the voltmeter, the customer must use other commands.

A Simple Example

For the following example, consult Figures PT1, Programming Tree for the 44701 Integrating Voltmeter, and EX1. Also consult the HP3852 Command Reference Manual for further information on the following commands. Begin the programming sequence with a USE command. This tells the

mainframe which module it will be programming. In this example, the voltmeter is in slot 0. Voltmeter programming should begin with a CONF command. This command sets the voltmeter to a known state which can be modified to suit the application. Alternately, the TRIG HOLD command can be used instead of the CONF command. TRIG HOLD does not set the voltmeter to a defined state. It only suspends triggering of the voltmeter. After CONF or TRIG HOLD, the FUNC command may be used. It is typically used with TRIG HOLD since CONF sets a function. Then the RANGE/ARANGE parameter should be specified. RANGE sets a fixed range for the voltmeter. ARANGE sets the voltmeter to autorange. In the example, the RANGE is set to three volts. Optionally, the terminals can be set next with TERM. Then the integration time is set with the NPLC command. This command sets the speed and resolution of the voltmeter. The example sets NPLC to 0.005. At this setting, a good blend of accuracy and speed is achieved. It provides 4 1/2 digit resolution at speeds approaching the maximum with this voltmeter. The NRDGS parameter sets the number of readings per trigger which translates to the number of readings per channel. AZERO selects autozero on or off. With autozero off, measurements are taken almost twice as fast. The autozero option can be specified with the ohms functions, but has no effect. Next, DELAY can be specified. DELAY sets the time between the trigger and when the actual measurement is taken. Use OCOMP next if offset-compensated four wire ohms is desired. The last parameter is the trigger source. It is set with the TRIG command. If TRIG SYS is used, the system trigger source must be set with TRG. After all these parameters are set, the MEAS command is used to begin taking the measurements. Any or all the commands between CONF and MEAS may be deleted as appropriate. If a programmer follows this sequence, most problems caused by programming errors will be avoided.

In the example, the voltmeter commands reside in a subroutine called TAKE_RDG. To begin taking measurements, call the subroutine.

```
200  OUTPUT 709;"CALL TAKE_RDG"
210  ENTER @CFORMOFF;A(*)
```

Notice that the last command in the example subroutine is:

```
180  OUTPUT 709;"VREAD RD,RL64"
```

This command tells the mainframe to output the readings from the array RD in RL64 binary format. This format is compatible with Series 200/300 internal REAL binary format so the ENTER is done with FORMAT OFF allowing faster transfer rates.

III. Applications

In addition to the previous example, three other applications will be considered. The first illustrates the use of a low level command, SWRITE, which can be used to talk to module registers under special circumstances. In this case, the program will set the integration time base to reject 400 Hz noise. A routine to unpack HP44701 readings in BASIC is also included. The second application example will demonstrate taking static strain measurements with the multiplexer and voltmeter

timed by the HP3852 system pacer. The final application shows real-time limit checking on temperature measurements. Consult the programming tree and the sample programs when following the text. For more information on the commands that are discussed, consult the HP3852 Command Reference Manual.

Example 1: 400 Hz Noise Rejection with 60 Hz Power

Aircraft systems can present special problems for data acquisition systems. In countries that use 60 Hz power, integrating voltmeters are generally configured to reject 60 Hz noise. They do this by integrating over an integral number of power line cycles. The HP44701 is set to either 60 or 50 Hz integration time at power up of the HP3852. If the HP44701 is configured for 60 Hz, it does not reject the 400 Hz noise from an aircraft power system. However, there is a procedure to change to 50 Hz operation which will reject 400 Hz noise.

In the example, Figure EX2, the voltmeter is configured normally with the USE, CONF, TERM and RANGE commands. These commands are followed by:

```
140  OUTPUT 709;"SWRITE 0,2,117"
```

This command writes directly to slot 0, register 2 which is the slot address of the HP44701. The value 117 is the decimal equivalent of a command to the voltmeter to set up 50 Hz operation at one power line cycle integration time of 20 msec. This integration time corresponds to eight power line cycles of 400 Hz. Since the integration period is more than one power line cycle of 400 Hz, additional noise rejection is obtained. If more rejection is required, the value may be set to 149. This configures the voltmeter for sixteen power line cycles at 50 Hz, which corresponds to 128 power line cycles at 400 Hz. The voltmeter can be returned to 60 Hz operation by executing an NPLC command.

The next portion of code in the example, shows a method of getting single readings from the HP44701 using the TRIG SGL and CHREAD commands. These commands are used when the voltmeter terminals are set to external. They can also be used in other applications where appropriate. Two sets of readings are taken in FOR/NEXT loops. The first is in the default format, RASC. The second set is taken in packed format with the PACK parameter for the CHREAD command. This format requires that the readings be unpacked later. An unpacking routine is included at the end of the program.

The packed data from the HP44701 consists of four bytes. Twenty-four bits represent the mantissa and the remaining eight bits represent the exponent. In the example, the variables X and Y become the mantissa and the variable Z contains the exponent information. The upper two bytes, variable X, are shifted left by multiplying by 256. Then the lowest byte of the mantissa, variable Y, is added to it. The mantissa is now correctly evaluated. The variable Z is sign corrected and manipulated to form the exponent. The result is then printed.

Example 2: Paced Strain Gauge Measurements

In this application, the customer wanted to take strain measurements at known time intervals. The most natural way to do this was to take advantage of the pacing function of the HP3852 to pace the readings. Refer to Figure EX3.

Two real arrays are configured in the HP3852 with the REAL command to hold the strained and unstrained readings. The unstrained readings are necessary to calculate the value of the strain for each channel. The readings in the array called STRAIN will be in microstrains.

The voltmeter is configured with the USE and CONF STRQ to take quarter bridge strain measurements. Then the trigger source is set to TRIG SCAN. The voltmeter will be triggered each time a channel is closed. The command MEAS STRUN then takes the unstrained measurements and stores them in the array UNSTR. The pacer now needs to be set up.

```
100  OUTPUT 709;"SADV PACER"
110  OUTPUT 709;"PACER 85E-3,26"
120  OUTPUT 709;"PDELAY .015"
```

The first command, SADV PACER, sets the scan advance source to the pacer output. Next the pacer is set to put out 26 pulses at a time interval of 85 milliseconds with the PACER command. Twenty-six pulses are output because the first pulse is used to close the first channel and the last pulse opens the last channel. The PDELAY command sets a delay after the pacer trigger before the first pulse comes out. When measuring strain in this manner, PDELAY should be set to a large enough value to ensure that a preliminary reading of the supply voltage on each multiplexer module can be accomplished. This time will vary depending on the NPLC setting and the number of different multiplexer modules involved. For instance, if NPLC is set to one and there are four multiplexer modules, then the delay should be sufficient to make at least four one power line cycle measurements plus some additional time for autoranging and processing (approx. 180 msec.).

A subroutine is used to trigger the pacer and take the strain measurements. In many cases, placing the measurement commands in a subroutine can yield speed improvements since only the string "CALL X" needs to be parsed by the mainframe rather than each command in the subroutine. If all the programming is done in subroutines, set up commands should be put in a separate subroutine than the measurement commands for best speed performance.

Example 3: Temperature Limit Checking

The customer in this application needed to monitor the temperature on a thermocouple and be signalled by the HP3852 when the temperature was out of limit. The real time limit checking feature of the HP3852 is a good match for this application. A limit subroutine is called in the HP3852 mainframe that asserts SRQ on the HPIB bus to signal the controller that the temperature is out of bounds.

Refer to Figure EX4 for the example program. The HP3852 does real time limit checking in the mainframe as the measurements are put into the

array (RTEMP in the example). If a measurement is out of limit, the index of that reading is placed into an integer array (IDX in the example). Also, an interrupt condition is set if enabled. This interrupt can cause an SRQ on the HPIB bus or other actions.

```
120  OUTPUT 709;"INTEGER IDX(0)"
130  OUTPUT 709;"REAL RTEMP(0),MIN,MAX"
140  OUTPUT 709;"MIN=22;MAX=28"
```

Note that IDX and RTEMP must be arrays. The variables MIN and MAX are the limit variables against which the measurements are compared. MIN and MAX can also be arrays if the expected measurements will vary greatly from channel to channel. MIN and MAX must also be real variables or arrays to prevent mixed mode arithmetic errors.

The DVM is set up in the default mode with the USE and CONF commands. Speed was not a consideration in this application. The limit subroutine (SUB WARN) displays a message, asserts SRQ and beeps. Then the limit conditions are set.

```
250  OUTPUT 709;"LMT MIN,MAX,IDX"
260  OUTPUT 709;"ON LMT CALL WARN"
270  OUTPUT 709;"ENABLE LMT"
280  OUTPUT 709;"RQS FPS,RQS ON"
```

The LMT command in line 250 sets up real time limit checking. The subroutine WARN will be called on limit (line 260). Then the limit interrupt is enabled. Line 280 sets the SRQ mask for software SRQ and enables SRQ on the bus.

The measurement subroutine (SUB MEAS_) takes the readings and displays a message. The channel list length is a significant variable when using real time limit checking. The limit interrupt is serviced at the end of the measurement subroutine. Therefore, if the channel list is very long and an out of limit reading is taken early in the channel list, the interrupt won't be serviced until the subroutine is finished. Short channel lists and command sequences are desirable when using this feature of the HP3852.

The controller is set up to receive the SRQ interrupt with the ON INTR and ENABLE commands. The ON INTR command directs the controller to stop when an interrupt is present on the HPIB bus. The ENABLE command activates the interrupt on the HPIB interface and sets the SRQ mask. Only SRQ interrupts will be recognized.

IV. Summary

Three application examples using the HP44701/HP3852 and a Series 200/300 controller running BASIC were presented as well as a simple example to show a way to get fast readings from the HP44701 integrating DVM. The DVM is quite flexible in choice of integration time to optimize it for a particular application. When coupled with the multiplexer control, conversion and interrupt features of the HP3852 mainframe, it can solve a great many data acquisition problems.

Programming Tree
for 44701 Integrating
Voltmeter

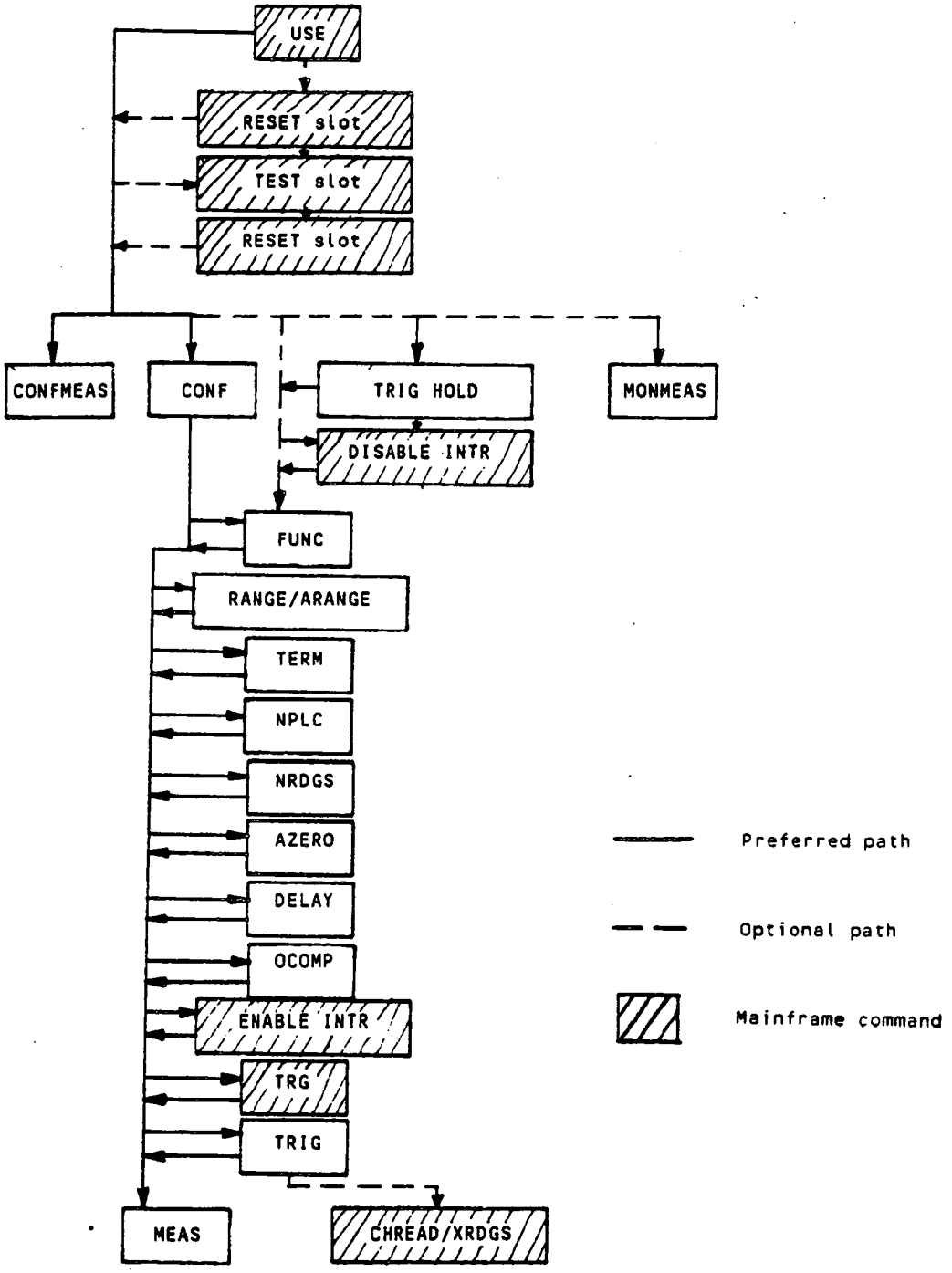


Figure PT1

```
10  ! THIS PROGRAM SHOWS A WAY TO GET HIGH SPEEDS FROM THE 44701
20  ! INTEGRATING DVM.
30  !
40  !
50  ASSIGN @Comp TO 709
60  ASSIGN @Cformoff TO 709;FORMAT OFF
70  REAL A(1:100)
80  !
90  CLEAR @Comp
100 OUTPUT @Comp;"RESET"
110 OUTPUT @Comp;"PACKED RD(399)"
120 OUTPUT @Comp;"SUB TAKE_RDG"
130 OUTPUT @Comp;"    USE 0"
140 OUTPUT @Comp;"    CONF DCV"
150 OUTPUT @Comp;"    RANGE 3"
160 OUTPUT @Comp;"    NPLC 0.005"
170 OUTPUT @Comp;"    AZERO OFF"
180 OUTPUT @Comp;"    MEAS DCV 100-119,100-119,100-119,100-119,100-119 INTO RD"
190 OUTPUT @Comp;"    VREAD RD,RL64"
200 OUTPUT @Comp;"SUBEND;DISP OFF"
210 !
220 !
230 OUTPUT @Comp;"CALL TAKE_RDG"
240 ENTER @Cformoff;A(*)
250 FOR I=20 TO 100 STEP 20
260    PRINT A(I)
270 NEXT I
280 END
```

Figure EX1

```
10    ! THIS PROGRAM DEMONSTRATES HOW TO REJECT 400Hz NOISE WITH THE
20    ! 44701 INTEGRATING DVM ON THE 3852.
30    ! THIS PROGRAM ALSO INCLUDES AN UNPACKING ROUTINE FOR THE DVM.
40    !
50    ASSIGN @Comp TO 709
60    ASSIGN @Cformoff TO 709;FORMAT OFF
70    DIM A(1:25)
80    INTEGER B(0:49),X,Y,Z,I,J
90    OUTPUT @Comp;"RST"
91    OUTPUT @Comp;"USE 0"
100   OUTPUT @Comp;"CONF DCV"
101   OUTPUT @Comp;"RANGE 3.0"
102   OUTPUT @Comp;"TERM EXT"
110   !
120   ! THE FOLLOWING COMMAND SETS UP THE DVM FOR 50HZ(400HZ) REJECTION.
130   !
140   OUTPUT @Comp;"SWRITE 0,2,117"    ! USE 149 FOR 128 PLC'S, 117 FOR 8 PLC'S
150   !
160   OUTPUT @Comp;"DISP 'REAL MEAS'"
170   WAIT 2
180   FOR I=1 TO 25                    ! THIS LOOP READS DATA IN RASC FORMAT
190     OUTPUT @Comp;"TRIG SGL;CHREAD 0"    ! AND PUTS IT INTO A REAL ARRAY.
200     ENTER @Comp;A(I)
210   NEXT I
220   !
230   OUTPUT @Comp;"DISP 'PACK MEAS'"
240   WAIT 2
250   FOR J=0 TO 24
260     OUTPUT @Comp;"TRIG SGL;CHREAD 0 PACK"
270     ENTER @Cformoff;B(2*J),B(2*J+1)
280   NEXT J
290   !
300   !
320   PRINT B(*)
330   PRINT A(*)
340   !
350   ! UNPACK ROUTINE STARTS HERE.
360   !
370   FOR I=0 TO 24
380     X=B(2*I)                   ! UPPER TWO BYTES
390     Z=BINAND(B(2*I+1),255)     ! LOWEST ORDER BYTE
400     Y=SHIFT(B(2*I+1),8)        ! SHIFT BYTE TO LOWEST BYTE
410     IF Z=128 THEN              ! OVERLOAD CONDITION
420       PRINT 1.E+38
430     ELSE
440       PRINT (X*256.+Y)*10^(Z-256*(Z>127)-6),    ! PRINT RESULT
450     END IF
460   NEXT I
470   !
480   ! END OF UNPACK ROUTINE.
490   !
500   END
```

Figure EX2

```
10  ! THIS PROGRAM DEMONSTRATES HOW TO TAKE STRAIN GAUGE MEASUREMENTS
20  ! THAT ARE PACED BY THE PACER. FIRST THE GAUGES ARE MEASURED UN-
30  ! STRAINED, THEN THE GAUGES ARE SCANNED USING THE PACER.
40  !
50  !
60  ! PROGRAM STARTS HERE.
70  !
80  !
90  !
100 !
110 REAL A(1:25),B(1:25)
120 ASSIGN @Comp TO 709
130 CLEAR @Comp
140 OUTPUT @Comp;"RESET"
150 OUTPUT @Comp;"REAL UNSTR(24),STRAIN(24)"
160 !
170 !
180 ! SET UP DVM
190 !
200 OUTPUT @Comp;"USE 0"
210 OUTPUT @Comp;"CONF STRQ"
220 OUTPUT @Comp;"TRIG SCAN"
230 OUTPUT @Comp;"MEAS STRUN 100-109,100-109,100-104 INTO UNSTR"
240 !
250 !
260 !
270 ! SET UP SCANNER
280 !
290 OUTPUT @Comp;"SADV PACER"
300 OUTPUT @Comp;"PACER 85E-3,26"
310 OUTPUT @Comp;"PDELAY .015"
320 !
330 OUTPUT @Comp;"SUB X"
340 OUTPUT @Comp;"    PTRIG SGL"
350 OUTPUT @Comp;"    MEAS STRQ 100-109,100-109,100-104,REF UNSTR INTO STRAIN"
360 OUTPUT @Comp;"SUBEND"
370 !
380 !
390 !
400 !
410  OUTPUT @Comp;"CALL X"
420  OUTPUT @Comp;"VREAD UNSTR"
430  ENTER @Comp;A(*)
440  OUTPUT @Comp;"VREAD STRAIN"
450  ENTER @Comp;B(*)
460  PRINT A(*)
470  PRINT B(*)
480 !
490 !
500 !
510 !
520 END
```

Figure EX3

```
10    ! THIS PROGRAM DEMONSTRATES AN EXAMPLE OF REAL TIME LIMIT
20    ! CHECKING ON THE HP3852. TEMPERATURE MEASUREMENTS ARE
30    ! TAKEN AND THEN COMPARED TO THE LIMITS. IF A LIMIT IS
40    ! EXCEEDED, A WARNING IS DISPLAYED.
50    !
60    !
70    !                                          !SET UP
80    ASSIGN @Comp TO 709
90    CLEAR @Comp
100   OUTPUT @Comp;"RESET"
110   !
120   OUTPUT @Comp;"INTEGER IDX(0)"              !DECLARE VARIABLES IN 3852
130   OUTPUT @Comp;"REAL RTEMP(0),MIN,MAX"
140   OUTPUT @Comp;"MIN=22;MAX=28"
150   !
160   OUTPUT @Comp;"USE 0;CONF TEMPT"            !SET UP DVM
170   !
180   !
190   OUTPUT @Comp;"SUB WARN"
200   OUTPUT @Comp;"    DISP 'WARNING'"
210   OUTPUT @Comp;"    SRQ"
220   OUTPUT @Comp;"    BEEP"
230   OUTPUT @Comp;"SUBEND"
240   !
250   OUTPUT @Comp;"LMT MIN,MAX,IDX"             ! SET UP 3852 CONDITIONS
260   OUTPUT @Comp;"ON LMT CALL WARN"
270   OUTPUT @Comp;"ENABLE LMT"
280   OUTPUT @Comp;"RQS FPS;RQS ON"
290   !
300   !
310   OUTPUT @Comp;"SUB MEAS_"                   ! MEASUREMENT ROUTINE
320   OUTPUT @Comp;"    MEAS TEMPT 100 USE 0 INTO RTEMP(0)"
330   OUTPUT @Comp;"    DISP 'TEMP OK'"
340   OUTPUT @Comp;"SUBEND"
350   !
360   !
370   !
380   INTEGER Flg,Mask                           ! SET UP CONTROLLER
390   Flg=1
400   ON INTR 7 GOSUB Stop_
410   Mask=2
420   ENABLE INTR 7;Mask
430   !
440   WHILE Flg                                  ! BEGIN MEASUREMENT LOOP
450     OUTPUT @Comp;"CALL MEAS_"
460     WAIT 1
470   END WHILE
480   !
490 Stop_:STOP
500   !
510   END
```

Figure EX4

Programming the HP44702/HP3852 with the Series 200/300

I. Introduction

The HP44702 High Speed Voltmeter is a successive approximation type of digital voltmeter which has great speed and flexibility. Successive approximation voltmeters sample the input for a short period of time and use a fast technique to digitize the reading. This technique allows reading speeds of one hundred thousand readings per second. The HP44702 also has some intelligence built into it. It can stand alone as a subsystem taking readings with high speed multiplexers connected to it via a ribbon cable. Intelligence and high speed make the HP44702 a potent addition to the HP3852 Data Acquisition System. With this capability comes complexity. The purpose of this document is to help the user better understand how to use the HP44702 to meet his or her data acquisition needs.

The HP44702 has two modes of operation, the system mode and the scanner mode. In the system mode, the user programs the HP44702 much like the HP44701 Voltmeter. In the scanner mode, the HP44702 can achieve 100 thousand readings per second speed while scanning or do a variety of other types of measurements. The high speed voltmeter can make both DC voltage and ohms readings. When using the MEAS command, which will be discussed later, temperature and strain measurements can be made. The HP44702 has differential inputs. Differential inputs allow only a moderate common mode input signal to be connected to them with respect to earth ground. The HP44701, which is a floating voltmeter, can tolerate much higher common mode voltages. Also, the high speed voltmeter has a reading buffer built into the module. This is useful to store readings until they can be conveniently retrieved. The HP44702A has an 8K reading buffer and the HP44702B has a 64K reading buffer.

II. High Level Programming Commands

Like the HP44701, the HP44702 High Speed Voltmeter is programmed by and can send its readings to the HP3852 Data Acquisition mainframe. The HP3852 has a high level command set to comunicate with the HP44702. This command set is similar to the one for the HP44701 but has more commands since the high speed voltmeter has more intelligence and capability than the integrating voltmeter. The command set provides a means to take measurements with a minimum of understanding of the internal workings of the high speed voltmeter. Other commands take advantage of the full feature set of the voltmeter. The customer has the choice.

The CONFMEAS command allows measurements to be taken most easily. Use this command in the system mode, which is the power up configuration. CONFMEAS sets up the voltmeter in the default mode and advances the multiplexer channels while the readings are taken. The readings are transferred to either the HPIB buffer or an array in memory. A typical command looks like this:

```
100 OUTPUT 709;"CONFMEAS DCV 400-419 USE 500"
```

The voltmeter and HP3852 mainframe work together to take twenty readings on channels 400 through 419 using the voltmeter in slot 5. The readings then go to the HPIB output buffer of the HP3852 where they can be accessed by the Series 200/300 controller. If the customer is content with the default values found in the Command Reference Manual under the CONF command, then using CONFMEAS is sufficient. However, to use the full capabilities of the voltmeter, the customer must use other commands.

Two Simple Examples

As mentioned before, the HP44702 high speed voltmeter has two modes of operation. In the system mode, the commands used to program the voltmeter look and act like the corresponding commands for the HP44701 integrating voltmeter. This promotes ease of use and compatibility between the two modules. The other mode of operation is the scanner mode. In this mode of operation, the full capability of the high speed voltmeter is available. The scanner mode allows the HP44702 to act as a measurement subsystem to take measurements without the intervention of the HP3852 mainframe. The scanner mode gives access to the two levels of triggering and all the features of the module. For the following examples, refer to Figure PT2, Programming Tree for the 44702 High Speed Voltmeter, as well as, the HP3852 Command Reference Manual.

System Mode Example

Refer to Figure SYS while following the example. Start with the USE command. This tells the HP3852 mainframe which slot you are programming. In the example, the voltmeter is in slot 6. Then use the SCANMODE command to set which mode the voltmeter is in. The example program sets the system mode for the voltmeter with SCANMODE OFF. Then, like the 44701, use the CONF command. CONF puts the voltmeter into a known default state. It makes a good starting point from which to modify the program to fit a particular application. The user may choose TRIG HOLD in place of CONF, but it does not set the voltmeter to a known state. It only suspends triggering. After CONF or TRIG HOLD, the FUNC command may be used. Use FUNC when specifying TRIG HOLD since no function is set by the TRIG command. Next, the RANGE/ARMODE command sets the voltmeter range or autorange mode. The use of ARMODE will be discussed under the scanner mode example. TERM selects the input terminals. The example selects the HP3852 analog backplane as the signal source. With the 44702, readings may be output to the HP3852 mainframe or the GPIO port. The RDGS command selects the output port. The example sends the readings to the HP3852 mainframe. RDGSMODE is a command that is unique to the 44702. The high speed voltmeter has many different interrupt modes. RDGSMODE chooses which interrupt will be used. Consult the Command Reference Manual for the choices available. The next choice in line on the Programming Tree is the NRDGS command. It sets the number of readings per trigger in the system mode. In the scanner mode, the action of NRDGS varies depending on the specified terminals. Specify DELAY next, if desired. It sets the time between the trigger and the start of the first measurement on a particular channel, as well as, the time between successive measurements on the same channel if NRDGS is set to more than one. Autozero on the

high speed voltmeter differs from the integrating voltmeter. It is only done when the AZERO command is executed. Another unique feature of the 44702 is its ability to trigger on a threshold level. The PERC and SLOPE commands set up this feature. PERC sets the trigger level to percentage of full scale. If you are using auto range, this trigger level changes with range but remains at the same percentage of full scale. The SLOPE command tells the voltmeter to trigger when the signal crosses the threshold on a rising or falling slope. SLOPE also sets the sense of the trigger edge for external triggers. TRIGOUT enables trigger pulses to be output on the EXTERNAL TRIGGER 0 BNC at the back of the voltmeter. Finally, the TRIG command sets the trigger source for the voltmeter. If TRIG SYS is used, then the TRG command must specify the system trigger source. When scanning, send the MEAS command after setting the trigger source. MEAS also makes it possible to take temperature measurements by using the appropriate function parameter. If the external terminals are used, then read the measurements with CHREAD or XRDGS. Any of the commands between CONF and MEAS may be deleted to fit the application. This programming sequence should avoid most of the programming errors introduced by users.

The example program has the voltmeter set up and measurement commands in a subroutine called HS_RDGS. To take measurements, simply call the subroutine.

```
250 OUTPUT 709;"CALL HS_RDGS"
260 ENTER @CFORMOFF;A(*)
```

As in the 44701 example, the last command in the subroutine is:

```
220 OUTPUT 709;"  VREAD RD,RL64"
```

This command tells the mainframe to output the readings from the array RD in RL64 binary format. This format is compatible with Series 200/300 internal REAL binary format. Therefore, the ENTER is done with FORMAT OFF allowing faster data transfer rates.

Scanner Mode Example

Refer to Figure SCN and Figure PT2 while following the example. In scanner mode, all of the commands described in the system mode example are valid plus a number of others not presented yet. Scanner mode gives the user full control of the 44702 voltmeter. Some of the commands described in the system mode discussion have slightly different functions in the scanner mode. These differences will be pointed out.

The 44702 High Speed Voltmeter has two levels of triggering. The two levels are the scan level and the measurement level. The scan level enables the measurement level. Each level has independent pacing and delay. At the scan level, the user can specify more than one PRESCAN or POSTSCAN. These terms will be discussed later. Independent trigger sources can be set for each level also. The voltmeter is quite flexible so the commands can be confusing. Consult the Command Reference Manual for more information, if needed.

As in system mode, the USE command should be used first. Then set SCANMODE ON. The high speed voltmeter is now in the scanner mode. The CONF command can be sent next, which puts the voltmeter in a predefined state. The state for scanner mode is different than the one for system mode. Consult the Command Reference for the differences. SCTRIG HOLD can be sent instead of CONF. It acts the same way as TRIG HOLD for system mode, that is, it only suspends triggering. Use the FUNC command with SCTRIG HOLD. Next set RANGE/ARMODE. ARMODE specifies when autorange will be performed. The AFTER parameter puts the voltmeter in a mode where autoranging is done after a measurement trigger. This is the default mode and will work well for most applications. The only exception is when gathering data for FFT's. In the ARMODE AFTER setting, there is a slight amount of timing jitter between the measurement trigger and the actual measurement. ARMODE BEFORE does away with the jitter by autoranging in a look-ahead fashion before the measurement trigger. The example sets a fixed range. TERM, RDGS and RDGSMODE have the same function as in system mode. NRDGS is slightly different in scanner mode. NRDGS sets the number of readings per channel. The number allowed depends on the TERM setting. AZERO, PERC and SLOPE have the same function as in system mode. SCSLOPE is a command used only in scanner mode. It sets the slope of the scan trigger source in SCTRIG EXT, MEAS or GPIO. TRIGOUT and TRG have the same functions as in the system mode. TRIG sets the source for the measurement triggers in scanner mode. This is the lowest trigger level of the voltmeter. Consult the 44702 Configuration and Programming Manual for details of scan and measurement triggering.

The following commands refer to scan and measurement programming. They are unique to the scanner mode. SPER sets the time interval between successive measurement triggers when TRIG INT is specified. SCDELAY sets the time between the scan trigger and the first measurement trigger at the start of a scan, as well as, the time between successive scan triggers. The number of scans before the stop trigger is enabled is set with PRESCAN. POSTSCAN sets the number of scans after the stop trigger. These two commands allow data to be captured around an event. There will be an example program showing how to do this later. After this, ASCAN sets whether scans will start automatically at the completion the previous scan (ASCAN ON), or will require a separate scan trigger for each pass through the scanlist (ASCAN OFF). CLWRITE is only valid in scanner mode with TERM set to RIBBON. CLWRITE specifies the channel and range list, as well as, setting the analog bus configuration on the ribbon cable. STSLOPE sets the slope of the stop trigger when the stop trigger is set to an external source or threshold trigger. STTRIG sets the stop trigger source. In scanner mode, the scan trigger source set with the SCTRIG command starts the high speed voltmeter. If the ribbon cable is used, after a SCTRIG command is issued, the voltmeter works independently. It can be thought of as a measurement subsystem. The CHREAD and XRDGS commands remove measurements from the HP44702 voltmeter. MEAS can also be used in the scanner mode. However, MEAS changes some parameters, so it must be used with caution. Consult the Command Reference to see the parameters that are changed. If MEAS is used, SCTRIG performs the same function as STRIG does in the system mode. MEAS is used in the example for simplicity. The other scanner mode commands will be used in the application examples.

III. Applications

The two previous examples illustrate how simple it is to take measurements with the HP44702 if the application is not complicated. This is often not the case. Many users will want to use the more advanced features of the high speed voltmeter. The following application examples show a few typical instances of the use of these more advanced features. Each application example illustrates a particular use of the HP44702 High Speed Voltmeter.

Example 1: Using Interrupts with the HP44702

One source of confusion arising from the use of the high speed voltmeter is how to use its interrupt capabilities. There are four interrupt choices available to the user. RDGSMODE specifies when interrupts occur and when data is available. The default mode is DAV which stands for Data AVailable. With this parameter specified, the voltmeter interrupts when it has any reading available. This is used when you want the voltmeter to signal when any reading is available or when you want to take out readings immediately. In the BURST mode, the voltmeter interrupts when there is only room for 4K more readings in the voltmeter reading buffer. Use this mode when you want the mainframe to be free to do other things (like service other interrupts) while readings are being taken. Data is not available until the interrupt occurs. When END is specified, the interrupt occurs after the end of the scan sequence. Data is available immediately, however. Use END when you want to signal the end of the scan, but wish to take some of the readings out of the buffer before the scan sequence is finished. If COMPLETE is set in the RDGSMODE command, data is available and the interrupt occurs at the end of the scan sequence. Since data may be overwritten, this mode can be used to capture a window of data around an event. The only limitation to the size of the window is the size of the voltmeter reading buffer. If a larger window of data is needed, other techniques must be used. These will be shown in a later example.

Refer to the program listing in Figure INTR during the following discussion. This example demonstrates the use of the voltmeter interrupt to signal the HP3852 mainframe when the scan sequence is finished. The mainframe is free to do other things while waiting for the interrupt. The readings are accessed when the interrupt is received. A subroutine called TAKE_RDG sets up the voltmeter. Then the subroutine called X transfers the readings.

The USE, SCANMODE, CONF and RANGE commands configure the voltmeter in the subroutine called TAKE_RDG. Then the interrupt is set up.

```
180  OUTPUT 709;"    RDGSMODE END"
190  OUTPUT 709;"    SPER 250E-6"
200  OUTPUT 709;"    ENABLE INTR"
210  OUTPUT 709;"    CLWRITE SENSE 500-509"
220  OUTPUT 709;"    SCTRIG SGL"
```

The RDGSMODE END command sets the voltmeter to interrupt after the scan is finished. Then the SPER command sets the period between measurement

triggers. The interrupt is enabled by the ENABLE INTR command. This command only sets up the interrupt at the module; the system level interrupt must also be enabled later. The CLWRITE command sets the configuration of the ribbon cable analog bus and sends the channel list to the voltmeter. SCTRIG then triggers it.

The subroutine called X, consists of a single command.

```
260  OUTPUT 709;"    XRDGS 600,RL64"
```

This command will transfer the readings from the voltmeter after the voltmeter has interrupted the HP3852 mainframe. Note that the format specified is RL64 to increase transfer speed. This binary format is compatible with the internal real binary format of the Series 200/300 so that no formatting on either the controller end or the HP3852 end is necessary.

```
290  OUTPUT 709;"ON INTR USE 600 CALL X"
300  OUTPUT 709;"ENABLE INTR SYS"
310  !
320  OUTPUT 709;"CALL TAKE_RDG"
330  ENTER @Cformoff;A(*)
```

The interrupt from the voltmeter is directed with the ON INTR command in line 290 to call subroutine X. System interrupts are enabled on the following line. Recall that the module interrupt was enabled earlier in line 200. The set up for measurement sequence is now complete. All that is required to set the process in motion is to CALL the measurement subroutine and acquire the data in lines 320 and 330. The controller will wait on the ENTER statement until the voltmeter completes its measurement sequence, signals the mainframe, and the mainframe begins to output data.

Example 2: Using GPIO with the HP44702

The full speed capabilities of the HP44702 are available with the use of the GPIO interface. Triggering, interrupts and data can all be passed via this interface. Refer to the program listing in Figure GPIO during the following discussion. This example demonstrates triggering, interrupt handling and data transfer over GPIO with the HP44702 High Speed Voltmeter. The voltmeter is set up via HPIB, then triggered over GPIO. When the scan sequence is complete, a GPIO interrupt is generated. The controller then reads in the data in packed format. The data is then unpacked using the routine furnished.

Setting up the I/O paths in the Series 200/300 is the first thing that must be done. The ASSIGN statements accomplish this. Next, program the GPIO interrupt.

```
130  ON INTR Gpio GOSUB Get_rdgs
140  Eir_=1
150  ENABLE INTR Gpio;Eir-
```

The ON INTR statement directs the controller to a routine called

Get_rdgs. This routine will acquire the data from the voltmeter. Then, the ENABLE INTR command enables the interrupt and sets the mask. Only EIR interrupts are enabled in this case.

Next, set up the voltmeter over HPIB. The USE and SCANMODE commands are similar to the commands in the previous example. The CONF command puts the voltmeter in a state to measure two wire ohms up to 10 K ohms. Then set up the voltmeter for GPIO operation.

```
220 OUTPUT 709;"RDGS GPIO"
230 OUTPUT 709;"RDGSMODE BURST"
240 OUTPUT 709;"SCSLOPE HL"
250 OUTPUT 709;"SPER .10"
260 OUTPUT 709;"ENABLE INTR"
270 OUTPUT 709;"CLWRITE COM 500-509"
280 OUTPUT 709;"SCTRIG GPIO"
```

The RDGS command directs the data and interrupts to the GPIO port of the voltmeter. The RDGSMODE command will not make data available until there is room for only 4096 readings in the voltmeter buffer or when the scan sequence is complete. In this case, RDGSMODE COMPLETE would have done the same thing as BURST since the number of readings is small. However, RDGSMODE COMPLETE allows readings to be overwritten which is undesirable in this application. Set the slope of the GPIO trigger with the SCSLOPE command. The SPER, ENABLE and CLWRITE commands set the sample period, enable the interrupt and select the channels to be scanned. Then the SCTRIG command sets the scan trigger source to GPIO. At this point, the voltmeter is ready to accept a trigger and begin the scan sequence. However, further programming over the GPIO interface is necessary to get data and trigger the voltmeter. Since the voltmeter is being programmed from two different sources, HPIB and GPIO, care must be taken to insure that the two interfaces do not get out of synchronizaton. A good way to do this is to monitor the BUSY bit in the status register with the SPOLL command. The BUSY bit will not be released until the HPIB commands have been completed. This assures that the voltmeter is programmed via HPIB before the user attempts to program it via the GPIO interface.

```
300 WHILE NOT BIT(SPOLL(709),4)
310 END WHILE
320 !
330 CONTROL Gpio,2;5
340 OUTPUT @Gpio USING "#,W";4
350 CONTROL Gpio,2;4
360 !
370 CONTROL Gpio,2;6
380 CONTROL Gpio,2;4
```

The WHILE loop waits until the HP3852 is ready (BUSY bit is cleared). Then the CONTROL and OUTPUT statements in lines 330 through 350 select the data register. These statements tell the voltmeter to output data on the GPIO interface. The voltmeter now only requires a trigger to begin. The CONTROL statements in lines 370 and 380 accomplish this. The voltmeter is actually triggered at the first CONTROL statement (remember the SCSLOPE command); the second command just returns the trigger bit to

its original state. The voltmeter now begins to take readings and will signal the controller, via the GPIO interrupt, when the readings are ready to be transferred. An ENTER statement accomplishes the transfer. The user must keep in mind that the data returned is in PACKED format since the HP3852 mainframe is not available to do formatting on the GPIO interface.

This brings us to the unpacking routine, which starts at the line label Unpack. This routine is written in BASIC but could just as easily be written in a CSUB. The packed data comes out as sixteen bits. The most significant bit is the good reading bit. It will always be a 1 for a good reading. This corresponds to a negative integer so the Unpack routine gives a bad reading (1.0E+38) if the integer is positive. The next two most significant bits are the range bits. A correct reading requires the mantissa to be divided by one of four values dictated by the range bits. The four values are shown in Figure UNPK, as well as, the bit configuration of the data word. The next most significant bit is the sign bit and the remaining twelve bits are the mantissa. The mantissa is the total number of voltmeter counts. The reading is evaluated as a positive number and then negated if the sign bit is negative.

Example 3: Capturing a Window of Data Around an Event

Often, a user wants to acquire data about an event that happens at an unknown time. In many of these cases, the user wants some data about the events leading up to the phenomenon, as well as, data after the event. Examples include seismic testing, failure analysis and capturing random transient events. The HP44702 Voltmeter can make this type of measurement. In this application, the voltmeter will record data both before and after an event. The event will be signalled by the value of the measurement on the first channel. When the value of the measurement crosses a threshold value in the proper direction, that will be the signal to stop. Data will be kept from before the event in addition to data taken after the event. The key to success in this applicaton is to use the stop trigger capability of the High Speed Voltmeter. Threshold triggering is used to reject similar but lower level events.

Refer to the example program in Figure EVNT during the following discussion. In this example, the voltmeter is set up and triggered. It takes readings until it receives a threshold stop trigger, then the final set of readings is taken and an interrupt is generated. The interrupt routine outputs the readings to the controller. The RESET, USE and SCANMODE commands are similar to the commands used in previous examples. Next, SCTRIG HOLD and FUNC commands disable scan triggers and set the function. A CONF command could have been used instead of SCTRIG and FUNC, but these commands were used for variety. Remember, when using SCTRIG and FUNC rather than CONF, all parameters of the voltmeter must be specified to insure a known state since these commands only suspend triggering and set function; they do not change any other settings. The range, terminals and readings destination are set next with RANGE, TERM and RDGS, respectively. The interrupt and threshold are set next.

```
220 OUTPUT 709;"RDGSMODE COMPLETE"
230 OUTPUT 709;"PERC 80"
```

With RDGSMODE COMPLETE set, data is not available nor is the interrupt asserted until the scan sequence, consisting of the number of prescans immediately preceding the stop trigger and the number of postscans after, is completed. Data in the buffer of the voltmeter is overwritten. Scans occur until a valid stop trigger is received. A valid stop trigger occurs when the measured value of the first channel in the scan list crosses a set percentage of the full scale range in the proper direction. The percentage is set with the PERC command and represents the percent of the full scale on the particular range. Use a range other than autorange when using threshold triggering since the value represented by the percent on one range is very different from the value represented by the same percent on a higher or lower range. The percent remains the same in autorange, but the value changes with each range. In this example, the threshold is set to 80% of full range (80% of 2.56V = 2.05V).

```
240 OUTPUT 709;"SPER .05"
250 OUTPUT 709;"SCDELAY .01,.5"
```

The combination of the SPER and SCDELAY commands with the channel list of ten channels causes readings to be taken at equally spaced intervals of .05 seconds. The SPER command sets the interval between successive measurement triggers. Then the SCDELAY command sets the interval between successive scan triggers. Since there are ten channels being measured, the scan pace parameter of SCDELAY (.5 seconds) is set to ten times the SPER setting (.05 seconds) to get equal intervals between all the channels. The interval between the last channel in the scan and the first channel in the next scan will be the same as the interval between the channels within the scan. All the measurements will be spaced at equal time intervals.

The interrupt is enabled next with ENABLE. Then the number of prescans and postscans is set to five each and the channel list is set with the CLWRITE command. The entire number of prescans must be completed before the stop trigger will be enabled. The channel that will be monitored for threshold trigger is channel 500 in this example. Remember that it is always the first channel in the channel list that is monitored.

The last thing that must be done in the set up is to set the slope and source for the stop trigger. In the example, the STSLOPE command sets the slope to a falling edge. The stop trigger is set with STTRIG MEAS to threshold triggering. The measurement process now only requires a scan trigger to begin.

As mentioned above, the intent in this application is to have the voltmeter hold the data in its reading buffer until the scan sequence completes and then pull the interrupt. When that interrupt is received, the readings need to be output to the controller. An interrupt service routine in the HP3852 mainframe can accomplish this. The subroutine is called X and consists of a single statement, XRDGS. Note again that the output format is RL64, a binary format. This was done to increase transfer speed. ON INTR directs the accessory interrupt to the subroutine, and the system interrupt is enabled to allow the interrupt to be

recognized and serviced. The SCTRIG INT command starts the scan sequence. The INT parameter is used so that the scan pace parameter of SCDELAY will be used to pace the scans. Once the SCTRIG command is sent, the voltmeter will take the specified number of prescans and then enable the stop trigger. Scans will continue until a stop trigger is received. That stop trigger will happen when the value of channel 500 crosses 2.05V from a higher previous value. The result will be one hundred readings, five on each channel before the stop trigger (50 total) and five on each channel (50 total) after the stop trigger. This data represents a window around the stop trigger event.

Example 4: Synchronizing Two or More Voltmeters

Some applications require that data be taken simultaneously on two or more different channels or sets of channels. A common way to accomplish this is to use two voltmeters and synchronize them. The HP44702 Voltmeter has a trigger out BNC connector to allow easy synchronization with other HP44702's. This application example will show a way to externally trigger a voltmeter, as well as, a way to synchronize them.

There are two basic ways of implementing this type of application. The first way is to simply set up both voltmeters identically and trigger them with the same signal. The voltmeters then work independently of each other after the initial trigger. Another approach to the problem is to have one voltmeter act as the master and the other voltmeter be the slave. In the first implementation, there is some risk of the voltmeters being slightly out of synchronization due to small variations in the pacer clocks of each voltmeter. The second implementation assures synchronization since only the pacer clock of the master is used to control all the voltmeters. This second type of implementation is used in the example.

Refer to the program listing in Figure SYNC during the following discussion. The two voltmeters in this application are set up and triggered in separate subroutines appropriately named MASTER and SLAVE. Then the two subroutines are called. The main program in the Series 200/300 controller pauses until the external triggers are received and all the readings taken. Then the readings are taken out of the voltmeters in REAL binary format.

The subroutine called MASTER sets up the master voltmeter first. The USE, SCANMODE and CONF commands do the initial set up. Note that the function set in the CONF command is two wire ohms. The slave will be set up to measure volts. It is not necessary that both voltmeters be set up for the same function, but they should have the same number of channels and scans in their respective scan sequences. The master will be set up to scan trigger externally so the SCSLOPE command is necessary to set the trigger edge. In this example, it is set to falling edge. The TRIGOUT command is the key to synchronizing multiple voltmeters. With TRIGOUT ON, measurement trigger pulses are output on the EXT 0 BNC. By connecting a cable from the EXT 0 BNC to the slave BNC external input, measurement trigger pulses from the master will trigger the slaves at the same time. In the example, the SPER, PRESCAN and CLWRITE commands are similar to those discussed previously. The last command in the

subroutine is SCTRIG EXT1. It sets the master voltmeter to accept external triggers at the EXT 1 input for each scan. These scan triggers need not come at regular intervals. Both voltmeters will remain synchronized with the measurement triggers being output at the EXT 0 BNC of the master. When the subroutine is called, the master will be waiting for external triggers to start its scans.

The slave voltmeter is set up in the subroutine called SLAVE. The slave voltmeter is set up with USE, SCTRIG HOLD, FUNC and TERM commands instead of the CONF command. Since the slave will be triggered from the master, its lowest level triggers, the measurement triggers, will be set to external. The SLOPE and TRIG commands set the trigger edge and source respectively. The ASCAN command causes the slave to automatically advance to the next scan while awaiting the next measurement trigger. Note that the number of PRESCANs and the number of channels in the scan list agree with the corresponding values set for the master. The last command in the subroutine is the SCTRIG SGL command. This command triggers the scan sequence and the slave voltmeter then waits for its external measurement triggers.

All that is necessary is for both subroutines to be called with the CALL command. The slave subroutine should be called first and then the master subroutine. When the slave subroutine is called, the first scan on the slave voltmeter begins and the voltmeter waits for measurement triggers to take readings. The measurement triggers will come from the master voltmeter. The master subroutine starts the whole measurement process. When it is called, the master will output trigger pulses after it is triggered from an external source. The controller is paused until all the readings are taken. Then the controller is continued and the readings are transferred to the controller.

Example 5: Programming Voltmeters in Real Time

In some applications, a single voltmeter set up won't do an adequate job of taking measurements. The ability to stop the voltmeter and reprogram it to a different state without unduly disrupting the measurement process could be valuable. This application example shows a method to reprogram the voltmeter in an interrupt service routine to a different state and then retrigger it to take readings again. All the readings are kept in the same array with the boundary indices of the different measurement states logged in variables to facilitate later evaluation.

Refer to the program listing in Figure RTPRG during the following discussion. In this application example, the HP44702 High Speed Voltmeter will be triggered by an interrupt on a digital input card to begin taking readings. The readings will be taken on a single channel at intervals of 20 milliseconds until a second interrupt is generated on the same digital input channel. Then the readings will be read into the array, the index logged, the voltmeter reprogrammed and the interrupt on the digital input redirected. When a third interrupt is generated on a different digital input channel, readings are read into the same array, the index logged and the voltmeter restarted. A fourth interrupt on the first digital input channel causes the measurement sequence to complete. The last readings are read out and the final array index saved.

Since this application is entirely interrupt driven, all the triggering and data transfer commands are in subroutines. The program listing shows the voltmeter being set up first to take continuous readings on one channel at 20 millisecond intervals. The range, function, scan delay and pacing, channel list and stop trigger are set accordingly. The interrupt subroutine called X issues the scan trigger to start the measurement sequence. Since a subroutine must exist before it can be called or have interrupts directed to it, the subroutines are entered into the HP3852 in reverse order. That means that the user must enter the last subroutine to be called in the HP3852 first.

```
260 OUTPUT 709;"SUB Z"
270 OUTPUT 709;"   STTRIG SGL"
280 OUTPUT 709;"   XRDGS 600 INTO A"
290 OUTPUT 709;"   INDEX? A INTO X3"
300 OUTPUT 709;"SUBEND"
```

This terminating subroutine consists of a stop trigger command which terminates the scan sequence, an XRDGS command to transfer the readings and a command to log the position of the of the last reading. All the other subroutines begin with commands similar to these three.

```
310 OUTPUT 709;"SUB Y"
320 OUTPUT 709;"   STTRIG SGL"
330 OUTPUT 709;"   XRDGS 600 INTO A"
340 OUTPUT 709;"   INDEX? A INTO X1"
350 OUTPUT 709;"   CLWRITE SENSE 500-501"
360 OUTPUT 709;"   SCDELAY 0,0.5"
370 OUTPUT 709;"   SCTRIG INT"
380 OUTPUT 709;"   ON INTR USE 216 CALL Z"
390 OUTPUT 709;"   ENABLE INTR USE 216"
400 OUTPUT 709;"SUBEND"
```

This subroutine has all the elements of the application. Measurements are gathered in the first three commands just as in SUB Z. The next command, CLWRITE, changes the channel list to include a total of two channels. The SCDELAY command changes the pacing of the scans. The time between measurement triggers on the two channels was set in the voltmeter initialization section with the SPER command. The voltmeter is then retriggered with SCTRIG. Finally, the interrupt on the digital input channel is redirected to use sub Z as a service routine and then enabled with ON INTR and ENABLE. Thus, the voltmeter has been reprogrammed and the interrupt redirected.

The starting subroutine, X, consists of just a scan trigger and an interrupt redirection set of commands. The last subroutine, T, is being called when an interrupt occurs on the second digital input. This subroutine gathers the data and logs the index, then it restarts the voltmeter. Since the second digital channel will be used as an interrupt source only once, it is not necessary to redirect it.

Before all the subroutines can be called, the interrupts must be set up for the digital input channels. This requires that the edges be set, the initial service subroutine designated and the channel and system

interrupts enabled. After this is done, the entire process is ready to begin.

When an edge appears at the first digital input, the subroutine called X is executed. The voltmeter begins to take readings at the rate of one reading every 20 milliseconds on channel 500. When the next valid edge occurs at the first digital input, the voltmeter is reprogrammed to take readings on channels 500 and 501 at 500 millisecond intervals. When an interrupt occurs at the second digital input channel, the voltmeter readings are gathered and the voltmeter restarted. Finally, at the last interrupt which occurs on the first digital input channel, the measurement sequence stops and the data is stored in the array.

Obviously, it takes some amount of time to reprogram the voltmeter and the interrupts. In this particular application, speed was not a critical issue. This type of approach will not work as well when interrupts are coming at close time intervals since the voltmeter may have just been restarted when a new interrupt occurs and will not have taken many readings. The example does illustrate that voltmeter measurement parameters can be changed in the middle of an application.

Example 6: Continuous Data Acquisition

There are often applications which require data to be taken for long periods of time. In many of these cases, a large amount of data is desired about an event but it is not known when the event will occur. The HP44702 voltmeter is a natural fit for this type of application since it can be programmed and left alone to take readings by itself. If less than 64K readings are desired, the output buffer of the HP44702B can accomodate all the data. However, if more than this number of readings is desired, other methods must be used. The basic idea presented here is to create a large circular buffer using high level software. Luckily, Series 200/300 BASIC with its rich I/O handling capabilities allows us to do this. This section will show three examples of continuous data acquisition with the HP3852 and a Series 200/300 controller running BASIC.

Hardware and Software Options

First we need to recognize some of the system limitations. The HP44702 can take readings at a rate of 100 thousand readings/second. Since each reading is two bytes, this translates to a transfer rate exceeding 200 thousand bytes/second. This can be a big problem if full voltmeter speed is needed. GPIO has the capability of transferring data at these kinds of speeds. If high speeds are not a necessity, other means besides GPIO can be used. Passing data over HPIB is an alternate possibility. However, some speed penalty is paid since all the data must first go through the HP3852 mainframe. Speeds achievable over HPIB are approximately 35 thousand readings/second. We will discuss the issues necessary to implement three different examples.

The I/O software capabilities of the Series 200/300 controllers running BASIC are quite good. The customer has two options available to move data from the HP3852 to the controller, the ENTER statement and the

TRANSFER statement. Each command involves choices in ease of programming, speed, the way data is stored and whether or not the task will run in foreground or background. Which command statement to use is often dictated by the way in which the data is to be stored in the controller. If the data can be stored in an array in the controller, both TRANSFER and ENTER may be used. If the data is to be stored in a file, then TRANSFER is the preferred command. The ENTER statement is the easiest to use. Since it runs serially, it is a foreground task and completely occupies the controller. For continuous data acquisition, customers may not want to tie up their controller in this way. The TRANSFER statement allows data transfer to run either in the foreground or in the background. Since the TRANSFER statement also gives the best data transfer speed, this note will concentrate on the use of the TRANSFER statement option.

Data Transfer Models

There are several ways to model the data transfer process. Each has advantages and disadvantages associated with it. One way to model the process is to pass the data into an array (see Fig. CDAARR). The ENTER command would be used normally in this situation. Since it runs serially, however, a speed penalty is encountered in some applications. There are other problems with this method also. If the number of readings to be transferred is not known, as is the general case in continuous data acquisition, the ENTER statement can hang up waiting for data and lock up the controller. This is a typical problem when trying to fill large arrays of data. Even when receiving data in small blocks, this model is cumbersome because of the difficulty of appending new data to data already in an array.

Passing the data into an array configured as a buffer affords much greater flexibility (see Fig. CDABUF). The customer can now use the TRANSFER command. This model turns out to be the easiest and most flexible to implement for continuous data acquisition. It can also be used for both HPIB and GPIO data transfers. It works like this. The high speed DVM takes data which is sent to the controller and stored in a buffer. The buffer is an array of data in the controller of sufficient size to hold all the data of interest. When the data fills the buffer, the TRANSFER statement terminates. Readings are then buffered in the local buffer of the high speed DVM until the TRANSFER is restarted. You can see that this is only possible if the transfer rate to the controller is faster than the reading rate. This is indeed the case and this model works well as a result.

A third model is one in which data is passed from the high speed DVM through the controller to a file (see Fig. CDAFL). Such a file can exist on disc or in a memory volume. The procedure is the same in either case. The model works like this. Readings are taken by the high speed DVM and sent to a buffer in the controller via GPIO. At the same time, readings are taken from the buffer and copied to a file residing in a memory volume or on disc. Again you can see that the reading rate must be slower than the transfer rate to the buffer or file if data is to be taken continuously. The writing speed of the disc is generally the limiting factor when sending data to a disc file. Check to be sure that

the disc drive can receive data at the speed necessary for the particular application. The buffer in the controller does not need to be as large as the amount of data to be recorded since it is just an intermediary in the process. TRANSFER statements are used for both the incoming and outgoing data transfers for maximum speed. For the following examples follow the flow chart for clarity (Fig. CDAFC).

Example 6A: Data Transfer Over GPIO to Buffer

Refer to Fig. CDA1 while reading about this example. The simplest example to implement is data transfer via GPIO to an array configured as a buffer in the controller. At first, using an array as a buffer may seem to be a restriction since only 32767 elements are allowed per dimension. However, more than one dimension can be used to obtain as large an array as necessary to hold the data. A typical group of set up commands would look like this.

```
        .
        .
        .
 200  INTEGER S300_buff(1:32767) BUFFER
        .
        .
        .
 310  ASSIGN @Hp3852 TO Hpib_addrs
 320  ASSIGN @Gpio TO Gpio_addrs
 330  ASSIGN @Path_3852 TO Gpio;WORD
 340  ASSIGN @S300_buff TO BUFFER S300_buff(*);FORMAT OFF
        .
        .
        .
```

In line 200, the INTEGER statement configures the array as a buffer with the BUFFER suffix. The first two ASSIGN statements are straighforward. The third ASSIGN statement sets up the GPIO path with the WORD attribute. This path will be used to handle the 16-bit data coming from the DVM. The FORMAT OFF attribute in the last ASSIGN statement is used to increase speed and insure data integrity since the data will be in a form that is a valid integer, even though the data is in a packed format.

The DVM must now be programmed to take the readings. In this example, the DVM will be set up to take DC voltage readings on one channel continuously. The high speed FET multiplexer will be attached to the DVM with the ribbon cable and will send its readings out over the GPIO port.

```
        .
        .
        .
 440  OUTPUT @Hp3852;"USE 600; SCANMODE ON; SCTRIG HOLD"
 450  OUTPUT @Hp3852;"FUNC DCV; RANGE 2.5; TERM RIBBON; RDGS GPIO"
 460  OUTPUT @Hp3852;"NRDGS 1; SPER 10E-6; SCDELAY 16E-3; ASCAN ON"
 470  OUTPUT @Hp3852;"CLWRITE SENSE 500; STTRIG HOLD"
```

```
      .
      .
      .
  510 CONTROL Gpio_addrs,2;5      ! Set control line 0 low, DVM expects addr
  520 OUTPUT Gpio_addrs USING "#,W";4   ! Set read register address to 4.
  530 CONTROL Gpio_addrs,2;4      ! Set control line 0 high, DVM sends data.
      .
      .
      .
```

The key command to set the DVM up to take continuous readings is the STTRIG HOLD command in line 470. This command disables the stop trigger on the high speed DVM which allows the DVM to continue taking readings indefinitely. With ASCAN ON in line 460, only an initial trigger is required. Subsequent scans start automatically. Note the CONTROL statements. They are necessary to set up the GPIO transfer. Consult the Plug-In Accessories Programming Manual (P/N 03852-90002, pp. 3-49 through 3-65) for further detail. The SPER and SCDELAY commands set the pacing of the readings.

To implement the circular buffer in software, we need to be able to restart the transfer process when the buffer in the controller fills and stop the process when the data of interest has been taken. Interrupts are used to accomplish this. The Series 200/300 is set up to interrupt on an END OF TRANSFER (EOT) and on an SRQ in this example. The interrupt in the HP3852 is set to pull SRQ on a rising edge of the first channel of a digital input card in slot 2.

```
      .
      .
      .
  580 ON EOT @Path_3852,5 GOSUB Xfer_done
  590 ON INTR 7 GOSUB Terminate     ! HPIB interrupt (SRQ)
  610 ENABLE INTR 7;SRQ_mask
  620 OUTPUT @Hp3852;"ENABLE INTR SYS"
  630 OUTPUT @Hp3852;"RQS ON; RQS INTR"
  640 OUTPUT @Hp3852;"EDGE LH USE 216"
  650 OUTPUT @Hp3852;"ENABLE INTR USE 216"
      .
      .
      .
```

All that is required now is to trigger the DVM and start the data transfer. A TRANSFER statement is executed which will terminate on a count and run continuously. Then a trigger is sent to the high speed DVM to start taking readings.

```
      .
      .
      .
  670 TRANSFER @Path_3852 TO @S300_buff;COUNT 65534,CONT
  680 OUTPUT @Hp3852;"SCTRIG SGL"
      .
      .
```

The high speed DVM in the HP3852 will now take readings and transfer them until the TRANSFER terminates. At that time, an EOT interrupt will be generated in the controller and the TRANSFER can be restarted in the interrupt subroutine called Xfer_done.

```
890  Xfer_done:RESET @S300_buff
900  TRANSFER @Path_3852 TO @S300_buff;COUNT 65534,CONT
920  RETURN
```

When the SRQ interrupt comes from the HP3852, a termination subroutine is called.

```
950  Terminate: WAIT .05       ! Take more data after the interrupt.
960  OFF EOT @Path_3852        ! Cancel inbound EOT.
970  ABORTIO @Path_3852        ! Clean up GPIO path.
980  ASSIGN @Path_3852 TO *
990  STATUS @S300_buff,3;Fill_ptr  ! Read ending fill pointer.
```

Reading the fill pointer in line 990 allows the programmer to know where the last reading was entered. The buffer can then be read back beginning with the last reading.

Example 6B: Data Transfer Over GPIO to File

Refer to Fig. CDA2 while reading about this example. This example is useful for storing the data of interest in a file. The file can be on a memory volume or on a disc. If storing data on disc, pay careful attention to the transfer rates available for the particular disc you are using. A good guide for approximate transfer rates can be found in Table 4-1, p.4-12 of the HP44456 System Software Manual (P/N 44456-90001). The set up commands look like this.

```
280  INITIALIZE ":MEMORY,0,1",512    ! Create memory volume
290  CREATE BDAT "File_name:MEMORY,0,1",384,256 ! Total bytes=98304
```

```
330  ASSIGN @Hp3852 TO Hpib_addrs
```

```
340  ASSIGN @Gpio TO Gpio_addrs
350  ASSIGN @Path_3852 TO Gpio_addrs;WORD
360  ASSIGN @S300_buff TO BUFFER [Buff_size];FORMAT OFF
370  ASSIGN @File_name TO "File_name:MEMORY,0,1"
       .
       .
       .
```

Lines 280 and 290 create a memory volume and a file on that volume. Note that the file is a binary data file. Also note that in line 360 an unnamed buffer is declared. This buffer will act as the intermediary buffer to hold the data between the inbound and outbound TRANSFER statements. The buffer size should be an integral number of 256 byte blocks for greatest speed. A typical size is 16 to 32 Kbytes.

The DVM set up is identical to the set up in the previous example. The interrupt set up is also the same except that the EOT is generated from the destination file TRANSFER.

```
       .
       .
       .
600  ON EOT @File_name,5  GOSUB Xfer_done
       .
       .
       .
```

Starting the DVM and transferring the readings is considerably more involved in this example.

```
       .
       .
       .
710  TRANSFER @Path_3852 TO @S300_buff;COUNT Xfer_length,
           EOR (COUNT Rec_length),CONT
720  TRANSFER @S300_buff TO @File_name;COUNT Xfer_length,
           EOR (COUNT Rec_length),CONT
730  OUTPUT @Hp3852;"SCTRIG SGL"
       .
       .
       .
```

The variables Xfer_length and Rec_length should be an integral number of 256 byte blocks for greatest speed. Xfer_length is the total number of bytes of data of interest. It is normally larger than the size of the buffer in the Series 200/300 controller. It is important that Rec_length be an integral number of blocks to speed up DMA transfer of the data. When an EOT is generated at the destination file, the subroutine Xfer_done is called as before. However, it is necessary to reset both the record and byte pointers to the start. This is done with a CONTROL command.

```
       .
       .
```

```
910  Xfer_done: CONTROL @File_name,5;1,1   ! Reset record and byte
                                             pointers to start.
920  TRANSFER ...      ! Same as line 710
930  TRANSFER ...      ! Same as line 720
940  RETURN
```

The HP3852, Series 200/300 controller and disc, if used, are now taking data continuously. To stop the process, the HP3852 will generate an SRQ from a digital input. A termination routine similar to the previous example is called in the controller to complete the data transfer.

```
970  Terminate: WAIT .05       ! Take more data after the interrupt.
980  OFF EOT @File_name        ! Cancel outbound EOT.
990  ABORTIO @Path_3852        ! Clean up GPIO path.
1000 ASSIGN @Path_3852 TO *
1010 ABORTIO @File_name        ! Clean up file path
1020 STATUS @File_name,5;End_rec_num,End_byte_num
1030 ASSIGN @File_name TO *
```

In this example, the ending record and ending byte numbers are stored so that the data may be read back starting with the last entry (see line 1020).

Example 6C: Data Transfer Over HPIB to Buffer

Refer to Fig. CDA3 containing the listing for this example. This example is different from the previous two since only the HPIB path is used. A different interrupt structure is used in the HP3852. Continuous data acquisition is still implemented but in a different way in the HP3852 since both data and control is transmitted via the HPIB link. The HP3852 mainframe must move the readings from the high speed DVM into an array in the HP3852. The array is then read out to the Series 200/300 controller. This process takes place serially so speed suffers. With this method, 35 thousand readings/second can be achieved.

It may be instructive to compare Example 6A to this example to understand the differences between the HPIB implementation and the GPIO implementation. The set up commands look like this.

```
200  INTEGER S300_buff(1:24576) BUFFER
```

```
      .
      .
300   ASSIGN @Hp3852 TO Hpib_addrs
310   ASSIGN @Path_3852 TO Hpib_addrs;FORMAT OFF
320   ASSIGN @S300_buff TO BUFFER S300_buff(*);FORMAT OFF
      .
      .
      .
```

As in Example 6A, the INTEGER statement declares a buffer in the controller. For greatest speed, note that the ASSIGN statements in lines 310 and 320 use the FORMAT OFF attribute.

The HP3852 and high speed DVM set up is also similar but somewhat different than Example 6A. The DVM will be programmed to take readings over the ribbon cable and one channel will be scanned repeatedly.

```
      .
      .
      .
400   OUTPUT @Hp3852;"PACKED A(8191);INTEGER X"
430   OUTPUT @Hp3852;"USE 600; SCANMODE ON; SCTRIG HOLD; FUNC DCV"
440   OUTPUT @Hp3852;"RANGE 2.5; TERM RIBBON; RDGSMODE BURST; SPER 27.5E-6"
450   OUTPUT @Hp3852;"SCDELAY 16E-3; POSTSCAN 4096; ASCAN ON"
460   OUTPUT @Hp3852;"CLWRITE SENSE 500; STTRIG HOLD"
      .
      .
      .
```

A packed array and an integer variable are declared in line 400. The array will be used to buffer readings in the HP3852 mainframe until they are read out. The variable will be used to hold a status word. In this particular example, a POSTSCAN command is used in line 450 to ensure that enough data will be available after the stop trigger to be read using an XRDGS command which will be described later. Also, the high speed DVM is set up to interrupt when there is room for only 4096 more readings in its buffer with the RDGSMODE BURST statement. This is done so that readings can be moved in large blocks into the memory array and out to the controller to increase speed.

The interrupts in this example are quite different than the previous examples. Two interrupts will be active in the HP3852 as well as the two in the controller which have already been described. One of the interrupt subroutines in the HP3852 will transfer the data from the high speed DVM into the array in the HP3852 and then from the internal array to the controller when the RDGSMODE BURST interrupt occurs. The other subroutine will send a stop trigger to the high speed DVM to end scanning. Then it will transfer a final block of data to the controller and assert SRQ. The terminating interrupt will again be generated by a digital input in the HP3852.

```
      .
      .
      .
```

```
540  ON EOT @Path_3852,5 GOSUB Xfer_done    ! Same as Example 1.
550  ON INTR 7 GOSUB Terminate    ! HPIB interrupt (SRQ)
570  ENABLE INTR 7;SRQ_mask
580  OUTPUT @Hp3852;"SUB INTROUT; XRDGS 600,4096 INTO A; VREAD A,PACK"
590  OUTPUT @Hp3852;"   ENABLE INTR USE 600; SUBEND"
600  OUTPUT @Hp3852;"SUB XIT; STTRIG SGL USE 600; XRDGS 600,4096 INTO A"
610  OUTPUT @Hp3852;"   VREAD A,PACK; STA? INTO X; SRQ; BEEP; SUBEND"
620   !
630  OUTPUT @Hp3852;"ON INTR USE 216 CALL XIT"   ! Call termination sub.
640  OUTPUT @Hp3852;"ON INTR USE 600 CALL INTROUT"  ! Call subroutine to
                                                      transfer data.
650  OUTPUT @Hp3852;"ENABLE INTR USE 600"
660  OUTPUT @Hp3852;"RQS FPS; RQS ON"     ! Enable SRQ and mask.
670  OUTPUT @Hp3852;"EDGE LH USE 216"     ! Set edge on digital card.
680  OUTPUT @Hp3852;"ENABLE INTR USE 216"
690  OUTPUT @Hp3852;"ENABLE INTR SYS"     ! Enable 3852 interrupts.
       .
       .
       .
```

Lines 580 through 590 contain the subroutine which sends the data to the controller. It is an interrupt routine that is called when an interrupt occurs from the high speed DVM (see RDGSMODE BURST in line 440). By using the XRDGS command followed by the VREAD command, we gain two advantages. The first is a speed increase since the command set up times are only incurred once for each command. The second advantage is that the HP3852 mainframe is not tied up sending data continuously and can thus do other tasks and service other interrupts. If you intend to use the HP3852 with this interrupt structure and the other tasks in the HP3852 take a long time to execute, the DVM buffer may fill before the interrupt subroutine can be called to take the data out. Be careful that this does not occur since the DVM will stop taking data when its buffer is full.

Lines 600 through 610 contain the termination subroutine. An STTRIG SGL command signals the DVM to begin the postscans. The last block of readings is then sent and SRQ is asserted. This signals the controller to begin its termination subroutine. The interrupts are now all activated.

The interrupts are now all activated. The whole data acquisition process must now be started by triggering the DVM and executing the TRANSFER statement in the controller.

```
       .
       .
       .
710 Start: OUTPUT @Hp3852;"SCTRIG INT"
720  TRANSFER @Path_3852 TO @S300_buff;COUNT 49152,EOR (END)
       .
       .
       .
```

The COUNT parameter in line 720 corresponds to the number of bytes in the buffer declared in the controller. The EOR (END) parameter is

specified so that the TRANSFER will not terminate since EOI is sent by the HP3852 after each block of data. When the buffer in the controller is full, an EOT interrupt is generated in the controller. The subroutine Xfer_done is called which resets the buffer and restarts the TRANSFER.

```
      .
      .
      .
930 Xfer_done: RESET @S300_buff
940  TRANSFER @Path_3852 TO @S300_buff;COUNT 49152, EOR (END)
960  RETURN
      .
      .
      .
```

The HP3852 and Series 200/300 controller are now taking data continuously. When the signal to stop is received by the digital input card, the HP3852 goes into its termination subroutine and then signals the controller via the SRQ line to go into its termination subroutine. The termination subroutine in the controller is identical to the one in Example 1.

Final Considerations

Three different examples of continuous data acquisition have been described. They are by no means an exhaustive list of possibilities for applications of this type. They are presented here as examples of the capabilities of the high speed DVM in the HP3852 working with a Series 200/300 controller. Many other variations are possible. For instance, error checking is available over GPIO though it was not demonstrated in any of the examples. Additional control capabilities are also available over GPIO. Examples of both HPIB and GPIO data transfers were presented. Speed as well as some control capability is sacrificed with HPIB, however, for many applications HPIB data transfer may be completely adequate.

Other triggering modes are also available. A set number of prescans and postscans may be desired. This can be easily accomplished by changing the programming of the high speed DVM in the HP3852. External triggering for either scan or stop triggers can also be programmed. Considerable flexibility is possible in this area.

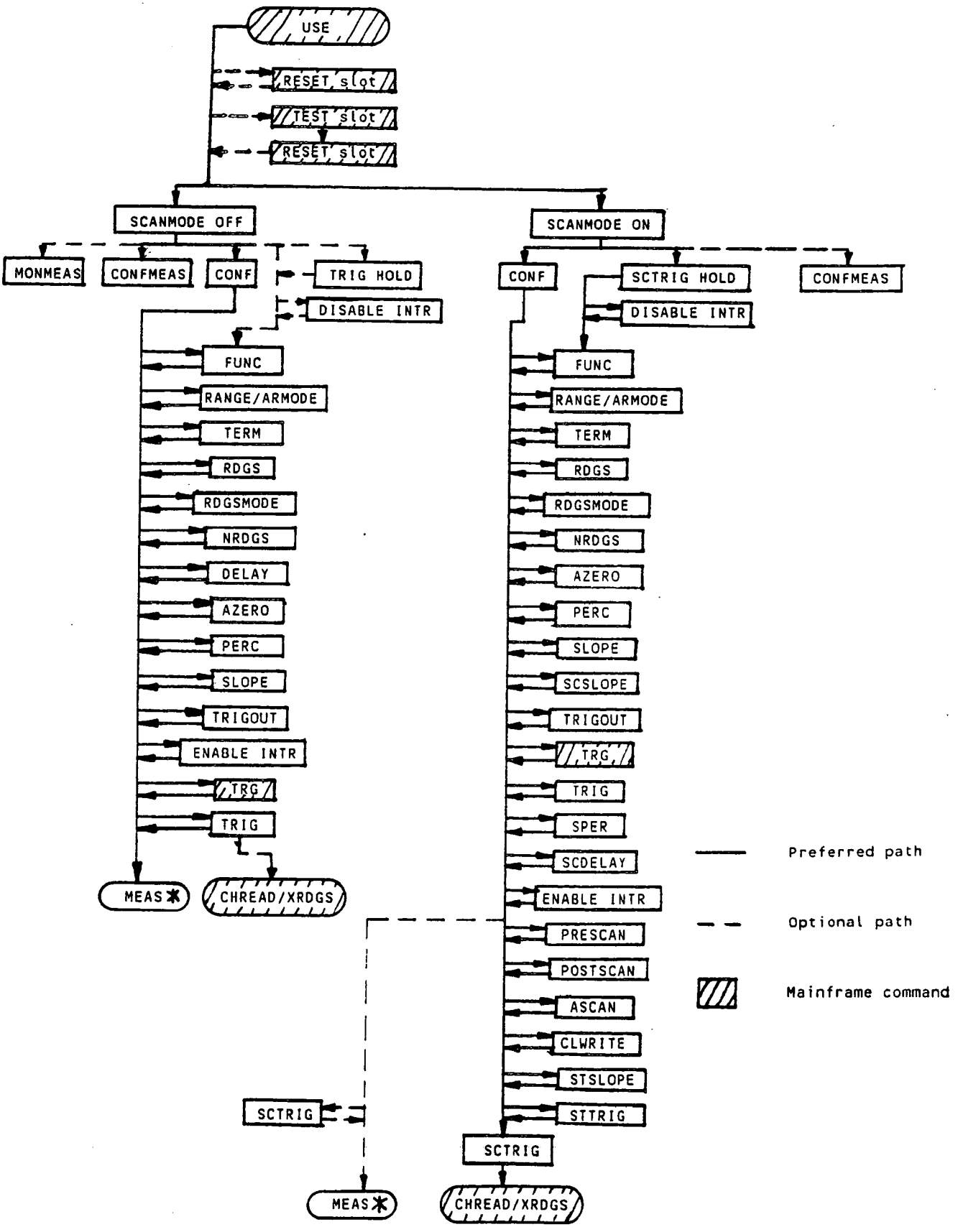
In these examples no mention was made as to how to get data out of the buffers or files. The key to this task is to save the pointers to the last entry in the buffer or file in the controller. Then the data can be read out of the buffer or file by beginning at the last entry and reading the data.

Where maximum speed is not important, more complex tasks can be performed in either the HP3852 or the Series 200/300 controller. The possibilities are many and varied. The examples presented here should be an aid in getting started on a continuous data acquisition application with the HP44702 and a Series 200/300 controller.

IV. Summary

Several application examples have been presented. They have ranged from simple examples like the first two which used the MEAS command, to very complex examples like the continuous data acquisition examples which used the low level commands. A GPIO example showed the capability of the 44702 with this interface. One example focused on the use of interrupts. Then interrupts were used in later examples. One example demonstrated a way to gather data around an event. Then the stop trigger feature was used in other examples. The external triggering capabilities were demonstrated in an example that showed how to synchronize two or more voltmeters. Interrupt driven programming was shown as an example also. The most difficult applications are the continuous data acquisition examples. These applications require complex interraction with the Series 200/300 controller, though they are similar in concept to gathering data around an event. The window is much larger, however. The low level command set gives the user access to a wide range of interrupt, pacing and triggering possibilities. The HP44702/HP3852 data acquisition system can solve some very difficult application problems with its capability and flexibility.

Programming Tree
for 44702
High Speed DVM



* MEAS command may change certain settings.

Figure PT2

```
10  ! THIS PROGRAM SHOWS A WAY TO TAKE DCV READINGS IN THE SYSTEM
20  ! MODE. THIS IS A BASIC APPLICATION AND USES CONF AND MEAS.
30  !
40  !
50  !
60  ASSIGN @Comp TO 709
70  ASSIGN @Cformoff TO 709;FORMAT OFF
80  REAL A(1:20)
90  !
100 CLEAR @Comp
110 OUTPUT @Comp;"RESET"
120 OUTPUT @Comp;"PACKED RD(39)"
130 OUTPUT @Comp;"SUB HS_RDGS"
140 OUTPUT @Comp;"      USE 600"
150 OUTPUT @Comp;"      SCANMODE OFF"
160 OUTPUT @Comp;"      CONF DCV"
170 OUTPUT @Comp;"      RANGE 2.5"
180 OUTPUT @Comp;"      TERM INT"
190 OUTPUT @Comp;"      RDGS SYS"
200 OUTPUT @Comp;"      RDGSMODE DAV"
210 OUTPUT @Comp;"      MEAS DCV 100-119 INTO RD"
220 OUTPUT @Comp;"      VREAD RD,RL64"
230 OUTPUT @Comp;"SUBEND"
240 !
250 OUTPUT @Comp;"CALL HS_RDGS"
260 ENTER @Cformoff;A(*)
270 PRINT A(*)
280 END
```

Figure SYS

```
10  ! THIS PROGRAM SHOWS A WAY TO TAKE DCV READINGS IN THE SCANNER
20  ! MODE. THIS IS A BASIC APPLICATION AND USES CONF AND MEAS.
30  !
40  !
50  !
60  ASSIGN @Comp TO 709
70  ASSIGN @Cformoff TO 709;FORMAT OFF
80  REAL A(1:20)
90  !
100 CLEAR @Comp
110 OUTPUT @Comp;"RESET"
120 OUTPUT @Comp;"PACKED RD(39)"
130 OUTPUT @Comp;"SUB HS_RDGS"
140 OUTPUT @Comp;"      USE 600"
150 OUTPUT @Comp;"      SCANMODE ON"
160 OUTPUT @Comp;"      CONF DCV"
170 OUTPUT @Comp;"      RANGE 2.5"
180 OUTPUT @Comp;"      TERM RIBBON"
190 OUTPUT @Comp;"      RDGS SYS"
200 OUTPUT @Comp;"      RDGSMODE DAV"
210 OUTPUT @Comp;"      MEAS DCV 500-504,500-504,500-504,500-504 INTO RD"
220 OUTPUT @Comp;"      VREAD RD,RL64"
230 OUTPUT @Comp;"SUBEND"
240 !
250 OUTPUT @Comp;"CALL HS_RDGS"
260 ENTER @Cformoff;A(*)
270 PRINT A(*)
280 END
```

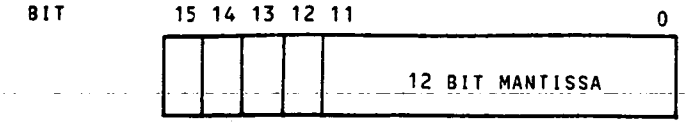
Figure SCN

```
10   ! THIS PROGRAM DEMONSTRATES A SIMPLE EXAMPLE OF USING INTERRUPTS
20   ! WITH THE HIGH SPEED DVM. THE MEASUREMENT SEQUENCE IS STARTED.
30   ! THEN AN INTERRUPT IS GENERATED AT THE END OF THE SEQUENCE, AND
40   ! THE READINGS ARE TRANSFERRED.
50   !
60   !
70   ASSIGN @Comp TO 709
80   ASSIGN @Cformoff TO 709;FORMAT OFF
90   REAL A(1:10)
100  !
110  CLEAR @Comp
120  OUTPUT @Comp;"RESET"
130  OUTPUT @Comp;"SUB TAKE_RDG"
140  OUTPUT @Comp;"    USE 600"
150  OUTPUT @Comp;"    SCANMODE ON"
160  OUTPUT @Comp;"    CONF DCV"
170  OUTPUT @Comp;"    RANGE .25"
180  OUTPUT @Comp;"    RDGSMODE END"
190  OUTPUT @Comp;"    SPER 250E-6"
200  OUTPUT @Comp;"    ENABLE INTR"
210  OUTPUT @Comp;"    CLWRITE SENSE 500-509"
220  OUTPUT @Comp;"    SCTRIG SGL"
230  OUTPUT @Comp;"SUBEND"
240  !
250  OUTPUT @Comp;"SUB X"
260  OUTPUT @Comp;"    XRDGS 600,RL64"
270  OUTPUT @Comp;"SUBEND"
280  !
290  OUTPUT @Comp;"ON INTR USE 600 CALL X"
300  OUTPUT @Comp;"ENABLE INTR SYS"
310  !
320  OUTPUT @Comp;"CALL TAKE_RDG"
330  ENTER @Cformoff;A(*)
340  PRINT A(*)
350  END
```

Figure INTR

```
10   ! THIS IS A PROGRAM THAT DEMONSTRATES GPIO TRIGGERING, INTERRUPT
20   ! HANDLING AND DATA TRANSFER. THE HIGH SPEED DVM IS SET UP OVER
30   ! HPIB, THEN TRIGGERED OVER GPIO. AFTER AN INTERRUPT IS GENERATED
40   ! ON GPIO, THE DATA IS READ OVER GPIO. FINALLY, THE DATA IS
50   ! UNPACKED IN THE CONTROLLER.
60   !
70   !
80    ASSIGN @Comp TO 709
90    ASSIGN @Gpio TO 12
100   Gpio=12
110   !
120   INTEGER A(1:10),Cnt
130   ON INTR Gpio GOSUB Get_rdgs
140   Eir_=1
150   ENABLE INTR Gpio;Eir_
160   !
170   CLEAR @Comp
180   OUTPUT @Comp;"RESET"
190   OUTPUT @Comp;"USE 600"
200   OUTPUT @Comp;"SCANMODE ON"
210   OUTPUT @Comp;"CONF OHM10K"
220   OUTPUT @Comp;"RDGS GPIO"
230   OUTPUT @Comp;"RDGSMODE BURST"
240   OUTPUT @Comp;"SCSLOPE HL"      ! SET TRIGGER SENSE
250   OUTPUT @Comp;"SPER .10"
260   OUTPUT @Comp;"ENABLE INTR"
270   OUTPUT @Comp;"CLWRITE COM 500-509"
280   OUTPUT @Comp;"SCTRIG GPIO"
290   !
300   WHILE NOT BIT(SPOLL(@Comp),4)  ! WAIT UNTIL READY
310   END WHILE
320   !
330   CONTROL Gpio,2;5            ! SET DATA REGISTER
340   OUTPUT @Gpio USING "#,W";4
350   CONTROL Gpio,2;4
360   !
370   CONTROL Gpio,2;6          ! TRIGGER DVM
380   CONTROL Gpio,2;4
390   !
400   Cnt=0
410   WHILE (Cnt<1000)
420     Cnt=Cnt+1
430     DISP Cnt
440   END WHILE
450   STOP
460   !
470 Get_rdgs:ENTER @Gpio USING "#,W":A(*)
480   !
490   !
500 Unpack:REAL R(0:3),M,V(1:10),X
510   DATA 256.,32.,4.,1.
520   READ R(*)
530   X=.001              ! FACTOR FOR OHM10K
540   FOR I=1 TO 10
550     M=BINAND(A(I),4095)
560     IF A(I)>0 OR M=4095. THEN
570       V(I)=1.0E+38
580     ELSE
590       V(I)=M*.0025/R(BINAND(SHIFT(A(I),13),3))/X
600       IF BIT(A(I),12) THEN V(I)=-V(I)
610     END IF
620   NEXT I
630   PRINT V(*)
640   !
650   !
660   END
```

Figure GP10



BIT 15= Good reading bit

BIT 14-13= Range bits

BIT 12= Sign bit

BIT 11-0= 12 Bit mantissa

BITS	14/13	RANGE FACTOR	FUNCTION	OHMS FACTOR
	0 0	256	DCV	1
	0 1	32	OHM10K	.001
	1 0	4	OHM100K	.0001
	1 1	1	OHM1M	.00001

Figure UNPK

```
10   ! THIS PROGRAM DEMONSTRATES AN EXAMPLE OF DATA BEING COLLECTED
20   ! BOTH BEFORE AND AFTER AN EVENT. THE HIGH SPEED DVM IS SET UP
30   ! AND TRIGGERED. IT WILL TAKE READINGS UNTIL THE VALUE ON THE
40   ! FIRST CHANNEL CROSSES THE THRESHOLD IN THE PROPER DIRECTION.
50   ! THEN THE READINGS WILL BE TAKEN OUT OF THE DVM.
60   !
70   !
80   ASSIGN @Comp TO 709
90   ASSIGN @Cformoff TO 709;FORMAT OFF
100  !
110  DIM A(1:100)
120  !
130  CLEAR @Comp
140  OUTPUT @Comp;"RESET"
150  OUTPUT @Comp;"USE 600"
160  OUTPUT @Comp;"SCANMODE ON"
170  OUTPUT @Comp;"SCTRIG HOLD"
180  OUTPUT @Comp;"FUNC DCV"
190  OUTPUT @Comp;"RANGE 2.5"
200  OUTPUT @Comp;"TERM RIBBON"
210  OUTPUT @Comp;"RDGS SYS"
220  OUTPUT @Comp;"RDGSMODE COMPLETE"
230  OUTPUT @Comp;"PERC 80"
240  OUTPUT @Comp;"SPER .05"
250  OUTPUT @Comp;"SCDELAY .01,.5"
260  OUTPUT @Comp;"ENABLE INTR"
270  OUTPUT @Comp;"PRESCAN 5"
280  OUTPUT @Comp;"POSTSCAN 5"
290  OUTPUT @Comp;"CLWRITE SENSE 500-509"
300  OUTPUT @Comp;"STSLOPE HL"
310  OUTPUT @Comp;"STTRIG MEAS"
320  !
330  OUTPUT @Comp;"SUB X"
340  OUTPUT @Comp;"    XRDGS 600,100,RL64"
350  OUTPUT @Comp;"SUBEND"
360  !
370  OUTPUT @Comp;"ON INTR USE 600 CALL X"
380  OUTPUT @Comp;"ENABLE INTR SYS"
390  !
400  OUTPUT @Comp;"SCTRIG INT"
410  !
420  ENTER @Cformoff;A(*)
430  PRINT A(*)
440  !
450  END
```

Figure EVNT

```
10   ! THIS PROGRAM DEMONSTRATES A WAY TO EXTERNALLY TRIGGER AND
20   ! SYNCHRONIZE TWO OR MORE 44702 HIGH SPEED DVM'S. THE DVM
30   ! IN SLOT 2 IS THE MASTER AND THE DVM IN SLOT 6 IS THE SLAVE.
40   !
50   !
60   ASSIGN @Comp TO 709
70   ASSIGN @Cformoff TO 709;FORMAT OFF
80   REAL A(1:100),B(1:100)
90   !
100  CLEAR @Comp
110  OUTPUT @Comp;"RESET"
111  OUTPUT @Comp;"SUB MASTER"
120  OUTPUT @Comp;"      USE 200"
130  OUTPUT @Comp;"      SCANMODE ON"
140  OUTPUT @Comp;"      CONF OHM10K"
160  OUTPUT @Comp;"      SCSLOPE HL"
170  OUTPUT @Comp;"      TRIGOUT ON"
180  OUTPUT @Comp;"      SPER .050"
190  OUTPUT @Comp;"      PRESCAN 10"
200  OUTPUT @Comp;"      CLWRITE COM 100-109"
210  OUTPUT @Comp;"      SCTRIG EXT1"
211  OUTPUT @Comp;"SUBEND"
220  !
230  !
231  OUTPUT @Comp;"SUB SLAVE"
232  OUTPUT @Comp;"      USE 600"
233  OUTPUT @Comp;"      SCANMODE ON"
234  OUTPUT @Comp;"      SCTRIG HOLD"
235  OUTPUT @Comp;"      FUNC DCV,2.5"
236  OUTPUT @Comp;"      TERM RIBBON"
238  OUTPUT @Comp;"      SLOPE HL"
239  OUTPUT @Comp;"      TRIG EXT0"
240  OUTPUT @Comp;"      PRESCAN 10"
241  OUTPUT @Comp;"      ASCAN ON"
242  OUTPUT @Comp;"      CLWRITE SENSE 500-509"
243  OUTPUT @Comp;"      SCTRIG SGL"
244  OUTPUT @Comp;"SUBEND"
245  !
246  !
248  OUTPUT @Comp;"CALL SLAVE"
249  OUTPUT @Comp;"CALL MASTER"
250  !
251  !
252  PAUSE
253  !
254  !
256  OUTPUT @Comp;"XRDGS 200,RL64"
257  ENTER @Cformoff;A(*)
258  OUTPUT @Comp;"XRDGS 600,RL64"
259  ENTER @Cformoff;B(*)
261  !
262  !
270  PRINT A(*)
271  PRINT B(*)
272  !
273  !
280  END
```

Figure SYNC

```
10    ! THIS IS A PROGRAM THAT ILLUSTRATES PROGRAMMING THE 44702
20    ! HIGH SPEED VOLTMETER OF THE HP3852 DURING AN APPLICATION.
30    ! THE HS DVM IS STARTED BY AN INTERRUPT ON A DIGITAL CARD.
40    ! THE DVM TAKES READINGS AT 20 MS INTERVALS UNTIL ANOTHER
50    ! INTERRUPT. THEN THE HS DVM IS REPROGRAMMED TO TAKE READINGS
60    ! ON TWO CHANNELS AT 200MS INTERVALS UNTIL A THIRD INTERRUPT.
70    ! THE THIRD INTERRUPT COMES FROM A DIFFERENT SOURCE THAN THE
80    ! FIRST INTERRUPT. THE THIRD INTERRUPT SUBROUTINE TRANSFERS
90    ! THE READINGS, LOGS THE INDEX AND RESTARTS THE VOLTMETER
100   ! UNTIL A FINAL INTERRUPT. THE LAST INTERRUPT OCCURS ON THE
110   ! ORIGINAL DIGITAL INPUT CHANNEL. THE READINGS ARE STORED IN
120   ! AN ARRAY AND ALL THE DATA BOUNDARIES (INDICES) ARE LOGGED.
130   !
140   ASSIGN @Comp TO 709
150   ASSIGN @Cformoff TO 709;FORMAT OFF
160   DIM A(500)
170   !
180 Dvm_setup:   !
190         OUTPUT @Comp;"RST;INBUF ON; OUTBUF ON"
200         OUTPUT @Comp;"REAL X1,X2,X3,A(500)"
210         OUTPUT @Comp;"USE 600; SCANMODE ON; SCTRIG HOLD; FUNC DCV"
220         OUTPUT @Comp;"RANGE 2.5; TERM RIBBON; TRIG INT; SPER 0"
230         OUTPUT @Comp;"SCDELAY 0,.02; CLWRITE SENSE 500; STTRIG HOLD"
240   !
250 Dig_intr:   !
260         OUTPUT @Comp;"SUB Z"                    ! THIS IS THE FINAL SUBROUTINE
270         OUTPUT @Comp;"    STTRIG SGL"
280         OUTPUT @Comp;"    XRDGS 600 INTO A"
290         OUTPUT @Comp;"    INDEX? A INTO X2"
300         OUTPUT @Comp;"SUBEND"
310         OUTPUT @Comp;"SUB Y"                      ! THIS IS THE SECOND
320         OUTPUT @Comp;"    STTRIG SGL USE 600"     ! INTERRUPT SUBROUTINE
330         OUTPUT @Comp;"    XRDGS 600 INTO A"
340         OUTPUT @Comp;"    INDEX? A INTO X1"
350         OUTPUT @Comp;"    CLWRITE SENSE 500-501"
360         OUTPUT @Comp;"    SCDELAY 0,0.5 USE 600"
370         OUTPUT @Comp;"    SCTRIG INT USE 600"
380         OUTPUT @Comp;"    ON INTR USE 216 CALL Z"
390         OUTPUT @Comp;"    ENABLE INTR USE 216"
400         OUTPUT @Comp;"SUBEND"
410         OUTPUT @Comp;"SUB X"                    ! THIS SUB STARTS EVERYTHING
420         OUTPUT @Comp;"    SCTRIG INT USE 600"
430         OUTPUT @Comp;"    ON INTR USE 216 CALL Y"
440         OUTPUT @Comp;"    ENABLE INTR USE 216"
450         OUTPUT @Comp;"SUBEND"
460         OUTPUT @Comp;"SUB T"                    ! THIS IS THE THIRD INTERRUPT
470         OUTPUT @Comp;"   BEEP;STTRIG SGL"       ! SUBROUTINE
480         OUTPUT @Comp;"   XRDGS 600 INTO A"
490         OUTPUT @Comp;"   INDEX? A INTO X3"
500         OUTPUT @Comp;"   SCTRIG INT USE 600"
510         OUTPUT @Comp;"SUBEND"
520         OUTPUT @Comp;"EDGE LH USE 216;EDGE LH USE 217"
530         OUTPUT @Comp;"ON INTR USE 216 CALL X;ENABLE INTR USE 216"
540         OUTPUT @Comp;"ON INTR USE 217 CALL T;ENABLE INTR USE 217"
550         OUTPUT @Comp;"ENABLE INTR SYS"
560   !
570   PAUSE
580   OUTPUT @Comp;"VREAD A,RL64"     ! GET ARRAY OF READINGS
590   ENTER @Cformoff;A(*)
600   OUTPUT @Comp;"VREAD X1,RL64"    ! GET FIRST DATA BOUNDARY
610   ENTER @Cformoff;X1
620   OUTPUT @Comp;"VREAD X2,RL64"    ! GET SECOND DATA BOUNDARY
630   ENTER @Cformoff;X2
640   OUTPUT @Comp;"VREAD X3,RL64"    ! GET THIRD DATA BOUNDARY
650   ENTER @Cformoff;X3
660   PRINT TABXY(5,5);"X1 IS ";X1
670   PRINT TABXY(5,8);"X2 IS ";X2
680   PRINT TABXY(5,11);"X3 IS ";X3
690   END
```

Figure RTPRG

CONTINUOUS DATA ACQUISITION FLOW CHART

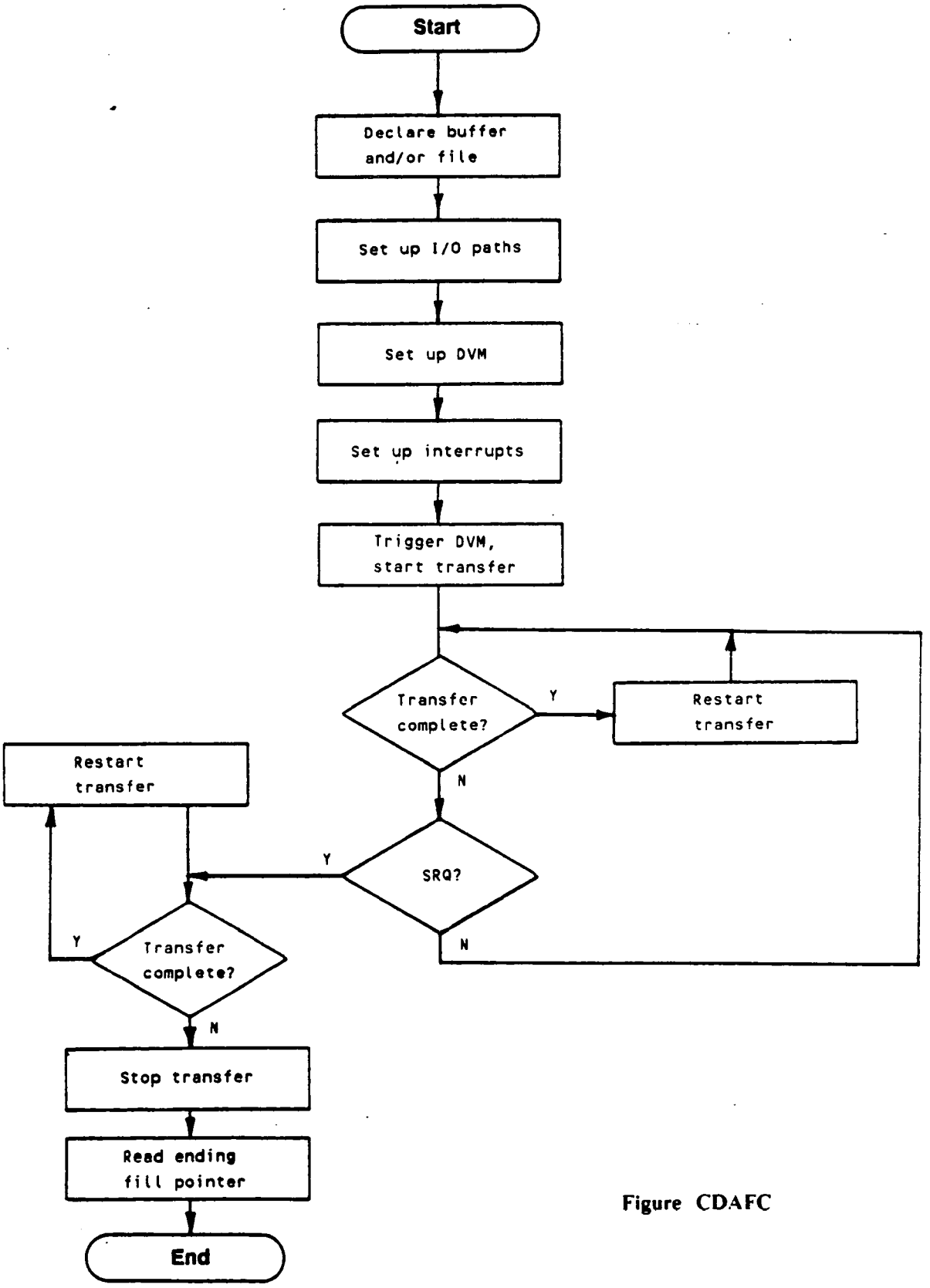


Figure CDAFC

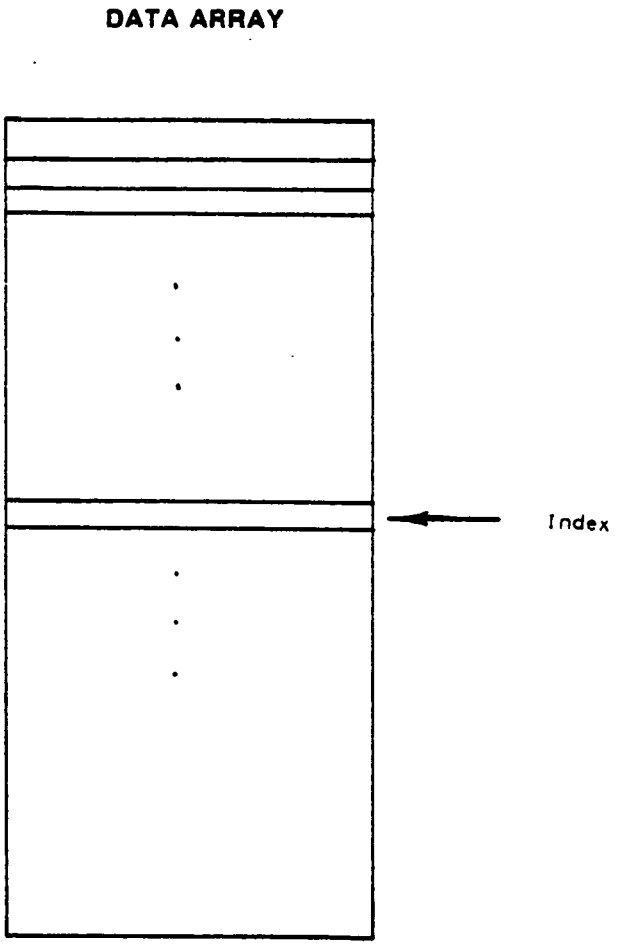


Figure CDAARR

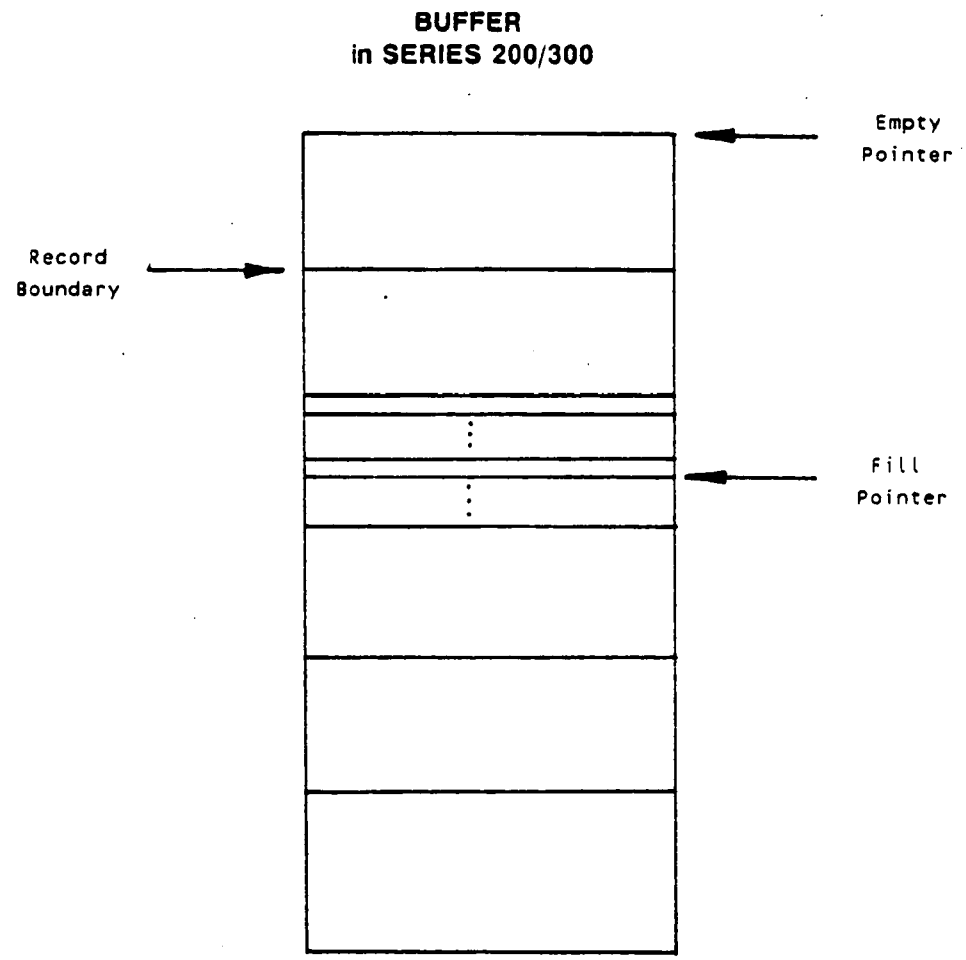


Figure CDABUF

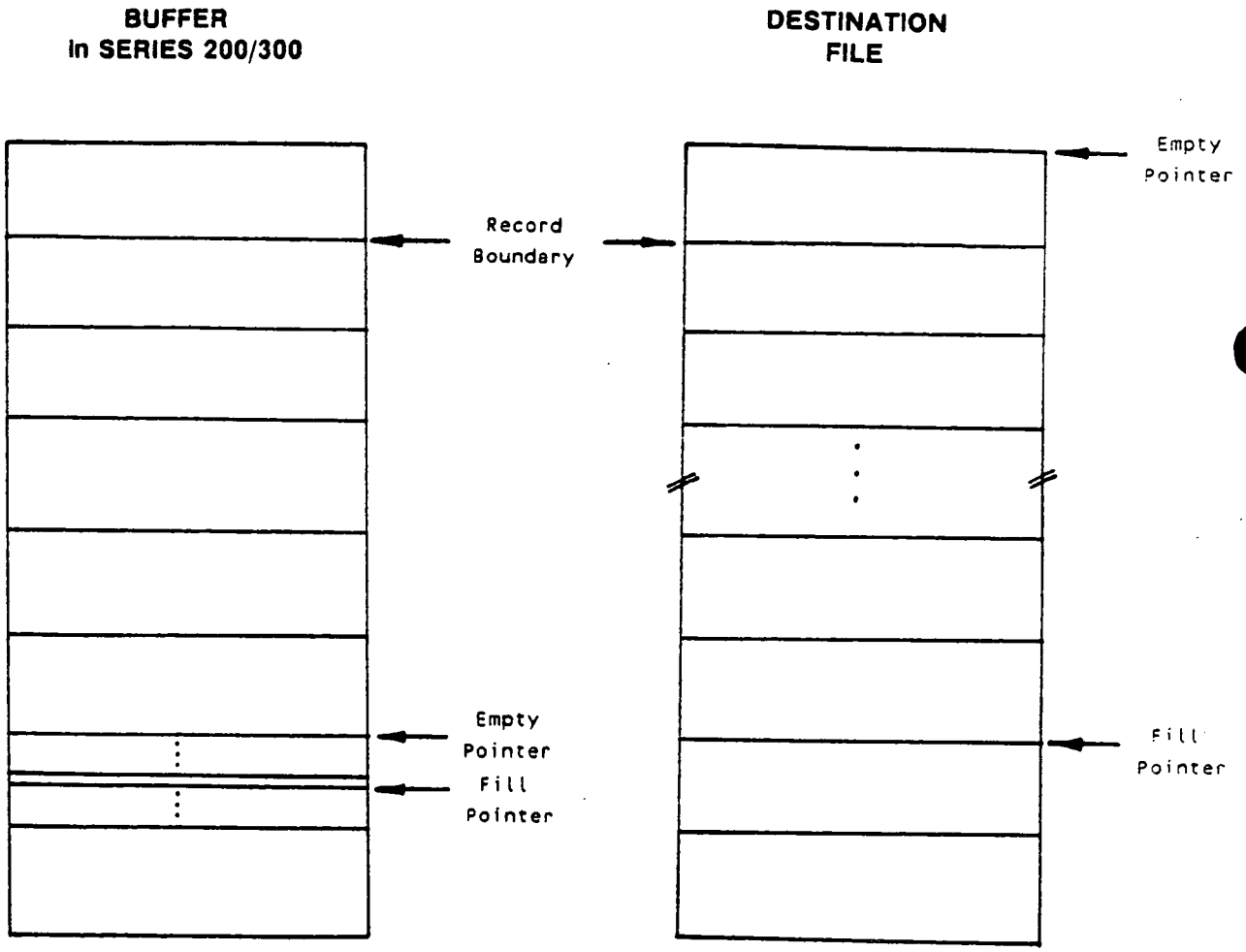


Figure CDAFL

```
10  ! This program demonstrates a method of doing continuous data acquisition
20  ! using the HP 3852A and a Series 300 BASIC computer. The computer must
30  ! have a GPIO card and approximately 1 Megabyte of memory. The HP 3852A
40  ! contains a HSDVM and is assumed to have no optional memory in the
50  ! HSDVM or in the mainframe. A 16 channel digital input card is also
60  ! necessary. For maximum throughput, a DMA card is necessary in the
70  ! Series 300 computer.
80  !
90  ! The program starts the TRANSFERs,then triggers the DVM to start
100 ! taking readings continuously. The TRANSFERs move the readings
110 ! into a buffer in the Series 300. No unwrap is performed, nor is the
120 ! DVM's data buffer cleared out. The DVM will stop when its data buffer
130 ! fills. The TRANSFERs will terminate on a digital interrupt over HPIB.
140 !
150 !
160   !
170   !
180 OPTION BASE 1
190 INTEGER Cntr,Done,In_stat_byt,Out_stat_byt,Mask,Gpio
200 INTEGER S300_buff(1:32767) BUFFER
210   !
220   !
230 Iopath_setup: ! This code sets up the file and IO paths.
240   !
250   ! Create a file of the length needed to accomodate the data of
260   ! interest. Highest performance can be obtained if the number of
270   ! bytes per record is an integral number of 256 byte blocks.
280   !
290   !
300   Gpio=12
310   ASSIGN @Hp3852 TO 709                ! HP 3852A IO address
320   ASSIGN @Gpio TO Gpio
330   ASSIGN @Path_3852 TO Gpio;WORD       ! Set up GPIO path using word attr
340   ASSIGN @S300_buff TO BUFFER S300_buff(*);FORMAT OFF  ! Series 300 buffer.
350   !
360 Dvm_setup:  ! This code sets up the HSDVM to take continuous readings.
370   ! DVM is in slot 6 with HS FET mux in slot 5. DVM is set up to take
380   ! DCV readings. One channel will be scanned repeatedly (500).
390   !
400   CLEAR @Hp3852
410   OUTPUT @Hp3852;"RST; OUTBUF ON; INBUF ON; DISP OFF"
420   !
430   !
440   OUTPUT @Hp3852;"USE 600; SCANMODE ON; SCTRIG HOLD"
450   OUTPUT @Hp3852;"FUNC DCV; RANGE 2.5; TERM RIBBON; RDGS GPIO"
460   OUTPUT @Hp3852;"NRDGS 1; SPER 10E-6; SCDELAY 16E-3; ASCAN ON"
470   OUTPUT @Hp3852;"CLWRITE SENSE 500; STTRIG HOLD"
480   !
490   !
500   !
510   CONTROL Gpio,2;5             ! Set control line low, DVM expects addr
520   OUTPUT Gpio USING "#,W";4    ! Set read reg addr to 4.
530   CONTROL Gpio,2;4             ! Set control line high, DVM sends data.
540   !
550 Intr_setup:  ! Set up interrupts in HP 3852A and Series 300.
560   !
570   !
580   ON EOT @Path_3852,5 GOSUB Xfer_done    ! When transfer stops, restart.
590   ON INTR 7 GOSUB Terminate              ! When 3852 interrupts, stop.
600   Mask=2                                 ! Set SRQ interrupt mask.
```

Figure CDA1

```
610     OUTPUT @Hp3852;"     VREAD A,PACK;STA? INTO X;SRQ;BEEP;SUBEND"
620     !
630     OUTPUT @Hp3852;"ON INTR USE 216 CALL XIT"
640     OUTPUT @Hp3852;"ON INTR USE 600 CALL INTROUT"
650     OUTPUT @Hp3852;"ENABLE INTR USE 600"  ! Enable interrupt on DVM.
660     OUTPUT @Hp3852;"RQS FPS;RQS ON"       ! Set up SRQ mask.
670     OUTPUT @Hp3852;"EDGE LH USE 216"      ! Set edge on digital input card.
680     OUTPUT @Hp3852;"ENABLE INTR USE 216"  ! Enable interrupt on digital card.
690     OUTPUT @Hp3852;"ENABLE INTR SYS"      ! Enable 3852 interrupts.
700     !
710     OUTPUT @Hp3852;"SCTRIG SGL"       ! Start transfers and trigger the DVM.
720 Start:TRANSFER @Path_3852 TO @S300_buff;COUNT 49152,EOR (END)  ! Start conti
nuous transfer.
730     !
740     !
750     !
760     Cntr=0
770     Done=0      ! The following code reads status while the transfers run.
780     REPEAT
790     STATUS @S300_buff,3;Fill_ptr
800     PRINT TABXY(10,5);"FILL POINTER IS";Fill_ptr
810     STATUS @S300_buff,10;In_stat_byt,Out_stat_byt
820     PRINT TABXY(10,8);"STATUS IN/OUT ";In_stat_byt;", ";Out_stat_byt
830     PRINT TABXY(10,11);"INDEX = ";Cntr
840     UNTIL Done
850     !
860     !
870     !
880     STOP
890     !
900     !
910     !
920     !
930 Xfer_done:RESET @S300_buff     ! Reset byte pointer to start.
940     TRANSFER @Path_3852 TO @S300_buff;COUNT 49152,EOR (END)  ! Restart transf
er.
950     Cntr=Cntr+1
960     RETURN
970     !
980     !
990 Terminate:WAIT .50              ! Take more data after intr.
1000    OFF EOT @Path_3852          ! Cancel inbound EOT.
1010    ABORTIO @Path_3852          ! Clean up HPIB path.
1020    ASSIGN @Path_3852 TO *
1030    STATUS @S300_buff,3;Fill_ptr
1040    PRINT TABXY(10,20);"ENDING FILL POINTER IS";Fill_ptr
1050    PRINT TABXY(10,18);"REACHED A GOOD END."
1060    STOP
1070    !
1080    !
1090    !
1100    END
```

```
10   ! This program demonstrates a method of doing continuous data acquisition
20   ! using the HP 3852A and a Series 300 BASIC computer. The computer must
30   ! have a GPIO card and approximately 1 Megabyte of memory. The HP 3852A
40   ! contains a HSDVM and is assumed to have no optional memory in the
50   ! HSDVM or in the mainframe. A 16 channel digital input card is also
60   ! necessary. For maximum throughput, a DMA card is necessary in the
70   ! Series 300 computer.
80   !
90   ! The program starts the TRANSFERs,then triggers the DVM to start
100  ! taking readings continuously. The TRANSFERs move the readings
110  ! into a buffer in the Series 300 and then into a disc or memory
120  ! file. No unwrap is performed, nor is the DVM's data buffer cleared
130  ! out. The DVM will stop when its data buffer fills. The TRANSFERs
140  ! will terminate on a digital interrupt over HPIB.
150  !
160     !
170     !
180  OPTION BASE 1
190  INTEGER Rec_length,Done,Rec,Byt,In_stat_byt,Out_stat_byt,End_rec_num,End_byt
e_num,Mask,Gpio
200     !
210     !
220  Iopath_setup: ! This code sets up the file and IO paths.
230     !
240     ! Create a file of the length needed to accomodate the data of
250     ! interest. Highest performance can be obtained if the number of
260     ! bytes per record is an integral number of 256 byte blocks.
270     !
280     INITIALIZE ":MEMORY,0,1",512          ! Create memory volume.
290     CREATE BDAT "HS_DATA:MEMORY,0,1",384,256   ! Total Bytes= 98304
300     !
310     Buff_size=65536
320     Gpio=12
330     ASSIGN @Hp3852 TO 709                 ! HP 3852A IO address
340     ASSIGN @Gpio TO Gpio
350     ASSIGN @Path_3852 TO Gpio;WORD        ! Set up GPIO path using word attr
360     ASSIGN @S300_buff TO BUFFER [Buff_size];FORMAT OFF  ! Series 300 buffer
370     ASSIGN @File TO "HS_DATA:MEMORY,0,1"      ! Destination file
380     !
390  Dvm_setup:  ! This code sets up the HSDVM to take continuous readings.
400     ! DVM is in slot 6 with HS FET mux in slot 5. DVM is set up to scan
410     ! taking DCV readings. One channel will be scanned repeatedly (500).
420     !
430     CLEAR @Hp3852
440     OUTPUT @Hp3852;"RST; OUTBUF ON; INBUF ON; DISP OFF"
450     !
460     !
470     OUTPUT @Hp3852;"USE 600; SCANMODE ON; SCTRIG HOLD"
480     OUTPUT @Hp3852;"FUNC DCV; RANGE 2.5; TERM RIBBON; RDGS GPIO"
490     OUTPUT @Hp3852;"NRDGS 1; SPER 10E-6; SCDELAY 10E-3; ASCAN ON"
500     OUTPUT @Hp3852;"CLWRITE SENSE 500; STTRIG HOLD"
510     !
520     !
530     CONTROL Gpio,2;5              ! Set control line low, DVM expects addr
540     OUTPUT Gpio USING "#,W";4     ! Set read reg addr to 4.
550     CONTROL Gpio,2;4              ! Set control line high, DVM sends data.
560     !
570  Intr_setup:  ! Set up interrupts in HP 3852A and Series 300.
580     !
590     !
```

Figure CDA2

```
600     ON EOT @File,5 GOSUB Xfer_done      ! When transfer stops, restart.
610     ON INTR 7 GOSUB Terminate           ! When 3852 interrupts, stop.
620     Mask=2                              ! Set SRQ interrupt mask.
630     ENABLE INTR 7;Mask
640     OUTPUT @Hp3852;"ENABLE INTR SYS"    ! Enable 3852 interrupts.
650     OUTPUT @Hp3852;"RQS ON; RQS INTR"   ! Send SRQ on interrupt at 3852.
660     OUTPUT @Hp3852;"EDGE LH USE 216"    ! Set edge on digital input card.
670     OUTPUT @Hp3852;"ENABLE INTR USE 216"! Enable interrupt on digital card.
680     !
690 Start: Xfer_length=98304               ! Xfer_length= 96 Rec_length
700     Rec_length=1024                     ! Rec_length= 512 readings
710     TRANSFER @Path_3852 TO @S300_buff;COUNT Xfer_length,EOR (COUNT Rec_length
),CONT
720     TRANSFER @S300_buff TO @File;COUNT Xfer_length,EOR (COUNT Rec_length),CON
T
730     OUTPUT @Hp3852;"SCTRIG S6L"         ! Start transfers and trigger the DVM.
740     !
750     !
760     Done=0      ! The following code reads status while the transfers run.
770     REPEAT
780     STATUS @S300_buff,3;Fill_ptr
790     PRINT TABXY(10,5);"FILL POINTER IS";Fill_ptr
800     STATUS @File,5;Rec,Byt
810     PRINT TABXY(10,8);"FILE RECORD AND BYTE= ";Rec;", ";Byt
820     STATUS @S300_buff,10;In_stat_byt,Out_stat_byt
830     PRINT TABXY(10,11);"STATUS IN/OUT ";In_stat_byt;", ";Out_stat_byt
840     UNTIL Done
850     !
860     !
870     STOP
880     !
890     !
900     !
910 Xfer_done:CONTROL @File,5;1,1     ! Reset record and byte pointer to start.
920     TRANSFER @Path_3852 TO @S300_buff;COUNT Xfer_length,EOR (COUNT Rec_length
),CONT
930     TRANSFER @S300_buff TO @File;COUNT Xfer_length,EOR (COUNT Rec_length),CON
T                               ! Restart the transfers.
940     RETURN
950     !
960     !
970 Terminate:WAIT .05              ! Take more data after intr.
980     OFF EOT @File               ! Cancel outbound EOT.
990     ABORTIO @Path_3852          ! Clean up GPIO path.
1000    ASSIGN @Path_3852 TO *
1010    ABORTIO @File               ! Clean up destination file path.
1020    STATUS @File,5;End_rec_num,End_byte_num
1030    ASSIGN @File TO *
1040    PRINT TABXY(10,20);"ENDING RECORD= ";End_rec_num;"  ENDING BYTE= ";End_by
te_num
1050    PRINT TABXY(10,18);"REACHED A GOOD END."
1060    STOP
1070    !
1080    !
1090    !
1100    END
```

```
10  ! This program demonstrates a method of doing continuous data acquisition
20  ! using the HP 3852A and a Series 300 BASIC computer. The computer must
30  ! have an HPIB card and approximately 1 Megabyte of memory. The HP 3852A
40  ! contains a HSDVM and is assumed to have no optional memory in the
50  ! HSDVM or in the mainframe. A 16 channel digital input card is also
60  ! necessary. For maximum throughput, a DMA card is necessary in the
70  ! Series 300 computer.
80  !
90  ! The program starts the TRANSFERs,then triggers the DVM to start
100 ! taking readings continuously. The TRANSFERs move the readings
110 ! into a buffer in the Series 300. No unwrap is performed, nor is the
120 ! DVM's data buffer cleared out. The DVM will stop when the number of
130 ! postscans is finished. The TRANSFERs will terminate on an SRQ interrupt
140 ! over HPIB.
150 !
160   !
170   !
180 OPTION BASE 0
190 INTEGER Cntr,Done,In_stat_byt,Out_stat_byt,Mask
200 INTEGER S300_buff(1:24576) BUFFER
210   !
220   !
230 Iopath_setup: ! This code sets up the file and IO paths.
240   !
250   ! Create a buffer of the length needed to accomodate the data of
260   ! interest. Highest performance can be obtained if the number of
270   ! bytes per record is an integral number of 256 byte blocks.
280   !
290   !
300   ASSIGN @Hp3852 TO 709                ! HP 3852A IO address
310   ASSIGN @Path_3852 TO 709;FORMAT OFF ! Set up HPIB path
320   ASSIGN @S300_buff TO BUFFER S300_buff(*);FORMAT OFF  ! Series 300 buffer.
330   !
340 Dvm_setup:  ! This code sets up the HSDVM to take continuous readings.
350   ! DVM is in slot 6 with HS FET mux in slot 5. DVM is set up to take
360   ! DCV readings. One channel will be scanned repeatedly (500).
370   !
380   CLEAR @Hp3852
390   OUTPUT @Hp3852;"RST;  OUTBUF ON; INBUF ON; DISP OFF"
400   OUTPUT @Hp3852;"PACKED A(8191);INTEGER X"
410   !
420   !
430   OUTPUT @Hp3852;"USE 600; SCANMODE ON; SCTRIG HOLD; FUNC DCV"
440   OUTPUT @Hp3852;"RANGE 2.5; TERM RIBBON; RDGSMODE BURST; SPER 27.5E-6"
450   OUTPUT @Hp3852;"SCDELAY 16E-3; POSTSCAN 4096; ASCAN ON"
460   OUTPUT @Hp3852;"CLWRITE SENSE 500; STTRIG HOLD"
470   !
480   !
490   !
500   !
510 Intr_setup:  ! Set up interrupts in HP 3852A and Series 300
520   !
530   !
540   ON EOT @Path_3852,5 GOSUB Xfer_done    ! When transfer stops, restart.
550   ON INTR 7 GOSUB Terminate            ! When 3852 interrupts, stop.
560   Mask=2                               ! Set SRQ interrupt mask.
570   ENABLE INTR 7;Mask
580   OUTPUT @Hp3852;"SUB INTROUT;XRDGS 600,4096 INTO A;VREAD A,PACK"
590   OUTPUT @Hp3852;"     ENABLE INTR USE 600;SUBEND"
600   OUTPUT @Hp3852;"SUB XIT;STTRIG SGL USE 600;XRDGS 600,4096 INTO A"
```

Figure CDA3

```
610     OUTPUT @Hp3852;"     VREAD A,PACK;STA? INTO X;SRQ;BEEP;SUBEND"
620     !
630     OUTPUT @Hp3852;"ON INTR USE 216 CALL XIT"
640     OUTPUT @Hp3852;"ON INTR USE 600 CALL INTROUT"
650     OUTPUT @Hp3852;"ENABLE INTR USE 600"  ! Enable interrupt on DVM.
660     OUTPUT @Hp3852;"RQS FPS;RQS ON"       ! Set up SRQ mask.
670     OUTPUT @Hp3852;"EDGE LH USE 216"      ! Set edge on digital input card.
680     OUTPUT @Hp3852;"ENABLE INTR USE 216"  ! Enable interrupt on digital card.
690     OUTPUT @Hp3852;"ENABLE INTR SYS"      ! Enable 3852 interrupts.
700     !
710     OUTPUT @Hp3852;"SCTRIG SGL"       ! Start transfers and trigger the DVM.
720 Start:TRANSFER @Path_3852 TO @S300_buff;COUNT 49152,EOR (END)  ! Start conti
nuous transfer.
730     !
740     !
750     !
760     Cntr=0
770     Done=0     ! The following code reads status while the transfers run.
780     REPEAT
790     STATUS @S300_buff,3;Fill_ptr
800     PRINT TABXY(10,5);"FILL POINTER IS";Fill_ptr
810     STATUS @S300_buff,10;In_stat_byt,Out_stat_byt
820     PRINT TABXY(10,8);"STATUS IN/OUT ";In_stat_byt;", ";Out_stat_byt
830     PRINT TABXY(10,11);"INDEX = ";Cntr
840     UNTIL Done
850     !
860     !
870     !
880     STOP
890     !
900     !
910     !
920     !
930 Xfer_done:ASSIGN @S300_buff     ! Reset byte pointer to start.
940     TRANSFER @Path_3852 TO @S300_buff;COUNT 49152,EOR(END)  ! Restart transf
er.
950     Cntr=Cntr+1
960     RETURN
970     !
980     !
990 Terminate:WAIT .50              ! Take more data after intr.
1000    OFF EOT @Path_3852          ! Cancel inbound EOT.
1010    ABORTIO @Path_3852          ! Clean up HPIB path.
1020    ASSIGN @Path_3852 TO *
1030    STATUS @S300_buff,3;Fill_ptr
1040    PRINT TABXY(10,20);"ENDING FILL POINTER IS";Fill_ptr
1050    PRINT TABXY(10,18);"REACHED A GOOD END."
1060    STOP
1070    !
1080    !
1090    !
1100    END
```

Binder Only Part No. 9282-1078

Made in U.S.A.