SOFTWARE TEST AND DRIVER PACKAGES FOR 620 INPUT/OUTPUT MODULES

Each Input/Output Module is delivered with a comprehensive software package of test and driver routines.

The test package exercises all the capabilities of a module, providing quick determination of its operational status. The test programs are randomly selectable under the operation of a test supervisor, allowing the operator to easily maintain interactive control.

The driver package provides convenient access to the modules without detailed knowledge of the hardware. With a simple CALL statement the module is fully functional and can be exercised without additional controlling instructions. The software drivers consist of two independent programs: Data Transfer Under Programmed Control and Data Transfer Using Direct Memory Access. They may be used by themselves or embedded in an operating system.

To illustrate a typical module's software package, the Analog Input Module (AIM) package is used in the following examples.

TEST PACKAGE EXAMPLE

The AIM Test Package consists of six different test programs selectable through a test supervisor:

Test No.	Description
1	Read one channel in random mode
2	Read N channels in sequential mode
3	Read N channels in random mode under timer control
4	Read one channel in random mode under BIC control
5	Read N channels in random mode under BIC control
6	Timer Test

These tests are selectable in any order and may be exercised as many times as desired with different parameters. Upon completion of a test, the following statistics are calculated and printed out:

- Minimum value in millivolts
- Average value in millivolts
- Maximum value in millivolts

The following frequency-of-occurrence readings are also calculated and printed out:

- Below average value, minus one count
- Average value, minus one count
- Average
- Average value, plus one count
- Above average value, plus one count

DRIVER PACKAGE EXAMPLE

The AIM Driver Package consists of the following programs: (1) Programmed Data Transfer, (2) Direct Memory Data Transfer, and (3) Status.

(1) Programmed Data Transfer Program

The activation of this program is accomplished by the following assembly language sequence:

```
CALL    PADC, MODE, CHANNELS, TIME, NUM,
        DESTINATION, EXIT
```

All entries in the calling sequence are either direct addresses or indirect addresses which point to the actual arguments. Multiple levels of indirect addresses are permitted. The underlined arguments represent inputs to the module. The interpretation of each argument follows:

MODE (integer)

The value of MODE specifies the technique for multiplexer channel selection.The options are as follows:

$$\underline{\text{MODE}} = \begin{cases} 0 & \text{Sequential channel selection, starting at 1 and ending at the value of CHANNELS for each frame.} \\ >0 & \text{Random channel selection; the value specifies the number of entries in the CHANNELS array.} \end{cases}$$

CHANNELS (integer vector)

The values in the CHANNELS array represent the channels and the order in which they will be collected for each frame. The entries in CHANNELS must be between 1 and 256.

TIME (integer)

The value of TIME represents the number of microseconds in one data acquisition frame. If TIME = 0, the start of each frame will be synchronized to an external signal.

NUM (integer)

The value of NUM specifies the total amount of data to be transferred from the AIM to the DESTINATION vector.

DESTINATION (integer vector)

The DESTINATION vector receives the incoming data. At least NUM words must be allocated to accommodate the data.

EXIT (label)

Control is transferred to EXIT when illegal input arguments are detected.

varian data machines • 611 hansen way • palo alto, california 94303 • (415) 326-4000

EXAMPLE: — The function and use of PADC program is shown in the following example:

Problem Statement — Acquire 300 data values from channels 10, 7, 8, 50 and 81. The order of channel selection for each frame is 50, 81, 7,8, and 10.

Assembly Language Solution

```
               . . .
               CALL     PADC, R, ADRS, T, N, (BADR)*, BADARG
               . . .
BADARG         CALL     ERPT          PRINT ERROR MESSAGE
               . . .
R              DATA     6             LENGTH OF 'ADRS'

*  CHANNEL SELECT VECTOR
ADRS           DATA     50
               DATA     81
               DATA     7
               DATA     8
               DATA     10
*  DATA             ARRAYS

T              DATA     500           500 MICROSECONDS FRAME TIME
N              DATA     300           300 INPUT VALUES
BADR           DATA     BUFFER        POINTER TO BUFFER
               . . .
BUFFER         BSS      300           RESERVE 300 WORDS
```

(2) Direct Memory Data Transfer Program

The activation of this program is accomplished by the following assembly language sequence.

```
CALL     DADC, BICNR, MODE, CHAN, TIME, NUM, DEST,
         EXIT
```

All entries in the calling sequence are either direct or indirect addresses of the actual arguments. Multiple levels of indirect addresses are permitted. The interpretation of each argument follows:

BICNR (integer)

The value of BICNR specifies which BIC will be used for the data transfer. The range of BICNR is from one to four corresponding to BIC's device addresses 20-21$_8$ to 26-27$_8$.

MODE (boolean)

The value of MODE determines whether sequential channels are to be scanned (MODE = 0) or data from an individual channel is to be acquired (MODE $\neq$ 0).

CHAN (integer)

The value of CHAN determines what channels are to be acquired. If MODE = 0, CHAN represents the last channel to be collected. If MODE $\neq$ 0, CHAN represents the channel to be acquired.

TIME (integer)

The value of TIME is the time in microseconds between each data input.

NUM (integer)

The value of NUM specifies the total amount of data to be transferred from the AIM to the DESTINATION vector.

DEST (integer vector)

The DEST vector receives the incoming data. At least NUM words must be allocated to accommodate the data.

EXIT (label)

Control is transferred to EXIT when illegal input arguments are detected.

(3) Status Program

This program determines the status of a previously started input operation. It is activated by the following assembly language sequence:

```
CALL     SADC, STATUS
```

The single entry in the calling sequence can be either direct or indirect addresses of the actual argument. Multiple levels and indirect addresses are permitted. The interpretation of the single output argument is:

STATUS (integer)

The value of STATUS specifies the status of the previously initiated input operations. The options are as follows:

$$\text{STATUS} = \begin{cases} 0 & \text{operation is not complete} \\ 1 & \text{operation is complete — no error conditions} \\ 2 & \text{operation aborted} \end{cases}$$

EXAMPLE: — The function and use of DADC and SADC programs are shown in the following example:

Problem Statement — Acquire 100 data values using the sequential scan mode starting from channel 1 through channel 10. The time between each data input should be 25 microseconds. The BIC with device address 20-21$_8$ will be used for the data transfer.

Assembly Language Solution

```
*  INITIATE DATA TRANSFER
          CALL     DADC, =1, =0, =10, =25, =100, VECT, ERR
          . . .
          . . .              concurrent processing
          . . .

*  WAIT FOR COMPLETION
WAIT      CALL     SADC, STATE    CHECK STATUS
          LDA      STATE          TEST STATUS WORD
          JAZ      WAIT           IF  O  WAIT
          . . .
          . . .
STATE     DATA     **             STATUS WORD
ERR       HLT
VECT      BSS      100            RESERVE 100 WORDS
```