

Applications Manual

*Data Acquisition Processor
Analog Accelerator Series*

Version 5.30

Microstar Laboratories, Inc.

This manual contains proprietary information which is protected by copyright. All rights are reserved. No part of this manual may be photocopied, reproduced, or translated to another language without prior written consent of Microstar Laboratories, Inc.

Copyright © 1985-2000

Microstar Laboratories, Inc.
2265 116 Avenue N.E.
Bellevue, WA 98004
Tel: (425) 453-2345
Fax: (425) 453-3199
[http:// www.mstarlabs.com](http://www.mstarlabs.com)

Microstar Laboratories, DAPcell, Data Acquisition Processor, DAP, DAPL, and DAPview are trademarks of Microstar Laboratories, Inc.

Microstar Laboratories requires express written approval from its President if any Microstar Laboratories products are to be used in or with systems, devices, or applications in which failure can be expected to endanger human life.

Microsoft, MS, and MS-DOS are registered trademarks of Microsoft Corporation. Windows is a trademark of Microsoft Corporation. IBM is a registered trademark of International Business Machines Corporation. Intel is a registered trademark of Intel Corporation. Novell and NetWare are registered trademarks of Novell, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

Contents

| | |
|---|------------|
| Contents | iii |
| 1. Introduction | 1 |
| Hardware Organization | 2 |
| Software Organization | 4 |
| 2. Applications Overview | 7 |
| Input..... | 7 |
| Basic Real-Time Processing..... | 8 |
| Output | 9 |
| Software Triggers..... | 10 |
| Further Real-Time Processing..... | 12 |
| Digital Signal Processing..... | 14 |
| Process Control..... | 15 |
| Communications | 16 |
| 3. Input | 17 |
| Application 1 — Sampling Three Inputs | 17 |
| Application 2 — Sampling 5000 Values | 21 |
| Application 3 — Digital Input | 22 |
| 4. Basic Real-Time Processing | 25 |
| Application 4 — Averaging | 25 |
| Application 5 — Peak Detection..... | 27 |
| Application 6 — Real-Time Data Analysis..... | 29 |
| Application 7 — DAPL Expressions | 31 |
| Application 8 — Extracting a Bit from a Digital Input..... | 32 |
| Application 9 — Finding Histograms | 34 |
| 5. Output | 37 |
| Application 10 — Asynchronous Output..... | 37 |
| Application 11 — Synchronous Output..... | 39 |
| Application 12 — Generating Waveforms | 40 |
| Application 13 — Generating Arbitrary Periodic Waveforms | 42 |
| Application 14 — Generating Periodic Waveforms by Copying | 43 |
| Application 15 — Generating Periodic Waveforms by Interpolation | 45 |
| Application 16 — Generating One-Shot Pulses..... | 47 |
| Application 17 — Using Analog Output Expansion..... | 49 |
| 6. Software Triggering | 51 |
| Application 18 — Software Triggers | 51 |
| Application 19 — Peak Detection..... | 54 |
| Application 20 — Implementing a Digital Oscilloscope | 56 |
| Application 21 — Using Hysteresis..... | 58 |
| Application 22 — Triggering on Two Conditions | 61 |
| Application 23 — Retriggering..... | 63 |
| Application 24 — Spike Detection | 65 |

| | |
|--|------------|
| Application 25 — Time Stamping Pulses | 68 |
| Application 26 — Detecting Bit Transitions | 71 |
| Application 27 — Using Triggers to Calculate Frequency | 73 |
| 7. Further Real-Time Processing..... | 75 |
| Application 28 — Finding Deviations between Inputs..... | 75 |
| Application 29 — Thermocouple Linearization | 77 |
| Improving Thermocouple Accuracy | 79 |
| Converting Temperatures to Fahrenheit | 80 |
| Sampling Several Thermocouples | 81 |
| Sampling Many Thermocouples | 82 |
| Sensing Reference Temperature | 83 |
| Application 30 — Interpolation..... | 87 |
| Application 31 — Autoranging | 89 |
| Application 32 — Identifying Maxima and Minima..... | 92 |
| Application 33 — Almost Simultaneous Sampling | 94 |
| Application 34 — Mixing Fast Inputs and Slow Inputs | 95 |
| Application 35 — Multiple Rate Data Transfer..... | 98 |
| Application 36 — Observing Timing of Rotating Machinery | 100 |
| 8. Digital Signal Processing..... | 105 |
| Application 37 — Digital Filtering..... | 105 |
| Application 38 — Spectral Analysis..... | 106 |
| Application 39 — Calculating Transfer Functions | 109 |
| 9. Process Control..... | 113 |
| Application 40 — Alarms | 113 |
| Application 41 — PID Control..... | 115 |
| Application 42 — Pulse Width Modulation | 117 |
| Heater Controller | 118 |
| 10. Communications | 121 |
| Application 43 — Text Communication..... | 121 |
| Application 44 — Text Communication from Several Tasks | 123 |
| Application 45 — Simultaneous Transfer of Text Data and Binary Data..... | 125 |
| Application 46 — Sending Data to a Data Acquisition Processor..... | 128 |
| Application 47 — Synchronizing Several Data Acquisition Processors..... | 130 |
| Application 48 — Serial Communication..... | 134 |
| Index..... | 137 |

1. Introduction

The Data Acquisition Processor from Microstar Laboratories is a complete data acquisition system that occupies one expansion slot in a personal computer. This Applications Manual introduces the Data Acquisition Processor by showing how to set up a wide variety of applications.

Two manuals complement the Applications Manual:

- The DAPL Manual describes the DAPL operating system, which runs in the Data Acquisition Processor.
- A Data Acquisition Processor hardware manual describes installation and use of Data Acquisition Processor hardware.

This version of the Applications Manual supports all versions of the DAPL operating system. Two applications require DAPL-specific commands: Application 37 and Application 40 each use a command supported only by DAPL 2000. See those applications for more information.

Note: The DAP 3400a is a unique board that requires a different syntax for some DAPL commands. Examples applications are provided in the DAP 3400a hardware documentation.

Chapter 2 outlines applications ranging from simple input and output to complex digital signal processing and real-time process control. These applications are described in detail in chapters 3 through 10.

To set up an application, the Data Acquisition Processor requires a series of commands. Typical commands set Data Acquisition Processor options, define variables, and define, start, and stop procedures.

Commands can be entered interactively from the keyboard using DAPview for Windows, or can be sent to the Data Acquisition Processor from a program running in the host PC. DAPview for Windows is an application development program from Microstar Laboratories; this program is described in the DAPview for Windows Manual.

A text file of Data Acquisition Processor commands is called a command file. A command file can be created using the integrated editor of DAPview Plus or using any text editor in the host PC. Definitions and formal syntax for DAPL commands are

found in the DAPL Manual. With DAPview Plus, the syntax of DAPL commands also is available online.

The sample command files in this manual can be entered using DAPview. The command files also are provided on the Data Acquisition Processor diskettes. To enter the commands, install the Data Acquisition Processor and the DAPview software following the instructions in the Data Acquisition Processor hardware manual.

The Data Acquisition Processor has a help facility and extensive error messages. DAPview and DAPview Plus both have help facilities.

Hardware Organization

Figure 1 shows the hardware organization of the Data Acquisition Processor. The Data Acquisition Processor is a complete microcomputer with the following features: It has its own microprocessor, random access memory (RAM) and read only memory (ROM), analog and digital inputs, analog and digital outputs, input/output control and timing circuits, timers, and direct memory access controllers (DMA) or first-in-first-out buffers (FIFO) for high-speed data transfer. Some Data Acquisition Processor models also have a digital signal processor with static RAM and some models also have a serial interface. The Data Acquisition Processor is connected to the PC through special FIFO interface hardware.

The timing control circuits control the analog and digital inputs and outputs so that the microprocessor is free to process the digital results. The timing control circuits select input pins and input pin gains. High-speed hardware implemented with a DMA controller or FIFO buffer transfers the digitized analog-to-digital converter output to the on-board memory. The timing control circuits also select analog and digital outputs. High-speed hardware, implemented with a DMA controller or FIFO buffer, transfers digital values from memory to the digital-to-analog converter or the digital output.

The Data Acquisition Processor provides analog and digital expansion signals for external multiplexing of analog and digital inputs. Using external multiplexers, the Data Acquisition Processor accepts up to 512 analog inputs and up to 128 digital inputs. The DAP 800 does not support external analog or digital expansion.

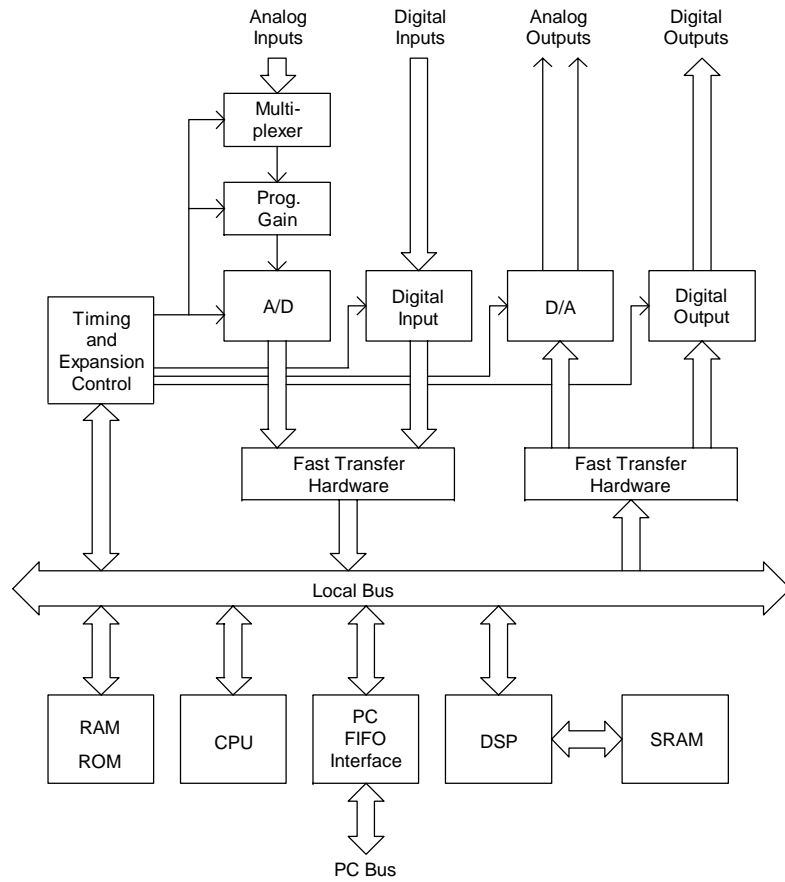


Figure 1. Hardware Organization

The analog voltage on one single-ended input pin or a differential pin pair is channeled from the analog input pins through a multiplexer and an instrumentation amplifier to a programmable gain amplifier. The gain is selected independently for each pin. A sample-and-hold circuit holds the voltage stable for the analog-to-digital converter, and the analog-to-digital converter converts the voltage to a binary number.

The Data Acquisition Processor provides two analog outputs through two digital-to-analog converters. Digital outputs are provided on a digital output port.

Software Organization

A procedure is a group of commands that together perform some function. DAPL allows the user to define input procedures, processing procedures, and output procedures for a Data Acquisition Processor. Within the processing procedure definitions are task definition commands. All the tasks in a procedure execute concurrently when the procedure is active. A command can be used several times to define distinct tasks within a procedure definition. A processing procedure, for example, might contain several commands setting up AVERAGE tasks that the Data Acquisition Processor executes concurrently on different input channel pipes.

An input procedure sets the sampling rate and selects the physical input pins on which voltages are sampled. An input procedure also may specify a sample count. Most applications have one input procedure, one processing procedure, and possibly one output procedure.

Every task is defined in a processing procedure. A task definition consists of a command and its parameters. Defining a task does not activate the task; a task only becomes active after the Data Acquisition Processor receives a START command to start the procedure containing the task.

Pipes are first-in-first-out software buffers which are used by DAPL to transfer data between DAPL tasks. Triggers are special pipes used to transfer synchronization information between DAPL tasks.

It is easy to create a DAPL application—simply:

- name all constants, variables, pipes and triggers
- define one or more procedures, and
- start one or more procedures.

Figure 2 shows the software organization of a typical application.

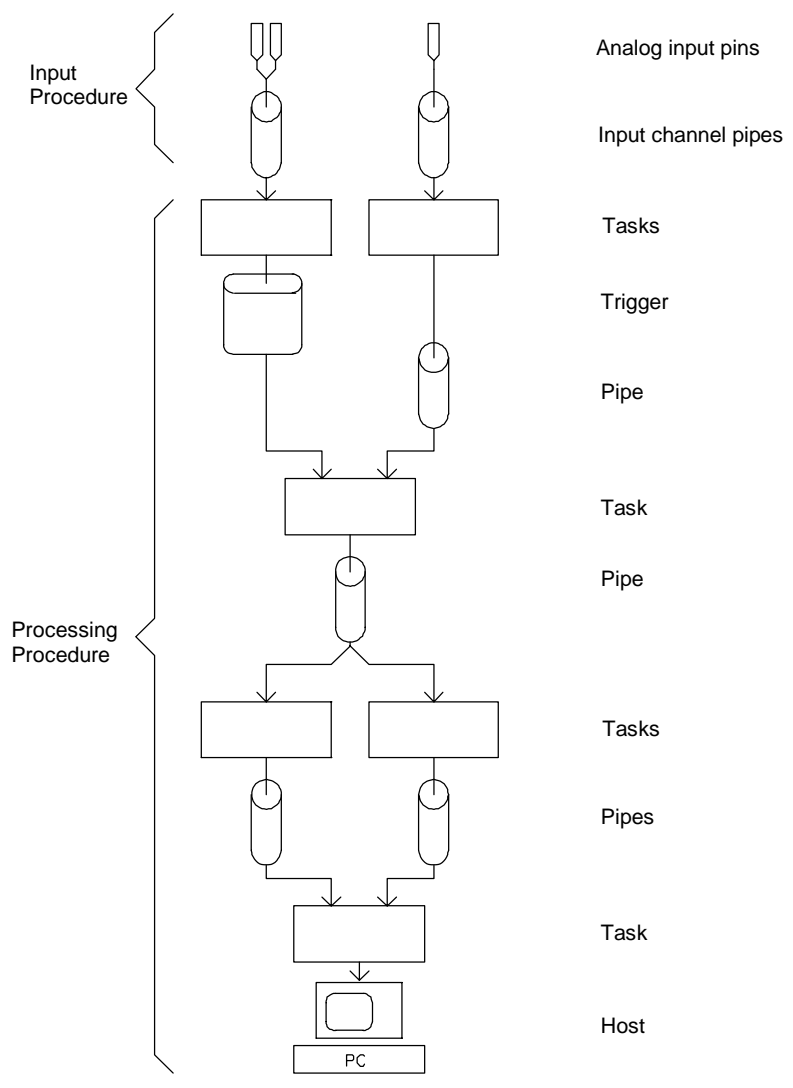


Figure 2. Software Organization

2. Applications Overview

This chapter lists a number of typical Data Acquisition Processor applications; the applications are described in detail in the following chapters of this manual. Each application introduces a DAPL command or a data acquisition configuration. Consult the DAPL Manual for detailed descriptions of the DAPL commands. The commands for each application are provided in DAPL command files in the APPS subdirectory of the Data Acquisition Processor software. Each of these DAPL command files can be run from DAPview.

A brief outline of the applications follows.

Input

Application 1 — Sampling Three Inputs

This application outlines a simple DAPL input configuration. Three input pins are digitized and the results are sent directly to the host PC.

Application 2 — Sampling 5000 Values

The COUNT input command limits the number of samples acquired by an input procedure. This ability is especially useful in conjunction with external hardware triggering, as illustrated in this application.

Application 3 — Digital Input

This application shows how to sample analog and digital inputs concurrently and send data to the host. Binary inputs are acquired on the digital input port.

Basic Real-Time Processing

Application 4 — Averaging

This application shows how to average data. Averaging reduces the rate at which the Data Acquisition Processor returns data to the host computer. Averaging may be used to slow the Data Acquisition Processor below its slowest data acquisition rate, or to improve accuracy by reducing the effects of noise.

Application 5 — Peak Detection

Some applications only require information about the maximum or minimum values of input data. This application demonstrates the use of the HI GH and RANGE commands to extract peak values.

Application 6 — Real-Time Data Analysis

This application demonstrates how commands can be chained to perform sophisticated data processing. In this example, derivatives, integrals, and square roots are computed.

Application 7 — DAPL Expressions

DAPL expressions define general arithmetic and logical operations on data. In this example, a DAPL expression converts raw data into engineering units.

Application 8 — Extracting a Bit from a Digital Input

In this example, an EXTRACT command extracts one bit from digital input port data.

Application 9 — Finding Histograms

This application demonstrates how to generate a histogram which summarizes how samples fall into user-defined ranges.

Output

Application 10 — Asynchronous Output

The Data Acquisition Processor provides two analog output pins and one digital output port. This application demonstrates how to control the analog and digital outputs.

Application 11 — Synchronous Output

DAPL provides commands which update the analog and digital outputs at precise time intervals. This application illustrates how to use an analog output to drive an external strip chart recorder.

Application 12 — Generating Waveforms

This application explains how the WAVEFORM command can be used to generate repetitive waveforms at the analog output pins.

Application 13 — Generating Arbitrary Periodic Waveforms

This example shows how to create arbitrary waveforms using output procedures.

Application 14 — Generating Periodic Waveforms by Copying

This example shows how to create an arbitrary waveform by copying a pipe to itself.

Application 15 — Generating Periodic Waveforms by Interpolation

This example shows how to create an arbitrary waveform by interpolation.

Application 16 — Generating One-Shot Pulses

This example shows how to generate one-shot pulses.

Application 17 — Using Analog Output Expansion

The Analog Output Expansion Board lets a Data Acquisition Processor drive more than two analog outputs. This example shows the DAPL commands required to use the Analog Output Expansion Board.

Software Triggers

Application 18 — Software Triggers

This application scans input data for a level trigger and prints data preceding and following each trigger event.

Application 19 — Peak Detection Using the PEAK command

The PEAK command asserts a trigger when it detects a local minimum or maximum in its input data. This application illustrates the use of PEAK .

Application 20 — Implementing a Digital Oscilloscope

This application uses a DAPL trigger to implement a four channel digital oscilloscope with pretriggering capability.

Application 21 — Using Hysteresis

This application demonstrates the use of trigger hysteresis to inhibit repeated triggering from a single analog event.

Application 22 — Triggering on Two Conditions

Logical combinations of triggers are required in many applications. In an automotive application, for example, two accelerometers oriented in the horizontal plane might be connected to one Data Acquisition Processor. Software triggering would be used to gather data whenever a large acceleration is observed along either axis. A TOR task is used in this application to implement a logical OR operation on triggers.

Application 23 — Retriggering

A WAIT task cannot respond to triggers which are separated by fewer samples than the number of samples transferred for each trigger event. This application uses a RETRIGGER task to guarantee that triggers which occur close together are not ignored.

Application 24 — Spike Detection

Triggers can be used to record high-speed spikes in input data without overloading the host PC. This application explains the use of wave extraction and wave averaging to perform data reduction and obtain characteristics of typical spikes.

Application 25 — Time Stamping Pulses

This application computes the sample number of each trigger event as well as the maximum value of the data around each trigger.

Application 26 — Detecting Bit Transitions

DAPL can assert a trigger when binary inputs change. This application explains how to select and trigger on various types of binary transitions.

Application 27 — Using Triggers to Calculate Frequency

Some applications only require summaries of the rates at which trigger assertions occur. This application demonstrates the use of the FREQUENCY command to convert trigger assertions into frequencies.

Further Real-Time Processing

Application 28 — Finding Deviations between Inputs

This application compares two inputs. Whenever the input channel pipes differ by more than a specified value, the sample number at which the difference occurs is printed.

Application 29 — Thermocouple Linearization

Thermocouples are sensors which generate voltages which vary with temperature. The conversion of raw thermocouple voltages to temperature readings requires sensor linearization. This application illustrates how thermocouple linearization is performed using the THERMO command.

Application 30 — Interpolation

Many sensors other than thermocouples require linearization. In this application an INTERP task implements an arbitrary mathematical function.

Application 31 — Autoranging

The software selectable gain capability of the Data Acquisition Processor can be used to implement input autoranging. This allows the Data Acquisition Processor to sample input signals possessing dynamic ranges much larger than the resolution of the analog-to-digital converter.

Application 32 — Identifying Maxima and Minima

In some applications, different tasks generate output data at different rates. These situations require multiple FORMAT commands. This application demonstrates how FORMAT prefixes can be used to identify the output data generated by DAPL.

Application 33 — Almost Simultaneous Sampling

This application explains how DAPL software can be used to provide "almost" simultaneous sampling of two or more analog input pins. A hardware solution using the Microstar Laboratories Simultaneous Sampling Board also is available.

Application 34 — Mixing Fast Inputs and Slow Inputs

Input procedures provide a "channel pipe list" facility which allows tasks to read data from more than one input channel pipe. This application explains why input channel pipe lists are useful when sampling a combination of fast and slow inputs.

Application 35 — Multiple Rate Data Transfer

In some applications it is necessary to transfer several data streams to the host PC at different data rates. This application shows how to use a MERGEF task to append identifiers while merging several data streams.

Application 36 — Observing Timing of Rotating Machinery

With a Counter/Timer Board, a Data Acquisition Processor can resolve variations in the speed of a rotating machine within one rotation. This application shows how to set up the Data Acquisition Processor and the Counter/Timer Board to study rotating machinery.

Digital Signal Processing

All Data Acquisition Processor models perform digital signal processing. Because Data Acquisition Processors with on-board digital signal processor (DSP) chips have optimized hardware for digital signal processing, they have significantly better performance than Data Acquisition Processors without DSP chips for the next three applications.

Application 37 — Digital Filtering

This application uses the `FIRFILTER` command to implement a bandpass filter.

Application 38 — Spectral Analysis

A spectrum analyzer is a device which determines the frequency components of an input signal. In this application, the `FFT` command is used in creating a DAPL spectrum analyzer.

Application 39 — Calculating Transfer Functions

The transfer function describes the frequency domain response of a device. This application shows two ways in which the Data Acquisition Processor can be used to calculate transfer functions.

Process Control

Application 40 — Alarms

In some applications it is necessary to monitor an analog voltage and switch a digital control if an out-of-range value is detected. This application shows how to use an ALARM task to respond quickly to an alarm condition.

Application 41 — PID Control

The Data Acquisition Processor can be configured as a process controller which requires no intervention from the host PC. This application implements a PID controller which reads an input voltage and controls an output signal so as to maintain the input at a specified value.

Application 42 — Pulse Width Modulation

Some applications require on/off, rather than continuous, control. This application uses pulse width modulation to feed the output of a PID controller to an on/off device.

Communications

Application 43 — Text Communication

For application in which text data needs to be sent to the PC, the Data Acquisition Processor can send text instead of binary data. This application shows how to use PRINT and FORMAT to send text to the PC.

Application 44 — Text Communication from Several Tasks

In many applications, the Data Acquisition Processor must return processed data from multiple tasks. This application shows two ways to send text data to the PC.

Application 45 — Simultaneous Transfer of Text Data and Binary Data

In some applications the Data Acquisition Processor must send two data streams, one at high speed with a large volume of data and one at low speed with a much smaller volume of data. This application combines binary communication for high volume data and text communication for low volume data.

Application 46 — Sending Data to a Data Acquisition Processor

Data can be transferred from the PC to a Data Acquisition Processor. This application illustrates how a Data Acquisition Processor reads binary data from a PC.

Application 47 — Synchronizing Several Data Acquisition Processors

Some applications require more real-time processing power than one Data Acquisition Processor can provide. Up to seven Data Acquisition Processors can be placed in one PC. This application shows how to use two synchronized Data Acquisition Processors.

Application 48 — Serial Communication

The DAP 801 has a serial port; this Data Acquisition Processor can communicate with external peripherals using serial communications. This application acquires analog data, sends the raw data to the PC, and simultaneously logs summary data to a serial printer.

3. Input

Application 1 — Sampling Three Inputs

This application configures the Data Acquisition Processor to sample three input signals and print the digitized values of these signals. The first signal is connected to single-ended input 2, the second signal is connected to single-ended input 5, and the final signal is connected to differential input 0. Each input pin is sampled every thirty milliseconds. The following DAPL commands configure the Data Acquisition Processor to perform this application.

The DAPL command file for this application is in the APPS subdirectory of the installed Data Acquisition Processor software. The file is named “APP01.DAP.”

```
RESET
I DEFINE A 3
  SET I PIPE0 S2
  SET I PIPE1 S5
  SET I PIPE2 D0
  TIME 10000
END
PDEFINE B
  BPRINT
END
```

Indentation is optional since DAPL ignores extra spaces. IPIPE can be abbreviated to IP.

The RESET command on the first line clears all definitions and errors. It is a good idea to start each application with a RESET. The next line begins an input procedure definition. An input procedure definition starts with the word IDEFINE and ends with the word END. IDEFINE usually is abbreviated to IDEF. The IDEFINE command requires the name of the input procedure and the number of input channel pipes read by the input procedure. A is the name chosen for the input procedure in this application. Input channel pipes are numbered consecutively from 0. Input procedure A has 3 input channel pipes, numbered 0, 1, and 2.

The three SET commands associate the analog input pins with the input channel pipes. Input channel pipe 0 is set to single-ended input 2 (S2). Input channel pipe 1 is set to single-ended input 5 (S5). Input channel pipe 2 is set to differential input 0 (D0).

The TIME command sets the sampling time to 10,000 microseconds. Since the input configuration samples three pins, each pin is sampled every 30,000 microseconds.

END marks the end of the input procedure definition.

The word PDEFINE begins a processing procedure definition. PDEFINE is usually abbreviated to PDEF. The PDEFINE command is followed by the name of the processing procedure. B is chosen for the name of the processing procedure in this application. You are free to choose other names for procedures in your applications.

The BPRINT command transfers binary data from all input channels to the binary communications pipe \$BINOUT. The Data Acquisition Processor transfers binary data in \$BINOUT directly to the PC. The BPRINT task continues until sampling is stopped.

END marks the end of the processing procedure definition.

DAPview Note: DAPview must be specially configured to read binary data from a Data Acquisition Processor. Use the Data Select menu to set the Data Type to Binary. Use the Binary Rec Size menu to set the number of binary words per record; in the first example above there are three input channel pipes that create three words per record. For most command lists, DAPview Plus makes the correct selections automatically if autoconfiguration is enabled.

DAPview Note: Data can be sent to the PC faster than DAPview can process the data. When this occurs, data are buffered in PC memory. When DAPview is processing buffered data, there is a time lag required to recognize DAPL commands, including STOP. This effect can be avoided by configuring applications to generate binary data at average rates which are slower than the maximum processing rate of DAPview.

DAPview Note: Using DAPview, DAPL commands can be entered either from the PC keyboard or from command files on disk in the host PC. Command files can be created using the integrated editor of DAPview, or using any other text editor in the host PC. The DAPview Run command sends a command file to the Data Acquisition Processor.

Figure 3 shows how this application passes data from the analog input pins to the host PC.

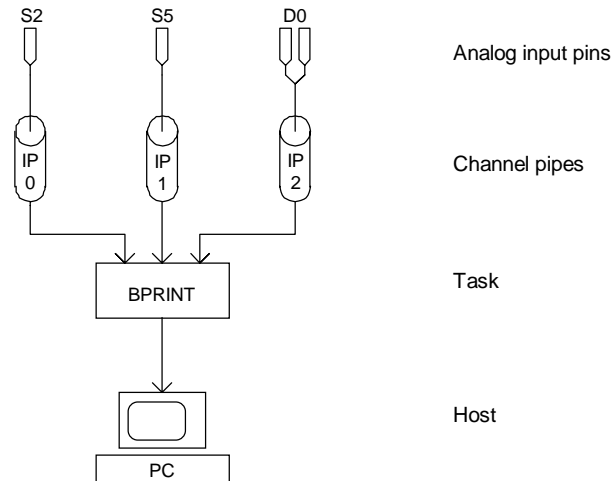


Figure 3. Sampling Three Inputs

The previous commands define how this application acquires and transfers data to the PC. Data collection begins when the Data Acquisition Processor receives a START command:

START A, B

The BPRINT task transfers data values from each input channel pipe in the order listed in the input procedure. The host program running on the PC must know the number of data channels sent from the Data Acquisition Processor in order to correctly display the data. Some programs such as DAPview Plus and DAPwindows automatically determine the number of data channels by examining the DAPL command file. Other programs such as DAPview must be user-configured.

Note that all the raw data are divisible by either 8 or 16, depending upon whether the Data Acquisition Processor is configured for unipolar or bipolar input. The binary format of the raw data from the Data Acquisition Processor is described in Chapter 2 of the DAPL Manual.

Because the sampling time has been set to 0.01 seconds per pin, and there are three pins, each pin is sampled every 0.03 seconds, or 33.3 times per second. The BPRINT task generates 100 values per second.

To stop sampling, the following command should be issued:

STOP

This command stops the input procedure and the processing procedure. Analog sampling is stopped and the BPRINT task is halted. The application can be restarted by reissuing the START command.

DAPview Note: Pressing function key <F9> displays a graph of the acquired data.

Application 2 — Sampling 5000 Values

An active input procedure normally continues sampling data until the Data Acquisition Processor receives a STOP command. An application may, however, require a specific number of samples. This capability is provided by the COUNT input command:

```
COUNT 5000
```

This command configures the input procedure to acquire 5000 samples and then stop acquisition. COUNT only stops acquisition; processing of the acquired data continues.

External triggering is another option that controls when sampling is performed. An external trigger is a digital input signal. When the external trigger signal is high, analog sampling begins. Sampling continues until the COUNT limit of the input procedure is reached, or until the Data Acquisition Processor receives a STOP command. The external trigger input is available on the Data Acquisition Processor analog expansion connector. See the Hardware Manual for additional details about the external trigger.

The following command list performs high-speed acquisition of 100 data values.

```
RESET
I DEFINE A 1
  SET IPIPEO 50
  TIME 20
  COUNT 100
  HTRIGGER ONESHOT
END
PDEFINE B
  BPRINT
END
START A, B
```

Single-ended input pin 0 is digitized at a rate of 50,000 samples per second. One hundred data values are acquired in two milliseconds. The HTRIGGER ONESHOT command selects external hardware triggering. To use an external trigger, the START command should be issued while the external trigger is low. Sampling begins when the external trigger makes a low to high transition.

Before restarting the procedures in an application, it is important to send a STOP command without parameters. A RESET command also will stop all processes.

Application 3 — Digital Input

Synchronous binary input allows the Data Acquisition Processor to monitor digital input lines with sampling times synchronized to the analog inputs. This has many applications, including precise triggering from conditions derived from the digital input lines.

One digital input port senses up to 16 digital lines simultaneously. Hardware discriminates between low voltages and high voltages. It also is possible to read digital signals using analog inputs, and then to discriminate between low and high signals in software. Reading digital signals through digital inputs, however, is more efficient than reading digital signals through analog inputs.

The following DAPL commands configure the Data Acquisition Processor to sample a digital input port and three analog inputs.

```
RESET
I DEFINE A 4
  SET I PIPE0 B0
  SET I PIPE1 S0
  SET I PIPE2 S1
  SET I PIPE3 S2
  TIME 10000
END
P DEFINE B
  BPRINT
  END
START A, B
```

The first SET command associates input channel pipe 0 with binary input port zero. Unless binary inputs are expanded externally, only one binary input port is available and the binary input port number is ignored. See the Hardware Manual for details on digital input expansion. With expansion, the Data Acquisition Processor supports eight 16-bit binary input ports. The DAP 800 has one 8-bit binary input port and does not support expansion. The digital input bits of the DAP 800 appear in the low-order byte, and the high order byte is zero.

The binary input port has 10K pull-up resistors on all inputs. Because of the pull-up resistors, unused inputs default to high TTL levels and are read by the Data Acquisition Processor as binary 1's.

Analog inputs pass through two pipeline stages, and digital data pass through one pipeline stage, before transmission to the Data Acquisition Processor microprocessor.

Therefore, binary data values on digital port B0 are sampled at the same time as the analog value is held on single-ended input S0. This feature permits sampling of a digital input at the same time as an analog input.

When input procedure A and processing procedure B are started, the BPRINT task transfers the data from the input channel pipes in order. The first value transferred represents the bits from the binary input port, interpreted as an integer.

4. Basic Real-Time Processing

Application 4 — Averaging

This application transfers the value of a single input channel pipe once every second. The Data Acquisition Processor maximum sampling interval is about 25 milliseconds, far too short for the desired data output rate. By using an AVERAGE task to average a number of data values, however, the volume of data can be reduced to one value every second. Such averaging not only reduces the data output rate but it also improves the accuracy of the output by reducing the effect of noise in the input signal. In applications where averaging is not desirable, the SKIP or IGNORE commands also reduce the output rate. The following commands perform input data averaging.

```
RESET
I DEFINE A 1
  SET I PIPE0 S6
  TIME 8333
  END
P DEFINE B
  AVERAGE (I PIPE0, 120, $BINOUT)
  END
START A, B
```

Figure 4 illustrates this application.

The I DEFINE command begins the definition of the input procedure. The sampling time is set to 8333 microseconds, corresponding to a sampling rate of 120 samples per second. This is a useful sampling rate for removing 60-Hz noise induced from power lines. By sampling at twice the power line frequency, an equal number of high and low noise values are sampled which cancel each other when averaged. (The sampling frequency in this application should be adjusted, depending on the local power supply frequency.) Power supply interference is a common source of noise in data acquisition applications.

The P DEFINE command begins the definition of a processing procedure named B. An AVERAGE command defines a task that reads 120 data values from input channel pipe 0, averages the values, and places the average into the binary communications pipe \$BINOUT. Since the sampling rate is 120 samples per second, one average is generated every second.

The command START A, B activates both procedures. The Data Acquisition Processor sends the one averaged data value to the PC per second.

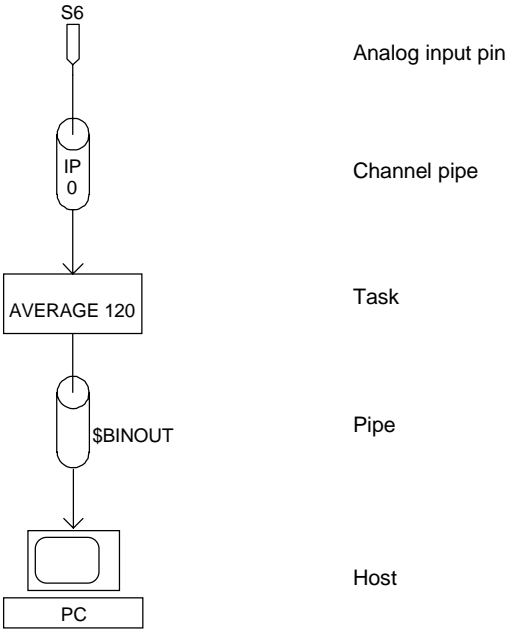


Figure 4. Averaging

Application 5 — Peak Detection

DAPL can be used to detect peaks in the input data. If peaks are well separated in time, the HIGH and LOW commands can be used to determine their heights and locations. For example, consecutive peaks of a normal canine heart electrocardiogram (EKG) are separated by minimum times of 0.3 seconds. This characteristic can be used to detect the height of each peak. The following command list illustrates how this is accomplished:

```
RESET
I DEFINE A 1
  SET I PIPE0 S0
  TIME 1000
  END
PDEFINE B
  HIGH (I PO, 300, $BI NOUT)
  END
START A, B
```

Figure 5 shows this application. The HIGH task receives data from input channel pipe 0 and scans blocks of 300 data values. With a sampling time of 1 millisecond, each block corresponds to 0.3 seconds of sampling time. Each number placed in \$BI NOUT corresponds to the maximum of a block. If a number is large, a peak occurred in this block. If the number is small, no peak occurred.

Note: The HIGH and LOW commands have optional fourth parameters which may be used to determine exactly where in each block of data a peak or valley occurs.

The above command list can be extended by adding a RANGE task to remove all data values which do not correspond to peaks:

```
RESET
PIPES PO
I DEFINE A 1
  SET I PIPE0 S0
  TIME 1000
  END
PDEFINE B
  HIGH (I PO, 300, PO)
  RANGE (PO, INSIDE, 5000, 30000, $BI NOUT)
  END
START A, B
```

The RANGE command reads data from pipe PO and passes data values to \$BINOUT if the values are between 5000 and 30000. In this manner, small data values generated by HIGH are never sent to the PC.

DAPL offers more sophisticated techniques for this sort of application, using software triggers and the PEAK command. See the [Detecting Peaks Using PEAK](#) application for an example.

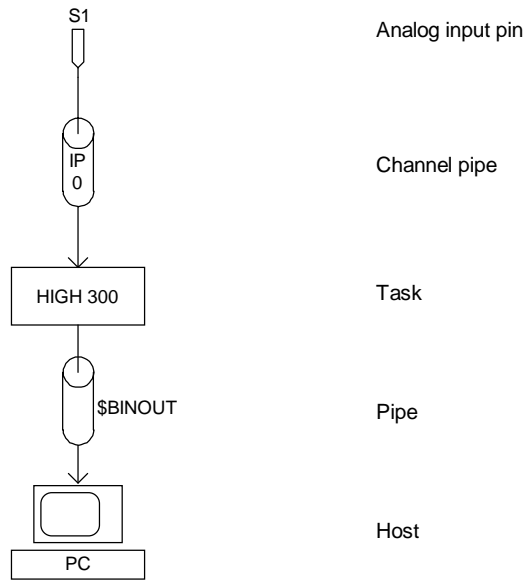


Figure 5. Peak Detection

Application 6 — Real-Time Data Analysis

DAPL permits linking the output of one task into the input of another task for sophisticated data processing. This application illustrates linking a number of commands. The Data Acquisition Processor is configured to sample an input pin, compute an average to smooth the data, and send the derivative and the square root of the integral of the averaged data to the host PC. The command file is:

```
RESET
PIPES P0, P1 LONG, P2, P3
I DEF A 1
  SET IPO S0
  TIME 10000
  END
PDEF B
  AVERAGE (IPO, 10, P0)
  INTEGRATE (P0, P1)
  Sqrt (P1, P2)
  DELTA (P0, P3)
  MERGE (P2, P3, $BINOUT)
  END
START A, B
```

A diagram of this application is shown in Figure 6.

The INTEGRATE task performs integration. Its output goes into long pipe P1. A square root operation is performed on the data by the Sqrt task. The DELTA task performs differentiation on the same data stream. Note that more than one command can read data from the same pipe.

Data are sent to the PC by the MERGE task. MERGE puts all data from the Sqrt and DELTA tasks into the binary communications pipe \$BINOUT.

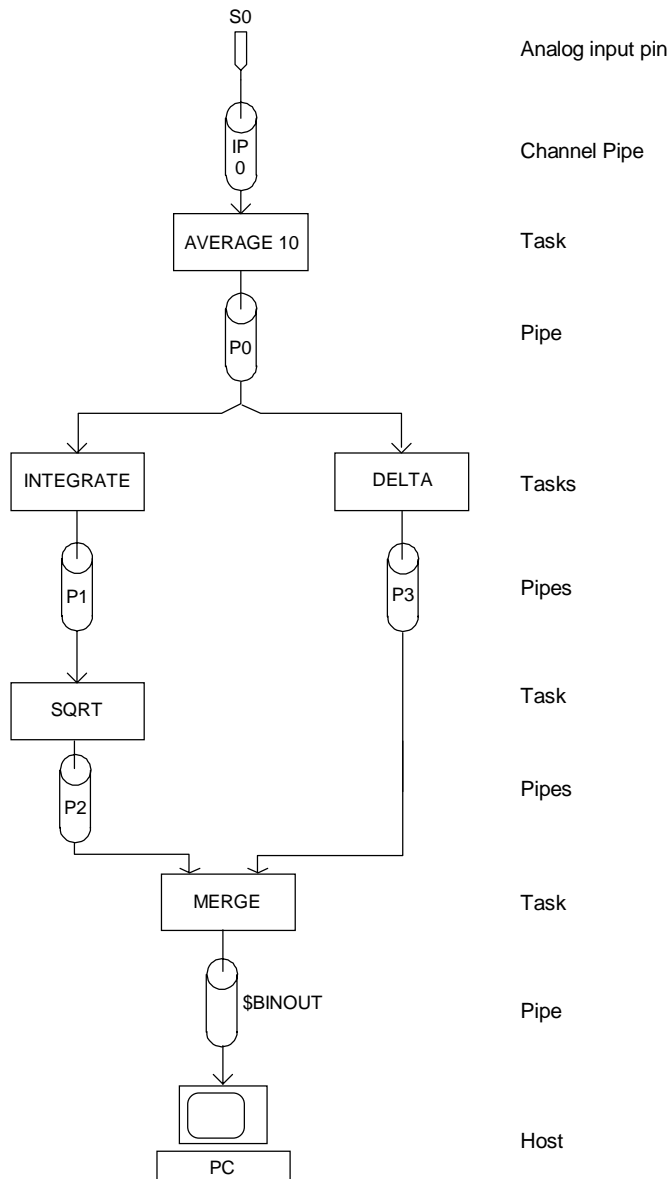


Figure 6. Real-Time Data Analysis

Application 7 — DAPL Expressions

DAPL expressions define arithmetic and logical operations on data. A DAPL expression defines a task which reads data, performs arithmetic or logical operations, and writes results to an output pipe. The complete syntax of DAPL expressions is given in Chapter 8 of the DAPL Manual.

The following example uses a DAPL expression to scale digitized input data into volts. The result of the DAPL expression is sent directly to the binary communications pipe \$BINOUT.

```
RESET
I DEF A 1
  SET IPO SO
  TIME 10000
  END
PDEF B
  $BINOUT = IPO * 5000 / 32767
  END
START A, B
```

If the Data Acquisition Processor analog input is configured for a -5 to +5 volt range, an input value of -32768 represents an input voltage of -5.000 volts and an input value of +32767 represents an input voltage of +4.998 volts. Therefore, multiplying input data by the factor 5000/32768 converts data into units of millivolts. Scale factors must be 16-bit numbers, so the ratio used in this example is adjusted to 5000/32767.

Application 8 — Extracting a Bit from a Digital Input

This application uses an EXTRACT command to extract the value of a single bit from the digital input port. The value of bit 3, for example, can be found with an EXTRACT command and placed in a pipe.

The format of the data from the digital input port is

```
xxxx xxxx xxxx bxxx
```

where b denotes the bit in position three, and x's denote bits which are to be ignored.

The command list for this application is:

```
RESET
I DEF A 1
  SET IPO B
  TIME 10000
  END
PDEF B
  EXTRACT (IPO, 3, $BINOUT)
  END
START A, B
```

A diagram of this application is shown in Figure 7.

This DAPL command list defines an input procedure which samples binary data with one input channel pipe. An EXTRACT command reads from input channel pipe 0 and calculates the value of bit 3. The result, either 0 or 1, is placed into \$BINOUT.

Note that DAPL expressions also can be used to manipulate and extract bits.

```
$BINOUT = (IPO >> 3) & 1
```

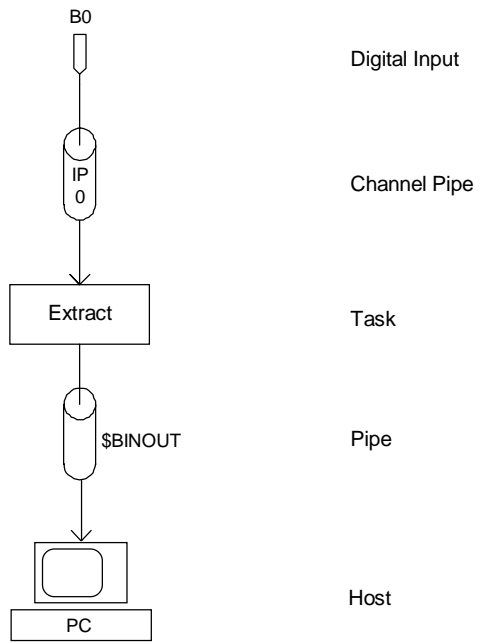


Figure 7. Extracting a Bit

Application 9 — Finding Histograms

This application uses the PCOUNT command to create a histogram which summarizes how many sample values fall into each of several user-defined ranges. A RANGE task transfers values in a specified range. A PCOUNT task reads data from a pipe, counts the data, and places the count in a variable.

```
RESET
PIPES P1, P2, P3, P4
VARIABLES V1, V2, V3, V4
IDEF A 1
  SET IPO SO
  TIME 10000
  COUNT 1000
  END
PDEF B
  RANGE (IPO, INSIDE, 1, 1000, P1)
  RANGE (IPO, INSIDE, 1001, 2000, P2)
  RANGE (IPO, INSIDE, 2001, 3000, P3)
  RANGE (IPO, INSIDE, 3001, 4000, P4)
  PCOUNT (P1, V1)
  PCOUNT (P2, V2)
  PCOUNT (P3, V3)
  PCOUNT (P4, V4)
  END
START A, B
```

Figure 8 shows a diagram of this application.

The input procedure configures the Data Acquisition Processor to sample an analog input one thousand times at 100 Hz.

The RANGE tasks transfer data values within specified ranges to pipes P1, P2, P3, and P4. Notice that four tasks read data from the same input channel pipe. DAPL allows any number of tasks to share input channel pipe and pipe data. The PCOUNT tasks count the numbers of values in each of the four pipes. These counts are placed in variables V1 through V4.

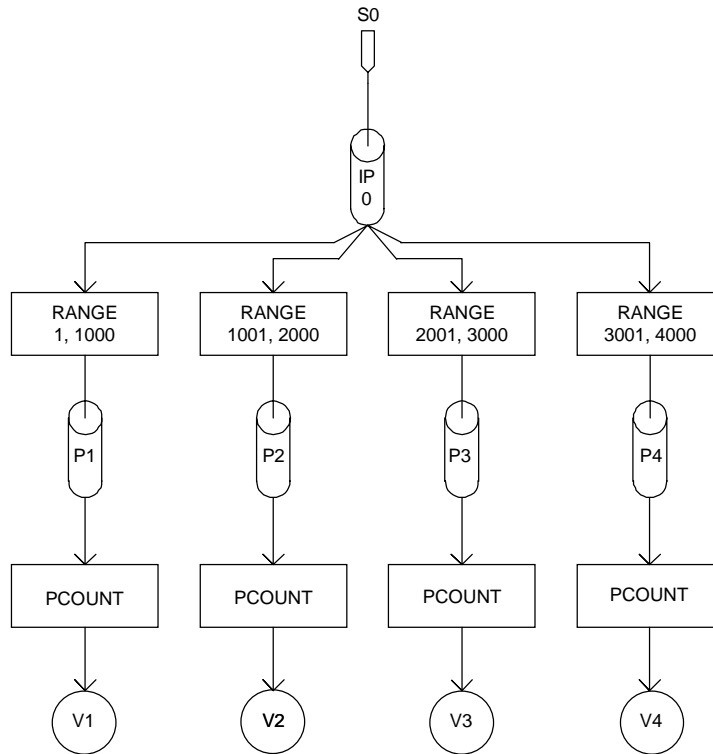


Figure 8. Finding Histograms

To run this application, start the input and processing procedures. After acquiring and processing all of the input data, issue the following command:

```
SDI SPLAY V1, V2, V3, V4
```

The SDI SPLAY (Symbol DI SPLAY) command prints information about DAPL symbols to the Data Acquisition Processor text communications pipe \$SYSOUT. In this case, SDI SPLAY prints a table in the following form:

```
34
235
663
68
```

This table presents the numbers of sample values in each of the four ranges. Since one thousand samples were taken, these numbers divided by ten represent percentages.

5. Output

Application 10 — Asynchronous Output

In addition to its analog and digital inputs, the Data Acquisition Processor provides two analog outputs and one digital output port. The DACOUT and DIGITALOUT commands give access to these outputs. Both commands read data from variables or pipes and transfer the data to the outputs. The following command list demonstrates analog and digital output task definitions:

```
RESET
PIPE P1
VARIABLE V1
PDEF B
  DACOUT (P1, 1)
  DIGITALOUT (V1, 0)
END
START B
```

Figure 9 is a diagram of this application.

When processing procedure B is started, DACOUT reads data from pipe P1 and writes the data to analog output #1. If the analog output is configured for the -5 to +5 volt range, a pipe value of -32768 generates an output voltage of -5.000 volts and a pipe value of +32767 generates an output voltage of +4.998 volts. While procedure B is active, the DIGITALOUT task continuously reads the value of variable V1 and sends the variable's current value to the digital output lines. When the value of V1 changes, the state of the digital output lines changes. The value of the variable V1 can be changed from DAPL using the LET command:

```
LET V1=256
```

Data for the pipe P1 normally would be generated by a task running in the Data Acquisition Processor. The host PC can put data into a pipe with a FILL command:

```
FILL P1 1024
```

The analog output voltage stays the same until the DACOUT task receives a new value from pipe P1.

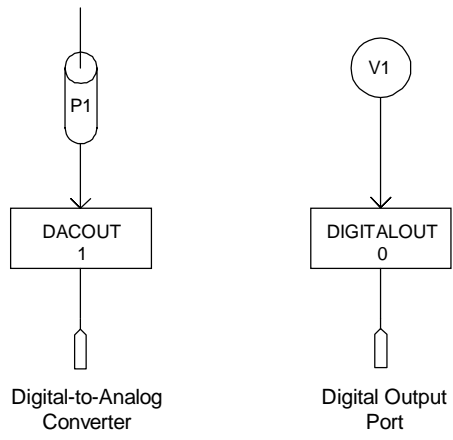


Figure 9. Digital and Analog Output

Application 11 — Synchronous Output

Because DAPL is a multitasking operating system, the rate of data flow within pipes is variable. DAPL is responsible for synchronizing tasks, so rate variations are not seen during typical data processing operations. DACOUT and DIGITALOUT are adversely affected by rate variations, however, since such variations result in timing jitter between successive updates to the output ports. At update rates faster than several Hertz, this jitter may be significant.

To eliminate timing jitter, Data Acquisition Processors have output procedures that update the output ports at precisely timed intervals. The following command list illustrates the use of RAVERAGE and an output procedure to create a smoothed signal to drive a strip chart recorder. By driving the strip chart recorder with smoothed data from the analog output pin, this application eliminates high frequency noise from the strip chart records.

```
RESET
I DEF A 1
  SET I PIPE0 S0
  TIME 8000
END
PDEF B
  RAVERAGE (I PIPE0, 100, O PIPE0)
END
ODEF C 1
  SET O PIPE0 A0
  TIME 8000
END
START A, B, C
```

In this application, an input pin is sampled at 125 Hz. A running average on windows of 100 data values smoothes the data. RAVERAGE reads the data from input channel pipe 0. The smoothed data are sent to output channel pipe 0. The output procedure C has a SET command to define O PIPE 0 for output on analog output pin 0. The TIME command specifies that the output pin is to be updated every 8000 microseconds—125 updates per second.

I PIPE and O PIPE usually are abbreviated to IP and OP respectively.

Application 12 — Generating Waveforms

The Data Acquisition Processor analog output need not be derived from an input channel pipe. The WAVEFORM command provides a means of generating common types of periodic output data. The following commands configure a Data Acquisition Processor to generate a 5 Hz sine wave output:

```
RESET
PDEF B
    WAVEFORM (2, 32000, 50, OPIPEO)
    END
ODEF C 1
    SET OPIPEO AO
    TIME 4000
    CYCLE 50
    END
START B, C
```

The first parameter of the WAVEFORM task specifies a sine wave output. Alternative outputs are triangle, sawtooth, and square wave. The second parameter selects an amplitude of 32000. This means that the values are sampled from an ideal sine wave of amplitude 32000. Because of sampling effects, the output values range from approximately -32000 to approximately +32000. The third parameter sets the period of the sine wave to 50 samples. The result is a 5 Hz sine wave because the output procedure reads 250 data values per second.

Note the CYCLE command in the definition of output procedure C. This command specifies that the output data should repeat after 50 values. To improve efficiency, the Data Acquisition Processor reads only the first 50 values from P1 and then cycles through the data repeatedly. With a CYCLE command, the Data Acquisition Processor can update its outputs up to the maximum output rate. See the Hardware Manual for details on the maximum output rate of the Data Acquisition Processor.

The WAVEFORM command includes powerful modulation capabilities. The basic signal generated by WAVEFORM can be modified by amplitude modulation or by frequency modulation. Modulation is controlled by specifying an additional pipe parameter which provides modulation data. Each time WAVEFORM generates a data value, it reads a modulation value from the modulation pipe. The modulation value must be a positive number and is interpreted as a signed binary fraction. The current amplitude or frequency is multiplied by this modulation value before WAVEFORM generates its next output value.

The following commands generate a sine wave which sweeps a frequency range of 6.25 to 25 Hertz.

```
RESET
PIPE P1, P2
PDEF B
  WAVEFORM (1, 12288, 100, P1)
  P2 = P1 + 20479
  WAVEFORM (2, 32000, 10, OPIPEO, 2, P2)
END
ODEF C 1
  SET OPIPEO AO
  TIME 4000
END
START B, C
```

Note that the CYCLE 50 command does not appear in this command list. Because of the modulation, the data do not repeat after 50 updates.

The output of the first WAVEFORM can be processed by the INTERP task to generate an arbitrary modulation function. For example, a sine wave's frequency may be exponentially swept, and, at the same time, its amplitude may be linearly modulated.

Application 13 — Generating Arbitrary Periodic Waveforms

Waveform data can be placed into a pipe with a FILL command. This application uses a FILL command to fill P1 with exponential waveform data. A CYCLE command allows analog output to start after 10 values have been copied into the output channel pipe. CYCLE also causes the output to repeat continuously the same 10 values.

```
RESET
PIPE P1
PDEF A
  COPY (P1, OPIPEO)
  END
ODEF B 1
  SET OPIPEO AO
  TIME 10000
  CYCLE 10
  END
START A, B
FILL P1 3 7 20 54 148 403 1096 2980 8103 22026
```

Application 14 — Generating Periodic Waveforms by Copying

Data in a pipe can be recycled by copying the pipe back to itself. This application generates a periodic waveform by maintaining a copy of the waveform data in a pipe. The following command list illustrates how to copy a pipe back to itself and use the data for periodic waveform generation.

```
RESET
PIPES P1
FILL P1 3 7 20 54 148 403 1096 2980 8103 22026
PDEF A
    COPY (P1, P1, OPIPE0)
    END
ODEF B 1
    SET OPIPE0 A0
    TIME 10000
    END
START A, B
```

Figure 10 is a diagram of this application.

A FILL command places an initial copy of exponential data into pipe P1. A COPY task copies P1 back to itself and to output channel pipe 0. This keeps data continuously stored in P1 and provides data for analog output.

Note that in the previous application the FILL command came after the START command. In this application, the FILL command must precede the START command. This avoids possible data mixing by ensuring that all data values are in the pipe before COPY begins recycling the values.

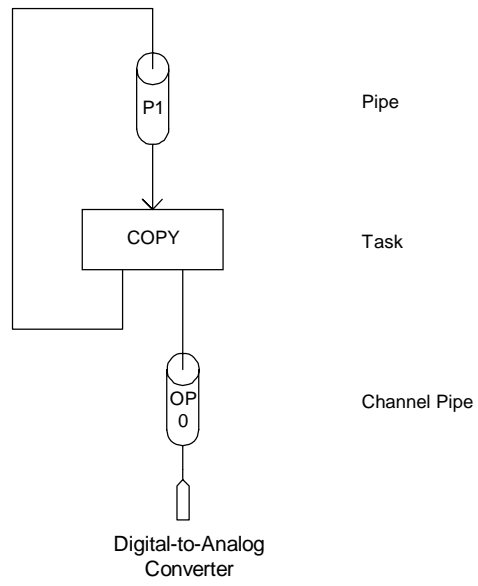


Figure 10. Using COPY to Save Periodic Waveforms

Application 15 — Generating Periodic Waveforms by Interpolation

When generating a periodic waveform, an INTERP task can interpolate between data points to provide fine resolution. This application uses INTERP to create a 50 point exponential wave from a vector with 11 data points.

```
RESET
VECTOR VECX=(-500, -400, -300, -200,
             -100, 0, 100, 200, 300, 400, 500)
VECTOR VECY=(0, 3, 7, 20, 54, 148, 403, 1096, 2980, 8103, 22026)
PIPES P1
PDEF A
  SAWTOOTH (500, 50, P1)
  INTERP (P1, VECX, VECY, OPIPEO)
END
ODEF B 1
  SET OPIPEO AO
  TIME 1000
  CYCLE 50
  END
START A, B
```

Figure 11 is a diagram of this application.

The DAPL command list defines two vectors for use by INTERP. VECX holds evenly incremented values, and VECY holds values which are an exponential function of VECX. A ramp is generated by a SAWTOOTH task which provides values that INTERP interpolates to provide exponential waveform data. The results are sent to the analog output.

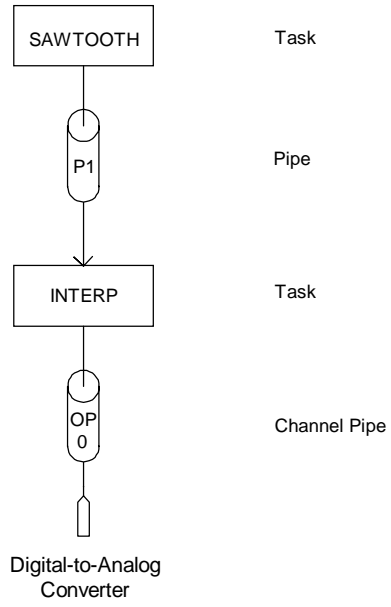


Figure 11. Using I NTERP to Create Waveforms

Application 16 — Generating One-Shot Pulses

The default configuration of a Data Acquisition Processor output procedure is oriented toward generating continuous analog output signals. In some applications, however, single pulses are desired, with analog updating stopping at the end of each pulse. This application demonstrates how to generate a 10 millisecond pulse.

When sending a single burst of analog output data, a Data Acquisition Processor output procedure should be non-cyclical (i.e. the output procedure should not include a CYCLE command). Instead, an UPDATE BURST command should be used to set the output procedure to burst mode. In burst mode, the output procedure stops updating after sending a burst of data. The output procedure stops as a result of emptying its output channel pipes. No underflow warning message is generated when the UPDATE BURST command is used. In burst mode, output updating automatically restarts when more data are available.

The following DAPL listing configures a Data Acquisition Processor to output a 10 millisecond pulse.

```
RESET
PIPE P1
FILL P1 6000 6000 6000 6000 6000
FILL P1 6000 6000 6000 6000 6000
FILL P1 0
PDEF A
    COPY (P1, OPI PEO)
END
ODEF B 1
    OUTPUTWAIT 11
    UPDATE BURST
    SET OPI PEO AO
    TIME 1000
END
START A, B
```

This listing uses a series of FILL commands to place 11 data values in pipe P1. The values in P1 are copied to output channel pipe 0.

An OUTPUTWAIT command in the output procedure sets the output to begin updating when 11 values are received in the output channel pipe. Output sampling time is set to 1000 microseconds to provide a 10 millisecond output pulse. The eleventh value in this example is a zero value to bring the output back to zero after the pulse.

When an output procedure is used to generate a single pulse, it usually is necessary to adjust the value of `OUTPUTWAIT`. The value of `OUTPUTWAIT` defaults output channel pipe buffering to approximately 500 milliseconds of data. The default value of `OUTPUTWAIT` can be calculated by dividing one-million microseconds by the output `TIME` parameter. This value is chosen to prevent underflow conditions in applications that perform continuous analog output. If an output procedure needs to generate a single burst of data with a duration of less than 1000 milliseconds, the default value of `OUTPUTWAIT` must be changed. Two situations are possible:

- If the length of the output pulse is less than the default value of `OUTPUTWAIT`, an `OUTPUTWAIT` command should be included in the output procedure. The parameter of `OUTPUTWAIT` should be equal to the length of the output pulse.
- If the length of the output pulse is greater than the default value of the `OUTPUTWAIT` parameter, the `OUTPUTWAIT` command is not needed.

Application 17 — Using Analog Output Expansion

This application uses an Analog Output Expansion Board to provide four synchronous analog signals. The Analog Output Expansion Board is connected to the digital output port, so it requires output data in a special output expansion format. Format conversion is provided by a DEXPAND task.

```
RESET
PIPES P1, P2, P3, P4, P5
PDEF A
  SINEWAVE (30000, 100, P1)
  SQUAREWAVE (30000, 100, P2)
  TRIANGLE (30000, 100, P3)
  SAWTOOTH (30000, 100, P4)
  MERGE (P1, P2, P3, P4, P5)
  DEXPAND (P5, (0, 1, 2, 3), OPIPEO)
END
ODEF B 1
  SET OPIPEO B
  TIME 250
  CYCLE 1600
END
START A, B
```

Figure 12 is a diagram of this application.

This DAPL command list defines four tasks to create four waveforms. The waveforms are merged into pipe P5 by the MERGE task. The DEXPAND task then converts the output data in P5 to the output expansion format of the Analog Output Expansion Board and places the result in output channel pipe 0; the list (0, 1, 2, 3) specifies that the analog outputs are at addresses 0, 1, 2, 3. An output procedure sends the data in output channel pipe 0 to the Analog Output Expansion Board which is connected to the digital output port.

DEXPAND produces four output values for every input value, so the CYCLE length in the output procedure must be four times the number of source data values.

Synchronous output to a Digital Expansion Board works in exactly the same way as synchronous output to an Analog Output Expansion Board. The Analog Output Expansion Board could be replaced by a Digital Expansion board in this application.

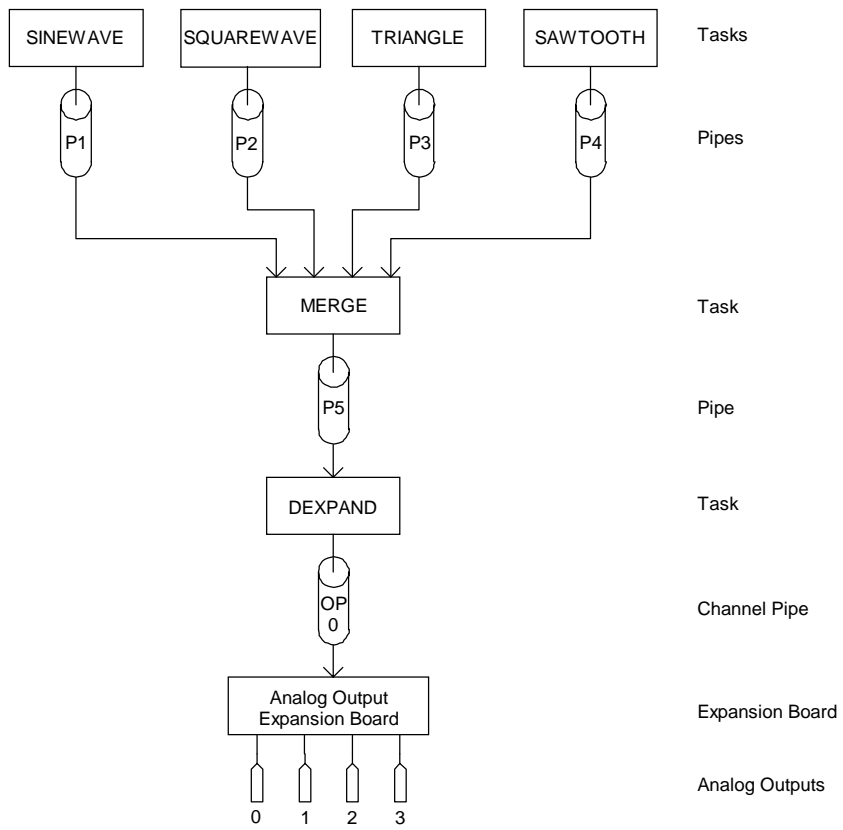


Figure 12. Using the Analog Output Expansion Board

6. Software Triggering

Application 18 — Software Triggers

In this application, the Data Acquisition Processor samples the voltage from a blood pressure sensor on a laboratory animal. The Data Acquisition Processor uses software triggering to transfer input channel pipe data only during the active part of each wave. This reduces the data to be recorded without losing useful information.

The input pin is sampled at 1,000 samples per second. A trigger event is recognized each time the input exceeds a trigger level of 1.5 volts. The Data Acquisition Processor takes a specified number of samples — 10 samples before the trigger and 40 samples after the trigger — and sends these to the host for display and storage. The command file for this application is:

```
RESET
TRIGGER T
I DEFINE A 1
  SET I PIPEO D3
  TIME 1000
  END
PDEFINE B
  LIMIT (I PO, OUTSIDE, -32768, 9830, T, OUTSIDE, -32768, 9830)
  WAIT (I PO, T, 10, 40, $BINOUT)
  END
START A, B
```

Figure 13 is a diagram of this application.

The second line of this command list defines trigger T. This trigger is used for synchronizing trigger events to actual trigger data.

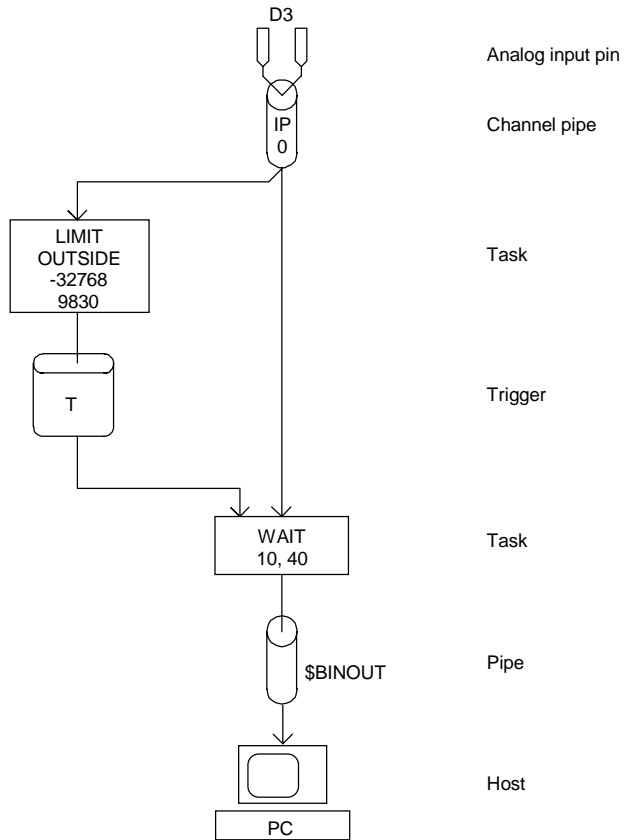


Figure 13. Trigger Scanning

The processing procedure for this application includes a LIMIT task definition and a WAIT task definition. LIMIT is a DAPL command that asserts a trigger when it detects a value inside a specified range. If the input voltage range of the Data Acquisition Processor is -5 to +5 volts, the analog-to-digital converter count corresponding to 0 volts is 0 and the analog-to-digital converter count corresponding to +5 volts is 32768. The count corresponding to 1.5 volts is $(1.5 * 32768 / 5) = 9830$. The lowest possible count for the analog-to-digital converter is -32768, and only samples above 9830 are of interest, so the region (OUTSIDE, -32768, 9830) is specified in the LIMIT task definition. A hysteresis region is specified by repeating the parameters for the trigger region. Hysteresis disables further trigger assertions until the input values leave the hysteresis region. Hysteresis is described in more detail later in this chapter.

When there is a trigger event, the LIMIT task asserts trigger T and puts the sample number of the event into trigger T. Trigger T is a software signal which the WAIT task uses to synchronize data transfers with trigger events. WAIT synchronizes data transfer to trigger events by matching the sample number of the trigger with the number of samples that WAIT reads. Matching sample numbers provides exact synchronization and is not dependent on response time between tasks.

A WAIT task continuously empties unwanted data out of the input channel pipe while waiting for a trigger event. For each trigger event, the WAIT task transfers a specified number of values before and after the trigger event from the input channel pipe to a pipe. LIMIT asserts a trigger on the rising edge of each wave. WAIT then transfers samples taken before and after the time of the rising edge of each wave.

In this application, the WAIT task transfers 10 pre-trigger and 40 post-trigger data values to the PC using the \$BINOUT communication pipe.

DAPview Note: When graphing data from a WAIT task, it often is desirable to display data from each trigger event in a new frame. In the example above, this is accomplished by setting Graph Number of Points to 50.

Application 19 — Peak Detection

A PEAK task finds maxima or minima and asserts a trigger at the locations of the peaks. The trigger can be used to extract data relative to the maxima or minima. The following application, illustrated in Figure 14, transfers 10 data values before and 10 data values after each peak in the data from input channel pipe 0.

```
RESET
PIPES P1
TRIGGER T1
IDEF A 1
  SET IPIPE0 S0
  TIME 1000
END
PDEF B
  AVERAGE (I P0, 32, P1)
  PEAK (P1, 2, T1)
  WAIT (P1, T1, 10, 10, $BINOUT)
END
START A, B
```

Electrical noise may cause the least significant bits of the conversion data to fluctuate. PEAK is sensitive to noise because it looks for changes in the derivatives of data values. Noise may result in detection of many spurious peaks when input channel pipe data is changing slowly.

To prevent noise from influencing the results of a PEAK task, data can be smoothed before peak detection. The above example smoothed the data by averaging blocks of 32 data points. Another way of masking noise is to implement a low pass digital filter with a FIRFILTER task.

PEAK allows an optional parameter to force detection of only those peaks which lie in a specified region. To ignore peaks occurring close to zero, for example, the parameter list of the previous PEAK process could be modified as follows:

```
PEAK (P1, 2, T1, OUTSIDE, -100, 100)
```

Another application could differentiate the data presented to PEAK using the DELTA process to find points of maximum slope.

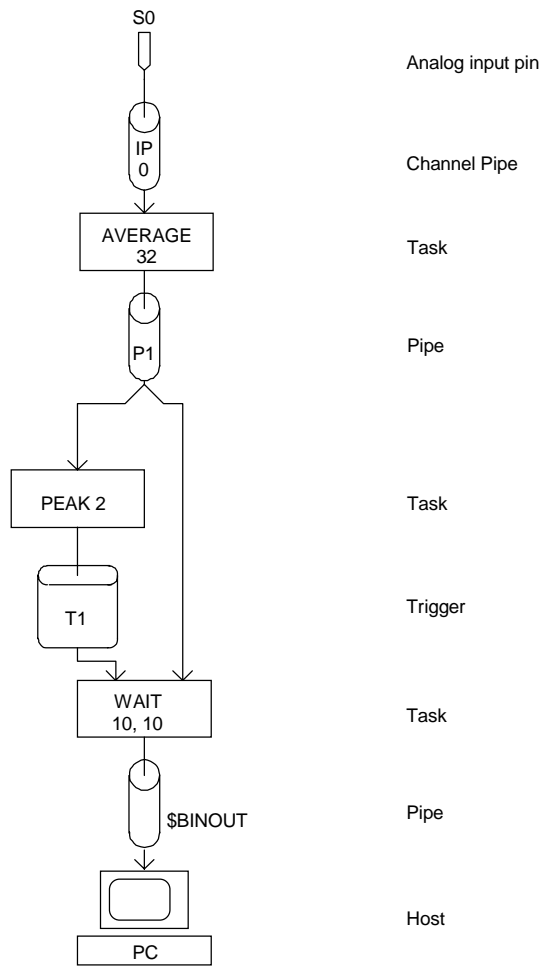


Figure 14. Peak Detection

Application 20 — Implementing a Digital Oscilloscope

This application implements a form of digital oscilloscope. An input channel pipe is scanned for a trigger event — in this case a positive slope exceeding 100. When a trigger event occurs, the values in four input channel pipes are transferred to the host. For each input channel pipe, 100 samples are acquired, 50 samples from before the trigger event and 50 samples from after the trigger event. A diagram of this application is in Figure 15.

```
RESET
PIPES P0, P1, P2, P3
TRIG T1 4
IDEF A 4
    SET IP0 S0
    SET IP1 S1
    SET IP2 S5
    SET IP3 S7
    TIME 5000
    END
PDEF B
    DLIMIT (IP0, OUTSIDE, -32768, 100, T1, OUTSIDE, -32768, 100)
    WAIT (IP0, T1, 50, 50, P0)
    WAIT (IP1, T1, 50, 50, P1)
    WAIT (IP2, T1, 50, 50, P2)
    WAIT (IP3, T1, 50, 50, P3)
    MERGE (P0, P1, P2, P3, $BINOUT)
    END
START A, B
```

The definition of trigger T1 is followed by the number 4. This specifies that T1 signals four tasks. In this example, T1 signals four WAIT tasks to provide triggering on four input channel pipes.

Notice that two tasks read data from input channel pipe 0. DAPL allows any number of tasks to share data from an input channel pipe.

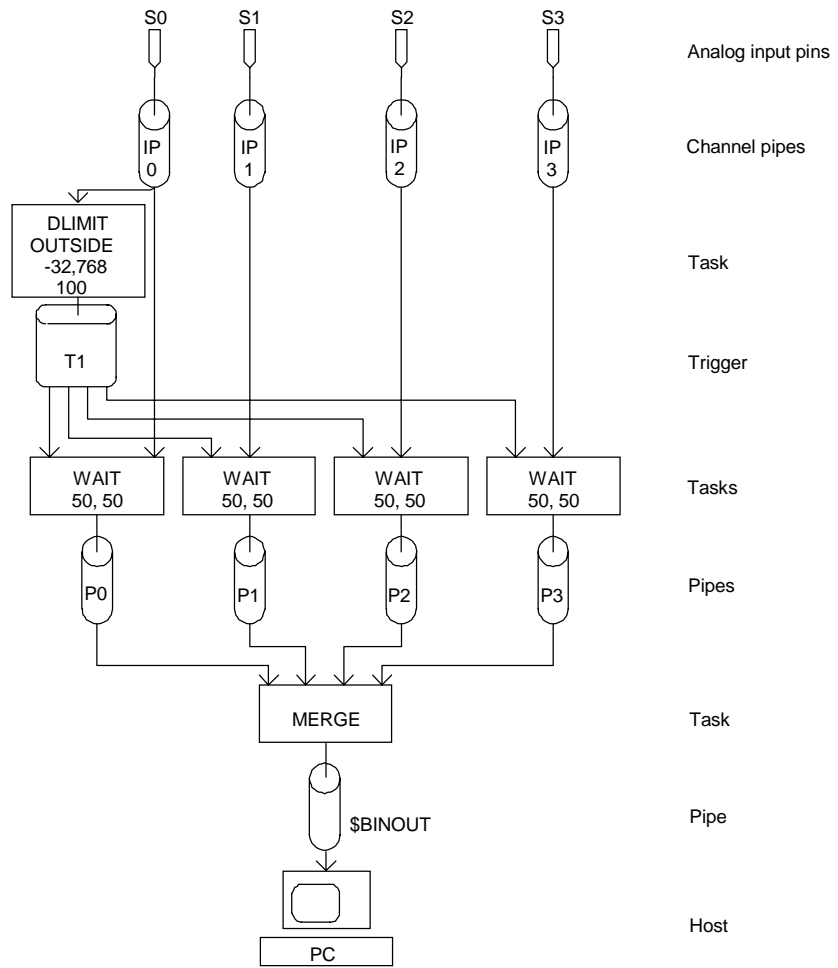


Figure 15. Digital Oscilloscope

Application 21 — Using Hysteresis

Both `LIMIT` and `DLIMIT` allow optional hysteresis specifications. Hysteresis prevents multiple triggering from a single event. For example, suppose a sawtooth waveform is being sampled, triggering when the wave exceeds a trigger value. See Figure 16.

The dotted line in figure 16 indicates the trigger threshold level. With no hysteresis, data points 7, 8, 9, 10, and 11 cause trigger assertions. If one trigger per wave is desired, only data point 7 should cause a trigger. The next trigger event should be the start of the next waveform — point 27. To accomplish this, a hysteresis region must be specified in the `LIMIT` parameter list:

```
LIMIT ( <pi pe>, <regi on1>, <tri gger>, <regi on2> )
```

After a value inside `<regi on1>` is found, the trigger is asserted. Subsequent values are ignored until a value outside `<regi on2>` is detected. The `LIMIT` task then resumes scanning the data for values inside `<regi on1>`.

For the sawtooth waveform example, the hysteresis region can be the same as the trigger region. Now, after a trigger, the data values must leave the hysteresis region before another trigger occurs. A possible `LIMIT` parameter list is:

```
LIMIT (P, INSIDE, 500, 32767, T, INSIDE, 500, 32767)
```

By varying the size of the hysteresis region, a wide variety of triggering conditions can be specified.

For example, the following command list configures a Data Acquisition Processor to scan an input channel pipe for values that are greater than or equal to 10,000 and less than or equal to 20,000 and to print the sample numbers of all samples in that range.

```

RESET
PIPE P1 LONG
TRIGGER T1
IDEF A 1
  SET IPO SO
  TIME 500
  END
PDEF B
  LIMIT (IPO, INSIDE, 10000, 20000, T1)
  TSTAMP (T1, P1)
  MERGE(P1, $BINOUT)
  END
START A, B

```

The LIMIT task asserts a trigger each time it detects a value in the region (INSIDE, 10000, 20000). The TSTAMP task extracts the time of each trigger assertion and puts the sample number into pipe P1.

Pipe P1 is declared as a long pipe. This means that a data value is stored internally as a 32-bit integer, a number in the range from -2,147,483,648 to 2,147,483,647. All trigger counts are represented as long integers. LONG integers appear in applications in which the normal 16-bit integer range from -32,768 to 32,767 is too restrictive.

The MERGE task transfers the 32-bit sample count from pipe P1 to the 16-bit binary communications pipe \$BINOUT. When MERGE transfers a long value to a word output pipe, two words are placed in the output pipe. The least significant word is followed by the most significant word.

DAPview Note: To display the 32-bit values generated by TSTAMP, replace the MERGE task with a FORMAT task. The FORMAT task sends values to the PC as text using the text communications pipe \$SYSOUT. For more information on text communication from the Data Acquisition Processor see the Communications chapter in this manual.

Unless values in the particular region (INSIDE, 10000, 20000) are rare, this application may overload the PC. Adding hysteresis reduces the number of trigger events. If the parameter list of LIMIT is changed to the line below, trigger T1 is asserted the first time a value in the region (INSIDE, 10000, 20000) is detected and is not asserted again until a value outside the region (INSIDE, 9000, 21000) is detected.

```

LIMIT (0, INSIDE, 10000, 20000, T1, INSIDE, 9000, 21000)

```

Hysteresis provides the ability to trigger only once. When a hysteresis region covers the entire range, no values can leave the hysteresis region to reset triggering. After the first trigger event, the procedure must be stopped and restarted to enable additional trigger events.

LIMIT (0, INSIDE, 10000, 20000, T1, INSIDE, -32768, 32767)

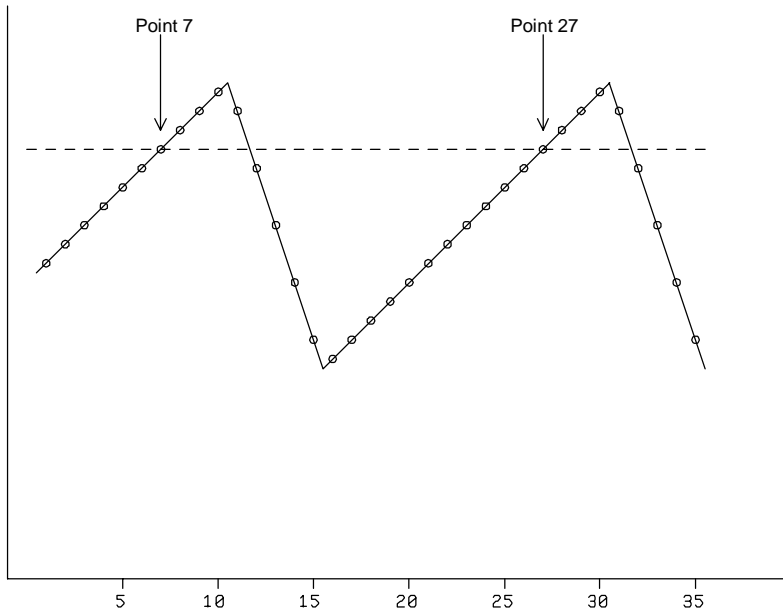


Figure 16. Triggering Hysteresis

Application 22 — Triggering on Two Conditions

This application monitors two input channel pipes. If values from either channel pipe rise above a trigger region, blocks of data from both input channel pipes are sent to the PC. A TOR task merges trigger events to allow both input channel pipes to send data.

```
RESET
PIPES P1, P2
TRIGGER T1, T2, T3 2
I DEF A 2
    SET IPO S0
    SET IP1 S1
    TIME 1000
END
PDEF B
    LIMIT (IPO, INSIDE, 10000, 32767, T1, INSIDE, 10000, 32767)
    LIMIT (IP1, INSIDE, 10000, 32767, T2, INSIDE, 10000, 32767)
    WAIT (IPO, T3, 1000, 19000, P1)
    WAIT (IP1, T3, 1000, 19000, P2)
    TOR (T1, T2, T3)
    MERGE(P1, P2, $BINOUT)
END
START A, B
```

Figure 17 is a diagram of this application.

The DAPL command list configures an input procedure to sample two input pins at 500 Hz. A processing procedure defines two LIMIT tasks that read data from the input channel pipes and assert triggers T1 and T2 whenever their respective values are at least 10,000. A hysteresis region is specified so that once a value is found in the trigger region, subsequent values are ignored until a value outside of the trigger region is detected. The triggers are merged into T3 by a TOR task. Two WAIT tasks transfer data from the input channel pipes to P1 and P2 whenever trigger T3 is asserted.

Note: There also is a TAND command for combining triggers with a logical AND operation.

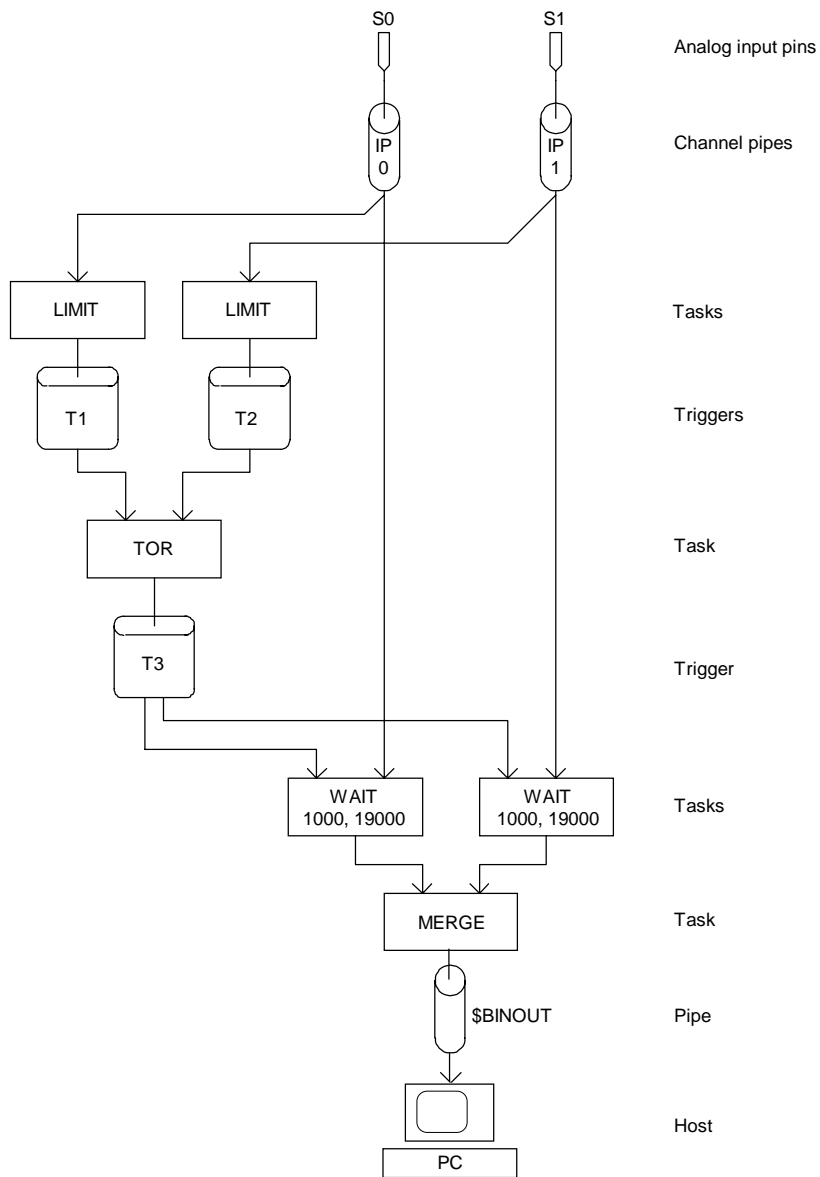


Figure 17. Combining Triggers

Application 23 — Retriggerring

A WAIT task transfers a block of data for each valid trigger event and ignores subsequent trigger events which occur within the block of data being transferred, as these would cause it to transfer overlapping blocks. In some applications, it is important to save a block of data around every trigger event, even when trigger events occur close together. This can be achieved by processing the triggers with a RETRIGGER task.

This application uses a RETRIGGER task to save a block of at least 1000 data values around every trigger event, 100 values before the trigger event and 900 values after.

```
RESET
TRIGGER T1, T2
I DEF A 1
  SET IPO S0
  TIME 10000
END
PDEF B
  WAIT (IPO, T2, 100, 900, $BINOUT)
  LIMIT (IPO, INSIDE, 0, 32767, T1, INSIDE, 0, 32767)
  RETRIGGER (T1, 1000, T2)
END
START A, B
```

Figure 18 is a diagram of this application.

The input procedure configures the Data Acquisition Processor to sample one input pin at 100 Hz. A WAIT task waits for a valid trigger event to begin transferring data to \$BINOUT. A LIMIT task scans input channel pipe 0 for a trigger condition and places trigger events in T1. RETRIGGER transfers trigger events to T2 in a way which ensures that WAIT transfers at least 100 values before each trigger event and 900 values after each trigger event. If triggers occur close together, RETRIGGER places trigger events 1000 counts apart.

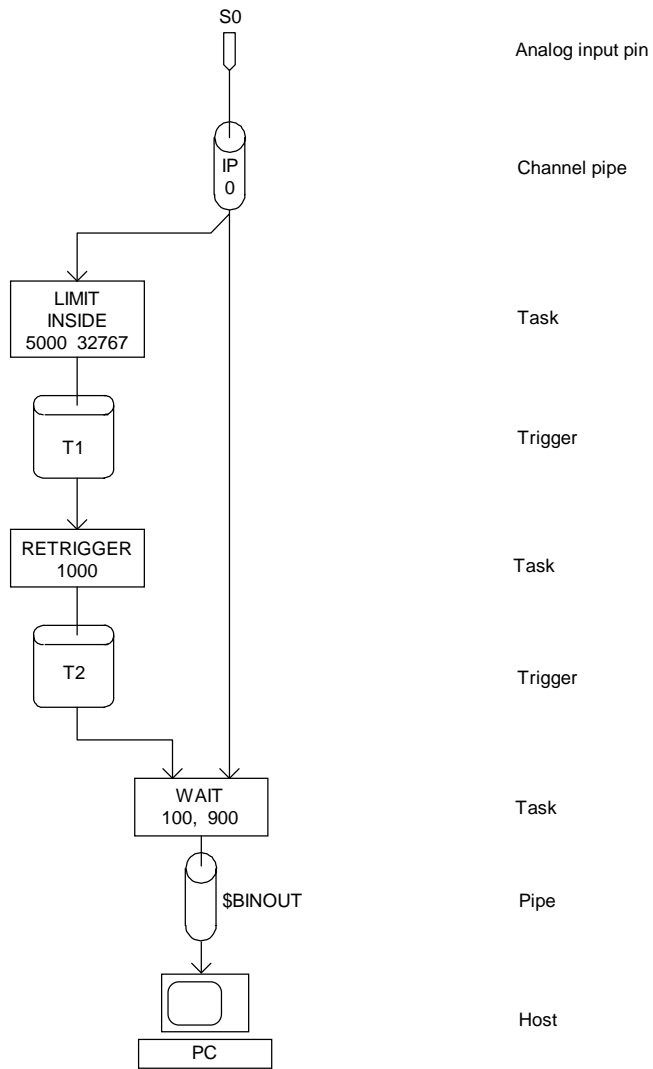


Figure 18. Capturing Every Trigger Event

Application 24 — Spike Detection

Some applications sample signals containing occasional high-speed spikes. To resolve the spikes, the sampling speed should be high. In order to avoid overloading the PC, a trigger should be used to let the Data Acquisition Processor discard most of the uninteresting data.

As an example, assume spikes have a typical duration of two milliseconds and that 20 samples of a spike waveform are required. If a sample value greater than 10,000 indicates that a spike has occurred, the following application produces the required results:

```
RESET
TRIG T1
I DEF A 1
  SET IPO S0
  TIME 100
  END
PDEF B
  LIMIT (IPO, OUTSIDE, -32768, 10000, T1, OUTSIDE, -32768, 10000)
  WAIT (IPO, T1, 10, 10, $BINOUT)
  END
START A, B
```

If a spike occurs approximately every second, this application transfers data at a rate of 20 values per second, even though data are being sampled and analyzed at 10,000 samples per second.

If spikes occur substantially faster than once per second, this application may produce too much data for the host PC to handle. A possible approach is to average groups of 100 spikes and return the averages, as shown in the following command list:

```
RESET
PIPE P1
TRIG T1
IDEF A 1
  SET IPO S0
  TIME 100
END
PDEF B
  LIMIT (IPO, OUTSIDE, -32768, 10000, T1, OUTSIDE, -32768, 10000)
  WAIT (IPO, T1, 10, 10, P1)
  BAVERAGE (P1, 20, 100, $BINOUT)
END
START A, B
```

The BAVERAGE task reads 100 blocks of 20 values and computes an averaged block of data. After the BAVERAGE task processes 100 blocks, it writes a single block of 20 values to \$BINOUT.

The MINTIME command also is available to restrict excessive triggering. A MINTIME task ignores triggers which occur too close together:

```
RESET
TRIGGERS T1, T2
IDEF A 1
  SET IPO S0
  TIME 100
END
PDEF B
  LIMIT (IPO, OUTSIDE, -32768, 10000, T1, OUTSIDE, -32768, 10000)
  WAIT (IPO, T2, 10, 10, $BINOUT)
  MINTIME (T1, 10000, T2)
END
START A, B
```

This example forces consecutive triggers T2 to be spaced at least 10,000 samples apart, corresponding to a maximum of one trigger per second.

Another data reduction alternative is to ignore a specified fraction of the trigger events. The NTH task in the following command list ignores 99 out of every 100 spikes.

```
RESET
TRIGGERS T1, T2
IDEF A 1
  SET IPO S0
  TIME 100
  END
PDEF B
  LIMIT (IPO, OUTSIDE, -32768, 10000, T1, OUTSIDE, -32768, 10000)
  WAIT (IPO, T2, 10, 10, $BINOUT)
  NTH (T1, 100, T2)
  END
START A, B
```

Application 25 — Time Stamping Pulses

This application records the times at which pulses occur and the peak values in the input channel pipe in a short time interval following the trigger. In order to make the observations less sensitive to noise, the application averages data in blocks of four values.

Figure 19 illustrates this application.

```
RESET
PIPES P1, P2, P3 LONG, P4
TRIGGER T 2, T1
IDEFINE A 1
    TIME 250
    SET IPO D3
END
PDEFINE B
    AVERAGE (IPO, 4, P1)
    LIMIT (P1, OUTSIDE, -32768, 9830, T, OUTSIDE, -32768,
           9830)
    WAIT (P1, T, 0, 50, P2)
    MINTIME (T, 50, T1)
    TSTAMP (T1, P3)
    HIGH (P2, 50, P4)
    MERGE (P3, P4, $BINOUT)
END
START A, B
```

Notice the number 2 in the definition of trigger T. This number tells DAPL that the trigger signals two tasks, as seen in Figure 19.

To allow for averaging, the sampling time is reduced from 1000 microseconds to 250 microseconds. Input channel pipe 0 passes data to an AVERAGE task. The AVERAGE task computes averages of groups of four samples and places the results in pipe P1.

The LIMIT task reads the averaged data from pipe P1. When it finds an average above the specified limit of 9830, it asserts trigger T. Hysteresis prevents multiple triggers from a single pulse. Each time a trigger event occurs, the WAIT task synchronized by trigger T transfers data from pipe P1 to pipe P2.

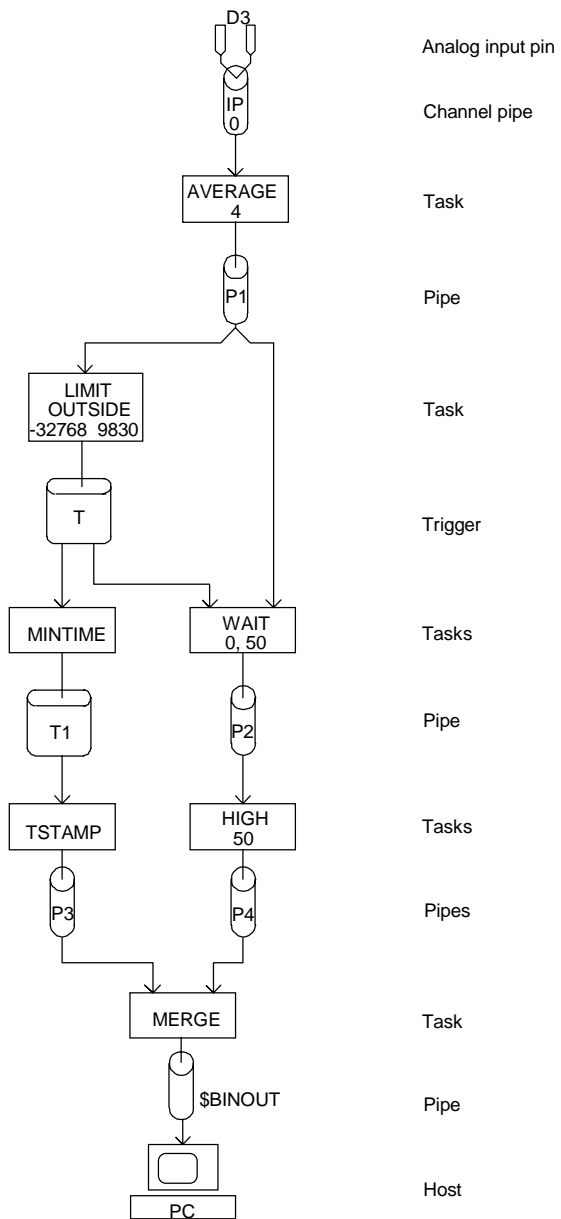


Figure 19. Time Stamping Pulses

The TSTAMP task waits for a trigger event, determines the sample count of each trigger event, and places the sample count in long pipe P3. The sample counts should be multiplied by the time between samples to give the time from the start of sampling. A MI NTI ME task processes the trigger to not allow triggers that are closer together than 50 values. This is necessary because triggers that are closer than 50 values apart are ignored by the WAIT task but are used by the TSTAMP task. In this application, MI NTI ME ensures that time stamp values are matched properly with data values.

The HI GH task reads each block of 50 data values generated by the WAIT task, determines the maximum of each block, and places the maxima in pipe P4.

Finally, the MERGE task transfers the contents of pipes P3 and P4 to \$BI NOUT. Since pipe P3 is declared as a long pipe, MERGE transfers a total of three words to \$BI NOUT for each trigger event. The first two words are the low and high-order words of the trigger event sample number from P3. The third word is the maximum value of the input channel pipe after the trigger.

It is important to note that this application generates data for P3 and P4 at the same rate — each trigger event generates one number for P3 and one number for P4. If P3 and P4 received data at different rates, a MERGEF task would be required to transfer flagged data to \$BI NOUT.

Application 26 — Detecting Bit Transitions

This application detects transitions at the digital input port. The application prints each sample number at which the digital input value changes, together with the new value. This application could be used, for example, to analyze the contact bounce behavior of a multiple pole switch.

```
RESET
PIPE P1, P2 LONG
TRIG T1 2
IDEF A 1
  SET IPO B0
  TIME 100
END
PDEF B
  CHANGE (IPO, T1)
  WAIT (IPO, T1, 0, 1, P1)
  TSTAMP (T1, P2)
  MERGE (P2, P1, $BINOUT)
END
START A, B
```

The CHANGE task scans input data and asserts a trigger each time an input value changes. The WAIT task passes each new data value to pipe P1. The TSTAMP task converts each trigger assertion to a time stamp. The sample count and new value are transferred by merging P1 and P2 to \$BINOUT.

CHANGE also can be used to scan for changes in analog data. In analog applications an optional third parameter usually is included. When this parameter is present, the CHANGE task asserts a trigger only when its input changes by more than the value of the parameter. For example:

```
CHANGE (IPIPE0, T1, 100)
```

This task asserts T1 when consecutive samples change by more than 100 counts. This is a useful command for implementing data compression for slowly varying data.

The LOGIC command provides another means of detecting changes in binary input data. LOGIC accepts three parameters which specify the bit transitions to detect. These parameters are described fully in the DAPL Manual. The following procedure outlines how to compute useful values for the three LOGIC parameters:

1. Determine which bits to use for triggering. Set the corresponding bits of the third parameter to one.
2. To trigger when a particular bit goes from 0 to 1, set the corresponding bit of second parameter to zero. To trigger when a particular bit goes from 1 to 0, set the corresponding bit of second parameter to one.
3. Set the fourth parameter to the number zero.
4. If more than one bit is used as a trigger, LOGIC asserts the trigger on the transition that causes all the enabled bits to be in their triggering states.

The following example asserts trigger T1 when bit 3 of the digital input port changes to one and bit 1 of the digital input port changes to zero:

```
RESET
TRIGGER T1
IDEF A 1
  SET IPIPEO B0
  TIME 1000
END
PDEF B
  LOGIC (IPIPEO, 2, 10, 0, T1)
END
```

Application 27 — Using Triggers to Calculate Frequency

A trigger carries information about the times at which trigger events occur. For many applications, the frequency at which events occur is important, but the times are not. The Data Acquisition Processor can convert trigger data to frequency data. The following command list computes the frequency at which the binary input port is changing its value:

```
RESET
TRIG T1
I DEF A 1
  SET IPO B0
  TIME 10000
  END
PDEF B
  CHANGE (IPO, T1)
  FREQUENCY (T1, 100, $BINOUT)
  END
START A, B
```

The numbers printed by this application indicate the numbers of trigger assertions in each 100 samples. Since the sampling frequency is 100 samples/second, the numbers represent the trigger assertion frequency in Hz. In other applications, it may be necessary to scale the output of a frequency task to produce a frequency in Hz.

7. Further Real-Time Processing

Application 28 — Finding Deviations between Inputs

A DAPL expression can be used in conjunction with `LIMIT` to find deviations between two inputs. For example, the following application determines the locations at which data from two input channel pipes differ by more than 100.

```
RESET
PIPES P1, P2 LONG
TRIG T1
IDEF A 2
  SET IPO S0
  SET IP1 S5
  TIME 10000
  END
PDEF B
  P1 = IPO - IP1
  LIMIT (P1, OUTSIDE, -100, 100, T1)
  TSTAMP (T1, P2)
  MERGE(P2, $BINOUT)
  END
START A, B
```

Figure 20 displays a diagram of this application.

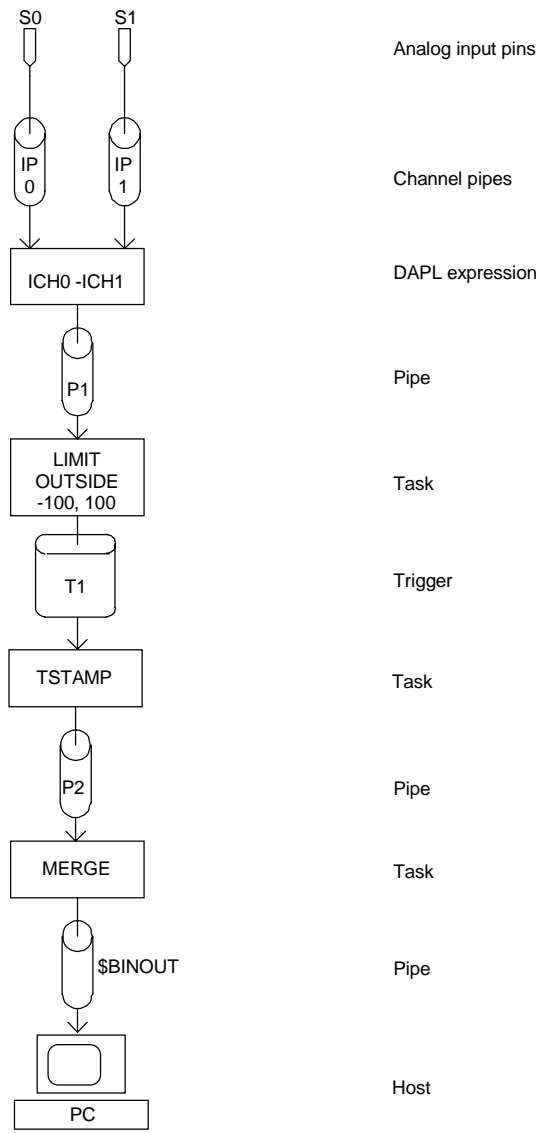


Figure 20. Scanning for Deviations Between Inputs

Application 29 — Thermocouple Linearization

Thermocouples are temperature sensors that generate voltages that depend on temperature. The relationship between voltage and temperature is nonlinear, so special computations must be performed to convert thermocouple voltages to temperatures.

Cold junction compensation establishes a reference voltage which is compared to the voltage generated by a thermocouple. Thermocouples usually require cold junction compensation.

The following command list can be used to sample a thermocouple, linearize the thermocouple data, and transfer one temperature reading every second. The two thermocouple leads should be connected to pins S0 and G0 of the Data Acquisition Processor. In this case the cold junction temperature is assumed to be fixed at 20.0 degrees; the cold junction temperature also can be measured by the Data Acquisition Processor.

```
RESET
PIPE P1
VARIABLE REF=200
I DEF A 1
    SET I PIPE0 S0 100
    TIME 10000
    END
PDEF B
    AVERAGE (I PIPE0, 100, P1)
    THERMO (P1, 1, 5000, 32767, $BINOUT, REF)
    END
START A, B
```

In this DAPL command list, the first THERMO parameter is the input pipe for thermocouple data. Notice the use of averaging to reduce noise. The second parameter specifies the type of thermocouple sensor being measured — in this case, a J-type thermocouple. The next two parameters define the input scale factor, which is derived below. The variable REF contains the temperature of the reference junction, as described below. Temperature data is sent to the host PC through the binary communications pin, \$BINOUT.

The THERMO command receives input voltages scaled so that each count represents ten microvolts. If the Data Acquisition Processor is configured for a -5 to +5 volt input range and the input pin is sampled at a gain of 100, then a digitized value of 32768 corresponds to a 0.05 volt input. Each count represents $0.05/32768$ or 1.53

microvolts. To convert these values to ten microvolt units, multiply digitized data by a scale factor of 1.53/10. This scale factor can be converted to a ratio of integers, 5000/32768. THERMO scale factors are 16-bit numbers, so the ratio used in the THERMO task is adjusted to 5000/32767.

The variable REF supplies information for cold junction compensation. Thermocouple cold junction compensation establishes a reference voltage which is compared to the voltage generated by the thermocouple. The reference voltage is determined by the temperature of the termination board to which the thermocouple leads are connected. In the above example, the reference temperature (expressed in tenths of a degree Celsius) is set to 20.0 degrees.

If the reference junction temperature changes over time, the Data Acquisition Processor can be configured to automatically sense the reference junction temperature and keep the variable REF updated using the PVALUE command. Some Analog Termination Boards from Microstar Laboratories have provisions for a thermistor circuit for sensing the reference temperature. See the end of this application for more information.

Improving Thermocouple Accuracy

The accuracy of thermocouple sampling can be improved by removing a DC ground offset from the thermocouple readings. The THERMO command allows a parameter which removes ground offsets. This parameter is a word variable that should be kept updated to the value of the Data Acquisition Processor internal ground reference. The following command list illustrates offset correction:

```
RESET
PIPES P1, P2
VARIABLE REF=200, GRND
I DEF A 2
  SET IPIPE0 G 100
  SET IPIPE1 SO 100
  TIME 5000
END
PDEF B
  AVERAGE (IP0, 100, P1)
  AVERAGE (IP1, 100, P2)
  PVALUE (P1, GRND)
  THERMO (P2, 1, 5000, 32767, GRND, $BINOUT, REF)
END
START A, B
```

The value of ground is sampled at the same gain as the thermocouple. The ground data values are averaged and placed in pipe P1. The PVALUE command keeps the variable GRND updated to the current ground reading. The THERMO command subtracts this ground value from the thermocouple readings.

Note: This command list updates the value of the ground offset variable, GRND, once per second. Since the value of GRND defaults to zero, the first temperature value displayed will not include a ground offset correction. This behavior can be avoided by using a separate set of input and processing procedures to initialize the value of GRND before running this application.

Converting Temperatures to Fahrenheit

The output units of the THERMO command are in tenths of a degree Celsius. A DAPL expression can be used to convert the data to degrees Fahrenheit:

```
RESET
PIPES P1, P2, P3
VARIABLE REF=200, GRND
IDEF A 2
  SET IPIPE0 G 100
  SET IPIPE1 SO 100
  TIME 5000
END
PDEF B
  AVERAGE (I P0, 100, P1)
  AVERAGE (I P1, 100, P2)
  PVALUE (P1, GRND)
  THERMO (P2, 1, 5000, 32767, GRND, P3, REF)
  $BINOUT = (P3*9/5) + 320
END
START A, B
```

In this example, if a value of 540 is sent to the PC, the temperature in Fahrenheit is 54.0°. If you want to send data to the PC with the decimal in place, send the data as text using a FORMAT command. Declare pipe P4 and use the following lines:

```
P4 = (P3*9/5) + 320
FORMAT (P4: 1)
```

The : 1 decimal point specification adds a decimal point to the data sent to the PC.

Sampling Several Thermocouples

Many thermocouples can be monitored simultaneously. The following command list illustrates reading three thermocouples every second, including linearization and conversion to Fahrenheit units:

```
RESET
PIPES P1, PA2, PB2, PC2, PA3, PB3, PC3, PA4, PB4, PC4
VARIABLE REF=200, GRND
IDEF A 4
  SET IPIPE0 G 100
  SET IPIPE1 S0 100
  SET IPIPE2 S1 100
  SET IPIPE3 S2 100
  TIME 2500
  END
PDEF B
  AVERAGE (IP0, 100, P1)
  AVERAGE (IP1, 100, PA2)
  AVERAGE (IP2, 100, PB2)
  AVERAGE (IP3, 100, PC2)
  PVALUE (P1, GRND)
  THERMO (PA2, 1, 5000, 32767, GRND, PA3, REF)
  THERMO (PB2, 1, 5000, 32767, GRND, PB3, REF)
  THERMO (PC2, 1, 5000, 32767, GRND, PC3, REF)
  PA4 = (PA3*9/5) + 320
  PB4 = (PB3*9/5) + 320
  PC4 = (PC3*9/5) + 320
  MERGE (PA4, PB4, PC4, $BINOUT)
  END
START A, B
```

Sampling Many Thermocouples

When sampling many thermocouples, it is possible to define more tasks than can fit into available heap memory. One way to avoid heap memory overflow is to merge all thermocouple inputs into a single data stream. The BAVERAGE and THERMO commands can process merged thermocouple data while keeping values in order so they can be separated later. Only a few commands are needed to process many thermocouples. The following example uses ten thermocouple inputs for illustration.

```
RESET
PIPES P1, P2, P3
VARIABLE REF=200, GRND
IDEF A 11
  SET IP0 G 100
  SET IP1 S0 100
  SET IP2 S1 100
  SET IP3 S2 100
  SET IP4 S3 100
  SET IP5 S4 100
  SET IP6 S5 100
  SET IP7 S6 100
  SET IP8 S7 100
  SET IP9 S8 100
  SET IP10 S9 100
  TIME 2000
END
PDEF B
  AVERAGE (IP0, 100, P1)
  PVALUE (P1, GRND)
  BAVERAGE (IP(1, 2, 3, 4, 5, 6, 7, 8, 9, 10), 10, 100, P2)
  THERMO (P2, 1, 5000, 32767, GRND, P3, REF)
  $BINOUT = (P3*9/5) + 320
END
START A, B
```

The BAVERAGE command averages corresponding values in blocks of data from input channel pipe 1 through 10 to get the average of each input channel pipe. A single THERMO command calculates the temperatures for each thermocouple value. Temperature values from all thermocouples are converted to Fahrenheit and transferred to \$BINOUT using a DAPL expression.

This technique can be expanded to process more thermocouples by adding more input channel pipes to the BAVERAGE command.

Note that all thermocouples in this example are the same type. One THERMO command is needed for each thermocouple type. More thermocouple types can be processed by adding more THERMO commands.

Sensing Reference Temperature

Data Acquisition Processor termination boards are available with reference temperature sensing circuits. A reference temperature circuit is used to measure the temperature of the termination board. Since the cold junction temperature is the same for all thermocouples connected to a termination board, only one reference temperature circuit is needed for any number of thermocouples.

The reference temperature sensing device on an Analog Termination Board is a thermistor. The reference temperature sensing device on a DAP 800 termination board and a DAP 1216a and DAP 2416a termination board is a solid state integrated circuit. The following two sections describe how to measure reference temperature with either type of temperature sensing device.

For the Analog Termination Board, the reference temperature circuit generates a temperature dependent differential voltage which is connected to differential input D4 of the termination board. The Data Acquisition Processor samples this voltage and the resultant information is used in the THERMO command for cold junction compensation.

Note: When the cold junction compensation circuit is installed, no other inputs should be connected to the S8 and S9 terminals of the termination board, as they are connected to the cold junction compensation circuit.

The following command list illustrates how cold junction compensation is used with an Analog Termination Board and a Data Acquisition Processor with an input range of +/- 5 volts. An example for a DAP 800, DAP 1216a, or DAP 2416a is provided later in this chapter.

```

RESET
PIPES P1, P2, P3, P4, P5, P6
VECTOR X=(-5000, -4000, -3000, -2000, -1000,
0, 1000, 2000, 3000, 4000, 5000)
VECTOR Y=(1086, 1378, 1663, 1943, 2221, 2500,
2783, 3071, 3368, 3678, 4004)
VARIABLES GRND, REF
IDEF A 3
SET IPIPE0 D4
SET IPIPE1 G 100
SET IPIPE2 SO 100
TIME 2500
END
PDEF B
AVERAGE (IP0, 100, P1)
AVERAGE (IP1, 100, P2)
AVERAGE (IP2, 100, P3)
PVALUE (P2, GRND)
P4 = P1 * 5000 / 32767
INTERP (P4, X, Y, P5)
P6 = P5 / 10
PVALUE (P6, REF)
THERMO (P3, 2, 5000, 32767, GRND, $BINOUT, REF)
END
START A, B

```

Figure 21 illustrates this application.

The vectors X and Y are used with the INTERP command and a DAPL expression to convert the nonlinear voltage output from the reference temperature circuit to units of tenths of degrees Celsius. This temperature is used as the cold junction reference temperature in the THERMO command. The values in vectors X and Y were taken from the data sheets for the thermistor.

If the Data Acquisition Processor is set up for a +/- 10 volt input range, this command list needs to be modified. Instead of 5000 in both the THERMO command and the DAPL expression that computes P4, 10000 is the correct number. This change provides the correct voltage scaling ratios for +/- 10 volt input range.

Note: When the cold junction compensation circuit is installed, no other inputs should be connected to the temperature reference input terminal of the termination board, as it is connected to the cold junction compensation circuit.

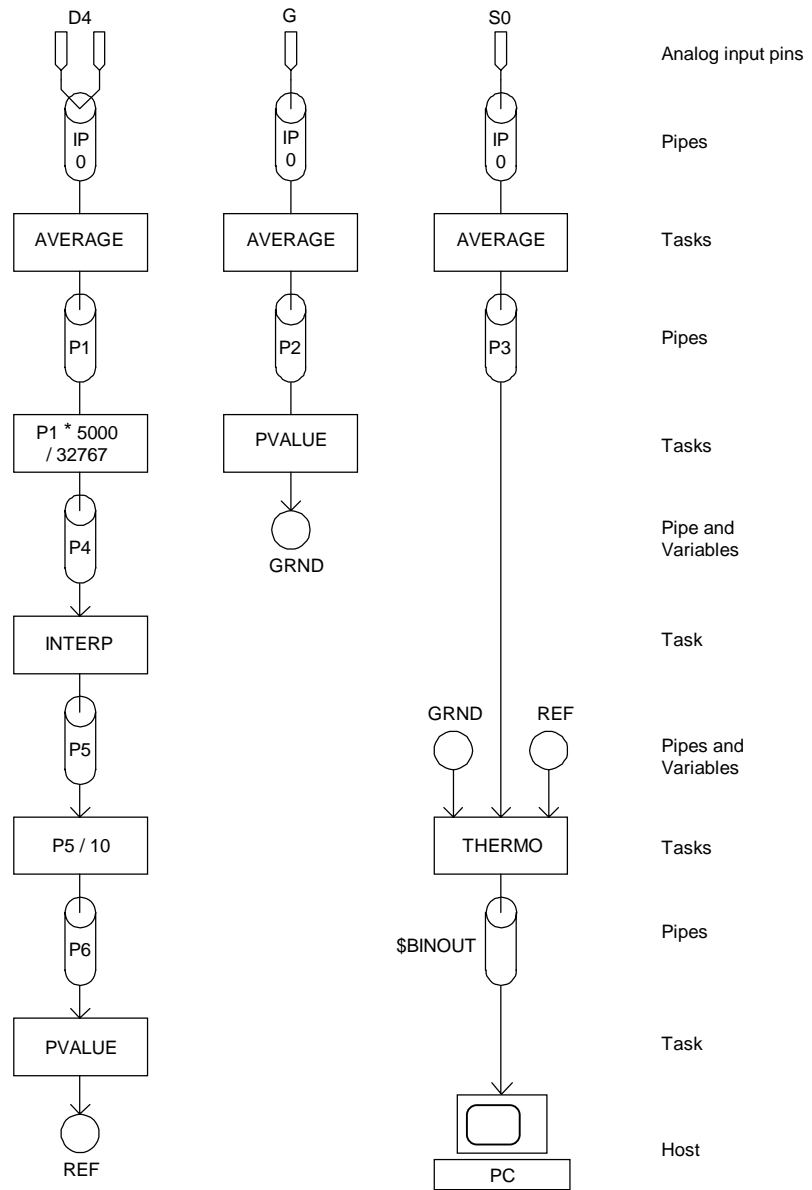


Figure 21. Thermocouple Cold Junction Compensation

For a DAP 800, DAP 1216a, and DAP 2416a, the reference temperature circuit generates a linear temperature dependent voltage. The reference temperature circuit is connected to single-ended input S7 of the DAP 800 termination Board, and S8 of the DAP 1216a and DAP 2416a termination board. The Data Acquisition Processor samples this voltage and the resultant information is used in the THERMO command for cold junction compensation.

The following command list illustrates how cold junction compensation is used with a DAP 800 input range of +/- 5 volts.

Note: For a DAP 1216a or DAP 2416a, change IPIPE0 to use S8 instead of S7.

```
RESET
PI PES P1, P2, P3, P4
VARIABLES GRND, REF
I DEF A 3
  SET I PIPE0 S7
  SET I PIPE1 G 100
  SET I PIPE2 SO 100
  TIME 2500
END
PDEF B
  AVERAGE (I P0, 100, P1)
  AVERAGE (I P1, 100, P2)
  AVERAGE (I P2, 100, P3)
  PVALUE (P2, GRND)
  P4 = P1 * 5000 / 32767
  PVALUE (P4, REF)
  THERMO (P3, 2, 5000, 32767, GRND, $BINOUT, REF)
END
START A, B
```

The voltage generated by the reference temperature circuit is converted to units of tenths of degrees Celsius with a DAPL expression. The THERMO command uses the resulting temperature value to perform cold junction compensation.

If the Data Acquisition Processor is set up for a +/- 10 volt input range, the previous DAPL command list needs to be modified. Instead of 5000 in both the THERMO task definition command and the DAPL expression defining the task which computes pipe P4, a value of 10,000 should be used. This change provides the correct voltage scaling ratios for +/- 10 volt input range.

Application 30 — Interpolation

Mathematical functions not provided in DAPL can be implemented with INTERP tasks. This application uses INTERP to provide a scaled fourth root function. Two vectors are defined, with the values in VECY equal to 1000 times the fourth roots of the values in VECX. INTERP uses these vectors for interpolation to provide a close approximation to the 4th root of any integer number from 0 to 32767. See Figure 22.

```
RESET
VECTOR VECX = (0, 1, 16, 81, 256, 1296, 4096,
               10000, 20736, 32767)
VECTOR VECY = (0, 1000, 2000, 3000, 4000, 6000, 8000,
               10000, 12000, 13454)

I DEF A 1
  SET IPO S0
  TIME 10000
END
PDEF B
  INTERP (IPO, VECX, VECY, $BINOUT)
END
START A, B
```

This DAPL listing defines vector VECX with 11 values from 0 to 32767. Vector VECY is defined with 11 values that correspond to the 1000 times the fourth roots of the values in VECX. Note that the values in each vector do not need to be evenly spaced.

For each sample value in input channel pipe 0, INTERP searches for the nearest values in VECX, interpolates between the corresponding values of VECY, and places the result in \$BINOUT.

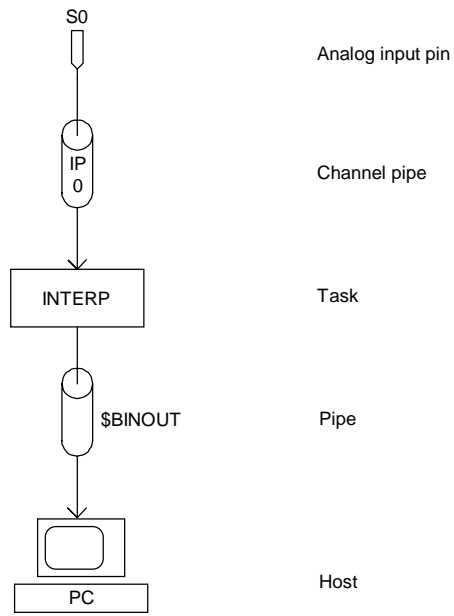


Figure 22. Exponential Calculation with I NTERP

Application 31 — Autoranging

This application demonstrates how to use the AUTORANGE command to increase the dynamic range of the Data Acquisition Processor. With autoranging, a Data Acquisition Processor configured for a -5 to +5 volt input range can resolve voltages as large as 5 volts and as small as 24.4 microvolts.

The following command list demonstrates autoranging. This application is illustrated in Figure 23.

```
RESET
EDIT $BINOUT WIDTH=LONG
PIPES P1, P2, P3, P4, P5, P6
IDEF A 6
  SET IPIPE0 DO 1
  SET IPIPE1 DO 10
  SET IPIPE2 DO 100
  SET IPIPE3 G 1
  SET IPIPE4 G 10
  SET IPIPE5 G 100
  TIME 1000
END
PDEF B
  AVERAGE (IP0, 16, P1)
  AVERAGE (IP1, 16, P2)
  AVERAGE (IP2, 16, P3)
  AVERAGE (IP3, 16, P4)
  AVERAGE (IP4, 16, P5)
  AVERAGE (IP5, 16, P6)
  AUTORANGE (6, P1, P2, P3, P4, P5, P6, $BINOUT, 16)
END
START A, B
```

A differential pin pair and six consecutive input channel pipes must be devoted to reading data for an AUTORANGE task. The differential pin pair is connected to the source of the input data. The first three input channel pipes read the input differential pin pair at gains of 1, 10, and 100. The next three input channel pipes read an internal ground reference at the same gains. This lets the software provide offset correction at each gain.

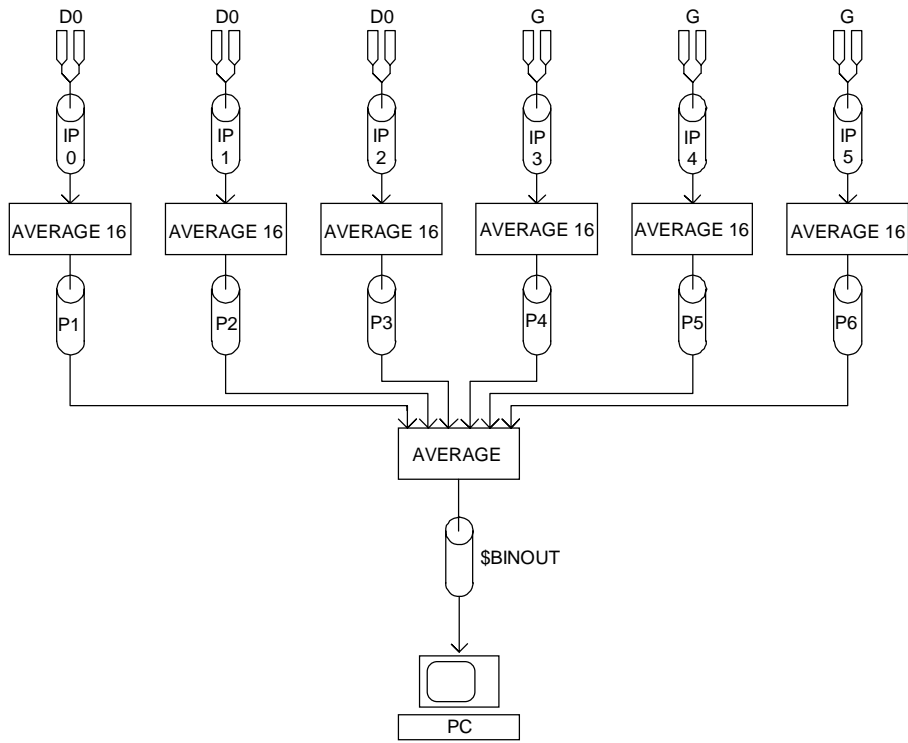


Figure 23. Autoranging

An AUTORANGE task normally receives data from an input channel pipe or an AVERAGE task. The dynamic range possible with autoranging makes the analog input sensitive to noise. To minimize noise effects, it is best to average the input before an AUTORANGE task. In the above application, the AUTORANGE task reads averaged data from pipes P1 through P6.

AUTORANGE interpolates three adjacent readings to form an extended range value. The AUTORANGE command compensates for interpolation errors caused by the different sampling times at the various gains. Averaging the input data allows the AUTORANGE command to reduce errors resulting from timing skew.

The autoranging operation generates a 23-bit integer. This is placed into \$BINOUT, which was modified to a long pipe by the EDIT task. The extended range integer resolves voltages as small as 24.4 microvolts on the -5 to +5 volt input range.

It is important to remember that AUTORANGE does not provide greater accuracy than the analog-to-digital converter can provide. AUTORANGE provides extended dynamic range by choosing the most precise value from one of three gains, multiplying the value by the gain, and placing the result in a long pipe. The long values still have the same relative accuracy as the analog-to-digital converter.

Application 32 — Identifying Maxima and Minima

The following application, illustrated in Figure 24, scans an input channel pipe for maxima and minima and prints the distances between successive maxima and the distances between successive minima:

```
RESET
PIPES P1, P2, P3 LONG, P4 LONG
PIPES P5 LONG, P6 LONG
TRIGGERS T1, T2
IDEF A 1
  SET IPO SO
  TIME 10000
  END
PDEF B
  AVERAGE (IPO, 16, P1)
  AVERAGE (IPO, 16, P2)
  PEAK (P1, 0, T1)
  PEAK (P2, 1, T2)
  TSTAMP (T1, P3)
  TSTAMP (T2, P4)
  DELTA (P3, P5)
  DELTA (P4, P6)
  MERGEF (P5, P6, $BINOUT)
  END
START A, B
```

To identify which data values are minima and which data values are maxima, the MERGEF commands is used in the above example. MERGEF includes an identifying flag with each data value; the flag shows which pipe the data value is from.

When using DAPview, two FORMAT tasks should be used in place of the MERGEF task shown above. The first FORMAT should print the data from pipe P5 and the second FORMAT should print the data from P6. DAPview does not accept long data values sent using MERGEF.

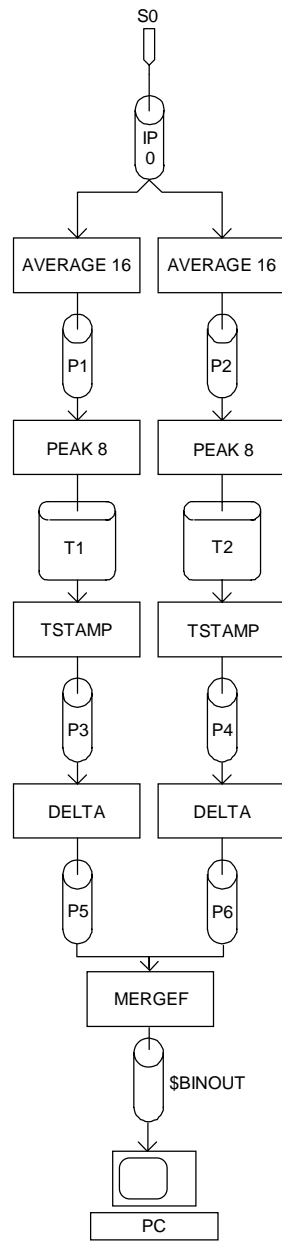


Figure 24. Using Prefixes to Identify Maxima and Minima

Application 33 — Almost Simultaneous Sampling

Some applications require sampling two or more analog inputs simultaneously. Although the Data Acquisition Processor does not support simultaneous sampling without external hardware, almost-simultaneous sampling can be implemented in software. With the following command list, the Data Acquisition Processor samples two inputs almost simultaneously at a rate of 1000 Hz.

```
RESET
I DEF A 100
  SET I P0 S0
  SET I P1 S1
  TIME 10
END
```

In input procedure A, 100 input channel pipes are specified. This causes the Data Acquisition Processor to sample two real input pins, S0 and S1, and 98 dummy input pins. Because of the input procedure declaration, the Data Acquisition Processor samples input pins S0 and S1 ten microseconds apart. The timing skew between input pins is only 1% of the effective sampling time.

Note: Dummy input channel pipes 2 through 99 do not cause input overflow since DAPL automatically purges data from input channel pipes that are not used by any tasks.

For true simultaneous sampling, Microstar Laboratories offers simultaneous sampling boards with either four or sixteen sample and hold amplifiers.

Application 34 — Mixing Fast Inputs and Slow Inputs

Some applications require sampling several input pins rapidly and several input pins slowly. Since the sampling time of an input procedure applies to all input pins that are defined, different sampling rates must be generated during data processing. The simplest way to provide different sampling speeds is to sample all pins at the speed of the fastest input pin and use `SKI P` commands to reduce the data rates of the slower input channel pipes.

For example, suppose an application requires sampling one input pin at 10,000 samples per second and four other input pins at 2,000 samples per second. The following DAPL commands accomplish this:

```
RESET
PIPES P1, P2, P3, P4, P5, P6, P7, P8, P9
I DEF A 5
  SET IPO S0
  SET IP1 S1
  SET IP2 S2
  SET IP3 S3
  SET IP4 S4
  TIME 20
  END
PDEF B
  SKI P (IP1, 0, 1, 4, P1)
  SKI P (IP2, 0, 1, 4, P2)
  SKI P (IP3, 0, 1, 4, P3)
  SKI P (IP4, 0, 1, 4, P4)
  HI GH (IPO, 100, P5)
  HI GH (P1, 100, P6)
  HI GH (P2, 100, P7)
  HI GH (P3, 100, P8)
  HI GH (P4, 100, P9)
  END
```

The input procedure samples each input pin at the rate of 10,000 samples per second. The four `SKI P` tasks, however, specify that only one out of every five samples is to be transferred its output pipe. This effectively reduces the sampling rate of the last four input pins to 2,000 samples per second. Input pin `S0` is sampled at 10,000 samples per second and no reduction is needed. In this application, `HI GH` commands illustrate typical processing.

This input procedure is inefficient, however, since the Data Acquisition Processor must acquire and process data at a rate of 50,000 samples per second, even though it ignores most of the data. A more efficient alternative is to use a channel pipe list, which is a list of input channel pipe numbers in parentheses. A channel pipe list specifies a list of input channel pipes from which an input task reads data, as illustrated by the first HI GH task definition below. See Figure 25 for a diagram of this application.

```

RESET
PIPE P1, P2, P3, P4, P5
IDEF A 10
  SET IP0 S0
  SET IP1 S1
  SET IP2 S0
  SET IP3 S2
  SET IP4 S0
  SET IP5 S3
  SET IP6 S0
  SET IP7 S4
  SET IP8 S0
  SET IP9 S5
  TIME 50
END
PDEF B
  HI GH (IP(0, 2, 4, 6, 8), 100, P1)
  HI GH (IP1, 100, P2)
  HI GH (IP3, 100, P3)
  HI GH (IP5, 100, P4)
  HI GH (IP7, 100, P5)
END

```

This input procedure samples ten input pins at a rate of 2,000 samples per second per input pin. The input pin sampling sequence is:

S0, S1, S0, S2, S0, S3, S0, S4, S0, S5, S0, S1,

Since the input procedure samples each input pin at 2,000 samples per second, the slow input channel pipes can be read directly by tasks. The SET commands in the input procedure set the input channel pipes so that the fast input pin is sampled at every second sample interval. The first HI GH task then reads from five of the input channel pipes to achieve an aggregate data rate of 10,000 samples per second.

Notice that the sampling rate of this input procedure is 2.5 times slower than the sampling rate in the previous input procedure, but with the same final data rates. This

allows more time for processing because the processor is not involved in reducing the data rate.

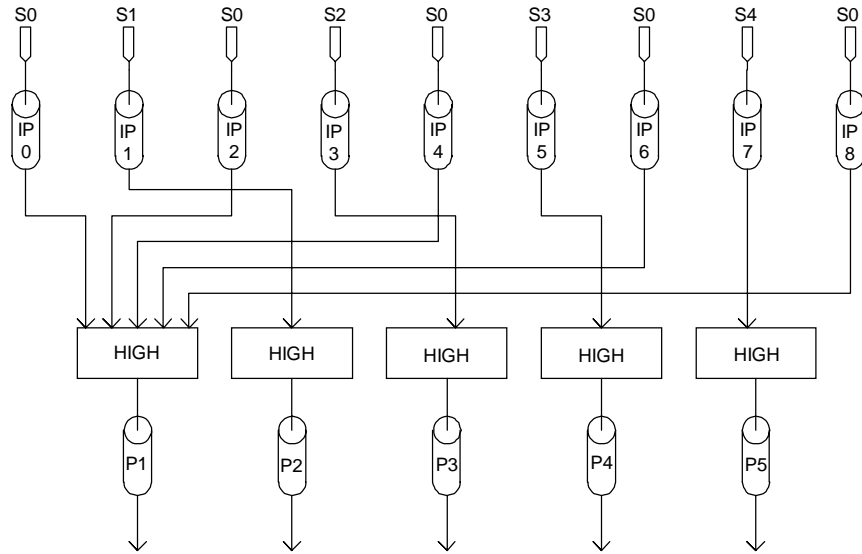


Figure 25. Mixing Fast Inputs and Slow Inputs

Application 35 — Multiple Rate Data Transfer

In some applications it is desirable to simultaneously transfer fast and slow input channel pipe data to a PC. A MERGEF task can merge pipes asynchronously.

This application uses MERGEF to combine pipes with different data rates into one pipe. MERGEF places an identifying flag before each value to identify the source of each data value. See Figure 26.

```
RESET
PIPES P1, P2, P3
IDEF A 4
  SET IP0 S0
  SET IP1 S1
  SET IP2 S2
  TIME 250
END
PDEF B
  AVERAGE (IP0, 10, P1)
  AVERAGE (IP1, 20, P2)
  AVERAGE (IP2, 50, P3)
  MERGEF (P1, P2, P3, $BINOUT)
END
START A, B
```

This DAPL command list defines three input pins sampled at 1000 samples per second. Four logical input channel pipes are specified in the input procedure definition to get the desired sample rate for each input pin. The fourth input channel pipe is ignored. The first three input channel pipes are averaged with different block sizes, so the data rates are different for each. The data rates in pipes P1, P2, and P3 are 100, 50, and 20 samples per second, respectively.

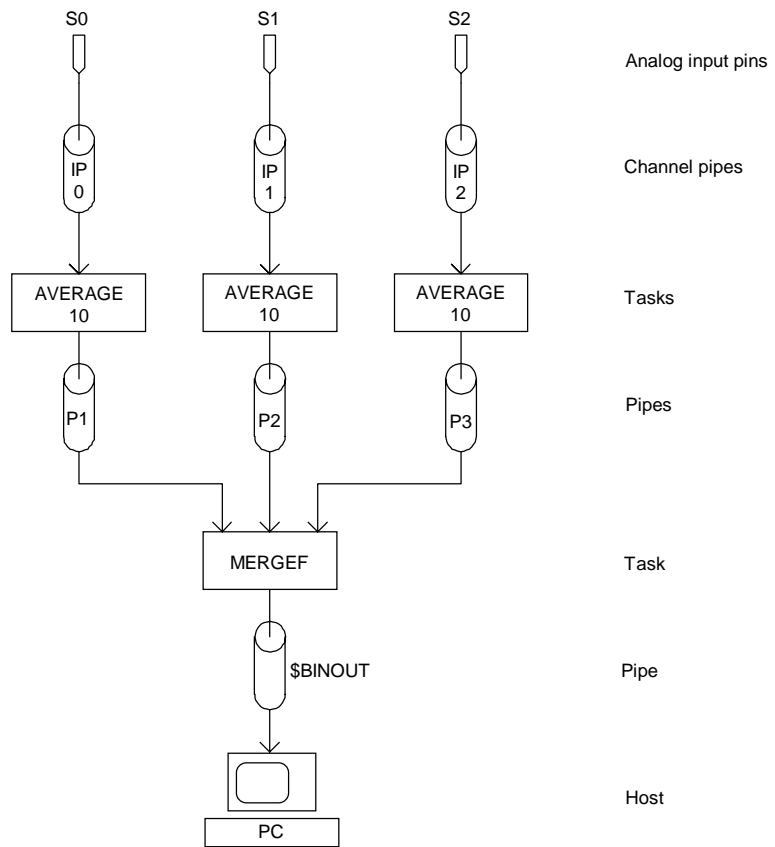


Figure 26. Multiple Rate Data Transfer

Application 36 — Observing Timing of Rotating Machinery

When combined with a Microstar Laboratories Counter Timer Board, a Data Acquisition Processor other than the DAP 800 can provide precise time resolution for repetitive events. This application shows how the combination of a Data Acquisition Processor and a Counter Timer Board can be used to generate an engine speed profile. This is typical of a variety of applications involving rotating machinery.

Figure 27 illustrates the hardware configuration for this application. An optical encoder has a disk which is attached to the engine's flywheel. The disk has two optical sensors. One passes light through a series of uniformly spaced slits to generate clock pulses; the other passes light through a single slit to generate a reference pulse. The output of the first sensor is connected to the external clock input of the Data Acquisition Processor. In a typical application, the optical encoder might have 256 slits, giving clock pulses every $360/256 = 1.41$ degrees. The output of the second sensor is connected to the hardware trigger of the Data Acquisition Processor to give absolute positional information.

Note: The optical encoder configuration used in this application also is useful in many applications involving fast Fourier transforms. In typical applications, the techniques of this application are combined with the spectral analysis techniques described in Chapter 8.

The Counter Timer Board is connected to the digital input/output port of the Data Acquisition Processor. By means of software commands, the Counter Timer Board can be configured either to count external pulses or to count cycles of its internal 5-MHz clock. With its 5-MHz clock, the Counter Timer Board gives a time resolution of 200 nanoseconds.

In this application, the Counter Timer Board is configured to count the number of internal clock cycles which occur between pulses from the optical encoder. At an engine speed of 600 RPM, for example, there are 1953 cycles of the 5 MHz clock between pulses of the optical encoder. This means that over the period of one encoder clock, the Counter Timer Board can resolve speed variations of about $1/1953 = 0.05\%$ of the engine speed. For greater resolution, the times can be averaged. The following DAPL listing illustrates a typical engine monitoring application.

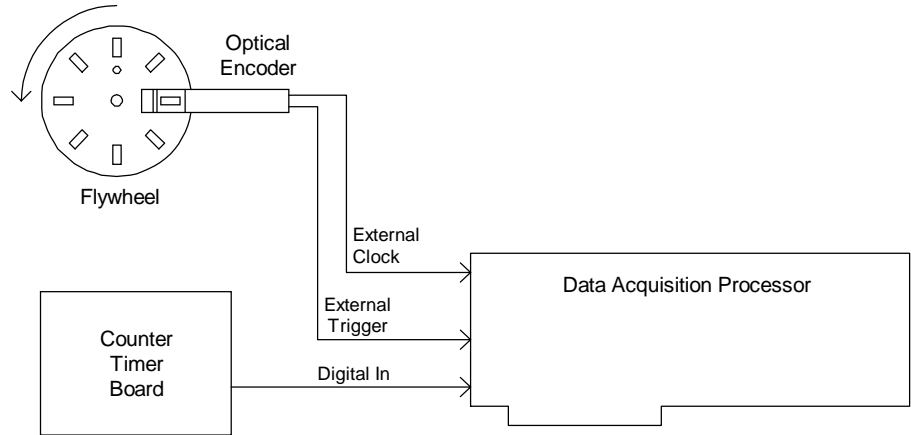


Figure 27. Engine Diagnostics Configuration

```
OUTPUT 0..7 TYPE=0
```

```
RESET
PIPES P1, P2
CONST INTERNAL=10
VARIABLE SCALE1=500000 LONG
I DEF A 2
  CLOCK EXTERNAL
  HTRIGGER ONESHOT
  SET IPO CTLO
  SET IP1 CTO INTERNAL
  TIME 100
END
PDEF B
  CTRATE (IP1, P1)
  BAVERAGE (P1, 256, 100, P2)
  $BINOUT = SCALE1 / P2
END
START A, B
```

The Counter Timer Board appears to the Data Acquisition Processor as several input/output ports. Because of this, the OUTPUT command is required for communication with the Counter Timer Board.

The clock output of the optical encoder is connected to the External Clock Input pin of the Data Acquisition Processor. The CLOCK EXTERNAL command configures the Data Acquisition Processor to use its external clock. The TIME command must set the

internal sampling time smaller than the shortest possible time between pulses of the flywheel encoder, and it should set the sampling time close to the shortest possible time between pulses of the flywheel encoder. See the Data Acquisition Processor hardware manual for more information about selecting the parameter for the TIME command.

The Data Acquisition Processor should be configured for Channel List Clocking. With Channel List Clocking, the Data Acquisition Processor reads all the input pins of its active input procedure each time it receives an external clock input. Channel List Clocking is a jumper option for the Data Acquisition Processor. See the Data Acquisition Processor hardware manual for details.

The reference pulse from the optical encoder is connected to the External Trigger pin on the Data Acquisition Processor. The HTRIGGER ONESHOT command synchronizes the start of sampling to a pulse at the External Trigger pin, giving absolute positional information. Sampling continues until the Data Acquisition Processor receives a STOP command.

Two forms of the SET command are dedicated to the Counter Timer Board. Each Counter Timer Board has two groups of five counters. Counters 0 through 4 form group 0, and counters 5 through 9 form group 1. A SET IPIPEX CTLY command sets up the Counter Timer Board to load all the counters in group 'y' each time the Data Acquisition Processor reads IPIPEX. A SET IPIPEW CTZ command sets up the Counter Timer Board to read counter z each time the Data Acquisition Processor reads IPIPEW. If this command is followed by the code 10, the input sample is clocked by the internal 5 MHz clock; otherwise, it is clocked externally.

Loading a group of counters takes one sample time. The load command for a group should be set at an input number which is lower than the input numbers for reading the inputs of the group.

In this application, the Data Acquisition Processor loads counter group 0 and then reads counter 0 each time the optical encoder generates a clock pulse. The CTRATE task definition command sets up an input task which calculates the differences between adjacent count values in input channel pipe 1 and places the differences in pipe P1. The speed of the flywheel is calculated easily from the differences.

It may be desirable to reduce the finite sampling time effects by averaging the speeds at corresponding points in many cycles. A BAVERAGE task in processing procedure B averages corresponding points in 100 blocks and puts the results in pipe P2. The block size for the BAVERAGE task is the same as the number of pulses from the flywheel encoder in a complete cycle of the engine.

Speed is calculated by a DAPL Expression, which places the result in \$BI NOUT.

8. Digital Signal Processing

Application 37 — Digital Filtering

Digital filtering provides many advantages over traditional analog filtering. This application implements a digital bandpass filter which attenuates input frequencies which are less than 10 Hz or greater than 15 Hz.

Each digital filtering task requires a vector of filter coefficients. This vector can be generated by the FGEN program.

```
RESET
VECTOR FV = (-19, 83, 246, -583, -1131,
             1871, 2722, -3553, -4217, 4586, 4586,
             -4217, -3553, 2722, 1871, -1131, -583,
             246, 83, -19)
I DEF A 1
  SET I PIPE0 S0
  TIME 10000
  END
PDEF B
  FIRFILTER (I PIPE0, FV, 0, 0, 0, 0, $BI NOUT)
  END
START A, B
```

In this command list, the VECTOR command defines a vector which specifies the frequency response of the filter. The FIRFILTER task filters the data from input channel pipe 0 and sends the filtered data to \$BI NOUT.

Note: FIRFILTER is available only in DAPL 2000. For previous versions of DAPL, use the RFILTER command.

Application 38 — Spectral Analysis

An important data processing operation performed by the Data Acquisition Processor is spectral analysis — computation of the frequency components of signals. Chapter 17 of the DAPL Manual provides background on spectral analysis using fast Fourier transforms.

The following commands configure the Data Acquisition Processor to compute fast Fourier transforms on successive blocks of 256 data values from input channel pipe 0. This application is illustrated in Figure 28. After computing the transform of each block of data, the Data Acquisition Processor prints the amplitude spectrum of the data.

```
RESET
I DEF A 1
  SET I PIPE 0
  TIME 10000
END
PDEF B
  FFT (5, 8, 0, I P0, $BI NOUT)
END
START A, B
```

The FFT task definition command is all that the Data Acquisition Processor requires to compute a fast Fourier transform. See Chapter 17 of the DAPL Manual for an interpretation of the fast Fourier transform.

DAPview Note: The recommended graphing options for spectral data are Accumulate X option off, Number of Points equal to $\langle \text{fft_size} \rangle / 2$, and X Unit Scaling equal to $\langle \text{sampling_frequency} \rangle / \langle \text{fft_size} \rangle$. For example, the previous application should be configured with Number of Points equal to 128 and X Unit Scaling equal to $100/256 = 0.391$.

The fast Fourier transform requires the input sampling speed to be at least twice the highest frequency in the input signal. This may require very fast sampling of an input pin. Fast sampling will, however, overload the PC with transform data. A SKIP command provides options for extracting blocks of data from an input channel pipe and ignoring fractions of each block:

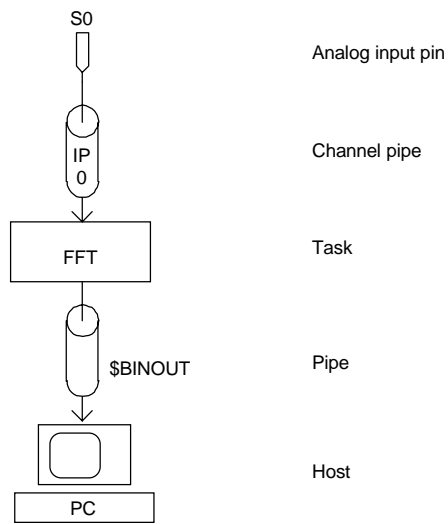


Figure 28. Spectral Analysis

```

I DEF A 1
  SET IPO S0
  TIME 100
  END
PDEF B
  SKIP (IPO, 0, 256, 768, P1)
  END

```

The SKIP task transfers a block of 256 data values and then ignores 768 data values — three blocks of 256 data values. This task reduces the data rate to FFT by a factor of four without slowing down the acquisition rate.

Some applications of spectral analysis require only the predominant input frequency and the size of the component at that frequency. A FINDMAX task can be used to find the dominant frequency.

```

RESET
PIPES P1, P2, P3
I DEF A 1
  SET IPO S0
  TIME 10000
  END
PDEF B
  FFT (5, 8, 0, IPO, P1)
  FINDMAX (P1, 128, INSIDE, 1, 127, P2, P3)
  MERGE (P2, P3, $BI NOUT)
  END
START A, B

```

The FINDMAX command allows optional restriction of searching to a portion of each block of data. In the FINDMAX task definition command above, the region (INSIDE, 1, 127) specifies that the FINDMAX task ignores the first frequency component. This component represents the DC offset of the input signal.

The MERGE command sends data to \$BI NOUT, and the PC displays the amplitude and the position of the dominant frequency component.

Application 39 — Calculating Transfer Functions

A Data Acquisition Processor with an on-board DSP chip is well-suited for frequency response testing. A device is driven by an input signal, and the input and output signals are digitized. The transfer function of the device under test is defined as the frequency domain ratio of the output to the input.

A RANDOM task can be used with an output procedure to drive a device under test with a broadband noise signal. Together with the techniques described in this application, this makes the Data Acquisition Processor into a complete transfer function tester. The output of the RANDOM task should be band-limited by a DAPL lowpass filter and a hardware anti-aliasing filter. Details of this setup are not included in this application.

The straightforward approach to calculating the transfer function is to calculate the fast Fourier transforms of the input signal and the output signal, and then to form the ratio of the output to the input at each frequency. The disadvantage of this approach is that it does not behave well under averaging, so it is sensitive to noise at those frequencies where the input signal does not have much power. An alternative involves calculating the crosspower spectrum and the autopower spectrum. With this approach, averaging is well behaved. The two command lists in this application show both approaches to calculating the transfer function.

If input and output are not sampled simultaneously, the calculated transfer function shows a phase error which is proportional to frequency. This error can be removed by a computation, or can be eliminated by using a Simultaneous Sampling Board. The command lists in this application use a Simultaneous Sampling Board.

The following command list calculates the transfer function directly using fast Fourier transforms.

```

RESET
PIPE S P1, P2, P3, P4, P5, P6, P7
IDEF A 3
  SET IP0 S256
  SET IP1 S0
  SET IP2 S1
  TIME 1000
END
PDEF B
  FFT (0, 10, 0, IP1, P4, P5)
  FFT (0, 10, 0, IP2, P6, P7)
  TFUNCTION1 (P4, P5, P6, P7, 100, P1, P2)
  P3 = (P2 * 180) / 32767
  MERGE (P1, P3, $BINOUT)
END
START A, B

```

In this application, single-ended inputs S0 and S1 are connected to the input and the output of the device under test, respectively. Inputs pins S0 and S1 are sampled simultaneously. Input channel pipe 0 is a dummy channel pipe which sets up the Simultaneous Sampling Board for sampling. See the Hardware Manual for details.

The fast Fourier transforms of the input and the output are calculated by two FFT tasks. These are combined in the TFUNCTION1 task to give the transfer function. The DAPL expression converts the phase output of TFUNCTION1 to degrees. Finally, the MERGE task sends the results back to the host PC through \$BINOUT.

The following command list calculates the transfer function using crosspower spectrum:

```

RESET
CONST NAVE=16
PIPES P1 LONG, P2 LONG, P3 LONG
PIPES P4 LONG, P5 LONG, P6 LONG, P7, P8, P9
I DEF A 3
    SET IPO S256
    SET IP1 S0
    SET IP2 S1
    TIME 1000
    END
PDEF B
    CROSSPOWER (IP1, IP2, 0, 10, 0, P1, P2, P3)
    BAVERAGE (P1, 512, NAVE, P4)
    BAVERAGE (P2, 512, NAVE, P5)
    BAVERAGE (P3, 512, NAVE, P6)
    TFUNCTION2 (P6, P4, P5, 100, P7, P8)
    P9 = (P8 * 180) / 32767
    MERGE (P7, P9, $BINOUT)
    END
START A, B

```

This command list uses a CROSSPOWER task in place of two FFT tasks. This allows block averaging, as performed by the three BAVERAGE tasks, before calculation of the transfer function. A TFUNCTION2 task must be used with the data from the CROSSPOWER task.

9. Process Control

Application 40 — Alarms

The following application sets an alarm bit on the digital output port when the value of differential input channel pipe 0 exceeds 999. Low latency is used to minimize the delay from input to alarm. See Figure 29 for a diagram of this application.

```
RESET
  OPTION SCHEDULING=FIXED, QUANTUM=200, BUFFERING=OFF
  IDEF A 1
    SET IPO DO
    TIME 1000
  END
  PDEF B
    ALARM (IPO, INSIDE, 1000, 32767, 7, 0)
  END
START A, B
```

The ALARM command sets digital output bit 7 if the value in input channel pipe 0 is in the range 1000...32767.

This application typically responds to an alarm condition within 2 milliseconds.

Note: OPTION SCHEDULING=FIXED, QUANTUM=200, BUFFERING=OFF is a DAPL 2000 command. For DAPL 4.x, use OPTION LLATENCY=ON.

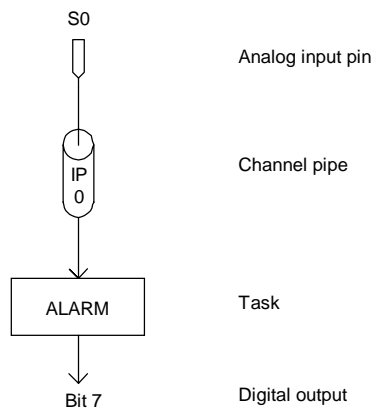


Figure 29. Rapid Response to Alarm Conditions

Application 41 — PID Control

Process control applications require feedback from input to output. A chemical reactor, for example, may need a temperature controller which maintains the reactor's internal temperature at a specified setpoint. One simple approach to this sort of temperature control is to use the DAPL ALARM command to turn a heater on when the temperature drops below the setpoint. This approach, however, can result in temperature overshoot.

A common control algorithm is the proportional integral derivative (PID) algorithm. A PID control loop drives an analog input toward a specified value, known as a setpoint, by generating an output signal which corrects for differences between the analog input and the setpoint. The PID algorithm combines three error signals which are proportional to the observed differences, their integrals, and their derivatives. By careful selection of the proportionality constants, it is possible to take system time lag and overshoot into consideration when defining a PID task.

One Data Acquisition Processor can control a large number of PID loops, so the Data Acquisition Processor can be an inexpensive PID controller. Sampling rates and acceptable latencies determine the number of PID loops one Data Acquisition Processor can control.

The following commands set up a typical PID application:

```
RESET
PIPES P1, P2
VARIABLES SETP=1000, P=-100, I=-50, D=-50
I DEF A 1
    SET IPO SO 10
    TIME 10000
    END
PDEF B
    AVERAGE (IPO, 100, P1)
    PID (P1, SETP, P, 1000, I, 1000, D, 1000, P2)
    DACOUT (P2, 0)
    END
START A, B
```

The PID task reads input data from pipe P1 and places correction data in pipe P2. If suitable parameters are selected, the PID task maintains the input data at the value of

variable SETP. See the DAPL Manual for more information about PID and its parameters.

Finally, the DACOUT task reads the output of the PID task and sends the data to analog output #0.

A LET command can be used to change the values of the SETP, P, I, or D variables at any time during execution of this application.

A low-latency mode should be used to reduce the latency between input and control output. The following commands set up a PID loop for minimum delay:

```
RESET
OPTI ONS SCHEDULI NG=FI XED, QUANTUM=200, BUFFERI NG=OFF
PI PES P1
VARI ABLES SETP=1000, P=-100, I=-50, D=-50
I DEF A 1
    SET I P0 S0
    TIME 10000
    END
PDEF B
    PID (I P0, SETP, P, 1000, I , 1000, D, 1000, P1)
    DACOUT (P1, 0)
    END
START A, B
```

This DAPL command list configures the Data Acquisition Processor for low latency with the OPTI ONS SCHEDULI NG=FI XED, QUANTUM=200, BUFFERI NG=OFF command. Low latency mode limits time delays to approximately 1 ms for each task, or 0.2 ms for a DAP 3000a or DAP 3200a. In this case the typical time between input and output is 2 ms, or 0.4 ms for a DAP 3000a or DAP 3200a.

Note: OPTI ONS settings are different between DAPL 2000 and previous versions of DAPL. Check the manual for the relevant version of DAPL for correct OPTI ONS settings. For example, in DAPL 4.4, use OPTI ONS LLATENCY = ON instead of the above syntax.

Application 42 — Pulse Width Modulation

Some applications require only digital (on/off) control, rather than analog control. A heater, for example, may be cycled on and off with a variable duty cycle.

The PWM command converts continuous control data into variable width control pulses; PWM stands for Pulse Width Modulation. The following commands assume that the heater in the previous application is controlled by bit 0 of the binary output port.

```
RESET
PIPES P1, P2
VARIABLES SETP=1000, P=-100, I=-50, D=-50
IDEF A 1
    SET IPIPEO S0 10
    TIME 10000
END
PDEF B
    PID (IPIPEO, SETP, P, 1000, I, 1000, D, 1000, P1)
    PWM (P1, -5000, 5000, 10, 0, 0, OPIPEO)
END
ODEFINE C 1
    SET OPIPEO B
    TIME 10000
END
START A, B, C
```

For the DAP 3200a, the output procedure should include the following command to ensure that enough values are buffered prior to starting the output procedure, which avoids output underflow:

```
OUTPUTWAIT 50
```

The PWM task reads 10 input values from pipe P1 and sends 10 output values to output channel pipe 0. PWM turns bit 0 on for zero to ten of the output values, depending on the average of the input data. If the average is less than -5000, bit 0 is turned off for all ten output values. If the average is above +5000 bit 0 is turned on for all ten output values. If the average is between -5000 and +5000, bit 0 is turned on for between zero and ten values. The output procedure sends PWM data to the digital output port.

The output procedure TIME command defines an update time of 10 milliseconds. The PWM command provides 10 values for each interval to cycle the heater on and off at

100 millisecond intervals. Within each interval, the output bit is high for between zero and 100 milliseconds, with 10 millisecond resolution.

Heater Controller

The following DAPL commands build on the previous example to implement a complete heater control application with a thermocouple for sensing temperature.

```
RESET
PI PES P0, P1, P2
VARIABLES SETP=450, P=-100, I=-50, D=-50
VARIABLES REF=200, GRND
I DEF A 2
  SET IPO G 100
  SET IP1 SO 100
  TIME 5000
END
PDEF B
  AVERAGE (IPO, 100, PO)
  PVALUE (PO, GRND)
  THERMO (IP1, 1, 5000, 32767, GRND, P1, REF)
  PID (P1, SETP, P, 1000, I, 1000, D, 1000, P2)
  PWM (P2, -5000, 5000, 10, 0, 0, OPO)
END
ODEFINE C 1
  SET OPO B
  TIME 10000
END
START A, B, C
```

For the DAP 3200a, the output procedure should include the following command to ensure that enough values are buffered prior to starting the output procedure, which avoids output underflow:

```
OUTPUTWAIT 50
```

In this example, a THERMO command converts thermocouple input voltages to tenths of degrees Celsius. A PID command calculates a control signal from the temperature input. Variable SETP sets the control temperature to 45.0 degrees Celsius. A PWM command reads the control signal from the PID command and sends a pulse sequence to an output procedure.

The output procedure TIME command defines an update time of 10 milliseconds. The PWM command provides 10 values for each interval to cycle the heater on and off at 100 millisecond intervals. Within each interval, the output bit is high for between zero and 100 milliseconds, with 10 millisecond resolution.

10. Communications

Application 43 — Text Communication

There are two options for sending data from a Data Acquisition Processor to the PC: binary or text transfer. Applications requiring high communication speed can transfer data in binary format. This improves efficiency by eliminating the conversions between binary format and text format. Binary transfer, however, limits data formatting options and may require more complex processing in the PC.

Tasks defined by DAPL PRINT and FORMAT commands convert binary data into formatted text and then transmit the text to the PC. Text supports many data formatting options and provides the most flexible data transmission. However, the conversions to and from text require extra processing on both the Data Acquisition Processor and its host PC.

The following command list configures the Data Acquisition Processor to transfer all data to the PC as text:

```
RESET
I DEF A 3
  SET I PIPE0 S0
  SET I PIPE1 S1
  SET I PIPE2 S2
  TIME 10000
  END
PDEF B
  PRINT
  END
START A, B
```

The PRINT task repeatedly reads one value from each of the three input channel pipes and sends the values to the PC as a line of text. The PRINT command continues until sampling is stopped.

The FORMAT task also sends text data to the PC. FORMAT is a powerful command for formatting data from processing procedures. Below is a simple example in which the FORMAT command receives data from DELTA and sends the data to the PC:

```
RESET
PIPE PO
IDEF A 1
  SET IPO SO
  TIME 100
END
PDEF B
  DELTA (IPO, PO)
  FORMAT (PO)
END
START A, B
```

The FORMAT command can accept up to 32 parameters. Some examples of formatting options are: printing line numbers, including a decimal specification, and printing the results of variables. See the DAPL manual for more details.

Application 44 — Text Communication from Several Tasks

When a PC receives data from more than one task, a FORMAT task is a good way to distinguish which data value came from which task. Multiple values can be printed with a FORMAT task, and each value will print on one line in the order listed.

```
RESET
PIPES P1, P2
IDEF A 2
  SET IPO S0
  SET IP1 S1
  TIME 100
END
PDEF B
  AVERAGE (IPO, 100, P1)
  AVERAGE (IP1, 100, P2)
  FORMAT (P1, P2)
END
START A, B
```

In the above example, FORMAT reads data values sequentially from pipes P1 and P2 and sends the values to the PC. The data rates into all input pipes must be the same for this FORMAT task to work.

Using two FORMAT tasks allows data to be sent to the PC from two tasks with different data rates. Each value is printed on a separate line. You can print an identifying prefix in front of each format line to identify which task each value is from.

```
RESET
PIPES P1, P2
TRIGGERS T1, T2
IDEF A 2
  SET IPO S0
  SET IP1 S1
  TIME 100
END
PDEF B
  DLIMIT (IPO, INSIDE, 100, 200, T1)
  DLIMIT (IP1, INSIDE, 100, 200, T2)
  TSTAMP (T1, P1)
  TSTAMP (T2, P2)
  FORMAT ("One: ", P1)
  FORMAT ("Two: ", P2)
END
START A, B
```

Application 45 — Simultaneous Transfer of Text Data and Binary Data

This application illustrates simultaneous binary and text data transfer to a PC. A binary com pipe is used for data which must be transferred at a high data rate, and a text com pipe is used for data which will be transferred at a low data rate.

Software triggering is used in this application to transfer a large block of data to a PC whenever an input value exceeds 10,000. Because the blocks are large, they are transferred through a binary com pipe. A TSTAMP task returns the sample count of each software trigger. Because there are relatively few sample count values, the results are transferred to the PC through a text com pipe. This application is illustrated in Figure 30.

```
RESET
PIPE P1 LONG
TRIGGER T1 2
IDEF A 5
  SET IP0 S0
  SET IP1 S1
  SET IP2 S2
  SET IP3 S3
  SET IP4 S4
  TIME 10
  END
PDEF B
LIMIT (IP0, INSIDE, 10000, 32767, T1, INSIDE, 10000, 32767)
WAIT (IP(0, 1, 2, 3, 4), T1, 100, 400, $BINOUT)
TSTAMP (T1, P1)
FORMAT (P1)
END
START A, B
```

This DAPL command list selects five input channel pipes for sampling at 20 KHz. A LIMIT task scans input channel pipe 0 and sets trigger T1 when a value above 10,000 is detected. Hysteresis is used to prevent generation of additional triggers until after a value below 10,000 is detected. A valid trigger in T1 causes a WAIT task to transfer 100 pre-trigger values and 400 post-trigger values from five input channel pipes to the binary com pipe \$BINOUT.

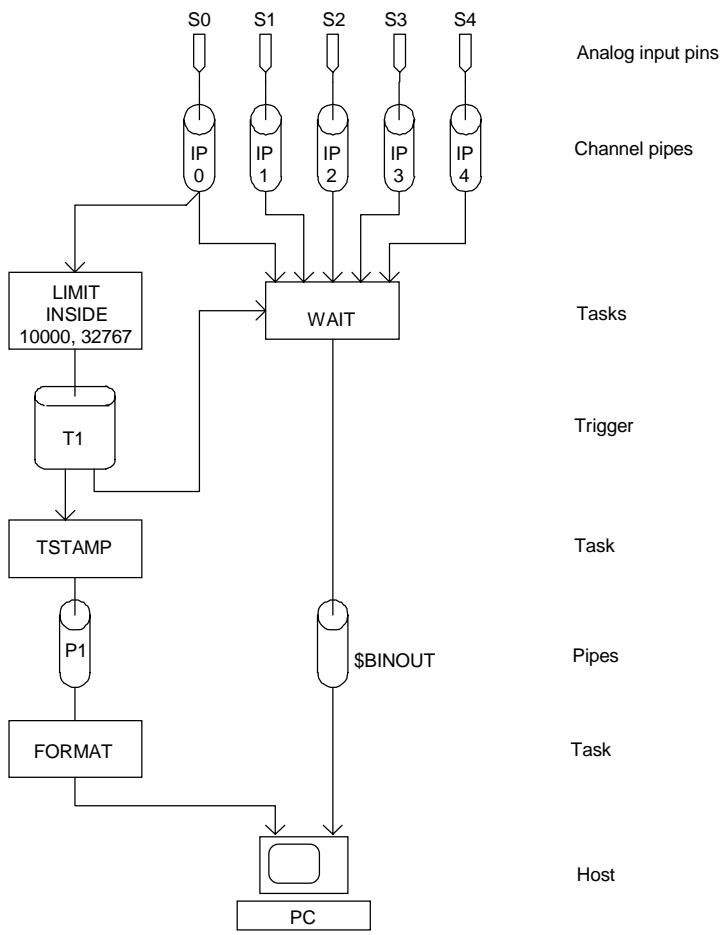


Figure 30. Text and Binary Transfer

Each time trigger T1 is asserted, a TSTAMP task places the sample count of the trigger event into long pipe P1. A FORMAT task transfers the data in P1 to the PC through a text com pipe.

DAPview Note: The recommended DAPview data settings for this application are Screen Data Select menu to Text and the logging Data Select menu to Binary. The graphics Data Select menu should be set to Text, unless real-time graphics are required. DAPview Plus applications should turn autoconfiguration off, to prevent automatic resetting of above options.

A high level language such as Pascal or BASIC also could be used to read the data from both text and binary communications pipes. The high level language should select com pipe 0 to read text data and com pipe 1 to read binary data.

Application 46 — Sending Data to a Data Acquisition Processor

A Data Acquisition Processor supports high-speed binary data transfer to and from a host PC. Receiving data from the PC is similar to sending data to the PC. This application provides two examples illustrating PC to Data Acquisition Processor binary transfer. The first example shows how to transfer previously logged data to a Data Acquisition Processor for analog output. The second example illustrates the use of a Data Acquisition Processor for postprocessing.

A Data Acquisition Processor reads data from the PC through the \$BININ communication pipe. Any task can use \$BININ as its input pipe. The following DAPL commands illustrate reading data from the PC and sending the data to output channel pipe 0.

```
RESET
PDEF A
  COPY ($BININ, OPIPEO)
  END
ODEF B 1
  SET OPIPEO AO
  TIME 10000
  END
START A, B
```

A COPY command reads data from \$BININ and sends the data to output channel pipe 0. An output procedure defines output channel pipe 0 for analog output on AO at 100 Hz.

DAPview Plus can transfer data from a file to the Data Acquisition Processor using the Data file option in the Control menu. Programs created with programming languages also can transfer data to a Data Acquisition Processor.

A Data Acquisition Processor also can process data from the PC and send the results back to the PC. The following example illustrates how a Data Acquisition Processor with an on-board DSP chip accelerates FFT processing of PC data.

```
RESET
PDEF A
  FFT (5, 8, 1, $BININ, $BINOUT)
  END
START A
```

An FFT command reads data from \$BI NI N, processes the data, and sends the data back to the PC through \$BI NOUT.

Note that the DAP 800 has fast data transfer only while sending data to the PC. When receiving data from the PC, the maximum data rate is limited to about 5 KHz. For this reason, it is best to use a different model for applications that require high-speed transfer from a PC to a Data Acquisition Processor.

Application 47 — Synchronizing Several Data Acquisition Processors

Several Data Acquisition Processors can be used in one PC to provide performance that would otherwise be impossible. This application uses two Data Acquisition Processors to provide an aggregate sample rate of 400 KHz. In addition, two Data Acquisition Processors provide twice the processing power of one board. In this application, one input on each board is sampled at 200 KHz for an aggregate sample rate of 400 KHz. Each Data Acquisition Processor performs FFT processing on a reduced set of data from each input channel pipe.

To achieve simultaneous synchronized sampling, one Data Acquisition Processor must be configured as a master and the other Data Acquisition Processor must be configured as a slave. See the Data Acquisition Processor hardware manual for instructions on installing several synchronous Data Acquisition Processors.

When several Data Acquisition Processors are installed in one PC, the Data Acquisition Processors communicate with the PC through different com pipes. In this application, the slave Data Acquisition Processor communicates with the PC through com pipes 0 and 1, and the master Data Acquisition Processor communicates with the PC through com pipes 2 and 3.

A diagram of this application is provided in Figure 31.

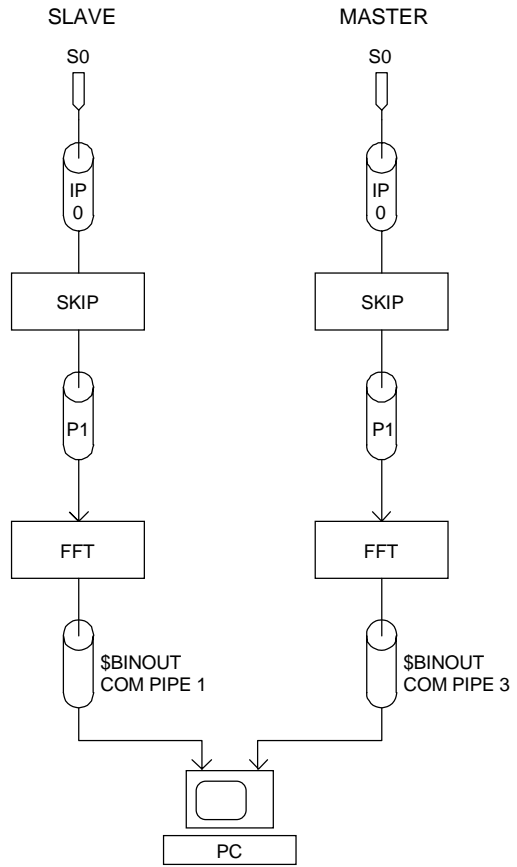


Figure 31. Synchronous Data Acquisition Processors

The following DAPL command list configures the master unit.

```
RESET
PIPE P1
IDEF AMASTER 1
  MASTER
  SET IPO SO
  TIME 5
  END
PDEF BMASTER
  SKIP (IPO, 0, 1024, 9216, P1)
  FFT (5, 10, 2, P1, $BINOUT)
  END
```

The following DAPL command list configures the slave unit.

```
RESET
PIPE P1
IDEF ASLAVE 1
  SLAVE
  SET IPO SO
  TIME 5
  END
PDEF BSLAVE
  SKIP (IPO, 0, 1024, 9216, P1)
  FFT (5, 10, 2, P1, $BINOUT)
  END
```

The DAPL command list for the master unit defines input procedure AMASTER to read from a single-ended analog input at 200,000 samples per second. The MASTER command defines the Data Acquisition Processor to be the master that supplies the sampling clock for all of the slave Data Acquisition Processors. A SKIP command transfers one block of 1024 values out of every ten blocks of input data to an FFT command. This reduces the data rate for the FFT but provides high-speed data within blocks. For Data Acquisition Processors without on-board DSP chips, further data reduction is required to compensate for the FFT processing speed. The FFT task performs high-speed FFT processing and the results are sent to the PC through the \$BINOUT com pipe.

In this application, the DAPL commands for the slave are nearly the same as the commands for the master. Note, however, that the commands are not required to be the same. Only the input procedure TIME commands must match between the master and slave. The SLAVE command sets the input procedure for slave operation. This forces the input sampling clock to come from the master unit. The inputs of the slave are sampled simultaneously with the inputs of the master. A processing procedure defines the same processing as the master. FFT values are sent to the PC through the \$BINOUT com pipe. The blocks of FFT data are synchronized with FFT data from the master unit.

Communication with several Data Acquisition Processors can be handled with DLOG or with a high level language such as C or BASIC. This application uses a high level language for communications.

To configure the master unit, a high level language program transfers the master DAPL commands to the master Data Acquisition Processor through com pipe 2. After the master unit is configured, the DAPL commands for the slave unit are sent through com pipe 0.

The input procedure of the slave unit must be started before that of the master unit. With the high level language, the following command is sent to the slave unit through com pipe 0.

```
START ASLAVE, BSLAVE
```

The following command is sent through com pipe 2 to start the master unit which causes both the master and slave to begin sampling.

```
START AMASTER, BMASTER
```

The high level language receives binary data from the slave unit and the master unit by reading from com pipe 1 and 3 respectively. When data acquisition is complete, the master unit should be stopped before the slave unit is stopped.

DLOG also can be used to communicate with several synchronous Data Acquisition Processors. DLOG starts each Data Acquisition Processor in the order they are numbered. The master unit is assumed to be the last Data Acquisition Processor. The I/O addresses of each Data Acquisition Processor should be configured so that the master unit is last.

Application 48 — Serial Communication

The DAP 801 has an onboard serial port that can communicate with external peripherals. This model also can operate in stand-alone mode. This application illustrates communication with an external peripheral, logging data summaries to an external serial printer. A hardcopy print is particularly useful for data reports and as a backup against accidental data loss.

```
RESET
PIPES P1, P2, P3
IDEF A 1
  SET IPO S0
  TIME 10000
  END
PDEF B
  HIGH (IPO, 100, P1)
  LOW (IPO, 100, P2)
  AVERAGE (IPO, 100, P3)
  FORMAT (IPO)
  FORMAT OUTPUT=$SEROUT (P1, P2, P3)
  END
START A, B
```

The first FORMAT command formats the raw data and sends the data to the PC. The second FORMAT command formats the high, low, and average values of blocks of 100 data values and sends the results to an external printer connected to the Data Acquisition Processor serial port. The printer generates a summary line once every second.

Com pipes are special pipes used for controlling DAPL communications. The OUTPUT= specification in the FORMAT command redirects the formatted output from the default text com pipe, \$SYSOUT, to the serial com pipe, \$SEROUT. When DAPL is initialized, six text com pipes are defined:

| | |
|----------|--------------------------|
| \$SYSOUT | - system output com pipe |
| \$SYSIN | - system input com pipe |
| \$SEROUT | - serial output com pipe |
| \$SERIN | - serial input com pipe |
| \$PAROUT | - PC output com pipe |
| \$PARIN | - PC input com pipe |

The system com pipes \$SYSIN and \$SYSOUT are assigned to either the serial com pipes or the PC com pipes, depending on the outcome of power-up communication initialization. See the Installation chapter of the Data Acquisition Processor hardware manual for more information about communication initialization.

When a Data Acquisition Processor with a serial port is operated inside a PC, the serial input port \$SERIN is available for reading serial input characters. The Developer's Toolkit for DAPL provides access to \$SERIN.

Index

| | |
|--|----------|
| \$BINOUT | 18, 121 |
| \$SERIN | 135 |
| \$SEROUT | 134, 135 |
| Analog output | 37, 39 |
| Application | |
| Alarms | 113 |
| Almost Simultaneous Sampling | 94 |
| Asynchronous Output | 37 |
| Autoranging | 89 |
| Averaging | 25 |
| Calculating Transfer Functions | 109 |
| DAPL Expressions | 31 |
| Detecting Bit Transitions | 71 |
| Digital Filtering | 105 |
| Digital Input | 23 |
| Digital Oscilloscope | 56 |
| Extracting a Bit from a Digital Input | 32 |
| Finding Deviations between Inputs | 75 |
| Finding Histograms | 34 |
| Generating One-Shot Pulses | 47 |
| Generating Periodic Waveforms | 42 |
| Generating Periodic Waveforms by Copying | 43 |
| Generating Periodic Waveforms by Interpolation | 45 |
| Generating Waveforms | 40 |
| Identifying Maxima and Minima | 92 |
| Interpolation | 87 |
| Mixing Fast Inputs and Slow Inputs | 95 |
| Multiple Rate Data Transfer | 98 |
| Observing Timing of Rotating Machinery | 100 |
| Peak Detection | 27 |
| PID Control | 115 |
| Pulse Width Modulation | 117 |
| Real-Time Data Analysis | 29 |
| Retriggering | 63 |
| Sampling 5000 Values | 21 |
| Sampling Three Inputs | 17 |
| Sending Data to a Data Acquisition Processor | 128 |
| Serial Communication | 134 |
| Simultaneous Transfer of Text Data and Binary Data | 125 |
| Software Triggers | 51 |
| Spectral Analysis | 106 |
| Spike Detection | 65 |
| Synchronizing Several DAPs | 130 |

| | |
|---|--------------|
| Synchronous Output | 39 |
| Text Communication | 121 |
| Text Communication from Several Tasks | 123 |
| Thermocouple Linearization | 77 |
| Time Stamping Pulses | 68 |
| Triggering on Two Conditions | 61 |
| Using Analog Output Expansion | 49 |
| Using Hysteresis | 58 |
| Using Triggers to Calculate Frequency | 73 |
| Applications Overview | 7 |
| Arithmetic Expressions | 31 |
| AUTORANGE | 89 |
| AVERAGE | 29 |
| Basic Real-Time Processing | 8, 25 |
| BAVERAGE | 66, 110 |
| Binary Input | 23 |
| Binary output | 37 |
| BPRINT | 18 |
| BUFFERING | 116 |
| CHANGE | 71 |
| Channel list | 95 |
| Channel pipe list | 95 |
| Communications | 121 |
| Converting Temperatures | 80 |
| COPY | 54 |
| COUNT | 21 |
| CTRATE | 100 |
| CYCLE | 40 |
| DACOUT | 37 |
| DAPview | 18 |
| DELTA | 29, 92 |
| DELTA2 | 75 |
| DEXPAND | 49 |
| Digital output | 37 |
| Digital Signal Processing | 105 |
| DIGITALOUT | 37 |
| DLIMIT | 56 |
| Dummy input channel pipes | 94 |
| END | 18 |
| Fast Fourier Transform | 106 |
| FFT | 106 |
| FILL | 47 |
| FINDMAX | 108 |
| FIRFILTER | 105 |
| FORMAT | 25, 121, 134 |
| Formatting prefixes | 92 |
| FREQUENCY | 73 |

| | |
|---------------------------------------|----------|
| Further Real-Time Processing | 12, 75 |
| Hardware Organization..... | 2 |
| Heater Controller..... | 118 |
| HIGH..... | 27 |
| HTRIGGER..... | 21, 100 |
| IDEFINE | 17 |
| Input | 17 |
| INT | 29 |
| INTERP..... | 87 |
| Introduction..... | 1 |
| IPIPE | 25 |
| LIMIT..... | 51, 65 |
| LOGIC..... | 71 |
| LONG..... | 29, 59 |
| LOW..... | 27 |
| Low latency | 116 |
| MASTER..... | 132 |
| Mathematical Expressions | 31 |
| MERGE..... | 49 |
| MERGEF..... | 98 |
| MINTIME | 66 |
| Modulation | 40 |
| Noise | 25, 54 |
| NTH..... | 66 |
| On-off control..... | 117 |
| OPTION | 116 |
| Output..... | 37 |
| OUTPUTWAIT | 47 |
| PCOUNT..... | 34 |
| PDEFINE | 25 |
| PEAK | 54, 92 |
| Peak Detection..... | 54 |
| PID | 116 |
| PRINT | 17, 121 |
| Process Control | 113, 115 |
| Proportional integral derivative..... | 115 |
| PVALUE..... | 79 |
| PWM | 117 |
| QUANTUM..... | 116 |
| RANGE | 27, 34 |
| RAVERAGE | 39 |
| RESET..... | 17 |
| RETRIGGER..... | 63 |
| Sampling Many Thermocouples..... | 82 |
| Sampling Several Thermocouples | 81 |
| SAWTOOTH..... | 49 |
| SCALE | 29 |

| | |
|--------------------------------------|------------|
| SCHEDULING | 116 |
| SDACOUT | 39, 40 |
| SDIGITALOUT | 39 |
| SDISPLAY..... | 36 |
| Sensing Reference Temperature | 83 |
| Serial port..... | 134 |
| Simultaneous sampling | 109 |
| SINEWAVE..... | 49 |
| Single shot software triggering..... | 60 |
| SLAVE..... | 132 |
| Software Organization..... | 4 |
| Software Triggering | 51 |
| Software Triggers..... | 10 |
| SQRT | 29 |
| SQUAREWAVE..... | 49 |
| START | 19 |
| STOP..... | 20 |
| Synchronous binary input | 23 |
| TFUNCTION1 | 109 |
| TFUNCTION2 | 110 |
| THERMO..... | 77 |
| Thermocouple Accuracy | 79 |
| TIME..... | 17 |
| TOR..... | 61 |
| TRIANGLE..... | 49 |
| Trigger..... | 51 |
| TSTAMP | 68, 125 |
| VARIABLE..... | 34 |
| VECTOR..... | 105 |
| WAIT | 51, 56, 65 |