RMPS A REALTIME PARALLEL COMPUTING SYSTEM

WILLIAM F. TROVER
Associate Director of Advanced Systems
Teledyne Controls
Los Angeles, California

ABSTRACT

Research and Test Activities have a continuing need to cope with more and more channels of data and at continually wider data bandwidths. There is a consensus in the test community that compressed EU and derived parameter data presented in realtime can significantly reduce total test costs because test engineers can make realtime judgements on the validity of a given test point (mode). Classical telemetry preprocessors usually cannot handle these more demanding realtime processing requirements because, when they were designed, it was assumed that raw data was in a single PCM word and only a simple mx+b EU conversion, or simple data compression was required. Present preprocessors typically use special bit slice technology to speed up the realtime process and they're only one or two bus systems whose processing capacity is typically less than 300k to 400k parameters per second. Furthermore, many cannot handle word concatenation (except for adjacent PCM words) and none can handle complex derived parameters such as thrust, lift, gross weight, center of gravity, stall speed, harmonic analysis, etc. To address these limitations, a massively parallel computer system has been developed based on up to sixty, general purpose, 1MFLOP floating point computers operating in parallel to support realtime processing of any type, at aggregate throughputs up to 1.5 Mwps. This system can merge realtime data from up to eight different asynchronous sources having word rates up to 2.0 Mwps from any source. Up to 32,768 different parameters can be accepted as inputs with an additional 32,768 ID tags available for concatenated and derived parameter identification. A powerful realtime software package permits the user of the computer system to apply any, or many algorithms) to any or an parameters being processed.

INTRODUCTION

All aspects of product research and development testing over the last decade have shown a continual increase in the number of measurements required to test or certify a product. This is especially true in aircraft, engine, missile and spacecraft testing. In addition to higher data bandwidths, due to the use of more sensors, there has been an increase in bandwidths per channel to acquire more vibration and acoustic data to assess its impact on

the unit being tested. Digital data bandwidth has been increasing, as more test facilities go away from analog data recording and process all data directly in PCM format.

With the advent of more semiconductor sensors and smart sensors with digital outputs, and the acceptance of digital data buses such as; MIL STD 1553 A/B and ARINC 429, the ratio of data from special pressure/temperature/position/sensors has decreased as compared to data from avionics computers and radars. Now it is not uncommon to have up to 80% of all test data already in digital format when sampled. This has led to sampling of variable or different length data values from different sources and packing them into constant size PCM words such as 8 to 12-bits per word. The result is the need for much more concatenation of multiple-PCM word measurements during realtime processing. Finally, test engineers want to see data in its most meaningful EU form such as thrust, lift, stall speed, center of gravity and spectral analysis of vibration. AR of these parameters represent derived data, which take many EU input terms to compute the result desired by the test engineer.

Most of these complex processes cannot be done in present preprocessors which try to offload from a host computer the I/O choking bandwidths of highspeed raw data and simple EU conversions and compression algorithms. Most present preprocessors use special purpose computers composed of bit/slice logic and microsequencers to obtain the maximum processing speed. These types of processors, however fast, must be micro-coded with specific algorithms. Thus what they can be programmed to do in realtime is limited, though fast, and reprograming is difficult. Even with these special preprocessors however, maximum throughput rates to convert data to EU values is typically less than 200k to 300k words per second. The solution to all of these problems is to develop a system of massively parallel, general purpose floating point computers with enough dedicated memory in each computer so that no disk accesses are required to perform the processing. This is especially true in the light of projected needs to handle from 500k wps to over 1.0 Mwps and to merge multiple asynchronous data sources in realtime.

This parallel computer system must have its hardware design dictated by the software required to process the data and distribute the processed results to a variety of destination (MMIF) devices. The software in turn must account for every type of presently known processing and allow for the future additions of unforeseen processes. This must include both synchronous and periodic processes, as well as asynchronous aperiodic processes. Enough ID tags must be provided to account for concurrent mixing of both raw, EU converted and derived measurements in the same data base. The data base manager must be modular and flexible so that identifier fields and acronyms can be easily changed to account for different user’s terminology and sensor calibration techniques and recertification periods. All processing sequences must be provided with hooks so that users may define, code and install their own algorithms at any point along the total

processing sequence. Finally, an incremental compiler must be provided so the user can make almost any needed change in the program in realtime without disturbing the realtime processing of unaffected parameters.

Teledyne Controls is developing such a computer hardware and software system called Realtime MultiProcessor System [RMPS]. This paper will present an overview of the RMPS architecture and its software, which is the place that many high speed computers have failed in realtime applications.

DESIGN TRADEOFFS AND CONSIDERATIONS

Prior to starting the development of the RMPS, considerable time was spent talking to test ranges and test facilities, trying to understand how existing preprocessors operated and what their limitations are in both data handling speed and in software. The conclusions reached from these studies was that most preprocessors were designed by hardware engineers using the highest speed device technology the state of the art offered. Little, if any, design inputs were given by the software people as most programmability was restricted to micro-code written by the hardware designer. No company appeared to use a general purpose computer, much less one with a floating point co-processor because for one or two CPU board systems, general purpose processors were too slow and/or too costly. Also, all computer hardware designers had been taught for years that memory was costly, so used it sparingly in the design. For these reasons most memories were less than 256kB. Finally, most preprocessor designers seemed to think that a 10MHz bus such as Versabus or Multibus is fast enough for any present and future realtime processing requirements. Thus, even recently introduced preprocessors only have single buses or dual buses even though addresses and data are sometimes treated as separate bus lines on the same bus.

The next analysis undertaken was to determine typical mixes of data types gathered during a test and what are the projected mixes of data (i.e. Analog/Discrete inputs to digital inputs) to be processed in realtime in the forseeable future. This study showed that until the general acceptance of MIL STD-1553 A/B and ARINC 429 avionics buses, most of the test data consisted of analog temperatures, pressures, positions and vibrations measured by temporally installed sensors and sampled by a PCM encoder with a fixed word length of from 8 to 12-bits per word. Only the wind tunnel community seemed to use a significant number of 16-bit ADCs in their test systems.

As avionics and radar systems became more complex, there were more on-board computers in airframes and engines and thus more 16-bit to 32-bit measurements needed to be sampled by the test system and mixed with the 10-bit to 12-bit classical temperatures,

pressures and positions to compare vehicle systems measurement accuracies with special, more accurate test sensors.

Teledyne elected to categorize all test data into three different categories:

1. Data which a complete measurement point can be contained in a single PCM word and converted to EU with a single process.

2. Data where one sample of a complete measurement must be spread across two or more PCM words requiring concatenation prior to processing.

3. Aperiodic data such as events or processed outputs asynochronous with sampling intervals, and derived parameter measurements taking a relatively long time to compute.

An informal world-wide survey of test organizations showed that for any given test (both now and projected for the future) the ratios of Category 1 to Category 2 data could vary from 20%/80% to 80%/20% respectively. There is a nearly universal desire to have derived parameter data and spectral analysis results (array processing) available in realtime along with other compressed EU data.

Category 1 data is easy to handle in an asynchronous parallel computer system and present preprocessors can also handle this type of data quite efficiently. Category 2 data can become a problem for asynchronous parallel computers at high throughput rates because data requiring concatenation or historical values of digitally filtered measurements must be stored in a common Global memory until all PCM words (or bytes) which make up one measurement are collected. At processing rates above 400k to 500k wps, the common processed data bus becomes overloaded (even a 10 MHz bus) when all of the asynchronous parallel processors need to access Global memory almost continually for concatenation words or for historical values of digitally filtered parameters.

The next problem to overcome in an asynchronous parallel computer environment is how to broadcast key data such as, IRIG time or event markers to all CPUs simultaneously, for concurrent data tagging or logical gating of processes. Another problem is how to collect processed data buffers for host transfer when some parameters may (generally do) take longer to process in one computer than similar parameters do in a different computer.

Similar problems of asynchronous parallel processing include how to communicate with the system’s host processor in realtime for mode changes, and how to resolve arbitration

on the common global processed data bus when as many as sixty (60) computers are operating in parallel.

One key problem to solve is how to apply different processes to the same raw parameter concurrently without realizing data staleness of a processed value because of sequential processing. An example of this problem is a parameter such as impact pressure that needs to be converted concurrently to indicated airspeed, calibrated airspeed, true airspeed, knots/second and feet (or meters)/second. Related system problems are where/how and when you ID tag the raw multiplexed data and how you route data that is already in EU from an avionics computer that is multiplexed in the same stream with raw pressure, temperature or position data which must be converted to EU. Another key consideration is how to handle aperiodic data such as an event triggered EU conversion or data compression based on rate or slope changes or other non-predictable aperiodic changes in the input signal level.

A major user requirement affecting system architecture is how to store a circular wraparound file of all raw data for anomaly investigations. The probability of the test engineer monitoring the parameters that cause the anomaly is generally near zero. When an anomaly is detected, how do you continue realtime processing of critical parameters which must be continually monitored whether an anomaly analysis is in process (or required).

There are, of course, many other critical considerations in the design of a high speed parallel computer system. One is having enough ID tags to handle all anticipated raw, concatenated and derived parameters concurrently in a common computer memory (such as when the Host CPU needs both raw and EU converted data). As part of Teledyne's user survey, it was discovered that one air frame manufacturer had already projected a need for over 40,000 different ID tags for processing data from a single test aircraft.

Another significant consideration is how to account for different processes being applied to the same parameter during different modes of a test, as well as routing data to different MMIFs at each phase of the test.

Along this lines are considerations of how to change processes as a function of events, or specific conditions such as loads or airspeed limit exceedances during a test. For rotor research testing on helicopters, data must be phase correlated to blade position or rotor speed and harmonic analysis must be made based on "n" times the rotor speed.

Finally, the most important consideration is how does the system operator change things affecting the man-machine interface on-line (during a test) without recompiling the realtime program. This is especially complicated in a asynchronous parallel computer environment where any computer (out of up to sixty) can process any parameter. Examples of this

dynamic reconfiguration problem include: how to change a parameter sensor cal file if the wrong one was assigned at system configeration time, how to change a parameter being displayed on one graph of a given graphics processor terminal, how to change limits and alarm levels in realtime, how to enable or disable alarm algorithms, how to disable wild point editing to see if there is a noisy sensor or whether the real data has step functions or some other unpredictable characteristic. The bottom line is, there are multitudes of problems to solve in designing an asynchronous parallel computer system - some in hardware, some in software and some with combined solutions. The key goal of all these considerations is to end up with a computer system that is easy to configure, easy to operate, and easy to expand the realtime throughput capacity. We will address some of these more critical considerations in both hardware and software.

ARCHITECTURAL SOLUTIONS

Defining optimum solutions to each of the problems to be solved in developing a massively parallel computer system had a significant impact on the system architecture. One of the most important conclusions was that to be able to minimize the number of computers required to process high aggregate word rates, it was necessary to have the fastest computer that it is possible to build with current microcomputer device technology. Next, in order to compute complex derived parameters in realtime, each Alogrithm Processor must be a general purpose computer with an independent floating point coprocessor and a large RAM memory to store all sensor cal files, lookup tables and processing algorithms necessary to process any (all) parameters). This is a critical consideration since no disk accesses can be permitted by any of the Algorithm Processors (APs) in realtime. A high speed pseudo cache memory must be incorporated to speed up table lookups and the most frequently used processes.

It was concluded that ID tagging must be accomplished in the decommutator or data source interface modules, because data from different sources must be ID tagged before they can be merged for processing, in order to preserve data source identification. Another key to speeding up the processing is the need to decommutate variable length PCM words in the Decom, to simplify ID tagging of measurements (parameters) less than 16-bits (or one PCM word) long. It turned out that Decom Systems, Inc. (DSI) was developing a new 10 MBPS, Motorola 68000 based decom that performed this function. Thus, an off-the-shelf module was available to perform these very important functions. Bit/bytes manipulation and ID tagging in the data source module (Decom) leaves the general purpose computers to mainly perform computations and not data handling.

It was initially thought that a two bus system would prove adequate to reach processing speeds of 1.5 Msps. This concept would use an input bus to merge ID tagged data from multiple data source modules (decoms) with all parallel computers connected to this bus so

that any available computer can process the next parameter on the bus. Each computer could then output computed data on the second bus, a global processed data bus for transfer to the host or to the MMIF devices. This proved to be an inadequate conclusion because detailed timing analyses showed there could easily be excessive traffic on a 10 MHz processed data bus which must also support global storage of Type 2 parameters, raw data wrapfile disk recording/reading and support of Current Value Tables [CVTs] in Global memory of synchronous and asynchronous data.

Additional heavy loads on the Global processed data bus could be generated by recirculation of parameters which are applied concurrently to different processes running in different Application Processors (APs). Another bus load complication was presented by the need to broadcast events or time data to all APs concurrently for precise time correllation. Finally, there is the problem of bus loading due to the need to reconfigure each computer's program periodically for Scenario changes at each sequential mode of the test flight. Analyses of these problems showed that it was necessary to have at least six separate parallel buses plus a serial bus for realtime commands and to divide the functions among the different buses.

DATA FLOW THROUGH THE SYSTEM

The RMPS architecture envolved through many interactions between the hardware designers, the software designers,and the users into the final form presented on Figure 1, which shows that the data flows from left to right in a pipeline manner. ID tagged data from the parallel (up to eight) data source modules must first be merged so that parameters from different sources can be sent to the same computer when they are part of a single process. Merged data from the Data Source Bus [DS-Bus] are sent to the Programmable Data Distributor (PDD) which provides preliminary data distribution. The PDD does a preliminary data routing lookup in 55 nanoseconds, modifies the ID tag and routes the tag/data pair concurrently to any of up to 64 different destinations. One destination is usually the circular wrapfile disk. The disk writes can be by either of two physical paths depending on system throughput requirements. For low to medium speed applications, the disk writes are via the Global processed data bus, the Versabus. For higher system throughput applications there is a separate wrapfile data bus (K-Bus) between the PDD and disk controller that permits direct reads/writes without any Versabus traffic.

Concurrently with wrapfile recording, the data can be sent to the Host DMA buffer (if the data is already in EU) and it's also sent via the undirectional, Data Input Bus (DI-Bus) to the Application Processors. If the measurement is a Category 1 parameter, determined by the PDD Look-Up Memory [LUM] tag transformation, any Algorithm Processor [AP] not busy can accept the measurement from the DI-Bus. Bus arbitration logic permits only one not busy CPU to take a parameter from the DI-Bus at one time. If the measurement is a

Category 2 or 3 parameter the Run Time Compiler has preassigned the parameter to a specific AP so that measurement bytes requiring concatenation or historical values for digital filtering can be stored locally in the assigned CPUs memory. When the PDD does a lookup on the ID tag appended by the Decom and finds it is a Category 2/3 measurement, it translates the tag so that the measurement is sent to the input FIFO of the assigned CPU. All CPUs have 64 word deep FIFOs on their inputs to account for up to eight preassigned parameters from different sources arriving sequentially at the same CPU.

The CPU accepting a measurement performs a secondary lookup on the ID tag, fetches the assigned Parameter Processing Packet (PPP) from local RAM and sequences through the PPP until all of the required processes are completed. This process use the sensor cal files, lookup tables and algorithms) stored in the Parameter Processing Algorithm (PPA) library of the CPUs local memory. One of the PPPs processes may be to transform the ID tag and send the measurement pair back to the PPP via the Inter-Processor Bus [IP-Bus] for further distribution to other CPUs and execution of a different process.

After the AP cycles through the complete Parameter Processing Packet, it routes the processed measurement to the different destination devices specified in the PPP via the Global processed data bus (Versabus) and the Output Bus (VME Bus). The distribution and further realtime use of processed data largely depends on whether the RMPS is functioning as a preprocessor to a Host, or whether it is configured with MMIF peripherals and mass storage devices to function as a standalone computer system.

CONTROL PROCESSOR

In any multiprocessor environment it’s generally necessary to have a “Master” or “Control” Processor to function as the local Host and control the downloading and functions of the other computers and programmable peripherals comprising the complete system. The selection of the computer for the Control Processor functions is extremely important as the hardware must be fast, general purpose and be ported for a multi-tasking realtime operating system. Since the Control Processor is an “overhead” function, the most desirable solution was to select a monoboard computer such as the Motorola MV68K03 with local memory. This would have represented the lowest cost for this function but unfortunately ported operating systems such as Versados and IDRIS proved to be very poor operating sytems.

Teledyne conducted a trade study of nearly a dozen operating systems and finally concluded that UNOS, a realtime, multi-tasking Unix derivative offered the best compromise for both realtime operation and time shared software development. UNOS, however, is presently only ported to Charles River Data Systems [CRDS] monoboard

computers. The selection of UNOS thus dictated the use of the CRDS Model CP32 computer as the Control Processor.

The CP32 is a high speed Motorola 68000 based computer with two MC 68000 CPUs, one operating at 12.5 MHz (faster than other current monoboard computers) and a separate I/O CPU operating at 6 MHz. The CP32 has no local RAM (a disadvantage in a realtime environment) but it does have a 4kB high speed cache memory which offsets most of the disadvantages of having to use part of the RMPS Global memory to support Control Processor operations. Figure 2 is a functional block diagram of the CRDS CP32 Control Processor. Attached to one of the RS-232 ports of the I/O processor is the RMPS system terminal. This is a Intecolor Model 2405D color graphics CRT terminal and keyboard which also functions as the local MMI and can provide limited graphics capability (optional software). A second RS-232 port on the I/O CPU connects to the S-Bus interconnecting the Control Processor to the Programmable Data Distributor and all of the parallel Algorithm Processors (APs). The third serial port connects to a line printer which serves as the RMPS data logger. The fourth port is unassigned and can be used as a host interface port or as the serial download bus to communicate with the microprocessors of each of the downloadable modules of a Teledyne Multiplexer system. When more than four serial ports are required for a given RMPS configuration, the CRDS TP308 8-channel UART card is added. This Versabus card provides eight asynchronous RS-232 ports connected to the I/O processor of the CP32.

The CP32 CP is designed to maximize the performance of the 68000 microprocessor operating at 12.5 MHz. The memory segmentation logic and cache of the CP32 are designed to permit the processor to run at high speed, without wait-states, while the interface to memory uses a full 32-bit data path to take advantage of the 20 MBps bandwidth of the Versabus. Figure 3 is a memory access timing chart for the CP32 processor.

CRDS claims that the CP32 processor provides approximately equivalent compute power as a DEC VAX 11/750. A 4kB cache memory takes data from Global memory in 32-bit transfers and provides fast access for both instruction and data accesses from the processor. The cache access time is 150 ns, which allows the 12.5 MHz 68000 to operate without wait-states. This permits the prefetch facility of the 68000 to operate at full speed for maximum 32-bit operation throughput. The cache uses write-through logic for processor memory modifications, so the data in the cache never needs to be “flushed” to memory. A cache validity array insures that cache accesses only return valid results, even when other bus devices are active. The 12.5 MHz CPU operates at a 70%/90% cache hit rate, depending on instruction mix.

The RMPS uses a common system memory to support both current value tables for all Algorithm Processors, as well as for the local RAM supporting the CP32 Control Processor. The Global memory is implemented by a one or more CRDS Model DM1024 one megabyte MOS RAM boards. The DM1024 is a Motorola Versabus board which has a cycle time of 300 ns per 32-bit transfer. The approximately 85% hit rate of the CP's 4kB cache memory means however, that very little Versabus traffic is used by the CP accessing program tasks from Global memory in realtime.

The Global memory is used to buffer minor frames (or cycles) of processed data being sent to the host computer via DMA interfaces, as well as for program storage of the realtime programs executing on the CP.

Mass storage for the Control Processor is provided for the System disk and the disk controller. There are two optional SMD disk controllers; one the Interphase Model 2190 formatter/controller which supports up to four SMD disk drives and the Interphase Model 3200 formatter/controller, which is faster but only supports two disk drives.

The RMPS generally uses two Winchester disks; one functions as a System disk to store the OS and the CP and AP programs while the other supports the realtime circular wraparound file of raw data used for anomaly analysis or intermaneuver monitoring of different parameters that are monitored in realtime.

The standard RMPS configuration has an Amcodyne Model 7110 dual 26MB disk drive with one fixed disk and one removable disk. The fixed disk is normally assigned the function of the RMPS system disk.

The Interphase model 3200 disk controller uses a 68000 processor plus two high speed bipolar state machines in a multitasking architecture which provides an average throughput rate of up to 2.4 megabytes/second. The virtual buffer architecture reduces or eliminates data transfer delays caused by disk rotational latency and data overrun/underrun problems associated with FIFO-based controllers. The V/SMD 3200 offers a 1:1 disk interleave and a highly intelligent caching scheme designed for UNIX. It provides programmable 8-,16- or 32-bit wide data transfers and 16-, 24-, or 32-bit addressing capability.

All information necessary for the setup, control and operation of RMPS is maintained in the Run Control data base and stored on the system disk under control of the CP. The accessing and manipulation of the data structures of the Run Control data base is exclusively performed by the data base control routines executed by the CP. All accesses to the Run Control data base by MMI software, the Run Control Compiler, Run Mode Initialization and Realtime Display control software are via these data base control routines. Data base control routines include the ability to operate on the data base records

to; create a record, delete a record, modify a record, obtain a record and verify a record.

The Run Control data base contains all the information necessary to control the acquisition, processing and distribution of ID tagged parameters acquired from data source modules in the RMPS. The Run Control data base contains the following types of files:

- Run Control Scenario File
- Parameter Definition File
- A/D Mux Definition File
- Data Stream Characteristics File
- RMPS Configuration File
- Realtime Display Definition File
- Run Control File

The Run Control File is generated by the Run Control Compiler based upon information contained in the other six files. The six input files are formatted test files and may be entered by an operator at the system CRT or may be transmitted to RMPS from a host processor. When RMPS is operated under the control of a host processor, the input files define the setup and control interface of the RMPS to the host processor.

The Run Control Scenario File contains up to ten Control Scenarios. Each is associated with a particular mode or phase of a test flight and may be oriented towards a specific testing discipline, e.g. handling quantities, propulsion, structures, avionics, etc. Each Run Control Scenario defines which input parameters are to be processed concurrently and it lists all the parameters used in the Scenario and specifies the Parameter Processing Algorithm names to be applied to each parameter. It lists all the destinations for distribution of processed data.

It is possible to assign a specified processing sequence to a block or blocks of parameters. This is important in reducing the time necessary to define a test flight processing configuration, as well as, the time to modify a test program to adapt to daily changes in the test configuration.

The Parameter Definition File contains definitions for all the parameters listed in the Run Control Scenarios plus any special or overhead parameters required for configuration control, documentary data, etc. Information in the Parameter Definition File is derived from sensor, aircraft instrumentation and calibration data bases residing on the host processor and includes coefficients, constants, limits and other values that may be required by any valid process on any parameter.

The RMPS Configuration File contains all the information necessary to define the hardware resources available at runtime. These include; the number of data source modules, type and base addresses, the Global memory configuration (Versabus or multipart) and amount of money, the number of TMPs, base addressing and revision level of each AP, the number of (display) terminals attached to RMPS and capabilities of each, the printer port, if any, the number and types of analog output modules and their addressing, the number and types of discrete output modules and their addressing, the mass storage configuration, DMA configuration to the host, and the wrapfile configuration.

When RMPS is under the control of an operator at the system CRT, displays of realtime data on the system CRT are supported. The Realtime Display Definition File supports up to ten Realtime Display Definition Groups where each Group is associated with one Run Control Scenario. Multiple Realtime Display Definition Groups may be defined for each Run Control Scenario containing: the Run Control Scenario number, the display type, i.e. tabular display, bar graph and/or annunciator display, the display title(s) and a parameter selection list defining the parameter name and color assignment.

Run Control Files are the primary output of the Control Processor's Run Control Compiler. Each Run Control File contains information derived from the Run Control Scenario File, the Parameter Definition File, the A/D Mux Definition File, the Data Stream Characteristics File, the RMPS Configuration File and the Realtime Display Definition File. One Run Control File is generated for each Scenario in the Run Control Scenario File and contains the bit, frame and subframe synchronizer card setups, the A/D Mux setup, the analog and discretes output card setups, the PDD setup, the parameter processing assignment, parameter processing packets and system control tables.

The Run Control Compiler operates in the CP and translates the information contained in the RMPS input files into a collection of hardware unit loadable images, parameter processing packets and algorithm control tables which are combined in the Run Control File. For each Run Control Scenario, one Run Control File is generated. The Run Control Compiler identifies all parameters needed in the Scenario and constructs the compressed mapping table from 16-bit tag to PPP vector offset. It constructs device images for each PCM stream and constructs A/D Mux device images for (RMPS) configurations that utilize a Teledyne multiplexer as a data source. It assigns parameter processing tasks to specific APs such that constraints on memory utilization, CPU loading and bus bandwidth are satisfied. It also generates the PDD Lookup Memory [LUM] image from the parameter assignments and from the wrapfile requirements specified in the setup files.

The compiler constructs a memory map for each AP memory and the Global memory and from this map generates the PPP vector table for each AP and constructs PPPs for each parameter listed in the Scenario, as well as constructing the PPA library callout list for

each AP. It stores the resulting Run Control File in the Run Control Data Base and generates printable text showing the expected CPU loading for each AP and the CP, as well as memory utilization of each AP local memory and the global RAM and the expected bus loading for each system bus. It generates an assigned parameter symbol table for each AP that can be uploaded to the host processor and constructs Realtime Display control tables for each Realtime Display Definition Group associated with the Scenario being processed.

The CP software also includes an incremental compiler which permits the operator during on-line operation to:

- Change sensor calibration files
- Modify an EU conversion algorithm
- Install a new EU algorithm
- Modify alarm limits
- Install new alarm limits
- Enable/disable digital filtering and alarms
- Modify or install a new filter algorithm
- Reroute data to different data destination devices
- Toggle wild point editing to see if a channel is really noisy or how noisy

These changes can be made by the system operator without affecting any other realtime processing and display functions not affected by the changes made. The changes are compiled incrementally into the data base without going off-line and performing a full runtime compilation (which could take from 5 minutes to 15 minutes).

DATA SOURCES MODULES

There are several standard data source modules for the RMPS. These include; PCM Decoms, an IRIG Time Code Interface, a multichannel ARINC 429 bus interface, and a MIL-STD 1553 A/B bus interface. All data source modules perform the same basic functions, that is, they synchronize their inputs with the data source, buffer the data, decommutate or concatenate the data (if required), add a 16-bit ID tag to each measurement and output ID tagged pairs (data/ID) through a FIFO onto the RMPS Data Source Bus (DS-Bus). The DS-Bus has arbitration logic to accommodate up to eight data source modules concurrently with different data source modules accepting different throughputs depending on the data rate from the source. For example, the ML STD 1553 A/B bus listener interface module has a maximum input of about 50k wps which is the limit of the 1 MBPS serial bit rate on the 1553 bus.

The DSI Model 7701 PCM Decom is the most generally used source module for decommutation of IRIG standard serial PCM data streams. The 7701 Decom is a two card Versabus source module with a 10 MBPS Frame Synchronizer/Data Tagger board [FS/DT] and a companion Subframe Synchronizer Board [SFS]. The SFS card is required to decommutate synchronous or asynchronous subframes. Any number of asynchronous subframes may be decommutated by the SFS as long as the SF sync pattern is the same in each subframe and the asynchronous subframes are the same length. For other asynchronous subframe strategies, a separate SFS card is required for each asynchronous subframe in the data cycle map.

One of the FS/DT boards is required for each separate PCM data stream to be merged by the RMPS. The FS/DT board is set up and controlled by the CRDS CP32 Control Processor. Its inputs are the serial PCM data streams and bit clocks. Its outputs are parallel 16-bit data words and a 16-bit channel ID tag word per PCM data word applied through a FIFO to the RMPS Data Source bus [DS-Bus].

The key feature of the FS/DT is that it can decommutate variable length PCM words of from one to sixteen bits in length in realtime and add a 16-bit ID tag to each measurement. This means that discretes are unpacked and ID tagged, and bytes of the same measurement in sequential PCM words are concatenated (up to 16 sequential bits such as two 8-bit bytes of a 1553B bus word) and ID tagged before merged on the DS-Bus with data from other source modules. The FS/DT/SFS card set operates on serial data up to 10 MBPS or 1MWPS (based on 10 bits per PCM word).

The Time Code Interface [TCI] card is another data source module for the RMPS. The TCI accepts parallel 48 line BCD IRIG time and divides each time word into three measurements. It adds a 16-bit ID tag to each 16-bit byte prior to outputting tagged time through a FIFO to the DS-Bus for merging with data from other source modules.

Another data source module is the ARINC 429 bus interface module. This card interfaces with either fourteen or twenty-eight ARINC 429 serial avionics buses. Each channel can be programmed to accept either l2k Hz or 100k Hz bus outputs. It decodes the programmed labels from each input bus and formats accepted data in a buffer where each measurement is appended with a 16-bit ID tag. ID tagged pairs are outputted through a FIFO onto the DS-Bus in the same manner as tagged data from any other Source module.

The DS-Bus is a 4 million words/second bus so the aggregate merged data rate is limited to 2 million measurements per second. All data placed on the DS-Bus is routed to the Programmable Data Distributor for distribution within the parallel processing system.

PROGRAMMABLE DATA DISTRIBUTOR

The Programmable Data Distributor [PDD] is a MC68000-based, high speed data distributor which performs both preliminary and secondary data distribution within the RMPS. It will accept data/ID tags from the data source modules at sustained word rates (16-bit words) of up to four million words/second and route them to any of 64 destination devices. The PDD has six external bus interfaces and three internal buses. The external bus interfaces are:

1. The Data Source Bus [DS-Bus], a 16-bit unidirectional bus which provides primary input data/tag pairs from up to eight data source modules.

2. The Data Input Bus [DI-Bus], a 18-bit unidirectional data output bus which routes all data to be processed by the RMPS in realtime to the Algorithm Processors (APs) on the Teledyne MultiProcessor (TMP) boards.

3. The Inter-Processor Bus [IP-Bus], a 16-bit wide bidirectional bus for broadcasting data to all Aps concurrently or for receiving data/tag pairs from an AP with a modified tag for the PDD to send to another destination.

4. The Versabus [V-BUS], a 32-bit wide data plus 24-bit wide address bus which functions as the global processed data bus of the RMPS. The Versabus is converted to a VME bus for transmitting of processed data to analog and discrete output cards in the Eurocard Cage.

5. The Serial Bus [S-Bus], a 19.2 kBaud bus connecting the PDD with all of the APs and the Control Processor [CP] for realtime communication of commands.

6. The Wrapfile Bus (K-Bus], a dedicated 17-bit wide bus (cable) between the wrap file FIFO on the PDD and the wrapfile disk controller (to permit bypassing the Versabus for high speed wrapfile

Three internal buses are used for the routing of data within the PDD to the various destination interfaces. The PDD does no processing, although it would be capable of some limited processing. It does provide the global memory management functions for data transfers between the APs and the CP32 and its companion DM 1024 1MB Global memory board. Figure 4 is a functional block diagram of the PDD showing all of the RMPS data bus interfaces.

The MC68000 microprocessor on the PDD has 32k words of EPROM to store operational firmware. It also has l6k (x 16 bits) of 120 ns static RAM for scratch pad. However, the

CPU on the PDD is not used for directing the flow of information from the DS-Bus to various destinations as it is too slow for this function. Data distribution is performed by a 128k word by 32-bit Look-Up Memory [LUM] driven by a 2910A micro-sequencer which will perform an ID tag lookup in 55 ns. The LUM is a 32,768 word by 32-bits wide high-speed memory, which stores the transformed tag of each PCM data word. The LUM contains instructions for data destination routing, including TMP assignment, wrapfile assignment and a 6-bit destination field to permit routing of a data word to any of up to 64 destination devices, including the MC68000 microprocessor on the PDD. Four bits of the LUM are reserved for future improvements or expansions. The LUM can process 2 million PCM words (a data word plus its 16-bit ID tag word) every second. The LUM process appends a two bit tag to all words. This tag identifies whether the data word and its ID tag are raw realtime data from the data source or whether it is stale raw data and tags being read back from the wrapfile disk.

Data routed by the PDD to any AP for processing can, as part of its processing, be returned to the PDD via the IP-Bus for rerouting to different destinations. This is the rerouting or recycling technique for parameters such as impact pressure that are needed for several concurrent processes such as IAS, CAS, TAS, knot/sec and/or feet/sec. Another feature of the PDD is its ability to broadcast any parameter (such as IRIG time) to all APs at the same instant via the IP-Bus. This is especially important when parallel processes must be time correlated to some event or function such as rotor blade position.

The PDD is the hardware source/destination for all data routed to the circular wraparound file disk for realtime analysis between flight maneuvers or for anomaly analysis. The wrapfile buffer is a 32k word deep by 32-bits wide FIFO that is used to buffer parameters and ID tags sent to the wrapfile disk. At lower aggregate word rates, the wrapfile disk writes are from the PDD via the Versabus to the Interphase Disk controller. For higher speed aggregate word rates (above 500k wps) where the wrapfile traffic may add too much traffic to the Versabus, the PDD K-Bus provides a direct cable path to the Interphase Disk controller. The K-Bus data path can be used for both disk writes and disk reads.

The PDD has the same DUART that is part of each AP. As with the AP, one of the UARTs is being used for standalone testing of the PDD outside the RMPS environment. The second UART is connected to the S-Bus, which interconnects the PDD to all APs and the CP. The S-Bus is used as a realtime command and control bus for changes in RMPS configurations initiated by the CP, as opposed to using the Versabus or the DI- or IP-buses for this function.

PARALLEL ALGORITHM PROCESSORS (APs)

The RMPS supports up to sixty parallel Algorithm Processors (APs) operating asynchronously. Each AP is two identical floating point computers with 500kB of RAM and 16kB pseudo cache memory per computer packaged on one Versabus Board (Teledyne Multi-Processor - TMP). Each AP shares common interfaces to the DI-Bus, IP-Bus and the Versabus. Each AP has a DUART providing two serial bus interfaces. One UART is used as a test port for connection of a terminal for standalone testing of the computer. The second UART connects to the S-Bus which ties all APs and PDD to the CP for transfer of realtime communications and commands. The TMP showing both APs and the bus interfaces is presented in functional block diagram form in Figure 5.

The memory in each AP is supported by a keep-alive battery at the system level. The size of this battery is user selectable from a few seconds to one minute capacity. The system level keep-alive battery has its own trickle charger to keep the battery fully charged so it may support the entire system through severe power line transients and/or short power outages. In addition, the V_{cc} supply lines coming onto the TMP board from the Versabus backplane are fused so that each board has its own isolated supply voltages. If a catastrophic device failure occurs in one TMP board, the V_{cc} supply fuse blows, isolating that TMP board from the rest of the system. Thus, no single point failure can bring down more than one TMP board, or one TMP failure will not bring down the entire DPA.

Each AP on the TMP is a Motorola MC68000 microprocessor operating with approximately a 12.5 MHz clock. The precise crystal frequency has been selected so each AP will experience no wait cycles or a minimum of wait cycles. The memory is 512kB of 120 ns RAM which will permit memory fetches in 4 clock cycles. The AP memory is designed for both byte and word transfers. It is mechanized with byte parity using eighteen 256k bit memory chips which will permit the memory size to be increased to 2,084kB when 1M bit chips become available.

Each AP has its own separate Floating Point Processor [FPP] that is packaged as a daughter board, one of which is plugged into each AP on the TMP. The FPP operates independently from the MC68000 CPU as a coprocessor to the AP's MC68000 main processor and it performs only floating point computations. The FPP consists of a control section, an address generator, an arithmetic processing unit [APU], floating point conversion logic and CPU interface logic. Figure 6 is a functional block diagram of the FPP showing the main CPU as a single 68000 block for reference purposes.

The standard FPP is a single precision computer working with 32-bit words. As an option, the FPP can be provided as a double precision FPP working with 64-bit words. The FPP has a maximum flowthrough rate of 8MFLOPS operating at a 16 MHz clock for pipeline

operations. For a linear conversion from a 16-bit operand to EU in DEC floating point format, the FPP throughput is about 0.8MFLOPS.

AP OPERATIONAL SOFTWARE

The AP operational software is the RMPS realtime processing component which resides in the local memory of each CPU. This software is stored on the RMPS system disk and the Run Mode initialization software of the CP downloads the AP operational software the RAM of each AP in the system. It provides three types of output buffers; raw data buffers, EU data buffers, and asynchronous data buffers. It also provides derived parameter processing, floating point format conversion and error processing.

The AP operational software is composed of an Executive, the Parameter Processing Sequencer and six generic classes of PPAs. It supports multitasking, where the processing of realtime data is a single background task. The AP executive resides in the pseudo cache RAM and it responds to external service request interrupts generated by the CP, including start, stop, restart, reconfigure and process wrapfile. It will respond to FIFO full interrupts by removing all FIFO entries and storing them in local RAM, as well as responding to IP-Bus broadcast buffer full interrupts by immediately processing the information in the broadcast buffer. It provides a facility to send ID tags and data words over the IP-Bus to the PDD for rerouting to another AP, as well as obtaining data from global RAM to support applications processing functions which require the involvement of more than one AP which cannot be met by serial processing. The AP executive provides a facility, using either Test-and-Set instructions on semaphores in global RAM, or server processes in the CP using event counts, whereby individual APs can synchronize access, and use shareable resources. It will output processed data to the global RAM, to a shared multipart memory and/or to analog and discrete output channels. It automatically schedules self-diagnostic routines on restart or upon receipt of a command from the CP.

The Parameter Processing Sequencer obtains ID tags and data from the DI-Bus input FIFO, locates a PPP and controls the processing of the measurement by calling all of Parameter Processing Algorithms (PPAs) listed in the PPP. The six generic classes of PPAs are:

- Data Acquisition
- Data Quality
- Engineering Unit Conversion
- Limits/Events/Alarm Processing
- Data Compression
- Output Processing

PPAs are small sequences of executable code that performs a single processing task (e.g., wildpoint detection, 1st through 7th order polynomial EU conversion, ZFN data compression, etc.). Each PPA has an initialization code section and a mainline code section. Based on the condition of an initialization flag, which is set during a data stream startup, cleared during normal data flow, and reset based on error conditions, the initialization code of a PPA is executed prior to the mainline code when the PPA is called by the Parameter Processing Sequencer. For every applicable PPA, support is provided for linear and non-linear data types. Linear data types permit many functions to be performed on the raw form of the data, in order to reduce the number of EU conversion operations required. Similarly a distinction is made between monotonic sensors and non-monotonic sensors for some compression algorithms, e.g., min/max.

Figure 7 presents the important elements controlled by the AP operational software, including the DI-Bus input FIFO, the Parameter Processing Sequencer, the PPP Vector, an example of a PPP, six classes of PPAs and a “software bus” for the transmission of various forms of data between the PPAs and the Parameter Processing Sequencer.

There is one PPP for each unique parameter ID tag assigned to an AP. The number of PPPs loaded into anyone AP is equal to the total number of Category 1 parameters plus the number of Category 2 parameters assigned to the AP. The number of PPPs loaded into APs dedicated to Category 3 processing is equal to the number of unique parameter ID tags assigned to each individual AP reserved for Category 3 processing.

Each PPP contains a list of parameter processing steps that are executed by the Parameter Processing Sequencer until the list is exhausted, whereupon the sequencer returns to the input FIFO. Each processing step contains a PPA address followed by any necessary constants, coefficients, lookup table addresses, global RAM addresses and previous values required to perform the processing step. The Parameter Processing Sequencer calls each PPA in the PPP. The PPA access the various forms of the data over a “software bus”. Each form of the data word (raw data, EU converted data, compressed data) is assigned to a specific AP hardware register.

Processing steps generally progress from data acquistion to output processing. However, variations in the order of PPA execution are permitted. For example, digital filtering may be performed before and/or after EU conversion, data compression may be performed before and/or after limit checking, and the sequence of EU converstion followed by output processing may be repeated for different EU conversion methods. The PPPn that is shown in Figure 7 may be considered a typical sequence that contains six generic processing steps as follows:

1. The data acquistion PPA performs a number system conversion (on the raw form of the data word) and replaces the converted value in the Raw Data register. If any conversion errors occur, the Status and a Message Pointer register is set with an error code and a pointer to an appropriate message string. The Parameter Processing Sequencer, depending upon specified options, may either abort processing for the particular data word or continue using a substitute value.

2. The data quality PPA performs a bandedge exceedance check on the Raw Data register. If the check fails, processing may be either aborted or continued with a substitute value.

3. The EU Conversion PPA performs a first thru seventh order polynomial conversion on the Raw Data and places the results in the EU Data register.

4. The Limit/Events/Alarm PPA performs an absolute limit check on the EU Data register. If any limits are exceeded, the Status and Message Point register is set with a limit failure code.

5. The Data Compression PPA performs a one of n data compression on the EU Data register. If a result is generated, it is placed in the Compressed Data register.

6. The Output Processing PPA sends the EU Data value to a DAC for output on a strip chart recorder. The EU Data and the Compressed Data (if generated) are output to a host computer buffer. The Raw Data value is output to a health monitoring CRT display.

This is just one example of a typical PPP which may be shorter or longer, depending on the processes to be applied to each parameter.

Data acquisition PPAs comprise all those processing tasks required to construct complete raw parameter values from PCM words. Raw data parameters that are distributed among portions of one or more PCM words are extracted and assembled into a single raw data quantity that can then be operated upon by the other processing steps, such as EU conversion. Supported data acquisition processing algorithms include; split word processing, multi-word concatenation, special format processing, accumulation of arrays of one parameter for digital filtering, FFT processing, etc., data type conversion and logical operations (OR, XOR, AND, NOT).

Data quality PPAs comprise those processing tasks that check, validate and improve the quality of data from the source flowing through the system (as opposed to data quality as a function of hardware errors within the system). When a data quality error is detected, an error message is generated and the raw data value is either used as is, replaced with a specific constant value or discarded. Supported data quality processing algorithms include; bandedge exceedance, data spike detection, wild point editing (discarding), N-point smoothing, and digital filtering. Each of these operations are performed on the raw data value.

The AP provides the capability of converting any raw parameter to engineering units via a first through seventh order polynomial or a lookup table where:

- $F(x) = ((((((Ax + B)x + C)x + D)x + E)x + G)x + H$, where the constants A through H are supplied by the Run Control File

- Table lookup is performed using a predefined table of point pairs with linear interpolation between defined points is used to determine the required EU value.

Each AP is capable of limit checking raw, EU converted and compressed data. The types of limits/events/alarms processing PPAs supported include: Maximum and minimum values, maximum oscillatory values, delta limit check, event occurrence detection, number of exceedances per defined time interval and alarm detection. All limit checking PPAs support up to three levels of limits. Out-of-limit conditions are communicated to the CP and/or the host computer upon detection, by posting an alarm message for the affected ID in the Asynchronous Message Buffer.

The AP is capable of applying data compression PPAs to any parameters. The following data compression algorithms are supported by the standard RMPS software:

- Average value of every n value
- Average value of last n values
- Min/max value
- In- or out-of-limits
- ZFN-floating aperature
- Every nth sample
- Bit match
- Fixed crossover
- Exclusive OR
- Harmonic Analysis

In addition to these compression algorithms, any user defined compression algorithm may be assigned to any parameter. Compression algorithms may be assigned to raw, EU or derived parameter data.

Output processing PPAs consist of those tasks necessary to route raw and/or processed data values to their final destination. This includes; transmitting EU data to the host, transmitting the compressed data to the host, transmitting alarms, event occurrences and limit exceedance messages to the host, maintaining a Current Value Table in global RAM, recording raw and/or EU data on stripchart recorders, driving event displays with raw data, EU data, compressed data, limit check messages, alarm messages and event notifications and driving analog outputs with raw, EU, derived and/or compressed scaled data.

MAN-MACHINE INTERFACE SOFTWARE

The Man-Machine Interface [MMI] source of control for the RMPS is either the local system CRT or a host processor CRT terminal connected to the CP via a serial communications line. In either case, all data transmitted across serial communications lines are ASCII characters. When the RMPS MMI is operating in the host processor mode, all commands, responses, status messages, etc. are terse, abbreviated forms of the communications used when the MMI is operating in the local operator mode.

MMI software permits the RMPS operator to prepare the RMPS data base for runtime operation and to control realtime operation of RMPS. MMI software comprises a hierarchical tree structure of menus, forms and displays that give the operator easy access to, and control of, all RMPS functional elements. This software is user friendly and it is essentially an interactive menu heirarchy which leads the system operator through all of the various levels of functional elements of the software. The menus are fill-in-the-blank types supported by “help” displays at all levels. The menu hierarchy starts with “User Log On” and divides the software into two different functional areas:

1. Setup Mode Functions
2. Run Mode Functions

The operator-system communication utilizes two types of interactive procedures. These are the top level menu type selection processes and the lower level fill-in-the-blank entry method. The menu procedures allow the operator to walk in and out of the various processes at will.

A fill-in-the-blank forms control technique is used at the lower level, where a large number of parameters or items must be entered. The operator-system interaction can be described

in terms of a “finite state machine”. That is, there are a set of defined system states with their corresponding screen formats, system responses, allowable selections and acceptable entries. The MMI menu tree, showing the four levels of menus used for setup and control of the RMPS are presented in Figure 8. The two top level functions of the RMPS software; the Setup Mode and the Run Mode are divided into four and five functional areas, respectively. The Flight Editor Librarian Selection Menu is one of the five main menu trees of the File Maintenance group of the RMPS Setup Mode software. Upon operator selection of the Setup Mode menu on the system CRT the operator may, in turn, select any of the following lower level set-up menus:

- File Maintenance Menu
- Realtime Display Definition Menu
- Diagnostics Control Menu
- Run Control Compiler Menu

The File Maintenance Menu, in turn, permits the operator to select any of the following lower level menus to perform maintenance functions on the input files:

- Run Control Scenario File
- Flight Editor File
- RMPS Configuration File
- RMDU SAT/M Hex File

The File Maintenance Menu is the broadest portion of the set-up mode operation as it is used to define the RMPS configuration and to establish most of the characteristics of the realtime processing to be done during each of up to ten different Scenarios during a test. Each file operation has a dynamic default directory which is replaceable in any file specification.

Selection of the diagnostics menu leads the operator through all of the system diagnostics, while the Realtime Display Definition menu presents a form on the system CRT that permits the operator to define the contents of the Realtime Display Definition File, including; selecting the display type, listing parameters selected for display, specifying titles, specifying display attributes of parameters.

At any level of menu operation the operator may Exit RMPS - terminate all processing, reset all special devices and return to the CP operating system command level.

After the operator selects the Setup Mode, the operator has the choice of either returning to the calling level (the RMPS Setup or Run Modes) or advancing to File Maintenance, Realtime Display Definition, Diagnostics or Run Control Compiler. To provide some

insight into the comprehensive MMIF we will present a few of the menus in the Flight Editor Maintenance menu tree which is one of the five lower level menus of the File Maintenance Menu of the Setup Mode (Figure 9).

When the File Maintenance Menu is selected, the operator has the choice of defining or editing; the RMPS Configuration File, the Run Control Scenario File, the Flight Editor File, the SAT-M Hex file or the Sensor Definition File. When Select Flight Library, (an option of Figure 9) is selected, the system responds by displaying the Flight Librarian Menu Tree as depicted in Figure 10. This tree shows all of the lower level menus of the flight librarian. The acronym list for the Flight Librarian Tree is presented in Figure 11. This is one of the Help forms which permits the operator to go further, without referring to a manual, in case he has forgotten the acronyms.

The Flight Librarian Menu Tree is a six-level deep and six-level wide organization that encompasses definition of the programmable variables of the airborne data acquisition system definition and related data categories. The Flight Librarian Selection Form shown as Figure 12 is utilized to initiate the creation of a data base for a new test aircraft (Selection 2). It also permits the the operator to delete a flight (one lower menu - Selection 3), copy a flight definition from one aircraft or flight number to another (Selection 4), merge a new aircraft flight with an existing aircraft flight (Selection 5) or proceed to the Data Selection Menu (Selection 6). It has three lower level menu options. These are the Select Flight (DATSEL), Copy Flight (FLTCPY) and Merge Flight (FLTMER).

The form corresponding to the Copy Flight option is illustrated in Figure 13. This form provides the means for creating additional copies of existing flight files. The options available to the operator at this menu level consists of copying the file, or returning to the Flight Selection Form.

These few examples of the MMIF menu tree respresent the hirarchal process where an operator can go from system “log on” to the lowest level of definition of a Test Scenario. Similar menu trees are used for on-line control of the system. It clearly shows that an operator need not be a software engineer or computer programmer to operate the RMPS.

CONCLUSIONS

There are many other features of the RMPS hardware and software which cannot be addressed in such a brief overview of the system. This product represents a state of the art advance in computer systems for realtime processing. It is a modular architecture using universally accepted standard buses, such as Motorola’s Versabus and VME bus. The

standard buses permit a large variety of off-the-shelf board level products to be incorporated into the system without any hardware or software redesign.

For example, it is possible at any time to add special computational board sets such as the Sky Computers, Inc. dual VME board Warrior array processor that has a 15MFLOP capability of processing IEEE P574 32-bit and 64-bit data, at memory access rates up to 20MB. The Warrior AP can do a 1024 point complex FFT in 3.9 ms (over 12 times faster than a TMP computer). To support this very capable array processor, Sky in turn provides a library of 100 sub-routines and 500 scientific routines which have been devloped in conjunction with the Numerical Algorithms Group of the United Kingdom.

The computation throughput capacity of the system can be expanded at anytime by merely plugging in more APs (TMP Cards). A large dual cage chassis is used to accommodate up to 18 Versabus modules plus 28 VME modules (10 full height, plus 18 half-height modules or larger quantities of each within the 20 full height slots available). Two chassis can be interconnected to double the card slot capacity to 36 Versabus modules and 56 VME Bus modules. By placing data source and PCM simulator modules in one chassis, with only computers (APs plus the CP) and peripheral interfaces in the second chassis, the realtime 10 MHz buses are not physically lengthened beyond 17 inches and dramatically slowed down, as is the case in other modular preprocessors where more chassis needed to process multiple data sources physically extend the CPU/Data buses. Multiple single or dual chassis RMPS can be interconnected via a 200 MB throughput Dataram multiport memory unit. Using the high speed Dataram as a Global CVT memory, the eight stream data source limit and the 1.5 Mwps throughput capacity of a single thread RMPS can both be multiplied factorially, again without software change or hardware redesign.

The use of standard buses and a multitasking operating system or the CP permits a wide variety of mass storage devices (tape transports and disks) to be added initially or at any future time to a system configuration. The same is true of graphics terminals and printer-plotters which permit a RMPS to function as a standalone computer system, as well as a compressed, EU data source for a super mainframe computer.

The use of an incremental compiler is the most important feature of the standard software, as changes in realtime processes are an essential ingredient of research and testing. At the same time, important changes must be able to be made without interrupting the processing of other parameters not affected by the changes being made. The system also provides graceful degradation of performance such, that with the failure of one computer, assigned parameters are reassigned to other computers and only a few parameters are lost to realtime display process while the reassignment is made (they are saved on the wrapfile disk). The CPU load leveling at run compile time is automatic and transparent to the operator. It is also failsafe, in that, if an operator requests more realtime processing than

there are available APs, the compiler will tell the operator to add more TMP cards. Even if this is not possible, the graceful degradation will permit elimination of the least critical processes until an acceptable computational load is achieved.

The RMPS horizontally and vertically expandable computer system architecture does not entice one with large theoretical capacities that are not achievable because of hidden realworld limits, as do most high performance systems. As the number of APs expand, so does the memory. In fact, memory generally expands faster than the need grows when more APs are added because of load leveling. All 65,536 ID tags are useable with one data source stream or with eight sources. Even this large number is expandable factorially when multiple RMPS are interconnected via the Dataram because each RMPS would have its own CVT buffers. Power supplies and thermal design have been calculated on worst-worst case configurations. Comprehensive diagnostics permit automatic trouble shooting to the pluggable module or subsystem level. Finally, all user programming is done with a high language, such as Fortran 77 or "C". The user does not have to worry about difficult microcode when he wants to

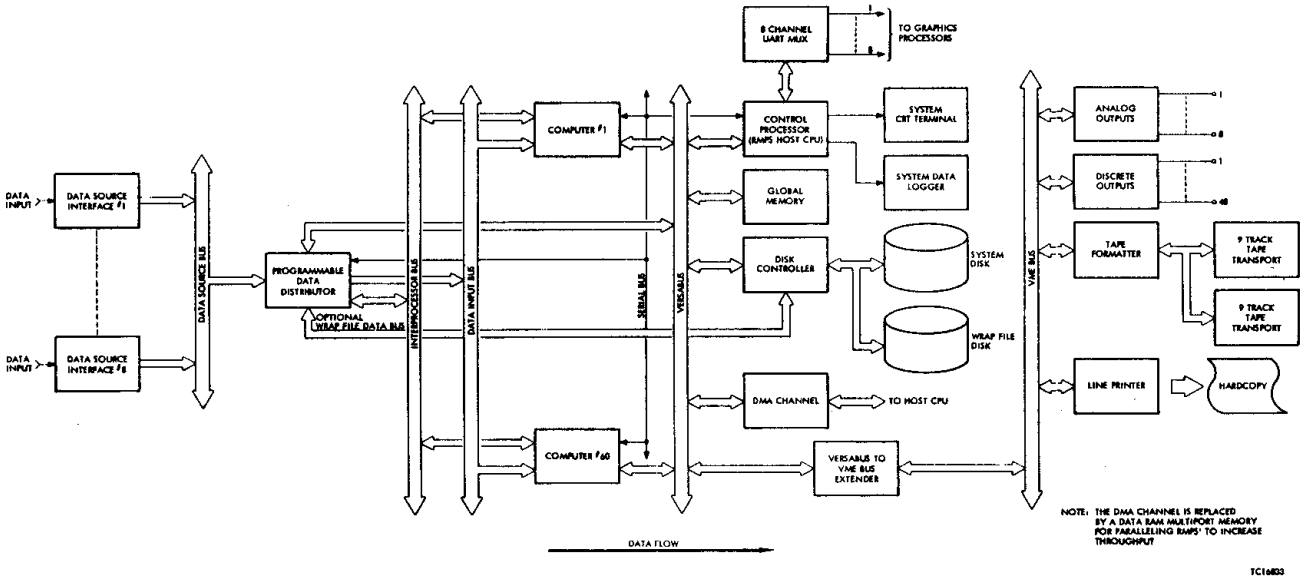

Figure 1. Data Flow Through RMPS

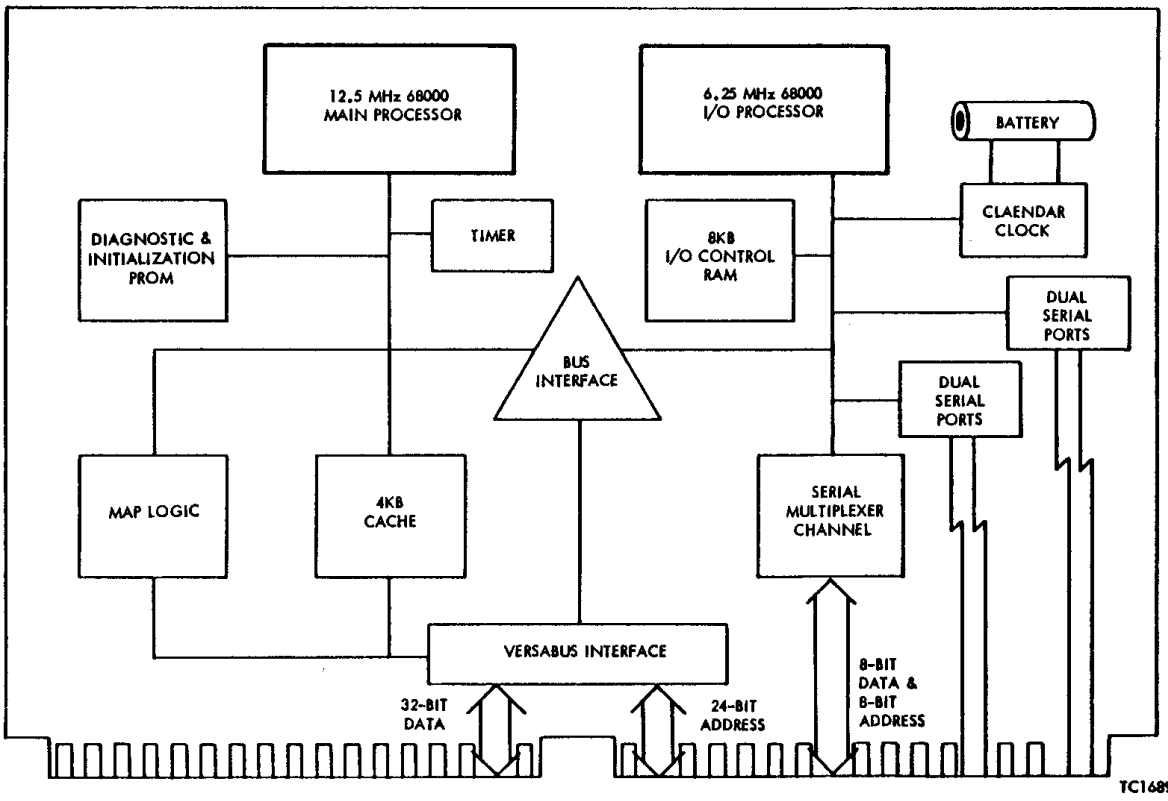

Figure 2. CRDS Model CP32 Functional Block Diagram

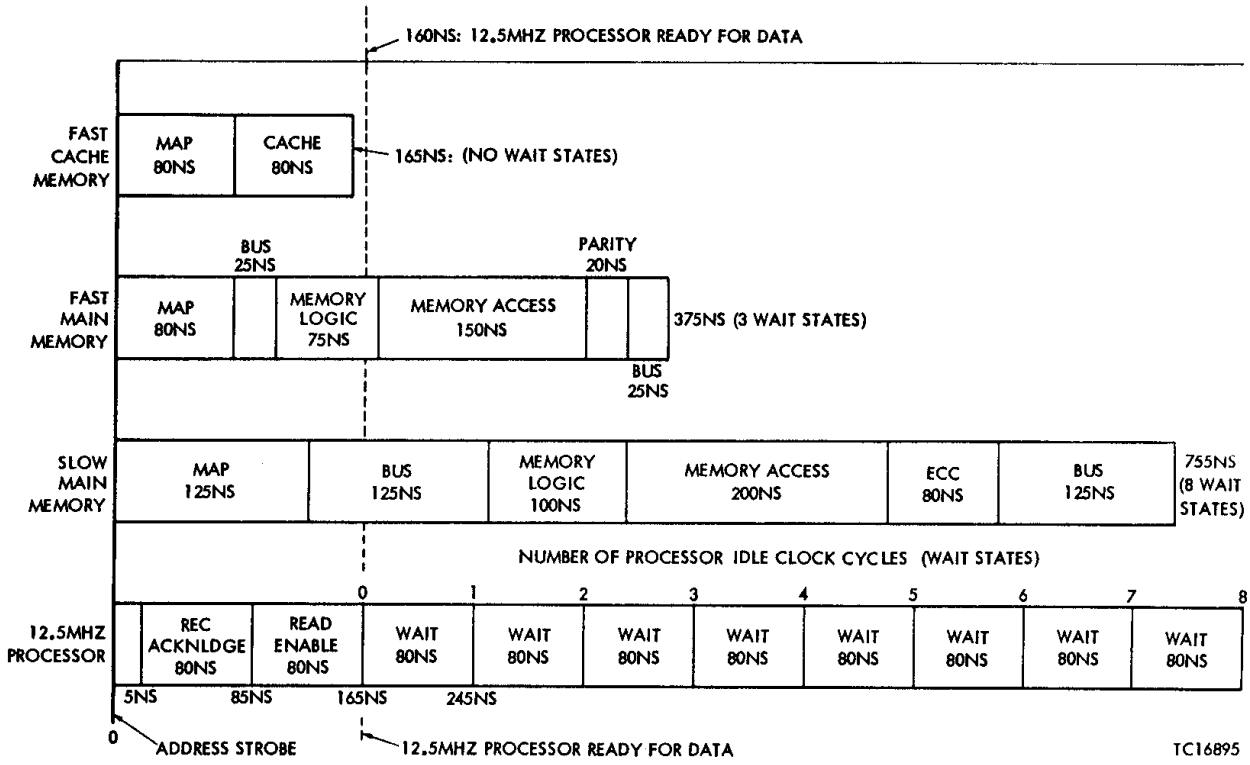

Figure 3. CP32 Control Processor Memory Access Timing

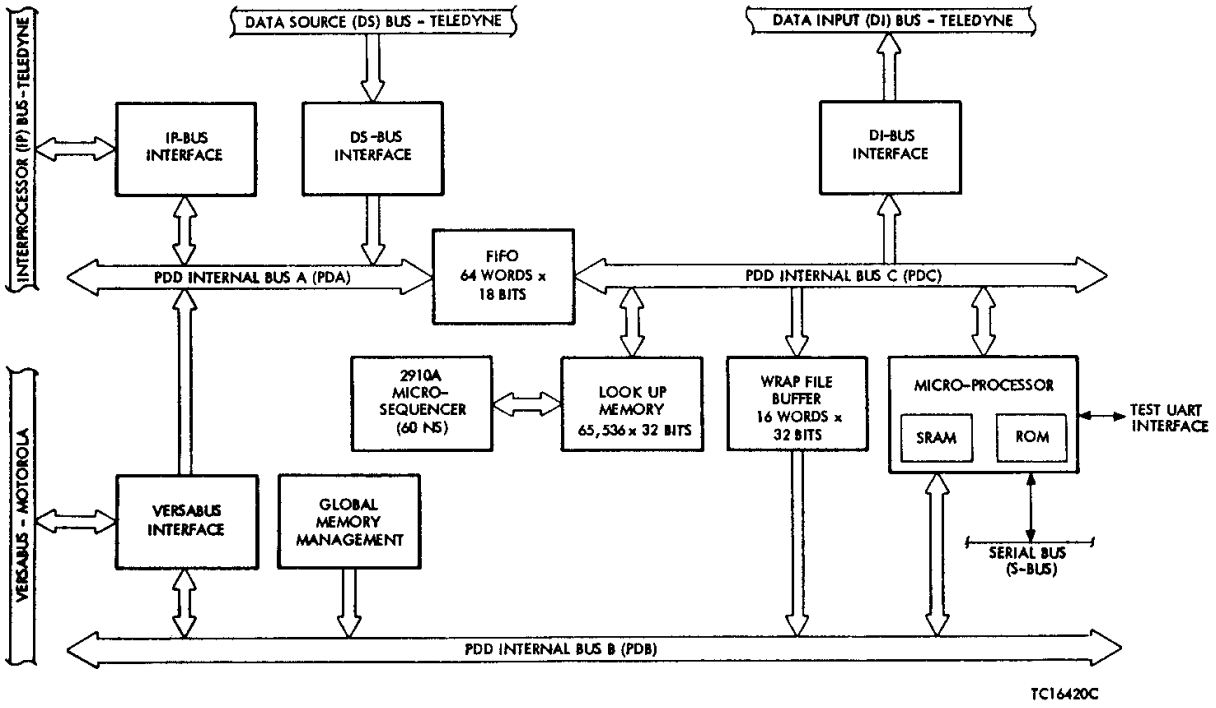

Figure 4. Programmable Data Distributor Block Diagram

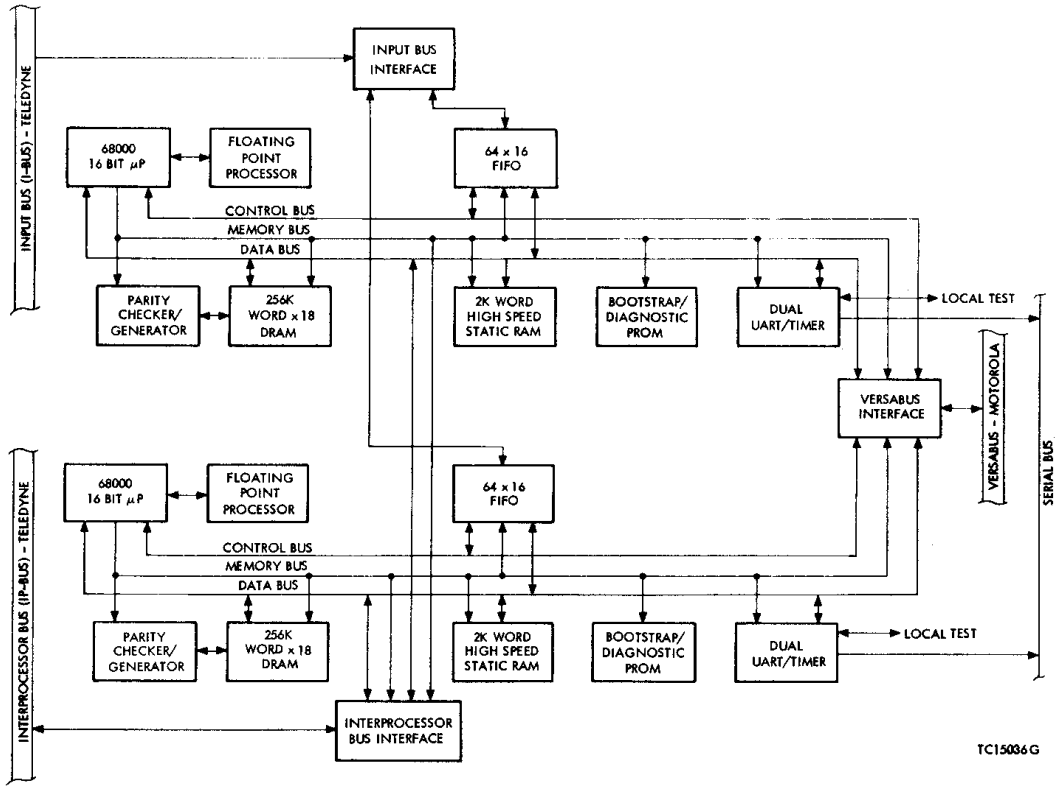

Figure 5. TMP Functional Block Diagram

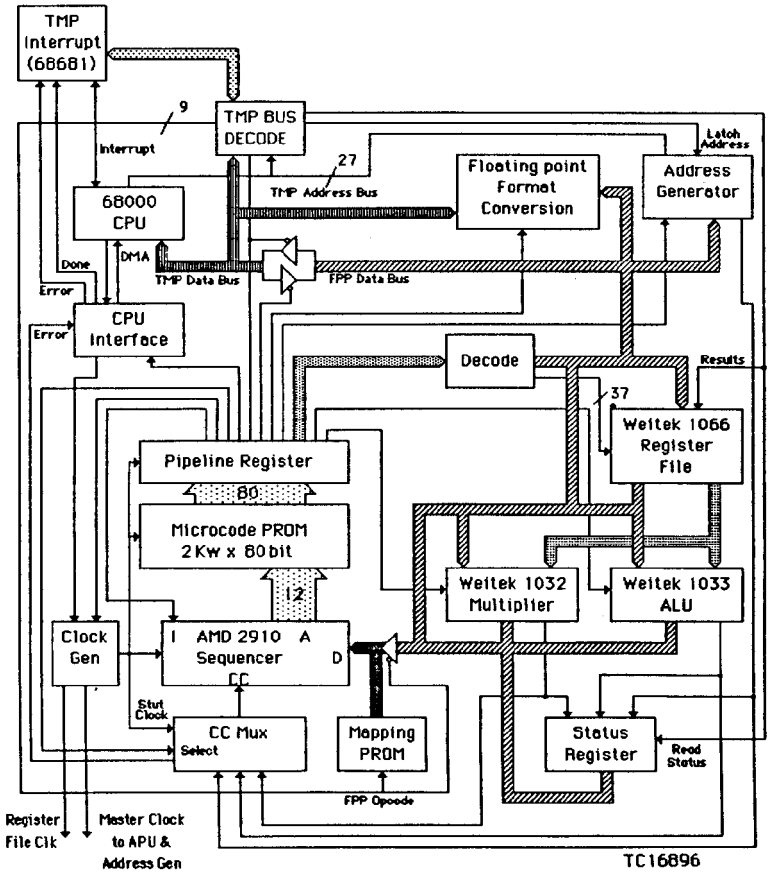

Figure 6. Floating Point Processor Diagram

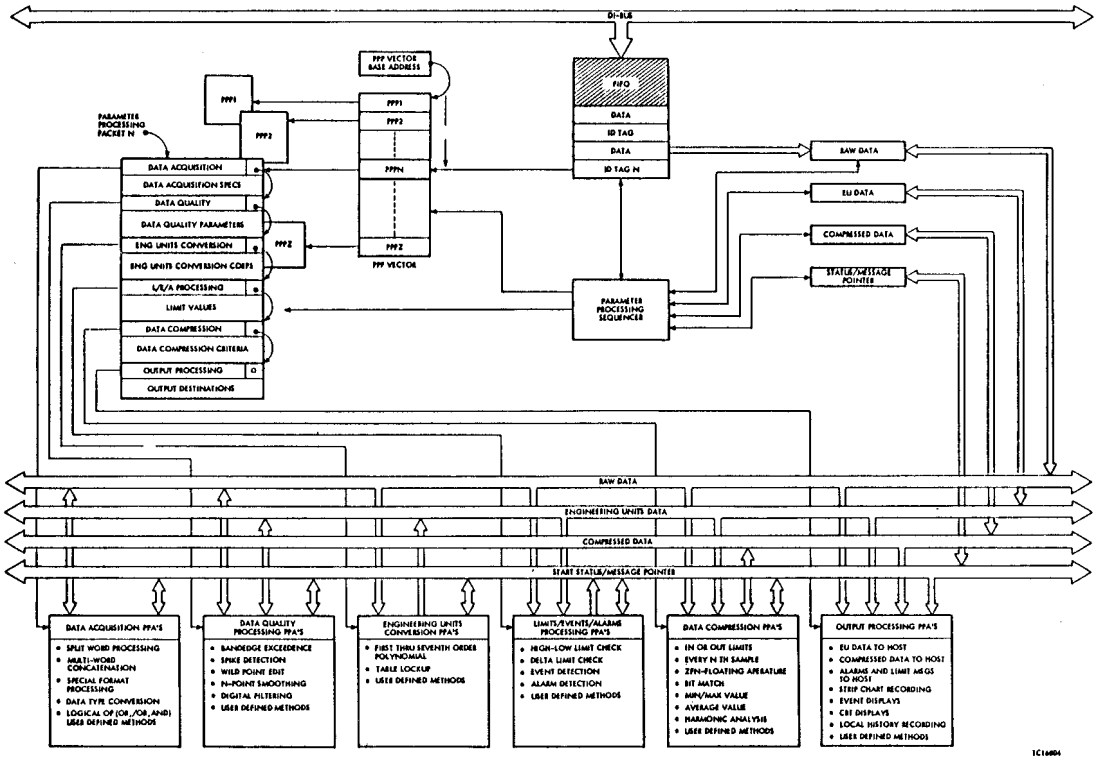

Figure 7. Parameter Processing Diagram

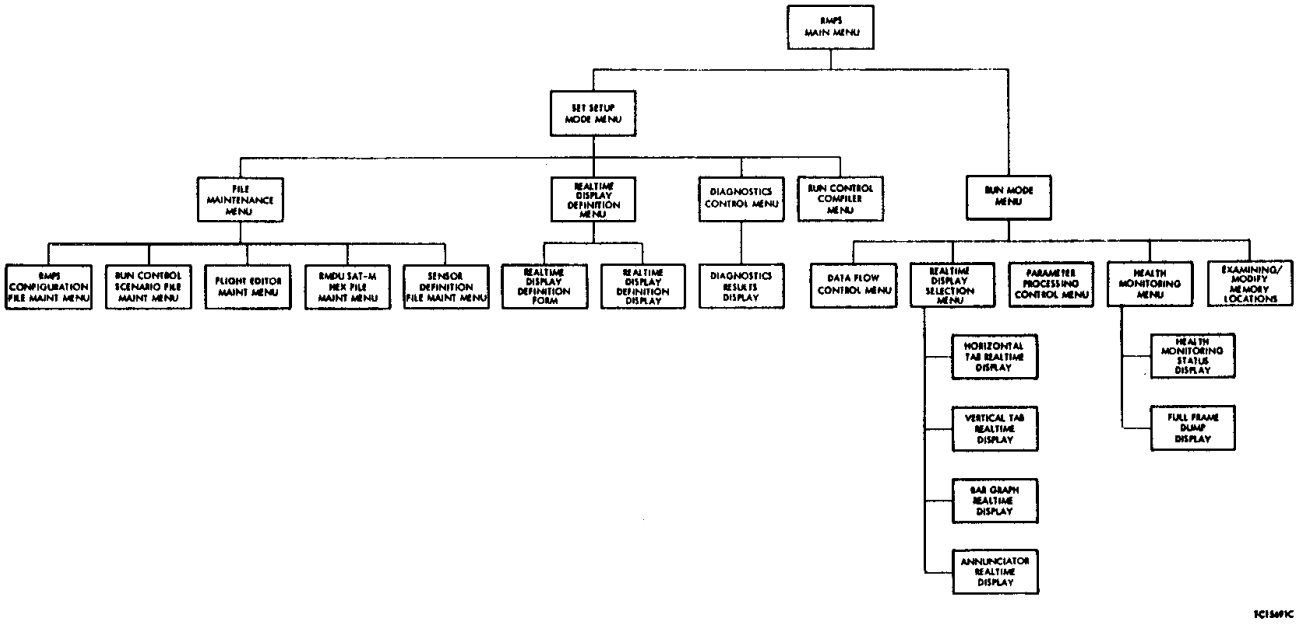

Figure 8. Man-Machine Interface Menu Tree

```
        1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
 1|                          FILE MAINTENANCE MENU                               | 1
 2|                                                                              | 2
 3|                                                                              | 3
 4|                    1. Exit RMPS Librarian                                    | 4
 5|                                                                              | 5
 6|                    2. Select Sensor Librarian                                | 6
 7|                                                                              | 7
 8|                    3. Select Flight Librarian                                | 8
 9|                                                                              | 9
10|                    4. Select Scenario Librarian                              |10
11|                                                                              |11
12|                    5. Select RMPS Configuration Library                      |12
13|                                                                              |13
14|                    6. Select RMDU SAT-M Hex Library                          |14
15|                                                                              |15
16|                       Select Option: _                                       |16
17|                                                                              |17
18|                                                                              |18
19|                                                                              |19
20|                                                                              |20
21|                                                                              |21
22|                                                                              |22
23|                                                                              |23
24|                                                                              |24
12345678901234567890123456789012345678901234567890123456789012345678901234567890
        1         2         3         4         5         6         7         8
```

TC16852

Figure 9. RMPS Flight Librarian Selection Menu

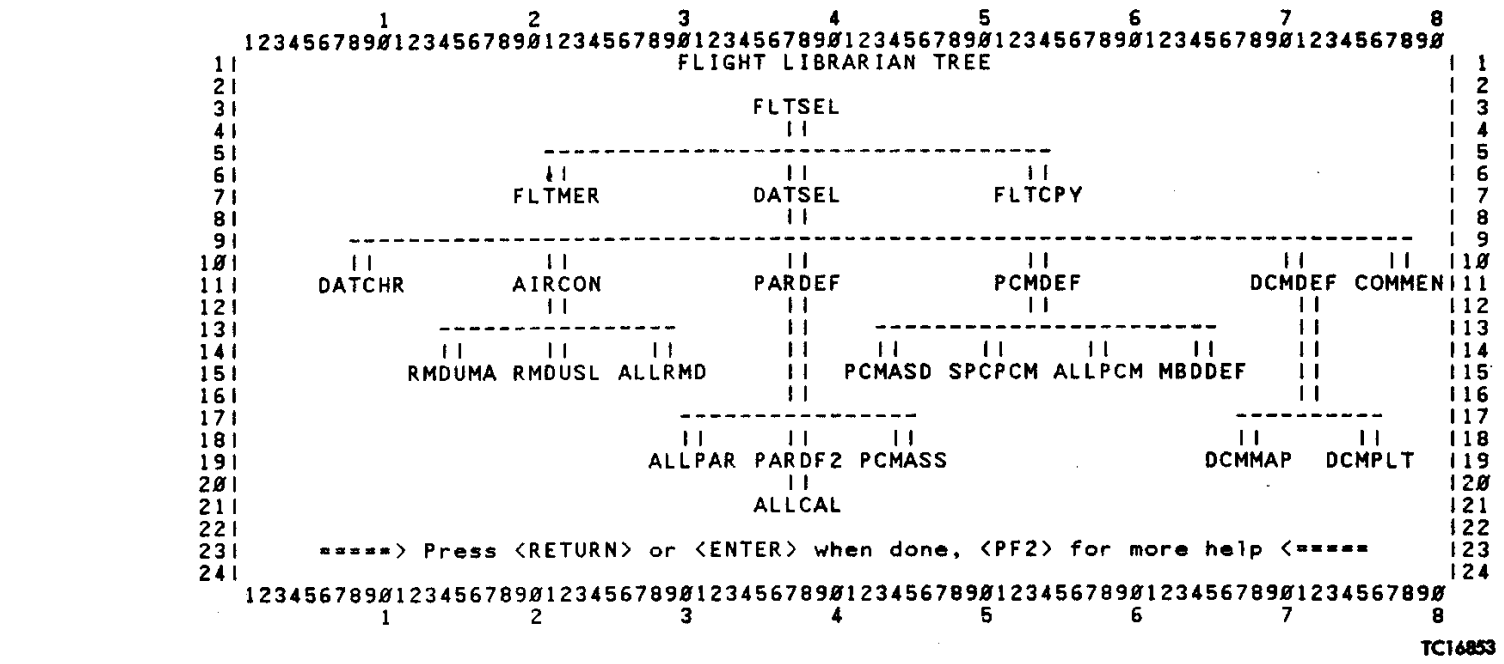

Figure 10. Flight Librarian Menu Tree

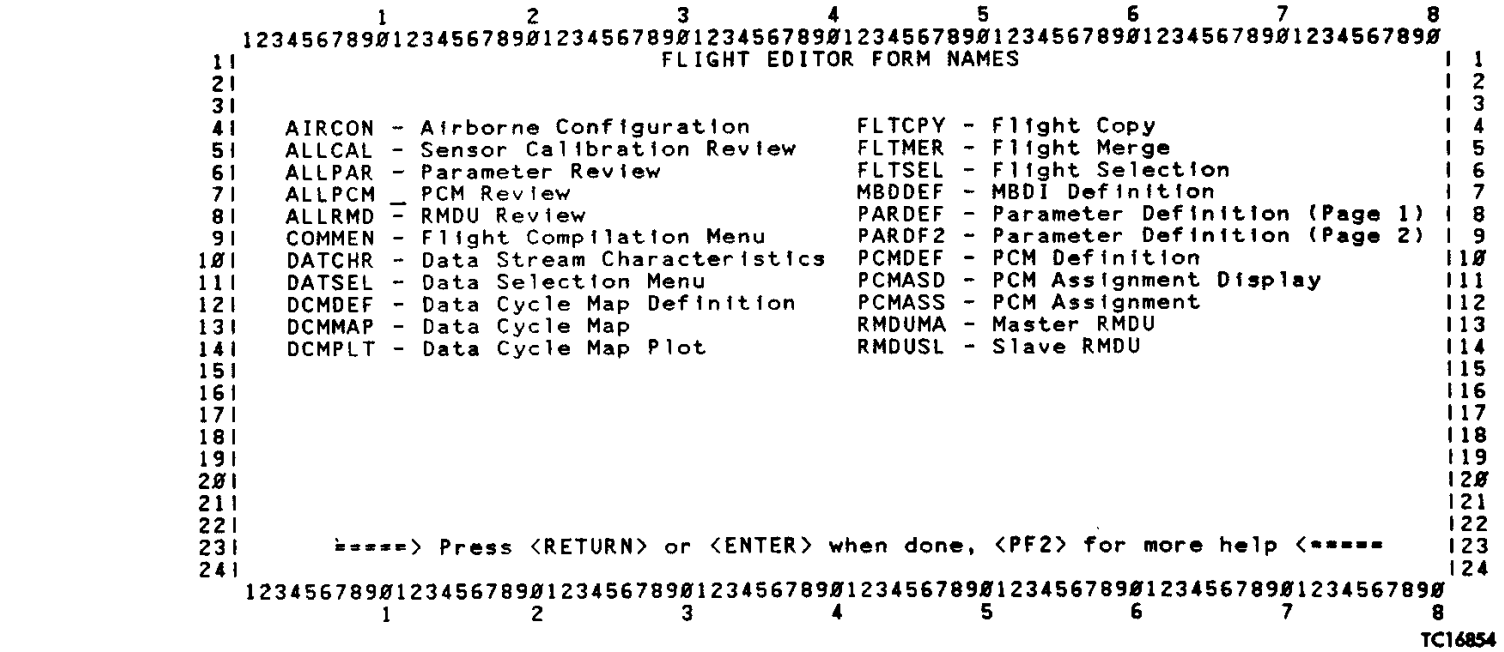

Figure 11. Acronym Help List

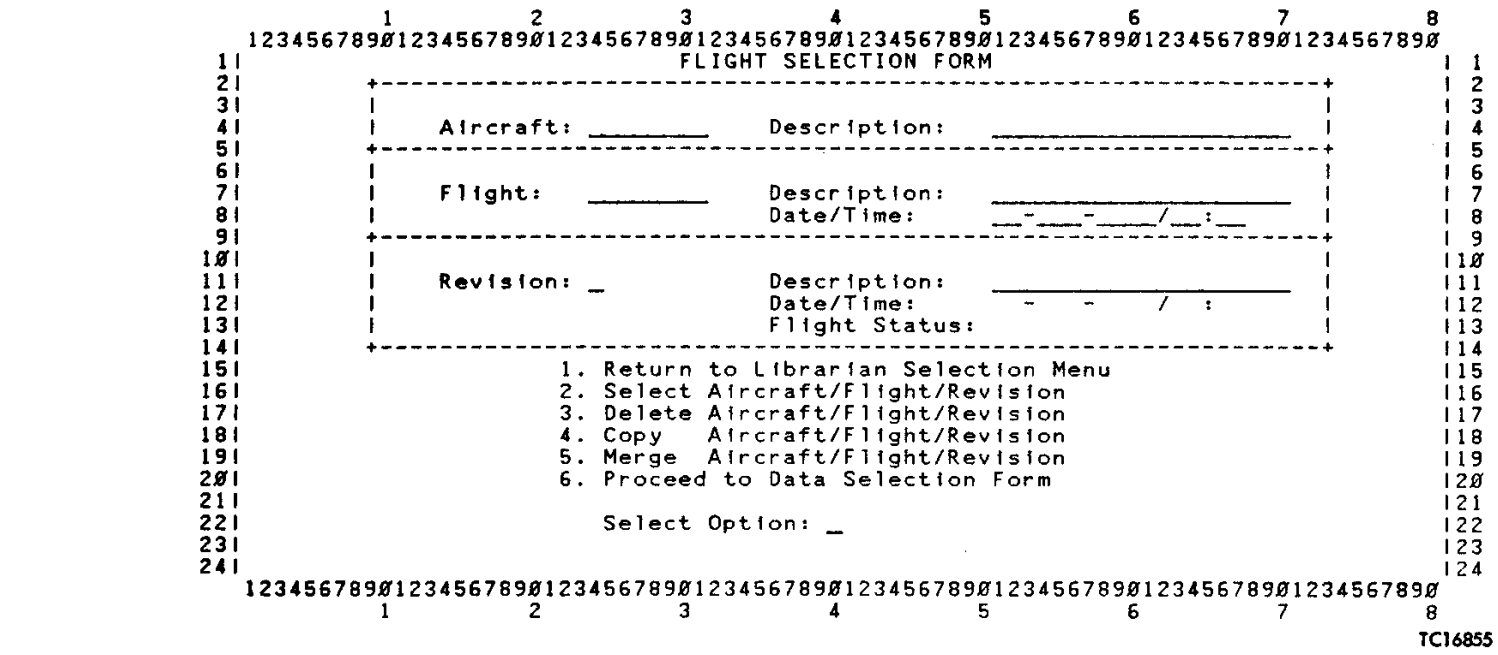

Figure 12. Flight Selection Form

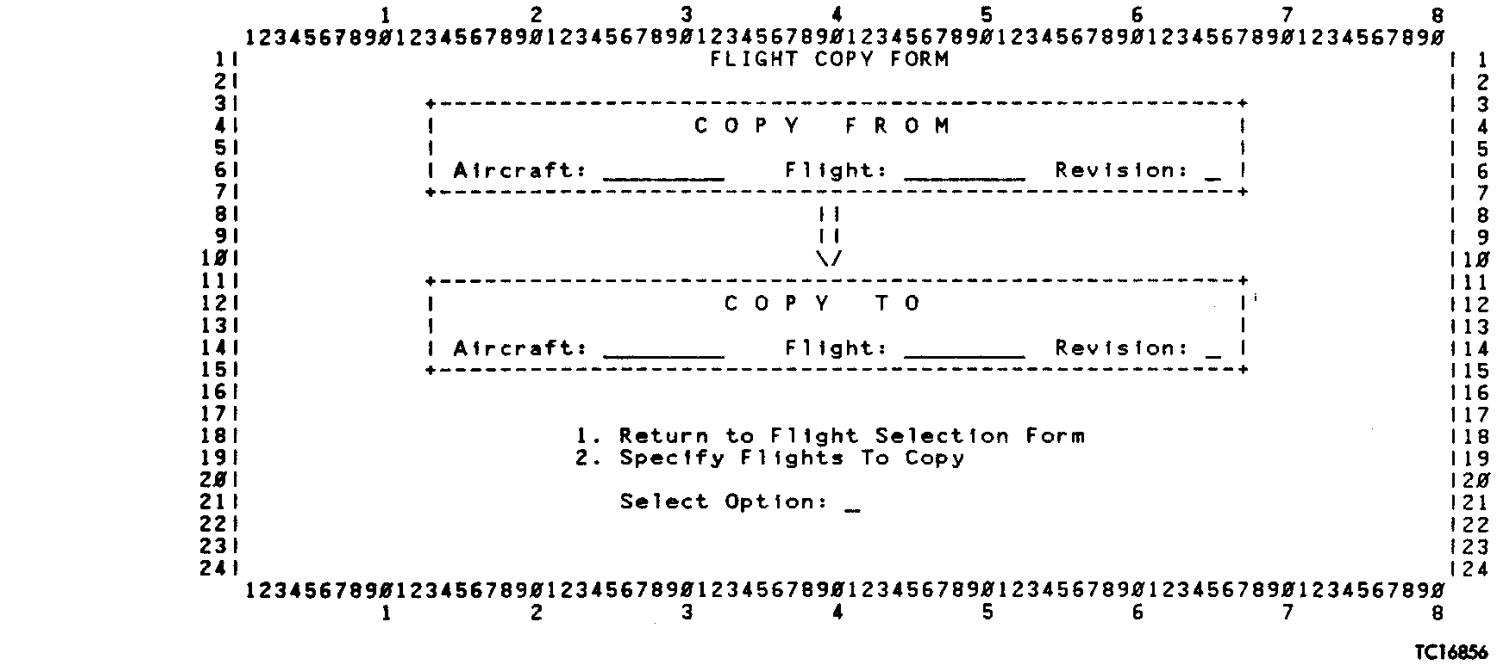

Figure 13. Flight Copy Form