PM5347

AUTOMATIC PROTECTION SWITCHING (APS) SOFTWARE REFERENCE DESIGN

PRELIMINARY

ISSUE 2:FEBUARY 1997

CONTENTS

1 INTRODUCTION

Automatic Protection Switching allows the recovery of a failed channel link between two nodes by switching traffic to a redundant protection channel. APS utilizes the K1 and K2 bytes in the SONET line overhead to implement a bit-oriented protocol for switching operation. The APS architecture consists of selectors, and bridges for each channel and a central controller.

This document describes the design, functionality, and implementation of an APS controller interfaced to PMC S/UNI-PLUS devices. This design is an extension to the Octal PLUS with APS reference design. The Octal PLUS reference with APS design document, PMC-960553, describes the hardware implementation of APS using S/UNI-PLUS devices.

Please refer to PMC-960505 for a tutorial on APS system operation and architecture.

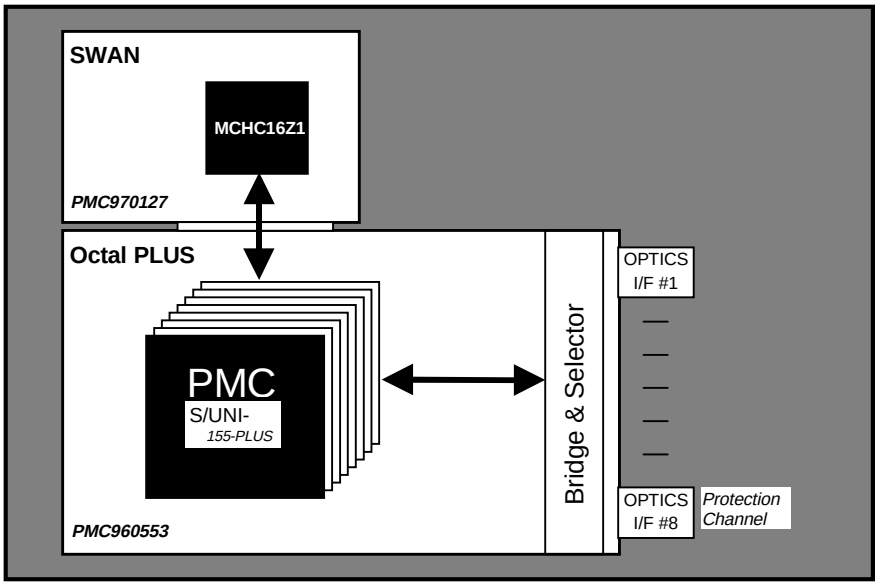
1.1 Audience

This software reference design document has been prepared for customers who are implementing Automatic Protection Switching using PMC-Sierra S/UNI or SONET/SDH family chipsets.

2 DESIGN OVERVIEW

The APS Controller is implemented by an interrupt driven software program written for the Motorola HC16Z1 series microcontroller. The microcontroller sits on the SWAN reference design board (PMC-970127) which is interfaced to the Octal PLUS board. The Octal PLUS consists of eight OC-3 ports, containing APS Bridge and Select hardware on board. Figure 1 depicts the APS setup using the SWAN and the Octal PLUS boards. Channel eight is used as the redundant channel to provide protection for the remaining seven working channels.

Figure 1: Automatic Protection Switching Setup



The APS controller monitors the SONET line overhead K1 and K2 bytes and controls the bridging and selecting of traffic to and from the protection channel. It updates the transmit K1 and K2 bytes according to the status of the local node and requests services from the remote node if needed.

For linear APS, two automatic switch initiation criteria are defined in Bellcore GR-253-CORE. The first criteria is associated with a hard failure and is labeled as Signal Fail (SF). SF is declared when any of the following conditions has been detected: Loss of Signal (LOS), Loss of Frame (LOF), Line Alarm Indication

Signal (AIS-L), or a Line BER threshold exceeding a range of 10^{-3} to 10^{-5}. The second criteria is associated with a soft failure and is called Signal Degrade (SD). SD is declared when a BER exceeding a user-provisionable range of 10^{-5} to 10^{-9} has been detected. After a failure condition due to either SD or SF has been detected, Bellcore GR-253-CORE requires the APS controller to complete an automatic switch within 50 ms. The software design in this application note will meet the 50 ms requirement and takes into account the APS bytes signaling time and software execution time.

The S/UNI-PLUS device provides all the hardware required to detect both SF and SD criteria. For the SF criteria, the S/UNI-PLUS generates a hardware interrupt when any of LOS, LOF, and AIS-L conditions has been declared. The integral BERM block can be programmed to detect a SF BER range of 10^{-4} to 10^{-5}. For the SD criteria, the S/UNI-PLUS's 20-bit BIP counter can used to detect a BER range of 10^{-5} to 10^{-9}. The APS controller polls the BIP counter at a defined interval to check if accumulated BIP error has exceed a maximum threshold calculated based on the BER defined.

In addition, the S/UNI-PLUS supports extraction and insertion of APS K1/K2 bytes and detection of APS byte failure. The APS controller can insert K1 and K2 bytes in the transmit stream by writing to the transmit K1 and K2 registers. The S/UNI-PLUS filters and captures the K1 and K2 bytes allowing them to be read via the receive K1 and K2 registers.

Figure 2 shows the flowchart for the APS controller software program. The software design is implemented as an interrupt driven program. The controller stays in the main loop until it is interrupted by a hardware or software interrupt. The interrupt service routines then perform the APS controller functions.

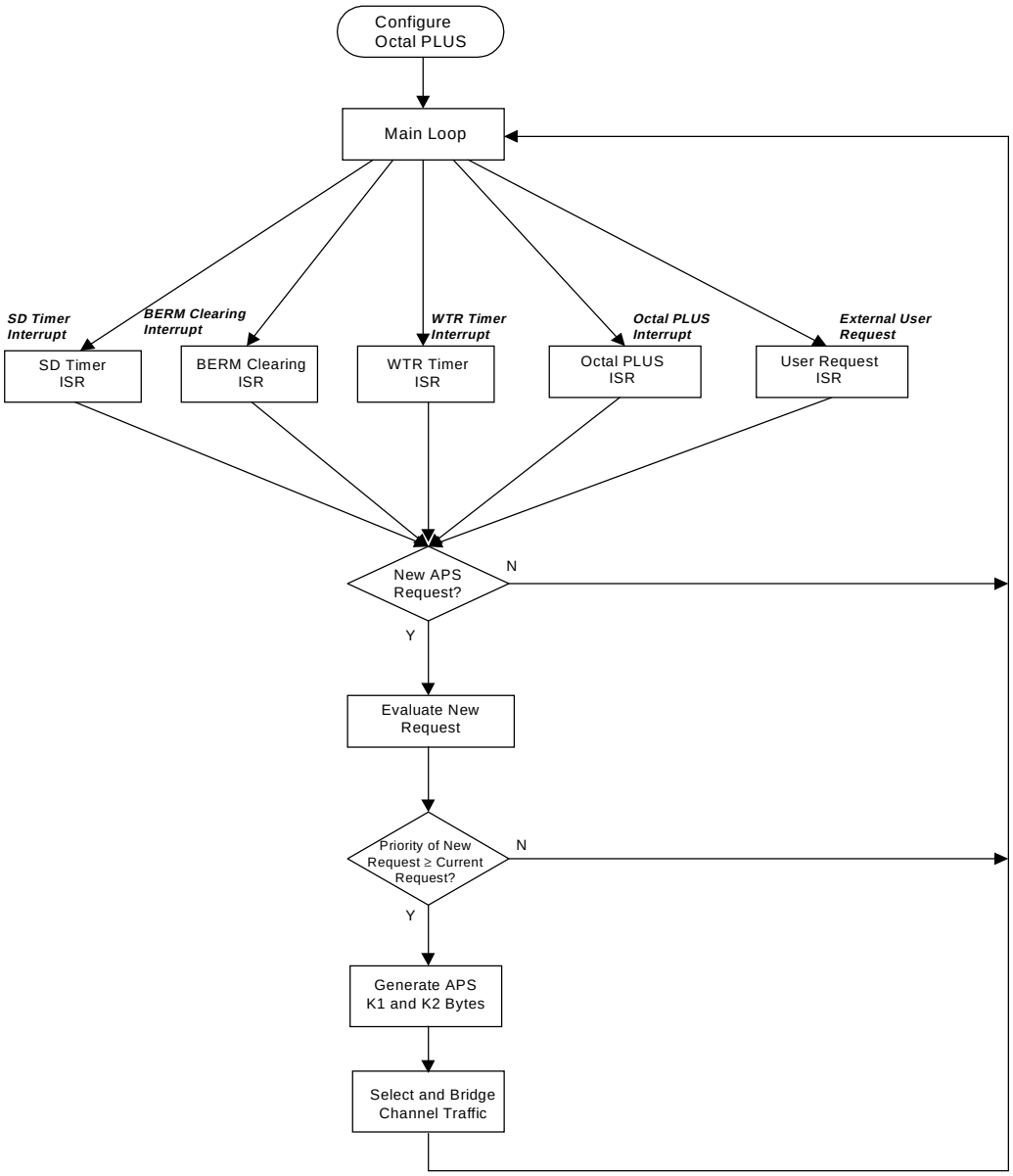
The configuration step sets up the S/UNI-PLUS for BERM detection, enables interrupts to detect LOS, LOF, AIS-L failures. The eighth S/UNI-PLUS is setup to detect change in the received K1/K2 bytes and APS bytes failure. After configuring the Octal PLUS, the program goes into a continuous loop and waits for interrupts to occur.

Four types of interrupts are defined for the controller program:

- SD Timer Software Interrupt
- BERM Timer Software Interrupt
- WTR Timer Software Interrupt
- Octal PLUS Hardware Interrupt

Each of the above four interrupts is serviced by its respective interrupt service routine (ISR). Note, the S/UNI-PLUS BERM block is configured to detect an SF condition. Therefore, this condition is incorporated in the interrupts generated from the Octal PLUS board.

Figure 2: Flowchart of the APS Controller



The SD Timer interrupt service routine polls the BIP count to check if it has exceeded the SD threshold. If the BIP count exceeds the preset threshold, SD condition is declared and a new APS request is initiated.

The BERM Clearing interrupt service routine checks whether a previously declared excessive SF BER has been cleared. The S/UNI-PLUS BERM block does not generate a clearing interrupt when the failure link has recovered from an excessive BER. To solve this problem, the APS controller polls the BERM block to determine whether a clearing threshold has been reached. If clearing has been detected, a new APS request is initiated.

The WTR Timer interrupt service routine keeps track of the wait time required for switching back to a working channel. The WTR Timer is only activated when a clearing condition for SF or SD has been declared. If WTR period has timed out, a new APS request is initiated.

The Octal PLUS interrupt service routine determines the source of the hardware interrupt from the Octal PLUS board. An interrupt from the Octal PLUS board indicates that one of the LOS, LOF, AIS-L, SF (BERM), APS bytes failure, or change in APS bytes conditions have occurred. Upon receiving an interrupt the controller will check the registers of all the S/UNI-PLUS devices and determine the source of the interrupt. It initiates a new APS request based on the result of the check.

At the end of each interrupt service routine, if a new APS request is initiated, the controller evaluates the new APS request and determine their priority level according to Bellcore GR-253-CORE specification. If the priority level of the new request is lower than the current request, the new request is logged for later execution and the controller returns to the main loop. If the new request has a higher priority, new APS K1 and K2 bytes will be generated and written into the transmit K1 and K2 registers on the S/UNI-PLUS. The APS signals are transmitted and received on the protection channel. The protection channel S/UNI-PLUS is responsible for receiving the APS bytes, checking if the bytes are valid, and to transmit the bytes out to the remote node.

New K1 and K2 APS bytes are generated for new requests and are written into the S/UNI-PLUS device to be transmitted over the protection channel. The controller will then set the selectors and bridges corresponding to the current transmit and receive K1 and K2 bytes.

After a protection switch has occurred, the eighth S/UNI-PLUS will select the failure channel line to monitor for line recovery. The protected working channel S/UNI-PLUS will take over the APS signaling duty from the eighth S/UNI-PLUS. The APS signaling will still be done through the protection line.

The user external request functionality has not been discussed so far. The external requests include lockout of protection, forced switch, and manual switch This functionality has not been implemented because this software design is meant for demonstrating an implementation of APS using PMC S/UNI-PLUS devices. The user external request routine does not require any hardware functionality and can be implemented as an extension to this design. The flowchart shows where the external request can be added to the program.

3 FUNCTIONAL DESCRIPTION

3.1 Octal PLUS Configuration

The following two register configurations are required for the seven working channel S/UNI PLUS devices on the Octal PLUS board.

3.1.1 Setting up the BERM Block

The BERM block is setup to monitor Signal Failure (SF) BER conditions of 10^{-4} or 10^{-5}. Table 1 shows the required values for accumulation period and threshold for the BERM block registers, 0x72 to 0x75.

Table 1: Accumulation and Threshold Values for BER

BER	Accumulation Period LSB	Accumulation Period MSB	Threshold LSB	Threshold MSB
10^{-4}	0x34	0x00	0x4D	0x00
10^{-5}	0x90	0x01	0x3E	0x00

3.1.2 Enabling S/UNI-PLUS Interrupts

Table 2 lists all the interrupt bits that need to be set to generate interrupts for the APS controller to detect APS Byte Failure, Change in Received APS Bytes, Loss of Frame, Loss of Signal, Line AIS and excessive BER conditions.

Table 2: Interrupt Enable Registers

Register	Bit	Name	Value	Description
0x10	1	LOFE	1	Enable LOF Interrupt
0x10	2	LOSE	1	Enable LOS Interrupt
0x19	5	LAISE	1	Enable LAIS Interrupt
0x70	0	BERE	1	Enable BERM interrupt
0x70	7	BERTEN	1	Enable BERM Monitoring

3.1.3 Protection Channel S/UNI-PLUS Setting

Table 3 lists the interrupt bits needed to be set for the protection channel S/UNI-PLUS. The APS bytes are sent and received on the protection channel. Thus, only the protection channel S/UNI-PLUS needs to monitor the change in APS bytes and APS byte failures. The transmit K1 byte is set to "0F" to indicate extra channel traffic is being carried on the protection channel and K2 byte is set to "FD" to indicate bidirectional 1:N linear APS.

Table 3: Interrupt Enable Register for the eighth S/UNI-PLUS

Register	Bit	Name	Value	Description
0x0B	6	COAPSE	1	Enable Change in APS bytes Interrupt
0x0B	7	PSBFE	1	Enable APS Failure Byte Interrupt

3.1.4 Octal PLUS Configuration Register

The configuration registers are set to their default values during initialization. For default settings, the APS Control/Status register in the configuration register is set to "00H". At default, the protection channel is used to carry extra channel traffic to and from the eighth S/UNI-PLUS at default. The extra channel traffic is unprotected and will be lost when the protection channel is used for protecting a failed working channel.

3.2 SD Timer ISR

The Signal Degrade (SD) condition is detected and cleared by polling the Line BIP registers of all the S/UNI-PLUS's to check if the accumulated error has exceeded a preset threshold. The polling period or integration period for the SD software timer is set to half of the APS initiation time criterion defined by Bellcore GR-253-CORE. This is explained in the APS application note, PMC-960505.

3.2.1 Detecting SD

To select a proper threshold count, Gaussian statistics are used to ensure 95% probability of detecting an excessive BER. Details regarding this calculation can be found in a application note, programming the Bit Error Rate Monitor (BERM),

on the PMC web site (PMC-950820). Table 4 lists the integration period and the threshold value associated with BER of 10^{-5}, 10^{-6}, and 10^{-7}.

The APS controller checks the BIP count for each S/UNI-PLUS device when it has been interrupted by the SD timer at the end of each integration period. It checks the accumulated errors in BIP registers, 0x1A, 0x1B, and 0x1C. If the polled BIP error exceeds the calculated threshold shown in the table, an SD condition is declared and a new APS request is initiated. If polled BIP error is within threshold set, the routine returns to the program to wait for the end of the next integration period.

Table 4: Integration Period and Threshold Count for SD Timer

BER (SD)	Switching Initiation Time Criteria (sec)	Integration Period (sec)	Threshold Count
10^{-5}	0.1	0.05	62
10^{-6}	1	0.5	62
10^{-7}	10	5.0	62

3.2.2 Clearing SD

After an SD condition has been declared, the APS controller uses the SD Timer for clearing the SD condition. Bellcore GR-253-CORE requires that clearing SD BER value be 1/10 of the declaring SD BER. Table 5 lists the threshold for clearing a SD BER. As soon as the polled accumulated BIP error falls within the threshold, the controller declares SD Clearing and issues a new APS request.

Table 5: SD Clearing Integration Period and Threshold Settings

BER (SD)	Clearing BER required	Switching Initiation Time Criteria (sec)	Integration Period (sec)	Threshold Count
10^{-5}	10^{-6}	1	0.5	62
10^{-6}	10^{-7}	10	5	62
10^{-7}	10^{-8}	83	41	51

3.3 BERM Clearing ISR

The BERM Timer is enabled when BERM block has declared a SF condition. Bellcore GR-253-CORE requires that the clearing SF BER value be 1/10 of the declaring SF BER. After an SF has been declared due to excessive BER, the controller will write new values into the BERM block as shown in table 5. The BERM block will stop generating interrupts only if BER falls below the clearing threshold.

Table 5: BERM Clearing Period and Threshold Settings

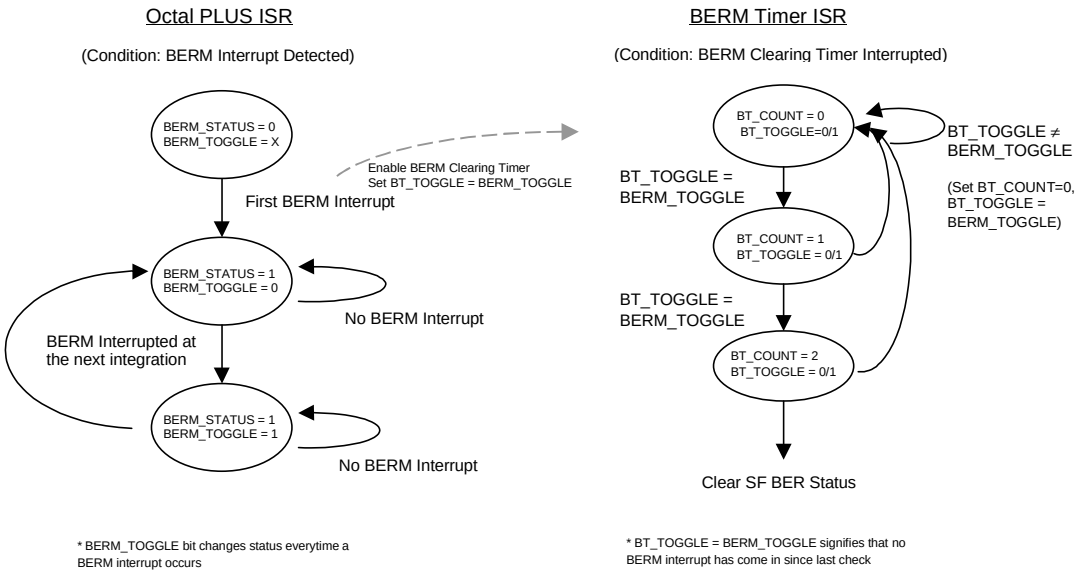
BER (SF)	Clearing BER required	Accumulation Period LSB	Accumulation Period MSB	Threshold LSB	Threshold MSB
10^{-4}	10^{-5}	0x90	0x01	0x3E	0x00
10^{-5}	10^{-6}	0xA0	0x0F	0x3E	0x00

Since the BERM block interrupt bit is cleared when it is read, it is difficult to ascertain if the SF condition has been cleared by just checking that bit within the BERM clearing ISR. Figure 3 shows the process used for clearing the SF status declared by the BERM block implemented in this design. The Octal PLUS Interrupt Service Routine uses a BERM_STATUS flag to indicate the current SF BER status and uses a BERM_TOGGLE flag to indicate if a new interrupt has arrived since the last integration period. The BERM clearing ISR uses a counter, BT_COUNT, to keep track and wait for a preset number of integration periods during which no further BERM interrupts have been detected before clearing the SF status. The BT_TOGGLE flag is used by the BERM ISR to check if interrupts are still being generated by comparing to the BERM_TOGGLE flag. For the first BERM interrupt, BERM_STATUS will be set to 1, BERM_TOGGLE to 0, BT_COUNT to 0, and BT_TOGGLE to 0. The BERM clearing timer will be enabled to interrupt with the same period as the BERM integration period.

The BERM block will continuously generate an interrupt if the detected threshold is higher than the clearing threshold set. The controller toggles BERM_TOGGLE every time it enters the Octal PLUS ISR. When the program enters the BERM Clearing ISR, the controller compares the value of BT_TOGGLE to the BERM_TOGGLE bit. If they are the same, it implies that no interrupt has occurred since the last check. The controller will increment BT_COUNT by one. If the toggle bits are different, it suggests that a new interrupt has occurred. The BT_TOGGLE bit will be set to the current BERM_TOGGLE bit and BT_COUNT will be reset to zero. When the BT_COUNT counter has reached 2, SF clearing is declared and a new APS request is generated.

Bellcore GR-253-CORE specifies that to reduce the chance of rapid switch between channel during SD or SF clearing due to intermittent failure conditions, a maximum delay of 10 seconds can be allotted for clearing. For this example, a delay of two integration period is employed to ensure the failure condition has been cleared. This delay counter can be set to any value within the 10 sec by the user.

Figure 3: BERM Clearing Process



3.4 WTR Timer ISR

The WTR Timer is enabled when a SD and SF condition has recovered and the controller is waiting for a preset duration before initiating to switch back to the working channel. The WTR Timer ISR counts and keeps track of the WTR period and generates a request when the WTR timer has expired. The WTR timer period can be user provisioned from 5 to 12 minutes at 1 minutes interval.

3.5 Octal PLUS Interrupt Service Routine

The Octal PLUS ISR examines all the interrupt status bits in all the S/UNI-PLUS devices to determine the source of the interrupt. For the working channel S/UNI-PLUS, the controller checks the interrupt registers for SF conditions. Table 6 lists all the registers and interrupt bits for SF declaration. By examining register 0x02, the APS controller can tell if an interrupt occurred for RSOP (LOS, LOF), RLOP (AIS-L), APS bytes, or BERM. All the failure conditions with the exception of the BERM block, generates both a detection and clearing interrupt. The ISR generates a new APS request based on the status of the interrupt bit read. The BERM clearing routine within the Octal PLUS ISR is discussed in the BERM clearing ISR section.

Table 6: Interrupt Status Bits for Working Channel S/UNI-PLUS

Register	Bit	Bit Name	Description
0x02	0	RSOPI	Logic one when interrupt occurred in RSOP
0x02	1	RLOPI	Logic one when interrupt occurred in RLOP
0x02	7	MISCI	Logic one when interrupt occurred for BERM
0x11	1	LOFV	Logic one when LOF is declared
0x11	2	LOSV	Logic one when LOS is declared
0x11	4	LOFI	Logic one when change in LOF state occurred
0x11	5	LOSI	Logic one when change in LOS state occurred
0x18	1	LAISV	Logic one when LAIS is declared
0x19	1	LAISI	Logic one when change in AIS state occurred
0x71	0	BERI	Logic one when BER exceeded threshold

For the protection channel S/UNI-PLUS, the controller checks for a change of APS bytes and APS bytes failure as shown in table 7. The received APS bytes are read from register 0x0C for K1 and 0x0D for K2 bytes in the S/UNI-PLUS.

Table 7: Interrupt Status Bits for Protection Channel S/UNI-PLUS

Register	Bit	Bit Name	Description
0x0B	0	PSBFV	Logic one when APS byte failure occurred
0x0B	2	COAPSI	Logic one when received new APS bytes
0x0B	3	PSBFI	Logic one when change in APS byte failure state occurred

All the interrupt status bits, PSBFI, LOFI, LOSI, LAISI, and BERI bits are cleared when read. The controller generates a new APS request for evaluation after determining the source of the interrupt.

3.6 Evaluate New Requests

After receiving requests from the interrupt service routines, the controller determines the priority level of the new request based on table 8. Any interrupts generated due to LOS, LOF, AIS-L, and excessive BER in the BERM block are considered as signal fail (SF) condition. The SD condition is declared from the SD timer routine indicating that BIP errors have exceeded the programmed threshold. Column two lists the requests that are generated locally and column three displays all the requests associated with received K1 byte. When more than one channel generates the same request, the lower channel number has higher priority than the higher channel number.

Table 8: Priority Level for Interrupts Generated

Priority Level	Conditions Detected	Received K1 Bits 8 - 5	Description
Highest		1111	Lockout of Protection
		1110	Forced Switch
	LOS, LOF, AIS-L, BERM Alarm	1101	SF : Higher Priority
		1100	SF : Lower Priority
	BIP > Threshold	1011	SD : Higher Priority
		1010	SD : Lower Priority
		1000	Manual Switch
	SF and SD clearing	0110	Wait-to-restore
		0010	Reverse Request
		0001	Do not revert
	WTR Timed Out		
Lowest		0000	No Request

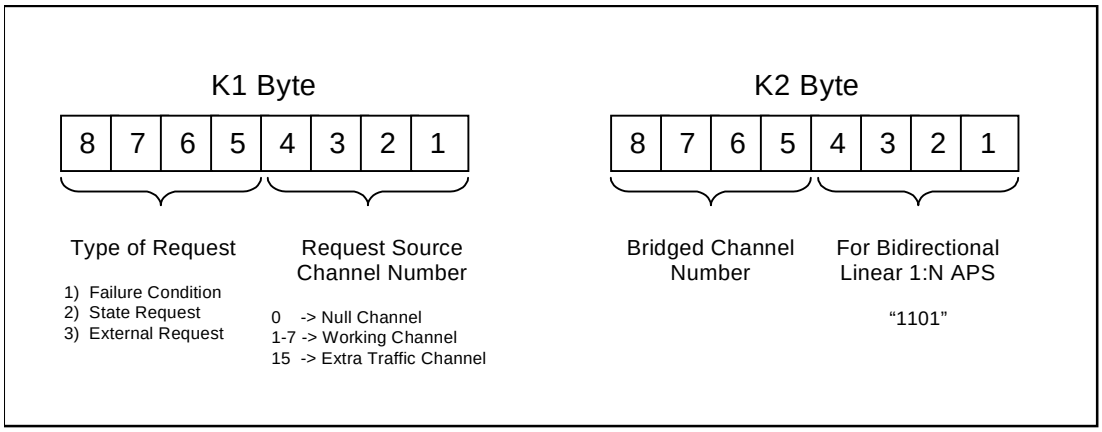
The priority of a new request due to a local failure condition or received K1/K2 bytes is compared with the current local request. A received “Reverse Request”

will not be considered in the comparison since it assumes the same priority as the request that has been previously sent. The new request is ignored if the current local request has a higher priority. If priorities are the same, it implies the channel is in the process of being switched to the protection channel or the status has not changed. If new request has a higher priority, the current local request will be replaced by it.

3.7 Generate K1 and K2 Bytes

APS bytes K1 and K2 are generated for new requests evaluated to be valid. The new APS bytes are written to the transmit K1 and K2 registers, 0x22 and 0x23, on the S/UNI-PLUS to be inserted into the transmit stream. Figure 4 shows the structure of the K1 and K2 bytes.

Figure 4: K1 and K2 Bytes Structure



Bits 5 to 8 in the transmit K1 byte indicates the type of request sent by local site. Table 4 shows the value associated with each specific request. The LSB nibble in the K1 byte indicates the channel requesting the switching. The default request channel will be "1111" when the protection channel is not used for switching. The MSB nibble in K2 byte represents the channel currently bridged on the protection channel. Again, "1111" or extra traffic channel will be the default value. The LSB nibble in the K2 byte will always be set to "1101" for l:N bidirectional APS.

The K1 and K2 values generated are based on three factors: the current received APS bytes, the current transmitting APS bytes, and any local failure or clearing request. The next section lists the APS K1 and K2 bytes generated for both local and remote requests.

3.7.1 Local Detected Conditions and Requests

3.7.1.1 Signal Fail (SF) and Signal Degrade (SD) Request

For SF and SD condition, the generated K1 and K2 byte will be the following:

K1 Byte		K2 Byte	
SF/SD Code	SF/SD Channel #	No Change	1101

3.7.1.2 Signal Failure and Signal Degrade Clearing Request

For revertive switching, when a local condition that caused an automatically initiated switch clears, the Wait-to-Restore (WTR) state is activated. The generated K1 and K2 byte is:

K1 Byte		K2 Byte	
0110	WTR Channel #	No Change	1101

3.7.1.3 Wait-To-Restore Timed Out Request

After the WTR times out, the APS controller will issue “No Request” request on the K1 Byte and requests to switch the protected traffic back to the working channel.

K1 Byte		K2 Byte	
No Request	1111 (Default)	No Change	1101

3.7.2 Remote Received Requests

3.7.2.1 Received SF or SD Request

When the controller evaluates the received SF or SD request to be the highest priority request, the transmit K1 byte will be set to Reverse Request to acknowledge the SF or SD request.

K1 Byte		K2 Byte	
0010	Same as Received K1	Same as K1 Channel #	1101

3.7.2.2 Received Reverse Request

The Reverse Request indicates that the remote node has acknowledged the local switch request. The controller will perform the bridge and select action.

K1 Byte		K2 Byte	
No Change	No Change	Change to Request Channel	1101

3.7.2.3 Received Wait-To-Restore Request

The received “Wait-to-Restore” request indicates that the remote site is initiating a wait period before requesting to switch back to the working channel. The local site does not have to take any actions and should still transmit the same APS bytes.

K1 Byte		K2 Byte	
No Change	No Change	No Change	1101

3.7.2.4 Received No-Request Request

The received “No Request” request indicates that the remote site is ready to switch traffic from the protection channel back to the working channel. The local site will issue a “No Request” to acknowledge the request and bridge the extra channel traffic back to the protection channel.

K1 Byte		K2 Byte	
0000	Same as Received K1	1111	1101

3.8 Select and Bridge Channel Traffic

The selector and bridge are controlled through on board configuration registers of the Octal PLUS. The APS control/status register is located in address x782. All the bits in the APS register can be either read or written.

Table 9: APS Control/Status Register x782

Bit	Bit Name	Description
7	unused	
6	APSS[2]	Control the selector to each of the seven working channels
5	APSS[1]	
4	APSS[0]	
3	unused	
2	APSB[2]	Control the bridge
1	APSB[1]	
0	APSB[0]	"000" => Extra Traffic

The values for the control signals are determined from the receive and the transmit APS K1 and K2 bytes.

Bridging takes place whenever the received K1 bytes requests a bridge unless the request is invalid. The channel to be bridged will be indicated in the lower nibble of the received K1 byte. The upper nibble of the transmit K2 byte will indicate the channel currently bridged on the protection channel.

Selecting takes place when there is a match between the transmitted K1 and received K2 bytes. The controller will select the channel indicated in the APS bytes by writing values into the APS register.

4 IMPLEMENTATION

4.1 Development Tools

The APS controller program is written in FORTH and HC16 Assembly languages using polyFORTH software development system from Forth, Inc. (www.forth.com). Some basic knowledge of FORTH is helpful in understanding the code. FORTH was chosen as the programming environment because it is well suited for code development for microcontrollers such as the HC16 sitting on the SWAN board. FORTH provides easy tools for development and debugging without the need for any extra program tools such as compilers and linkers. The polyFORTH environment is also interactive so there is no need to recompile the code for each change implemented.

Each FORTH program is defined and constructed from words. There are a number of core words that are common in every FORTH, and by using these words more words can be constructed. Routines in FORTH are a collection of words. In contrast to other programming languages like C and FORTRAN, FORTH operates in reverse polish notation (RPN). This requirement arises from the simple architecture of FORTH; built on the concept of a parameter stack and a return stack. This architecture lends itself well to microcontroller environments.

The majority of the APS controller program has been written in assembly to minimize software execution time. The SWAN board uses a 16.78 MHz HC16 microcontroller. Each instruction cycle is 30 ns and each write or read to the Octal PLUS boards takes approximately 200 ns. New APS bytes require three frames to be captured and filtered by the S/UNI-PLUS devices. A complete protection switch requires a total of three APS bytes changes to take place. Actual measurement of switching time was taken from the implemented APS setup as described. The measurement was taken from when a failure has been detected to when the local site has finished selecting the protected line. The measured time was within the Bellcore GR-253-CORE switching requirement of 50 ms.

Appendix A contains the entire code listing in both assembly and forth. Because the assembly code was developed in the FORTH RPN environment, the assembly operand precedes the mnemonic. The '\' character suppresses compilation and is used to add comments to the code. In addition, FORTH comments are also demarcated by '(' and ')'. These are used interchangeably throughout the code.

4.2 Program Structure and Routines

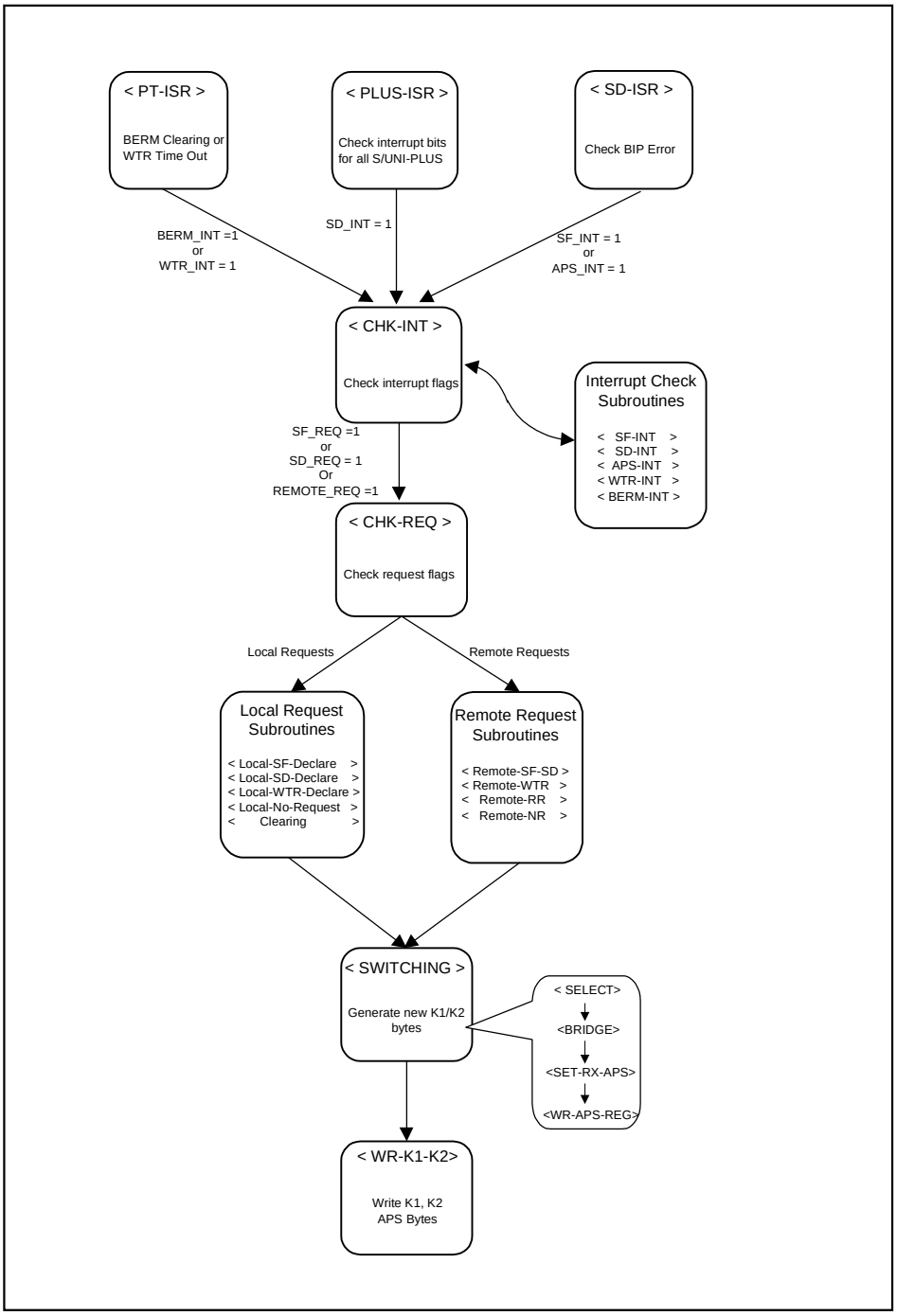
The controller program is implemented by various interrupt service routines as described in the functional description. This section goes through the names and functions of the main routines implemented for the program. The naming convention used in this program for labeling routines is with "<" and ">". Figure 5 shows the overall program structure of the controller software.

All the interrupts generated from the Octal PLUS board are processed by the interrupt service routine, <PLUS-INT>. The interrupt service routines are defined as labels and are defined by the interrupt vector number and the word "EXCEPTION". The <PLUS-INT> ISR checks the interrupts bits of all the S/UNI-PLUS devices when executed. A 500 μs delay is introduced within the ISR before reading the interrupt registers. This ensures that all the interrupts generated from the Octal-PLUS board are detected and cleared when exiting from the ISR. The <PLUS-INT> routine sets either the SF_INT or APS_INT flag to indicate change in SF status or change in APS bytes. The <PLUS-INT> calls the <CHK-INT> routine to check whether the new interrupt warrants a new request.

The SD Timer Routine derives its timing from the IC1 input of the HC16 microcontroller. The SWAN board supplies a 250 μs pulse to the HC16's IC1 port. At 250 μs interval, IC1 generates a software timer interrupt. The SD Interrupt service routines, <SD-INT>, which services the IC1 interrupt, accumulates a time counter (SD_COUNT) until it reaches the integration period of BER detection. At this point, the routine polls the line-BIP counter to check if threshold has exceeded. By the same token, if a SD condition has been previously defined, the <SD-INT> will change its parameters to generate a clearing status when BER reaches a level $1/10^{th}$ of declaring BER. The SD_INT will be asserted when <SD-INT> has detected either SD declaring or clearing condition. Again, the <CHK-INT> will be called to carry out the SD interrupt.

The periodic timer within the HC16 System Integration Module (SIM) is used for two purposes: clearing SF conditions generated by the BERM block and timing out the WTR period. The periodic timer interrupt service routine, <PT-ISR>, either performs BERM clearing action and set BERM_INT flag or WTR time out action and set WTR_INT flag based on the BERM_ENABLE flag. The <WTR> subroutine within <PT-ISR> accumulates the WTR counter until it reaches a pre-set period. The BERM clearing part of the <PT-ISR> performs the BERM clearing process as described in the functional description.

Figure 5: Program Structure



The <CHK-INT> routine checks interrupt status flags, SF_INT, SD_INT, BERM_INT, APS_INT, and WTR_INT and it calls the corresponding subroutines for issuing requests. After a request flag has been set, <CHK-INT> calls <CHK-REQ> to carry out requests. Table 10 shows interrupt check subroutines called as discussed above.

Table 10: Interrupt Check Subroutines

Interrupt Check Subroutines	Action
< SF - INT >	Set SF_REQ flag if interrupt result in a valid request
< SD – INT >	Set SD_REQ flag if interrupt result in a valid request
< APS - INT >	Load Received K1, K2 Bytes, Set REMOTE_REQ flag
< BERM - INT >	Reset BERM Values, calls < SF – INT > to evaluate
< WTR – INT >	Jumps to issue No Request subroutine

The <CHK-REQ> routine checks three request flags: SF_REQ, SD_REQ, and REMOTE_REQ. Based on the current status, <CHK-REQ> determines whether the SF_REQ or SD_REQ is a declaring or a clearing request. For a valid REMOTE_REQ, <CHK-REQ> calls <REMOTE-PRI> to determine the priority of received APS bytes. Once the priority of the remote request has been determined, <CHK-REQ> calls <REMOTE-REQ> to evaluate and carry out requests.

For local requests, <CHK-REQ> either declares a new SF or SD condition clears an existing condition. For declaration, local request subroutines are called to carry out generation of K1/K2 bytes and perform switching. For clearing, <CLEARING> routine is called to check if outstanding failure exist on other channel. <CLEARING> enables the WTR timer to time out the WTR period if no other failures exists else it calls local request subroutines, <LOCAL-SF-DECLARE>, or <LOCAL-SD-DECLARE> to generate a new switching request.

Table 11 lists all the local request subroutines in the controller program. Each of the request subroutines updates TX_K1 and TX_K2 to reflect current status, calls <SWITCHING> subroutine to perform bridging and selecting, and calls <WR-K1- to write K1 and K2 byte to the transmit APS channel.

Table 11: Local Request Subroutines

Local Request Subroutines	Action
< LOCAL – SF – DECLARE >	Set TX_K1 to send “D” as request
< LOCAL – SD – DECLARE >	Set TX_K1 to send “B” as request
< LOCAL – WTR – DECLARE >	Set TX_K1 to send “6” as request
< LOCAL – NO – REQUEST >	Set TX_K1 to send “0” as request

For remote request, <REMOTE-REQ> evaluates the received K1 byte’s upper nibble to determine the request it has received. Based on this, <REMOTE-REQ> calls one of the remote request subroutines listed below in table 12.

Table 12: Remote Request Subroutines

Remote Request Subroutines	Action
< REMOTE – SF – SD >	Set TX_K1 to send Reverse Request (2)
< REMOTE – WTR >	Declare SD,SF if any local SF,SD outstanding
< REMOTE – RR >	Acknowledge Reverse Request, Perform Switch
< REMOTE – NR >	Set TX_K1 to send No Request (0)

The <SWITCHING> routine looks at the current transmitting K1, K2 bytes and received K1, K2 bytes to determine bridging and selecting actions as described in the functional description. The <SWITCHING> routine, calls four subroutines that will update the channel address map after a bridge and a select has taken place, and enable and disable appropriate interrupt bits in each of the S/UNI-PLUS. Section 4.2.2 discusses in detail the addressing convention used for the program.

Table 13: Switching Subroutines

Switching Subroutines	Function
< BRIDGE >	Set up the transmit APS channel, APS_TX, address
< SELECT >	Set up the Octal PLUS logic address map
< SET-RX-APS >	Enable APS byte detection interrupt for the current APS receive (APS_RX) channel
< WR-APS-REG >	Write bridge and select values into Octal-PLUS APS Control and Status Register

4.2.1 Conventions Used

In the APS controller program implemented, SF condition is represented by “D” (higher priority) and SD condition is represented by “B” (higher priority). For carrying extra channel traffic, K1 is set to “0F” and K2 to “FD”.

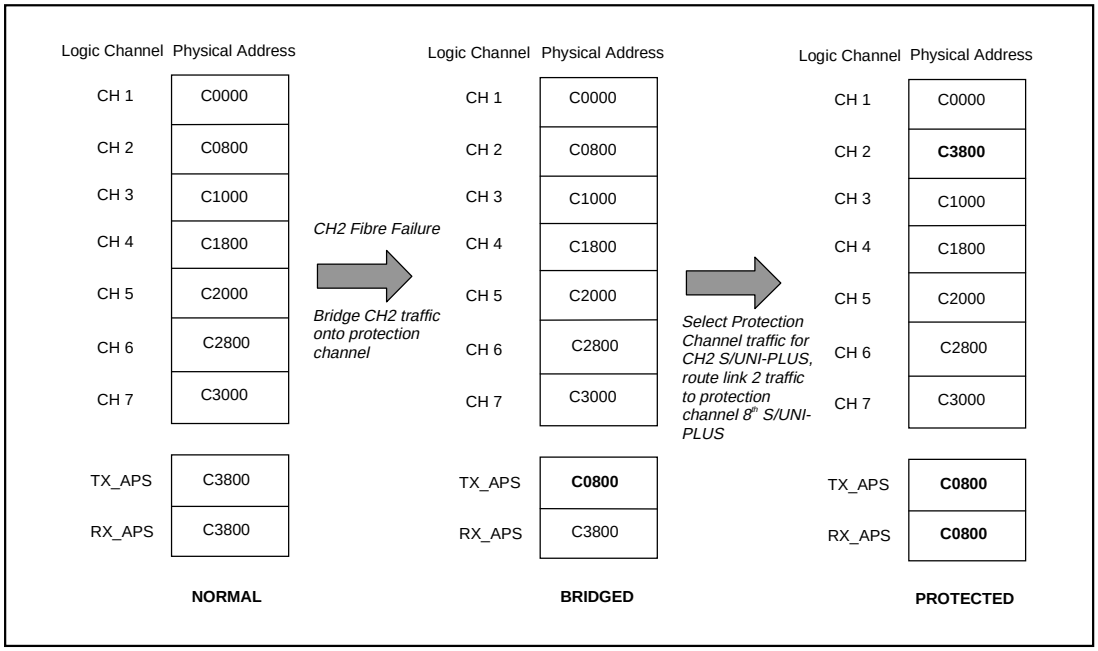
4.2.2 Channel Addressing

After a working channel has been switched to the protection channel, the protection channel S/UNI-PLUS will select the failure channel to listen for recovery and the protected working channel S/UNI-PLUS will perform APS signaling. As a result, the logic channel address need to be changed when a protection switch has occurred. Figure 5 shows the addressing of the Octal PLUS logic channels during a switch.

A total of eight chip selects are used to control and monitor the 8 S/UNI-PLUS devices on the Octal PLUS board. During configuration, address C0000H to C3800H are assigned to the eight chip selects at a spacing of 800H bytes. The left address map in figure 5 shows the normal addressing setup with CH1 at C0000H and proceeds up to C3000 for CH7. As for the APS channel, the addressing is separate into two components: transmit APS channel (APS_TX) and receive APS channel (APS_RX). Both TX_APS and RX_APS are assigned to address C3800H (8th S/UNI-PLUS) during configuration. At this point, both transmit and receive APS K1 and K2 bytes signaling are done through the eighth channel S/UNI-PLUS.

When a channel failure has occurred, the local site bridges the working channel’s transmit traffic over the protection channel. The change in addressing is shown in figure 5 as the middle address map. The TX_APS address has changed to 2nd S/UNI-PLUS’s address (C0800H). This allows the second S/UNI-PLUS now to act as the TX_APS channel. This addressing change is transparent to the remote site, since it continues to receive APS bytes signaling on the protection channel line. The right hand address map shows the addressing after a selection has taken place. RX_APS has been changed to 2nd S/UNI-PLUS since it will now act as the receive APS channel. Logical channel 2 address has been changed to C3800H to reflect that the 8th S/UNI-PLUS is listening to channel 2 line for recovery. When the 8th S/UNI-PLUS has detected that the channel 2 line has recovered, the address map will revert back to the original logic channel to physical address mapping shown in figure 5.

Figure 5: Address Map after Bridge and Selection



5 REFERENCES

- PMC-Sierra, Inc., PM5347 S/UNI-PLUS Data Sheet, Issue 5, September 1996
- PMC-Sierra, Inc., Network Survivability Using Automatic Protection Switching(APS) Over SONET/SDH Point-to-Point & Ring Networks, Issue 2, June 1996
- Bell Communication Research – SONET Transport Systems: Common Generic Criteria, GR-253-CORE, Issue 2, December 1995
- ANSI, Synchronous Optical Network (SONET) Automatic Protection Switching, ANSI T1.105.01-1994
- Fiber Network Survivability, Tsong-Ho Wu
- Motorola Inc., M68HC16 Z Series User's Manual, MC68HC16ZUM/AD, 1997

6 APPENDIX A

Automatic Protection Switching Program Code

```
\  AUTOMATIC PROTECTION SWITCHING (APS) PROGRAM
\
\
\  This program provides APS controller functionality to the
\  Octal-PLUS with APS Reference Design Board
\
\  This program runs on the HC16 on the SWAN reference board
\
\
\
\
\  Edward Chen
\  Applications
\  PMC-Sierra, Inc.
\  January 1998

\ SIM DEFINITIONS

 HOST HEX
 FFA00 CONSTANT SIMMCR   \ Module Configuration Register
 FFA04 CONSTANT SYNCR    \ System Clock Control Register Byte
 FFA11 CONSTANT PORTE    \ Port E data reg.
 FFA15 CONSTANT DDRE     \ Port E data direction reg.
 FFA17 CONSTANT PEPAR    \ Port E pin assignment reg.
 FFA1B CONSTANT PORTF    \ Port F data reg.
 FFA1D CONSTANT DDRF     \ Port F data direction reg.
 FFA1F CONSTANT PFPAR    \ Port F pin assignment reg.




\ SIM DEFINITIONS CONT'D
HOST EDATE HEX
FFA44 CONSTANT CSPAR0            FFA46 CONSTANT CSPAR1
FFA48 CONSTANT CSBARBT           FFA4A CONSTANT CSORBT
FFA4C CONSTANT CSBAR0            FFA4E CONSTANT CSOR0
FFA50 CONSTANT CSBAR1            FFA52 CONSTANT CSOR1
FFA54 CONSTANT CSBAR2            FFA56 CONSTANT CSOR2
FFA58 CONSTANT CSBAR3            FFA5A CONSTANT CSOR3
FFA5C CONSTANT CSBAR4            FFA5E CONSTANT CSOR4
FFA60 CONSTANT CSBAR5            FFA62 CONSTANT CSOR5
FFA64 CONSTANT CSBAR6            FFA66 CONSTANT CSOR6
FFA68 CONSTANT CSBAR7            FFA6A CONSTANT CSOR7
FFA6C CONSTANT CSBAR8            FFA6E CONSTANT CSOR8
FFA70 CONSTANT CSBAR9            FFA72 CONSTANT CSOR9
FFA74 CONSTANT CSBAR10           FFA76 CONSTANT CSOR10

\ INITIALIZE CHIP SELECTS   ( S/UNI-PLUS #1 TO #4 )
 HOST DEFINITIONS HEX
 : INIT-CHIP-SELECTS1
 AAAA CSPAR0 F E!
 02AA CSPAR1 F E!

 FC00 CSBAR2 F E!         ( PLUS-X0, C0000, CH1 )
 5B70 CSOR2  0F E!
 FC08 CSBAR3 F E!         ( PLUS-X1, C0800, CH2 )
 5B70 CSOR3  0F E!
 FC10 CSBAR4 F E!         ( PLUS-X2, C1000, CH3 )
 5B70 CSOR4  0F E!
 FC18 CSBAR5 F E!         ( PLUS-X3, C1800, CH4 )
 5B70 CSOR5  0F E!
 ;
```

```
\ INITIALIZE CHIP SELECTS   ( S/UNI-PLUS #5 TO #8 )
  HOST DEFINITIONS  HEX
 : INIT-CHIP-SELECTS2
 FC20 CSBAR7 F E!          ( PLUS-Y0, C2000, CH5 )
 5B70 CSOR7  0F E!
 FC28 CSBAR8 F E!          ( PLUS-Y1, C2800, CH6 )
 5B70 CSOR8  0F E!
 FC30 CSBAR9 F E!          ( PLUS-Y2, C3000, CH7 )
 5B70 CSOR9 0F E!
 FC38 CSBAR1 F E!          ( PLUS-Y3, C3800, CH8 APS )
 5B70 CSOR1 0F E!

 FFF8 CSBAR10 F E!         ( IRQ3 IACK )
 4841 CSOR10 0F E!   ;


\ SERIAL OUTPUT ROUTINE
 VARIABLE CH-SEND
 HOST HEX
 CREATE ASC
 30 C, 31 C, 32 C, 33 C, 34 C, 35 C, 36 C, 37 C,
 20 C, 0D C, 0A C,


\   0,    1,    2,    3,    4,    5,    6,    7,
\  SP,   CR,   LF,

 HOST DEFINITIONS HEX
 : INIT-SCI
 0037 SCCR0 F E!
 0008 SCCR1 F E!
 ;
\ INITIALIZE PORTE AND PORTF
 HOST DEFINITIONS HEX
 : INIT-PORTF  ( --- ) ( Initializes port F )
 ( MODCLK/PF0=RED LED; IRQ1/PF1=YELLOW LED; IRQ2/PF2=GREEN LED )
 ( rest are set as inputs )
  00  PORTF F EC! ( all zero at port outputs )
  08  PFPAR F EC! ( all I/O , CONFIGURE IRQ3 )
  07  DDRF  F EC! ( PF0,1,2 = o/p ; rest i/p )
 ;

 HOST HEX
 : INIT-PORTE  ( --- ) ( Initializes port E  )
 EF  PORTE F EC! ( all ones at port outputs )
 10  PEPAR F EC! ( PE five for xilinx, all I/O except PE4 - )
 E2  DDRE  F EC! ( PE1,PE5,PE6,PE7 O/P, rest I/O )
 ;
 \ on-chip RAM                                    ( 30 Nov 1997)
 HOST DEFINITIONS HEX
 FFB00 CONSTANT  RAMMCR
 FFB04 CONSTANT  RAMBAH
 FFB06 CONSTANT  RAMBAL

 HOST DEFINITIONS HEX
 : INIT-ON-CHIP-RAM  ( --- )
 ( initialize on chip ram to sit in FF0000 TO FF03FF )
 ( 0000 RAMMCR F E! ) ( unlock base addres registers )
 00FF RAMBAH F E! ( base address high word       )
 0000 RAMBAL F E! ( base address low  word       )
 0800 RAMMCR F E! ( lock RAM base address so it can not be ch)
 400 0 DO 0 I F E! 2 +LOOP ;
```

```
\ GENERAL PURPOSE TIMER DEFINITIONS
HOST EDATE HEX
FF900 CONSTANT GPTMCR      FF904 CONSTANT GPTICR
FF906 CONSTANT DDRGP       FF907 CONSTANT PDR
FF908 CONSTANT OC1M        FF90A CONSTANT TCNT
FF90C CONSTANT PACTL       FF90D CONSTANT PACNT
FF90E CONSTANT TIC1        FF910 CONSTANT TIC2
FF912 CONSTANT TIC3        FF914 CONSTANT TOC1
FF916 CONSTANT TOC2        FF918 CONSTANT TOC3
FF91A CONSTANT TOC4        FF91C CONSTANT TI4/O5
FF91E CONSTANT TCTL1       FF91F CONSTANT TCTL2
FF920 CONSTANT TMSK1       FF921 CONSTANT TMSK2
FF922 CONSTANT TFLG1       FF923 CONSTANT TFLG2
FF924 CONSTANT CFORC       FF925 CONSTANT PWMC
FF926 CONSTANT PWMA        FF927 CONSTANT PWMB

\ GENERAL PURPOSE TIMER DEFINITIONS CONT'D
HOST EDATE HEX
( GENERAL-PURPOSE TIMER MODULE REGISTERS )
FF928 CONSTANT PWMCNT
FF92A CONSTANT PWMBUFA     FF92B CONSTANT PWMBUFB
FF92C CONSTANT PRESCL




\ INITIALIZE GPT FOR IC1 INPUT
 HOST EDATE HEX
 CODE INIT-GPT
  D ,E ,X ,Y ,Z ,K PSHM
   0F # LDAB TBEK  0083 # LDD GPTMCR STD \ interr. arbitration
   0450 # LDD GPTICR STD      \ vectr base addr 50 , req. level
   0001 # LDD TCTL1 STD       \ IC1 on RISING edge only
   0106 # LDD TMSK1 STD       \ enable IC1, /256
   0000 # LDD TFLG1 STD       \ clear IC1 flag
   00 # LDAB TBEK
  D ,E ,X ,Y ,Z ,K PULM
 NEXT



\ ROUTINE FOR SELECTING CURRENT S/UNI-PLUS

 VARIABLE NDEV
 HOST HEX DEFINITIONS
 : DEV
   CR ." DEVICE SET TO ADDR: " NDEV @ U. CR
 ;
 : SET
   800 * 800 - NDEV !
 ;
```

```
\ SHOWS S/UNI-PLUS # 1 REGISTERS

 HOST DEFINITIONS HEX
: Z1 ( --- ) ( read all S/UNI-PLUS registers )
 CR ."                  S/UNI-PLUS REGISTERS          "     CR
   ."          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F "
CR   80 0 DO       I 2 U.R  ."  --- "
     10 0 DO       I J + C EC@ ."  " 2 U.R      LOOP
                                      CR  10 +LOOP  ;
 HOST HEX
 : WX0 ( b a --- )  C EC! ;
 HOST HEX  : RX0 ( a --- ) C EC@ . ;
 HOST HEX : RMX0 ( m a --- ) C EC@ AND . ;
 HOST HEX : RSX0 ( A --- ) C EC@ ;


\ SHOWS S/UNI-PLUS # 2 REGISTERS

 HOST DEFINITIONS HEX
: Z2 ( --- ) ( read all S/UNI-PLUS registers )
 CR ."                  S/UNI-PLUS REGISTERS          "     CR
   ."          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F "
CR   80 0 DO       I 2 U.R  ."  --- "
     10 0 DO       I J + 800 + C EC@ ."  " 2 U.R      LOOP
                                      CR  10 +LOOP  ;
 HOST
 : WX1 ( b a --- )
 800 + C EC! ;
 HOST
 : RX1 ( a --- ) 800 + C EC@ . ;


\ SHOWS S/UNI-PLUS # 3 REGISTERS

 HOST DEFINITIONS HEX
: Z3 ( --- ) ( read all S/UNI-PLUS registers )
 CR ."                  S/UNI-PLUS REGISTERS          "     CR
   ."          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F "
CR   80 0 DO       I 2 U.R  ."  --- "
     10 0 DO       I J + 1000 + C EC@ ."  " 2 U.R      LOOP
                                      CR  10 +LOOP  ;
 HOST
 : WX2 ( b a --- )
 1000 + C EC! ;
 HOST
 : RX2 ( a --- ) 1000 + C EC@ . ;


\ SHOWS S/UNI-PLUS # 4 REGISTERS

 HOST DEFINITIONS HEX
: Z4 ( --- ) ( read all S/UNI-PLUS registers )
 CR ."                  S/UNI-PLUS REGISTERS          "     CR
   ."          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F "
CR   80 0 DO       I 2 U.R  ."  --- "
     10 0 DO       I J + 1800 + C EC@ ."  " 2 U.R      LOOP
                                      CR  10 +LOOP  ;
 HOST
 : WX3 ( b a --- )
 1800 + C EC! ;
 HOST
 : RX3 ( a --- ) 1800 + C EC@ . ;
```

```
\ SHOWS S/UNI-PLUS # 5 REGISTERS

 HOST DEFINITIONS HEX
: Z5 ( --- ) ( read all S/UNI-PLUS registers )
 CR ."                   S/UNI-PLUS REGISTERS          "     CR
   ."          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F "
CR   80 0 DO       I 2 U.R  ."  --- "
     10 0 DO       I J + 2000 + C EC@ ."  " 2 U.R      LOOP
                                      CR  10 +LOOP  ;
 HOST
 : WY0 ( b a --- )
 2000 + C EC! ;
 HOST
 : RY0 ( a --- ) 2000 + C EC@ . ;


\ SHOWS S/UNI-PLUS # 6 REGISTERS

 HOST DEFINITIONS HEX
: Z6 ( --- ) ( read all S/UNI-PLUS registers )
 CR ."                   S/UNI-PLUS REGISTERS          "     CR
   ."          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F "
CR   80 0 DO       I 2 U.R  ."  --- "
     10 0 DO       I J + 2800 + C EC@ ."  " 2 U.R      LOOP
                                      CR  10 +LOOP  ;
 HOST
 : WY1 ( b a --- )
 2800 + C EC! ;
 HOST
 : RY1 ( a --- ) 2800 + C EC@ . ;


\ SHOWS S/UNI-PLUS # 7 REGISTERS

 HOST DEFINITIONS HEX
: Z7 ( --- ) ( read all S/UNI-PLUS registers )
 CR ."                   S/UNI-PLUS REGISTERS          "     CR
   ."          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F "
CR   80 0 DO       I 2 U.R  ."  --- "
     10 0 DO       I J + 3000 + C EC@ ."  " 2 U.R      LOOP
                                      CR  10 +LOOP  ;
 HOST
 : WY2 ( b a --- )
 3000 + C EC! ;
 HOST
 : RY2 ( a --- ) 3000 + C EC@ . ;


\ SHOWS S/UNI-PLUS # 8 REGISTERS

 HOST DEFINITIONS HEX
: Z8 ( --- ) ( read all S/UNI-PLUS registers )
 CR ."                   S/UNI-PLUS REGISTERS          "     CR
   ."          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F "
CR   80 0 DO       I 2 U.R  ."  --- "
     10 0 DO       I J + 3800 + C EC@ ."  " 2 U.R      LOOP
                                      CR  10 +LOOP  ;
 HOST
 : WY3 ( b a --- )
 3800 + C EC! ;
 HOST
```

```
 : RY3 ( a --- ) 3800 + C EC@ . ;


\ SHOWS CURRENT S/UNI-PLUS REGISTERS

HOST DEFINITIONS  HEX EDATE
: ZZ ( --- ) ( read all S/UNI-PLUS registers )
 CR CR ." S/UNI-PLUS CHANNEL " NDEV @ 800 + 800 / .
 CR ."                  S/UNI-PLUS REGISTERS          "     CR
   ."          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F "
CR   80 0 DO       I 2 U.R  ."  --- "
     10 0 DO       I J + NDEV @ + C EC@ ."  " 2 U.R      LOOP
                                        CR  10 +LOOP  ;
 HOST HEX : WDZ ( b a --- ) NDEV @ + C EC! ;
 HOST HEX : RDZ ( a --- ) NDEV @ + C EC@ . ;
 HOST HEX : RDMZ ( m a --- ) NDEV @ + C EC@ AND . ;
 HOST HEX : RDSZ ( a --- ) NDEV @ + C EC@ ;


\ DISABLE AND ENABLE INTERRUPTS

HOST DEFINITIONS  HEX
 CODE EI ( --- ) \ ENABLE INTERRUPTS
 FF1F # ANDP  \ CPU lowest priority , everybody disturbs
 NEXT
 CODE DI ( --- ) \ DISABLE INTERRUPTS
 00C0 # ORP   \ CPU highest priority , nobody can disturb
 NEXT
```

```
\ STATUS MONITOR                                  15:39 12-01-97
HEX DEFINITIONS HEX
: STAT
 CR CR ." S/UNI-PLUS CHANNEL " NDEV @ 800 + 800 / .
CR
." SUNI_INTR (02) " FF 02 RDMZ ."     RLOPI: " 02 02 RDSZ AND 0
 > ABS .  ."     RSOPI: " 01 02 RDSZ AND 0 > ABS . CR
." APS_STATS (0B) " 1D 0B RDMZ CR
." RSOP_INTR (11) " 7F 11 RDSZ AND DUP DUP . ."     LOSV : " 04
AND 0 > ABS .  ."     LOFV: " 02 AND 0 > ABS . CR
." RLOP_INTR (19) " 0F 19 RDMZ CR
." RLOP_STAT (18) " 03 18 RDMZ ."     LAISV: " 02 18 RDSZ AND 0
 > ABS . CR
." BERM_INTR (71) " 01 71 RDMZ CR
CR ;

\ PERFORMANCE MONITORING                          12:23 12-01-97
HOST DEFINITIONS HEX

: PMON
 CR CR ." S/UNI-PLUS CHANNEL " NDEV @ 800 + 800 / .
CR 00 00 WDZ
." RSOP_BIP  = " 13 RDZ 12 RDZ CR
." RLOP_BIP  = " 0F 1C RDMZ 1B RDZ 1A RDZ CR
." RLOP_FEBE = " 0F 1F RDMZ 1E RDZ 1D RDZ CR
." RPOP_BIP  = " 39 RDZ 38 RDZ CR
." RPOP_FEBE = " 3B RDZ 3A RDZ CR
." RCELL_CNT = " 1F 5B RDMZ 5A RDZ 59 RDZ CR
```

```
." TCELL_CNT = " 1F 66 RDMZ 65 RDZ 64 RDZ CR
CR
;

\ INIT-PROGRAM                                    12:23 12-01-97
HOST DEFINITIONS  HEX
 0 17 THRU
HOST DEFINITIONS  HEX
 : INIT-MICRO
 DI INIT-ON-CHIP-RAM
 INIT-CHIP-SELECTS1
 INIT-CHIP-SELECTS2
 INIT-PORTF
 INIT-PORTE
 EI
 ;



\
```

```
\  REGISTER DEFINITIONS FOR S\UNI-PLUS
HOST DEFINITIONS  HEX
 0000 CONSTANT  MASTER_RESET              ( 0x00 )
 0002 CONSTANT  MASTER_INTERRUPT          ( 0x02 )

 0B   CONSTANT  APS_CONTROL_STATUS        ( 0x0B )
 0C   CONSTANT  RECEIVE_K1                ( 0x0C )
 0D   CONSTANT  RECEIVE_K2                ( 0x0D )
 10   CONSTANT  RSOP_CONTROL              ( 0x10 )
 11   CONSTANT  RSOP_STATUS               ( 0x11 )
 19   CONSTANT  RLOP_INTERRUPT            ( 0x19 )
 1A   CONSTANT  RLOP_BIP_LSB              ( 0x1A )
 1B   CONSTANT  RLOP_BIP_ISB              ( 0x1B )
 1C   CONSTANT  RLOP_BIP_MSB              ( 0x1C )
 20   CONSTANT  TLOP_CONTROL              ( 0x20 )

\  PLUS REG'S CONT
HOST DEFINITIONS  HEX
 22   CONSTANT  TRANSMIT_K1               ( 0x22 )
 23   CONSTANT  TRANSMIT_K2               ( 0x23 )

 70   CONSTANT  BERM_CONTROL              ( 0x70 )
 71   CONSTANT  BERM_INTERRUPT            ( 0x71 )
 72   CONSTANT  BERM_LINE_BIP_LSB         ( 0x72 )
 73   CONSTANT  BERM_LINE_BIP_MSB         ( 0x73 )
 74   CONSTANT  BERM_LINE_BIP_THR_LSB     ( 0x74 )
 75   CONSTANT  BERM_LINE_BIP_THR_MSB     ( 0x75 )
```

```
 82   CONSTANT  OCTAL_APS_CONTROL          ( 0x82 )



\  APS VARIABLE LIST                              ( 02 Dec 1997)
 HOST HEX
 VARIABLE PLUS_LOS  9 ALLOT        \ LOS STATUS
 VARIABLE PLUS_LOF  9 ALLOT        \ LOF STATUS
 VARIABLE PLUS_AISL 9 ALLOT        \ AISL STATUS
 VARIABLE PLUS_BERM 9 ALLOT        \ BERM STATUS
 VARIABLE SF_VAL 9 ALLOT
 VARIABLE SD_VAL 9 ALLOT

 VARIABLE SF_INT      VARIABLE SD_INT    \ INTERRUPT FLAGS
 VARIABLE BERM_INT    VARIABLE WTR_INT
 VARIABLE APS_INT

 VARIABLE SF_REQ      VARIABLE BERM_REQ  \ REQUESTS FLAGS
 VARIABLE SD_REQ      VARIABLE WTR_REQ

\ APS VARIABLE LIST CONT'D                        10:39 21/11/97
 HOST HEX
 VARIABLE CUR_REQ_PRI   VARIABLE NEW_REQ_PRI  \ PRIORITY VARS
 VARIABLE REQ_CHAN

 VARIABLE BERM_ENABLE   VARIABLE BERM_TOGGLE  \ BERM VARIABLES

 VARIABLE TEMP1         VARIABLE TEMP2
 VARIABLE TEMP3         VARIABLE TEMP

 VARIABLE TX_K1         VARIABLE TX_K2     \ STORES K1/K2 BYTES
 VARIABLE RX_K1         VARIABLE RX_K2
 VARIABLE SF_SUM
 VARIABLE REMOTE_REQ    VARIABLE REMOTE_CMD
 VARIABLE APS_REG

\ APS VARIABLE LIST CONT'D                        ( 01 Dec 1997)
  VARIABLE SD_COUNT         VARIABLE SD_LIMIT
  VARIABLE BIP_COUNT        VARIABLE SD_THR
  VARIABLE SD_CLEAR_WAIT
  VARIABLE SD_CLEAR_CNT  9 ALLOT

  VARIABLE PICRV         VARIABLE  PITRV
  VARIABLE WTR_LIMIT     VARIABLE WTR_COUNT
  VARIABLE BT_FLAG
  VARIABLE BT_COUNT     VARIABLE BT_LIMIT
  VARIABLE BT_TOGGLE
  VARIABLE SD_CHK       VARIABLE SF_CHK
  VARIABLE CLEAR_CHAN
  VARIABLE CNT          VARIABLE PT_USED
  VARIABLE PLUS_ADDR 20 ALLOT
  VARIABLE APS_RX       VARIABLE APS_TX
\ INITIALIZE S/UNI-PLUS REGISTERS TO DETECT ERROR

HOST DEFINITIONS HEX
 : INIT-APS-REG
 CR ." INITIALIZING APS" CR
    06 RSOP_CONTROL           WDZ
 ( BERM SETUP TO DETECT 1e-4 BER  )
    81 BERM_CONTROL           WDZ
    34 BERM_LINE_BIP_LSB      WDZ
    00 BERM_LINE_BIP_MSB      WDZ
```

```
    4D BERM_LINE_BIP_THR_LSB  WDZ
    00 BERM_LINE_BIP_THR_MSB  WDZ
    20 TLOP_CONTROL           WDZ
;


\ INITIALIZE REGISTER FOR APS SIGNALLING AND DETECTION
HOST DEFINITIONS HEX
 : INIT-APS-REGA
 CR ." INITIALIZING APS SUNI-PLUS " CR
    40 APS_CONTROL_STATUS     WDZ
    20 TLOP_CONTROL           WDZ
 ;

\ PHYSICAL PERMENENT ADDRESS OF S/UNI-PLUS
 HOST HEX
 CREATE ADDR_MAP
 0000 , 0000 ,  800 , 1000 , 1800 , 2000 ,
 2800 , 3000 , 3800 ,



 \ INITIALIZE VARIABLES
 HOST HEX DEFINITIONS
  : INIT-APS-VAR1
   8 0 DO 0 PLUS_LOS  I + !  0 PLUS_LOF I + !
          0 PLUS_BERM I + !  0 PLUS_AISL I + !
          0 SF_VAL    I + !  0 SD_VAL  I + !
          0 SD_CLEAR_CNT I + !  LOOP
   00 SF_INT !
   00 APS_INT !    00 WTR_INT !
   00 BERM_TOGGLE !  00 BERM_INT !
   00 TEMP1 ! 00 TEMP2 ! 00 TEMP3 !
   00 REMOTE_CMD !
   00 REMOTE_REQ !
   00 CUR_REQ_PRI !
   00 APS_REG C!
   ;
 \ INITIALIZE VARIABLE CONT'D
 HOST DEFINITIONS HEX
 : INIT-APS-VAR2
   0F TX_K1 C!         FD TX_K2 C!
   00 SF_SUM C!
   00 SF_REQ ! 00 BT_COUNT !
   00 SD_REQ !
   00 RX_K1 C! 00 RX_K2 C!
   01 REQ_CHAN C!
   00 NEW_REQ_PRI C!
   00 TEMP C!
   00 SF_CHK !  00 SD_CHK !
   00 PT_USED !      00 CLEAR_CHAN !
   9 0 DO -800 800 I * + PLUS_ADDR I 2 * + ! LOOP
   00 PLUS_ADDR ! 3800 APS_TX ! 3800 APS_RX !
   ;
 \ INIT-APS                                        ( 05 Dec 1997)
 HOST DEFINITIONS HEX
 : INIT-APS
  INIT-APS-VAR1
  INIT-APS-VAR2
  INIT-APS-REG
 ;

 HOST DEFINITIONS HEX
```

```
 : INIT-VARS
  INIT-APS-VAR1
  INIT-APS-VAR2
 ;



 \ DISPLAY APS STATUS
 HOST DEFINITIONS HEX
 : SHOW-OCTAL-STATUS
 CR
 ." APS CONTROL STATUS REGISTER [ SELECT | BRIDGE ]:  "
                                  APS_REG C@ U. CR
 CR
 ." TRANSMIT K1 BYTE: " TX_K1 C@ U.
 CR ."          K2 BYTE: " TX_K2 C@ U.
 CR CR
 ." RECEIVED K1 BYTE: " RX_K1 C@ U. CR
 ."          K2 BYTE: " RX_K2 C@ U.
 CR
 CR
 ;

\ INITIALIZE IC1 USE FOR SD DETECTION

   HOST DEFINITIONS HEX
   CODE INIT-SD
    D ,E ,X ,Y ,Z ,K PSHM
    00 # LDAB TBEK
    00C8 # LDD SD_LIMIT STD         ( 0.05 SEC, BER => -5 )
    0000 # LDD SD_COUNT STD
    003E # LDD SD_THR STD           ( 62, for -5, -6, -7 )
    0000 # LDD SD_INT STD
    0014 # LDAB SD_CLEAR_WAIT STAB  ( WAIT 5 SEC TO CLEAR)
    D ,E ,X ,Y ,Z ,K PULM
   NEXT



\ PERIODIC INTERRUPT TIMER ROUTINE
HOST DEFINITIONS HEX
LABEL <E-PT>              \ ENABLE PERIODIC TIMER
 00 # LDAB TBEK
 PICRV LDE  0F # LDAB TBEK  PICR STE
 00 # LDAB TBEK
 PITRV LDE  0F # LDAB TBEK  PITR STE
 00 # LDAB TBEK
 RTS

HOST DEFINITIONS HEX
LABEL <D-PT>              \ DISABLE PERIODIC TIMER
 0F # LDAB TBEK
 0040 # LDD PICR STD
 00 # LDAB TBEK
 RTS
\ <CLEAR-BERM>                                    ( 06 Jan 1998)
\ RESET THRESHOLD VALUES TO 1/10 OF ORIGINAL
 HOST DEFINITIONS HEX
 LABEL <CLEAR-BERM>                ( RECOVERS BER 1e-4 )
  D ,E ,X ,Y ,Z ,K PSHM
  00 # LDAB TBEK TBYK 0C # LDAB TBXK
  PLUS_ADDR # LDY REQ_CHAN LDAB ASLB ABY 0 ,Y LDX
\ 90 # LDAB BERM_LINE_BIP_LSB ,X STAB
```

```
\ 01 # LDAB BERM_LINE_BIP_MSB ,X STAB
\ 3E # LDAB BERM_LINE_BIP_THR_LSB ,X STAB
  00 # LDAB TBEK 01 # LDD BERM_ENABLE STD
  0540 # LDD PICRV STD
  0101 # LDD PITRV STD        ( 0.0625 SEC INTERVAL )
  0005 # LDD BT_LIMIT STD
  0000 # LDD BT_COUNT STD

\ <CLEAR-BERM> CONT'D
 HEX
  00 # LDD BT_COUNT STD   01 # LDD PT_USED STD
  00 # LDD BT_TOGGLE STD
  <E-PT> JSR  D ,E ,X ,Y ,Z ,K PULM RTS
\ SET THRESHOLD BACK TO ORIGINAL VALUES
HOST DEFINITIONS HEX
 LABEL <RESET-BERM>    \ SET IT BACK TO DETECTING 1e-4
  D ,E ,X ,Y ,Z ,K PSHM
  00 # LDAB TBEK TBYK 0C # LDAB TBXK
  PLUS_ADDR # LDY REQ_CHAN LDAB 2 # LDAA MUL ABY 0 ,Y LDD XGDX
  34 # LDAB BERM_LINE_BIP_LSB ,X STAB
  00 # LDAB BERM_LINE_BIP_MSB ,X STAB
  4D # LDAB BERM_LINE_BIP_THR_LSB ,X STAB
  00 # LDD PT_USED STD D ,E ,X ,Y ,Z ,K PULM
 RTS
\ ENABLE PT TO RUN WAIT-TO-RESTORE ROUTINE
 HOST DEFINITIONS HEX
 LABEL <RUN-WTR>
  D ,E ,X ,Y ,Z ,K PSHM
  00 # LDAB TBEK
  0540 # LDD PICRV STD
  0108 # LDD PITRV STD        ( 0.5 SEC INTERVAL )

  000A # LDD WTR_LIMIT STD    ( 20 COUNT = 10 SEC  )
  0000 # LDD WTR_COUNT STD    ( RESET COUNTER      )

  0000 # LDD BERM_ENABLE STD
  <E-PT> JSR
  D ,E ,X ,Y ,Z ,K PULM
 RTS

 \  CLEAN ALL REGISTERS
 HOST HEX DEFINITIONS
 : CLR
 1 SET STAT PMON
 2 SET STAT PMON
 3 SET STAT PMON
 4 SET STAT PMON
 8 SET STAT PMON

 ;




  \ SET FLAG VALUES
  HOST DEFINITIONS HEX
  : SFLAG
  1 PLUS_LOS 1 + C!
  1 PLUS_LOF 1 + C!
  ;
```

```
 HOST DEFINITIONS HEX
 : CFLAG
 0 PLUS_LOS 1 + C!
 0 PLUS_LOF 1 + C!
 ;



\ DEBUG OUTPUT
 HOST DEFINITIONS HEX
 : PLUSV
 CR ." REQ_CHAN: " REQ_CHAN C@ U. CR
 ." TEMP1: " TEMP1 @ U. CR
 ." BERM_INT: " BERM_INT @ U. CR
 ." TEMP2: " TEMP2 @ U.  ."            TEMP3: " TEMP3 @ U. CR
 ." SF_INT: " SF_INT @ U.
 ."           APS_INT: " APS_INT @ U. CR
 ." SF_SUM: " SF_SUM C@ U. ."         WTR_INT: " WTR_INT @ U.
CR  ." SD_REQ: " SD_REQ @ U. ."           SF_REQ: " SF_REQ @ U.
 CR ." REMOTE_REQ: " REMOTE_REQ @ U. ."       REMOTE_CMD: "
 REMOTE_CMD @ U. CR
 ." NEW_REQ_PRI: " NEW_REQ_PRI C@ U. ."      CUR_REQ_PRI: "
 CUR_REQ_PRI C@ U. CR ." APS REGISTER: " APS_REG C@ U. CR  ;









\ SHOW SF AND SD STATUS FOR EIGHT CHANNELS

 HOST HEX DEFINITIONS
 : SHOW-OCTAL-REG
 CR ." PLUS_LOS:   " 8 1 DO PLUS_LOS I + C@ U. ."  " LOOP
 CR
 CR ." PLUS_LOF:   " 8 1 DO PLUS_LOF I + C@ U. ."  " LOOP
 CR
 CR ." PLUS_BERM:  " 8 1 DO PLUS_BERM I + C@ U. ."  " LOOP
 CR
 CR ." SF_VAL:     " 8 1 DO SF_VAL I + C@ U. ."  " LOOP
 CR
 CR ." SD_VAL:     " 8 1 DO SD_VAL I + C@ U. ."  " LOOP
 CR
 ;

\ SHOW APS STATUS

  HOST DEFINITIONS HEX
  : ILOOP
  CR CR ." S/UNI-PLUS  1  2  3  4  5  6  7 "
     CR ." --------------------------------- "
```

```
 SHOW-OCTAL-REG
 CR
 ;

 HOST HEX
 : APS
 SHOW-OCTAL-STATUS
 ILOOP
 ;

\ PLUS-ADDR OUTPUT
HOST DEFINITIONS HEX
: ADDRV
CR ."            PLUS ADDRESS " CR
."    1    2    3    4    5    6    7 " CR
." ------------------------------------ " CR
 ."    " PLUS_ADDR 2 + @ . ."  "
 PLUS_ADDR 4 + @ . PLUS_ADDR 6 + @ . PLUS_ADDR 8 + @ .
 PLUS_ADDR A + @ . PLUS_ADDR C + @ . PLUS_ADDR E + @ .
 CR CR ." APS_TX: " APS_TX @ U. CR
 CR ." APS_RX: " APS_RX @ U. CR
 ;
 HOST DEFINITIONS HEX
 : OCTAL
 ILOOP
 ADDRV ;
\ WTR VARIABLES OUTPUT

HOST DEFINITIONS HEX
: WTRV
 CR ." PICRV: " PICRV @ U. CR
 ." PITRV: " PITRV @ U. CR
 ." PICR: " PICR F E@ U. CR ." PITR: " PITR F E@ U. CR
 ." WTR_COUNT: " WTR_COUNT @ U. CR ." WTR_LIMIT: " WTR_LIMIT @
  U. CR  ;




\ BERM VARIABLES OUTPUT

HOST DEFINITIONS HEX
: BERMV
 CR ." PICRV: " PICRV @ U.  ."     PITRV: " PITRV @ U. CR
    ." PICR: " PICR F E@ U. ."     PITR: " PITR F E@ U. CR
 ." BT_COUNT: " BT_COUNT @ U. ."     BT_LIMIT: " BT_LIMIT @
 U. CR
 ."     BERM_ENABLE " BERM_ENABLE @ U. CR
 ." BERM_TOGGLE " BERM_TOGGLE @ U. ."   BT_TOGGLE " BT_TOGGLE @
 U. CR  ." PT_USED " PT_USED @ U. CR
 ;



\  DISPLAY SD ISR VARIABLE VALUES

   HOST DEFINITIONS HEX
   : SDV
   CR ." SD_LIMIT: " SD_LIMIT @ .  ."      SD_COUNT: " SD_COUNT
```

```
   @ . CR ." SD_THR: " SD_THR @ .  ."      SD_VAL: " SD_VAL C@
   U. CR ." SD_INT: " SD_INT @ U.  CR
   ." SD_CLEAR_WAIT: " SD_CLEAR_WAIT C@ U. CR
   ." SD_VAL: " SD_VAL C@ U. CR
     CR ." CHANNEL       1  2  3  4  5  6  7 "
     CR ." ------------------------------- "
     CR ." SD_VAL: " SD_VAL 8 DUMP
     CR ." SD CLEAR_CNT: " SD_CLEAR_CNT 8 DUMP CR
   ;


\ READ OUT REGISTERS                              08:15 12-01-97
 HOST DEFINITIONS HEX
   : KOUT
   CR ." TX_K1: " TX_K1 C@ U.
      ."          TX_K2: " TX_K2 C@ U. CR
      ." RX_K1: " RX_K1 C@ U.
      ."          RX_K2: " RX_K2 C@ U. CR
   ." REQ_CHAN: " REQ_CHAN C@ U.
   CR ." SF_CHK: " SF_CHK C@ U.
   CR ." SD_CHK: " SD_CHK C@ U.
   CR ;
```

```
\ <INT-OFF> : TURNS OFF INTERRUPT ENABLE BITS

 HOST HEX
 LABEL <CLEAR-REG>
  00 # LDAB RSOP_CONTROL ,X STAB
           BERM_CONTROL ,X STAB
           APS_CONTROL_STATUS ,X STAB
 RTS

 HOST HEX
 LABEL <INT-OFF>
  00 # LDAB TBEK 0C # LDAB TBXK 00 # LDX
  00 # LDX <CLEAR-REG> JSR
  3800 # LDX <CLEAR-REG> JSR
  RTS

\ <INT-ON> : TURN ON INTERRUPT ENABLE BITS

 HOST HEX
 LABEL <SET-REG>
  06 # LDAB RSOP_CONTROL ,X STAB
  81 # LDAB BERM_CONTROL ,X STAB
  \         APS_CONTROL_STATUS ,X STAB
 RTS

 HOST HEX
```

```
 LABEL <INT-ON>
  00 # LDAB TBEK 0C # LDAB TBXK 00 # LDX
  00 # LDX <SET-REG> JSR
  3800 # LDX <SET-REG> JSR
  RTS

\ ROUTINE TO WRITE K1, K2 VALUES TO REGISTERS

 HOST DEFINITIONS HEX
   LABEL <WR-K1-K2>
   00 # LDAB TBEK 0C # LDAB TBXK     \ SET BANKS
   APS_TX LDX
   TX_K2 LDAA TRANSMIT_K2 ,X STAA
   TX_K1 LDAA TRANSMIT_K1 ,X STAA
   RTS




\ CLEARS INTERRUPTS AFTER SWITCHING              ( 28 Dec 1997)
HOST HEX LABEL <CLEAR-INT>
  0C # LDAB TBXK
   MASTER_INTERRUPT ,X LDAA       \ CLEAR INTERRUPTS
   RSOP_STATUS ,X LDAA
   RLOP_INTERRUPT ,X LDAA
   BERM_INTERRUPT ,X LDAA
   APS_CONTROL_STATUS ,X LDAA
   TEMP3 STX
   TEMP3 STX
  00 # LDAB MASTER_RESET ,X STAB
  RLOP_BIP_LSB ,X LDAA
  RLOP_BIP_ISB ,X LDAA
  RLOP_BIP_MSB ,X LDAA
 RTS

\ WRITE VALUES TO APS CONTROL REGISTER AND CLEAR INTERRUPTS
HOST HEX
 LABEL <WR-APS-REG>
 D ,E ,X ,Y ,Z ,K PSHM
 00 # LDAB TBEK TBZK 0C # LDAB TBXK 3800 # LDX
 OCTAL_APS_CONTROL ,X LDAB LSRB LSRB LSRB LSRB
   0= IF 08 # LDAB THEN
 ASLB ADDR_MAP # LDZ ABZ 0 ,Z LDE TEMP1 STE
 APS_REG LDAB LSRB LSRB LSRB LSRB 0= IF 08 # LDAB THEN
 ASLB ADDR_MAP # LDZ ABZ 0 ,Z LDE TEMP2 STE
 APS_REG LDAB OCTAL_APS_CONTROL ,X STAB
 300 # LDZ BEGIN -1 # AIZ 0= UNTIL
  TEMP1 LDX <CLEAR-INT> JSR        \ CLEAR INTERRUPTS AND BIP
  TEMP2 LDX <CLEAR-INT> JSR        \ ERROR COUNTERS
  3800 # LDX <CLEAR-INT> JSR
 D ,E ,X ,Y ,Z ,K PULM RTS
\ SET UP ADDRESS FOR RX APS
\ CLEAR THE REST S/UNI-PLUS FROM DETECTING NEW APS BYTES

HOST HEX
  LABEL <SET-RX-APS>
  00 # LDAB TBEK   0C # LDAB TBXK 00 # LDAB TBYK
  0000 # LDX 00 # LDAB 08 # LDAA CNT STAA
  BEGIN  APS_CONTROL_STATUS ,X STAB 800 # AIX
  CNT LDAA 1 # SUBA CNT STAA   0= UNTIL
```

```
  APS_RX LDX 40 # LDAB APS_CONTROL_STATUS ,X STAB

  RTS



\ <SELECT>
\ SETUP ADDRESS MAP FOR SELECTION
 HOST HEX
 LABEL <SELECT>
  00 # LDAB TBEK TBZK TBYK 0C # LDAB TBXK 3800 # LDX
  APS_REG LDAB LSRB LSRB LSRB LSRB 07 # ANDB TEMP1 STAB
  OCTAL_APS_CONTROL ,X LDAA LSRA LSRA LSRA LSRA 07 # ANDA
  CBA 0= NOT IF  00 # LDAB CNT STAB
    BEGIN PLUS_ADDR # LDZ ADDR_MAP # LDY
       CNT LDAB 1 # ADDB CNT STAB ASLB ABY
       ABZ 0 ,Y LDD 0 ,Z STD CNT LDAB 5 # SUBB 0= UNTIL
    TEMP1 LDAB 0= IF 3800 # LDE  APS_RX STE
                  ELSE TEMP1 LDAB ASLB ADDR_MAP # LDZ ABZ
                  0 ,Z LDE APS_RX STE PLUS_ADDR # LDY TEMP1
                  LDAB ASLB ABY 3800 # LDE  0 ,Y STE  THEN THEN
  RTS
 \ <BRIDGE>
 \ SETUP ADDRESS MAP FOR BRIDGE
 HOST HEX
 LABEL <BRIDGE>
  00 # LDAB TBEK TBZK 0C # LDAB TBXK 3800 # LDX
  APS_REG LDAB 07 # ANDB OCTAL_APS_CONTROL ,X LDAA 07 # ANDA
     NEGA ABA 0= NOT IF                       \ NOT EQUAL, NEW
        APS_REG LDAA 07 # ANDA 0=
          IF 3800 # LDE APS_TX STE <WR-K1-K2> JSR
          ELSE
          APS_REG LDAB 07 # ANDB ASLB ADDR_MAP # LDZ ABZ
          0 ,Z LDE APS_TX STE <WR-K1-K2> JSR
          THEN
     THEN
 RTS

 \ <SWITCHING> : DETERMINE IF TO SELECT & BRIDGE
  HOST DEFINITIONS HEX
  LABEL <SWITCHING> D ,E ,X ,Y ,Z ,K PSHM
  00 # LDAB TBEK 0C # LDAB TBXK
  RX_K1 LDAB 0F # ANDB 0F # SUBB 0= IF
         00 # LDAA APS_REG STAA
    ELSE RX_K1 LDAB 0F # ANDB APS_REG LDAA F0 # ANDA ABA
         APS_REG STAA
         TX_K1 LDAB ASLB ASLB ASLB ASLB
         RX_K2 LDAA F0 # ANDA CBA 0= IF
         RX_K2 LDAA F0 # ANDA APS_REG LDAB 0F # ANDB ABA
         APS_REG STAA THEN THEN
         <BRIDGE>  JSR      <SELECT> JSR
         <SET-RX-APS> JSR   <WR-APS-REG> JSR
       D ,E ,X ,Y ,Z ,K PULM RTS
```

```
\ LOCAL REQUEST SUBROUTINES
  HOST DEFINITIONS HEX
  LABEL <LOCAL-SF-DECLARE>  \ SET TO SEND D AS SF
 \  0F # LDAB TBEK 0040 # LDD PICR STD
    00 # LDAB TBEK \ 0000 # LDD PT_USED STD
    REQ_CHAN LDAA F0 # ORAA DF # ANDA TX_K1 STAA
    <WR-K1-K2> JSR
  RTS

  HOST DEFINITIONS HEX      \ SET TO SEND B AS SD
  LABEL <LOCAL-SD-DECLARE>
 \   0F # LDAB TBEK 0040 # LDD PICR STD
    00 # LDAB TBEK \ 0000 # LDD PT_USED STD
    REQ_CHAN LDAA F0 # ORAA BF # ANDA TX_K1 STAA
    <WR-K1-K2> JSR
  RTS
 \ LOCAL SUBROUTINES CONT'D
 HOST DEFINITIONS HEX
 LABEL <LOCAL-WTR-DECLARE>  \ SET TO SEND 6 AS WTR
    D ,E ,X ,Y ,Z ,K PSHM
   00 # LDAB TBEK <SWITCHING> JSR
   REQ_CHAN LDAB 60 # ORAB TX_K1 STAB
   <WR-K1-K2> JSR
    D ,E ,X ,Y ,Z ,K PULM
  RTS
 HOST DEFINITIONS HEX
 LABEL <LOCAL-NO-REQUEST>   \ SET TO SEND 0 AS NR
   00 # LDAB TBEK
      0F # LDAB TX_K1 STAB <SWITCHING> JSR
      <WR-K1-K2>  JSR  00 # LDAB CUR_REQ_PRI STAB
      NEW_REQ_PRI STAB
   RTS









 \ CHECK IF OTHER FAILURE CONDITON EXIST WHEN ONE CHANNEL
 \ HAS RECOVERD
 HOST DEFINITIONS HEX
 LABEL <RESTCHK>
 00 # LDAB TBEK TBXK 01 # LDAB CNT STAB
 00 # LDAB SD_CHK STAB SF_CHK STAB
 BEGIN
   SF_VAL # LDX CNT LDAB ABX
   0 ,X LDAB 0= NOT IF
      CNT LDAB REQ_CHAN STAB 08 # LDAB CNT STAB
      1 # LDAB SF_CHK STAB
   ELSE CNT LDAB 1 # ADDB CNT STAB
   THEN CNT LDAB 08 # SUBB
 0= UNTIL


 \ RESTCHK CONT'D
   SF_CHK LDAA 0=
```

```
    IF 01 # LDAB CNT STAB
    BEGIN
      SD_VAL # LDX CNT LDAB ABX
      0 ,X LDAB 0= NOT
      IF
           CNT LDAB REQ_CHAN STAB 08 # LDAB CNT STAB
           1 # LDAB SD_CHK STAB
      ELSE CNT LDAB 1 # ADDB CNT STAB
      THEN
           CNT LDAB 08 # SUBB
    0= UNTIL
    THEN
  RTS

\ <SF-DECLARE> : DETERMINE IF IT'S A NEW LOCAL SF REQ
HOST DEFINITIONS HEX
LABEL <SF-DECLARE> 00 # LDAB TBEK
   REQ_CHAN LDAA COMA F0 # ORAA 4F # ANDA   \ FIGURE OUT PRI
   NEW_REQ_PRI STAA                         \ BASED ON PRI & C
   REMOTE_CMD LDD 0=                        \ NOT REMOTE REQ
    IF CUR_REQ_PRI LDAB 40 # ANDB 0=        \ CHK IF IT'S SAME
       IF <LOCAL-SF-DECLARE> JSR
        NEW_REQ_PRI LDAB CUR_REQ_PRI STAB
       THEN THEN
   REMOTE_CMD LDD 0>
    IF NEW_REQ_PRI LDAA CUR_REQ_PRI LDAB CBA 0>
     IF <LOCAL-SF-DECLARE> JSR 0 # LDD  REMOTE_CMD STD
     NEW_REQ_PRI LDAB CUR_REQ_PRI STAB
       THEN THEN
    RTS
\ <SD-DECLARE> : DETERMINE IF IT'S A NEW LOCAL SD REQ
HOST DEFINITIONS HEX
LABEL <SD-DECLARE>
   00 # LDAB TBEK
   REQ_CHAN LDAA COMA F0 # ORAA 3F # ANDA   \ FIGURE OUT PRI
   NEW_REQ_PRI STAA                         \ BASED ON PRI & C
   REMOTE_CMD LDD 0=
    IF CUR_REQ_PRI LDAB F0 # ANDB 40 # BITB 0=
      IF 30 # SUBB 0= NOT
         IF <LOCAL-SD-DECLARE> JSR NEW_REQ_PRI LDAB
CUR_REQ_PRI STAB THEN THEN THEN
   REMOTE_CMD LDD 0>
    IF NEW_REQ_PRI LDAA CUR_REQ_PRI LDAB CBA 0>
     IF <LOCAL-SD-DECLARE> JSR NEW_REQ_PRI LDAB
     CUR_REQ_PRI STAB 0 # LDD REMOTE_CMD STD  THEN THEN
RTS
\ <REMOTE-PRIORITY> : DETERMINE REMOTE REQ PRIORITY
HOST DEFINITIONS HEX
LABEL <REMOTE-PRIORITY>
   00 # LDAB TBEK RX_K1 LDAB LSRB LSRB LSRB LSRB
   TBA 0D # SUBA 0= IF 40 # LDAA NEW_REQ_PRI STAA THEN
   TBA 0B # SUBA 0= IF 30 # LDAA NEW_REQ_PRI STAA THEN
   TBA 06 # SUBA 0= IF 20 # LDAA NEW_REQ_PRI STAA THEN
   TBA 02 # SUBA 0= IF 10 # LDAA NEW_REQ_PRI STAA THEN
   TBA 00 # SUBA 0= IF 00 # LDAA NEW_REQ_PRI STAA THEN
   RX_K1 LDAB 0F # ANDB REQ_CHAN STAB COMB 0F # ANDB
   NEW_REQ_PRI LDAA ABA NEW_REQ_PRI STAA

RTS
```

```
 \  <CLEARING> : CHECK TO SEE IF OUTSTANDING SD, SF
 HOST DEFINITIONS HEX
 LABEL <CLEARING>  00 # LDAB TBEK
   REMOTE_CMD LDD 0=                      \ LOCAL CLEAR
    IF CUR_REQ_PRI LDAA F0 # ORAA COMA    \ CHK CH#
      REQ_CHAN LDAB SBA
        0= IF 00 # LDAB CUR_REQ_PRI STAB
          <RESTCHK> JSR
          SF_CHK LDAB 0= NOT
            IF <SF-DECLARE> JSR
            ELSE SD_CHK LDAB 0= NOT IF <SD-DECLARE> JSR
               ELSE ABBA # LDD TEMP1 STD
                 <LOCAL-WTR-DECLARE> JSR <RUN-WTR> JSR
              00 # LDAB TBEK REQ_CHAN LDAA COMA F0 # ORAA
              2F # ANDA CUR_REQ_PRI STAA
    THEN THEN THEN THEN RTS
\ REMOTE REQUEST SUBROUTINES

 HOST DEFINITIONS HEX
 LABEL <REMOTE-SF-SD>
  0F # LDAB TBEK 0040 # LDD PICR STD
  00 # LDAB TBEK 0000 # LDD PT_USED STD
  RX_K1 LDAB 0F # ANDB 20 # ORAB TX_K1 STAB
  RX_K1 LDAB 0F # ANDB ASLB ASLB ASLB ASLB 0D # ORAB
  TX_K2 STAB
    <SWITCHING> JSR   <WR-K1-K2> JSR
  1 # LDD REMOTE_CMD STD
  RTS



\ REMOTE REQUEST SUBROUTINES CONT'D

 HOST DEFINITIONS HEX
   LABEL <REMOTE-WTR>
     <RESTCHK> JSR
     SF_CHK LDAB 0= NOT
       IF <SF-DECLARE> JSR
       ELSE SD_CHK LDAB 0= NOT
         IF <SD-DECLARE> JSR
         THEN
       THEN
   RTS



\ REMOTE REQUEST SUBROUTINES CONT'D

 HOST DEFINITIONS HEX
 LABEL <REMOTE-RR>
  00 # LDAB TBEK <SWITCHING> JSR
  RX_K1 LDAB 0F # ANDB 10 # LDAA MUL 0D # ORAB
  TX_K2 STAB <WR-K1-K2> JSR
 RTS
```

```
\ REMOTE REQUEST SUBROUTINES CONT'D

 HOST DEFINITIONS HEX
 LABEL <REMOTE-NR>
  <SWITCHING> JSR
  0F # LDAB TX_K1 STAB
  RX_K1 LDAB 0F # ANDB 10 # LDAA MUL 0D # ORAB
  TX_K2 STAB
  <WR-K1-K2> JSR
 RTS




\  <REMOTE-REQ> : DETERMINE WHICH REMOTE REQ SUBROUTINE TO CALL

\ ONE OF THE FIVE :
\   1)  SF
\   2)  SD
\   3)  WTR
\   4)  RR
\   5)  NR

\ HELPER FUNCTION
 HOST HEX
 LABEL <NEW-REMOTE-SF-SD>
   <REMOTE-SF-SD> JSR
   NEW_REQ_PRI LDAB CUR_REQ_PRI STAB
 RTS

\  <REMOTE-REQ> : DETERMINE WHICH REMOTE REQ SUBROUTINE TO CALL
 HOST DEFINITIONS HEX
 LABEL <REMOTE-REQ> 00 # LDAB TBEK
  NEW_REQ_PRI LDAA 40 # BITA 0= NOT IF ( 1 )
  REMOTE_CMD LDD 0= NOT IF ( 2 )
    <RESTCHK> JSR
    SF_CHK LDAB 0= NOT IF ( 3 )
      REQ_CHAN LDAA COMA F0 # ORAA 4F # ANDA
      NEW_REQ_PRI LDAB SBA 0> IF ( 4 ) 00 # LDAB
      CUR_REQ_PRI STAB NEW_REQ_PRI STAA <SF-DECLARE> JSR
      ELSE <NEW-REMOTE-SF-SD> JSR THEN ( 4 )
      ELSE <NEW-REMOTE-SF-SD> JSR THEN ( 3 )
    ELSE NEW_REQ_PRI LDAA
    CUR_REQ_PRI LDAB SBA 0< NOT IF ( 5 ) <NEW-REMOTE-SF-SD> JSR
    01 # LDD REMOTE_CMD STD
   THEN ( 5 ) THEN ( 2 ) THEN ( 1 )
\  <REMOTE-REQ> CONT'D
 HEX
  NEW_REQ_PRI LDAA F0 # ANDA 30 # EORA 0= IF ( 1 )
  REMOTE_CMD LDD 0= NOT IF ( 2 )
  <RESTCHK> JSR
  SF_CHK LDAB 0= NOT IF ( 3 )
      REQ_CHAN LDAA COMA F0 # ORAA 3F # ANDA 00 # LDAB
      CUR_REQ_PRI STAB NEW_REQ_PRI STAA <SF-DECLARE> JSR
  ELSE
    SD_CHK LDAB 0= NOT IF ( 4 )
      REQ_CHAN LDAA COMA F0 # ORAA 3F # ANDA
      NEW_REQ_PRI LDAB SBA 0> IF ( 5 ) 00 # LDAB
      CUR_REQ_PRI STAB NEW_REQ_PRI STAA <SD-DECLARE> JSR
```

```
      ELSE <NEW-REMOTE-SF-SD> JSR THEN ( 5 )
    ELSE <NEW-REMOTE-SF-SD> JSR THEN ( 4 )
  THEN ( 3 )
\  <REMOTE-REQ> : DETERMINE WHICH REMOTE REQ SUBROUTINE TO CALL
 HEX

  ELSE
      NEW_REQ_PRI LDAA
      CUR_REQ_PRI LDAB SBA 0< NOT IF ( 6 )
        <NEW-REMOTE-SF-SD> JSR
        01 # LDD REMOTE_CMD STD
      THEN ( 6 )
  THEN ( 2 )
 THEN ( 1 )




 \ <REMOTE-REQ> CONT'D
 HEX
   NEW_REQ_PRI LDAA F0 # ANDA 20 # EORA 0= IF
    00 # LDD REMOTE_CMD STD NEW_REQ_PRI LDAB CUR_REQ_PRI STAB
    <REMOTE-WTR> JSR
   THEN

   NEW_REQ_PRI LDAA F0 # ANDA 10 # EORA 0= IF
    <REMOTE-RR> JSR
   THEN

   NEW_REQ_PRI LDAA F0 # ANDA 0= IF
     00 # LDAB CUR_REQ_PRI STAB NEW_REQ_PRI STAB
    <REMOTE-NR> JSR
   THEN
 RTS
 \ <CHK-REQ> : DETERMINE WHICH REQUEST IS VALID  ( 28 Dec 1997)
 HOST DEFINITIONS HEX
 LABEL <CHK-REQ>
  0F # LDAB TBEK PORTF LDAA COMA PORTF STAA     \ TOGGLE LED
  00 # LDAB TBEK TBXK                          \ BANK 0
  SF_REQ LDD 0= NOT
    IF SF_VAL # LDX REQ_CHAN LDAB ABX 0 ,X LDAB 0=
    IF <CLEARING> JSR
    ELSE   <SF-DECLARE> JSR THEN
     00 # LDD SF_REQ STD THEN

  SD_REQ LDD 0= NOT
    IF SD_VAL # LDX REQ_CHAN LDAB ABX 0 ,X LDAB 0=
     IF  <CLEARING> JSR
     ELSE <SD-DECLARE> JSR THEN
    00 # LDD SD_REQ STD THEN
 \ <CHK_REQ>                                    ( 05 Jan 1998)
 HEX
  REMOTE_REQ LDD 0= NOT
   IF <REMOTE-PRIORITY> JSR
      <REMOTE-REQ> JSR
      00 # LDD REMOTE_REQ STD
   THEN
   RTS
```

```
 \ <ADD-SF>
 \ FIGURE OUT SUM OF SIGNAL FAILURE CONDITIONS
 HOST DEFINITIONS HEX
 LABEL <ADD-SF>
   PLUS_LOS # LDY REQ_CHAN LDAB ABY
   PLUS_LOF # LDZ REQ_CHAN LDAB ABZ
   0 ,Y LDAB 0 ,Z LDAA ABA SF_SUM STAA
   PLUS_BERM # LDY REQ_CHAN LDAB ABY
   PLUS_AISL # LDZ REQ_CHAN LDAB ABZ
   0 ,Y LDAA SF_SUM ADDA 0 ,Z LDAB ABA SF_SUM STAA
   RTS




\  INTERRUPT CHECK SUBROUTINES                  12:31 30/12/97
 HEX
 LABEL <SF-INT>

  SF_VAL # LDX REQ_CHAN LDAB ABX 00 # LDAB TBEK
  <ADD-SF> JSR
  0 ,X LDAA 0=                        \ CHK IF SF PREV SET
   IF SF_SUM LDAA 0>                  \ NEW SF DECLARED
      IF 1 # LDAB 0 ,X STAB           \ SET FLAGS
        1 # LDD SF_REQ STD THEN

  ELSE                                \ SF PREV DECLARED
   SF_SUM LDAA 0=                     \ CLEARED
   IF 0 # LDAB 0 ,X STAB              \ SET FLAGS
   1 # LDD SF_REQ STD THEN
  THEN
\  INTERRUPT CHECK SUBROUTINES CONT'D           12:31 30/12/97
HEX
   SF_SUM LDAA 1 # SUBA 0= IF

   PLUS_BERM # LDY REQ_CHAN LDAB ABY
   0 ,Y LDAB TEMP2 STAB 0= NOT
    IF PT_USED LDD 0= IF
    <CLEAR-BERM> JSR
  THEN THEN THEN


  RTS




 \ INTERRUPT CHECK SUBROUTINES
 HEX
  LABEL <SD-INT>
  SF_VAL # LDY REQ_CHAN LDAB ABY
    0 ,Y LDAB 0= IF
      00 # LDAB TBEK
      01 # LDD SD_REQ STD
    THEN
  RTS
```

```
 \ INTERRUPT CHECK SUBROUTINES CONT'D

 \ LOAD RECEIVE K1 AND K2 BYTES FROM S/UNI-PLUS
  HOST HEX
  LABEL <APS-INT>
   00 # LDAB TBEK 0C # LDAB TBXK
   APS_RX LDX
   RECEIVE_K1 ,X LDAA RX_K1 STAA
   RECEIVE_K2 ,X LDAA RX_K2 STAA
   1 # LDD REMOTE_REQ STD
  RTS




 \ INTERRUPT CHECK SUBROUTINES CONT'D
  HOST HEX DEFINITIONS HEX
  LABEL <BERM-INT>
    PLUS_BERM # LDY REQ_CHAN LDAB ABY
    0 # LDAB 0 ,Y STAB
    00 # LDD PT_USED STD
    <RESET-BERM> JSR
    <SF-INT> JSR
  RTS
  HOST HEX
  LABEL <WTR-INT>
    00 # LDAB TBEK <RESTCHK> JSR
    SF_CHK LDAB 0= IF  SD_CHK LDAB 0= IF
     <LOCAL-NO-REQUEST> JSR 00 # LDAB CUR_REQ_PRI STAB
     THEN THEN  00 # LDD WTR_INT STD
   RTS
\ <CHK-INT> : CHECK WHICH INTERRUPT OCCURED
HOST DEFINITIONS HEX
  LABEL <CHK-INT>
  00 # LDAB TBEK TBXK TBYK TBZK

  SF_INT LDD 0= NOT IF                    \ SF INTERRUPT
   <SF-INT> JSR
   00 # LDD SF_INT STD
   <CHK-REQ> JSR
   THEN

  APS_INT LDD 0= NOT IF                   \ RECEIVED APS INTERR
   <APS-INT> JSR
   00 # LDD APS_INT STD
   <CHK-REQ> JSR
   THEN
\  <CHK_INT> CONT'D                                04:07 12-02-9
 HEX

  WTR_INT LDD 0= NOT IF               \ WTR INTERRUPT OCCURED
    <WTR-INT> JSR
  THEN

  SD_INT LDD 0= NOT IF                \ SD INTERRUPT OCCURED
    <SD-INT> JSR
    00 # LDAB SD_INT STD
```

```
    <CHK-REQ> JSR
  THEN



 \  <CHK_INT> CONT'D
 HEX

  BERM_INT LDD 0= NOT IF           \ BERM INTERRUPT OCCURED
  <BERM-INT> JSR
  00 # LDD BERM_INT STD
  <CHK-REQ> JSR
  THEN

  FF1F # ANDP
  D ,E ,X ,Y ,Z ,K PULM  RTI
```

```
\ <PLUS-ISR> ISR ROUTINE                          ( 28 Dec 1997)
HOST DEFINITIONS HEX
LABEL <PLUS-ISR>
D ,E ,X ,Y ,Z ,K PSHM 00C0 # ORP
 F700 # LDD BEGIN 0001 # ADDD 0= UNTIL       \ WAIT ALL INTERRUP
 00 # LDAB TBEK TBYK TBZK
 0C # LDAB TBXK 0000 # LDX
 01 # LDAB CNT STAB 0F # LDAB REQ_CHAN STAB
 BEGIN
 PLUS_ADDR # LDZ CNT LDAB ASLB ABZ 0 ,Z LDX TEMP2 STX
 MASTER_INTERRUPT ,X LDAA PSHA 01 # ANDA 0= NOT \ CHK IF RSOP =
 IF RSOP_STATUS ,X LDAA \ 20 # BITA 0>      \ CHK IF LOSI = 1
  \ IF                                        \ CHK IF LOSV = 1
  PLUS_LOS # LDY CNT LDAB ABY


\ <PLUS-ISR> ISR ROUTINE CONT'D
HEX
    04 # BITA 0= NOT
    IF  1 # LDAB 0 ,Y STAB             \ WR 1 TO FLAG
    ELSE 0 # LDAB 0 ,Y STAB            \ WR 0 TO FLAG
    THEN \ CNT LDAB REQ_CHAN STAB 01 # LDE SF_INT STE
    THEN

    10 # BITA 0>
     IF                                \ CHK LOFI & LOFV
     PLUS_LOF # LDY CNT LDAB ABY
     02 # BITA 0>
     IF 1 # LDAB 0 ,Y STAB               \ WR 1 TO FLAG
```

```
     ELSE 0 # LDAB 0 ,Y STAB
     THEN CNT LDAB REQ_CHAN STAB 01 # LDE SF_INT STE
   THEN
\ <PLUS-ISR> ISR ROUTINE CONT'D
\ CHECK BERM REGISTER
 HEX
   PULA 00 # LDAB TBEK
   80 # ANDA 0= NOT
   IF
   BERM_INTERRUPT ,X LDAA 01 # BITA 0>
   IF PLUS_BERM # LDY CNT LDAB ABY
      00 # LDAB TBEK 0 ,Y LDAA 0=
     IF 1 # LDD SF_INT STD 0 ,Y STAB     \ WRITE BERM VAL
        CNT LDAB REQ_CHAN STAB           \ SET CHAN
        01 # LDD SF_INT STD              \ SET INT FLAG
     ELSE BERM_TOGGLE LDD 0001 # EORD BERM_TOGGLE STD
          CNT LDAB REQ_CHAN STAB
     THEN THEN THEN

\ <PLUS-ISR> ISR ROUTINE CONT'D
HEX

  CNT LDAB 1 # ADDB CNT STAB
  CNT LDAB 08 # SUBB
  0= UNTIL









\ <PLUS-ISR> ISR ROUTINE CONT'D

\ CHECK APS_RX REGISTER FOR NEW APS BYTES
 HEX
  00 # LDAB TBEK 0C # LDAB TBXK
  APS_RX   LDX APS_CONTROL_STATUS ,X LDAA 04 # BITA 0= NOT
   IF 00 # LDAB TBEK 1 # LDD APS_INT STD
   THEN









\ <PLUS-ISR> ISR ROUTINE CONT'D
 HEX
   00 # LDAB TBXK
   <CHK-INT> BRA

   <PLUS-ISR> 0013  EXCEPTION
```

```
\ <PT-ISR> ISR ROUTINE
  HOST DEFINITIONS HEX
  LABEL <WTR>
    0F # LDAB TBEK PORTF LDAA COMA PORTF STAA  \ TOGGLE LED
    00 # LDAB TBEK WTR_COUNT LDD  1 # ADDD     \ ADD COUNTER
    WTR_COUNT STD WTR_LIMIT LDE                \ CHK IF REACHED
    SDE 0=  IF 0F # LDAB TBEK 0040 # LDD PICR STD
            00 # LDAB TBEK 1 # LDD WTR_INT STD
            <CHK-INT> BRA
    THEN D ,E ,X ,Y ,Z ,K PULM RTI

  LABEL <PT-ISR>
   D ,E ,X ,Y ,Z ,K PSHM
   00 # LDAB TBEK BERM_ENABLE LDD 0=
   IF <WTR> BRA

\ <PT-ISR> ISR ROUTINE CONT'D
 HEX
  ELSE
  00 # LDAB TBEK BT_TOGGLE LDD BERM_TOGGLE LDE
  SDE 0=                                       \ CHK IF =
    IF BT_COUNT LDD 1 # ADDD BT_COUNT STD      \ ADD COUNT
    ELSE 0000 # LDD BT_COUNT STD               \ CLEAR COUNT
         BERM_TOGGLE LDD BT_TOGGLE STD
    THEN
    BT_LIMIT LDE BT_COUNT LDD SDE 0=           \
    IF 0001 # LDD BERM_INT STD
      0F # LDAB TBEK 0040 # LDD PICR STD
      <CHK-INT> BRA
  THEN THEN D ,E ,X ,Y ,Z ,K PULM RTI
  <PT-ISR> 40 EXCEPTION
```

```
\ <SD-ISR> ISR ROUTINE
HOST DEFINITIONS HEX
  LABEL <OVER>                        \ THRESHOLD EXCEEDED
    0 # LDAB TBEK
    SD_VAL # LDY CNT LDAB ABY
    SD_CLEAR_CNT # LDZ CNT LDAB ABZ 00 # LDAB 0 ,Z STAB
    0 ,Y LDAB 0=
      IF 01 # LDAB 0 ,Y STAB
         01 # LDD SD_INT STD
         CNT LDAB REQ_CHAN STAB
      THEN
    RTS



\ <SD-ISR> ISR ROUTINE CONT'D
HOST DEFINITIONS HEX
  LABEL <UNDER>                         \ CLEARING DETECTED
    0 # LDAB TBEK
    SD_VAL # LDY CNT LDAB ABY 0 ,Y LDAB
     0= NOT IF
       SD_CLEAR_CNT # LDZ CNT LDAB ABZ
       0 ,Z LDAB 0001 # ADDB 0 ,Z STAB
       SD_CLEAR_WAIT LDAA 0 ,Z LDAB SBA 0=
         IF 00 # LDAB 0 ,Z STAB
            01 # LDD SD_INT STD
            00 # LDAB 0 ,Y STAB
            CNT LDAB REQ_CHAN STAB
         THEN
    THEN
    RTS
\ <SD-ISR> ISR ROUTINE CONT'D
  HOST DEFINITIONS HEX
  LABEL <SD-ISR>
  D ,E ,X ,Y ,Z ,K PSHM
  00 # LDAB TBEK 0C # LDAB TBXK
  SD_COUNT LDD 0001 # ADDD SD_COUNT STD SD_LIMIT LDD
  SD_COUNT SUBD 0=                                \ CHK COUNT
    IF 00 # LDAB TBEK 0000 # LDD SD_COUNT STD   \ RESET COUNT
    00 # LDAB TBEK TBYK TBZK 01 # LDAB CNT STAB
    0000 # LDX
  BEGIN
    PLUS_ADDR # LDZ CNT LDAB 2 # LDAA MUL ABZ 0 ,Z LDD XGDX
    00 # LDAB MASTER_RESET ,X STAB             \ LOAD BIP
    RLOP_BIP_LSB ,X LDAB RLOP_BIP_ISB ,X LDAA


\ <SD-ISR> ISR ROUTINE CONT'D
 HEX
      TEMP1 STD TDE SD_THR LDD TEMP2 STD NEGD ADE 0>
        IF <OVER> JSR
        ELSE  <UNDER> JSR
        THEN

     CNT LDAB 1 # ADDB CNT STAB
     CNT LDAB 08 # SUBB
     0= UNTIL
```

```
\ <SD-ISR> ISR ROUTINE CONT'D
 HEX
    THEN
    0F # LDAB TBEK TFLG1 LDD FEFF # ANDD TFLG1 STD

    <CHK-INT> BRA

    <SD-ISR> 51 EXCEPTION


\                                                   00:54 12/09/97


\ SETUP2                                            ( 05 Jan 1998)
  HOST DEFINITIONS HEX
  : SETUP2
  INIT-MICRO
  ZZ
  DI CLR
  1 SET INIT-APS
  2 SET INIT-APS
  3 SET INIT-APS
  ( 4 SET INIT-APS  )
  8 SET INIT-APS-REGA INIT-APS
  ( INIT-SD INIT-GPT ) 00 82 WDZ
  CLR CLR CLR EI
  ;


                                                    ( 28 Dec 1997)
 HOST HEX
 18 LOAD
 HOST HEX
 19 76 THRU
```

NOTES

CONTACTING PMC-SIERRA, INC.

PMC-Sierra, Inc.
105-8555 Baxter Place Burnaby, BC
Canada V5A 4V7

Tel: (604) 415-6000

Fax: (604) 415-6200

Document Information: document@pmc-sierra.com
Corporate Information: info@pmc-sierra.com
Application Information: apps@pmc-sierra.com
Web Site: http://www.pmc-sierra.com

PM-971116 (R2) Issue date: Febuary 1998