April 16, 1999
Revision 3.0
(for 10K based control modules)

PP Controller Card

The PP can accept up to four controller cards. Each controller card plugs into a 96 pin connector slot in the PP mother board. The controller card circuitry is divided into three sections.

- The State Assembly register
- The Controller State Memory
- The Control Module

Each of these sections is described under a separate heading below.

1. State Assembly Register

This is a 96 bit wide register which is used to assemble each controller state word before it is written to the controller state memory. Words are assembled in the register by writing individual 16 bit fields from the OBD bus. The register presents the assembled state word on the 96 bit wide SM bus. The register also contains a read back multiplexer so that the contents of the 16 bit fields on the local SM bus can be read back to the OBD bus. This allows the state memory contents to be read for diagnostic purposes.

The state assembly register fields are accessible to the PP CPU by via the memory address map. Each PP controller slot has an assembly register base address at which the registers are accessible (see the PP Memory Map document, section II-L). The functions of the individual fields are detailed in section 3.2 of the present document. Access to the assembly register is write only. Reading an assembly register field will return the contents of that field in the currently addressed state memory word, not the contents of the field in the assembly register.

The state assembly register is interfaced to the following card connector slot signals:

Signal Name	Type	Description
OBD[15:0]	I/O	This 16 bit data bus is used to write data to fields in the state assembly register and to read data fields from the SM bus.
BA[6:1]	Input	This bus specifies the address (position in the state word) of the field that is to be written to the state assembly register from the SM bus or read from the state memory to the SM bus.
RD_MEM	Input	When this signal is asserted the contents of the selected field from the SM bus are driven on to the OBD bus. This action is performed only if SELECT is also asserted.
WR_ASMB	Input	The active edge of this signal cause the contents of the OBD bus to be written into the selected field of the state assembly register. This action is performed only if SELECT is also asserted.
SELECT	Input	This signal qualifies the RD_MEM and WR_ASMB inputs. RD and WRT will only perform their selected actions if SELECT is also asserted.

2. Controller State Memory

The controller state memory has a word width of 96 bits and is 65,536 words deep. The memory is connected to the state assembly register and the control module by the 96 bit wide SM bus. The memory address is provided over the MA bus by the control module. The control module also generates the signals that control the reading and writing of the memory.

3. Control Module

The control module consists of a single FPGA. The control module is a state machine that is responsible for generating the memory address from which the next state will be fetched and for determining when the next state is loaded. The next state is determined by the current contents of the registers in the control module and the contents of control state memory word for the currently addressed state. The following sections describe the interface signals for the control module FPGA, the fields in the state memory control word, etc.

3.1. External Interface

Signal Name	Type	Description
		Local Signals (These signals *are not* on the slot connector)
-SM_CE	Output	Asserted to chip enable the local controller state memory.
-SM_OE	Output	Asserted to output enable the local controller state memory.
-SM_WE	Output	Asserted to write enable the local controller state memory.
SM[95:0]	Input	Data lines from the controller state memory
		External Signals (The remainder of the signals are all on the slot connector)
		State Memory Controls
MA[15:0]	Output	State Memory address
LD_ST L	Output	Asserted to request that all output registers on associated output cards be loaded on the next clock edge with the currently addressed contents of the state memory.
MEMOE H	Output	Asserted to enable the state memory outputs on all associated output cards.
WR_STM H	Output	Asserted to write the state memory on all associated output cards with the current contents of the state assembly register.
CLR_OUT L	Output	Asserted to hold the state output registers on all associated output cards in the reset condition.
SCLK H	Input	Synchronous clock for the control module address generator.
		CPU Interface
OBD[15:0]	Tristate	Data bus for writing to or reading from internal registers.
BA[6:1]	Input	Buffered address signals from the CPU. Asserted by the CPU to specify the read or write operation to be performed.
SELECT	Input	This input qualifies the WR_ASMB, RD_MEM, RD_CTR, and WRT_CTR inputs by indicating that this control module is selected to perform the requested action.
WR_ASMB	Input	Asserted to write the field in the state assembly register specified by BA[6:1] with the contents of OBD[15:0]. The action is performed only if SELECT is also asserted.

RD_MEM	Input	Asserted to read the field of the currently addressed State Memory word selected by BA[6:1] onto OBD[15:0].
RD_CTR H	Input	Asserted by the CPU to read the register specified by BA[6:1] from the control module onto OBD[15:0]. The action is performed only if SELECT is also asserted.
WRT_CTR H	Input	Asserted by the CPU to write the register (or execute a command) specified by BA[6:1] using the data from OBD[15:0]. The action is performed only if SELECT is also asserted.
RESET	Input	Resets control module to a known state
BUSY H	Output	Asserted by the controller to indicate a state memory write operation is in progress.
FIFO FULL H	Output	Asserted to indicate the state FIFO is full.
INTRPT	Output	Asserted to indicate that the control module state has changed from halted to running or from running to halted, or that one of the experiment state registers has been changed. The signal is deasserted by the action of an INIT command or a CLR INTRPT command from the CPU interface.
		External Controls
EXT_HLT L	Input/Output (open collector)	Whenever a controller halts it asserts this open collector signal for one clock cycle following the halt to notify other controllers that the halt occurred. This will halts all other controller connected to this common open collector signal.
EXT_FLT H	Input	Asserted by external logic to indicate a spectrometer fault has been detected. The controller will be held in the halted condition as long as this input is asserted.
SYN_OUT1 H	Output	The controller asserts this signal to indicate that has reached the end of a state requesting external synchronization with another controller.
SYN_OUT2 H	Output	
SYN_OUT3 H	Output	
SYN_IN1 H	Input	When a controller is asserting an SYN_OUT*n* signal, it will wait at the end of the state until it detects that the corresponding SYN_IN*n* is asserted. Each controller is connected to each of the other three controllers by a SYN_IN*n* - SYN_OUT*n* signal pair.
SYN_IN2 H	Input	
SYN_IN3 H	Input	
		Special State Outputs (Controller 0 only)
AD_STRB L	Output	Asserted for one clock cycle at the beginning of a state to request that the A-D converters digitize one point.
CMD_RDY L	Output	Asserted to indicate that the state contains a command or a command parameter for the DAP.
CMD_PARAM H	Output	Asserted to indicate that the infomrmation for the DAP is a command parameter rather than a command.
CMD_ACK L	Input	Asserted by the DAP to signal the control module that the command or command parameter has been acknowledged.

3.2. Controller State Memory Fields

The 96 bit wide controller state memory word is divided up into the following logical fields. Since fields are addressed by BA[6:1] as 16 bit entities, but logical fields may be up to 32 bits wide, each logical field may occupy up to two physical field addresses in a memory word. Each entry shows the starting physical field address (in parenthesis), relative to the assembly register base for this card, followed by the actual bit position in the memory word.

The field addresses are byte boundary addresses, however, fields are not accessible as byte entities.

3.2.1. Timing Field (0)- SM[31:0]

This 32 bit field specifies the state duration in 50 nS units. A value of zero corresponds to a duration of 50 nS, a value of one corresponds to 100 nS, etc.

3.2.2. Repeat/Data Field (4) - SM[55:32]

This 24 bit field serves two purposes.

3.2.2.1. Repeat Count

If the repeat bit in the control field is set this field specifies the number of times the state should be repeated. A repeat count value of zero specifies that the state should be repeated once (i.e., it will execute normally once and then be repeated once). A value of one specifies two repeats, etc.

3.2.2.2. Data for Experiment State Register

If any of the four *ESR n* bits are set in the *control* field, the contents of the *repeat/data* field are loaded into the corresponding Experiment State register.

3.2.3. Call Address Field (8) - SM[71: 56]

If the *call/return* bit in the *control* field is set, and the state was fetched from the FIFO, this 16 bit field specifies the address of the next state.

3.2.4. Control Field (12)- SM[95: 72]

Control Field		
Bit #	**Function**	**Description**
0	call/return	If this bit is set, and the current state was fetched from the FIFO, then the next state will be fetched from the subprogram RAM address specified by the *call address* field. If the state also contains a set *repeat* bit, the call will be repeated each time the flow of control returns from the subprogram until such time as the repeat count is exhausted. The next sequential state will then be fetched from the FIFO. If the call/return bit is set, and the current state was fetched from the subprogram RAM, then the next state will be fetched from the FIFO. If the state also contains a set *repeat* bit, the return will not occur until such time as the repeat count is exhausted.
1	FIFO sync	Whenever a state with a set *FIFO sync* bit is loaded into the FIFO by the service processor, the *FIFO sync counter* is incremented. Whenever a state with a set *FIFO sync* bit is removed from the FIFO and loaded into the output register, the *FIFO sync counter* is decremented.
2	repeat	If this bit is set, repeat the state the number of times specified by the *repeat/data* field.
3	cm sync 1	The *cm sync n* bit requests synchronization with the Control Module specified by n.
4	cm sync 2	
5	cm sync 3	
6	halt	If this bit is set in the current state, then the Control Module will halt at the end of the current state.

7	load ESR 1	If this bit is set in the current state, then *experiment state register 1* will be loaded with the contents of the *repeat/data* field.
8	load ESR 2	
9	load ESR 3	
10	unused	
11	conditional action 1	If this bit it is set in the current state, the state is identified as one where a conditional halt or pause may occur.
12	conditional action 2	
13	conditional action 3	
14	conditional action 4	
15	AD strobe	If this bit is set, the AD_STRB signal output is asserted for 50 nS at the beginning of the state.
16	DAP notify	If this bit is set, the CMD_RDY signal output is asserted for the duration of the state to indicate that the DAP Direct Command field should be read by the DAP.
17	DAP data	If this bit is set, the CMD_PARAM signal output is asserted for the duration of the state to indicate that the contents of the DAP Direct Command field are a command parameter rather than a command.
18	DAP nowait	If this bit is *not* asserted, the controller will pause at the end of a state which contains a set *DAP Notify* bit until such time as the DAP acknowledges the command or parameter by strobing the CMD_ACK input signal. If this bit is set, a DAP Wait Error will occur if a state containing a set *DAP Notify* bit is not acknowledged before the end of the state.

3.3. CPU Command Interface

The control module has a command interface to the CPU on the PP motherboard. This allows the CPU to read and write registers in the control module and to execute commands that request actions by the control module. The command to be executed is specified by buffered address bits BA[6:1]. Since BA0 is not included (and is assumed to be zero) all commands correspond to even addresses on the CPU data bus.

3.3.1. Write Commands

Commands that write data to registers that are wider that 16 bit are divided into two commands; one to write the least significant 16 bits and a second command to write the remaining most significant bits. In all such cases, the command to write the 16 least significant bits writes to a holding register rather that the actual target. When the command to write the most significant 16 bits is executed the target register is then written using the data supplied with the second command as well as the data from the holding register. Note that there is only one holding register. This implies that both commands to write a particular register should be executed without intervening write commands that might alter the holding register.

Command Name	BA[6:0]	Description
WRT SM1L	00	Write state match register 1 least significant bits with OBD[15:0]
WRT SM1H	02	Write state match register 1 most significant bits with OBD[15:0]
WRT SM2L	04	
WRT SM2H	06	
WRT SM3L	08	
WRT SM3H	0a	
WRT CAM	10	Write conditional action mask register with D[7:0]

WRT SMWA	12	Write state memory write address register with D[15:0]
WRT FBSA	14	Write FIFO buffer start address with D[15:0]
WRT ST	16	Write the state memory location address by the SMWA register with the contents of the state assembly register. If bit 73 of the state assembly register (FIFO sync) is set then the *FIFO sync counter* is also incremented.
RUN	18	Start the controller
HALT	1a	Halt the controller
CONTINUE	1c	Continue from conditional pause state
CLR STATUS	1e	Clear status interrupt.
INC FSYNC	20	Increment the FIFO sync counter
INIT	22	Initialize the controller

3.3.2. Read Commands

Commands that read data from registers that are wider that 16 bit are divided into two commands; one to read the least significant 16 bits and a second command to read the remaining most significant bits. In all such cases, the command to read the 16 least significant bits reads the entire source register and stores the most significant bits in a holding register so that they can be read by another command. The command to read the most significant bits reads only the holding register. Note that there is only one holding register. This implies that both commands to read a particular register should be executed without intervening read commands that might alter the holding register.

Command Name	**CMD[4:0]**	**Description**
RD ESR1L	00	Read experiment state register 1 least significant bits to OBD[15:0]
RD ESR1H	02	Read experiment state register 1 most significant bits to OBD[15:0]
RD ESR2L	04	
RD ESR2H	06	
RD ESR3L	08	
RD ESR3H	0a	
RD STATUS	28	Read status register to OBD[15:0]

3.4. Status register

The CPU reads the control module status register to determine the state of the control module. The bits in the status register are assigned the following meanings.

Status		
Bit #	**Name**	**Meaning**

0	**Run**	When set, this bit indicates that the Control Module is running. The *Run* bit becomes set by the action of a RUN command from the CPU interface. As soon as the *Run bit* becomes set, the Control Module fetches the first state from the FIFO and expresses it. The Control Module will continue expressing states until the *Run* bit is cleared. The *Run* bit is cleared when the Control Module is halted by any of the following events: o A state was completed in which the *halt* bit was set in the *control* field. o A HALT command from the CPU interface o The EXT_HLT or EXT_FLT input signals were asserted. o An error occurred
1	**FIFO Sync Pause**	When set, this bit indicates that the Control Module is paused because the *FIFO sync counter* is zero. The pause will end when a state is loaded into the FIFO that contains a set *FIFO sync* bit in the *control* field.
2	**Conditional Halt**	When set this bit indicates that a halt occurred because the condition set by the *halt mask* register is satisfied.
3	**DAP Pause**	When set, this bit indicates that the Control Module is paused awaiting a response from the DAP.
4	**CM Sync1 Pause**	Paused waiting for synchronization with control module #1.
5	**CM Sync2 Pause**	Paused waiting for synchronization with control module #2.
6	**CM Sync3 Pause**	Paused waiting for synchronization with control module #3.
7	**Stop**	When set, this bit indicates that a halt occurred because the Control module received a stop command from the CPU interface.
8	**Local Halt**	When set, this bit indicates that a halt occurred because this Control Module completed a state in which the *halt* bit was set in the *control* field.
9	**FIFO Sync = 0**	Indicates that the current contents of the FIFO sync counter are zero.
10	**External Halt**	When set, this bit indicates that a halt occurred because the EXT_HLT input signal was asserted.
11	**External Fault**	When set, this bit indicates that a halt occurred because the EXT_FLT input signal was asserted.
12	**FIFO Empty Error**	When set, this bit indicates that the Control Module failed to fetch the next state from the FIFO because the FIFO was found to be empty at the end of the current state.
13	**DAP Error**	When set, this bit indicates that the DAP did not respond before the end of a state that had a set *DAP Notify* bit and a set *DAP nowait* bit in the *control* field.
14	**RAM Error**	When set, this bit indicates that the Control Module attempted to fetch a subprogram state from a location in state memory that was not allocated as subprogram RAM. This can happen in two ways: o The *call address* field of a state fetched from the FIFO contained an illegal address. o A sequential fetch of a subprogram state was from an illegal address. This can occur if the *call/return* bit is not set in the last state of a subprogram. The *RAM Error* bit is reset by an INIT command from the host.
15	**Conditional Pause**	When set, this bit indicates that the Control Module is paused because the current state met the conditions for a conditional pause. The Control Module will remain paused until a CONTINUE command is executed.

3.5. Experiment State Register

The Control Module provides three general purpose *experiment state registers* that can be loaded by action of the pulse program. These registers are each 24 bits in length and can be used to maintain current values for experiment state parameters such as "experiment number" and "scan number". In order to provide a mechanism for loading these *experiment state registers*, the *control* field contains three bits each of which will cause the loading of one of the three registers at the beginning of any state in which the bit is set. The value loaded is taken from the 24 bit *repeat/data* field of the same state. This obviously precludes loading a register from a repeated state.

3.6. State Match Registers

Each 24 bit *experiment state register* (see Section 3.5) has a associated *state match register,* which can be loaded by command from the host (see Section 3.3.1), and comparator logic to determine if the two values are equal. This mechanism can be used to cause the Control module to pause or halt when a match occurs. The action that occurs is controlled by the *condition mask register* (see Section 3.7).

3.7. Conditional Action Mask Register

The *condition mask register* is writable by the host (see Section3.3.1) and contains seven bits that determine the logical combination required to produce a conditional action. The eighth bit in the register determines whether the action will be a *pause* or a *halt*. In order for a conditional action to occur, the logical "and" of the conditions specified by each set bit in the mask must be true. The condition specified by each bit in the mask is as follows:

Bit #	**Condition Specified**
0	*Conditional Action 1* is set in state
1	*Conditional Action 2* is set in state
2	*Conditional Action 3* is set in state
3	*Conditional Action 4* is set in state
4	*Experiment State Register 1* match
5	*Experiment State Register 2* match
6	*Experiment State Register 3* match
7	*Pause/Halt*

For example, if only bit 0 of the pause condition mask is set, a pause will occur at the end of the next state that has the *conditional action 1* bit set in its *control* word. If, in a more complicated case, both bits 1 and 5 are set, the pause will occur only if the state has the *conditional action 2* bit set in its *control* word **and** the contents of *experiment state register 2* match those of *state match register 2.* In either case the action will occur at the end of the state. If bit seven is set the action will be a *pause*; otherwise it will be a *halt*. If a *pause* occurs, the Control Module will remain paused at the end of the state until such time as the host executes a CONTINUE command (see Section 3.3.1)

3.8. State Address Generator Rules

The primary function of the control module is to generate the memory address from which each program state is fetched. The following section documents the major elements of the address generator state machine.

3.8.1. Address Generator State Variables

These are the variables that contain the current state of the address generator. The next address is determined by applying the rules to the state variables and to the contents of the currently addressed state word.

3.8.1.1. TMR[31:0]

This the state timer variable. This variable is managed as a down-counter which is loaded at the beginning of each state. This counter is loaded from state memory whenever a new state is loaded. A value of zero indicates that TIMER_DONE will be set on the next clock. Thus a value of zero specifies a state duration of two clocks, which is the minimum possible.

3.8.1.2. TIMER_DONE

This variable is set on the next clock after TMR[31:0] reaches zero. When set, this variable indicates that it is time to load a new state. It is cleared whenever a new state is loaded.

3.8.1.3. SUBA

This variable is set to indicate that the currently expressed address is the address of a subprogram state in the portion of the state memory that is managed as RAM. When clear, the variable indicates that the currently expressed address is the address of a state in the FIFO portion of the state memory.

3.8.1.4. RPT_F[23:0], RPT_R[23:0]

These are the state repeat counters. There are two, one associated with FIFO states and one associated with subprogram states in the RAM portion of state memory. The appropriate counter is loaded whenever the first repetition of a repeated state is loaded. Complete rules for loading and decrementing follow in the next section.

3.8.1.5. RPTIP_F, RPTIP_R

These are the repeat in progress variables. There is one associated with each of the two repeat counters. When set they indicate that a repeated state is in progress.

3.8.1.6. RA[15:0]

This variable contains the return address. This is the FIFO address from which the next state will be fetched after a subprogram finishes execution.

3.8.1.7. CMRA[15:0]

This variable contains the current memory read address. This is the address of the state that is about to be loaded into memory. When a state is loaded, this variable is updated with the address of the next state to be read from memory.

3.8.2. State Fields affecting Address Generation

The following are the fields in the memory word describing a state that control the duration of a state, the number of times it will be repeated, and the address from which the next state will be fetched.

3.8.2.1. SM[31:0] - TIME

This field specifies state duration in 50 nS units. A value of zero corresponds to the minimum state time of 100 nS (two clocks).

3.8.2.2. SM[55:32] - REPC

This field specifies the state repeat count. A value of zero requests that the state be repeated once, i.e., that the state be executed a total of two times. A state is only repeated if the RPTE bit for in the control field of that state is set.

3.8.2.3. SM[71:56] - CA

This field specifies the 16 bit address of a called subprogram.

3.8.2.4. SM[72] - C_R

This is the call/return bit in the control field.

3.8.2.5. SM[74] - RPTE

This is the repeat enable bit in the control field.

3.8.3. Next Address Generation Rules

Each time a state is loaded, the address of the next state is generated on the same clock that generated the address. This is done by applying a set of rules to the state variables a state field bits described above. The four possible types of “next” address are described in the paragraphs that follow. The categories are then further subdivided and the Boolean equations show the circumstance under which each case occurs.

3.8.3.1. Next Sequential Address

This is the case where the flow of address is not modified by a subprogram call or return, or by a state being repeated. This can be subdivided into two major cases:

3.8.3.1.1. No repeat is requested or in progress

The FIFO and RAM cases differ only in which repeat in progress bit must be consulted:

```
 !SUBA * !C_R * !RPTE * !RPTIP_F
+ SUBA * !C_R * !RPTE * !RPTIP_R
```

3.8.3.1.2. A repeat is in progress but the repeat counter is exhausted

The FIFO and RAM cases differ in which repeat in progress bit must be consulted and in which repeat counter must be checked. Note that if a repeat is in progress we do not need to check RPTE, since the instruction we are repeating must contain a repeat enable to have been repeated.

```
 !SUBA * !C_R * RPTIP_F * (RPT_F == 0)
+ SUBA * !C_R * RPTIP_R * (RPT_R == 0)
```

3.8.3.2. Same Address

This occurs only if a state is being repeated. This can be divided into three major cases:

3.8.3.2.1. Start of a repeat, no call or return

```
 !SUBA * !C_R * RPTE * !RPTIP_F
+ SUBA * !C_R * RPTE * !RPTIP_R
```

3.8.3.2.2. Unexhausted repeat in progress, no call or return

```
 !SUBA * !C_R * RPTIP_F * (RPT_F != 0)
+ SUBA * !C_R * RPTIP_R * (RPT_R != 0)
```

3.8.3.2.3. Return with unexhausted repeat

A subprogram state with a return and a repeat does not execute the return until the repeat is exhausted.

```
SUBA * C_R * RPTIP_R * (RPT_R != 0)
```

3.8.3.3. Call Address

A call occurs if a FIFO state has the C_R bit set. When a call occurs the next address is always the call address from the CA field of the state making the call. If the state contains a repeat, the call will be made each time the state is repeated.

```
!SUBA * C_R
```

3.8.3.4. Return Address

A return occurs if a subprogram state has the C_R bit set. If the state also contains a repeat the return does not execute until the repeat is exhausted. When a return occurs the next address is always the contents of the RA register.

```
  SUBA * C_R * !RPTE * !RPTIP_R
+ SUBA * C_R * RPTIP_R * (RPT_R == 0)
```

3.8.4. Return Address Generation Rules

Whenever a call occurs, the RA variable is loaded with the address of the FIFO state that will occur when the subprogram returns from the call. This is always the same as the address of the next FIFO state that would have been loaded if the call had not occurred.

3.8.4.1. Return Address is Next Sequential Address

This is the case where the flow of addresses is not modified by a repeat. This occurs if no repeat is requested, or if a repeat is in progress but exhausted.

```
  !SUBA * C_R * !RPTE * !RPTIP_F
+ !SUBA * C_R * RPTIP_F * (RPT_F == 0)
```

3.8.4.2. Return Address is the Same Address as the Calling State

This occurs only if a calling state is being repeated and the repeat is not exhausted.

```
  !SUBA * C_R * RPTE * !RPTIP_F
+ !SUBA * C_R * RPTIP_F * (RPT_F !=0)
```

3.8.5. Repeat Counter Control Rules

There are two repeat counters, one for repeated FIFO states and one for repeated subprogram states. At the end of each state the repeat counter is either loaded, decremented, or it remains the same.

3.8.5.1. Load FIFO State Repeat Counter

The counter is loaded if this is the first occurrence (i.e., not a repetition) of a repeated state.

```
!SUBA * RPTE * !RPTIP_F
```

3.8.5.2. Load Subprogram State Repeat Counter

This is similar to the FIFO state repeat counter case.

```
SUBA * RPTE * !RPTIP_R
```

3.8.5.3. Decrement FIFO State Repeat Counter

The FIFO state repeat counter (RPT_F) is decremented at the *logical* end of each repeated FIFO state. If the state *is not* a call, the logical end is the same as the actual end of the state. If the state *is* a call, the logical end of state is at the end of the last state in the called subprogram. The fact that the state is a call is determinable by the fact that the next address is a subprogram address (SUBA is set). Thus RPT_F is decremented only if SUBA is not set. Likewise, if the current state is a subprogram state with a return, the next state is from the FIFO and thus SUBA is not set. Thus if a FIFO state repeat is in progress, (RPTIP_F is set) and SUBA is not set, the current state can be either the actual FIFO state being repeated (if it is not a call) or the last state of the subprogram that it called. In either case, RPT_F should be decremented. Note that the counter is never decremented past zero.

```
!SUBA * RPTIP_F * (RPT_F != 0)
```

3.8.5.4. Decrement Subprogram State Repeat Counter

The subprogram state repeat counter is decremented at the end of each repeated subprogram state. The fact that the state is a repeated subprogram state is fully determinable by the fact that RPTIP_R is set, since this variable can only become set in a subprogram, and a subprogram cannot return until all repeated states have been completed.

```
RPTIP_R * (RPT_R != 0)
```

3.8.6. Rules for Updating other State Variables

3.8.6.1. SUBA

This variable indicates that the current address (i.e., the address of the next state to be loaded) is the address of a subprogram state. If the current address is that of a FIFO state, and the state contains a set C_R bit, then it is a call and the next address generated will be a subprogram address. SUBA will then remain set until a return occurs.

```
SUBA := !SUBA * C_R
      + SUBA * !(C_R *(!RPTE * !RPTIP_R + RPTIP_R * (RPT_R
== 0)))
```

3.8.6.2. RPTIP_F

This variables indicates that a FIFO state repeat is in progress. RPTIP_F remains set until a FIFO logical end of state occurs in which the FIFO repeat counter is exhausted.

```
RPTIP_F := !SUBA * RPTE * !RPTIP_F
         + RPTIP_F * !( !SUBA * (RPT_F == 0))
```

3.8.6.3. RPTIP_R

This variable indicates that a subprogram state repeat is in progress. The variable remains set until the subprogram repeat counter is exhausted.

```
RPTIP_R := SUBA * RPTE * !RPTIP_R
         + RPTIP_R * !(RPT_R == 0)
```

4. Notes

1) The inter-controller synchronization mechanism assumes that a state containing a *cm sync n* operation will not contain a *FIFO sync,* a *conditional action,* or a *DAP notify* operation. If this constaint is violated, the controller may signal that it is ready to continue when it is in fact paused waiting for the FIFO to fill, waiting for a continue command, or waiting for the DAP to respond.

2) The controller checks for the condition where the current memory read address (CMRA) is equal to the FIFO buffer starting address (FBSA) and the address is a subroutine address. If an attempt to load a state is made in this circumstance the controller halts with the RAM Error bit set. This can occur if a subroutine does not end with a state that has the call/return bit set. It *does not* check for the case where CMRA > FBSA for a subroutine address as might occur if an invalid call address were specified.

3) Any state which may contain a pause, i.e., any state that contains a set *DAP notfy, FIFO sync,* or *cm sync n* bit, must be at least three clocks long.