PAL is clocked by Clk20H (rising edge), with WrEn and Go gated externally to be asserted for just the second 40 MHz cycle (the low half of Clk20H).

Assumes message includes meaningless destination address with Head bit set.

IGC messages may not cross message boundaries; if unexpected 'Head' seen, will generate error and begin parsing next message.

Must wait at least two 40 MHz cycles after Go asserted before testing for !Busy, because of latch on Busy.

IF AEO and ALSB are asserted when Head is asserted, entire message is discarded. This feature is not active if un-expected Head is seen (one which generates SPErr). Flushed messages are not parsed.

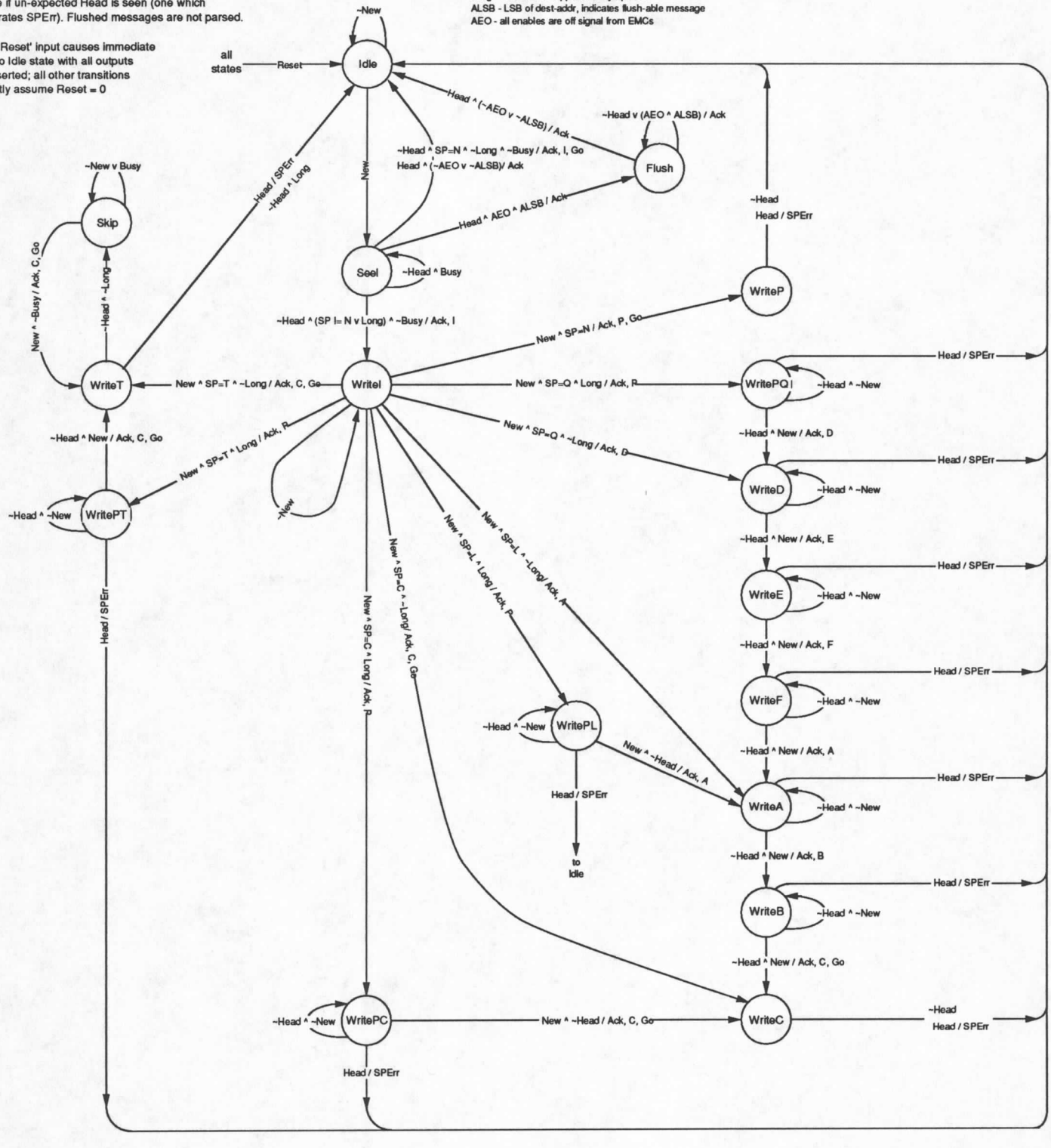
Note: 'Reset' input causes immediate jump to Idle state with all outputs de-asserted; all other transitions implicitly assume Reset = 0

Inputs:

Reset - unconditonally resets to Idle state
Head - resets to Idle and generates SP error if occurs at other than opcode position
New - indicates next word available in RxLatch
Busy - indicates IGC busy, no writes permitted
SP - stream parser code (3 bits), set to
N for no coefficients
C for C only
L for A, B, and C only
Q for A, B, C, D, E, and F
T for lookup table mode
Long - indicates P supplementary opcode
ALSB - LSB of dest-addr, indicates flush-able message
AEO - all enables are off signal from EMCs

Outputs:

Ack - asserted to indicate ready for next word
write (I, P, A, B, C, D, E, or F) - assert T<0:2>LLPR for specified word type and assert WrEnLLPR
Go - assert active low GoLLPR to the IGC, indicating that a new instruction is loaded
SPErr - assert active low error strobe SPErrL for one cycle to indicate reseting due to unexpected Head bit seen



State Diagram for Stream Parser FSM (RDSP)

State Diagram for IGC Port Status Transmitter FSM

PAL is clocked by Clk20H (rising edge). It is the type of FSM in which outputs are a function of the current state only.

high-low transition of SendStH triggers transmission of a status message. To send a status message, the IGC first waits for StIPH=0, then asserts ExtOp (latched to produce SendStH) for 32 cycles to shift in return address, and 3 bytes of status ID during the last 3 cycles (low byte first), and then de-asserts SendStH.

An error message is enabled by SendErrH input, and the message is sent after any message in progress is completed.

Error messages are always sent to Ring address 0x00000000. Their status ID word is that of the previous ordinary status message, or of one in progress that was interrupted.

If an error occurs after IGC has detected StIPH=0, but before it has asserted SendStH, the error message will be sent, and the status message is lost.

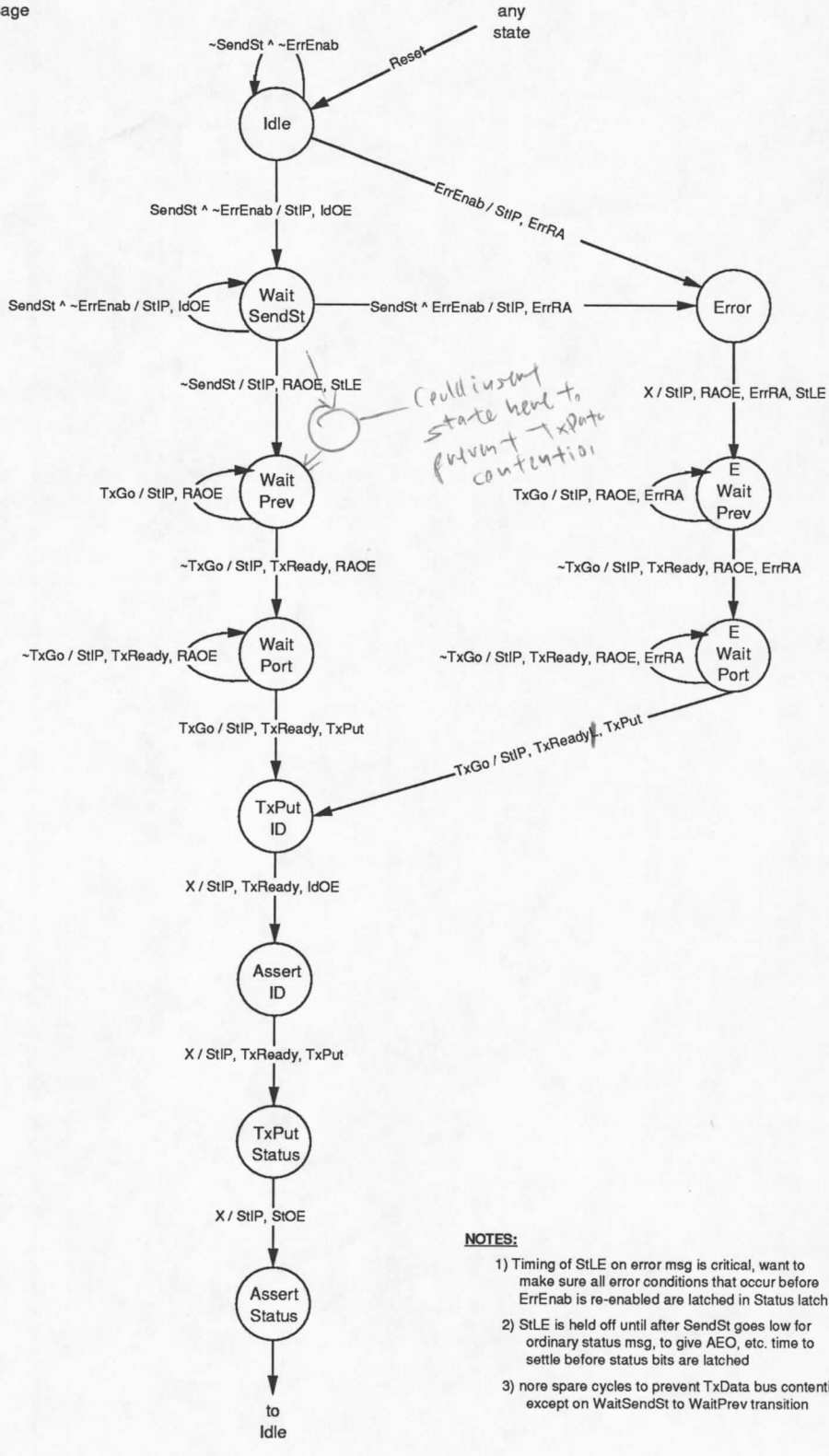
'Reset' input causes immediate jump to Idle state. All other transitions implicitly assume ~Reset.

Inputs:

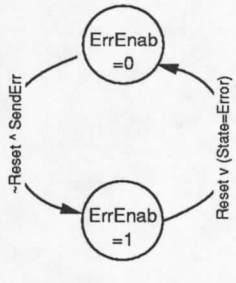
- Reset - unconditonally resets to Idle state
- TxGo - handshake signal from Ring transmit port
- SendSt - goes high (for many cycles) then low to initiate transmission of status message
- SendErr - sets the ErrEnab flag, which itself is a state machine input which causes the transmission of an error message at the next available opportunity (SendErr represents the logical-OR of various error strobes)

Outputs:

- StIP - assert StIPHL20R, indicating that transmission of a status or error message is in progress; SendSt must not be asserted when StIP is high
- TxReady - assert active low transmit handshake signal TxReadyL to Ring
- TxPut - asserted one cycle before valid data onto TxData
- StOE - assert active low StOEL to enable status latch onto TxData wires
- !dOE - assert active low IdOEL to enable status mesage ID onto TxData wires
- RAOE - assert active low RAOEL to enable return address onto TxData wires
- ErrRA - assert active low ErrRAL to clear return address latch
- StLE - assert active low StLEL to latch status word



ErrEnab is a flag which acts as an input to the state machine. It is set by 'SendErr' and cleared when the state machine leaves the 'Error' state. ErrEnab canot go high again without first going low (clear overrides set).



NOTES:

1) Timing of StLE on error msg is critical, want to make sure all error conditions that occur before ErrEnab is re-enabled are latched in Status latch
2) StLE is held off until after SendSt goes low for ordinary status msg, to give AEO, etc. time to settle before status bits are latched
3) nore spare cycles to prevent TxData bus contention except on WaitSendSt to WaitPrev transition