

AMD PCSCSI driver State machine transitions

This document describes the various state machines implemented in chip-specific modules of the AMDPCSCSI driver. Each description includes the name of the instance variable variable which contains the state (e.g., **scState**), it data type (e.g., **scState_t**), and a list describing where (what method) and when each transition occurs.

scState (scState_t)

This state machine is the basic phase-change following mechanism.

from	to	where	when
-----	-----	-----	
unknown	SCS_UNINITIALIZED	-archInit	initial probe
xxx	SCS_DISCONNECTED	-hwReset	Any hardware reset
xxx	SCS_DISCONNECTED	-hwInterrupt	IS_DISCONNECT interrupt
SCS_DISCONNECTED	SCS_SELECTING	-hwStart:	start of select sequence
SCS_DISCONNECTED	SCS_ACCEPTINGMSG	-fsmDisconnected	valid Reselect detected
SCS_SELECTING	SCS_DISCONNECTED	-fsmSelecting	Select Timeout
SCS_SELECTING	SCS_INITIATOR	-fsmSelecting	select sequence interrupt (may be incomplete)
SCS_SELECTING	SCS_DISCONNECTED	-fsmSelecting	reselect during select attempt

SCS_INITIATOR	SCS_SENDINGCMD	-fsmPhaseChange	Phase change to cmd out
SCS_INITIATOR	SCS_COMPLETING	-fsmPhaseChange	Phase change to status in
SCS_INITIATOR	SCS_GETTINGMSG	-fsmPhaseChange	Phase change to msg in
SCS_INITIATOR	SCS_SENDINGMSG	-fsmPhaseChange	Phase change to msg out, sent msg byte(s) to FIFO
SCS_INITIATOR	SCS_DMAING	-dmaStart	DMA setup after phase change to data in/out
SCS_SENDINGCMD	SCS_INITIATOR	-fsmSendingCmd	good interrupt after sending cmd
SCS_COMPLETING	SCS_ACCEPTINGMSG	-fsmCompleting	Good SCMD_INIT_CMD_CMPLT
SCS_COMPLETING	SCS_INITIATOR	-fsmCompleting	SCMD_INIT_CMD_CMPLT, but only status byte seen
SCS_GETTINGMSG	SCS_ACCEPTINGMSG	-fsmGettingMsg	SCMD_MSG_ACCEPTED cmd sent
SCS_ACCEPTINGMSG	SCS_GETTINGMSG	-fsmAcceptingMsg	Phase still msg in
SCS_ACCEPTINGMSG	SCS_INITIATOR	-fsmAcceptingMsg	Phase != msg in
SCS_ACCEPTINGMSG	SCS_DISCONNECTED	-fsmAcceptingMsg	Msg = Cmd complete
SCS_ACCEPTINGMSG	SCS_DISCONNECTED	-fsmAcceptingMsg	Msg = disconnect
SCS_ACCEPTINGMSG	SCS_INITIATOR	-fsmAcceptingMsg	Any other msg
SCS_DMAING	SCS_INITIATOR	-fsmDMAing	DMA complete

SCS_SENDINGMSG SCS_INITIATOR -fsmSendingMsg always

msgOutState (msgOutState_t)

The purpose of this is to provide a means to send a message out to the target. Basically, to do a message out:

- Set ATN true
- Place message byte(s) in currMsgOut[]
- Set currMsgOutCnt to length of message
- Set msgOutState = MOS_WAITING

All of these are done in -messageOut: for a single-byte message. For extended messages, context-specific code must perform these manually.

When we see a phase change to message out, -fsmPhaseChange sends the message byte(s) from currMsgOut[] to the FIFO, does a SCMD_TRANSFER_INFO, and sets msgOutState to MOS_SAWMSGOUT. We go back to MOS_NONE when we see any other phase.

from	to	where	when
-----	-----	-----	
xxx	MOS_NONE	-hwReset	Hardware reset
xxx	MOS_NONE	-hwStart:	start of select sequence
MOS_NONE	MOS_WAITING	-fsmSelecting	Select complete, set ATN
			true, trying to send

MOS_NONE	MOS_WAITING	-fsmAcceptingMsg	Target-init SDTR message
MOS_NONE	MOS_WAITING	-messageOut:	Set ATN in hopes of doing a single-byte msg out
MOS_WAITING	MOS_SAWMSGOUT	-fsmPhaseChange	Phase change to msg out, sent msg byte(s) to FIFO
MOS_SAWMSGOUT	MOS_NONE	-fsmPhaseChange	Phase != msg in or msg out

SDTR_State (SDTR_State_t)

This state machine keeps track of SDTR negotiations, both target- and host-initiated.

from	to	where	when
xxx	SNS_NONE	-hwReset	Hardware reset
xxx	SNS_HOST_INIT_NEEDED	-hwStart:	We want to do SDTR
xxx	SNS_NONE	-hwStart:	We don't want to do
SNS_HOST_INIT_NEEDED	SNS_NONE	-fsmSelecting	No message phase after Selection complete
SNS_HOST_INIT_NEEDED	SNS_HOST_INIT	-fsmPhaseChange	Phase = msg out after selection

SNS_HOST_INIT	SNS_NONE	-fsmAcceptingMsg	Our SDTR resulted in msg reject
SNS_HOST_INIT	SNS_NONE	-fsmAcceptingMsg	Target's SDTR response recv'd
SNS_HOST_INIT	SNS_NONE	-fsmPhaseChange	Attempted host-init'd SDTR, but no msg in
SNS_NONE	SNS_TARGET_INIT	-fsmGettingMsg	Saw target-init'd SDTR
SNS_TARGET_INIT	SNS_NONE	-fsmSendingMsg	Completed target-init'd SDTR