.DSP Driver Routines

Overview

The .DSP Driver implements the special communications protocol called the **AppleTalk DSP (ADSP)**. You send commands to **ADSP** and obtain information about **ADSP** by executing **The .DSP Driver** routines described in this section.

Each routine is implemented as a call to the **Device Manager**'s **PBControl** function, as follows:

OSErr	**PBControl(***thePBptr, async***);**	
ParmBlkPtr	*thePBptr*;	address of a parameter block structure
Boolean	*async*;	0=await completion; 1=immediate return
	returns	Error Code; 0=no error

thePBptr is a pointer to the parameter block used by the **PBControl** function for .DSP routines

async is a Boolean that specifies whether the function is to execute synchronously or asynchronously. Set the async parameter to TRUE to execute the function asynchronously.

The parameter block is shown in the **DSPParamBlock** . The parameters used with each function are described in this section.

For a general discussion of the use of **ADSP**, see **Using ADSP** in the section **AppleTalk DSP (ADSP)**.

Establishing and Terminating an ADSP Connection

You can use the routines described in this section to

- establish a connection end
- set the values for parameters that control the behavior of a connection end
- open a connection
- assign an identification number to a connection end
- close a connection end
- eliminate a connection end

dspInit

Parameter block

Out-In	Name	Type	Size	Offset	Description
←	ioResult	short	2	16	Result code
→	ioCRefNum	short	2	24	Driver reference number
→	csCode	short	2	26	Always dspInit
←	ccbRefNum	short	2	32	Reference number of **CCB**
→	ccbPtr	long	4	34	Pointer to**CCB**
→	userRoutine	long	4	38	Pointer to routine to call on connection events
→	sendQSize	short	2	42	Size in bytes of the send queue
→	sendQueue	long	4	44	Pointer to send queue

→	recvQSize	short	2	48	Size in bytes of the receive queue
→	recvQueue	long	4	50	Pointer to receive queue
→	attnPtr	long	4	54	Pointer to buffer for incoming attention messages
↔	localSocket	char	1	58	DDP socket number for this connection end

The dspInit routine establishes a connection end; that is, it assigns a specific socket for use by **ADSP** and initializes the variables that **ADSP** uses to maintain the connection. The dspInit routine does not open the connection end or establish a connection with a remote connection end; you must follow the dspInit routine with the dspOpen routine to perform those tasks. Use the dspCLInit routine to establish a connection listener. Use the dspRemove routine to eliminate a connection end.

When you send bytes to a remote connection end, **ADSP** stores the bytes in a buffer called the *send queue.* Until the remote connection end acknowledges their receipt, **ADSP** keeps the bytes you sent in the send queue so that they are available to be retransmitted if necessary. When the local connection end receives bytes, it stores them in a buffer called the *receive queue* until you read them.

You must allocate memory for the send and receive queues and for a buffer that holds incoming attention messages. You must also allocate a nonrelocatable block of memory for the **CCB** for this connection end.

Note: When you call the dspInit routine, the memory that you allocate becomes the property of **ADSP**. You cannot write any data to this memory except by calling **ADSP** routines, and you must ensure that the memory remains locked until you call the dspRemove routine to eliminate the connection end.

Field descriptions

ioResult	The result of the routine. When you execute the routine asynchronously, the routine sets this parameter to 1 and returns a routine result of noErr as soon as the routine begins execution. When the routine completes execution, it sets the ioResult parameter to the actual result code.
ioCRefNum	The driver reference number. This parameter is returned by the **OpenDriver** function. You must specify this number every time you call **The .DSP Driver**.
csCode	The routine selector, always equal to dspInit for this routine.
ccbRefNum	The **CCB** reference number. The dspInit routine returns this number. You must provide this number in all subsequent calls to this connection end.
ccbPtr	A pointer to the **CCB** that you allocated. The **CCB** is 242 bytes in size and is described in **Connection Control Block** .
userRoutine	A pointer to a routine that is to be called each time the connection end receives an unsolicited connection event. Specify NIL for this parameter if you do not want to supply a user routine. Connection events and user routines are discussed in the section entitled **Writing a Connection Routine** .

sendQSize	The size in bytes of the send queue. A queue size of 600 bytes should work well for most applications. If you are using **ADSP** to send a continuous flow of data, a larger data buffer improves performance. If your application is sending the user's keystrokes, a smaller buffer should be adequate. The constant minDSPQueueSize indicates the minimum queue size that you can use.
sendQueue	A pointer to the send queue that you allocated.
recvQSize	The size in bytes of the receive queue. A queue size of 600 bytes should work well for most applications. If you are using **ADSP** to receive a continuous flow of data, a larger data buffer improves performance. If your application is receiving a user's keystrokes, a smaller buffer should be adequate. The constant minDSPQueueSize indicates the minimum queue size that you can use.
recvQueue	A pointer to the receive queue that you allocated.
attnPtr	A pointer to the attention-message buffer that you allocated. The attention-message buffer must be the size of the constant attnBufSize.
localSocket	The DDP socket number of the socket that you want **ADSP** to use for this connection end. Specify 0 for this parameter to cause **ADSP** to assign the socket. In the latter case, **ADSP** returns the socket number when the dspInit routine completes execution.

Result codes

noErr	(0)	No error
ddpSktErr	(-91)	Error opening socket
DSPQueueSize	(-1274)	Send or receive queue is too small

dspOptions

Parameter block

Out-In	Name	Type	Size	Offset	Description
←	ioResult	short	2	16	Result code
→	ioCRefNum	short	2	24	Driver reference number
→	csCode	short	2	26	Always dspInit
→	ccbRefNum	short	2	32	Reference number of **CCB**
→	sendBlocking	short	2	34	Send-blocking threshold
→	badSeqMax	char	1	38	Threshold to send retransmit advice
→	useCheckSum	char	1	39	DDP checksum flag

The dspOptions routine allows you to set values for several parameters that affect the behavior of the local connection end. You can set the options for any established connection end, whether open or not.

Field descriptions

ioResult	The result of the routine. When you execute the routine asynchronously, the routine sets this parameter to 1 and returns a routine result of noErr as soon as the routine begins execution. When the routine completes execution, it sets the ioResult parameter to the actual result code.

ioCRefNum	The driver reference number. This parameter is returned by the **OpenDriver** function. You must specify this number every time you call **The .DSP Driver**.
csCode	The routine selector, always equal to dspOptions for this routine.
ccbRefNum	The **CCB** reference number that was returned by the dspInit routine.
sendBlocking	The maximum number of bytes that may accumulate in the send queue before **ADSP** sends a packet to the remote connection end. **ADSP** sends a packet before the maximum number of bytes accumulates if the period specified by the send timer expires, if you execute the dspWrite routine with the flush parameter set to 1, or if a connection event requires that the local connection end send an acknowledgment packet to the remote connection end. You can set the sendBlocking parameter to any value from 1 byte to the maximum size of a packet (572 bytes). If you set the sendBlocking parameter to 0, the current value for this parameter is not changed. The default value for the sendBlocking parameter is 16 bytes.
badSeqMax	The maximum number of out-of-sequence data packets that the local connection end can receive before requesting the remote connection end to retransmit the missing data. Because a connection end does not acknowledge the receipt of a data packet received out of sequence, the retransmit timer of the remote connection end will expire eventually and the connection end will retransmit the data. The badSeqMax parameter allows you to cause the data to be retransmitted before the retransmit timer of the remote connection end has expired. You can set the badSeqMax parameter to any value from 1 to 255. If you set the badSeqMax parameter to 0, the current value for this parameter is not changed. The default value for the badSeqMax parameter is 3.
useCheckSum	A flag specifying whether DDP should compute a checksum and include it in each packet that it sends to the remote connection end. Set this parameter to 1 if you want DDP to use checksums or to 0 if you do not want DDP to use checksums. The default value for useCheckSum is 0. **ADSP** cannot include a checksum in a packet that has a short DDP header-that is, a packet being sent over LocalTalk to a remote socket that is on the same cable as the local socket. Note that the useCheckSum parameter affects only whether **ADSP** includes a checksum in a packet that it is sending. If **ADSP** receives a packet that includes a checksum, it validates the checksum regardless of the setting of the useCheckSum parameter.

Result codes

noErr	(0)	No error
RefNum	(-1280)	Bad connection reference number

dspOpen

Parameter block

Out-In	Name	Type	Size	Offset	Description
←	ioResult	short	2	16	Result code
→	ioCRefNum	short	2	24	Driver reference number
→	csCode	short	2	26	Always dspInit
→	ccbRefNum	short	2	32	Reference number of**CCB**
←	localCID	short	2	34	ID of this connection end
↔	remoteCID	short	I2	36	ID of remote connection end
↔	remoteAddress	long	4	38	Remote internet address
→	filterAddress	long	4	42	Filter for open-connection requests
↔	sendSeq	long	4	46	Initial send sequence number
↔	sendWindow	short	2	50	Initial size of remote receive queue
→	recvSeq	long	4	52	Initial receive sequence number
↔	attnSendSeq	long	4	56	Attention send sequence number
→	attnRecvSeq	long	4	60	Attention receive sequence number
→	ocMode	char	1	64	Connection-opening mode
→	ocInterval	char	1	65	Interval between open requests
→	ocMaximum	char	1	66	Retries of open-connection request

You use the ocMode field of the parameter block to specify the opening mode that the dspOpen routine is to use. The dspOpen routine puts a connection end into one of the four following opening modes:

- The ocRequest mode, in which **ADSP** attempts to open a connection with the socket at the internet address you specify with the remoteAddress parameter. If the socket you specify as a remote address is a connection listener, it is possible that your application will receive a connection acknowledgment and request from a different address than the one to which you sent the open-connection request. You can use the filterAddress parameter to restrict the addresses with which you will accept a connection.

  The dspOpen routine completes execution in the ocRequest mode when one of the following occurs: **ADSP** establishes a connection, your connection end receives a connection denial from the remote connection end, your connection end denies the connection request returned by a connection listener, or **ADSP** cannot complete the connection within the maximum number of retries that you specified with the ocMaximum parameter.

- The ocPassive mode, in which the connection end waits to receive an open-connection request from a remote connection end. You can use the filterAddress parameter to restrict the addresses from which you will accept a connection request.

  The dspOpen routine completes execution in the ocPassive mode when **ADSP** establishes a connection or when either connection end receives a connection denial.

- The ocAccept mode, used by connection servers to complete an open-connection dialog. When a connection server is informed by its connection listener that the connection listener has received an open-connection request, the connection server calls the dspInit routine to establish a connection end and then calls the dspOpen routine in ocAccept mode to complete the connection. You must obtain the following parameters from the dspCLListen routine and provide them to the dspOpen routine: *remoteAddress*, *remoteCID*, *sendSeq*, *sendWindow*, and *attnSendSeq*. Connection listeners and connection servers are described in **Making a Connection Listener** and in **Establishing and Terminating an ADSP Connection**. under the section **.DSP Driver Routines**

The dspOpen routine completes execution in the ocAccept mode when **ADSP** establishes a connection or when either connection end receives a connection denial.

- The ocEstablish mode, in which **ADSP** considers the connection end established and the connection state open. This mode is for use by clients that determine their connection-opening parameters without using **ADSP** or **The .DSP Driver** to do so.

  You must first use the dspInit routine to establish a connection end and then execute the **dspNewCID** routine to obtain an identification number (ID) for the local connection end. You must then communicate with the remote connection end to send it the local connection ID and to determine the values of the following parameters: remoteAddress, remoteCID, sendSeq, sendWindow, recvSeq, attnSendSeq, and attnRecvSeq. Only then can you execute the dspOpen routine in the ocEstablish mode.

  The dspOpen routine completes execution in the ocEstablish mode immediately.

The use of parameters by the dspOpen routine depends on the mode in which the routine is executed, as follows:

ocRequest	**ocPassive**	**ocAccept**	**ocEstablish**
← ioResult	← ioResult	← ioResult	← ioResult
→ ioCRefNum	→ ioCRefNum	→ ioCRefNum	→ ioCRefNum
→ csCode	→ csCode	→ csCode	→ csCode
→ ccbRefNum	→ ccbRefNum	→ ccbRefNum	→ ccbRefNum
← localCID	← localCID	← localCID	– localCID
← remoteCID	← remoteCID	→ remoteCID	→ remoteCID
→ remoteAddress	← remoteAddress	→ remoteAddress	→ remoteAddress
→ filterAddress	→ filterAddress	– filterAddress	– filterAddress
← sendSeq	← sendSeq	→ sendSeq	→ sendSeq
← sendWindow	← sendWindow	→ sendWindow	→ sendWindow
– recvSeq	– recvSeq	– recvSeq	→ recvSeq
← attnSendSeq	← attnSendSeq	→ attnSendSeq	→ attnSendSeq
– attnRecvSeq	– attnRecvSeq	– attnRecvSeq	→ attnRecvSeq
→ ocMode	→ ocMode	→ ocMode	→ ocMode
→ ocInterval	→ ocInterval	→ ocInterval	– ocInterval
→ ocMaximum	→ ocMaximum	→ ocMaximum	– ocMaximum

Key:
→ input ← output - not used

Field descriptions

ioResult	The result of the routine. When you execute the routine asynchronously, the routine sets this parameter to 1 and returns a routine result of noErr as soon as the routine begins execution. When the routine completes execution, it sets the ioResult parameter to the actual result code.
ioCRefNum	The driver reference number. This parameter is returned by the **OpenDriver** function. You must specify this number every time you call **The .DSP Driver**.
csCode	The routine selector, always equal to dspOpen for this routine.
ccbRefNum	The **CCB** reference number that was returned by the dspInit routine for the connection end that you want to use.
localCID	The identification number of the local connection end. This number is assigned by **ADSP** when the connection is opened. **ADSP** includes this number in every packet sent to a remote connection end. Before you call the dspOpen routine in ocEstablish mode, you must call the dspNewCID routine to cause **ADSP** to assign this value.
remoteCID	The identification number of the remote connection end. This parameter is returned by the dspOpen routine in the ocRequest and ocPassive modes. A connection server must provide this number to the dspOpen routine when the server executes the routine in ocAccept mode; in this case, the connection server obtains the *remoteCID* value from the dspCLListen routine. You must provide the *remoteCID* value to the dspOpen routine when you use the routine in ocEstablish mode.
remoteAddress	The internet address of the remote socket with which you wish to establish communications. This address consists of a 2-byte network number, a 1-byte node ID, and a 1-byte socket number. You must provide this parameter when you call the dspOpen routine in the ocRequest or ocEstablish mode. This parameter is returned by the dspOpen routine when you call the routine in the ocPassive mode. When you call the dspOpen routine in the ocAccept mode, you must use the value for the remoteAddress parameter that was returned by the dspCLListen routine.
filterAddress	The internet address of the socket from which you will accept a connection request. The address consists of three fields: a 2-byte network number, a 1-byte node ID, and a 1-byte socket number. Specify 0 for any of these fields for which you wish to impose no restrictions. If you specify a filter address of 0x000082500, for example, the connection end accepts a connection request from any socket at node 0x025 of network 0x00008. Set the filterAddress parameter equal to the remoteAddress parameter to accept a connection only with the socket to which you sent a connection request. When you execute the dspOpen routine in the ocPassive mode,

you can receive a connection request from any **ADSP** connection end on the internet. When you execute the dspOpen routine in the ocRequest mode, your connection end can receive a connection request acknowledgment from an address different from the one you specified in the remoteAddress parameter only if the remote address you specified was that of a connection listener. In either case, you can use the filterAddress parameter to avoid acknowledging unwanted connection requests.

When you execute the dspOpen routine in the ocAccept mode, your connection listener has already received and decided to accept the connection request. You can specify a filter address for a connection listener with the dspCLListen routine. A connection server can use the dspCLDeny routine to deny a connection request that was accepted by its connection listener.

You cannot use the filter address when you execute the dspOpen routine in ocEstablish mode.

sendSeq	The sequence number of the first byte that the local connection end will send to the remote connection end **ADSP** uses this number to coordinate communications and to check for errors **ADSP** returns a value for the sendSeq parameter when you execute the dspOpen routine in the ocRequest or ocPassive mode. When you execute the dspOpen routine in the ocAccept mode, you must specify the value for the sendSeq parameter that was returned by the dspCLListen routine. You must provide the value for this parameter when you execute the dspOpen routine in the ocEstablish mode.
sendWindow	The sequence number of the last byte that the remote connection end has buffer space to receive. **ADSP** uses this number to coordinate communications and to check for errors. **ADSP** returns a value for the sendWindow parameter when you execute the dspOpen routine in the ocRequest or ocPassive mode. When you execute the dspOpen routine in the ocAccept mode, you must specify the value for the sendWindow parameter that was returned by the dspCLListen routine. You must provide the value for this parameter when you execute the dspOpen routine in the ocEstablish mode.
recvSeq	The sequence number of the next byte that the local connection end expects to receive. **ADSP** uses this number to coordinate communications and to check for errors. You must provide the value for this parameter when you execute the dspOpen routine in the ocEstablish mode. The dspOpen routine does not use this parameter when you execute it in any other mode.
SendSeq	The sequence number of the next attention packet that the local connection end will transmit. **ADSP** uses this number to coordinate communications and to check for errors. **ADSP** returns a value for the attnSendSeq parameter when you execute the dspOpen routine in the ocRequest or ocPassive mode. When you execute the dspOpen routine in the ocAccept mode, you must specify the value for the attnSendSeq parameter that was returned by the dspCLListen routine. You must provide the value

for this parameter when you execute the dspOpen routine in the ocEstablish mode.

attnRecvSeq	The sequence number of the next attention packet that the local connection end expects to receive. **ADSP** uses this number to coordinate communications and to check for errors. You must provide the value for this parameter when you execute the dspOpen routine in the ocEstablish mode. The dspOpen routine does not use this parameter when you execute it in any other mode.
ocMode	The mode in which the dspOpen routine is to operate, as follows:

Mode	**Value**	**Meaning**
ocRequest	1	**ADSP** attempts to open a connection with the socket you specify.
ocPassive	2	The connection end waits to receive a connection request.
ocAccept	3	The connection server accepts and acknowledges receipt of a connection request.

ocInterval	The period between transmissions of open-connection requests. If the remote connection end does not acknowledge or deny an open-connection request, **ADSP** retransmits the request after a time period specified by this parameter. The time period used by **ADSP** is (ocInterval×10)ticks, or (ocInterval / 6)seconds. For example, if you set the ocInterval parameter to 3, the time period between retransmissions is 30 ticks (1/2 second). You can set the ocInterval parameter to any value from 1 (1/6 second) to 180 (30 seconds). If you specify 0 for the ocInterval parameter, **ADSP** uses the default value of 6 (1second). You must provide a value for the ocInterval parameter when you execute the dspOpen routine in the ocRequest, ocPassive, or ocAccept mode. The dspOpen routine does not use this parameter when you execute it in the ocEstablish mode.
ocMaximum	The maximum number of times to retransmit an open-connection request before **ADSP** terminates execution of the dspOpen routine. If you specify 0 for the ocMaximum parameter, **ADSP** uses the default value of 3. If you specify 255 for the ocMaximum parameter, **ADSP** retransmits the open-connection request indefinitely until the remote connection end either acknowledges or denies the request. You must provide a value for the ocMaximum parameter when you execute the dspOpen routine in the ocRequest, ocPassive, or ocAccept mode. The dspOpen routine does not use this parameter when you execute it in the ocEstablish mode.

Result codes

noErr	(0)	No error
OpenDenied	(-1273)	Open request denied by recipient
errOpening	(-1277)	Attempt to open connection failed
errState	(-1278)	Connection end must be closed
errAborted	(-1279)	Request aborted by the dspRemove or by the dspClose routine
errRefNum	(-1280)	Bad connection reference number

dspNewCID

Parameter block

Out-In	Name	Type	Size	Offset	Description
←	ioResult	short	2	16	Result code
→	ioCRefNum	short	2	24	Driver reference number
→	csCode	short	2	26	Always dspNewCID
→	ccbRefNum	short	2	32	Reference number of **CCB**
←	newCID	short	2	34	ID of new connection

The dspNewCID routine causes **ADSP** to assign an ID to a connection end without opening the connection end or attempting to establish a connection with a remote connection end. Use this routine only if you implement your own protocol to establish communication with a remote connection end. You must first use the dspInit routine to establish a connection end. Next, you must call the dspNewCID routine to obtain a connection-end ID. Then you must establish communication with a remote connection end and pass the ID to the remote connection end. Finally, you must call the dspOpen routine in ocEstablish mode to cause ADSP to open the connection. See the description of the dspOpen routine for more information on establishing a connection in this fashion.

The ioResult parameter returns the result of the routine. If you call the routine asynchronously, the routine sets this field to 1 as soon as it begins execution, and it changes the field to the actual result code when it completes execution. The ioCRefNum parameter is the driver reference number returned by the **OpenDriver** function. You must specify this number every time you call **The .DSP Driver**. The csCode parameter is the routine selector; it is always dspNewCID for this routine. The ccbRefNum parameter is the CCB reference number that was returned by the dspInit routine. The newCID parameter is the connection-end ID returned by this routine. You must provide this number to the client of the remote connection end so that it can use it for the *remoteCID* parameter when it calls the dspOpen routine.

	Result codes	
noErr	(0)	No error
errState	(-1278)	Connection is not closed
errRefNum	(-1280)	Bad connection reference number

dspClose

Parameter block

Out-In	Name	Type	Size	Offset	Description
←	ioResult	short	2	16	Result code
→	ioCRefNum	short	2	24	Driver reference number
→	csCode	short	2	26	Always dspClose
→	ccbRefNum	short	2	32	Reference number of **CCB**
→	abort	char	1	34	Abort send requests if not 0

The dspClose routine closes the connection end. The connection end is still established; that is, **ADSP** retains ownership of the **CCB**, send queue, receive queue, and attention-message buffer. You can continue to read bytes from the receive queue after you have called the dspClose routine. Use the dspRemove routine instead of the dspClose routine if you are through reading bytes from the receive queue and want to release the memory associated with the connection end. The dspClose routine does not return an error if you call it for a connection end that is already closed.

The ioResult parameter returns the result of the routine. If you call the routine asynchronously, the routine sets this field to 1 as soon as it begins execution, and it changes the field to the actual result code when it completes execution. The ioCRefNum parameter is the driver reference number returned by the **OpenDriver** function. You must specify this number every time you call **The .DSP Driver**. The csCode parameter is the routine selector; it is always dspClose for this routine. The ccbRefNum parameter is the **CCB** reference number that was returned by the dspInit routine. If the abort parameter is nonzero, **ADSP** cancels any outstanding requests to send data packets (such as the dspAttention routine) and discards all data in the send queue. If the abort parameter is 0, **ADSP** does not close the connection end until all of the data in the send queue and all outstanding attention messages have been sent and acknowledged.

	Result codes	
noErr	(0)	No error
errRefNum	(-1280)	Bad connection reference number

dspRemove

Parameter block

Out-In	Name	Type	Size	Offset	Description
←	ioResult	short	2	16	Result code
→	ioCRefNum	short	2	24	Driver reference number
→	csCode	short	2	26	Always dspRemove
→	ccbRefNum	short	2	32	Reference number of **CCB**
→	abort	char	1	34	Abort connection if not 0

The dspRemove routine closes any open connection and eliminates the connection end; that is, **ADSP** no longer retains control of the **CCB**, send queue, receive queue, and attention-message buffer. You cannot continue to read bytes from the receive queue after you have called the dspRemove routine. After you call the dspRemove routine, you can release all of the memory you allocated for the connection end if you do not intend to reopen the connection end.

The ioResult parameter returns the result of the routine. If you call the routine asynchronously, the routine sets this field to 1 as soon as it begins execution, and it changes the field to the actual result code when it completes execution. The ioCRefNum parameter is the driver reference number returned by the **OpenDriver** function. You must specify this number every time you call **The .DSP Driver**. The csCode parameter is the routine selector, always dspRemove for this routine. The ccbRefNum parameter is the **CCB** reference number that was returned by the dspInit routine. If the abort parameter is nonzero, ADSP cancels any outstanding requests to send data packets (such as the dspAttention routine) and discards all data in the send queue. If the abort parameter is 0, **ADSP** does not close the connection end until all of the data in the send queue has been sent and acknowledged.

	Result codes	
noErr	(0)	No error
errRefNum	(-1280)	Bad connection reference number