

Devices_Manual

COLLABORATORS

	<i>TITLE :</i> Devices_Manual		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 18, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Devices_Manual	1
1.1	Amiga® RKM Devices: 8 Narrator Device	1
1.2	8 Narrator Device / Narrator Device Commands and Functions	2
1.3	8 Narrator Device / Device Interface	3
1.4	8 / Device Interface / The Amiga Speech System	4
1.5	8 / Device Interface / Opening The Narrator Device	5
1.6	8 / Device Interface / Closing The Narrator Device	7
1.7	8 Narrator Device / Writing to the Narrator Device	7
1.8	8 Narrator Device / Reading from the Narrator Device	11
1.9	8 Narrator Device / How to Write Phonetically for Narrator	14
1.10	8 / How to Write Phonetically for Narrator / Phonetic Spelling	15
1.11	8 / Phonetic Spelling / Choosing the Right Vowel	16
1.12	8 / Phonetic Spelling / Choosing the Right Consonant	17
1.13	8 / Phonetic Spelling / Contractions and Special Symbols	17
1.14	8 / How to Write Phonetically for Narrator / Stress And Intonation	18
1.15	8 // Stress And Intonation / How and Where to Put the Stress Marks	18
1.16	8 // Stress And Intonation / Which Stress Value Do I Use?	19
1.17	8 / How to Write Phonetically for Narrator / Punctuation	20
1.18	8 / How to Write Phonetically / Hints For Intelligibility	20
1.19	8 / Writing Phonetically / English And Phonetic Text Example	21
1.20	8 / How to Write Phonetically for Narrator / Concluding Remarks	21
1.21	8 Narrator Device / A More Technical Explanation	21
1.22	8 Narrator Device / Additional Information on the Narrator Device	23

Chapter 1

Devices_Manual

1.1 Amiga® RKM Devices: 8 Narrator Device

This chapter describes the narrator device which, together with the translator library, provides all of the Amiga's text-to-speech functions. The narrator device is used to produce high-quality human-like speech in real time.

NEW NARRATOR FEATURES FOR VERSION 2.0

Feature	Description	Function
NDB_NEWIORB	Flag	Use V37 features
NDB_WORDSYNC	Flag	Synchronize speech/mouth on words
NDB_SYLSYNC	Flag	Synchronize speech/mouth on syllables
F0enthusiasm	narrator_rb field	F0 excursion factor
F0perturb	narrator_rb field	Amount of F0 perturbation
F1adj	narrator_rb field	F1 adjustment in $\pm 5\%$ steps
F2adj	narrator_rb field	F2 adjustment in $\pm 5\%$ steps
F3adj	narrator_rb field	F3 adjustment in $\pm 5\%$ steps
A1adj	narrator_rb field	A1 adjustment in decibels
A2adj	narrator_rb field	A2 adjustment in decibels
A3adj	narrator_rb field	A3 adjustment in decibels
articulate	narrator_rb field	Transition time multiplier
centralize	narrator_rb field	Degree of vowel centralization
centphon	narrator_rb field	Pointer to central ASCII phon
AVbias	narrator_rb field	Amplitude of voicing bias
AFbias	narrator_rb field	Amplitude of frication bias
priority	narrator_rb field	Priority while speaking

Compatibility Warning:

The new features for the 2.0 narrator device are not backwards compatible.

Narrator Device Commands and Functions
 Device Interface
 Writing to the Narrator Device
 Reading from the Narrator Device
 How to Write Phonetically for Narrator
 A More Technical Explanation

Example Speech and Mouth Movement Program
 Additional Information on the Narrator Device

1.2 8 Narrator Device / Narrator Device Commands and Functions

Command -----	Operation -----
CMD_FLUSH	Purge all active and queued requests for the narrator device.
CMD_READ	Read mouth shapes associated with an active write from the narrator device.
CMD_RESET	Reset the narrator port to its initialized state. All active and queued I/O requests will be aborted. Restarts the device if it has been stopped.
CMD_START	Restart the currently active speech (if any) and resume queued I/O requests.
CMD_STOP	Stop any currently active speech and prevent queued I/O requests from starting.
CMD_WRITE	Write a stream of characters to the narrator device and generate mouth movement data for reads.

Exec Functions as Used in This Chapter

-----	-----
AbortIO()	Abort a command to the narrator device. If the command is in progress, it is stopped immediately. If it is queued, it is removed from the queue.
BeginIO()	Initiate a command and return immediately (asynchronous request). This is used to minimize the amount of system overhead.
CloseDevice()	Relinquish use of the narrator device. All requests must be complete.
CheckIO()	Return the status of an I/O request.
CloseLibrary()	Relinquish use of a previously opened library.
DoIO()	Initiate a command and wait for completion (synchronous request). Should be used with care because it will not return control if the request does not complete.
OpenDevice()	Obtain use of the narrator device.
OpenLibrary()	Obtain use of a library.
SendIO()	Initiate a command and return immediately (asynchronous request).

WaitIO() Wait for the completion of an asynchronous request. When the request is complete the message will be removed from reply port.

Exec Support Functions as Used in This Chapter

CreateExtIO() Create an extended I/O request structure of type narrator_rb. This structure will be used to communicate commands to the narrator device.

CreatePort() Create a signal message port for reply messages from the narrator device. Exec will signal a task when a message arrives at the port.

DeleteExtIO() Delete an extended I/O request structure created by CreateExtIO().

DeletePort() Delete the message port created by CreatePort().

1.3 8 Narrator Device / Device Interface

The narrator device operates like all other Amiga devices. To use the narrator device, you must first open it. This initializes certain global areas, opens the audio device, allocates audio channels, and performs other housekeeping functions. Once open, the device is ready to receive I/O commands (most typically CMD_WRITE and CMD_READ). Finally, when finished, the user should close the device. This will free some buffers and allow the entire device to be expunged should the system require memory. See the Introduction to Amiga System Devices chapter for general information on device usage.

The narrator device uses two extended I/O request structures: narrator_rb for write commands (to produce speech output) and mouth_rb for read commands (to receive mouth shape changes and word/syllable synchronization events). Both I/O request structures have been expanded (in a backwards compatible fashion) for the V37 narrator device with several new fields defined.

```
struct narrator_rb
{
    struct IOStdReq  message; /* Standard IORequest Block          */
    UWORD   rate;           /* Speaking rate (words/minute) */
    UWORD   pitch;          /* Baseline pitch in Hertz      */
    UWORD   mode;           /* Pitch mode                   */
    UWORD   sex;            /* Sex of voice                  */
    UBYTE   *ch_masks;      /* Pointer to audio allocation maps */
    UWORD   nm_masks;       /* Number of audio allocation maps */
    UWORD   volume;         /* Volume. 0 (off) thru 64      */
    UWORD   sampfreq;       /* Audio sampling frequency     */
    UBYTE   mouths;         /* If non-zero, generate mouths */
    UBYTE   chanmask; /* Which ch mask used (internal - do not modify) */
    UBYTE   numchan; /* Num ch masks used (internal- do not modify) */
    UBYTE   flags;      /* New feature flags            */
    UBYTE   F0enthusiasm; /* F0 excursion factor          */
}
```

```

    UBYTE    F0perturb;        /* Amount of F0 perturbation      */
    BYTE     F1adj;            /* F1 adjustment in +- 5% steps  */
    BYTE     F2adj;            /* F2 adjustment in +- 5% steps  */
    BYTE     F3adj;            /* F3 adjustment in +- 5% steps  */
    BYTE     A1adj;            /* A1 adjustment in decibels     */
    BYTE     A2adj;            /* A2 adjustment in decibels     */
    BYTE     A3adj;            /* A3 adjustment in decibels     */
    UBYTE     articulate;      /* Transition time multiplier    */
    UBYTE     centralize;      /* Degree of vowel centralization */
    char      *centphon;       /* Pointer to central ASCII phon */
    BYTE     AVbias;           /* Amplitude of voicing bias     */
    BYTE     AFbias;           /* Amplitude of frication bias   */
    BYTE     priority;         /* Priority while speaking       */
    BYTE     padl;             /* For alignment                 */
};

struct mouth_rb
{
    struct    narrator_rb voice; /* Speech IORequest Block        */
    UBYTE     width;            /* Mouth width (returned value)  */
    UBYTE     height;          /* Mouth height (returned value) */
    UBYTE     shape;           /* Internal use, do not modify   */
    UBYTE     sync;            /* Returned sync events          */
};

```

Details on the meaning of the various fields of the two I/O request blocks can be found in the Writing to the Narrator Device and Reading from the Narrator Device sections later in this chapter. See the include file `devices/narrator.h` for the complete structure definitions.

The Amiga Speech System
 Opening The Narrator Device
 Closing The Narrator Device

1.4 8 / Device Interface / The Amiga Speech System

The speech system on the Amiga is divided into two subsystems:

- * The translator library, consisting of a single function: `Translate()`, which converts an English string into its phonetic representation, and
- * The narrator device, which uses the phonetic representation (generated either manually or by the translator library) as input to generate human-like speech and play it out via the audio device.

The two subsystems can be used either together or individually. Generally, hand coding phonetic text will produce better quality speech than using the translator library, but this requires the programmer to "hard code" the phonetic text in the program or otherwise restrict the input to phonetic text only. If the program must handle arbitrary English input, the translator library should be used.

Below is an example of how you would use the translator library to translate a string for the narrator device.

```

#define BUFLen 500

APTR EnglStr;           /* pointer to sample input string */
LONG EnglLen;           /* input length */
UBYTE PhonBuffer[BUFLen]; /* place to put the translation */
LONG rtnCode;           /* return code from function */

struct narrator_rb *VoiceIO; /* speaking I/O request block */
struct mouth_rb *MouthIO;    /* mouth movement I/O request block */

EnglStr = "This is Amiga speaking."; /* a test string */
EnglLen = strlen(EnglStr);
rtnCode = Translate(EnglStr, EnglLen, (APTR)&PhonBuffer[0], BUFLen);

voice_io->message.io_Command = CMD_WRITE;
voice_io->message.io_Offset  = 0;
voice_io->message.io_Data    = PhonBuffer;
voice_io->message.io_Length  = strlen(PhonBuffer);
DoIO((struct IORequest *)VoiceIO)

```

This chapter discusses only the narrator device; refer to the "Translator Library" chapter of the Amiga ROM Kernel Reference Manual: Libraries for more information on the translator library.

While the narrator device on the Amiga supports all of the major device commands (see the Narrator Device Commands and Functions section), two of these commands do most of the work in the device. They are:

- * **CMD_WRITE** - This command is used to send a phonetic string to the device to be spoken. The narrator_rb I/O request block also contains several parameters which can be set to control various aspects of the speech, such as pitch, speaking rate, male/female voice, and so on. Some of the options are rather arcane. See the Writing to the Narrator Device section for a complete list of options and their descriptions.
- * **CMD_READ** - The narrator device can be told to generate various synchronization events which the user can query. These events are: mouth shape changes, word sync, and/or syllable sync. The events can be generated singly or in any combination, as requested by the user. Word and syllable synchronization events are new to system 2.0 and later (V37 and later of the narrator device). See the Reading from the Narrator Device section for more details.

1.5 8 / Device Interface / Opening The Narrator Device

Three primary steps are required to open the narrator device:

- * Create a message port using `CreatePort()`. Reply messages from the device must be directed to a message port.
- * Create an extended I/O request structure of type `narrator_rb`. The `narrator_rb` structure is created by the `CreateExtIO()` function.

- * Open the narrator device. Call `OpenDevice()` passing the I/O request.

```
struct MsgPort *VoiceMP;
struct narrator_rb *VoiceIO;

if (VoiceMP = CreatePort("speech_write",0))
    if (VoiceIO = (struct narrator_rb *)
        CreateExtIO(VoiceMP,sizeof(struct narrator_rb));
        if (OpenDevice("narrator.device", 0, VoiceIO, 0))
            printf("narrator.device did not open\n");
```

When the narrator device is first opened, it initializes certain fields in the user's narrator_rb I/O request structure. In order to maintain backwards compatibility with older versions of the narrator device, a mechanism was needed for the device to ascertain whether it was being opened with a V37 or pre-V37 style I/O request structure. The pad field in the pre-V37 narrator_rb I/O request structure (which no one should have ever touched!) has been replaced by the flags field in the V37 narrator_rb structure, and is our path to upward compatibility. The device checks to see if a bit is set in this flags field. This bit must be set before opening the device if V37 or later features of the narrator device are to be used. There are two defined constants in the include file, `NDB_NEWIORB` and `NDF_NEWIORB`. `NDB_NEWIORB` specifies the bit which must be set in the flags field, `NDF_NEWIORB` is the field definition of the bit (`1 << NDB_NEWIORB`).

Once the device is opened, the mouth_rb (read) I/O request structure can be set up. Each `CMD_READ` request must be matched with an associated `CMD_WRITE` request. This is necessary for the device to match the various sync events with a particular utterance. The read I/O request structure is easily set up as follows:

- * Create a read message port using the `CreatePort()` function.
- * Allocate memory for the mouth_rb extended I/O request structure using `AllocMem()`.
- * Copy the narrator_rb I/O request structure used to open the device into the voice field of the mouth_rb I/O request structure. This will set the fields necessary for the device to make the correct correspondence between read and write requests.
- * Copy the pointer to the read message port returned from `CreatePort()` into the `voice.message.io_Message.mn_ReplyPort` field of the mouth_rb structure.

The following code fragment, in conjunction with the `OpenDevice()` code fragment above, shows how to set up the mouth_rb structure:

```
struct MsgPort *MouthMP;
struct mouth_rb *MouthIO;

if (MouthMP = CreatePort("narrator_read", 0))
    if (!(MouthIO = (struct mouth_rb *)
        AllocMem(sizeof(struct mouth_rb),MEMF_PUBLIC|MEMF_CLEAR)))
    {
```

```

        /* Copy I/O request used in OpenDevice */
        MouthIO->voice = *VoiceIO;
        /* Set port */
        MouthIO->voice.message.io_Message.mn_ReplyPort=MouthMP;
    }
    else
        printf("AllocMem failed\n");
    else
        printf("CreatePort failed\n");

```

1.6 8 / Device Interface / Closing The Narrator Device

Each `OpenDevice()` must be eventually matched by a call to `CloseDevice()`. This is necessary to allow the system to expunge the device in low memory conditions. As long as any task has the device open, or has forgotten to close it before terminating, the narrator device will not be expunged.

All I/O requests must have completed before the task can close the device. If any requests are still pending, the user must abort them before closing the device.

```

    if (!(CheckIO(VoiceIO))
    {
        AbortIO(VoiceIO); /* Abort queued or in progress request */
    }
    WaitIO((struct IORequest *)VoiceIO); /* Wait for abort to do its job */
    CloseDevice(VoiceIO);                /* Close the device */

```

1.7 8 Narrator Device / Writing to the Narrator Device

You write to the narrator device by passing a `narrator_rb` I/O request to the device with `CMD_WRITE` set in `io_Command`, the number of bytes to be written set in `io_Length` and the address of the write buffer set in `io_Data`.

```

VoiceIO->message.io_Command = CMD_WRITE;
VoiceIO->message.io_Offset = 0;
VoiceIO->message.io_Data = PhonBuffer;
VoiceIO->message.io_Length = strlen(PhonBuffer);
DoIO((struct IORequest *)VoiceIO);

```

You can control several characteristics of the speech, as indicated in the `narrator_rb` struct shown in the Device Interface section.

Generally, the narrator device attempts to speak in a non-regional dialect of American English. With pre-V37 versions of the device, the user could change only a few of the more basic aspects of the speaking voice such as pitch, male/female, speaking rate, etc. With the V37 and later versions of the narrator device, the user can now change many more aspects of the speaking voice. In addition, in the pre-V37 device, only mouth shape changes could be queried by the user. With the V37 device, the user can also receive start of word and start of syllable synchronization events.

These events can be generated independently, giving the user much greater flexibility in synchronizing voice to animation or other effects.

The following describes the fields of the `narrator_rb` structure:

`message.io_Data`

Points to a NULL-terminated ASCII phonetic input string. For backwards compatibility issues, the string may also be terminated with a "#" symbol. See the How to Write Phonetically for Narrator section of this chapter for details.

`message.io_Length`

Length of the input string. The narrator device will parse the input string until either a NULL or a "#" is encountered, or until `io_Length` characters have been processed.

`rate`

The speaking rate in words/minute. Range is from 40 to 400 wpm.

`pitch`

The baseline pitch of the speaking voice. Range is 65 to 320 Hertz.

`mode`

The F0 (pitch) mode. `ROBOTICF0` produces a monotone pitch, `NATURALF0` produces a normal pitch contour, and `MANUALF0` (new for V37 and later) gives the user more explicit control over the pitch contour by creative use of accent numbers. In `MANUALF0` mode, a given accent number will have the same effect on the pitch regardless of its position in the sentence and its relation to other accented syllables. In `NATURALF0` mode, accent numbers have a reduced effect towards the end of sentences (especially long ones). In addition, the proximity of other accented syllables, the number of syllables in the word, and the number of phrases and words in the sentence all affect the pitch contour. In `MANUALF0` mode these things are ignored and it's up to the user to do the controlling. This has the advantage of being able to have the pitch be more expressive. The `F0enthusiasm` field will scale the effect.

`sex`

Controls the sex of the speaking voice (MALE or FEMALE). In actuality, only the formant targets are changed. The user must still change the pitch and speaking rate of the voice to get the correct sounding sex. See the include files for default pitch and rate settings.

`ch_masks`

Pointer to a set of audio allocation maps. See the "Audio Device" chapter for details.

`nm_masks`

Number of audio allocation maps. See the "Audio Device" chapter for details.

`volume`

Sets the volume of the speaking voice. Range 0 - 64.

`sampfreq`

The synthesizer is ``tuned" to a sampling frequency of 22,200 Hz. Changing sampfreq affects pitch and formant tunings and can be used to create unusual vocal effects. For V37 and later, it is recommended that F1, F2, and F3adj be used instead to achieve this effect.

mouths

If set to a non-zero value will direct the narrator device to generate mouth shape changes and send this data to the user in response to read requests. See the Reading from the Narrator Device section for more details.

chanmask

Used internally by the narrator device. The user should not modify this field.

numchan

Used internally by the narrator device. The user should not modify this field.

flags (V37)

Used to specify V37 features of the device. Possible bit settings are: NDB_NEWIORB - I/O request block uses V37 features. NDB_WORDSYNC - Device should generate start of word sync events. NDB_SYLSYNC - Device should generate start of syllable sync events. These bit definitions and their corresponding field definitions (NDF_NEWIORB, NDF_WORDSYNC, and NDF_SYLSYNC) can be found in the include files.

F0enthusiasm (V37)

The value of this field controls the scaling of pitch (F0) excursions used on accented syllables and has the effect of making the narrator device sound more or less "enthusiastic" about what it is saying. It is calibrated in 1/32s with unity (32) being the default value. Higher values cause more F0 variation, lesser values cause less. This feature is most useful in manual F0 mode.

F0perturb (V37)

Non-zero values in this field cause varying amounts of random low-frequency modulation of the pitch (F0). In other words, the pitch shakes in much the same way as an elderly person's voice does. Range is 0 to 255.

F1adj, F2adj, F3adj (V37)

Changes the tuning of the formant frequencies. A formant is a major vocal tract resonance, and the frequencies of these formants move continuously as we speak. Traditionally, they have been given the abbreviations of F1, F2, F3... with F1 being the one lowest in frequency. Moving these formants away from their normal positions causes drastic changes in the sound of the voice and is a very powerful tool in the creation of character voices. This adjustment is in $\pm 5\%$ steps. Positive values raise the formant frequencies and vice versa. The default is zero. Use these adjustments instead of changing sampfreq.

A1adj, A2adj, A3adj (V37)

In a parallel formant synthesizer, the amplitudes of the formants

need to be specified along with their frequencies. These fields bias the amplitudes computed by the narrator device. This is useful for creating different tonal balances (bass or treble), and listening to formants in isolation for educational purposes. The adjustments are calibrated directly in ± 1 db (decibel) steps. Using negative values

will cause no problems; use of positive numbers can cause clipping. If you want to raise an amplitude, try cutting the others the same relative amount, then bring them all up equally until clipping is heard, then back them off. This should produce an optimum setting. This field has a +31 to -32 db range and the value -32db is equivalent to -infinity, shutting that formant off completely.

articulate (V37)

According to the popular theories of speech production, we move our articulators (jaw, tongue, lips, etc.) smoothly from one "target" position to the next. These articulatory targets correspond to acoustic targets specified by the narrator device for each phoneme. The device calculates the time it should take to get from one target to the next and this field allows you to intervene in that process. Values larger than the default will cause the transitions to be proportionately longer and vice versa. This field is calibrated in percent with 100 being the default. For example, a value of 50 will cause the transitions to take half the normal time, with the result being "sharper", more deliberate sounding speech (not necessarily more natural). A value of 200 will cause the transitions to be twice as long, slurring the speech. Zero is a special value in the narrator device will take special measures to create no transitions at all and each phoneme will simply be abutted to the next.

centralize (V37)

This field together with centphon can be used to create regional accent effects by modifying vowel sounds. centralize specifies the degree (in percent) to which vowel targets are "pulled" towards the targets of the vowel specified by centphon. The default value of 0% indicates that each vowel in the utterance retains its own target values. The maximum value of 100% indicates that each vowel's targets are replaced by the targets of the specified vowel. Intermediate values control the degree of interpolation between the utterance vowel's targets and the targets of the vowel specified by centphon.

centphon (V37)

Pointer to an ASCII string specifying the vowel whose targets are used in the interpolation specified by centralize. The vowels which can be specified are: IY, IH, EH, AE, AA, AH, AO, OW, UH, ER, UW. Specifying other than these will result in an error code being returned.

AVbias, AFbias (V37)

Controls the relative amplitudes of the voiced and unvoiced speech sounds. Voiced sounds are those made with the vocal cords vibrating, such as vowels and some consonants like y, r, w, and m. Unvoiced sounds are made without the vocal cords vibrating and use the sound of turbulent air, such as s, t, sh, and f. Some sounds are combinations of both such as z and v. AVbias and AFbias change the default amplitude of the voiced and unvoiced components of the sounds

respectively. (AV stands for Amplitude of Voicing and AF stands for Amplitude of Frication). These fields are calibrated in ± 1 db steps ←

and have the same range as the other amplitude biases, namely +31 to -32 db. Again, positive values may cause clipping. Negative values are the most useful.

priority (V37)

Task priority while speaking. When the narrator device begins to synthesize a sentence, the task priority remains unchanged while it is calculating acoustic parameters. However, when speech begins at the end of this process, the priority is bumped to 100 (the default value). If you wish, you may change this to anything you want. Higher values will tend to lock out most anything while speech is going on, and lower values may cause audible breaks in the speech output. The following example shows how to issue a write request to the narrator device. The first write is done with the default parameter settings. The second write is done after modifying the first and third formant loudness and using the centralization feature.

The following example shows how to issue a write request to the narrator device. The first write is done with the default parameter settings. The second write is done after modifying the first and third formant loudness and using the centralization feature.

```
Speak_Narrator.c
```

1.8 8 Narrator Device / Reading from the Narrator Device

All read requests to the narrator device must be matched to an associated write request. This is done by copying the narrator_rb structure used in the OpenDevice() call into the voice field of the mouth_rb I/O request structure. You must do this after the call to OpenDevice(). Matching the read and write requests allows the narrator device to coordinate I/O requests across multiple uses of the device.

In pre-V37 versions of the narrator device, only mouth shape changes can be queried from the device. This is done by setting the mouths field of the narrator_rb I/O request structure (the write request) to a non-zero value. The write request is then sent asynchronously to the device and while it is in progress, synchronous read requests are sent to the device using the mouth_rb I/O request structure. When the mouth shape has changed, the device will return the read request to the user with bit 0 set in the sync field of the mouth_rb. The fields width and height of the mouth_rb structure will contain byte values which are proportional to the actual width and height of the mouth for the phoneme currently being spoken. Read requests sent to the narrator device are not returned to the user until one of two things happen: either the mouth shape has changed (this prevents the user from having to constantly redraw the same mouth shape), or the speech has completed. The user can check io_Error to determine if the mouth shape has changed (a return code of 0) or if the speech has completed (return code of ND_NoWrite).

In addition to returning mouth shapes, reads to the V37 narrator device can also perform two new functions: word and syllable sync. To generate

word and/or syllable sync events, the user must specify several bits in the flags field of the write request (narrator_rb structure). The bits are NDB_WORDSYNC and NDB_SYLSYNC, for start of word and start of syllable synchronization events, respectively, and, of course, NDB_NEWIORB, to indicate that the V37 I/O request is required.

NDB_WORDSYNC and NDB_SYLSYNC tell the device to expect read requests and to generate the appropriate event(s). As with mouth shape change events, the write request is sent asynchronously to the device and, while it is in progress, synchronous read requests are sent to the device. The sync field of the mouth_rb structure will contain flags indicating which events (mouth shape changes, word sync, and/or syllable sync) have occurred.

The returned sync field flags are:

```

bit 0 (0x01) -> mouth shape change event
bit 1 (0x02) -> start-of-word synchronization event
bit 2 (0x04) -> start-of-syllable synchronization event

```

and 1 or more flags may be set for any particular read.

As with mouth shape changes, read requests will not return until the requested event(s) have occurred, and the user must test the io_Error field of the mouth_rb structure to tell when the speech has completed (an error return of ND_NoWrite).

Several read events can be compressed into a single event. This can occur in two ways: first when two dissimilar events occur between two successive read requests. For example, a single read may return both a mouth change and a syllable sync event. This should not present a problem if the user checks for all events. The second is when multiple events of the same type occur between successive read requests. This is of no great concern in dealing with mouth shape changes because, presumably, mouth events are used to drive animation, and the animation procedure will simply draw the current mouth shape.

Watch Those Sync Events.

When word or syllable sync is desired, the narrator device may compress multiple sync events into a single sync event. Missing a word or syllable sync may cause word highlighting (for example) to lose sync with the speech output. A future version of the device will include an extension to the mouth_rb I/O request structure which will contain word and syllable counts and, possibly, other synchronization methods.

The following code fragment shows the basics of how to perform reads from the narrator device. For a more complete example, see the sample program at the end of this chapter. For this fragment, take the code of the previous write example as a starting point. Then the following code would need to be added:

```

struct  mouth_rb  *MouthIO;      /* Pointer to read IORquest block */
struct  MsgPort   *MouthMP;     /* Pointer to read message port */

/*

```

```

* (1) Create a message port for the read request.
*/
    if (!(MouthMP = CreatePort("narrator_read", 0L)))
        BellyUp("Read CreatePort failed");

/*
* (2) Create an extended IORequest of type mouth_rb.
*/
    if (!(MouthIO = (struct mouth_rb *)
        CreateExtIO(MouthMP, sizeof(struct mouth_rb))))
        BellyUp("Read CreateExtIO failed");

/*
* (3) Set up the read IORequest. Do this after the call to OpenDevice().
* We assume that the write IORequest and the OpenDevice have been done
*/
    MouthIO->voice = *SpeakIO;
    MouthIO->voice.message.io_Message.mn_ReplyPort = ReadMsgPort;
    MouthIO->voice.message.io_Command = CMD_READ;

/*
* (4) Set the flags field of the narrator_rb write request to return the
*     desired sync events. If mouth shape changes are required, then the
*     mouths field of the IORequest should be set to a non-zero value.
*/

    SpeakIO->mouths = 1;           /* Generate mouth shape changes */
    SpeakIO->flags = NDF_NEWIORB | /* Indicates V37 style IORequest */
                  NDF_WORDSYNC | /* Request start-of-word sync events */
                  NDF_SYLSYNC;   /* Request start-of-syll sync events */

/*
* (5) Issue asynchronous write request. The driver initiates the write
*     request and returns immediately.
*/

    SendIO(SpeakIO);

/*
* (6) Issue synchronous read requests. For each request we check the sync
*     field to see which events have occurred. Since any combination of
*     events can be returned in a single read, we must check all
*     possibilities. We continue looping until the read request returns
*     an error of ND_NoWrite, which indicates that the write request has
*     completed.
*/

    for (DoIO(MouthIO); MouthIO->voice.message.io_Error
        != ND_NoWrite; DoIO(MouthIO))
    {
        if (MouthIO->sync & 0x01) DoMouthShape();
        if (MouthIO->sync & 0x02) DoWordSync();
        if (MouthIO->sync & 0x04) DoSyllableSync();
    }

/*
* (7) Finally, we must perform a WaitIO() on the original write request.

```

*/

```
WaitIO(SpeakIO);
```

1.9 8 Narrator Device / How to Write Phonetically for Narrator

This section describes in detail the procedure used to specify phonetic strings to the narrator speech synthesizer. No previous experience with phonetics is required. The only thing you may need is a good pronunciation dictionary for those times when you doubt your own ears. You do not have to learn a foreign language or computer language. You are just going to learn how to write down the English that comes out of your own mouth. In writing phonetically you do not have to know how a word is spelled, just how it is said.

TABLE OF PHONEMES

Vowels

Phoneme	Example	Phoneme	Example
-----	-----	-----	-----
IY	beet, eat	IH	bit, in
EH	bet, end	AE	bat, ad
AA	bottle, on	AH	but, up
AO	ball, awl	UH	book, soot
ER	bird, early	OH	border
AX*	about, calibrate	IX*	solid, infinite

* AX and IX should never be used in stressed syllables.

Diphthongs

Phoneme	Example	Phoneme	Example
-----	-----	-----	-----
EY	bay, aid	AY	bide, I
OY	boy, oil	AW	bound, owl
OW	boat, own	UW	brew, boolean

Consonants

Phoneme	Example	Phoneme	Example
-----	-----	-----	-----
R	red	L	long
W	wag	Y	yellow, comp(Y)uter
M	men	N	no
NX	sing	SH	shy
S	soon	TH	thin
F	fed	ZH	pleasure
Z	has, zoo	DH	then
V	very	WH	when
CH	check	J	judge
/H	hole	/C	loch
B	but	P	put

D	dog	T	toy
K	keg, copy	G	guest

Special Symbol

Phoneme	Example	Explanation
-----	-----	-----
DX	pity	tongue flap
Q	kitt(Q)en	glottal stop
QX		silent vowel

Contractions (see text)

UL = AXL
 IL = IXL
 UM = AXM
 IM = IXM
 UN = AXN
 IN = IXN

Digits and Punctuation

Digits 1-9	Syllabic stress, ranging from secondary through emphatic
.	Period - sentence final character.
?	Question mark - sentence final character
-	Dash - phrase delimiter
,	Comma - clause delimiter
()	Parentheses - noun phrase delimiters (see text)

The narrator device works on utterances at the sentence level. Even if you want to say only one word, it will treat it as a complete sentence. Therefore, narrator wants one of two punctuation marks to appear at the end of every sentence - a period or a question mark. The period is used for almost all utterances and will cause a final fall in pitch to occur at the end of a sentence. The question mark is used at the end of yes/no questions only, and results in a final rise in pitch.

For example, the question, Do you enjoy using your Amiga? would take a question mark at the end, while the question, What is your favorite color? should be followed (in the phonetic transcription) with a period. If no punctuation appears at the end of a string, narrator will append a dash to it, which will result in a short pause. Narrator recognizes other punctuation marks as well, but these are left for later discussion.

Phonetic Spelling	Hints For Intelligibility
Stress And Intonation	Example Of English And Phonetic Texts
Punctuation	Concluding Remarks

1.10 8 / How to Write Phonetically for Narrator / Phonetic Spelling

Utterances are usually written phonetically using an alphabet of symbols known as IPA (International Phonetic Alphabet). This alphabet is found at

the front of most good dictionaries. The symbols can be hard to learn and were not readily available on computer keyboards, so the Advanced Research Projects Agency (ARPA) came up with the ARPABET, a way of representing each symbol using one or two upper case letters. Narrator uses an expanded version of the ARPABET to specify phonetic sounds.

A phonetic sound, or phoneme, is a basic speech sound, a speech atom. Working backwards: sentences can be broken into words, words into syllables, and syllables into phonemes. The word cat has three letters and (coincidentally) three phonemes. Looking at the table of phonemes we find the three sounds that make up the word cat. They are the phonemes K, AE, and T, written as KAET. The word cent translates as SEHNT. Notice that both words begin with the letter c, but because they are pronounced differently they have different phonetic spellings. These examples introduce a very important concept of phonetic spelling: spell it like it sounds, not like it looks.

Choosing the Right Vowel

Choosing the Right Consonant

Contractions and Special Symbols

1.11 8 / Phonetic Spelling / Choosing the Right Vowel

Phonemes, like letters, are divided into two categories: vowels and consonants. Loosely defined, a vowel is a continuous sound made with the vocal cords vibrating and air exiting the mouth (as opposed to the nose). A consonant is any other sound, such as those made by rushing air (like S or TH), or by interruptions in the air flow by the lips or tongue (B or T). All vowels use a two letter ASCII phonetic code while consonants use a one or two letter code.

In English we write with only five vowels: a, e, i, o, and u. It would be easy if we only said five vowels. However, we say more than 15 vowels. Narrator provides for most of them. Choose the proper vowel by listening: Say the word aloud, perhaps extending the vowel sound you want to hear and then compare the sound you are making to the sounds made by the vowels in the examples on the phoneme list. For example, the a in apple sounds the same as the a in cat, not like the a in Amiga, talk, or made. Notice also that some of the example words in the list do not even use any of the same letters contained in the phoneme code; for example AA as in bottle.

Vowels are divided into two groups: those that maintain the same sound throughout their durations and those that change their sound. The ones that change are called diphthongs. Some of us were taught the terms long and short to describe vowel sounds. Diphthongs fall into the long category, but these two terms are inadequate to fully differentiate between vowels and should be avoided. The diphthongs are the last six vowels listed in the table. Say the word made out loud very slowly. Notice how the a starts out like the e in bet but ends up like the e in beet. The a, therefore, is a diphthong in this word and we would use EY to represent it. Some speech synthesis systems require you to specify the changing sounds in diphthongs as separate elements, but narrator takes care of the assembly of diphthongal sounds for you.

1.12 8 / Phonetic Spelling / Choosing the Right Consonant

Consonants are divided into many categories by phoneticians, but we need not concern ourselves with most of them. Picking the correct consonant is very easy if you pay attention to just two categories: voiced and unvoiced. A voiced consonant is made with the vocal cords vibrating, and an unvoiced one is made when the vocal cords are silent. Sometimes English uses the same letter combinations to represent both. Compare the th in thin with the th in then. Notice that the first is made with air rushing between the tongue and upper teeth. In the second, the vocal cords are vibrating also. The voiced th phoneme is DH and the unvoiced one is TH. Therefore, thin is phonetically spelled as THIHN while the word then is spelled DHEHN.

A sound that is particularly subject to mistakes is voiced and unvoiced S, phonemes Z and S, respectively. Clearly the word bats ends with an S and the word has ends with a Z. But, how do you spell close? If you say "What time do you close?", you spell it with a Z, and if you are saying "I love to be close to you." you use an S.

Another sound that causes some confusion is the r sound. There are two different r-like phonemes in the Narrator alphabet: R under the consonants and ER under the vowels. Use ER if the r sound is the vowel sound in the syllable like in bird, absurd, and flirt. Use the R if the r sound precedes or follows another vowel sound in that syllable as in car, write, and craft.

1.13 8 / Phonetic Spelling / Contractions and Special Symbols

There are several phoneme combinations that appear very often in English words. Some of these are caused by our laziness in pronunciation. Take the word connector for example. The o in the first syllable is almost swallowed out of existence. You would not use the AA phoneme; you would use the AX phoneme instead. It is because of this relaxation of vowels that we find ourselves using AX and IX very often. Since this relaxation frequently occurs before l, m, and n, narrator has a shortcut for typing these combinations. Instead of personal being spelled PERSIXNAXL, we can spell it PERSINUL, making it a little more readable. Anomaly goes from AXNAAMAXLIY to UNAAMULIY, and KAAMBIXNEYSHIXN becomes KAAMBINEYSHIN for combination. It may be hard to decide whether to use the AX or IX brand of relaxed vowel. The only way to find out is to use both and see which sounds best.

Other special symbols are used internally by narrator. Sometimes they are inserted into or substituted for part of your input sentence. You can type them in directly if you wish. The most useful is probably the Q or glottal stop, an interruption of air flow in the glottis. The word Atlantic has one between the t and the l. Narrator knows there should be a glottal stop there and saves you the trouble of typing it. But narrator is only close to perfect, so sometimes a word or word pair might slip by that would have sounded better with a Q stuck in someplace.

1.14 8 / How to Write Phonetically for Narrator / Stress And Intonation

It is not enough to tell narrator what you want said. For the best results you must also tell narrator how you want it said. In this way you can alter a sentence's meaning, stress important words, and specify the proper accents in polysyllabic words. These things improve the naturalness and thus the intelligibility of the spoken output.

Stress and intonation are specified by the single digits 1-9 following a vowel phoneme code. Stress and intonation are two different things, but are specified by a single number.

Stress is, among other things, the elongation of a syllable. A syllable is either stressed or not, so the presence of a number after the vowel in a syllable indicates stress on that syllable. The value of the number indicates the intonation. These numbers are referred to here as stress marks but keep in mind that they also affect intonation.

Intonation here means the pitch pattern or contour of an utterance. The higher the stress mark, the higher the potential for an accent in pitch. A sentence's basic contour is comprised of a quickly rising pitch gesture up to the first stressed syllable in the sentence, followed by a slowly declining tone throughout the sentence, and finally, a quick fall to a low pitch on the last syllable. The presence of additional stressed syllables causes the pitch to break its slow, declining pattern with rises and falls around each stressed syllable. Narrator uses a very sophisticated procedure to generate natural pitch contours based on how you mark the stressed syllables.

How and Where to Put the Stress Marks
Which Stress Value Do I Use?

1.15 8 // Stress And Intonation / How and Where to Put the Stress Marks

The stress marks go immediately to the right of vowel phoneme codes. The word cat has its stress marked after the AE, e.g., KAE5T. You generally have no choice about the location of a number; there is definitely a right and wrong location. A number should either go after a vowel or it should not. Narrator will not flag an error if you forget to put a stress mark in or if you place it on the wrong vowel. It will only tell you if a stress mark has been put after a non-vowel, i.e., consonant or punctuation.

The rules for placing stress marks are as follows:

- * Always place a stress mark in a content word. A content word is one that contains some meaning. Nouns, verbs, and adjectives are all content words, they tell the listener what you are talking about. Words like but, if, and the are not content words. They do not convey any real world meaning, but are required to make the sentence function, so they are given the name function words.
- * Always place a stress mark on the accented syllable(s) of polysyllabic words, whether they are content or function words. A

polysyllabic word is any word of more than one syllable. Commodore has its stress (often called accent) on the first syllable and would be spelled KAA5MAXDOHR, while computer is stressed on the second syllable: KUMPYUW5TER.

If you are in doubt about which syllable gets the stress, look up the word in a dictionary and you will find an accent mark over the stressed syllable. If more than one syllable in a word receives stress, they usually are not of equal value. These are referred to as primary and secondary stresses. The word understand has its first and last syllables stressed, with the syllable stand getting the primary stress and the syllable un getting the secondary stress. This produces the phonetic representation AH1NDERSTAE4ND. Syllables with secondary stress should be marked with a value of only 1 or 2.

Compound words (words with more than one root) such as baseball, software, and lunchwagon can be written as one word, but should be thought of as separate words when marking stress. Thus, lunchwagon would be spelled LAH5NCHWAE2GIN. Notice that the lunch got a higher stress mark than the wagon. This is common in compound words, the first word usually receives the primary stress.

1.16 8 // Stress And Intonation / Which Stress Value Do I Use?

If you get the spelling and stress mark positions correct, you are 95 percent of the way to a good sounding sentence. The next thing to do is decide on the stress mark values. They can be roughly related to parts of speech, and you can use the table shown below as a guide to assigning values.

RECOMMENDED STRESS VALUES

Part of Speech -----	Stress Value -----
Exclamations	9
Adverbs	7
Quantifiers	7
Nouns	5
Adjectives	5
Verbs	4
Pronouns	3
Secondary stress	1 or 2
Everything else	None

The above values merely suggest a range. If you want attention directed to a certain word, raise its value. If you want to downplay a word, lower it. Sometimes even a function word can be the focus of a sentence. It is quite conceivable that the word to in the sentence Please deliver this to Mr. Smith. could receive a stress mark of 9. This would add focus to the word, indicating that the item should be delivered to Mr. Smith in person.

1.17 8 / How to Write Phonetically for Narrator / Punctuation

In addition to the period or question mark that is required at the end of a sentence, Narrator also recognizes dashes, commas, and parentheses.

The comma goes where you would normally put a comma in an English sentence. It causes narrator to pause with a slightly rising pitch, indicating that there is more to come. The use of additional commas, that is, more than would be required for written English – is often helpful. They serve to set clauses off from one another. There is a tendency for a listener to lose track of the meaning of a sentence if the words run together. Read your sentence aloud while pretending to be a newscaster. The locations for additional commas should leap out at you.

The dash serves almost the same purpose as the comma, except that the dash does not cause the pitch to rise so severely. A rule of thumb is: Use dashes to divide phrases and commas to divide clauses.

Parentheses provide additional information to narrator's intonation function. They should be put around noun phrases of two or more content words. This means that the noun phrase, a giant yacht should be surrounded with parentheses because it contains two content words, giant and yacht. The phrase my friend should not have parentheses around it because it contains only one content word. Noun phrases can get fairly large, like the best time I've ever had or a big basket of fruit and nuts. The parentheses are most effective around these large phrases; the smaller ones can sometimes go without. The effect of parentheses is subtle, and in some sentences you might not notice their presence. In sentences of great length, however, they help provide for a very natural contour.

1.18 8 / How to Write Phonetically / Hints For Intelligibility

There are a few tricks you can use to improve the intelligibility of a sentence. Often, a polysyllabic word is more recognizable than a monosyllabic word. For instance, instead of saying huge, say enormous. The longer version contains information in every syllable, thus giving the listener a greater chance to hear it correctly.

Another good practice is to keep sentences to an optimal length. Writing for reading and writing for speaking are two different things. Try not to write a sentence that cannot be easily spoken in one breath. Such a sentence tends to give the impression that the speaker has an infinite lung capacity and sounds unnatural. Try to keep sentences confined to one main idea; run-on sentences tend to lose their meaning.

New terms should be highly stressed the first time they are heard. This gives the listener something to cue on, and can aid in comprehension.

The insertion of the glottal stop phoneme Q at the end of a word can sometimes help prevent slurring of one word into another. When we speak, we do not pause at the end of each word, but instead transition smoothly between words. This can sometimes reduce intelligibility by eliminating word boundary cues. Placing a Q, (not the silent vowel QX) at the end of a word results in some phonological effects taking place which can restore

the word boundary cues.

1.19 8 / Writing Phonetically / English And Phonetic Text Example

Cardiomyopathy. I had never heard of it before, but there it was listed as the form of heart disease that felled not one or two but all three of the artificial heart recipients. A little research produced some interesting results. According to an article in the Nov. 8, 1984, New England Journal of Medicine, cigarette smoking causes this lethal disease that weakens the heart's pumping power. While the exact mechanism is not clear, Dr. Arthur J. Hartz speculated that nicotine or carbon monoxide in the smoke somehow poisons the heart and leads to heart failure.

KAA1RDIYOWMAYAA5PAXTHIY. AY /HAED NEH1VER /HER4D AXV IHT BIXFOH5R, BAHT DHEH5R IHT WAHZ - LIH4STIXD AEZ (DHAX FOH5RM AXV /HAA5RT DIHZIY5Z) DHAET FEH4LD (NAAT WAH5N OHR TUV5) - BAHT (AO7L THRIY5 AXV DHAX AA5RTAXFIHSHUL /HAA5RTQ RIXSIH5PIYINTS). (AH LIH5TUL RIXSER5CH) PROHDUW5ST (SAHM IH5NTRIHSIHNX RIXZAH5LTS). AHKOH5RDIHNX TUV (AEN AA5RTIHKUL IHN DHAX NOWVEH5MBER EY2TH NAY5NTIYNEYTIYFOH1R NUW IY5NXGLIND JER5NUL AXV MEH5DIXSIN), (SIH5GEREHT SMOW5KIHNX) KAO4ZIHZ (DHIHS LIY5THUL DIHZIY5Z) DHAET WIY4KINZ (DHAX /HAA5RTS PAH4MPIHNX PAW2ER). WAYL (DHIY IHGZAE5KT MEH5KINIXZUM) IHZ NAAT KLIY5R, DAA5KTER AA5RTHIR JEY2 /HAARTS SPEH5KYULEYTIHD DHAET NIH5KAXTIYN, OHR KAA5RBIN MUNAA5KSAYD IHN DHAX SMOW5K - SAH5M/HAW1 POY4ZINZ DHAX /HAA5RT, AEND LIY4DZ TUV (/HAA5RT FEY5LYER).

1.20 8 / How to Write Phonetically for Narrator / Concluding Remarks

This guide should get you off to a good start in phonetic writing for Narrator. The only way to get really proficient is to practice. Many people become good at it in as little as one day. Others make continual mistakes because they find it hard to let go of the rules of English spelling, so trust your ears.

1.21 8 Narrator Device / A More Technical Explanation

The narrator speech synthesizer is a computer model of the human speech production process. It attempts to produce accurately spoken utterances of any English sentence, given only a phonetic representation as input. Another program in the Amiga speech system, the translator device, derives the required phonetic spelling from English text. Timing and pitch contours are produced automatically by the synthesizer software.

In humans, the physical act of producing speech sounds begins in the lungs. To create a voiced sound, the lungs force air through the vocal folds (commonly called the vocal cords), which are held under tension and

which periodically interrupt the flow of air, thus creating a buzz-like sound. This buzz, which has a spectrum rich in harmonics, then passes through the vocal tract and out the lips and nose, which alters its spectrum drastically. This is because the vocal tract acts as a frequency filter, selectively reinforcing some harmonics and suppressing others. It is this filtering that gives a speech sound its identity. The amplitude versus frequency graph of the filtering action is called the vocal tract transfer function. Changing the shape of the throat, tongue, and mouth retunes the filter system to accentuate different frequencies.

The sound travels as a pressure wave through the air, and it causes the listener's eardrum to vibrate. The ear and brain of the listener decode the incoming frequency pattern. From this the listener can subconsciously make a judgement about what physical actions were performed by the speaker to make the sound. Thus the speech chain is completed, the speaker having encoded his physical actions on a buzz via selective filtering and the listener having turned the sound into guesses about physical actions by frequency decoding.

Now that we know how humans produce speech, how does the Amiga do it? It turns out that the vocal tract transfer function is not random, but tends to accentuate energy in narrow bands called formants. The formant positions move fairly smoothly as we speak, and it is the formant frequencies to which our ears are sensitive. So, luckily, we do not have to model throat, tongue, teeth and lips with our computer, we can imitate formant actions instead.

A good representation of speech requires up to five formants, but only the lowest three are required for intelligibility. The pre-V37 Narrator had only three formants, while the V37 Narrator has five formants for a more natural sounding voice. We begin with an oscillator that produces a waveform similar to that which is produced by the vocal folds, and we pass it through a series of resonators, each tuned to a different formant frequency. By controlling the volume and pitch of the oscillator and the frequencies of the resonators, we can produce highly intelligible and natural-sounding speech. Of course the better the model the better the speech; but more importantly, experience has shown that the better the control of the model's parameters, the better the speech.

Oscillators, volume controls, and resonators can all be simulated mathematically in software, and it is by this method that the narrator system operates. The input phonetic string is converted into a series of target values for the various parameters. A system of rules then operates on the string to determine things such as the duration of each phoneme and the pitch contour. Transitions between target values are created and smoothed to produce natural, continuous changes from one sound to the next.

New values are computed for each parameter for every 8 milliseconds of speech, which produces about 120 acoustic changes per second. These values drive a mathematical model of the speech synthesizer. The accuracy of this simulation is quite good. Human speech has more formants than the narrator model, but they are high in frequency and low in energy content.

The human speech production mechanism is a complex and wonderful thing. The more we learn about it, the better we can make our computer simulations. Meanwhile, we can use synthetic speech as yet another

computer output device to enhance the man/machine dialogue.

1.22 8 Narrator Device / Additional Information on the Narrator Device

Additional programming information on the narrator device can be found in the include files and the Autodocs for the narrator device and the Autodocs for the translator library. All are contained in the Amiga ROM Kernel Reference Manual: Includes and Autodocs.

Narrator Device Information

INCLUDES	devices/narrator.h devices/narrator.i
AUTODOCS	narrator.doc translator.doc