

# Dreamcast

## 組み込みマニュアル

### 簡易音声認識ライブラリ

2000年05月26日

Ver . 1.00

2000年05月31日

Ver . 1.10

2000年08月23日

Ver . 1.20

2000年09月11日

Ver . 1.21

## 変 更 履 歴

年月日	バージョン	変 更 内 容
2000.05.26	1.00	・ 新規作成。
2000.05.31	1.10	・ NsrGetResultSentence, NsrSetVoiceDetectMode 関数の追加。 ・ 記述漏れの追記
2000.08.23	1.20	・ ライブラリ使用方法にユーザワードトレーニングサンプルを追加。 ・ NSR ハンドルの動作状態にユーザワードトレーニングを追加。 ・ NsrCreateClass, NsrDestroyClass, NsrEnterTrainMode, NsrStartTrain, NsrTerminateTrainInput, NsrStopTrain, NsrGetUserWord, NsrGetTrainCount, NsrGetMallocMultiStat 関数の追加。
2000.09.11	1.21	・ ライブラリのバージョンアップに伴いバージョン番号を更新。

# 目 次

1. 概 要 .....	1
1.1 目 的 .....	1
1.2 モジュール構成 .....	1
2. NSR の仕組み .....	2
3. ライブラリの使用方法 .....	3
4. NSR ライブラリの動作 .....	5
4.1 NSR ハンドルの動作状態 .....	5
5. データの作成方法 .....	9
5.1 言語データ .....	9
5.2 コンテキストデータ .....	9
5.3 クラスデータ .....	9
6. データ仕様 .....	10
6.1 定 数 .....	11
6.2 データ型 .....	13
7. 関数仕様 .....	14
7.1 初期化と終了処理 .....	15
7.2 基本動作処理 .....	16
7.3 拡張動作処理 .....	22
7.4 状態取得処理 .....	27
7.5 エラー処理 .....	31

## 1. 概 要

### 1.1 目 的

本ライブラリは、音声認識ライブラリ ASR1600/C (以下 ASR)を簡易に利用するためのライブラリです。ASR や Shinobi の音声サンプリングライブラリ wssip を直接使用せずに音声認識を行うことができます。

本ライブラリはソースを公開していますので無保証での提供となります。

### 1.2 モジュール構成

簡易音声認識ライブラリ(NSR)のモジュール構成図を以下に示します。

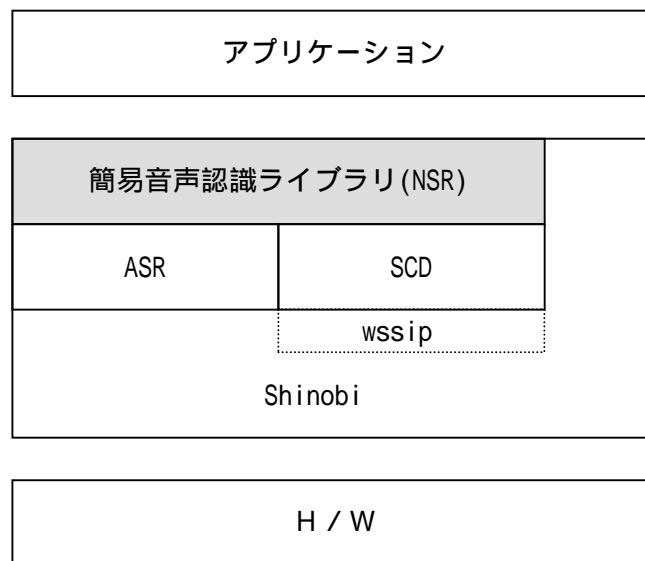
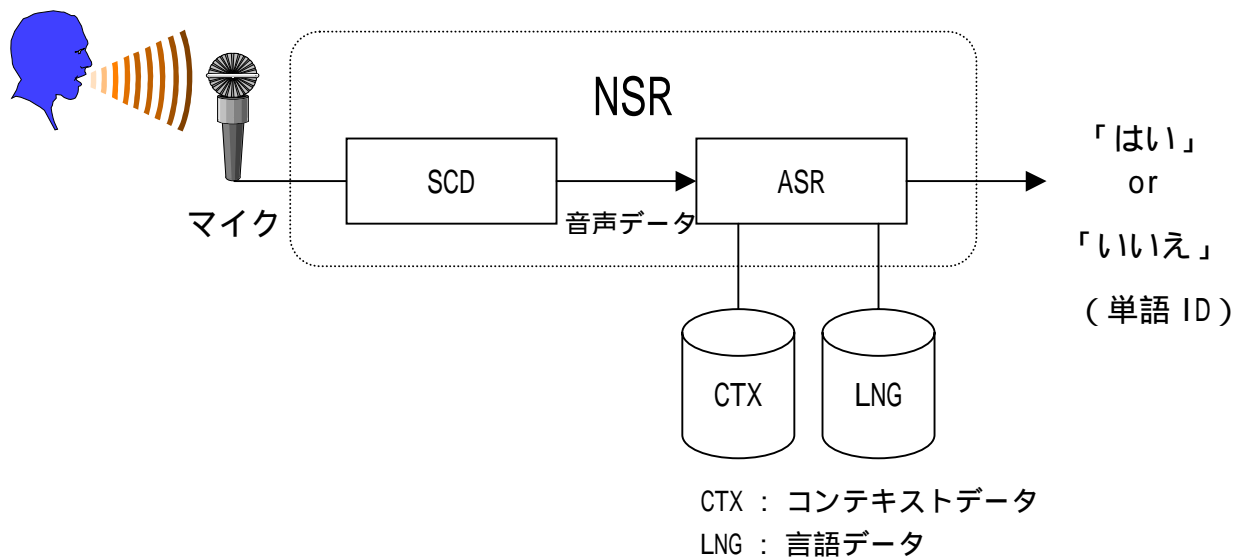


図 1 - 1 モジュール構成

SCD は 音声サンプリングライブラリ wssip を簡易に利用できるようにしたモジュール

## 2. NSR の仕組み



### 3. ライブラリの実用方法

以下に、Push to talk 形式の音声認識サンプルプログラムを示します。

#### < サンプルプログラム >

```
void main(void)
{
    NSR          nsr;                /* NSR ハンドル */
    NSR_STAT     nsr_status;         /* NSR の状態 */
    Sint32       word_id;            /* 認識結果の単語 ID */
    Sint32       conf;               /* 認識結果の信頼値 */
    long         *lngbuff;           /* 言語データ */
    void         *ctxbuff;           /* コンテキストデータ */

    NsrInit();                      /* ライブラリの初期化 */
    nsr = NsrCreate(SCDE_DEV_A2, lngbuff, ctxbuff); /* ハンドルの生成 */
    for (;;) {
        per = pdGetPeripheral(PDD_PORT_A0);

        if( per->press & PDD_DGT_TA ){
            NsrStart(nsr);           /* 認識開始 */
        }
        if( per->release & PDD_DGT_TA ){
            NsrTerminateInput(nsr);  /* 音声データの入力停止 */
        }
        nsr_status = NsrGetStat(nsr); /* 状態の取得 */

        if(nsr_status == NSRE_STAT_DONE){
            word_id = NsrGetResult(nsr, &conf); /* 認識結果取得 */
            /* 必要に応じて 信頼値 conf による棄却処理を追加 */
        }
        NsrExecServer();             /* サーバ関数 */
        /* アプリケーション処理 */
        njWaitVSync();
    }

    NsrDestroy(nsr);                /* ハンドルの消去 */
    NsrFinish();                    /* ライブラリの終了処理 */
}
```

以下に、ユーザワードトレーニングのサンプルプログラムを示します。ユーザワードの認識は通常の認識と同じ方法で行えますのでここでは省略します。

### < サンプルプログラム >

```
void main(void)
{
    NSR          nsr;                /* NSR ハンドル */
    NSR_STAT     nsr_status;         /* NSR の状態 */
    Uint32       first_uw_id;        /* ユーザワード追加時の先頭単語 ID */
    Uint32       uw_id_num;          /* ユーザワード追加時の単語 ID 個数 */
    long         *lngbuff;           /* 言語データ */
    void         *ctxbuff;           /* コンテキストデータ */
    void         *clsbuff;           /* クラスデータ */
    void         *uwbuff;            /* ユーザワードデータ */
    Uint32       uwbuff_size;        /* ユーザワードデータのサイズ */

    NsrInit();                      /* ライブラリの初期化 */
    nsr = NsrCreate(SCDE_DEV_A2, lngbuff, ctxbuff); /* ハンドルの生成 */
    NsrCreateClass(nsr, clsbuff);    /* クラスハンドルの生成 */
    for (;;) {
        per = pdGetPeripheral(PDD_PORT_A0);
        nsr_status = NsrGetStat(nsr); /* 状態の取得 */
        switch(nsr_status){
            case NSRE_STAT_STOP:
                NsrEnterTrainMode(nsr); /* ユーザワードトレーニングモード */
                break;
            case NSRE_STAT_TRAIN_READY:
            case NSRE_STAT_TRAIN_EXEC:
                if( per->press & PDD_DGT_TA ){
                    NsrStartTrain(nsr); /* トレーニング開始 */
                }
                if( per->release & PDD_DGT_TA ){
                    NsrTerminateInputTrain(nsr); /* 音声データの入力停止 */
                }
                break;
            case NSRE_STAT_TRAIN_DONE:
                uwbuff_size = NsrGetUserWord(nsr, &uwbuff); /* ユーザワード取得 */
                NsrAddUserWord(nsr, "userword", uwbuff, /* ユーザワードをクラスに追加 */
                               uwbuff_size, &first_uw_id, &uw_id_num);
                NsrStopTrain(nsr); /* ユーザワードトレーニングモード停止 */
                break;
            default:
                break;
        }
        NsrExecServer(); /* サーバ関数 */
        /* アプリケーション処理 */
        njWaitVSync();
    }
    NsrDestroyClass(nsr); /* クラスハンドルの消去 */
    NsrDestroy(nsr);      /* ハンドルの消去 */
    NsrFinish();          /* ライブラリの終了処理 */
}
```

## 4. NSR ライブラリの動作

### 4.1 NSR ハンドルの動作状態

ハンドルの動作状態を下記に示します。

表 4 - 1 NSR ハンドルの動作状態

状態	説 明
STOP	停止状態
EXEC	認識実行状態
DONE	認識完了状態
ERR	エラー状態
TRAIN_READY	ユーザワードトレーニング実行可能状態
TRAIN_EXEC	ユーザワードトレーニング実行状態
TRAIN_DONE	ユーザワードトレーニング完了状態

状態遷移図を以下に示します。

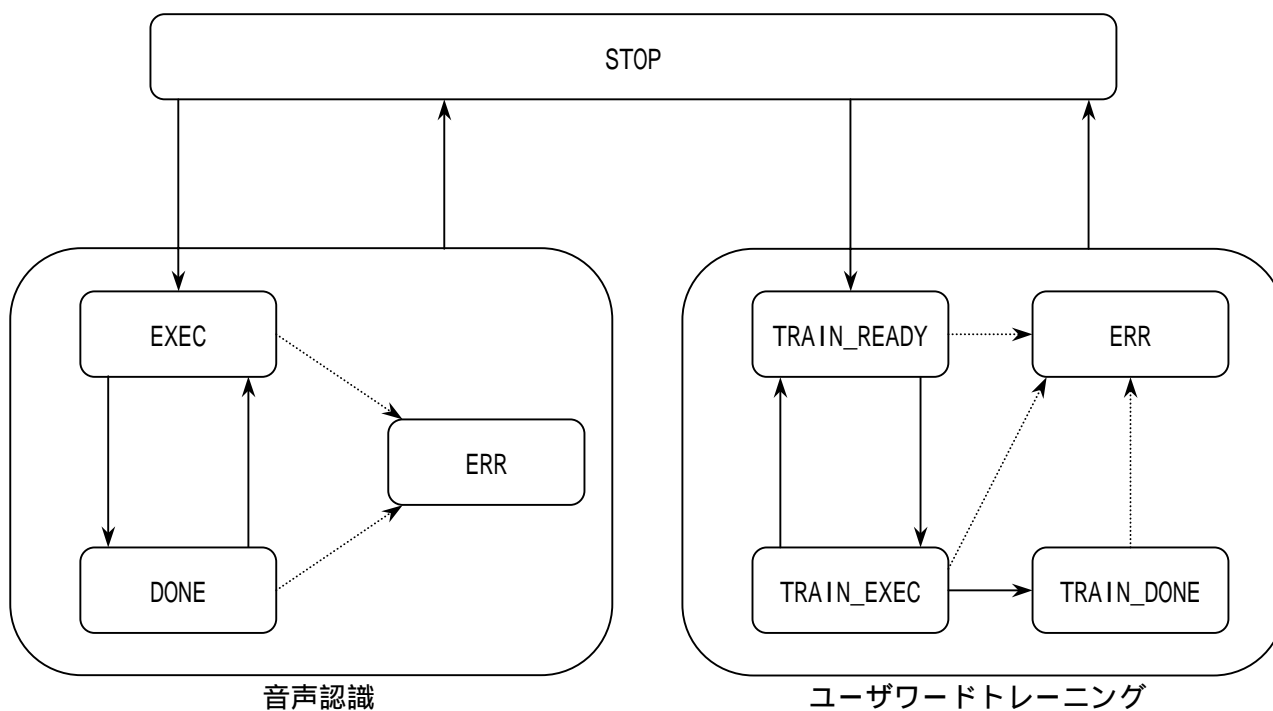


図 4 - 1 状態遷移図



状態遷移図の音声認識部分を以下に示します。

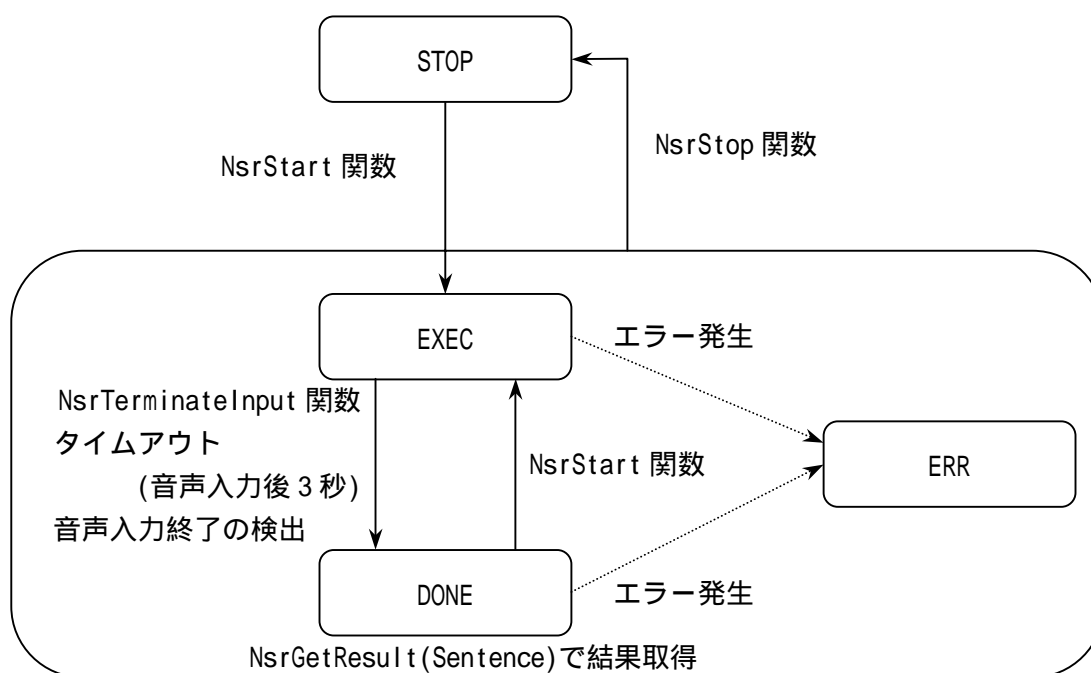


図 4 - 2 音声認識状態遷移図

ハンドルを生成した直後は STOP 状態となり、認識を開始すると EXEC 状態になります。

認識エンジンへの音声データの入力を停止させるか、タイムアウトで DONE 状態になります。DONE 状態は認識結果が更新されたことを示します。

認識を停止させると STOP 状態になり、認識結果は更新されません。

NSR や SCD にエラーが発生すると、ERR 状態に遷移します。ERR 状態は通常起こりませんが ERR 状態になった場合、その後 NSR の動作は保証できません。システムの初期化が必要になります。

状態遷移図のユーザワードトレーニング部分を以下に示します。

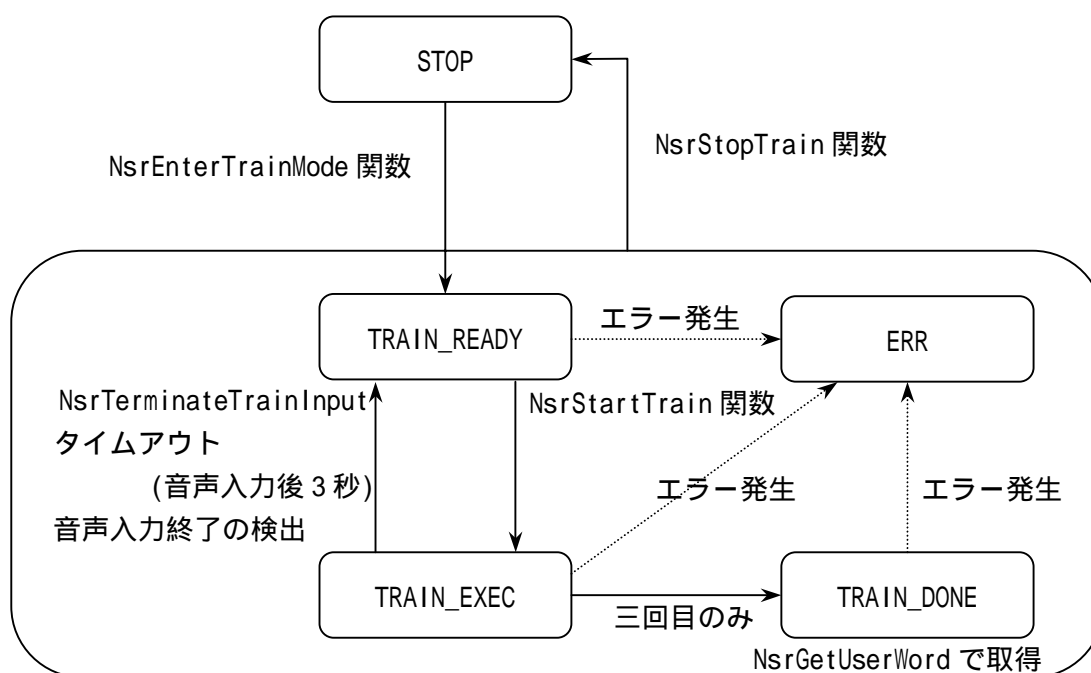


図 4 - 3 ユーザワードトレーニング状態遷移図

STOP 状態は図 4 - 2 の STOP 状態と同じものを指します。NsrEnterTrainMode 関数を実行すると TRAIN\_READY 状態になります。NsrStartTrain 関数により音声データの入力を開始すると、TRAIN\_EXEC 状態になります。認識エンジンへの音声データの入力を停止させるか、タイムアウトで TRAIN\_READY 状態に戻ります。TRAIN\_READY と TRAIN\_EXEC を三度繰り返すと、TRAIN\_READY に戻るのではなく、TRAIN\_DONE 状態になります。TRAIN\_DONE 状態はトレーニングが終了したことを示します。

トレーニングを停止させると STOP 状態になり、トレーニング内容は破棄されます。

NSR や SCD にエラーが発生すると、ERR 状態に遷移します。ERR 状態は通常起こりませんが ERR 状態になった場合、その後 NSR の動作は保証できません。システムの初期化が必要になります。

ユーザワードトレーニングの動作の流れを以下に示します。

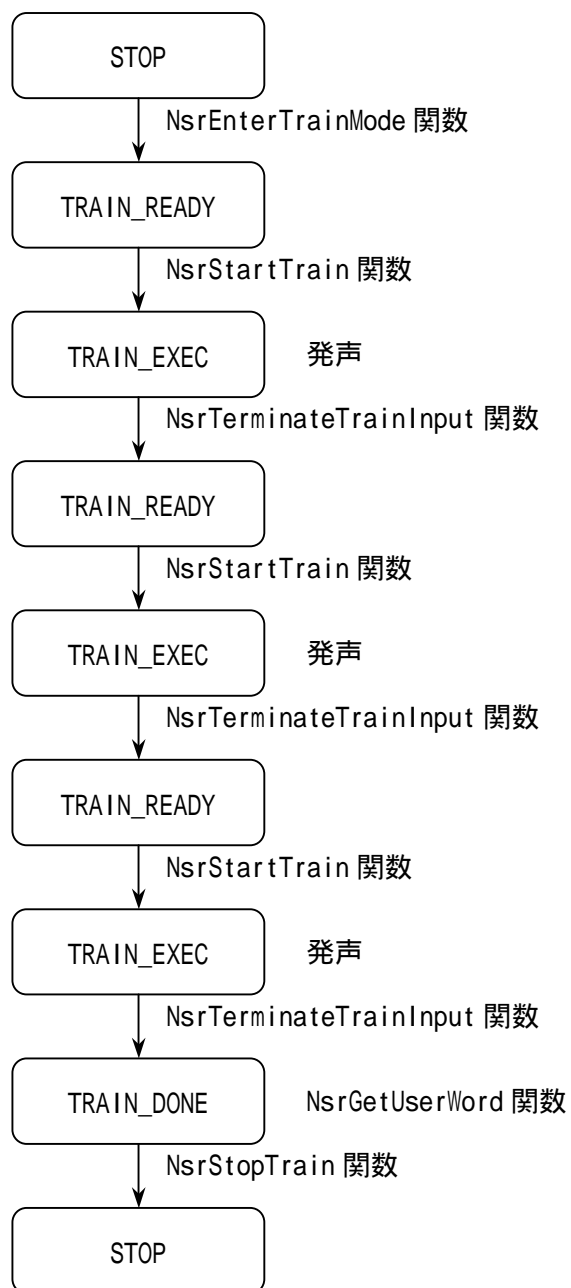


図 4 - 4 ユーザワードトレーニングフロー

## 5. データの作成方法

本ライブラリでは言語データとコンテキストデータの2つのデータを使用します。

各データや用語についての詳細については「ドリームキャストプラットフォーム用 ASR 1600/C 音声認識アプリケーション開発ガイド」を参照してください。

### 5.1 言語データ

言語データ(.lng)とは、言語に依存する音声認識エンジンデータと不特定話者ユーザモデルデータへのリンクが含まれたデータを指します。

言語データは以下の通りです。この中から任意の言語データを1つ使用します。

- (a) asr16v2\_jpv113\_float\_8b.lng (日本語用)
- (b) asr16v2\_jpv113\_float\_4b.lng (日本語用圧縮版)
- (c) asr16v2\_aev200\_float\_8b.lng (アメリカ英語用)
- (d) asr16v2\_aev200\_float\_4b.lng (アメリカ英語用圧縮版)

### 5.2 コンテキストデータ

コンテキストデータ(.ctx)とは、認識対象語彙のデータを指し、アプリケーション開発者が作成します。

コンテキストデータは認識対象語彙を規定するテキスト形式の文法ファイル(BNF ファイル)からツールを使用して作成します。詳細については、「コンテキスト作成 導入マニュアル」を参照してください。

### 5.3 クラスデータ

クラスデータ(.cls)とは、コンテキストデータ内の単語クラスのデータを指し、アプリケーション開発者が作成します。

クラスデータは単語クラスの記述を含む文法ファイル(BNF ファイル)からツールを使用して作成します。詳細については、「Development Tools User 's Guide」を参照してください。

## 6. データ仕様

ライブラリのデータ一覧を以下に示します。

表 6 - 1 データ一覧

データ名		機 能	番号
定数			
	NSRE_STAT_ ~	NSR ハンドルの状態	1.1
	NSRE_ABCOND_ ~	入力音声の異常状態	1.2
	NSRE_VDMODE_ ~	音声区間検出モード	1.3
	SCDE_STAT_ ~	SCD ハンドルの状態	1.4
データ型			
	NSR	NSR ハンドル	2.1

## 6.1 定 数

Title	Data Name	Data	No
データ	NSRE_STAT_~	NSR ハンドルの状態	1.1

以下の定数は、NSR ハンドルの状態を示します。

定数名	説 明
NSRE_STAT_STOP	停止状態
NSRE_STAT_EXEC	認識実行状態
NSRE_STAT_DONE	認識完了状態
NSRE_STAT_ERR	エラー状態
NSRE_STAT_TRAIN_READY	ユーザワードトレーニング実行可能状態
NSRE_STAT_TRAIN_EXEC	ユーザワードトレーニング実行状態
NSRE_STAT_TRAIN_DONE	ユーザワードトレーニング完了状態

Title	Data Name	Data	No
データ	NSRE_ABCOND_~	入力音声の異常状態	1.2

以下の定数は、入力音声の異常状態を示します。

定数名	説 明
NSRE_ABCOND_NORMAL	正常
NSRE_ABCOND_BADSNR	S/N 比が低すぎる
NSRE_ABCOND_OVERLOAD	音量が大きすぎる
NSRE_ABCOND_TOOQUIET	音量が小さすぎる
NSRE_ABCOND_NOSIGNAL	AGC 開始後、信号が確認できない
NSRE_ABCOND_GARBLED_SOUND	非常に短い雑音を検出された
NSRE_ABCOND_POORMIC	マイクの性能が良好ではない
NSRE_ABCOND_UNKNOWN	未定義の異常状態

Title データ	Data Name NSRE_VDMODE_ ~	Data 音声区間検出モード	No 1.3
--------------	-----------------------------	-------------------	-----------

以下の定数は、音声区間検出モードを示します。

定数名	説 明
NSRE_VDMODE_MANUAL	手動検出モード。 音声区間を検出しないのでアプリケーションで NsrStart, NsrTerminateInput 関数により認識処理を制御してください。
NSRE_VDMODE_ENDAUTO	終了検出モード。 音声入力の開始を検出しないので、NsrStart によって認識処理を開始します。音声入力の終了を検出し、自動的に認識完了状態(NSRE_STAT_DONE)に遷移します。また NsrTerminateInput 関数も使用できます。
NSRE_VDMODE_FULLLAUTO	自動検出モード。 NsrStart 関数実行後、音声入力の開始を検出して認識処理を開始します。音声入力の終了を検出し、自動的に認識完了状態(NSRE_STAT_DONE)に遷移します。また NsrTerminateInput 関数も使用できます。ユーザワードトレーニング時は音声入力の開始の検出は行われず、終了検出モードと同様の動作となります。ユーザワードトレーニング終了後は自動検出モードの動作に戻ります。

Title データ	Data Name SCDE_STAT ~	Data SCD ハンドルの状態	No 1.4
--------------	--------------------------	---------------------	-----------

以下の定数は、SCD ハンドルの状態を示します。

定数名	説 明
SCDE_STAT_DISCNCT	音声入力デバイス未接続
SCDE_STAT_STOP	リングバッファへの取り込み停止状態
SCDE_STAT_REC	リングバッファへの取り込み中
SCDE_STAT_ERR	エラー
SCDE_STAT_RETRY	エラーリトライ中
SCDE_STAT_RESTART	音声入力デバイスの CPU によってリスタート中

## 6.2 データ型

Title	Data Name	Data	No
データ	NSR	NSR ハンドル	1.1

NSR を制御するためのハンドルです。NsrCreate 関数で生成されます。



## 7. 関数仕様

ライブラリの関数一覧を以下に示します。

表 7 - 1 関数一覧

関数名	機 能	番号
初期化と終了処理		
NsrInit	ライブラリの初期化	1.1
NsrFinish	ライブラリの終了処理	1.2
基本動作処理		
NsrExecServer	サーバ関数	2.1
NsrCreate	ハンドルの生成	2.2
NsrDestroy	ハンドルの消去	2.3
NsrStart	認識開始	2.4
NsrTerminateInput	音声データの入力停止	2.5
NsrStop	認識停止	2.6
NsrGetResult	認識結果の取得（1 単語）	2.7
NsrGetResultSentence	認識結果の取得（連続単語）	2.8
NsrSetVoiceDetectMode	音声区間検出モードの設定	2.9
拡張動作処理		
NsrCreateClass	クラスハンドルの生成	3.1
NsrDestroy	クラスハンドルの消去	3.2
NsrEnterTrainMode	ユーザワードトレーニングモードに入る	3.3
NsrStartTrain	ユーザワードトレーニング開始	3.4
NsrTerminateTrainInput	トレーニング音声データの入力停止	3.5
NsrStopTrain	ユーザワードトレーニングモード停止	3.6
NsrGetUserWord	ユーザワード(トレーニング結果)の取得	3.7
NsrAddUserWord	ユーザワード追加	3.8
NsrDeleteUserWord	ユーザワード削除	3.9
状態取得処理		
NsrGetStat	NSR の状態取得	4.1
NsrGetAsrStat	ASR の状態取得(デバッグ用)	4.2
NsrGetScdStat	SCD の状態取得	4.3
NsrGetAbnormalCondition	入力音声の異常状態取得	4.4
NsrClearAbnormalCondition	入力音声の異常状態クリア	4.5
NsrGetTrainCount	ユーザワードトレーニング回数の取得	4.6
NsrGetMallocMultiStat	NSR 用 Multi heap の状態取得	4.7
エラー処理		
NsrEntryErrFunc	エラーコールバック関数の登録	5.1

## 7.1 初期化と終了処理

Title	Function Name	Function	No
関 数	NsrInit	ライブラリの初期化	1.1

[書 式] void NsrInit(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] ライブラリを初期化します。

Title	Function Name	Function	No
関 数	NsrFinish	ライブラリの終了処理	1.2

[書 式] void NsrFinish(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] ライブラリの終了処理を行います。

## 7.2 基本動作処理

Title	Function Name	Function	No
関 数	NsrExecServer	サーバ関数	2.1

[ 書 式 ] void NsrExecServer(void);

[ 入 力 ] なし

[ 出 力 ] なし

[ 関数値 ] なし

[ 機 能 ] ライブラリの内部状態を更新し、音声データの取込・認識処理を行います。

[ 備 考 ] V-Sync 毎に呼び出す必要があります。

NSRE\_STAT\_EXEC 状態は 3 秒間 NsrTarminateInput 関数が呼ばれない場合自動的に NSRE\_STAT\_DONE 状態に遷移します。

Title 関 数	Function Name NsrCreate	Function ハンドルの生成	No 2.2
--------------	----------------------------	---------------------	-----------

[ 書 式 ] NSR NsrCreate(Sint32 pno, PDATA lngbuff, PCONT ctxbuff);

[ 入 力 ] pno : ポート番号  
lngbuff : 言語データ  
ctxbuff : コンテキストデータ

[ 出 力 ] なし

[ 関数値 ] NSR ハンドル

[ 機 能 ] ハンドルを生成します。  
ポート番号を示す引数 pno には次の値が定義されています。

定数名	説 明
SCDE_DEV_A1	コントロールポート A の拡張ソケット 1
SCDE_DEV_A2	コントロールポート A の拡張ソケット 2
SCDE_DEV_A3	コントロールポート A の拡張ソケット 3
SCDE_DEV_A4	コントロールポート A の拡張ソケット 4
SCDE_DEV_A5	コントロールポート A の拡張ソケット 5
SCDE_DEV_B1	コントロールポート B の拡張ソケット 1
SCDE_DEV_B2	コントロールポート B の拡張ソケット 2
SCDE_DEV_B3	コントロールポート B の拡張ソケット 3
SCDE_DEV_B4	コントロールポート B の拡張ソケット 4
SCDE_DEV_B5	コントロールポート B の拡張ソケット 5
SCDE_DEV_C1	コントロールポート C の拡張ソケット 1
SCDE_DEV_C2	コントロールポート C の拡張ソケット 2
SCDE_DEV_C3	コントロールポート C の拡張ソケット 3
SCDE_DEV_C4	コントロールポート C の拡張ソケット 4
SCDE_DEV_C5	コントロールポート C の拡張ソケット 5
SCDE_DEV_D1	コントロールポート D の拡張ソケット 1
SCDE_DEV_D2	コントロールポート D の拡張ソケット 2
SCDE_DEV_D3	コントロールポート D の拡張ソケット 3
SCDE_DEV_D4	コントロールポート D の拡張ソケット 4
SCDE_DEV_D5	コントロールポート D の拡張ソケット 5

Title 関 数	Function Name NsrDestroy	Function ハンドルの消去	No 2.3
--------------	-----------------------------	---------------------	-----------

[書 式] void NsrDestroy(NSR nsr);  
 [入 力] nsr : NSR ハンドル  
 [出 力] なし  
 [関数値] なし  
 [機 能] ハンドルを消去します。

Title 関 数	Function Name NsrStart	Function 認識開始	No 2.4
--------------	---------------------------	------------------	-----------

[書 式] void NsrStart(NSR nsr);  
 [入 力] nsr : NSR ハンドル  
 [出 力] なし  
 [関数値] なし  
 [機 能] 音声認識を開始します。

Title 関 数	Function Name NsrTerminateInput	Function 音声データの入力停止	No 2.5
--------------	------------------------------------	------------------------	-----------

[書 式] void NsrTerminateInput(NSR nsr);  
[入 力] nsr : NSR ハンドル  
[出 力] なし  
[関数値] なし  
[機 能] 認識エンジンへの音声データの入力を停止します。

Title 関 数	Function Name NsrStop	Function 認識停止	No 2.6
--------------	--------------------------	------------------	-----------

[書 式] void NsrStop(NSR nsr);  
[入 力] nsr : NSR ハンドル  
[出 力] なし  
[関数値] なし  
[機 能] 音声認識を停止します。  
[備 考] 認識結果を取得することはできません。

Title 関 数	Function Name NsrGetResult	Function 認識結果取得 (1 単語)	No 2.7
--------------	-------------------------------	---------------------------	-----------

[ 書 式 ] Sint32 NsrGetResult(NSR nsr, Sint32 \*confidence);

[ 入 力 ] nsr : NSR ハンドル

[ 出 力 ] confidence : 信頼値 (0 ~ 10000)

[ 関数値 ] 認識した単語 ID

[ 機 能 ] 認識の結果を取得します。

[ 備 考 ] NSR の動作状態が NSRE\_STAT\_EXEC から NSRE\_STAT\_DONE に遷移したときに認識結果が更新されます。

認識結果は次の結果に更新されるまで値を保持します。

Title 関 数	Function Name NsrGetResultSentence	Function 認識結果取得 (連続単語)	No 2.8
--------------	---------------------------------------	---------------------------	-----------

[ 書 式 ] Sint32 NsrGetResultSentence

(NSR nsr, Sint32 num\_word, Sint32 \*word\_id, Sint32 \*confidence);

[ 入 力 ] nsr : NSR ハンドル

num\_word : 認識する最大単語数

[ 出 力 ] word\_id : 認識した単語 ID

confidence : 信頼値 (0 ~ 10000)

[ 関数値 ] 認識した単語の数

[ 機 能 ] 連続単語認識の結果を取得します。

[ 備 考 ] NSR の動作状態が NSRE\_STAT\_EXEC から NSRE\_STAT\_DONE に遷移したときに認識結果が更新されます。

認識結果は次の結果に更新されるまで値を保持します。

関数値は、num\_word に関わらず認識した単語数になります。

Title 関 数	Function Name NsrSetVoiceDetectMode	Function 音声区間検出モードの設定	No 2.9
--------------	--	--------------------------	-----------

[書 式] void NsrSetVoiceDetectMode(NSR nsr, NSRE\_VDMODE vd\_mode);

[入 力] nsr : NSR ハンドル  
vd\_mode : 音声区間検出モード

[出 力] なし

[関数値] なし

[機 能] 音声区間検出モードを設定します。

[備 考] NSR が停止状態 (NSRE\_STAT\_STOP)の時に実行してください。  
それ以外の状態の場合は無視されます。

音声区間検出モード vd\_mode に指定する定数は以下の通りです。

定数名	説 明
NSRE_VDMODE_MANUAL	手動検出モード。 音声区間を検出しないのでアプリケーションで NsrStart,NsrTerminateInput 関数により認識処 理を制御してください。
NSRE_VDMODE_ENDAUTO	終了検出モード。 音声入力を開始を検出しないので、NsrStart によ って認識処理を開始します。音声入力の終了を検 出し、自動的に認識完了状態(NSRE_STAT_DONE)に 遷移します。また NsrTerminateInput 関数も使用 できます。
NSRE_VDMODE_FULLAUTO	自動検出モード。 NsrStart 関数実行後、音声入力を開始を検出して 認識処理を開始します。音声入力の終了を検出 し、自動的に認識完了状態(NSRE_STAT_DONE)に遷 移します。また NsrTerminateInput 関数も使用で きます。



### 7.3 拡張動作処理

Title	Function Name	Function	No
関 数	NsrCreateClass	クラスハンドルの生成	3.1

[書 式] void NsrCreateClass(NSR nsr, PCLASSES clsbuff);

[入 力] nsr : NSR ハンドル  
clsbuff : クラスバッファ

[出 力] なし

[関数値] なし

[機 能] 単語クラスのハンドルを生成し、NSR ハンドルに登録します。NSR ハンドル 1 つにつき単語クラスのハンドルは 1 つのみ生成・登録できます。ユーザワードを利用するには、この単語クラスのハンドルが必要です。  
C++のクラスとは関係ありません。

Title	Function Name	Function	No
関 数	NsrDestroyClass	クラスハンドルの消去	3.2

[書 式] void NsrDestroyClass(NSR nsr);

[入 力] nsr : NSR ハンドル

[出 力] なし

[関数値] なし

[機 能] 単語クラスのハンドルを消去します。

Title 関 数	Function Name NsrEnterTrainMode	Function ユーザワードトレーニングモードに入る	No 3.3
--------------	------------------------------------	--------------------------------	-----------

[書 式] void NsrEnterTrainMode(NSR nsr);  
 [入 力] nsr : NSR ハンドル  
 [出 力] なし  
 [関数値] なし  
 [機 能] ユーザワードトレーニングモードに移行します。

Title 関 数	Function Name NsrStartTrain	Function ユーザワードトレーニング開始	No 3.4
--------------	--------------------------------	----------------------------	-----------

[書 式] void NsrStartTrain(NSR nsr);  
 [入 力] nsr : NSR ハンドル  
 [出 力] なし  
 [関数値] なし  
 [機 能] ユーザワードトレーニングを開始します。

Title 関 数	Function Name	Function	No
	NsrTerminateTrainInput	トレーニング音声データの入力停止	3.5

[書 式] void NsrTerminateTrainInput(NSR nsr);

[入 力] nsr : NSR ハンドル

[出 力] なし

[関数値] なし

[機 能] 認識エンジンへのトレーニング音声データの入力を停止します。

Title 関 数	Function Name	Function	No
	NsrStopTrain	ユーザワードトレーニングモード停止	3.6

[書 式] void NsrStop(NSR nsr);

[入 力] nsr : NSR ハンドル

[出 力] なし

[関数値] なし

[機 能] ユーザワードトレーニングを停止し、ユーザワードトレーニングモードから抜けます。

Title 関 数	Function Name	Function	No
	NsrGetUserWord	ユーザワード（トレーニング結果）の取得	3.7

[ 書 式 ] Uint32 NsrGetUserWord(NSR nsr, USERWORDBUF \*uw\_buff);

[ 入 力 ] nsr : NSR ハンドル

[ 出 力 ] uw\_buff : ユーザワードデータ

[ 関数値 ] ユーザワードデータのサイズ

[ 機 能 ] トレーニング結果であるユーザワードのデータを取得します。

[ 備 考 ] NSR の動作状態が NSRE\_STAT\_TRAIN\_DONE の時にユーザワードデータが取得できます。

トレーニングの音声入力 が 3 回とも短すぎた場合などユーザワードデータのサイズが 0 バイトになる場合があります。この場合は単語クラスに追加することはできません。

NsrStopTrain 関数によりユーザワードトレーニングモードを終了すると、

NsrGetUserWord 関数により取得したユーザワードデータは無効となります。必要に応じてあらかじめ別領域に保存してください。

Title 関 数	Function Name	Function	No
	NsrAddUserWord	ユーザワード追加	3.8

[ 書 式 ] void NsrAddUserWord(NSR nsr, Sint8 \*class\_name, USERWORDBUF uw\_buff,  
 Uint32 uw\_buff\_size, Uint32 \*first\_uw\_id, Uint32 \*uw\_id\_num);

[ 入 力 ] nsr : NSR ハンドル

class\_name : 単語クラス名

uw\_buff : ユーザワードデータ

uw\_buff\_size : ユーザワードデータのサイズ

[ 出 力 ] first\_uw\_id : 割り当てられた最初のユーザワード ID (0x40000000 ~)

uw\_id\_num : 割り当てられたユーザワードの数

[ 関数値 ] なし

[ 機 能 ] ユーザワードをコンテキストの単語クラスに追加します。

[ 備 考 ] 1 つのコンテキストに 1 つの単語クラスが複数回使用されているとユーザワード ID が複数個生成されます。

ユーザワード ID は 0x40000000 以降に割り当てられます。(ASR では 0x80000000 ~)

Title 関 数	Function Name NsrDeleteUserWord	Function ユーザワード削除	No 3.9
--------------	------------------------------------	----------------------	-----------

[ 書 式 ] void NsrDeleteUserWord(NSR nsr, Sint8 \*class\_name,  
 Uint32 first\_uw\_id, Uint32 uw\_id\_num);

[ 入 力 ] nsr : NSR ハンドル  
 class\_name : 単語クラス名  
 first\_uw\_id : 削除する最初のユーザワード ID  
 uw\_id\_num : 削除するユーザワードの数

[ 出 力 ] なし

[ 関数値 ] なし

[ 機 能 ] ユーザワードをコンテキストの単語クラスから削除します。

[ 備 考 ] 直前に追加したユーザワードのみが削除可能です。

## 7.4 状態取得処理

Title 関 数	Function Name NsrGetStat	Function NSR の状態の取得	No 4.1
--------------	-----------------------------	------------------------	-----------

[ 書 式 ] NSR\_STAT NsrGetStat(NSR nsr);  
 [ 入 力 ] nsr : NSR ハンドル  
 [ 出 力 ] なし  
 [ 関数値 ] 現在の NSR の動作状態  
 [ 機 能 ] 現在の NSR の動作状態を取得します。  
 動作状態は以下の通りです。

定数名	説 明
NSRE_STAT_STOP	停止状態
NSRE_STAT_EXEC	認識実行状態
NSRE_STAT_DONE	認識完了状態
NSRE_STAT_ERR	エラー状態

Title 関 数	Function Name NsrGetAsrStat	Function ASR の状態取得	No 4.2
--------------	--------------------------------	-----------------------	-----------

[ 書 式 ] Sint32 NsrGetAsrStat(NSR nsr);  
 [ 入 力 ] nsr : NSR ハンドル  
 [ 出 力 ] なし  
 [ 関数値 ] 現在の ASR の動作状態  
 [ 機 能 ] 現在の ASR の動作状態を取得します。  
 動作状態は以下の通りです。

定数名	説 明
CASR_BOOT	コンテキスト、言語、ユーザ、いずれも非アクティブな初期状態
CASR_DATAREADY	エンジンに必要なすべてのデータがアクティブ化されている状態
CASR_IDLE	コンテキスト、言語、ユーザがアクティブな状態
CASR_SLEEP	低 CPU 負荷の音声検出システムが動作中の状態
CASR_RUN	音声認識中
CASR_RECOVER	後続無音が検出された、あるいは、CasrStop 関数が呼び出された状態。音声認識結果を出力し、その旨を通知します。
CASR_PROMPT	ユーザワードトレーニング中の発話開始のユーザフィードバック待ち状態
CASR_RECORD	ユーザワードの発話を録音中
CASR_ACCEPT	ユーザワードの発話の受理依頼待機状態
CASR_CONFIRM	ユーザワードの発話を録音中

Title 関 数	Function Name NsrGetScdStat	Function SCD の状態の取得	No 4.3
--------------	--------------------------------	------------------------	-----------

[ 書 式 ] Sint32 NsrGetScdStat(NSR nsr);  
[ 入 力 ] nsr : NSR ハンドル  
[ 出 力 ] なし  
[ 関数値 ] 現在の SCD の動作状態  
[ 機 能 ] 現在の SCD の動作状態を取得します。  
動作状態は以下の通りです。

定数名	説 明
SCDE_STAT_DISCNCT	音声入力デバイス未接続
SCDE_STAT_STOP	リングバッファへの取り込み停止状態
SCDE_STAT_REC	リングバッファへの取り込み中
SCDE_STAT_ERR	エラー
SCDE_STAT_RETRY	エラーリトライ中
SCDE_STAT_RESTART	音声入力デバイスの CPU によってリスタート中

Title 関 数	Function Name NsrGetAbnormalCondition	Function 入力音声の異常状態取得	No 4.4
--------------	--	-------------------------	-----------

[ 書 式 ] NSR\_ABCOND NsrGetAbnormalCondition(NSR nsr);  
[ 入 力 ] nsr : NSR ハンドル  
[ 出 力 ] なし  
[ 関数値 ] 入力音声の異常状態  
[ 機 能 ] 入力音声の異常状態を取得します。  
[ 備 考 ] 入力音声の異常状態は以下の通りです。

定数名	説 明
NSRE_ABCOND_NORMAL	正常
NSRE_ABCOND_BADSNR	S/N 比が低すぎる
NSRE_ABCOND_OVERLOAD	音量が大き過ぎる
NSRE_ABCOND_TOOQUIET	音量が小さ過ぎる
NSRE_ABCOND_NOSIGNAL	AGC 開始後、信号が確認できない
NSRE_ABCOND_GARBLED SOUND	非常に短い雑音を検出された
NSRE_ABCOND_POORMIC	マイクの性能が良好ではない
NSRE_ABCOND_UNKNOWN	未定義の異常状態

Title 関 数	Function Name NsrClearAbnormalCondition	Function 入力音声の異常状態クリア	No 4.5
--------------	--	--------------------------	-----------

[書 式] void NsrClearAbnormalCondition(NSR nsr);

[入 力] nsr : NSR ハンドル

[出 力] なし

[関数値] なし

[機 能] 入力音声の異常状態をクリアします。

[備 考] 入力音声の異常状態を NSRE\_ABCOND\_NORMAL ( 正常 ) にセットします。

Title 関 数	Function Name NsrGetTrainCount	Function ユーザーワードトレーニング回数取得	No 4.6
--------------	-----------------------------------	-------------------------------	-----------

[書 式] Sint32 NsrGetTrainCount(NSR nsr);

[入 力] nsr : NSR ハンドル

[出 力] なし

[関数値] ユーザーワードトレーニング回数

-1 : ユーザワードトレーニング中ではない

1,2,3: 現在のユーザワードトレーニング回数

4 : ユーザワードトレーニング終了時(備考参照)

[機 能] 現在のユーザーワードトレーニング回数を取得します。

[備 考] トレーニングを完了した回数でないことに注意してください。

トレーニング完了判断はこの関数値で行わないでください。必ず NSR ハンドルの状態を取得し NSRE\_STAT\_TRAIN\_DONE であるか否かで判断してください。



Title 関 数	Function Name NsrGetMallocMultiStat	Function NSR 用 Multi heap の状態取得	No 4.7
--------------	--	------------------------------------	-----------

[ 書 式 ] void NsrGetMallocMultiStat(UINT32 \*wholeFreeSize, UINT32 \*biggestFreeBlockSize);

[ 入 力 ] なし

[ 出 力 ] wholeFreeSize : 利用可能なヒープ領域の残り  
biggestFreeBlockSize : 一回の確保で可能な最大サイズ

[ 関数値 ] なし

[ 機 能 ] NSR で使用しているサブヒープの使用状況を取得します。

[ 備 考 ] NSR を Multi heap を不使用にしてコンパイルした場合は出力は 0 となります。  
NSR 内でサブヒープを使い切ると動作しなくなり、その後の動作は保証されません。  
NSR は標準で 256K バイト確保します。必要に応じて変更してください。

## 7.5 エラー処理

Title	Function Name	Function	No
関 数	NsrEntryErrFunc	エラーコールバック登録	5.1

[ 書 式 ] void NsrEntryErrFunc(void(\*fn)(void \*obj, char \*msg), void \*obj);

[ 入 力 ] fn : ユーザのコールバック関数  
obj : コールバック関数の第 1 引数

[ 出 力 ] なし

[ 関数値 ] なし

[ 機 能 ] エラーコールバック関数を登録します。エラーが発生すると、登録されたコールバック関数が以下の形式で呼び出されます。

```
(*fn)(void *obj, char *msg);
```

この関数の第 1 引数 obj は、NsrEntryErrFunc 関数の第 2 引数となります。また、第 2 引数 msg は、エラーメッセージです。この関数は、デバッグ用の関数なので、アプリケーションのマスタアップ時には、何もしない関数に置き換えてください。