EPSON
EXCEED YOUR VISION

S1V30080 シリーズ
I2C インタフェース
サンプルプログラム仕様書

セイコーエプソン株式会社

Rev.1.00

本資料のご使用につきましては、次の点にご留意願います。

本資料の内容については、予告無く変更することがあります。

1. 本資料の一部、または全部を弊社に無断で転載、または、複製など他の目的に使用することは堅くお断りいたします。
2. 本資料に掲載される応用回路、プログラム、使用方法等はあくまでも参考情報であり、これら起因する第三者の権利(工業所有権を含む)侵害あるいは損害の発生に対し、弊社はいかなる保証を行うものではありません。また、本資料によって第三者または弊社の工業所有権の実施権の許諾を行うものではありません。
3. 特性値の数値の大小は、数直線上の大小関係で表しています。
4. 本資料に掲載されている製品のうち「外国為替及び外国貿易法」に定める戦略物資に該当するものについては、輸出する場合、同法に基づく輸出許可が必要です。
5. 本資料に掲載されている製品は、生命維持装置その他、きわめて高い信頼性が要求される用途を前提としていません。よって、弊社は本(当該)製品をこれらの用途に用いた場合のいかなる責任についても負いかねます。

目次

1. 概要

本ドキュメントは、S1V30080 をご使用のお客様向けに提供するサンプルプログラムについて記述したものです。本サンプルプログラムは、お客様のホストシステム上で S1V30080 を制御するためのサンプルとなっており、その中で使用している API 関数の仕様等についても説明します。

本ドキュメントは、S1V30080 シリーズメッセージプロトコル仕様書と合わせてご使用ください。

※ 本サンプルプログラムは、弊社にて S1V30080 制御評価で使用したホストシステム向けに作成したものです。そのため、お客様のシステムに搭載する上での動作を保証するものではありません。

2. ファイルリスト

本サンプルプログラムには、以下のディレクトリに全てのソースファイルが含まれています。

¥i2c_src　　(各種ソースファイル)

2.1 メインプログラムファイル

メインプログラムファイルには、ホストプロセッサからの制御により、オーディオ再生やシンセサイザー再生、または、オーディオ再生とシンセサイザー再生のミキシングを行うための一連の制御プログラムが記述されています。

表 2.1 にメインプログラムファイルの一覧を示します。表中の各ファイルは、各々が main 関数を含むメインプログラムとなっています。ご使用の際は、メインプログラムファイルのうち 1 つを選択してご使用ください。

表 2.1　メインプログラムファイル一覧

ファイル名	説明
main_audio_main.c	オーディオ・メイン再生用のメインプログラムファイル
main_audio_multi.c	オーディオ・マルチ再生用のメインプログラムファイル
main_synthesizer_melody.c	シンセサイザー・メロディ再生用のメインプログラム
main_synthesizer_buzztone.c	シンセサイザー・ブザートーン再生用のメインプログラム
main_synthesizer_melodystreaming.c	シンセサイザー・メロディ再生(ストリーミング)用のメインプログラム
main_mixing_main_multi_melody.c	ミキシング再生用メインプログラム (オーディオ・メイン、オーディオ・マルチ、シンセサイザー・メロディの 3 系統の再生方式によるミキシング)
main_mixing_main_multi_buzztone.c	ミキシング再生用メインプログラム (オーディオ・メイン、オーディオ・マルチ、シンセサイザー・ブザートーンの 3 系統の再生方式によるミキシング)
main_mixing_main_multi_melodystreaming.c	ミキシング再生用メインプログラム (オーディオ・メイン、オーディオ・マルチ、ストリーミング再生によるシンセサイザー・メロディの 3 系統の再生方式のミキシング)
main_mixing_easy_start.c	ミキシング再生用メインプログラム (ISC_EASY_START_REQ を使用したミキシング)

2.2 S1V30080 制御用API関数定義ファイル

S1V30080 制御用 API 関数定義ファイルは、S1V30080 を制御するための API 関数が定義されるファイルです。

表 2.2 に S1V30080 制御用 API 関数定義ファイルの一覧を示します。

API 関数仕様の詳細は、「3.2 S1V30080 制御用 API 関数仕様」を参照ください。

表 2.2 S1V30080 制御用 API ソースファイル一覧

ファイル名	説明
i2c_api.c	シリアル通信制御用 API 関数が定義されるソースファイル
i2c_api.h	シリアル通信制御用 API 関数が宣言されるヘッダファイル
gpio.c	シリアル通信以外のデバイス制御用 API 関数が定義されるソースファイル
gpio.h	シリアル通信以外のデバイス制御用 API 関数が宣言されるヘッダファイル

※ “i2c_api.c”は、弊社にて S1V30080 制御評価で使用したホストシステム上の I2C を例にした制御プログラムです。お客様のシステムに組み込む際には、お客様のシステムの仕様に合わせて変更いただく必要があります。

2.3 メッセージファイル

メッセージファイルには、REQ メッセージの配列テーブルが定義されています。

表 2.3 にメッセージファイルの一覧を示します。

表 2.3 メッセージファイル一覧

ファイル名	説明
isc_msgs.c	サンプルプログラム用 REQ メッセージのデータ列が定義されるソースファイル

メッセージファイルには、表 2.4 に示すようなデータが定義されており、メッセージ開始パディングおよび開始コマンドの“0x00, 0xAA”を含む REQ メッセージのデータ列が記述されています。

表 2.4 メッセージファイルの例

```
unsigned char aucIscClkdivConfigReq[] = {
    0x00, 0xAA, 0x06, 0x00, 0x14, 0x00, 0x00, 0x00,
};
```

2.4 その他のソースファイル

表 2.5 に、上記以外のソースファイルの一覧を示します。

表 2.5 その他のソースファイル一覧

ファイル名	説明
isc_msgs.h	S1V30080 メッセージの Length 及び ID が定義されるヘッダファイル
reg_host.h	弊社ホストシステムのレジスタマップが定義されるヘッダファイル

注:“reg_host.h”は、弊社にて S1V30080 制御評価で使用したホストシステム上のレジスタマップが記載されています。お客様のホストシステムに組み込む際には、ホストシステムの仕様に合わせて変更を行う必要があります。

3. サンプルプログラム仕様

本サンプルプログラムで使用する API 関数仕様およびメインプログラムの詳細について説明します。

3.1 メインプログラム

I2C 対応のサンプルプログラムとして、9 種類のメインプログラムを用意しています。各メインプログラムは、下記の手順に沿った制御を行います。

3.1.1 main_audio_main.c / main_audio_multi.c

オーディオ・メイン再生用、オーディオ・マルチ再生用のメインプログラムです。このプログラムでは、単純再生、再生中の強制停止、再生中のミュート制御、再生中のボリューム制御の処理を含んでいます。

1) システム(Host)の初期化を行う
2) I2C の初期化を行う
3) S1V30080 の初期化を行う
(SYSTEM_EN 端子に Low->High を入力)
4) S1V30080 との通信開始前に 1ms のウェイトを入れる
5) チェックサムの有効、無効を設定する
(ISC_CHKSUM_CONFIG_REQ 送信)
6) クロックの分周の設定を行う
(ISC_CLKDIV_CONFIG_REQ 送信)
7) サンプリング周波数やボリュームのサウンド出力の設定を行う
(ISC_AUDIO_CONFIG_REQ 送信)

＜play simple＞

8) オーディオ再生の設定を行う
(ISC_SEQUENCER_(MAIN/MULTI)_CONFIG_REQ 送信)
9) オーディオ再生を開始する
(ISC_SOUND_START_REQ 送信)
10) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play and stop＞

11) オーディオ再生の設定を行う
(ISC_SEQUENCER_(MAIN/MULTI)_CONFIG_REQ 送信)
12) オーディオ再生を開始する
(ISC_SOUND_START_REQ 送信)
13) 再生開始 1 秒後に再生を停止する
(ISC_SOUND_STOP_REQ 送信)
14) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play with mute control＞

15) オーディオ再生の設定を行う
(ISC_SEQUENCER_(MAIN/MULTI)_CONFIG_REQ 送信)

16) オーディオ再生を開始する
(ISC_SOUND_START_REQ 送信)

17) 再生開始 1 秒後にミュートを開始する
(ISC_AUDIO_MUTE_REQ 送信)

18) ミュート開始 1 秒後にミュートを解除する
(ISC_AUDIO_MUTE_REQ 送信)

19) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play with volume control＞

20) オーディオ再生の設定を行う
(ISC_SEQUENCER_(MAIN/MULTI)_CONFIG_REQ 送信)

21) オーディオ再生を開始する
(ISC_SOUND_START_REQ 送信)

22) 再生開始直後にボリュームを徐々に下げ、また、徐々に上げる
(ISC_AUDIO_VOLUME_REQ 繰り返し送信)

23) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

3.1.2 main_synthesizer_melody.c

シンセサイザー・メロディ再生用のメインプログラムです。このプログラムでは、単純再生、再生中の強制停止、再生中のミュート制御、再生中のボリューム制御の処理を含んでいます。

1) システム(Host)の初期化を行う
2) I2C の初期化を行う
3) S1V30080 の初期化を行う
(SYSTEM_EN 端子に Low->High を入力)
4) S1V30080 との通信開始前に 1ms のウェイトを入れる
5) チェックサムの有効/無効を設定する
(ISC_CHKSUM_CONFIG_REQ 送信)
6) クロックの分周の設定を行う
(ISC_CLKDIV_CONFIG_REQ 送信)
7) サンプリング周波数やボリュームのサウンド出力の設定を行う
(ISC_AUDIO_CONFIG_REQ 送信)
8) シンセサイザー・メロディ再生のためのテンポやエンベロープの設定を行う
(ISC_SYNTHESIZER_MELODY_CONFIG_REQ 送信)

＜play simple＞

9) シンセサイザー・メロディ再生のための音符データの設定を行う
(ISC_SYNTHESIZER_MELODY_SET_REQ 送信)
10) シンセサイザー・メロディ再生を開始する
(ISC_SOUND_START_REQ 送信)
11) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play and stop＞

12) シンセサイザー・メロディ再生のための音符データの設定を行う
(ISC_SYNTHESIZER_MELODY_SET_REQ 送信)
13) シンセサイザー・メロディ再生を開始する
(ISC_SOUND_START_REQ 送信)
14) 再生開始 1 秒後に再生を停止する
(ISC_SOUND_STOP_REQ 送信)
15) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play with mute control＞

16) シンセサイザー・メロディ再生のための音符データの設定を行う
(ISC_SYNTHESIZER_MELODY_SET_REQ 送信)
17) シンセサイザー・メロディ再生を開始する
(ISC_SOUND_START_REQ 送信)
18) 再生開始 1 秒後にミュートを開始する
(ISC_AUDIO_MUTE_REQ 送信)
19) ミュート開始 1 秒後にミュートを解除する
(ISC_AUDIO_MUTE_REQ 送信)

20) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play with volume control＞

21) シンセサイザー・メロディ再生のための音符データの設定を行う
(ISC_SYNTHESIZER_MELODY_SET_REQ 送信)

22) シンセサイザー・メロディ再生を開始する
(ISC_SOUND_START_REQ 送信)

23) 再生開始直後にボリュームを徐々に下げ、また、徐々に上げる
(ISC_AUDIO_VOLUME_REQ 繰り返し送信)

24) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

3.1.3 main_synthesizer_buzztone.c

シンセサイザー・ブザートーン再生用のメインプログラムです。このプログラムでは、単純再生、再生中の強制停止、再生中のミュート制御、再生中のボリューム制御の処理を含んでいます。

1) システム(Host)の初期化を行う
2) I2C の初期化を行う
3) S1V30080 の初期化を行う
(SYSTEM_EN 端子に Low->High を入力)
4) S1V30080 との通信開始前に 1ms のウェイトを入れる
5) チェックサムの有効/無効を設定する
(ISC_CHKSUM_CONFIG_REQ 送信)
6) クロックの分周の設定を行う
(ISC_CLKDIV_CONFIG_REQ 送信)
7) サンプリング周波数やボリュームのサウンド出力の設定を行う
(ISC_AUDIO_CONFIG_REQ 送信)

＜play simple＞

8) シンセサイザー・ブザートーン再生の設定を行う
(ISC_SYNTHESIZER_BUZZ_TONE_SET_REQ 送信)
9) シンセサイザー・ブザートーン再生を開始する
(ISC_SOUND_START_REQ 送信)
10) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play and stop＞

11) シンセサイザー・ブザートーン再生の設定を行う
(ISC_SYNTHESIZER_BUZZ_TONE_SET_REQ 送信)
12) シンセサイザー・ブザートーン再生を開始する
(ISC_SOUND_START_REQ 送信)
13) 再生開始 1 秒後に再生を停止する
(ISC_SOUND_STOP_REQ 送信)
14) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play with mute control＞

15) シンセサイザー・ブザートーン再生の設定を行う
(ISC_SYNTHESIZER_BUZZ_TONE_SET_REQ 送信)
16) シンセサイザー・ブザートーン再生を開始する
(ISC_SOUND_START_REQ 送信)
17) 再生開始 1 秒後にミュートを開始する
(ISC_AUDIO_MUTE_REQ 送信)
18) ミュート開始 1 秒後にミュートを解除する
(ISC_AUDIO_MUTE_REQ 送信)
19) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play with volume control＞

20) シンセサイザー・ブザートーン再生の設定を行う
(ISC_SYNTHESIZER_BUZZ_TONE_SET_REQ 送信)

21) シンセサイザー・ブザートーン再生を開始する
(ISC_SOUND_START_REQ 送信)

22) 再生開始直後にボリュームを徐々に下げ、また、徐々に上げる
(ISC_AUDIO_VOLUME_REQ 繰り返し送信)

23) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

3.1.4 main_synthesizer_melodystreaming.c

シンセサイザー・メロディ再生(ストリーミング)用のメインプログラムです。このプログラムでは、単純再生、再生中の強制停止、再生中のミュート制御、再生中のボリューム制御の処理を含んでいます。

1) システム(Host)の初期化を行う
2) I2C の初期化を行う
3) S1V30080 の初期化を行う
(SYSTEM_EN 端子に Low->High を入力)
4) S1V30080 との通信開始前に 1ms のウェイトを入れる
5) チェックサムの有効/無効を設定する
(ISC_CHKSUM_CONFIG_REQ 送信)
6) クロックの分周の設定を行う
(ISC_CLKDIV_CONFIG_REQ 送信)
7) サンプリング周波数やボリュームのサウンド出力の設定を行う
(ISC_AUDIO_CONFIG_REQ 送信)
8) シンセサイザー・メロディ再生のためのテンポやエンベロープの設定を行う
(ISC_SYNTHESIZER_MELODY_CONFIG_REQ 送信)

＜play simple＞

9) シンセサイザー・メロディ再生用音符データを送信する
(ISC_SYNTHESIZER_MELODY_START_REQ 送信)
10) S1V30080 に対して次のメッセージが受付可能になるまで待機する
(MSG_RECEIVE 端子の出力信号が Low->High->Low に変化することを確認)
11) 次のシンセサイザー・メロディ再生用音符データを送信する
(ISC_SYNTHESIZER_MELODY_START_REQ 送信)
12) シンセサイザー・メロディ再生用音符データが終端に到達するまで、10)、11)を繰り返す
13) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play and stop＞

14) シンセサイザー・メロディ再生用音符データを送信する
(ISC_SYNTHESIZER_MELODY_START_REQ 送信)
15) S1V30080 に対して次のメッセージが受付可能になるまで待機する
(MSG_RECEIVE 端子の出力信号が Low->High->Low に変化することを確認)
16) 次のシンセサイザー・メロディ再生用音符データを送信する
(ISC_SYNTHESIZER_MELODY_START_REQ 送信)
17) シンセサイザー・メロディ再生用音符データが終端に到達するまで、15)、16)を繰り返す
18) 再生開始 1 秒後に再生を停止する
(ISC_SOUND_STOP_REQ 送信)
19) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play with mute control＞

20) シンセサイザー・メロディ再生用音符データを送信する
(ISC_SYNTHESIZER_MELODY_START_REQ 送信)

21) S1V30080 に対して次のメッセージが受付可能になるまで待機する
(MSG_RECEIVE 端子の出力信号が Low->High->Low に変化することを確認)

22) 次のシンセサイザー・メロディ再生用音符データを送信する
(ISC_SYNTHESIZER_MELODY_START_REQ 送信)

23) シンセサイザー・メロディ再生用音符データが終端に到達するまで、21)、22)を繰り返す

24) 再生開始 1 秒後にミュートを開始する
(ISC_AUDIO_MUTE_REQ 送信)

25) ミュート開始 1 秒後にミュートを解除する
(ISC_AUDIO_MUTE_REQ 送信)

26) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play with volume control＞

27) シンセサイザー・メロディ再生用音符データを送信する
(ISC_SYNTHESIZER_MELODY_START_REQ 送信)

28) S1V30080 に対して次のメッセージが受付可能になるまで待機する
(MSG_RECEIVE 端子の出力信号が Low->High->Low に変化することを確認)

29) 次のシンセサイザー・メロディ再生用音符データを送信する
(ISC_SYNTHESIZER_MELODY_START_REQ 送信)

30) シンセサイザー・メロディ再生用音符データが終端に到達するまで、28)、29)を繰り返す

31) 再生開始直後にボリュームを徐々に下げ、また、徐々に上げる
(ISC_AUDIO_VOLUME_REQ 繰り返し送信)

32) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

3.1.5 main_mixing_main_multi_melody.c

ミキシング再生用のメインプログラムです。このプログラムでは、オーディオ・メイン、オーディオ・マルチ、シンセサイザー・メロディの 3 系統の再生方式で、ミキシング再生を行います。

1) システム(Host)の初期化を行う
2) I2C の初期化を行う
3) S1V30080 の初期化を行う
(SYSTEM_EN 端子に Low->High を入力)
4) S1V30080 との通信開始前に 1ms のウェイトを入れる
5) チェックサムの有効/無効を設定する
(ISC_CHKSUM_CONFIG_REQ 送信)
6) クロックの分周の設定を行う
(ISC_CLKDIV_CONFIG_REQ 送信)
7) サンプリング周波数やボリュームのサウンド出力の設定を行う
(ISC_AUDIO_CONFIG_REQ 送信)
8) オーディオ・メイン再生の設定を行う
(ISC_SEQUENCER_MAIN_CONFIG_REQ 送信)
9) オーディオ・マルチ再生の設定を行う
(ISC_SEQUENCER_MULTI_CONFIG_REQ 送信)
10) シンセサイザー・メロディ再生用のテンポ及びエンベロープの設定を行う
(ISC_SYNTHESIZER_MELODY_CONFIG_REQ 送信)
11) シンセサイザー・メロディ再生用の音符データをセットする
(ISC_SYNTHESIZER_MELODY_SET_REQ 送信)

＜play synchronous-mixing＞

12) 全ての再生方式の再生開始フラグを Enable にし、ミキシング再生を開始する
(ISC_SOUND_START_REQ 送信)
13) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play asynchronous-mixing＞

14) オーディオ・メインの再生開始フラグのみ Enable にして再生を開始する
(ISC_SOUND_START_REQ 送信)
15) オーディオ・メイン再生開始 1 秒後に、オーディオ・マルチの再生開始フラグのみ Enable にして再生を開始する
(ISC_SOUND_START_REQ 送信)
16) オーディオ・マルチ再生開始 1 秒後に、シンセサイザー・メロディの再生開始フラグのみ Enable にして再生を開始する
(ISC_SOUND_START_REQ 送信)
17) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

※ 14)、15)、16)の各再生方式の再生開始タイミングの順番を入れ換えてもミキシング再生可能です。

3.1.6 main_mixing_main_multi_buzztone.c

ミキシング再生用のメインプログラムです。このプログラムでは、オーディオ・メイン、オーディオ・マルチ、シンセサイザー・ブザートーンの 3 系統の再生方式で、ミキシング再生を行います。

1) システム(Host)の初期化を行う
2) I2C の初期化を行う
3) S1V30080 の初期化を行う
(SYSTEM_EN 端子に Low->High を入力)
4) S1V30080 との通信開始前に 1ms のウェイトを入れる
5) チェックサムの有効/無効を設定する
(ISC_CHKSUM_CONFIG_REQ 送信)
6) クロックの分周の設定を行う
(ISC_CLKDIV_CONFIG_REQ 送信)
7) サンプリング周波数やボリュームのサウンド出力の設定を行う
(ISC_AUDIO_CONFIG_REQ 送信)
8) オーディオ・メイン再生の設定を行う
(ISC_SEQUENCER_MAIN_CONFIG_REQ 送信)
9) オーディオ・マルチ再生の設定を行う
(ISC_SEQUENCER_MULTI_CONFIG_REQ 送信)
10) シンセサイザー・ブザートーン再生の設定を行う
(ISC_SYNTHESIZER_BUZZ_TONE_SET_REQ 送信)

＜play synchronous-mixing＞

11) 全ての再生方式の再生開始フラグを Enable にし、ミキシング再生を開始する
(ISC_SOUND_START_REQ 送信)
12) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play asynchronous-mixing＞

13) オーディオ・メインの再生開始フラグのみ Enable にして再生を開始する
(ISC_SOUND_START_REQ 送信)
14) オーディオ・メイン再生開始 1 秒後に、オーディオ・マルチの再生開始フラグのみ Enable にして再生を開始する
(ISC_SOUND_START_REQ 送信)
15) オーディオ・マルチ再生開始 1 秒後に、シンセサイザー・ブザートーンの再生開始フラグのみ Enable にして再生を開始する
(ISC_SOUND_START_REQ 送信)
16) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

※ 13)、14)、15)の各再生方式の再生開始タイミングの順番を入れ換えてもミキシング再生可能です。

3.1.7 main_mixing_main_multi_melodystreaming.c

ミキシング再生用のメインプログラムです。このプログラムでは、オーディオ・メイン、オーディオ・マルチ、シンセサイザー・メロディ(ストリーミング)の 3 系統の再生方式で、ミキシング再生を行います。

1) システム(Host)の初期化を行う
2) I2C の初期化を行う
3) S1V30080 の初期化を行う
(SYSTEM_EN 端子に Low->High を入力)
4) S1V30080 との通信開始前に 1ms のウェイトを入れる
5) チェックサムの有効/無効を設定する
(ISC_CHKSUM_CONFIG_REQ 送信)
6) クロックの分周の設定を行う
(ISC_CLKDIV_CONFIG_REQ 送信)
7) サンプリング周波数やボリュームのサウンド出力の設定を行う
(ISC_AUDIO_CONFIG_REQ 送信)
8) オーディオ・メイン再生の設定を行う
(ISC_SEQUENCER_MAIN_CONFIG_REQ 送信)
9) オーディオ・マルチ再生の設定を行う
(ISC_SEQUENCER_MULTI_CONFIG_REQ 送信)
10) シンセサイザー・メロディ再生用のテンポやエンベロープの設定を行う
(ISC_SYNTHESIZER_MELODY_CONFIG_REQ 送信)

＜play mixing＞

11) 最初の音符データを送信し、シンセサイザー・メロディの再生を開始する
(ISC_SYNTHESIZER_MELODY_START_REQ 送信)
12) S1V30080 に対して次のメッセージが受付可能になるまで待機する
(MSG_RECEIVE 端子の出力信号が Low->High->Low に変化することを確認)
13) 次のシンセサイザー・メロディ再生用音符データを送信する
(ISC_SYNTHESIZER_MELODY_START_REQ 送信)
14) シンセサイザー・メロディ再生用音符データが終端に到達するまで、12)、13)を繰り返す
15) シンセサイザー・メロディ再生開始 1 秒後に、オーディオ・メインの再生開始フラグのみ Enable にして再生を開始する
(ISC_SOUND_START_REQ 送信)
16) オーディオ・メイン再生開始 1 秒後に、オーディオ・マルチの再生開始フラグのみ Enable にして再生を開始する
(ISC_SOUND_START_REQ 送信)
17) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

※ 上記フローでは、ストリーミングによるシンセサイザー・メロディ再生中に、オーディオ再生を開始していますが、オーディオ再生中に、ストリーミングによるシンセサイザー・メロディ再生を開始することも可能です。

3.1.8 main_easy_start.c

ミキシング再生用のメインプログラムです。このプログラムでは、ISC_EASY_START_REQ メッセージを使用してミキシング再生を行います。

※ このプログラムを使用する場合は、音声 IC の内蔵 ROM、もしくは、外部フラッシュメモリにメモリ格納コマンドが含まれている必要があります。

1) システム(Host)の初期化を行う
2) I2C の初期化を行う
3) S1V30080 の初期化を行う
(SYSTEM_EN 端子に Low->High を入力)
4) S1V30080 との通信開始前に 1ms のウェイトを入れる

＜play simple＞

5) メモリ格納コマンドのインデックス番号 0 を指定して再生を開始する
(ISC_EASY_START_REQ 送信)
6) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play and stop＞

7) メモリ格納コマンドのインデックス番号 0 を指定して再生を開始する
(ISC_EASY_START_REQ 送信)
8) 再生開始 1 秒後に再生を停止する
(ISC_SOUND_STOP_REQ 送信)
9) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play with mute control＞

10) メモリ格納コマンドのインデックス番号 0 を指定して再生を開始する
(ISC_EASY_START_REQ 送信)
11) 再生開始 1 秒後にミュートを開始する
(ISC_AUDIO_MUTE_REQ 送信)
12) ミュート開始 1 秒後にミュートを解除する
(ISC_AUDIO_MUTE_REQ 送信)
13) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play with volume control＞

14) メモリ格納コマンドのインデックス番号 0 を指定して再生を開始する
(ISC_EASY_START_REQ 送信)
15) 再生開始直後にボリュームを徐々に下げ、また、徐々に上げる
(ISC_AUDIO_VOLUME_REQ 繰り返し送信)
16) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

＜play mixing＞

17) メモリ格納コマンドのインデックス番号 0 を指定して再生を開始します
(ISC_EASY_START_REQ 送信)

18) 17)の再生開始 1 秒後、メモリ格納コマンドのインデックス番号 1 を指定して再生を開始します
(ISC_EASY_START_REQ 送信)

19) 18)の再生開始 1 秒後、メモリ格納コマンドのインデックス番号 2 を指定して再生を開始します
(ISC_EASY_START_REQ 送信)

20) 再生が終了するまで待機する
(SOUND_PLAYING 端子の出力信号が High->Low に変化することを確認)

※ 本メインプログラムを使用してミキシング再生を行う場合、コマンドインデックス 0, 1, 2 で指定されるメモリ格納コマンドが、それぞれ異なる再生方式(オーディオ・メイン再生、オーディオ・マルチ再生、シンセサイザー再生)である必要があります。同じ再生方式のメモリ格納コマンドによるミキシング再生は、仕様上できませんので、ご注意ください。ミキシング再生に関する詳細については、「S1V30080 シリーズ メッセージプロトコル仕様書」をご参照ください。

3.2 S1V30080 制御用API関数仕様

S1V30080 を制御するための API 関数仕様を以下に記載します。

※ 以下に記載する API 関数は、弊社にて S1V30080 制御評価で使用したホストシステム上の I2C を例にした制御プログラムです。お客様のシステムに組み込む際には、お客様のシステムの仕様に合わせて API 関数内部を変更いただく必要があります。

3.2.1 I2C_Initialize

［構文］

void I2C_Initialize (void)

［機能］

I2C の初期設定を行います。

［入力引数］

なし

［出力引数］

なし

［戻り値］

なし

［機能説明］

I2C の各種レジスタの初期化を行います。本関数では、弊社評価システムで使用したホストプロセッサの I2C 仕様に準拠しており、下記の設定を行っています。

(1) I2C を無効にする。

(2) I2C の割り込み設定を無効にする。

(3) I2C のポート入出力方向を設定する。

(4) I2C のボーレートを設定する。

(5) 転送モードを MSB ファーストに設定する。

(6) I2C を有効にする。

3.2.2 I2C_SendMessage

［構文］

```
int         I2C_SendMessage (
                unsigned char   *pucSendMessage,
                int             iCheckSumFlag )
```

［機能］

S1V30080 へメッセージを送信します。

［入力引数］

pucSendMessage	送信メッセージが格納されたメモリ領域のアドレスを指定します。
iCheckSumFlag	チェックサムの有効/無効を指定します。 (0：無効 / 1：有効)

［出力引数］

なし

［戻り値］

正常終了した場合は 0 を返し、それ以外は以下に示すエラーコードを返します。

エラーコード	値	説明
I2CERR_SUCCESS	0	正常終了した場合に返します
I2CERR_NULL_PTR	-1	引数が NULL ポインタの場合に返します

［機能説明］

S1V30080 に対して REQ メッセージを送信します。本関数は、送信対象となる REQ メッセージ中の length フィールドの値を参照してデータを送信します。
本関数の内部では、メッセージの送信後に、S1V30080 の MESSAGE_RECEIVE 端子から出力される信号の Low->High の変化を確認しています。ただし、以下に挙げるメッセージについては、ストリーミングによるメロディ再生時のメッセージプロトコル仕様により、信号の変化を確認せずに関数を終了します。

- ISC_SYNTHESIZER_MELODY_START_REQ
- ISC_AUDIO_MUTE_REQ
- ISC_AUDIO_VOLUME_REQ

3.2.3 I2C_ReceiveMessage

［構文］

int I2C_ReceiveMessage (
unsigned char aucReceivedMessage[])

［機能］

S1V30080 からのメッセージを受信します。

［入力引数］

なし

［出力引数］

aucReceivedMessage 受信メッセージを格納する配列を指定します。

［戻り値］

正常終了した場合は 0 を返し、それ以外は以下に示すエラーコードを返します。

エラーコード	値	説明
I2CERR_SUCCESS	0	正常終了した場合に返します
I2CERR_NULL_PTR	-1	引数が NULL ポインタの場合に返します
I2CERR_TIMEOUT	1	設定したタイムアウト時間内に、S1V30080 から通知されるメッセージが受信されない場合に返します

［機能説明］

S1V30080 から通知される RESP メッセージを受信します。メッセージ開始コマンド(0xAA)が検出されるまで、S1V30080 からのデータを受信し続けます。メッセージ開始コマンド検出後、受信データに含まれる length フィールドの値を参照してデータの受信を続けます。
出力引数である aucReceivedMessage には、受信したメッセージのデータ列が格納されます。

3.2.4 I2C_SendByte

［構文］

```
void        I2C_SendByte (
                        unsigned char       ucSendData )
```

［機能］

1 バイトのデータを送信します。

［入力引数］

ucSendData　　1 バイトの送信データを指定します。

［出力引数］

なし

［戻り値］

なし

［機能説明］

S1V30080 に対して I2C 経由で 1 バイトのデータを送信します。

3.2.5 I2C_ReceiveByte

［構文］

```
unsigned char   I2C_ReceiveByte (
                    char            cAck )
```

［機能］

1 バイトのデータを受信します。

［入力引数］

cAck	スレーブデバイスである S1V30080 に返す Ack 情報を指定します。(0：Ack / 1：Nack)

［出力引数］

なし

［戻り値］

1 バイトの受信データを返します。

［機能説明］

S1V30080 から I2C 経由で 1 バイトのデータを受信します。

3.2.6 I2C_SetStartCondition

［構文］

```
char    _I2cSetStartCondition (
                char            cAddressMode,
                unsigned short  usAddress)
```

［機能］

スタート条件を設定します。

［入力引数］

cAddressMode	スレーブアドレスモードを指定します。7 を指定してください。
usAddress	スレーブアドレスを指定します。01101100’b(0x6C)を指定してください。

［出力引数］

なし

［戻り値］

S1V30080 から受信した Ack 情報を返します。(0：Ack / 1：Nack)

［機能説明］

スタート条件の波形を出力し、スレーブアドレスを送信します。その後、スレーブデバイスである S1V30080 の Ack 情報を受信し、戻り値として返します。

3.2.7 I2C_SetStopCondition

［構文］

void I2C_SetStartCondition (void)

［機能］

ストップ条件を設定します。

［入力引数］

なし

［出力引数］

なし

［戻り値］

なし

［機能説明］

ストップ条件の波形を出力します。

3.2.8 S080_system_en

［構文］

void S080_system_en (int iValue)

［機能］

S1V30080 の SYSTEM_EN 端子への入力信号を制御します。

［入力引数］

iValue	SYSTEM_EN 端子に入力する信号を指定します。 (0：Low / 1：High)

［出力引数］

なし

［戻り値］

なし

［機能説明］

S1V30080 へのリセットを制御するために、SYSTEM_EN 端子への入力信号を制御します。S1V30080 へのリセットは、S1V30080 の SYSTEM_EN 端子に Low->High->Low を入力して行います。
本関数を使用する場合は、S1V30080 の SYSTEM_EN 端子に、ホストプロセッサ上の GPIO 端子を接続して信号を制御する必要があります。

3.2.9 S080_sound_playing

［構文］

void S080_sound_playing (void)

［機能］

S1V30080 の SOUND_PLAYING 端子からの出力信号の状態を取得します。

［入力引数］

なし

［出力引数］

なし

［戻り値］

音声出力の状態を返します。(0: Low / 1: High)

［機能説明］

S1V30080 から音声出力の状態を知るために、SOUND_PLAYING 端子から出力される信号の状態を取得します。音声が出力されている場合は High、音声が出力されていない場合は Low を返します。
本関数を使用する場合は、S1V30080 の SOUND_PLAYING 端子に、ホストプロセッサ上の GPIO 端子を接続して信号を制御する必要があります。

3.2.10 S080_message_receive

［構文］

void S080_message_receive

［機能］

MESSAGE_RECEIVE 端子からの出力信号の Low から High への変化を検出します。

［入力引数］

なし

［出力引数］

なし

［戻り値］

出力信号の Low から High への変化の状態を返します。
(0: 変化なし / 1: 変化あり)

［機能説明］

S1V30080 の MESSAGE_RECEIVE 端子からの出力信号が Low から High に変化したかどうかを検出します。検出した場合は 1 を、検出しない場合は 0 を返します。
本関数では、信号の変化を割り込み処理により検出しています。そのため、本関数を使用する場合は、S1V30080 の SOUND_PLAYING 端子に、ホストプロセッサ上の GPIO 端子を接続し、GPIO 端子に割り込みを適用して制御する必要があります。

3.2.11 その他の関数について

本サンプルプログラムでは、上記で説明した関数以外にも、以下に示す関数が使用されています。

- _i2cSendDataAsync　スレーブへデータを送信する
- _i2cReceiveAck　スレーブから Ack/Nack を受信する
- _i2cSendAck　スレーブへ Ack/Nack を送信する
- _i2cCreateStart　スタート条件信号を生成する
- _i2cCreateStop　ストップ条件信号を生成する

※ 上記関数は､弊社の S1V30080 制御評価用に使用したホストシステムに搭載される 2 チャンネルの双方向シリアル I/F(DCSIO)を使用して I2C I/F を実現するために用意したソフトウェア制御用関数です。お客様のホストシステムに組み込む場合には、お客様システムの I2C I/F 仕様に合わせて変更いただく必要があります。

改訂履歴表

年月日	改訂内容			
	Rev.	頁	種別	内　容
09/02/16	1.00	全頁	新規	新規作成

セイコーエプソン株式会社

半導体事業部　IC 営業部

<IC 国内営業グループ>

東京　〒191-8501　東京都日野市日野 421-8

TEL(042)587-5313(直通)　　FAX(042)587-5116

大阪　〒541-0059　大阪市中央区博労町 3-5-1　エプソン大阪ビル 15F

TEL(06)6120-6000(代表)　　FAX(06)6120-6100

ドキュメントコード：411657500

2009 年 2 月 作成ⓓ