EPSON
EXCEED YOUR VISION

S1V30080 Series
I2C Interface
Sample Program Specifications

SEIKO EPSON CORPORATION

Rev.1.00

NOTICE

No part of this material may be reproduced or duplicated in any form or by any means without the written permission of Seiko Epson. Seiko Epson reserves the right to make changes to this material without notice. Seiko Epson does not assume any liability of any kind arising out of any inaccuracies contained in this material or due to its application or use in any product or circuit and, further, there is no representation that this material is applicable to products requiring high level reliability, such as, medical products. Moreover, no license to any intellectual property rights is granted by implication or otherwise, and there is no representation or warranty that anything made in accordance with this material will be free from any patent or copyright infringement of a third party. This material or portions thereof may contain technology or the subject relating to strategic products under the control of the Foreign Exchange and Foreign Trade Law of Japan and may require an export license from the Ministry of Economy, Trade and Industry or other approval from another government agency.

All other product names mentioned herein are trademarks and/or registered trademarks of their respective companies.

©SEIKO EPSON CORPORATION 2009, All rights reserved.

Table of Contents

1. Overview

This document describes the sample programs provided to clients using the S1V30080. These sample programs are samples for controlling the S1V30080 on the client host system. Also described are specifications for the API functions used within the sample programs.

This document should be used alongside the *S1V30080 Series Message Protocol Specifications*.

* The sample programs were created for the host system used by Seiko Epson for S1V30080 control evaluations. Seiko Epson makes no compatibility guarantees with respect to client systems.

2. File List

All source files for the sample programs are located in the following directory.

\i2c_src (source files)

2.1 Main program files

The main program files include the set of control programs for audio playback processing, synthesizer playback processing, and audio playback and synthesizer playback mixing processing controlled by the host processor.

Table 2.1 lists the main program files. These files are main programs, which include main functions. Select a single main program file for use.

Table 2.1 Main program file list

Filename	Description
main_audio_main.c	Main program file for audio-main playback
main_audio_multi.c	Main program file for audio-multi playback
main_synthesizer_melody.c	Main program file for synthesizer melody playback
main_synthesizer_buzztone.c	Main program file for synthesizer buzzer tone playback
main_synthesizer_melodystreaming.c	Main program file for synthesizer melody (streaming) playback
main_mixing_main_multi_melody.c	Main program file for mixing playback (mixing by the following three playback systems: audio-main, audio-multi, and synthesizer melody)
main_mixing_main_multi_buzztone.c	Main program file for mixing playback (mixing by the following three playback systems: audio-main, audio-multi, and synthesizer buzzer tone)
main_mixing_main_multi_melodystreaming.c	Main program file for mixing playback (mixing by the following three playback systems: audio-main, audio-multi, and synthesizer melody streaming playback)
main_mixing_easy_start.c	Main program for mixing playback (mixing using ISC_EASY_START_REQ)

2.2 S1V30080 control API function definition files

The S1V30080 control API function definition files define API functions used to control the S1V30080.

Table 2.2 lists API function definition files for controlling the S1V30080.

For detailed information on API function specifications, refer to "3.2 S1V30080 control API function specifications."

Table 2.2 S1V30080 control API source file list

Filename	Description
i2c_api.c	Source file defining API functions for serial communications control
i2c_api.h	Header file declaring API functions for serial communications control
gpio.c	Source file defining API functions for device control not using serial communications
gpio.h	Header file declaring API functions for device control not using serial communications

* "i2c_api.c" is a control program for I2C on the host system used by Seiko Epson for S1V30080 control evaluations. Modifications must be made to suit client system specifications before incorporation into the client system.

2.3 Message file

The message file defines an array table for REQ messages.

Table 2.3 shows the message file.

Table 2.3 Message file

Filename	Description
isc_msgs.c	Source file defining a data array for REQ messages for sample programs

The message file defines the data shown in Table 2.4. It describes the REQ message data sequence, including message start padding and start command 0x00 and 0xAA.

Table 2.4 Sample message file

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

2.4 Other source files

Table 2.5 lists source files other than those already described above.

Table 2.5 List of other source files

Filename	Description
isc_msgs.h	Header file defining the lengths and IDs of S1V30080 messages
reg_host.h	Header file defining the register map of the host system used by Seiko Epson

Note: “reg_host.h” describes the register map on the host system used by Seiko Epson for S1V30080 control evaluations. Modifications must be made to suit client host system specifications before incorporation into the client system.

3. Sample Program Specifications

The following describes the specifications for the API functions used by the sample programs and details of the main programs.

3.1 Main programs

Nine main programs are provided as I2C-compatible sample programs. Each main program provides control functions in the following sequence.

3.1.1 main_audio_main.c, main_audio_multi.c

This is the main program for audio-main playback and audio-multi playback. The program includes simple playback processing, forced stop processing during playback, mute control processing during playback, and volume control processing during playback.

1) Initializes the system (host).
2) Initializes the I2C.
3) Initializes the S1V30080.
   (Inputs Low → High to SYSTEM_EN pin)
4) Inserts a wait time of 1 ms before initiating communications with the S1V30080.
5) Enables/disables checksum verification.
   (Sends ISC_CHKSUM_CONFIG_REQ)
6) Sets the clock frequency division.
   (Sends ISC_CLKDIV_CONFIG_REQ)
7) Sets the sample frequency and sound output for volume.
   (Sends ISC_AUDIO_CONFIG_REQ)

< play simple >

8) Sets audio playback.
   (Sends ISC_SEQUENCER_(MAIN/MULTI)_CONFIG_REQ)
9) Starts audio playback.
   (Sends ISC_SOUND_START_REQ)
10) Waits for playback to end.
    Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play and stop >

11) Sets audio playback.
    (Sends ISC_SEQUENCER_(MAIN/MULTI)_CONFIG_REQ)
12) Starts audio playback.
    (Sends ISC_SOUND_START_REQ)
13) Stops playback 1 second after playback starts.
    (Sends ISC_SOUND_STOP_REQ)
14) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play with mute control >

15) Set audio playback.
(Sends ISC_SEQUENCER_(MAIN/MULTI)_CONFIG_REQ)

16) Starts audio playback.
(Sends ISC_SOUND_START_REQ)

17) Activates mute 1 second after playback starts.
(Sends ISC_AUDIO_MUTE_REQ)

18) Cancels mute 1 second after mute activation.
(Sends ISC_AUDIO_MUTE_REQ)

19) Waits for playback to end.
(Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play with volume control >

20) Sets audio playback.
(Sends ISC_SEQUENCER_(MAIN/MULTI)_CONFIG_REQ)

21) Starts audio playback.
(Sends ISC_SOUND_START_REQ)

22) Gradually reduces volume immediately after playback starts, then gradually increases volume.
(Repeatedly sends ISC_AUDIO_VOLUME_REQ)

23) Waits for playback to end.
(Confirms the change in SOUND_PLAYING pin output signal from High to Low)

3.1.2 main_synthesizer_melody.c

This is the main program for synthesizer melody playback. The program includes simple playback processing, forced stop processing during playback, mute control processing during playback, and volume control processing during playback.

1) Initializes the system (host).
2) Initializes the I2C.
3) Initializes the S1V30080.
   (Inputs Low → High to SYSTEM_EN pin)
4) Inserts a wait time of 1 ms before initiating communications with the S1V30080.
5) Enables/disables checksum verification.
   (Sends ISC_CHKSUM_CONFIG_REQ)
6) Sets the clock frequency division.
   (Sends ISC_CLKDIV_CONFIG_REQ)
7) Sets the sample frequency and sound output for volume.
   (Sends ISC_AUDIO_CONFIG_REQ)
8) Sets the tempo and envelope for synthesizer melody playback.
   (Sends ISC_SYNTHESIZER_MELODY_CONFIG_REQ)

< play simple >

9) Sets musical note data for synthesizer melody playback.
   (Sends ISC_SYNTHESIZER_MELODY_SET_REQ)
10) Starts synthesizer melody playback.
    (Sends ISC_SOUND_START_REQ)
11) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play and stop >

12) Sets musical note data for synthesizer melody playback.
    (Sends ISC_SYNTHESIZER_MELODY_SET_REQ)
13) Starts synthesizer melody playback.
    (Sends ISC_SOUND_START_REQ)
14) Stops playback 1 second after playback starts.
    (Sends ISC_SOUND_STOP_REQ)
15) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play with mute control >

16) Sets musical note data for synthesizer melody playback.
    (Sends ISC_SYNTHESIZER_MELODY_SET_REQ)
17) Starts synthesizer melody playback.
    (Sends ISC_SOUND_START_REQ)
18) Activates mute 1 second after playback starts.
    (Sends ISC_AUDIO_MUTE_REQ)
19) Cancels mute 1 second after mute activation.
    (Sends ISC_AUDIO_MUTE_REQ)
20) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play with volume control >

21) Sets musical note data for synthesizer melody playback.
(Sends ISC_SYNTHESIZER_MELODY_SET_REQ)

22) Starts synthesizer melody playback.
(Sends ISC_SOUND_START_REQ)

23) Gradually reduces volume immediately after playback starts, then gradually increases volume.
(Repeatedly sends ISC_AUDIO_VOLUME_REQ)

24) Waits for playback to end.
(Confirms the change in SOUND_PLAYING pin output signal from High to Low)

3.1.3 main_synthesizer_buzztone.c

This is the main program for synthesizer buzzer tone playback. The program includes simple playback processing, forced stop processing during playback, mute control processing during playback, and volume control processing during playback.

1) Initializes the system (host).
2) Initializes the I2C.
3) Initializes the S1V30080.
   (Inputs Low → High to SYSTEM_EN pin)
4) Inserts a wait time of 1 ms before initiating communications with the S1V30080.
5) Enables/disables checksum verification.
   (Sends ISC_CHKSUM_CONFIG_REQ)
6) Sets the clock frequency division.
   (Sends ISC_CLKDIV_CONFIG_REQ)
7) Sets the sample frequency and sound output for volume.
   (Sends ISC_AUDIO_CONFIG_REQ)

< play simple >

8) Sets synthesizer buzzer tone playback.
   (Sends ISC_SYNTHESIZER_BUZZ_TONE_SET_REQ)
9) Starts synthesizer buzzer tone playback.
   (Sends ISC_SOUND_START_REQ)
10) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play and stop >

11) Sets synthesizer buzzer tone playback.
    (Sends ISC_SYNTHESIZER_BUZZ_TONE_SET_REQ)
12) Starts synthesizer buzzer tone playback.
    (Sends ISC_SOUND_START_REQ)
13) Stops playback 1 second after playback starts.
    (Sends ISC_SOUND_STOP_REQ)
14) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play with mute control >

15) Sets synthesizer buzzer tone playback.
    (Sends ISC_SYNTHESIZER_BUZZ_TONE_SET_REQ)
16) Starts synthesizer buzzer tone playback.
    (Sends ISC_SOUND_START_REQ)
17) Activates mute 1 second after playback starts.
    (Sends ISC_AUDIO_MUTE_REQ)
18) Cancels mute 1 second after mute activation.
    (Sends ISC_AUDIO_MUTE_REQ)
19) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play with volume control >

20) Sets synthesizer buzzer tone playback.
(Sends ISC_SYNTHESIZER_BUZZ_TONE_SET_REQ)

21) Starts synthesizer buzzer tone playback.
(Sends ISC_SOUND_START_REQ)

22) Gradually reduces volume immediately after playback starts, then gradually increases volume.
(Repeatedly sends ISC_AUDIO_VOLUME_REQ)

23) Waits for playback to end.
(Confirms the change in SOUND_PLAYING pin output signal from High to Low)

3.1.4 main_synthesizer_melodystreaming.c

This is the main program for synthesizer melody (streaming) playback. The program includes simple playback processing, forced stop processing during playback, mute control processing during playback, and volume control processing during playback.

1) Initializes the system (host).
2) Initializes the I2C.
3) Initializes the S1V30080.
   (Inputs Low → High to SYSTEM_EN pin)
4) Inserts a wait time of 1 ms before initiating communications with the S1V30080.
5) Enables/disables checksum verification.
   (Sends ISC_CHKSUM_CONFIG_REQ)
6) Sets the clock frequency division.
   (Sends ISC_CLKDIV_CONFIG_REQ)
7) Sets the sample frequency and sound output for volume.
   (Sends ISC_AUDIO_CONFIG_REQ)
8) Sets the tempo and envelope for synthesizer melody playback.
   (Sends ISC_SYNTHESIZER_MELODY_CONFIG_REQ)

< play simple >

9) Sends musical note data for synthesizer melody playback.
   (Sends ISC_SYNTHESIZER_MELODY_START_REQ)
10) Waits until the S1V30080 is ready to receive the next message.
    (Confirms the change in MSG_RECEIVE pin output signal from Low to High then back to Low)
11) Sends musical note data for the next synthesizer melody playback.
    (Sends ISC_SYNTHESIZER_MELODY_START_REQ)
12) Repeats 10) and 11) until reaching the end of the musical note data for synthesizer melody playback.
13) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play and stop >

14) Sends musical note data for synthesizer melody playback.
    (Sends ISC_SYNTHESIZER_MELODY_START_REQ)
15) Waits until the S1V30080 is ready to receive the next message.
    (Confirms the change in MSG_RECEIVE pin output signal from Low to High then back to Low)
16) Sends musical note data for the next synthesizer melody playback.
    (Sends ISC_SYNTHESIZER_MELODY_START_REQ)
17) Repeats 15) and 16) until reaching the end of the musical note data for synthesizer melody playback.
18) Stops playback 1 second after playback starts.
    (Sends ISC_SOUND_STOP_REQ)
19) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play with mute control >

20) Sends musical note data for synthesizer melody playback.
(Sends ISC_SYNTHESIZER_MELODY_START_REQ)

21) Waits until the S1V30080 is ready to receive the next message.
(Confirms the change in MSG_RECEIVE pin output signal from Low to High then back to Low)

22) Sends musical note data for the next synthesizer melody playback.
(Sends ISC_SYNTHESIZER_MELODY_START_REQ)

23) Repeats 21) and 22) until reaching the end of the musical note data for synthesizer melody playback.

24) Activates mute 1 second after playback starts.
(Sends ISC_AUDIO_MUTE_REQ)

25) Cancels mute 1 second after mute activation.
(Sends ISC_AUDIO_MUTE_REQ)

26) Waits for playback to end.
(Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play with volume control >

27) Sends musical note data for synthesizer melody playback.
(Sends ISC_SYNTHESIZER_MELODY_START_REQ)

28) Waits until the S1V30080 is ready to receive the next message.
(Confirms the change in MSG_RECEIVE pin output signal from Low to High then back to Low)

29) Sends musical note data for the next synthesizer melody playback.
(Sends ISC_SYNTHESIZER_MELODY_START_REQ)

30) Repeats 28) and 29) until reaching the end of the musical note data for synthesizer melody playback.

31) Gradually reduces volume immediately after playback starts, then gradually increases volume.
(Repeatedly sends ISC_AUDIO_VOLUME_REQ)

32) Waits for playback to end.
(Confirms the change in SOUND_PLAYING pin output signal from High to Low)

3.1.5 main_mixing_main_multi_melody.c

This is the main program for mixing playback. This program performs mixing playback using the three playback systems: audio-main, audio-multi, and synthesizer melody.

1) Initializes the system (host).
2) Initializes the I2C.
3) Initializes the S1V30080.
   (Inputs Low → High to SYSTEM_EN pin)
4) Inserts a wait time of 1 ms before initiating communications with the S1V30080.
5) Enables/disables checksum verification.
   (Sends ISC_CHKSUM_CONFIG_REQ)
6) Sets the clock frequency division.
   (Sends ISC_CLKDIV_CONFIG_REQ)
7) Sets the sample frequency and sound output for volume.
   (Sends ISC_AUDIO_CONFIG_REQ)
8) Sets audio-main playback.
   (Sends ISC_SEQUENCER_MAIN_CONFIG_REQ)
9) Sets audio-multi playback.
   (Sends ISC_SEQUENCER_MULTI_CONFIG_REQ)
10) Sets the tempo and envelope for synthesizer melody playback.
    (Sends ISC_SYNTHESIZER_MELODY_CONFIG_REQ)
11) Sets musical note data for synthesizer melody playback.
    (Sends ISC_SYNTHESIZER_MELODY_SET_REQ)

< play synchronous-mixing >

12) Enables playback start flags for all playback systems and starts mixing playback.
    (Sends ISC_SOUND_START_REQ)
13) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play asynchronous-mixing >

14) Enables only the audio-main playback start flag and starts playback.
    (Sends ISC_SOUND_START_REQ)
15) Enables only the audio-multi playback start flag 1 second after the start of audio-main playback, then starts playback.
    (Sends ISC_SOUND_START_REQ)
16) Enables only the synthesizer melody playback start flag 1 second after the start of audio-multi playback, then starts playback.
    (Sends ISC_SOUND_START_REQ)
17) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

* Mixing playback is also possible even if the playback sequence of the three playback systems indicated in 14), 15), and 16) is modified.

3.1.6 main_mixing_main_multi_buzztone.c

This is the main program for mixing playback. This program performs mixing playback using the three playback systems: audio-main, audio-multi, and synthesizer buzzer tone.

1) Initializes the system (host).
2) Initializes the I2C.
3) Initializes the S1V30080.
   (Inputs Low → High to SYSTEM_EN pin)
4) Inserts a wait time of 1 ms before initiating communications with the S1V30080.
5) Enables/disables checksum verification.
   (Sends ISC_CHKSUM_CONFIG_REQ)
6) Sets the clock frequency division.
   (Sends ISC_CLKDIV_CONFIG_REQ)
7) Sets the sample frequency and sound output for volume.
   (Sends ISC_AUDIO_CONFIG_REQ)
8) Sets audio-main playback.
   (Sends ISC_SEQUENCER_MAIN_CONFIG_REQ)
9) Sets audio-multi playback.
   (Sends ISC_SEQUENCER_MULTI_CONFIG_REQ)
10) Sets synthesizer buzzer tone playback.
    (Sends ISC_SYNTHESIZER_BUZZ_TONE_SET_REQ)

< play synchronous-mixing >

11) Enables playback start flags for all playback systems and starts mixing playback.
    (Sends ISC_SOUND_START_REQ)
12) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play asynchronous-mixing >

13) Enables only the audio-main playback start flag and starts playback.
    (Sends ISC_SOUND_START_REQ)
14) Enables only the audio-multi playback start flag 1 second after the start of audio-main playback, then starts playback.
    (Sends ISC_SOUND_START_REQ)
15) Enables only the synthesizer buzzer tone playback start flag 1 second after the start of audio-multi playback, then starts playback.
    (Sends ISC_SOUND_START_REQ)
16) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

* Mixing playback is also possible even if the playback sequence of the three playback systems indicated in 13), 14), and 15) is modified.

3.1.7 main_mixing_main_multi_melodystreaming.c

This is the main program for mixing playback. This program performs mixing playback using the three playback systems: audio-main, audio-multi, and synthesizer melody (streaming).

1) Initializes the system (host).
2) Initializes the I2C.
3) Initializes the S1V30080.
   (Inputs Low → High to SYSTEM_EN pin)
4) Inserts a wait time of 1 ms before initiating communications with the S1V30080.
5) Enables/disables checksum verification.
   (Sends ISC_CHKSUM_CONFIG_REQ)
6) Sets the clock frequency division.
   (Sends ISC_CLKDIV_CONFIG_REQ)
7) Sets the sample frequency and sound output for volume.
   (Sends ISC_AUDIO_CONFIG_REQ)
8) Sets audio-main playback.
   (Sends ISC_SEQUENCER_MAIN_CONFIG_REQ)
9) Sets audio-multi playback.
   (Sends ISC_SEQUENCER_MULTI_CONFIG_REQ)
10) Sets the tempo and envelope for synthesizer melody playback.
    (Sends ISC_SYNTHESIZER_MELODY_CONFIG_REQ)

< play mixing >

11) Sends the first musical tone data and starts synthesizer melody playback.
    (Sends ISC_SYNTHESIZER_MELODY_START_REQ)
12) Waits until the S1V30080 is ready to receive the next message.
    (Confirms the change in MSG_RECEIVE pin output signal from Low to High then back to Low)
13) Sends musical note data for the next synthesizer melody playback.
    (Sends ISC_SYNTHESIZER_MELODY_START_REQ)
14) Repeats 12) and 13) until reaching the end of the musical note data for synthesizer melody playback.
15) Enables only the audio-main playback start flag 1 second after the start of synthesizer melody playback, then starts playback.
    (Sends ISC_SOUND_START_REQ)
16) Enables only the audio-multi playback start flag 1 second after the start of audio-main playback, then starts playback.
    (Sends ISC_SOUND_START_REQ)
17) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

* Although audio playback in the above process flow starts in the middle of synthesizer melody streaming playback, it is also possible to start synthesizer melody streaming playback during audio playback.

3.1.8 main_easy_start.c

This is the main program for mixing playback. This program performs mixing playback using the ISC_EASY_START_REQ message.

* To run this program, the voice IC's built-in ROM or external flash memory must contain a memory storage command.

1) Initializes the system (host).
2) Initializes the I2C.
3) Initializes the S1V30080.
   (Inputs Low → High to SYSTEM_EN pin)
4) Inserts a wait time of 1 ms before initiating communications with the S1V30080.

< play simple >

5) Specifies Index No. 0 for the memory storage command, then starts playback.
   (Sends ISC_EASY_START_REQ)
6) Waits for playback to end.
   (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play and stop >

7) Specifies Index No. 0 for the memory storage command, then starts playback.
   (Sends ISC_EASY_START_REQ)
8) Stops playback 1 second after playback starts.
   (Sends ISC_SOUND_STOP_REQ)
9) Waits for playback to end.
   (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play with mute control >

10) Specifies Index No. 0 for the memory storage command, then starts playback.
    (Sends ISC_EASY_START_REQ)
11) Activates mute 1 second after playback starts.
    (Sends ISC_AUDIO_MUTE_REQ)
12) Cancels mute 1 second after mute activation.
    (Sends ISC_AUDIO_MUTE_REQ)
13) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play with volume control >

14) Specifies Index No. 0 for the memory storage command, then starts playback.
    (Sends ISC_EASY_START_REQ)
15) Gradually reduces volume immediately after playback starts, then gradually increases volume.
    (Repeatedly sends ISC_AUDIO_VOLUME_REQ)
16) Waits for playback to end.
    (Confirms the change in SOUND_PLAYING pin output signal from High to Low)

< play mixing >

17) Specifies Index No. 0 for the memory storage command, then starts playback.
(Sends ISC_EASY_START_REQ)

18) Specifies Index No. 1 for the memory storage command and starts playback 1 second after playback starts described in 17).
(Sends ISC_EASY_START_REQ)

19) Specifies Index No. 2 for the memory storage command and starts playback 1 second after playback starts described in 18).
(Sends ISC_EASY_START_REQ)

20) Waits for playback to end.
(Confirms the change in SOUND_PLAYING pin output signal from High to Low)

* When using this main program for mixing playback, the memory storage commands specified by Command Indexes 0, 1, and 2 must be for different playback systems (audio-main playback, audio-multi playback, synthesizer playback). Note that mixing playback cannot be performed by using memory storage commands for the same playback system due to specification restrictions. For detailed information on mixing playback, refer to the *S1V30080 Series Message Protocol Specifications*.

3.2 S1V30080 control API function specifications

The following describes specifications for API functions for controlling the S1V30080.

* The API functions described below are control programs for I2C on the host system used by Seiko Epson for S1V30080 control evaluations. Modifications must be made to API functions to suit client system specifications before incorporation into the client system.

3.2.1 I2C_Initialize

[Syntax]

void I2C_Initialize (void)

[Function]

Sets initial I2C settings.

[Input arguments]

None

[Output arguments]

None

[Return values]

None

[Function description]

Initializes I2C registers. This function complies with the I2C specifications of the host processor used for the Seiko Epson evaluation system and performs the following actions.

(1) Disables the I2C.

(2) Disables I2C interrupt settings.

(3) Sets the I2C port input/output direction.

(4) Sets the I2C baud rate.

(5) Sets the transfer mode to “MSB first.”

(6) Enables the I2C.

3.2.2 I2C_SendMessage

[Syntax]

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

[Function]

Sends messages to the S1V30080.

[Input arguments]

pucSendMessage	Specifies the address of the memory area containing the message to be sent.
iCheckSumFlag	Enables/disables checksum verification. (0: Disable / 1: Enable)

[Output arguments]

None

[Return values]

Returns 0 for normal termination; otherwise returns the following error codes.

Error code	Value	Description
I2CERR_SUCCESS	0	Returned for normal termination.
I2CERR_NULL_PTR	-1	Returned if the argument is a null pointer.

[Function description]

Sends the REQ message to the S1V30080. This function sends data by referring to the length field value in the REQ message to be sent.
After the message is sent, the function internally confirms the change in S1V30080's MESSAGE_RECEIVE pin signal output from Low to High. However, in the case of the following messages, this function ends without confirming the signal change due to message protocol specifications for melody streaming playback.

- ISC_SYNTHESIZER_MELODY_START_REQ
- ISC_AUDIO_MUTE_REQ
- ISC_AUDIO_VOLUME_REQ

3.2.3 I2C_ReceiveMessage

[Syntax]

int I2C_ReceiveMessage (
unsigned char aucReceivedMessage[])

[Function]

Receives messages from the S1V30080.

[Input arguments]

None

[Output arguments]

aucReceivedMessage Specifies the array for storing the received message.

[Return values]

Returns 0 for normal termination; otherwise returns the following error codes.

Error code	Value	Description
I2CERR_SUCCESS	0	Returned for normal termination.
I2CERR_NULL_PTR	-1	Returned if the argument is a null pointer.
I2CERR_TIMEOUT	1	Returned when the message from the S1V30080 is not received within the preset timeout period.

[Function description]

Receives RESP messages sent from the S1V30080. Data from the S1V30080 continues to be received until the message start command (0xAA) is detected. After the message start command is detected, the function refers to the length field value contained in the received data and continues receiving data.
The aucReceivedMessage output argument contains the data array of the received message.

3.2.4 I2C_SendByte

[Syntax]

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

[Function]

Sends 1 byte of data.

[Input arguments]

ucSendData Specifies 1 byte of the data to be sent.

[Output arguments]

None

[Return values]

None

[Function description]

Sends 1 byte of data to the S1V30080 via I2C.

3.2.5 I2C_ReceiveByte

[Syntax]

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

[Function]

Receives 1 byte of data.

[Input arguments]

cAck	Specifies Ack information returned to the S1V30080 functioning as the slave device. (0: Ack / 1: Nack)

[Output arguments]

None

[Return values]

Returns 1 byte of received data.

[Function description]

Receives 1 byte of data from the S1V30080 via I2C.

3.2.6 I2C_SetStartCondition

[Syntax]

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

[Function]

Sets the start conditions.

[Input arguments]

cAddressMode	Specifies the slave address mode. Specify 7.
usAddress	Specifies the slave address. Specify 01101100’b(0x6C).

[Output arguments]

None

[Return values]

Returns the Ack information received from the S1V30080. (0: Ack / 1: Nack)

[Function description]

Outputs the start condition waveform and sends the slave address. The Ack information is then received from the S1V30080 functioning as the slave device and returned as a return value.

3.2.7 I2C_SetStopCondition

[Syntax]

void I2C_SetStartCondition (void)

[Function]

Sets stop conditions.

[Input arguments]

None

[Output arguments]

None

[Return values]

None

[Function description]

Outputs the stop condition waveform.

3.2.8 S080_system_en

[Syntax]

void S080_system_en (int iValue)

[Function]

Controls the input signal to the SYSTEM_EN pin of the S1V30080.

[Input arguments]

iValue	Specifies the signal to be input to the SYSTEM_EN pin. (0: Low / 1: High)

[Output arguments]

None

[Return values]

None

[Function description]

Controls the input signal to the SYSTEM_EN pin to control the reset of the S1V30080. Reset is achieved by the input of Low → High → Low to the SYSTEM_EN pin of the S1V30080.
To use this function, you must connect the GPIO pin of the host processor to the S1V30080 SYSTEM_EN pin for signal control.

3.2.9 S080_sound_playing

[Syntax]

void S080_sound_playing (void)

[Function]

Acquires the status of the output signal from the S1V30080 SOUND_PLAYING pin.

[Input arguments]

None

[Output arguments]

None

[Return values]

Returns the voice output state. (0: Low / 1: High)

[Function description]

Acquires the signal output state from the SOUND_PLAYING pin to detect the condition of the voice output from the S1V30080. Returns High when voice is output or Low when voice is not output.
To use this function, you must connect the GPIO pin of the host processor to the S1V30080 SOUND_PLAYING pin for signal control.

3.2.10 S080_message_receive

[Syntax]

void S080_message_receive

[Function]

Detects the change in the MESSAGE_RECEIVE pin output signal from Low to High.

[Input arguments]

None

[Output arguments]

None

[Return values]

Returns the state of the change in output signal from Low to High.
(0: Without change / 1: With change)

[Function description]

Detects whether the output signal from the MESSAGE_RECEIVE pin of the S1V30080 changes from Low to High. Returns 1 when a change is detected or 0 when no change is detected.
This function detects a change in the signal using interrupt processing. To use this function, you must connect the GPIO pin of the host processor to the S1V30080 SOUND_PLAYING pin to apply and control the interrupt to the GPIO pin.

3.2.11 Other functions

In addition to the functions described thus far, the sample programs use the following functions.

- _i2cSendDataAsync Sends data to the slave.
- _i2cReceiveAck Receives Ack/Nack from the slave.
- _i2cSendAck Sends Ack/Nack to the slave.
- _i2cCreateStart Generates the start condition signal.
- _i2cCreateStop Generates the stop condition signal.

* The above functions are software control functions prepared to implement the I2C I/F using the bidirectional serial I/F (DCSIO) of the two channels mounted on the host system used by Seiko Epson for S1V30080 control evaluations. Modifications must be made to suit client system I2C I/F specifications before incorporation into the client host system.

Revision History

Date	Revision details			
	Rev.	Page	Type	Details
02/16/2009	1.00	All	New	Newly established

EPSON

International Sales Operations

AMERICA

EPSON ELECTRONICS AMERICA, INC.
2580 Orchard Parkway,
San Jose, CA 95131, USA
Phone: +1-800-228-3964 FAX: +1-408-922-0238

EUROPE

EPSON EUROPE ELECTRONICS GmbH
Riesstrasse 15, 80992 Munich,
GERMANY
Phone: +49-89-14005-0 FAX: +49-89-14005-110

ASIA

EPSON (CHINA) CO., LTD.
7F, Jinbao Bldg., No.89 Jinbao St.,
Dongcheng District,
Beijing 100005, CHINA
Phone: +86-10-6410-6655 FAX: +86-10-6410-7320

SHANGHAI BRANCH
7F, Block B, Hi-Tech Bldg., 900 Yishan Road,
Shanghai 200233, CHINA
Phone: +86-21-5423-5522 FAX: +86-21-5423-5512

SHENZHEN BRANCH
12F, Dawning Mansion, Keji South 12th Road,
Hi-Tech Park, Shenzhen 518057, CHINA
Phone: +86-755-2699-3828 FAX: +86-755-2699-3838

EPSON HONG KONG LTD.
20/F, Harbour Centre, 25 Harbour Road,
Wanchai, Hong Kong
Phone: +852-2585-4600 FAX: +852-2827-4346
Telex: 65542 EPSCO HX

EPSON TAIWAN TECHNOLOGY & TRADING LTD.
14F, No. 7, Song Ren Road,
Taipei 110, TAIWAN
Phone: +886-2-8786-6688 FAX: +886-2-8786-6660

EPSON SINGAPORE PTE., LTD.
1 HarbourFront Place,
#03-02 HarbourFront Tower One, Singapore 098633
Phone: +65-6586-5500 FAX: +65-6271-3182

SEIKO EPSON CORP.
KOREA OFFICE
50F, KLI 63 Bldg., 60 Yoido-dong,
Youngdeungpo-Ku, Seoul 150-763, KOREA
Phone: +82-2-784-6027 FAX: +82-2-767-3677

SEIKO EPSON CORP.
SEMICONDUCTOR OPERATIONS DIVISION

IC Sales Dept.
IC International Sales Group
421-8, Hino, Hino-shi, Tokyo 191-8501, JAPAN
Phone: +81-42-587-5814 FAX: +81-42-587-5117

Document Code: 411657600
First Issue February 2009 Ⓓ