EPSON
EXCEED YOUR VISION

S1V30120
Sample Program
Specification Manual

SEIKO EPSON CORPORATION

Rev.1.01

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.

Table of Contents

1. Overview 1

2. Sample Program Source List 2

2.1 Main Programs 2

2.2 Command Control Program 2

2.3 SPI Control Program 2

2.4 Header Files 3

2.5 Data Files 3

3. Sample Program Specifications 4

3.1 Main Programs 4

3.1.1 main_adpcm_normal.c 4

3.1.2 main_adpcm_etc.c 5

3.1.3 main_tts_normal.c 6

3.1.4 main_tts_etc.c 8

3.1.5 main_pman.c 9

3.1.6 main_gpio.c 10

3.2 Command Control API Specifications 11

3.2.1 func_ISC_BOOT_LOAD_REQ 11

3.2.2 func_ISC_BOOT_LOAD_RESP 13

3.2.3 func_ISC_BOOT_RUN_REQ 14

3.2.4 func_ISC_BOOT_RUN_RESP 15

3.2.5 func_ISC_TEST_REQ 16

3.2.6 func_ISC_VERSION_REQ 18

3.2.7 func_ISC_VERSION_RESP 19

3.2.8 func_ISC_AUDIO_CONFIG_REQ 20

3.2.9 func_ISC_AUDIO_VOLUME_REQ 22

3.2.10 func_ISC_AUDIO_MUTE_REQ 24

3.2.11 func_ISC_AUDIO_MUTE_RESP 26

3.2.12 func_ISC_AUDIO_POSTFILTER_CONFIG_REQ 28

3.2.14 func_ISC_SPCODEC_CONFIG_REQ 30
3.2.15 func_ISC_SPCODEC_START_REQ 32
3.2.16 func_ISC_SPCODEC_START_RESP 34
3.2.17 func_ISC_SPCODEC_PAUSE_REQ 36
3.2.18 func_ISC_SPCODEC_PAUSE_RESP 38
3.2.19 func_ISC_SPCODEC_STOP_REQ 40
3.2.20 func_ISC_SPCODEC_STOP_RESP 41
3.2.21 func_ISC_SPCODEC_FINISHED_IND 43
3.2.22 func_ISC_SPCODEC_READY_IND 45
3.2.23 func_ISC_PMAN_CONFIG_REQ 47
3.2.24 func_ISC_PMAN_STANDBY_ENTRY_REQ 49
3.2.25 func_ISC_PMAN_STANDBY_EXIT_IND 50
3.2.26 func_ISC_TTSCONFIG_REQ 51
3.2.27 func_ISC_TTS_SPEAK_REQ 53
3.2.28 func_ISC_TTS_SPEAK_RESP 55
3.2.29 func_ISC_TTS_STOP_REQ 57
3.2.30 func_ISC_TTS_STOP_RESP 58
3.2.31 func_ISC_TTS_PAUSE_REQ 60
3.2.32 func_ISC_TTS_PAUSE_RESP 62
3.2.33 func_ISC_TTS_FINISHED_IND 64
3.2.34 func_ISC_TTS_READY_IND 66
3.2.35 func ISC_TTS_UDICT_DATA_REQ 68
3.2.36 func_ISC_XXX_RESP 70
3.2.37 func_ISC_GPIO_REGISTER_REQ 72
3.2.38 func_ ISC_GPIO_OUTPUT_CONFIG_REQ 74
3.2.39 func_ISC_GPIO_OUTPUT_CONFIG_RESP 75
3.2.41 func_ISC_GPIO_OUTPUT_SET_REQ 77
3.2.42 func_ISC_GPIO_OUTPUT_SET_RESP 79
3.3 SPI Control API Specifications 81
3.3.1 SPI_initialise 81
3.3.2 SPI_transfer_commands 83

3.3.3 SPI_ReceiveCommands 84
3.3.4 SPI_TransferPaddingData 85
Appendix A Binary File Conversion Tool 86
Appendix B Typical SPI Register Specifications 87

1. Overview

This document describes the sample programs provided to clients using the S1V30120. These sample programs are incorporated into the host processor to control the S1V30120 on client systems. Also described here are the specifications for the corresponding APIs.

Use this document in conjunction with the message protocol specifications manual.

These sample programs are not guaranteed to function properly when incorporated into client systems.

2. Sample Program Source List

The S1V30120 demo kit contains the following sample source files.

\src (sample source files)

\header_files (header files)

Certain parts of the source code must be modified for compatibility with the client system when incorporating the sample programs.

* These sample programs are in a 32-bit processor (C33) and little-endian data format.

2.1 Main Programs

The main source files listed below include the set of control programs used for ADPCM decode playback processing, TTS (text-to-speech), and power management processing with the S1V30120 controlled by the host processor.

- main_adpcm_normal.c (ADPCM decode playback control program)
- main_adpcm_etc.c (ADPCM decode playback control program 2)
- main_tts_normal.c (TTS control program)
- main_tts_etc.c (TTS control program 2)
- main_pman.c (Power management control program)

The individual files comprise the main programs for these sample programs. All include program booting and system initialization. Individual files contain sample code that handles ADPCM decode playback/stop, pause, volume adjustment and power management control.

2.2 Command Control Program

The following source file implements command control functions:

- spi_command.c (sample program for command control)

The command control API included within this source file is called by the main programs described in 2.1.

2.3 SPI Control Program

The following source file implements SPI control:

- spi_api.c (SPI initialization and transmission/receipt API sample program)

The SPI control API included within this source file is called by the command programs described in 2.2.

*** "spi_api.c" is a control program based on an SPI on the host system used by Seiko Epson for S1V30120 control evaluation. The file must be modified to ensure compatibility when used with the client system.**

2.4 Header Files

The header files for the sample programs are indicated below.

- typedef.h (type declaration)
- spi_reg.h (SPI register map)
- spi_api.h (SPI downstream level API definition)
- spi_command_headers.h (API definition)
- s1v30120_gpio_msg_data.h (GPIO control definition)
- s1v30120_commonalg_msg_data.h" (ADPCM decode playback control definition)
- s1v30120_tts_msg_data.h (TTS control definition)
- s1v30120_audio_msg_data.h (audio configuration control definition)
- s1v30120_postfilt_msg_data.h (post filter control definition)
- s1v30120_pman_msg_data.h (power management control definition)
- s1v30120_boot_msg_data.h (boot control definition)
- s1v30120_msg_data.h (message protocol joint control definition)
- s1v30120_isc_messages.h (message protocol overall definition)
- isc_messages_debug.h (debug display function definition)

*** "spi_reg.h" contains the register map as is for the SPI on the host system used by Seiko Epson for S1V30120 control evaluation. Modifications must be made to ensure compatibility when used with the client system.**
("Appendix B Typical SPI Register Specifications" provides the specifications for the SPI register on the host system used by Seiko Epson.)

2.5 Data Files

The data files for the sample programs are shown below:

- adpcm.c (ADPCM data configuration)

"adpcm.c" is intended for use as a C source ADPCM data configuration but is not provided in the appropriate form and must be created by the client. Create by converting an ADPCM format binary file generated using the Audio Synthesis Authoring Tool into an ASCII format readable by the C source. Use "bin2text.exe" where necessary.

(Refer to "Appendix A Binary File Conversion Tool" for detailed information on using the "bin2text.exe" conversion tool.)

3. Sample Program Specifications

The specifications for the APIs used by the sample programs and specifics of the main programs are described below. The API group declarations used by the sample programs are all implemented by "spi_command_headers.h."

* The sample programs do not use interrupt processing. If interrupt processing is required for the client host system, modifications must be made to ensure compatibility with host system interrupt specifications.

3.1 Main Programs

Six programs are provided. The programs are controlled as described below.

3.1.1 main_adpcm_normal.c

Performs ADPCM decode playback after program booting and initialization.

(1) Initializes the SPI register set.

(2) Sends the boot code to S1V30120.
(ISC_BOOT_LOAD_REQ/RESP sent and received)

(3) Issues instruction for program booting.
(ISC_BOOT_RUN_REQ/RESP sent and received)

(4) Guarantees 120 ms standby for the S1V30120.

(5) Registers the device to S1V30120 (ISC_TEST_REQ/RESP sent and received)

(6) Requests notification of firmware version to S1V30120.
(ISC_VERSION_REQ/RESP sent and received)

(7) Issues instruction to initialize S1V30120 audio control.
(ISC_AUDIO_CONFIG_REQ/RESP sent and received)

(8) Issues instruction to initialize the S1V30120 power management function.
(ISC_PMAN_CONFIG_REQ/RESP sent and received)

(9) Issues instructions for configuration settings for S1V30120 ADPCM decode playback.
(ISC_SPCODEC_CONFIG_REQ/RESP sent and received)

(10) Issues instruction for transmission and playback of ADPCM streaming data to S1V30120. (ISC_SPCODEC_START_REQ/RESP sent and received)

(11) Awaits transmission if READY_IND has not been issued by the S1V30120.
(Awaits ISC_SPCODEC_READY_IND)

(12) Repeats (10) until the entire data stream has been transmitted and played back.

(13) Awaits a data stream complete instruction message from the S1V30120.
(Awaits ISC_SPCODEC_FINISHED_IND)

3.1.2 main_adpcm_etc.c

Performs ADPCM decode playback after program booting and initialization. Processing is performed midway for post-filtering, volume adjustment, and pausing. Data is played back without gaps after normal playback.

(1) Initializes the SPI register set.

(2) Sends the boot code to S1V30120.
(ISC_BOOT_LOAD_REQ/RESP sent and received)

(3) Issues instruction for program booting.
(ISC_BOOT_RUN_REQ/RESP sent and received)

(4) Guarantees 120 ms standby for the S1V30120.

(5) Registers the device to S1V30120.
(ISC_TEST_REQ/RESP sent and received)

(6) Requests notification of firmware version to S1V30120.
(ISC_VERSION_REQ/RESP sent and received)

(7) Issues instruction to initialize S1V30120 audio control.
(ISC_AUDIO_CONFIG_REQ/RESP sent and received)

(8) Issues instruction to initialize the S1V30120 power management function.
(ISC_PMAN_CONFIG_REQ/RESP sent and received)

(9) Issues instruction to initialize S1V30120 post filter.
(ISC_POSTFILTER_CONFIG_REQ/RESP sent and received)

(10) Issues instruction for S1V30120 audio volume control.
(ISC_AUDIO_VOLUME_REQ/RESP sent and received)

(11) Issues instructions for configuration settings for S1V30120 ADPCM decode playback.
(ISC_SPCODEC_CONFIG_REQ/RESP sent and received)

(12) Issues instruction for transmission and playback of ADPCM streaming data to S1V30120. (ISC_SPCODEC_START_REQ/RESP sent and received)

(13) Awaits transmission if READY_IND has not been issued by the S1V30120.
(Awaits ISC_SPCODEC_READY_IND)

(14) Issues instruction for muting during ADPCM playback in (12).

(15) Issues instruction for pausing during ADPCM playback in (12).

(16) Repeats (12) until the entire data stream has been transmitted and played back.

(17) Awaits data stream complete instructions from the S1V30120.
(Awaits ISC_SPCODEC_FINISHED_IND)

3.1.3 main_tts_normal.c

Performs TTS playback after program booting and initialization.

(1) Initializes the SPI register set.

(2) Sends the boot code to S1V30120.
(ISC_BOOT_LOAD_REQ/RESP sent and received)

(3) Issues instruction for program booting.
(ISC_BOOT_RUN_REQ/RESP sent and received)

(4) Guarantees 120 ms standby for the S1V30120.

(5) Registers the device to S1V30120.
(ISC_TEST_REQ/RESP sent and received)

(6) Requests notification of firmware version to S1V30120.
(ISC_VERSION_REQ/RESP sent and received)

(7) Issues instruction to initialize S1V30120 audio control.
(ISC_AUDIO_CONFIG_REQ/RESP sent and received)

(8) Issues instruction to initialize the S1V30120 power management function.
(ISC_PMAN_CONFIG_REQ/RESP sent and received)

(9) Issues instructions for configuration settings for S1V30120 TTS playback.
(ISC_TTS_CONFIG_REQ/RESP sent and received)

(10) Issues instruction for transmission and playback of TTS data to S1V30120.
(ISC_TTS_SPEAK_REQ/RESP sent and received)

(11) Awaits transmission if READY_IND has not been issued by the S1V30120.
(Awaits ISC_TTS_READY_IND)

(12) Issues instruction for TTS completion processing after playback ends.
(ISC_TTS_CLOSE_REQ/RESP sent and received)

(13) Awaits TTS data end instructions from the S1V30120.
(Awaits ISC_TTS_FINISHED_IND)

(14) Issues instructions for configuration settings for S1V30120 TTS playback.
(ISC_TTS_CONFIG_REQ/RESP sent and received)

(15) Issues instruction for transmission and playback of TTS data to S1V30120.
(ISC_TTS_SPEAK_REQ/RESP sent and received)

(16) Awaits transmission if READY_IND has not been issued by the S1V30120.
(Awaits ISC_TTS_READY_IND)

(17) Issues instruction for muting during TTS playback in (15).

(18) Issues instruction for pausing during TTS playback in (15).

(19) Repeats (15) until the TTS data has been transmitted and played back.

(20) Issues instruction for TTS playback end processing after playback ends.
(ISC_TTS_CSTOP_REQ/RESP sent and received)

(21) Issues instructions for configuration settings for S1V30120 TTS playback.
(ISC_TTS_CONFIG_REQ/RESP sent and received)

(22) Issues instruction for transmission and playback of TTS data to S1V30120.
(ISC_TTS_SPEAK_REQ/RESP sent and received)

(23) Awaits transmission if READY_IND has not been issued by the S1V30120.
(Awaits ISC_TTS_READY_IND)

(24) Repeats until TTS data in (22) ends, then issues instruction for TTS playback to S1V30120.

(25) Issues instruction for TTS completion processing after playback ends.
(ISC_TTS_CLOSE_REQ/RESP sent and received)

(26) Awaits TTS data end instructions from the S1V30120.
(Awaits ISC_TTS_FINISHED_IND)

3.1.4 main_tts_etc.c

Registers the dictionary and plays back the registered data after program booting and initialization.

(1) Initializes the SPI register set.

(2) Sends the boot code to S1V30120.
(ISC_BOOT_LOAD_REQ/RESP sent and received)

(3) Issues instruction for program booting.
(ISC_BOOT_RUN_REQ/RESP sent and received)

(4) Guarantees 120 ms standby for the S1V30120.

(5) Registers the device to S1V30120.
(ISC_TEST_REQ/RESP sent and received)

(6) Requests notification of firmware version to S1V30120.
(ISC_VERSION_REQ/RESP sent and received)

(7) Issues instruction to initialize S1V30120 audio control.
(ISC_AUDIO_CONFIG_REQ/RESP sent and received)

(8) Issues instruction to initialize the S1V30120 power management function.
(ISC_PMAN_CONFIG_REQ/RESP sent and received)

(9) Issues instructions for configuration settings for S1V30120 TTS playback.
(ISC_TTS_CONFIG_REQ/RESP sent and received)

(10) Registers the dictionary for S1V30120 TTS playback.
(ISC_TTS_UDICT_DATA_REQ/RESP sent and received)

(11) Issues configuration settings for TTS playback of data registered in the S1V30120.
(ISC_TTS_CONFIG_REQ/RESP sent and received)

(12) Issues instruction for transmission and playback of TTS data to S1V30120.
(ISC_TTS_SPEAK_REQ/RESP sent and received)

(13) Awaits transmission if READY_IND has not been issued by the S1V30120.
(Awaits ISC_TTS_READY_IND)

(14) Issues instruction for TTS completion processing after playback ends.
(ISC_TTS_CLOSE_REQ/RESP sent and received)

(15) Awaits TTS data end instructions from the S1V30120.
(Awaits ISC_TTS_FINISHED_IND)

3.1.5 main_pman.c

Performs power management processing after program booting and initialization.

(1) Initializes the SPI register set.

(2) Sends the boot code to S1V30120.

(ISC_BOOT_LOAD_REQ/RESP sent and received)

(3) Issues instruction for program booting.

(ISC_BOOT_RUN_REQ/RESP sent and received)

(4) Guarantees 120 ms standby for the S1V30120.

(5) Registers the device to S1V30120.

(ISC_TEST_REQ/RESP sent and received)

(6) Requests notification of firmware version to S1V30120.

(ISC_VERSION_REQ/RESP sent and received)

(7) Issues instruction to initialize S1V30120 audio control.

(ISC_AUDIO_CONFIG_REQ/RESP sent and received)

(8) Issues instruction to initialize the S1V30120 power management function.

(ISC_PMAN_CONFIG_REQ/RESP sent and received)

(9) Instructs the S1V30120 to enter standby mode.
(ISC_PMAN_STANDBY_ENTRY_REQ/RESP sent and received)

(10) (In standby mode)

(11) Transmits ISC_PMAN_STANDBY_EXIT_IND to S1V30120 to instruct recovery from standby mode.

(ISC_PMAN_STANDBY_EXIT_IND sent)

(12) Awaits standby mode end instructions from the S1V30120.
(ISC_PMAN_STANDBY_EXIT_IND received)

3.1.6 main_gpio.c

Performs GPIO control after program booting and initialization.

(1) Initializes the SPI register set.

(2) Sends the boot code to S1V30120.

(ISC_BOOT_LOAD_REQ/RESP sent and received)

(3) Issues instruction for program booting.

(ISC_BOOT_RUN_REQ/RESP sent and received)

(4) Guarantees 120 ms standby for the S1V30120.

(5) Registers the device to S1V30120.

(ISC_TEST_REQ/RESP sent and received)

(6) Requests notification of firmware version to S1V30120.

(ISC_VERSION_REQ/RESP sent and received)

(7) Issues instruction to initialize S1V30120 audio control.

(ISC_AUDIO_CONFIG_REQ/RESP sent and received)

(8) Issues instruction to initialize the S1V30120 power management function.

(ISC_PMAN_CONFIG_REQ/RESP sent and received)

(9) Starts S1V30120 GPIO control.
(ISC_GPIO_REGISTER_REQ/RESP sent and received)

(10) Issues instruction to initialize for S1V30120 GPIO output control.
(ISC_GPIO_OUTPUT_CONFIG_REQ/RESP sent and received)

(11) Sets the S1V30120 GPIO5 to GPIO11 output to High/Low.
(ISC_GPIO_OUTPUT_SET_REQ/RESP sent and received)

** **Recovery from standby mode in 3.1.5 requires sharing of S1V30120 NSCSS and GPIOA-4 signal lines. For detailed information, refer to the message protocol specifications manual.**

3.2 Command Control API Specifications

The specifications for the command control related program APIs (defined by “spi_command.c”) used by the sample programs are described below.

3.2.1 func_ISC_BOOT_LOAD_REQ

[Format]

```
int     func_ISC_BOOT_LOAD_REQ (    uWord8   *transfer_data,
                                    uWord8   received_data[ ],
                                    uWord16 transfer_length)
```

[Function]

Sends the boot code to S1V30120.

ISC_BOOT_LOAD_REQ is sent with the boot code attached.

The response is received by the func_ISC_BOOT_RESP_RESP function.

16 bytes of padding are sent following the command sequence.

[Input arguments]

*transfer_data	Indicates the data pointer for the boot code sent. Excludes information such as command header (0xAA), message ID, data length, and padding.
transfer_length	Indicates the data length for the boot code sent. Excludes information such as command header (0xAA), message ID, data length, and padding.

[Output arguments]

received_data[ ]	Specifies the work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.2 func_ISC_BOOT_LOAD_RESP

[Format]

```
int    func_ISC_BOOT_LOAD_RESP(    int    iReceivedCounts,
                                   uWord8  received_data[ ],
                                   uWord8  **errData)
```

[Function]

Receives the response for the data sent by the func_ISC_BOOT_RUN_REQ function.

16 bytes of padding are sent after the command sequence is received.

[Input arguments]

iReceivedCounts Data size received by the func_ISC_BOOT_LOAD_REQ function.

[Output arguments]

received_data[ ] Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

**errData Indicates the data pointer if a response contains an error code.

[Returned values]

Returns 0 if ISC_BOOT_LOAD_RESP is received normally.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.3 func_ISC_BOOT_RUN_REQ

[Format]

int func_ISC_BOOT_RUN_REQ(uWord8 received_data[ ])

[Function]

Sends ISC_BOOT_RUN_REQ to issue instruction for program booting.

The response is received by the func_ISC_BOOT_RUN_RESP function.

[Input arguments]

None

[Output arguments]

received_data[ ] Specifies the work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_BOOT_RUN_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.4 func_ISC_BOOT_RUN_RESP

[Format]

```
int     func_ISC_BOOT_RUN_RESP(     int     iReceivedCounts,
                                            uWord8  received_data[ ],
                                            uWord8  **errData)
```

[Function]

Receives the response for the data sent by the func_ISC_BOOT_RUN_REQ function.

8 bytes of padding are sent after the command sequence is received.

[Input arguments]

iReceivedCounts	Data size received by the func_ISC_BOOT_RUN_REQ function.

[Output arguments]

received_data[ ]	Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).
**errData	Indicates the data pointer if a response contains an error code.

[Returned values]

Returns 0 if ISC_BOOT_RUN_RESP is received normally.

The following errors are issued if the value is less than 0:

RCV_SPI_TIMEOUT (-1):	Data reception timeout
RCV_ISC_ERROR_IND (-2):	Fatal error occurred in device
RCV_UNEXPECTED_ID (-3):	Unexpected data returned
RCV_BUFF_OVERFLOW (-4):	Receiving buffer overflow
RCV_DEVICE_ERROR (-5):	Response message received containing error code (non-fatal error)
RCV_DEVICE_FAUAL (-6):	Response message received containing error code (fatal error)

3.2.5 func_ISC_TEST_REQ

[Format]

```
int     func_ISC_TEST_REQ(      uWord8          register,
                                uWord8          received_data[ ])
```

[Function]

Issues instruction for device registration/cancellation to S1V30120.

Sends device registration parameters to ISC_TEST_REQ.

The response is received by the func_ISC_XXX_RESP function.

(The expected value of the received data is ISC_TEST_RESP.)

16 bytes of padding are sent following the command sequence.

[Input arguments]

register	0x0000:	Issues instruction for device registration cancellation.
	0x0001:	Issues instruction for device registration.

[Output arguments]

received_data[ ]	Specifies the work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.6 func_ISC_VERSION_REQ

[Format]

int func_ISC_VERSION_REQ(uWord8 received_data[ ])

[Function]

Sends ISC_VERSION_REQ to obtain S1V30120 version details.

The response is received by the func_ISC_VERSION_RESP function.

(The expected value of the received data is ISC_VERSION_RESP.)

16 bytes of padding are sent following the command sequence.

[Input arguments]

None

[Output arguments]

received_data[ ] Specifies the work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.7 func_ISC_VERSION_RESP

[Format]

```
int     func_ISC_VERSION_RESP(      int     iReceivedCounts,
                                    uWord8  received_data[ ],
                                    uWord8  errData[])
```

[Function]

Receives the response for the data sent by the func_ISC_VERSION_REQ function.

16 bytes of padding are sent after the command sequence is received.

[Input arguments]

iReceivedCounts	Data size received by the func_ISC_VERSION_REQ function

[Output arguments]

received_data[ ]	Specifies the command sequence received. This includes the initial padding (0x00) and command header (0xAA), message ID, data length, and padding. Thus, the initial value of the sequence will be the initial padding (0x00).
errData[]	Returns the data if a response contains an error code.

[Returned values]

Returns 0 if ISC_VERSION_RESP is received normally.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.8 func_ISC_AUDIO_CONFIG_REQ

[Format]

int func_ISC_AUDIO_CONFIG_REQ(uWord8 received_data[ ])

[Function]

Sends ISC_AUDIO_CONFIG_REQ to initialize of the S1V30120 AUDIO control. The response is received by the func_ISC_XXX_RESP function.

(The expected value of the received data is ISC_AUDIO_CONFIG_RESP.)

The volume is set to 0 dB but can be reset to the desired value by altering the “audio_gain” parameter inside this API. The sampling frequency output is decoded from the stream header and automatically set by the decoder.

(“audio_sample_rate” inside the API is set to “Don’t care.”)

16 bytes of padding are sent following the command sequence.

[Input arguments]

None

[Output arguments]

received_data[ ] Specifies the work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.9 func_ISC_AUDIO_VOLUME_REQ

[Format]

```
int     func_ISC_AUDIO_VOLUME_REQ(  uWord16     gain,
                                    uWord8      received_data[ ])
```

[Function]

Sends ISC_AUDIO_VOLUME_REQ to control the S1V30120 VOLUME. (May be specified during ADPCM playback.)

The response is received by the func_ISC_XXX_RESP function.

(The expected value of the received data is ISC_AUDIO_VOLUME_RESP.)

Increases or decreases from the current volume setting can be specified in dB using the "gain" argument.

16 bytes of padding are sent following the command sequence.

[Input arguments]

gain	Indicates the increase or decrease from the current volume in dB. (Example: 0x0006 is a gain of +6 dB.) Saturation processing is performed using the maximum (initial value +18 dB) or minimum value (initial value -48 dB) for increases or decreases beyond the specified range. (The value range is between -48 dB and +18 dB from the initial value.)

[Output arguments]

received_data[ ]	Specifies the work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.10 func_ISC_AUDIO_MUTE_REQ

[Format]

int	func_ISC_AUDIO_MUTE_REQ(	uWord16	enable,
		uWord8	received_data[ ])

[Function]

Sends ISC_AUDIO_MUTE_REQ to control the S1V30120 MUTE. (Can be specified during ADPCM playback.)

The response is received by the func_ISC_AUDIO_MUTE_RESP function.

Mute is enabled or disabled by the “enable” argument.

16 bytes of padding are sent following the command sequence.

[Input arguments]

enable	0x0000:	Disables MUTE.
	0x0001:	Enables MUTE.

[Output arguments]

received_data[ ]	Specifies the work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_AUDIO_MUTE_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.11 func_ISC_AUDIO_MUTE_RESP

[Format]

```
int     func_ISC_AUDIO_MUTE_RESP(    int     iReceivedCounts,
                                     uWord8  received_data[ ],
                                     int     *IndFlag,
                                     uWord8  errData[])
```

[Function]

Receives the response for the data sent by the func_ISC_AUDIO_MUTE_REQ function.

16 bytes of padding are sent after the command sequence is received.

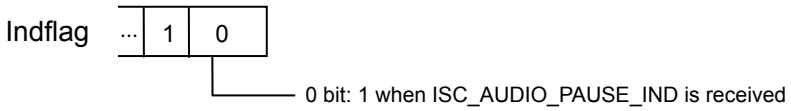
[Input arguments]

iReceivedCounts Data size received by the func_ISC_AUDIO_MUTE_REQ function.

[Output arguments]

received_data[ ] Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

*IndFlag The ISC_AUDIO_PAUSE_IND receipt status is indicated by the following bit:



errData[] Returns the data if a response contains an error code.

[Returned values]

Returns 0 if ISC_AUDIO_MUTE_RESP is received normally.

The following errors are issued if the value is less than 0:

RCV_SPI_TIMEOUT (-1):	Data reception timeout
RCV_ISC_ERROR_IND (-2):	Fatal error occurred in device
RCV_UNEXPECTED_ID (-3):	Unexpected data returned
RCV_BUFF_OVERFLOW (-4):	Receiving buffer overflow
RCV_DEVICE_ERROR (-5):	Response message received containing error code (non-fatal error)
RCV_DEVICE_FATAL (-6):	Response message received containing error code (fatal error)

3.2.12 func_ISC_AUDIO_POSTFILTER_CONFIG_REQ

[Format]

```
int     func_ISC_AUDIO_POSTFILTER_CONFIG_REQ        (
                                        uWord8      *data_ptr,
                                        uWord16     enable,
                                        uWord8      received_data[ ])
```

[Function]

Sends the post-filter coefficients to the S1V30120 and attaches the post-filter coefficients to ISC_POSTFILTER_CONFIG_REQ to enable or disable post-filtering.

The response is received by the func_ISC_XXX_RESP function.

(The expected value of the received data is ISC_POSTFILTER_CONFIG_RESP.)

Enabling or disabling is specified by the “enable” argument.

16 bytes of padding are sent following the command sequence.

[Input arguments]

data_ptr		Indicates the coefficient data pointer for the post-filter sent. Excludes information such as the command header (0xAA), message ID, data length, and padding.
enable	0x0000:	Disables the post-filter.
	0x0001:	Enables the post-filter.

[Output arguments]

received_data[ ]	Specifies the work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.14 func_ISC_SPCODEC_CONFIG_REQ

[Format]

```
int     func_ISC_SPCODEC_CONFIG_REQ(
                                        uWord8   input_source,
                                        uWord8   received_data[ ])
```

[Function]

Sends ISC_SPCODEC_CONFIG_REQ to issue configuration settings for S1V30120 ADPCM decode playback.

The response is received by the func_ISC_XXX_RESP function.

(The expected value of the received data is ISC_SPCODEC_CONFIG_RESP.)

16 bytes of padding are sent following the command sequence.

[Input arguments]

input_source Specifies the playback data type.

- 0x00: Embedded file system
- 0x01: Active IF

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.15 func_ISC_SPCODEC_START_REQ

[Format]

```
int     func_ISC_SPCODEC_START_REQ    (
                                              uWord8        *transfer_data,
                                              uWord8        received_data[ ],
                                              uWord16       transfer_length)
```

[Function]

Sends the ADPCM data attached to ISC_SPCODEC_START_REQ to issue instructions for transmission or playback of ADPCM streaming data to the S1V30120.

The response is received by the func_ISC_SPCODEC_START_RESP function.

16 bytes of padding are sent following the command sequence.

[Input arguments]

*transfer_data — Indicates the data pointer for the ADPCM stream to be sent. This excludes information such as the command header (0xAA), message ID, data length, and padding.

transfer_length — Indicates the data length for the ADPCM stream to be sent. This excludes information such as the command header (0xAA), message ID, data length, and padding.

[Output arguments]

received_data[ ] — Specifies the work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_SPCODEC_START_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.16 func_ISC_SPCODEC_START_RESP

[Format]

```
int     func_ISC_SPCODEC_START_RESP (
                                        int     iReceivedCounts,
                                        uWord8  received_data[ ],
                                        int     *IndFlag,
                                        uWord8  **errData)
```

[Function]

Receives the response for the data sent by the ISC_SPCODEC_START_RESP function.

16 bytes of padding are sent after the command sequence is received.

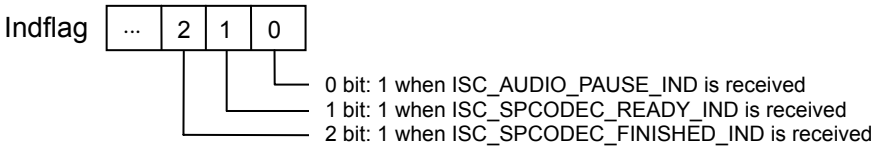
[Input arguments]

iReceivedCounts Data size received by the func_ISC_SPCODEC_START_REQ function.

[Output arguments]

received_data[ ] Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

*IndFlag The ISC_AUDIO_PAUSE_IND/ISC_SPCODEC_READY_IND and ISC_SPCODEC_FINISHED_IND receipt statuses are indicated by the bits shown below:



**errData Indicates the data pointer if a response contains an error code.

[Returned values]

Returns 0 if ISC_SPCODEC_START_RESP is received normally.

The following errors are issued if the value is less than 0:

RCV_SPI_TIMEOUT (-1):	Data reception timeout
RCV_ISC_ERROR_IND (-2):	Fatal error occurred in device
RCV_UNEXPECTED_ID (-3):	Unexpected data returned
RCV_BUFF_OVERFLOW (-4):	Receiving buffer overflow
RCV_DEVICE_ERROR (-5):	Response message received containing error code (non-fatal error)
RCV_DEVICE_FAUAL (-6):	Response message received containing error code (fatal error)

3.2.17 func_ISC_SPCODEC_PAUSE_REQ

[Format]

int func_ISC_SPCODEC_PAUSE_REQ(

uWord16 enable,

uWord8 received_data[ ])

[Function]

Sends ISC_SOCODEC_PAUSE_REQ to pause during S1V30120 ADPCM playback. (Can be specified during ADPCM playback).

The response is received by the func_ISC_SPCODEC_PAUSE_RESP function.

PAUSE is enabled or disabled by the “enable” argument.

16 bytes of padding are sent following the command sequence.

[Input arguments]

enable 0x0000: Disables PAUSE.

0x0001: Enables PAUSE.

[Output arguments]

received_data[ ] Specifies the work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_SPCODEC_PAUSE_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.18 func_ISC_SPCODEC_PAUSE_RESP

[Format]

```
int     func_ISC_SPCODEC_PAUSE_RESP (
                                        int         iReceivedCounts,
                                        uWord8      received_data[ ],
                                        int         *IndFlag,
                                        uWord8      **errData)
```

[Function]

Receives the response for the data sent by the func_ISC_SPCODEC_PAUSE_REQ function.

16 bytes of padding are sent after the command sequence is received.

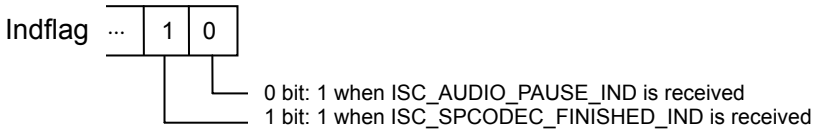
[Input arguments]

iReceivedCounts — Data size received by the func_ISC_SPCODEC_START_REQ function.

[Output arguments]

received_data[ ] — Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

*IndFlag — The ISC_AUDIO_PAUSE_IND and ISC_SPCODEC_FINISHED_IND receipt statuses are indicated by the bits shown below:



**errData — Indicates the data pointer if a response contains an error code.

[Returned values]

Returns 0 if ISC_SPCODEC_PAUSE_RESP is received normally.

The following errors are issued if the value is less than 0:

RCV_SPI_TIMEOUT (-1):	Data reception timeout
RCV_ISC_ERROR_IND (-2):	Fatal error occurred in device
RCV_UNEXPECTED_ID (-3):	Unexpected data returned
RCV_BUFF_OVERFLOW (-4):	Receiving buffer overflow
RCV_DEVICE_ERROR (-5):	Response message received containing error code (non-fatal error)
RCV_DEVICE_FAUAL (-6):	Response message received containing error code (fatal error)

3.2.19 func_ISC_SPCODEC_STOP_REQ

[Format]

int func_ISC_SPCODEC_STOP_REQ_(uWord8 received_data[ ])

[Function]

Stop command for ADPCM (Auto stop is available). Sends ISC_SPCODEC_STOP_REQ to stop ADPCM decoding to the S1V30120. (Can be specified during ADPCM playback.)

The response is received by the func_ISC_SPCODEC_STOP_RESP function.

16 bytes of padding are sent following the command sequence.

[Input arguments]

None

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_SPCODEC_STOP_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.20 func_ISC_SPCODEC_STOP_RESP

[Format]

```
int     func_ISC_SPCODEC_STOP_RESP(
                                        int     iReceivedCounts ,
                                        uWord8  received_data[ ],
                                        int     *IndFlag,
                                        uWord8  **errData)
```

[Function]

Receives the response for the data received by the func_ISC_SPCODEC_STOP_REQ function.

16 bytes of padding are sent after the command sequence is received.

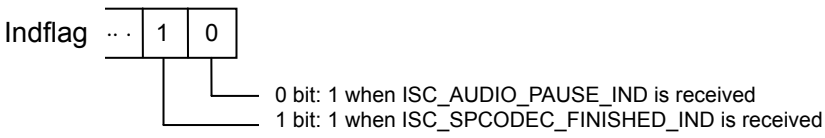
[Input arguments]

iReceivedCounts Data size received by the func_ISC_SPCODEC_STOP_REQ function.

[Output arguments]

received_data[ ] Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

*IndFlag The ISC_AUDIO_PAUSE_IND and ISC_SPCODEC_FINISHED_IND receipt statuses are indicated by the bits shown below:



**errData Indicates the data pointer if a response contains an error code.

[Returned values]

Returns 0 if ISC_SPCODEC_STOP_RESP is received normally.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.21 func_ISC_SPCODEC_FINISHED_IND

[Format]

```
int    func_ISC_SPCODEC_FINISHED_IND (    int       iReceivedCounts,
                                          int       *IndFlag,
                                          uWord8    received_data[ ])
```

[Function]

Receives ISC_SPCODEC_FINISHED_IND.

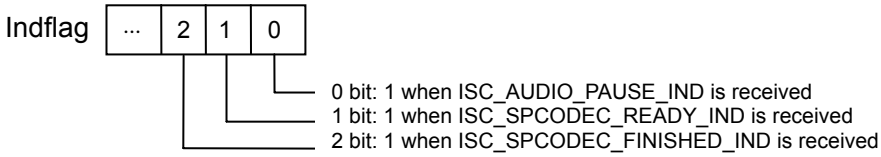
16 bytes of padding are sent after the command sequence is received.

[Input arguments]

iReceivedCounts — Specifies the buffer data size when using the work-receiving buffer received by the func_*_REQ function. Specifies 0 in all other cases.

[Output arguments]

*IndFlag — The ISC_AUDIO_PAUSE_IND/ISC_SPCODEC_READY_IND and ISC_SPCODEC_FINISHED_IND receipt statuses are indicated by the bits shown below:



received_data[ ] — Specifies the buffer data size when using the work-receiving buffer received by the func_*_REQ function. Specifies receiving buffer in all other cases.
Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

[Returned values]

Returns 0 if ISC_SPCODEC_FINISHED_IND is received normally.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.22 func_ISC_SPCODEC_READY_IND

[Format]

```
int    func_ISC_SPCODEC_READY_IND (    int       iReceivedCounts,
                                       int       *IndFlag,
                                       uWord8    received_data[ ])
```

[Function]

Receives ISC_SPCODEC_READY_IND.

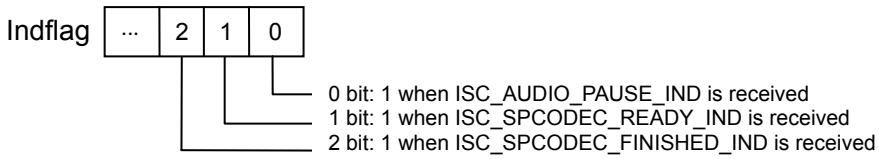
16 bytes of padding are sent after the command sequence is received.

[Input arguments]

iReceivedCounts Specifies the buffer data size when using the work-receiving buffer received by the func_*_REQ function. Specifies 0 in all other cases.

[Output arguments]

*IndFlag The ISC_AUDIO_PAUSE_IND/ISC_SPCODEC_READY_IND and ISC_SPCODEC_FINISHED_IND receipt statuses are indicated by the bits shown below:



received_data[ ] Specifies the buffer data size when using the work-receiving buffer received by the func_*_REQ function. Specifies the receiving buffer in all other cases.
Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

[Returned values]

Returns 0 if ISC_SPCODEC_READY_IND is received normally.

The following errors are issued if the value is less than 0:

RCV_SPI_TIMEOUT (-1):	Data reception timeout
RCV_ISC_ERROR_IND (-2):	Fatal error occurred in device
RCV_UNEXPECTED_ID (-3):	Unexpected data returned
RCV_BUFF_OVERFLOW (-4):	Receiving buffer overflow
RCV_DEVICE_ERROR (-5):	Response message received containing error code (non-fatal error)
RCV_DEVICE_FAUAL (-6):	Response message received containing error code (fatal error)

3.2.23 func_ISC_PMAN_CONFIG_REQ

[Format]

```
int     func_ISC_PMAN_CONFIG_REQ      (
                                          uWord16   mode,
                                          uWord8    host_mhz,
                                          uWord8    received_data[ ])
```

[Function]

Issues instruction to initialize the S1V30120 power management function.

Sends ISC_PMAN_CONFIG_REQ and obtains the command sequence received.

The response is received by the func_ISC_XXX_RESP function.

(The expected value of the received data is ISC_PMAN_CONFIG_RESP.)

The power management function is supported by “Minimum delay audio mode,” “Minimum power audio mode,” and “Audio PLL constant on.” It specifies the desired functions depending on the mode.

16 bytes of padding are sent following the command sequence.

[Input arguments]

enable 0x0000: Specifies Minimum delay audio mode.

0x0001: Specifies Minimum power audio mode.

0x0002: Specifies Audio PLL constant on.

host_mz Specifies the SPI bit clock frequency used by the host processor. The value is specified in MHz, rounded down to the nearest 1 MHz.

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.24 func_ISC_PMAN_STANDBY_ENTRY_REQ

[Format]

int func_ISC_PMAN_STANDBY_ENTRY_REQ(uWord8 received_data[ ])

[Function]

Sends ISC_PMAN_STANDBY_ENTRY_REQ to issue instruction for S1V30120 power management entry.

The response is received by the func_ISC_XXX_RESP function.

(The expected value of the received data is ISC_PMAN_STANDBY_ENTRY_RESP.)

16 bytes of padding are sent following the command sequence.

[Input arguments]

None

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.25 func_ISC_PMAN_STANDBY_EXIT_IND

[Format]

int func_ISC_PMAN_STANDBY_EXIT_IND(uWord8 received_data[ ])

[Function]

Sends ISC_PMAN_STANDBY_EXIT_IND to issue instruction for S1V30120 recovery from power management.

The response is received by the func_ISC_XXX_RESP function.

(The expected value of the received data is ISC_PMAN_STANDBY_EXIT_IND.)

16 bytes of padding are sent following the command sequence.

[Input arguments]

None

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

RCV_SPI_TIMEOUT (-1):	Data reception timeout
RCV_ISC_ERROR_IND (-2):	Fatal error occurred in device
RCV_UNEXPECTED_ID (-3):	Unexpected data returned
RCV_BUFF_OVERFLOW (-4):	Receiving buffer overflow
RCV_DEVICE_ERROR (-5):	Response message received containing error code (non-fatal error)
RCV_DEVICE_FATAL (-6):	Response message received containing error code (fatal error)

3.2.26 func_ISC_TTSCONFIG_REQ

[Format]

int func_ISC_TTS_CONFIG_REQ(

msg_tts_config_req_t *ttsConfigInfo,

uWord8 received_data[])

[Function]

Sends ISC_TTS_CONFIG_REQ to set the configuration for S1V30120 TTS playback.

The response is received by the func_ISC_XXX_RESP function.

(The expected value of the received data is ISC_TTS_CONFIG_RESP.)

16 bytes of padding are sent following the command sequence.

[Input arguments]

ttsConfigInfo: TTS configuration setting structure
(For detailed information on the TTS configuration setting structure, refer to the *S1V30120 Message Protocol Specification* manual.)

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.27 func_ISC_TTS_SPEAK_REQ

[Format]

```
int     func_ISC_TTS_SPEAK_REQ    (
                                        uWord8     flush_enable,
                                        uWord8     *data_ptr,
                                        uWord8     received_data[],
                                        uWord16    length)
```

[Function]

Sends ISC_TTS_SPEAK_REQ for S1V30120 TTS playback.

The response is received by the func_ISC_TTS_SPEAK_RESP function.

16 bytes of padding are sent following the command sequence.

[Input arguments]

flush_enable	0x0000	TTS playback after request is completed
	0x0001	TTS playback immediately
*data_ptr	Indicates the data pointer for the text to be sent. Excludes information such as command header (0xAA), message ID, data length, and padding.	
length	Indicates the data length for the text to be sent. Excludes information such as command header (0xAA), message ID, data length, and padding.	

[Output arguments]

received_data[ ]	Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_TTS_SPEAK_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.28 func_ISC_TTS_SPEAK_RESP

[Format]

```
int    func_ISC_TTS_SPEAK_RESP(   int    iReceivedCounts ,
                                  Word8   received_data[ ],
                                  int     *IndFlag,
                                  uWord8  errData[])
```

[Function]

Receives the response for the data sent by the func_ISC_TTS_SPEAK_REQ function.

16 bytes of padding are sent after the command sequence is received.

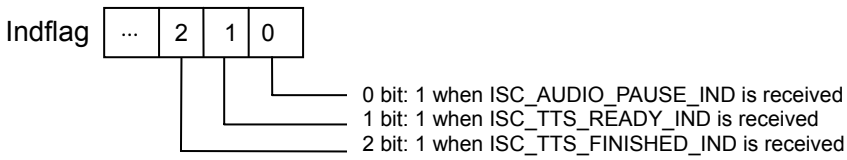
[Input arguments]

iReceivedCounts Data size received by the func_ISC_SEQUENCER_START_REQ function.

[Output arguments]

received_data[ ] Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

*IndFlag The ISC_AUDIO_PAUSE_IND/ISC_TTS_READY_IND and ISC_TTS_FINISHED_IND receipt statuses are indicated by the bits shown below:



errData[] Returns the data if a response contains an error code.

[Returned values]

Returns 0 if ISC_SEQUENCER_START_RESP is received normally.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.29 func_ISC_TTS_STOP_REQ

[Format]

int func_ISC_TTS_STOP_REQ(

uWord16 reset_tts,

uWord8 received_data[ ])

[Function]

Sends ISC_TTS_STOP_REQ to stop S1V30120 playback.

The response is received by the func_ISC_TTS_STOP_RESP function.

16 bytes of padding are sent following the command sequence.

[Input arguments]

reset_tts 0x0000: Does not reset TTS on stopping.

0x0001: Resets TTS on stopping.

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_TTS_STOP_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.30 func_ISC_TTS_STOP_RESP

[Format]

```
int    func_ISC_TTS_STOP_RESP(    int      iReceivedCounts,
                                  uWord8   aucReceivedData[],
                                  int      * IndFlag,
                                  uWord8   errData[])
```

[Function]

Receives the response for the data sent by the func_ISC_TTS_STOP_REQ function.

16 bytes of padding are sent after the command sequence is received.

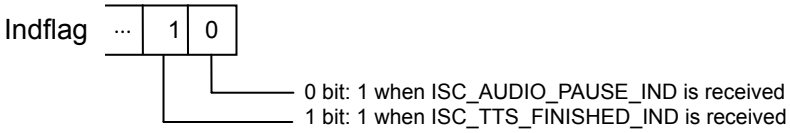
[Input arguments]

iReceivedCounts Data size received by the func_ISC_SEQUENCER_STOP_RESP function.

[Output arguments]

received_data[ ] Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

*IndFlag The ISC_AUDIO_PAUSE_IND and ISC_TTS_FINISHED_IND receipt statuses are indicated by the bits shown below:



errData[] Returns the data if a response contains an error code.

[Returned values]

Returns 0 if ISC_TTS_STOP_RESP is received normally.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.31 func_ISC_TTS_PAUSE_REQ

[Format]

```
int    func_ISC_TTS_PAUSE_REQ(    uWord8    pause_info,
                                  uWord8    received_data[ ])
```

[Function]

Issues instruction for a pause during S1V30120 TTS playback.

Transfers pause or pause cancel information in the argument and sends ISC_TTS_PAUSE_REQ.

The response is received by the func_ISC_TTS_PAUSE_RESP function.

16 bytes of padding are sent following the command sequence.

[Input arguments]

pause_info 0x0000: Pause cancel

0x0001: Pause

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_TTS_STOP RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.32 func_ISC_TTS_PAUSE_RESP

[Format]

```
int    func_ISC_TTS_PAUSE_RESP(    int      iReceivedCounts,
                                   uWord8   aucReceivedData[],
                                   int      * IndFlag,
                                   uWord8   errData[])
```

[Function]

Receives the response for the data sent by the func_ISC_TTS_PAUSE_REQ function.

16 bytes of padding are sent after the command sequence is received.

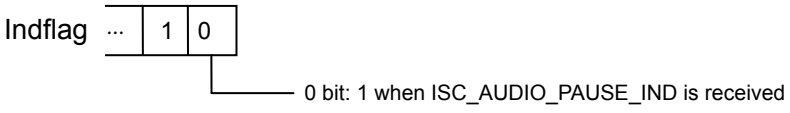
[Input arguments]

iReceivedCounts Data size received by the func_ISC_TTS_PAUSE_REQ function

[Output arguments]

received_data[ ] Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

*IndFlag The ISC_AUDIO_PAUSE_IND receipt status is indicated by the bit shown below:



errData[] Returns the data if a response contains an error code.

[Returned values]

Returns 0 if ISC_TTS_PAUSE_RESP is received normally.

The following errors are issued if the value is less than 0:

RCV_SPI_TIMEOUT (-1):	Data reception timeout
RCV_ISC_ERROR_IND (-2):	Fatal error occurred in device
RCV_UNEXPECTED_ID (-3):	Unexpected data returned
RCV_BUFF_OVERFLOW (-4):	Receiving buffer overflow
RCV_DEVICE_ERROR (-5):	Response message received containing error code (non-fatal error)
RCV_DEVICE_FATAL (-6):	Response message received containing error code (fatal error)

3.2.33 func_ISC_TTS_FINISHED_IND

[Format]

```
int    func_ISC_TTS_FINISHED_IND (    int       iReceivedCounts,
                                      int       *IndFlag,
                                      uWord8    received_data[ ])
```

[Function]

Receives ISC_TTS_FINISHED_IND.

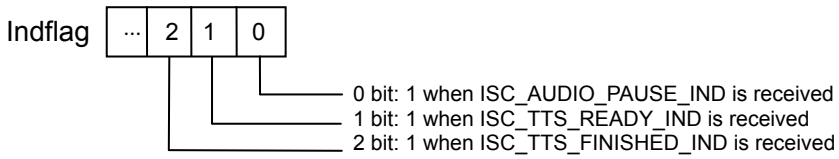
16 bytes of padding are sent after the command sequence is received.

[Input arguments]

iReceivedCounts Specifies the buffer data size when using the work-receiving buffer received by the func_*_REQ function. Specifies 0 in all other cases.

[Output arguments]

*IndFlag The ISC_AUDIO_PAUSE_IND/ISC_TTS_READY_IND and ISC_TTS_FINISHED_IND receipt statuses are indicated by the bits shown below:



received_data[ ] Specifies the buffer when using the work-receiving buffer received by the func_*_REQ function. Specifies the receiving buffer in all other cases.
The data received indicates the command sequence. This includes the initial padding (0x00) and command header (0xAA), message ID, data length, and padding. The initial value of the sequence will therefore be the initial padding (0x00).

[Returned values]

Returns 0 if ISC_TTS_FINISHED_IND is received normally.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.34 func_ISC_TTS_READY_IND

[Format]

```
int     func_ISC_TTS_READY_IND (    int     iReceivedCounts,
                                    int     *IndFlag,
                                    uWord8  received_data[ ])
```

[Function]

Receives ISC_TTS_READY_IND.

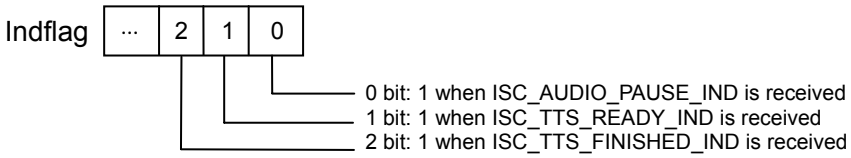
16 bytes of padding are sent after the command sequence is received.

[Input arguments]

iReceivedCounts Specifies the buffer data size when using the work-receiving buffer received by the func_*_REQ function. Specifies 0 in all other cases.

[Output arguments]

*IndFlag The bits shown below indicate the receipt status for ISC_AUDIO_PAUSE_IND/ISC_TTS_READY_IND and ISC_TTS_FINISHED_IND receipt status:



received_data[ ] Specifies the buffer when using the work-receiving buffer received by the func_*_REQ function. Specifies the receiving buffer in all other cases.
The data received indicates the command sequence. This includes the initial padding (0x00) and command header (0xAA), message ID, data length, and padding. The initial value of the sequence will therefore be the initial padding (0x00).

[Returned values]

Returns 0 if ISC_TTS_READY_IND is received normally.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FAUAL (-6):`	Response message received containing error code (fatal error)

3.2.35 func ISC_TTS_UDICT_DATA_REQ

[Format]

```
int     func_ISC_TTS_UDICT_DATA_REQ (
                                        uWord16         clear_udict,
                                        uWord8          *data_ptr,
                                        uWord8          aucReceivedData[],
                                        uWord16         length)
```

[Function]

Issues instruction to register the S1V30120 TTS playback dictionary.

The response is received by the func_ISC_XXX_RESP function.

16 bytes of padding are sent following the command sequence.

[Input arguments]

clear_udict 0x00000: Does not delete existing dictionary data.

0x00001: Deletes all existing dictionary data.

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.36 func_ISC_XXX_RESP

[Format]

```
int     func_ISC_XXX_RESP (     int         iMessageID,
                                int         iReceivedCounts,
                                uWord8      aucReceivedData[],
                                            uWord8      errData[])
```

[Function]

Receives the response for the data sent by the func_ISC_.....REQ function.

(Use the dedicated RESP function, if available, for func_ISC....REQ.)

16 bytes of padding are sent after the command sequence is received.

[Input arguments]

iMessageID	Message ID received
iReceivedCounts	Data size received by the func_ISC_SEQUENCER_PAUSE_REQ function.

[Output arguments]

received_data[ ]	received_data[ ] Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).
errData[]	Returns the data if a response contains an error code.

[Returned values]

Returns 0 if the same massage as that contained for the iMessageID is received.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.37 func_ISC_GPIO_REGISTER_REQ

[Format]

```
int     func_ISC_GPIO_REGISTER_REQ (
                                        uWord16     enable_registration,
                                        uWord8      received_data[ ])
```

[Function]

This function is used for S1V30120 GPIO control.

The response is received by the func_ISC_XXX_RESP function.

(The expected value of the received data is ISC_GPIO_REGISTER_RESP.)

16 bytes of padding are sent following the command sequence.

[Input arguments]

enable_registration 0x0000: de-register for control of GPIO I/F

0x0001: register for control of GPIO I/F

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_XXX_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.38 func_ ISC_GPIO_OUTPUT_CONFIG_REQ

[Format]

int func_ ISC_GPIO_OUTPUT_CONFIG_REQ (

uWord8 received_data[ ])

[Function]

This function is used in initialization settings to control S1V30120 GPIO output.

The response is received by the ISC_GPIO_OUTPUT_CONFIG_RESP function.

16 bytes of padding are sent following the command sequence.

[Input arguments]

None

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the ISC_GPIO_OUTPUT_CONFIG_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.39 func_ISC_GPIO_OUTPUT_CONFIG_RESP

[Format]

```
int     func_ISC_GPIO_OUTPUT_CONFIG_RESP (
                                          int       iReceivedCounts,
                                          uWord8    received_data[ ],
                                          int       *IndFlag,
                                          uWord8    **errData)
```

[Function]

Receives the response for the data sent by the func_ISC_GPIO_OUTPUT_CONFIG_REQ function.

16 bytes of padding are sent after the command sequence is received.

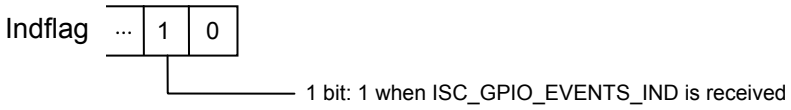
[Input arguments]

iReceivedCounts Data size received by the func_ISC_GPIO_OUTPUT_REQ function.

[Output arguments]

received_data[ ] Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

*IndFlag The ISC_GPIO_EVENTS_IND receipt status is indicated by the bit shown below:



**errData Indicates the data pointer if a response contains an error code.

[Returned values]

Returns 0 if func_ISC_GPIO_OUTPUT_CONFIG_RESP is received normally.

The following errors are issued if the value is less than 0:

RCV_SPI_TIMEOUT (-1):	Data reception timeout
RCV_ISC_ERROR_IND (-2):	Fatal error occurred in device
RCV_UNEXPECTED_ID (-3):	Unexpected data returned
RCV_BUFF_OVERFLOW (-4):	Receiving buffer overflow
RCV_DEVICE_ERROR (-5):	Response message received containing error code (non-fatal error)
RCV_DEVICE_FAUAL (-6):	Response message received containing error code (fatal error)

3.2.41 func_ISC_GPIO_OUTPUT_SET_REQ

[Format]

```
int     func_ISC_GPIO_OUTPUT_SET_REQ (
                                        uWord32     output_value,
                                        uWord8      received_data[ ])
```

[Function]

Use this function when switching S1V30120 GPIO output between High and Low.

The response is received by the func_ISC_GPIO_OUTPUT_SET_RESP function.

16 bytes of padding are sent following the command sequence.

[Input arguments]

output_value
- Bit[0]: set to 0
- Bit[1]: set to 0
- Bit[2]: set to 0
- Bit[3]: set to 0
- Bit[4]: set to 0
- Bit[5]: set to the value for GPIO5
- Bit[6]: set to the value for GPIO6
- Bit[7]: set to the value for GPIO7
- Bit[8]: set to the value for GPIO8
- Bit[9]: set to the value for GPIO9
- Bit[10]: set to the value for GPIO10
- Bit[11]: set to the value for GPIO11

[Output arguments]

received_data[ ] Specifies work-receiving buffer. This receiving buffer and received data size are transferred as the received data is processed by the func_ISC_GPIO_OUTPUT_SET_RESP function.

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow
`RCV_DEVICE_ERROR (-5):`	Response message received containing error code (non-fatal error)
`RCV_DEVICE_FATAL (-6):`	Response message received containing error code (fatal error)

3.2.42 func_ISC_GPIO_OUTPUT_SET_RESP

[Format]

```
int     func_ISC_GPIO_OUTPUT_SET_RESP (
                                        int         iReceivedCounts,
                                        uWord8      received_data[ ],
                                        uWord8      **errData)
```

[Function]

Receives the response for the data sent by the func_ISC_GPIO_OUTPUT_SET_REQ function.

16 bytes of padding are sent after the command sequence is received.

[Input arguments]

iReceivedCounts	Data size received by the func_ISC_GPIO_OUTPUT_SET_REQ function.

[Output arguments]

received_data[ ]	Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).
**errData	Indicates the data pointer if a response contains an error code.

[Returned values]

Returns 0 if func_ISC_GPIO_OUTPUT_SET_RESP is received normally.

The following errors are issued if the value is less than 0:

RCV_SPI_TIMEOUT (-1):	Data reception timeout
RCV_ISC_ERROR_IND (-2):	Fatal error occurred in device
RCV_UNEXPECTED_ID (-3):	Unexpected data returned
RCV_BUFF_OVERFLOW (-4):	Receiving buffer overflow
RCV_DEVICE_ERROR (-5):	Response message received containing error code (non-fatal error)
RCV_DEVICE_FAUAL (-6):	Response message received containing error code (fatal error)

3.3 SPI Control API Specifications

Described below are the program API specifications used by these sample programs for SPI control (those defined by “spi_api.c”).

*** The APIs described below are control programs based on an SPI on the host system used by Seiko Epson for S1V30120 control evaluation. Modifications to customer specifications are needed before incorporation into the customer's system.**

3.3.1 SPI_initialise

[Format]

void SPI_initialise (void)

[Function]

Initializes the SPI registers.

This API sets the following settings based on the SPI specifications for the host processor used by the Seiko Epson evaluation system.

(1) Prohibits SPI interrupts for settings.

(2) Sets GPIO.

(3) Sets the SPI clock frequency.

(4) Sets SPI master mode transmission and receipt.

(5) Sets the wait cycles between data transfers.

(6) Sets the received data mask.

(7) Sets interrupt prohibition for transmission data empty, received data overflow, and received data full.

(8) Permits SPI interrupt.

[Input arguments]

None

[Output arguments]

None

[Returned values]

None

* For detailed information, refer to “Appendix B Typical SPI Register Specifications.”

3.3.2 SPI_transfer_commands

[Format]

```
int    SPI_transfer_commands    (  uWord8    *transfer_data,
                                   int        transfer_length,
                                   uWord8     received_data[ ])
```

[Function]

Sends data to the S1V30120 via the SPI.

Inputs all data to be sent (including padding and command headers), since command analysis is not performed inside the API.

Data is received by the SPI_ReceiveCommands function, but the work-receiving buffer is used, since multiple continuous commands may be received when commands are sent.

[Input arguments]

*transfer_data	Indicates the data pointer to be sent.
transfer_length	Indicates the data length to be sent.

[Output arguments]

received_data[ ]	Indicates the command sequence received in the work-receiving buffer. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow

3.3.3 SPI_ReceiveCommands

[Format]

int SPI_ReceiveCommands (uWord8 received_data[ ])

[Function]

Receives the response for the data sent by the SPI_transfer_commands function.

[Input arguments]

None

[Output arguments]

received_data[ ] Specifies the command sequence received. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

[Returned values]

The received data size if 0 or more.

The following errors are given if less than 0.

RCV_SPI_TIMEOUT (-1):	Data reception timeout
RCV_ISC_ERROR_IND (-2):	Fatal error occurred in device
RCV_UNEXPECTED_ID (-3):	Unexpected data returned
RCV_BUFF_OVERFLOW (-4):	Receiving buffer overflow

*** The command stored in the SPI_transfer_commands work-receiving buffer must be processed if data is stored in the buffer before processing the command received by the SPI_ReceiveCommands function.**

3.3.4 SPI_TransferPaddingData

[Format]

```
int    SPI_TransferPaddingData  (  uWord16     transfer_length,
                                   uWord8      aucReceivedData[])
```

[Function]

Sends padding data to the S1V30120 via the SPI.

[Input arguments]

transfer_length Indicates the padding data length to be sent. (Data is 0x00.)

[Output arguments]

received_data[ ] Specifies the command sequence received in the work-receiving buffer. This includes the initial padding (0x00), command header (0xAA), message ID, data length, and padding. Thus, the initial value for the sequence will be the initial padding (0x00).

[Returned values]

The received data size stored in the work-receiving buffer if 0 or more.

The following errors are issued if the value is less than 0:

`RCV_SPI_TIMEOUT (-1):`	Data reception timeout
`RCV_ISC_ERROR_IND (-2):`	Fatal error occurred in device
`RCV_UNEXPECTED_ID (-3):`	Unexpected data returned
`RCV_BUFF_OVERFLOW (-4):`	Receiving buffer overflow

Appendix A Binary File Conversion Tool

The S1V30120 demonstration kit includes a tool ("bin2text.exe") for converting binary files into ASCII format capable of being read as C source files.

This tool is designed to convert the ADPCM binary files generated by the speech generation authoring tool provided in the demonstration kit into ASCII format capable of being read as C source files to provide a setup readily incorporated into customer systems.

This tool is provided at the following location in the S1V30120 demonstration kit:

\utility

The binary format speech data (adpcm_data) is provided at the following location:

\adpcm_data

<Usage directions>

Type the following at the command prompt:

\utility> bin2text ..\adpcm_data\1_Hello.adpcm adpcm.c

This allows creation of ASCII format "adpcm.h" from the binary format "hello.adpcm." "boot.h" is arranged as shown below. Data is output in little-endian format.

```
--------------------------------------------------------------------------------
unsigned char /* PLEASE FILL THE NAME OF ARRAY HERE ! */ []={

0x18,0xf0,0x9f,0xe5,0x18,0xf0,0x9f,0xe5,0x18,0xf0,0x9f,0xe5,0x18,0xf0,0x9f,0xe5,

0x18,0xf0,0x9f,0xe5,0x00,0x00,0xa0,0xe1,0x14,0xf0,0x9f,0xe5,0x14,0xf0,0x9f,0xe5,
     . . . . .
     . . . . .
};

int /* PLEASE FILL THE NAME OF NUMBER FOR ARRAY HERE ! */ = 7152;
--------------------------------------------------------------------------------
```

The file generated above includes the data sequence and data size (bytes). The data sequence and data size declaration names are included as comments. These declaration names can be replaced by names specified by the customer to enable ready data incorporation into the system as part of the C source file.

Appendix B Typical SPI Register Specifications

The SPI register specifications listed below are extracted from the host processor register specifications used by these sample programs.

Table V.2.7.1 SPI control register list

Address	Register	Size	Function
0x00301700	SPI Receive Data Register (pSPI_RXD)	32	Received data
0x00301704	SPI Transmit Data Register (pSPI_TXD)	32	Transmitted data
0x00301708	SPI Control Register 1 (pSPI_CTL1)	32	SPI transfer condition setting
0x0030170C	SPI Control Register 2 (pSPI_CTL2)	32	Slave mode control
0x00301710	SPI Wait Register (pSPI_WAIT)	32	Wait cycle between character setting
0x00301714	SPI Status Register (pSPI_STAT)	32	SPI transfer/error status
0x00301718	SPI Interrupt Control Register (pSPI_INT)	32	SPI interrupt control
0x0030171C	SPI Receive Data Mask Register (pSPI_RXMK)	32	Received data bit mask setting

The SPI control registers are discussed individually below.

The SPI control registers are assigned to the 32-bit device area 0x301700 to 0x30171C. Word access is possible.

Note:

- SPI control registers are valid for word access only. Do not read or write using half-word or byte-size.
- When setting the SPI control registers, always write 0 to “Reserved” bits, not 1.

0x301700: SPI Receive Data Register (pSPI_RXD)

Register name	Address	Bit	Name	Function	Setting	Init.	R/W	Remarks
SPI receive data register (pSPI_RXD)	00301700 (W)	D31 | D0	SPIRXD31 | SPIRXD0	SPI receive data SPIRXD31 = MSB SPIRXD0 = LSB	0x0 to 0xFFFFFFFF	0x0	R	

D[31: 0] SPIRXD[31: 0]: SPI Receive Data Bits

Comprise the received data. (Default: 0x0)

RDFF (D2/0x301714) is set to 1 (data full) once the data is received and the shift register data has been transferred to this register. A received data full interrupt request is generated simultaneously. Data can subsequently be read until all next data is received. If the next data is received before this register is read, it is overwritten by the new data received, and RDOF (D3/0x301714) is set to 1 (data overflow). A received data overflow interrupt request is generated simultaneously.

The serial data input from the SDI terminal is converted to parallel as MSB with the high level bit as 1 and the low level bit as 0, then loaded into this register.

The specified number of high-order bits can be masked (0) when loading from the shift register using the SPI Receive Data Mask Register (0x30171C) settings.

This register is for reading only and must not be written to.

0x301704: SPI Transmit Data Register (pSPI_TXD)

Register name	Address	Bit	Name	Function	Setting	Init.	R/W	Remarks
SPI transmit data register (pSPI_TXD)	00301704 (W)	D31 | D0	SPITXD31 | SPITXD0	SPI transmit data SPITXD31 = MSB SPITXD0 = LSB	0x0 to 0xFFFFFFFF	0x0	R	

D[31: 0] SPITXD[31: 0]: SPI Transmit Data Bits

Set the transmitted data. (Default: 0x0)

In master mode, transmission is initiated by writing data to this register. In slave mode, transmission is initiated when the clock is input from the master and register details are sent to the shift register.

TDEF (D4/0x301714) is set to 1 (empty) once the data written to this register has been transferred to the shift register. A transmitted data empty interrupt request is also generated simultaneously.

The next transmission data can be subsequently written even while data is being sent. The data converted to serial from the SDO terminal is output with MSB first and with the bit set to 1 as high level and the bit set to 0 as low level.

If the transfer data bit quantity is set to less than 32 in BPT[4: 0] (D[14: 10]/0x301708), only the specified number of low-order bits of the register are sent.

0x301708: SPI Control Register 1 (pSPI_CTL1)

Register name	Address	Bit	Name	Function	Setting	Init	R/W	Remarks
SPI control register 1 (pSPI_CTL1)	00301708 (W)	D31 | D15	-	reserved	-	-	-	0 when being read.
		D14 D13 D12 D11 D10	BPT4 BPT3 BPT2 BPT1 BPT0	Number of data bits per transfer	Number of data bits per transfer = BPT + 1	0	R/W	
		D9	CPHA	SPI_CLK phase selection	1 Phase 1 / 0 Phase 0	0	R/W	
		D8	CPOL	SPI_CLK polarity selection	1 Active low / 0 Active high	0	R/W	
		D7	MWEN	reserved	Fix at 0.	0	-	
		D6 D5 D4	MCBR2 MCBR1 MCBR0	Master clock bit rate (in master mode only)	Master clock divided value = 4×2^{MCBR}	0	R/W	
		D3	TXDE	Transmit DMA enable	1 Enabled / 0 Disabled	0	R/W	
		D2	RXDE	Receive DMA enable	1 Enabled / 0 Disabled	0	R/W	
		D1	MODE	SPI mode selection	1 Master / 0 Slave	0	R/W	
		D0	ENA	SPI enable	1 Enabled / 0 Disabled	0	R/W	

D[31: 15] Reserved

D[14: 10] BPT[4: 0]: Number of Data Bits Per Transfer Setup Bits

Sets the number of transfer data bits. (Default: 0x0)

This register setting +1 (1 to 32) represents the number of bits sent and received in a single transfer.

D9 CPHA: SPI_CLK Phase Select Bit

Selects the SPI clock phase. (Default: 0)

Sets data transfer timing together with CPOL (D8). (See Figure V.2.7.1.)

D8 CPOL: SPI_CLK Polarity Select Bit

Selects the SPI clock polarity.

1 (R/W): Active low

0 (R/W): Active high (Default)

Sets data transfer timing together with CPHA (D9). (See Figure V.2.7.1.)

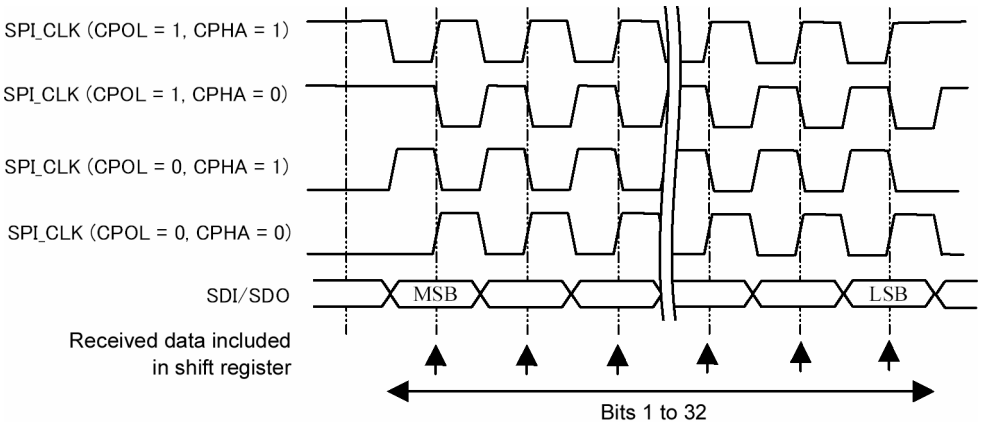


Figure V.2.7.1 Clock and data transfer timing

D7 **Reserved** (Do not write 1.)

D[6: 4] MCBR[2: 0]: Master Clock Bit Rate Setup Bits

Set the source clock division ratio for generating the SPI clock. This setting is used to determine the bit rate.

Table V.2.7.2 Clock frequency settings

MCBR2	MCBR1	MCBR0	Clock frequency (Hz)
1	1	1	MCLK/512
1	1	0	MCLK/256
1	0	1	MCLK/128
1	0	0	MCLK/64
0	1	1	MCLK/32
0	1	0	MCLK/16
0	0	1	MCLK/8
0	0	0	MCLK/4

(Default: 0x000)

There is no need to set the bit rate in slave mode, since the clock is input from the master.

D3 XDE: Transmit DMA Enable Bit

Permits or prohibits transmission DMA interrupts.

1 (R/W): Permitted

0 (R/W): Prohibited (Default)

Setting TXDE to 1 permits the output of transmission DMA interrupt requests to the ITC. Transmission DMA interrupt requests occur when data written to the SPI Transmit Data Register (0x301704) is transferred to the shift register (when transmission starts). If TXDE is set to 1 (Permit), the ITC interrupt request flag FSPITX (D5/0x300289) is set to 1 at that point. This interrupt request can also initiate HSDMA.

Transmission DMA interrupts do not occur if TXDE is set to 0.

D2 RXDE: Receive DMA Enable Bit

Permits or prohibits receipt DMA interrupts.

1 (R/W): Permitted

0 (R/W): Prohibited (Default)

Setting RXDE to 1 permits the output of receipt DMA interrupt requests to the ITC. Receipt DMA interrupt requests occur when data received by the shift register is transferred to the SPI Receive Data Register (0x301700) (when receipt is complete). If RXDE is set to 1 (Permit), the ITC interrupt request flag FSPIRX (D4/0x300289) is set to 1 at that point. This interrupt request can also initiate HSDMA.

Receipt DMA interrupts do not occur if RXDE is set to 0.

D1 MODE: SPI Mode Select Bit

Sets the SPI to master mode or slave mode.

1 (R/W): Master mode

0 (R/W): Slave mode (Default)

Setting MODE to 1 selects master mode while setting to 0 selects slave mode. In master mode, data is transferred using the clock generated within this module. In slave mode, data is transferred with the clock input from the master.

D0 ENA: SPI Enable Bit

Permits or prohibits SPI module operation.

1 (R/W): Permit (On)

0 (R/W): Prohibit (Off) (Default)

Setting ENA to 1 starts SPI module operation and allows data transfer.

Setting ENA to 0 stops SPI module operation.

ENA should be set to 0 before setting data transfer conditions.

0x30170C: SPI Control Register 2 (pSPI_CTL2)

Register name	Address	Bit	Name	Function	Setting	Init.	R/W	Remarks
SPI control register 2 (pSPI_CTL2)	0030170C (W)	D31-12	-	reserved	-	-	-	0 when being read.
		D11	SSA	reserved	Fix at 0.	0	-	
		D10	SS	Slave select control	Fix at 0.	0	R/W	Master mode
					1 SPI select / 0 SPI deselect			Slave mode
		D9	SSP	reserved	Fix at 0.	0	-	Slave mode
		D8	SSC	reserved	Fix at 0.	0	-	
		D7-3	-	reserved	-	-	-	0 when being read.
		D2	RDYP	reserved	Fix at 0.	0	-	
		D1	RDYS	reserved	Fix at 0.	0	-	
		D0	RDYE	reserved	Fix at 0.	0	-	

D[31: 11] Reserved (Do not write 1.)

D10 SS: Slave Select Control Bit

Selects the slave.

1 (R/W): Selected

0 (R/W): Not selected (Default)

Set SS to 1 before sending and receiving in slave mode. If ENA = 1 and SS = 1, clock input from the master is enabled, allowing sending and receiving in slave mode.

SS should be fixed at 0 in master mode.

D[9: 0] Reserved (Do not write 1.)

0x301710: SPI Wait Register (pSPI_WAIT)

Register name	Address	Bit	Name	Function	Setting	Init.	R/W	Remarks
SPI wait register (pSPI_WAIT)	00301710 (W)	D31 - D0	SPIW31 | SPIW0	Wait cycle control SPIW31 = MSB SPIW0 = LSB	Number of wait cycles = SPIW[31: 0] + 1 (1 to 65536)	0x0	R/W	

D[31: 0] SPIW[31: 0]: Wait Cycle Control Bits

Set the number of wait cycles to be inserted between separate data transfers. The value of this register +1 forms the number of wait cycles. It can be specified within the SPI_CLK range of 1 to 65,536 cycles.

0x301714: SPI Status Register (pSPI_STAT)

Register name	Address	Bit	Name	Function	Setting				Init.	R/W	Remarks
SPI status register (pSPI_STAT)	00301714 (W)	D31-7	-	reserved	-				-	-	0 when being read.
		D6	BSYF	Transfer busy flag	1	Busy	0	Idle	0	R	Master mode
		D5	MFEF	reserved	-				-	-	0 when being read.
		D4	TDEF	Transmit data empty flag	1	Empty	0	Not empty	1	R	
		D3	RDOF	Receive data overflow flag	1	Occurred	0	Not occurred	0	R	
		D2	RDFF	Receive data full flag	1	Full	0	Not full	0	R	
		D1-0	-	reserved	-				-	-	0 when being read.

D[31: 7] Reserved

D6 BSYF: Transfer Busy Flag

Indicates that the SPI is transmitting or receiving data. (In master mode)

1 (R): Transmitting/receiving

0 (R): Standby (Default)

BSYF is set to 1 when the SPI begins transmitting or receiving in master mode and remains at 1 while transmitting or receiving data, including during wait cycles. It is cleared to 0 once transmitting and receiving has ended.

BSYF is disabled (always 0) in slave mode.

D5 Reserved

D4 TDEF: Transfer Data Empty Flag

Indicates the SPI Transmit Data Register (0x301704) status.

1 (R): Empty (Default)

0 (R): Data present

TDEF is set to 0 when transmission data is written to the SPI Transmit Data Register (0x301704) and is set to 1 when the data is transferred to the shift register (when transmission starts).

Transmission data is written when this bit is 1.

D3 RDOF: Receive Data Overflow Flag

Indicates the received data overflow status.

1 (R): Overflow

0 (R): No overflow (Default)

RDOF is set to 1 when the next data is received before the received data is read in the SPI Receive Data Register (0x301700), indicating that the register has been overwritten.

It is returned to 0 when the data in the SPI Receive Data Register (0x301700) is read out.

D2 RDFF: Receive Data Full Flag

Indicates the SPI Receive Data Register (0x301700) status.

1 (R): Data full

0 (R): No data (Default)

RDFF is set to 1 when the data received in the shift register is transferred to the SPI Receive Data Register (0x301700) (when all data is received), indicating that the data can be read. It is returned to 0 when the data is read out.

D[1: 0] Reserved

0x301718: SPI Interrupt Control Register (pSPI_INT)

Register name	Address	Bit	Name	Function	Setting				Init.	R/W	Remarks
SPI interrupt control register (pSPI_INT)	00301718 0 (W)	D31-6	-	reserved	-				-	-	0 when being read.
		D5	MFIE	reserved	Fix at 0.				0	-	
		D4	TEIE	Transmit data empty int. enable	1	Enabled	0	Disabled	0	R/W	
		D3	ROIE	Receive overflow interrupt enable	1	Enabled	0	Disabled	0	R/W	
		D2	RFIE	Receive data full interrupt enable	1	Enabled	0	Disabled	0	R/W	
		D1	MIRQ	Manual IRQ set/clear	1	Set	0	Clear	0	R/W	
		D0	IRQE	Interrupt request enable	1	Enabled	0	Disabled	0	R/W	

D[31: 5] Reserved (Do not write 1.)

D4 TEIE: Transmit Data Empty Interrupt Enable Bit

Permits or prohibits SPI interrupts for transmission data empty.

1 (R/W): Permitted

0 (R/W): Prohibited (Default)

Setting TEIE to 1 permits SPI interrupt requests to be output to the ITC for transmission data empty. These interrupt requests occur when data written to the SPI Transmit Data Register (0x301704) is transferred to the shift register (when transmission starts). The ITC interrupt request flag FP8 (D0/0x3002A9) is set to 1 as soon as TEIE and IRQE (D0) are set to 1 (Permit).

SPI interrupts do not occur for transmission data empty if TEIE is set to 0.

D3 ROIE: Receive Data Overflow Interrupt Enable Bit

Permits or prohibits SPI interrupts for received data overflows.

1 (R/W): Permitted

0 (R/W): Prohibited (Default)

Setting ROIE to 1 permits SPI interrupt requests to be output to the ITC for received data overflows. These interrupt requests occur when the next received data is loaded before the received data in the SPI Receive Data Register (0x301700) is read out. The ITC interrupt request flag FP8 (D0/0x3002A9) is set to 1 as soon as ROIE and IRQE (D0) are set to 1 (Permit).

SPI interrupts do not occur for received data overflows if ROIE is set to 0.

D2 RFIE: Receive Data Full Interrupt Enable Bit

Permits or prohibits SPI interrupts for received data full.

1 (R/W): Permitted

0 (R/W): Prohibited (Default)

Setting RFIE to 1 permits SPI interrupt requests to be output to the ITC for received data full.

These interrupt requests occur when the data received in the shift register is transferred to the SPI Receive Data Register (0x301700) (when all data is received). The ITC interrupt request flag FP8 (D0/0x3002A9) is set to 1 as soon as RFIE and IRQE (D0) are set to 1 (Permit).

SPI interrupts do not occur for received data full if RFIE is set to 0.

D1 MIRQ: Manual IRQ Set/Clear Bit

Generates a manual SPI interrupt request to the ITC.

1 (R/W): Sets interrupt request

0 (R/W): Clears interrupt request (Default)

Setting MIRQ to 1 when IRQE (D0) is 1 enables SPI interrupt requests to the ITC and sets the ITC interrupt request flag FP8 (D0/0x3002A9) to 1.

Writing 0 to MIRQ clears the interrupt request. Note that writing 0 does not clear the interrupt request flag FP8 (D0/0x3002A9).

D0 IRQE: Interrupt Request Enable Bit

Permits or prohibits SPI interrupt request output to the ITC.

1 (R/W): Permitted

0 (R/W): Prohibited (Default)

Setting IRQE to 1 outputs an interrupt request to the ITC as soon as the permitted SPI interrupt request occurs or 1 is written to the MIRQ (D1). This sets the interrupt request flag FP8 (D0/0x3002A9) to 1.

If IRQE is set to 0, interrupt requests are not issued to the ITC, even if SPI interrupt requests are individually permitted. Manual interrupt requests using MIRQ (D1) are also prohibited.

0x30171C: SPI Receive Data Mask Register (pSPI_RXMK)

Register name	Address	Bit	Name	Function	Setting	Init	R/W	Remarks
SPI receive data mask register (pSPI_RXMK)	0030171C (W)	D31-15	-	reserved	-	-	-	0 when being read.
		D14 D13 D12 D11 D10	RXMASK4 RXMASK3 RXMASK2 RXMASK1 RXMASK0	Bit mask for reading received data	0x0 to 0x1F	0 0 0 0 0	R/W	
		D9-2	-	reserved	-	-	-	0 when being read.
		D1	RXME	Receive data mask enable	1 Enabled 0 Disabled	0	R/W	
		D0	-	reserved	-	-	-	Do not write 1.

D[31: 15] Reserved

D[14: 10] RXMASK[4: 0]: Receive Data Mask Setup Bits

Specifies the low-order bits to enable when reading received data by masking all except the required low-order bits. (Default: 0x0)

The value set is the MSB of the enabled bit. (E.g. 31 = no mask, 15 = mask D[31: 16])

RXME (D1) must be set to 1 to enable this bit mask. Enabling the bit mask allows the received data to be read out from the SPI Receive Data Register (0x301700) with the masked bits set to 0.

D[9: 2] Reserved

D1 RXME: Receive Data Mask Enable Bit

Enables the RXMASK[4: 0] (D[14: 10]) setting.

1 (R/W): Enabled

0 (R/W): Disabled (Default)

Setting RXME to 1 masks the high-order bits (sets to 0) as specified by RXMASK[4: 0] when loading received data to the SPI Receive Data Register (0x301700) from the received data buffer. Setting RXME to 0 masks all bits except the data bits (low-order bits) specified by BPT[4: 0] (D[14: 10]/0x301708) when data received in the shift register is loaded to the SPI Receive Data Register (0x301700).

Figure V.2.7.2 illustrates the relationship between the mask control bit settings and the received data loaded to the SPI Receive Data Register (0x301700).

D0 Reserved

Do not set to 1.

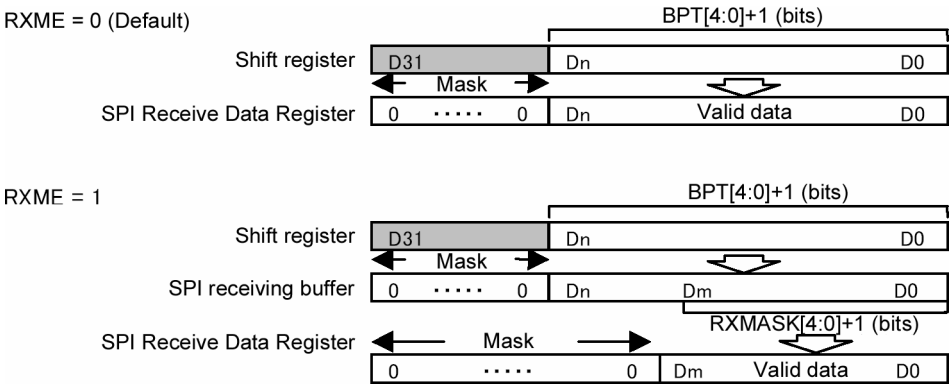


Figure V.2.7.2 Received data mask

V.2.8 Caution Points

- Always use 32-bit access commands for SPI control register (0x301700 to 0x30171C) reading and writing. Never use 16-bit or 8-bit access commands.
- Do not access SPI Control Register 1 (0x301708), SPI Control Register 2 (0x30170C), or SPI Wait Register (0x301710) while BSYF (D6/0x301714) is 1 (while data is being transferred).

  * **.BSYF**: Transfer Busy Flag in the SPI Status Register (D6/0x301714)

- To prevent malfunctions, always write 0x0 to the SPI InterruptControl Register (0x301718) to block all SPI interrupt requests before stopping the SPI circuit (setting ENA (D0/0x301708) to 0).

  * **.ENA:** SPI Enable Bit in the SPI Control Register 1 (D0/0x301708)

Revision History

Date	Revision details			
	Rev.	**Page**	**Type**	**Details**
10/12/2007	1.00	All	New	Newly established
12/10/2007	1.01	All	Update	Deleted ISC_AUDIO_PAUSE_IND and ISC_GPIO_EVENT_IND

EPSON

International Sales Operations

AMERICA

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

SALES OFFICES
Northeast
301 Edgewater Place, Suite 210
Wakefield, MA 01880, U.S.A.
Phone: +1-800-922-7667 FAX: +1-781-246-5443

EUROPE

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

ASIA

EPSON (CHINA) CO., LTD.
23F, Beijing Silver Tower 2# North RD DongSanHuan
ChaoYang District, Beijing, CHINA
Phone: +86-10-6410-6655 FAX: +86-10-6410-7320

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

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 Electronic Technology Development (Shenzhen) LTD.
12/F, Dawning Mansion, Keji South 12th Road,
Hi- Tech Park, Shenzhen
Phone: +86-755-2699-3828 FAX: +86-755-2699-3838

EPSON TAIWAN TECHNOLOGY & TRADING LTD.
14F, No. 7, Song Ren Road,
Taipei 110
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 CORPORATION
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

GUMI OFFICE
2F, Grand B/D, 457-4 Songjeong-dong,
Gumi-City, KOREA
Phone: +82-54-454-6027 FAX: +82-54-454-6093

SEIKO EPSON CORPORATION
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: 411297000
First Issue November 2007
Printed December 2007 in JAPAN C