RL78/G10

Wi-Fi module (ESP-WROOM-02) control sample software for TCP/IP Slave Transmission/Reception

Introduction

This application note describes how to use the software for controlling the Wi-Fi module ESP-WROOM-02 through the RL78/G10 and its sample application program.
The ESP-WROOM-02 supports AT commands. The RL78/G10 sends AT commands through UART communication to control the ESP-WROOM-02.
Using this control software enables TCP/IP client transmission and reception of data.

Target Device

ESP-WROOM-02 (ESP8266EX) manufactured by ESPRESSIF SYSTEMS CO., LTD.

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Contents

1. Specifications

The RL78/G10 controls the Wi-Fi module ESP-WROOM-02 through UART communication.
The sample program provided together with this application note connects the Wi-Fi module to a server as a TCP/IP client and transmits and receives data.
Table 1.1 gives an outline of the system functions and Figure 1.1 shows an example of system configuration.

Table 1.1 Outline of System Functions

Wi-Fi module Specification	Wireless LAN standard IEEE 802.11 b/g/n (2.4GHz Wi-Fi)
	Supported protocols IPv4 TCP Security: WPA/WPA2 Encryption: AES
Connection between Wi-Fi module and MCU	Communication method Master: RL78/G10 from Renesas Electronics Slave: ESP-WROOM-02 (ESP8266EX) from Espressif Systems Communication mode: UART at 115200 bps Module control Control through AT commands (UART communication) Used in the Station mode Connection and disconnection with an access point Data translation and reception through the network as a client of TCP communication

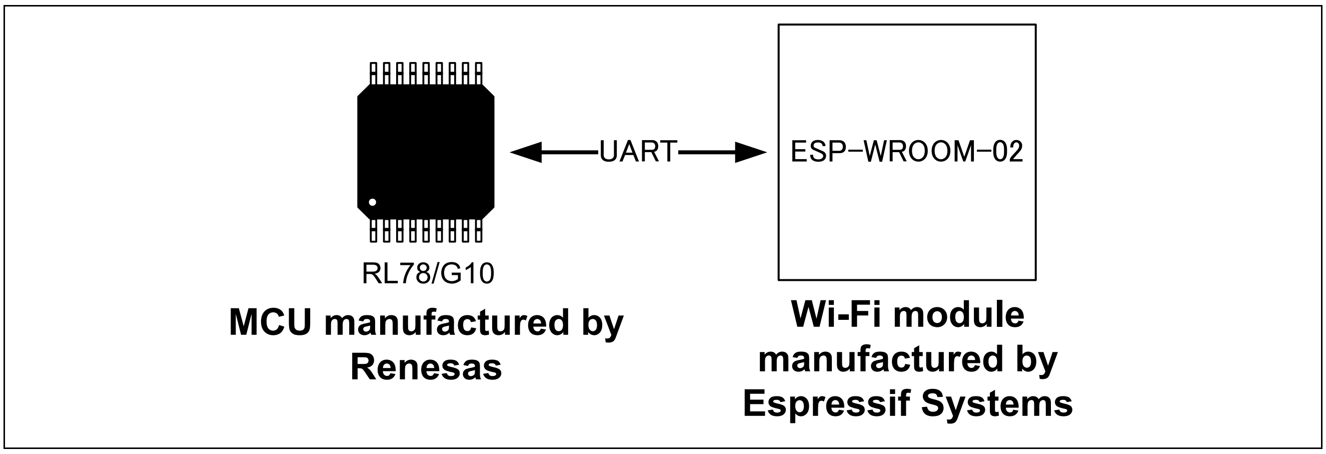

UART
ESP−WROOM−02
RL78/G10
MCU manufactured by Renesas
Wi-Fi module manufactured by Espressif Systems

Figure 1.1 Example of System Configuration

1.1 Overview of Wi-Fi Module ESP-WROOM-02 (ESP8266EX) Operation

1.1.1 Boot Mode

The ESP-WROOM-02 provides two boot modes: Flash Boot Mode for executing a program stored in the flash memory and UART Download Mode for writing a program to the flash memory.
The example in this application note only uses the Flash Boot Mode to execute the official firmware stored in advance in the flash memory. The boot mode is selected by pin setting when the Wi-Fi module is reset. In the example in this application note, set up the pins as shown in Table 1.2.

Table 1.2 Boot Mode Selection

Boot Mode	GPIO15	GPIO2	GPIO0
Flash Boot Mode	L	H	H

1.1.2 Control Method

The ESP-WROOM-02 is controlled by AT commands through UART communication. Executing API functions of the provided control software enables transmission of AT commands to the Wi-Fi module and reception of result codes.
The formats of AT commands (instructions to be sent to the module) and result codes (responses from the module) in the ESP-WROOM-02 (ESP8266EX) are shown below.

AT Command:

A	T	Command	Parameters	CR	LF

Result Code:

CR	LF	Result Code	CR	LF

Note: For details of the AT commands for the ESP-WROOM-02, refer to the ESP8266EX AT Instruction Set.

1.1.3 Operation Mode

The ESP-WROOM-02 provides three operating modes: Station, SoftAP, and SoftAP + Station Modes.
In the Station mode, the Wi-Fi module works as a wireless client, which is usually connected to an access point. In the SoftAP mode, the Wi-Fi module works as an access point (base station).
This application note assumes that the Wi-Fi module is connected to an access point and describes the Station mode only.

2. Conditions of Operation Confirmation Test

The sample code with this application note runs properly under the conditions below.

Table 2.1 Operation Confirmation Conditions

Items	Contents
MCU used	RL78/G10 (R5F10Y47) ROM: 4 KB, RAM: 512 B
Operating frequencies	• High-speed on-chip oscillator clock (fIH): 20 MHz • CPU/peripheral hardware clock: 20 MHz
Operating voltage	3.3 V
Integrated development environment (CS+)	Renesas Electronics Corporation CS+ for CC V8.07.00
C compiler (CS+)	Renesas Electronics Corporation CC-RL V1.11.00
Integrated development environment (e^2 studio)	Renesas Electronics Corporation e^2 studio V2022-01 (22.1.0)
C compiler (e^2 studio)	Renesas Electronics Corporation CC-RL V1.11.00
Integrated development environment (IAR)	IAR Systems IAR Embedded Workbench for Renesas RL78 V4.21.3
C compiler (IAR)	IAR Systems IAR C/C++ Compiler for Renesas RL78 V4.21.3.2447
Wi-Fi Module	ESP-WROOM-02 (ESP8266EX) manufactured by Espressif Systems AT version:1.3.0.0 SDK version: 2.0.0

Note 1: Before building the project for R5F10Y47 on IAR Embedded Workbench, rewrite the linker configuration file partly as below.
Target configuration file:
{IAR Embedded Workbench installation directory}\rl78\config\lnkr5f10y47.icf
Details of change:
Setting of RAM address area
From:

```
define region RAM_near = mem:[from 0xFFCE0 to 0xFFE1F];
define region RAM_far  = mem:[from 0xFFCE0 to 0xFFE1F];
define region RAM_huge = mem:[from 0xFFCE0 to 0xFFE1F];
```

To:

```
define region RAM_near = mem:[from 0xFFCE0 to 0xFFEDF];
define region RAM_far  = mem:[from 0xFFCE0 to 0xFFEDF];
define region RAM_huge = mem:[from 0xFFCE0 to 0xFFEDF];
```

3. Hardware

3.1 Example of Hardware Configuration

Figure 3.1 shows an example of the hardware configuration used in this application note.

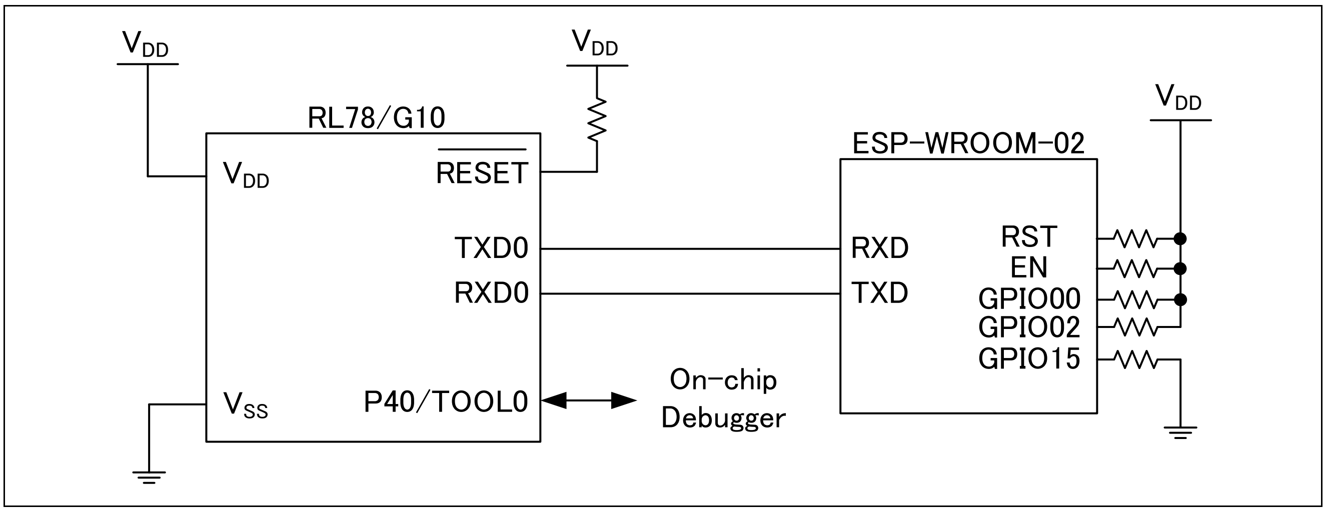


Figure 3.1 Hardware Configuration

Note 1: This simplified circuit diagram was created to show an overview of connections only.
When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements. (Connect each input-only port to V_{DD} or V_{SS} through a resistor.)

3.2 Used Pins

Table 3.1 shows list of used Pins and assigned functions.

Table 3.1 List of Pins and Functions

Pin Name	Input/Output	Function
TXD0	Output	Serial Data Transmission (UART)
RXD0	Input	Serial Data Reception (UART)

4. Software Explanation

4.1 Example of Software Configuration

Figure 4.1 shows an example of software configuration using the control software.

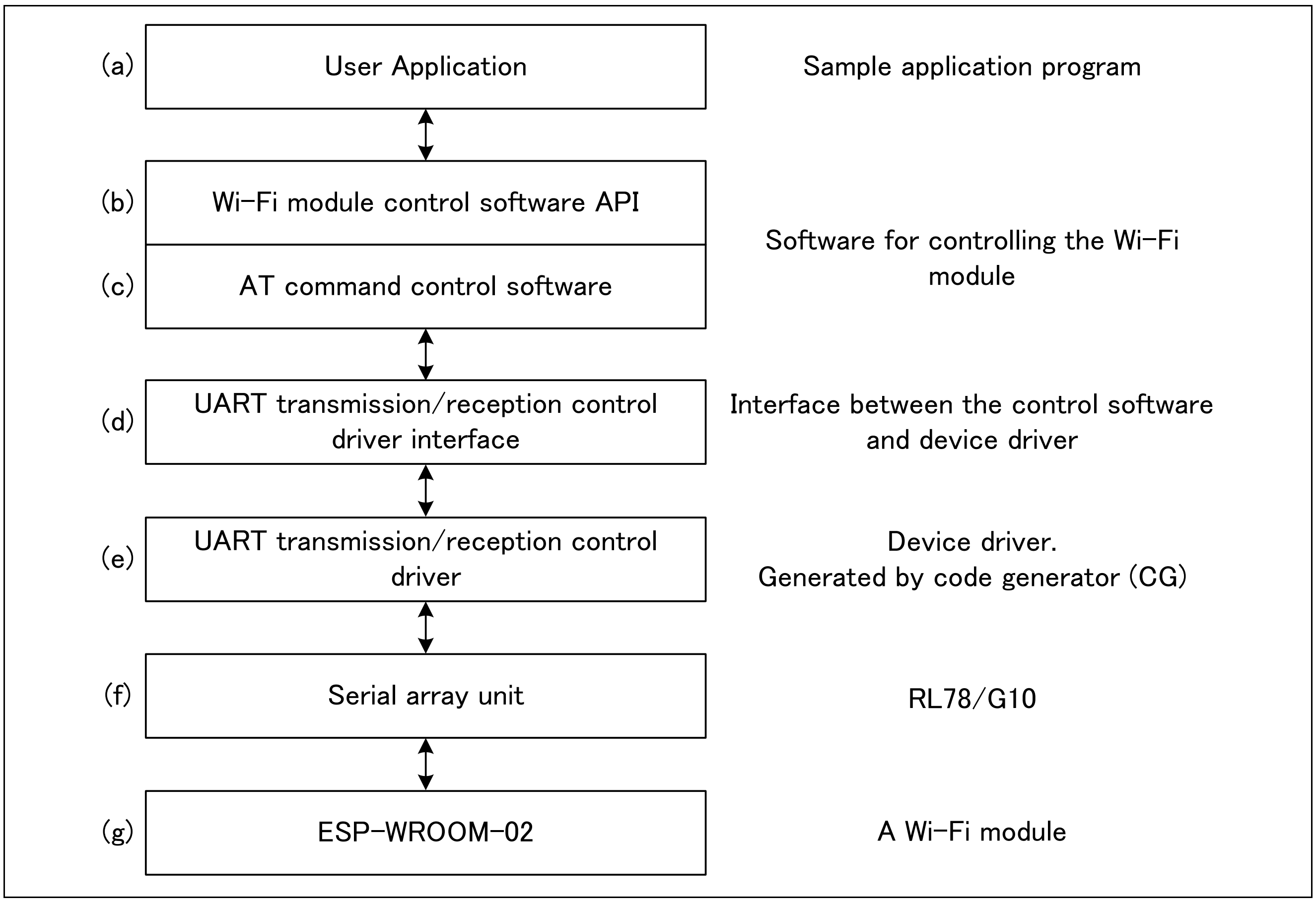

(a)
User Application
Sample application program
(b)
Wi-Fi module control software API
(c)
AT command control software
Software for controlling the Wi-Fi module
(d)
UART transmission/reception control driver interface
Interface between the control software and device driver
(e)
UART transmission/reception control driver
Device driver. Generated by code generator (CG)
(f)
Serial array unit
RL78/G10
(g)
ESP-WROOM-02
A Wi-Fi module

Figure 4.1 Example of Software Configuration

(a) User application

An application program that calls API functions of the control software. A sample application program is provided together with the control software.

(b) Wi-Fi module control software API

API of the control software.

(c) AT command control software

Software for controlling UART transmission and reception between RL78/G10 and the Wi-Fi module.

(d) UART transmission/reception control driver interface

Interface between the control software and driver. Modify the settings in accordance with the target device and the UART configuration.

(e) UART transmission/reception control driver

Software for controlling UART communication. The code generator provided as a tool in the Renesas integrated development environment is used to generate this software for the RL78/G10.

Generate an appropriate code for the target device.

(f) Serial array unit

The serial array unit (SAU) is used in the UART transmission/reception mode.

(g) ESP-WROOM-02

A Wi-Fi module.

4.2 Required memory size (CC-RL V1.11.00)

Required memory size	Size [Byte]
ROM	3144
RAM	140
Stack size	184

Note: The required memory size depends on the version of the C compiler and the compile option settings. The above values include those for the sample application program.

4.3 Option Byte Settings

Table 4.1 lists the option byte settings.

Table 4.1 Option Byte Settings

Address	Setting Value	Contents
000C0H	11101111B	Operation of Watchdog timer is stopped（counting is stopped after reset.）
000C1H	11110111B	SPOR detection voltage Rising edge: TYP. 2.90 V (2.76 to 2.87 V) Falling edge: TYP. 2.84 V (2.70to 2.81 V)
000C2H	11111001B	High-speed on-chip oscillator clock: 20 MHz
000C3H	10000101B	On-chip debugging enabled

4.4 Folder Configuration

Table 4.2 shows a folder configuration of this Application Note.

Table 4.2 Folder configuration

Folder configuration	Outline	Corresponding item in figure 4
\an-r01an4791xx0100-rl78g10-communication <DIR>	Root folder of this Application Note	-
r01an4791ej0100-rl78g10.pdf	PDF file of this Application Note	-
\workspace\{IDE}\ESP CS+\src <DIR>	Folder for program source	-
\sample <DIR>	Folder for Application sample program	-
testmain.c	Application sample program	(a)
\r_esp_wroom_02 <DIR>	Folder for Wi-Fi module control program	-
r_esp.c	Wi-Fi module control program	(b), (c)
r_esp.h	Header file of Wi-Fi module control program	(b), (c)
\cg_src <DIR>	Folder for MCU drivers	-
r_cg_cgc.c[Note]	CGC module	(a)
r_cg_cgc.h[Note]	Header file of CGC module	(a)
r_cg_cgc_user.c[Note]	CGC module	(a)
r_cg_macrodriver.h[Note]	Header file of driver macros	(a)
r_cg_main.c[Note]	Main processing module	(a)
r_cg_sau.c[Note]	SAU module	(e)
r_cg_sau.h[Note]	Header file of SAU module	(e)
r_cg_sau_user.c[Note]	SAU module	(e)
r_cg_systeminit.c[Note]	System initialization module	(a)
r_cg_userdefine.h[Note]	Header file of user's definitions	(a)
r_esp_hw.c[Note]	Driver interface module	(d)
r_esp_hw.h[Note]	Header file of driver interface module	(d)

Note: Generate appropriate modules for the target device by using the code generator. When integrating generated modules into an existing project, confirm that they do not conflict with existing software.

4.5 Return Value

Table 4.3 lists return value of Wi-Fi module controlling API functions.

Table 4.3 Return Value of API Functions

Label	Contents
ESP_OK	Wi-Fi module responded “OK”
ESP_ERROR	Wi-Fi module responded “ERROR”
ESP_ALREADY_CONNECT	Wi-Fi module responded “ALREADY CONNECT”
ESP_TIMEOUT	Timeout
ESP_OVERFLOW	Buffer overflow happen
ESP_OTHERS	Other Errors

4.6 Constants

Table 4.4 lists the constants used in Wi-Fi module controlling API functions.

Table 4.4 Constants in the Sample Program

Constant Name	Value	Contents
FCLK_MHZ	20	CPU clock frequency (fCLK) [MHz]
ESP_RECV_BUFSIZE	128	Size of UART receive buffer [B]
ESP_LOOPNUM_1MS	FCLK_MHZ * 143	Loop count for 1 ms wait

4.7 Structures

Table 4.5lists the structures used in Wi-Fi module controlling API functions.

Table 4.5 Structures in the Sample Program

[Structure Name] esp_params_t		
Outline	Structure for storing Wi-Fi module control parameters	
Variables	uint8_t * const ssid	SSID of Access Point
	uint8_t * const pwd	Password of Access Point
	uint16_t timeout_ms	Timeout value to wait response from Wi-Fi module [ms]
	uint8_t ip_address[15]	Buffer to store IP Address of Wi-Fi module
	uint8_t mac_address[17]	Buffer to store MAC Address of Wi-Fi module
	uint8_t * const ip_address_target	IP address of TCP server
	uint8_t * const port_target	Port number of TCP server

4.8 API Functions List

Table 4.6 lists functions of Wi-Fi module control API.

Table 4.6 List of API Functions

Function Name	Outline
R_ESP_Init	Initializes the Wi-Fi module.
R_ESP_SendCommandToESP	Sends an AT command to the Wi-Fi module.
R_ESP_Receive	Starts reception.
R_ESP_WaitResultFromESP	Waits for result code reception from the Wi-Fi module.
R_ESP_ConnectToAP	Connects to an access point.
R_ESP_DisconnectFromAP	Disconnects from the current access point.
R_ESP_TCP_Open	Establishes a TCP connection.
R_ESP_TCP_Close	Terminates the current TCP connection.
R_ESP_GetIPAddress	Gets the IP address of the Wi-Fi module.
R_ESP_GetMACAddress	Gets the MAC address of the Wi-Fi module.
R_ESP_SendDataToNetwork	Sends data to the network.
R_ESP_WaitDataFromNetwork	Waits for data reception from the network.
R_ESP_GetDataFromNetwork	Gets data from the network.
R_WaitMilliSeconds	Waits for the specified milliseconds.
R_Wait1MilliSecond	Waits for one millisecond.
R_strstr_WithTail	Searches for a specified pattern in a string.

4.9 API Function Specification

The following describes detailed specifications of the API functions for controlling the Wi-Fi module.

4.9.1 Initializing the Wi-Fi Module

[Function Name] R_ESP_Init

Outline	Initializes the Wi-Fi module.	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_Init(esp_params_t* params);	
Description	Initializes the Wi-Fi module AT Commands Echoing: off Wi-Fi mode: Station mode Call once at system starting up.	
Arguments	esp_params_t* params	Wi-Fi module control parameters
Return value	Response from Wi-Fi module	
	ESP_OK	Wi-Fi module responded “OK”
	ESP_ERROR	Wi-Fi module responded “ERROR”
	ESP_TIMEOUT	Timeout
Remarks	None	

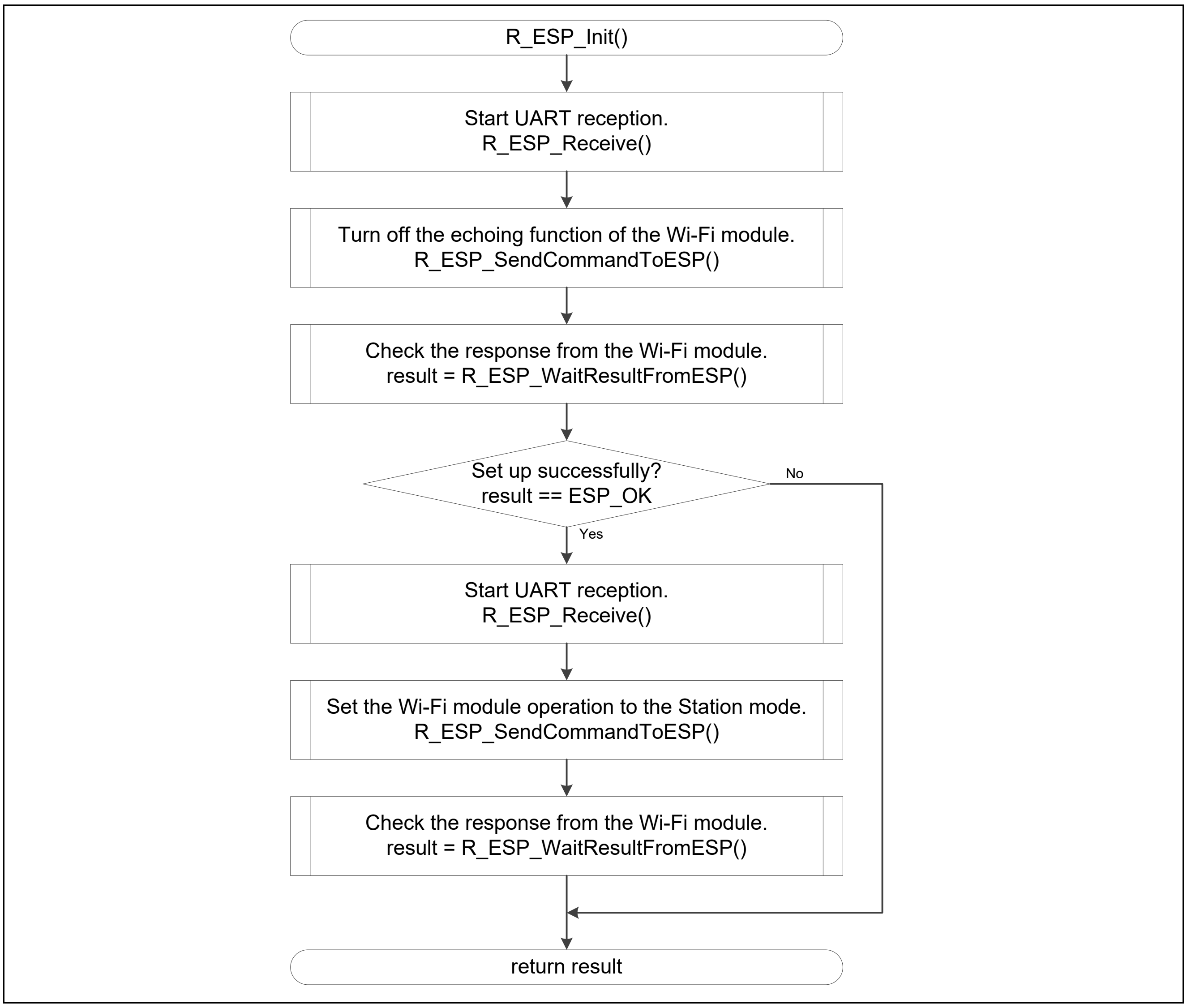

R_ESP_Init()
Start UART reception.
R_ESP_Receive()
Turn off the echoing function of the Wi-Fi module.
R_ESP_SendCommandToESP()
Check the response from the Wi-Fi module.
result = R_ESP_WaitResultFromESP()
Set up successfully?
result == ESP_OK
No
Yes
Start UART reception.
R_ESP_Receive()
Set the Wi-Fi module operation to the Station mode.
R_ESP_SendCommandToESP()
Check the response from the Wi-Fi module.
result = R_ESP_WaitResultFromESP()
return result

Figure 4.2 Initializing the Wi-Fi Module

4.9.2 Sending an AT Command to the Wi-Fi Module

[Function Name]	R_ESP_SendCommandToESP
Outline	Sends an AT command to the Wi-Fi module.
Header	r_esp.h
Declaration	esp_err_t R_ESP_SendCommandToESP(uint8_t * const buf_command);
Description	Send AT command passed as an argument to Wi-Fi module Please add CR, LF, and null character ('\0') at the end of the command
Arguments	uint8_t * const buf_command — Address of send command buffer
Return value	ESP_OK

Remarks	Call the following functions in this order to send an AT command to the Wi-Fi module and to receive a result code.
	R_ESP_Receive()
	R_ESP_SendCommandToESP()
	R_ESP_WaitResultFromESP()

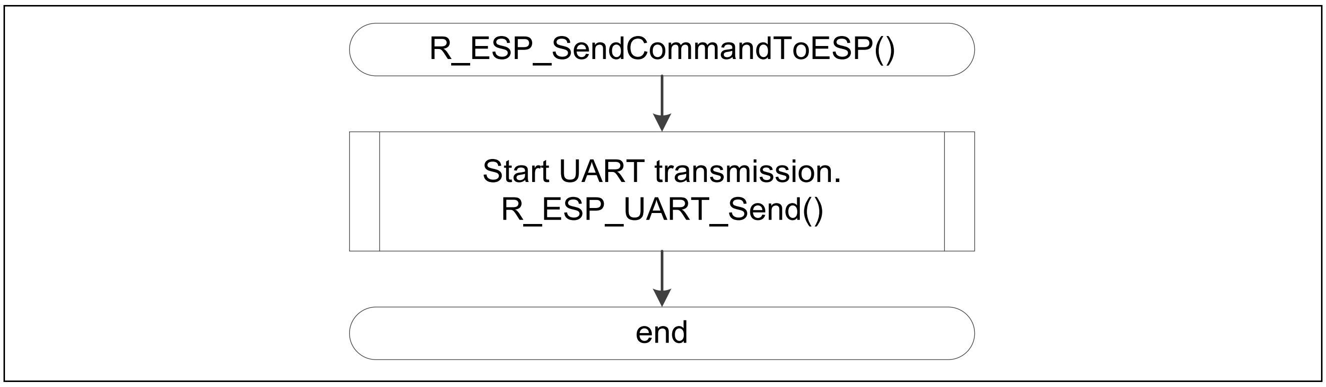


Figure 4.3 Sending an AT Command to the Wi-Fi Module

4.9.3 Starting Data Reception from the Wi-Fi Module

[Function Name] R_ESP_Receive

Outline	Starts data reception from the Wi-Fi module.	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_Receive(esp_params_t* params);	
Description	Clear receive buffer, and call UART driver to start UART receiving	
Arguments	esp_params_t* params	Wi-Fi module control parameters
Return value	ESP_OK	
Remarks	None	

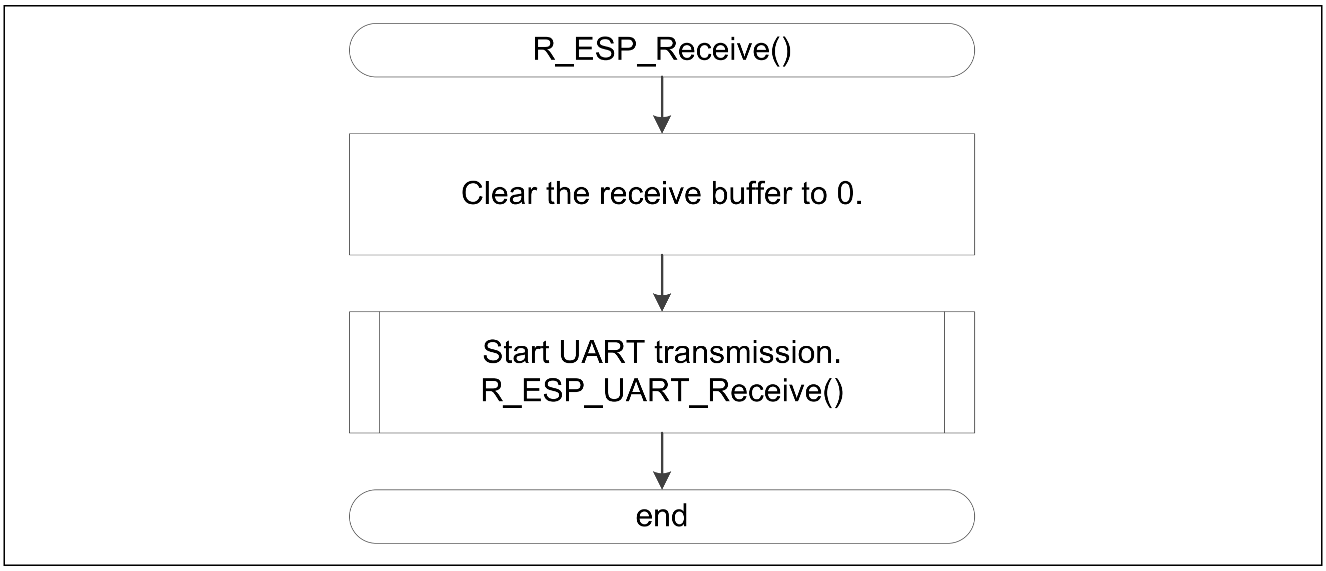
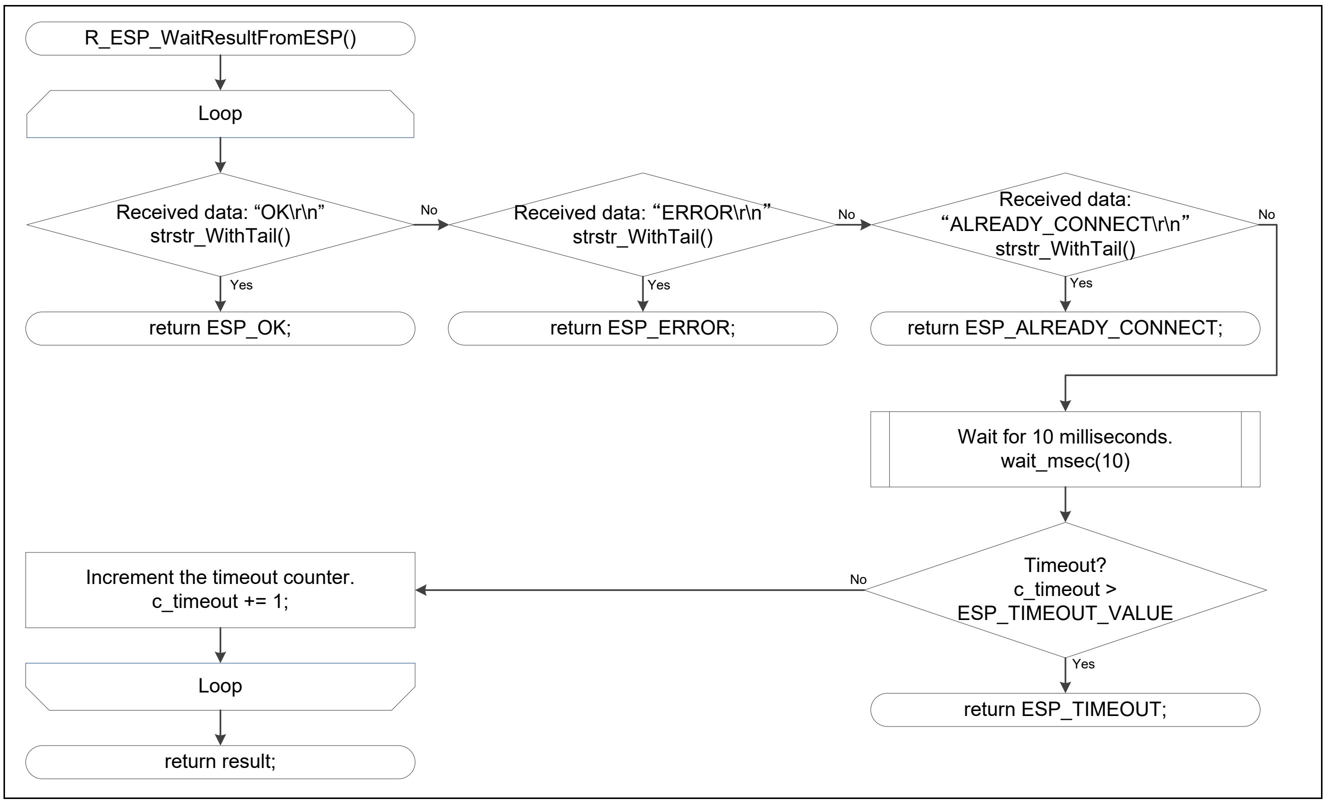


Figure 4.4 Starting Data Reception from the Wi-Fi Module

4.9.4 Waiting for Result Code Reception from the Wi-Fi Module

[Function Name]	R_ESP_WaitResultFromESP	
Outline	Wiats for result code reception from the Wi-Fi module.	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_WaitResultFromESP(esp_params_t* params);	
Description	Scan receive buffer and wait until timeout or until receive "OK", "ERROR", or "ALREADY CONNECTED"	
Arguments	esp_params_t* params	Wi-Fi module control parameters
Return value	Response from Wi-Fi module	
	ESP_OK	Wi-Fi module responded “OK”
	ESP_ERROR	Wi-Fi module responded “ERROR”
	ESP_ALREADY_CONNECT	Wi-Fi module responded “ALREADY_CONNECT”
	ESP_TIMEOUT	Timeout

Remarks	Call the following function in this order to send an AT command to the Wi-Fi module and to receive a result code.
	R_ESP_Receive()
	R_ESP_SendCommandToESP()
	R_ESP_WaitResultFromESP()	



R_ESP_WaitResultFromESP()
Loop
Received data: “OK\r\n” strstr_WithTail()
No
Yes
return ESP_OK;
Received data: “ERROR\r\n” strstr_WithTail()
return ESP_ERROR;
Received data: “ALREADY_CONNECT\r\n” strstr_WithTail()
return ESP_ALREADY_CONNECT;
Wait for 10 milliseconds. wait_msec(10)
Timeout? c_timeout > ESP_TIMEOUT_VALUE
return ESP_TIMEOUT;
Increment the timeout counter. c_timeout += 1;
Loop
return result;

Figure 4.5 Waiting for Result Code Reception from the Wi-Fi Module

4.9.5 Connecting to an Access Point

[Function Name]	R_ESP_ConnectToAP	
Outline	Connects to an access point.	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_ConnectToAP(esp_params_t* params);	
Description	Connects the Wi-Fi module to an access point.	
Arguments	esp_params_t* params	Wi-Fi module control parameters
Return value	Returns the result of connection to the access point.	
	ESP_OK	Wi-Fi module responded “OK”
	ESP_ERROR	Wi-Fi module responded “ERROR”
	ESP_TIMEOUT	Timeout
Remarks	None	

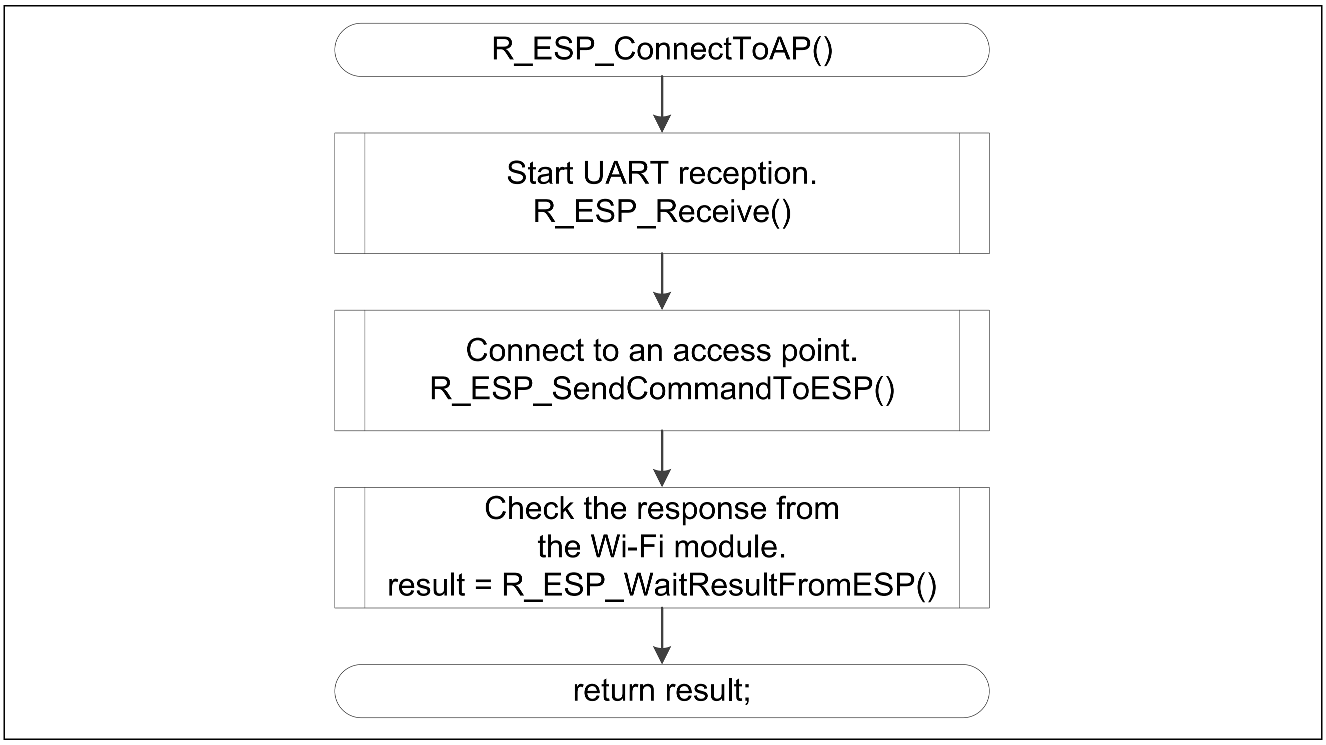

R_ESP_ConnectToAP()
Start UART reception.
R_ESP_Receive()
Connect to an access point.
R_ESP_SendCommandToESP()
Check the response from the Wi-Fi module.
result = R_ESP_WaitResultFromESP()
return result;

Figure 4.6 Connecting to an Access Point

4.9.6 Disconnecting from the Current Access Point

[Function Name]	R_ESP_DisconnectFromAP	
Outline	Disconnects from the current access point.	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_DisconnectFromAP(esp_params_t* params);	
Description	Disconnects the Wi-Fi module from the current access point connected.	
Arguments	esp_params_t* params	Wi-Fi module control parameters
Return value	Returns the result of disconnection from the access point.	
	ESP_OK	Wi-Fi module responded “OK”
	ESP_TIMEOUT	Timeout
Remarks	None	

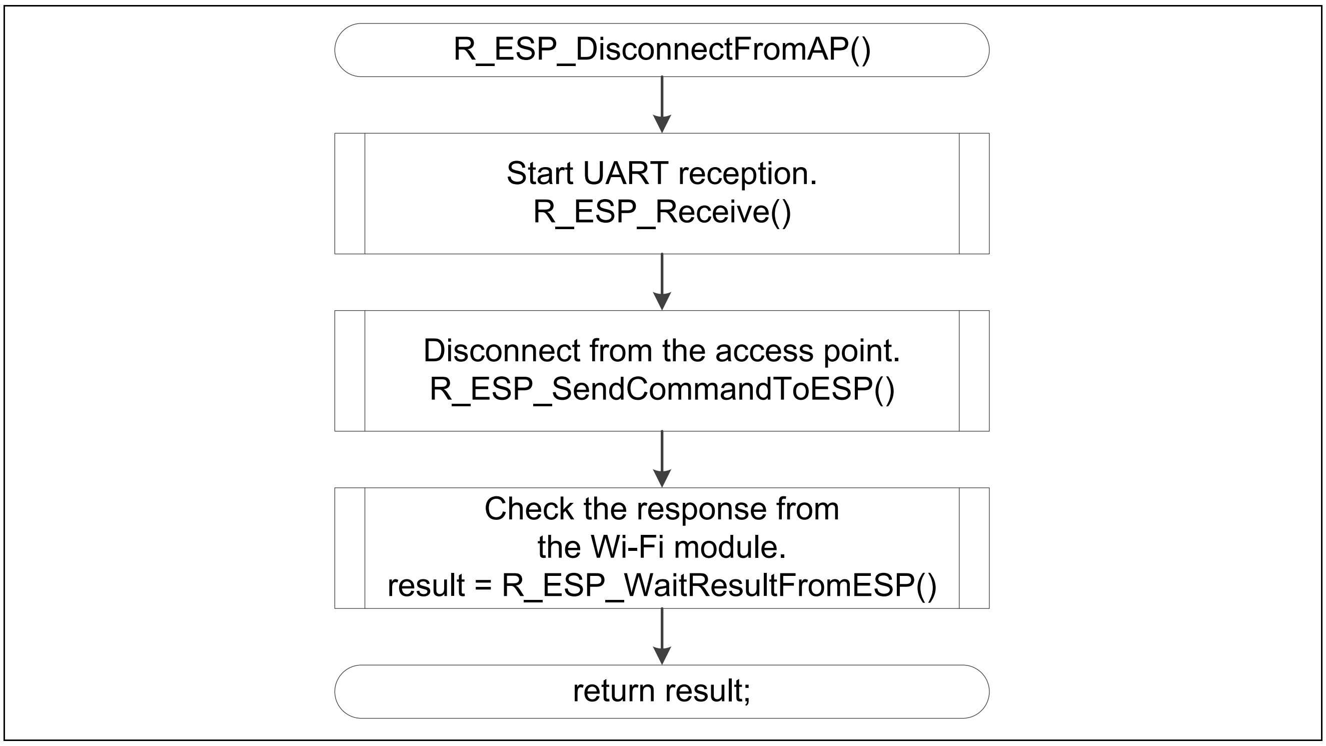

R_ESP_DisconnectFromAP()
Start UART reception.
R_ESP_Receive()
Disconnect from the access point.
R_ESP_SendCommandToESP()
Check the response from the Wi-Fi module.
result = R_ESP_WaitResultFromESP()
return result;

Figure 4.7 Disconnecting from the Current Access Point

4.9.7 Establishing a TCP Connection

[Function Name]	R_ESP_TCP_Open	
Outline	Establishes a TCP connection.	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_TCP_Open(esp_params_t* params);	
Description	Connects to a server as a TCP client and establishes a connection.	
Arguments	esp_params_t* params	Wi-Fi module control parameters
Return value	Returns the result of establishing a TCP connection.	
	ESP_OK	Wi-Fi module responded “OK”
	ESP_ERROR	Wi-Fi module responded “ERROR”
	ESP_ALREADY_CONNECT	Wi-Fi module responded “ALREADY CONNECT”
	ESP_TIMEOUT	Timeout
Remarks	None	

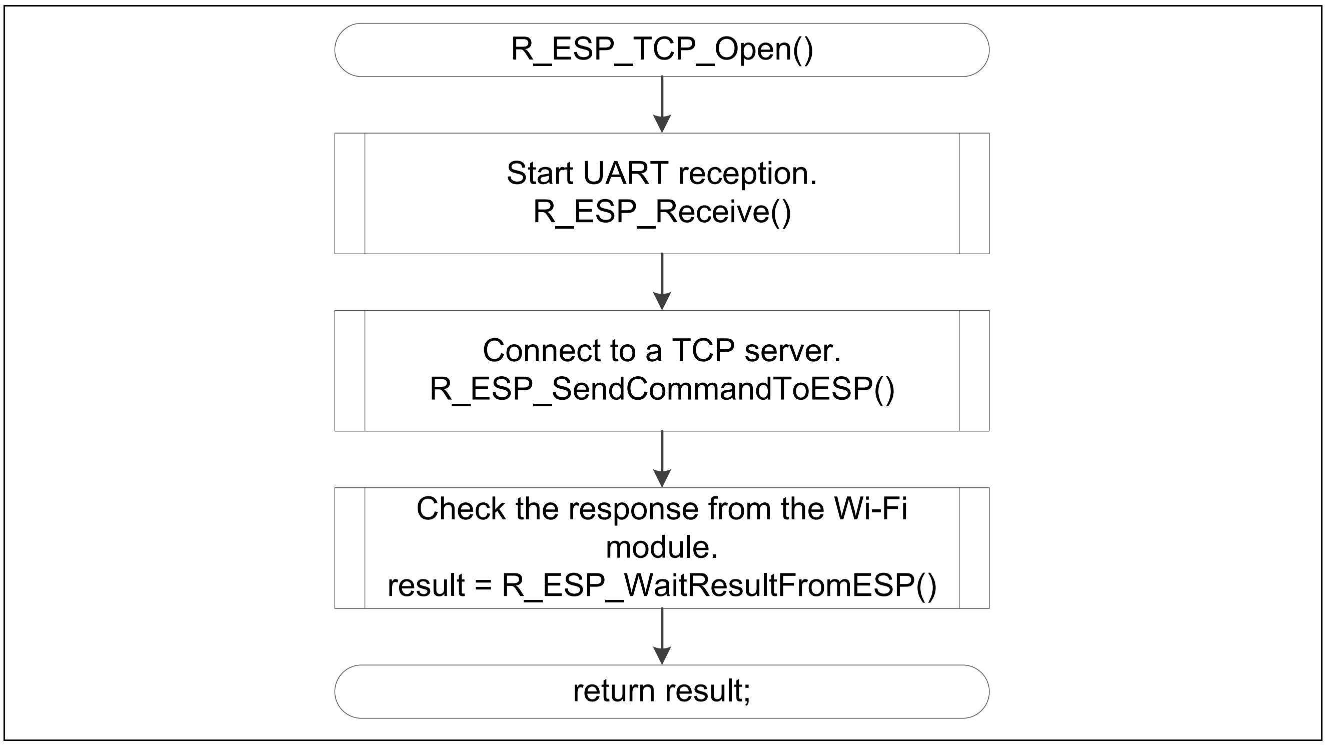


Figure 4.8 Establishing a TCP Connection

4.9.8 Terminating the Current TCP Connection

[Function Name]	R_ESP_TCP_Close	
Outline	Disconnects from the TCP server.	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_TCP_Close(esp_params_t* params);	
Description	Terminates the current TCP connection.	
Arguments	esp_params_t* params	Wi-Fi module control parameters
Return value	Returns the result of terminating the TCP connection.	
	ESP_OK	Wi-Fi module responded “OK”
	ESP_TIMEOUT	Timeout
Remarks	None	

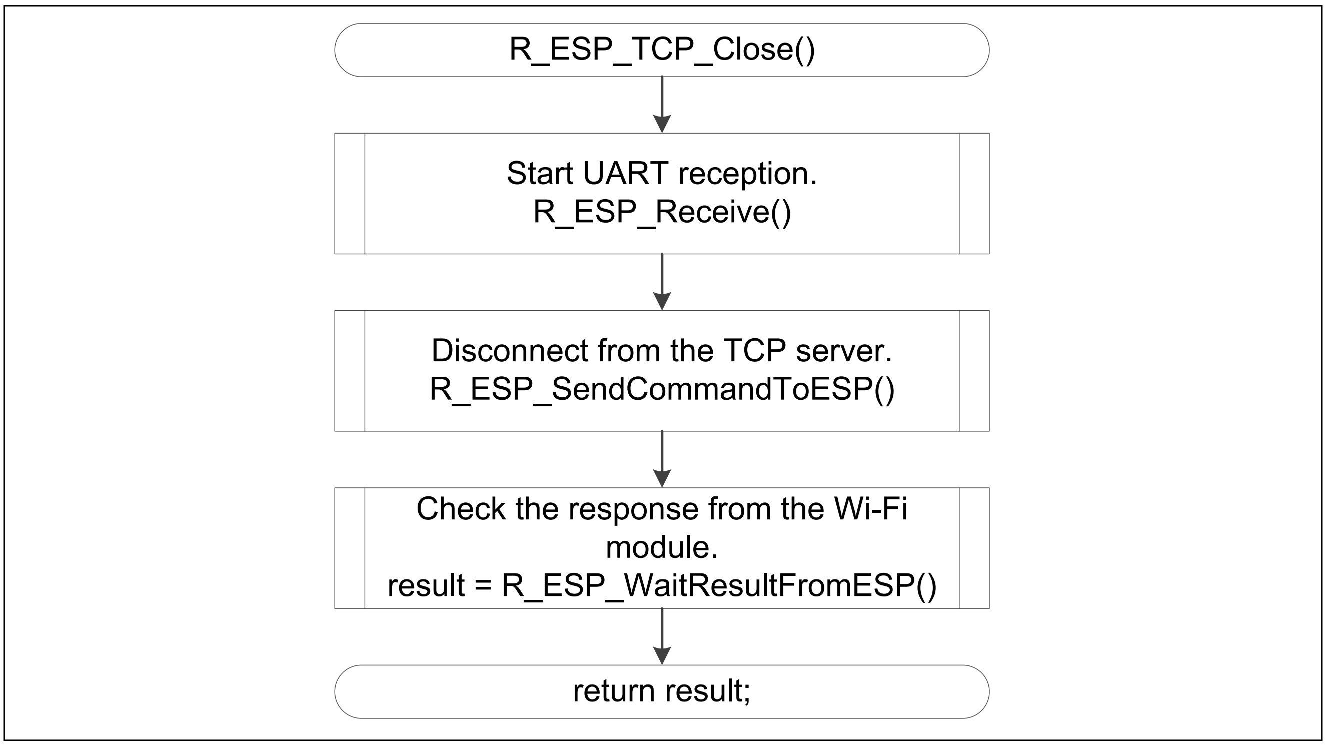

R_ESP_TCP_Close()
Start UART reception.
R_ESP_Receive()
Disconnect from the TCP server.
R_ESP_SendCommandToESP()
Check the response from the Wi-Fi module.
result = R_ESP_WaitResultFromESP()
return result;

Figure 4.9 Terminating the Current TCP Connection

4.9.9 Getting the IP address of the Wi-Fi Module

[Function Name]	R_ESP_GetIPAddress	
Outline	Gets the IP address of the Wi-Fi module.	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_GetIPAddress(esp_params_t* params);	
Description	Obtains the IP address assigned to the Wi-Fi module. An IP address is a string of up to 15 characters separated by commas.	
Arguments	esp_params_t* params	Wi-Fi module control parameters
Return value	Returns the result of getting the IP address.	
	ESP_OK	Wi-Fi module responded “OK”
	ESP_ERROR	Wi-Fi module responded “ERROR”
	ESP_TIMEOUT	Timeout
Remarks	None	

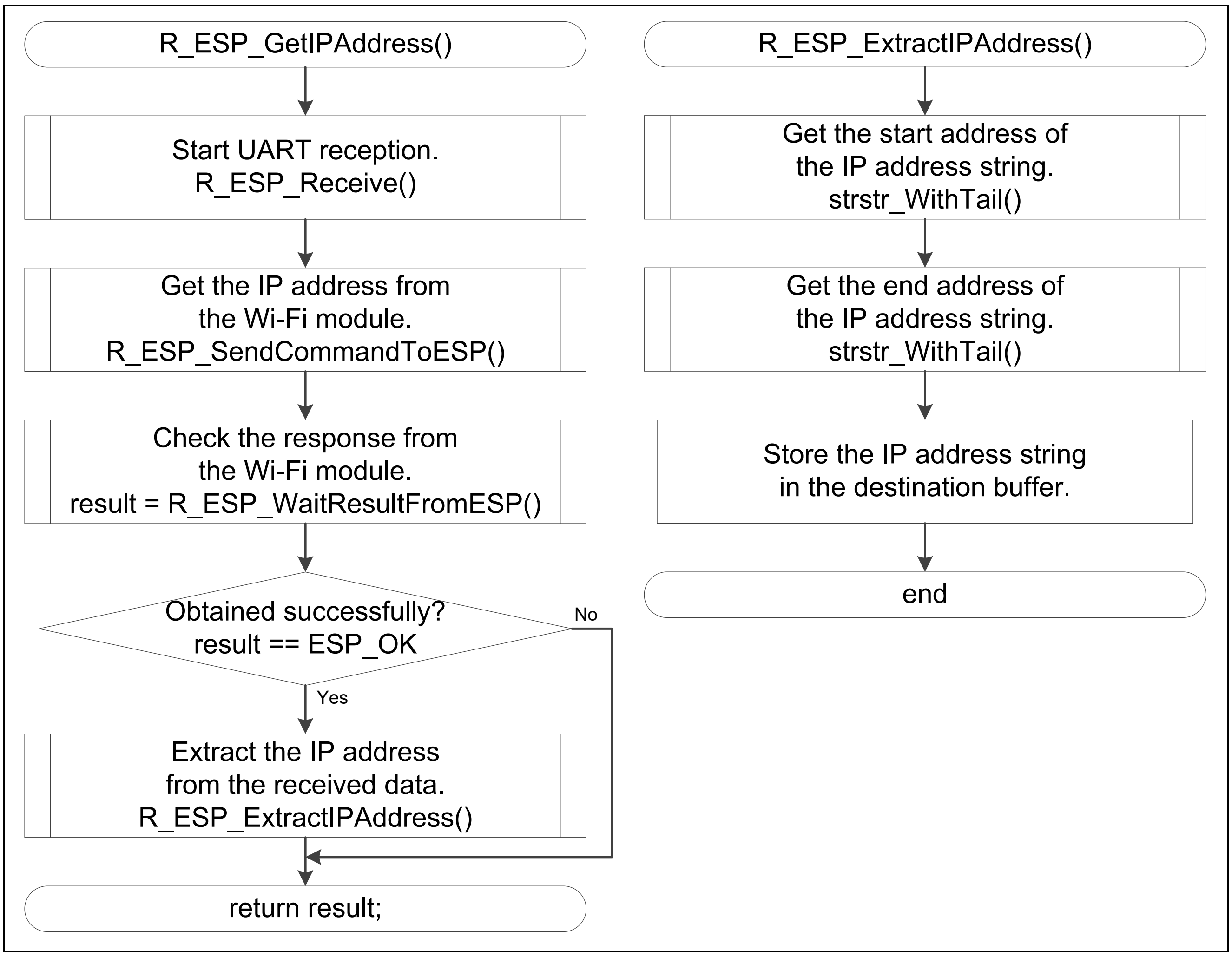

R_ESP_GetIPAddress()
Start UART reception.
R_ESP_Receive()
Get the IP address from the Wi-Fi module.
R_ESP_SendCommandToESP()
Check the response from the Wi-Fi module.
result = R_ESP_WaitResultFromESP()
Obtained successfully?
result == ESP_OK
No
Yes
Extract the IP address from the received data.
R_ESP_ExtractIPAddress()
return result;
R_ESP_ExtractIPAddress()
Get the start address of the IP address string.
strstr_WithTail()
Get the end address of the IP address string.
strstr_WithTail()
Store the IP address string in the destination buffer.
end

Figure 4.10 Getting the IP address of the Wi-Fi Module

4.9.10 Getting the MAC Address of the Wi-Fi Module

[Function Name]	R_ESP_GetMACAddress	
Outline	Gets the MAC address of the Wi-Fi module.	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_GetMACAddress(esp_params_t* params);	
Description	Gets the MAC address of the Wi-Fi module. A MAC address is a string of up to 17 characters separated by semicolons.	
Arguments	esp_params_t* params	Wi-Fi module control parameters
Return value	Returns the result of getting the MAC address.	
	ESP_OK	Wi-Fi module responded “OK”
	ESP_ERROR	Wi-Fi module responded “ERROR”
	ESP_TIMEOUT	Timeout
Remarks	None	

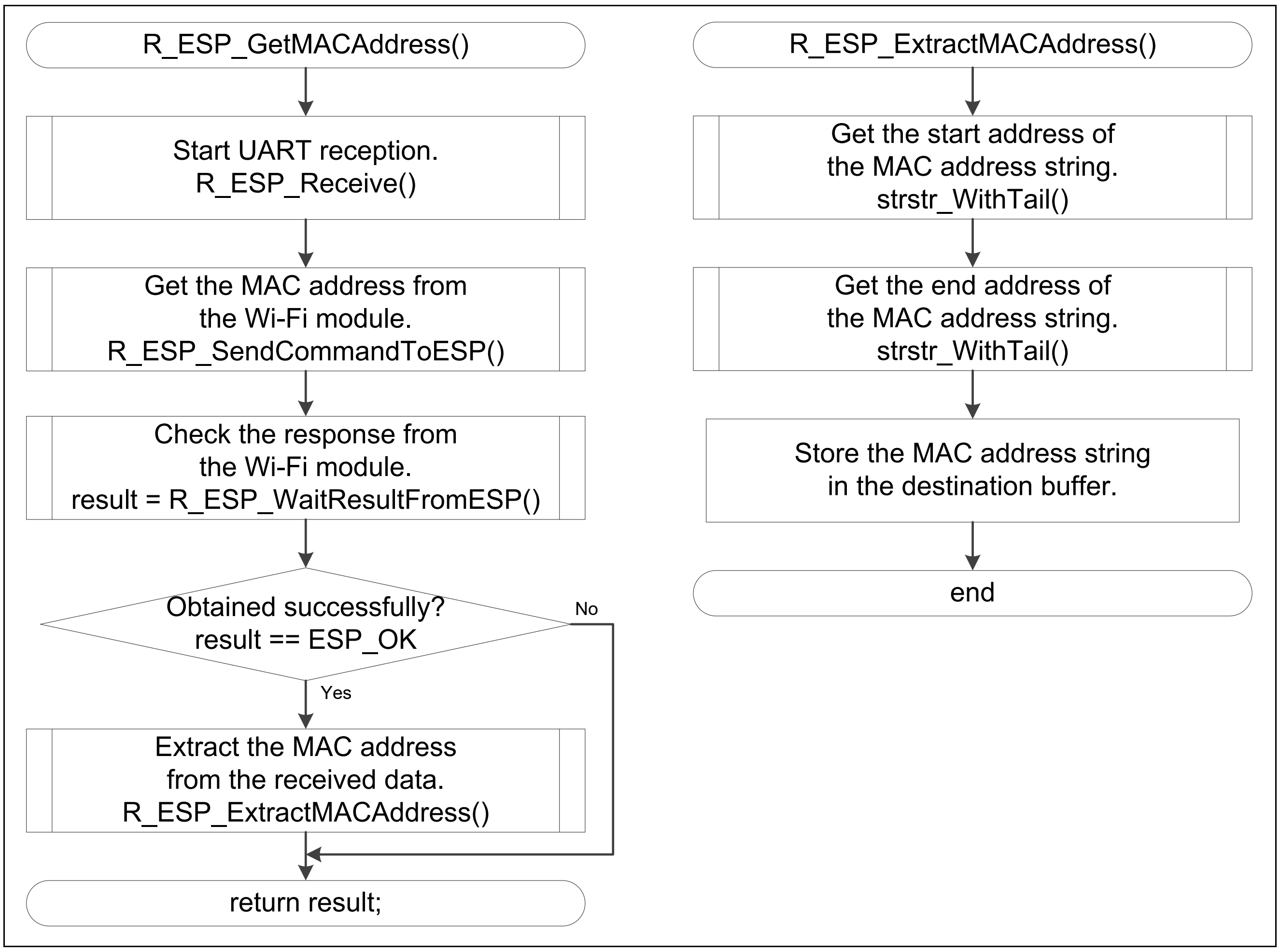

R_ESP_GetMACAddress()
Start UART reception.
R_ESP_Receive()
Get the MAC address from the Wi-Fi module.
R_ESP_SendCommandToESP()
Check the response from the Wi-Fi module.
result = R_ESP_WaitResultFromESP()
Obtained successfully?
result == ESP_OK
No
Yes
Extract the MAC address from the received data.
R_ESP_ExtractMACAddress()
return result;
R_ESP_ExtractMACAddress()
Get the start address of the MAC address string.
strstr_WithTail()
Get the end address of the MAC address string.
strstr_WithTail()
Store the MAC address string in the destination buffer.
end

Figure 4.11 Getting the MAC Address of the Wi-Fi Module

4.9.11 Sending Data to the Network

[Function Name]	R_ESP_SendDataToNetwork	
Outline	Sends data to the network.	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_SendDataToNetwork(esp_params_t* params, uint8_t * const buf_senddata);	
Description	Sends data to the connected IP address.	
Arguments	esp_params_t* params	Wi-Fi module control parameters
	uint8_t * const buf_senddata	Start address of the area where data to be sent is stored
Return value	Returns the result of data transmission.	
	ESP_OK	Wi-Fi module responded “OK”
	ESP_ERROR	Wi-Fi module responded “ERROR”
	ESP_TIMEOUT	Timeout
Remarks	Maximum length of transmit data: 2048 bytes	

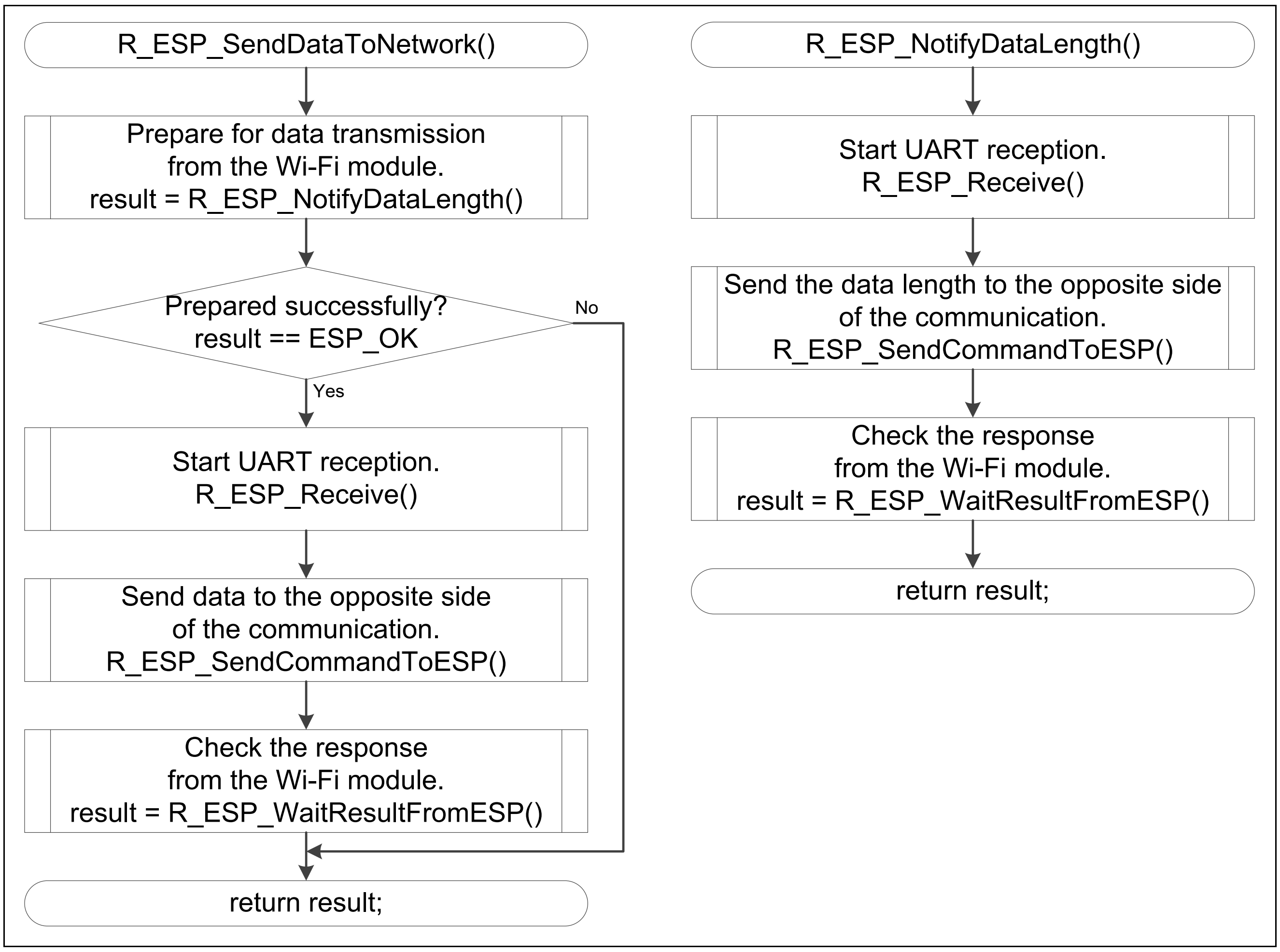

R_ESP_SendDataToNetwork()
Prepare for data transmission from the Wi-Fi module. result = R_ESP_NotifyDataLength()
Prepared successfully? result == ESP_OK
No
Yes
Start UART reception. R_ESP_Receive()
Send data to the opposite side of the communication. R_ESP_SendCommandToESP()
Check the response from the Wi-Fi module. result = R_ESP_WaitResultFromESP()
return result;
R_ESP_NotifyDataLength()
Start UART reception. R_ESP_Receive()
Send the data length to the opposite side of the communication. R_ESP_SendCommandToESP()
Check the response from the Wi-Fi module. result = R_ESP_WaitResultFromESP()
return result;

Figure 4.12 Sending Data to the Network

4.9.12 Waiting for Data Reception from the Network

[Function Name	R_ESP_WaitDataFromNetwork	
Outline	Waits for data reception from the network.	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_WaitDataFromNetwork(esp_params_t* params);	
Description	Monitors the UART receive buffer and waits until the pattern “+IPD:” is found in the received data of the timeout period has elapsed.	
Arguments	esp_params_t* params	Wi-Fi module control parameters
Return value	Returns the result of waiting for data reception.	
	ESP_OK	Wi-Fi module responded “OK”
	ESP_TIMEOUT	Timeout
Remarks	None	

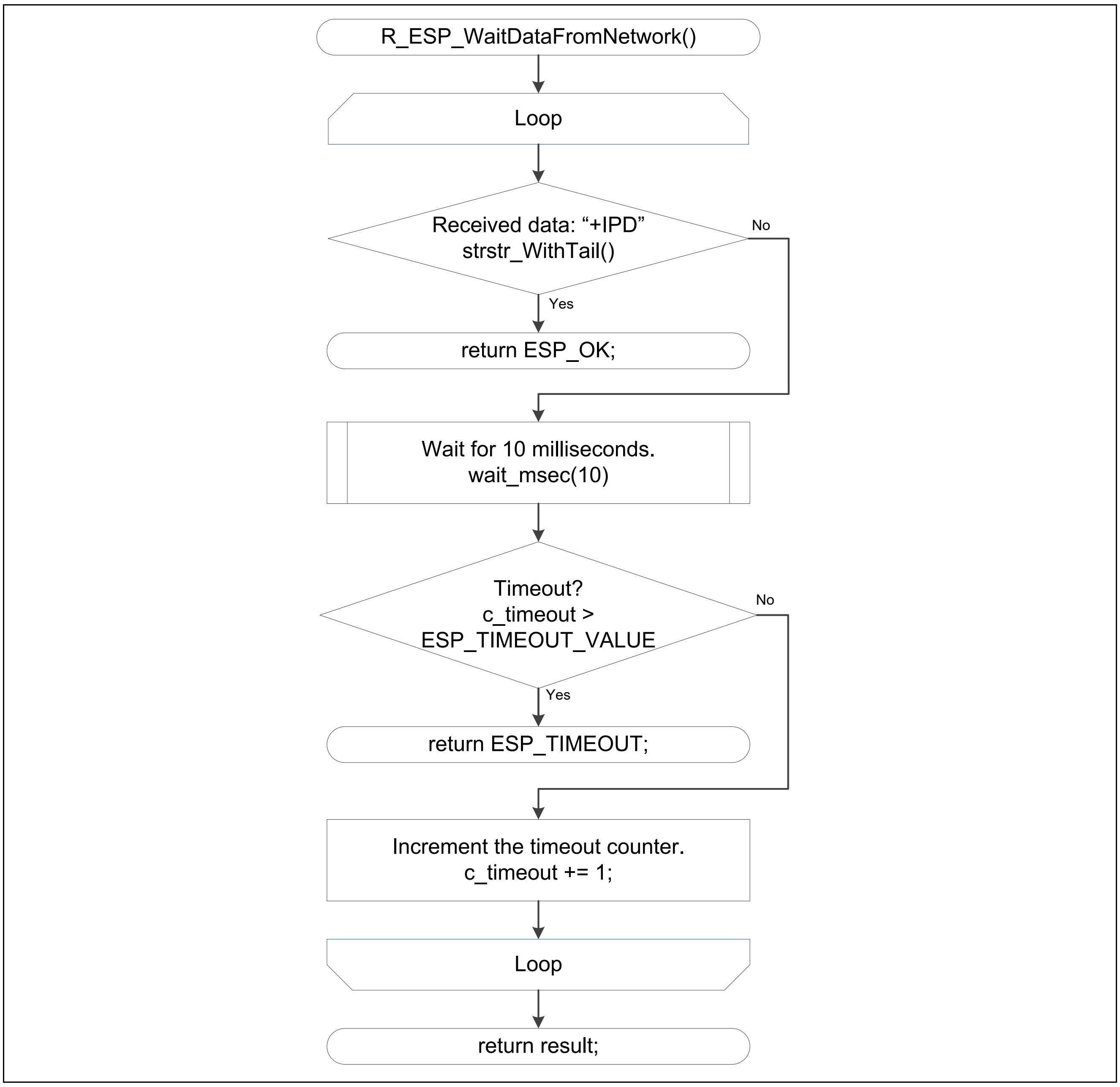


Figure 4.13 Waiting for Data Reception from the Network

4.9.13 Getting Data from the Network

[Function Name]	R_ESP_GetDataFromNetwork	
Outline	Gets data from the network	
Header	r_esp.h	
Declaration	esp_err_t R_ESP_GetDataFromNetwork(uint8_t *buf_dest, uint8_t buf_length);	
Description	Extracts network data from the result code stored in the receive buffer.	
Arguments	uint8_t *buf_dest	Destination buffer for storing the extracted data.
	uint8_t buf_length	Size of the destination buffer.
Return value	Returns the result of getting data.	
	ESP_OK	Wi-Fi module responded "OK"
	ESP_TIMEOUT	Timeout
Remarks	None	

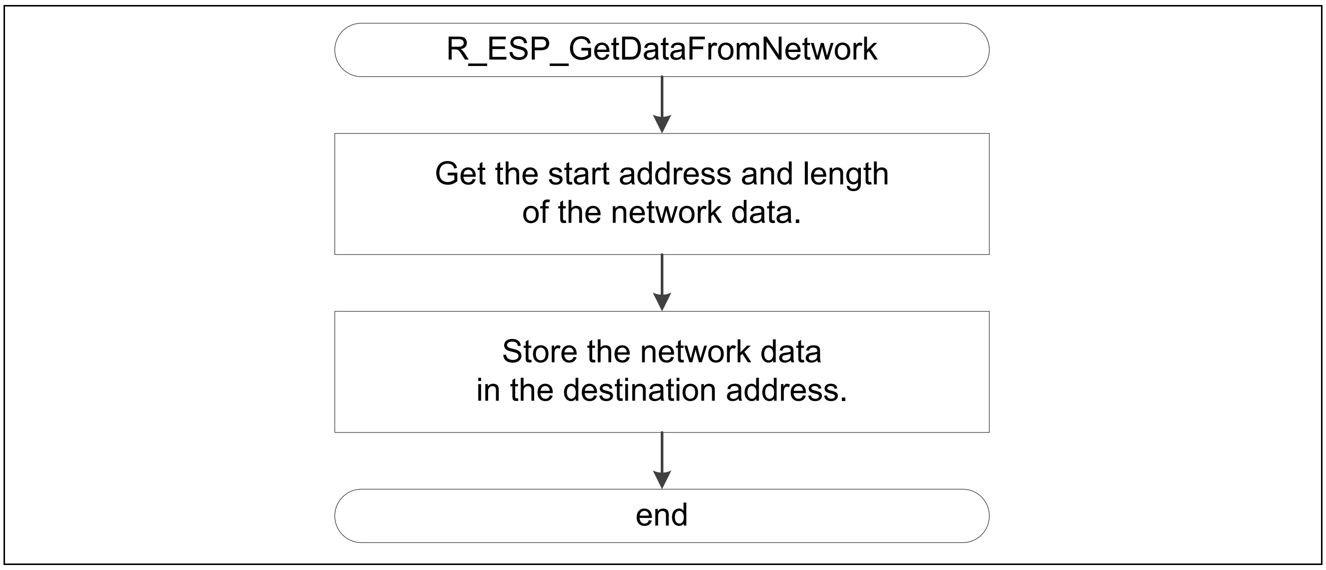

R_ESP_GetDataFromNetwork
Get the start address and length of the network data.
Store the network data in the destination address.
end

Figure 4.14 Getting Data from the Network

4.9.14 Waiting for the Specified Milliseconds

[Function Name]	**R_WaitMilliSeconds**	
Outline	Waits for the specified milliseconds.	
Header	r_esp.h	
Declaration	void R_WaitMilliSeconds(uint16_t msec);	
Description	Calls the R_Wait1MilliSeconds function repeatedly for the number of milliseconds specified in the argument.	
Arguments	uint16_t msec	Number of milliseconds to wait.
Return value	None	
Remarks	None	

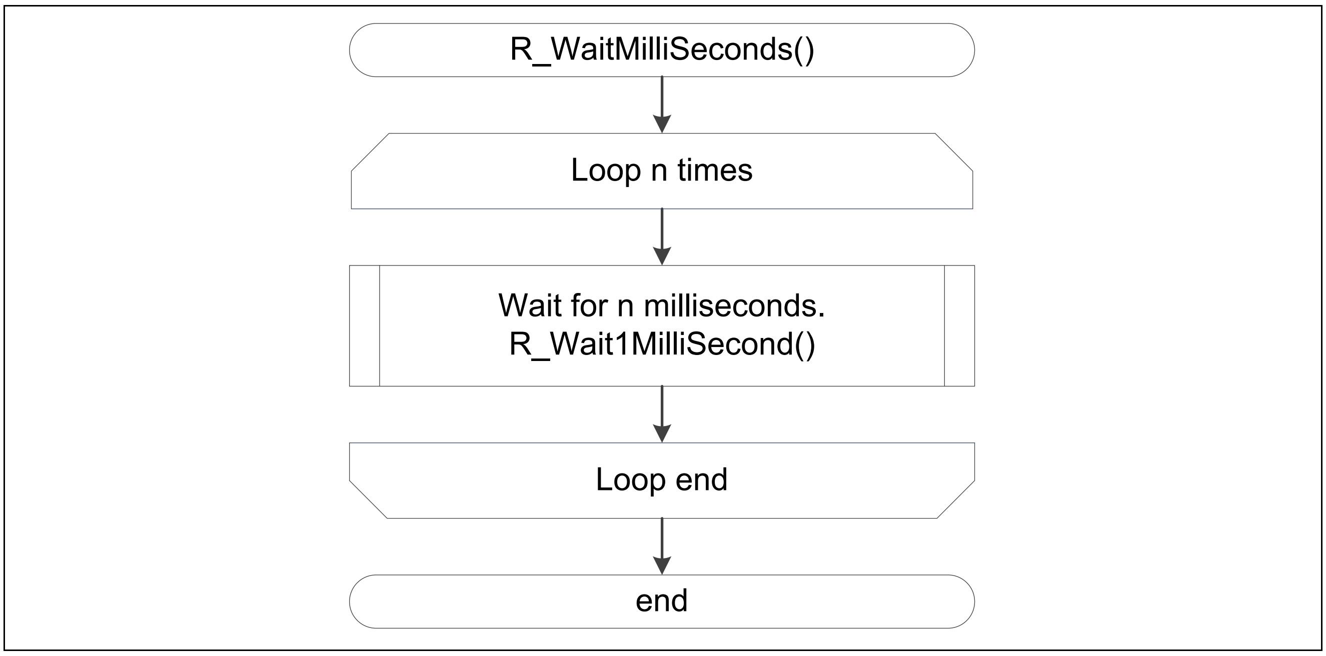


Figure 4.15 Waiting for the Specified Milliseconds

4.9.15 Waiting for One Millisecond

[Function Name]	R_Wait1MilliSecond
Outline	Waits for one millisecond.
Header	r_esp.h
Declaration	void R_Wait1MilliSecond(void);
Description	Waits for one millisecond by instruction execution in the CPU.
Arguments	None
Return value	None
Remarks	None

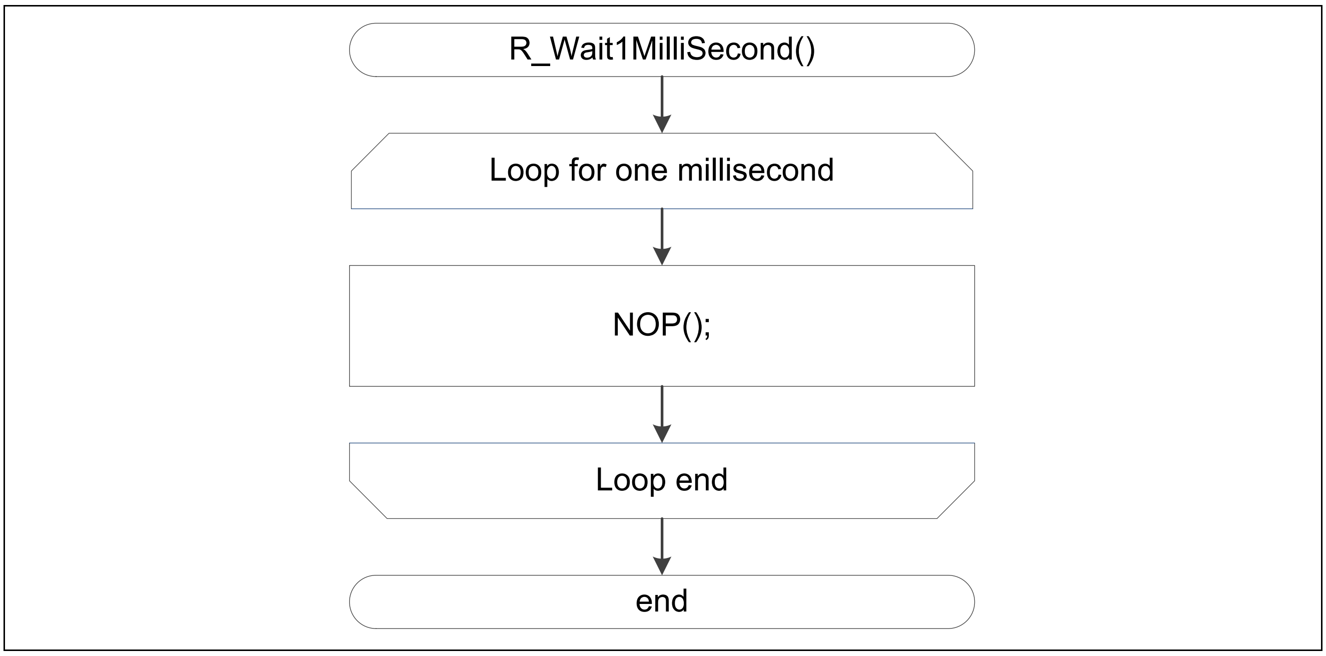
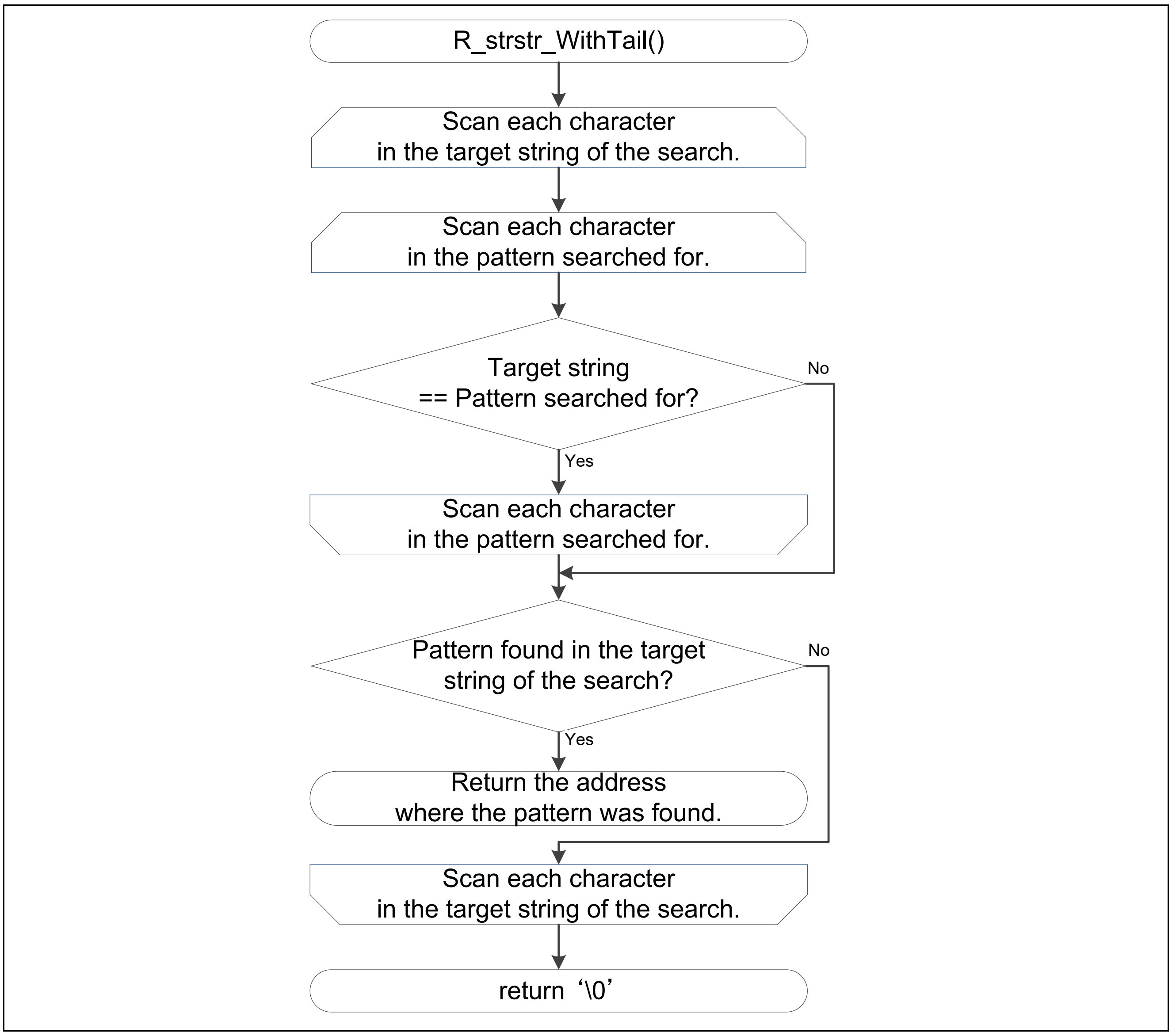


Figure 4.16 Waiting for One Millisecond

4.9.16 Searching for a Specified Pattern in a String

[Function Name]	R_strstr_WithTail	
Outline	Searches for a specified pattern in a string.	
Header	r_esp.h	
Declaration	uint8_t* R_strstr_WithTail(uint8_t *str_head, uint8_t *str_tail, uint8_t *key);	
Description	Searching for a specified pattern in a character string and returns the first address where the pattern is found.	
Arguments	uint8_t *str_head	Start address of the target string of the search
	uint8_t *str_tail	End address of the target string of the search
	uint8_t *key	Pattern to be searched for
Return value	Returns the result of searching for a pattern.	
	\0	The pattern was not found.
	Othres	First address where the pattern was found.
Remarks	None	



R_strstr_WithTail()
Scan each character in the target string of the search.
Scan each character in the pattern searched for.
Target string == Pattern searched for?
No
Yes
Scan each character in the pattern searched for.
Pattern found in the target string of the search?
No
Yes
Return the address where the pattern was found.
Scan each character in the target string of the search.
return '\0'

Figure 4.17 Searching for a Specified Pattern in a String

4.10 List of Driver Interface Functions

Table 4.7 lists the driver interface functions

Table 4.7 **List of Driver Interface Functions**

Function Name	Outline
R_ESP_UART_Send	Starts ART transmission
R_ESP_UART_Receive	Starts ART reception
R_ESP_UART_GetRxCount	Gets the number of bytes received.
R_ESP_UART_Start	Starts UART processing
R_ESP_UART_Stop	Stops UART processing

4.11 Driver Interface Function Specifications

The following describes the specifications of the driver interface functions of the Wi-Fi module control software. The interface functions are set up to call the UART0 driver. Modify the settings in accordance with the SAU configuration.

4.11.1 Starting UART Transmission Processing

[Function Name] R_ESP_UART_Send

Outline	Starts UART transmission.	
Header	r_esp_hw.h	
Declaration	void R_ESP_UART_Send(uint8_t * const tx_buf, uint16_t tx_num);	
Description	Starts the UART transmission processing. In this sample program, this function calls R_UART0_Send().	
Arguments	uint8_t * const tx_buf	Start address of the area where data to be sent is stored
	uint16_t tx_num	Length of data to be sent
Return value	None	
Remarks	None	

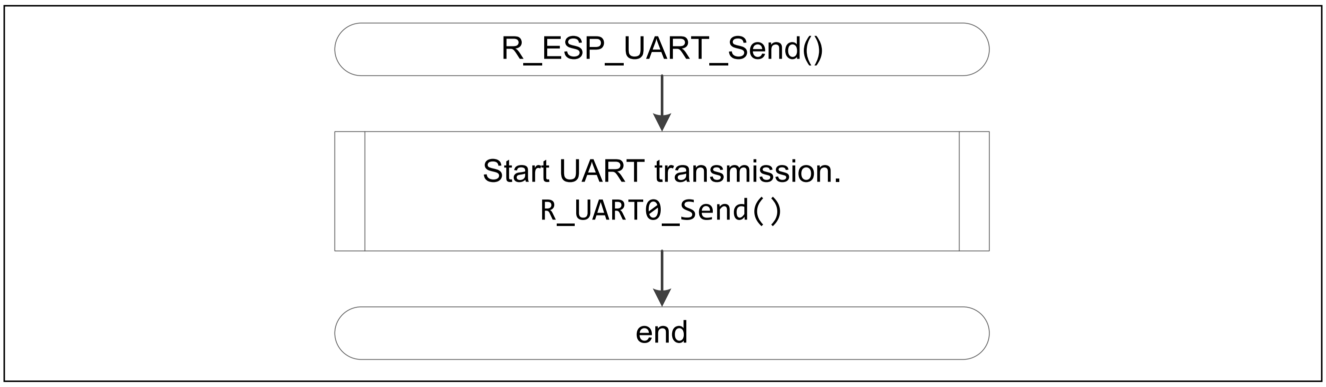


Figure 4.18 Starting UART Transmission Processing

4.11.2 Starting UART Reception Processing

[Function Name]	R_ESP_UART_Receive	
Outline	Starts UART reception.	
Header	r_esp_hw.h	
Declaration	void R_ESP_UART_Receive(uint8_t * const rx_buf, uint16_t rx_num);	
Description	Starts the UART reception processing. In this sample program, this function calls R_UART0_Receive().	
Arguments	uint8_t * const rx_buf	Start address of the area for storing the received data
	uint16_t rx_num	Length of data to be received
Return value	None	
Remarks	None	

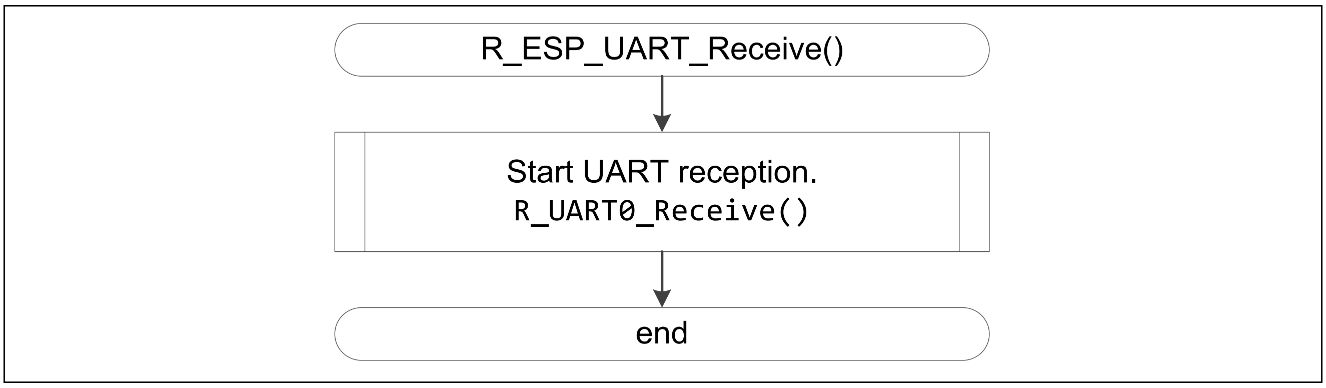


Figure 4.19 Starting UART Reception Processing

4.11.3 Getting the Length of Data Received through UART

[Function Name]	R_ESP_UART_GetRxCount
Outline	Gets the length of data received through UART
Header	r_esp_hw.h
Declaration	uint16_t R_ESP_UART_GetRxCount(void);
Description	Gets the value of the UART received data counter. In this sample program, this function gets the value of g_uart0_rx_count.
Arguments	None
Return value	Value of the UART received data counter.
Remarks	None

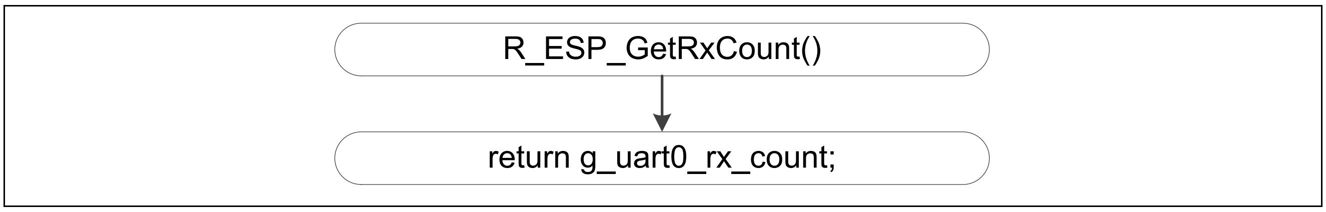


Figure 4.20 Getting the Length of Data Received through UART

4.11.4 Starting UART Processing

[Function Name]	R_ESP_UART_Start
Outline	Starts UART transmission.
Header	r_esp_hw.h
Declaration	void R_ESP_UART_Start(void);
Description	Start UART processing
Arguments	None
Return value	None
Remarks	None

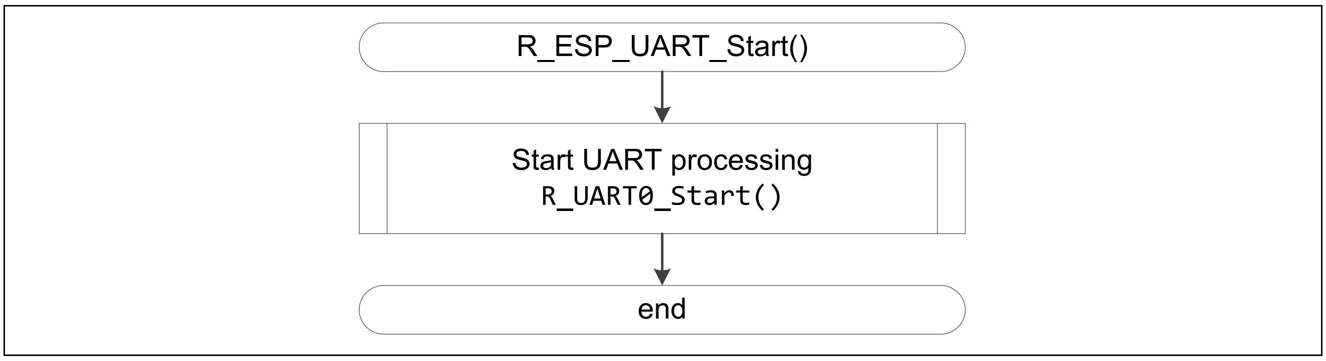


Figure 4.21 Starting UART Processing

4.11.5 Stopping UART Processing

[Function Name]	R_ESP_UART_Stop
Outline	Stops UART transmission.
Header	r_esp_hw.h
Declaration	void R_ESP_UART_Stop(void);
Description	Stop UART processing
Arguments	None
Return value	None
Remarks	None

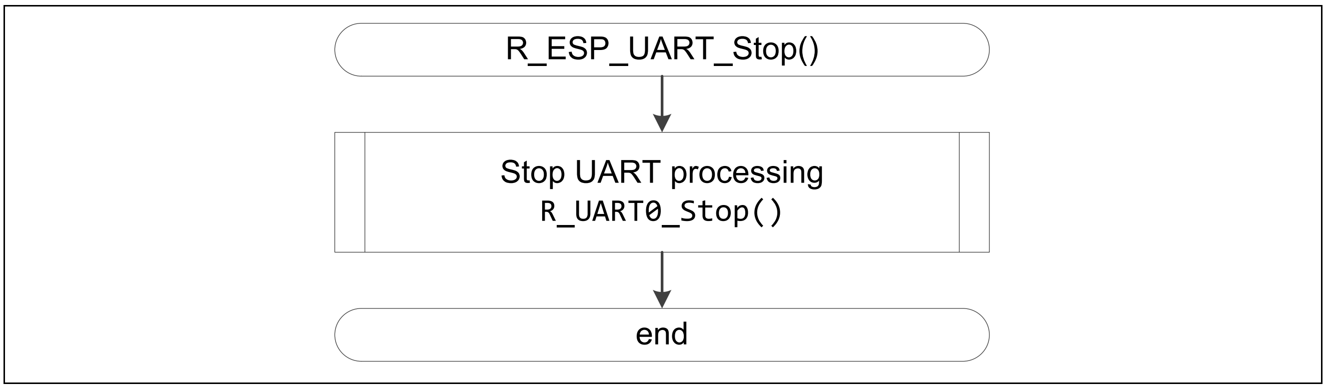
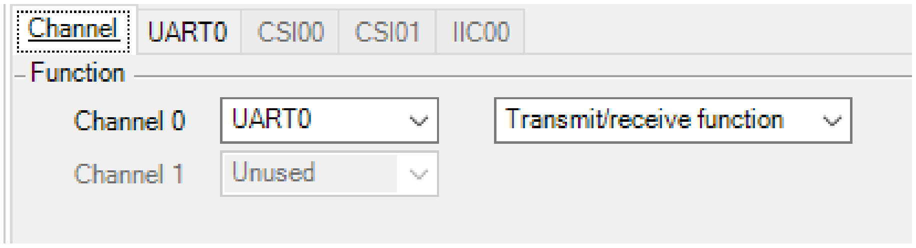


Figure 4.22 Stopping UART Processing

4.12 Device Driver

4.12.1 Generating the Driver Code

The device driver in this application note is generated by using the code generator (CG). For the CG settings, refer to Figure 4.23, Figure 4.24, and Figure 4.25.

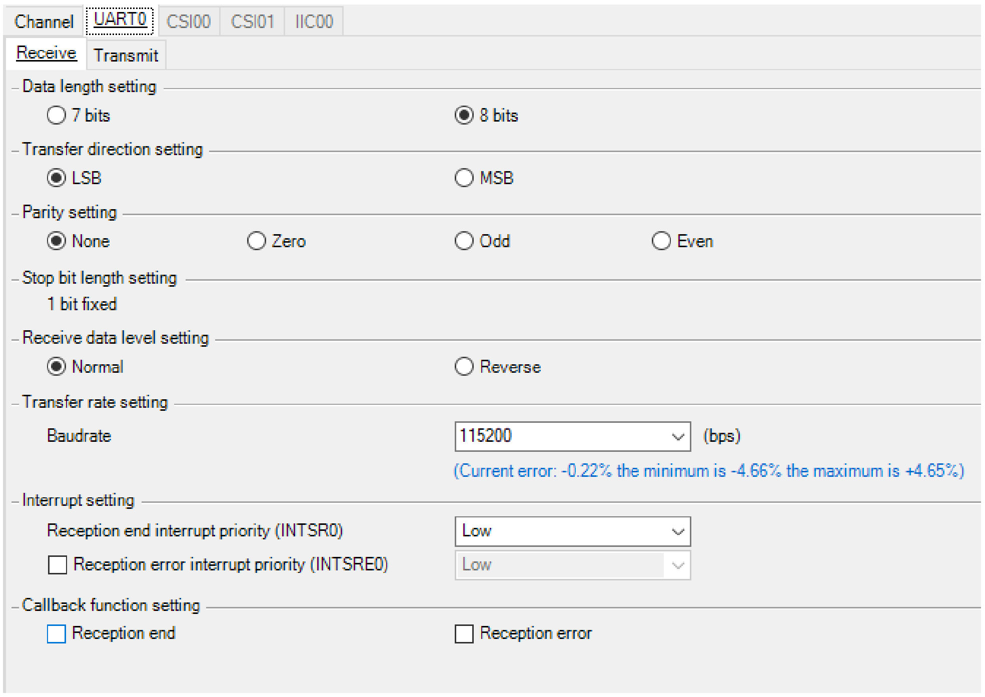


Function

Channel 0: UART0 — Transmit/receive function

Channel 1: Unused

Figure 4.23 SAU CG Configuration (1/3)



Data length setting: [✓ 8 bits] (7 bits)

Transfer direction setting: [✓ LSB] (MSB)

Parity setting: [✓ None] (Zero, Odd, Even)

Stop bit length setting: 1 bit fixed

Receive data level setting: [✓ Normal] (Reverse)

Transfer rate setting

Baudrate: 115200 (bps)

(Current error: -0.22% the minimum is -4.66% the maximum is +4.65%)

Interrupt setting

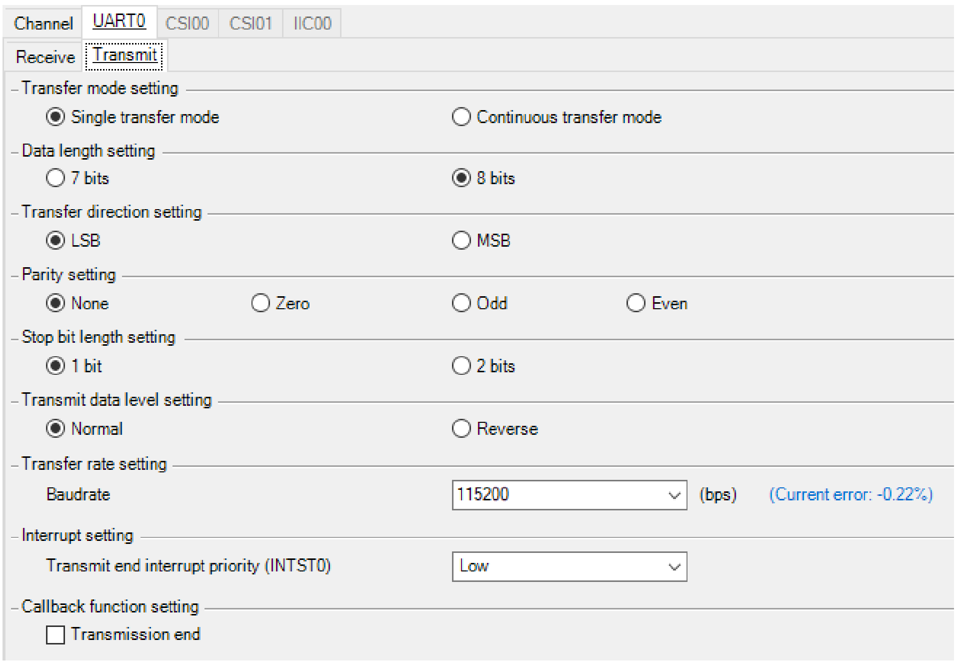
Reception end interrupt priority (INTSR0): Low

☐ Reception error interrupt priority (INTSRE0): Low

Callback function setting

☐ Reception end ☐ Reception error

Figure 4.24 SAU CG Configuration (2/3)



Transfer mode setting
(●) Single transfer mode ( ) Continuous transfer mode

Data length setting
( ) 7 bits (●) 8 bits

Transfer direction setting
(●) LSB ( ) MSB

Parity setting
(●) None ( ) Zero ( ) Odd ( ) Even

Stop bit length setting
(●) 1 bit ( ) 2 bits

Transmit data level setting
(●) Normal ( ) Reverse

Transfer rate setting
Baudrate 115200 (bps) (Current error: -0.22%)

Interrupt setting
Transmit end interrupt priority (INTST0) Low

Callback function setting
☐ Transmission end

Figure 4.25 SAU CG Configuration (3/3)

5. Application Example

A sample application program, testmain.c, is stored in the "sample" folder (see Table 4.2).
This sample application program accesses an access point and sends and receives data to and from a TCP server as a sample usage of the control software. Figure 5.1 shows an example of system configuration for the sample application program.

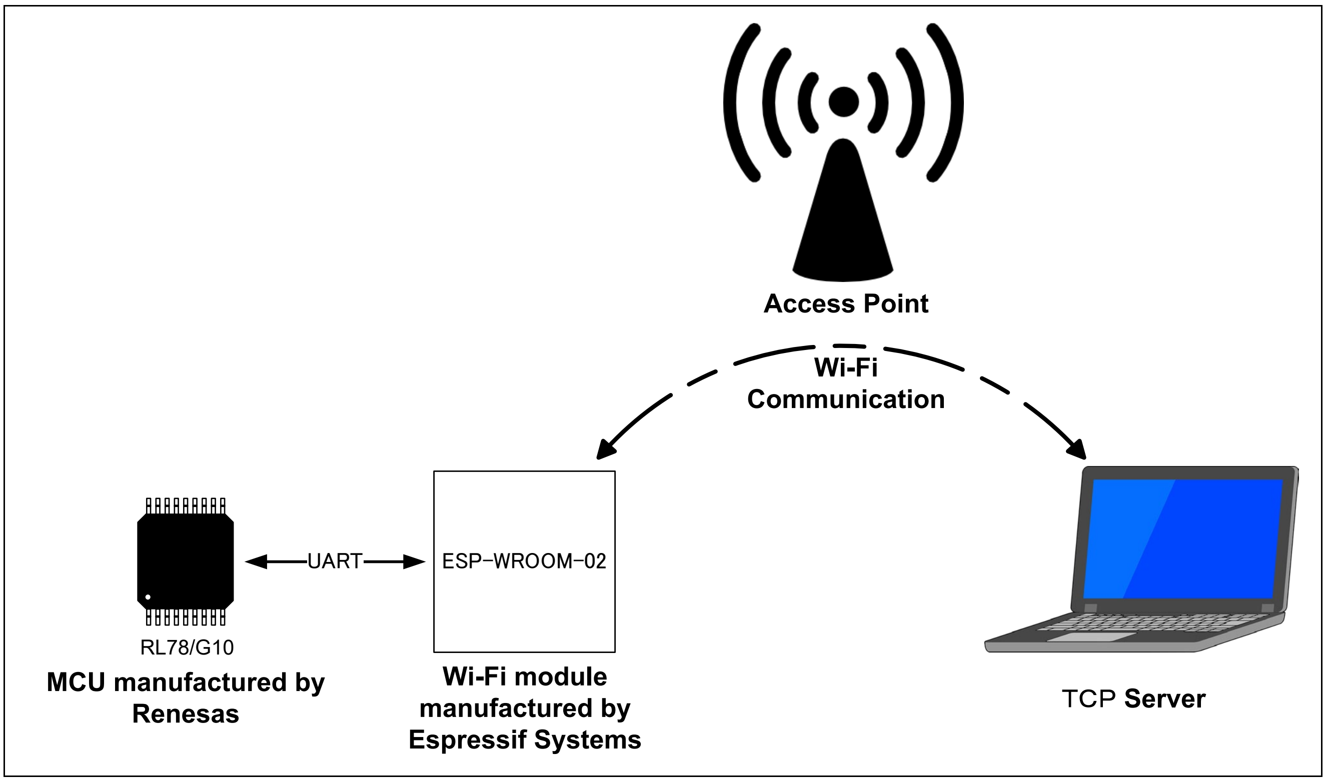

Access Point
Wi-Fi Communication
UART
ESP-WROOM-02
RL78/G10
MCU manufactured by Renesas
Wi-Fi module manufactured by Espressif Systems
TCP Server

Figure 5.1 Example of System Configuration for the Sample Application Program

5.1 Overview of the Sample Application Program

An overview of the sample application program is given below.

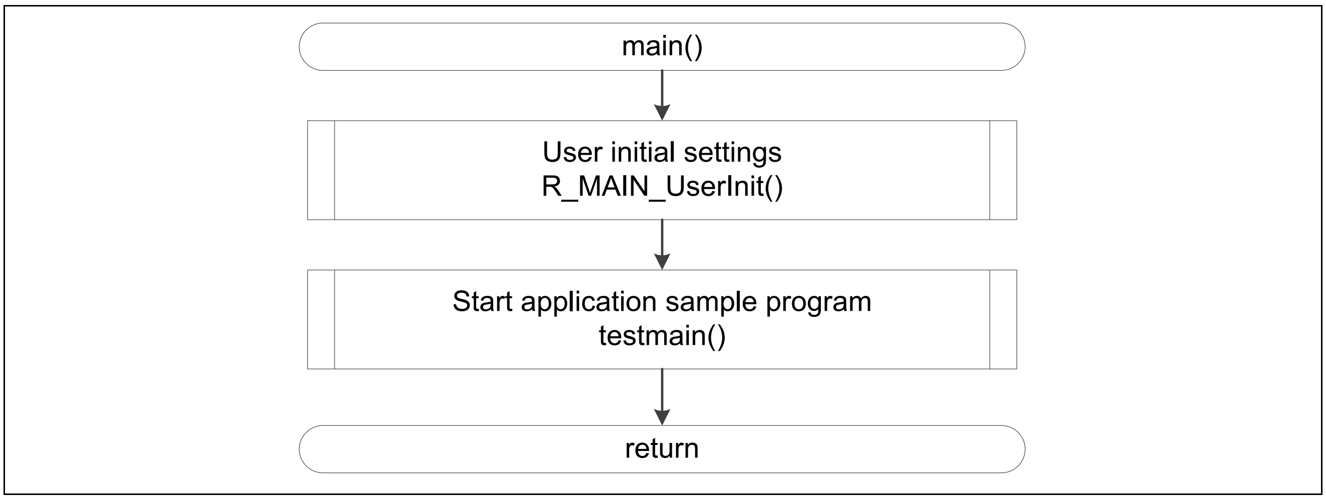
(1) The RL78/G10 makes initial settings of the Wi-Fi module through UART communication.
(2) The RL78/G10 connects the Wi-Fi module to an access point.
(3) The RL78/G10 gets the IP address from the Wi-Fi module.
(4) The RL78/G10 gets the MAC address from the Wi-Fi module.
(5) The RL78/G10 connects the Wi-Fi module to a TCP server.
(6) The RL78/G10 sends data to the TCP server.
(7) The RL78/G10 waits for a response from the TCP server.
(8) The RL78/G10 stores the received data in a buffer (on-chip RAM).
(9) The RL78/G10 waits for one second by using software.
(10) Steps (6) to (9) are repeated.

5.2 Flowchart of Application Example

Function specifications and flowcharts of the sample application program provided together with this application note are shown in this section.

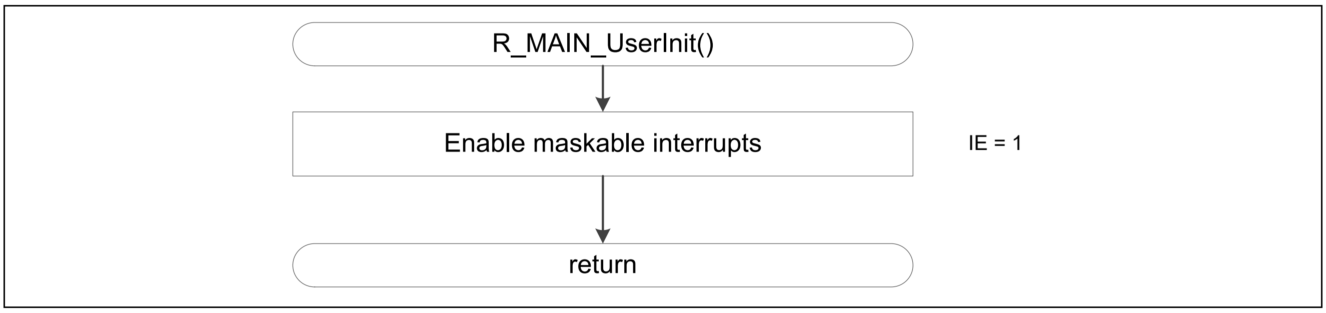
5.2.1 Main processing

[Function Name]	main
Outline	Main processing
Header	None
Declaration	void main(void);
Description	After calling R_MAIN_UserInit(), start application sample program
Arguments	None
Return value	None
Remarks	None



5.2.2 Starting UART Transmission Processing

[Function Name]	R_MAIN_UserInit
Outline	User initial setting
Header	None
Declaration	static void R_MAIN_UserInit(void);
Description	Enable interrupt processing by the EI instruction
Arguments	None
Return value	None
Remarks	None



5.2.3 Application sample function

[Function Name]	testmain
Outline	Application sample program
Header	None
Declaration	void testmain(void);
Description	Initialize the Wi-Fi module, then execute TCP transmission and reception
Arguments	None
Return value	None
Remarks	None

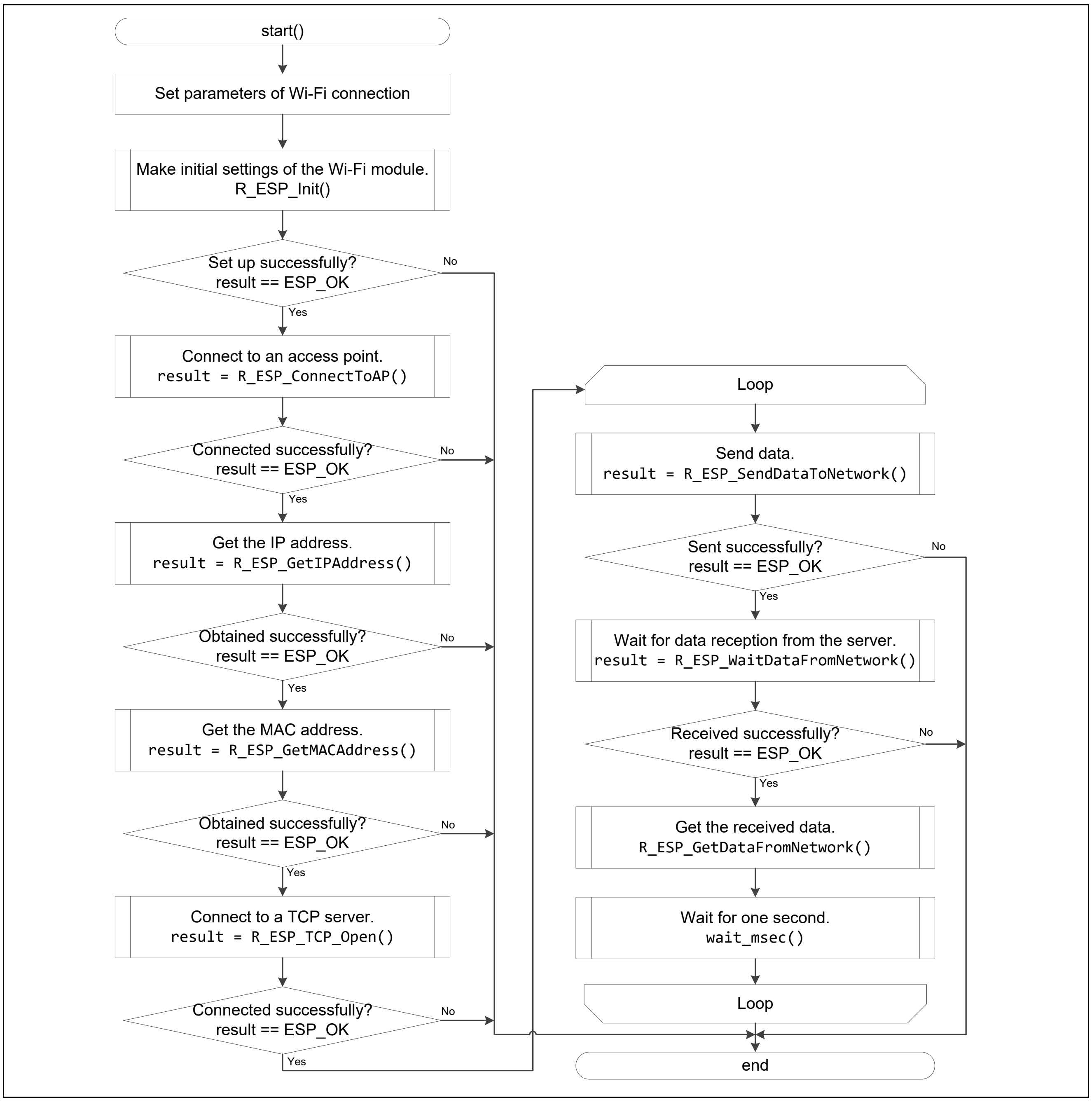


Figure 5.2 Flowchart of Sample Application Program

5.3 Modification of Sample Code

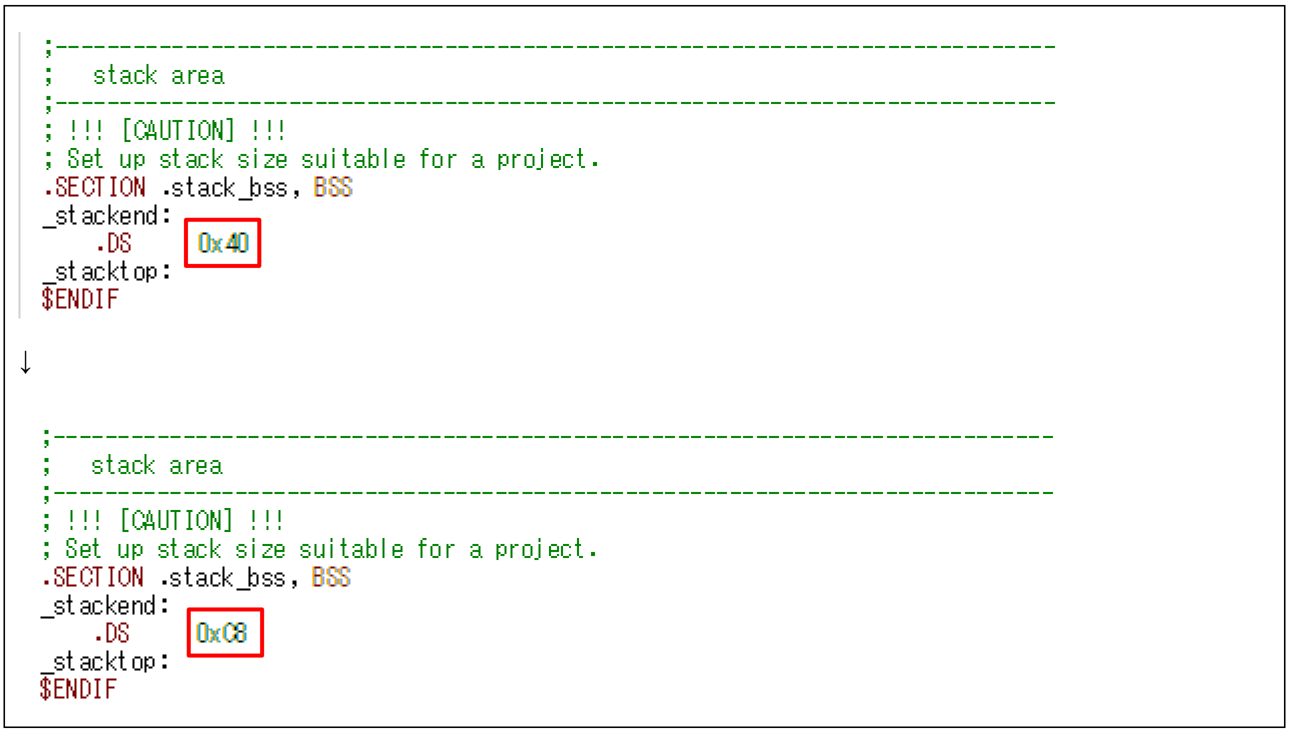
If code generation is to be redone, it may be necessary to modify the file and project as follows.

Target environment : CS+ version, e2studio version

File name : cstart.asm

Correction point : Place where "stack area" is written

DS 0x40 generated in cstart.asm is changed to an arbitrary stack size.



```
;-------------------------------------------------------------------------------
;   stack area
;-------------------------------------------------------------------------------
; !!! [CAUTION] !!!
; Set up stack size suitable for a project.
.SECTION .stack_bss, BSS
_stackend:
    .DS     0x40
_stacktop:
$ENDIF
```

↓

```
;-------------------------------------------------------------------------------
;   stack area
;-------------------------------------------------------------------------------
; !!! [CAUTION] !!!
; Set up stack size suitable for a project.
.SECTION .stack_bss, BSS
_stackend:
    .DS     0xC8
_stacktop:
$ENDIF
```

Target environment : IAR version

Fixes : Where to enter the stack size

From the project options, go to "General Options" > "Stack/Heap" > "Override default " > "Heap Size (bytes)" and enter the desired stack size.

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	July 4, 2019	-	First edition
1.10	June 24, 2022	6	Updated Operation check condition
		9	Updated required memory size
		10	Updated directory structure
		37	Added corrections to sample code

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.