V850 Microcontrollers

V850ES/Jx3-L

UART Communication Using UARTA

R01AN0780EJ0100
Rev. 1.00
Dec 7, 2011

Summary

This application note describes how to perform UART communication with another device by using asynchronous serial interface A (UARTA) incorporated in V850ES/Jx3-L Series microcontrollers.

In the processing described in this application note, ASCII characters sent from another device are analyzed and a response is returned.

Target devices

V850ES/JC3-L

V850ES/JE3-L

V850ES/JF3-L

V850ES/JG3-L

When this application note is applied to other microcontrollers, make the necessary changes according to the specifications of the microcontroller and verify them thoroughly.

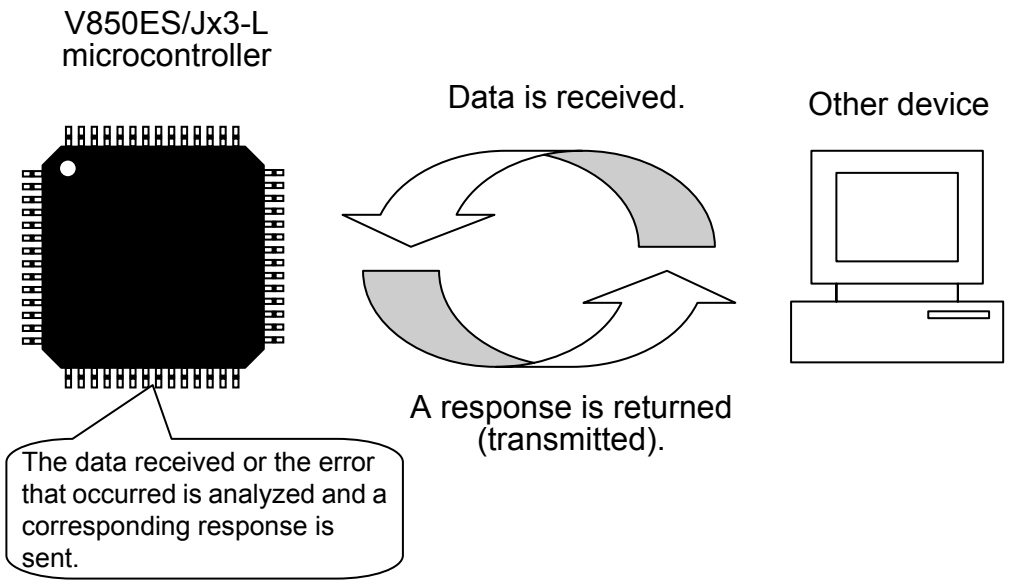
CONTENTS

1. SPECIFICATIONS

This application note shows examples of using asynchronous serial interface A (UARTA). In these examples, ASCII characters sent from another device are analyzed and a response is returned by using UART communication.

An overview of the operations performed by the sample code is shown below.

[Operation Overview]



If data is received normally when performing UART communication with another device, the received data is stored by the microcontroller and a value corresponding to the received data is transmitted to the other device. See Table 1.1 for the values that are transmitted by the microcontroller.

If an error occurs while data is being received, the microcontroller checks what type of error occurred and transmits a value corresponding to that error. See Table 1.2 for the values that are transmitted by the microcontroller.

Table 1.1 Values Returned (Transmitted) When Data Is Received

Received data	Returned (transmitted) value
T (54H)	O (4FH), K (4BH), "CR" (0DH), "LF" (0AH)
t (74H)	o (6FH), k (6BH), "CR" (0DH), "LF" (0AH)
Other than above	U (55H), C (43H), "CR" (0DH), "LF" (0AH)

Table 1.2 Values Returned (Transmitted) When an Error Occurs

Error	Returned (transmitted) value
Parity error	P (50H), E (45H), "CR" (0DH), "LF" (0AH)
Framing error	F (46H), E (45H), "CR" (0DH), "LF" (0AH)
Overrun error	O (4FH), E (45H), "CR" (0DH), "LF" (0AH)

Table 1.3 shows the peripheral functions used and their applications, and Table 1.4 shows the settings used for UARTA communication.

Table 1.3 Peripheral Functions Used and Their Applications

Peripheral function	Application
Asynchronous serial interface A (UARTA)	UART communication with another device

Table 1.4 UART Communication Settings

Item	Setting
Baud rate	38,400 bps
Data length	8 bits
Stop bit length	1 bit
Parity	Even
Transfer direction	LSB first

Remark In the processing described in this application note, UART communication is performed by using asynchronous serial interface A (UARTA), but communication can also be performed in the same way by using asynchronous serial interface C (UARTC).

Note that the number of channels used for the UART interface in V850ES/Jx3-L Series microcontrollers differs depending on the product, as shown in Table 1.5 below.

Table 1.5 Differences in Available UART Channels

Name	Part name	Flash memory/ RAM	Number of UART channels
V850ES/JC3-L (40-pin)	μPD70F3797	16 KB/8 KB	UARTA: 2 channels
	μPD70F3798	32 KB/8 KB	
	μPD70F3799	64 KB/8 KB	
	μPD70F3800	128 KB/8 KB	
	μPD70F3838	256 KB/16 KB	
V850ES/JC3-L (48-pin)	μPD70F3801	16 KB/8 KB	UARTA: 3 channels
	μPD70F3802	32 KB/8 KB	
	μPD70F3803	64 KB/8 KB	
	μPD70F3804	128 KB/8 KB	
	μPD70F3839	256 KB/16 KB	
V850ES/JE3-L (64-pin)	μPD70F3805	16 KB/8 KB	UARTA: 3 channels
	μPD70F3806	32 KB/8 KB	
	μPD70F3807	64 KB/8 KB	
	μPD70F3808	128 KB/8 KB	
	μPD70F3840	256 KB/16 KB	
V850ES/JF3-L (80-pin)	μPD70F3735	128 KB/8 KB	UARTA: 3 channels
	μPD70F3736	256 KB/16 KB	
V850ES/JG3-L	μPD70F3737	128 KB/8 KB	UARTA: 3 channels
	μPD70F3738	256 KB/16 KB	
	μPD70F3792	384 KB/32 KB	UARTA: 6 channels
	μPD70F3793	512 KB/40 KB	UARTC: 1 channel
	μPD70F3841	768 KB/80 KB	
	μPD70F3842	1 MB/80 KB	
V850ES/JG3-L (with USB)	μPD70F3794	256 KB/40 KB	UARTA: 6 channels
	μPD70F3795	384 KB/40 KB	UARTC: 1 channel
	μPD70F3796	512 KB/40 KB	
	μPD70F3843	768 KB/80 KB	
	μPD70F3844	1 MB/80 KB	

2. OPERATION-VERIFIED CONDITIONS

The sample code described in this application note has been verified to operate correctly under the following conditions:

Table 2.1 Operation-Verified Conditions

Item	Description
Microcontroller used	V850ES/JG3-L (with USB) (μPD70F3796GC)
Operating frequency	• CPU clock: 20 MHz • Peripheral clock: 20 MHz • Main clock oscillation frequency: 5 MHz
Operating voltage	3.0 V (2.7 V to 3.6 V) (when CPU clock operates at 20 MHz)
Integrated development environment	Renesas Electronics CubeSuite+ V1.00.01
C compiler	Renesas Electronics CA850 V3.50
Board used	V850ES/JG3-L (USB) target board (QB-V850ESJG3LUSB-TB)
Tool used (other device)	Hyper terminal
Tool settings	• Bits per second: 38,400 • Data bits: 8 • Parity: Even • Stop bits: 1 • Flow control: None

3. RELATED APPLICATION NOTE

The following application note is related to this document. Refer to this note together with this document.

V850ES/Jx3-L Sample Program (Initial Settings) LED Lighting Switch Control (R01ANxxxxEJ0100)

4. HARDWARE DESCRIPTION

4.1 Pins Used

Table 4.1 shows the pins used and their functions.

Table 4.1 Pins Used and Their Functions

Pin name	I/O	Description
P90/TXDA1	Output	UARTA1 transmission output
P91/RXDA1	Input	UARTA1 reception input

4.2 Hardware Configuration Example

Figure 4.1 shows an example of the hardware configuration used in this application note.

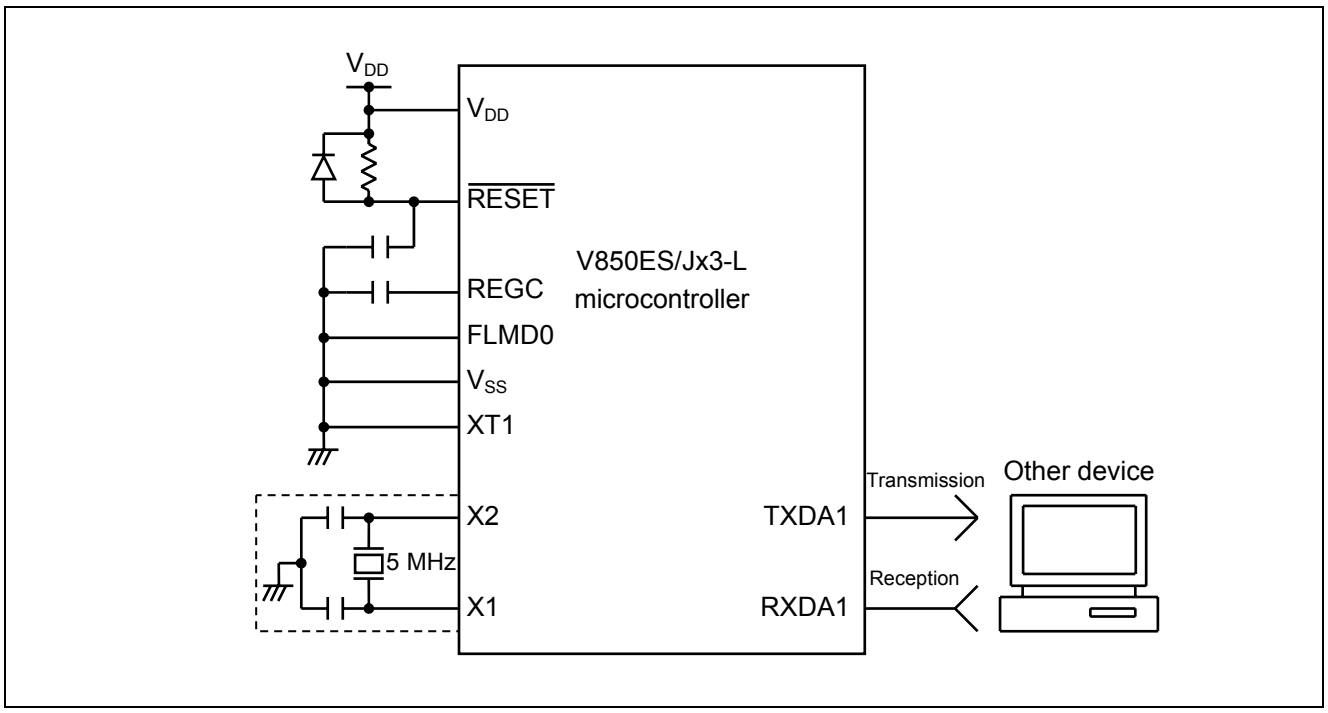


Figure 4.1 Hardware Configuration

Cautions 1. Use V_{DD} in a range of 2.7 V < V_{DD} ≤ 3.6 V. Note that this range applies when the CPU clock is operating at 20 MHz.

2. Set the EV_{DD} pin and AV_{REF0} pin to the same potential as V_{DD}.

3. Set the EV_{SS} pin to the same potential as GND.

4. Connect REGC to GND via a capacitor (recommended value: 4.7 μF).

5. Connect the FLMD0 pin to GND in the normal operation mode.

6. Leave the unused ports open because they are handled as output ports.

7. When using the main clock oscillator, wire as follows in the area enclosed by the dotted lines in the above figure to avoid an adverse effect from wiring capacitance:

- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with the other signal lines.**
- **Do not route the wiring near a signal line through which a high fluctuating current flows.**
- **Always make the ground point of the oscillator capacitor the same potential as V_{SS}.**
- **Do not ground the capacitor to a ground pattern through which a high current flows.**
- **Do not fetch signals from the oscillator.**

5. SOFTWARE DESCRIPTION

5.1 Operation Overview

In the processing described in this application note, UART communication is performed with another device by using asynchronous serial interface A (UARTA). The microcontroller receives data from the other device and returns (transmits) a value in response. If an error occurs during reception, the microcontroller returns a value corresponding to the error.

Setting conditions

- Set the baud rate to 38,400 bps.
- Set the data length to 8 bits.
- Set the number of stop bits to 1.
- Specify even parity.
- Set the transfer direction to LSB first.
- Specify use of the reception completion interrupt (INTUA1R) and transmission enable interrupt (INTUA1T).

(1) Specify the initial settings of asynchronous serial interface A (UARTA).

(2) Enable the reception completion interrupt (INTUA1R), and shift to the HALT mode.

(3) Specify that the receive data or error status be analyzed and a corresponding value be returned (transmitted) when a reception completion interrupt (INTUA1R) is generated by the reception of data from another device or the occurrence of an error during reception. Enable the transmission enable interrupt (INTUA1T) and transmit the response value to the other device.

(4) Generate the transmission enable interrupt (INTUA1T) after every byte of data that is transmitted in order to initiate transmission of the next byte. Once all the data has been transmitted, disable the transmission enable interrupt (INTUA1T) and shift to HALT mode.

5.2 Option Byte Settings

Table 5.1 shows the option byte settings.

Table 5.1 Option Byte Settings

Address	Setting value	Description
0000007AH	00000101B	The operation clock for watchdog timer 2 (f_X/f_T/f_R) can be selected. INTWDT2/WDTRES mode can be selected.
		The internal oscillator can be stopped by software.
		The oscillation stabilization time is 2^{15}/fx.

5.3 Constants

Table 5.2.1 shows the constants used in the sample code. Table 5.2.2 shows the tables used in the sample code.

Table 5.2.1 Constants Used in Sample Code

Constant name	Setting value	Description
BUFFSIZE	16	Size of buffer for received data
TXDNUM	4	Number of bytes in transmitted data
CODE_NORMAL1	0x54	ASCII code “T”
CODE_NORMAL2	0x74	ASCII code “t”
CODE_ERRORP	0xFF	Code indicating parity error
CODE_ERRORF	0xFE	Code indicating framing error
CODE_ERRORO	0xFD	Code indicating overrun error
REP_NONE	0x00	No response status: There is no unresponded data.
REP_STORE	0x01	No response status: There is unresponded data.
REP_FULL	0x02	No response status: The unresponded data buffer is full.

Table 5.2.2 Tables Used in Sample Code

Table name	Setting value	Description
gStringOK	O K \r \n	Value transmitted when “T” is received
gStringok	o k \r \n	Value transmitted when “t” is received
gStringUC	U C \r \n	Value transmitted when other character code is received
gStringPE	P E \r \n	Value transmitted when parity error occurs
gStringFE	F E \r \n	Value transmitted when framing error occurs
gStringOE	O E \r \n	Value transmitted when overrun error occurs

5.4 Variables

Table 5.3 shows the global variables.

Table 5.3 Global Variables

Type	Variable name	Description	Function used
UCHAR*	gpUarta1TxAddress	Pointer for transmitted data	MD_INTUA1T MD_TxControl
USHORT	gUarta1TxCnt	Number of data items transmitted	MD_INTUA1T MD_TxControl
UCHAR*	gpUarta1RxAddress	Pointer for receive buffer	MD_INTUA1R
USHORT	gUarta1RxCnt	Number of data items received	MD_INTUA1R MD_TxControl
USHORT	gUarta1RxLen	Maximum number of data items that can be received	MD_INTUA1R
UCHAR	gRxBuffer	Receive buffer	MD_INTUA1R
UCHAR	gTxPointer	Pointer to responded data in receive buffer	MD_INTUA1T MD_TxControl
UCHAR	gRepStatus	No response status	MD_INTUA1R MD_INTUA1T MD_TxControl

5.5 Functions

Table 5.4 shows the functions.

Table 5.4 Functions

Function name	Overview
MD_UartVarInit	Specification of initial values of variables used for communication
MD_TxControl	Transmission control processing
MD_INTUA1R	INTUA1R interrupt processing
MD_INTUA1T	INTUA1T interrupt processing

5.6 Function Specifications

This section shows the function specifications of the sample code.

Function name: `MD_UartVarInit`

Overview	Specification of initial values of variables used for communication
Header	`CG_serial.h`
Declaration	`void MD_UartVarInit( void )`
Description	Specifies the initial values of the variables used to perform communication.
Parameter	None
Return value	None
Remark	None

Function name: `MD_INTUA1R`

Overview	INTUA1R interrupt processing
Header	`CG_serial.h`
Declaration	`__interrupt void MD_INTUA1R(void)`
Description	Saves the received data or the status of the error that occurred in the buffer. However, if the buffer is full of unresponded data, the received data or error status is not saved.
Parameter	None
Return value	None
Remark	None

Function name: `MD_INTUA1T`

Overview	INTUA1T interrupt processing
Header	`CG_serial.h`
Declaration	`__interrupt void MD_INTUA1T(void)`
Description	Transmits data.
Parameter	None
Return value	None
Remark	None

Function name: `MD_TxControl`

Overview	Transmission control processing
Header	`CG_serial.h`
Declaration	`void MD_TxControl( void )`
Description	Specifies the value to be returned in accordance with the received data or error status. Also starts and stops transmission processing.
Parameter	None
Return value	None
Remark	None

5.7 State Transition Diagram

In this sample code, settings such as selecting the clock frequency, stopping watchdog timer 2, specifying the I/O port settings, and specifying the UARTA settings are performed during initialization.

After initialization is complete, the system shifts to HALT mode. When UART communication with another device starts, the microcontroller analyzes the data received or the status of the error that occurred and transmits a corresponding response to the other device. Once transmission of the response data is complete, the system shifts back to HALT mode.

The details are shown in the following state transition diagram (state chart).

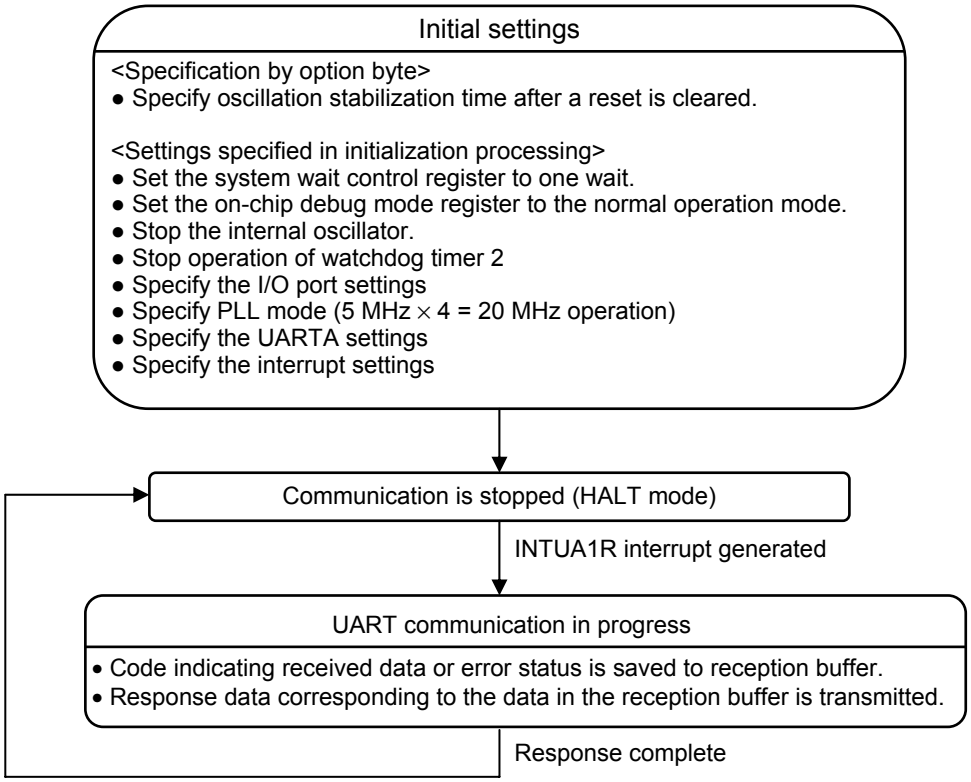


Figure 5.1 State Transition Diagram

5.8 Flow Charts

5.8.1 Main processing

Figure 5.2 shows the flow of main processing.

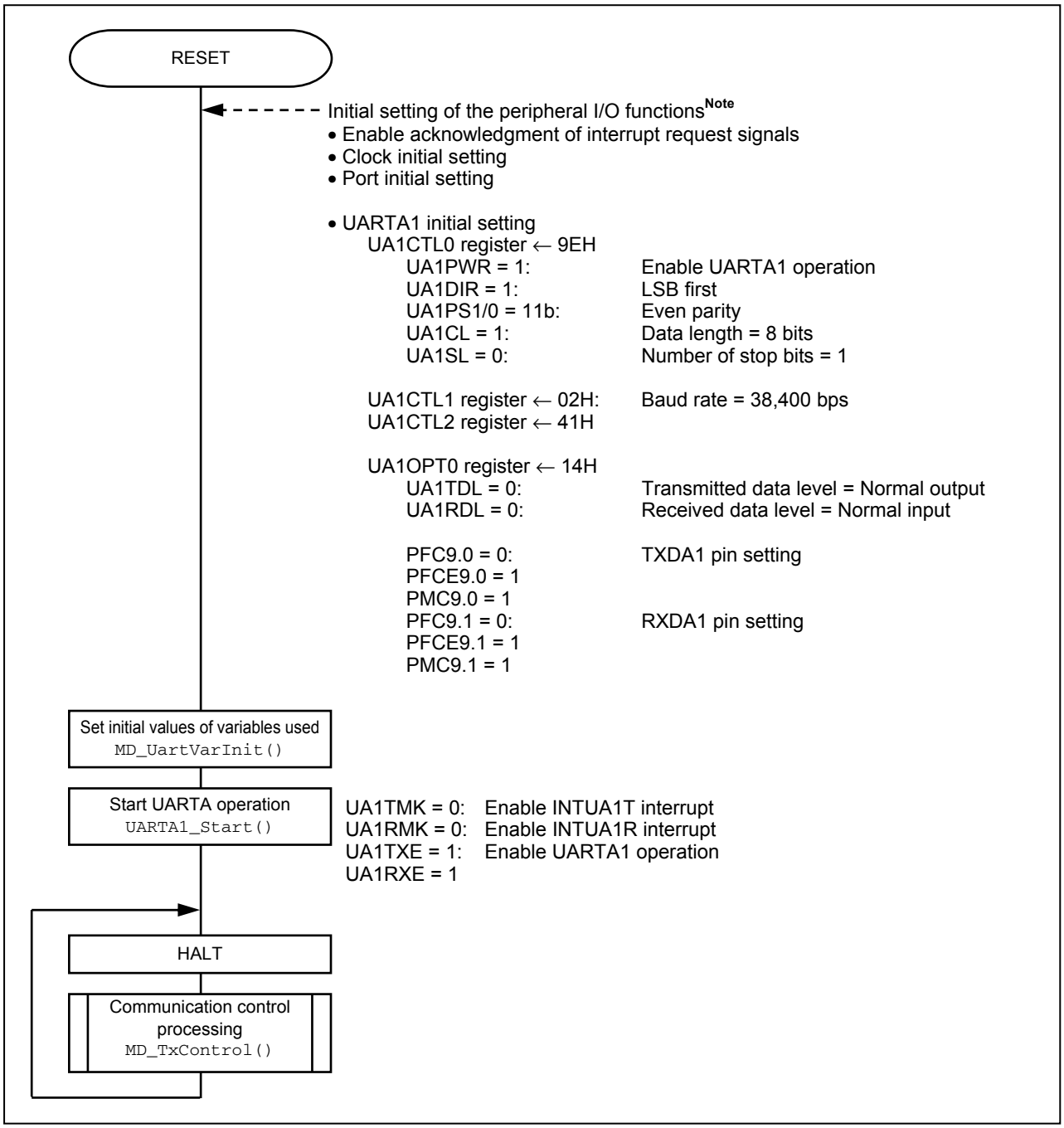


Figure 5.2 Main Processing

Note Initialization of the peripheral I/O functions is performed in the startup processing (`CG_start.s`).

5.8.2 Interrupt processing

Figure 5.3.1 shows the flow of INTUA1T interrupt processing, and Figure 5.3.2 shows the flow of INTUA1R interrupt processing.

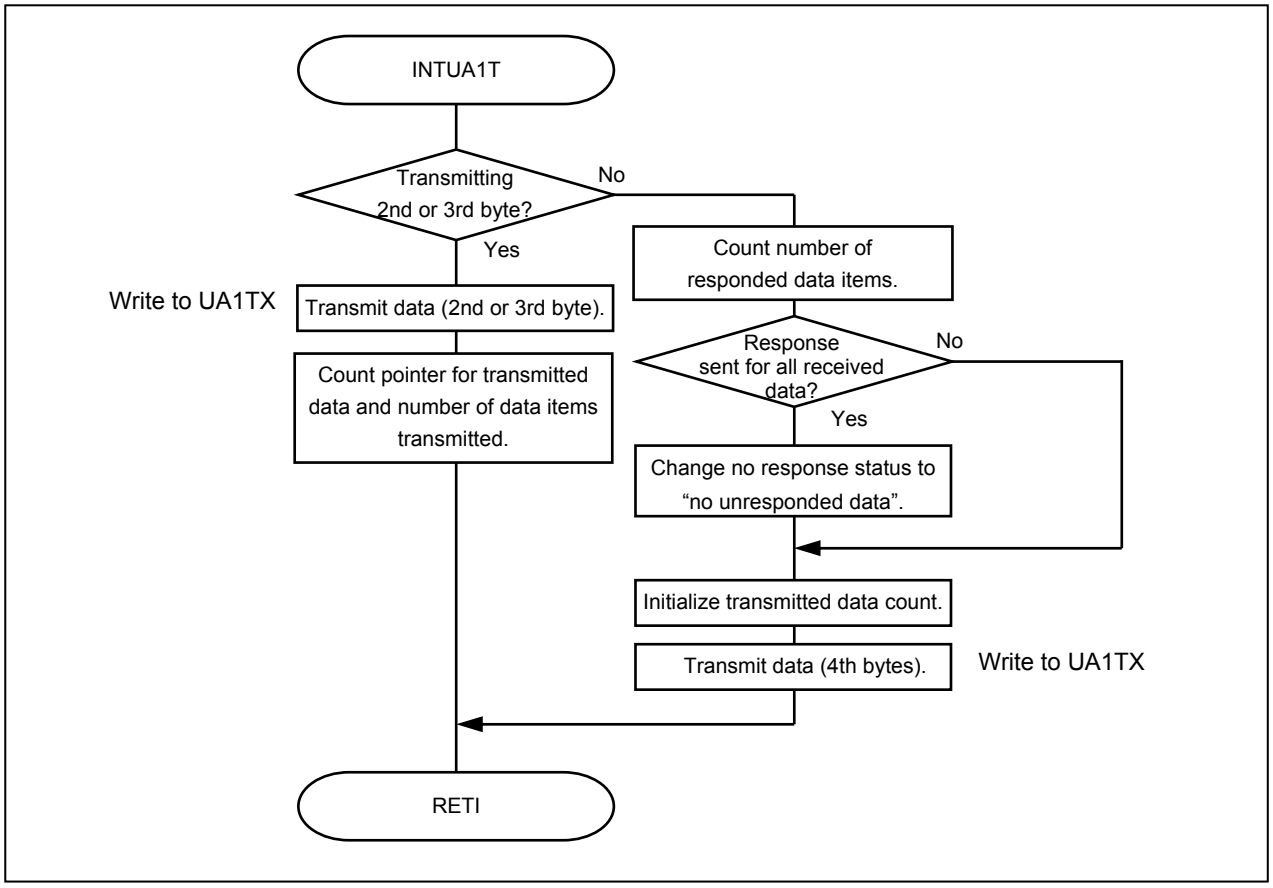


Figure 5.3.1 INTUA1T Interrupt Processing

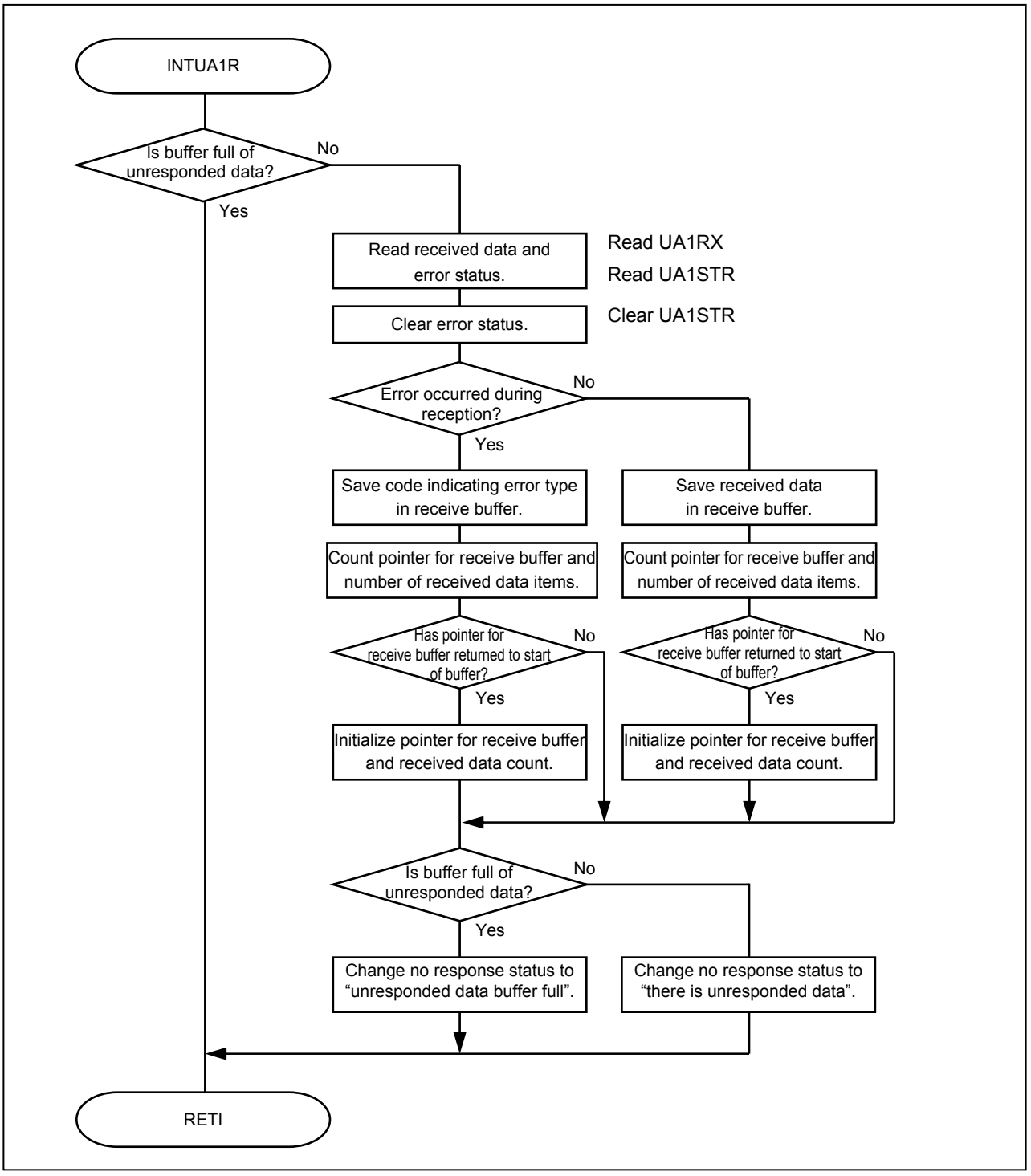


Figure 5.3.2 INTUA1R Interrupt Processing

5.8.3 Communication control processing

Figure 5.4 shows the flow of communication control processing.

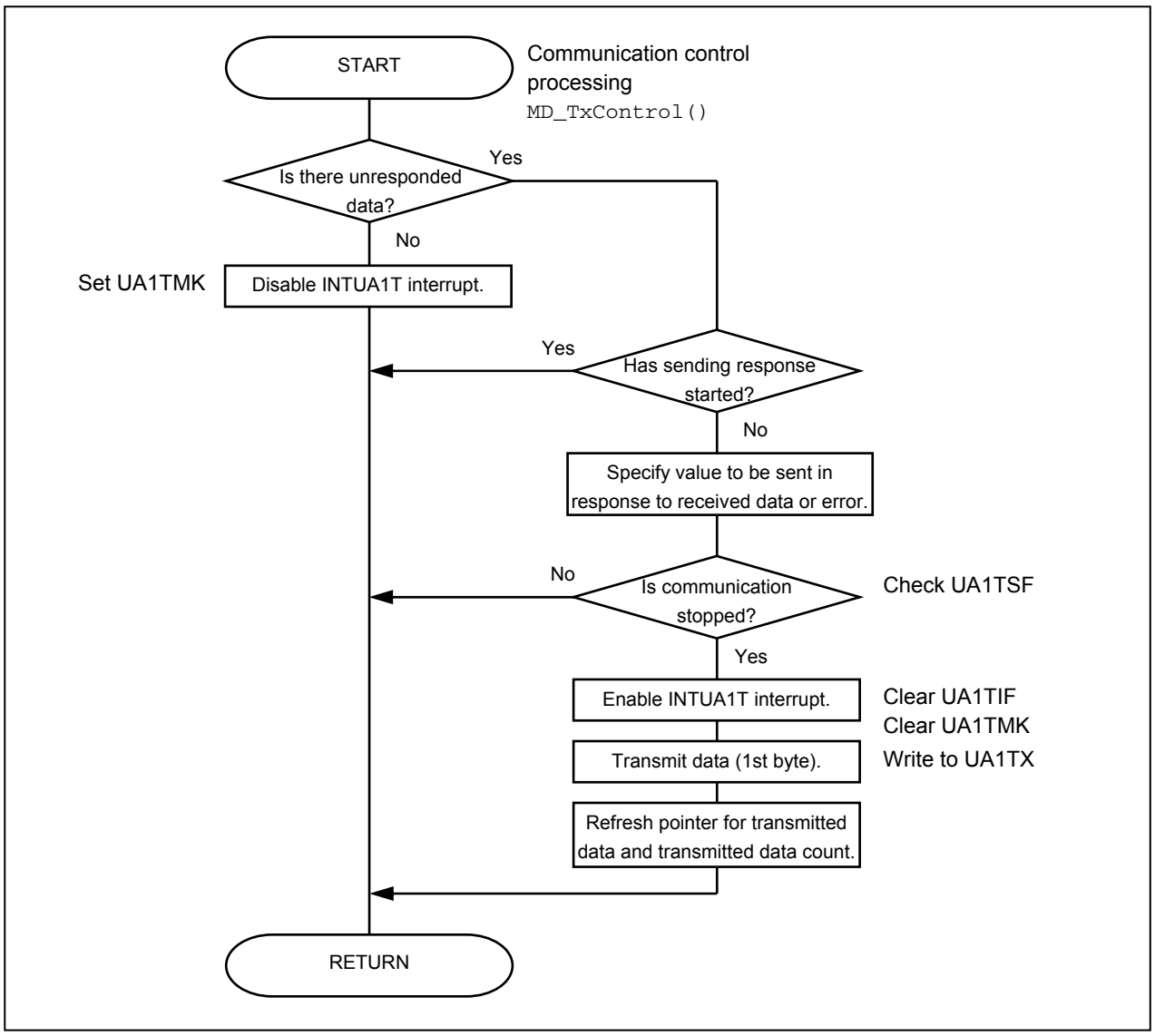
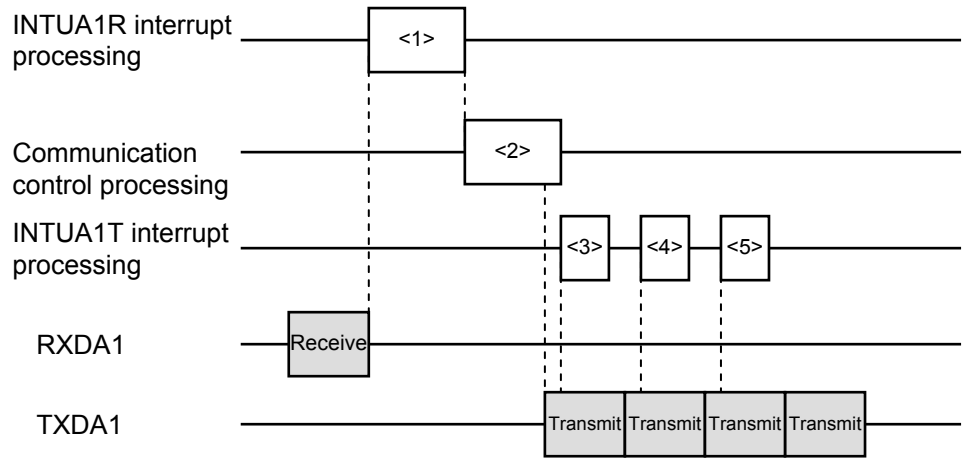


Figure 5.4 Communication Control Processing

Figure 5.5 shows the timing from receiving data to sending a response.



<1> The INTUA1R interrupt is generated and processing starts. The received data is stored in the receive buffer.
<2> The data in the receive buffer is analyzed and the corresponding response data is selected. The INTUA1T interrupt is enabled and the 1st byte of the response data is written to UA1TX. The system then shifts to HALT mode.
<3> The INTUA1T interrupt is generated and processing starts. The 2nd byte of the response data is written to UA1TX. The system then shifts to HALT mode.
<4> The INTUA1T interrupt is generated and processing starts. The 3rd byte of the response data is written to UA1TX. The system then shifts to HALT mode.
<5> The INTUA1T interrupt is generated and processing starts. The 4th byte of the response data is written to UA1TX. The INTUA1T interrupt is disabled and the system shifts to HALT mode.

Figure 5.5 Timing of a 16-Byte Consecutive Transmission/Reception

5.9 UARTA Settings

5.9.1 UARTA1 control register 0 (UA1CTL0)

The UA1CTL0 register is an 8-bit register that controls the serial transfer operation of UARTA1.

This register can be read or written in 8-bit or 1-bit units.

Reset sets this register to 10H.

UARTA1 control register 0 (UA1CTL0)
Address: FFFFFA10H

7	6	5	4	3	2	1	0
UA1PWR	UA1TXE	UA1RXE	UA1DIR	UA1PS1	UA1PS0	UA1CL	UA1SL

UA1PWR	UARTA1 operation disable/enable
0	Disable UARTA1 operation (UARTA1 is reset asynchronously).
1	Enable UARTA1 operation.

UA1TXE	Transmission disable/enable
0	Disable transmission.
1	Enable transmission.

UA1RXE	Reception disable/enable
0	Disable reception.
1	Enable reception.

UA1DIR	Data transfer order
0	MSB first
1	LSB first

UA1PS1	UA1PS0	Transmission parity selection	Reception parity selection
0	0	Do not output parity.	Receive data with no parity.
0	1	Output 0 parity.	Receive data with 0 parity.
1	0	Output odd parity.	Judge the parity as odd.
1	**1**	Output even parity.	Judge the parity as even.

UA1CL	Character length of 1 data frame during transmission and reception
0	7 bits
1	8 bits

UA1SL	Number of stops bits in transmitted data
0	1 bit
1	2 bits

Figure 5.6.1 Format of UARTA1 Control Register 0 (UA1CTL0)

5.9.2 UARTA1 control register 1 (UA1CTL1)

The UA1CTL1 register is an 8-bit register that selects the UARTA1 base clock.

This register can be read or written in 8-bit units.

Reset sets this register to 00H.

Caution Clear the UA1CTL0.UA1PWR bit to 0 before rewriting the UA1CTL1 register.

UARTA1 control register 1 (UA1CTL1)
Address: FFFFFA11H

7	6	5	4	3	2	1	0
0	0	0	0	UA1CKS3	UA1CKS2	UA1CKS1	UA1CKS0

UA1CKS3	UA1CKS2	UA1CKS1	UA1CKS0	Communication clock (f_{UCLK})
0	0	0	0	f_{XX}
0	0	0	1	$f_{XX}/2$
0	**0**	**1**	**0**	$f_{XX}/4$
0	0	1	1	$f_{XX}/8$
0	1	0	0	$f_{XX}/16$
0	1	0	1	$f_{XX}/32$
0	1	1	0	$f_{XX}/64$
0	1	1	1	$f_{XX}/128$
1	0	0	0	$f_{XX}/256$
1	0	0	1	$f_{XX}/512$
1	0	1	0	$f_{XX}/1024$
1	0	1	1	External clock (ASCKA0 pin)
Other than above				Setting prohibited

Remark fxx: Main clock frequency

Figure 5.6.2 Format of UARTA1 Control Register 1 (UA1CTL1)

5.9.3 UARTA1 control register 2 (UA1CTL2)

The UA1CTL2 register is an 8-bit register that selects the baud rate (serial transfer speed) clock of UARTA1.

The baud rate clock is generated by dividing the serial clock specified by this register by two.

This register can be read or written in 8-bit units.

Reset sets this register to FFH.

UARTA1 control register 2 (UA1CTL2)
Address: FFFFFA12H

7	6	5	4	3	2	1	0
UA1BRS7	UA1BRS6	UA1BRS5	UA1BRS4	UA1BRS3	UA1BRS2	UA1BRS1	UA1BRS0

UA1 BRS7	UA1 BRS6	UA1 BRS5	UA1 BRS4	UA1 BRS3	UA1 BRS2	UA1 BRS1	UA1 BRS0	Default (k)	Serial clock
0	0	0	0	0	0	X	X	X	Setting prohibited
0	0	0	0	0	1	0	0	4	$f_{UCLK}/4$
0	0	0	0	0	1	0	1	5	$f_{UCLK}/5$
0	0	0	0	0	1	1	0	6	$f_{UCLK}/6$
...	...	...	...	...	...	...	...	...	...
0	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**65**	**$f_{UCLK}/65$**
...	...	...	...	...	...	...	...	...	...
1	1	1	1	1	1	0	0	252	$f_{UCLK}/252$
1	1	1	1	1	1	0	1	253	$f_{UCLK}/253$
1	1	1	1	1	1	1	0	254	$f_{UCLK}/254$
1	1	1	1	1	1	1	1	255	$f_{UCLK}/255$

f_{UCLK}: Clock frequency selected by the UA1CTL1.UA1CKS3 to UA1CTL1.UA1CKS0 bits

Figure 5.6.3 Format of UARTA1 Control Register 2 (UA1CTL2)

Representative examples of baud rate settings are shown below.

Table 5.5.1 Baud Rate Generator Setting Data

Baud Rate (bps)	f_{XX} = 20 MHz			f_{XX} = 16 MHz			f_{XX} = 10 MHz		
	UA1CTL1	UA1CTL2	ERR (%)	UA1CTL1	UA1CTL2	ERR (%)	UA1CTL1	UA1CTL2	ERR (%)
300	08H	82H	0.16	07H	D0H	0.16	07H	82H	0.16
600	07H	82H	0.16	06H	D0H	0.16	06H	82H	0.16
1200	06H	82H	0.16	05H	D0H	0.16	05H	82H	0.16
2400	05H	82H	0.16	04H	D0H	0.16	04H	82H	0.16
4800	04H	82H	0.16	03H	D0H	0.16	03H	82H	0.16
9600	03H	82H	0.16	02H	D0H	0.16	02H	82H	0.16
19200	02H	82H	0.16	01H	D0H	0.16	01H	82H	0.16
31250	01H	A0H	0	01H	80H	0	00H	A0H	0
38400	01H	82H	0.16	00H	D0H	0.16	00H	82H	0.16
76800	00H	82H	0.16	00H	68H	0.16	00H	41H	0.16
153600	00H	41H	0.16	00H	34H	0.16	00H	21H	–1.36
312500	00H	20H	0	00H	1AH	–1.54	00H	10H	0
625000	00H	10H	0	00H	0DH	–1.54	00H	08H	0

Remark f_{XX}: Main clock frequency

ERR: Baud rate error (%)

5.9.4 UARTA1 status register (UA1STR)

The UA1STR register is an 8-bit register that displays the UARTA1 transfer status and reception error contents.

This register can be read or written in 8-bit or 1-bit units, but the UA1TSF bit is a read-only bit, while the UA1PE, UA1FE, and UA1OVE bits can be both read and written. However, these bits can only be cleared by writing 0; they cannot be set by writing 1 (even if 1 is written to them, the previous value is retained).

The conditions for clearing the UA1STR register are shown below.

Table 5.5.2 Conditions for Clearing STR Register

Register/Bit	Conditions for Clearing
UA1STR register	• Reset • UA1CTL0.UA1PWR bit = 0
UA1TSF bit	• UA1CTL0.UA1TXE bit = 0
UA1PE, UA1FE, UA1OVE bits	• Writing 0 • UA1CTL0.UA1RXE bit = 0

UARTA1 status register (UA1STR)
Address: FFFFFA14H

7	6	5	4	3	2	1	0
UA1TSF	0	0	0	0	UA1PE	UA1FE	UA1OVE

UA1TSF	Transfer status flag
0	The transmit shift register does not have data. • When the UA1PWR bit or the UA1TXE bit has been set to 0. • When, following transfer completion, there was no next data transfer from UA1TX register.
1	The transmit shift register has data. (Write to UA1TX register)

UA1PE	Parity error flag
0	• When the UA1PWR bit or the UA1RXE bit has been set to 0. • When 0 has been written.
1	The received parity bit does not match the specified parity.

UA1FE	Framing error flag
0	• When the UA1PWR bit or the UA1RXE bit has been set to 0. • When 0 has been written.
1	When no stop bit is detected during reception.

UA1OVE	Overrun error flag
0	• When the UA1PWR bit or the UA1RXE bit has been set to 0. • When 0 has been written.
1	When receive data has been set to the UA1RX register and the next receive operation is completed before that receive data has been read.

Figure 5.6.4 Format of UARTA1 Status Register (UA1STR)

5.9.5 UARTA1 receive data register (UA1RX)

The UA1RX register is an 8-bit buffer register that stores parallel data converted by the receive shift register.

The data stored in the receive shift register is transferred to the UA1RX register when 1 character of data has been received.

This register is read-only, in 8-bit units.

In addition to reset input, the UA1RX register can be set to FFH by clearing the UA1CTL0.UA1PWR bit to 0.

UARTA1 receive data register (UA1RX)
Address: FFFFFA16H

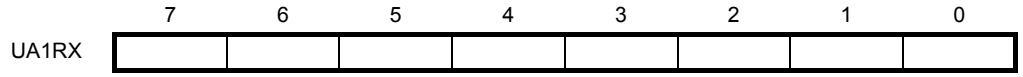


Figure 5.6.5 Format of UARTA1 Receive Data Register (UA1RX)

5.9.6 UARTA1 transmit data register (UA1TX)

The UA1TX register is an 8-bit register used to specify the data to be transmitted.

Writing data to the UA1TX register with transmission enabled (UA1CTL0.UA1TXE bit = 1) triggers transmission. When transfer of the UA1TX register data to the UARTA1 transmit shift register is complete, the transmission enable interrupt request signal (INTUA1T) is generated.

This register can be read or written in 8-bit units.

Reset sets this register to FFH.

Caution Writing the UA1TX register with transmission enabled (UA1PWR bit = 1 and UA1TXE bit = 1) triggers transmission. If the same value as the one immediately before is written, therefore, the same data is transmitted twice. To write new transmit data during processing of the preceding data, wait until the transmission enable interrupt request signal (INTUA1T) has been generated. Note that even if transmission is enabled after data is written to the UA1TX register with transmission disabled (UA1PWR bit = 0 or UA1TXE bit = 0), transmission does not start.

UARTA1 transmit data register (UA1TX)
Address: FFFFFA17H

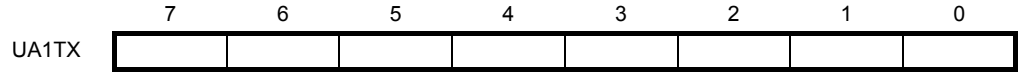


Figure 5.6.6 Format of UARTA1 Transmit Data Register (UA1TX)

5.9.7 UARTA1 pin settings

To use UARTA1, the UARTA1 transmit data output pin (TXDA1) and UARTA1 reception data input pin (RXDA1) must be set.

TXDA1 is set by using bit 0 of the port 9 function control register (PFC9), bit 0 of the port 9 function control expanded register (PFCE9), and bit 0 of the port 9 mode control register (PMC9).

RXDA1 is set by using bit 1 of the port 9 function control register (PFC9), bit 1 of the port 9 function control expanded register (PFCE9), and bit 1 of the port 9 mode control register (PMC9).

Table 5.5.7 TXDA1 Setting

PMC9.0	PFC9.0	PFCE9.0	Pin function specification
0	x	x	I/O port (P90)
1	**1**	**0**	TXDA1 output
1	1	1	SDA02 I/O

Remark x: don't care

Table 5.5.8 RXDA1 Setting

PMC9.1	PFC9.1	PFCE9.1	Pin function specification
0	x	x	I/O port (P91)
1	**1**	**0**	RXDA1 input
1	1	1	SCL02 I/O

Remark x: don't care

6. SAMPLE CODE

This chapter provides the sample code for the V850ES/JG3-L.

• CG_main.c

```
/*
***********************************************************************
* Copyright(C) 2008, 2011 Renesas Electronics Corporation
* RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY
* This program must be used solely for the purpose for which
* it was furnished by Renesas Electronics Corporation. No part of this
* program may be reproduced or disclosed to others, in any
* form, without the prior written permission of Renesas Electronics
* Corporation.
*
* This device driver was created by CodeGenerator for V850ES/Jx3
* 32-Bit Single-Chip Microcontrollers
* Filename: CG_main.c
* Abstract: This file implements main function.
* APIlib:   CodeGenerator for V850ES/Jx3 V1.00.01 [07 Jun 2011]
* Device:   uPD70F3796
* Compiler: CA850
* Creation date:  2011/07/20
***********************************************************************
*/

/*
***********************************************************************
** Pragma directive
***********************************************************************
*/
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*
***********************************************************************
** Include files
***********************************************************************
*/
```

```
#include "CG_macrodriver.h"
#include "CG_system.h"
#include "CG_port.h"
#include "CG_serial.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*#include "CG_userdefine.h"*/

/*
***********************************************************************************
**  Global define
***********************************************************************************
*/
/* Start user code for global. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/*
**-----------------------------------------------------------------------------
**
**  Abstract:
**    This function implements main function.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-----------------------------------------------------------------------------
*/
void main(void)
{
      /* Start user code. Do not edit comment generated here */

      MD_UartVarInit(); /* Variable initialization for UART */
      UARTA1_Start();   /* Start UARTA1 operation */

      /* Main loop */
      while (1U)
```

```
      {
            HALT();      /* Cpu standby */

            NOP();
            NOP();
            NOP();
            NOP();
            NOP();

            MD_TxControl();   /* UART transmission control */
      }

      /* End user code. Do not edit comment generated here */
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

```
• CG_serial.c

  Note that the following functions are not used in this sample code.
  • UARTA1_Stop
  → Stops the operation of UARTA1.
  • MD_STATUS UARTA1_ReceiveData
  → Saves received data in the buffer.
  • MD_STATUS UARTA1_SendData
  → Transmits data.
  • UARTA1_SoftOverRunCallback
  → Processing performed when the amount of data received exceeds the size of the
  receive buffer.

/*
***********************************************************************************
* Copyright(C) 2008, 2011 Renesas Electronics Corporation
* RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY
* This program must be used solely for the purpose for which
* it was furnished by Renesas Electronics Corporation. No part of this
* program may be reproduced or disclosed to others, in any
* form, without the prior written permission of Renesas Electronics
* Corporation.
*
* This device driver was created by CodeGenerator for V850ES/Jx3
* 32-Bit Single-Chip Microcontrollers
* Filename: CG_serial.c
* Abstract: This file implements device driver for Serial module.
* APIlib:   CodeGenerator for V850ES/Jx3 V1.00.01 [07 Jun 2011]
* Device:   uPD70F3796
* Compiler: CA850
* Creation date:  2011/07/20
***********************************************************************************
*/

/*
***********************************************************************************
** Pragma directive
***********************************************************************************
*/
/* Start user code for pragma. Do not edit comment generated here */
```

```
#pragma interrupt INTUA1R MD_INTUA1R
#pragma interrupt INTUA1T MD_INTUA1T
/* End user code. Do not edit comment generated here */
/*
**********************************************************************************
** Include files
**********************************************************************************
*/
#include "CG_macrodriver.h"
#include "CG_serial.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*#include "CG_userdefine.h"*/

/*
**********************************************************************************
**  Global define
**********************************************************************************
*/
volatile UCHAR  *gpUarta1TxAddress; /* uarta1 transmit buffer address */
volatile USHORT gUarta1TxCnt; /* uarta1 transmit data number */
volatile UCHAR  *gpUarta1RxAddress; /* uarta1 receive buffer address */
volatile USHORT gUarta1RxCnt; /* uarta1 receive data number */
volatile USHORT gUarta1RxLen; /* uarta1 receive data length */
/* Start user code for global. Do not edit comment generated here */
/*
**********************************************************************************
** Private global variables and constants
**********************************************************************************
*/
      #define     BUFFSIZE    16    /*  Receive data buffer size */
static UCHAR gRxBuffer[BUFFSIZE];   /* Receive data buffer */
static UCHAR gTxPointer;            /* Buffer pointer for transmission */
      #define     TXDNUM      4     /* Transmit data number */
static UCHAR gRepStatus;            /* Reply status */
      #define     REP_NONE    0x00  /*  All reply finished */
      #define     REP_STORE   0x01  /*  Some no-reply data stored */
      #define     REP_FULL    0x02  /*  Buffer is full of no-reply data */
```

```
/* Transmit code */
static const UCHAR gStringOK[TXDNUM] = { "OK\r\n" };
static const UCHAR gStringok[TXDNUM] = { "ok\r\n" };
static const UCHAR gStringUC[TXDNUM] = { "UC\r\n" };
static const UCHAR gStringPE[TXDNUM] = { "PE\r\n" };
static const UCHAR gStringFE[TXDNUM] = { "FE\r\n" };
static const UCHAR gStringOE[TXDNUM] = { "OE\r\n" };

/* Receive code */
#define     CODE_NORMAL1      'T'   /* Normal code 1 : 'T' */
#define     CODE_NORMAL2      't'   /* Normal code 2 : 't' */

/* Error code */
#define     CODE_ERRORP 0xFF  /* Parity error code */
#define     CODE_ERRORF 0xFE  /* Framing error code */
#define     CODE_ERRORO 0xFD  /* Overrun error code */
/* End user code. Do not edit comment generated here */

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function initializes the UARTA1 module.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
void UARTA1_Init(void)
{
      UA1TXE = 0U;      /* disable UARTA1 transmission operation */
      UA1RXE = 0U;      /* disable UARTA1 reception operation */
      UA1PWR = 0U;      /* disable UARTA1 operation */
      UA1TMK = 1U;      /* disable INTUA1T interrupt */
      UA1TIF = 0U;      /* clear INTUA1T interrupt flag */
```

```
      UA1RMK = 1U;      /* disable INTUA1R interrupt */
      UA1RIF = 0U;      /* clear INTUA1R interrupt flag */
      /* Set INTUA1T level low priority */
      UA1TIC |= 0x07U;
      /* Set INTUA1R level low priority */
      UA1RIC |= 0x07U;
      /* Baud rate : 38400bps */
      UA1CTL1 = _02_UARTA_BASECLK_FXX_4;
      UA1CTL2 = _41_UARTA1_BASECLK_DIVISION;
      /* LSB first, Even parity, Data length 8bit, Stop bit 1bit */
      UA1CTL0 = _10_UARTA_TRANSFDIR_LSB | _0C_UARTA_PARITY_EVEN |
_02_UARTA_DATALENGTH_8BIT | _00_UARTA_STOPLENGTH_1BIT;
      /* Normal output, Normal input */
      UA1OPT0 = _14_UARTA_UAnOPT0_INITIALVALUE | _00_UARTA_TRAN_DATALEVEL_NORMAL |
_00_UARTA_REC_DATALEVEL_NORMAL;
      UA1PWR = 1U;      /* enable UARTA1 operation */
      /* Set TXDA1 pin */
      PFC9L &= 0xFEU;
      PFCE9L |= 0x01U;
      PMC9L |= 0x01U;
      /* Set RXDA1 pin */
      PFC9L &= 0xFDU;
      PFCE9L |= 0x02U;
      PMC9L |= 0x02U;
}
/*
**-----------------------------------------------------------------------------
**
**  Abstract:
**    This function starts the UARTA1 operation.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-----------------------------------------------------------------------------
*/
void UARTA1_Start(void)
```

```
{
      UA1TIF = 0U;      /* clear INTUA1T interrupt flag */
      UA1TMK = 0U;      /* enable INTUA1T interrupt */
      UA1RIF = 0U;      /* clear INTUA1R interrupt flag */
      UA1RMK = 0U;      /* enable INTUA1R interrupt */
      UA1TXE = 1U;      /* enable UARTA1 transmission operation */
      UA1RXE = 1U;      /* enable UARTA1 reception operation */
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function stops the UARTA1 operation.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
/*void UARTA1_Stop(void)
{
      UA1TXE = 0U;*/    /* disable UARTA1 transmission operation */
/*    UA1RXE = 0U;*/    /* disable UARTA1 reception operation */
/*    UA1TMK = 1U;*/    /* disable INTUA1T interrupt */
/*    UA1TIF = 0U;*/    /* clear INTUA1T interrupt flag */
/*    UA1RMK = 1U;*/    /* disable INTUA1R interrupt */
/*    UA1RIF = 0U;*/    /* clear INTUA1R interrupt flag */
/*}*/
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function receives UARTA1 data.
**
**  Parameters:
**    rxbuf: receive buffer pointer
```

```
**    rxnum: buffer size
**
**  Returns:
**    MD_OK
**    MD_ARGERROR
**
**-------------------------------------------------------------------------
*/
/*MD_STATUS UARTA1_ReceiveData(UCHAR *rxbuf, USHORT rxnum)
{
      MD_STATUS status = MD_OK;

      if (rxnum < 1U)
      {
            status = MD_ARGERROR;
      }
      else
      {
            gUarta1RxCnt = 0U;
            gUarta1RxLen = rxnum;
            gpUarta1RxAddress = rxbuf;
      }

      return (status);
}*/
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function sends UARTA1 data.
**
**  Parameters:
**    txbuf: transfer buffer pointer
**    txnum: buffer size
**
**  Returns:
**    MD_OK
**    MD_ARGERROR
**    MD_DATAEXISTS
```

```
**
**-------------------------------------------------------------------------
*/
/*MD_STATUS UARTA1_SendData(UCHAR *txbuf, USHORT txnum)
{
      MD_STATUS status = MD_OK;

      if (txnum < 1U)
      {
            status = MD_ARGERROR;
      }
      else
      {
            gpUarta1TxAddress = txbuf;
            gUarta1TxCnt = txnum;
            if((UA1STR & 0x80U) == 0U)
            {
                  UA1TMK = 1U;*/    /* disable INTUA1T interrupt */
/*                UA1TX = *gpUarta1TxAddress;
                  gpUarta1TxAddress++;
                  gUarta1TxCnt--;
                  UA1TMK = 0U;*/    /* enable INTUA1T interrupt */
/*          }
            else
            {
                  status = MD_DATAEXISTS;
            }
      }

      return (status);
}*/

/* Start user code for adding. Do not edit comment generated here */
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function is INTUA1R interrupt service routine.
**
```

```
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
__interrupt void MD_INTUA1R(void)
{
      UCHAR rx_data;
      UCHAR err_type;

      /* Receive data buffer is mask */
      if( gRepStatus == REP_FULL )
      {
            NOP();      /* Don't read/store receive data */
      }
      else
      {
            rx_data = UA1RX;              /* Read receive data */
            err_type = (UCHAR)(UA1STR & 0x07U); /* Read reception error */
            UA1STR = (UCHAR)~0x07U;       /* Clear reception error */

            /* Error is detected */
            if (err_type != 0U)
            {
                  UARTA1_ErrorCallback(err_type);     /* Error function */
            }
            /* Reception ends normally */
            else
            {
                  /* Receive data buffer is not full */
                  if (gUarta1RxLen > gUarta1RxCnt)
                  {
                        *gpUarta1RxAddress = rx_data; /* Store receive data */
                        gpUarta1RxAddress++;           /* Increment receive
buffer address */
                        gUarta1RxCnt++;          /* Count receive data number */
```

```
                        /* Receive data buffer is end */
                        if (gUarta1RxLen == gUarta1RxCnt)
                        {
                              UARTA1_ReceiveEndCallback();  /* Reset receive data
number */
                        }
                  }
                  /* Receive data number is over */
                  else
                  {
                        /*UARTA1_SoftOverRunCallback(rx_data);*/  /* Overrun
function */
                  }
            }

            /* Receive data buffer is full of no-reply data */
            if( gTxPointer == gUarta1RxCnt )
            {
                  gRepStatus = REP_FULL;  /* Set reply status */
            }
            /* Some data stored */
            else
            {
                  gRepStatus = REP_STORE; /* Set reply status */
            }
      }
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function is INTUA1T interrupt service routine.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
```

```
*/
__interrupt void MD_INTUA1T(void)
{
      /* During transmission */
      if (gUarta1TxCnt > 0U)
      {
            UA1TX = *gpUarta1TxAddress;    /* Output transmit data */
            gpUarta1TxAddress++;           /* Increment transmit data address */
            gUarta1TxCnt--;                /* Count data transfer */
      }
      /* Last data transfer */
      else
      {
            UARTA1_SendEndCallback();      /* Last data transfer function */
      }
}
/*
**-----------------------------------------------------------------------------
**
**  Abstract:
**    This function is a callback function of UARTA1.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-----------------------------------------------------------------------------
*/
void UARTA1_ReceiveEndCallback(void)
{
      /* Start user code. Do not edit comment generated here */

      gUarta1RxCnt = 0;                          /* Clear receive data number */
      gpUarta1RxAddress = (UCHAR *)gRxBuffer;    /* Reset receive data buffer
address */

      /* End user code. Do not edit comment generated here */
}
```

```
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function is a callback function of UARTA1.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
void UARTA1_SendEndCallback(void)
{
      /* Start user code. Do not edit comment generated here */

      /* Pointer update */
      gTxPointer++;
      gTxPointer %= BUFFSIZE;

      /* All reply finished */
      if( gTxPointer == gUarta1RxCnt )
      {
            gRepStatus = REP_NONE;  /* Set reply status */
      }

      gUarta1TxCnt = (TXDNUM - 1);  /* Clear transmit number */
      UA1TX = *gpUarta1TxAddress;   /* Output transmit data */

      /* End user code. Do not edit comment generated here */
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function is a callback function of UARTA1.
**
```

```
**  Parameters:
**    err_type: error type
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
void UARTA1_ErrorCallback(UCHAR err_type)
{
      /* Start user code. Do not edit comment generated here */

      /* Parity error */
      if( err_type & 0b00000100 )
      {
            /* Store parity error code for buffer */
            MD_StoreError( CODE_ERRORP );
      }

      /* Framing error */
      if( err_type & 0b00000010 )
      {
            /* Store framing error code for buffer */
            MD_StoreError( CODE_ERRORF );
      }

      /* Overrun error */
      if( err_type & 0b00000001 )
      {
            /* Store overrun error code for buffer */
            MD_StoreError( CODE_ERRORO );
      }

      /* End user code. Do not edit comment generated here */
}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
```

```
**    Store error code for buffer.
**
**  Parameters:
**    Error code
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
void MD_StoreError(UCHAR code)
{

      *gpUarta1RxAddress = code;    /* Set error code */
      gpUarta1RxAddress++;          /* Increment receive buffer address */
      gUarta1RxCnt++;               /* Count receive data number */

      /* Receive data buffer is full */
      if (gUarta1RxLen <= gUarta1RxCnt)
      {
            gUarta1RxCnt = 0;       /* Clear receive data number */

            /* Reset receive data buffer address */
            gpUarta1RxAddress = (UCHAR *)gRxBuffer;
      }

}
/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    Variable initialization for UART.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
```

```
**-------------------------------------------------------------------------
*/
void MD_UartVarInit( void )
{

      gUarta1TxCnt = (TXDNUM - 1);              /* Clear transmit number */
      gpUarta1RxAddress = (UCHAR *)gRxBuffer;   /* Set receive data buffer address
*/
      gUarta1RxLen = BUFFSIZE;      /* Set maximum value of receive data number */

      gUarta1RxCnt = 0;       /* Clear receive data number */
      gTxPointer = 0;         /* Clear buffer pointer for transmission */
      gRepStatus = REP_NONE;  /* Reply status : All reply finished */

}

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    UART transmission control.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
void MD_TxControl( void )
{

      /* All reply finished */
      if( gRepStatus == REP_NONE )
      {
            UA1TMK = 1; /* Disable INTUA1T */
      }
      /* No-reply data stored */
      else
```

```
{
        /* First byte of transmit data */
        if( gUarta1TxCnt >= (TXDNUM - 1) )
        {
                /* Set transmit data address */
                switch( gRxBuffer[ gTxPointer ] )
                {
                        /* Normal code 1 receive */
                        case CODE_NORMAL1:
                                /* Set transmit data 'OK' */
                                gpUarta1TxAddress = (UCHAR *)gStringOK;
                                break;
                        /* Normal code 2 receive */
                        case CODE_NORMAL2:
                                /* Set transmit data 'ok' */
                                gpUarta1TxAddress = (UCHAR *)gStringok;
                                break;
                        /* Parity error */
                        case CODE_ERRORP:
                                /* Set transmit data 'PE' */
                                gpUarta1TxAddress = (UCHAR *)gStringPE;
                                break;
                        /* Framing error */
                        case CODE_ERRORF:
                                /* Set transmit data 'FE' */
                                gpUarta1TxAddress = (UCHAR *)gStringFE;
                                break;
                        /* Overrun error */
                        case CODE_ERRORO:
                                /* Set transmit data 'OE' */
                                gpUarta1TxAddress = (UCHAR *)gStringOE;
                                break;
                        /* Other code receive */
                        default:
                                /* Set transmit data 'UC' */
                                gpUarta1TxAddress = (UCHAR *)gStringUC;
                                break;
                }
```

```
                    /* Transmission stopped */
                    if( !UA1TSF )
                    {
                          UA1TIF = 0;
                          UA1TMK = 0; /* Enable INTUA1T */

                          gUarta1TxCnt--;          /* Count transmit number */
                          UA1TX = *gpUarta1TxAddress;    /* Data output */
                          gpUarta1TxAddress++;     /* Increment transmit data address
*/
                    }
              }
      }

}
/* End user code. Do not edit comment generated here */
```

- CG_systeminit.c

```
/*
***********************************************************************************
* Copyright(C) 2008, 2011 Renesas Electronics Corporation
* RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY
* This program must be used solely for the purpose for which
* it was furnished by Renesas Electronics Corporation. No part of this
* program may be reproduced or disclosed to others, in any
* form, without the prior written permission of Renesas Electronics
* Corporation.
*
* This device driver was created by CodeGenerator for V850ES/Jx3
* 32-Bit Single-Chip Microcontrollers
* Filename: CG_systeminit.c
* Abstract: This file implements system initializing function.
* APIlib:   CodeGenerator for V850ES/Jx3 V1.00.01 [07 Jun 2011]
* Device:   uPD70F3796
* Compiler: CA850
* Creation date:  2011/07/20
***********************************************************************************
*/

/*
***********************************************************************************
** Pragma directive
***********************************************************************************
*/
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "CG_macrodriver.h"
#include "CG_system.h"
#include "CG_port.h"
#include "CG_serial.h"
```

```
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*#include "CG_userdefine.h"*/

/*
***************************************************************************
**  Global define
***************************************************************************
*/
/* Start user code for global. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

extern unsigned long _S_romp; /* ROMization symbol */

void systeminit(void);

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function initializes each macro.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
void systeminit(void)
{
      _rcopy(&_S_romp, -1);   /* Call ROMization */
      DI(); /* disable interrupt */
      PORT_Init();
      UARTA1_Init();
      EI(); /* enable interrupt */
}
```

```
/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

- CG_system.c

```
/*
***********************************************************************************
* Copyright(C) 2008, 2011 Renesas Electronics Corporation
* RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY
* This program must be used solely for the purpose for which
* it was furnished by Renesas Electronics Corporation. No part of this
* program may be reproduced or disclosed to others, in any
* form, without the prior written permission of Renesas Electronics
* Corporation.
*
* This device driver was created by CodeGenerator for V850ES/Jx3
* 32-Bit Single-Chip Microcontrollers
* Filename: CG_system.c
* Abstract: This file implements device driver for System module.
* APIlib:   CodeGenerator for V850ES/Jx3 V1.00.01 [07 Jun 2011]
* Device:   uPD70F3796
* Compiler: CA850
* Creation date:  2011/07/20
***********************************************************************************
*/

/*
***********************************************************************************
** Pragma directive
***********************************************************************************
*/
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "CG_macrodriver.h"
#include "CG_system.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

```
/*#include "CG_userdefine.h"*/

/*
******************************************************************************
**  Global define
******************************************************************************
*/
/* Start user code for global. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/*
**-----------------------------------------------------------------------------
**
**  Abstract:
**    This function initializes the clock generator module.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-----------------------------------------------------------------------------
*/
void CLOCK_Init(void)
{
      UCHAR psval = 0U;
      UINT i = 0U;
      /* Set WDT2 (stop) */
      WDTM2 = _00_WDT2_OPERMODE_STOP;
      /* Set fXX and fCPU */
      psval = _00_CG_SUBCLK_FEEDBACK_USE | _00_CG_MAINCLK_ENABLE |
_00_CG_MAINCLK_FEEDBACK_USE | _00_CG_CPUCLK_MAIN0;
      PRCMD = psval;
      PCC = psval;      /* fCPU = fXX */
      /* Select PLL Mode */
      SELPLL = 1U;
      /* Set fR (disable) */
      RSTOP = 1U;
      /* Set fBRG (disable) */
```

```
      /*BGCE0 = 0U;*/
      /* Set Stand-by function */
      /*psval = _00_CG_STANDBY_INTWDT2EN | _00_CG_STANDBY_NMIEN |
_00_CG_STANDBY_MASKIEN;
      PRCMD = psval;
      PSC = psval;*/
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

- CG_port.c

```
/*
***********************************************************************************
* Copyright(C) 2008, 2011 Renesas Electronics Corporation
* RENESAS ELECTRONICS CONFIDENTIAL AND PROPRIETARY
* This program must be used solely for the purpose for which
* it was furnished by Renesas Electronics Corporation. No part of this
* program may be reproduced or disclosed to others, in any
* form, without the prior written permission of Renesas Electronics
* Corporation.
*
* This device driver was created by CodeGenerator for V850ES/Jx3
* 32-Bit Single-Chip Microcontrollers
* Filename: CG_port.c
* Abstract: This file implements device driver for PORT module.
* APIlib:   CodeGenerator for V850ES/Jx3 V1.00.01 [07 Jun 2011]
* Device:   uPD70F3796
* Compiler: CA850
* Creation date:  2011/07/26
***********************************************************************************
*/

/*
***********************************************************************************
** Pragma directive
***********************************************************************************
*/
/* Start user code for pragma. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "CG_macrodriver.h"
#include "CG_port.h"
/* Start user code for include. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

```
/*#include "CG_userdefine.h"*/

/*
***************************************************************************
**  Global define
***************************************************************************
*/
/* Start user code for global. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */

/*
**-------------------------------------------------------------------------
**
**  Abstract:
**    This function initializes setting for Port I/O.
**
**  Parameters:
**    None
**
**  Returns:
**    None
**
**-------------------------------------------------------------------------
*/
void PORT_Init(void)
{
      /* P90 : TXDA1 */
      /* P91 : RXDA1 */
      /* P40-P42,PCM0,PDL5 : On-chip debug */
      /* Other port : Low level output(no use) */

      /* PM0  : 0b10000011 */
      PM0 = _00_PMn2_MODE_OUTPUT | _00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT |
_00_PMn5_MODE_OUTPUT | _00_PMn6_MODE_OUTPUT | _83_PM0_DEFAULT;
      /* PM1  : 0b11111100 */
      PM1 = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _FC_PM1_DEFAULT;
      /* PM3L : 0b00111000 */
      PM3L = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT |
_00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT | _38_PM3L_DEFAULT;
      /* PM3H : 0b11111100 */
```

```
    PM3H = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _FC_PM3H_DEFAULT;

    /* PM4  : No care */

    /* PM5  : 0b11000000 */

    PM5 = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT |
_00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT | _00_PMn5_MODE_OUTPUT |
_C0_PM5_DEFAULT;

    /* PM7L : 0b00000000 */

    PM7L = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT |
_00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT | _00_PMn5_MODE_OUTPUT |
_00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT;

    /* PM7H : 0b11110000 */

    PM7H = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT |
_00_PMn3_MODE_OUTPUT | _F0_PM7H_DEFAULT;

    /* PM9L : 0b00000011 */

    PM9L = _01_PMn0_MODE_UNUSED | _02_PMn1_MODE_UNUSED | _00_PMn2_MODE_OUTPUT |
_00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT | _00_PMn5_MODE_OUTPUT |
_00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT;

    /* PM9H : 0b00000000 */

    PM9H = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT |
_00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT | _00_PMn5_MODE_OUTPUT |
_00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT;

    /* PMCM : 0b11110000 */

    PMCM = _00_PMn0_MODE_OCD | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT |
_00_PMn3_MODE_OUTPUT | _F0_PMCM_DEFAULT;

    /* PMCT : 0b10101100 */

    PMCT = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT |
_00_PMn6_MODE_OUTPUT | _AC_PMCT_DEFAULT;

    /* PMDH : 0b11100000 */

    PMDH = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT |
_00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT | _E0_PMDH_DEFAULT;

    /* PMDLL: 0b00000000 */

    PMDLL = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT |
_00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT | _20_PMn5_MODE_UNUSED |
_00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT;

    /* PMDLH: 0b00000000 */

    PMDLH = _00_PMn0_MODE_OUTPUT | _00_PMn1_MODE_OUTPUT | _00_PMn2_MODE_OUTPUT |
_00_PMn3_MODE_OUTPUT | _00_PMn4_MODE_OUTPUT | _00_PMn5_MODE_OUTPUT |
_00_PMn6_MODE_OUTPUT | _00_PMn7_MODE_OUTPUT;


    /* PMC0  : 0b00000000 */

    PMC0 = _00_PMCn2_OPER_PORT | _00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT |
_00_PMCn5_OPER_PORT | _00_PMCn6_OPER_PORT;

    /* PMC3L : 0b00000000 */

    PMC3L = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn6_OPER_PORT | _00_PMCn7_OPER_PORT;
```

```
    /* PMC3H : 0b00000000 */
    PMC3H = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT;
    /* PMC4  : No care */
    /* PMC5  : 0b00000000 */
    PMC5 = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT | _00_PMCn5_OPER_PORT;
    /* PMC9L : 0b00000000 */
    PMC9L = _00_PMCn2_OPER_PORT | _00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT |
_00_PMCn5_OPER_PORT | _00_PMCn6_OPER_PORT | _00_PMCn7_OPER_PORT;
    /* PMC9H : 0b00000000 */
    PMC9H = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT | _00_PMCn5_OPER_PORT |
_00_PMCn6_OPER_PORT | _00_PMCn7_OPER_PORT;
    /* PMCCM : 0b00000000 */
    PMCCM = _00_PMCn0_OPER_OCD | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT;
    /* PMCCT : 0b00000000 */
    PMCCT = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn4_OPER_PORT |
_00_PMCn6_OPER_PORT;
    /* PMCDH : 0b00000000 */
    PMCDH = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT;
    /* PMCDLL: 0b00100000 */
    PMCDLL = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT | _20_PMCn5_OPER_ALTER |
_00_PMCn6_OPER_PORT | _00_PMCn7_OPER_PORT;
    /* PMCDLH: 0b00000000 */
    PMCDLH = _00_PMCn0_OPER_PORT | _00_PMCn1_OPER_PORT | _00_PMCn2_OPER_PORT |
_00_PMCn3_OPER_PORT | _00_PMCn4_OPER_PORT | _00_PMCn5_OPER_PORT |
_00_PMCn6_OPER_PORT | _00_PMCn7_OPER_PORT;
}

/* Start user code for adding. Do not edit comment generated here */
/* End user code. Do not edit comment generated here */
```

7. REFERENCE DOCUMENTS

V850ES/JC3-L, V850ES/JE3-L User's Manual: Hardware (R01UH0018)

V850ES/JF3-L User's Manual: Hardware (R01UH0017)

V850ES/JG3-L User's Manual: Hardware (R01UH0165)

V850ES/JG3-L (on-chip USB controller) User's Manual Hardware (R01UH0001)

(Obtain the latest versions from the Renesas Electronics website.)

Technical updates and technical news

(Obtain the latest information from the Renesas Electronics website.)

Website and Support

Renesas Electronics Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/inquiry

Revision Record	V850ES/Jx3-L - UART Communication Using UARTA

Rev.	Date	Description	
		Page	Summary
1.00	Dec. 7, 2011	–	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord 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 unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   - The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different part number, confirm that the change will not lead to problems.
   - The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   - "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.1