RH850/U2B Group

R01AN7076EJ0100
Rev.1.00

FlexRay Application Note

Summary

This document describes the procedure for performing the FlexRay communication using RH850/U2Bx.

[Note] About the procedure example of each item, there is the part that describes the reference example of the specific application. Since the RENESAS program is supplied in the function unit, the procedure example and the program configuration may differ.

Summary

1. Initial Setting

In RH850/U2Bx, perform the following setting for usings the FlexRay communication function.RH850/U2Bx.

- Port Setting
- Interrupt Setting
- Initialization of FlexRay Module
- FlexRay Clock Setting
- Setting for FlexRay, etc.
- MessageRAM Setting

The following shows each initialization.

1.1 Port Setting

In this operation example, allocate the FlexRay to the port 12 and 22. **Table 1-1** shows the FlexRay pin setting, and Table 1-2 shows the function of each pin.

Table 1-1 Pin Setting of FlexRay Module

Port Name	Pin Name	Setting Value of Port Control Register (PCRn_m)
P12_1	FLX0TXDA	PCR12_1 = 0x00000045
P12_0	FLX0TXENA	PCR12_0 = 0x00000045
P12_7	FLX0RXDA	PCR12_7 = 0x00000057
P12_8	FLX0RXDB	PCR12_8 = 0x00000055
P22_11	FLX0TXDB	PCR22_11 = 0x00000044
P12_9	FLX0TXENB	PCR12_9 = 0x00000045

Table 1-2 Pin Explanation of FlexRay Module

Pin Name	Function
FLX0TXDA	Channel A transmit data output pin
FLX0RXDA	Channel A receive data input pin
FLX0TXENA	Channel A transmit data enable pin “H” : Transmit disable “L” : Transmit enable
FLX0TXDB	Channel B transmit data output pin
FLX0RXDB	Channel B receive data input pin
FLX0TXENB	Channel B transmit data enable pin “H” : Transmit disable “L” : Transmit enable

1.2 Initialization of FlexRay Module

Immediately after the power resetting and the CLEAR_RAMS command executing by FLXAnFRSUCC1 register, the internal RAM initialization of the FlexRay module (all “0”) is executed.

While the internal RAM is being initialized, confirm the internal RAM initialization process is completed since the settings cannot be made to the FlexRay registers.

1.3 FlexRay Clock

The sampling clock supplied to FlexRay is the high-speed peripheral clock “CLK_HSB (80MHz)”.

1.3.1 Setting of FlexRay Sampling Clock and FlexRay Communication Time Unit

This section explains each clock setting using in the FlexRay module. These clocks generate within the FlexRay module by dividing down the high-speed peripheral clock (CLK_HSB).

Before performing the communication in FlexRay, severally set the clock that is sampling the FlexRay bus value, Macrotick that is common time unit in FlexRay network (hereinafter MT), and Microtick that is local time unit in the node (hereinafter uT).

Table 1-3 shows the relationship between high-speed peripheral clock (CLK_HSB) and FlexRay communication speed.

Table 1-3 Relationship between High-speed Peripheral Clock and FlexRay

High-speed Peripheral Clock (CLK_HSB)	FLXAnFRPRTC1 Register BRP[1:0] Setting Value	Bit Clock (Bit/s)	Communication Speed
80MHz	00	100ns	10Mbps
80MHz	01	200ns	5Mbps
80MHz	1x	400ns	2.5Mbps

■Setting Example

Table 1-4 shows the setting value example of each time unit when the high-speed peripheral clock (CLK_HSB) is 80Mhz and the communication bit rate is 10Mbps. Figure 1-1 shows the setting procedure example.

Table 1-4 Setting Value Example of Each Time Unit

Name	Setting Value
Sampling Period	12.5ns
Communication Speed	10Mbps
Microtick	25ns
Macrotick	1us
Communication Cycle	1ms

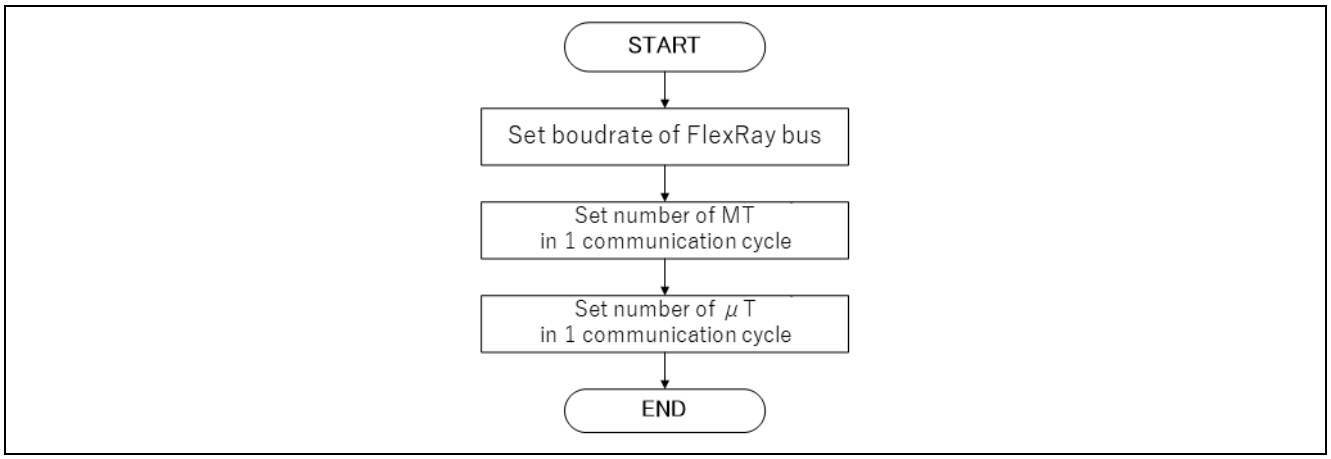


Figure 1-1 Setting Procedure Example of Each Time Unit

[Explanation of Setting Procedure Example]

If BRP bit = “00” in FLXAnFRPRTC1 register is set when high-speed peripheral clock (CLK_HSB) is 80MHz, the sampling period is 80MHz (12.5ns, no division).

Communication bit rate is 1/8 of sampling period (fixed by FlexRay specification), therefore 80MHz/8 ⇒ 10Mbps.

When BRP bit in FLXAnFRPRTC1 register = “00”, 1uT length is 12.5ns*2 = 25ns since uT is two peripheral (fixed) of the high-speed peripheral clock (CLK_HSB).

Communication cycle length is uT length (25ns) *400,000 ⇒ 1ms.

1MT ⇒ 40*uT ⇒ 1us since the Macrotick period is one communication cycle 1,000MT = 40,000uT.

[Note] 1. No register for setting number of uT per MT (pMicroPerMacroNom). The communication controller automatically calculates pMicroPerMacroNom by setting the number of MT per communication cycle and number of uT per communication cycle.

2. Perform the setting in CONFIG state (POCS bit = “001111” in FLXAnFRCCSV register). In any other state, writing to the FLXAnFRPRTC1, FLXAnFRGTUC1, and FLXAnFRGTUC2 registers is not enabled.

1.4 Setting for FlexRay Communication Parameter, etc.

1.4.1 Communication Cycle Setting

In FlexRay module, determine the communication cycle configuration by setting the static segment, dynamic segment, and NIT start position. Figure 1-2 shows the configuration of the communication cycle.

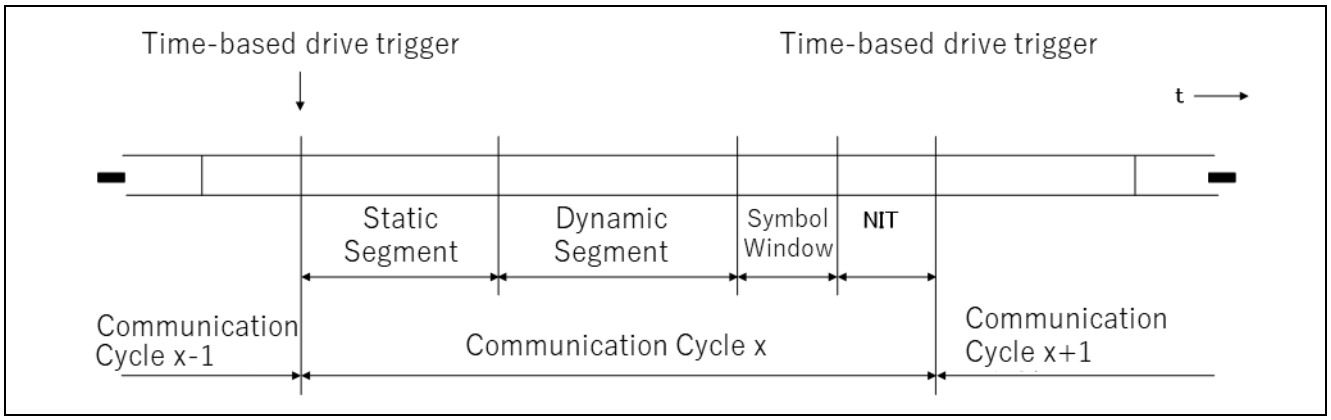


Figure 1-2 Communication Cycle Configuration

The following shows the setting procedure example of the communication cycle. Table 1-5 shows the setting value example of communication cycle. Figure 1-3 shows the setting procedure example.

Table 1-5 Setting Value Example of Communication Cycle

Name	Setting Value
1 communication cycle length	1000 MT
Static slot length	54MT
Number of static slot	6 slot
ActionPoint position	5MT
Mini slot length	24MT
Number of mini slot	6slot
Mini slot ActionPoint position	2MT
NIT start position	990MT

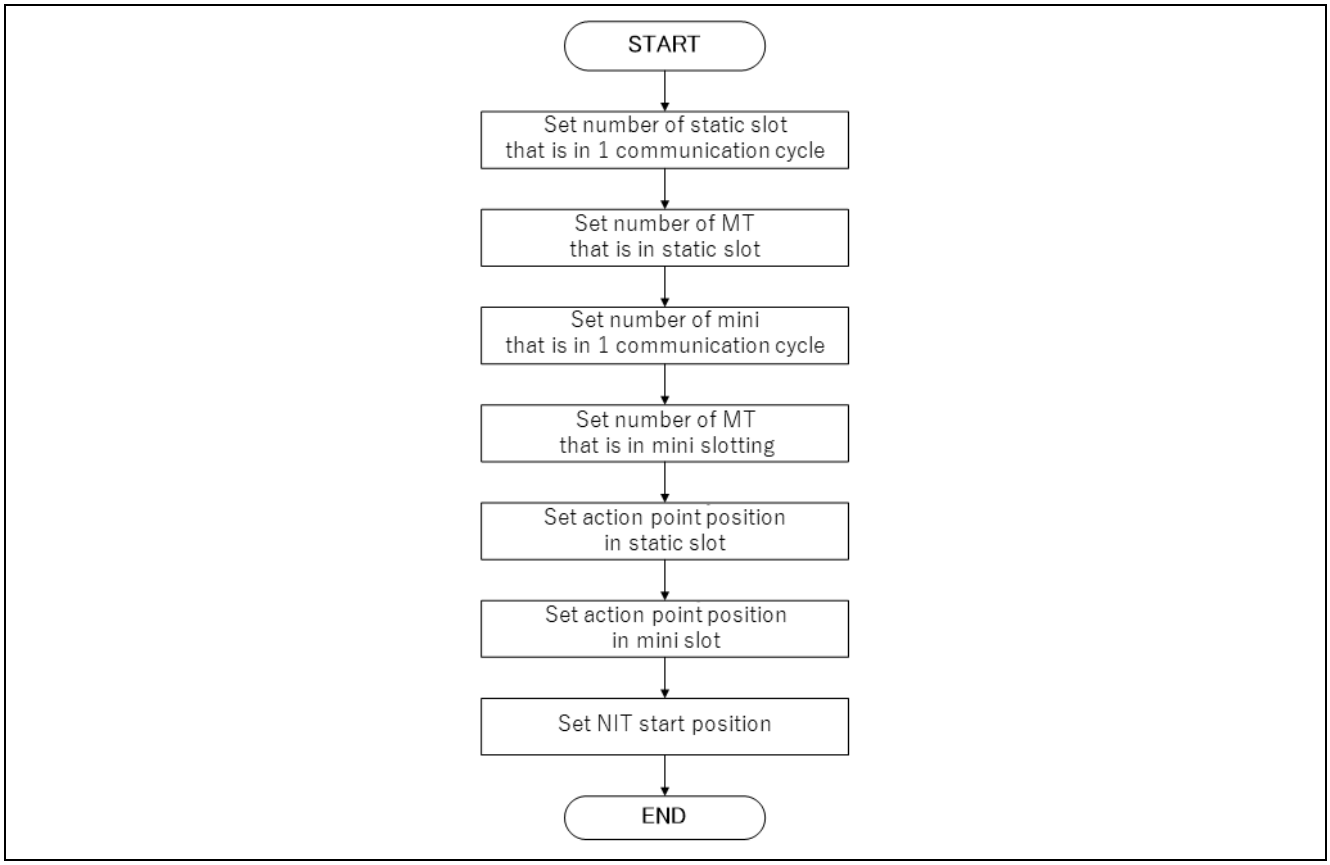


Figure 1-3 Communication Cycle Setting Procedure Example

[Note] 1. No register for setting the start position/length of symbol window. Dynamic segment end to NIT start position is symbol window.

2. Perform the communication cycle setting in CONFIG state (POCS bit = “001111” in FLXAnFRCCSV regsiter). In any other state, writing to the FLXAnFRGTUC2, FLXAnFRGTUC4, and FLXAnFRGTUC7 to 9 registers is not enabled.

1.4.2 Role of Node (Setting of Startup Node)

In FlexRay, the role of the node is required to determine when initial setting for “possibility of startup frame transmission”.

The startup frames must be at the same time.

Figure 1-4 shows the state transfer course when setting each flag in FLXAnFRSUCC1.

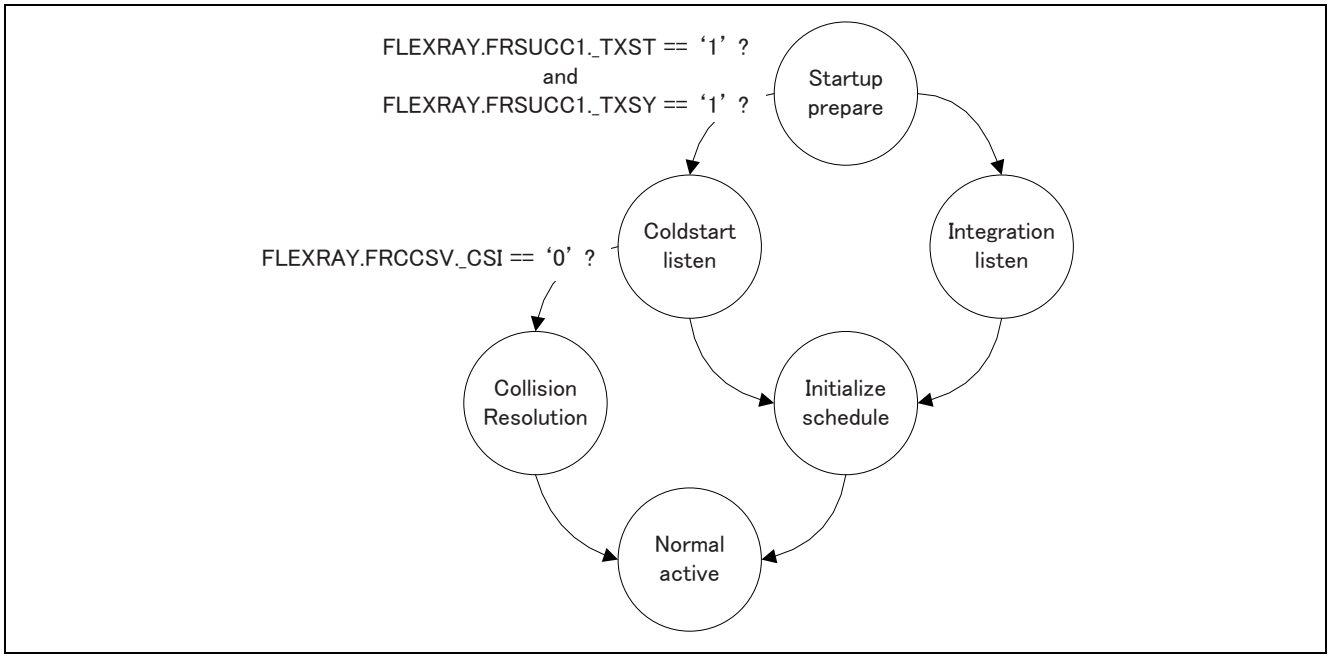


Figure 1-4 Role Setting of Node

The following shows the setting procedure and reference program for configuring as the startup node.

Table 1-6 Setting Example Configuring Startup Node

Name	Setting Value
Startup frame transmission	Yes
Sync frame transmission	Yes
Coldstart Inhibit flag	Yes
Startup retry times	31 times

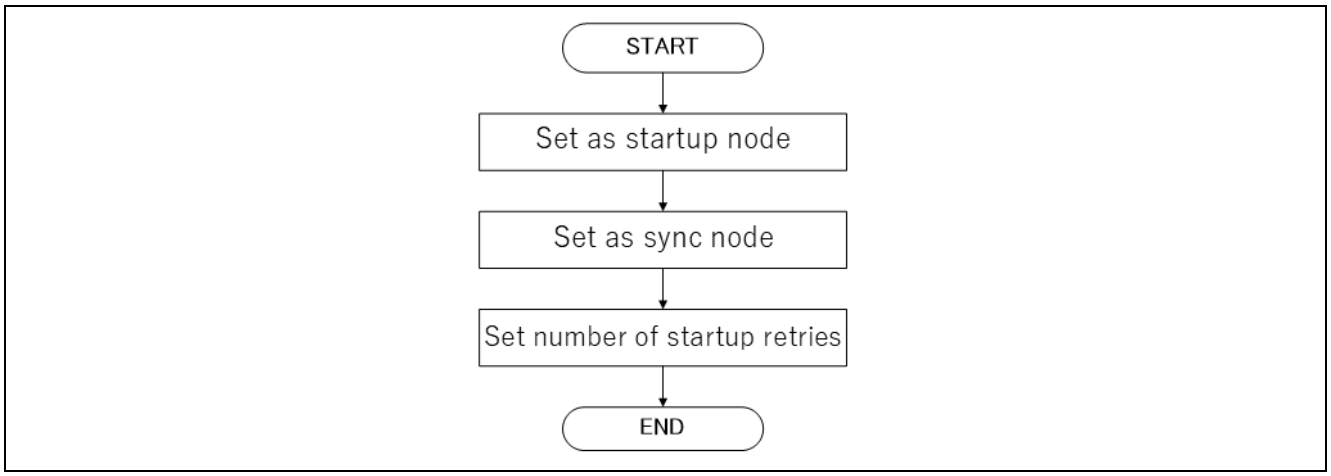


Figure 1-5 Control Procedure Example

[Explanation of Setting Procedure Example]

Enable the operation as Startup node and Sync node by TXST bit and TXSY bit in FLXAnFRSUCC1 register.

When setting as Startup node, set “1” to TXST bit in FLXAnFRSUCC1. When setting as Sync node, set “1” to TXSY bit.

Startup/Sync frame is only transmitted from the message buffer 0 and 1.

Set the frame ID (pKeySlotId) using as Startup/Sync frame in the header part of the message buffer 0 and 1.

When setting “0” to SPLM register in FLXAnFRMRC register, the message buffer 0 is only used. When setting “1” to SPLM register in FLXAnFRMRC register, the message buffer 1 is only used. In this case, Startup/Sync frame with different payload data can be transmitted to the channel A and B.

[Note] 1. Perform the setting in CONFIG state (POCS bit = “001111” in FLXAnFRCCSV register). In any other state, writing to the FLXAnFRPRTC1, FLXAnFRGTUC1, and FLXAnFRGTUC2 registers is not enabled.
2. When CSI (Coldstart Inhibit) in FLXAnFRCCSV register is “1”, Cold_start is disabled. This flag automatically be “1” when transferring to READY state.
For enabling Cold_start, “0” clear CSI bit in FLXAnFRCCSV register by writing “1001” (ALLOW_COLDSTART command) to CMD bit in FLXAnFRSUCC1 register.
3. Setting for TXSY bit = 0 in FLXAnFRSUCC1 register and TXST bit = 1 in FLXAnFRSUCC1 register are disabled.
4. Confirm the setting consistency of SPLM bit in FLXAnFRMRC register and CH bit (channel filter) in FLXAnFRWRHS1 register.

1.5 Message RAM Setting

1.5.1 Buffer Configuration of Message RAM

In FlexRay module, 8KB Message RAM is bilt-in, and the number of buffers is arbitrary settable in the range from 0 to 128.

Writing to the specific message buffer, and transmit enable/disable setting to static segment are possible by setting of SEC bit in FLXAnFRMRC register.

Figure 1-6 shows the buffer configuration in MessageRAM.

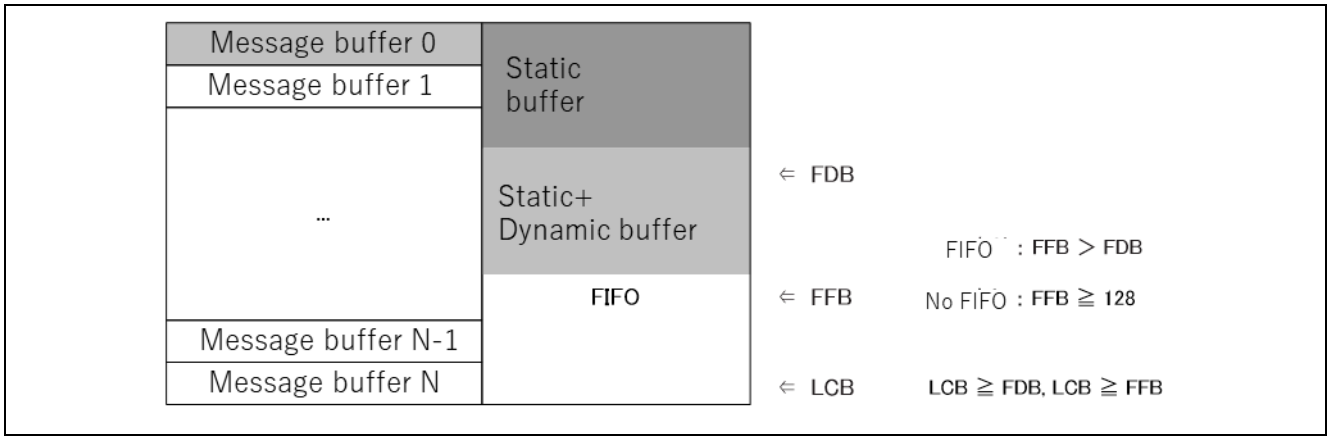


Figure 1-6 Message Buffer Configuration

[Note]

1. Please note when setting FLXAnFRMRC register.
   The size of FIFO buffer is decided by FFB bit and LCB bit.
   When using Dynamic buffer and FIFO buffer, set the setting value of FFB bit that is lager than the setting value of FDB bit.
   When you do not use the Dynamic buffer, set the FDB bit to 128 or a value greater than 128.
   When you do not use a FIFO buffer, set the FFB bit to 128 or a value greater than 128.
   The LCB bit setting value should be larger than the FFB bit setting value when using a FIFO buffer, and the FDB bit setting value when using a Dynamic buffer.
2. The FlexRay module does not check whether the MessageRAM configuration is set incorrectly.
3. Perform MessageRAM setting in CONFIG state (POCS bit = “001111” in FLXAnFRCCSV register). In any other state, writing to the FLXAnFRMHDC and FLXAnFRMRC registers is not enabled.
4. After resetting the hardware, it will be in mDefault_config state (POCS bit in FLXAnFRCCSV register = “000000”).
   Before initial resetting, transfer to CONFIG state by CHI CONFIG command (CMD bit in FLXAnFRSUCC1 register = “0001”).

■Header Part Configuration Example (1) (When only configuring Static buffer to MessageRAM)

Table 1-8 shows the configuration example when only configuring Static buffer to MessageRAM, and Figure 1-7 shows the setting example.

Table 1-7 Configuration Example of Message RAM

RAM (4 bytes)	Buffer Number	Configuration	Contents
0...3	Message buffer 0	↓Static buffer	Header part
4...7	Message buffer 1		
	....		
56...59	Message buffer 7		
60...63	Message buffer 8		←LCB
64			Data part
	....		
2047			

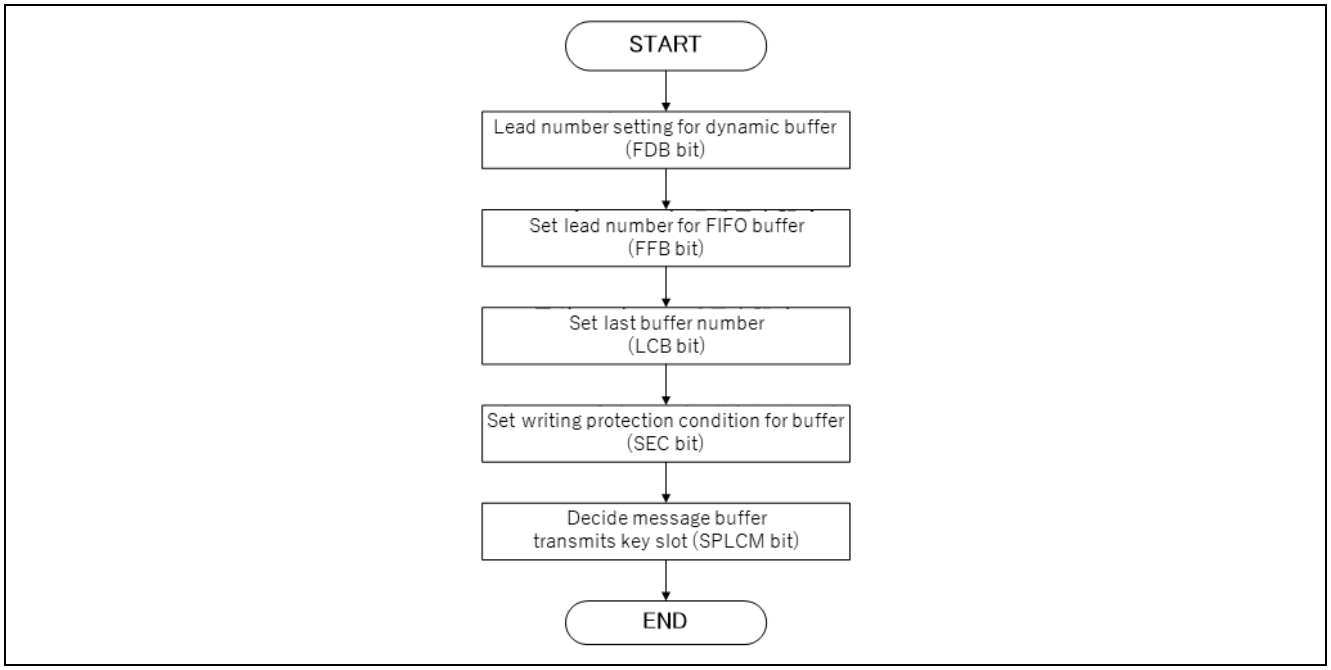


Figure 1-7 Setting Procedure Example

[Explanation for Setting Procedure Example]

In this example, Dyanamic buffer and FIFO buffer is not configured since Static buffer is only configured.

Set FDB bit = “128” in FLXAnFRMRC register and FFB bit = “128” in FLXAnFRMRC register.

Set LCB bit =”7” in FLXAnFRMRC register since the number of message buffers is 8.

■Header Part Configuration Example (2) (When configuring Static buffer and FIFO buffer to MessageRAM)

Table 1-8 shows the configuration example when configuring Static buffer and FIFO buffer to MessageRAM.

Table 1-8 Configuration Example of Message RAM

RAM (4bytes)	Buffer Number	Configuration	Contents
0...3	Message buffer 0	↓Static buffer	Header part
4...7	Message buffer 1		
8...11	Message buffer 2		←FFB
12...15	Message buffer 3		←LCB
16			Data part
	....		
2047			

[Explanation for Setting Procedure Example and Program Example]

Set LCB bit in FLXAnFRMRC register = “3” since the number of message buffers is 4.

In this operation example, configure two Static buffers and two FIFO buffers. Set FDB bit =”128” in FLXAnFRMRC register since Dynamic buffer is not configured.

FFB bit = “2” in FLXAnFRMRC register and LCB bit = “3” since FIFO buffers are two.

Change the macro definition value: MRAMC_SET in the flexray.c file as necessary.

■Header Part Configuration Example (3) (When configuring Static buffer and FIFO buffer)

Table 1-9 shows the configuration example when configuring Static buffer and Dynamic buffer to MessageRAM.

Table 1-9 Configuration Example of Message RAM

RAM (4bytes)	Buffer Number	Configuration	Contents
0...3	Message buffer 0	↓Static buffer	Header part
4...7	Message buffer 1		
8...11	Message buffer 2		
12...15	Message buffer 3		
16...19	Message buffer 4		
20...23	Message buffer 5		
24...27	Message buffer 6	↓Static+Dynamic buffer	
28...31	Message buffer 7		←FDB
32...35	Message buffer 8		
36...39	Message buffer 9		
40...43	Message buffer 10		
44...47	Message buffer 11		←LCB
48			Data part
	....		
2047			

[Explanation for Setting Procedure Example and Program Example]

Set LCB bit = “11” in FLXAnFRMRC register since the number of message buffers is 12.

In this operation example, configure six Static buffers and six FIFO buffers. Set FFB bit =”128” in FLXAnFRMRC register since FIFO buffer is not configured.

Set FDB bit in FLXAnFRMRC register = “6” since the start number of the header part in Dynamic buffer is “6”. Set LCB bit in FLXAnFRMRC register = “11” since the Dynamic buffer is “6”

Change the macro definition value: MRAMC_SET in the flexray.c file as necessary.

1.5.2 Configuration of Data Part and Header Part

In FlexRay, the data length of each buffer is any settable in range from 0 to 254 bytes in 2-byte units.

In Message RAM, the frame information is separated as the header part and the data part.

The header part (the size is fixed at 16 bytes) and the data part (the size is variable) are required for each message buffer.

When setting transmit/receive frame, set the header part in FLXAnFRWRHS1 to FLXAnFRWRHS1 register, and the data part to FLXAnFRWRDS1 to FLXAnFRWRDS64 register.

The set register to each register is transferred to the message buffer in Message RAM via IBF (input buffer). For the transfer method, please refer to “3 Data Setting to MessageRAM & Data Read from MessageRAM”. In this chapter, the configuration is only explained.

Figure 1-8 shows the configuration example of internal Message RAM, and Table 1-10 shows the data length per buffer and the number of buffers that can be secured.

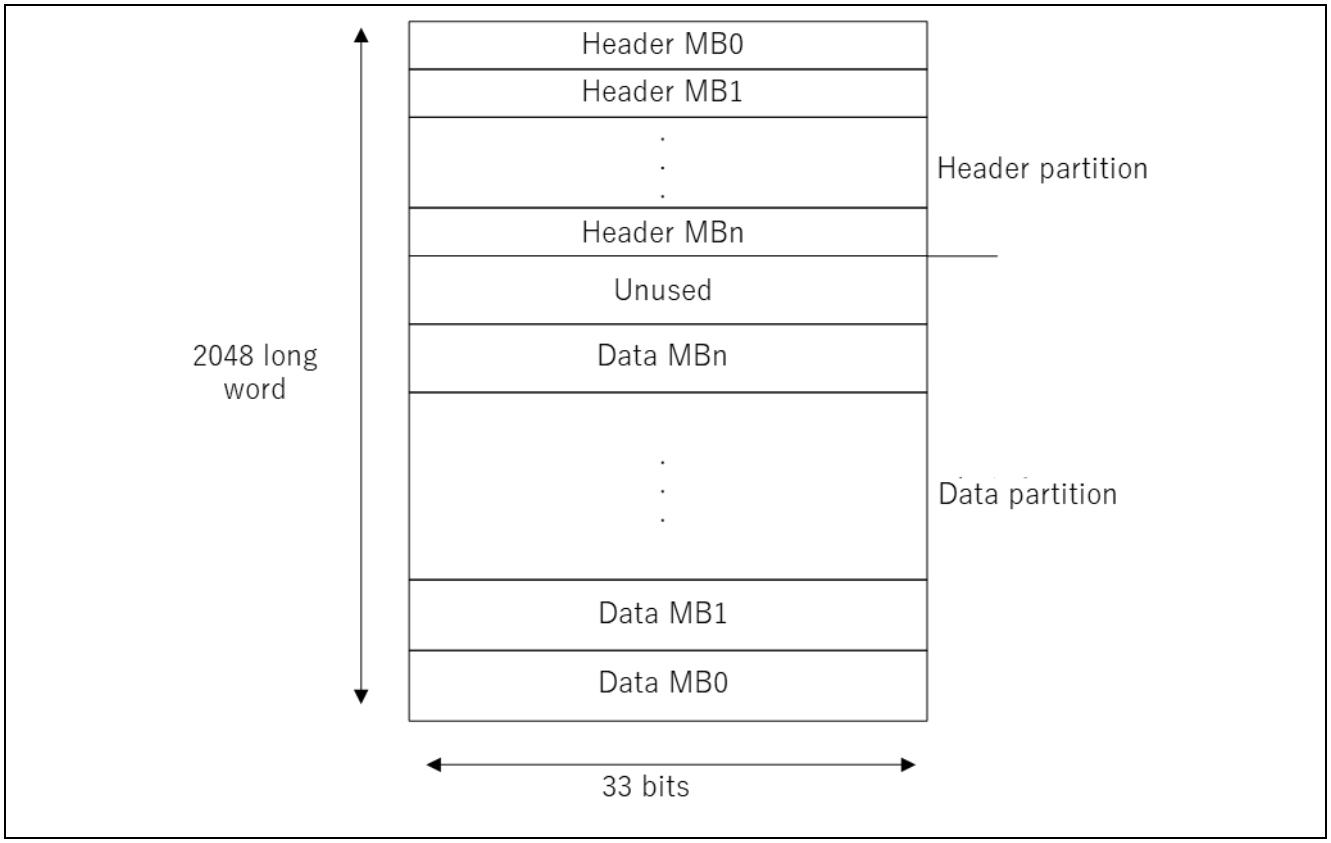


Figure 1-8 Configuration of Message RAM Inside

Table 1-10 Data Length per Buffer & Number of Buffers that Can Be Secured

Data Length per Buffer	Number of Buffers that Can Be Secured	Note
254 bytes	30	
128 bytes	56	
48 bytes	128	

[Note] 1. Set the data length for each buffer so that the total of "header part + data part" of all buffers fits within 8KB. When setting the data length of all buffers is set to be the same, indicates the number of buffers that can be secured.

2. Make sure that the payload length value set in the header section of the FIFO buffer and the length of the data section are the same value. These are settable by PLC0 to PLC6 bit in FLXAnFRWRHS2 register and DP0 to DP10 in FLXAnFRWRHS3 register.

1.5.3 Data Pointer

FlexRay module calculates the leading address of the data part from the data pointer of the header part when reading/writing the frame information to the data part of message buffer.

Data pointer sets the value of "Data Part Leading Address÷4 bytes" by assuming the start address of Message RAM (address of message buffer 0) as "H'0". Set the data pointer value to DP0 to DP10 bit in FLXAnFRWRHS3 register.

[Note] 1. Due to the above specifications, set the data part leading address to be "multiple of 4".
2. In operating, do not change the value of DP0 to DP10 bit in FLXAnFRWRHS3 register.
3. Even if you set an incorrect value, such as specifying the same address in multiple buffers whose data pointers point to the header part, the FlexRay module will still store data in the message buffer. In this case, it might be possible to lose the payload data in the data part and execute the wrong writing.
If try to write the frame data to the header, WAHP bit in FLXAnFRMHDF register will be "1". In his case, the header part is protected against the writing, and the data is lost.

■Data Pointer Calculate Example (1)

Table 1-11 shows configuration example of MessageRAM and the data pointer value for each buffer.

Table 1-11 Configuration Example of Message RAM

RAM (4 bytes)	Buffer Number	Configuration		Contents
0...3	Message buffer 0	Static buffer, DP = 12		Header Part
4...7	Message buffer 1	Static buffer, DP = 14		
8...11	Message buffer 2	Static buffer, DP = 16		←LCB
12		MB0 data 1	MB0 data 0	Data Part←DP MB0
13		Unused*	MB0 data 2	
14		MB1 data 1	MB1 data 0	←DP MB1
15		Unused*	MB1 data 2	
16		MB2 data 1	MB2 data 0	←DP MB2
17		Unused*	MB2 data 2	
18...2046		Unused		
2047		Unused		

[Note] *This area is not accessible.

[Explanation for Control Procedure Example]

LCB bit in FLXAnFRMRC = "2" since the number of message buffers is 3. In this case, the data pointer must be greater than or equal to (LCB +1 of FLXAnFRMRC register)*4 = "12".

Set SFDL bit in FLXAnFRMHDC register = "3"since the data length of static frame is 6 bytes.

The following shows the data pointer of each message buffer.

The data pointer of the message buffer 0 is DP bit in FLXAnFRWRHS3 register = "12".

The data pointer of the message buffer 1 is DP bit in FLXAnFRWRHS3 register = "14".

The data pointer of the message buffer 2 is DP bit in FLXAnFRWRHS3 register = "16".

In this operation example, FDB bit in FLXAnFRMRC register = "128" and FFB bit in FLXAnFRMRC register = "128" since FIFO buffer and dyamic buffer are not used.

■Data Pointer Calculation Example (2)

Table 1-12 shows the configuration example of MessageRAM and the data pointer value for each buffer.

Table 1-12 Configuration Example of Message RAM

RAM (4bytes)	Buffer Number	Configuration		Contents
0...3	Message buffer 0	Static buffer, DP = 2046		Header part
4...7	Message buffer 1	Static buffer, DP = 2044		
8...11	Message buffer 2	Dynamic buffer, DP = 2042		←FDB
12...15	Message buffer 3	Dynamic buffer, DP = 2040		←LCB
16...2039		Unused		Data part
2040		MB3 data 1	MB3 data 0	←DP MB3
2041		MB3 data 3	MB3 data 2	
2042		MB2 data 1	MB2 data 0	←DP MB2
2043		MB2 data 3	MB2 data 2	
2044		MB1 data 1	MB1 data 0	←DP MB1
2045		Unused*	MB1 data 2	
2046		MB0 data 1	MB0 data 0	←DP MB0
2047		Unused*	MB0 data 2	

[Note] *This area is not accessible.

[Explanation for Control Procedure Example]

LCB bit in FLXAnFRMRC = “3” since the number of message buffers is 4. In this case, the data pointer must be greater than or equal to (LCB +1 of FLXAnFRMRC register)*4 = “16”.

In this example, set two Static buffers and two Dynamic buffers. FIFO buffer is not used. Set FDB bit in FLXAnFRMRC register = “2”, LCB bit in FLXAnFRMRC register = “3”, and FFB bit in FLXAnFRMRC register = “128”.

Set SFDL bit in FLXAnFRMHDC register = “3” since the data length of the Static frame is 6 bytes.

Set PLC bit in FLXAnFRWRHS2 register of the message buffer 2 and 3 = “4” since the payload length of Dynamic buffer is 8 bytes.

The following shows the data pointer of each message buffer.

The data pointer of the message buffer 0 is DP bit in FLXAnFRWRHS3 register = “2046”

The data pointer of the message buffer 1 is DP bit in FLXAnFRWRHS3 register = “2044”

The data pointer of the message buffer 2 is DP bit in FLXAnFRWRHS3 register = “2042”

The data pointer of the message buffer 3 is DP bit in FLXAnFRWRHS3 register = “2040”

■Data Pointer Calculation Example (3)

Table 1-13 shows the configuration of MessageRAM and the pointer value for each buffer.

Table 1-13 Configuration Example of Message RAM

RAM (4bytes)	Buffer Number	Configuration		Contents
0...3	Message buffer 0	Static buffer, DP = 1100		Header part
4...7	Message buffer 1	Static buffer, DP = 1101		
8...11	Message buffer 2	Dynamic buffer, DP = 1985		←FDB
12...15	Message buffer 3	Dynamic buffer, DP = 1987		
16...19	Message buffer 4	FIFO buffer, DP = 79		←FFB
20...23	Message buffer 5	FIFO buffer, DP = 81		
24...27	Message buffer 6	FIFO buffer, DP = 83		
28...31	Message buffer 7	FIFO buffer, DP = 85		←LCB
32		Unused		Data part
33...78		Unused		
79		MB4 data 1	MB4 data 0	←DP MB4
80		MB4 data 3	MB4 data 2	
81		MB5 data 1	MB5 data 0	←DP MB5
82		MB5 data 3	MB5 data 2	
83		MB6 data 1	MB6 data 0	←DP MB6
84		MB6 data 3	MB6 data 2	
85		MB7 data 1	MB7 data 0	←DP MB7
86		MB7 data 3	MB7 data 2	
87...1099		Unused		
1100		MB0 data 1	MB0 data 0	←DP MB0
1101		MB1 data 1	MB1 data 0	←DP MB1
1102...1984		Unused		
1985		MB2 data 1	MB2 data 0	
1986		Unused*	MB2 data 2	←DP MB2
1987		MB3 data 1	MB3 data 0	
1988		Unused*	MB3 data 2	←DP MB3
1989...2046		Unused*		
2047		Unused*		

[Note] *This area is not accessible.

[Explanation for Control Procedure Example]

LCB bit in FLXAnFRMRC = “7” since the number of message buffers is 8. In this case, the data pointer must be more than or equal to (LCB +1 of FLXAnFRMRC register)*4 = "36".

In this example, for using two Static buffers and four Dynamic buffers, FDB bit in FLXAnFRMRC register = “2”, FFB bit in FLXAnFRMRC register = “4”, LCB bit in FLXAnFRMRC register = “7”.

Set SFDL bit in FLXAnFRMHDC register = “2” since the data length of the Static frame is 4 bytes.

Set PLC bit in FLXAnFRWRHS2 register of the message buffer 2 and 3 = “3” since the payload length of Dynamic buffer is 6 bytes.

Set PLC bit in FLXAnFRWRHS2 register of the message buffer 4 to 7 = “4” since the payload length of FIFO buffer is 8 bytes.

The following shows the data pointer of each message buffer.

The data pointer of the message buffer 0 is DP bit in FLXAnFRWRHS3 register = “1100”

The data pointer of the message buffer 1 is DP bit in FLXAnFRWRHS3 register = “1101”

The data pointer of the message buffer 2 is DP bit in FLXAnFRWRHS3 register = “1985”

The data pointer of the message buffer 3 is DP bit in FLXAnFRWRHS3 register = “1987”

The data pointer of the message buffer 4 is DP bit in FLXAnFRWRHS3 register = “79”

The data pointer of the message buffer 5 is DP bit in FLXAnFRWRHS3 register = “81”

The data pointer of the message buffer 6 is DP bit in FLXAnFRWRHS3 register = “83”

The data pointer of the message buffer 7 is DP bit in FLXAnFRWRHS3 register = “85”

2. State of Communication Control

2.1 Communication Control State Transferring Diagram

The transferring between each state performs by writing the command to CMD bit in FLXAnFRSUCC1 register.

Figure 2-1 shows the flow from the power supply to the network communication start.

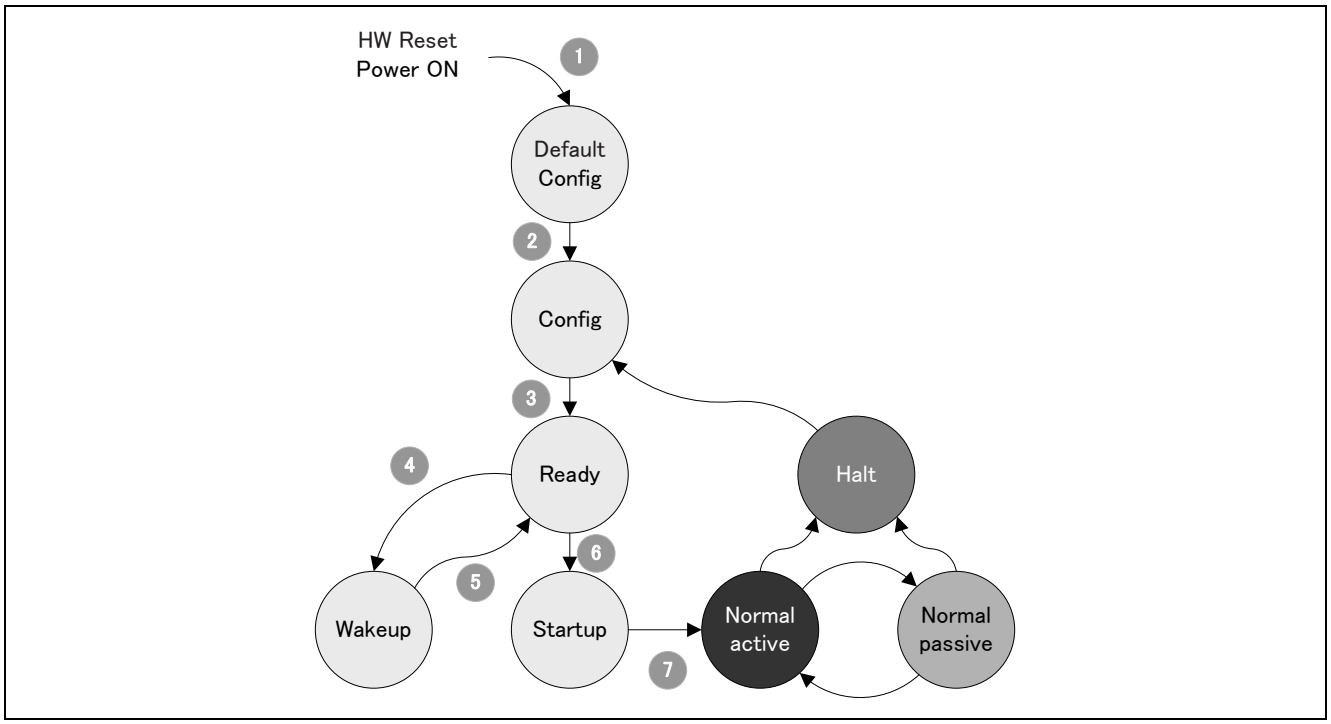


Figure 2-1 Flow until Network Communication Start

Table 2-1 shows the required operation for transferring between each state.

Table 2-1 Required Operation for Transferring between Each State

State before transferring	State after transferring	Required Operation
①All States	Default_config	―
②Default Config	Config	Write “0001” (Config) to CMD bit in FLXAnFRSUCC1 register.
③Config	Ready	After operating lock release for CLK bit in FLXAnFRLCK register, write “0010” (Ready) to CMD bit in FLXAnFRSUCC1 register.
④Ready	Wakeup	Write “0011” (Wakeup) to CMD bit in FLXAnFRSUCC1 register.
⑤Wakeup	Ready	Write “0010” (Ready) to CMD bit in FLXAnFRSUCC1 register, - (or automatically transfer by wakeup completion and suspension.
⑥Ready	Startup	Write “0100” (Run) to CMD bit in FLXAnFRSUCC1 register.
⑦ Startup	Normal active	- (Automatically transfer by startup completion)
Startup Normal active	Ready	Write “0010” (Run) to CMD bit in FLXAnFRSUCC1 register.

2.2 Default Config to Ready State

After resetting, it is in the Default Config state.

After transferring to Config state by CHI CONFIG command (CMD bit in FLXAnFRSUCC1 register), perform the initial setting for the various register. For transferring from CONFIG state to the other state, the CLK bit of the FLXAnFRLCK register must be unlocked before writing the command to CMD bit. This function prevents erroneous operation of the communication controller due to unintended actions such as software runaway.

Figure 2-2 shows the operation procedure example.

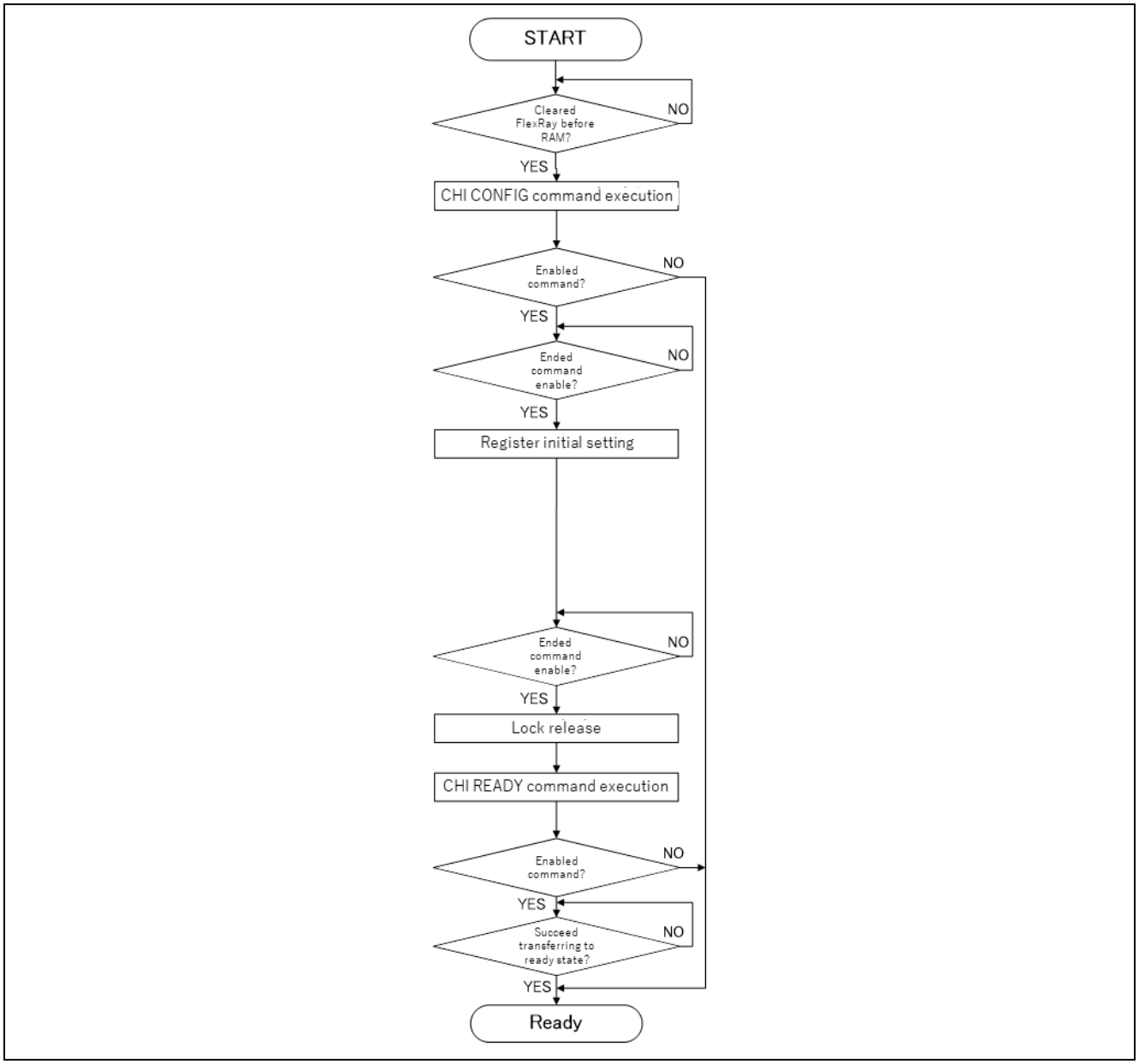


Figure 2-2 Setting Procedure Example

[Note] 1. After resetting, do not perform the setting for each register in clearing Message RAM. Reset the setting for each register After confirming that PBSY bit in FLXAnFRSUCC1 register has become “0”.

2. When transferring Config state to the other state, it is required to unlock by FLXAnFRLCK register. Continuously perform the writing to FLXAnFRLCK register when unlocking. If other processing is performed between the two writing, it needs to start from the first write again.

2.3 Wakeup State

In FlexRay, all channels need to be active state until starting communication.

In wakeup state, wakeup master node activates the nodes connected to the channel.

When the connecting channel is not active and the own node is wakeup master node, it is transferred to the wakeup state by CHI WAKEUP command (CMD bit in FLXAnFRSUCC1 register) and the wakeup sequence is executed.

When the one side of channel is active and the other side of channel wakes up, it returns to ready sate to config state, and it transferred to wakeup state after changing channel that transmitted wakeup pattern.

The changing of the wakeup status can be confirmed by WST bit in FLXAnFRSIR register and WSV0 to WSV2 bit in FLXAnFRCCSV register. When received the effective wakeup pattern, WUPA bit and WUPB in FLXAnFRSIR are set.

Figure 2-3 shows the state transferring diagram in wakeup.

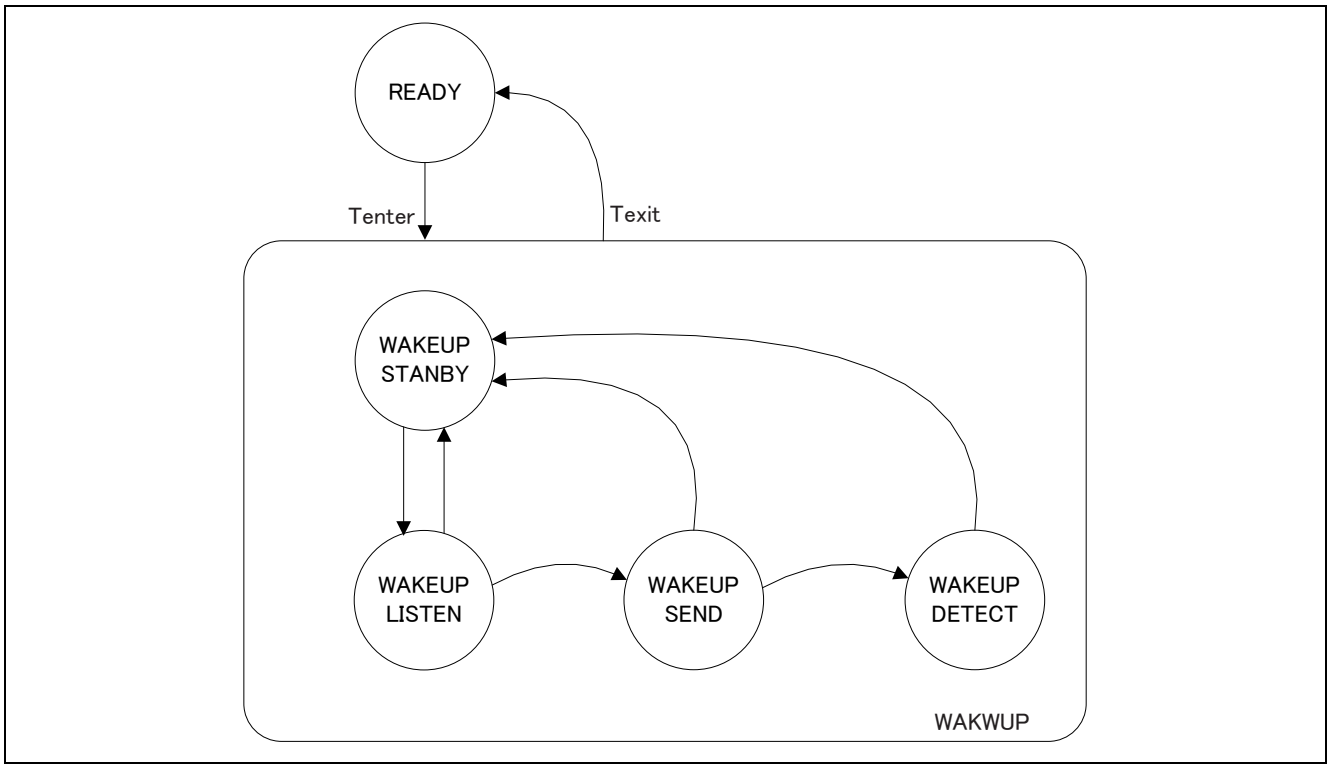


Figure 2-3 State Transferring Diagram in Wakeup

[Note] 1. In wakeup state, it is necessary to supply the minimum power necessary for waking up to the bus driver.
2. In FlexRay, it is not possible to check whether all nodes have become active state in wakeup state. For Cold_start node, it is necessary to consider the time required for all nodes to wake up and wait. before starting startup.
3. Select wakeup channel by WUCS bit in FLXAnFRSUCC1 register.
In FlexRay, it is disabled for one node to wake up two channels at the same time.

Figure 2-4 to Figure 2-7 show the wakeup procedure example when operating the wakeup master node and cold-start node.

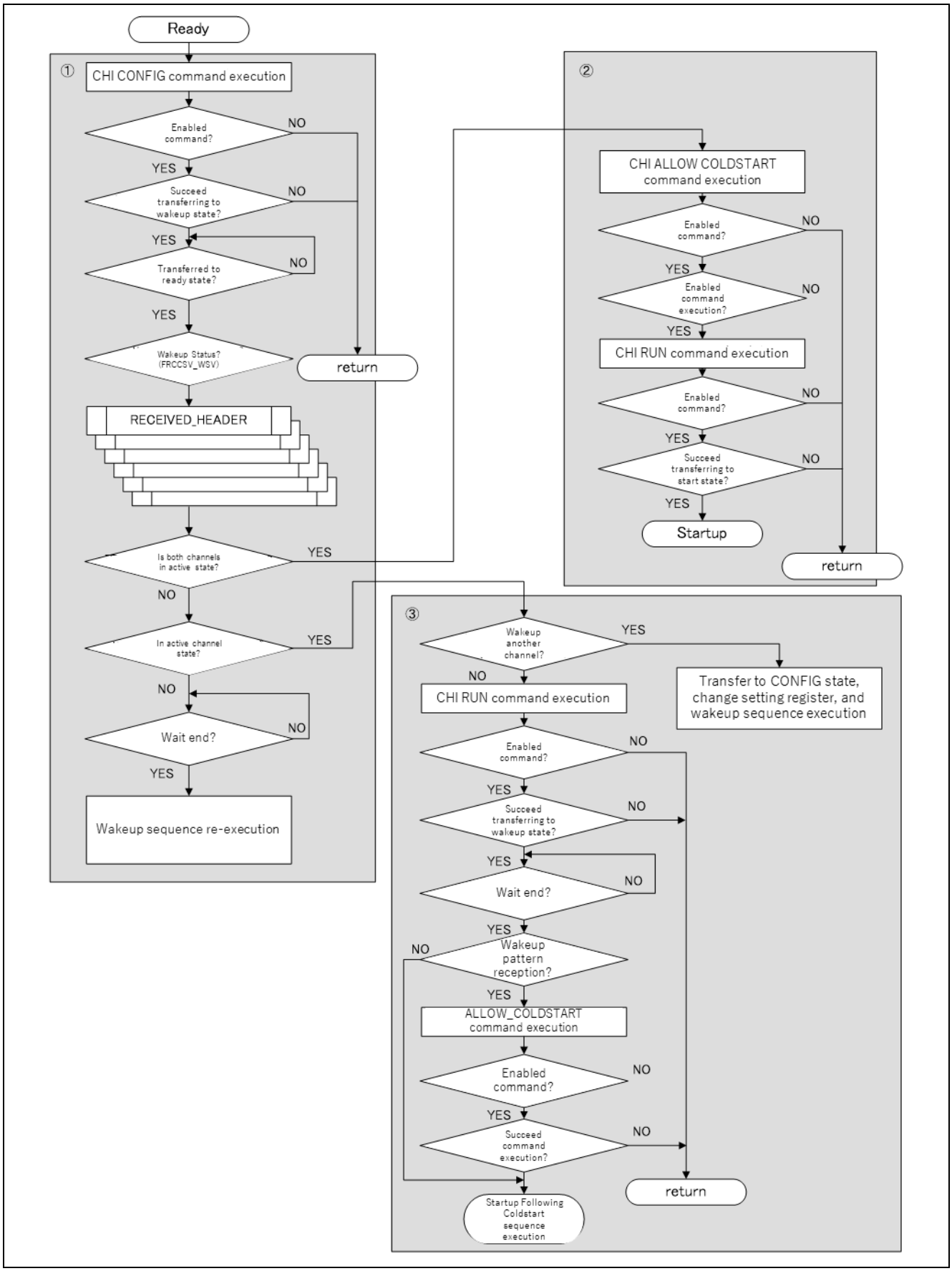


Figure 2-4 Wakeup Sequence Example (1)

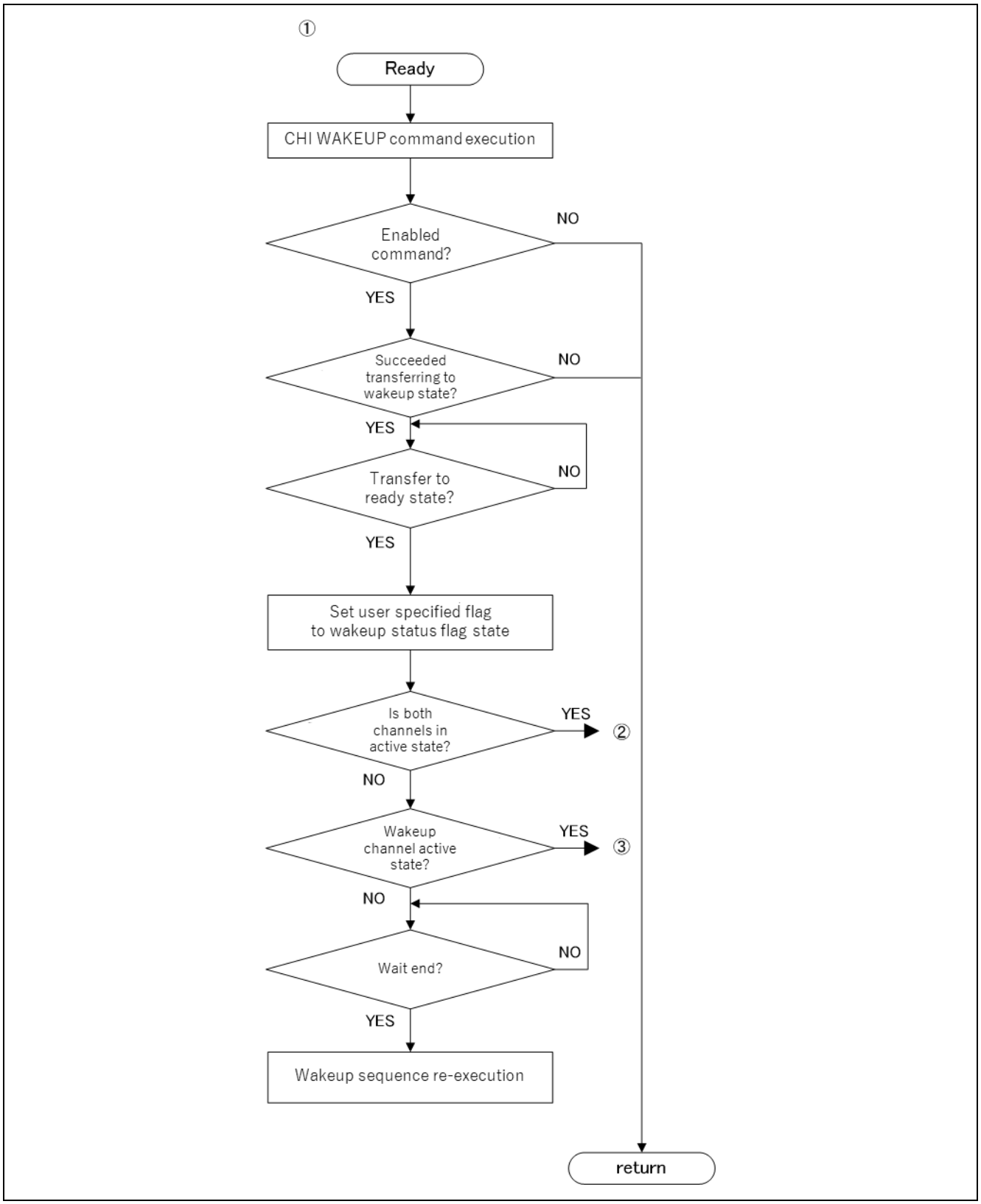


Figure 2-5 Wakeup Sequence Example (2)

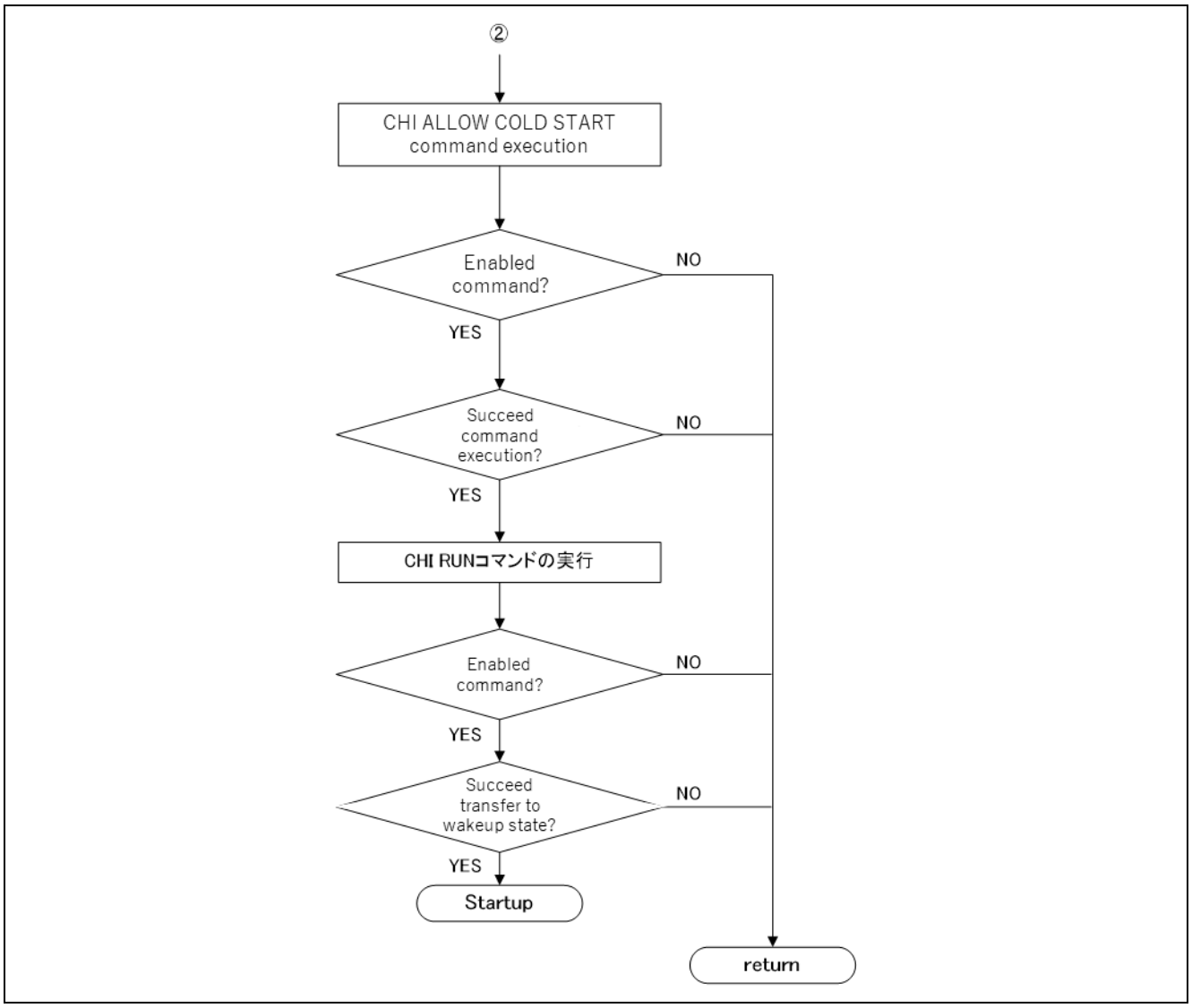


Figure 2-6 Wakeup Sequence Example (3)

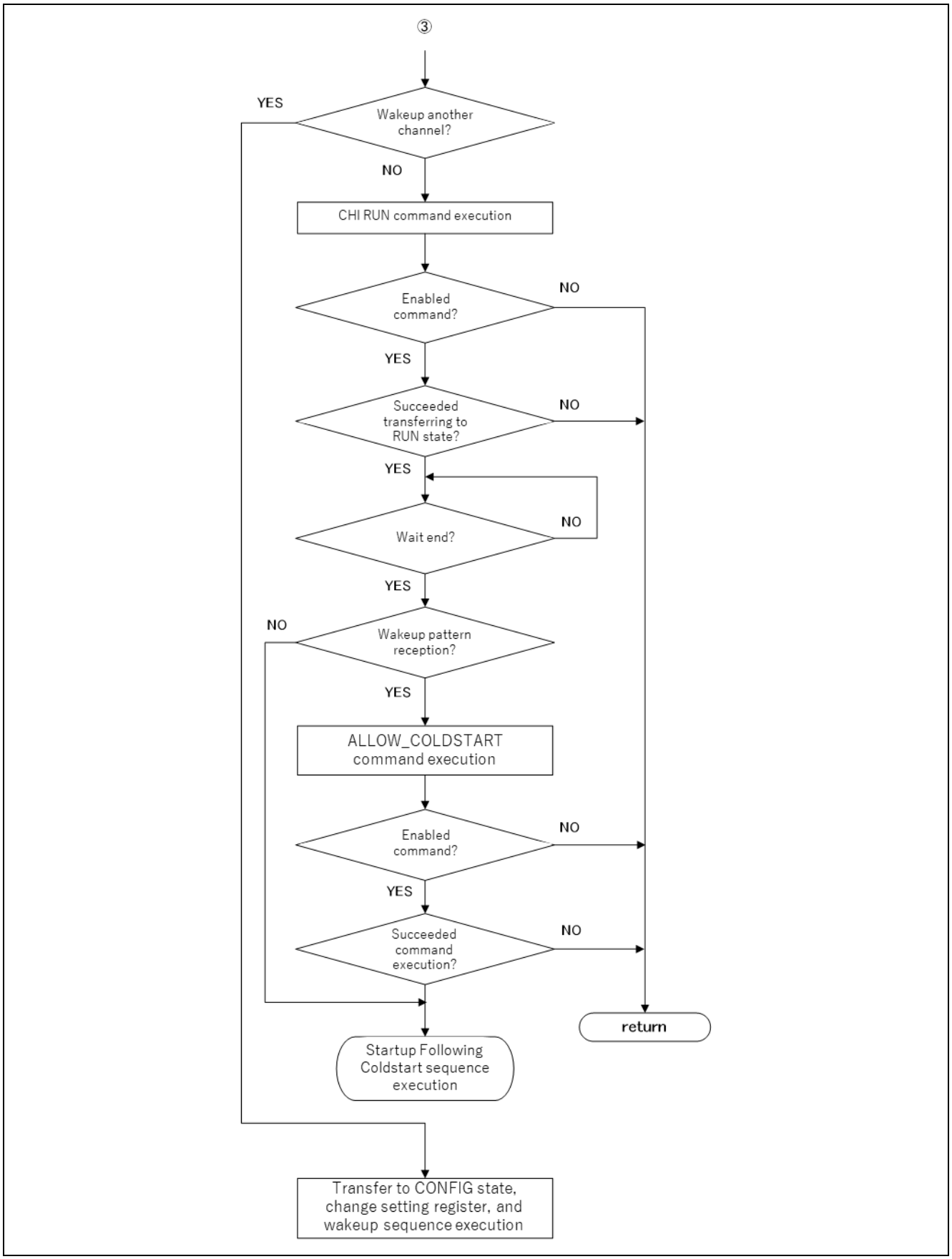


Figure 2-7 Wakeup Sequence Example (4)

2.3.1 Processing Example in RECEIVED_HEADER or COLLISION_HEADER

In RECEIVED_HEADER (WSV bit = “001” in FLXAnFRCCSV register) or COLLISION_HEADER (WSV bit = “011” in FLXAnFRCCSV register) state, abort the wakeup process, assuming that the cluster that the local node is trying to wake up on is already in the communication state.

Figure 2-8 shows the operation outline example.

In operation outline example, the own node transitions to the Startup state as a Leading Cold_start node.

Refer to Figure 2-4 for the control procedure example. Since both channels are active, control ② is performed.

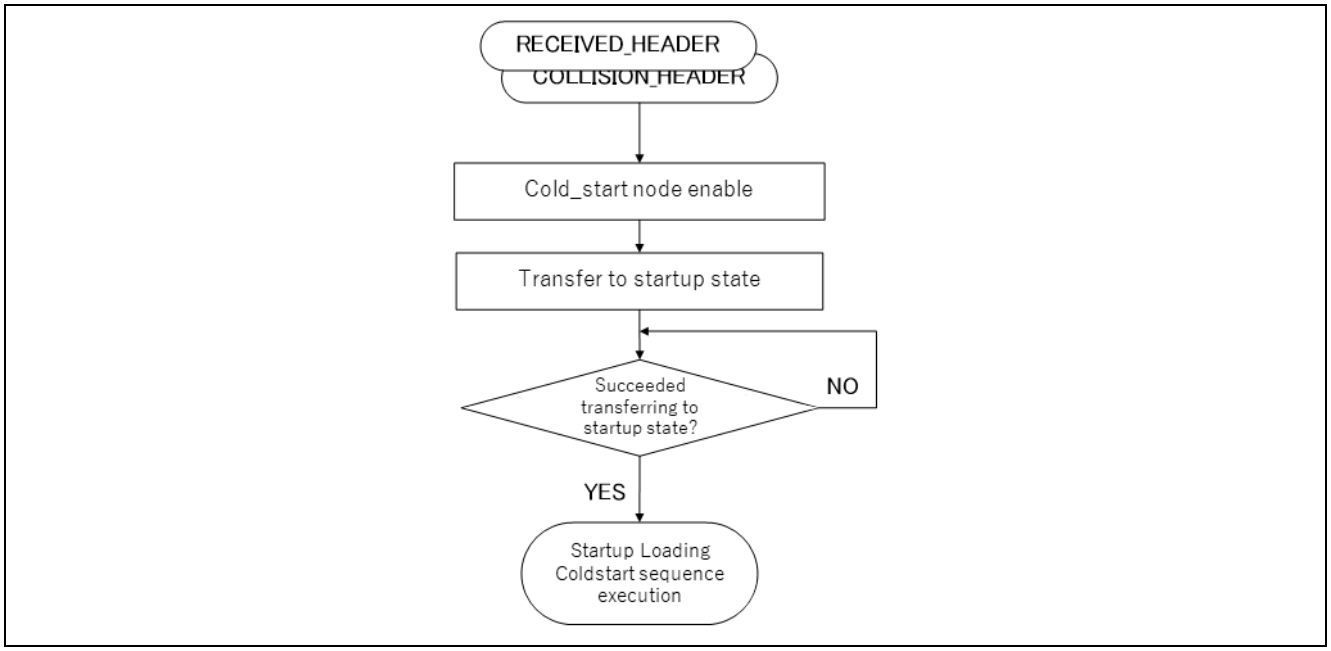


Figure 2-8 Setting Procedure Example

2.3.2 Processing Example in RECEIVED_WUP, COLLISION_WUP or TRANSMITTED

In RECEIVED_WUP (WSV bit in FLXAnFRCCSV register = “010”) or COLLISION_WUP (WSV bit = “100” in FLXAnFRCCSV register = “100”) state, abort the wakeup processing since the own node that attempts to wakeup is already wakening up by the other node.

In TRANSMITTED (WSV bit in FLXAnFRCCSV register = “110”) State, transfer to the startup state when the transmission of the wakeup pattern is succussed. In this case, wait for the time required for other nodes to perform wakeup processing.

When only one channel is active and the own node does not execute wakeup for the other channel, the CHI RUN command (CMD bit in FLXAnFRSUCC1 register) will transition to the startup state. In this case, wait for the other node to wake up the other channel. As a Following Cold_start node, it waits for the startup of other Cold_start nodes and integrates them.

Figure 2-9 shows the setting operation procedure example.

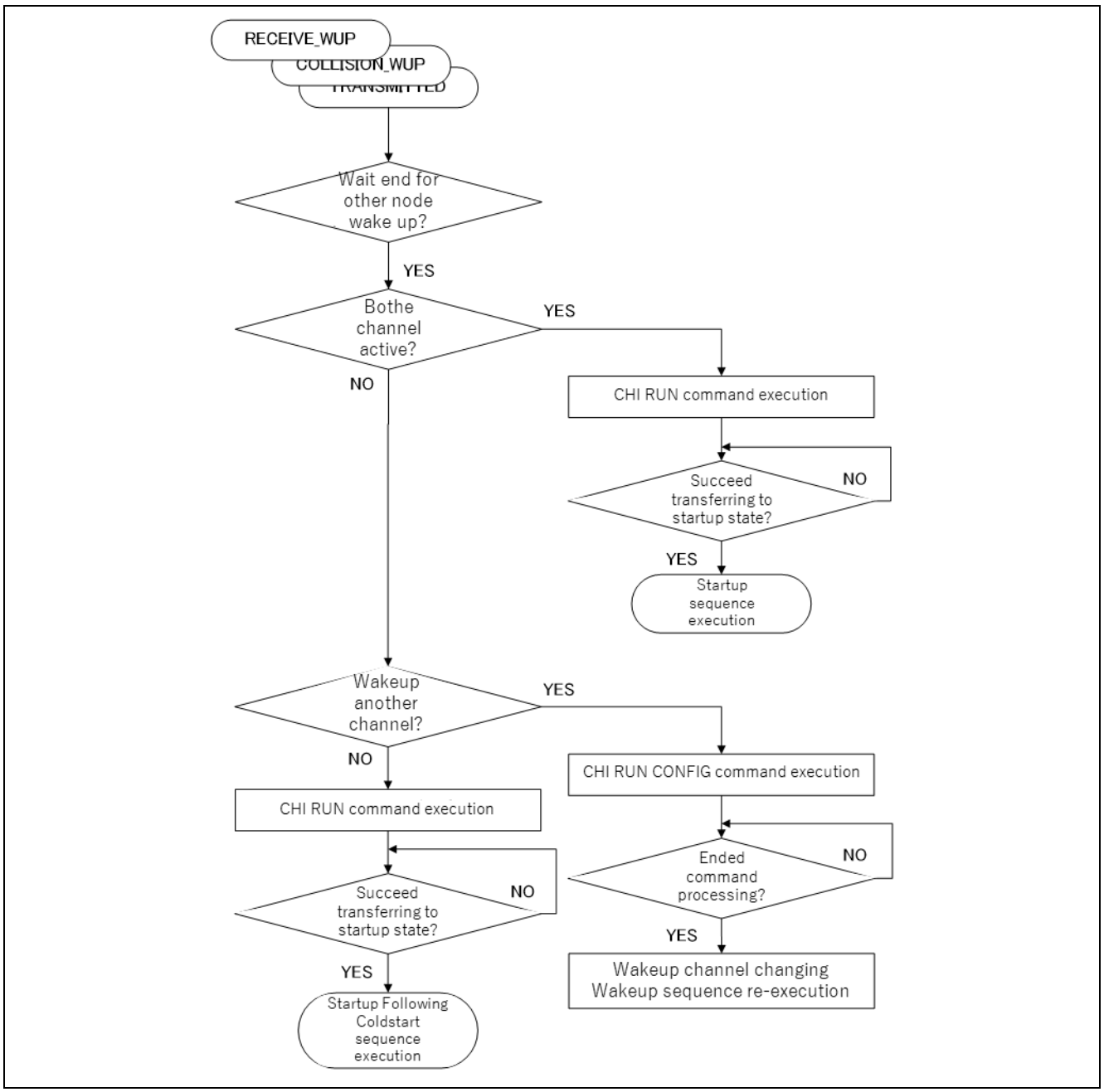


Figure 2-9 Setting Procedure Example

2.3.3 Processing Example in COLLISION_UNKNOWN

Stop the wakeup processing by escaping from WAKEUP_DETECT state after reaching the internal wakeup timer to the specified value without receiving the effective wakeup pattern and the header part of the effective frame.

When the communication controller is in such a status, it can be inferred that the influence of noise is large or that there is an abnormality in communication.

When execute the wakeup processing again, wait for a certain time for the internal processing of the other node.

[Note] Listen timeout noise timer value is settable by LTN0~LTN3 in FLXAnFRSUCC2 register.

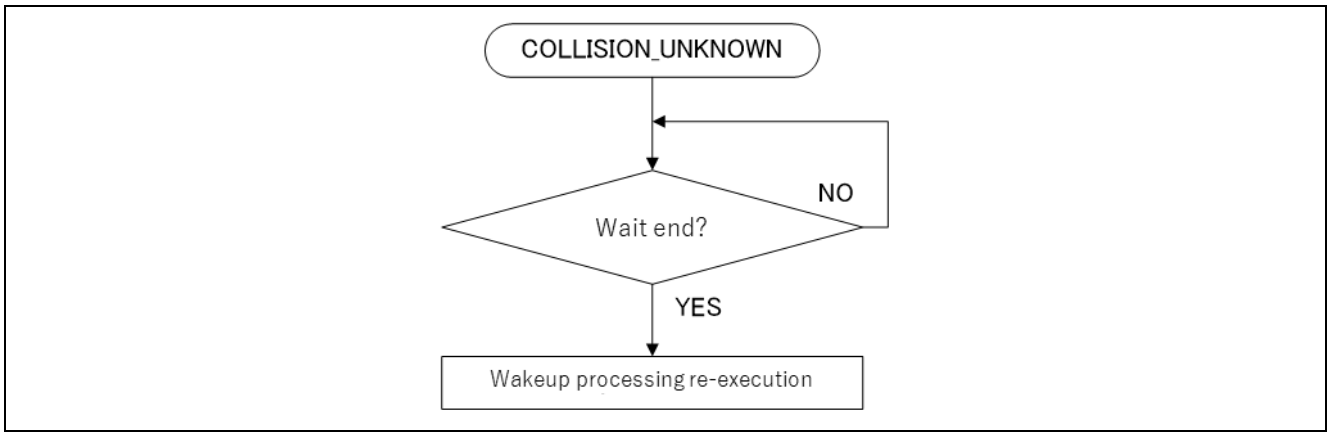


Figure 2-10 Control Procedure Example

2.4 Startup Sate

In FlexRay, it is required to synchronize the clocks in the entire cluster before starting the communication. Each node performs the startup processing and integrate the clock of own node to the clock of Cold_start node.

At least two Cold_start nodes are required for starting up the cluster.

Refer to “1. Initial Setting” for setting for operating as startup node and sync node.

Figure 2-11 shows the state transferring diagram in startup.

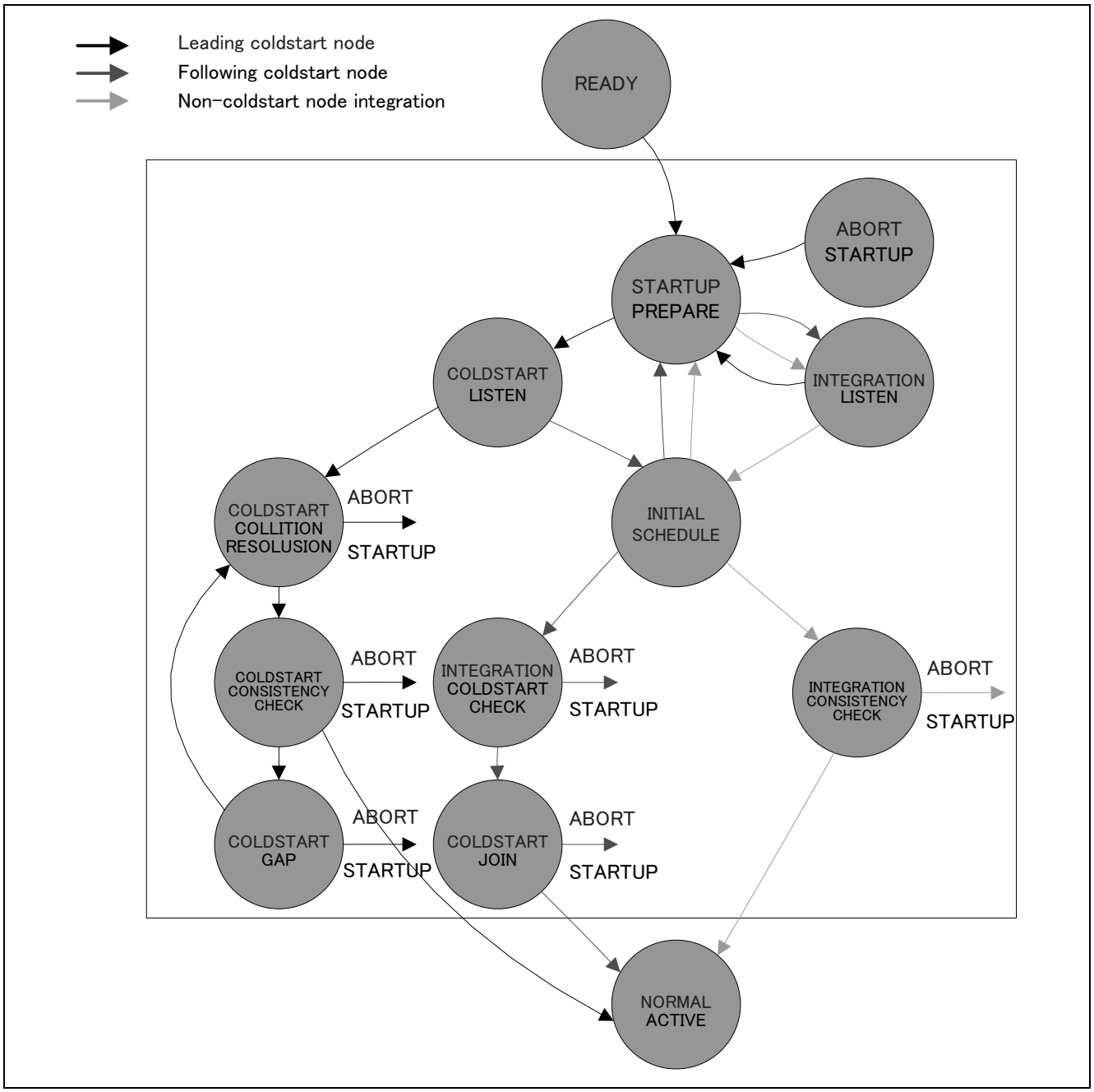


Figure 2-11 State Transferring in Startup

2.4.1 Operation for Cold_start Node

In the ready state (POCS bit in FLXAnFRCCSV register = “000001”), transfer to the startup sate by executing CHI RUN command (CMD bit in FLXAnFRSUCC1 register = “0100”).

Set the number of trials for Cold_start by CSA0 to CSA4 bit in FLXAnFRSUCC1 register. The number of Cold_start trials for Cold_start node should be at least 2 and the same for all Cold_start nodes in the cluster.

Startup node starts the startup as Cold_start node when CSI bit in FLXAnFRCCSV register is “0”. This bit becomes “0” by executing CHI ALLOW_COLDSTART (CMD bit in FLXAnFRSUCC1 register = “1001”).

Nevertheless, when the communication is continuing, the starting Startup at the same time as the Cold_start node transitions to the Startup state may disrupt communication. For avoiding above, it is required to transfer to the startup state in the state that CSI bit set to “1” (startup enable).

In this case, Cold_start node enters to INTEGRATION_LISTEN state after CHI RUM command execution (transfer to startup state).

If there is no detection of the communication state, CHI ALLOW_COLDSTART command (CMD bit in FLXAnFRSUCC1 register = “1001”) is executed, and CSI bit is cleared. transfer to COLDSTART_LISTEN state. This results in a COLDSTART_LISTEN state.

[Note] When Cold_start is failed, read the status register and the error register.
When executing the startup again, please consider disabling Cold_start using the CSI bit of the FLXAnFRCCSV register to avoid interfering with ongoing communication.

Figure 2-12 shows the control procedure example of Cold_start node.

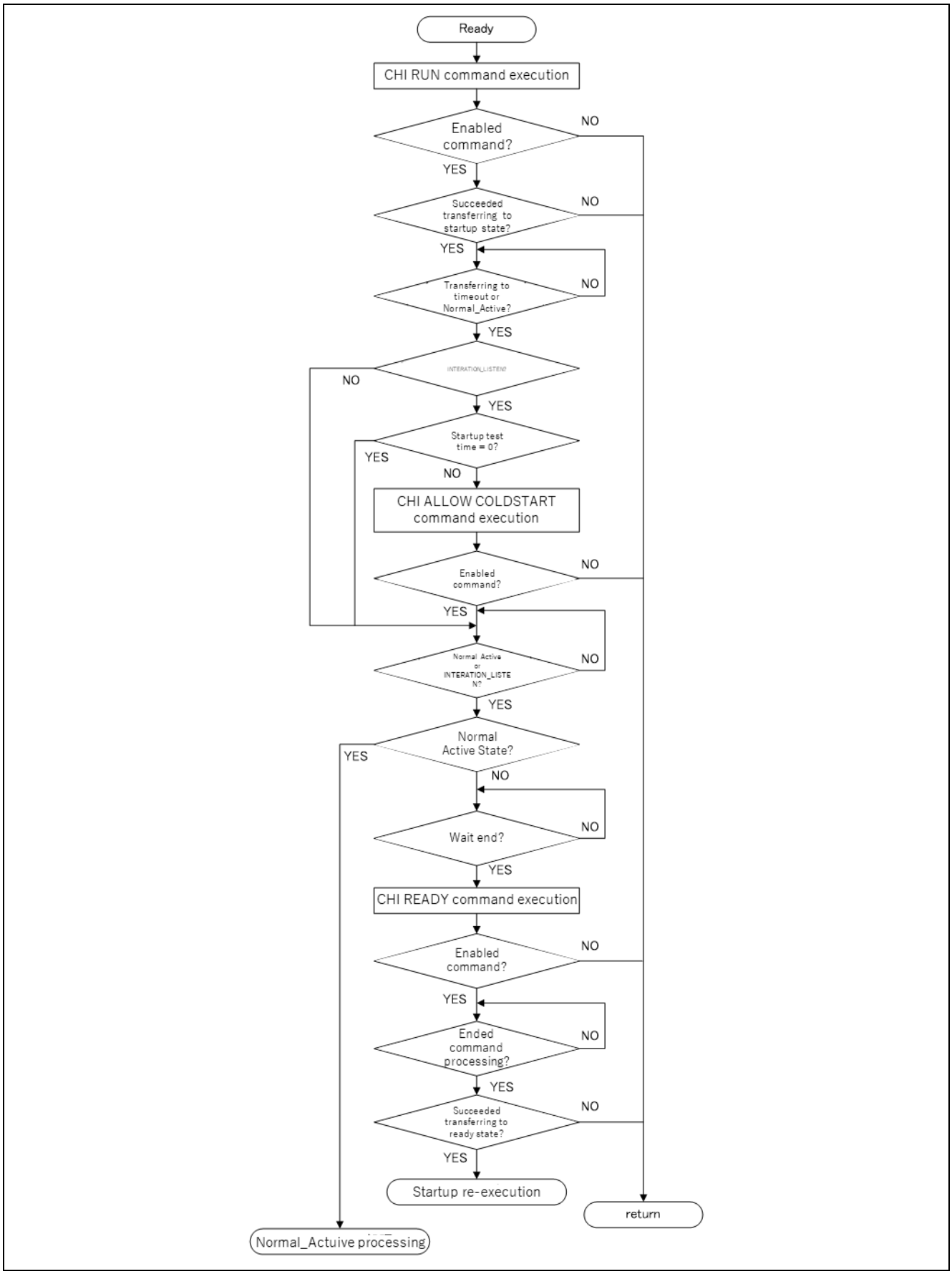


Figure 2-12 Control Procedure Example for Cold_start Node

■Pass for Leading Cold_start Node

When the communication is not executed, the node is transferred to COLDSTART_COLLISION_RESOLUTION state and transmit CAS symbol. After transmitting CAS symbol, the cycle 0 is started.

Multiple Cold_start nodes may initiate Startup at the same time, but FlexRay avoids this situation in the first four cycles.

If the Cold_start node receives the CAS symbol or frame header during the first four cycles (cycle 0 to cycle 3), it immediately returns to the COLDSTART_LISTEN state. Therefore, only one node remains on Leading Cold_start node sequence.

In cycle 4, Cold_start nodes other than the Leading Cold_start node start sending frames.

The Leading Cold_start node transitions to the COLDSTART_CONSISTENCY_CHECK state in four cycles after COLDSTART_COLLISION_RESOLUTION. In cycles 4 and 5, the node corrects its own clock by receiving Startup frames from other Cold_start nodes.

If the clock correction is successful and the 2-cycle startup frame is successfully received from at least one Cold_start node other than the own node, it will transition to the Normal_Active state.

The number of Cold Startup retries that can be executed by Cold_start node is set by CSA0 to CSA4 in FLXAnFRSUCC1 register. The remaining number of retries can be referenced with RCA0 to RCA4 in FLXAnFRCCSV register. The values of RCA0 to RCA4 are decremented by 1 each time retry.

It is possible to transition to the COLDSTART_COLLISION_RESOLUTION state only when this value is greater than "0", and to the OLDSTART_LISTEN state only when this value is greater than "1". If the number of retries is set to "1", it is prohibited for the own node to start Cold Start, but integration with other nodes is possible.

[Note] Set the values for CSA0 to CSA4 bits to be "2" or more and set the same values for the cluster.

■Pass for Following Cold_start Node

The Following Cold_start node transitions to the COLDSTART_LISTEN state, receives two cycles of valid Startup frames from the Leading Cold_start node, and obtains schedule and clock correction information.

First, it transitions to the INITIALIZE_SCHEDULE state as soon as it receives one effective startup frame.

If the second effective startup frame is successfully received, clock synchronization and schedule information acquisition is successful, it transitions to the INTEGRATION_COLDSTART_CHECK state.

During the two cycles of INTEGRATION_COLDSTART_CHECK state, it receives all sync frames from other nodes and performs clock correction. After performing clock correction without errors and ensuring that the Leading Cold_start node is in effective state, it transitions to the COLDSTART_JOIN state.

Transmission of startup frame is started in COLDSTART_JOIN state.

In this state, the Leading Cold_start and Following Cold_start nodes check the schedules of each other. If the clock correction error is detected, the Startup process aborts.

The node transitions to the Normal_active state if it receives at least one startup frame in every even cycle and at least one pair of startup frames in every two consecutive cycles.

As a result, the Following Cold_start node exits the startup state at least one cycle after the Leading Cold_start node.

2.4.2 Operation for non-cold_start Node

The nodes other than Cold_start node is integrated to Cold_start of the clock as the non-cold_star node.

It becomes the startup state by executing CHI RUN command (CMD bit in FLXAnFRSUCC1 register = “ 0100”) in the ready state (POCS bit = FLXAnFRCCSV register = “000001”).

Figure 2-13 shows the control procedure example for non-cold_start node.

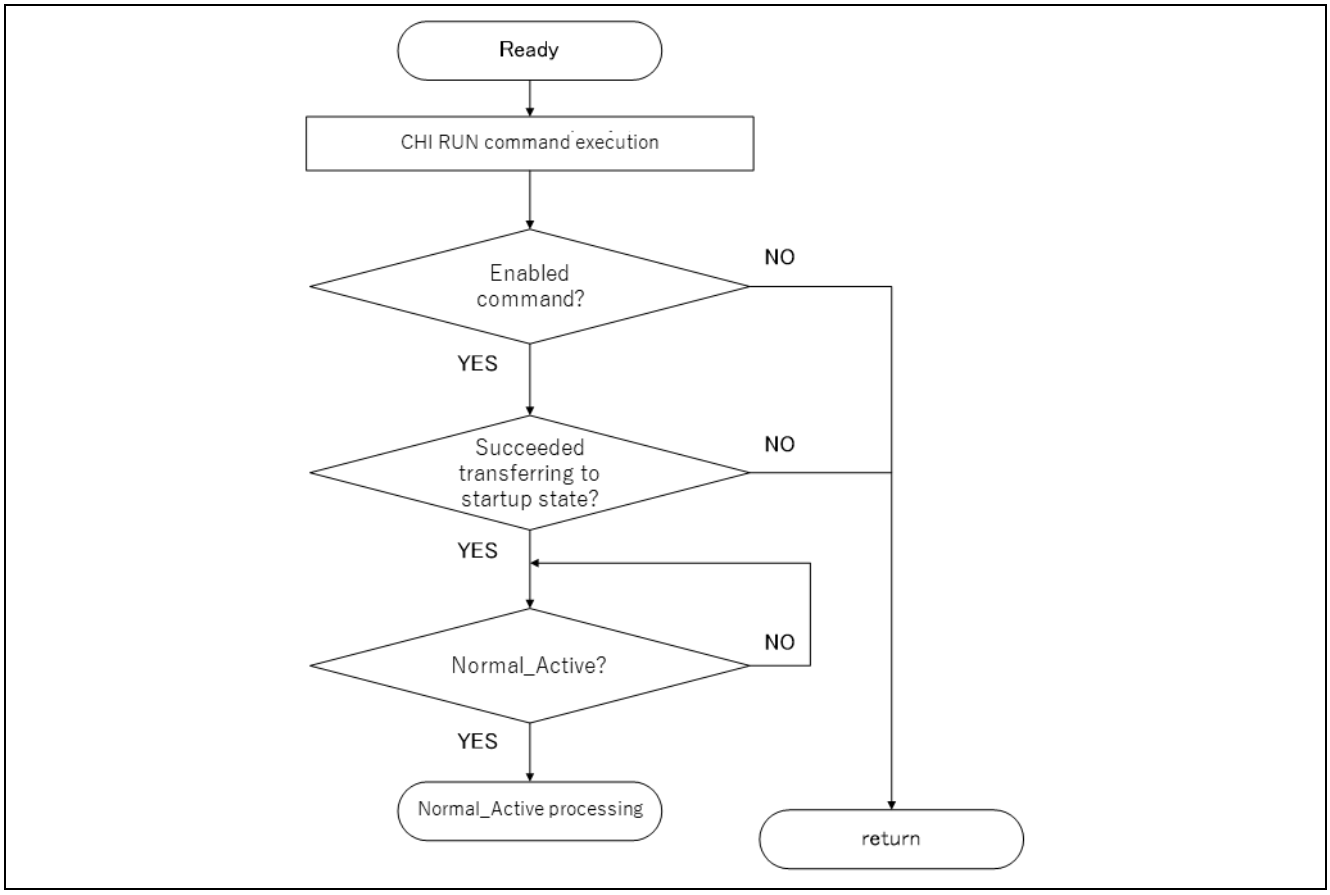


Figure 2-13 Operation Procedure for Non-Cold_start Node

After executing CHI RUN command (CMD bit in FLXAnFRSUCC1 register = “0100”), non-cold_start node checks the state of the connection channel for a period of time by transferring to INTEGRATION_LISTEN state.

When receiving the enable startup, it transfers to INITIALIZE_SCHEDULE state. When it successfully synchronizes the clock to the second effective startup frame and obtains the schedule information. It transitions to the INTEGRATION_CONSISTENCY_CHECK state.

In the INTEGRATION_CONSISTENCY_CHECK state, the node verifies that clock synchronization can be performed accurately and that (at least two) Cold_start nodes send Startup frames containing the same schedule information as itself. Aborted if the error is detected.

Non-Cold_start node must receive two valid Startup frames, or the Startup frame of the node it has integrated, in an even cycle in this state. The node suspends the integration and aborts when the reception is failed.

Also, it must receive two valid pairs of startup frames, or pairs of Startup frames for nodes that this node has integrated, in the first two cycles. If reception fails, the node stops integrating and aborts.

When the node fails to receive two or more startup frames within an even number of cycles after the first two cycles, or fails to receive two valid startup frame pairs within two cycles, the node aborts the integration and aborts.

In this state, the node receives two valid Startup frames in two consecutive even-odd cycle pairs and transitions to the Normal_active state.

As a result, the node will exit the Startup state at the end of an odd cycle at least two cycles after the Leading Cold_start node.

3. Data Setting to MessageRAM & Data Read from MessageRAM

3.1 Configuration for Transmit Frame Data

Set the header part (transmit/receive filtering condition) by FLXAnFRWRHS1 to FLXAnFRWRHS3 register.

Set the data part by FLXAnFRWRDS1 to FLXAnFRWRDS64 register.

The data set by FLXAnFRWRHS1 to FLXAnFRWRHS3 register and FLXAnFRWRDS1 to FLXAnFRWRDS64 register are transferred to MessageRAM (message buffer) via IBF (input buffer).

Table 3-1 shows the setting example of header part. Figure 3-1 shows the control procedure example.

Table 3-1 Setting Example of Header Part

Name	Setting Value	Note
Frame ID	1	
Cycle filtering	Every communication cycle	
Transmit/Receive channel	Both Ch. A/B	
Transmit/Receive Setting	Transmission	
Network control flag	Not use	
Transmit mode	Continuous transmit mode	
Transmit/Receive completion interrupt	Not use	
Header CRC	0xXXXX	
Data length	16 bytes	
Data pointer	Data part lead address H'400	

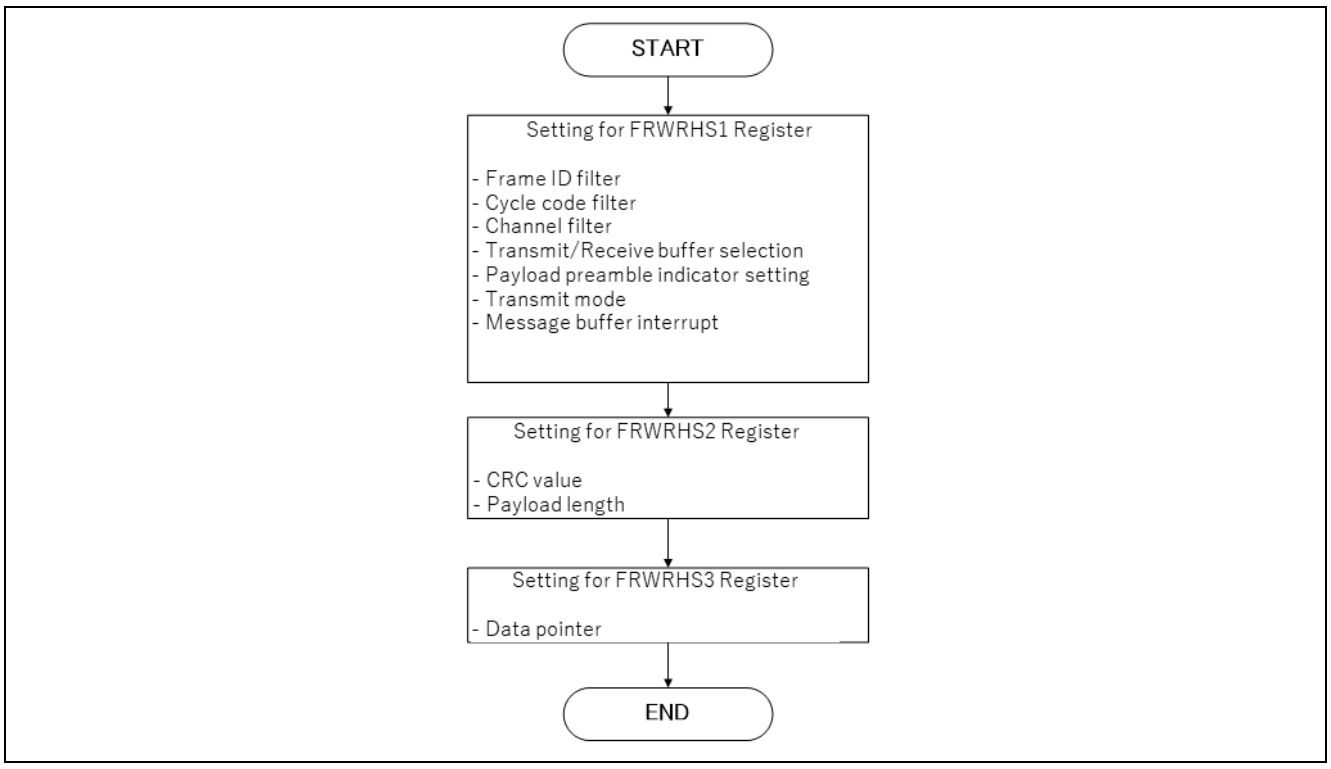


Figure 3-1 Control Procedure Example

■Header CRC

FlexRay introduces HeaderCRC to detect data corruption in the header. The HeaderCRC value must be calculated from the values of SyncFrameIndicator, StartupFrameIndicator, FrameID, and PayloadLength in the header.

[Note] According to the FlexRay specification, the communication controller does not calculate the Header CRC when transmitting frames. Header CRC must be calculated in advance and set in the header using a configuration tool, Host MCU, etc.

3.2 IBF (Input Buffer) : Transfer to MessageRAM

In FlexRay module, the writing to Message RAM is performed via IBF for ensuring the consistency of transmitted data. Also, the two-faces structure of the host side (CPU side)/shadow side (Message RAM side) is implemented to IBF part for improving the data transmission performance.

3.2.1 Transfer Method

(1) Set the data part and the header part to IBF (host side).

(FLXAnFRWRDS1 to FLXAnFRWRDS64 register, FLXAnFRWRHS1 to FLXAnFRWRHS3 register, and FLXAnFRIBCM register).

(2) Set the buffer number of transfer destination to IBRH bit in FLXAnFRIBCR register.

(3) The writing of (2) becomes the transmit request, IBF host and IBF shadow are switched, and the transferring of IBF and MessageRAM in shadow side is started.

In this time, IBSYS bit becomes “1” and is indicated that it is busy state.

(4) If there is the next transferring data, the data part and the header part are set to IBF (host side).

In this time, IBSYH bit becomes “1” when IBSYS bit is “1”.

(5) When completing the transfer, IBF host and IBF shadow are switched, and the next transfer that is waited is started.

In this time, IBSYS bit holds “1” and IBSYH bit becomes “0”.

[Note] When executing the setting for requesting the next transfer to IBRH bit in the states that IBSYS bit and IBSYH bit are “1”, it will be error and IIBA bit in FLXAnFREIR register becomes “1”. In this time, the value of IBF is not changed.

Figure 3-2 shows the double buffer configuration of input data.

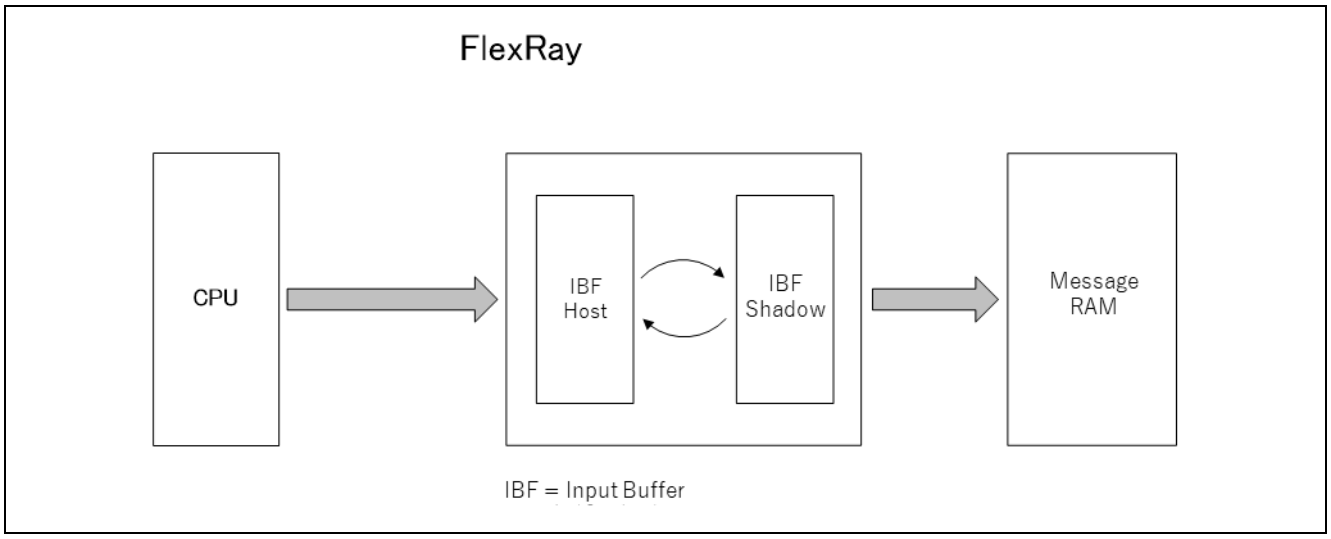


Figure 3-2 Double Buffer of Input Buffer

Figure 3-2 shows the setting example when transferring the data of the transmit frame to the message buffer of MessageRAM. Figure 3-3 shows the control procedure example.

Table 3-2 Setting Example of Transmit Frame Data

Name	Setting value	Note
Header part	Transmit	
Data part	Transmit	
Transmit request	Transmit request ON	
Transmit destination buffer number	1	Transmit to 2nd buffer since the buffer number is counted form “0”.

[Note] 1. Do not the header part to the locked buffer by SEC0 to SEC1 bit in FLXAnFRMRC register.
2. It is exclusively reception for FIFO buffer. Do not transmit to the data part.

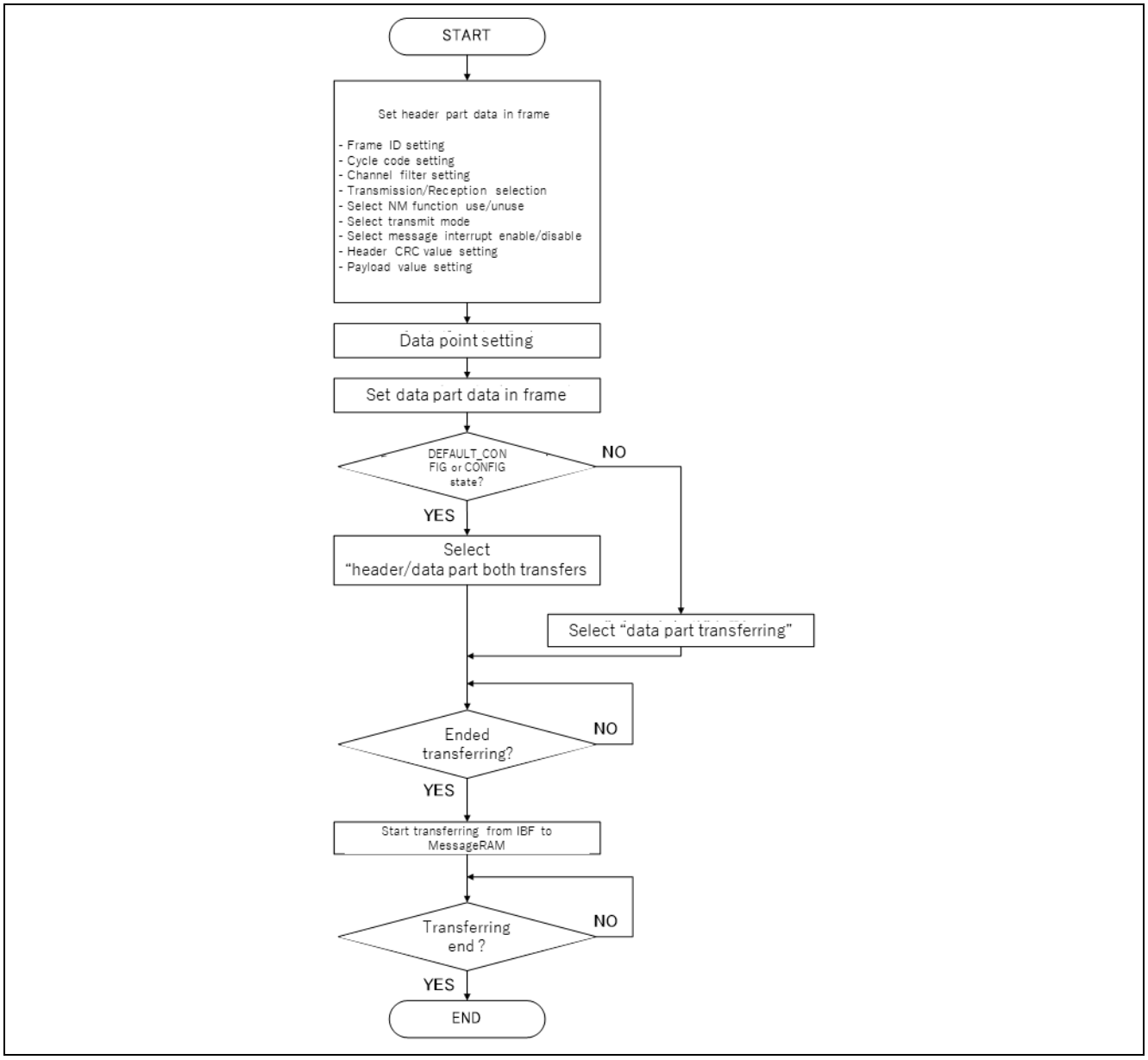


Figure 3-3 Control Procedure Example

3.3 OBF (Output Buffer) : Reading from MessageRAM

In FlexRay module, the writing to Message RAM is performed via OBF for ensuring the consistency of reception data. Also, the two-faces structure of the host side (CPU side)/shadow side (Message RAM side) is implemented.

The header part read from MessageRAM is stored to FLXAnFRRDHS1 to FLXAnFRRDHS3 register, and the data part from MessageRAM is stored to FLXAnFRRDHS1 to FLXAnFRRDHS3 register.

3.3.1 Transmission Method

(1) Set the buffer number to OBRS bit in FLXAnFROBCR register.

(2) After checking that OBSYS bit is “0”, set REQ bit to “1”.

By this setting, the transmission from the message buffer to OBF is started.

(3) OBSYS bit automatically becomes “0” when completed the transmission.

(4) After checking that OBSYS bit is “0”, set VIEW bit to “1”.

By this setting, OBF host and OBF shadow in OBF buffer are switched.

(5) Read the data from OBF buffer.

[Note] 1. When OBSYS bit is “1” and REQ bit is “1”, the error is occurred and IOBA bit in FLXAnFREIR register becomes “1”.
2. When transferring the data form FIFO buffer, set the lead number of FIFO buffer is set to OBRS bit.

Figure 3-4 shows the double-buffer of the output buffer.

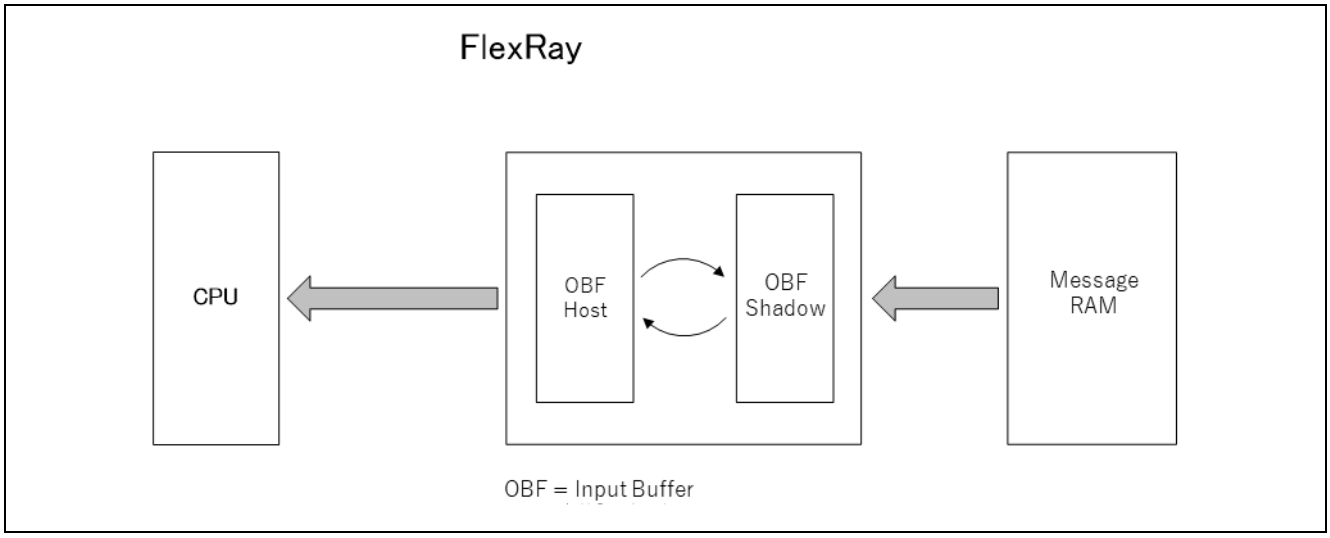


Figure 3-4 Double-buffer Configuration of Output Buffer

Table 3-3 shows the setting example when reading the frame received by the buffer number 1 of MessageRAM to OBF. Figure 3-5 shows the control procedure example.

Table 3-3 Setting Example in Transmit Frame Read

Name	Setting Value	Note
Header part	Transmit	
Data part	Transmit	
Transmit Source Buffer Number	1	

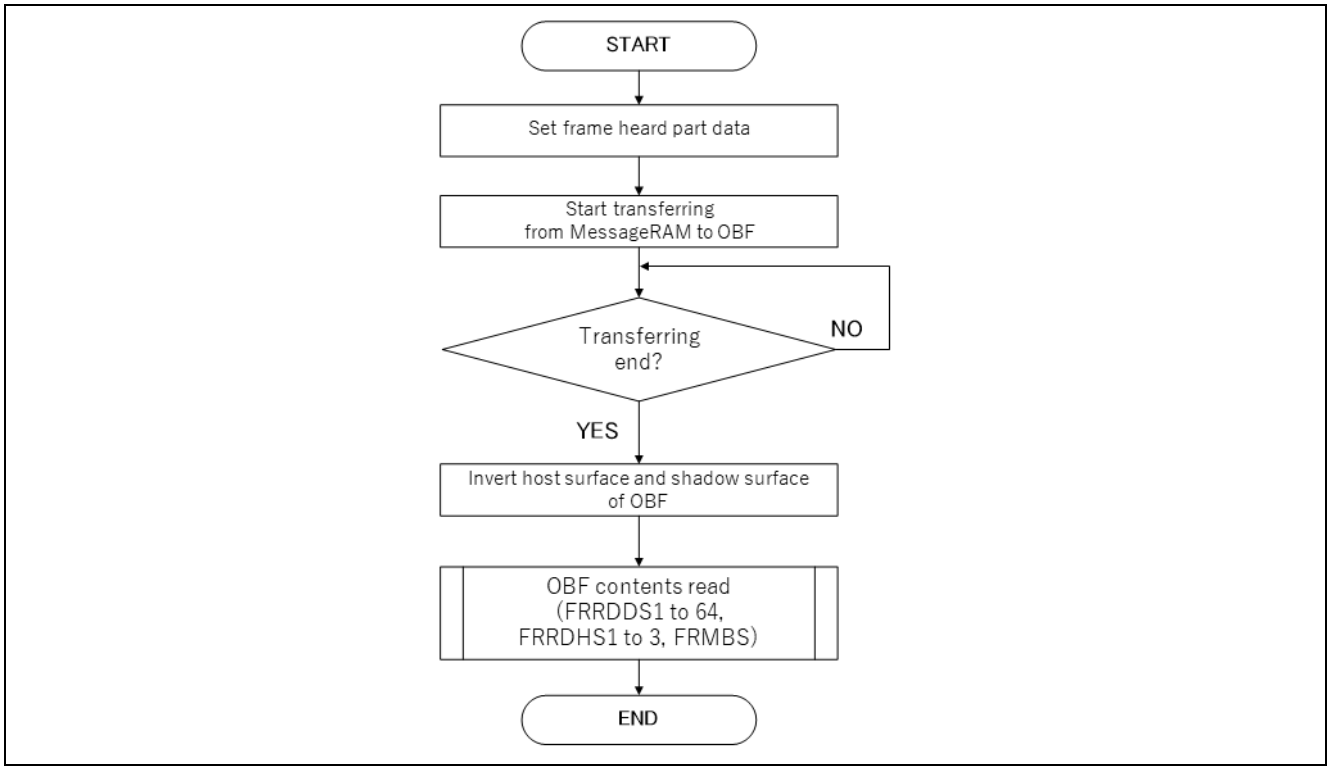


Figure 3-5 Control Procedure Example

4. Frame Transmission/Reception

4.1 Frame Transmission

In FlexRay, the communication is performed based on TDMA method. Therefore, the setting for the frame transmit timing detail is not required. The communication controller controls frame transmission timing according to the preset communication schedule.

The following shows the flow for the transmit data standby to the frame transmission.

(1) Set the target frame to IBF. (In this time, set CFG bit in FLXAnFRWRHS1 register = “1”.)

(2) Transfer the transmit frame data to IBF⇒Message RAM.

When STXRH in the header part is “1”, the transmit request flag of the target message buffer (the flag corresponding to FLXAnFRTXRQ1 to 4 registers) becomes “1” and transmission waiting status.

(3) The frame transmission is started by ActionPoint of the applicable slot.

(4) When setting the transmission completion interrupt to “enable” (MBI bit in FLXAnFRWRHS1 register), the interrupt request is issued when completed the transmission.

In this time, the message buffer number that is succussed the transmission is stored to MBT bit in FLXAnFRMHDS register.

[Note] It is required to start the transmission from IBF buffer to MessageRAM until the slot starting that transmits the frame. (Example: For transmitting the frame by the slot “5”, it is required to start the transmission to MessageRAM from IBF buffer until the slot “5” starting.)

■Continuous Transmission Mode and Single-shot Mode

Select TXM bit in FLXAnFRWRHS1 register by the transmission mode.

If selecting the single-shot mode, TXR flag becomes “0” when completing the transmission.

If selecting the continuous transmission mode, TXR flag does not become “0” when completing the transmission.

■Padding Function

In static segment, when the payload length that sets to the header part of the message buffer by the value that is set to SFDL bit in FLXAnFRMHDC register, the padding data is added. The padding value is “0”.

4.2 Frame Reception

In FlexRay, the communication is performed based on TDMA method. This communication controller confirms not only “format of receive frame is normal or not” but also “received frame timing is suitable or not” and “various flag in frame is normal or not”.

The following shows the flow from the data detection to the frame reception on the bus.

(1) Confirm if the frame format is correct.

(When detected CRC error, etc, it treats as Syntax Error.)

(2) Confirm if the header content is normal.

(When detected cycle counter value error, etc, it is treated as Syntax Error.)

(3) Confirm whether the conditions for storing in the receive buffer are met.

(4) Confirm whether the conditions for storing in the receive FIFO are met.

Among frames that meet the various reception conditions stipulated by the protocol, only frames that satisfy the filtering conditions (Frame ID, Channel ID, Cycle counter) set in the receive buffer/receive FIFO are stored in the receive buffer.

The flag corresponding to FLXAnFRNDAT1 to 4 register and FLXAnFRMBSC1 to 4 register becomes “1” by the storing to the receive buffer.

In this time, the message buffer number that is succeeded the reception is stored to MBU bit in FLXAnFRMHDS register.

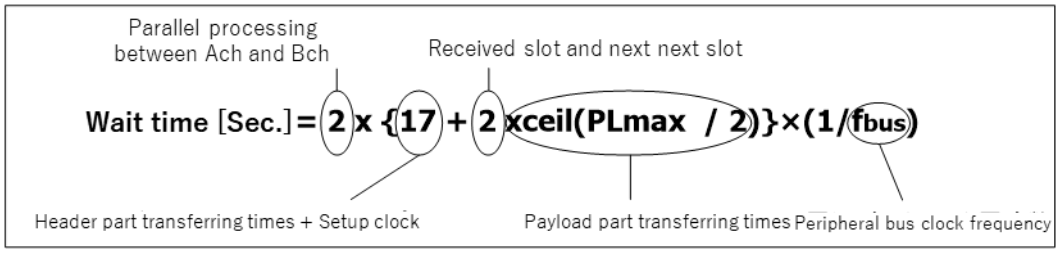
When enabling the receive completion interrupt (MBI bit in FLXAnFRWRHS1 register = “1”) in setting the header, the interrupt request by the frame reception completion (RXI bit in FLXAnFRSIR register = “1”) is issued.

[Note] Each flag of FLXAnFRNDAT1 to FLXAnFRNDAT4 register is automatically cleared by the transferring of the data part from MessageRAM to OBF. Even if only the header part of the frame is transferred to OBF, the reception completion flags in the FLXAnFRNDAT1 to FLXAnFRNDAT4 registers are not cleared.

■ Read Timing of Receive Message

Received messages can be reliably read by waiting the worst time calculated by using the following formula from the beginning of the next received slot.

Formula:



PLmax : Maximum payload length (word)

Ceil(x) : Minimum integer value exceeds x

The following shows the calculation example of the waiting time.

(1) Payload is 16 words (32bytes) and Peripheral bus clock is 80MHz

Waiting time = 2 x {17 + 2 x ceil(16 / 2)} x {1 / (80 x 106)}
= 2 x (17 + 2 x 8) x {1 / (80 x 106)}
= 82.5 [nsec]

(2) Payload is 128 words (254 bytes) and Peripheral bus clock is 80MHz

Waiting time = 2 x {17 + 2 x ceil(128 / 2)} x {1 / (80 x 106)}
= 2 x (17 + 2 x 64) x {1 / (80 x 106)}
= 3.6 [usec]

5. Interrupt

5.1 Interrupt Control

Table 5-1 shows FlexRay related interrupt of RH850/P1x. Refer to RH850/P1x hardware manual: “5. Interrupt” for the interrupt details.

Table 5-1 FlexRay Related Interrupt

Interrupt Source	EIINT Interrupt Channel Number	Offset Address (Table Reference Method)
FlexRay 0 interrupt	722	+5A4H
FlexRay 1 interrupt	723	+5A6H
Timer 0 interrupt	724	+5A8H
Timer 1 interrupt	725	+5AAH
Timer 2 interrupt	726	+5ACH
FIFO transfer interrupt	727	+5AEH
FIFO transfer caution interrupt	728	+560H
Output transfer caution interrupt	729	+562H
Output transfer end interrupt	730	+564H
Input que full-interrupt	731	+566H
Input que empty-interrupt	732	+568H

5.2 FlexRay0 Interrupt and FlexRay1 Interrupt

FlexRay0 and FlexRay1 interrupt can be issued the interrupt by the following conditions.

- Frame transmit/receive completion
- Detected communication error
- Each status changing
- Timer counter reached to specified value
- Transmission completion between IBF/OBF ⇔ Message RAM

Each interrupt can be set by enable/disable by FLXAnFREIES (Set)/FLXAnFREIER(Reset) register and FLXAnFRSIES (Set)/FLXAnFRSIER (Reset) register. The interrupt is enabled by setting the target bit in the set register to “1”, and the interrupt is disabled by setting the target bit in the reset register to “1” (“0” writing is not affected to the register value).

The status of each interrupt is reflected to FLXAnFREIR register and FLXAnFRSIR register regardless of whether interrupts are enabled or disabled. When each bit in the status register is cleared, set “1” to the target bit. (“0” writing is not affected to the register value.

Each interrupt can be selected whether to use the FlexRay0 interrupt or FlexRay1 interrupt using the interrupt output selection register (FLXAnFREILS/FLXAnFRSILS).

Table 5-2 shows the interrupt setting example. Figure 5-1 shows the control procedure example when using the channel A error detection interrupt.

Table 5-2 Interrupt, Request Flag and Assign Example

Interrupt Name	Request Flag	Interrupt Assign
Receive interrupt	FLXAnFRSIR_RXI	FlexRay1 interrupt
POC error mode change	FLXAnFREIR_PEMC	FlexRay0 interrupt
Channel A error detection	FLXAnFREIR_EDA	FlexRay0 interrupt
Channel B error detection	FLXAnFREIR_EDB	FlexRay0 interrupt

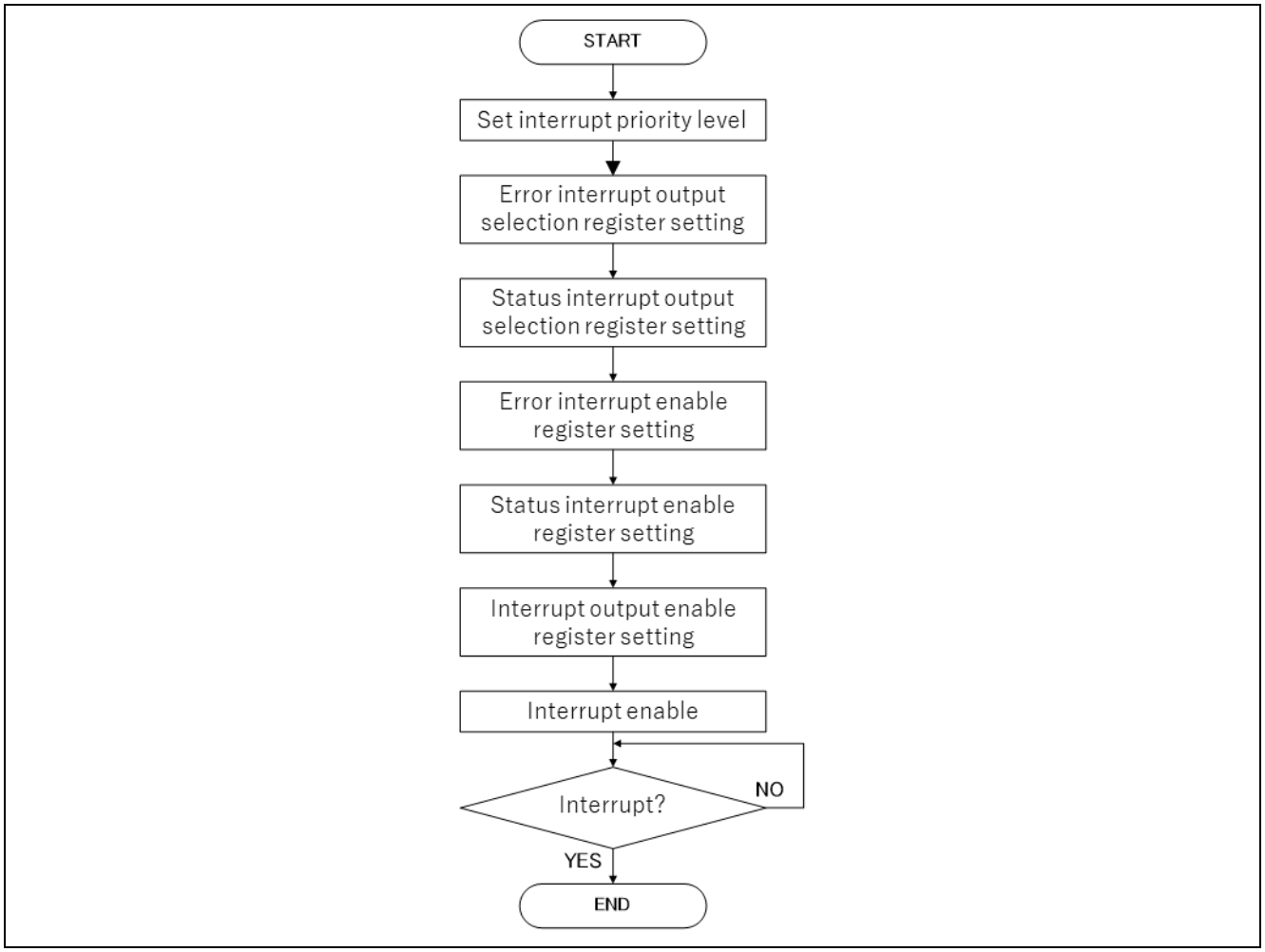


Figure 5-1 Control Procedure Example of Channel A Error Detection Example

[Note] When using the receive interrupt, set “1” to MBI bit in the target message buffer.

5.3 Interrupt Request Clear

In FlexRay module, the interrupt request is cleared by writing “1” to each bit in the interrupt status register (FLXAnFREIR and FLXAnFRSIR).

When clearing the interrupt request, write the value that is only the relevant bit set to “1” (immediate value) to the target register. If the interrupt request bit is cleared using a bit field structure or logical operation, other interrupt requests may be cleared by mistake.

6. Timer

FlexRay module is built-in the following two types of timers.

- Timer 0
- Timer 1

6.1 Timer 0

Absolute Timer. Set the interrupt issuing timing by the offset value (MT value) by the cycle count value and the cycle start. Timer 0 starts by setting “1” to T0RC in FLXAnFRT0C register. The interrupt issuing timing is set with the T0CC bit and T0MO in FLXAnFRT0C register.

【注】 1. When setting the timer value to FLXAnFRT0C register, previously set “0” to T0RC bit and stop the timer T0.
2. Timer 0 is possible to operate in Normal_Active state. When transitioning to another state, timer 0 stops.

Table 6-1 shows the timer 0 setting example. Figure 6-1 shows the control procedure example.

Table 6-1 Timer 0 Setting Example

Name	Setting	Note
Cycle counter value	Each 2 communication cycle	Occurred interrupt each even communication cycle.
MT offset value	1,000 MT	After 1,000MT from communication cycle head
Timer operation mode	Continuous mode	Occurred interrupt request every time the set conditions are met.

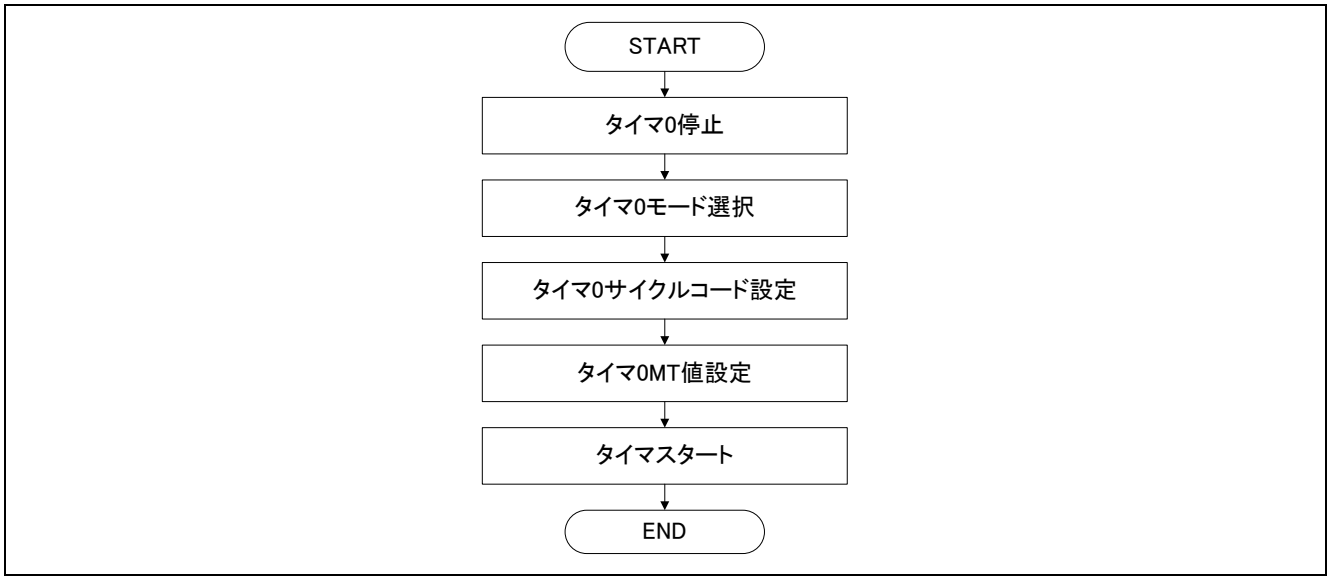


Figure 6-1 Control Procedure Example

6.2 Timer 1

Relative value. Set the interrupt issuing timing by the passed time (MT value) from timer starting. Timer 1 starts by setting “1” to T1RC in FLXAnFRT1C register. The interrupt issuing timing is set with the T1MC bit in FLXAnFRT1C register.

[Note] 1. When setting the timer value to FLXAnFRT1C register, previously set “0” to T1RC bit and stop the timer 1.
2. Timer 1 is possible to operate in NORMAL_ACTIVE state. When transitioning to another state, timer 1 stops.

Table 6-2 shows the timer 1 setting example. Figure 6-2shows the control procedure example.

Table 6-2 Timer 1 Setting Example

Name	Setting	Note
MT count	After 100 MT	
Timer operation mode	Single-shot mode	Stop by timer 1 interrupt occurrence

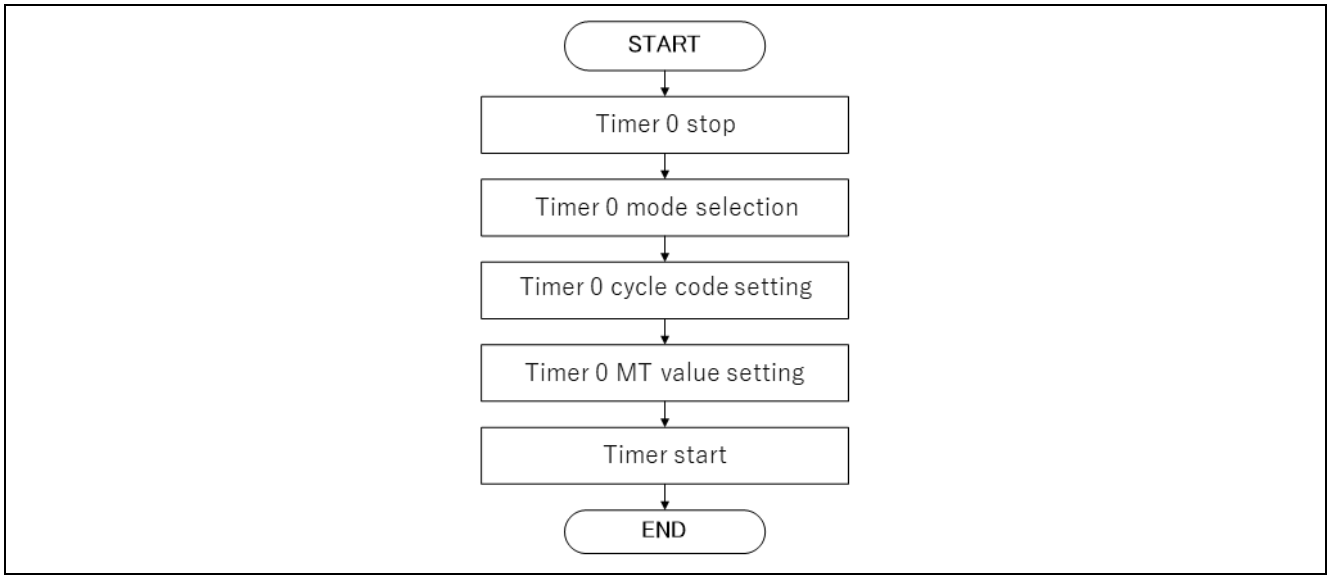


Figure 6-2 Control Procedure Example

7. Stopwatch Timer

The stop capture timer captures the cycle counter value, slot counter value, and MT value, and stores then to the following register when the trigger event is occurred.

- Cycle counter value : SCCV0 to SCCV5 bit in FLXAnFRSTPW1 register
- MT value : SMTV0 to SMTV13 bit in FLXAnFRSTPW1 register
- Channel A slot counter value : SSCVA0 to SSCVA10 bit in FLXAnFRSTPW2 register
- Channel B slot counter value : SSCVB0 to SSCVB10 bit in FLXAnFRSTPW2 register

The trigger event of the stopwatch timer is the following.

- FlexRay 0 interrupt request issuing
- FlexRay 1 interrupt request issuing
- Software trigger (SWWT bit in FLXAnFRSTPW1 = “1”)

[Note] 1. FlexRay0/1 interrupt trigger and software trigger are not possible to enable at the same time.
2. Each value is captured by the stopwatch trigger when the trigger occurs and at the start of the next MT after the trigger occurs.

Table 7-1 shows the setting example of the stopwatch trigger. Figure 7-1 shows the control procedure example.

Table 7-1 Setting Example of Stopwatch Trigger

Name	Setting	Note
FlexRay0/1 interrupt trigger	Disable	
Stopwatch operation mode	Single-shot mode	Stopped by stopwatch interrupt occurrence
Software trigger	Enable	

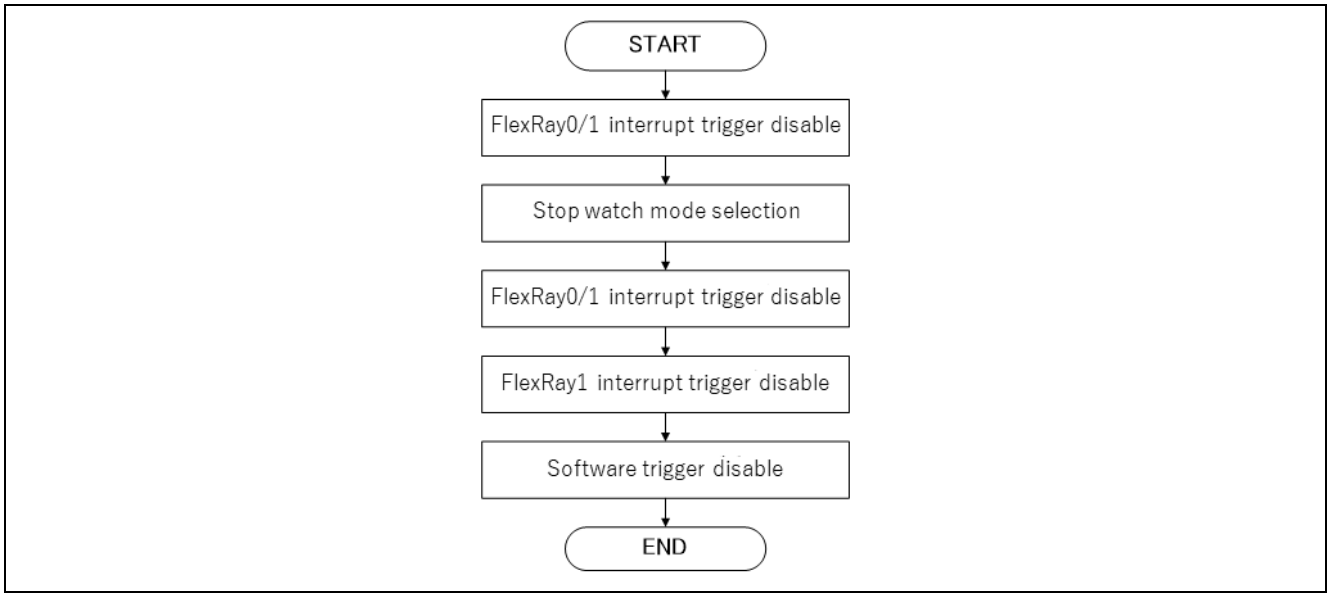


Figure 7-1 Control Procedure Example

8. Network Control Function

In FlexRay module, it has a built-in network management function that extracts the beginning part of the frame whose payload preamble indicator (PPI) flag is set to "1" from among the frames received in the static segment as the "NM vector."

The communication controller performs the bitwise OR (logical sum) of all NM vectors received during one communication cycle and saves it as NM vector information. Compares the NM vector information from the previous communication cycle and generates an interrupt request if the values differ.

[Note] NM vector information (FLXAnFRNMV1 to 3 register) is updated when communication cycle is updated.

Table 8-1 shows the network control function setting example.

Table 8-1 Network Control Function Setting Example

Name	Setting	Note
Network control function	Use	Set unused in sample program.
NM vector length	4 bytes	Set area of 0 to 12 bytes and 1 byte unit

Macro definition values in Node_1.h file if required : Change NEMC_SET.

9. Receive FIFO

FlexRay module is built-in the receive FIFO function.

When configurating the FIFO buffer to MessageRAM, it is possible to set the start buffer number to FFB0 to FFB7 bit in FRMRC register and the end buffer number to LCB0 to LCB7 bit in FLXAnFRMRC.

FIFO buffer is possible to configure up to 128. Refer to “1. Initial Setting” for the configuration method of MessageRAM.

When the receive data is not matched with the filtering conditions of the static buffer and dynamic buffer, and matched with the filtering of FIFO buffer, it stored to FIFO buffer. In this case, MBS bit in the target message buffer is rewritten by the frame ID, the payload length, the receive cycle counter, and the status of the receive frame.

9.1 FIFO Filtering

The filtering of FIFO buffer is configured by the channel filter, the frame ID filter, and the cycle counter filter.

9.1.1 Configuration for FIFO Rejection Filter

Set the reject conditions for storing received data in the FIFO buffer or not in the FLXAnFRFRF and FLXAnFRFRFM registers.

When RSS bit in FLXAnFRFRF register is “1”, all received message in Static segment is rejected by FIFO.

When RNF bit in FLXAnFRFRF register is “1”, the received Null frame is not stored to FIFO buffer.

When the null frame is not rejected by the FIFO rejection filter, it is stored to FIFO buffer same with the other data. In this time, the target bit in NDAT register becomes “1”.

Set the rejection filtering condition of the frame ID by FID0 to FID10 bit in FLXAnFRFRF register. Whether to apply the filtering conditions of each bit set in the FID0 to FID10 bits of the FLXAnFRFRF register by the MFID0 to MFID10 bits of the FLXAnFRFRFM register, and determine the reject frame ID to actually be used. When “1” is set to MFID0 to MFID10 bit, corresponding MFID0 to MFID10 bit setting is ignored.

Table 9-1 to Table 9-5 show the frame ID reject filter setting example.

Table 9-1 Reject Filter Setting Example 1

Bit Name	Setting Value
FLXAnFRFRF register FID bit	000 0000 0011b
FLXAnFRFRFM register MFID bit	000 0000 0000b
Reject frame ID	000 0000 0011b

[Explanation]
Frame ID = 3 is rejected.

Table 9-2 Reject Filter Setting Example 2

Bit Name	Setting Value
FRF register FID bit	000 0000 0011b
FLXAnFRFRFM register MFID bit	000 0000 0001b
Reject frame ID	000 0000 001Xb (X is optional)

[Explanation]
Frame ID2 and Frame ID3 are rejected.

Table 9-3 Reject Filter Setting Example 3

Bit Name	Setting Value
FLXAnFRFRF register FID bit	000 0000 1000b
FLXAnFRFRFM register MFID bit	000 0000 0111b
Reject frame ID	000 0000 1XXXb (X is optional)

[Explanation]
Frame ID8 to Frame ID15 are rejected.

Table 9-4 Reject Filter Setting Example 4

Bit Name	Setting Value
FLXAnFRFRF register FID bit	000 0000 1000b
FLXAnFRFRFM register MFID bit	000 0000 0100b
Reject frame ID	000 0000 1X00b (X is optional)

[Explanation]
Frame ID8 and Frame ID12 are rejected.

Table 9-5 Reject Filter Setting Example 5

Bit Name	Setting Value
FLXAnFRFRF register FID bit	000 0000 1000b
FLXAnFRFRFM register MFID bit	000 0001 1111b
Reject frame ID	000 000X XXXXb (X is optional)

[Explanation]
Frame ID1 and Frame ID31 are rejected.

9.2 Reding for FIFO Buffer

For reading the stored frame to the received FIFO, perform “FIFO header buffer contents reading processing” to the OBF buffer.

For staring the transfer trigger from FIFO buffer to OBF buffer without DEFAULT_CONFIG or COFIG state, set the first buffer number of FIFO buffer to OBRS0 to OBRS6 bit in FLXAnFROBCR register. This writing becomes the transfer request.

The first buffer number of FIFO buffer can be referred by FFB0 to FFB7 bit in FLXAnFRMRC register.

For reading the data from FIFO buffer, perform the following sequences. Refer to “3.3. OBF (Output Buffer) : Reading from MessageRAM” for the details.

Table 9-6 shows the reading setting example from FIFO buffer. Figure 9-1 shows the control procedure example.

Table 9-6 Setting Example

Name	Setting	Note
Header part transferring	Perform	
Data part transferring	Perform	
FIFO buffer header number	8	

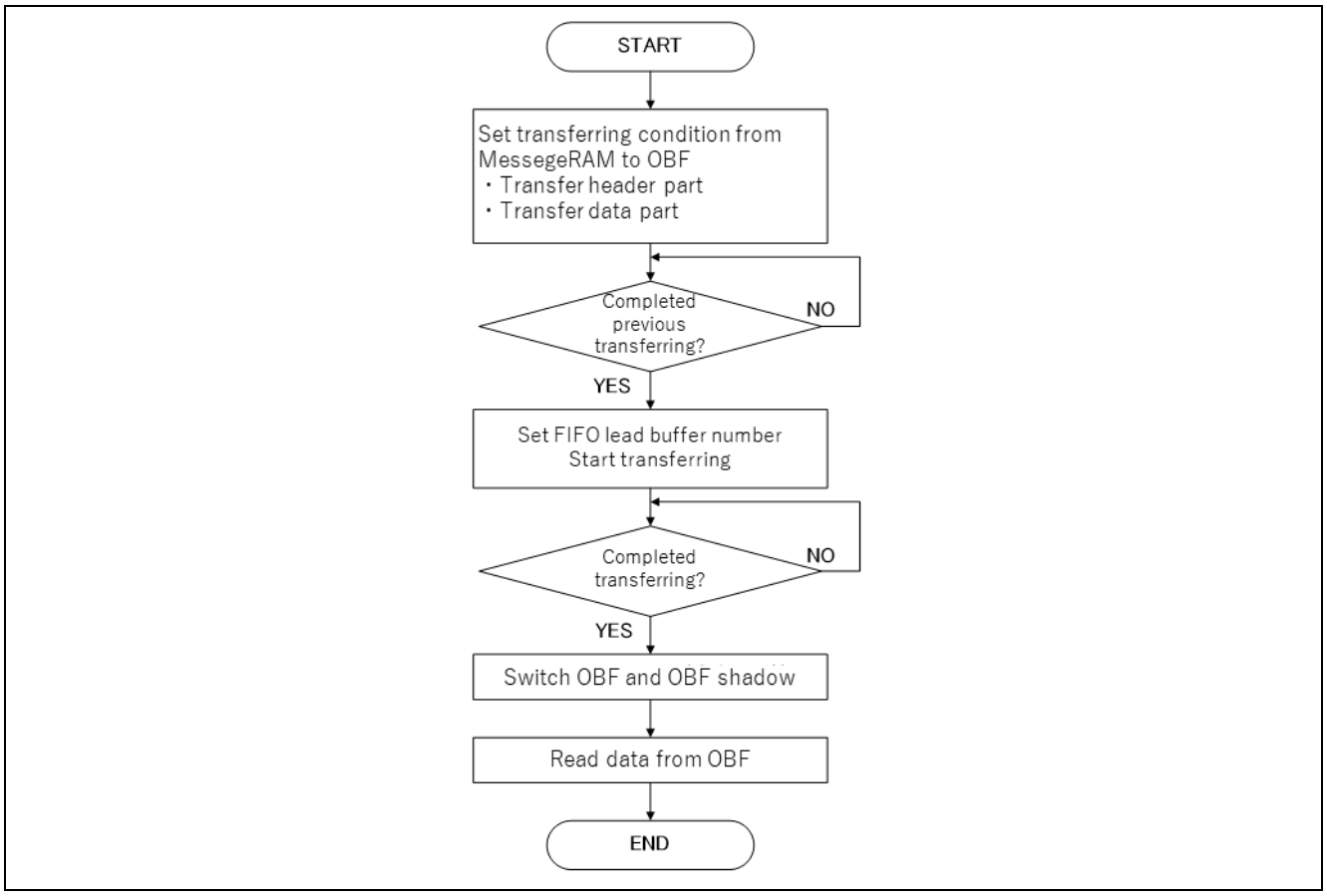


Figure 9-1 Control Procedure Example

[Note] 1. Do not perform the reading process for FIFO buffers other than "FIFO first buffer".
2. Do not perform the empty-reding FIFO buffer.
When perform empty-reding, EFA (Empty FIFOAccess) flag in FLXAnFREIR register is set.
Please perform the reding process after configuring that data is stored in the FIFO by FLXAnFRSIR register RFNE (Receive FIFO Not Empty), etc.
3. When reading the data from FIFO buffer, FLXAnFRRDHS1 register only holds the frame ID (FID bit) in the receive frame. All other bits become “0” when read.

4. When transferring the data part from FIFO buffer to OBF buffer (RDSS bit in FLXAnFRWRHS2 register = “1”), the data of the set payload length (PLC0 to PLC6 bit in FLXAnFRWRHS2 register and PLC0 to PLC6 in FLXAnFRRDHS2 register) is transferred.
   The payload length of the received frame (PLR0 to PLR6 in FLXAnFRRDHS2 register) is not transferred.
   When the message is stored to FIFO buffer, the received payload (PLR0 to PLR6 in FLXAnFRRDHS2 register) and the set payload length (PLC0 to PLC6 bit in FLXAnFRWRHS2 register and PLC0～PLC6 bit in FLXAnFRRDHS2 register) is the following.
   - PLR[6:0] > PLC[6:0] :
     The stored payload data to the message buffer is cut to the set Payload length. It is the PLC+1 length.
   - PLR[6:0] ≤ PLC[6:0] :
     The received payload data is stored to the data part in the message buffer. The data in the data part is filled by the undefined value up to the number of bytes indicated by the PLC.
   - PLR[6:0] = “0” :
     The data part in the message buffer is filled by the undefined value.
   - PLC[6:0] = “0” :
     The message buffer does not configure the data part. The data is not stored to the data part in the message buffer.

9.3 Writing to FIFO Buffer

For the writing of FLXAnFRWRHS1 to FLXAnFRWRHS3 register to FIFO buffer, it is same control with “3. Data Setting to MessageRAM & Data Read from MessageRAM”.

When using the message buffer as FIFO buffer, the receive conditions in the FLXAnFRWRHS1 register are ignored and the receive conditions in the FLXAnFRFRF and FLXAnFRFRFM registers are used.

Figure 9-2 shows the writing control procedure example to FIFO buffer that is allocated to the message buffer 2.

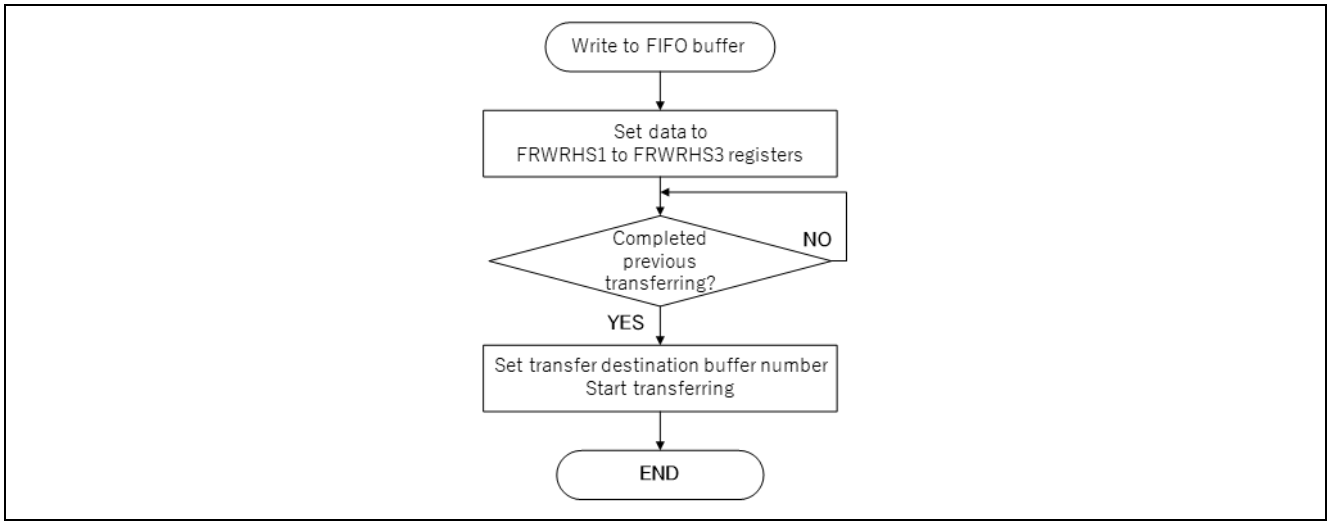


Figure 9-2 Control Setting Procedure

[Note] Make FIFO buffers the same data length. (PLC0～PLC6 bit in FLXAnFRWRHS2 register)

Our Company’s Website and Inquiry

Website

http://japan.renesas.com/

Inquiry

http://japan.renesas.com/contact/

Revision History

Rev.	Date	Description	
		Page	Point
1.00	2024.3.8	-	-

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.

Notice

1. 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating the safety of the final products or systems 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall 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. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

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/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.