M16C/62 Group Application Note <Simple I^2C Bus> (Tentative)

Mitsubishi Electric Corporation, Kitaitami Works
Mitsubishi Electric Semiconductor Systems Corporation
Mitsubishi Electric System LSI Design Corporation
REV.B

Safety Considerations

Mitsubishi has made every effort to improve the quality and reliability of its products. Nevertheless, a small percentage of semiconductor products can malfunction. Please incorporate redundancy features, fire countermeasures, malfunction prevention features, and other safety measures to prevent bodily injury, fire, or inconvenience to the public.s

Using this Document

This document is intended to serve as a reference to permit the appropriate use of the Mitsubishi semiconductor products for customer applications. It does not represent consent for the implementation of intellectual property or other rights by Mitsubishi Electric or third-parties.

Mitsubishi Electric takes no responsibility for damage or infringement of third-party rights arising from the use of the data, figures, tables, or circuit charts presented in this document.

The data, figures, tables, and other information are current as of the publication of this document; this information is subject to change without notice as product features are improved by Mitsubishi Electric. Please have your company's technical specialists contact Mitsubishi Electric or your retailer as needed for information about using the products presented here.

Mitsubishi Denki semiconductors are not been designed or manufactured for use in equipment upon which safety and human life depend. Please contact Mitsubishi Denki or your retailer for information regarding transportation, moving vehicle, medical, aerospace, nuclear power control, or sea-floor communications devices and systems, or other special applications.

Written permission must be obtained from Mitsubishi Electric or Mitsubishi Semiconductor Systems prior to excerpting or duplicating this documentation.

Export permits based on all foreign exchange and overseas trade control laws applicable to strategic products and materials described herein must be obtained prior to export.

Please contact Mitsubishi Electric or your retailer with any inquiries or comments you may have about this document.

Foreword

The "M16C/62 Series Application Notes, “Simple I^2C Bus Mode" are reference materials intended to help you control the simple I^2C bus mode which is contained in the Mitsubishi CMOS 16 bit microcomputer M16C/62 series. These materials are not intended to serve as a guarantee of the communications operations of the I^2C bus. As the user, you are responsible for performing an adequate performance evaluation.

Please use these materials along with the "M16C/60 Series Software Manual" for information about the M16C/62 series command system. Refer to the user manuals appropriate for each of the hardware devices you are using, and see the operating descriptions for information about development support tools.

Summary

The M16C/62 Series has a serial I/O circuit (UART2) which is provided with a simple I^2C bus circuit. Using a simple I^2C bus circuit in combination with software enables control of the PC bus interface. This Application Note outlines the I^2C bus specifications and introduces the various functions and programs that are used to enable the I^2C bus interface with the simple I^2C bus function.

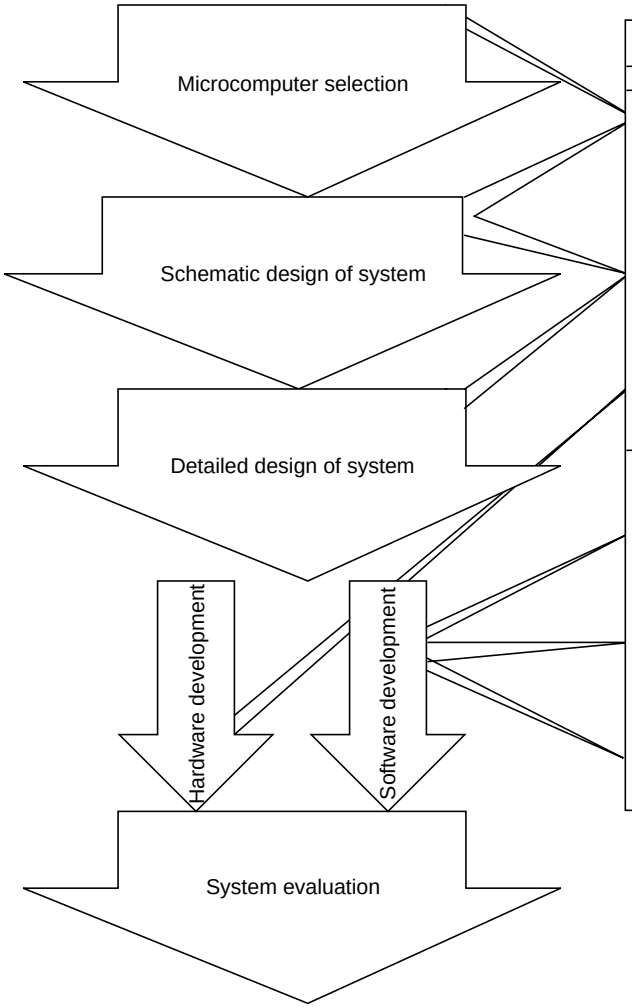
To use this document, you need a fundamental knowledge about electrical circuits, logic circuits, and microcomputers.

The document consists of four chapters. You can refer to the appropriate chapters/sections for information responding to the following needs:

I need to learn about ...	Refer to ...
I^2C bus protocol	Chapter 1, "Summary of I^2C Bus Mode Specifications"
M16C/62 serial I/O configuration	Chapter 2, "M16C/62 UART2 Functions" Section 2.1
M16C/62 simple I^2C bus block schematic and the register configuration	Chapter 2, "M16C/62 UART Functions" Sections 2.2~2.4
How to use functions of M16C/62 simple I^2C bus mode	Chapter 3, “Functions of the Simple I^2C Bus Mode”
Precautions regarding use of M16C/62 simple I^2C bus mode	Chapter 4, "Precautions Concerning the Simple I^2C Bus Mode”
Refer to the I^2C bus mode interface program used by M16C/62	The reference program

List of Documents for the M16C Family

Applications (Microcomputer development flow)



Type of Document		Contents
Hardware	Datasheets/ Databooks	Hardware specifications (pin assignment, memory maps, specification of peripheral functions, electrical properties, timing)
	Users' manual	Details of hardware specifications, operations, application examples (connections with peripherals, relationship with software)
Software	Programming manual	Assembly language, programming with C-language
	Software manual	Details of each command operation (assembly language)
	Reference program collection	Assembly language reference program collection

Chapter 1. Summary of I²C Bus Mode Specifications

	Chapter 1 Section Title
1.1	I²C Bus Mode Features
1.2	II²C Bus Mode Concept
1.3	Data Transmission
1.4	Communication Coordination
1.5	Definition of First Byte
1.6	Standard Mode and High-Speed Mode
1.7	I²C Bus SDA and SCL Bus Line Characteristics

The I²C bus is a multi-master bus communications inter-IC control protocol developed by Philips that is now used by many IC's. Refer to information published by Philips for detailed I²C bus specifications.

1.1 I²C Bus Mode Features

I²C bus mode provides efficient control between IC's. It is a two-way bus line with a simple two-wire structure consisting of a serial data line (SDA) and a serial clock line (SCL). I²C bus mode also includes the following features:

Each of the devices connected to the bus has its own individual address. A simple master[1] and slave[2] relationship is always established. The master device functions as either a transmitting device or reception device; the slave device functions as either a slave transmitting device or reception device.

Collision detection[3] and communications coordination procedures[4] are incorporated into the device to prevent data destruction in the event that several masters attempt to start data transmission simultaneously, permitting multi-master[5] operations.

Two-way serial data transfer is enabled in high-speed mode.

Note 1 Devices that start data transmission, generate clock signals, and end data transmission.
Note 2 Devices that have addresses designated by the master.
Note 3 A function that detects data transmitting at levels other than its own when more than one master device transmits data.
Note 4 A procedure that enables bus control by only one master device when multiple masters attempt to control simultaneously, ensuring that messages are not lost and contents not changed.
Note 5 This feature allows several masters to control the bus simultaneously without losing messages.

1.2 I^2C Bus Mode Concept

With two wires (SDA and SCL), the I^2C bus performs data transmission among devices connected to the bus. Each piece of equipment is recognized by its individual address and operated as either a transmitting or receiving device, according to the functions of the equipment. Master devices are those devices that transmit clock signals as they transmit data; slave devices are those devices that designate addresses in response to one or several masters.

When master devices transfer data, the master transmits a clock signal to obtain the data transmission timing and designates the slave address (transmission). The transmitting device then transmits data to the receiving device; data transmission is ended by the master. A clock signal is always generated by the master with the I^2C bus, so each master generates its own clock signal whenever data exchanges take place at the bus. If the clock signal generated by the master is a for a low-speed slave device that maintains the SCL at "low" , then the bus may change to another master during communications coordination procedures. (Refer to 1.4, "Coordinating Procedures Using SCL Synchronization" and "SCL Synchronization".)

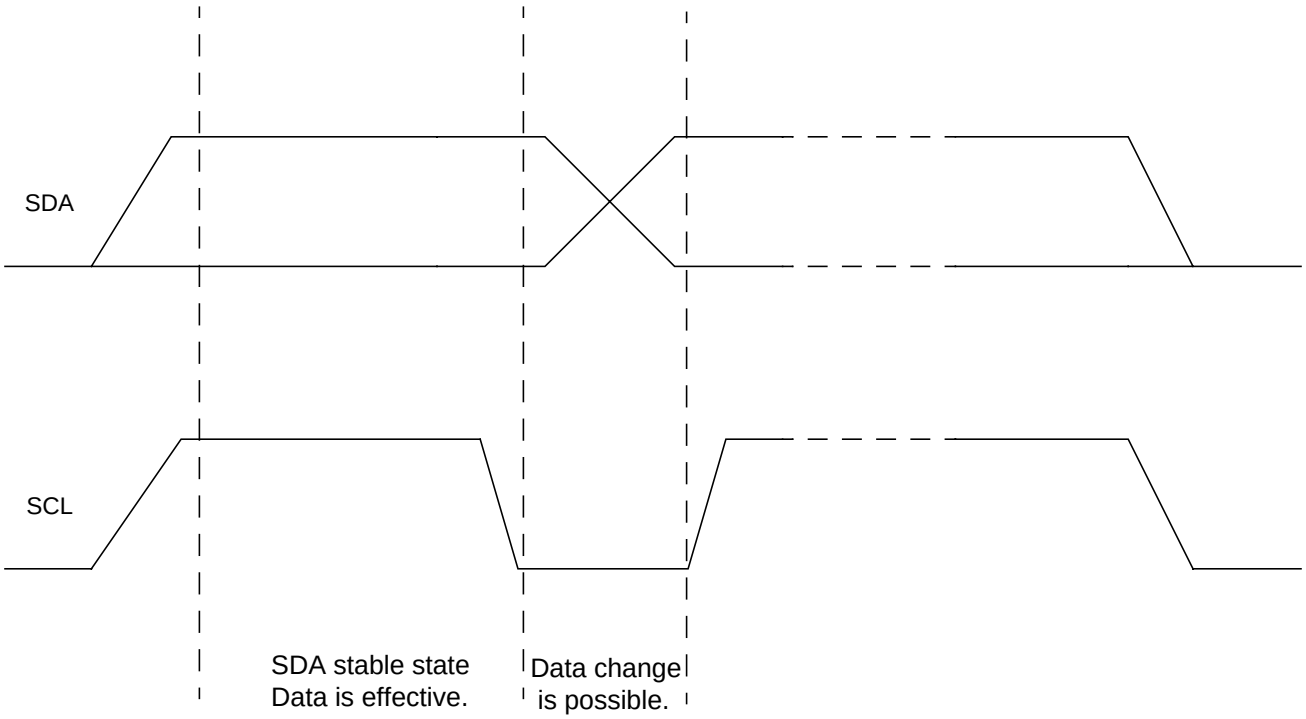
Both the SDA and SCL are two-way lines that are connected to the positive power source through parallel resistance. When the bus is free, both lines are put into "high" status. Since the output levels of the devices connected to the bus execute AND connection functions, either an open drain or an open collector are needed. *

* The SDA and SCL output terminals of the M16C/62 are N-channel open drain output.

1.3 Data Transmission

The I^2C bus data transmission format is defined as follows:

Data effectiveness - SDA status between the SCL highs must remain constant.
Changes in SDA level are limited to when the SCL is low.



1.3 Data Transmission

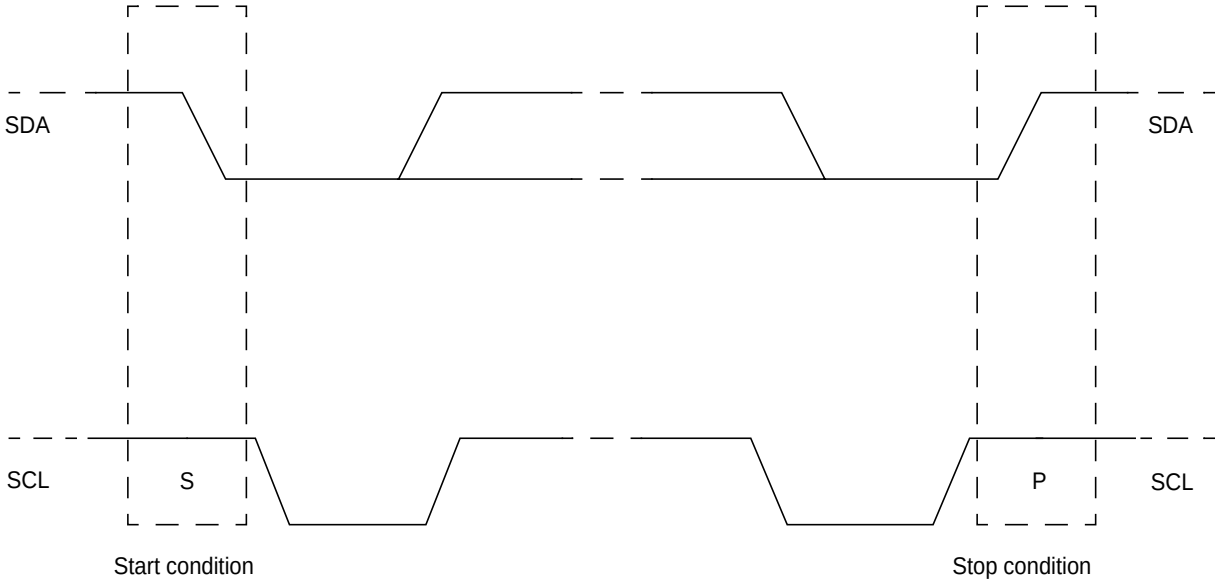
Start and Stop Conditions

In I^2C bus communications procedures, data transmission begins with the transmitting of a start condition by the master; data transmission stops with the transmitting of a stop condition.

When the SCL is high, the situation where the SDA changes from high to low is called a start condition.[1] The situation where the SDA changes from low to high is called a stop condition.[2]

Other than start and stop conditions, the SDA level does not change when the SCL is high.

Both start and stop conditions are always generated by the master. The bus goes into busy status after a start condition is generated. The bus goes into free status after a stop condition is generated.[3]



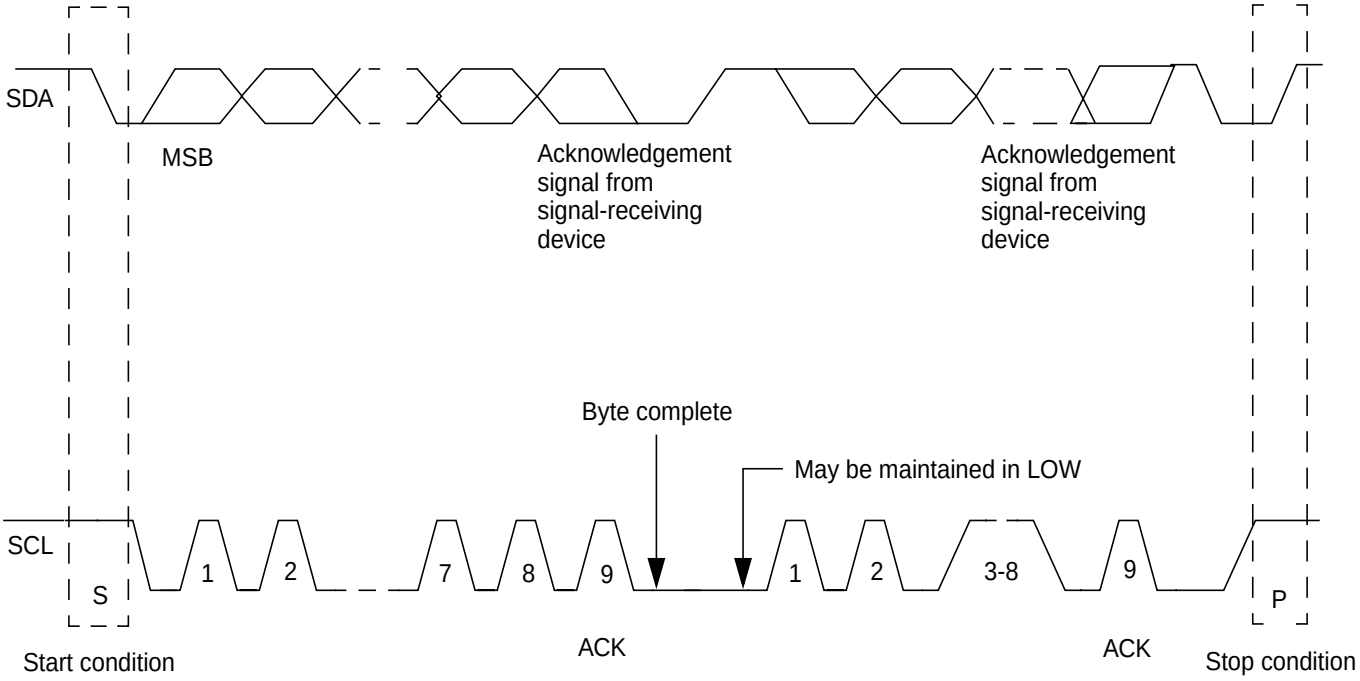
1 The M16C/62 simple I^2C bus mode has a “start condition detection interrupt” designed to detect start conditions. (Refer to Section 3.2, “Start/Stop Condition Detection”.)

2 The M16C/62 simple I^2C bus mode has a “stop condition detection interrupt” designed to detect stop conditions. (Refer to Section 3.2, “Start/Stop Condition Detection”.)

3 The M16C/62 simple I^2C bus mode has a “bus busy” flag indicating the condition of the bus. (Refer to Section 3.2, “Bus Busy Detection”.)

Byte Format

The length of each byte of the data output by the SDA must be eight bits. There is no limit to the number of bytes that can be sent in one transmission (from the time a start condition is generated to the time a stop condition is generated. Any number of bytes can be sent continuously. Data is sent in sequence from the uppermost bit (MSB) and an acknowledge bit is attached after each byte (the ninth bit).

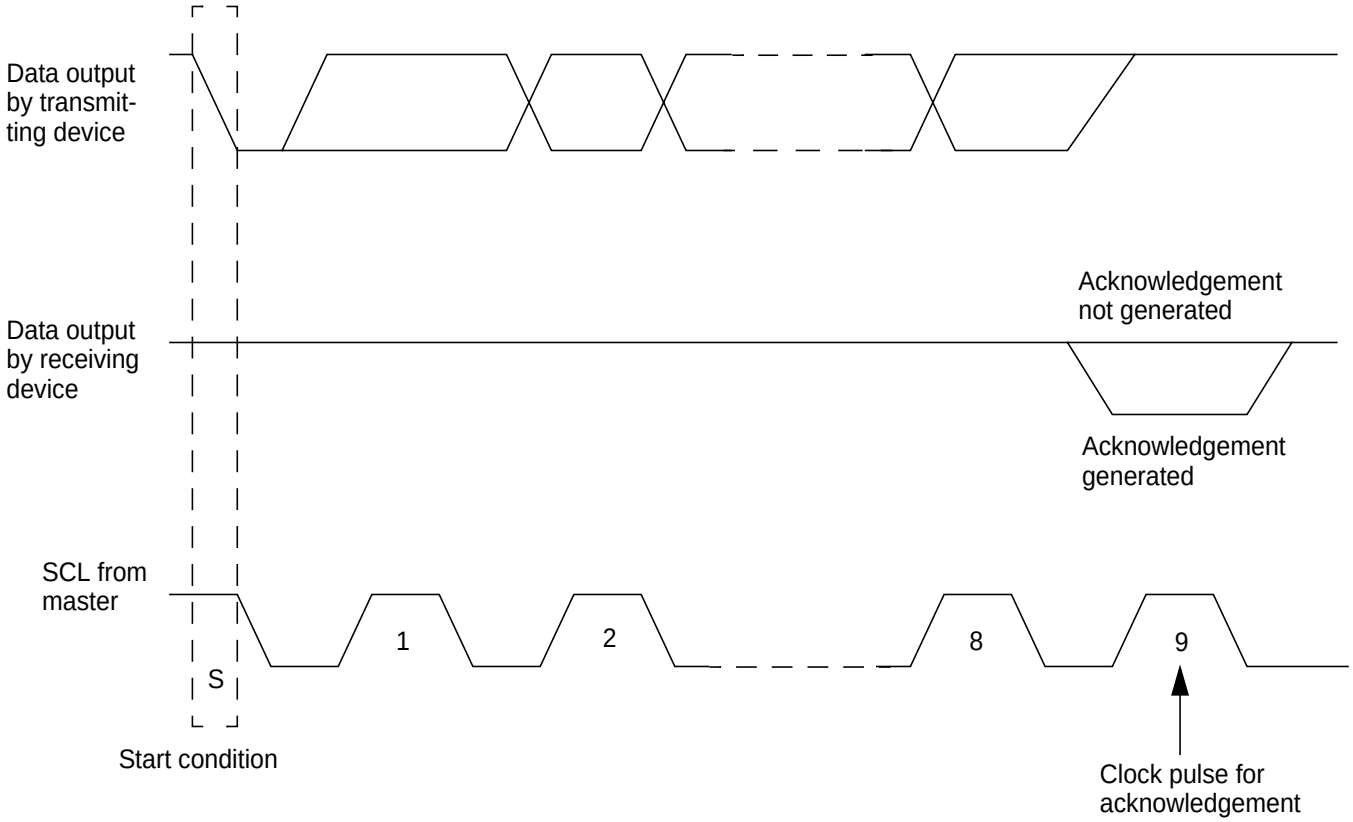


Acknowledge

Acknowledgements are needed in data transmission. The clock signal for acknowledgements are generated by the master in the SCL. When the clock signal for acknowledgement (the ninth bit) is generated, the SDA is released on the transmitting device end (high status). When the acknowledgement clock signal rises at each receiving device, an acknowledgement signal is generated because the SDA remains in a stable low status.[1]

Receiving devices that have had addresses designated must ordinarily generate acknowledgements whenever the completion of each byte is received. If the slave receiver device does not confirm the address, the slave maintains the SDA in high status when the acknowledgement signal rises and does not generate an acknowledgement signal.2 At this time, the master transmits a stop condition and data transmission can be halted. When data cannot be received during a transmission, even if the slave receiving device has confirmed the address, the condition is indicated by the fact that an acknowledgement is not generated. At this time, the slave maintains the SDA in high status and the master generates a stop condition.

When the master becomes a receiving device, it informs the slave transmitting device of the end of the data by not generating an acknowledgement for the last data byte transmitted from the slave. At this time, the slave transmitting device releases the SDA and the master is able to generate a stop condition.,



[1] The M16C/62 simple I^{2}C bus mode has an “acknowledgement detected” interrupt to detect acknowledgement generation conditions. (Refer to Section 3.3, “Acknowledgement Detection”.)

[2] The M16C/62 simple I^{2}C bus mode has an “acknowledgement not detected” interrupt to detect when acknowledgements have not been generated. (Refer to Section 3.3, “Acknowledgement Detection”.)

1.4 Communication Coordination

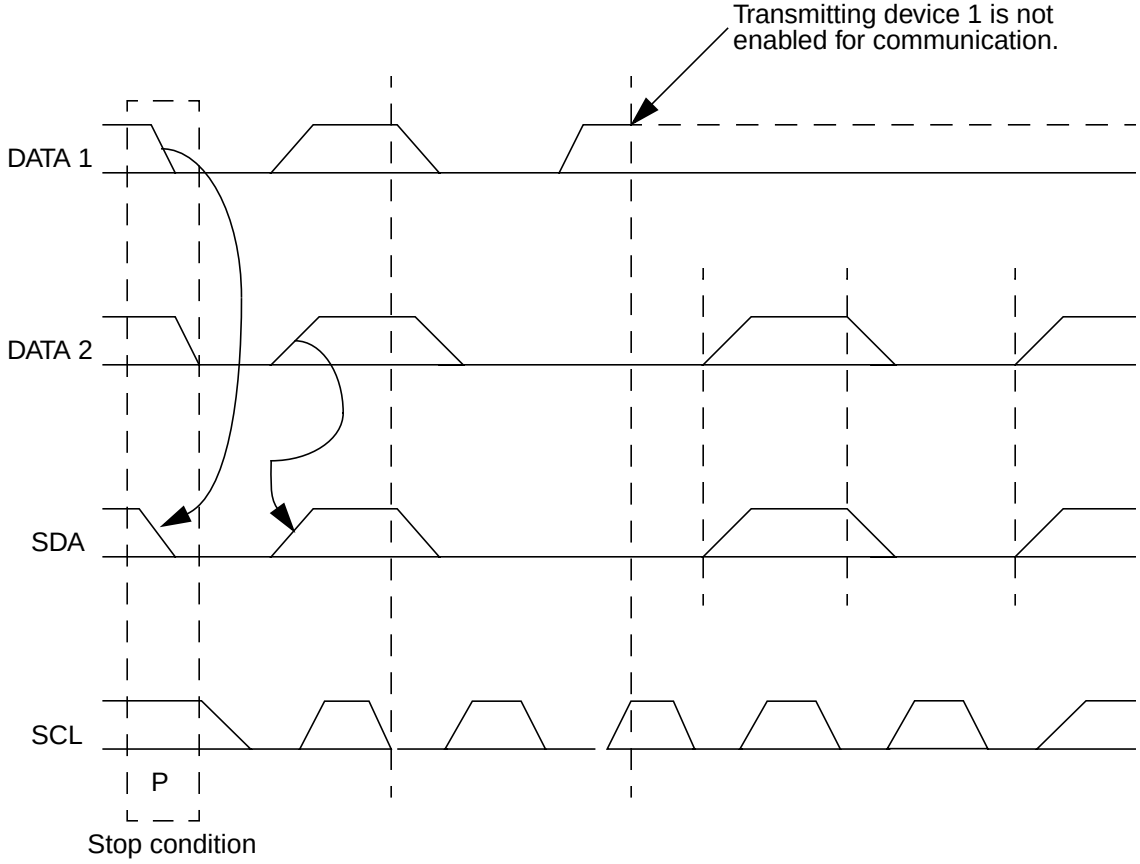
Communication is performed according to the following procedures for communications among several devices connected through one I^2C bus.

Communication Enabling Procedures

The master can only start data transmission when the bus is in a free condition. However, since the I^2C is a multi-master bus, there are situations in which two or more masters generate a start condition at precisely the same time. Communication enabling procedures exist to prevent confusion when this occurs. These procedures are performed by using the SDA and the AND feature of each device's open drain or the open collector output terminal.

When the SCL is at a high level, masters that are transmitting to the SDA at a high level turn the data output level off, determining that communcation is not enabled when a low SDA level (when an arbitration loss occurs) is detected (since other masters are transmitting with the SDA at a low level.)[1] In this way, communication enabling procedures are executed through the SDA if several masters attempt to transmit their own data. Simultaneous transmissions limit the master that can actually transmit data to another device. The masters that do not receive enabling are immediately switched over to slave signal-receiving mode if they have slave capabilities. The clock signal can be generated until the end of the byte that made the determination not to grant communication enabling is shut off.

The diagram below shows the communication enabling procedures that take place between two masters. When the internal data level and actual SDA levels are different in the master-generating DATA1, the master's data output is turned off. Thus the data transmission that was started by the master that was given communication enabling is not affected in any way. These procdures are used only when many masters start at the same time; the procedures do not set up priority masters or an order of priority.



[1] The M16C/62 simple I^2C bus mode has a function that detects disparities between internal data levels and the SDA levels timed to the SCL rise (refer to Section 3.5, "Arbitration Loss Detection Function") and a function for turning output level output off (refer to Section 3.5, "SDA Output Prohibition Function when Arbitration Loss Occurs").

SCL Synchronization

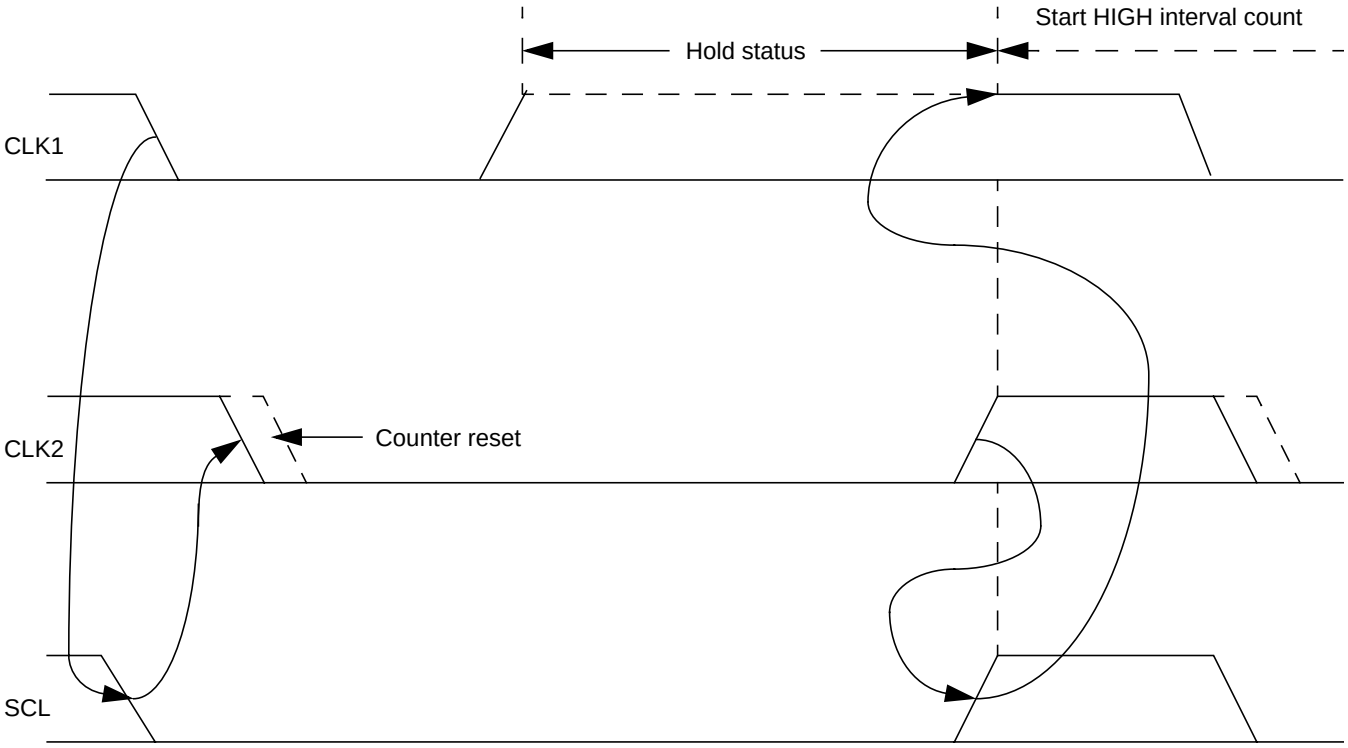
The M30622 chip does not have a register for I^2C and does not operate in multimaster environments.

The SCL has procedures to enable communication when several masters attempt to start data transmission at the same time. A clock signal is output until the last byte for masters where communication is not enabled and when arbitration is lost, but among the masters that are each generating their own clock signals occurring at the bus.

Synchronization of the different clock signals can be performed using the SCL and the AND feature of the open drain output or open collector output terminals of each device. If the clock signal of a device is low, the low interval count is started while the SCL is maintained in low. Even if the low interval of the device ends and the clock signal goes from low to high, the SCL does not change if the clock signals of other devices remain in the low interval. The duration of the low interval is actually determined by the device with the longest low interval. During this interval, the clock signal remains high for devices with the shorter low interval (remaining in high-impedance state).

When all the devices finish the low interval, the clock signal is released and the system goes into high interval. The clock signals output by the devices and the SCL are now in the same status and counting begins for the high intervals of each device. The SCL reverts to low status according to the device that first finishes its high interval.

The low interval is determined by the device that has the longest low interval, while the high interval is determined by the device with the shortest high interval, thereby synchronizing the SCL.



[1] The M16C/62 simple I^2C bus mode has an “SCL Synchronization” function to obtain SCL synchronization. (Refer to Section 3.5, “SCL Synchronization Function.”)

1.5 Definition of First Byte

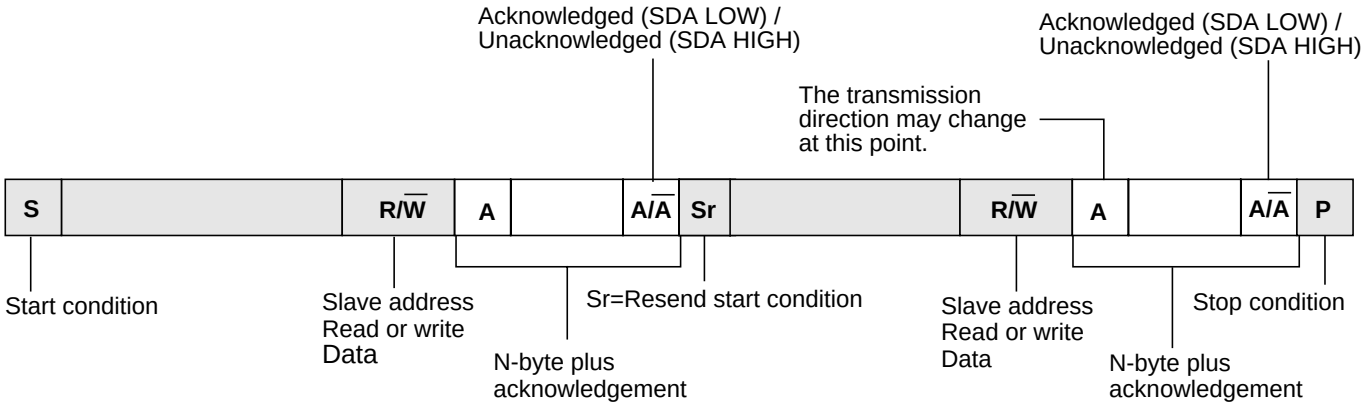
In the I^2C bus, the first byte sent after a start condition contains important data which designates the slave. The following is the definition of the first byte.

7-bit Address Format

.

In the I^2C bus, the address with which the master selects the slave is determined by the first byte continuing from the start condition. The following description is an explanation of the seven-bit address format.

The schematic below shows the data transmission format. The slave address is sent after the start condition (S). This address consists of seven bits, with the eighth bit constituting the data direction bit (R/W). Transmitting to the slave is indicated if this data direction bit is 0; a data request to the slave is indicated if the bit is 1. However, a retransmit start condition (SR) is generated without first generating a stop condition and another address can be designated with the next byte if the master wants to continue communication at the bus. In this case, the slave address is sent with the seventh bit after the retransmit start condition, with the eighth bit serving as the data direction bit (R/W).



1.5 Definition of First Byte

Definitions of Each Bit of the First Byte

After the start condition, the first seven bits of the first byte show the slave address. The eighth bit is the data direction bit (R/W), the bit which determines the direction of the message.

When the address is sent, and after the start condition, each device in the system compares the first seven bits with their own individual addresses. When the addresses agree, the device understands that it has been designated as a slave by the master and puts the SDA line in low in time with the nineth bit clock pulse, returns an acknowledgement signal, and performs data reception (data direction bit "W") or transmitting (data direction bit "R").

* The M16C/62 simple I^2C bus mode has a function that puts the master into standby mode until the address determination processing and acknowledgement transmitting can be completed.

1.6 Standard Mode and High-Speed Mode

You can set 7-bit addresses in the standard mode of the I^2C bus at a maximum data transmitting rate of 100 kilobits/second. By increasing from standard to high-speed mode, 7-bit addresses as well as 10-bit addresses can be set with data transmitting rates of up to 400 kilobits/second. All new devices which have I^2C bus interfaces can handle high-speed mode.

* The M16C/62 simple I^2C bus mode also supports high-speed mode, with the exception of some restricted items. (Refer to Chapter 4, "Precautions Concerning the Simple I^2C Bus Mode"). Note,however,that address transmitting and receiving is performed by software. (Refer to "Example of Auto Address Determination".)

1.7 I^2C Bus SDA and SCL Bus Line Characteristics

The following table presents the I^2C bus standard and high-speed mode electrical characteristics and the SDA and SCA bus line definitions:

Electrical Characteristics

Parameters	Symbol	Standard Mode		High-Speed Mode		Units
		Min.	Max.	Min.	Max.	
Low level input voltage: When input level is constant When input level varies according to V_{DD}	V_{IL}	-0.5 -0.5	1.5 $0.3V_{DD}$	-0.5 -0.5	1.5 $0.3V_{DD}$	V
High level input voltage: When input level is constant When input level varies according to V_{DD}	V_{IH}	3.0 $0.7V_{DD}$	(1) (1)	3.0 $0.7V_{DD}$	(1) (1)	V
Input hysteresis is a Schmidt trigger: When input level is constant When input level varies according to V_{DD}	V_{hys}	n/a n/a	n/a n/a	0.2 $0.05V_{DD}$	-- --	V
Spike pulse width controlled by input filter	t_{SP}	n/a	n/a	0	50	ns
Low level output voltage (open drain or open collector: When sink current is 3mA When sink current is 6mA	 V_{OL1} V_{OL2}	 0 n/a	 0.4 n/a	 0 0	 0.4 0.6	V
Output fall time from V_{Ihmin} to V_{Ilmax} when the bus capacitance is from 10pF to 400pF (up to maximum 6mA through V_{OL2} parallel resistance): Maximum sink current 3mA at V_{OL1} Maximum sink current 6mA at V_{OL2}	t_{OF}	 -- n/a	 250 (2) n/a	 $20+0.1C_b$ (2) $20+0.1C_b$ (2)	 250 250 (3)	ns
Input current at each VO pin when input current is 0.4 V~ 0.9 V_{DDmax} .	Ii	-10	10	-10 (3)	10 (3)	µa
Capacitance of each VO pin	Ci	--	10	--	10	pF

n/a = not applicable

Note 1 Max $V_{IH} = V_{DD}$ " 0.5V.

Note 2 Cb = Capacitance (units: pF) of 1 bus line. The maximum t_F of the SDA and SCL bus lines (200 ns) greater than the maximum t_{OF} (250ns) at the output step.
Thus, a series protector resistor (R_S) can be connected between the SDA/SCL pins and the SDA/SCL bus line without exceeding the maximum rated I_F.

Note 3 There can be no interference by the I/O pin with the SDA and the SCL lines when the V_{DD} supply is cut.

Related Note: The electrical properties of the M16C/62 are different from the standards for the I^2C bus. (Refer to 4.1, "Electrical Characteristics".)

Timing Definitions

Parameters	Symbol	Standard I^2C Mode Bus		High-Speed Standard I^2C Mode Bus		Units
		Min.	Max.	Min.	Max.	
SCL clock frequency	f_{SCL}	0	100	0	400	KHz
Bus free time between start and stop conditions	t_{BUF}	4.7	--	1.3	--	µ secs
Hold time (retransmit) "start". Initial clock pulse generated after this interval.	t_{HDSTA}	4.0	--	0.6	--	µ secs
SCL clock "low"status hold time	t_{LOW}	4.7	--	1.3	--	µ secs
SCL clock "high" status hold time	t_{HIGH}	4.0	--	0.6	--	µ secs
Retransmit "start" condition setup time	t_{SU-STA}	4.7	--	0.6	--	µ secs
Data hold time: For the CBUS compatibility master (see 9.3, "Precautions") For the I^2C bus	t_{HD-DAT}	5.0 0 Note 1	-- --	-- 0 Note 1	-- 0.9 Note 2	µ secs µ secs
Data setup time	t_{SU-DAT}	250	--	100 Note 3	--	n secs
SDA and SCL signal rise times	t_R	--	1000	20 + 0.1 C_b Note 4	300	n secs
SDA and SCL signal fall times	t_F	--	300	20 + 0.1 C_b Note 4	300	n secs
"Stop" condition setup time	t_{SU-STD}	4.0	--	0.6	--	µ secs
Load capacity of each bus line	C_b	--	400	--	400	pF

The above values all correspond to $V_{IH\ max}$ or $V_{IL\ min}$ levels.

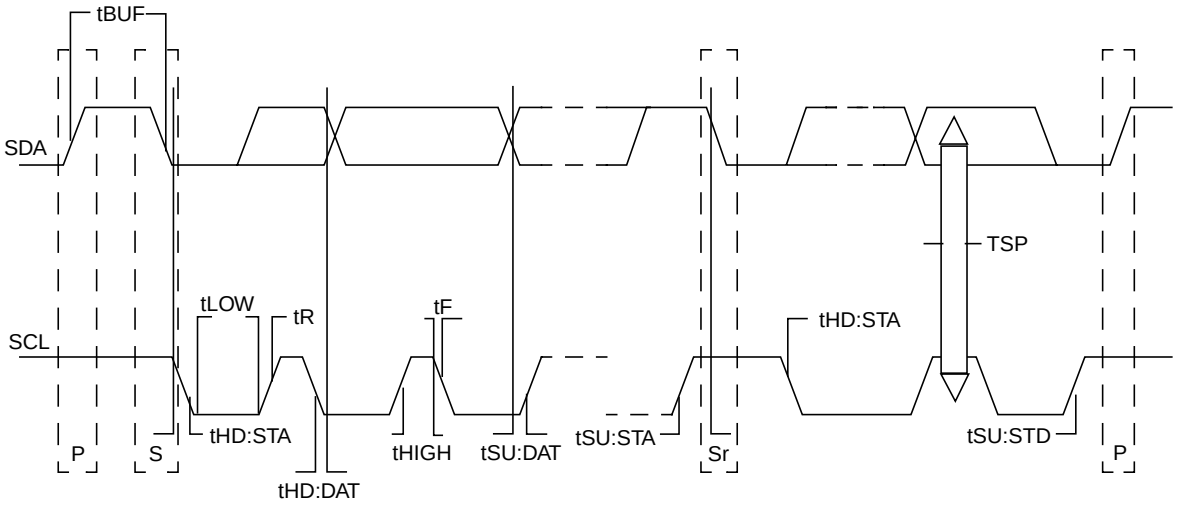
Note 1 To fill the undefined region of the SCL fall end, a 300-ns minimum hold time for the SDA signal (at SCL signal $V_{IH\ min}$) must be provided internally.

Note 2 When the device does not extend the SCL signal "low" (t_{LOW}) hold time, only the maximum data hold time $t_{HD:DAT}$ must be met.

Note 3 Although the high-speed mode of the I^2C bus can be used in the standard mode I^2C bus system, $t_{SU:DAT}$>250ns is a condition that must be satisfied. If the device does not extend the SCL signal's "low" (tLOW) hold time, the satisfaction of this condition is requested to be a non-condition. If the device does extend the SCL signal's "low status hold time, the following data bits need to be sent to the SDA line before $t_{Rmax.} + t_{su:dat} = 1000 + 250 = 1250$ nx, releasing the SCL line (according to specifications of standard I^2C mode bus.

Note 4 C_b = 1 bus line total capacitance (units: pF)

Note 5 The electrical properties of the M16C/62 are different from the standards for the I^2C bus. Refer to 4.1, "Electrical Characteristics".


tBUF
SDA
SCL
tLOW
tR
tF
tHD:STA
tHIGH
tSU:DAT
tSU:STA
tHD:STA
TSP
tSU:STD
P
S
tHD:DAT
Sr
P

Chapter 2. M16C/62 UART2 Functions

The M16C/62 serial I/O consists of five channels, UART0, UART1, UART2, as well as S I/O3 and S I/O4. Each of these channels has its own timer for generating dedicated transmitting clocks, and these timers function independently. In this chapter, we discuss detailed settings for the simple I^2C bus mode, which is one of the UART2.

2.1 Functions of the UART2

With the exception of some different functions, the functions of the UART0~ UART2 channels are essentially the same. Of these channels, UART2 in particular makes use of modes which can handle a SIM interface, an IE bus interface, and an I^2C bus interface.

UART0~ UART2 make selective use of either a clock synchronization-type serial I/O mode or a clock non-synchronization-type serial I/O mode.

The clock non-synchronization-type serial I/O mode in particular can handle an IE bus interface and a SIM interface. The M16C/62 has a serial data switching function and a bus collision detection function in order to enable IE bus interface and SIM interface. See the M16C/62 data sheets for detailed information about these functions.

Further, the UART2 clock synchronization-type serial I/O mode makes it possible to handle the I^2C bus interface.

In this chapter, we present the M16C/62 simple I^2C bus mode block diagrams and each type of relevant register. In the next chapter, we discuss the functions needed to make the I^2C bus interface work with the M16C/62.

The IE bus interface, SIM interface, and I^2C bus interface are the functions which are limited to UART2.

M16C/62 serial I/O configuration:

UART0	Clock synchronization-type serial I/O
	Clock non-synchronization-type serial I/O
UART1	Clock synchronization-type serial I/O
	Clock non-synchronization-type serial I/O
UART2	Clock synchronization-type serial I/O
	For the I^2C bus interface

Clock non-synchronization-type serial I/O

	For the IE bus interface
	For the SIM interface
S I/O3	Clock synchronization-type serial I/O
S I/O4	Clock non-synchronization-type serial I/O

2.2 Block Diagram of the Simple I^2C Bus Mode

The M16C/62 simple I^2C bus mode is used to enable the I^2C bus interface. The simple I^2C bus mode effectuates the circuit for the I^2C bus interface by setting the I^2C selector bit (IICM) to "1."

The following is the I^2C bus mode block diagram.

2.3 Terminal Functions That Change in I²C Bus Mode and Interrupt Causes

The M16C/62 simple I²C bus mode is put into effect by setting the I²C mode selector bit (IICM). The following tables show each function when the simple I²C bus mode is selected.

Terminal Functions

	(IICM) =1 (Simple I²C bus mode)	(IICM) = 0 (Common S I/O mode)
P7_0 terminal function	SDA (input/output)	TxD2 (output)
P7_0 output initial value	Set value in P7_0 when the serial I/O is not valid	H level
P7_1 terminal function	SCL (input/output)	RxD1 (input)
P7_1 terminal read	The terminal reads, regardless of the directional register	In accordance with the directional register settings
P7_2 terminal function	Port P7_2	CLK2

Main Interrupt Causes

	(IICM) = 1 (Simple I²C bus mode)		(IICM) = 0 (Common S I/O mode)
	(IICM2) = 0	(IICM2) = 0	
Cause: interrupt no. 10	Start/stop condition detected		Bus collision detected
Cause: interrupt no. 15	Acknowledge not detected	UART2 transmission	UART2 transmission
Cause: interrupt no. 16	Acknowledge detected	UART2 reception	UART2 reception
DMA1 cause when the DMA request factor selector bit = "1101"	Acknowledge detected	UART2 reception	UART2 reception

Precautions concerning bit processing commands for the port:
When the input/output port data register (port latch), is rewritten using bit processing commands, the bits which do not have values designated for them may change.

Reason:
Bit processing commands are read-modify-write commands which perform reading and writing in bit units. Therefore, when these commands are executed in relation to certain bits of the input/output ports of the data register, the following processing takes place in relation to the entire data register:

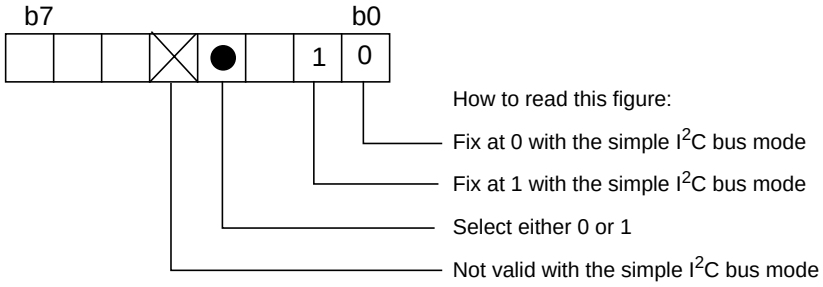
Bits set with input
The terminal value is read by the CPU, and this bit is read in after bit processing.
Bits for which values have been set
The data register bit values are read by the CPU, and these bits are written in after bit processing.

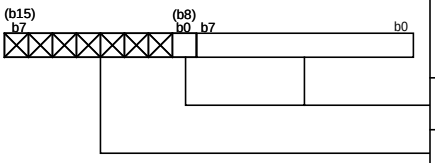
* Please note that the SCL and SDA output values may be changed when read-modify-write commands are executed for port P7.

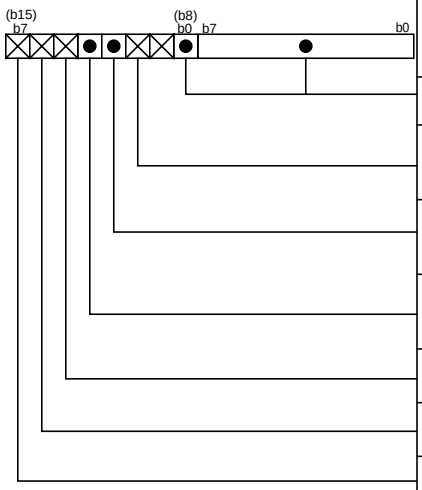
2.4 Register Setup When in Simple I²C Bus Mode

Registered are configured as follows when using the simple I²C bus mode in the M16C/62 UART2:

UART2-relevant register



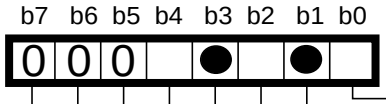
UART2 transmitting buffer register	Function	Values that can be set	R	W
	**Symbol =** U2TB **Address =** $037B_{16}$, $037E_{16}$ **At reset =** Undefined			
	Transmitting data (bit 8 is ACK)	00_{16} ~ FF_{16}	X	O
	Nothing located here. Put in 0 when writing. Value is undefined when reading.		--	--

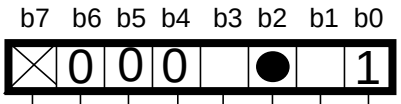
UART2 reception buffer register	Bit	Bit Name	Function	R	W
	**Symbol =** U2RB **Address =** $037F_{16}$, $037E_{16}$ **At reset =** Undefined				
	--	--	Reception data	O	X
	Nothing located here. Put in 0 when writing. Value is undefined when reading.			--	--
	ABT	Arbitration lost detection flag	0: Not detected (success) 1: Detected (defeat)	O	O
	OER	Overrun error flag	0: No overrun 1: Overrun	O	X
	FER	Framing error flag	Not valid in I²C bus mode	O	X
	PER	Parity error flag	Not valid in I²C bus mode	O	X
	SUM	Error sum flag	Not valid in I²C bus mode	O	X

UART2 transmitting speed register	Function	Values that can be set	R	W
b7 b0	**Symbol =** U2BRG **Address =** 0379_{16} **At reset =** Undefined			
	If the set value is n, BRG2 will n+1 divide the count source.	00_{16} ~ FF_{16}	X	O

UART2 transmit/receive mode register	Bit	Bit Name	Function	R	W
b7 b6 b5 b4 b3 b2 b1 b0 0 X X X 0 1 0	**Symbol =** U2MR **Address =** 0378_{16} **At reset =** 00_{16}				
	SMD0	Serial i/o mode selector bit	b2=0 b1=0 b0=0: serial I/O not valid (port control) b2=0 b1=1 b0=0: simple I^2C bus mode	O	O
	SMD1			O	O
	SMD2			O	O
	CKDIR	Internal clock selector bit	0: Internal clock (set at 0 when master) 1: External clock (set at 1 when slave)	O	O
	STPS	Stop bit length selector bit	Not valid in I^2C bus mode	O	O
	PRY	Parity odd/even selector bit	Not valid in I^2C bus mode	O	O
	PRYE	Parity authorization bit	Not valid in I^2C bus mode	O	O
	IOPOL	TxD, RxD input/output polarity switching bit	0: Reversal 1: No reversal (set at 0 when in I^2C mode)	O	O

UART2 transmit/receive control register 0	Bit	Bit Name	Function	R	W
b7 b6 b5 b4 b3 b2 b1 b0 1 0 X 1 ● X	**Symbol =** U2C0		**Address =** $037C_{16}$	**At reset =** 08_{16}	
	CLK0	BRG count source selector bit	b1=0 b0=0: f_1 selected b1=0 bo=1: f_8 selected b1=1 b0=0: f_{32} selected b1=1 b0=1: Use is not permitted.	O	O
	CLK1			O	O
	CRS	CTS/RTS function selector bit	Not valid when bit 4=1	O	O
	TXEPT	Transmitting register blank flag	0: Data in transmitting register 1: No data in transmitting register	O	X
	CRD	CTS/RTS prohibition bit	0: CTS/RTS function allowed 1: CTS/RTS function prohibited. (set at 1 when in I^2C mode)	O	O
	Nothing located here. Put in a 0 when writing. Value is 0 when reading.			--	--
	CKPOL	Clock polarity selection bit	0: Transmitting data output at fall of transmitting clock and reception data output at rise. 1: Transmitting data output at rise of transmitting clock and reception data output at fall.	O	O
	UFORM	Transmitting format selector bit	0: LSB format 1: MSB format	O	O

UART2 transmit/receive control register 1	Bit	Bit Name	Function	R	W
	**Symbol =** U2C1		**Address =** $037D_{16}$	**At reset =** 02_{16}	
	TE	Transmitting authorization bit	0: Transmitting denied 1: Transmitting authorized	O	O
	TI	Transmitting buffer open flag	0: Data in transmitting buffer register 1: No data in transmitting buffer register	O	X
	RE	Reception authorization bit	0: Reception denied 1: Reception authorized	O	O
	RI	Reception completed flag	0: Data in transmitting buffer register 1: No data in transmitting buffer register	O	X
	U2IRS	UART2 transmit/ receive interrupt cause selector bit (Note: U2IRS is not valid when (IICM)-1 and (IICM2).	0: Transmitting buffer open (TI=1) 1: Transmitting complete (TXEPT=1)	O	O
	U2RRM	UART2 continuous reception mode authorization bit	0: UART2 clock 1: 0 output	O	O
	U2LCH	Data logic selector bit	0: No reversal 1: Reversal (set at 0 in simple I^2C bus mode)	O	O
	U2ERE	Error signal output authorization bit	Set at 0 in simple I^2C bus mode.	O	O

UART2 special mode register	Bit	Bit Name	Function	R	W
	**Symbol =** U2SMR **Address =** 0377_{16} **At reset =** 00_{16}				
	IICM	I^2C mode selector bit (Note: Set serial I/O mode selector bit at 0 1 0 when in simple I^2C mode.)	0: Normal mode 1: Simple I^2C bus mode	O	O
	ABC	Arbitration lost detection flag control (Note: Set at 0 when (IICM) -1 and (IICM2).	0: Update for each bit 1: Update for each byte	O	O
	BBS	Bus busy flag (can be written as 0)	0: Stop condition detected 1: Start condition detected	O	O
	LSYN	SCLL synchronization output authorization bit (Note: Valid only when in port control; this bit is not valid when in serial I/O).	0: Denied 1: Authorized	O	O
	ABSCS	Bus collision detection sampling clock selector bit	Fix at 0.	O	O
	ACSE	Transmitting authorization bit and automatic clearing function selector bit	Fix at 0.	O	O
	SSS	Transmitting start condition selector bit	Fix at 0.	O	O
	Nothing located here. Put in a 0 when writing. Value is 0 when reading.			--	--

UART2 special mode register 2	Bit	Bit Name	Function	R	W	Chapter explaining bits
b7 b6 b5 b4 b3 b2 b1 b0	**Symbol =** U2SMR2		**Address =** 0376_{16}		**At reset =** 00_{16}	
	IICM2	I^2C mode selector bit	See Table 1.	O	O	2.3
	CSC	Clock synchronization bit	0: Denied 1: Authorized	O	O	3.5
	SWC	SCL wait output bit	0: Denied 1: Authorized	O	O	3.4
	ALS	SDA output stop bit	0: Denied 1: Authorized	O	O	3.5
	STAC	UART2 initialization bit	0: Denied 1: Authorized	O	O	3.6
	SWC2	SCL wait output bit 2	0: UART2 clock 1: 0 output	O	O	3.2
	SDHI	SDA output prohibition bit	0: Denied 1: Authorized (high impedance)	O	O	3.6
	SHTC	Start/stop condition control bit (Table 2)	1: This must be set to 1 when in I^2C mode	O	O	--

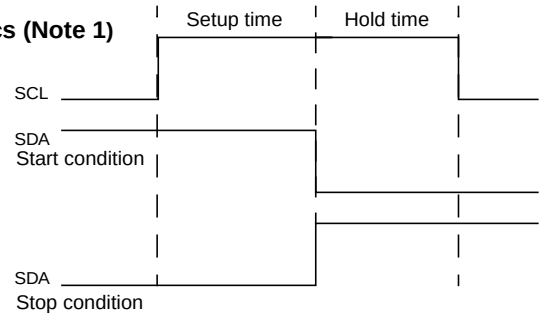
Table 1

	Function	IICM2 = 0	IICM2 = 1
1	Interrupt signal cause 15	No acknowledgement detected (NACK)	UART2 transmitting (rise of the last bit of the transmitting clock)
2	Interrupt signal cause 16	Acknowledgement detected (ACK)	UART2 reception (fall of the last bit of the reception clock)
3	DMA cause when DMA request cause selector bit = 1 1 0 1	Acknowledgement detected (ACK)	UART2 reception (fall of the last bit of the reception clock)
4	Timing of data transmitting from UART2 reception shift register to reception buffer	Rise of last bit of reception clock	Fall of last bit of reception clock
5	Timing of UART2 reception/acknowledgement detection interrupt request generation	Rise of last bit of reception clock (acknowledgement detected)	Fall of last bit of reception clock (UART2 reception)

Table 2. Start/stop condition detection timing characteristics (Note 1)

3~6 cycles < setup time (Note 2)
3~6 cycles < hold time (Note 2)

Note 1; When the start/stop condition control bit SHTC=1.
Note 2: The number of cycles indicates the main clock input oscillation frequency (XIN) number of cycles.



Chapter 3 Functions of the Simple I^2C Bus Mode

In this chapter, we discuss methods for using the hardware functions when the M16C/62 uses the simple I^2C bus mode to realize the I^2C bus interface.
I

	Chapter 3 Section Title
3.1	Byte-Data Transmission/Reception Methods
3.2	Start/Stop Conditions
3.3	Acknowledgement
3.4	Own-address Designation Determination
3.5	Communication Coordination
3.6	Other Functions

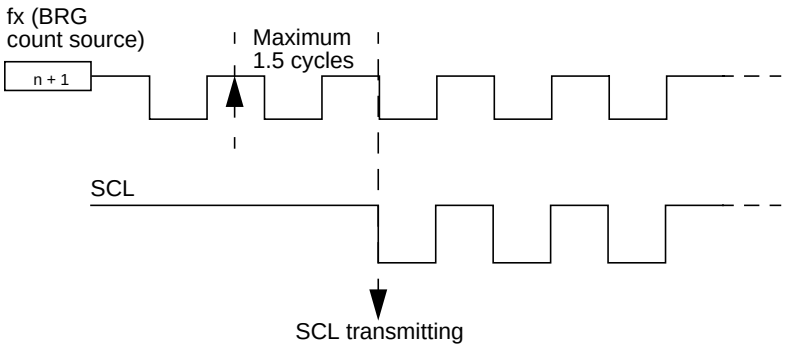
3.1 Byte-Data Transmission/Reception Methods

Here we present the simple I^2C bus mode SCL transmitting setup method where the M16C/62 is used as a master, as well as 1-byte transmitting and setup methods for reception.

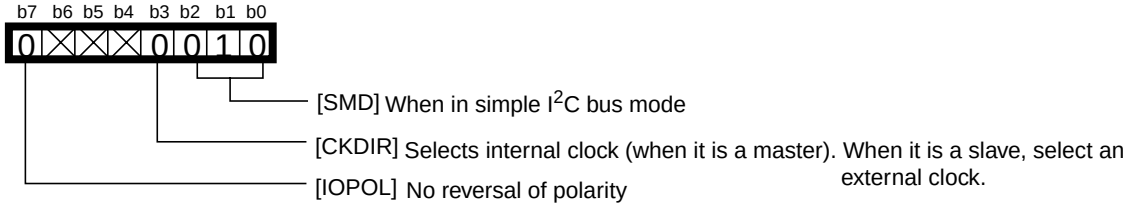
SCL Generation Method (as a master)

The sending clock (SCL) speed must be set up when using the M16C/62 as a master. The following registers are set up just as they would be for normal serial I/O transmitting. After the SCL writes to the transmitting data buffer, transmitting is performed at 1.5 cycles of the SCL.

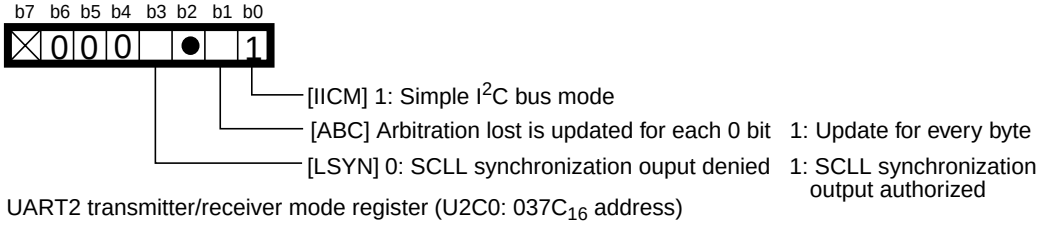
SCL transmitting timing



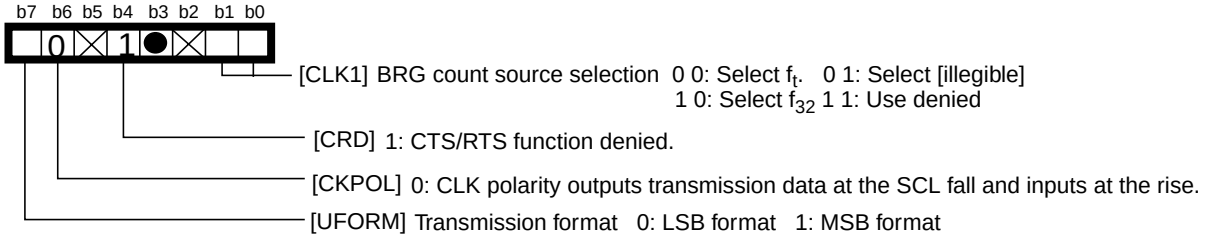
UART2 transmitter/receiver mode register (U2MR: 0378_{16} address)



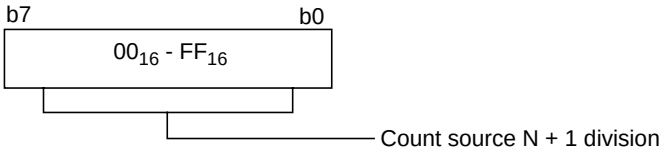
UART2 transmitter/receiver mode register (U2SMR: 0377_{16} address)



UART2 transmitter/receiver mode register (U2C0: $037C_{16}$ address)



UART2 transmission speed register (U2BRG: 0379_{16} address)



Setup example

When using a 1-MHz source oscillation and transmission speed of 100 Kbps:

U2MR = 00000010b (I^2C mode, internal clock selected)

U2C0 = 10010000b (BRG f1 count source selected)

U2BRG = 49

Settings when the device is a slave

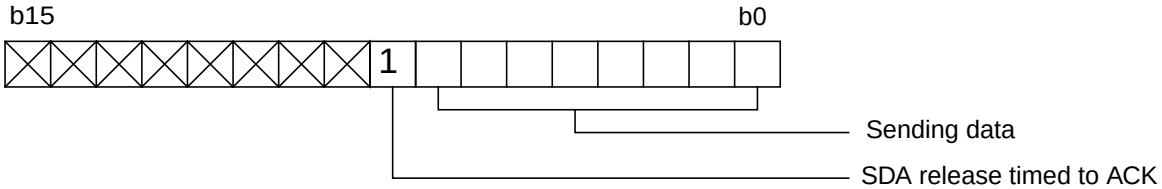
When using the device as a slave, set bit 3 (CKDIR) of the UART2 send/receive mode register (U2MR) at 1 and select an external clock. At this time, the BRG count source selector bit (CLK0) (CLK1) and the UART2 transmission speed register (U2BRG) settings are not valid.

3.1 Byte-Data Transmission/Reception Methods

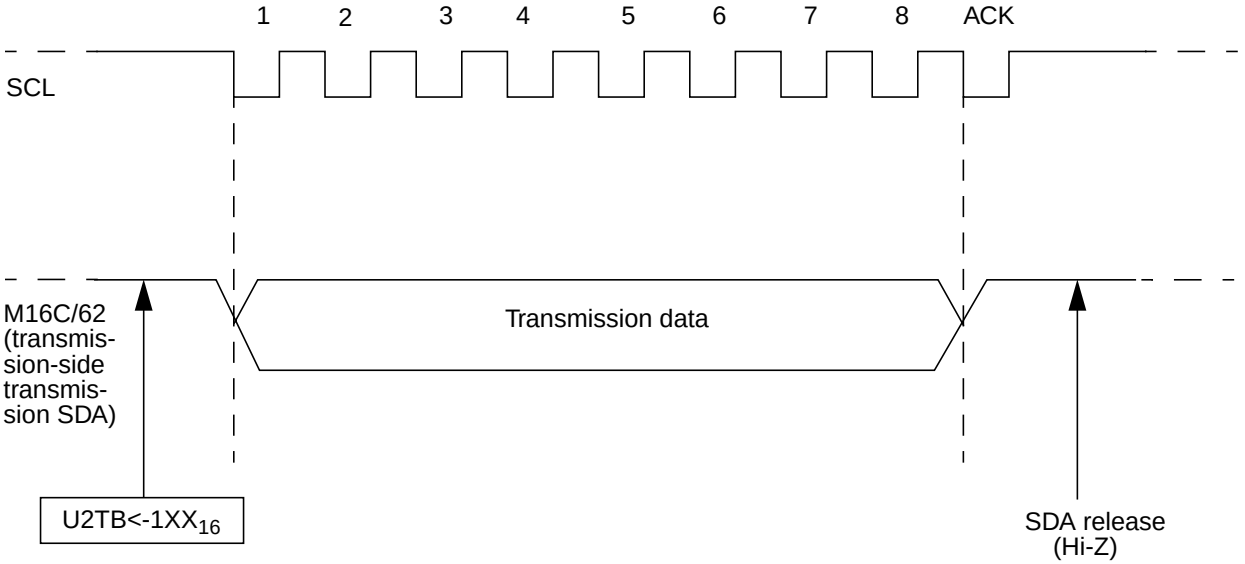
When the M16C/62 is used as a transmitting device, eight-bit transmission data is sent from the SDA terminal, but the SDA terminal must be free to receive the transmission clock acknowledgement at the ninth bit. This operation can be performed with data that sets up the transmission buffer.

Nine-bit data is used to set up the transmission buffer. In the I^2C bus, data is sent with MSB fast [*or "first"*]. In the M16C/62, the transsmission format is set to MSB fast and nine-bit data is sent in b7>b6>...>b0>b8 order. Since the uppermost bit is timed for transmission when the acknowledgement is received, and the SDA terminal is free and the uppermost bit is set at 1, the M16C/62's SDA is put into high-impedance status.

Relevant register



Timing pattern



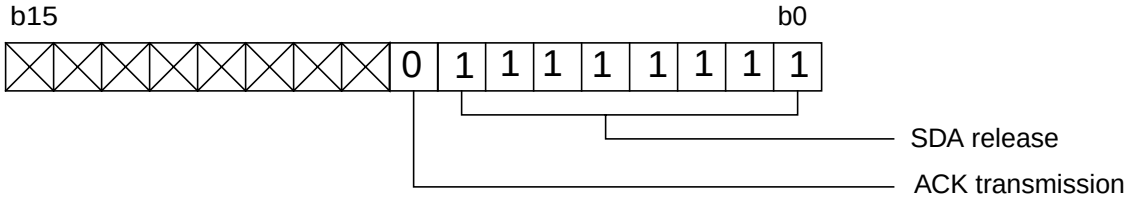
Byte Data Reception Methods

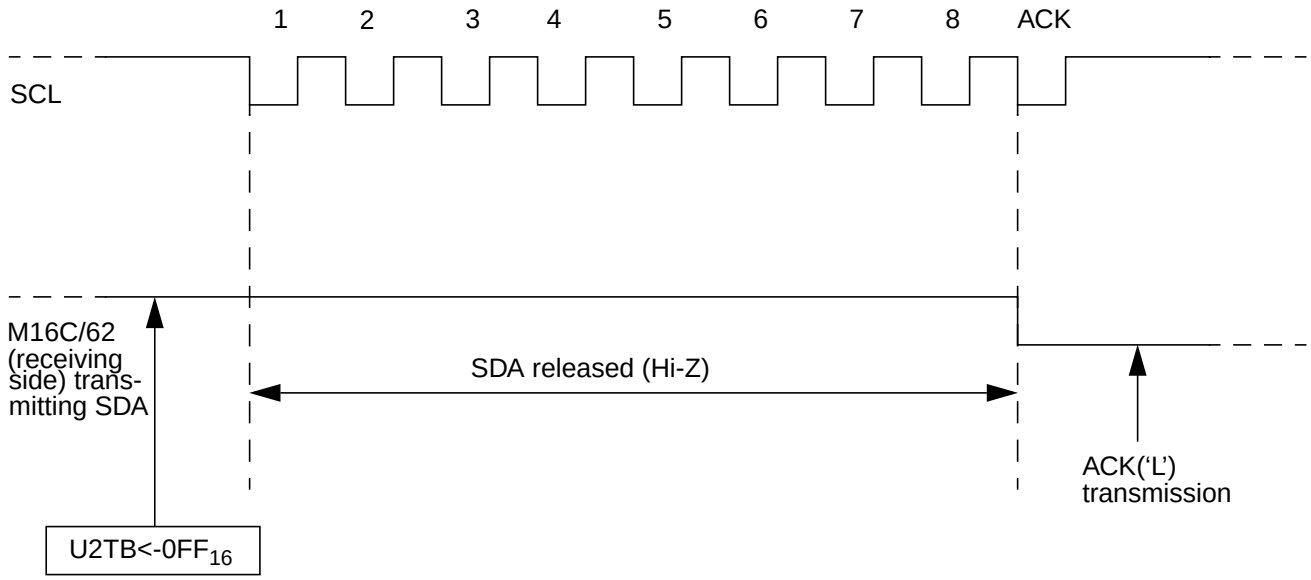
When the M16C/62 is being used as a receiving device, the M16C/62 SDA terminal needs to be released while receiving eight-bit data from the SDA terminal. Also, the SDA terminal must be put into L and generate an acknowledgement at the ninth bit of the clock. With this, address determination of the receiving device that has been designated by the master is completed. Acknowledgement transmissions can easily be executed at the value that has been written to the transmission buffer when it has been confirmed that transmission has taken place to this master-designated device.

Also, the nine-bit data in the transmission buffer is set as dummy data when the M16C/62 receives data. While the lower eight bits are sent, these bits are set at 1 because the SDA terminal is released. Moreover, the last bit transmitted (b8) is set at 0 to generate an acknowledgement. This indicates that byte data is received.

Relevant register

UART2 transmission buffer register (U2TB, $037B_{16}$, $037A_{16}$ address)





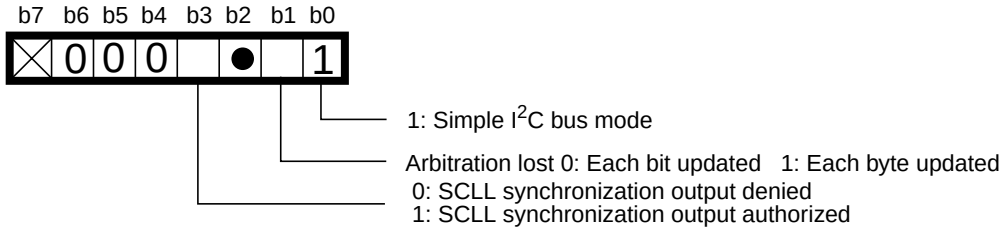
Transmission Interrupt/Reception Interrupt

The completion of data transmission can be detected with aUART2 transmission interrupt when the M16C/62 is used as a transmission device. Also, the completion of data reception can be detected with a UART2 reception interrupt when the M16C/62 is used as a receiving device. These interrupts are allocated to interrupt registers 15 and 16, and the interrupt causes UART2 transmission and reception, respectively, are determined by the I^2C mode selector bit 2 (IICM2) being set to 1. The timing of the generation of a transmission interrupt in this case is determined by the fall of the clock's start bit when the UART2 transmission interrupt cause selector bit (U2IRS) is set at 0. The generation of a transmission interrupt is timed to the rise of the (U2IRS) selector bit when it is set at 1. The timing of the generation of a reception interrupt is coordinated with the fall of the last bit of the reception clock. (See 2.4, "Register Setting when in Simple I^2C Bus Mode", UART2-relevant register (4), Table 1.)

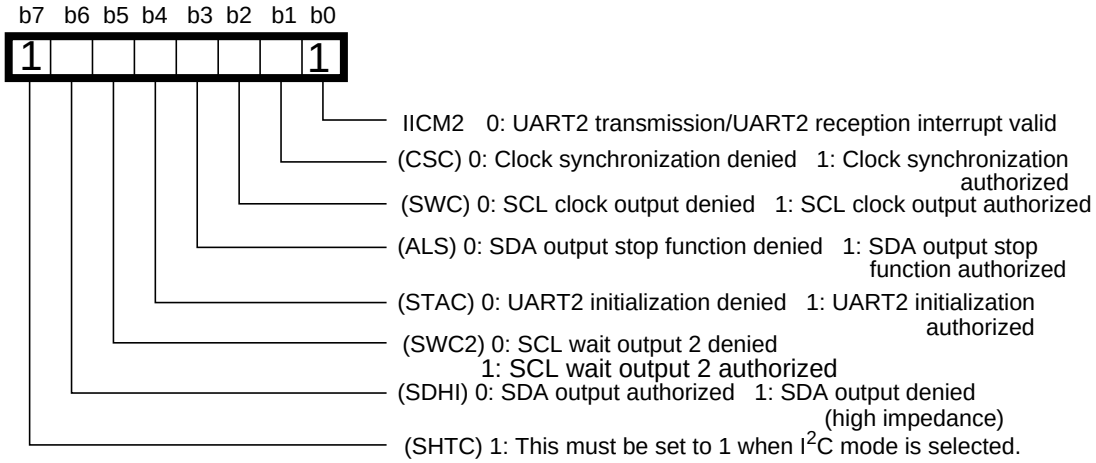
Note that if the reception buffer is read before the rise of the last bit of the reception clock (during simple I^2C bus mode reception complete interrupt), the reception data will be read with the bit positions altered.

Relevant register

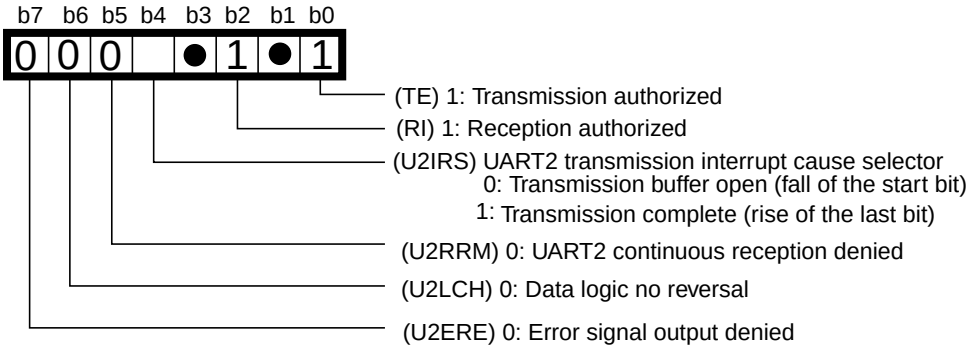
UART2 special mode register 2 (U2SMR : 0377_{16} address)



UART2 special mode register 2 (U2SMR : 0376_{16} address)

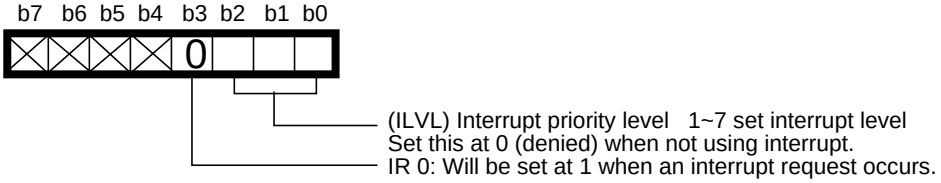


UART2 send/receive control register 1 (U2C1: $037D_{16}$ address)

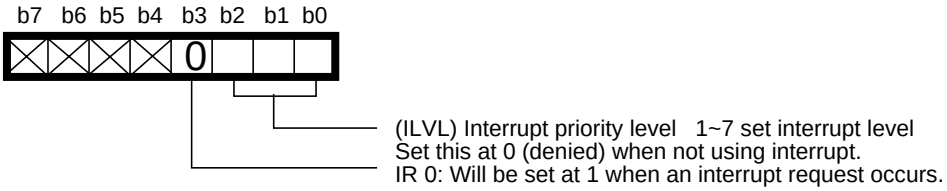


3.2 Start Conditions/Stop Conditions

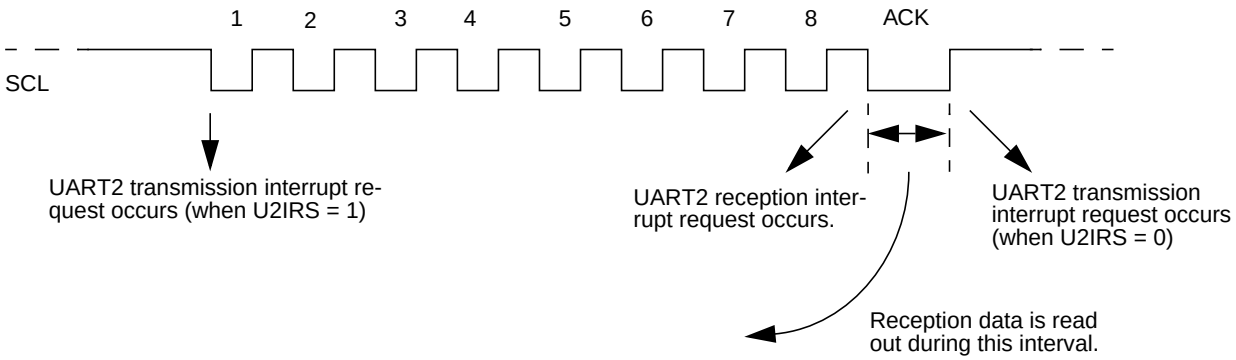
UART2 transmission interrupt control register (S2TIC : $004F_{16}$) when transmission interrupt is used.



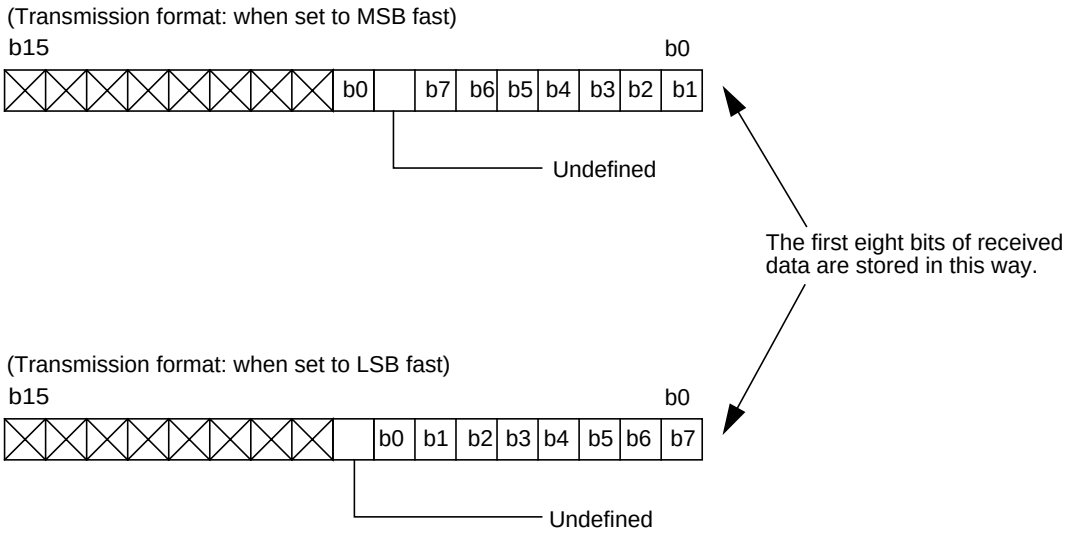
UART2 transmission interrupt control register (S2TIC : 0050_{16}) when reception interrupt is used.



Timing pattern



UART2 reception buffer register (U2RB : $037F_{16}$ $037E_{16}$ addresses)



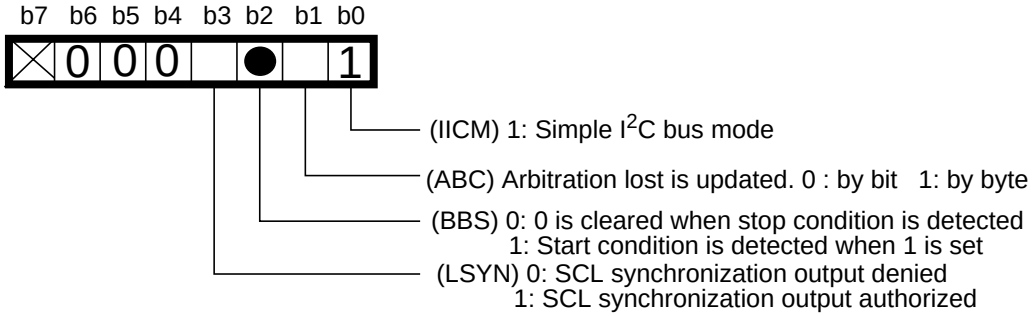
Start Condition/Stop Condition Detection

The M16C/62 is provided with a “start condition/ stop condition detection interrupt” to detect start conditions where the SDA changes from high to low when the SCL is high, and to detect stop conditions when the SCL is high and the SDA goes from low to high. This interrupt is allocated tosoftware interrupt number 10. When the I^2C bus mode is selected ([IICM] =1), the interrupt number 10 causes changes to “start condition/stop condition detection interrupt. Determine whether a start or stop condition has occurred in bus busy flag (BBS) when this interrupt is detected.

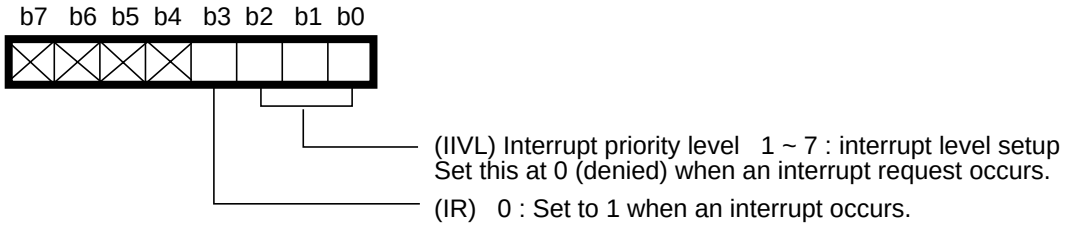
Please be aware that the start or stop condition setup and hold times may vary from the I^2C bus standartds. (See 4.1, “Start/stop condition setup and hold times.)

Relevant register

UART2 special mode register (U2SMR : 0337_{16} address)

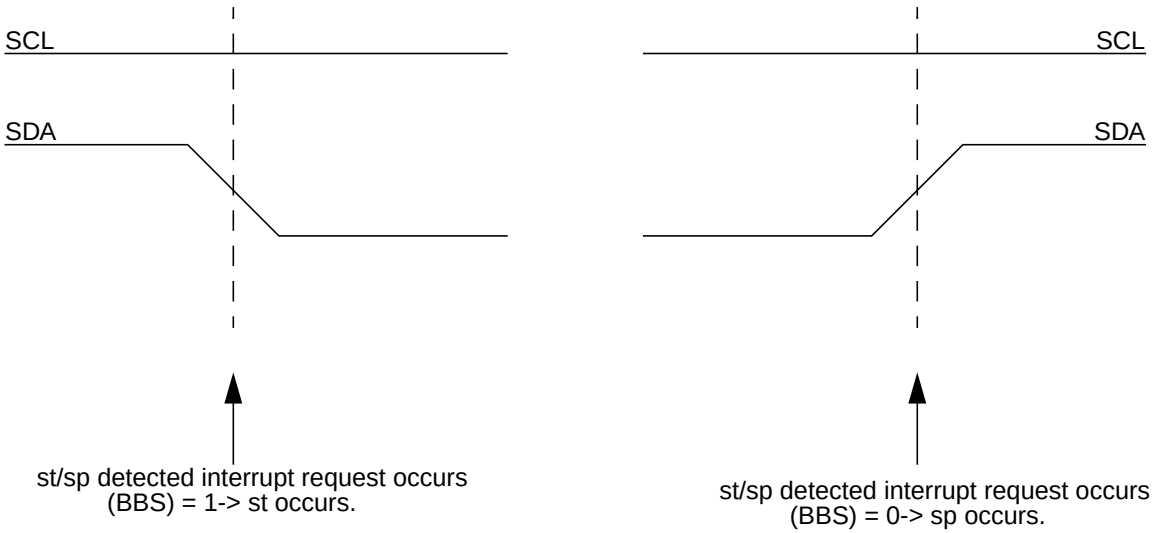


Bus collision detection interrupt control register 0 (BCNIC : $004A_{16}$ address)



I flag = 1 (when an interrupt is generated by an interrupt request)

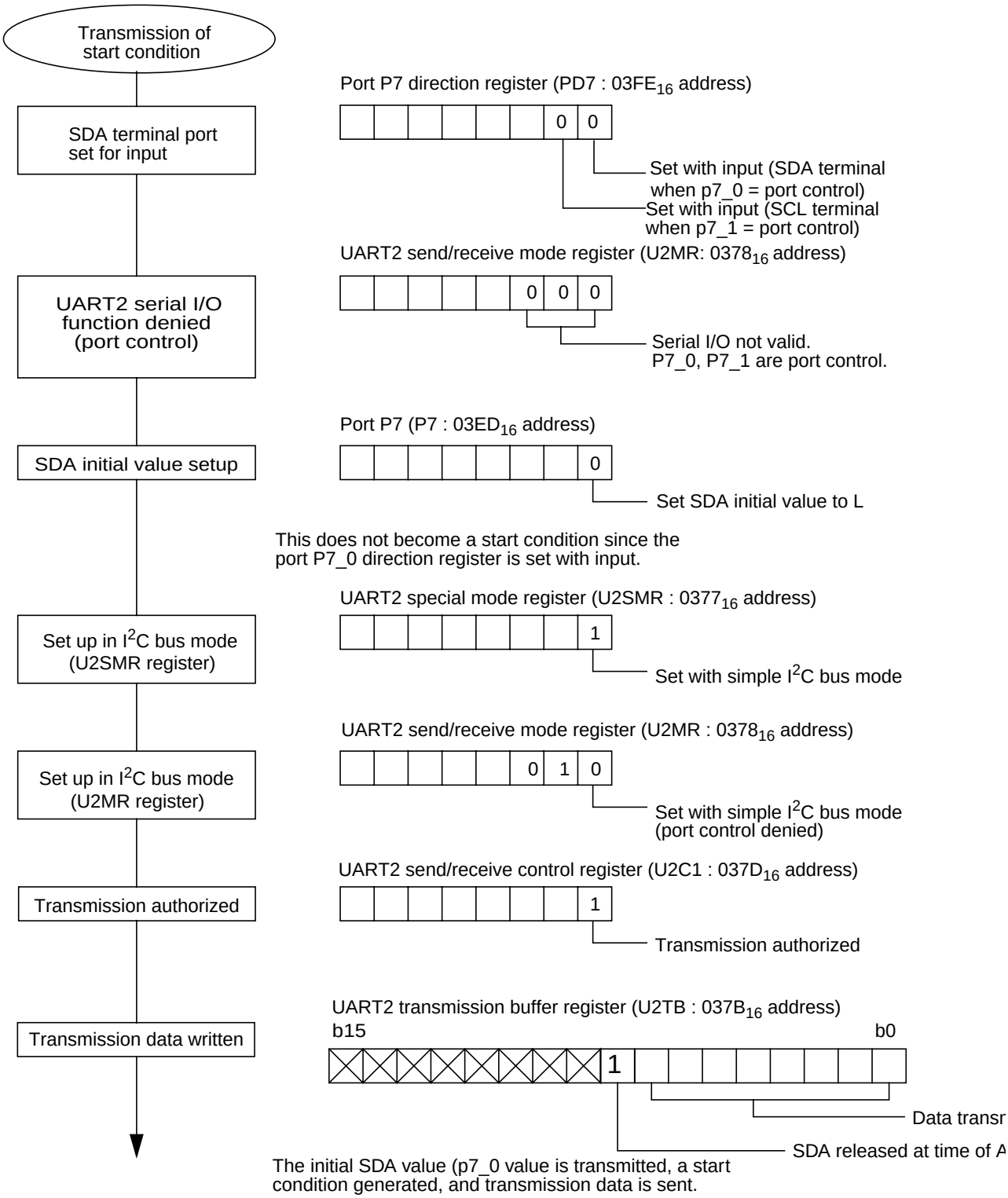
Timing pattern st: start condition sp: stop condition



3.2 Start Condition/Stop Condition

When the M16C/62 is used as a master, port control generates start conditions. In the simple I2C bus mode of the M16C/62, the SDA transmission initial value becomes the value that was set in the port p7_0 when serial I/O control was denied (at time of port control), so this function is used. The following is an example of control that generates a start condition.

Flow Chart

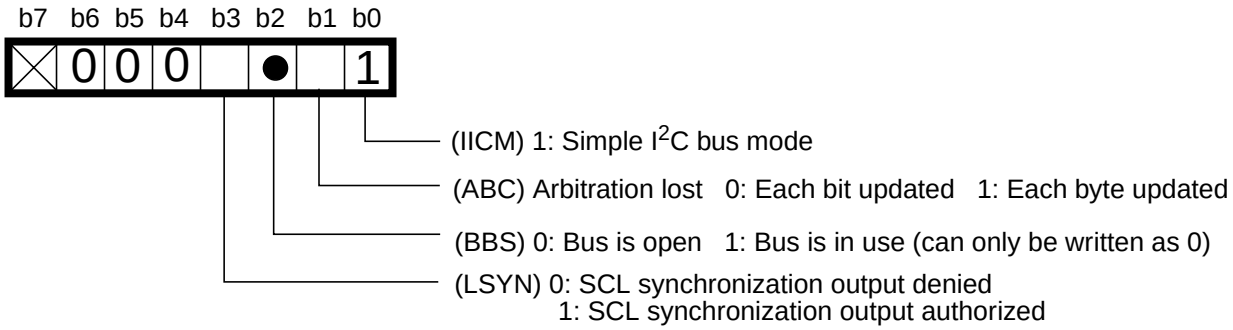


Bus Busy Detection

It is necessary to confirm that the bus is open before sending a start condition. In the M16C/62 simple I^2C bus mode, the bus status can be detected with the bus busy flag (BBS).

When a start condition is detected, the BBS is set to 1. When a stop condition is detected, the BBS is cleared to 0. Therefore, the bus is in use when the BBS equals 1 when the device attempts to send a start condition, so the device must wait to start sending untl the BBS clears to 0.

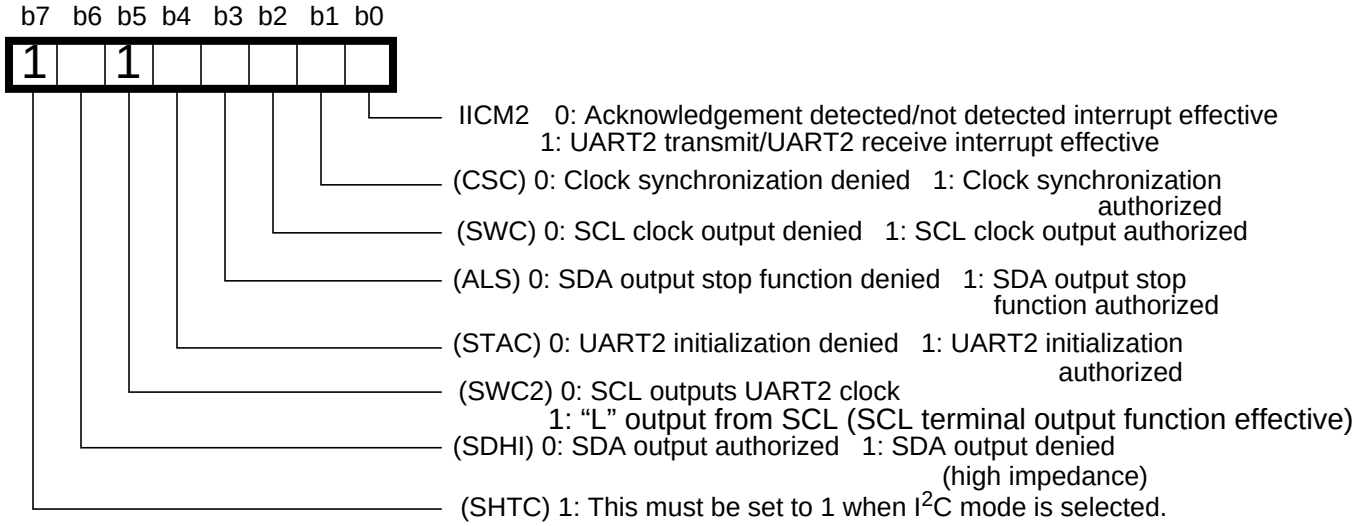
Relevant register



3.2 Start Condition/Stop Condition

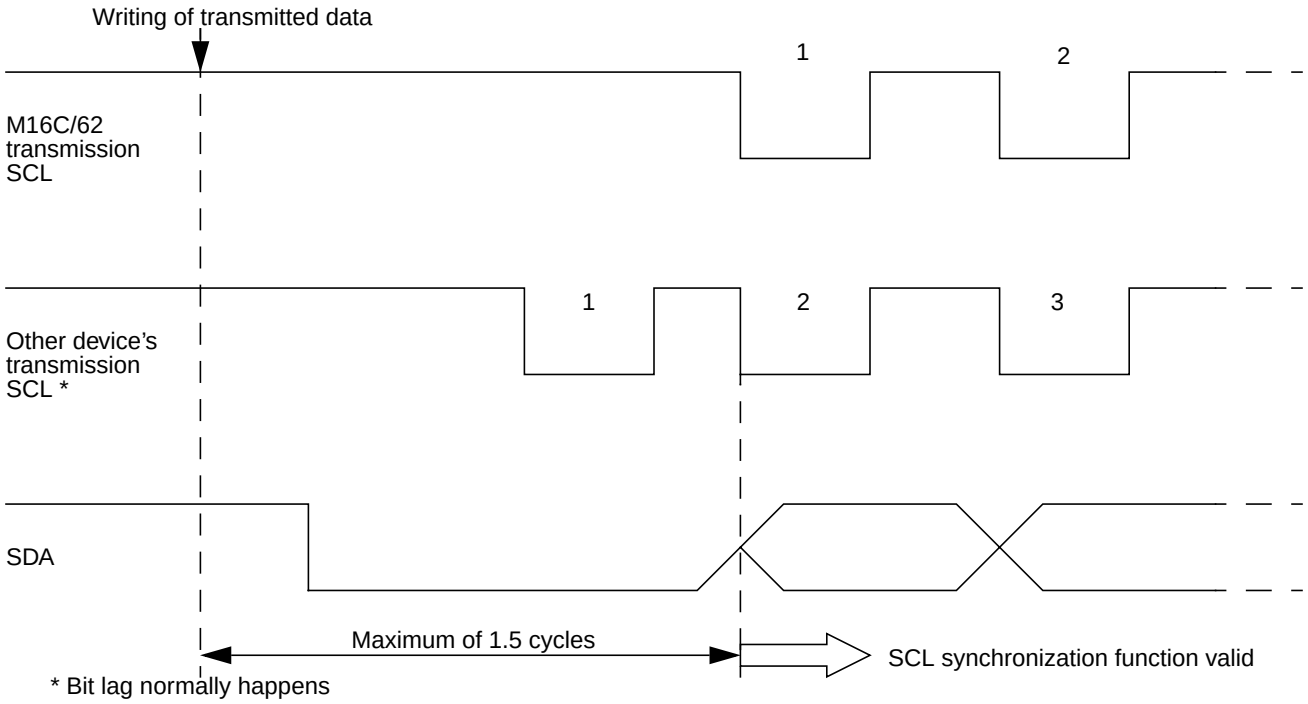
SCL Terminal L Output Function 2

A maximum period of 1.5 cycles of the transmission clock (SCL) is needed by the M16C/62 serial I/O from when the transmission data is written into the transmission buffer until transmission of the transmission clock (the SCL in the simple I^2C bus mode). Also, there is a possibility that bit lag could be caused (upper timing pattern) if another device transmits at the first bit in the interval from the time the start condition is generated until the clock line (SCL) synchronization function (see 3.5, Communication Coordination) is effective from the first bit of the SCL transmission. The M16C/62 has an SCL terminal L-output function to prohibit clock transmissions from other devices after the start condition. By using this function, I is output from the SCL terminal and other devices can be put into wait status at the same time that data is written into the buffer (lower timing pattern). This function is made operable by assigning a value of 1 to the wait output bit 2 (SWC2), which is L output from the SCL. The function is released by making the wait output bit 2 equal to 0.

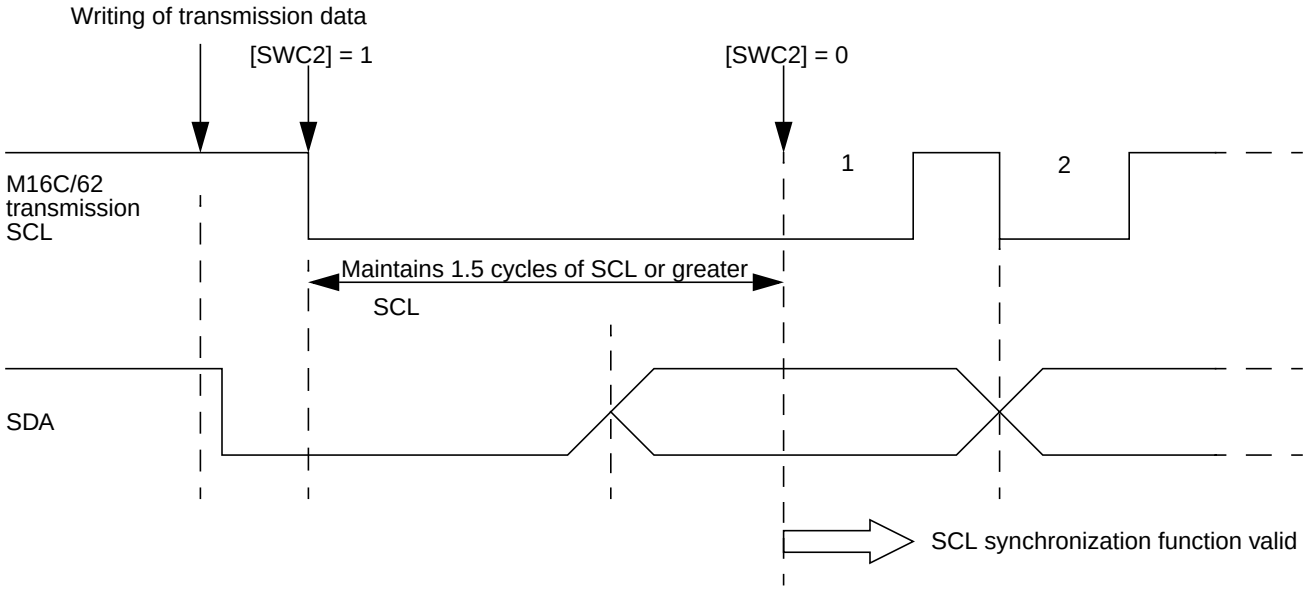


Timing pattern

When the SCL terminal output function is not used



When the SCL terminal output function is used



3.3 Acknowledgement

An acknowledgement is added to each byte of transmission data.

There must be a means, byte by byte, for the receiving device to detect whether or not there is an acknowledgement when the M16C/62 is being used as a transmission device. This is why the hardware is provided with an "acknowledgement detection interrupt" and an "acknowledgement undetected interrupt." Also, an acknowledgement can easily be generated by setting the 9^{th} bit of transmission data to "0" when the M16C/62 is used as a receiving device and one-to-one communication performed. (See 3.1, "Byte Data Reception Method.")

The I^2C mode selector bit 2 (IICM2) must be set to "0" when an "acknowledgement detection interrupt" and "acknowledgement undetected interrupt" are used. This setting makes interrupt numbers 15 and 16 into "acknowledgement detection interrupt" and " acknowledgement undetected interrupt," respectively. In this case, data transmission from the UART2 reception register to the reception buffer register is timed for the rise of the last bit of the reception clock. (See 2.4, "Register Setup When in Simple I^2C Bus Mode.")

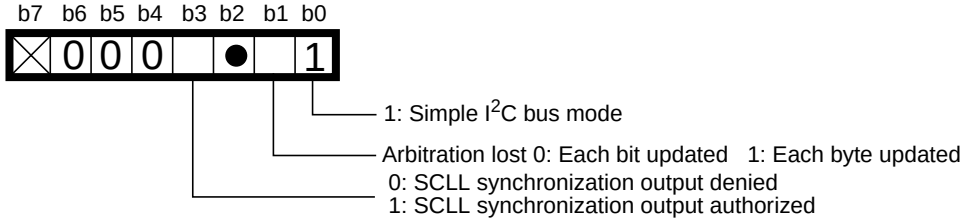
Acknowledgement Detected

The reception device can determine that an acknowledgement has occurred at the time of the rise of the transmission clock's 9th bit when the open SDA line on the transmitting side level reaches "L." The M16C/62 can detect this condition with the "acknowledgement detection interrupt" function. This interrupt is assigned "software interrupt number 16." The interrupt number 16 interrupt cause "acknowledgement detection interrupt" is the case only when the I^{2}C mode is selected ([IICM] = "1") and the I^{2}C mode selector bit 2 [IICM2]= "0."

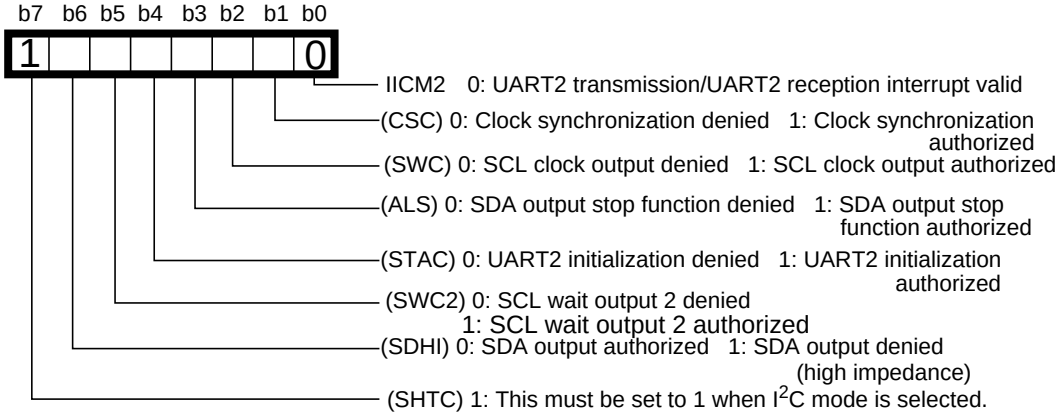
The reception device can determine that an acknowledgement has occurred at the time of the rise of the transmission clock's ninth bit when the open SDA line on the transmitting side level reaches L. The M16C/62 can detect this condition with the "acknowledgement detection interrupt" function. This interrupt is assigned software interrupt number 16. The interrupt number 16 causes the "acknowledgement detection interrupt" in this case only when the I^2C mode is selected ([IICM] = 1) and the I^2C mode selector bit 2 [IICM2] = 0.

Relevant registers

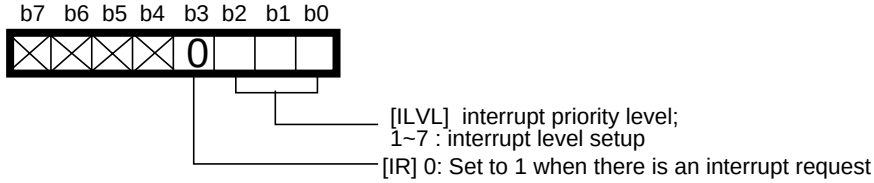
UART2 special mode register (U2SMR : 0377_{16} address)



UART2 special mode register 2 (U2SMR : 0376_{16} address)



UART2 reception control register 0 (S2RIC : 0050_{16} address)
When acknowledgement detection interrupt is used.



I flag = 1 (when an interrupt occurs because of an interrupt request)

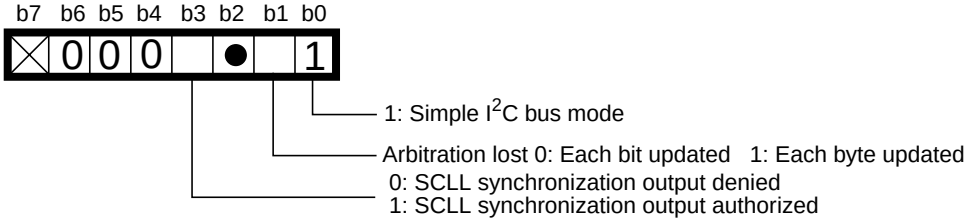
Timing pattern

Acknowledgement Undetected

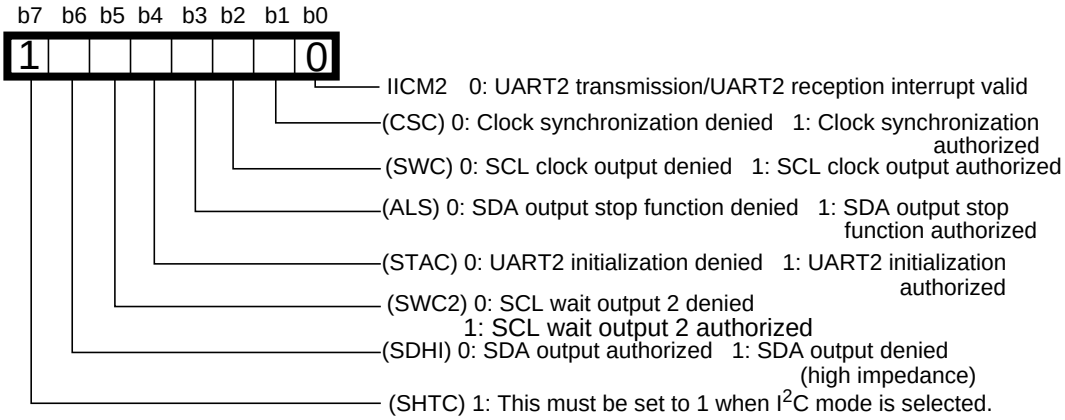
The reception device can determine that an acknowledgement has occurred at the time of the rise of the transmission clock's ninth bit when the open SDA line on the transmitting side level reaches H. The M16C/62 can detect this condition with the "acknowledgement undetected interrupt" function. This interrupt is assigned software interrupt number 15. The interrupt number 15 causes the "acknowledgement undetected interrupt" in this case only when the I^2C mode is selected ([IICM] = 1) and the I^2C mode selector bit 2 [IICM2] = 0.

Relevant registers

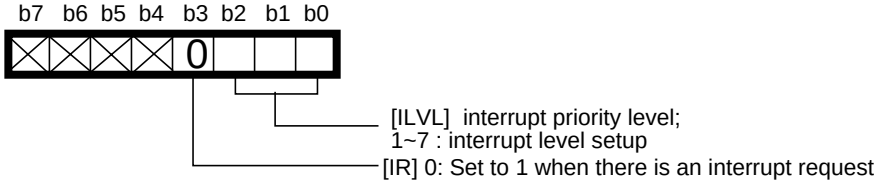
UART2 special mode register (U2SMR : 0377_{16} address)



UART2 special mode register 2 (U2SMR : 0376_{16} address)

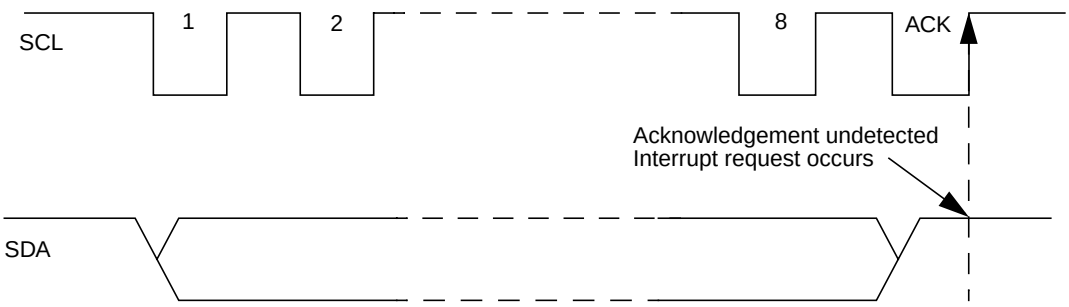


UART2 reception control register 0 (S2RIC : 0050_{16} address)
When acknowledgement detection interrupt is used.



I flag = 1 (when an interrupt occurs because of an interrupt request)

Timing pattern



3.4 Own-address Designation Determination

The reception device can determine that an acknowledgement has occurred at the time of the rise of the transmission clock's ninth bit when the open SDA line on the transmitting side level reaches H. The M16C/62 can detect this condition with the "acknowledgement undetected interrupt" function. The interrupt is assigned software interrupt number 15. The interrupt number 15 causes the "acknowledgement undetected interrupt" in this case only when the I2C mode is selected ([IICM] = 1) and the I2C mode selector bit 2 [IICM2] = 0.

SCL Terminal L Output Function

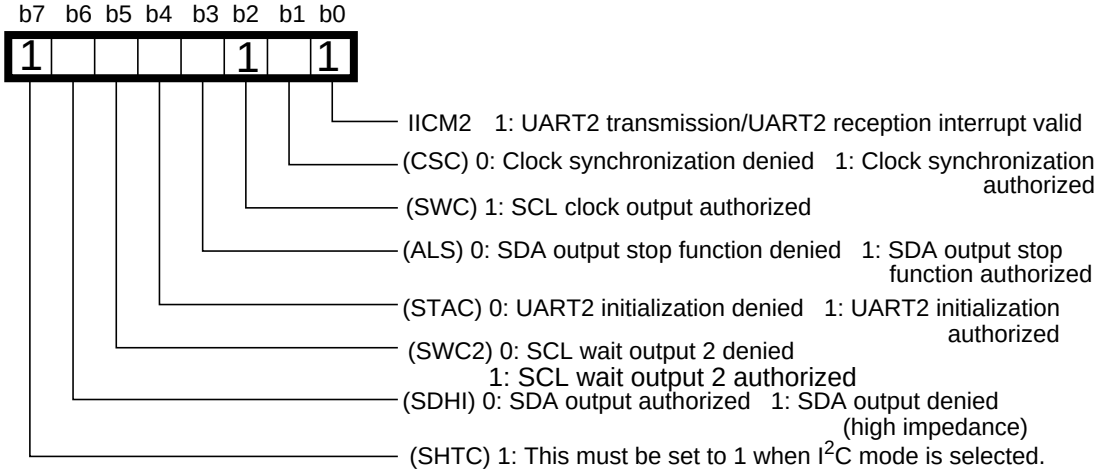
With the I^2C bus, the designated slave's address is sent to the first byte after the start condition is detected (when in seven-bit address mode). The slave must compare with its own address the reception data of the first seven bits of tthe clock sent by another master in the first byte and perform processing to generate (or not to generate), a synchronizing acknowledgement in the clock's ninth bit. The M16C/62 has the SCL terminal L output function to perform this processing. By using this function, the M16C/62's SCL terminal outputs in L timed to the SCL's ninth bit after receiving data in the first eight bits, forcing the master into waiting status. Then after the software completes address comparison processing, port control can generate or not generate an acknowledgement. (See 3.1, "Byte Data Reception Method" for one-to-one communications or other use where address reception and acknowledgement generation can be performed.

Permission to run this function is given by setting the wait output bit [SWC] to 1 and is denied by setting it to 0. Also when the SCL terminal is 0 with this function, the function is released by setting [SWC] to 0.

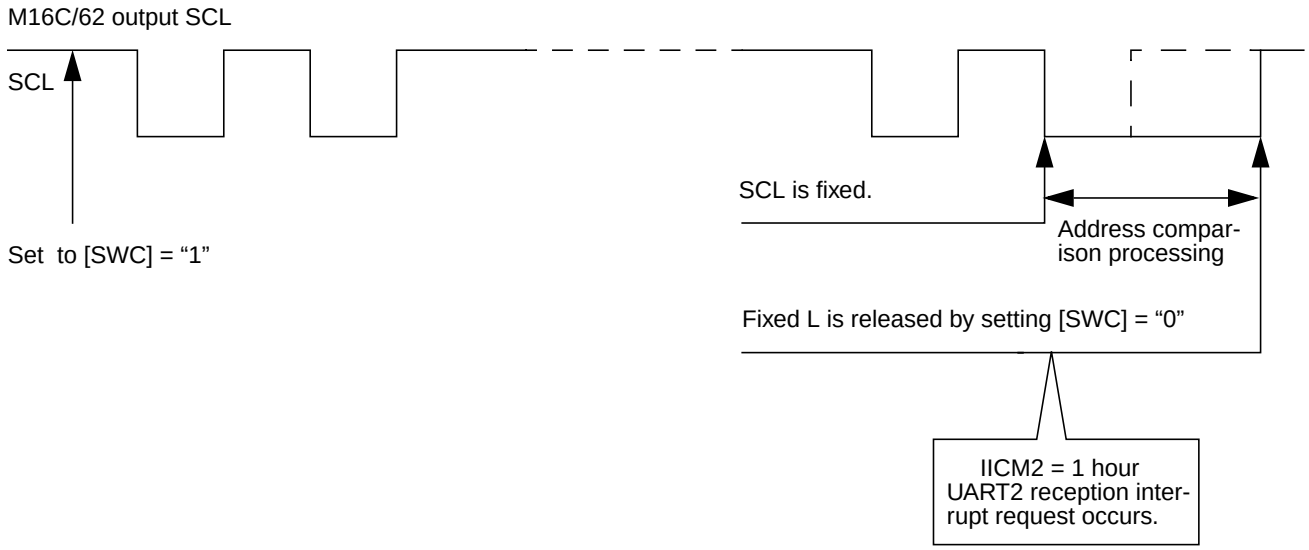
When address comparison processing is performed with this function, the contents of the reception buffer are read prior to the rise of the last bit of the clock, so be aware of the fact that the bit location of the reception data which has been read out is changed. (See 3.1, "Transmission Interrupt/Reception Interrupt", timing pattern.)

Relevant register

UART2 special mode register 2 (U2SMR2: 0376_{16} address)



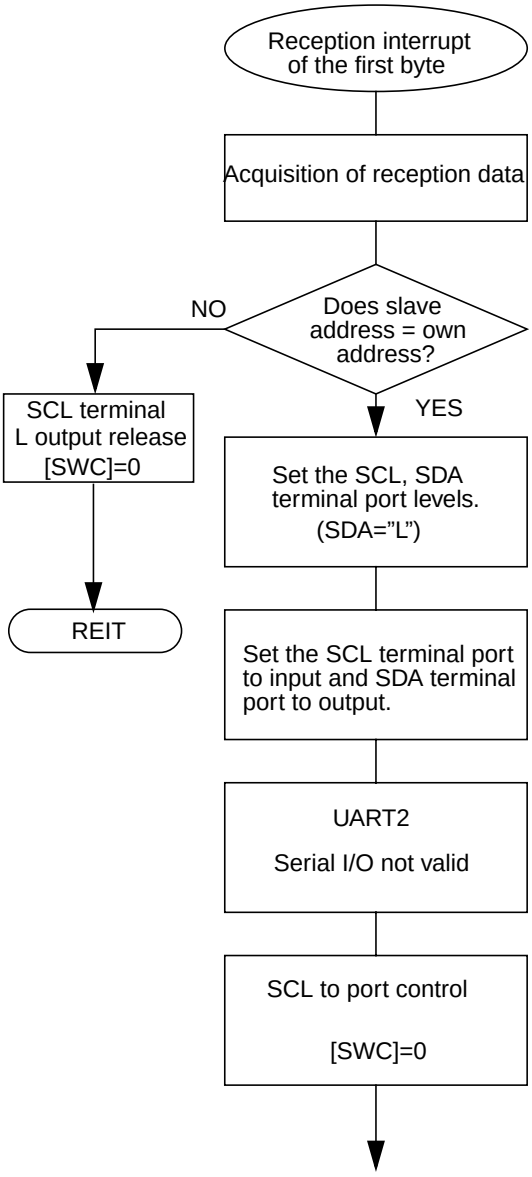
Timing pattern



Example of Own Address Recognition

There are two types of address designation formats: 7-bit addresses and 10-bit addresses. Although the following is an example of 7-bit address recognition, the same kind of control may be used to recognize 10-bit addresses. This example shows the generation of a 1st byte reception interrupt after the reception of the start bit. Here, [SWC] = "1" (SCL terminal L-output function authorization), is already set before receiving the 1st byte, and only a portion of the reception interrupt routine is shown.

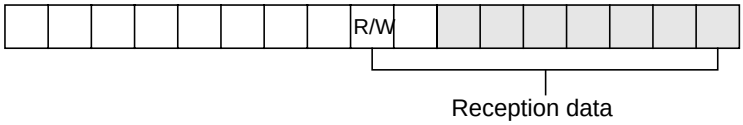
Flow Chart



Example of main routing settings

SWC = 1 (SCL terminal L-output function authorization)
IICM = 1 (interrupt cause #16: UART2 reception interrupt, interrupt generation timing: rise of last bit)
S2RIC = 7 (UART2 reception interrupt authorization)
I Flag = 1 (interrupt authorized)

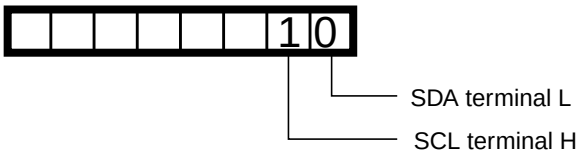
UART2 reception buffer register (U2RB : $037F_{16}$, $037E_{16}$)



Slave address

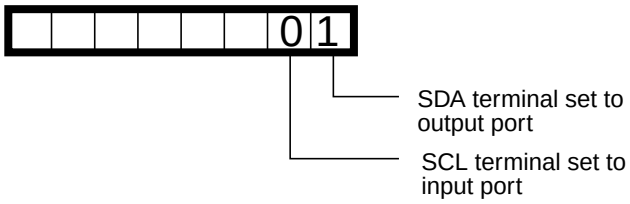
R/W R/W

Port P_7 (P_7 : $03ED_{16}$ address)



(Input/output settings when set to port control)

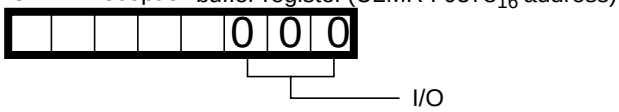
Port P_7 direction register (P_7 : $03EF_{16}$ address)



(Input/output settings when set to port control)

ACK sent

UART2 reception buffer register (U2MR : 0378_{16} address)



3.5 Communication Coordination

It is possible that more than one master could generate a start condition and attempt to start sending data at the same time (an arbitration occurrence), when the I^2C bus is used as a multiplexed master. In this case, communication coordination is performed between the masters in the I^2C bus system. In the M16C/62 simple I^2C bus mode, the hardware is provided with the "arbitration lost detection function" and the "SDA output prohibition function at time of arbitration lost occurrence" in order to recover communications when there is an arbitration lost occurrence. Also, an "SCL synchronization function" is provided apart from arbitration lost as one of the means of communication coordination using clock synchronization.

3.5 Communication Coordination

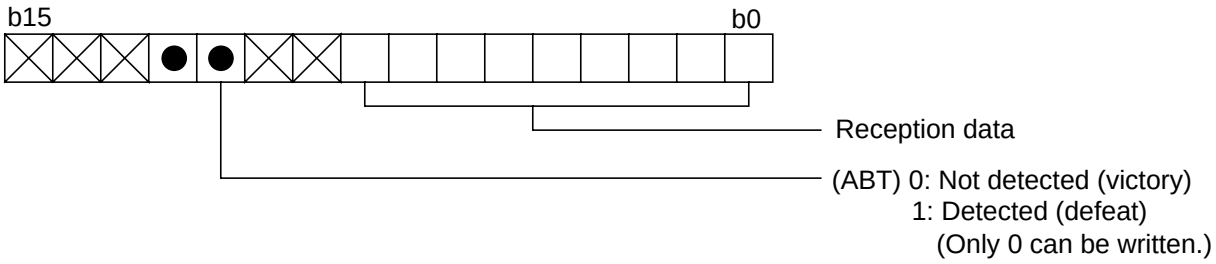
Arbitration Lost Detection Function

The M16C/62 simple I2C bus mode has an arbitrartion lost detection flag (ABT). This detection flag is located in bit 3 of the UART2 reception buffer register. The arbtration lost detection flag (SBT) set to 1 when the internal data level and the SDA level do not agree at the time of the SCL rise, indicating an error. The arbitration lost detction flag control (ABC) updates the arbitration lost detection flag and can select by bits (0) or bytes (1). Fix the flag at (ABC) = 0 when both the I2C mode selector bit (IICM) and the I2C mode selector bit 2 (IICM2) are set to 1.

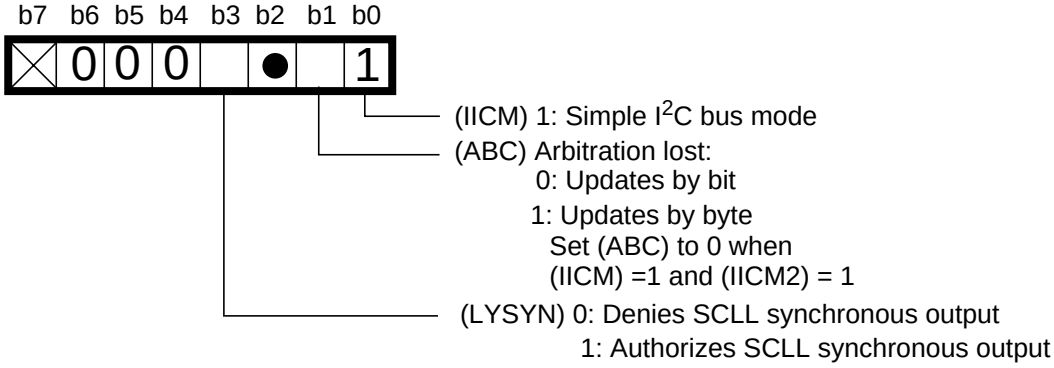
Output becomes H and input becomes I when the acknowledgement is received and the arbitration lost detection flag ends up being set. Therefore, at the time of the next transmission, perform sending after the arbitration lost detection flag clears to 0.

Relevant register

UART2 reception buffer register (U2RB: $037F_{16}$ address)



UART2 special mode register (U2SMR: 0377_{16} address)



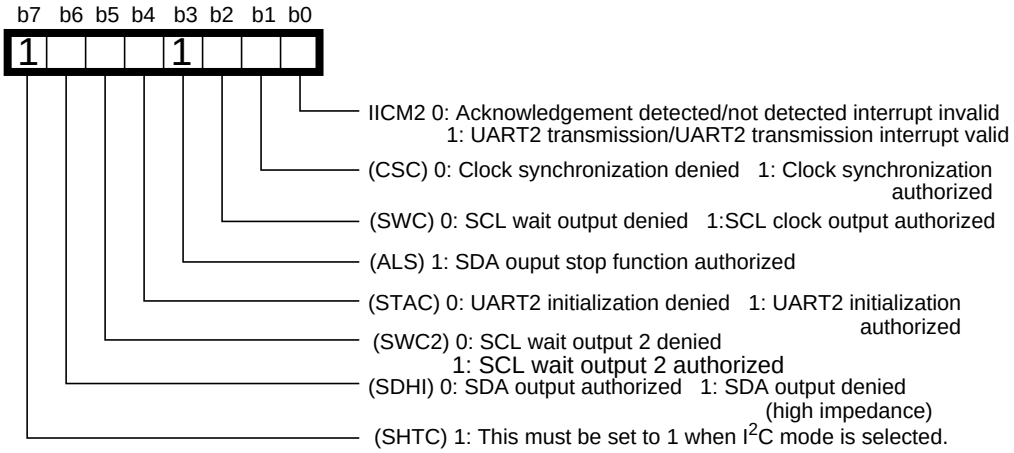
3.5 Communication Coordination

SDA Output Prohibition Function at Time of Arbitration Lost Occurrence

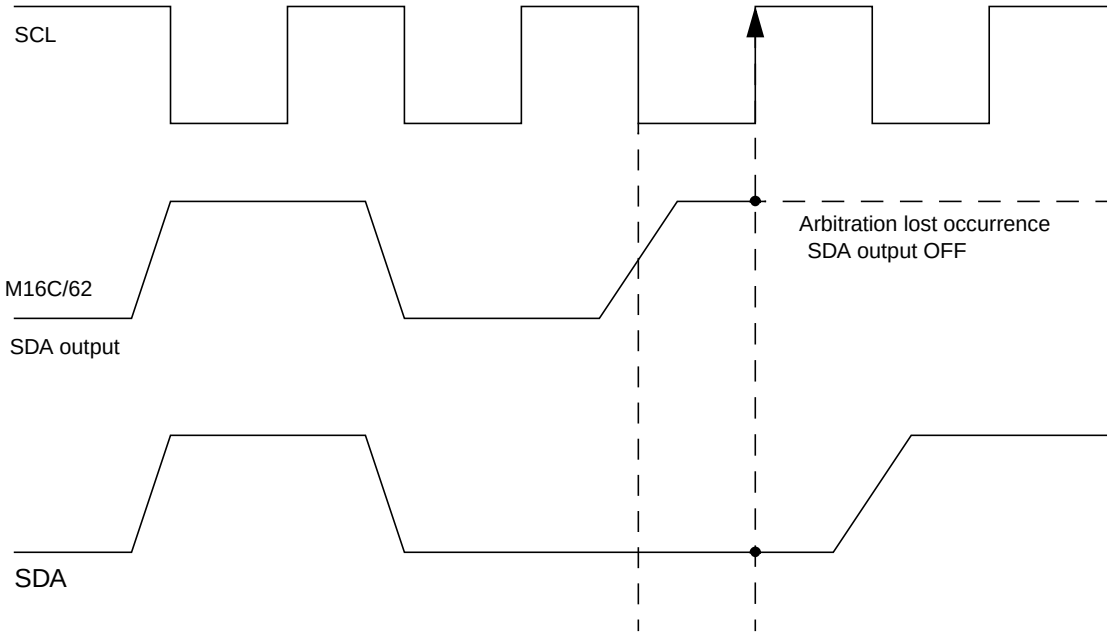
The SDA output of masters that have detected an arbitration lost must be turned off at that time. The M16C/62 simple I2C bus mode can select a function that automatically turns off SDA with the hardware when there is an arbitration lost occurrence. This function is authorized by setting the SDA output stop bit (ALS) to 1 and is denied by setting this bit to 0. When the SDA output is turned off with this function, the function is released by clearing either the SDA output stop bit (ALS) or the arbitration detection bit (ABT) before sending byte data, since an arbitration lost was determined to have occurred and output was turned off, even at the timing of the acknowledgement. Also, fix the arbitration detection flag control (ABC) at 0.

Relevant register

UART2 special mode register 2 (U2RB: 0376_{16} address)

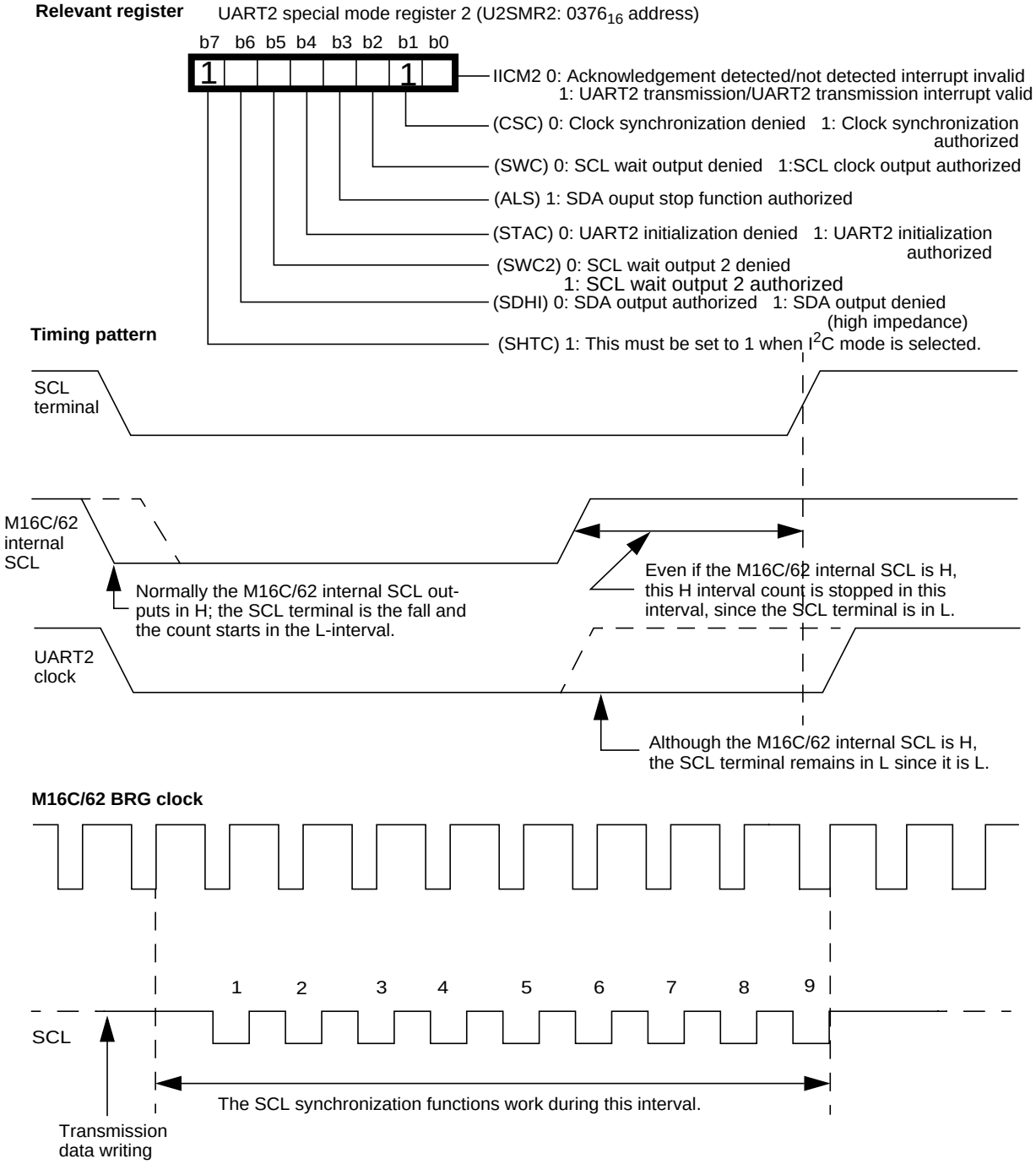


Timing pattern



3.5 Communication Coordination

When the M16C/62 is connected to a slower device, the other device may put an L-hold on the SCL and put the clock sent from the master into a forced waiting status. The M16C/62 simple I2C bus mode has an SCL synchronization function that automatically enters wait status in response to an L-hold from the other device and that also releases the wait status by releasing the L-hold. The operation of this function is authorized by setting the clock synchronization bit (CSC) to 1. This function should be used only when the M16C/62 is set as a master (internal clock mode).



SCL Synchronization Function

3.6 Other Functions

In addition to the functions described above, the M16C/62 simple I^2C bus mode is provided with the following hardware designed to make I^2C bus control easier.

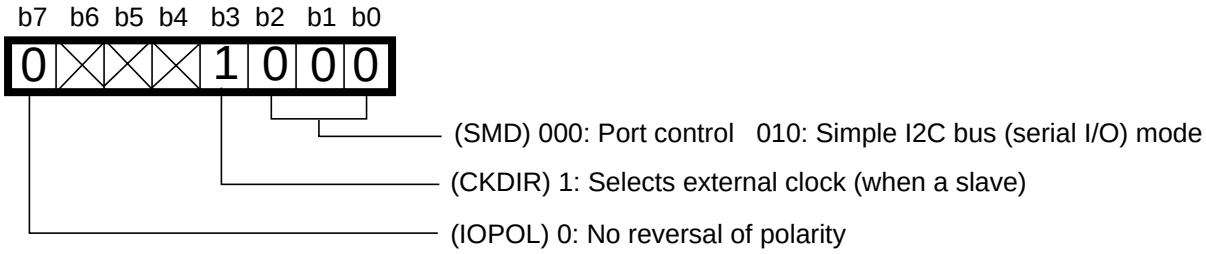
SCL L-Synchronized Output Function

The M16C/62 simple I2C bus mode has an SCL L-synchronized output function. This function is authorized by setting the SCL L-synchronized output authorization bit (LSYN) to 1 and is denied by setting the bit to 0. By setting the SCL L-synchronized output authorization bit (LSYN) to 1, the SCL terminal is synchronized by the level going to L and the P7_1 data register (the port assigned by the SCL terminal) is reset to 0. This function is valid when SCL/SDA are used as ports (when the serial I/O mode selector bit (SMD0, SMD1, SMD2 = 000) and is invalid in serial I/O mode.

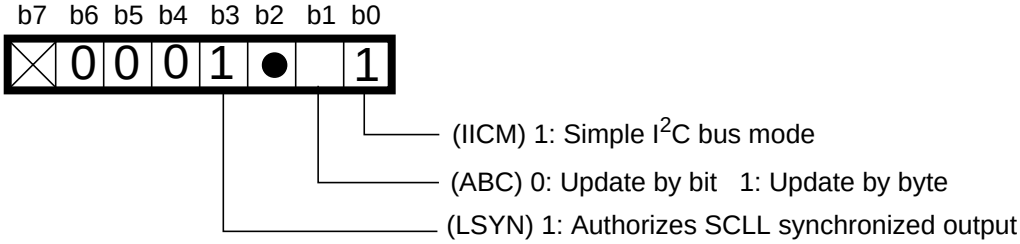
This function can be used to transmit a first-byte acknowledgement whe the M16C/62 is a slave receiver, but normally you should use the SCL terminal L-output function instead.

Relevant registers

UART2 reception mode register (U2MR: 0378_{16} address)



UART2 special mode register (U2SMR: 0377_{16} address)



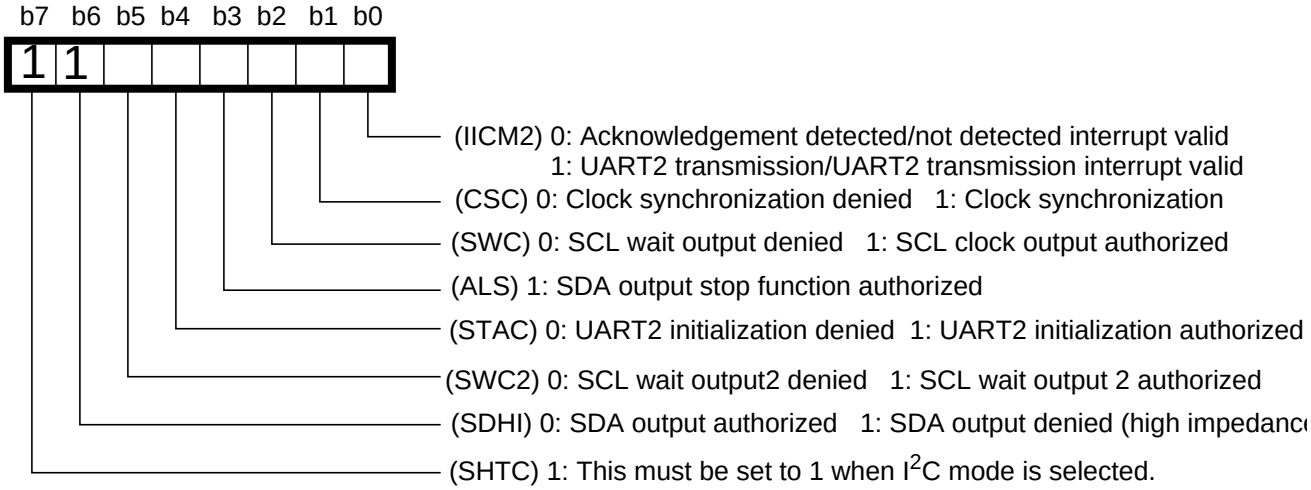
3.6 Communication Coordination

SDA Output Prohibition Function

When the M16C/62 is used as a slave, the M16C/62 must turn the SDA output off (high impedance), if the address designated by the master is different from its own address when the address is determined after the reception of a start condition. In this case, the SDA is turned off by the M16C/62's setting the transmission buffer register to 1FFh at every ninth clock of the SCL (at each occurrence of a reception interrupt request). Also, the SDA output can be turned off even if the SDA output prohibition function provided with the M16C/62 is used. The function is made valid by setting the SDA output prohibition bit (SDHI) to 1, so that the M16C/62's SDA output can be high impedance, even if the transmission buffer register is set to 1FFh. This function is released by setting the (SDHI) to 0 and that value that is set in the synchronized transmission buffer is output in the next SCL input.

Relevant register

UART2 reception mode register 2 (U2SMR2: 0376_{16} address)



3.6 Communication Coordination

UART2 Initialization Function

The M16C/62 simple I2C bus mode is provided with a function that automatically initializes UART2 at a time when a start condition is detected. The function is used when the M16C/62 is a slave. This function is authorized by setting the start condition initialization bit (STAC) to 1; it is denied by setting the bit to 0. The following initialization is performed when a start condition is detected:

1 The transmission register is initialized and the contents of the transmission buffer register are transferred to the transmission register. In so doing, there is no need to reset the data in the transmission register when data is received. Transmission begins with the input of the next clock as the first bit. Note, however, that you should prohibit the output of transmission data by setting the SDA output prohibition bit (SDHI) to 1, since this transmission data is identical to the last transmission data.

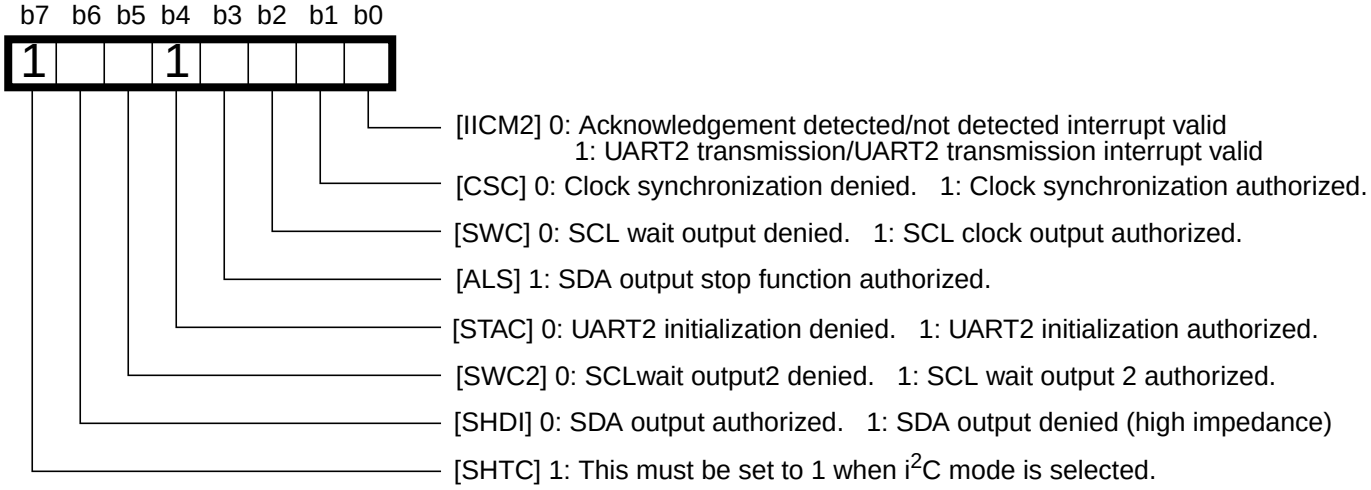
2 The receiving register is initialized and the next clock is input as the first bit, thus starting reception. Overrun errors do not occur, even though initialization and reception is timed to begin prior to reading the reception buffer register.

3 The wait output bit (SWC) is set to 1. In so doing, the SCL terminal L-output function becomes valid and L is output from the SCL terminal at the fall of the ninth bit of the transmission clock.

Use this function only when an external clock has been selected. Also be aware that the transmission buffer open flag's value does not change if this function is used and UART2 transmission is started.

Relevant register

UART2 special mode register 2 (U2SMR2: 0376_{16} address)



Chapter 4 Precautions Concerning the Simple I^2C Bus Mode

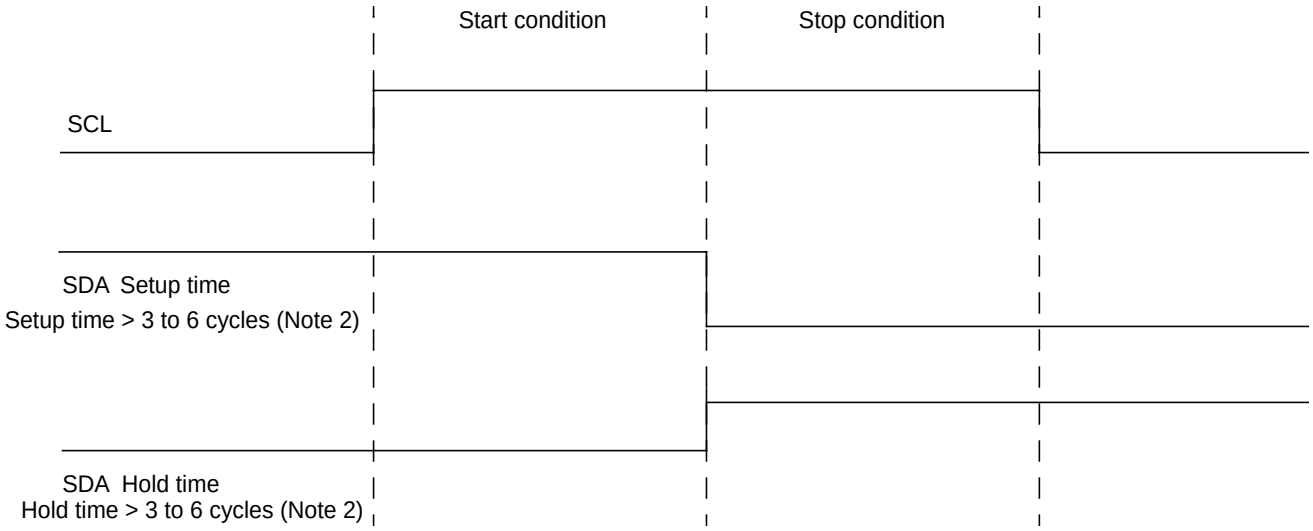
Please observe the following precautions and restrictions for I^2C bus protocol control when using the M16C/62 simple I^2C bus mode.

4.1 Electrical Characteristics

There is one difference in the electrical characteristics of the M16C/62 with the I^2C bus standard: (See the I^2C Bus Standard, 1.7 "I^2C Bus SDA and SDA Bus Line Characteristics")

Start/Stop Condition Setup Time/Hold Time

The M16C/62 setup and hold times may vary from I^2C bus standard values when start/stop conditions are detected (when in high-speed mode). During start/stop conditions, the M16C/62 setup and hold times are detected as having the following values. (Note 1)



Note 1: Be sure to set the start/stop condition SHTG bit to 1.

Note 2: The number of cycles indicates the main clock input oscillation frequency $f(X_{IN})$ number of cycles.

In high-speed mode, the I^2C bus standard has start/stop condition setup and hold times of minimum 600 nanoseconds. In contrast, the M16C/62 ssetup/hold time is a minimum 6 cycles ($f(X_{IN})$ cycles). Therefore, when the main clock ($f(X_{IN})$ is used at 10 MHz, the M16C/62 simple I^2C bus mode setup/hold time is a minimum of 600 nanoseconds and can handle the high-speed mode of the I^2C bus standard. However, setup and hold times can no longer satisfy the high-speed mode I^2C bus standard when the main clock is used at less than 10 MHz.

4.1 Electrical Characteristics

Low Level/High Level Input Voltages

The electrical characteristics of the M16C/62 when it runs at 2.7V~5.5V are:

"H" input voltage (VIH) = min. 0.8 Vxx (guaranteed level)
"L" input voltage (VIL) = max. 0.2 Vxx (guaranteed level)

Therefore, these are different from the I^2C standard values of:

Running at 5V: VIH = 3V, VIL = 1.5V
Running at other than 5V: VIH = 0.7V, VIL = 0.3V

Also, when the M16C/62 "L" output voltage is V_{CC} = 5V, IOL = 5mA:
"L" output voltage (VOL) = max. 2.0V (guaranteed level)

Which is different from the I^2C bus standard value of:
"L" output voltage (VOL) = max. 0.6V (when IOL = 6mA)

However, the standard M16C/62 characteristics when V_{CC} = 5V and IOL = 5mA:
"L" output voltage (VOL) = 0.6V.

4.1 Electrical Characteristics

Data Hold Time

The provision of a minimum 300ns hold time for the SDA signal (in the SCL signal VIH min.), is requested in order to fill in the undefined region of the SCL fall width in the I^2C bus standard.

However, in the M16C/62,
TxDi hold time = min.0ns

The 300ns hold time requested for the I^2C standard is not generated internally.

4.2 Maximum Transmission Speed Limits with BRG Count Source

The time it takes for the M16C/62 to recognize the SCL level depends on sampling cycle. A maximum BRG count source of 3 clocks is needed. Therefore, the maximum transmission speed of an I^2C bus connected to the M16C/62 simple I^2C bus is limited according to the operating frequency and the bits set for the BRG count source speed. There is a danger of bit lag if transmission speeds do not satisfy the following conditions:

I^2C bus maximum transmission speed (Hz) < BRG count source (Hz)) 3

Example: Source oscillation 10Hz, with BRG count source fc32 selected:

I^2C bus maximum transmission speed (Hz) < 10MHz)32 / 3 = 104Kbps
(BRG count source)

In this example, the I^2C bus maximum transmission speed is 104Kbps.

Appendix

Reference Programs

Appendix and Reference Programs

In conclusion, we present reference programs to be used when sending and receiving is performed using the M16C/62 simple I^2C bus mode as a multi-master. These programs do not guarantee communication operations, so please evaluate them fully when using them.

Appendix

Soft I^{2}C Bus Control Specification

Contents

Appendix Reference Programs

1. Summary

This software controls the simple I^2C bus mode hardware installed in the M16C/62 series, enabling I^2C bus communications protocol. I^2C bus communications protocol is based upon the following use conditions:

Operating conditions

Source frequency: 10MHz (non-wait, no frequency divider)

Specification restrictions[1]

It is assumed that the bus line for the I^2C is not locked. If the bus line is locked, this software's processing will also lock up and it will be impossible to restore the user's program. We recommend bus line monitoring and recovery processing from lock status (such as a watchdog timer), as upper level applications to deal with this issue.

Communication with slave units having 10-bit addresses is not supported.

Mixes of C-BUS, M3L-BUS, and other I^2C bus interchangeable protocols are not supported.

Do not reuse communication formats in the bus that have restart conditions and which switch over to slaves. (These will bring about communication abnormalities.)

2. Explanation of functions

2.1 Address

Master Device
Sending and receiving with slaves having 7-bit addresses.

Slave Device
Has 7-bit address

Note: Sending/receiving with special addresses (such as general call addresses), is not supported.

2.2 Transmission Speed

Transmission speeds are between 0~100Kbps, so the device cannot communicate with high-speed mode masters. In this program, a 15us software wait is inserted until the SCL synchronization function becomes valid when a start condition is sent. This assumes communication at 100Kbps. Change the timing of the software wait insertion when communication takes place at speeds other than 100Kbps.

1. Could be a typo for "Use restrictions." [translator]

2.3 Transmission Data Length

Master Device

Can send/receive data between 1 and 256 bytes in length.

Slave Device

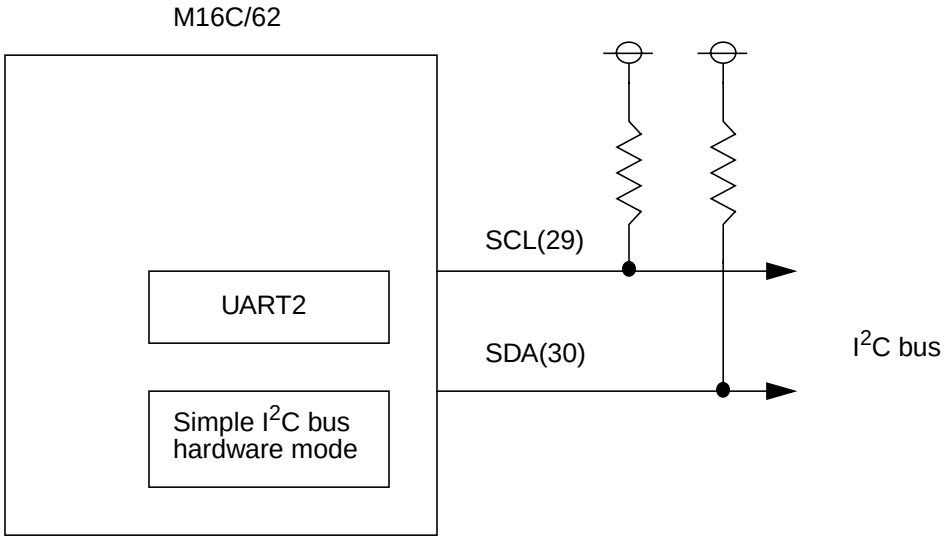
Operates the same as the 24LC01 (E2PROM supporting 128-byte I^2C bus). However, the soft I^2C bus can optionally set its own device address with the user program. This function is primarily for other masters which are connected to the bus, and there is no status provided for it.
Also, it has the advantage of permitting writing or reading of its own device into this domain without going through the bus.

2.4 Multi-Master

Data can be relayed from several devices which are connected to the I^2C bus to other devices. However, you cannot use composite formats as a master. (This is because a restart condition cannot be generated.) Therefore, E2PROM random access reading cannot be performed between M16C/62s which are doing bus control of the same I^2C bus.

3. Explanation of Hardware

Only the M16C/62 UART2 + internal simple I^2C bus hardware are used in order to implement the I^2C bus protocol. Choose the pullup resistance which is most appropriate for the system.



4. Method of Use

4.1 How to Build In

User Program

Include i2c.h, since the function provided by the soft I^2C bus is used.

ncrt.a30

_Add the following descriptions as the final section entries in the *near* region. By so doing, you will establish a location used by the soft I^2C bus 14 byte RAM region.

```
.section        iicbus,data,align
```

,Add the following interrupt vectors:

Software interrupt number 10 (bus collision detection interrupt)

```
.glb             stsp_int
.lword           stsp_int
```

Software interrupt number 16 (UART2 reception interrupt)

```
.glb             u2rcv_int
.lword           u2rcv_int
```

i2cbus.a30

Sets the maximum interrupt level used by the soft I^2C bus.

The second place of i2cbus.a30

```
IICIPL          .equ    X
```

Enter 2~7 as descriptions for the X area.

In the soft I^2C bus, IPL X-1~X are used.

Please select the most appropriate value since there is a strong relationship between the value of X and the communication speed and execution speed. It is possible that the larger the value of X, the higher the communication speed, but there is also the chance that the interrupt prohibition time would be extended.

This program uses UART2 reception interrupt and bus collision detection; the 2 interrupt levels relate to each other as follows:
UART2 reception interrupt level < bus collision detection interrupt level

Both interrupts have the same priority level, and in software interrupt denied status, they also detect stop conditions, performing the following operations when a bus collision detection interrupt and a UART2 reception interrupt occur due to the reception of the 1^{st} byte of the next frame.

UART2 reception interrupts are executed in order of priority according to the hardware priority when an interrupt is authorized from the status outlined above, and then bus collision detection interrupt processing is executed. In this case, it is difficult to determine whether frame received previously (which was ended by the bus collision detection), was normal or not.
For these reasons, the UART2 reception interrupt level is set one level lower than the bus collision interrupt, allowing bus collision detection interrupt processing to be performed on a priority basis.

Access Denied Register
Do not change the following registers:

Address	Name of Register		7	6	5	4	3	2	1	0
04A16	Bus collision detection interrupt register		X	X	X	X	X	X	X	X
04F16	UART2 transmission interrupt control register		X	X	X	X	X	X	X	X
05016	UART2 reception interrupt control register		X	X	X	X	X	X	X	X
37616	UART2 special mode register 2		X	X	X	X	X	X	X	X
37716	UART2 special mode register		X	X	X	X	X	X	X	X
37816	UART2 send/receive mode register		X	X	X	X	X	X	X	X
37916	UART2 transmission speed register		X	X	X	X	X	X	X	X
37A16	UART2 transmission buffer register		X	X	X	X	X	X	X	X
37B16			X	X	X	X	X	X	X	X
37C16	UART2 send/receive control register 0		X	X	X	X	X	X	X	X
37D16	UART2 send/receive control register 1		X	X	X	X	X	X	X	X
37E16	UART2 reception buffer		X	X	X	X	X	X	X	X
37F16			X	X	X	X	X	X	X	X
3ED16	Port 7								X	X
3EF16	Port 7 direction register								X	X

Note:When registers combining access denied bits and authorization bits are written, use bset, bclr commands, etc., and use the 1 command write-operate and complete method.

4.2 Memory Used

RAM size	data	14 bytes
	stack	9 bytes
	Interrupt stack	16 bytes
ROM size		1,239 bytes

Functions provided by the soft I^2C bus

By including the i2c.h, the following functions can be used:

void iic_ini(unsigned char ID, unsigned char *E2PROM)

Functionality: Performs initialization processing for transmission with the I^2C bus. The device operates as a slave if this processing is completed and if the status is such that an interrupt is authorized. On the other hand, the device will operate as a master if the following functions which start master transmitting/ receiving are called:
Stack used: 5 bytes
Arguments: ID - Set own address
Set own address as the last 7 bits of addresses other than special addresses.
*E2PROM - Set the leading address RAM used as for E2PROM. Prepare an **unsigned char**-type array as a global variable 128 bytes in size with the user program, and set that leading address.
Return values: None
Other: Interrupts are denied while this is being executed.

unsigned char iic_stop(void)

Functionality: Halts I^2C bus send/receive functions.
I^2C bus send/receive functions can be halted with functions during slave transmitting/receiving. However, transmitting/receiving functionality cannot be halted. As soon as the function is used to halt transmitting/receiving, transmitting/receiving operations cannot be started, even if the function starting of master transmitting and receiving functionality is called, since I^2C bus communication operations are not taking place. You will need to call **iic_ini** if you want to restart transmitting/receiving functionality.
Stack used: 5 bytes
Arguments: None
Return values: 0 - I^2C bus transmitting/receiving functionality was able to be halted.
1 - - I^2C bus transmitting/receiving functionality could not be halted since the device itself is a master.
Other: Interrupts are denied while this is being executed.

unsigned char iic_mr_start(unsigned char MR_LNG,
unsigned char*MR_DATA,
unsigned char MR_SLAVE)

Functionality: Starts master reception.
You will need to put the I^2C bus in usable status with **iic_ini** to use this function.
Stack used: 9 bytes.
Arguments: **MR_LNG**: Designates data length received by master. (Note that 0 means 256 bytes.)
***MR_DATA**: Designates leading address stored by data received by the master.
MR_SLAVE: Designates device which you want to have receive data.

Indicate the address of the slave device in the latter 7 bits.
The upper 1 bit is not reflected.

Return values: 0 è Reception by the master has begun.
1 è Reception by the master has not begun due the bus being busy. (Does

1. As in a mathematical function [the translator]

not perform retry.)
Other: Interrupts are denied while this is being executed.

unsigned char iic+mw_start(unsigned char MW_LNG,
unsigned char *MW_DATA,
unsigned char MW_SLAVE

Functionality: Starts master reception
You will need to put the I^2C bus in usable status with **iic_ini** to use this function.
Stack usage: 9 bytes
Arguments: **MR_LNG**: Designates data length received by master. (Note that 0 means 256 bytes.)
***MR_DATA**: Designates leading address stored by data received by the master.
MR_SLAVE: Designates device which you want to have receive data.
Indicate the address of the slave device in the latter 7 bits.
The upper 1 bit is not reflected.

Return values: 0 è Reception by the master has begun.
1 è Reception by the master has not begun due the bus being busy. (Does not perform retry.)
Other: Interrupts are denied while this is being executed.

Soft IIC Bus Functions Made by the User

The following functions are called by the soft I^2C bus as arguments for the transmit/ receive status and its data number.

The user must provide these functions for the soft I^2C bus.

void iic_mw_end(unsigned char MW_STATUS,unsigned char MW_LNG

Functionality: Uses the following two arguments to inform the user of the master's transmission complete status:
Stack usage: 16 bytes (interrupt stack) + auto variable in the function itself.
Arguments: **MW_STATUS**: Indicates completion of master's transmission.
0 = Master transmission completed normally.
1 = Slave returned NACK in the first byte.
2 = Slave returned NACK in data region.
3 = BUS competition detected and master operation completed.
4 = Improper start condition detected.
5 = Improper stop condition detected.
MW_LNG: Indicates the master reception data number.
Return values: None
Other: Calls are made from within soft I^2C bus interrupt processing.

void iic_mr_end(unsigned char MR_STATUS,unsigned char MR_LNG

Functionality: Uses the following two arguments to inform the user of the master's transmission complete status:
Stack usage: 16 bytes (interrupt stack) + auto variable in the function itself.
Arguments: **MR_STATUS**: Indicates completion of master's reception.
0 = Master reception completed normally.
1 = Slave returned NACK in the first byte.
2 = Slave returned NACK in data region.
3 = BUS competition detected and master operation completed.
4 = Improper start condition detected.
5 = Improper stop condition detected.
MW_LNG: Indicates the master reception data number.
Return values: None
Other: Calls are made from within soft I^2C bus interrupt processing.

4.4 Communication Methods

4.4.1 Preparation

In the program operation initial stage, the following **iic_ini** calls are needed in the initial routine in order to perform I^2C bus communications.

At this point, it makes no difference whether the I flag is 0 or 1, and interrupts are not permitted during the execution of **iic_ini**. 2 arguments are transferred when **iic_ini** is called. The first argument indicates the device's own address, while those arguments after this one receive slave assignments from other masters. Do not designate the functional address in the device's own address.

The 2^{nd} argument designates the RAM region leading address used in I^2C bus salve transmission/ reception. Therefore, you will need to set up this region before issuing the **iic_ini** call. Reserve 128 bytes in the static variable region near attributes of the region.

Example:

```
//global variable declarator

unsigned char iic_ram[128]          //RAM region used in I2C slave transmission and recep-
tion.
                                    //global variable which becomes like a static variable

System initialization processing
iic_ini(0x54,iic_ram);              //I2C=BUS initialization
                                    //Own address 1010100b
                                    //iic_ram leading address

asm("fset I");                      // I-flag set
```

4.4.2 Master Communications

(1) Master Transmission

iic_mw_start is called to start master transmission. Three arguments are transferred when **iic_mw_start** is called. The first argument sets the transmission length. A "0" setting indicates the maximum transmission data length, sending 256 bytes.

The second argument designates the leading address of the transmission data storage destination. If this transmission data storage destination is not released until the master transmission is complete, the transmission data storage destination can be assigned anywhere in the near attribute RAM region.

The third argument designates the transmission counterpart address. Do not designate the functional address in the transmission counterpart address. Also, **iic_mw_start** has return values, a "0" is returned when master transmission starts, and a "1" is returned if it does not start.

The following example shows a 5-byte data transmission from **iic_ram** in a slave device which has an address called 55_{16}:

Example:

```
if (iic_mw_start(0x05,iic_ram,0x55) !=0) { //master transmission failure confirmation processing
}
else {
        //master transmission start confirmation processing
}
```

When the master transmission is complete, the soft I^2C bus calls **iic_mw_end**. The user must create this function. Two arguments are transferred when the soft I^2C bus calls **iic_mw_end**. The first argument indicates completion of master transmission. Section 4.3 shows the status contents.

The second argument indicates the actual number of data transmitted.

The following is an example of **iic_mw_end**.

(2) Master Reception

iic_mw_start is called to start master transmission. Three arguments are transferred when **iic_mw_start** is called. The first argument sets the transmission length. A "0" setting indicates the maximum transmission data length, sending 256 bytes.

The second argument designates the leading address of the transmission data storage destination. If this transmission data storage destination is not released until the master transmission is complete, the transmission data storage destination can be assigned anywhere in the near attribute RAM region.

The third argument designates the transmission counterpart address. Do not designate the functional address in the transmission counterpart address. Also, **iic_mw_start** has return values, a "0" is returned when master transmission starts, and a "1" is returned if it does not start.

The following example shows a 5-byte data transmission from **iic_ram** in a slave device which has an address called 55_{16}:

Example:

```
if (iic_mw_start(0x05,iic_ram,0x55) !=0) {
        //master transmission failure confirmation processing
}
else {
        //master transmission start confirmation processing
}
```

[translator's note: this page duplicates page 70]

When the master reception is complete, the soft I^2C bus calls **iic_mr_end**. The user must create this function. Two arguments are transferred when the soft I^2C bus calls **iic_mr_end**. The first argument indicates completion of master transmission.
The second argument shows the actual number of data received.

The following is an example of **iic_mr_end**.

4.4.3 Slave Communications

Is controlled the same as the 24LC01 (E2PROM supporting 128-byte I^2C bus).
However, it differs in that device address (with the exception of the functional address) can be freely chosen, and in that the address increment is 128-bytes linear.

5. Evaluation

We recommend that when you evaluate the device that you set the 3rd line and following of the **i2cbus.a30** file as follows and that you use a logic analyzer to check operational performance. Set the unused ports in lines 5 and 6, and set the ports stated here to output with your user program initialization. (Port 9 in this example.)

The execution time of the software varies according to the system with which it is integrated, so use this method to check actual execution time (interrupt prohibited time).

Keep safety first in your circuit designs!

- Mitsubishi Electric Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of non-flammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Mitsubishi semiconductor product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Mitsubishi Electric Corporation or a third party.
- Mitsubishi Electric Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
- All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Mitsubishi Electric Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor for the latest product information before purchasing a product listed herein.
  The information described here may contain technical inaccuracies or typographical errors. Mitsubishi Electric Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
  Please also pay attention to information published by Mitsubishi Electric Corporation by various means, including the Mitsubishi Semiconductor home page (http://www.mitsubishichips.com).
- When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Mitsubishi Electric Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Mitsubishi Electric Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- The prior written approval of Mitsubishi Electric Corporation is necessary to reprint or reproduce in whole or in part these materials.
- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
  Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
- Please contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semicon ductor product distributor for further details on these materials or the products con tained therein.