Serial Communications

Chapter 7

Renesas Electronics America Inc.
Embedded Systems using the RX63N

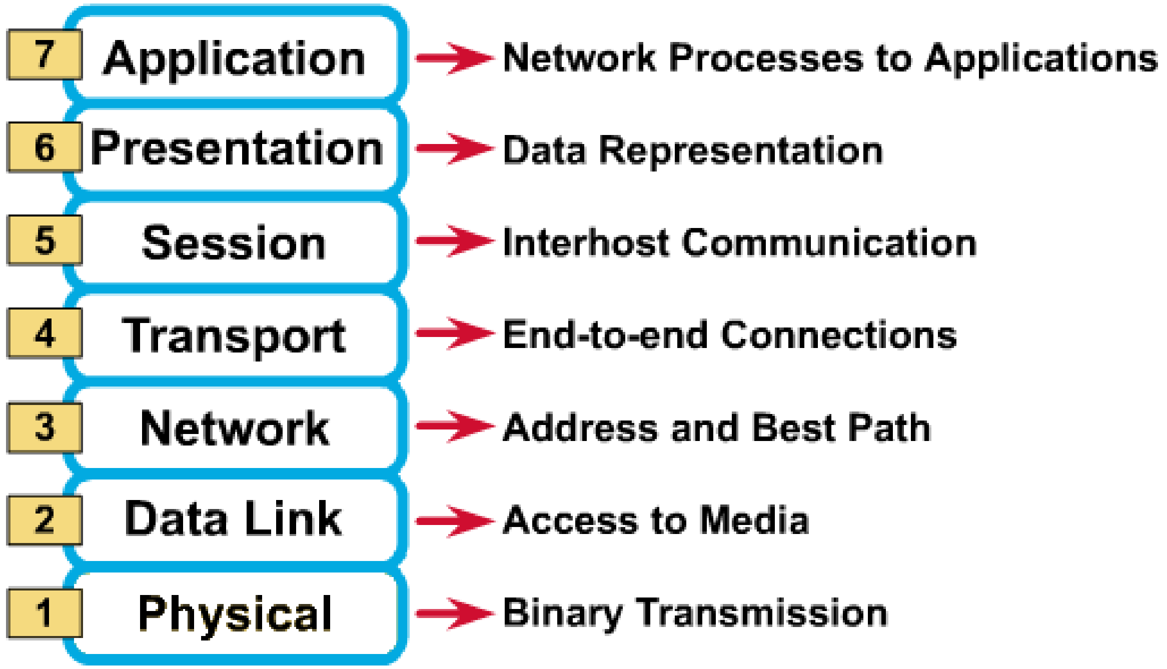
Rev. 1.0

In this chapter we will learn:

- General Communications
- Serial Communications
  - RS232 Standard
  - UART Operation
  - RSPI Operation
  - I^2C

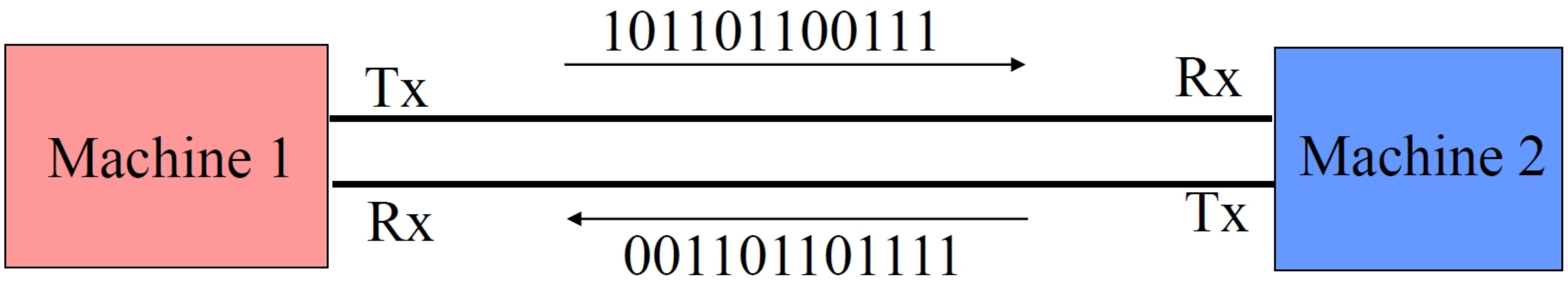
Data Communications

- The International Organization for Standardization (ISO) has established a reference model which organizes network function in seven layers.
- Each layer provides a service to the layer above and communicates with the same layer's software or hardware on other computers.
- Layers 5-7 are concerned with services for the applications.
- Layers 1-4 are concerned with the flow of data from end to end through the network.


7 Application → Network Processes to Applications
6 Presentation → Data Representation
5 Session → Interhost Communication
4 Transport → End-to-end Connections
3 Network → Address and Best Path
2 Data Link → Access to Media
1 Physical → Binary Transmission

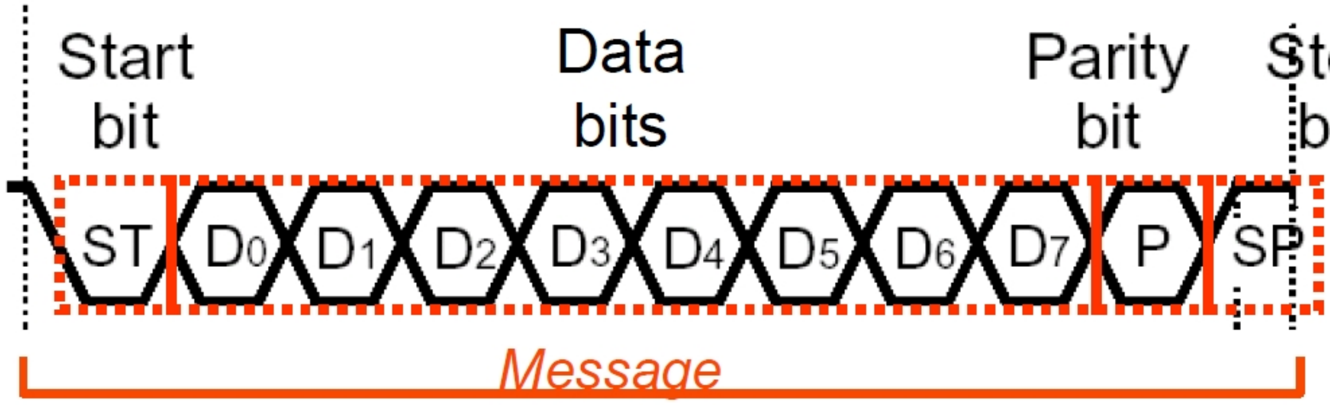
Physical Layer (1) – Serial Communications

- The basic premise of serial communications is that one or two wires are used to transmit digital data.
  - An extra ground reference wire is also needed.
- Communication can be one way or two way, however, most often two way, hence the need for two communication wires.
- Other wires are often used for other aspects of the communications such as: ground, “clear-to-send,” “data terminal ready,” etc.



Serial Communication Basics

- Send one bit of the message at a time.
- Message field consists of:
  - Start bit (one bit)
  - Data (LSB first or MSB, and size – 7, 8, 9 bits)
  - Optional parity bit is used to make total number of ones in data even or odd
  - Stop bit (one or two bits)
- All devices on network or link must use the same communications parameters, such as speed, for example.



Bit Rate vs. Baud Rate

- **Bit Rate:**
  - How many **data bits** are transmitted per second
- **Baud Rate:**
  - How many **symbols** are transmitted per second
  - A symbol may be represented by a voltage level, a sine wave’s frequency or phase, etc.
- Extra symbols (channel changes) may be inserted for framing, error detection, acknowledgment, etc. These reduce the bit rate.
- A single symbol might encode more than one bit. This increases the bit rate.

UART Concepts

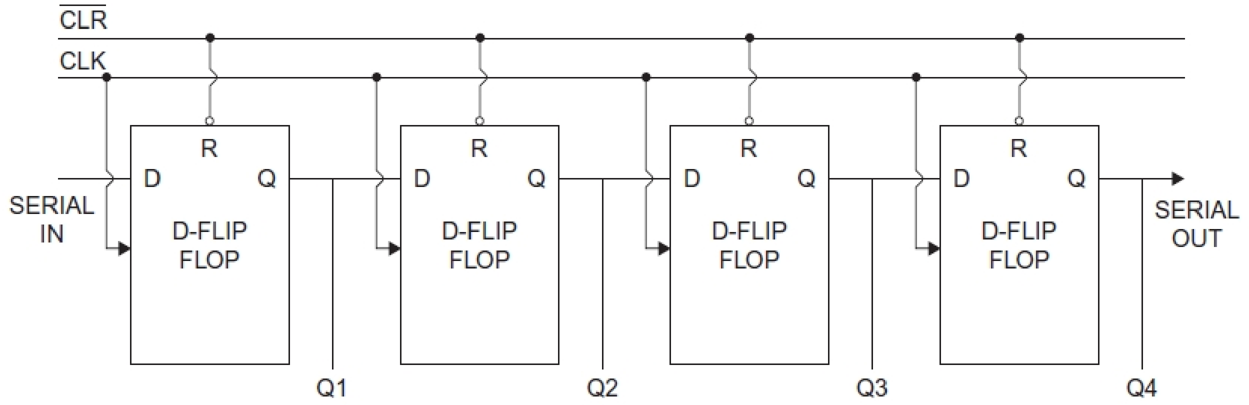
- UART stands for Universal Asynchronous Receiver/Transmitter
- **Universal**
  - Configurable to fit protocol requirements
- **Asynchronous**
  - No clock line needed to de-serialize bits
- **Receiver / Transmitter**
  - Signals can be both received and transmitted

General UART Concepts

- The UART subsystem consists of:
- Two shift registers
  - Parallel to serial for transmit
  - Serial to parallel for receive
- Programmable clock source
  - Clock must run at 16x desired bit rate
- Error detection
  - Detect bad stop or parity bits
  - Detect receive buffer overwrite
- Interrupt generators
  - Character received
  - Character transmitted, ready to send another

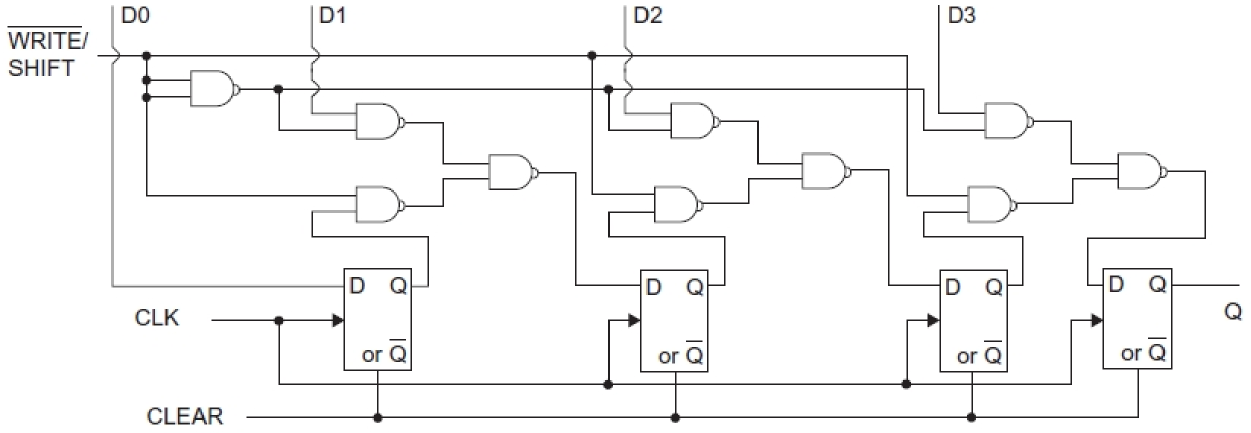
General UART Concepts cont.

- Here is a circuit representation of a Serial Input Parallel Output (SIPO) shift register,



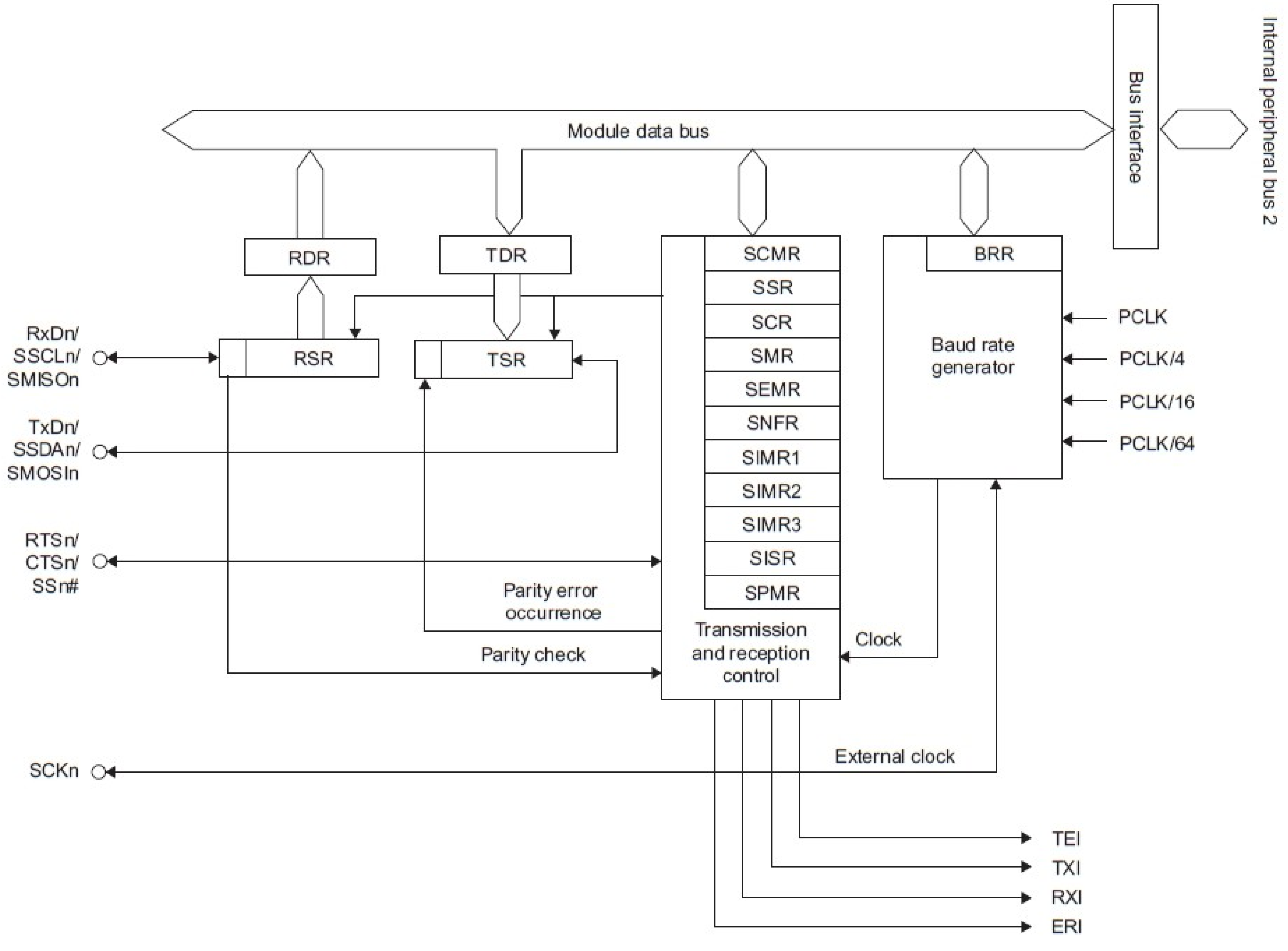
[1]

- and a Parallel Input Serial Output (PISO) shift register.



[1]

Block Diagram of RX63N Serial Communications Interface



RSR:	Receive shift register	SSR:	Serial status register	SIMR1:	I²C mode register 1
RDR:	Receive data register	SCMR:	Smart card mode register	SIMR2:	I²C moder register 2
TSR:	Transmit shift register	BRR:	Bit rate register	SIMR3:	I²C moder register 3
TDR:	Transmit data register	SEMR:	Serail extended mode register	SISR:	I²C status register
SMR:	Serial mode register	SNFR:	Noise filter setting register	SPMR:	SPI mode register
SCR:	Serial control register				

[2]

SCI in UART Mode

- In order to communicate from the RX63N chip, you need to set up several registers, including:
  - Mode
  - Speed
  - Parity
  - Stop bits
  - Configuration
- There are two primary “Data Registers”:
  - SCIx.RDR (Receive Data Register)
  - SCIx.TDR (Transmit Data Register)

Serial Mode Register (SMR)

- This special function register is concerned with operational variations of the UART.

	b7	b6	b5	b4	b3	b2	b1	b0
	CM	CHR	PE	PM	STOP	MP	CKS[1:0]	
Value after reset:	0	0	0	0	0	0	0	0

[2]

- The bits related to the SMR are:
  - CKS: transmission speed
  - MP: Multi processor (set to 0)
  - STOP: Stop bits
  - PM: Parity mode
  - PE: Parity Enable
  - CHR: Length of data
  - CM: Communications mode
- The following slide contains the values each bit can be set to.

Serial Mode Register (SMR) cont.

Bit	Symbol	Bit Name	Description	R/W
b1, b0	CKS[1:0]	Clock Select	b1 b0 0 0: PCLK clock (n = 0)*1 0 1: PCLK/4 clock (n = 1)*1 1 0: PCLK/16 clock (n = 2)*1 1 1: PCLK/64 clock (n = 3)*1	R/W*4
b2	MP	Multi-Processor Mode	(Valid only in asynchronous mode) 0: Multi-processor communications function is disabled 1: Multi-processor communications function is enabled	R/W*4
b3	STOP	Stop Bit Length	(Valid only in asynchronous mode) 0: 1 stop bit 1: 2 stop bits	R/W*4
b4	PM	Parity Mode	(Valid only when the PE bit is 1 in asynchronous mode) 0: Selects even parity 1: Selects odd parity	R/W*4
b5	PE	Parity Enable	(Valid only in asynchronous mode) • When transmitting 0: Parity bit addition is not performed 1: The parity bit is added • When receiving 0: Parity bit checking is not performed 1: The parity bit is checked	R/W*4
b6	CHR	Character Length	(Valid only in asynchronous mode) 0: Selects 8 bits as the data length*2 1: Selects 7 bits as the data length*3	R/W*4
b7	CM	Communications Mode	0: Asynchronous mode 1: Clock synchronous mode	R/W*4

Note 1. n is the decimal notation of the value of n in BRR (see section 35.2.9, Bit Rate Register (BRR)).
Note 2. In clock synchronous mode, this bit setting is invalid and a fixed data length of 8 bits is used.
Note 3. LSB-first is fixed and the MSB (bit 7) in TDR is not transmitted in transmission.
Note 4. Writable only when TE in SCR = 0 and RE in SCR = 0 (both serial transmission and reception are disabled).

[2]

Serial Control Register (SCR)

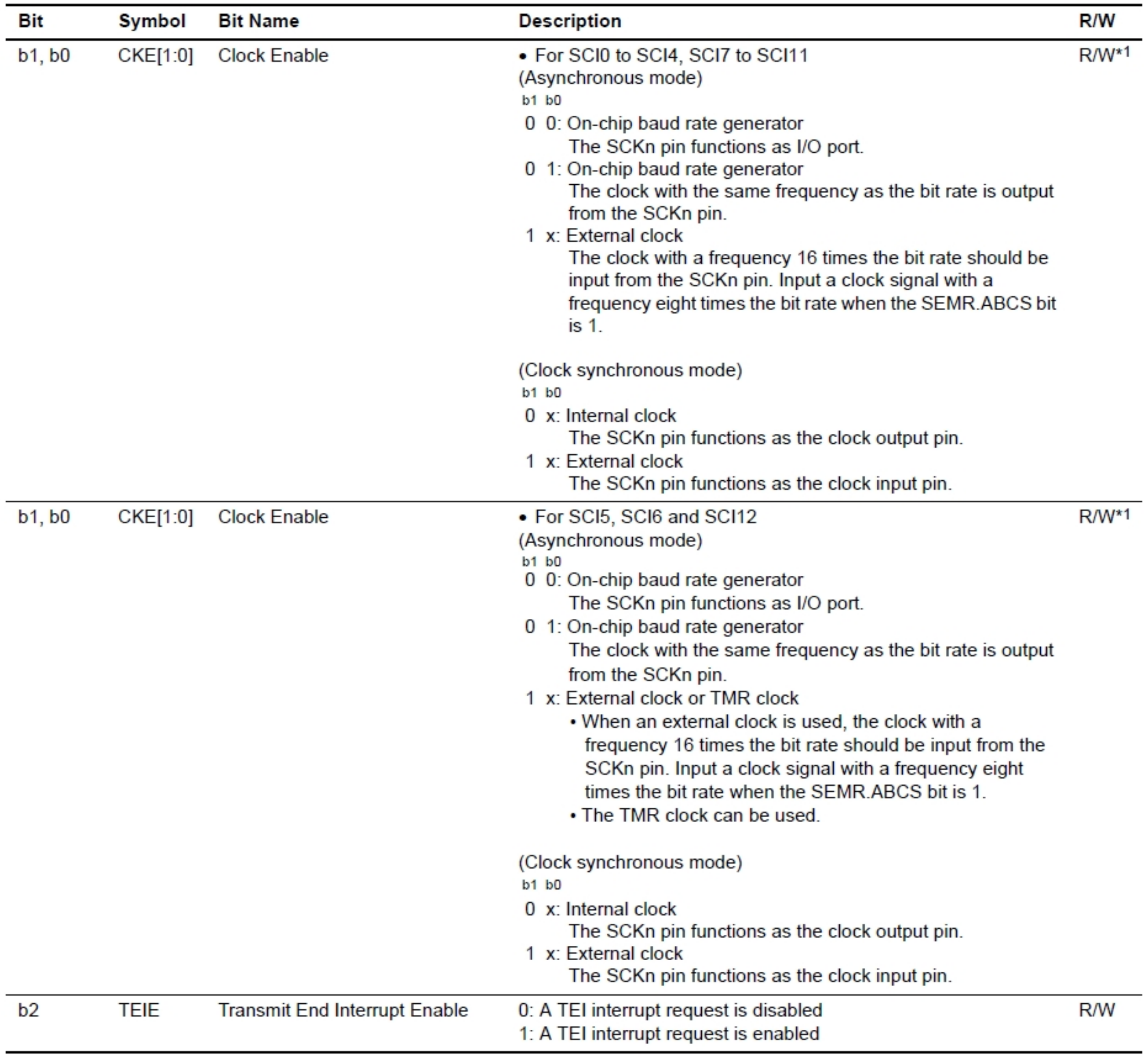
- This register is responsible for controlling whether the Serial Communications Interface is turned on or off, the choice of input clock to the shift register, and function of the SCK pin

	b7	b6	b5	b4	b3	b2	b1	b0
	TIE	RIE	TE	RE	MPIE	TEIE	CKE[1:0]	
Value after reset:	0	0	0	0	0	0	0	0

[2]

- The following two slides contain the values each bit can be set to, as well as their description.

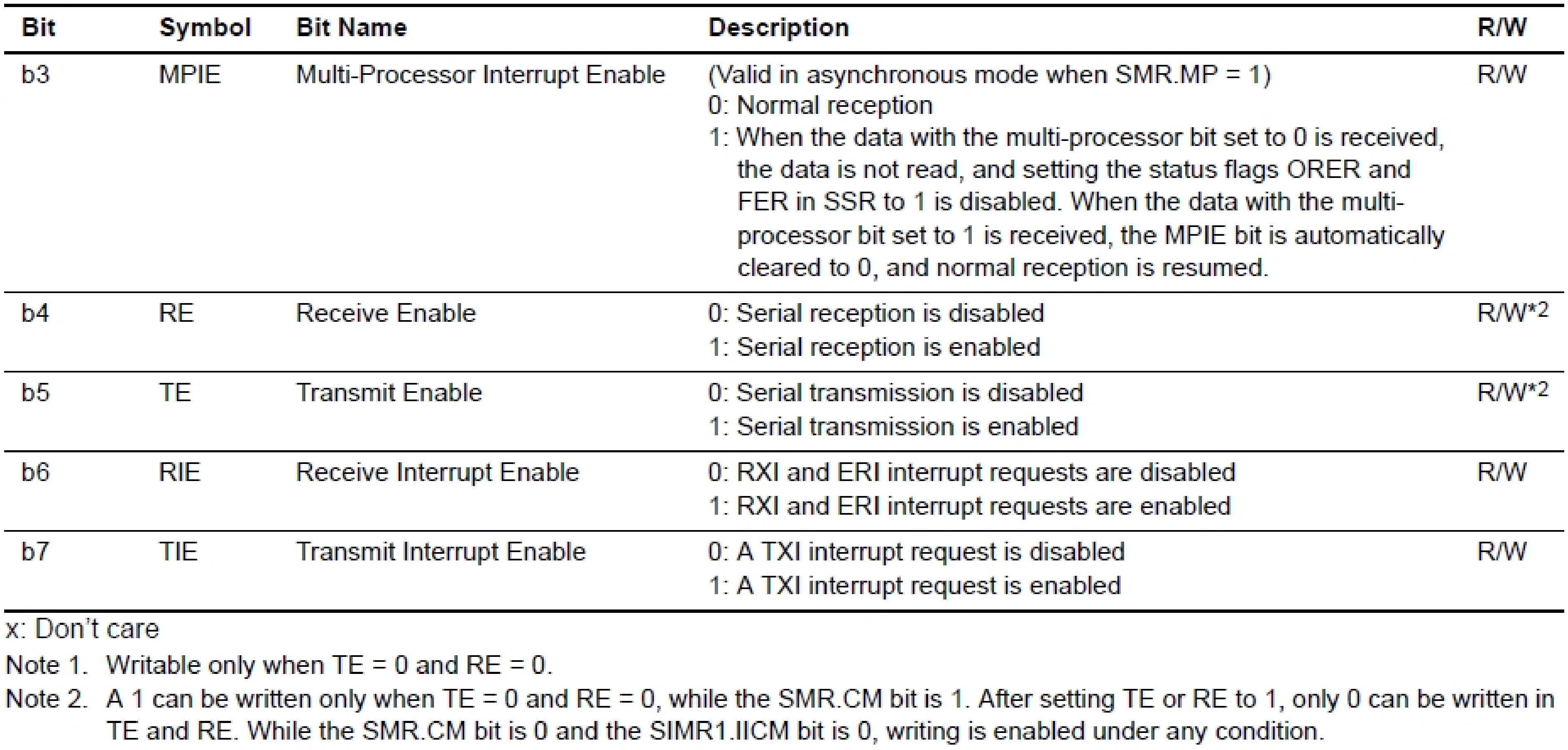
Serial Control Register (SCR) cont.



Bit	Symbol	Bit Name	Description	R/W
b1, b0	CKE[1:0]	Clock Enable	• For SCI0 to SCI4, SCI7 to SCI11 (Asynchronous mode) b1 b0 0 0: On-chip baud rate generator The SCKn pin functions as I/O port. 0 1: On-chip baud rate generator The clock with the same frequency as the bit rate is output from the SCKn pin. 1 x: External clock The clock with a frequency 16 times the bit rate should be input from the SCKn pin. Input a clock signal with a frequency eight times the bit rate when the SEMR.ABCS bit is 1. (Clock synchronous mode) b1 b0 0 x: Internal clock The SCKn pin functions as the clock output pin. 1 x: External clock The SCKn pin functions as the clock input pin.	R/W*1
b1, b0	CKE[1:0]	Clock Enable	• For SCI5, SCI6 and SCI12 (Asynchronous mode) b1 b0 0 0: On-chip baud rate generator The SCKn pin functions as I/O port. 0 1: On-chip baud rate generator The clock with the same frequency as the bit rate is output from the SCKn pin. 1 x: External clock or TMR clock • When an external clock is used, the clock with a frequency 16 times the bit rate should be input from the SCKn pin. Input a clock signal with a frequency eight times the bit rate when the SEMR.ABCS bit is 1. • The TMR clock can be used. (Clock synchronous mode) b1 b0 0 x: Internal clock The SCKn pin functions as the clock output pin. 1 x: External clock The SCKn pin functions as the clock input pin.	R/W*1
b2	TEIE	Transmit End Interrupt Enable	0: A TEI interrupt request is disabled 1: A TEI interrupt request is enabled	R/W

[2]

Serial Control Register (SCR) cont.



Bit	Symbol	Bit Name	Description	R/W
b3	MPIE	Multi-Processor Interrupt Enable	(Valid in asynchronous mode when SMR.MP = 1) 0: Normal reception 1: When the data with the multi-processor bit set to 0 is received, the data is not read, and setting the status flags ORER and FER in SSR to 1 is disabled. When the data with the multi-processor bit set to 1 is received, the MPIE bit is automatically cleared to 0, and normal reception is resumed.	R/W
b4	RE	Receive Enable	0: Serial reception is disabled 1: Serial reception is enabled	R/W*2
b5	TE	Transmit Enable	0: Serial transmission is disabled 1: Serial transmission is enabled	R/W*2
b6	RIE	Receive Interrupt Enable	0: RXI and ERI interrupt requests are disabled 1: RXI and ERI interrupt requests are enabled	R/W
b7	TIE	Transmit Interrupt Enable	0: A TXI interrupt request is disabled 1: A TXI interrupt request is enabled	R/W

x: Don't care

Note 1. Writable only when TE = 0 and RE = 0.

Note 2. A 1 can be written only when TE = 0 and RE = 0, while the SMR.CM bit is 1. After setting TE or RE to 1, only 0 can be written in TE and RE. While the SMR.CM bit is 0 and the SIMR1.IICM bit is 0, writing is enabled under any condition.

[2]

Serial Status Register (SSR)

- The SSR is a read only register which indicates the status of the currently received byte over the corresponding SCI.

	b7	b6	b5	b4	b3	b2	b1	b0
	—	—	ORER	FER	PER	TEND	MPB	MPBT
after reset:	x	x	0	0	0	1	0	0

[2]

- **TEND:**
  - This flag is set at the end of transmission of a byte from the TDR or in case the serial transmission is disabled.
- **PER:**
  - Parity error flag.
- **FER:**
  - This flag indicates if there is a framing error.
- **ORER:**
  - Overrun error flag.
- MPB and MPBT bits are multi-processor related.

Setting up the Speed of the Serial Port

- The speed of communications is a combination of:
  - PCLK
  - Bits CKS in the SMR
  - The Bit Rate Register (BRR)
- Formula: $N = \frac{PCLK * 10^6}{64 * 2^{2n-1} * B} - 1$
- *B*=bit rate, *N*=BRR setting, *n*=CKS setting
- If you, for example, want to communicate at 38,400 bps, and your PCLK is 50 MHz, *n* should be set to 0 and *N* should be set to 40.
- SCI0.BRR.BYTE = 40

Error Rate

- Since you cannot get an exact value of xx.0 there is an error rate associated with calculating the bit rate
- Formula: $Error\ (\%) = \left\{\frac{PCLK * 10^6}{B * 64 * 2^{2n-1} * (N+1)} - 1\right\} * 100$
- For example, communication at 38,400 bps, with a PCLK of 50 MHz, *n* set to 0 and *N* set to 40, the percent error will be:

$$Error\ (\%) = \left\{\frac{50 * 10^6}{38400 * 64 * 2^{2*0-1} * (40+1)} - 1\right\} * 100$$

$$Error\ (\%) = -0.75\%$$

Bit Rates and Percent Errors

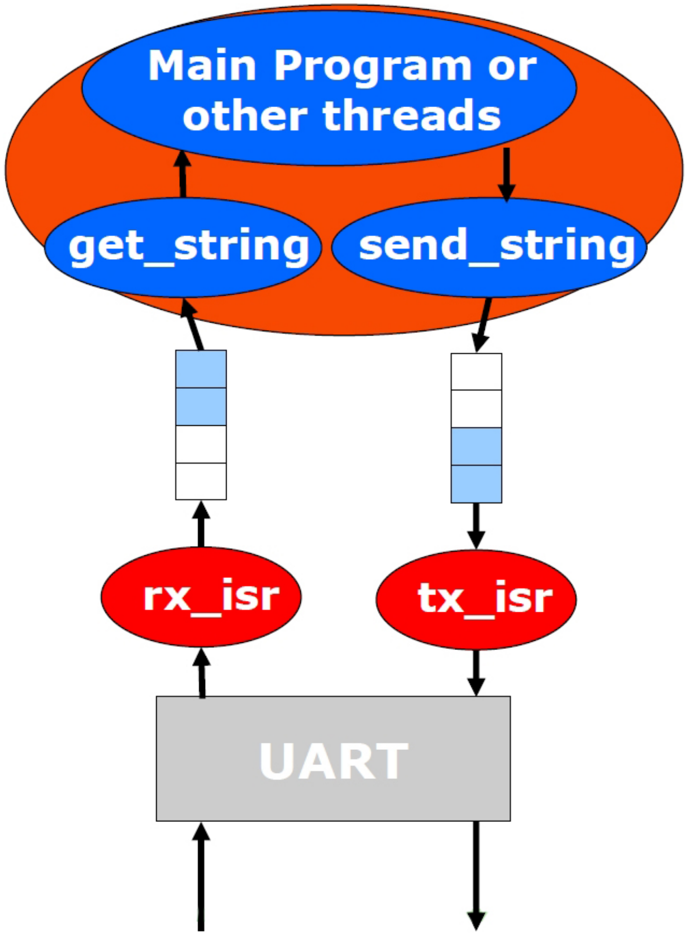
(N = BRR value, n = Clock source setting with CKS bits in SMR).

BIT RATE (bit/s)	OPERATING FREQUENCY PCLK (MHz)											
	25			30			33			50		
	n	N	ERROR (%)	n	N	ERROR (%)	n	N	ERROR (%)	n	N	ERROR (%)
110	3	110	−0.02	3	132	0.13	3	145	0.33	3	221	−0.02
150	3	80	0.47	3	97	−0.35	3	106	0.39	3	162	−0.15
300	2	162	−0.15	2	194	0.16	2	214	−0.07	3	80	0.47
600	2	80	0.47	2	97	−0.35	2	106	0.39	2	162	−0.15
1200	1	162	−0.15	1	194	0.16	1	214	−0.07	2	80	0.47
2400	1	80	0.47	1	97	−0.35	1	106	0.39	1	162	−0.15
4800	0	162	−0.15	0	194	0.16	0	214	−0.07	1	80	0.47
9600	0	80	0.47	0	97	−0.35	0	106	0.39	1	40	−0.77
19200	0	40	−0.76	0	48	−0.35	0	53	−0.54	0	80	0.47
31250	0	24	0.00	0	29	0.00	0	32	0	0	49	0.00
38400	0	19	1.73	0	23	1.73	0	26	−0.54	0	40	−0.77

[1]

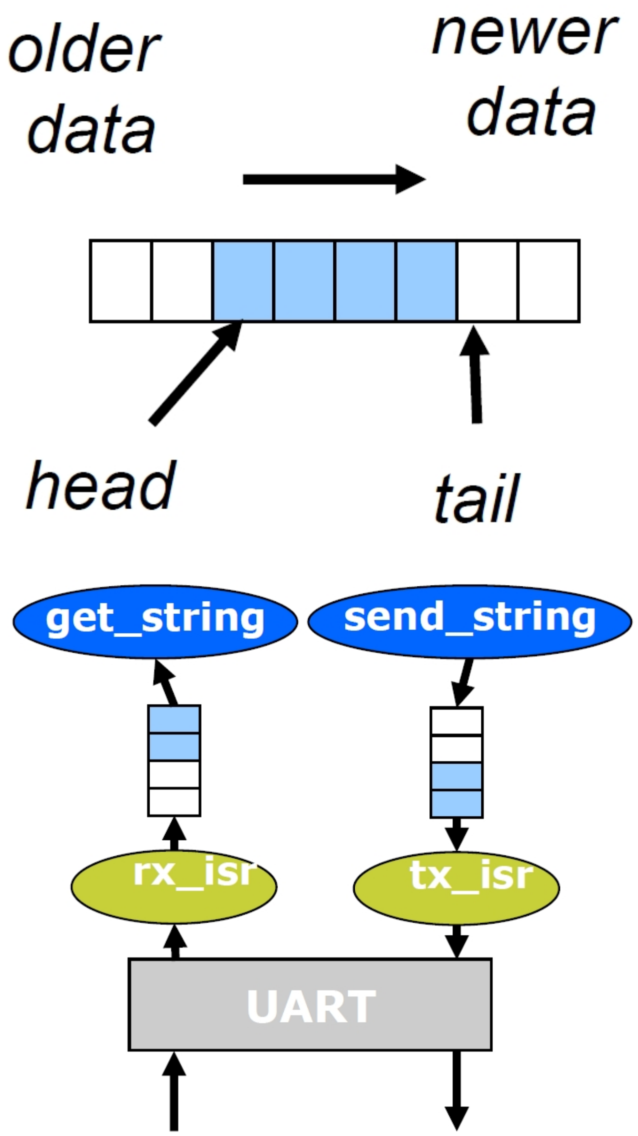
Serial Communications and Interrupts

- There are three separate threads of control in the program:
  - **Main program** and subroutines it calls
  - **Transmit ISR**
    - Executes when UART is ready to send another character
  - **Receive ISR**
    - Executes when UART receives a character
- Problem: Information needs to be buffered between threads:
  - Solution: circular queue with head and tail pointers
  - One for Tx and one for Rx



Code Implementing Queues

- Enqueue at tail
  - tail_ptr points to next free entry
- Dequeue from head
  - head_ptr points to item to remove
- #define the queue size makes it easy to change in the future
- One queue direction
  - Tx ISR unloads tx_q
  - Rx ISR loads rx_q
- Other threads (e.g., main) load tx_q and unload rx_q
- Queue is empty if size == 0
- Queue is full if size == Q_SIZE



Defining the Queues

```
#define Q_SIZE (32)

typedef struct {
    unsigned char Data[Q_SIZE];
    unsigned int Head; //points to oldest data element
    unsigned int Tail; //points to next free space
    unsigned int Size; //quantity of elements in queue
} Q_T;
Q_T tx_q, rx_q;
```

Initialization and Status Inquiries

```
void Q_Init(Q_T * q) {
    unsigned int i;
    for (i=0; i<Q_SIZE; i++)
            q->Data[i] = 0; //simplifies debugging
    q->Head = 0;
    q->Tail = 0;
    q->Size = 0;
}

int Q_Empty(Q_T * q) {
    return q->Size == 0;
}
int Q_Full(Q_T * q) {
    return q->Size == Q_SIZE;
}
```

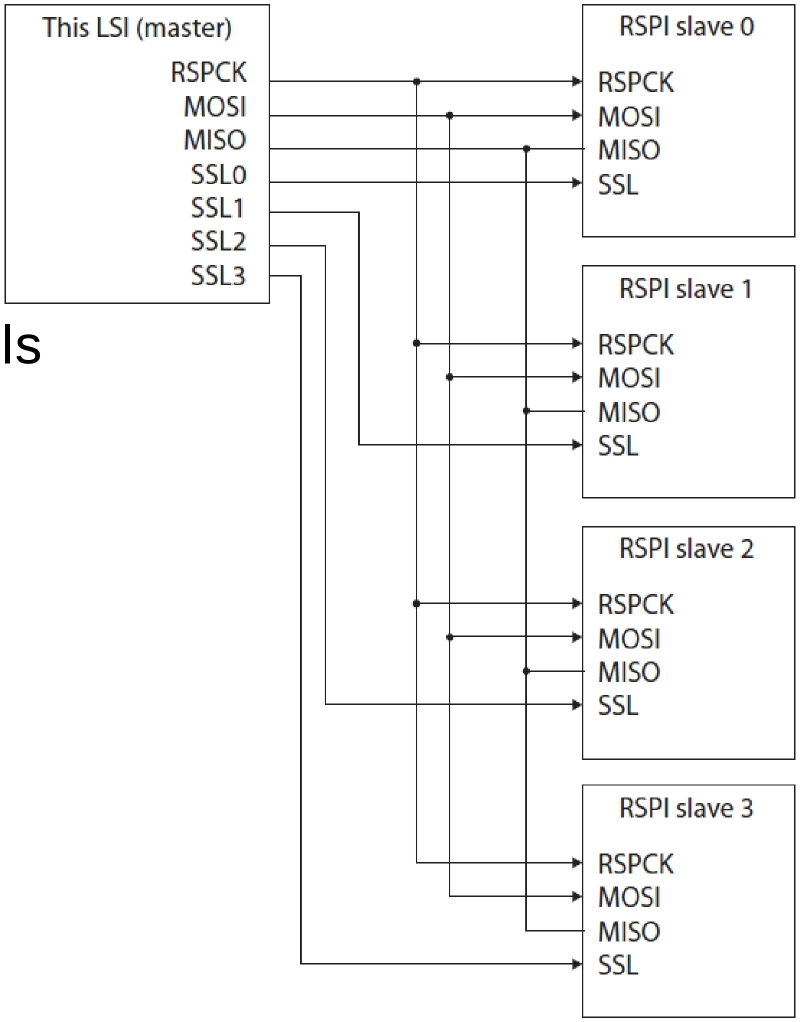
Enqueue and Dequeue

```
// Q_Enqueue – Called by a UART ISR – put a char on the queue
int Q_Enqueue(Q_T * q, unsigned char d) {
    if (!Q_Full(q)) { // Check if queue is full
            q->Data[q->Tail++] = d;
            q->Tail %= Q_SIZE;
            q->Size++;
            return 1; // success
    } else
            return 0; // failure
}

// Q_Dequeue–called by a consumer function–take a char from queue
unsigned char Q_Dequeue(Q_T * q) {
    unsigned char t=0;
    if (!Q_Empty(q)) { //Check to see if queue is empty
            t = q->Data[q->Head];
            q->Data[q->Head++] = 0; // to simplify debugging, clear
            q->Head %= Q_SIZE;
            q->Size--;
    }
    return t;
}
```

Renesas Serial Peripheral Interface (RSPI)

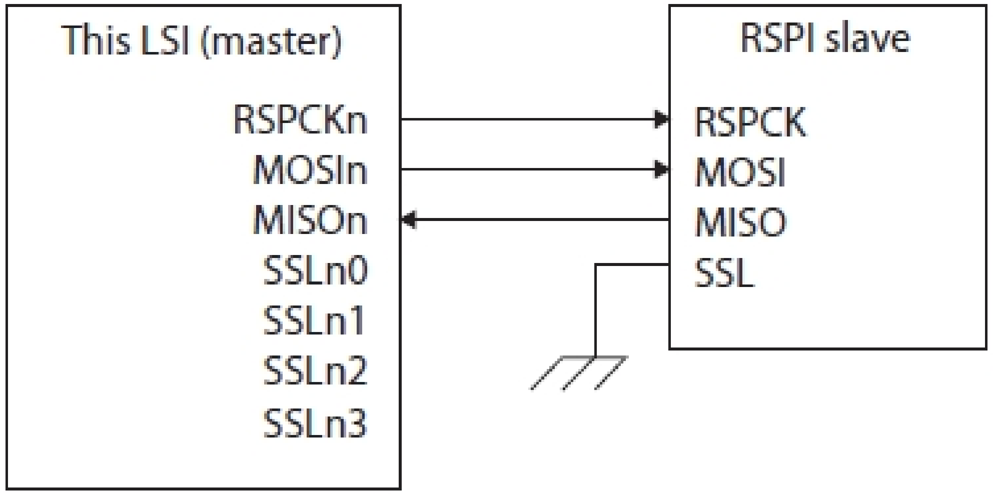
- Synchronous communication
- Can work with as few as three wires, but more needed to access additional devices
- Better method to access peripherals than parallel I/O
- Common clock means you can transmit at 25.0 Mbps
- Intended for very short distances (i.e., on-board)
- The RX63N has three SPI masters



[1]

SPI Details

- Serial Clock (RSPCK)
- Master Out, Slave in (MOSI)
  - Transmission from RX63N
- Master In, Slave Out (MISO)
  - Transmission from peripheral)
- Slave Select (SSLx)
  - Select one of the peripheral devices\
- We will only cover SPI in Slave Mode



[1]

SPI Registers

- Serial Peripheral Control Register (SPCR)
- Serial Peripheral Control Register (SPCR2)
- Slave Select Polarity (SSLP)
- Serial Peripheral Pin Control Register (SPPCR)
- Serial Peripheral Status (SPSR)
- Serial Peripheral Data Register (SPDR)
- Serial Peripheral Bit Rate Register (SPBR)
- Serial Peripheral Clock Delay Register (SPCKD)

Serial Peripheral Control Register (SPCR)

- This register controls the operating mode of the RSPI.

	b7	b6	b5	b4	b3	b2	b1	b0
	SPRIE	SPE	SPTIE	SPEIE	MSTR	MODFEN	TXMD	SPMS
Value after reset:	0	0	0	0	0	0	0	0

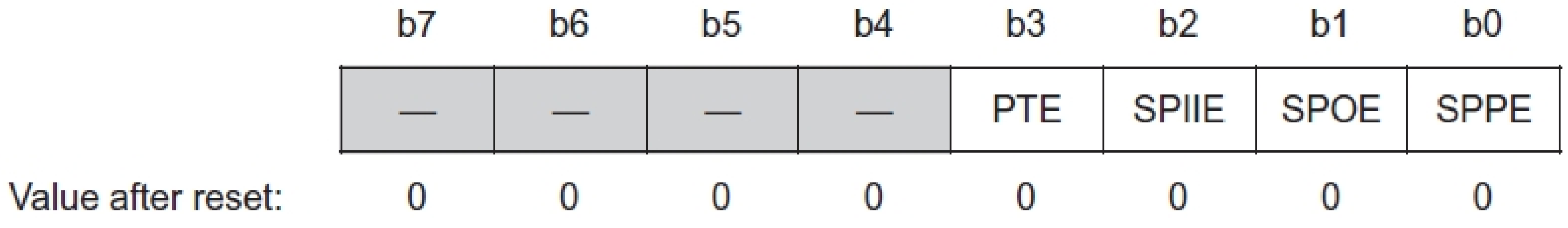
[2]

Bit	Symbol	Bit Name	Description	R/W
b0	SPMS	RSPI Mode Select	0: SPI operation (four-wire method) 1: Clock synchronous operation (three-wire method)	R/W
b1	TXMD	Communications Operating Mode Select	0: Full-duplex synchronous serial communications 1: Serial communications consisting of only transmit operations	R/W
b2	MODFEN	Mode Fault Error Detection Enable	0: Disables the detection of mode fault error 1: Enables the detection of mode fault error	R/W
b3	MSTR	RSPI Master/Slave Mode Select	0: Slave mode 1: Master mode	R/W
b4	SPEIE	RSPI Error Interrupt Enable	0: Disables the generation of RSPI error interrupt requests 1: Enables the generation of RSPI error interrupt requests	R/W
b5	SPTIE	RSPI Transmit Interrupt Enable	0: Disables the generation of RSPI transmit interrupt requests 1: Enables the generation of RSPI transmit interrupt requests	R/W
b6	SPE	RSPI Function Enable	0: Disables the RSPI function 1: Enables the RSPI function	R/W
b7	SPRIE	RSPI Receive Interrupt Enable	0: Disables the generation of RSPI receive interrupt requests 1: Enables the generation of RSPI receive interrupt requests	R/W

[2]

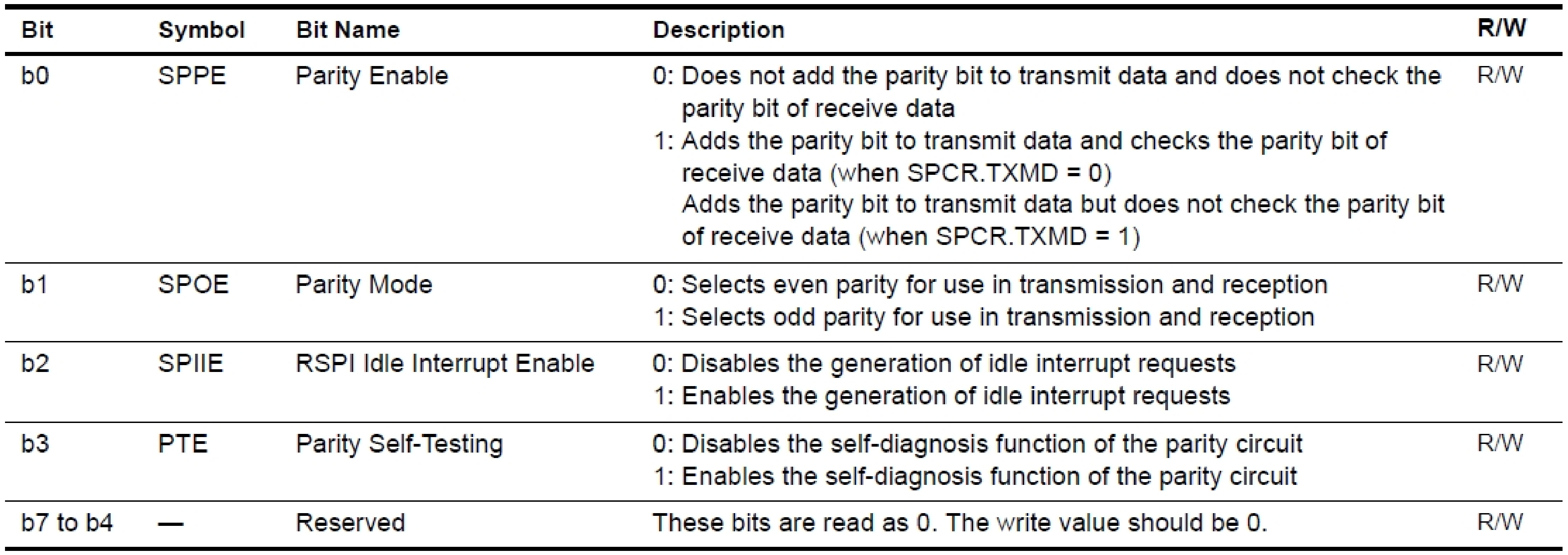
Serial Peripheral Control Register (SPCR2)

- This register adds to the controllability of the operating mode of the Renesas SPI.



	b7	b6	b5	b4	b3	b2	b1	b0
	—	—	—	—	PTE	SPIIE	SPOE	SPPE
Value after reset:	0	0	0	0	0	0	0	0

[2]



Bit	Symbol	Bit Name	Description	R/W
b0	SPPE	Parity Enable	0: Does not add the parity bit to transmit data and does not check the parity bit of receive data 1: Adds the parity bit to transmit data and checks the parity bit of receive data (when SPCR.TXMD = 0) Adds the parity bit to transmit data but does not check the parity bit of receive data (when SPCR.TXMD = 1)	R/W
b1	SPOE	Parity Mode	0: Selects even parity for use in transmission and reception 1: Selects odd parity for use in transmission and reception	R/W
b2	SPIIE	RSPI Idle Interrupt Enable	0: Disables the generation of idle interrupt requests 1: Enables the generation of idle interrupt requests	R/W
b3	PTE	Parity Self-Testing	0: Disables the self-diagnosis function of the parity circuit 1: Enables the self-diagnosis function of the parity circuit	R/W
b7 to b4	—	Reserved	These bits are read as 0. The write value should be 0.	R/W

[2]

Slave Select Polarity (SSLP):

- This register sets the polarity of the slave select lines SSL0 to SSL3 of the Renesas SPI module.

	b7	b6	b5	b4	b3	b2	b1	b0
	—	—	—	—	SSL3P	SSL2P	SSL1P	SSL0P
Value after reset:	0	0	0	0	0	0	0	0

[2]

Bit	Symbol	Bit Name	Description	R/W
b0	SSL0P	SSL0 Signal Polarity Setting	0: SSL0 signal is active low 1: SSL0 signal is active high	R/W
b1	SSL1P	SSL1 Signal Polarity Setting	0: SSL1 signal is active low 1: SSL1 signal is active high	R/W
b2	SSL2P	SSL2 Signal Polarity Setting	0: SSL2 signal is active low 1: SSL2 signal is active high	R/W
b3	SSL3P	SSL3 Signal Polarity Setting	0: SSL3 signal is active low 1: SSL3 signal is active high	R/W
b7 to b4	—	Reserved	These bits are read as 0. The write value should be 0.	R/W

[2]

Serial Peripheral Pin Control Register (SPPCR)

- This register sets the modes of the RSPI pins.

	b7	b6	b5	b4	b3	b2	b1	b0
	—	—	MOIFE	MOIFV	—	SPOM	SPLP2	SPLP
Value after reset:	0	0	0	0	0	0	0	0

[2]

Bit	Symbol	Bit Name	Description	R/W
b0	SPLP	RSPI Loopback	0: Normal mode 1: Loopback mode (reversed transmit data = receive data)	R/W
b1	SPLP2	RSPI Loopback 2	0: Normal mode 1: Loopback mode (transmit data = receive data)	R/W
b3, b2	—	Reserved	These bits are read as 0. The write value should be 0.	R/W
b4	MOIFV	MOSI Idle Fixed Value	0: The level output on the MOSIn pin during MOSI idling corresponds to 0. 1: The level output on the MOSIn pin during MOSI idling corresponds to 1.	R/W
b5	MOIFE	MOSI Idle Value Fixing Enable	0: MOSI output value equals final data from previous transfer 1: MOSI output value equals the value set in the MOIFV bit	R/W
b7, b6	—	Reserved	These bits are read as 0. The write value should be 0.	R/W

[2]

Serial Peripheral Status (SPSR)

- This register is an indicator of the current operating status of the RSPI.

	b7	b6	b5	b4	b3	b2	b1	b0
	—	—	—	—	PERF	MODF	IDLNF	OVRF
Value after reset:	x	0	x	0	0	0	0	0

[2]

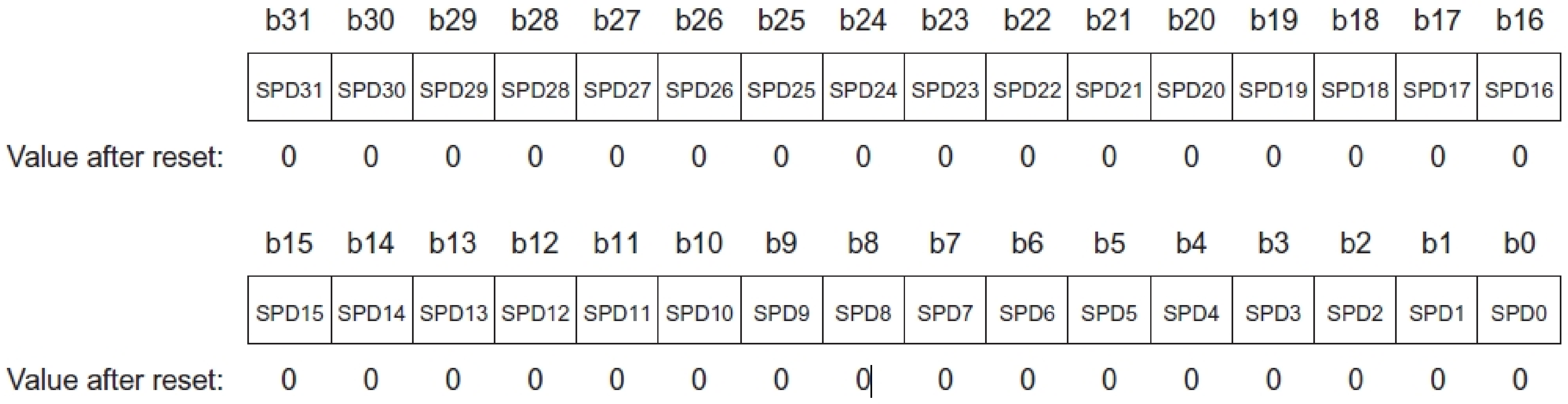
Bit	Symbol	Bit Name	Description	R/W
b0	OVRF	Overrun Error Flag	0: No overrun error occurs 1: An overrun error occurs	R/(W) *1
b1	IDLNF	RSPI Idle Flag	0: RSPI is in the idle state 1: RSPI is in the transfer state	R
b2	MODF	Mode Fault Error Flag	0: No mode fault error occurs 1: A mode fault error occurs	R/(W) *1
b3	PERF	Parity Error Flag	0: No parity error occurs 1: A parity error occurs	R/(W) *1
b4	—	Reserved	This bit is read as 0. The write value should be 0.	R/W
b5	—	Reserved	The read value is undefined. The write value should be 1.	R/W
b6	—	Reserved	This bit is read as 0. The write value should be 0.	R/W
b7	—	Reserved	The read value is undefined. The write value should be 1.	R/W

[2]

Note 1. Only 0 can be written to clear the flag after reading 1.

Serial Peripheral Data Register (SPDR)

- This register contains data to be transmitted and data received over the SPI channel.



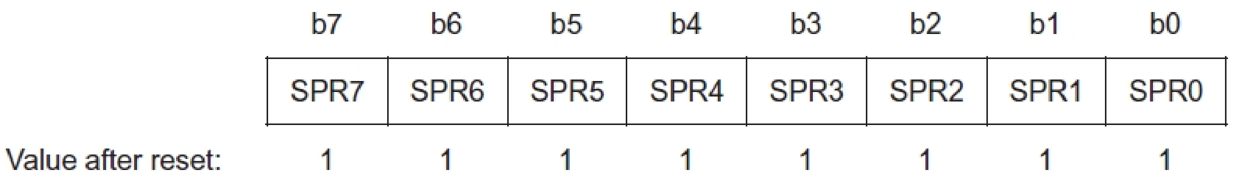
	b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
	SPD31	SPD30	SPD29	SPD28	SPD27	SPD26	SPD25	SPD24	SPD23	SPD22	SPD21	SPD20	SPD19	SPD18	SPD17	SPD16
Value after reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
	SPD15	SPD14	SPD13	SPD12	SPD11	SPD10	SPD9	SPD8	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0
Value after reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

[2]

Serial Peripheral Bit Rate Register (SPBR)

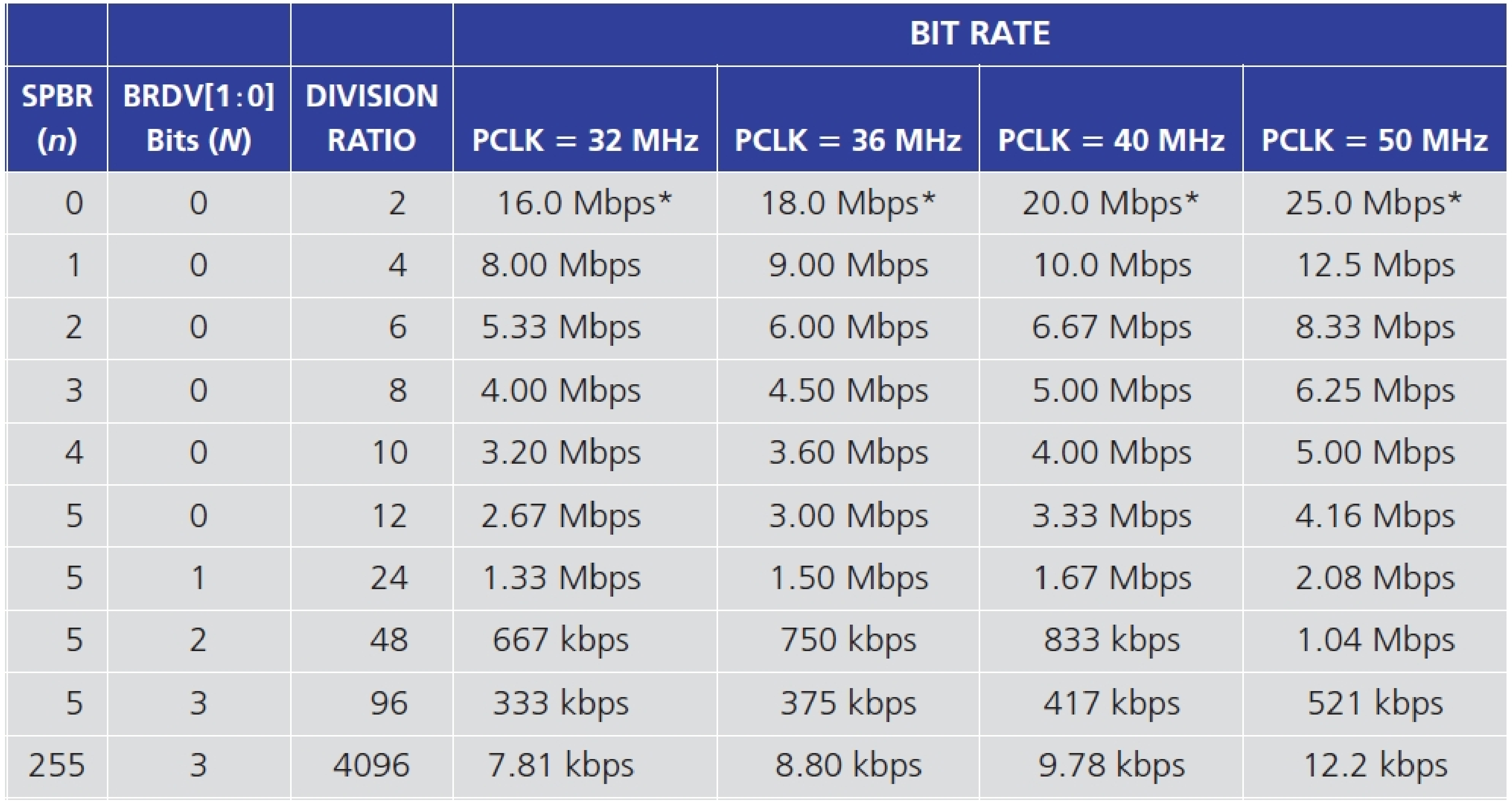
- The value of this register determines the rate of data transfer.



	b7	b6	b5	b4	b3	b2	b1	b0
	SPR7	SPR6	SPR5	SPR4	SPR3	SPR2	SPR1	SPR0
Value after reset:	1	1	1	1	1	1	1	1

$$\text{Bit rate} = \frac{f\,(\text{PCLK})}{2 \times (n + 1)\, 2^{N}}$$

[2]

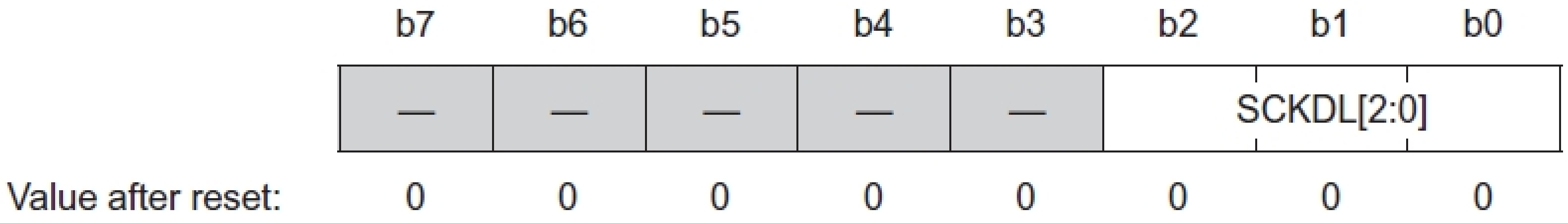


			BIT RATE			
SPBR (*n*)	BRDV[1:0] Bits (*N*)	DIVISION RATIO	PCLK = 32 MHz	PCLK = 36 MHz	PCLK = 40 MHz	PCLK = 50 MHz
0	0	2	16.0 Mbps*	18.0 Mbps*	20.0 Mbps*	25.0 Mbps*
1	0	4	8.00 Mbps	9.00 Mbps	10.0 Mbps	12.5 Mbps
2	0	6	5.33 Mbps	6.00 Mbps	6.67 Mbps	8.33 Mbps
3	0	8	4.00 Mbps	4.50 Mbps	5.00 Mbps	6.25 Mbps
4	0	10	3.20 Mbps	3.60 Mbps	4.00 Mbps	5.00 Mbps
5	0	12	2.67 Mbps	3.00 Mbps	3.33 Mbps	4.16 Mbps
5	1	24	1.33 Mbps	1.50 Mbps	1.67 Mbps	2.08 Mbps
5	2	48	667 kbps	750 kbps	833 kbps	1.04 Mbps
5	3	96	333 kbps	375 kbps	417 kbps	521 kbps
255	3	4096	7.81 kbps	8.80 kbps	9.78 kbps	12.2 kbps

[1]

Serial Peripheral Clock Delay Register (SPCKD)

- The value of this register sets a period from the beginning of SSL signal assertion to the clock oscillations on the RSPK line.



	b7	b6	b5	b4	b3	b2	b1	b0
	—	—	—	—	—	SCKDL[2:0]		
Value after reset:	0	0	0	0	0	0	0	0

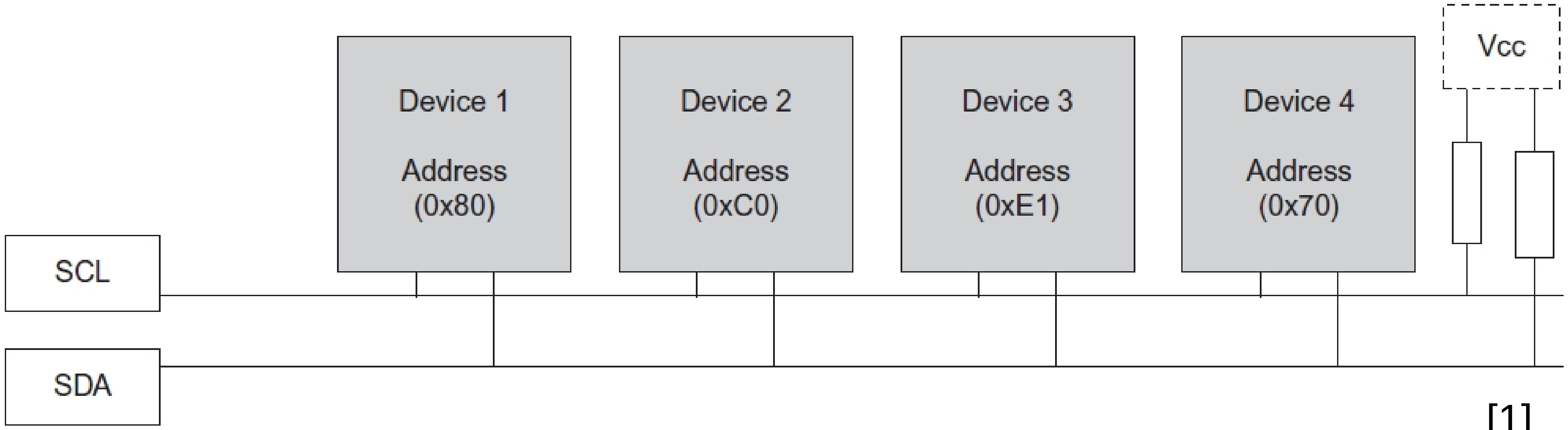
[2]

Bit	Symbol	Bit Name	Description	R/W
b2 to b0	SCKDL[2:0]	RSPCK Delay Setting	b2 b0 0 0 0: 1 RSPCK 0 0 1: 2 RSPCK 0 1 0: 3 RSPCK 0 1 1: 4 RSPCK 1 0 0: 5 RSPCK 1 0 1: 6 RSPCK 1 1 0: 7 RSPCK 1 1 1: 8 RSPCK	R/W
b7 to b3	—	Reserved	These bits are read as 0. The write value should be 0.	R/W

[2]

I²C (IIC)

- Inter-Integrated Circuit Bus
- A two line bus for communicating data at high speeds
- Multiple devices on the same bus with only one master controlling the bus
- Needs pull up resistors and is kept at a digital high level when idle



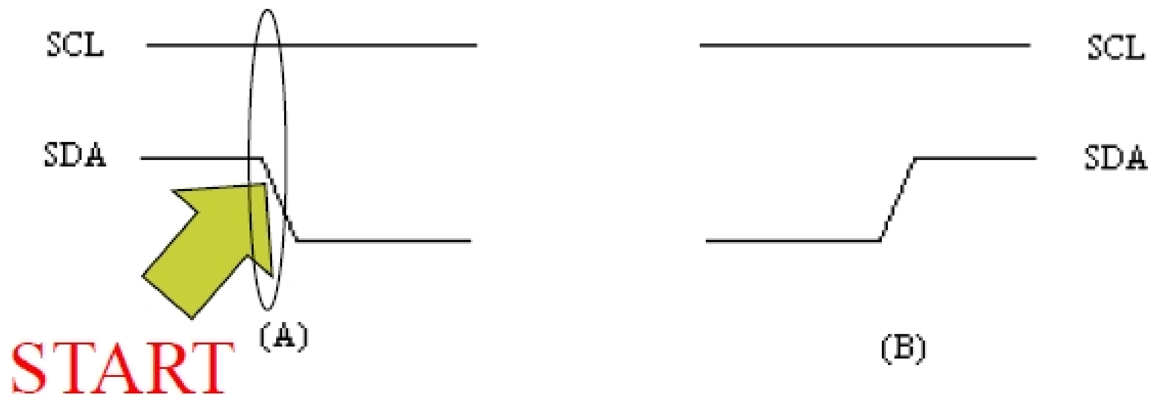
[1]

I^{2}C Working

- Two wires:
  - SCL (Serial Clock): Synchronizing data transfer on the data line.
  - SDA (Serial Data): Responsible for transferring data between devices.
- Together they can toggle in a controlled fashion to indicated certain important conditions that determine the status of the bus and intentions of the devices on the bus.
- Before any form of data transfer takes place, a device wanting to transfer data must take control of the bus (monitor the bus).

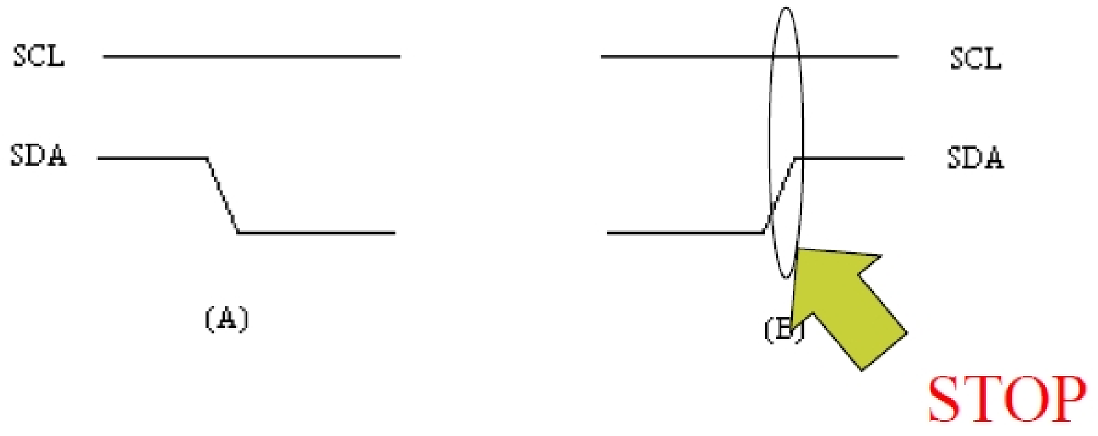
I²C Working cont.

- If the bus is held high, then it is free. A device may issue a START condition and take control of the bus.
- If a START condition is issued, no other device will transmit data on the bus (predetermined behavior for all devices).



I²C Working cont.

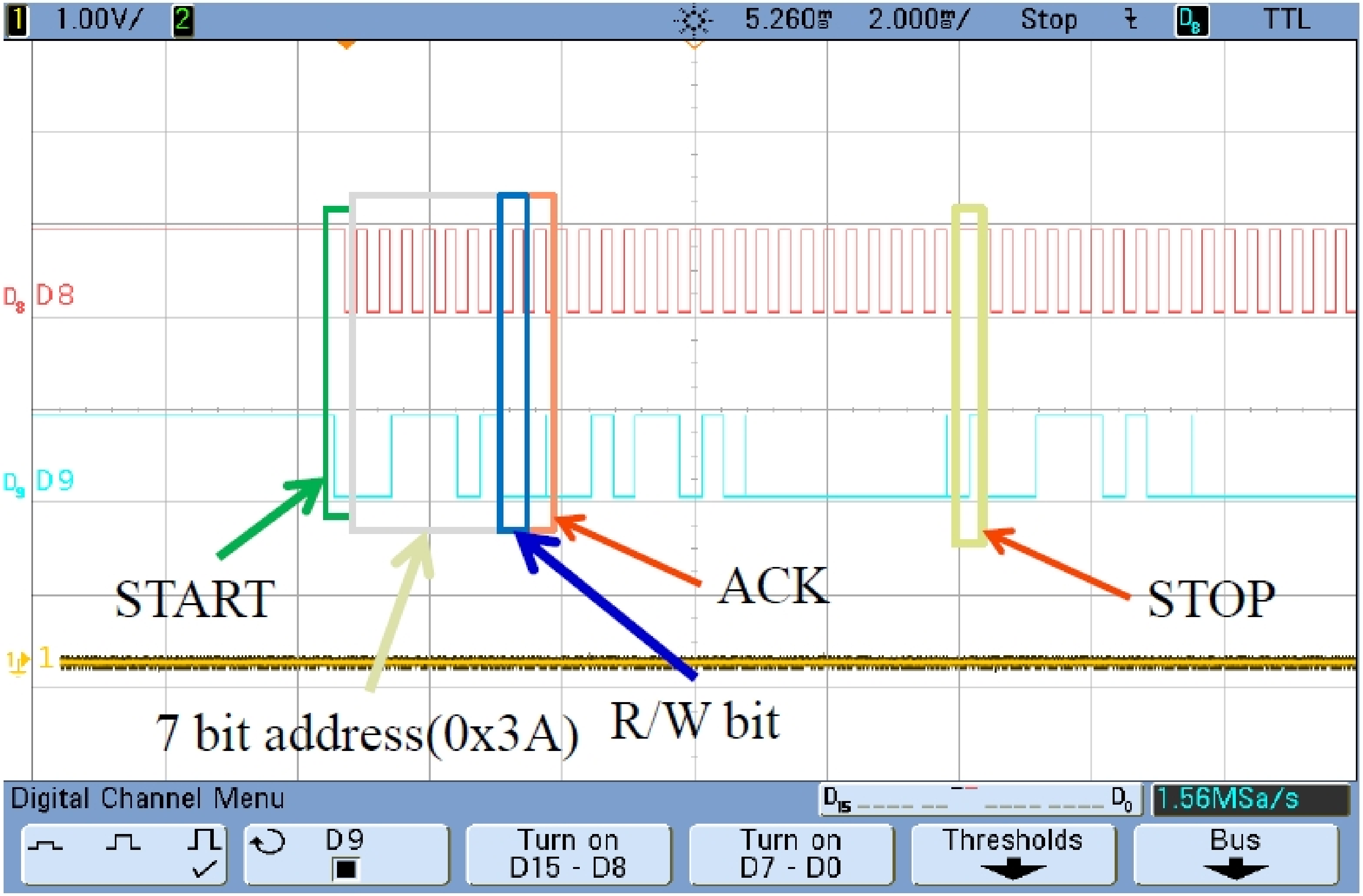
- When device is ready to give up control of the bus, it issues a STOP condition.
- STOP condition is one in which the SDA line gets pulled high while the SCL line is high.
- Other conditions: RESTART (combination of a START and STOP signal).



I^{2}C Working cont.

- Address the slave device with one byte of data which consists of a 7-bit address + 1 bit (R/W).
- If this bit is low, it indicates that the master wants to write to the slave device; if high, the master device wishes to read from the slave. This determines whether the next transactions are going to be read from or written to the addressed slave devices.
- A ninth bit (clock) is transmitted with each byte of data transmitted (ACK(Logic 0)/NACK(logic 1) bit). The slave device must provide an ACK within the ninth cycle to acknowledge receipt of data.

I²C Working cont.



What we have covered

- Basics of communication
- Creating queues
- Various transmission protocols and how to operate them:
  - RS232
  - UART
  - RSPI
  - I^2C

References

- [1] Embedded Systems, An Introduction Using the Renesas RX63N Microcontroller
- [2] Renesas Electronics, Inc. (February, 2013). *RX63N Group, RX631 Group User's Manual: Hardware, Rev.1.60.*

Renesas Electronics America Inc.