ICS library
Function manual

Index

1. Introduction

1.1. Introduction

This document is a manual for ICS library manual.

1.2. Caution

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Desk Top Laboratories Inc products listed herein, please confirm the latest product information with a Desk Top Laboratories Inc. Also, please pay regular and careful attention to additional and different information to be disclosed by Desk Top Laboratories Inc such as that disclosed through our website.

2. Desk Top Laboratories Inc does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of Desk Top Laboratories Inc products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Desk Top Laboratories Inc or others.

3. You should not alter, modify, copy, or otherwise misappropriate any Desk Top Laboratories Inc product, whether in whole or in part.

4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Desk Top Laboratories Inc assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Desk Top Laboratories Inc products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Desk Top Laboratories Inc products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.

6. Desk Top Laboratories Inc has used reasonable care in preparing the information included in this document, but Desk Top Laboratories Inc does not warrant that such information is error free. Desk Top Laboratories Inc assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

7. Desk Top Laboratories Inc products are classified to the experimental use. Especially, you may not use any Desk Top Laboratories Inc product for any application for Transportation equipment (automobiles, trains, ships, etc.), traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support; Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support.

8. You should use the Desk Top Laboratories Inc products described in this document within the range specified by Desk Top Laboratories Inc, especially with respect to the maximum rating,

operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Desk Top Laboratories Inc shall have no liability for malfunctions or damages arising out of the use of Desk Top Laboratories Inc products beyond such specified ranges.

9. Although Desk Top Laboratories Inc endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Desk Top Laboratories Inc products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Desk Top Laboratories Inc product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

10. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Desk Top Laboratories Inc.

ICS is a product of RENESAS electronics. Desk Top Laboratories Inc performs ICS related support business, such as the directions for ICS and a library.

2. ICS library overview

2.1. ICS communication specification / Library source code

ICS library source code and the communication protocol are not disclosed. Here, we will discuss the important items to use ICS.

2.2. Limitations of the data transfer interval

In order to transfer the data from your CPU side, user CPU needs to call ics_watchpoint() function. How to call this function, the following restrictions apply:

Minimum calling period:　250us (communication speed 1Mbps)
Maximum calling period:　5ms

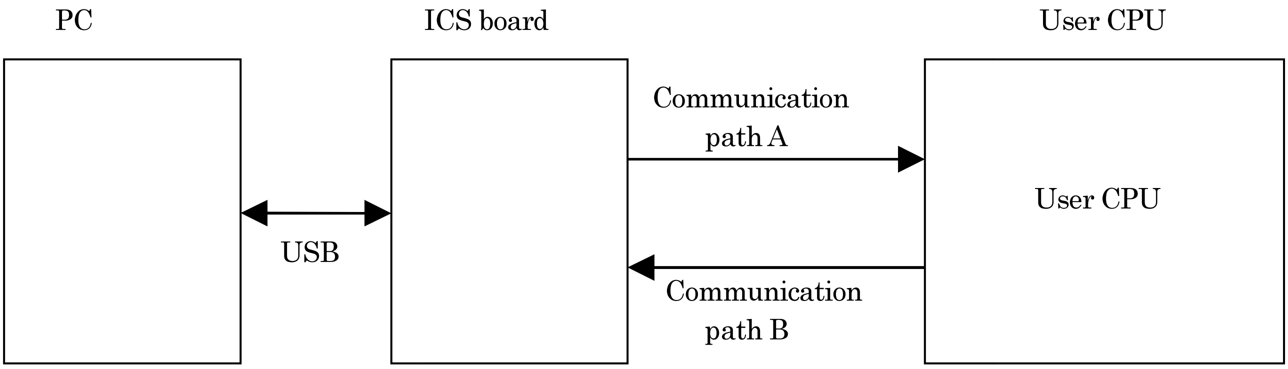

Fig. 1　ICS system structure

In this ICS, there is a limit of data transfer interval. This restriction is caused by communication rate upper limit of the channel B in Fig.1. In the ICS system, whenever it calls the below-mentioned data transfer function ics_watchpoint(), fixed-length data is sent to an ICS board from the target. The shortest time restriction of the transmission interval occurs from this data transferring time, the delay of the time by interrupt of the target CPU and ICS board operation overheads. If it becomes below this time, transmission is not performed well and ICS may not carry out normal operation.

The shortest time restriction of the transmission interval of ICS is greatly dependent on a transfer rate. When transmission speed is 1Mbps as an example, the shortest time constraint serves as 250us. Please refer to the statement of each library portion for other transmission speed. Moreover, there is also restriction of the maximum latency time interval of an ics_watchpoint() function, and it has been 5 ms irrespective of the library.

2.3. Difference between 16bit and 32bit libraries

There are a 16bit library and a 32bit library in ICS. The difference between these libraries is defined in the maximum bit length to be supported by the wave waveform display function rather than the bit length of the CPU itself. In 16bit CPU, such as, RL78, it offers 16bit libraries, 32bit library in 32b it CPU, such as, SH, RX, RZ, RH850 series.

Data to be transferred at one time at time intervals described in the portion of [limitations of the data transfer interval] is 16byte.

1) Operation in 16bit library mode

For numerical display, it works for all type of 8, 16 and 32bits. However, there are restrictions about the waveform display. If the data is 8bits, it is extended to 16bits with sign. If the data is 16bits, it remains as it is. And they will transmit a part for 8ch at once. 32bits data cannot be transmitted.

2) Operation in 32bit library mode

For numerical display, it works for all type of 8, 16 and 32bits. ics_watchpoint()function is called, it will capture 8ch of 8bits, 16bits and 32bits data. And it transmits 4ch of the data. The next time ics_watchpoint() function is called, it will not capture the data, and it transfers the rest of the 4ch data. In other words, in the case of a 32bits library, it transfers for one 8ch is carried out by two times of ics_watchpoint() functions.

	Merit	Demerit
16bit library	Waveform update interval is short	Impossible to display 32bit variable waveform
32bit library	Possible to display 32bit variable waveform	Waveform update rate is twice the 16bit library.

2.4. Restriction at the time of numeric display window use

In ICS, since the numeric display and the waveform display are shared by one communication path, when performing a numeric display and a waveform display simultaneously, restrictions of a waveform display occur. Since waveform data is transmitted each time when the waveform display is performed and the numeric display is not performed, data is displayed as it is. However, when the numeric display and the waveform display are performed simultaneously, data is not updated by one sampling at tens of ms, but the part of displayed waveform may become flat. When carrying out data measurement and such a situation is not suitable, please suspend the “AUTO REFRESH” function of ICS temporarily.

2.5. Filename and library name

ICS library is made up of the following two files.
ics_<CPUNAME>.h
ics_<CPUNAME>.obj

And it is made up of the following two functions.

```
void   ics_init( void*  addr,   char  unitpin,  char  level );
void   ics_watchpoint(void);
```

However, depending on the CPU, the name may be different.

*Caution 1:

Depending on CPU, an used interrupt is different.

*Caution 2:

In the library of free distribution, DTC uses the standard address mode. The vector table of the DTC, you must be located in RAM. You must be located the vector table of DTC in RAM.

If you use a short address mode in DTC, if you want to use the big-endian, if you want to place a DTC table in ROM, if it is different from the specification of the standard, free library cannot be used.

*Caution 3:

Option switch of the compiler assembler linker when generating a standard library takes advantage of the state in which it was generated by the default project.　If you have changed memory model, endian, register mode and so on to be used in your project, a part of the ICS library or all functions may not work. Please use ICS library after confirming the state of the compiler switch which is to be used.

3. Resources and Library

3.1. RX62T series

3.1.1. RX62T resources

CPU name	RX62T series
Develop environment	CubeSuite+ Ver.2.01.00
Library version	Ver.2.0 ～ Ver.2.1
Communication rate	$Rate = \frac{PCLKB}{48}[Mbps]$ Standard Clock 1Mbps @PCLKB = 48MHz
Status	SCI0, SCI1, SCI2 support all ports
Library type	32bit Library
Library file name	ics_RX62T.obj
Header file name	ics_RX62T.h

Used CPU resources	Support ICS	Support variable type
・Used internal resources SCI0 INT SCI0 RXI DTC INT216 (TXI0) ICU.DTCER[216].BIT.DTCE SCI0 (all registers) DTC (all registers) ICU.IPR[0x80].BYTE ICU.IER[0x1A].BIT.IEN6 ICU.IER[0x1A].BIT.IEN7 ICU.IER[0x1B].BIT.IEN0 ICU.IER[0x1B].BIT.IEN1 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRA.BIT.B31 SYSTEM.MSTPCRB.BIT.B31 PORTB.ICR.BIT.B1 = 1 External pin PB2: TXD0 PB1: RXD0 SCI1 INT SCI1 RXI DTC INT220 (TXI1) ICU.DTCER[220].BIT.DTCE	Support ICS *W1001 H/W model 1 H/W Ver. 1 S/W Ver. 1.22 (after) *W1003 H/W model 4 H/W Ver. 1 S/W Ver. 1.22 (after) ICS PC software After Ver. 2.5.0.0	Numeric display 8bit unsigned char 8bit signed char 16bit unsigned short 16bit signed short 32bit unsigned int 32bit signed int 32bit IEEE754 floating point 8bit BOOL type 8bit LOGIC type Waveform display 8bit unsigned char 8bit signed char 16bit unsigned short 16bit signed short 32bit unsigned int 32bit signed int 32bit IEEE754 floating point

SCI1 (all registers) DTC (all registers) ICU.IPR[0x81].BYTE ICU.IER[0x1B].BIT.IEN2 ICU.IER[0x1B].BIT.IEN3 ICU.IER[0x1B].BIT.IEN4 ICU.IER[0x1B].BIT.IEN5 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRA.BIT.B31 SYSTEM.MSTPCRB.BIT.B30 PORTDICR.BIT.B5= 1 External pin PD3 TXD1 PD5: RXD1 SCI2 (PB5, PB6) INT SCI2RXI DTC INT224 (TXI2) ICU.DTCER[224].BIT.DTCE SCI2 (all registers) DTC (all registers) ICU.IPR[0x82].BYTE ICU.IER[0x1B].BIT.IEN6 ICU.IER[0x1B].BIT.IEN7 ICU.IER[0x1C].BIT.IEN0 ICU.IER[0x1C].BIT.IEN1 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRA.BIT.B31 SYSTEM.MSTPCRB.BIT.B29 PORTB.ICR.BIT.B6 = 1 IOPORT.PFFSCI.BIT.SCI2S External pin PB5: TXD2 PB6: RXD2 SCI2 (P81, P80) INT SCI2RXI DTC INT224 (TXI2) ICU.DTCER[224].BIT.DTCE SCI2 (all registers) DTC (all registers) ICU.IPR[0x82].BYTE ICU.IER[0x1B].BIT.IEN6 ICU.IER[0x1B].BIT.IEN7		

ICU.IER[0x1C].BIT.IEN0 ICU.IER[0x1C].BIT.IEN1 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRA.BIT.B31 SYSTEM.MSTPCRB.BIT.B29 PORT8.ICR.BIT.B0 = 1 IOPORT.PFFSCI.BIT.SCI2S External pin P81: TXD2 P80: RXD2		

3.1.2. RX62T function library

Lib Ver.2.0 ～Ver.2.1 on CubeSuite+ Ver.2.01.00

Initialize function void ics_init(void* addr, char port, char level);

This function initializes ICS relation including a pin definition. Be careful to destroy neither the definition of the resource pin used by ICS indicated for the preceding clause, nor a setup of a standby control register etc., after initialization of this function.

First parameter:

Please specify the head address of the vector table of DTC. Before calling an ics_init() function, a user needs to secure a DTC vector table. 12bits of lower ranks of this address need to be '0'.

Second parameter:

The port number of SCI and the pins which SCI uses are set up. For this parameter, please use the string that is defined in the ICS_<CPUNAME>.h.

Third parameter:

Please specify the interrupt level of SCI to be used in ICS. There is a need to meet the following conditions.
There is a possibility that the 2ms interrupt occurs at the minimum interval, as a system, please set the interrupt level that can tolerate this interrupt interval. Receive interrupt of the SCI is the longest processing time. It is about 10us, but if there is an interrupt source that cannot tolerate interrupt disable time, please set the interrupt level higher than the interrupt level setting.

Transfer function void ics_watchpoint(void);

This is the data transfer function. Normally an user puts this function in the carrier interrupt function. However, in the sample software, to make it easier to understand how to write the software, it is written in the main routine.

This function reads the data of the variable specified by the PC, and copy it to the transfer buffer for the DTC.

When the communication speed is 1Mbps, this function should maintain the interval of 250us or more of minimum, and less than 5ms and please call it. When the communication speed is not 1Mbps, please keep and call the time defined by the following formula.

$$MinimumPeriod = 1/(CommunicationSpeed[bps]) \times 180 + 70[us]$$

When the communication speed is 1Mbps, let 1Mbps into this formula.

$$MinimumPeriod = 1/(1[Mbps]) \times 180 + 70[us] = 250[us]$$

*Caution: The interrupt interval in the user software is a relation of other interrupt, and generating of interrupt may be delayed. Please also take that interrupt timing shifts into consideration and call it.

Interrupt functions

Since the following interrupt vector is used, please register the following function into the interrupt vector of user software. When you use the project automatically generated with the standard compiler for RENESAS, please add these functions to the file which indicated the interrupt processing "intprg.c".

The case of SCI0

```
// SCI0 ERI0
void Excep_SCI0_ERI0(void){ ics_int_sci_eri(); }
// SCI0 ERI0
void Excep_SCI0_RXI0(void){ ics_int_sci_rxi(); }
```

The case of SCI1

```
// SCI1 ERI1
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
// SCI1 RXI1
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

The case of SCI2

```
// SCI2 ERI2
void Excep_SCI2_ERI2(void){ ics_int_sci_eri(); }
// SCI2 RXI2
void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); }
```

3.1.3. RX62T functions usage

This document explains the setting method of the user program for using ICS, using attached sample software.

1) To secure the BDTCTBL section in the development environment.

The section of BDTCTBL is assigned as the address on RAM that 12 bits of low ranks are set to 0. This address is set as development environment and carried out. Since the models of the minimum RAM size are 8kbytes, the address which can be set up is 0x0000 or 0x1000. If the models of RAM size are 16kBytes, the address which can be set up is 0x0000, 0x1000, 0x2000, 0x3000. Here, please set up at 0x0000.

When you use emulator, such as E1 or something, please keep a user RAM domain, the domain of a DTC table and domain of E1 emulator from overlapping.

2) Define DTC table in user program

Please define the DTC table variable "unsigned long dtc_table[256];
At the top of ICS_sample.c

```
#pragma section DTCTBL
unsigned long dtc_table[256];        // caution alignment 0x000
#pragma section
```

3) Call "ics_init()" as following

Please put the initialization function "ics_init((void*)dtc_table, ICS_SCI2_PB5_PB6, 6)"
at the user initialization part.

First parameter is the address to be secured at 1).
Second parameter is the port name you want to use defined in the ICS_<CPUNAME>.h.
Third parameter is the interrupt level using in the ICS. Normally we choose the level lower than the carrier interrupt.

------------ List 1 main.c ---

```
#pragma section DTCTBL
unsigned long dtc_table[256];        // caution alignment 0x000
#pragma section

void main(void)
{
        ics_init((void*)dtc_table, ICS_SCI2_PB5_PB6, 6);    /* Interrupt level 6            */
        while(1)
        {    nop();    }
}
```

4) Installation of ics_watchpoint() function

In this sample software, ics_watchpoint() function is called in the main routine. But normally this is called in the carrier interrupt.

And this function must be called below 5ms period and above 250us. If the carrier interrupt period is below 250us, please decimate function call of ics_watchpoint() as in the List 2.

----------- List 2 ics_watchpoint() decimation ------------------------

```
int     deci = 0;

void    int_TM0(void)     /* 100us period */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics_watchpoint();
        }
}
```

5) Modification of "intprg.c"

The case of SCI0

```
// SCI0 ERI0
void Excep_SCI0_ERI0(void){ ics_int_sci_eri(); }
// SCI0 RXI0
void Excep_SCI0_RXI0(void){ ics_int_sci_rxi(); }
```

The case of SCI1

```
// SCI1 ERI1
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
// SCI1 RXI1
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

The case of SCI2

```
// SCI2 ERI2
void Excep_SCI2_ERI2(void){ ics_int_sci_eri(); }
// SCI2 RXI2
void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); }
```

3.1.4. ICS on board clock for RX62T

When use this library, please choose the clock on an ICS board as follows according to a setup of the clock of the CPU side. In the case of the model which cannot change the on board clock of the ICS, please use the PCLK=96MHz.

On board clock frequency of ICS = (PCLKB / 6) MHz

Example:
The case of PCLKB = 50MHz: ICS CLOCK = 50/6 = 8.333MHz
The case of PCLKB = 48MHz: ICS CLOCK = 48/6 = 8.000MHz

Desk Top Laboratories is preparing the stock of 8.000MHz, 8.333MHz and 10.000MHz parts.

*Caution:

W1001 (No external clock module type)
This type can not change the clock, so you can use only 8MHz clock.

W1003 (Support external clock module)
In the case of using ICS clock except 8MHz, you need to change clock module.

W1004 (Optical fiber type)
This model supports variable clock function, so you can change master clock from the PC software.

3.2. RX111 series

3.2.1. RX111 resources

CPU name	RX111 series
Develop environment	CubeSuite+ Ver.2.01.00
Library version	Ver.2.0
Communication rate	$Rate = \frac{PCLKB}{32}[Mbps]$ Standard Clock 1Mbps @PCLKB = 32MHz
Status	SCI1, SCI5, SCI12 support all ports
Library type	32bit Library
Library file name	ics_RX111.obj
Header file name	ics_RX111.h

Used CPU resources	Support ICS	Support variable type
・Used internal resources SCI1 INT SCI1 RXI DTC INT220 (TXI1) ICU.DTCER[220].BIT.DTCE SCI1 (all registers) DTC (all registers) ICU.IPR[218].BYTE ICU.IER[0x1B].BIT.IEN2 ICU.IER[0x1B].BIT.IEN3 ICU.IER[0x1B].BIT.IEN4 SYSTEM.MSTPCRB.BIT.B30 SYSTEM.MSTPCRB.BIT.B26 SYSTEM.MSTPCRB.BIT.B4 MPC.PWPR.BIT.B0WI MPC.PWPR.BIT.PFSWE ・Ext pin, PC7:TXD1, PC6:RXD1 MPC.PC6PFS.BIT.PSEL MPC.PC7PFS.BIT.PSEL PORTC.PMR.BIT.B7 PORTC.PMR.BIT.B6 ・Ext pin, P26:TXD1, P30:RXD1 MPC.P26PFS.BIT.PSEL MPC.P30PFS.BIT.PSEL PORT2.PMR.BIT.B6 PORT3.PMR.BIT.B0 ・Ext pin P16:RXD1, P15:RXD1	Support ICS *W1001 H/W model 1 H/W Ver. 1 S/W Ver. 1.22 (after) *W1003 H/W model 4 H/W Ver. 1 S/W Ver. 1.22 (after) ICS PC software After Ver. 2.5.0.0	Numeric display 8bit unsigned char 8bit signed char 16bit unsigned short 16bit signed short 32bit unsigned int 32bit signed int 8bit BOOL type 8bit LOGIC type Waveform display 8bit unsigned char 8bit signed char 16bit unsigned short 16bit signed short 32bit unsigned int 32bit signed int

MPC.P16PFS.BIT.PSEL MPC.P15PFS.BIT.PSEL PORT1.PMR.BIT.B6 PORT1.PMR.BIT.B5 SCI5 INT　SCI5 RXI DTC　INT224　(TXI5) ICU.DTCER[224].BIT.DTCE SCI5　　(all registers) DTC　　(all registers) ICU.IPR[222].BYTE ICU.IER[0x1B].BIT.IEN6 ICU.IER[0x1B].BIT.IEN7 ICU.IER[0x1C].BIT.IEN0 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRA.BIT.B31 SYSTEM.MSTPCRB.BIT.B30 MPC.PWPR.BIT.B0WI MPC.PWPR.BIT.PFSWE ・Ext pin, PA4:TXD5, PA3:RXD5 MPC.PA3PFS.BIT.PSEL MPC.PA4PFS.BIT.PSEL PORTA.PMR.BIT.B4 PORTA.PMR.BIT.B3 ・Ext pin, PC3:TXD5, PC2:RXD5 MPC.PC3PFS.BIT.PSEL MPC.PC2PFS.BIT.PSEL PORTC.PMR.BIT.B3 PORTC.PMR.BIT.B2 SCI12 INT　SCI12 (RXI12) DTC　INT240　(TXI12) ICU.DTCER[240].BIT.DTCE SCI2　　(all registers) DTC　　(all registers) ICU.IPR[238].BYTE ICU.IER[0x1D].BIT.IEN6 ICU.IER[0x1D].BIT.IEN7 ICU.IER[0x1E].BIT.IEN0 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRA.BIT.B31 SYSTEM.MSTPCRB.BIT.B29		

MPC.PWPR.BIT.B0WI MPC.PWPR.BIT.PFSWE ・Ext pin,PE1:TXD12, PE2:RXD12 MPC.PE1PFS.BIT.PSEL MPC.PE2PFS.BIT.PSEL PORTE.PMR.BIT.B1 PORTE.PMR.BIT.B2 ・Ext pin, P14:TXD12, P17:RXD12 MPC.P14PFS.BIT.PSEL MPC.P17PFS.BIT.PSEL PORT1.PMR.BIT.B4 PORT1.PMR.BIT.B7		

3.2.2. RX111 function library

Lib Ver.2.0 on CubeSuite+ Ver.2.01.00

Initialize function void ics_init(void* addr, char port, char level);

This function initializes ICS relation including a pin definition. Be careful to destroy neither the definition of the resource pin used by ICS indicated for the preceding clause, nor a setup of a standby control register etc., after initialization of this function.

First parameter:
Please specify the head address of the vector table of DTC. Before calling an ics_init() function, a user needs to secure a DTC vector table. 12bits of lower ranks of this address need to be ‘0’.

Second parameter:
The port number of SCI and the pins which SCI uses are set up. For this parameter, please use the string that is defined in the ICS_<CPUNAME>.h.

Third parameter:
Please specify the interrupt level of SCI to be used in ICS. There is a need to meet the following conditions.
There is a possibility that the 2ms interrupt occurs at the minimum interval, as a system, please set the interrupt level that can tolerate this interrupt interval. Receive interrupt of the SCI is the longest processing time. It is about 10us, but if there is an interrupt source that cannot tolerate interrupt disable time, please set the interrupt level higher than the interrupt level setting.

Transfer function void ics_watchpoint(void);

This is the data transfer function. Normally an user puts this function in the carrier interrupt function. However, in the sample software, to make it easier to understand how to write the software, it is written in the main routine.

This function reads the data of the variable specified by the PC, and copy it to the transfer buffer for the DTC.

When the communication speed is 1Mbps, this function should maintain the interval of 250us or more of minimum, and less than 5ms and please call it. When the communication speed is not 1Mbps, please keep and call the time defined by the following formula.

$$MinimumPeriod = 1/(CommunicationSpeed[bps]) \times 180 + 70[us]$$

When the communication speed is 1Mbps, let 1Mbps into this formula.

$$MinimumPeriod = 1/(1[Mbps]) \times 180 + 70[us] = 250[us]$$

*Caution: The interrupt interval in the user software is a relation of other interrupt, and generating of interrupt may be delayed. Please also take that interrupt timing shifts into consideration and call it.

Interrupt functions

Since the following interrupt vector is used, please register the following function into the interrupt vector of user software. When you use the project automatically generated with the standard compiler for RENESAS, please add these functions to the file which indicated the interrupt processing “intprg.c”.

The case of SCI1

```
// SCI1 ERI1
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
// SCI1 ERI1
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

The case of SCI5

```
// SCI5 ERI5
void Excep_SCI5_ERI5(void){ ics_int_sci_eri(); }
// SCI5 RXI5
void Excep_SCI5_RXI5(void){ ics_int_sci_rxi(); }
```

The case of SCI12

```
// SCI12 ERI12
void Excep_SCI12_ERI12(void){ ics_int_sci_eri(); }
// SCI12 RXI12
void Excep_SCI12_RXI12(void){ ics_int_sci_rxi(); }
```

3.2.3. RX111 function usage

This document explains the setting method of the user program for using ICS, using attached sample software.

1) To secure the BDTCTBL section in the development environment.

The section of BDTCTBL is assigned as the address on RAM that 12 bits of low ranks are set to 0. This address is set as development environment and carried out. Since the models of the minimum RAM size are 8kbytes, the address which can be set up is 0x0000 or 0x1000. If the models of RAM size are 16kBytes, the address which can be set up is 0x0000, 0x1000, 0x2000, 0x3000. Here, please set up at 0x0000.

When you use emulator, such as E1 or something, please keep a user RAM domain, the domain of a DTC table and domain of E1 emulator from overlapping.

2) Define DTC table in user program

Please define the DTC table variable "unsigned long dtc_table[256];
At the top of ICS_sample.c

```
#pragma section DTCTBL
unsigned long dtc_table[256];        // caution alignment 0x000
#pragma section
```

3) Call "ics_init()" as following

When you use SCI1,
Please put the initialization function "ics_init((void*)dtc_table, ICS_SCI1_PC7_PC6, 6)"
at the user initialization part.

First parameter is the address to be secured at 1).
Second parameter is the port name you want to use defined in the ICS_<CPUNAME>.h.
Third parameter is the interrupt level using in the ICS. Normally we choose the level lower than the carrier interrupt.

```
----------- List 1   main.c -----------------------------------------------
#pragma section DTCTBL
unsigned long dtc_table[256];        // caution alignment 0x000
#pragma section

void main(void)
{
      ics_init((void*)dtc_table, ICS_SCI1_PC7_PC6, 6);     /* Interrupt level 6            */
      while(1)
      {    nop();    }
}
```

4) Installation of ics_watchpoint() function

In this sample software, ics_watchpoint() function is called in the main routine. But normally this is called in the carrier interrupt.

And this function must be called below 5ms period and above 250us. If the carrier interrupt period is below 250us, please decimate function call of ics_watchpoint() as in the List 2.

----------- List 2 ics_watchpoint() decimation ------------------------

```
int     deci = 0;

void   int_TM0(void)      /* 100us period */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics_watchpoint();
        }
}
```

5) Modification of “intprg.c”

The case of SCI1

```
// SCI1 ERI1
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
// SCI1 ERI1
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

The case of SCI5

```
// SCI5 ERI5
void Excep_SCI5_ERI5(void){ ics_int_sci_eri(); }
// SCI5 RXI5
void Excep_SCI5_RXI5(void){ ics_int_sci_rxi(); }
```

The case of SCI12

```
// SCI12 ERI12
void Excep_SCI12_ERI12(void){ ics_int_sci_eri(); }
// SCI12 RXI12
void Excep_SCI12_RXI12(void){ ics_int_sci_rxi(); }
```

3.2.4. ICS on board clock for RX111

When use this library, please choose the clock on an ICS board as follows according to a setup of the clock of the CPU side. In the case of the model which cannot change the on board clock of the ICS, please use the PCLK=32MHz.

If you use RX111 with external clock, please calculate following equations
On board clock frequency of ICS = (PCLKB / 4) MHz

Desk Top Laboratories is preparing the stock of 8.000MHz, 8.333MHz and 10.000MHz parts.

*Caution:

W1001 (No external clock module type)
This type can not change the clock, so you can use only 8MHz clock.

W1003 (Support external clock module)
In the case of using ICS clock except 8MHz, you need to change clock module.

W1004 (Optical fiber type)
This model supports variable clock function, so you can change master clock from the PC software.

3.3. RL78G14 series

3.3.1. RL78G14 resources

CPU name	RL78G14series			
Develop evvironment	CubeSuite+ Ver.2.01.00			
Library version	Ver2.00			
Communication rate	$Communication\_rate = \frac{CLK}{32}[Mbps]$ Standard CLK = 32MHz ：1Mbps			
status	SCI0, SCI1, SCI2 support			
Library type	16bit library			
Library file name	R5F104xx.rel example：if you use R5F104LE, the name is “R5F104LE.rel”.			
Header file name	R5F104xx.h example：if you use R5F104LE, the name is “R5F104LE.h”.			
Momory model	Medium (ROM=1MB, RAM=64kB)			
DTC address mode	Standard			
Endian	Little			
Support port	R5F104AE (30pin)	#define	ICS_SCI0_P51_P50	(0x00)
		#define	ICS_SCI0_P17_P16	(0x01)
		#define	ICS_SCI1_P00_P01	(0x11)
	R5F104BE (32pin)	#define	ICS_SCI0_P51_P50	(0x00)
		#define	ICS_SCI0_P17_P16	(0x01)
		#define	ICS_SCI1_P00_P01	(0x11)
	R5F104CE (36pin)	#define	ICS_SCI0_P51_P50	(0x00)
		#define	ICS_SCI0_P17_P16	(0x01)
		#define	ICS_SCI1_P00_P01	(0x11)
	R5F104EE (40pin)	#define	ICS_SCI0_P51_P50	(0x00)
		#define	ICS_SCI0_P17_P16	(0x01)
		#define	ICS_SCI1_P00_P01	(0x11)
	R5F104FE (44pin)	#define	ICS_SCI0_P51_P50	(0x00)
		#define	ICS_SCI0_P17_P16	(0x01)
		#define	ICS_SCI1_P00_P01	(0x11)
	R5F104GE (48pin)	#define	ICS_SCI0_P51_P50	(0x00)
		#define	ICS_SCI0_P17_P16	(0x01)
		#define	ICS_SCI1_P00_P01	(0x11)
	R5F104JE (52pin)	#define	ICS_SCI0_P51_P50	(0x00)
		#define	ICS_SCI0_P17_P16	(0x01)
		#define	ICS_SCI1_P02_P03	(0x10)
		#define	ICS_SCI2_P77_P76	(0x20)
	R5F104LE (64pin)	#define	ICS_SCI0_P51_P50	(0x00)
		#define	ICS_SCI0_P17_P16	(0x01)
		#define	ICS_SCI1_P02_P03	(0x10)
		#define	ICS_SCI2_P77_P76	(0x20)
		#define	ICS_SCI2_P13_P14	(0x21)

Used CPU resources	Support ICS	Support variable type
・Used internal resources	Support ICS hardware	Numeric display
		8bit unsigned char
DTC	*W1001	8bit signed char
	H/W model　1	16bit unsigned short
*SCIx_Pab_Pcd	H/W Ver.　1	16bit signed short
X：SCI number	S/W Ver.　1.22　(after)	32bit unsigned int
a, b, c, d：port number		32bit signed int
	*W1003	8bit　BOOL type
SCIx　all resources	H/W model　4	8bit　LOGIC type
	H/W Ver.　1	
PFC	S/W Ver.　1.22　(after)	Waveform display
PIOR0.1		8bit unsigned char
PMCa.b	ICS PC software	8bit signed char
PMCc.d	After Ver. 2.5.0.0	16bit unsigned short
Pa.b = 1	Recommendation	16bit signed short
PMa.b = 0	After　Ver.2.7.3.0	
PMc.d = 1		
External pin		
Pab　for TXDx		
Pcd　for RXDx		
CLOCK		
SPS0　bit4〜7		
INTC		
STPR0x		
STPR1x		
SRPR0x		
SRPR1x		
SREPR0x		
SREPR1x		

3.3.2. RL78G14 series function library

Lib Ver.2.00 on CubeSuite+ Ver.2.01.00

Initialize function void ics_init(unsigned int addr, char pin, char level, unsigned char num);

This function initializes ICS relation including a pin definition. Be careful to destroy neither the definition of the resource pin used by ICS indicated for the preceding clause, nor a setup of a standby control register etc., after initialization of this function.

First parameter:

Please specify the head address of the 16bits of lower ranks of the vector table address of DTC. Before calling an ics_init() function, a user needs to secure a DTC vector table. 8bits of lower ranks of this address need to be '0'.

Second parameter:

The port number of SCI and the pins which SCI uses are set up. For this parameter, please use the string that is defined in the ICS_<CPUNAME>.h.

```
#define    ICS_SCI0_P51_P50    (0x00)
#define    ICS_SCI0_P17_P16    (0x01)
#define    ICS_SCI1_P02_P03    (0x10)
#define    ICS_SCI2_P77_P76    (0x20)
```

Third parameter:

Specify the interrupt level of SCI to be used in ICS. There is a need to meet the following conditions. There is a possibility that the 2ms interrupt occurs at the minimum interval, as a system, please set the interrupt level that can tolerate this interrupt interval. Receive interrupt of the SCI is the longest processing time. It is about 10us, but if there is an interrupt source that cannot tolerate interrupt disable time, please set the interrupt level higher than the interrupt level setting.

Fourth parameter:

The top address of the DTC struct. This parameter can be chosen from 0x40, 0x48, 0x50... 0xF8.

Transfer function void ics_watchpoint(void);

This is the data transfer function. Normally an user puts this function in the carrier interrupt function. However, in the sample software, to make it easier to understand how to write the software, it is written in the main routine.

This function reads the data of the variable specified by the PC, and copy it to the transfer buffer for the DTC.

When the communication speed is 1Mbps, this function should maintain the interval of 250us or more of minimum, and less than 5ms and please call it. When the communication speed is not 1Mbps, please keep and call the time defined by the following formula.

$$MinimumPeriod = 1/(CommunicationSpeed[bps]) \times 180 + 70[us]$$

When the communication speed is 1Mbps, let 1Mbps into this formula.

$$MinimumPeriod = 1/(1[Mbps]) \times 180 + 70[us] = 250[us]$$

*Caution: The interrupt interval in the user software is a relation of other interrupt, and generating of interrupt may be delayed. Please also take that interrupt timing shifts into consideration and call it.

Interrupt function

This library uses these interrupts
INTST0, INTSR0, INTSRE0
INTST1, INTSR1, INTSRE1
INTST2, INTSR2, INTSRE2

```
#ifdef ICS_SCI0
__interrupt void   Excep_INTST0(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR0(void)   { int_ics_sci_rx(); }
__interrupt void   Excep_INTSRE0(void) { int_ics_sci_err(); }
#endif

#ifdef ICS_SCI1
__interrupt void   Excep_INTST1(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR1(void)   { int_ics_sci_rx(); }
__interrupt void   Excep_INTSRE1(void) { int_ics_sci_err(); }
#endif


#ifdef ICS_SCI2
__interrupt void   Excep_INTST2(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR2(void)   { int_ics_sci_rx(); }
__interrupt void   Excep_INTSRE2(void) { int_ics_sci_err(); }
#endif
```

3.3.3. RL78G14 series : function usage

his document explains the setting method of the user program for using ICS, using attached sample software.

1) To secure the DTC table

There are some ways to keep the DTC table memory. We introduce the way we can check from the source code.

To keey the DTC table, please add the following description. This case keeps 0xD0 byte from address 0xFFE00. This address must keep 8bits of the lower ranks of the address.

```
#pragma section @@DATA   @@DTCTBL at 0xFFE00
char   dtc_tbl[0xD0];
#pragma section @@DATA   @@DATA
```

When you use emulator, such as E1 or something, please keep a user RAM domain, the domain of a DTC table and domain of E1 emulator from overlapping.

2) Cal ics_init()

Please put the initialization function “ics_init((void*)dtc_table, ICS_SCI2_P77_P76, 2, 0x40)” at the user initialization part.

First parameter is the address to be secured at 1).
Second parameter is the port name you want to use defined in the ICS_<CPUNAME>.h.
Third parameter is the interrupt level using in the ICS. Normally we choose the level lower than the carrier interrupt.
Fourth parameter is normally 0x40. If you don’t use other DTC channels.

```
------------ List 1   main.c ------------------------------------------------
#pragma   SFR
#pragma   DI
#pragma   EI
#pragma   NOP

#include "ICS_define.h"
#include "low_level_init.h"
#include "ics_R5F104LE.h"

/*********** KEEP DTC TABLE AREA ***********/
#pragma section @@DATA   @@DTCTBL at 0xFFE00
char   dtc_tbl[0xD0];
#pragma section @@DATA   @@DATA

    ics_init(0xFE00, 0x40, 2, ICS_SCI2_P77_P76); /* T5101 CN4 o */
```

3) Installation of ics_watchpoint() function

In this sample software, ics_watchpoint() function is called in the main routine. But normally this is called in the carrier interrupt.

And this function must be called below 5ms period and above 250us. If the carrier interrupt period is below 250us, please decimate function call of ics_watchpoint() as in the List 2.

----------- List 2 ics_watchpoint() ------------------------

```
__interrupt void   int_TM0(void)
{
        theta_e_est = theta_e_est + 60;
        if (theta_e_est>4095)
        {
                theta_e_est = theta_e_est - 4096;
        }

        /********** pwm reference generation **************/
        refu = R_FIX_sin_int16(theta_e_est);
        refv = R_FIX_sin_int16(theta_e_est-1333);
        refw = R_FIX_sin_int16(theta_e_est-2666);

        RPECTL = 0x80U;
        ics_watchpoint();
}
```

4) Add interrupt functions

The case of SCI0

```
__interrupt void   Excep_INTST0(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR0(void)   { int_ics_sci_rx(); }
__interrupt void   Excep_INTSRE0(void) { int_ics_sci_err(); }
```

The case of SCI1

```
__interrupt void   Excep_INTST1(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR1(void)   { int_ics_sci_rx(); }
__interrupt void   Excep_INTSRE1(void) { int_ics_sci_err(); }
```

The case of SCI2

```
__interrupt void   Excep_INTST2(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR2(void)   { int_ics_sci_rx(); }
__interrupt void   Excep_INTSRE2(void) { int_ics_sci_err(); }
```

3.3.4. ICS on board clock for RL78G14

When use this library, please choose the clock on an ICS board as follows according to a setup of the clock of the CPU side. In the case of the model which cannot change the on board clock of the ICS, please use the CLK=32MHz.

On board clock frequency of ICS = (CLK /4) MHz

Desk Top Laboratories is preparing the stock of 8.000MHz, 8.333MHz and 10.000MHz parts.

*Caution:

W1001 (No external clock module type)
This type can not change the clock, so you can use only 8MHz clock.

W1003 (Support external clock module)
In the case of using ICS clock except 8MHz, you need to change clock module.

W1004 (Optical fiber type)
This model supports variable clock function, so you can change master clock from the PC software.

3.4. RL78F14 series

3.4.1. RL78F14 resources

CPU name	RL78F14 series (R5F10PMF only)	
Develop evvironment	CubeSuite+ Ver.2.01.00	
Library version	Ver2.00	
Communication rate	$Communication\_rate = \frac{CLK}{32}[Mbps]$ Standard CLK = 32MHz :1Mbps	
status	SCI0, SCI1	
Library type	16bit library	
Library file name	R5F10PMF.rel example : if you use R5F10PMF, the name is “R5F10PMF.rel”.	
Header file name	R5F10PMF.h example : if you use R5F10PMF, the name is “R5F10PMF.h”.	
Momory model	Medium (ROM=1MB, RAM=64kB)	
DTC address mode	Standard	
Endian	Little	
Support port	R5F10PMF (80pin)	#define ICS_SCI0_P62_P61 (0x00) #define ICS_SCI1_P74_P75 (0x10) #define ICS_SCI1_P12_P11 (0x11)

Used CPU resources	Support ICS	Support variable type
・Used internal resources DTC *SCIx_Pab_Pcd x : SCI number a, b, c, d: port number SCIx all resources PFC PIOR4 Pa.b = 1 PMa.b = 0 PMc.d = 1 External pins Pab for TXDx Pcd for RXDx CLOCK SPS0 bit4～7 INTC STPR0x STPR1x SRPR0x SRPR1x	Support ICS hardware *W1001 H/W model 1 H/W Ver. 1 S/W Ver. 1.22 (after) *W1003 H/W model 4 H/W Ver. 1 S/W Ver. 1.22 (after) ICS PC software After Ver. 2.5.0.0 Recommendation After Ver.2.7.3.0	Numeric display 8bit unsigned char 8bit signed char 16bit unsigned short 16bit signed short 32bit unsigned int 32bit signed int 8bit BOOL type 8bit LOGIC type Waveform display 8bit unsigned char 8bit signed char 16bit unsigned short 16bit signed short

3.4.2. RL78F14 function library

Lib Ver.2.00 on CubeSuite+ Ver.2.01.00

Initialize function void ics_init(unsigned int addr, char pin, char level, unsigned char num);

This function initializes ICS relation including a pin definition. Be careful to destroy neither the definition of the resource pin used by ICS indicated for the preceding clause, nor a setup of a standby control register etc., after initialization of this function.

First parameter:

Please specify the head address of the 16bits of lower ranks of the vector table address of DTC. Before calling an ics_init() function, a user needs to secure a DTC vector table. 8bits of lower ranks of this address need to be '0'.

Second parameter:

The port number of SCI and the pins which SCI uses are set up. For this parameter, please use the string that is defined in the ICS_<CPUNAME>.h.

```
#define    ICS_SCI0_P62_P61     (0x00)
#define    ICS_SCI1_P74_P75     (0x10)
#define    ICS_SCI1_P12_P11     (0x11)
```

Third parameter:

Specify the interrupt level of SCI to be used in ICS. There is a need to meet the following conditions. There is a possibility that the 2ms interrupt occurs at the minimum interval, as a system, please set the interrupt level that can tolerate this interrupt interval. Receive interrupt of the SCI is the longest processing time. It is about 10us, but if there is an interrupt source that cannot tolerate interrupt disable time, please set the interrupt level higher than the interrupt level setting.

Fourth parameter:

The top address of the DTC struct. This parameter can be chosen from 0x40, 0x48, 0x50... 0xF8.

Transfer function void ics_watchpoint(void);

This is the data transfer function. Normally an user puts this function in the carrier interrupt function. However, in the sample software, to make it easier to understand how to write the software, it is written in the main routine.

This function reads the data of the variable specified by the PC, and copy it to the transfer buffer for the DTC.

When the communication speed is 1Mbps, this function should maintain the interval of 250us or more of minimum, and less than 5ms and please call it. When the communication speed is not 1Mbps, please keep and call the time defined by the following formula.

$$MinimumPeriod = 1/(CommunicationSpeed[bps]) \times 180 + 70[us]$$

When the communication speed is 1Mbps, let 1Mbps into this formula.

$$MinimumPeriod = 1/(1[Mbps]) \times 180 + 70[us] = 250[us]$$

*Caution: The interrupt interval in the user software is a relation of other interrupt, and generating of interrupt may be delayed. Please also take that interrupt timing shifts into consideration and call it.

Interrupt function

This library uses these interrupts

INTST0, INTSR0
INTST1, INTSR1

```
#ifdef ICS_SCI0
__interrupt void   Excep_INTST0(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR0(void)   { int_ics_sci_rx(); }
#endif

#ifdef ICS_SCI1
__interrupt void   Excep_INTST1(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR1(void)   { int_ics_sci_rx(); }
#endif
```

3.4.3. RL78F14 series function usage

his document explains the setting method of the user program for using ICS, using attached sample software.

1) To secure the DTC table

There are some ways to keep the DTC table memory. We introduce the way we can check from the source code.

To keep the DTC table, please add the following description. This case keeps 0xD0 byte from address 0xFFE00. This address must keep 8bits of the lower ranks of the address.

```
#pragma section @@DATA   @@DTCTBL at 0xFFE00
char   dtc_tbl[0xD0];
#pragma section @@DATA   @@DATA
```

When you use emulator, such as E1 or something, please keep a user RAM domain, the domain of a DTC table and domain of E1 emulator from overlapping.

2) Cal ics_init()

Please put the initialization function “ics_init((void*)dtc_table, ICS_SCI0_P62_P61, 2, 0x40)” at the user initialization part.

First parameter is the address to be secured at 1).
Second parameter is the port name you want to use defined in the ICS_<CPUNAME>.h.
Third parameter is the interrupt level using in the ICS. Normally we choose the level lower than the carrier interrupt.
Fourth parameter is normally 0x40. If you don’t use other DTC channels.

```
----------- List 1   main.c -----------------------------------------------
#pragma   SFR
#pragma   DI
#pragma   EI
#pragma   NOP

#include "ICS_define.h"
#include "low_level_init.h"
#include "ics_R5F104LE.h"

/*********** KEEP DTC TABLE AREA ***********/
#pragma section @@DATA   @@DTCTBL at 0xFFE00
char   dtc_tbl[0xD0];
#pragma section @@DATA   @@DATA

    ics_init(0xFE00, 2, ICS_SCI1_P12_P11, 0x40);
```

3) Installation of ics_watchpoint() function

In this sample software, ics_watchpoint() function is called in the main routine. But normally this is called in the carrier interrupt.

And this function must be called below 5ms period and above 250us. If the carrier interrupt period is below 250us, please decimate function call of ics_watchpoint() as in the List 2.

----------- List 2 ics_watchpoint() ------------------------

```
__interrupt void    int_TM0(void)
{
        theta_e_est = theta_e_est + 60;
        if (theta_e_est>4095)
        {
                theta_e_est = theta_e_est - 4096;
        }

        /********** pwm reference generation **************/
        refu = R_FIX_sin_int16(theta_e_est);
        refv = R_FIX_sin_int16(theta_e_est-1333);
        refw = R_FIX_sin_int16(theta_e_est-2666);

        RPECTL = 0x80U;
        ics_watchpoint();
}
```

4) Add interrupt functions

The case of SCI0

```
__interrupt void    Excep_INTST0(void)    { int_ics_sci_tx(); }
__interrupt void    Excep_INTSR0(void)    { int_ics_sci_rx(); }
```

The case of SCI1

```
__interrupt void    Excep_INTST1(void)    { int_ics_sci_tx(); }
__interrupt void    Excep_INTSR1(void)    { int_ics_sci_rx(); }
```

3.4.4. ICS on board clock for RL78F14

When use this library, please choose the clock on an ICS board as follows according to a setup of the clock of the CPU side. In the case of the model which cannot change the on board clock of the ICS, please use the CLK=32MHz.

On board clock frequency of ICS = (CLK /4) MHz

Desk Top Laboratories is preparing the stock of 8.000MHz, 8.333MHz and 10.000MHz parts.

*Caution:

W1001 (No external clock module type)
This type can not change the clock, so you can use only 8MHz clock.

W1003 (Support external clock module)
In the case of using ICS clock except 8MHz, you need to change clock module.

W1004 (Optical fiber type)
This model supports variable clock function, so you can change master clock from the PC software.

3.5. RL78G1F series

3.5.1. RL78F14 resources

CPU name	RL78G1F series (R5F11BLEACFB only)	
Develop evvironment	CS+ Ver.3.xx.xx	
Library version	Ver2.01	
Communication rate	$Communication\_rate = \frac{CLK}{32}[Mbps]$ Standard CLK = 32MHz ：1Mbps	
status	SCI0, SCI2	
Library type	16bit library	
Library file name	Library name depends on the CPU name Library name is affected IC package Example 64pin, ROM=64kB R5F11BLEAFB -> "ICS_RL78G1F_Lx.lib" 48pin, ROM=64kB R5F11BGEAFB -> "ICS_RL78G1F_Gx.lib" 36pin, ROM=64kB R5F11BCEALA -> "ICS_RL78G1F_Cx.lib" 32pin, ROM=64kB R5F11BBEAFP -> "ICS_RL78G1F_Bx.lib" 24pin, ROM=64kB R5F11B7EANA -> "ICS_RL78G1F_7x.lib"	
Header file name	Header file name rule is same as library. 64pin, ROM=64kB R5F11BLEAFB -> "ICS_RL78G1F_Lx.h" 48pin, ROM=64kB R5F11BGEAFB -> "ICS_RL78G1F_Gx.h" 36pin, ROM=64kB R5F11BCEALA -> "ICS_RL78G1F_Cx.h" 32pin, ROM=64kB R5F11BBEAFP -> "ICS_RL78G1F_Bx.h" 24pin, ROM=64kB R5F11B7EANA -> "ICS_RL78G1F_7x.h"	
Momory model	Medium (ROM=1MB, RAM=64kB)	
DTC address mode	Standard	
Endian	Little	
Support port	R5F11BLx (64pin)	#define ICS_SCI0_P51_P50 (0x00) #define ICS_SCI0_P17_P16 (0x01) #define ICS_SCI2_P77_P76 (0x20)
	R5F11BGx (48pin)	Not supported now
	R5F11BCx (36pin)	Not supported now
	R5F11BBx (32pin)	#define ICS_SCI0_P51_P50 (0x00)
	R5F11B7x (24pin)	Not supported now

Used CPU resources	Support ICS	Support variable type
・Used internal resources	1) Support ICS hardware	Numeric display
		8bit unsigned char
1) ICS_SCI0_P51_P50	*W1001	8bit signed char
DTC	H/W model 1	16bit unsigned short
SCI0 all resources	H/W Ver. 1	16bit signed short
PFC	S/W Ver. 1.22 (after)	32bit unsigned int
PIOR0.1 = 0		32bit signed int
P5.0 = 1	*W1003	8bit BOOL type
PM5.1 = 0	H/W model 4	8bit LOGIC type
PM5.0 = 1	H/W Ver. 1	
External pin	S/W Ver. 1.22 (after)	Waveform display
P51 for TXD0		8bit unsigned char
P50 for RXD0		8bit signed char
CLOCK	ICS PC software	16bit unsigned short
SPS0 bit4～7	After Ver. 2.5.0.0	16bit signed short
INTC	Recommendation	
STPR00	After Ver.2.7.3.0	
STPR10		
SRPR00		
SRPR10	2) Support ICS++ hardware	
2) ICS_SCI0_P17_P16	*W1004	
DTC	All version.	
SCI0 all resources		
PFC		
PIOR0.1 = 1		
P1.7 = 1		
PM1.7 = 0		
PM1.6 = 1		
External pin		
P17 for TXD0		
P16 for RXD0		
CLOCK		
SPS0 bit4～7		
INTC		
STPR00		
STPR10		
SRPR00		
SRPR10		
3) ICS_SCI2_P77_P76		
DTC		
SCI2 all resources		
PFC		

PIOR0.1 = 1 P7.7 = 1 PM7.7 = 0 PM7.6 = 1 External pin P77 for TXD2 P76 for RXD2 CLOCK SPS0 bit4～7 INTC STPR02 STPR12 SRPR02 SRPR12		

3.5.2. RL78G1F function library

Lib Ver.2.01 on CS+ Ver.3.xx.xx

Initialize function void ics_init(unsigned int addr, char pin, char level, unsigned char num);

This function initializes ICS relation including a pin definition. Be careful to destroy neither the definition of the resource pin used by ICS indicated for the preceding clause, nor a setup of a standby control register etc., after initialization of this function.

First parameter:

Please specify the head address of the 16bits of lower ranks of the vector table address of DTC. Before calling an ics_init() function, a user needs to secure a DTC vector table. 8bits of lower ranks of this address need to be '0'.

Second parameter:

The port number of SCI and the pins which SCI uses are set up. For this parameter, please use the string that is defined in the ICS_<CPUNAME>.h.

Example: (R5F11BLEAFB)

```
#define    ICS_SCI0_P51_P50    (0x00)
#define    ICS_SCI1_P17_P16    (0x01)
#define    ICS_SCI1_P77_P76    (0x20)
```

Third parameter:

Specify the interrupt level of SCI to be used in ICS. There is a need to meet the following conditions. There is a possibility that the 2ms interrupt occurs at the minimum interval, as a system, please set the interrupt level that can tolerate this interrupt interval. Receive interrupt of the SCI is the longest processing time. It is about 10us, but if there is an interrupt source that cannot tolerate interrupt disable time, please set the interrupt level higher than the interrupt level setting.

Fourth parameter:

The top address of the DTC struct. This parameter can be chosen from 0x40, 0x48, 0x50... 0xF8.

Transfer function void ics_watchpoint(void);

This is the data transfer function. Normally an user puts this function in the carrier interrupt function. However, in the sample software, to make it easier to understand how to write the software, it is written in the main routine. This function reads the data of the variable specified by the PC, and copy it to the transfer buffer for the DTC.

When the communication speed is 1Mbps, this function should maintain the interval of 250us or more of minimum, and less than 5ms and please call it. When the communication speed is not 1Mbps, please keep and call the time defined by the following formula.

$$MinimumPeriod = 1/(CommunicationSpeed[bps]) \times 180 + 70[us]$$

When the communication speed is 1Mbps, let 1Mbps into this formula.

$$MinimumPeriod = 1/(1[Mbps]) \times 180 + 70[us] = 250[us]$$

*Caution: The interrupt interval in the user software is a relation of other interrupt, and generating of interrupt may be delayed. Please also take that interrupt timing shifts into consideration and call it.

Interrupt function

This library uses these interrupts

INTST0, INTSR0
INTST1, INTSR1

```
#ifdef ICS_SCI0
__interrupt void   Excep_INTST0(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR0(void)   { int_ics_sci_rx(); }
#endif

#ifdef ICS_SCI2
__interrupt void   Excep_INTST2(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR2(void)   { int_ics_sci_rx(); }
#endif
```

3.5.3. RL78G1F series function usage

his document explains the setting method of the user program for using ICS, using attached sample software.

1) To secure the DTC table

There are some ways to keep the DTC table memory. We introduce the way we can check from the source code.

To keep the DTC table, please add the following description. This case keeps 0xD0 byte from address 0xFFE00. This address must keep 8bits of the lower ranks of the address.

```
#pragma section @@DATA   @@DTCTBL at 0xFFE00
char   dtc_tbl[0xD0];
#pragma section @@DATA   @@DATA
```

When you use emulator, such as E1 or something, please keep a user RAM domain, the domain of a DTC table and domain of E1 emulator from overlapping.

2) Cal ics_init()

Please put the initialization function “ics_init((void*)dtc_table, ICS_SCI0_P17_P16, 2, 0x40)” at the user initialization part.

First parameter is the address to be secured at 1).
Second parameter is the port name you want to use defined in the ICS_<CPUNAME>.h.
Third parameter is the interrupt level using in the ICS. Normally we choose the level lower than the carrier interrupt.
Fourth parameter is normally 0x40. If you don’t use other DTC   channels.

```
----------- List 1   main.c ----------------------------------------------
#pragma   SFR
#pragma   DI
#pragma   EI
#pragma   NOP

#include "ICS_define.h"
#include "low_level_init.h"
#include "ics_RL78G1F_Lx.h"

/*********** KEEP DTC TABLE AREA ***********/
#pragma section @@DATA   @@DTCTBL at 0xFFE00
char   dtc_tbl[0xD0];
#pragma section @@DATA   @@DATA

    ics_init(0xFE00, 2, ICS_SCI0_P17_P16, 0x40);
```

3) Installation of ics_watchpoint() function

In this sample software, ics_watchpoint() function is called in the main routine. But normally this is called in the carrier interrupt.

And this function must be called below 5ms period and above 250us. If the carrier interrupt period is below 250us, please decimate function call of ics_watchpoint() as in the List 2.

----------- List 2 ics_watchpoint() ------------------------

```
__interrupt void   int_TM0(void)
{
        theta_e_est = theta_e_est + 60;
        if (theta_e_est>4095)
        {
                theta_e_est = theta_e_est - 4096;
        }

        /********** pwm reference generation **************/
        refu = R_FIX_sin_int16(theta_e_est);
        refv = R_FIX_sin_int16(theta_e_est-1333);
        refw = R_FIX_sin_int16(theta_e_est-2666);

        RPECTL = 0x80U;
        ics_watchpoint();
}
```

4) Add interrupt functions

The case of SCI0

```
__interrupt void   Excep_INTST0(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR0(void)   { int_ics_sci_rx(); }
```

The case of SCI2

```
__interrupt void   Excep_INTST2(void)   { int_ics_sci_tx(); }
__interrupt void   Excep_INTSR2(void)   { int_ics_sci_rx(); }
```

3.5.4. ICS on board clock for RL78G1F

When use this library, please choose the clock on an ICS board as follows according to a setup of the clock of the CPU side. In the case of the model which cannot change the on board clock of the ICS, please use the CLK=32MHz.

On board clock frequency of ICS = (CLK /4) MHz

Desk Top Laboratories is preparing the stock of 8.000MHz, 8.333MHz and 10.000MHz parts.

*Caution:

W1001 (No external clock module type)
This type can not change the clock, so you can use only 8MHz clock.

W1003 (Support external clock module)
In the case of using ICS clock except 8MHz, you need to change clock module.

W1004 (Optical fiber type)
This model supports variable clock function, so you can change master clock from the PC software.

3.6. RX64M series

3.6.1. RX64M resources

CPU name	RX64M series
Develop environment	CubeSuite+ Ver.2.0x.xx
Library version	Ver.2.0
Communication rate	$Rate = \frac{PCLKB}{48}[Mbps]$ Standard Clock 1.25Mbps @PCLKB = 60MHz
Status	SCI0, SCI1, SCI12 support
Library type	32bit Library
Library file name	ics_RX64M.obj
Header file name	ics_RX64M.h

Used CPU resources	Support ICS	Support variable type
･Used internal resources CI0 (P32, P33) INT SCI0 RXI INT SCI0 TXI DTC INT59 (TXI0) ICU.DTCER[59].BIT.DTCE SCI0 (all registers) DTC (all registers) ICU.IPR[58].BYTE ICU.IPR[59].BYTE ICU.IER[0x07].BIT.IEN2 ICU.IER[0x07].BIT.IEN3 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRB.BIT.B31 MPC.P32PFS.BYTE MPC.P33PFS.BYTE PORT3.PMR.BIT.B2 = 1 PORT3.PMR.BIT.B3 = 1 External pin P32: TXD0 P33: RXD0 SCI1 (P16, P15) INT SCI1 RXI INT SCI1 TXI DTC INT61 (TXI1) ICU.DTCER[61].BIT.DTCE SCI1 (all registers)	Support ICS *W1001 H/W model 1 H/W Ver. 1 S/W Ver. 1.22 (after) *W1003 H/W model 4 H/W Ver. 1 S/W Ver. 1.22 (after) ICS PC software After Ver. 2.5.0.0	Numeric display 8bit unsigned char 8bit signed char 16bit unsigned short 16bit signed short 32bit unsigned int 32bit signed int 32bit IEEE754 floating point 8bit BOOL type 8bit LOGIC type Waveform display 8bit unsigned char 8bit signed char 16bit unsigned short 16bit signed short 32bit unsigned int 32bit signed int 32bit IEEE754 floating point

DTC　(all resisters) ICU.IPR[60].BYTE ICU.IPR[61].BYTE ICU.IER[0x07].BIT.IEN4 ICU.IER[0x07].BIT.IEN5 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRB.BIT.B30 MPC.P16PFS.BYTE MPC.P15PFS.BYTE PORT1.PMR.BIT.B6 = 1 PORT1.PMR.BIT.B5 = 1 External pin P16: TXD1 P15: RXD1 SCI2　(P13, P12) INT　SCI2RXI INT　SCI2TXI DTC　INT63　(TXI2) ICU.DTCER[63].BIT.DTCE SCI2　(all resisters) DTC　(all resisters) ICU.IPR[62].BYTE ICU.IPR[63].BYTE ICU.IER[0x07].BIT.IEN6 ICU.IER[0x07].BIT.IEN7 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRB.BIT.B29 MPC.P13PFS.BYTE MPC.P12PFS.BYTE PORT1.PMR.BIT.B3 = 1 PORT1.PMR.BIT.B2 = 1 External pin P13: TXD2 P12: RXD2		

3.6.2. RX64M function library

Lib Ver.2.0 on CubeSuite+ Ver.2.0x.xx

Initialize function void ics_init(void* addr, char port, char level);

This function initializes ICS relation including a pin definition. Be careful to destroy neither the definition of the resource pin used by ICS indicated for the preceding clause, nor a setup of a standby control register etc., after initialization of this function.

First parameter:
Please specify the head address of the vector table of DTC. Before calling an ics_init() function, a user needs to secure a DTC vector table. 12bits of lower ranks of this address need to be '0'.

Second parameter:
The port number of SCI and the pins which SCI uses are set up. For this parameter, please use the string that is defined in the ICS_<CPUNAME>.h.

Third parameter:
Please specify the interrupt level of SCI to be used in ICS. There is a need to meet the following conditions.
There is a possibility that the 2ms interrupt occurs at the minimum interval, as a system, please set the interrupt level that can tolerate this interrupt interval. Receive interrupt of the SCI is the longest processing time. It is about 10us, but if there is an interrupt source that cannot tolerate interrupt disable time, please set the interrupt level higher than the interrupt level setting.

Transfer function void ics_watchpoint(void);

This is the data transfer function. Normally an user puts this function in the carrier interrupt function. However, in the sample software, to make it easier to understand how to write the software, it is written in the main routine.
This function reads the data of the variable specified by the PC, and copy it to the transfer buffer for the DTC.
When the communication speed is 1Mbps, this function should maintain the interval of 250us or more of minimum, and less than 5ms and please call it. When the communication speed is not 1Mbps, please keep and call the time defined by the following formula.

$$MinimumPeriod = 1/(CommunicationSpeed[bps]) \times 180 + 70[us]$$

When the communication speed is 1Mbps, let 1Mbps into this formula.

$$MinimumPeriod = 1/(1[Mbps]) \times 180 + 70[us] = 250[us]$$

*Caution: The interrupt interval in the user software is a relation of other interrupt, and generating of interrupt may be delayed. Please also take that interrupt timing shifts into consideration and call it.

Interrupt functions

Since the following interrupt vector is used, please register the following function into the interrupt vector of user software. When you use the project automatically generated with the standard compiler for RENESAS, please add these functions to the file which indicated the interrupt processing “intprg.c”.

The case of SCI0

```
// SCI0 ERI0
void Excep_SCI0_RXI0(void){ ics_int_sci_rxi(); }
```

The case of SCI1

```
// SCI1 RXI1
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

The case of SCI2

```
// SCI2 RXI2
void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); }
```

3.6.3. RX64M functions usage

This document explains the setting method of the user program for using ICS, using attached sample software.

1) To secure the BDTCTBL section in the development environment.

The section of BDTCTBL is assigned as the address on RAM that 12 bits of low ranks are set to 0. This address is set as development environment and carried out. Here, please set up at 0x0000.
When you use emulator, such as E1 or something, please keep a user RAM domain, the domain of a DTC table and domain of E1 emulator from overlapping.

2) Define DTC table in user program

Please define the DTC table variable “unsigned long dtc_table[256];
At the top of ICS_sample.c

```
#pragma section DTCTBL
unsigned long dtc_table[256];        // caution alignment 0x000
#pragma section
```

3) Call “ics_init()” as following

Please put the initialization function “ics_init((void*)dtc_table, ICS_SCI0_P32_P33, 6)”
at the user initialization part.

First parameter is the address to be secured at 1).
Second parameter is the port name you want to use defined in the ICS_<CPUNAME>.h.
Third parameter is the interrupt level using in the ICS. Normally we choose the level lower than the carrier interrupt.

----------- List 1 main.c ---

```
#pragma section DTCTBL
unsigned long dtc_table[256];        // caution alignment 0x000
#pragma section

void main(void)
{
        ics_init((void*)dtc_table, ICS_SCI0_P32_P33, 6);      /* Interrupt level 6              */
        while(1)
        {    nop();    }
}
```

4) Installation of ics_watchpoint() function

In this sample software, ics_watchpoint() function is called in the main routine. But normally this is called in the carrier interrupt.

And this function must be called below 5ms period and above 250us. If the carrier interrupt period is below 250us, please decimate function call of ics_watchpoint() as in the List 2.

----------- List 2 ics_watchpoint() decimation ------------------------

```
int     deci = 0;

void    int_TM0(void)      /* 100us period */
{
          deci = deci + 1;
          if (deci >=3)
          {
               deci = 0;
               ics_watchpoint();
          }
}
```

5) Modification of "intprg.c"

The case of SCI0

```
// SCI0 RXI0
void Excep_SCI0_RXI0(void){ ics_int_sci_rxi(); }
```

The case of SCI1

```
// SCI1 RXI1
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

The case of SCI2

```
// SCI2 RXI2
void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); }
```

3.6.4. ICS on board clock for RX64M

When use this library, please choose the clock on an ICS board as follows according to a setup of the clock of the CPU side. In the case of the model which cannot change the on board clock of the ICS, please use the PCLK=96MHz.

On board clock frequency of ICS = (PCLKB / 6) MHz

Example:

The case of PCLKB = 60MHz: ICS CLOCK = 60/6 = 10.000MHz
The case of PCLKB = 48MHz: ICS CLOCK = 48/6 = 8.000MHz

Desk Top Laboratories is preparing the stock of 8.000MHz, 8.333MHz and 10.000MHz parts.

*Caution:

W1001 (No external clock module type)
This type can not change the clock, so you can use only 8MHz clock.

W1003 (Support external clock module)
In the case of using ICS clock except 8MHz, you need to change clock module.

W1004 (Optical fiber type)
This model supports variable clock function, so you can change master clock from the PC software.

3.7. V850E2M/FJ4 series

3.7.1. V850E2M/FJ4 resources

CPU name	V850E2M/Fx4 series
Develop environment	CubeSuite+ Ver.2.02.00
Library version	Ver.2.0
Communication rate	$Rate = \frac{PCLK}{80}[Mbps]$ Standard Communication rate 1Mbps @PCLK = 80MHz
Status	UARTE4, UARTE5, UARTE10, UARTE11
Library type	32bit Library
Library file name	ics_V850FJ4.obj
Header file name	ics_V850FJ4.h

<table>
<tr><th>Used CPU resources</th><th>Support ICS</th><th>Support variable type</th></tr>
<tr><td>・Used internal resources
UARTE4
INT INTLMA4IR
INT INTLMA4IS
DMA3 INTLMA4IT
ICU.DTCER[215].BIT.DTCE
UARTE4 全て
DMA3 全て

/* Set URTE4RX pin */
FCLA27CTL2 = 0x80U;
PFC1 |= 0x0200U;
PFCE1 |= 0x0200U;
PMC1 |= 0x0200U;
PM1 |= 0x0200U;
/* Set URTE4TX pin */
PFC1 |= 0x0100U;
PFCE1 |= 0x0100U;
PMC1 |= 0x0100U;
PM1 &= (~0x0100U);

UARTE5
INT INTLMA5IR
INT INTLMA5IS
DMA3 INTLMA5IT
UARTE5 全て
DMA3 全て

/* Set URTE5RX pin */
FCLA27CTL3 = 0x80U;
PFC25 &= (~0x4000U);</td><td>Support ICS

*W1001
H/W model 1
H/W Ver. 1
S/W Ver. 1.22 (after)

*W1003
H/W model 4
H/W Ver. 1
S/W Ver. 1.22 (after)

ICS PC software
After Ver. 2.5.0.0</td><td>Numeric display
8bit unsigned char
8bit signed char
16bit unsigned short
16bit signed short
32bit unsigned int
32bit signed int
32bit IEEE754 floating point
8bit BOOL type
8bit LOGIC type

Waveform display
8bit unsigned char
8bit signed char
16bit unsigned short
16bit signed short
32bit unsigned int
32bit signed int
32bit IEEE754 floating point</td></tr>
</table>

PFCE25 |= 0x4000U;
PMC25 |= 0x4000U;
PM25 |= 0x4000U;
/* Set URTE5TX pin */
PFC25 &= ~(0x8000U);
PFCE25 |= 0x8000U;
PMC25 |= 0x8000U;
PM25 &= ~(0x8000U);
 	
UARTE10
INT INTLMA10IR
INT INTLMA10IS
DMA3 INTLMA10IT
UARTE10 全て
DMA3 全て
 	
/* Set URTE10RX pin */
FCLA7CTL0 = 0x80U;
PFC4 |= 0x0010U;
PFCE4 &= (~0x0010U);
PMC4 |= 0x0010U;
PM4 |= 0x0010U;
/* Set URTE10TX pin */
PFC4 |= 0x0008U;
PFCE4 &= (~0x0008U);
PMC4 |= 0x0008U;
PM4 &= (~0x0008U);
 	
UARTE11
INT INTLMA11IR
INT INTLMA11IS
DMA3 INTLMA11IT
UARTE11 全て
DMA3 全て
 	
/* Set URTE11RX pin */
FCLA7CTL1 = 0x80U;
PFC0 &= (~0x0080U);
PFCE0 &= (~0x0080U);
PMC0 |= 0x0080U;
PM0 |= 0x0080U;
/* Set URTE11TX pin */
PFC0 &= (~0x0040U);
PFCE0 &= (~0x0040U);
PMC0 |= 0x0040U;
PM0 &= (~0x0040U);		

3.7.2. V850E2M/FJ4 function library

Lib Ver.2.0 on CubeSuite+ Ver.2.02.00

Initialize function void ics_init(unsigned char port, unsigned char level);

This function initializes ICS relation including a pin definition. Be careful to destroy neither the definition of the resource pin used by ICS indicated for the preceding clause, nor a setup of a standby control register etc., after initialization of this function.

First parameter:

The port number of SCI and the pins which SCI uses are set up. For this parameter, please use the string that is defined in the ICS_<CPUNAME>.h.

Second parameter:

Please specify the interrupt level of SCI to be used in ICS. There is a need to meet the following conditions.

There is a possibility that the 2ms interrupt occurs at the minimum interval, as a system, please set the interrupt level that can tolerate this interrupt interval. Receive interrupt of the SCI is the longest processing time. It is about 10us, but if there is an interrupt source that cannot tolerate interrupt disable time, please set the interrupt level higher than the interrupt level setting.

Transfer function void ics_watchpoint(void);

This is the data transfer function. Normally an user puts this function in the carrier interrupt function. However, in the sample software, to make it easier to understand how to write the software, it is written in the main routine.

This function reads the data of the variable specified by the PC, and copy it to the transfer buffer for the DTC.

When the communication speed is 1Mbps, this function should maintain the interval of 250us or more of minimum, and less than 5ms and please call it. When the communication speed is not 1Mbps, please keep and call the time defined by the following formula.

$$MinimumPeriod = 1/(CommunicationSpeed[bps]) \times 180 + 70[us]$$

When the communication speed is 1Mbps, let 1Mbps into this formula.

$$MinimumPeriod = 1/(1[Mbps]) \times 180 + 70[us] = 250[us]$$

*Caution: The interrupt interval in the user software is a relation of other interrupt, and generating of interrupt may be delayed. Please also take that interrupt timing shifts into consideration and call it.

Interrupt functions

Since the following interrupt vector is used, please register the following function into the interrupt vector of user software.

The case of UARTE4

```
// UARTE4 RXI4, ERI4
```

```
#pragma interrupt INTLMA4IR R_UARTE4_Interrupt_Receive multi
void R_UARTE4_Interrupt_Receive(void) { ics_int_sci_rxi(); }
#pragma interrupt INTLMA4IS R_UARTE4_Interrupt_Error
void R_UARTE4_Interrupt_Error(void) { ics_int_sci_eri(); }
```

3.7.3. RX63U functions usage

This document explains the setting method of the user program for using ICS, using attached sample software.

1) Call “ics_init()” as following

Please put the initialization function “ics_init(ICS_UARTE4_P19_P110, 6)” at the user initialization part.

First parameter is the port name you want to use defined in the ICS_<CPUNAME>.h.

Second parameter is the interrupt level using in the ICS. Normally we choose the level lower than the carrier interrupt.

------------ List 1 main.c --

```
void main(void)
{
      ics_init(ICS_UARTE4_P19_P110,   6);      /* Interrupt level 6            */
      while(1)
      {   nop();   }
}
```

2) Installation of ics_watchpoint() function

In this sample software, ics_watchpoint() function is called in the main routine. But normally this is called in the carrier interrupt.

And this function must be called below 5ms period and above 250us. If the carrier interrupt period is below 250us, please decimate function call of ics_watchpoint() as in the List 2.

------------ List 2 ics_watchpoint() decimation -------------------------

```
int     deci = 0;

void   int_TM0(void)      /* 100us period */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics_watchpoint();
        }
}
```

3) Modification of a file of interrupt vector.

The case of UARTE4

```
#pragma interrupt INTLMA4IR R_UARTE4_Interrupt_Receive multi
void R_UARTE4_Interrupt_Receive(void) { ics_int_sci_rxi(); }
#pragma interrupt INTLMA4IS R_UARTE4_Interrupt_Error
void R_UARTE4_Interrupt_Error(void) { ics_int_sci_eri(); }
```

The case of UARTE5

```
#pragma interrupt INTLMA5IR R_UARTE5_Interrupt_Receive multi
void R_UARTE5_Interrupt_Receive(void) { ics_int_sci_rxi(); }
#pragma interrupt INTLMA5IS R_UARTE5_Interrupt_Error
void R_UARTE5_Interrupt_Error(void) { ics_int_sci_eri(); }
```

The case of UARTE10

```
#pragma interrupt INTLMA10IR R_UARTE10_Interrupt_Receive multi
void R_UARTE10_Interrupt_Receive(void) { ics_int_sci_rxi(); }
#pragma interrupt INTLMA10IS R_UARTE10_Interrupt_Error
void R_UARTE10_Interrupt_Error(void) { ics_int_sci_eri(); }
```

The case of UARTE11

```
#pragma interrupt INTLMA11IR R_UARTE11_Interrupt_Receive multi
void R_UARTE11_Interrupt_Receive(void) { ics_int_sci_rxi(); }
#pragma interrupt INTLMA11IS R_UARTE11_Interrupt_Error
void R_UARTE11_Interrupt_Error(void) { ics_int_sci_eri(); }
```

3.7.4. ICS on board clock for V850E2/Fx4

When use this library, please choose the clock on an ICS board as follows according to a setup of the clock of the CPU side. In the case of the model which cannot change the on board clock of the ICS, please use the PCLK=80MHz.

On board clock frequency of ICS = (PCLK / 10) MHz

Example:
The case of PCLK = 80MHz: ICS CLOCK = 80/10 = 8.000MHz

Desk Top Laboratories is preparing the stock of 8.000MHz, 8.333MHz and 10.000MHz parts.

*Caution:

W1001 (No external clock module type)
This type can not change the clock, so you can use only 8MHz clock.

W1003 (Support external clock module)
In the case of using ICS clock except 8MHz, you need to change clock module.

W1004 (Optical fiber type)
This model supports variable clock function, so you can change master clock from the PC software.

3.8. RX63T series

3.8.1. RX63T resources

CPU name	RX63T series
Develop environment	CS+ Ver.3.00.00
Library version	Ver.2.0 / Ver.2.1
Communication rate	$Rate = \frac{PCLK}{48}[Mbps]$ Standard Clock 1.00Mbps @PCLK = 48MHz
Status	SCI0, SCI1, SCI12 support
Library type	32bit Library
Library file name	ics_RX63T.obj
Header file name	ics_RX63T.h

Used CPU resources	Support ICS	Support variable type
・Used internal resources SCI0 (PB2, PB1) INT SCI0 RXI INT SCI0 TXI DTC INT215 (TXI0) ICU.DTCER[215].BIT.DTCE SCI0 (all resisters) DTC (all resisters) ICU.IPR[214].BYTE ICU.IER[0x1A].BIT.IEN6 ICU.IER[0x1A].BIT.IEN7 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRB.BIT.B31 MPC.PB2PFS.BYTE MPC.PB1PFS.BYTE PORTB.PMR.BIT.B2 = 1 PORTB.PMR.BIT.B1 = 1 External pin PB2: TXD0 PB1: RXD0 SCI2 (P02, P03) INT SCI2RXI INT SCI2TXI DTC INT221 (TXI2) ICU.DTCER[221].BIT.DTCE SCI2 (all resisters) DTC (all resisters) ICU.IPR[220].BYTE	Support ICS *W1001 H/W model 1 H/W Ver. 1 S/W Ver. 1.22 (after) *W1003 H/W model 4 H/W Ver. 1 S/W Ver. 1.22 (after) ICS PC software After Ver. 2.5.0.0	Numeric display 8bit unsigned char 8bit signed char 16bit unsigned short 16bit signed short 32bit unsigned int 32bit signed int 32bit IEEE754 floating point 8bit BOOL type 8bit LOGIC type Waveform display 8bit unsigned char 8bit signed char 16bit unsigned short 16bit signed short 32bit unsigned int 32bit signed int 32bit IEEE754 floating point

ICU.IER[0x1B].BIT.IEN4 ICU.IER[0x1B].BIT.IEN5 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRB.BIT.B29 MPC.P02PFS.BYTE MPC.P03PFS.BYTE PORT0.PMR.BIT.B2 = 1 PORT0.PMR.BIT.B3 = 1 External pin P02: TXD2 P03: RXD2 Support from Ver.2.01 SCI2 (PG0, PG1) INT SCI2RXI INT SCI2TXI DTC INT221 (TXI2) ICU.DTCER[221].BIT.DTCE SCI2 (all registers) DTC (all registers) ICU.IPR[220].BYTE ICU.IER[0x1B].BIT.IEN4 ICU.IER[0x1B].BIT.IEN5 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRB.BIT.B29 MPC.PG1PFS.BYTE MPC.PG0PFS.BYTE PORTG.PMR.BIT.B1 = 1 PORTG.PMR.BIT.B0 = 1 External pins PG0: TXD2 PG1: RXD2 SCI3 (P35, P34) INT SCI3 RXI INT SCI3 TXI DTC INT224 (TXI3) ICU.DTCER[224].BIT.DTCE SCI3 (all registers) DTC (all registers) ICU.IPR[223].BYTE ICU.IER[0x1B].BIT.IEN7 ICU.IER[0x1C].BIT.IEN0 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRB.BIT.B28 MPC.P35PFS.BYTE		

MPC.P34PFS.BYTE PORT3.PMR.BIT.B5 = 1 PORT3.PMR.BIT.B4 = 1 External pin P35: TXD3 P34: RXD3		

3.8.2. RX63T function library

Lib Ver.2.0 on CS+ Ver.3.00.00

Initialize function void ics_init(void* addr, char port, char level);

This function initializes ICS relation including a pin definition. Be careful to destroy neither the definition of the resource pin used by ICS indicated for the preceding clause, nor a setup of a standby control register etc., after initialization of this function.

First parameter:
Please specify the head address of the vector table of DTC. Before calling an ics_init() function, a user needs to secure a DTC vector table. 12bits of lower ranks of this address need to be ‘0’.

Second parameter:
The port number of SCI and the pins which SCI uses are set up. For this parameter, please use the string that is defined in the ICS_<CPUNAME>.h.

Third parameter:
Please specify the interrupt level of SCI to be used in ICS. There is a need to meet the following conditions.
There is a possibility that the 2ms interrupt occurs at the minimum interval, as a system, please set the interrupt level that can tolerate this interrupt interval. Receive interrupt of the SCI is the longest processing time. It is about 10us, but if there is an interrupt source that cannot tolerate interrupt disable time, please set the interrupt level higher than the interrupt level setting.

Transfer function void ics_watchpoint(void);

This is the data transfer function. Normally an user puts this function in the carrier interrupt function. However, in the sample software, to make it easier to understand how to write the software, it is written in the main routine.
This function reads the data of the variable specified by the PC, and copy it to the transfer buffer for the DTC.
When the communication speed is 1Mbps, this function should maintain the interval of 250us or more of minimum, and less than 5ms and please call it. When the communication speed is not 1Mbps, please keep and call the time defined by the following formula.

$$MinimumPeriod = 1/(CommunicationSpeed[bps]) \times 180 + 70[us]$$

When the communication speed is 1Mbps, let 1Mbps into this formula.

$$MinimumPeriod = 1/(1[Mbps]) \times 180 + 70[us] = 250[us]$$

*Caution: The interrupt interval in the user software is a relation of other interrupt, and generating of interrupt may be delayed. Please also take that interrupt timing shifts into consideration and call it.

Interrupt functions

Since the following interrupt vector is used, please register the following function into the interrupt vector of user software. When you use the project automatically generated with the standard compiler for RENESAS, please add these functions to the file which indicated the interrupt processing “intprg.c”.

The case of SCI0

```
// SCI0 ERI0
void Excep_SCI0_RXI0(void){ics_int_sci_rxi();}
void Excep_SCI0_TXI0(void){ ics_int_sci_txi(); }
```

The case of SCI1

```
// SCI1 RXI1
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
void Excep_SCI1_TXI1(void){ ics_int_sci_txi(); }
```

The case of SCI2

```
// SCI2 RXI2
void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); }
void Excep_SCI2_TXI2(void){ ics_int_sci_txi(); }
```

3.8.3. RX63T functions usage

This document explains the setting method of the user program for using ICS, using attached sample software.

1) To secure the BDTCTBL section in the development environment.

The section of BDTCTBL is assigned as the address on RAM that 12 bits of low ranks are set to 0. This address is set as development environment and carried out. Here, please set up at 0x0000.

When you use emulator, such as E1 or something, please keep a user RAM domain, the domain of a DTC table and domain of E1 emulator from overlapping.

2) Define DTC table in user program

Please define the DTC table variable “unsigned long dtc_table[256];
At the top of ICS_sample.c

```
#pragma section DTCTBL
unsigned long dtc_table[256];        // caution alignment 0x000
#pragma section
```

3) Call “ics_init()” as following

Please put the initialization function “ics_init((void*)dtc_table, ICS_SCI0_P32_P33, 6)”
at the user initialization part.

First parameter is the address to be secured at 1).
Second parameter is the port name you want to use defined in the ICS_<CPUNAME>.h.
Third parameter is the interrupt level using in the ICS. Normally we choose the level lower than the carrier interrupt.

------------ List 1 main.c --

```
#pragma section DTCTBL
unsigned long dtc_table[256];        // caution alignment 0x000
#pragma section

void main(void)
{
        ics_init((void*)dtc_table, ICS_SCI0_PB2_PB1, 6);     /* Interrupt level 6              */
        while(1)
        {    nop();    }
}
```

4) Installation of ics_watchpoint() function

In this sample software, ics_watchpoint() function is called in the main routine. But normally this is called in the carrier interrupt.

And this function must be called below 5ms period and above 250us. If the carrier interrupt period is below 250us, please decimate function call of ics_watchpoint() as in the List 2.

------------ List 2 ics_watchpoint() decimation -------------------------

```
int     deci = 0;

void    int_TM0(void)      /* 100us period */
{
          deci = deci + 1;
          if (deci >=3)
          {
               deci = 0;
               ics_watchpoint();
          }
}
```

5) Modification of “intprg.c”

The case of SCI0

```
// SCI0 RXI0
void Excep_SCI0_RXI0(void){ ics_int_sci_rxi(); }
void Excep_SCI0_TXI0(void){ ics_int_sci_txi(); }
```

The case of SCI1

```
// SCI1 RXI1
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
void Excep_SCI1_TXI1(void){ ics_int_sci_txi(); }
```

The case of SCI2

```
// SCI2 RXI2
void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); }
void Excep_SCI2_TXI2(void){ ics_int_sci_txi(); }
```

3.8.4. ICS on board clock for RX63T

When use this library, please choose the clock on an ICS board as follows according to a setup of the clock of the CPU side. In the case of the model which cannot change the on board clock of the ICS, please use the PCLK=96MHz.

On board clock frequency of ICS = (PCLK / 6) MHz

Example:
The case of PCLK = 50MHz: ICS CLOCK = 50/6 = 8.333MHz
The case of PCLK = 48MHz: ICS CLOCK = 48/6 = 8.000MHz

Desk Top Laboratories is preparing the stock of 8.000MHz, 8.333MHz and 10.000MHz parts.

*Caution:

W1001 (No external clock module type)
This type can not change the clock, so you can use only 8MHz clock.

W1003 (Support external clock module)
In the case of using ICS clock except 8MHz, you need to change clock module.

W1004 (Optical fiber type)
This model supports variable clock function, so you can change master clock from the PC software.

4. Revision history

Version	Date	Note
Ver.1.02	2013-11-06	・First English version release
Ver.1.03	2014-01-06	・Add RX111 libary
Ver.1.04	2014-02-10	・Add RL78G14 library ・Add RL78F14 library
Ver.1.06	2014-02-25	・Add RX63U series library
Ver.1.07	2014-03-12	・Add RX64M series library
Ver.1.08	2014-06-18	・Remove RX63U ・Add V850E2M/FJ4
Ver.1.09	2014-10-03	・Add RX63T
Ver.1.12		・Add RX63T new support port
Ver.1.13	2015-10-14	・Add RL78/G1F support

ICS Library function manual

Issue date: 14-Oct-2015 Ver.1.13EN.

Issue: Desk Top Laboratories Inc.
101, 35-7, Matsugi, Hachiouji-shi, TOKYO, JAPAN, 1920362