Desk Top Lab

ICS++ library Ver.3.6x
Function reference manual

RENESAS CS+ CC-RX compiler

Index

1. Introduction

1.1. Introduction

This document is a manual for ICS series W1001, W1002, W1003, T2001B, and ICS++ series W1004, W2001, W2002. T2001C, T2006.

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 Desk Top Laboratories Inc.

2. Difference between ICS/ICS++ models

2.1. Introduction of ICS / ICS++ series

There are many kinds of ICS / ICS++ series, as described below. Please understand the name of your ICS according to the explanation below and read the explanation of the following function.

2.1.1. ICS++ W2002 series

It is a new type of ICS ++ series that connects by optical fiber.
It supports a range of 0.5 Mbps to 8 Mbps. In addition, it supports 12ch mode.

Fig 1 W2002 ICS++

This model is described on W2002 and seal on the board. Some lots for initial shipment may not have a seal. In these cases, it is possible to distinguish by the number stated with silk on the board or the model number displayed by software DTLScope on the PC.
Discrimination by silk: When there is description as P00301-D1-009, it becomes W2002.

2.1.2. Subset ICS++ on T2001C / T2006A

ICS ++ installed in T2001C / T2006A is classified as W2002 series.

The main difference from W2002 is the memory length, there are two differences from T2001C / T2006A.
1) Record length up to 1024 points
2) Waveform Display Up to 8 channels
The functions are restricted as described above.

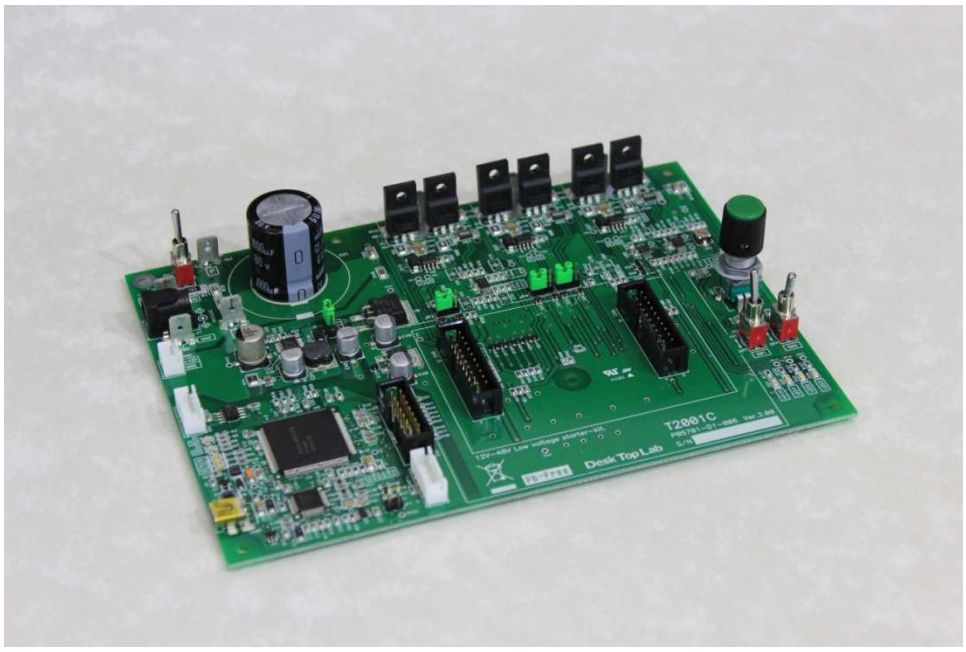


Fig 2　T2001C　Low voltage inverter（Successor model of T2001B）

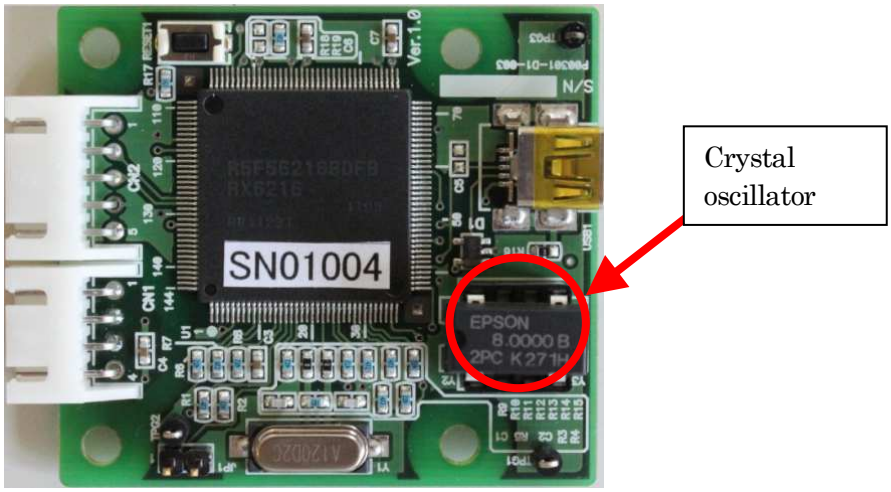
Fig 3　T2006A　Low voltage inverter（Three phase inverter 3 port verion）

2.1.3. ICS++ W1004 series (Obsolete)

It is ICS ++ series of type connected by optical fiber.
The communication rate of the target CPU is set to 0.5 Mbps to 1.25 Mbps.

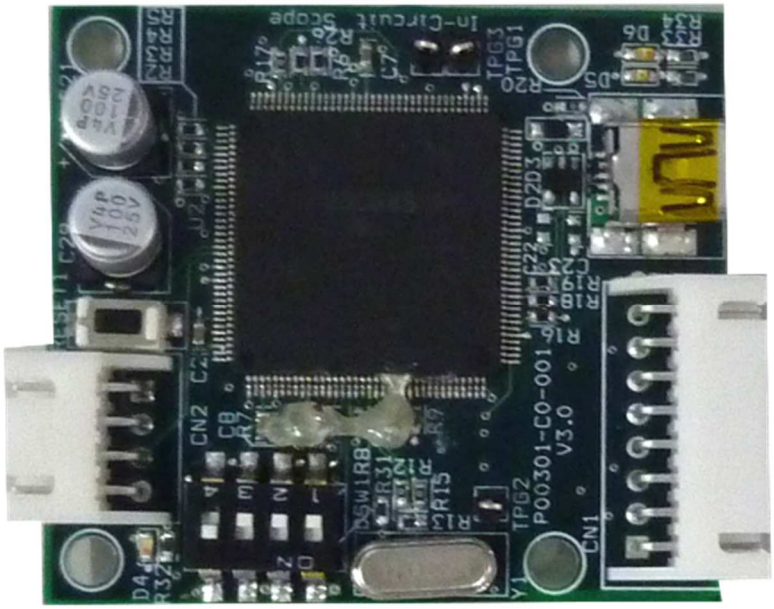
2.1.4. ICS++ W1003 series (Obsolete)

ICS of the type which fixes the communication rate by exchanging the crystal oscillator mounted on the socket on the board like the picture below.



2.1.5. ICS++ W1001 series (Obsolete)

It is an ICS of 1 Mbps fixed type like the picture below.

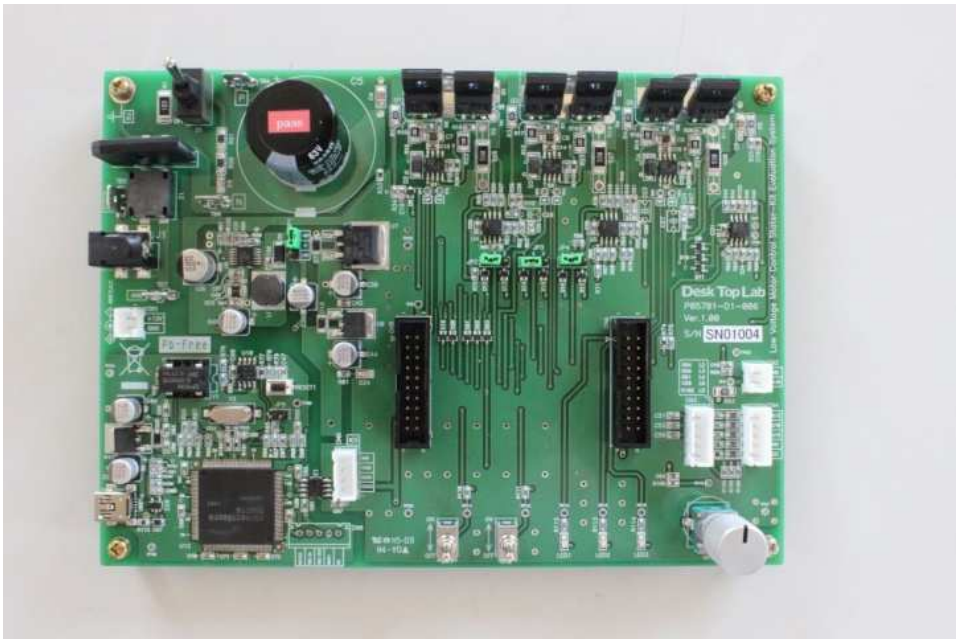


2.1.1. Subset ICS++ on T2001B / T2002B (Obsolete)

ICS installed in T2001B / T2002B is classified as W1003 series.

ICS installed in T2001B, T2002B is a subset of W1003.

Because it is a tool for positioning as a trial version, the record length is very short, up to 1024 points.



2.2. Difference in function of each series

Table 1 ICS / ICS++ Specifications

	ICS series W1001	ICS series W1003 T2002B T2001B	ICS++ series W1004	ICS++ series W2002 T2001C T2006A
Communication speed	1Mbps 固定	0.5Mbps～1.25Mbps	0.5Mbps～1.25Mbps	0.5Mbps～8Mbps
Max channel	8ch	8ch	8ch	W2002: 12ch T2001C, T2006A 8ch
Isolation	Isolation by IC	Isolation by IC	Optical fiber	W2002: Optical fiber T2001C, T2006A ; Isolatio by IC
USB speed	11Mbps	11Mbps	11Mbps	480Mbps
PC soft	InCircuitScope DTLScope	InCircuitScope DTLScope	DTLScope.exe	DTLScope.exe

2.3. Setting the transfer rate

When using the library, it is necessary to decide the transfer rate. Normally, it is better to set the communication rate as fast as possible, but it is restricted by the ICS / ICS ++ hardware to be used, the type of CPU used, and the clock frequency. Normally, set the highest communication rate by the following procedure.

2.4. ICS / ICS++ hardware constraints

As shown in "Table 1 ICS / ICS ++ Specifications", the maximum transferable rate varies depending on each hardware. Please set the communication rate so that it falls within the range of this constraint.

2.5. Target CPU / clock constraints

Depending on each CPU, the clock frequency actually used, and the library version, the settable frequencies exist intermittently. For example, for RX23T, it is as follows.

$$Communicat\ ionRate = \frac{PCLKB}{8 \times (speed + 1)}[Mbps]$$

Here, PCLKB is the clock frequency of RX23T actually used. “speed” is an integer value greater than or equal to 0.

2.6. Communication rate setting example in actual system

Example A) RX23T　When PCLKB = 40 MHz,
The communication rate is as shown in the table below.

Speed	Communication rate
0	5Mbps
1	2.5Mbps
2	1.67Mbps
3	1.25Mbps
4	1Mbps
5	0.833Mbps

In the case of W 1003
Since 0.5 Mbps to 1.25 Mbps can be selected, 1 Mbps is selected.

In the case of W1004
Since 0.5 Mbps to 1.25 Mbps can be selected, 1 Mbps or 1.25Mbps is selected.

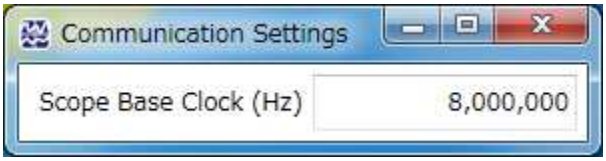
In the case of W 2002,
Since 0.5 Mbps to 8 Mbps can be selected, select 5 Mbps.

2.7. How to set the communication rate to ICS ++ hardware

When using this library, select the clock on the ICS ++ board as follows according to the setting of the clock on the CPU side.

2.7.1. In the case of W1004, W2001, W2002, T2001C, T2006A

Since the variable clock is built in, operation from the PC side becomes possible.
Please set the frequency which is 8 times the communication rate with PC software (DTLScope.exe).
Launch DTLScope.exe,
Settings -> Communication Settings
When you click, the following window will be displayed.
Please enter a value 8 times the communication rate below.



2.7.2. In the case of W1001

Since the clock is fixed, it can not be used with a clock other than the communication clock 8 MHz.

2.7.3. In the case of W1003

It is possible to change the clock by replacing the crystal oscillator mounted on the socket on the board. Replace with a crystal oscillator module with a frequency eight times the communication rate.

The calculation method of the set clock frequency is as follows

It is necessary to set the communication rate to 1.25 Mbps or less. Please replace the crystal oscillator which is 8 times the selected clock with the crystal oscillator on the board. In the desktop laboratory, stocks of 8.000 MHz, 8.333 MHz, 10.000 MHz are prepared as standard products.

The recommended part is EPSON SG - 8002 DC 3.3 V type.
This recommended part can be purchased with Digikey. Frequency can be specified.

3. ICS++ library overview

3.1. 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.

3.2. 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:

1) In the case of W2002, T2001C, T2006A
   Minimum time = 180/(Communication rate[Mbps]) + 30 [us]
   Example A) Min 210us @1Mbps
   Example B) Min 66us @5Mbps
2) In the case of W1001, W1003
   Minimum time = 180/(Communication rate[Mbps]) + 70 [us]
   Example A) Min 250us @1Mbps

Maximum calling period: 5ms

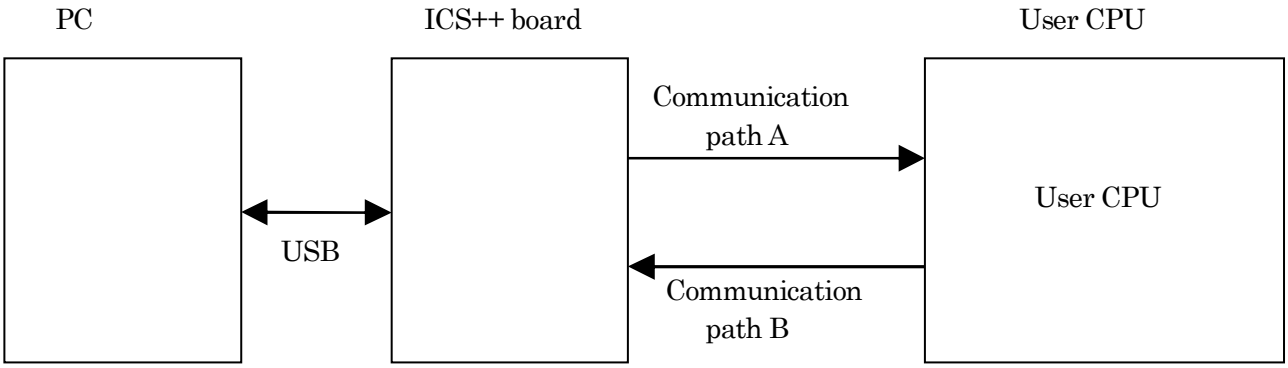


Fig. 1 ICS system structure

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.

3.3. Difference between Transfer mode 0, 1, 2, 3, 4, 5, 6

There are four transfer modes in ICS ++. Hereafter, it is called mode 0, mode 1, mode 2, mode 3. The difference between these modes is the maximum bit length supported by the waveform display and the difference how many times the ics2_watchpoint () function transfers data of one sampling. (This transfer mode will be extended in the future)

1) mode 0 (8/16bit mode)

For numerical display, it operates on all types of 8/16/32 bits. However, there are constraints on the type of waveform display. For 8-bit data, it is expanded to 16 bits according to the type of the variable, and if it is 16 bits, 8 channels are transferred at once without change. 32 bit data can not be transferred. Normally it is not supported by 32bit CPU. It can be used in all ICS models.

This mode transfers eight 16bit data at one time, when ics2_watchpoint() function is called.

2) mode 1 (8/16/32bit mode 8 channel two times transfer mode)

When the ics2_watchpoint() function is called, 8-bit, 16-bit, 32-bit data for 8 channels specified at the same time is captured. In addition, data for 4 channels is transferred. Next, when the ics2_watchpoint() function is called, it does not capture data and transfers the remaining 4 channels of data yet to be transferred. It can be used in all ICS models.

In other words, in the case of 32-bit 8-channel mode, the ics2_watchpoint () function is used twice to transfer eight channels at a time.

3) mode 2 (8/16/32bit 4 channel 1 time transfer mode)

When the ics2_watchpoint() function is called, function samples 4 channel data, and transfers 4ch data. And the next function call is the same. This mode supports only 4 channels waveform display function.

This mode is supported only W1004, W2001, W2002, T2001C and T2006.

4) mode 3 (8/16/32bit 12 channel 3 times transfer mode)

When the ics2_watchpoint () function is called, the 8-bit, 16-bit, 32-bit data for the specified 12 ch are loaded at once. In addition, data for 4 channels is transferred. Next, when the ics2_watchpoint () function is called, it does not capture data and transfers the remaining 4 channels of data not yet transferred. And the 3rd times the ics2_watchpoint () function is called, the last 4 ch data is transferred. This mode is supported only W2002, T2001C and T2006. (T2001C and T2006A support first 8channels.)

5) mode 4 (8/16 bit 15channel 2 times transfer mode)

For the numerical display function, it works on all 8bit, 16bit and 32bit data types. Waveform display function can work for 8bit and 16bit variables. But it cannot work for 32bit variables.

ics2_watchdpoint() function works followings. First function call samples 15 channel data, and transfers 8ch data, and the second function call transfers left of 7ch data. So two times function call send the one set of the sampling data.

***Causion This mode is supported after W2002 Firmware Ver1.2.**

6) mode 5 (Reserved for future use)

7) mode 6 (16bit only support 1 time transfer mode)

This mode is almost same the mode 0. But this mode doesn't support 8bit variables on waveform display. Instead, the execution time of the ics2_watchpoint() function is faster than mode 0.

	Merit	Demerit
Mode 0 8/16bit mode	Waveform update interval is short	Impossible to display 32bit variable waveform
Mode 1 8/16/32bit 8ch two times transfer mode	Possible to display 32bit variable waveform Possible to display 8 channels variable waveform	Waveform update time is twice of the 16bit library.
Mode 2 8/16/32bit 4ch 1 time transfer mode	Possible to display 32bit variable waveform. Waveform update interval is half of the mode 1.	Only 4 channel waveform support.
Mode 3 8/16/32bit 12ch 3 times transfer mode	This mode supports 12ch waveform display	
Mode 4 8/16bit 15ch 2 times mode		
Mode 5 Reserved for future use		
Mode 6 16bit 8ch 1 times mode		

16bit CPU for example RL78 series does not support "Mode 1/2/3/5".
32bit CPU for example RX series does not support "Mode 0/4/6"

3.4. 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.

3.5. Filename and library name

ICS++ library is made up of the following two files.
ics2_<CPUNAME>.h
ics2_<CPUNAME>.lib

And it is made up of the following two functions.
void ics2_init(void* addr, char unitpin, char level, char speed, char mode);
void ics2_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 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.

4. Resources and Library

4.1. RX13T series (CC compiler)

4.1.1. RX13T resources

CPU name	RX13T series
Develop environment	CS+ Ver.8.03.00　CC-RX 3.01.00
Library version	Ver.3.60
Communication rate	0.5Mbps – 4Mbps Transfer speed rate to be set $Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)} [Mbps]$　[speed≧0] Standard Clock　4.00Mbps　speed = 0　@PCLKB = 32MHz 1.00Mbps　speed = 3　@ PCLKB = 32MHz
Support port	#define　ICS_SCI1_PD3_PD5　(0x10) #define　ICS_SCI1_PB6_PB7　(0x11) #define　ICS_SCI5_PB6_PB7　(0x50) #define　ICS_SCI5_PB2_PB1　(0x51) #define　ICS_SCI5_P23_P24　(0x52) #define　ICS_SCI12_PB0_P94　(0xC0)
Library file name	ics2_RX13T.lib
Header file name	ics2_RX13T.h

Used CPU resources	Support variable type
・Used internal resources INT　SCIx RXI INT　SCIx TXI DTC　(TXIx) ICU.DTCER[xx].BIT.DTCE SCIx　(all resisters) DTC　(all resisters) ICU.IPR[xx].BYTE　corresponding part ICU.IER[xx]　corresponding part SYSTEM.MSTPCRA.BIT.Bxx　corresponding part SYSTEM.MSTPCRB.BIT.B31 MPC　corresponding part PORTx.　Corresponding part	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

4.1.2. RX13T function library

Initialize function　void ics2_init(void* addr, char　port,　char level, char speed, char mode);

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 ics2_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 ics2_<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 less than 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.

Forth parameter:

Transfer speed rate to be used in the ICS++ system. The way to calculate the frequency is following

$$Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)} [Mbps]$$

Fifth parameter:

Definition of the transfer　mode

0 : Do not use　(Reserved for future use)

1: 32bit 8 channel two times transfer mode

2: 32bit 4 channel one time transfer mode

This mode is supported on W1004, W2001, W2002, T2001C and T2006A.

3 : 32bit 12channel three times transfer mode

This mode is supported on W2002, T2001C and T2006A.

4 : Do not use　(Reserved for future use)

Transfer function	void	ics2_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.

Please keep and call the time defined by the following formula.

Case of W1001, W1003, ICS++ W1004:

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

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

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

Case of W2001, W2002, T2001 and T2006A

$$MinimumPeriod = 1/(Transfer\_speed\_rate[bps]) \times 180 + 30[us]$$

*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 ICS uses interrupt vector, please register the following functions into the interrupt vector of user software.

*Case1: When using a project that is automatically generated by the CC-RL compiler running on CS+
Please add these functions to the file which indicated the interrupt processing “intprg.c”.

The case of SCI1

```
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
```

*Case2: When using a project that is automatically generated by SmartConfigurator.
Please add four interrupt functions. Since SmartConfigurator does not generated SCI interrupt functions used by ICS. Pay attention to the vector number, since vector number is changed according to the SCI number.

```
#pragma interrupt (Excep_SCI5_ERI5(vect=222))
#pragma interrupt (Excep_SCI5_RXI5(vect=223))
#pragma interrupt (Excep_SCI5_TXI5(vect=224))
#pragma interrupt (Excep_SCI5_TEI5(vect=225))
void Excep_SCI5_ERI5(void){ /* no code */ }
void Excep_SCI5_RXI5(void){ ics_int_sci_rxi(); }
void Excep_SCI5_TXI5(void){ /* no code */ }
void Excep_SCI5_TEI5(void){ /* no code */ }
```

4.1.3. RX13T function usage

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

1) Place the DTC table.
Please use either method. In the example, we use A)

A) Place the DTC table at the absolute address using #pragma address direct directive.

```
#pragma address dtc_table=0x02000
uint32_t dtc_table[256];
```

B) In section specification of the development environment, specify the section address of dtc_table.

```
#pragma section DTCTBL
uint32_t    dtc_table[256];        // caution alignment 0x000
#pragma section
```

In CS+ specify the address of the BDTCTBL
Project Tree
- ➔ Build tool
- ➔ Property
- ➔ Link Option
- ➔ Section
- ➔ BDTCTBL

DTC table address must be placed at 12 bits of low ranks are set to 0.

2) Call “ics2_init()” as following

Please put the initialization function “ics2_init((void*)dtc_table, ICS_SCI5_PB6_PB7, 6, 0, 1)”
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 ICS2_<CPUNAME>.h.
Third parameter is the interrupt level using in the ICS. Normally we choose the level lower than the carrier interrupt.
Forth parameter is “2”.
Fifth parameter is normally “1”.

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

```
#pragma address dtc_table=0x02000
uint32_t dtc_table[256];

void main(void)
{
      ics2_init(dtc_table, ICS_SCI1_PB6_PB7, 6, 0, 1);
      while(1)
      {   nop();   }
}
```

3) Installation of ics2_watchpoint() function

In this sample software, ics2_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. (In the case of W1004). If the carrier interrupt period is below 250us, please decimate function call of ics2_watchpoint() as in the List 2.

---------- List 2 ics2_watchpoint() decimation ----------

```
int    deci = 0;

void   int_TM0(void)     /* 100us period */
{
       deci = deci + 1;
       if (deci >=3)
       {
          deci = 0;
          ics2_watchpoint();
       }
}
```

5) Modification of “intprg.c”

The case of SCI1

```
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

The case of SCI5

```
void Excep_SCI5_RXI5(void){ ics_int_sci_rxi(); }
```

*Caution

When using SmartConfigurator, the user also needs to add TXI, TEI, ERI functions according to the description of the interrupt function.

4.2. RX23T series (CC compiler)

4.2.1. RX23T resources

CPU name	RX23T series
Develop environment	CS+ Ver.6.00.00　CC-RX 2.07.00
Library version	Ver.3.60
Communication rate	0.5Mbps – 5Mbps Transfer speed rate to be set $Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)} [Mbps]$　[speed≧0] Standard Clock　1.25Mbps　speed = 3　@PCLKB = 40MHz 1.00Mbps　speed = 3　@ PCLKB = 32MHz
Support port	SCI1　TXD1:PD3,　RXD1:PD5 SCI5　TXD5:PB5,　RXD5:PB6 SCI5　TXD5:PB2,　RXD5:PB1
Library file name	ics2_RX23T.lib
Header file name	ics2_RX23T.h

Used CPU resources	Support variable type
・Used internal resources INT　SCIx RXI INT　SCIx TXI DTC　(TXIx) ICU.DTCER[xx].BIT.DTCE SCIx　(all resisters) DTC　(all resisters) ICU.IPR[xx].BYTE　corresponding part ICU.IER[xx]　corresponding part SYSTEM.MSTPCRA.BIT.Bxx　corresponding part SYSTEM.MSTPCRB.BIT.B31 MPC　corresponding part PORTx.　Corresponding part	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

4.2.2. RX23T function library

Initialize function　void ics2_init(void* addr, char　port,　char level, char speed, char mode);

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 ics2_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 ics2_<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 less than 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.

Forth parameter:

Transfer speed rate to be used in the ICS++ system. The way to calculate the frequency is following

$$Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)} [Mbps]$$

Fifth parameter:

Definition of the transfer　mode

0 : Do not use　(Reserved for future use)

1: 32bit 8 channel two times transfer mode

2: 32bit 4 channel one time transfer mode

This mode is supported on W1004, W2001, W2002, T2001C and T2006A.

3 : 32bit 12channel three times transfer mode

This mode is supported on W2002, T2001C and T2006A.

4 : Do not use　(Reserved for future use)

Transfer function	void	ics2_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.

Please keep and call the time defined by the following formula.

Case of W1001, W1003, ICS++ W1004:

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

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

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

Case of W2001, W2002, T2001 and T2006A

$$MinimumPeriod = 1/(Transfer\_speed\_rate[bps]) \times 180 + 30[us]$$

*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 ICS uses interrupt vector, please register the following functions into the interrupt vector of user software.

*Case1: When using a project that is automatically generated by the CC-RL compiler running on CS+
Please add these functions to the file which indicated the interrupt processing “intprg.c”.

The case of SCI1

```
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
```

*Case2: When using a project that is automatically generated by SmartConfigurator.
Please add four interrupt functions. Since SmartConfigurator does not generated SCI interrupt functions used by ICS. Pay attention to the vector number, since vector number is changed according to the SCI number.

```
#pragma interrupt (Excep_SCI5_ERI5(vect=222))
#pragma interrupt (Excep_SCI5_RXI5(vect=223))
#pragma interrupt (Excep_SCI5_TXI5(vect=224))
#pragma interrupt (Excep_SCI5_TEI5(vect=225))
void Excep_SCI5_ERI5(void){ /* no code */ }
void Excep_SCI5_RXI5(void){ ics_int_sci_rxi(); }
void Excep_SCI5_TXI5(void){ /* no code */ }
void Excep_SCI5_TEI5(void){ /* no code */ }
```

4.2.3. RX23T function usage

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

2) Place the DTC table.
Please use either method. In the example, we use A)

C) Place the DTC table at the absolute address using #pragma address direct directive.

```
#pragma address dtc_table=0x02000
uint32_t dtc_table[256];
```

D) In section specification of the development environment, specify the section address of dtc_table.

```
#pragma section DTCTBL
uint32_t    dtc_table[256];        // caution alignment 0x000
#pragma section
```

In CS+ specify the address of the BDTCTBL
Project Tree
- ➔ Build tool
- ➔ Property
- ➔ Link Option
- ➔ Section
- ➔ BDTCTBL

DTC table address must be placed at 12 bits of low ranks are set to 0.

2) Call “ics2_init()” as following

Please put the initialization function “ics2_init((void*)dtc_table, ICS_SCI5_PB2_PB1, 6, 2, 1)”
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.
Forth parameter is “2”.
Fifth parameter is normally “1”.

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

```
#pragma address dtc_table=0x02000
uint32_t dtc_table[256];

void main(void)
{
      ics2_init(dtc_table, ICS_SCI5_PB5_PB6, 6, 0, 1); // CN3
      while(1)
      {   nop();   }
}
```

3) Installation of ics2_watchpoint() function

In this sample software, ics2_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. (In the case of W1004). If the carrier interrupt period is below 250us, please decimate function call of ics2_watchpoint() as in the List 2.

---------- List 2 ics2_watchpoint() decimation ----------

```
int    deci = 0;

void   int_TM0(void)     /* 100us period */
{
       deci = deci + 1;
       if (deci >=3)
       {
          deci = 0;
          ics2_watchpoint();
       }
}
```

5) Modification of "intprg.c"

The case of SCI1

```
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
```

The case of SCI5

```
void Excep_SCI5_RXI5(void){ ics_int_sci_rxi(); }
void Excep_SCI5_ERI5(void){ ics_int_sci_eri(); }
```

*Caution

When using SmartConfigurator, the user also needs to add TXI, TEI, ERI functions according to the description of the interrupt function.

4.3. RX24T series (CC compiler)

4.3.1. RX24T resources

CPU name	RX24T series
Develop environment	CS+ Ver.6.00.00　CC-RX 2.07.00
Library version	Ver.3.60
Communication rate	0.5Mbps – 5Mbps Transfer speed rate to be set $Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)} [Mbps]$　[speed≧0] Standard Clock　1.25Mbps　speed = 3　@PCLKB = 40MHz 1.00Mbps　speed = 3　@ PCLKB = 32MHz
Support port	SCI1　TXD1:PD3,　RXD1:PD5 SCI5　TXD5:PB5,　RXD5:PB6 SCI6　TXD6:PB2,　RXD6:PB1 SCI6　TXD6:PB0,　RXD6:PA5 SCI6　TXD6:P81,　RXD6:P80
Library file name	ics2_RX24T.lib
Header file name	ics2_RX24T.h

Used CPU resources	Support variable type
・Used internal resources INT　SCIx RXI INT　SCIx TXI DTC　(TXIx) ICU.DTCER[xx].BIT.DTCE SCIx　(all resisters) DTC　(all resisters) ICU.IPR[xx].BYTE　corresponding part ICU.IER[xx]　corresponding part SYSTEM.MSTPCRA.BIT.Bxx　corresponding part SYSTEM.MSTPCRB.BIT.B31 MPC　corresponding part PORTx.　Corresponding part	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

4.3.2. RX24T function library

Initialize function　void ics2_init(void* addr, char　port,　char level, char speed, char mode);

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 ics2_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 ics2_<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.

Forth parameter:

Transfer speed rate to be used in the ICS++ system. The way to calculate the frequency is following

$$Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)} [Mbps]$$

Fifth parameter:

Definition of the transfer　mode

0 : Do not use　(Reserved for future use)

1: 32bit 8 channel two times transfer mode

2: 32bit 4 channel one time transfer mode

This mode is supported on W1004, W2001, W2002, T2001C and T2006A.

3 : 32bit 12channel three times transfer mode

This mode is supported on W2002, T2001C and T2006A.

4 : Do not use　(Reserved for future use)

Transfer function　　void　　ics2_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. Please keep and call the time defined by the following formula. Case of W1001, W1003, ICS++ W1004, T2001A/B, T2002A/B: $MinimumPeriod = 1/(Transdfer\_speed\_rate[bps]) \times 180 + 70[us]$ When the communication speed is 1Mbps, let 1Mbps into this formula. $MinimumPeriod = 1/(1[Mbps]) \times 180 + 70[us] = 250[us]$ Case of W2001, W2002, T2001C, T2006A $MinimumPeriod = 1/(Transfer\_speed\_rate[bps]) \times 180 + 30[us]$ *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 ICS uses interrupt vector, please register the following functions into the interrupt vector of user software. *Case1: When using a project that is automatically generated by the CC-RL compiler running on CS+ Please add these functions to the file which indicated the interrupt processing “intprg.c”. The case of SCI1 void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); } void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); } *Case2: When using a project that is automatically generated by SmartConfigurator. Please add four interrupt functions. Since SmartConfigurator does not generated SCI interrupt functions used by ICS. Pay attention to the vector number, since vector number is changed according to the SCI number. #pragma interrupt (Excep_SCI5_ERI5(vect=222)) #pragma interrupt (Excep_SCI5_RXI5(vect=223)) #pragma interrupt (Excep_SCI5_TXI5(vect=224)) #pragma interrupt (Excep_SCI5_TEI5(vect=225)) void Excep_SCI5_ERI5(void){ /* no code */ } void Excep_SCI5_RXI5(void){ ics_int_sci_rxi(); } void Excep_SCI5_TXI5(void){ /* no code */ } void Excep_SCI5_TEI5(void){ /* no code */ }

4.3.3. RX24T function usage

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

1) Place the DTC table.
Please use either method. In the example, we use A)

E) Place the DTC table at the absolute address using #pragma address direct directive.

```
#pragma address dtc_table=0x02000
uint32_t dtc_table[256];
```

F) In section specification of the development environment, specify the section address of dtc_table.

```
#pragma section DTCTBL
uint32_t    dtc_table[256];        // caution alignment 0x000
#pragma section
```

In CS+ specify the address of the BDTCTBL
Project Tree
- ➔ Build tool
- ➔ Property
- ➔ Link Option
- ➔ Section
- ➔ BDTCTBL

DTC table address must be placed at 12 bits of low ranks are set to 0.

2) Call “ics2_init()” as following

Please put the initialization function “ics2_init((void*)dtc_table, ICS_SCI5_PB5_PB6, 6, 2, 1)”
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.
Forth parameter is “2”.
Fifth parameter is normally “1”.

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

```
#pragma address dtc_table=0x02000
uint32_t dtc_table[256];

void main(void)
{
      ics2_init((void*)dtc_table, ICS_SCI5_PB5_PB6, 6, 0, 1);      /* Interrupt level 6              */
      while(1)
      {    nop();    }
}
```

3) Installation of ics2_watchpoint() function

In this sample software, ics2_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. (In the case of W1004). If the carrier interrupt period is below 250us, please decimate function call of ics2_watchpoint() as in the List 2.

---------- List 2 ics2_watchpoint() decimation ----------

```
int     deci = 0;

void    int_TM0(void)      /* 100us period */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics2_watchpoint();
        }
}
```

4) Modification of “intprg.c”

The case of SCI1

```
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
```

*Caution

When using SmartConfigurator, the user also needs to add TXI, TEI, ERI functions according to the description of the interrupt function.

4.4. RX26T series (CC compiler)

4.4.1. RX26T resources

CPU name	RX26T series
Develop environment	CS+ Ver.8.09.00　CC-RX 3.01.00
Library version	Ver.3.60
Communication rate	0.5Mbps – 7.5Mbps Transfer speed rate to be set $Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)} [Mbps]$　[speed≧0] Exception　1Mbps @speed=251, PCLKB=60MHz, Standard Clock　3.75Mbps　speed = 1　@ PCLKB = 60MHz
Support port	#define　ICS_SCI1_PD3_PD5　(0x10) #define　ICS_SCI5_PD7_PE0　(0x50) #define　ICS_SCI5_PB5_PB6　(0x51) #define　ICS_SCI6_P81_P80　(0x60) #define　ICS_SCI6_PB2_PB1　(0x61) #define　ICS_SCI12_PD4_PD6　(0xC0) #define　ICS_SCI12_P01_P00　(0xC1) #define　ICS_SCI12_P81_P80　(0xC2) #define　ICS_SCI12_P23_P22　(0xC3) #define　ICS_SCI12_PB5_PB6　(0xC4)
Library file name	ics2_RX26T.lib
Header file name	ics2_RX26T.h

Used CPU resources	Support variable type
・Used internal resources INT　SCIx RXI INT　SCIx TXI DTC　(TXIx) ICU.DTCER[xx].BIT.DTCE SCIx　(all resisters) DTC　(all resisters) ICU.IPR[xx].BYTE　corresponding part ICU.IER[xx]　corresponding part SYSTEM.MSTPCRA.BIT.Bxx　corresponding part MPC　corresponding part PORTx.　Corresponding part	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

4.4.2. RX26T function library

Initialize function void ics2_init(void* addr, char port, char level, char speed, char mode);

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 ics2_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 ics2_<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.

Forth parameter:

Transfer speed rate to be used in the ICS++ system. The way to calculate the frequency is following

$$Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)} [Mbps]$$

Fifth parameter:

Definition of the transfer mode

0 : Do not use (Reserved for future use)

1: 32bit 8 channel two times transfer mode

2: 32bit 4 channel one time transfer mode

This mode is supported on W1004, W2001, W2002, T2001C and T2006A.

3 : 32bit 12channel three times transfer mode

This mode is supported on W2002, T2001C and T2006A.

4 : Do not use (Reserved for future use)

Transfer function　　void　　ics2_watchpoint(void);
This is the data transfer function. Normally a 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. Please keep and call the time defined by the following formula. Case of W1001, W1003, ICS++ W1004, T2001A/B, T2002A/B: $MinimumPeriod = 1/(Transdfer\_speed\_rate[bps]) \times 180 + 70[us]$ When the communication speed is 1Mbps, let 1Mbps into this formula. $MinimumPeriod = 1/(1[Mbps]) \times 180 + 70[us] = 250[us]$ Case of W2001, W2002, T2001C, T2006A $MinimumPeriod = 1/(Transfer\_speed\_rate[bps]) \times 180 + 30[us]$ *Caution: The interrupt interval in the user software is a relation of another interrupt, and generating of interrupt may be delayed. Please also take that interrupt timing shifts into consideration and call it.
Interrupt functions
Since ICS uses interrupt vector, please register the following functions into the interrupt vector of user software. *Case1: When using a project that is automatically generated by the CC-RL compiler running on CS+ Please add these functions to the file which indicated the interrupt processing “intprg.c”. The case of SCI1 void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); } void Excep_SCI1_TXI1(void) { /* no code */ } void Excep_SCI5_RXI5(void){ ics_int_sci_rxi(); } void Excep_SCI5_TXI5(void){ /* no code */ }

4.4.3. RX26T function usage

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

2) Place the DTC table.
Please use either method. In the example, we use A)

G) Place the DTC table at the absolute address using #pragma address direct directive.

```
#pragma address dtc_table=0x0F000
uint32_t dtc_table[256];
```

H) In section specification of the development environment, specify the section address of dtc_table.

```
#pragma section DTCTBL
uint32_t    dtc_table[256];        // caution alignment 0x000
#pragma section
```

In CS+ specify the address of the BDTCTBL
Project Tree
- ➔ Build tool
- ➔ Property
- ➔ Link Option
- ➔ Section
- ➔ BDTCTBL

DTC table address must be placed at 12 bits of low ranks are set to 0.

2) Call “ics2_init()” as following

Please put the initialization function “ics2_init((void*)dtc_table, ICS_SCI1_PD3_PD5, 6, 1, 1)”
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.
Forth parameter is “1”.
Fifth parameter is normally “1”.

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

```
#pragma address dtc_table=0x0F000
uint32_t dtc_table[256];

void main(void)
{
      ics2_init((void*)dtc_table, ICS_SCI1_PD3_PD5, 6, 1, 1);      /* Interrupt level 6            */
      while(1)
      {   nop();   }
}
```

3) Installation of ics2_watchpoint() function

In this sample software, ics2_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. (In the case of W1004). If the carrier interrupt period is below 250us, please decimate function call of ics2_watchpoint() as in the List 2.

---------- List 2 ics2_watchpoint() decimation ----------

```
int     deci = 0;

void    int_TM0(void)      /* 100us period */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics2_watchpoint();
        }
}
```

4) Modification of “intprg.c”

The case of SCI1

```
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
void Excep_SCI1_TXI1(void){    }
```

*Caution

When using SmartConfigurator, the user also needs to add TXI functions according to the description of the interrupt function.

4.5. RX62T series (CC compiler)

4.5.1. RX62T resources

CPU name	RX62T series
Develop environment	CS+ Ver.6.00.00　CC-RX 2.07.00
Library version	Ver.3.60
Communication rate	0.5Mbps – 3.125Mbps Transfer speed rate to be set $Transfer\_speed\_rate = \frac{PCLKB}{16 \times (speed + 1)} [Mbps]$ [speed≧0] Standard Clock　1.00Mbps　speed = 2　@PCLKB = 48MHz
Support port	SCI0　TXD0:PB2,　RXD0:PB1 SCI1　TXD1:PD3,　RXD1:PD5 SCI2　TXD2:PB5,　RXD2:PB6 SCI2　TXD2:P81,　RXD2:P80
Library file name	ics2_RX62T.lib
Header file name	ics2_RX62T.h

Used CPU resources	Support variable type
・Used internal resources INT　SCIx RXI INT　SCIx TXI DTC　(TXIx) ICU.DTCER[xx].BIT.DTCE SCIx　(all resisters) DTC　(all resisters) ICU.IPR[xx].BYTE　corresponding part ICU.IER[xx].BIT.IEN7　corresponding part SYSTEM.MSTPCRA.BIT.Bxx　corresponding part SYSTEM.MSTPCRB.BIT.B31 MPC　corresponding part PORTx.　Corresponding part	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

4.5.2. RX62T function library

Initialize function　void ics2_init(void* addr, char　port,　char level, char speed, char mode);

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 ics2_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 ics2_<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.

Forth parameter:

Transfer speed rate to be used in the ICS++ system. The way to calculate the frequency is following

$$Transfer\_speed\_rate = \frac{PCLKB}{16 \times (speed + 1)}[Mbps]$$

Fifth parameter:

Definition of the transfer　mode

0 : Do not use　(Reserved for future use)

1: 32bit 8 channel two times transfer mode

2: 32bit 4 channel one time transfer mode

This mode is supported on W1004, W2001, W2002, T2001C and T2006A.

3 : 32bit 12channel three times transfer mode

This mode is supported on W2002, T2001C and T2006A.

4 : Do not use　(Reserved for future use)

Transfer function	void	ics2_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.

Please keep and call the time defined by the following formula.

Case of W1001, W1003, ICS++ W1004, T2001A/B, T2002A/B:

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

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

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

Case of W2001, W2002, T2001C, T2006A

$$MinimumPeriod = 1/(Transfer\_speed\_rate[bps]) \times 180 + 30[us]$$

*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 ICS uses interrupt vector, please register the following functions into the interrupt vector of user software.

*Case1: When using a project that is automatically generated by the CC-RL compiler running on CS+
Please add these functions to the file which indicated the interrupt processing “intprg.c”.

The case of SCI1

```
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
```

*Case2: When using a project that is automatically generated by SmartConfigurator.
Please add four interrupt functions. Since SmartConfigurator does not generated SCI interrupt functions used by ICS. Pay attention to the vector number, since vector number is changed according to the SCI number.

```
#pragma interrupt (Excep_SCI2_ERI2(vect=222))
#pragma interrupt (Excep_SCI2_RXI2(vect=223))
#pragma interrupt (Excep_SCI2_TXI2(vect=224))
#pragma interrupt (Excep_SCI2_TEI2(vect=225))
void Excep_SCI2_ERI2(void){ /* no code */ }
void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); }
void Excep_SCI2_TXI2(void){ /* no code */ }
void Excep_SCI2_TEI2(void){ /* no code */ }
```

4.5.3. RX62T function usage

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

3) Place the DTC table.
Please use either method. In the example, we use A)

I) Place the DTC table at the absolute address using #pragma address direct directive.

```
#pragma address dtc_table=0x03000
uint32_t dtc_table[256];
```

J) In section specification of the development environment, specify the section address of dtc_table.

```
#pragma section DTCTBL
uint32_t    dtc_table[256];        // caution alignment 0x000
#pragma section
```

In CS+ specify the address of the BDTCTBL
Project Tree
- ➔ Build tool
- ➔ Property
- ➔ Link Option
- ➔ Section
- ➔ BDTCTBL

DTC table address must be placed at 12 bits of low ranks are set to 0.

2) Call "ics2_init()" as following

Please put the initialization function "ics2_init((void*)dtc_table, ICS_SCI0_PB2_PB1, 6, 2, 1)"
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.
Forth parameter is "2".
Fifth parameter is normally "1".

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

```
#pragma address dtc_table=0x03000
uint32_t dtc_table[256];

void main(void)
{
      ics2_init((void*)dtc_table, ICS_SCI0_PB2_PB1, 6, 0, 1);      /* Interrupt level 6              */
      while(1)
      {    nop();    }
}
```

3) Installation of ics2_watchpoint() function

In this sample software, ics2_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. (In the case of W1004). If the carrier interrupt period is below 250us, please decimate function call of ics2_watchpoint() as in the List 2.

---------- List 2 ics2_watchpoint() decimation ----------

```
int     deci = 0;

void    int_TM0(void)      /* 100us period */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics2_watchpoint();
        }
}
```

4) Modification of “intprg.c”

The case of SCI0

```
void Excep_SCI0_RXI0(void){ ics_int_sci_rxi(); }
void Excep_SCI0_ERI0(void){ ics_int_sci_eri(); }
```

*Caution

When using SmartConfigurator, the user also needs to add TXI, TEI, ERI functions according to the description of the interrupt function.

4.6. RX63T series (CC compiler)

4.6.1. RX63T resources

CPU name	RX63T series
Develop environment	CS+ Ver.6.00.00　CC-RX 2.07.00
Library version	Ver.3.60
Communication rate	0.5Mbps – 3.125Mbps Transfer speed rate to be set $Transfer\_speed\_rate = \frac{PCLKB}{16 \times (speed + 1)} [Mbps]$ [speed≧0] Standard Clock　1.00Mbps　speed = 2　@PCLKB = 48MHz
Support port	SCI0　TXD0:PB2,　RXD0:PB1 SCI0　TXD0:P30,　RXD0:P24 SCI0　TXD0:PA4,　RXD0:PA5 SCI0　TXD0:P23,　RXD0:P22 SCI1　TXD1:PD3,　RXD1:PD5 SCI1　TXD1:P94,　RXD1:P93 SCI1　TXD1:PF3,　RXD1:PF2 SCI1　TXD1:P95,　RXD1:P96 SCI2　TXD2:P02,　RXD2:P03 SCI2　TXD2:PG0,　RXD2:PG1 SCI2　TXD2:PA1,　RXD2:PA2 SCI3　TXD3:P35,　RXD3:P34 SCI3　TXD3:PG3,　RXD3:PG4 SCI12　TXD12:PB5,　RXD12:PB6　// 64, 48pin version is not supported SCI12　TXD12:P81,　RXD12:P80
Library file name	ics2_RX63T.lib
Header file name	ics2_RX63T.h

Used CPU resources	Support variable type
・Used internal resources INT　SCIx RXI INT　SCIx TXI DTC　(TXIx) ICU.DTCER[xx].BIT.DTCE SCIx　(all resisters) DTC　(all resisters) ICU.IPR[xx].BYTE　corresponding part ICU.IER[xx].BIT.IEN7　corresponding part SYSTEM.MSTPCRA.BIT.Bxx　corresponding part SYSTEM.MSTPCRB.BIT.B31 MPC　corresponding part PORTx.　Corresponding part	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 / signed char 16bit unsigned short/ signed short 32bit unsigned int / signed int 32bit IEEE754 floating point

4.6.2. RX63T function library

Initialize function　void ics2_init(void* addr, char　port,　char level, char speed, char mode);

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 ics2_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 ics2_<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.

Forth parameter:

Transfer speed rate to be used in the ICS++ system. The way to calculate the frequency is following

$$Transfer\_speed\_rate = \frac{PCLKB}{16 \times (speed + 1)}[Mbps]$$

Fifth parameter:

Definition of the transfer　mode

0 : Do not use　(Reserved for future use)

1: 32bit 8 channel two times transfer mode

2: 32bit 4 channel one time transfer mode

This mode is supported on W1004, W2001, W2002, T2001C and T2006A.

3 : 32bit 12channel three times transfer mode

This mode is supported on W2002, T2001C and T2006A.

4 : Do not use　(Reserved for future use)

Transfer function	void	ics2_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.

Please keep and call the time defined by the following formula.

Case of W1001, W1003, T2001A/B, T2002A/B, ICS++ W1004,

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

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

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

Case of W2001, W2002, T2001C, T2006A

$$MinimumPeriod = 1/(Transfer\_speed\_rate[bps]) \times 180 + 30[us]$$

*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 ICS uses interrupt vector, please register the following functions into the interrupt vector of user software.

*Case1: When using a project that is automatically generated by the CC-RL compiler running on CS+
Please add these functions to the file which indicated the interrupt processing “intprg.c”.

The case of SCI1

```
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
```

*Case2: When using a project that is automatically generated by SmartConfigurator.
Please add four interrupt functions. Since SmartConfigurator does not generated SCI interrupt functions used by ICS. Pay attention to the vector number, since vector number is changed according to the SCI number.

```
#pragma interrupt (Excep_SCI2_RXI2(vect=220))
#pragma interrupt (Excep_SCI2_TXI2(vect=221))
#pragma interrupt (Excep_SCI2_TEI2(vect=222))
void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); }
void Excep_SCI2_TXI2(void){ /* no code */ }
void Excep_SCI2_TEI2(void){ /* no code */ }
```

4.6.3. RX63T function usage

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

4) Place the DTC table.
Please use either method. In the example, we use A)

K) Place the DTC table at the absolute address using #pragma address direct directive.

```
#pragma address dtc_table=0x03000
uint32_t dtc_table[256];
```

L) In section specification of the development environment, specify the section address of dtc_table.

```
#pragma section DTCTBL
uint32_t    dtc_table[256];        // caution alignment 0x000
#pragma section
```

In CS+ specify the address of the BDTCTBL
Project Tree
- ➔ Build tool
- ➔ Property
- ➔ Link Option
- ➔ Section
- ➔ BDTCTBL

DTC table address must be placed at 12 bits of low ranks are set to 0.

2) Call "ics2_init()" as following

Please put the initialization function "ics2_init((void*)dtc_table, ICS_SCI0_PB2_PB1, 6, 2, 1)"
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.
Forth parameter is "2".
Fifth parameter is normally "1".

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

```
#pragma address dtc_table=0x03000
uint32_t dtc_table[256];

void main(void)
{
      ics2_init((void*)dtc_table, ICS_SCI0_PB2_PB1, 6, 0, 1);      /* Interrupt level 6              */
      while(1)
      {   nop();   }
}
```

3) Installation of ics2_watchpoint() function

In this sample software, ics2_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. (In the case of W1004). If the carrier interrupt period is below 250us, please decimate function call of ics2_watchpoint() as in the List 2.

---------- List 2 ics2_watchpoint() decimation ----------

```
int     deci = 0;

void    int_TM0(void)      /* 100us period */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics2_watchpoint();
        }
}
```

4) Modification of “intprg.c”

The case of SCI0

```
void Excep_SCI0_RXI0(void){ ics_int_sci_rxi(); }
```

*Caution

When using SmartConfigurator, the user also needs to add TXI, TEI functions according to the description of the interrupt function.

4.7. RX66T series (CC compiler)

4.7.1. RX66T resources

CPU name	RX66T series
Develop environment	CS+ Ver.8.00.00　CC-RX 3.00.00
Library version	Ver.3.60
Communication rate	0.5Mbps – 7.5Mbps Transfer speed rate to be set SCI1, SCI5, SCI6, SCI8, SCI9, SCI12 PCLKB= 40MHz @CPU CLK=160MHz　(PCLKB max 60MHz) $Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)}[Mbps]$ [speed≧0] Standard Clock　7.5Mbps　speed = 0　@PCLKB = 60MHz *This library does not support SCI11
Support port	#define　ICS_SCI1_PD3_PD5　(0x10) #define　ICS_SCI5_PD7_PE0　(0x50) #define　ICS_SCI5_PB5_PB6　(0x51) #define　ICS_SCI6_PB0_PB1　(0x60) #define　ICS_SCI8_PD0_PD1　(0x80) #define　ICS_SCI8_PC1_PC0　(0x81) #define　ICS_SCI8_PA4_PA5　(0x82) #define　ICS_SCI8_P23_P22　(0x83) #define　ICS_SCI9_P01_P00　(0x90) #define　ICS_SCI9_PA3_PA2　(0x91) #define　ICS_SCI9_PG1_PG0　(0x92) #define　ICS_SCI12_PB5_PB6　(0xC0) #define　ICS_SCI12_P23_P22　(0xC1) #define　ICS_SCI12_P01_P00　(0xC2)
Library file name	ics2_RX66T.lib
Header file name	ics2_RX66T.h

Used CPU resources	Support variable type
・Used internal resources INT　SCIx RXI INT　SCIx TXI DTC　(TXIx) ICU.DTCER[xx].BIT.DTCE SCIx　(all resisters) DTC　(all resisters) ICU.IPR[xx].BYTE　corresponding part ICU.IER[xx].BIT.IEN7　corresponding part SYSTEM.MSTPCRA.BIT.Bxx　corresponding part SYSTEM.MSTPCRB.BIT.B31 MPC　corresponding part PORTx.　Corresponding part	Numeric display 8bit unsigned char / signed char 16bit unsigned char / signed char 32bit unsigned char / signed char 32bit IEEE754 floating point 8bit　BOOL type 8bit　LOGIC type Waveform display 8bit unsigned char / signed char 16bit unsigned short/ signed short 32bit unsigned int / signed int 32bit IEEE754 floating point

4.7.2. RX66T function library

Initialize function　void ics2_init(void* addr, char　port,　char level, char speed, char mode);

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 ics2_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 ics2_<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.

Forth parameter:

Transfer speed rate to be used in the ICS++ system. The way to calculate the frequency is following

$$Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)}[Mbps]$$

Fifth parameter:

Definition of the transfer　mode

0 : Do not use　(Reserved for future use)

1: 32bit 8 channel two times transfer mode

2: 32bit 4 channel one time transfer mode

This mode is supported on W1004, W2001, W2002, T2001C and T2006A.

3 : 32bit 12channel three times transfer mode

This mode is supported on W2002, T2001C and T2006A.

4 : Do not use

5: Do not use

6: Do not use

Transfer function　　void　　ics2_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. Please keep and call the time defined by the following formula. Case of W2001, W2002, T2001C, T2006A $MinimumPeriod = 1/(Transfer\_speed\_rate[bps]) \times 180 + 30[us]$ Previous products Case of W1001, W1003, T2001A/B, T2002A/B, ICS++ W1004, $MinimumPeriod = 1/(Transdfer\_speed\_rate[bps]) \times 180 + 70[us]$ 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 ICS uses interrupt vector, please register the following functions into the interrupt vector of user software. *Case1: When using a project that is automatically generated by the CC-RL compiler running on CS+ Please add these functions to the file which indicated the interrupt processing “intprg.c”. The case of SCI1 void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); } *Case2: When using a project that is automatically generated by SmartConfigurator. Please add four interrupt functions. Since SmartConfigurator does not generated SCI interrupt functions used by ICS. Pay attention to the vector number, since vector number is changed according to the SCI number. #pragma interrupt (Excep_SCI1_RXI1(vect=60)) #pragma interrupt (Excep_SCI1_TXI1(vect=61)) void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); } void Excep_SCI1_TXI1(void){ /* no code */ }

4.7.3. RX66T function usage

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

5) Place the DTC table.
Please use either method. In the example, we use A)

M) Place the DTC table at the absolute address using #pragma address direct directive.

```
#pragma address dtc_table=0x0F000
uint32_t dtc_table[256];
```

N) In section specification of the development environment, specify the section address of dtc_table.

```
#pragma section DTCTBL
uint32_t    dtc_table[256];        // caution alignment 0x000
#pragma section
```

In CS+ specify the address of the BDTCTBL
Project Tree

- ➔ Build tool
- ➔ Property
- ➔ Link Option
- ➔ Section
- ➔ BDTCTBL

DTC table address must be placed at 12 bits of low ranks are set to 0.

2) Call "ics2_init()" as following

Please put the initialization function "ics2_init((void*)dtc_table, ICS_SCI6_PB0_PB1, 6, 0, 1)" 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.
Forth parameter is "0".
Fifth parameter is normally "1".

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

```
#pragma address dtc_table=0x0F000
uint32_t dtc_table[256];

void main(void)
{
    ics2_init((void*)dtc_table, ICS_SCI6_PB0_PB1, 6, 0, 1);    /* Interrupt level 6          */
    while(1)
    {   nop();   }
}
```

3) Installation of ics2_watchpoint() function

In this sample software, ics2_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. (In the case of W1004). If the carrier interrupt period is below 250us, please decimate function call of ics2_watchpoint() as in the List 2.

---------- List 2 ics2_watchpoint() decimation ----------

```
int     deci = 0;

void    int_TM0(void)      /* 100us period */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics2_watchpoint();
        }
}
```

4) Modification of "intprg.c"

The case of SCI1

```
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

The case of SCI3

```
void Excep_SCI5_RXI5(void){ ics_int_sci_rxi(); }
```

*Caution

When using SmartConfigurator, the user also needs to add TXI functions according to the description of the interrupt function.

4.8. RX71M series (CC compiler)

4.8.1. RX71M resources

CPU name	RX71M series
Develop environment	CS+ Ver.6.00.00　CC-RX 2.07.00
Library version	Ver.3.60
Communication rate	0.5Mbps – 7.5Mbps Transfer speed rate to be set $Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)}[Mbps]$ [speed≧0] Standard Clock　7.5Mbps　speed = 0　@PCLKB = 60MHz
Support port	SCI0　TXD0:P32,　RXD0:P33 SCI0　TXD0:P20,　RXD0:P21 SCI1　TXD1:P26,　RXD1:P30 SCI1　TXD1:P16,　RXD1:P15 SCI2　TXD2:P13,　RXD2:P12 SCI2　TXD2:P50,　RXD2:P52 SCI3　TXD3:P23,　RXD3:P25 SCI3　TXD3:P16,　RXD3:P17 SCI4　TXD4:PB1,　RXD4:PB0 SCI5　TXD5:PC3,　RXD5:PC2 SCI5　TXD5:PA4,　RXD5:PA3 SCI6　TXD6:P00,　RXD6:P01 SCI6　TXD6:P32,　RXD6:P33 SCI6　TXD6:PB1,　RXD6:PB0 SCI7　TXD7:P90,　RXD7:P92
Library file name	ics2_RX71M.lib
Header file name	ics2_RX71M.h

Used CPU resources	Support variable type
・Used internal resources INT　SCIx RXI INT　SCIx TXI DTC　(TXIx) ICU.DTCER[xx].BIT.DTCE SCIx　(all resisters) DTC　(all resisters) ICU.IPR[xx].BYTE　corresponding part ICU.IER[xx].BIT.IEN7　corresponding part SYSTEM.MSTPCRA.BIT.Bxx　corresponding part SYSTEM.MSTPCRB.BIT.B31 MPC　corresponding part PORTx.　Corresponding part	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 / signed char 16bit unsigned short/ signed short 32bit unsigned int / signed int 32bit IEEE754 floating point

4.8.2. RX71M function library

Initialize function　void ics2_init(void* addr, char　port,　char level, char speed, char mode);

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 ics2_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 ics2_<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.

Forth parameter:

Transfer speed rate to be used in the ICS++ system. The way to calculate the frequency is following

$$Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)}[Mbps]$$

Fifth parameter:

Definition of the transfer　mode

0 : Do not use　(Reserved for future use)

1: 32bit 8 channel two times transfer mode

2: 32bit 4 channel one time transfer mode

This mode is supported on W1004, W2001, W2002, T2001C and T2006A.

3 : 32bit 12channel three times transfer mode

This mode is supported on W2002, T2001C and T2006A.

4 : Do not use　(Reserved for future use)

Transfer function　　void　　ics2_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. Please keep and call the time defined by the following formula. Case of W1001, W1003, T2001A/B, T2002A/B, ICS++ W1004, $MinimumPeriod = 1/(Transdfer\_speed\_rate[bps]) \times 180 + 70[us]$ When the communication speed is 1Mbps, let 1Mbps into this formula. $MinimumPeriod = 1/(1[Mbps]) \times 180 + 70[us] = 250[us]$ Case of W2001, W2002, T2001C, T2006A $MinimumPeriod = 1/(Transfer\_speed\_rate[bps]) \times 180 + 30[us]$ *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 ICS uses interrupt vector, please register the following functions into the interrupt vector of user software. *Case1: When using a project that is automatically generated by the CC-RL compiler running on CS+ Please add these functions to the file which indicated the interrupt processing “intprg.c”. The case of SCI1 void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); } *Case2: When using a project that is automatically generated by SmartConfigurator. Please add four interrupt functions. Since SmartConfigurator does not generated SCI interrupt functions used by ICS. Pay attention to the vector number, since vector number is changed according to the SCI number. #pragma interrupt (Excep_SCI2_RXI2(vect=62)) #pragma interrupt (Excep_SCI2_TXI2(vect=63)) void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); } void Excep_SCI2_TXI2(void){ /* no code */ }

4.8.3. RX71M function usage

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

6) Place the DTC table.
Please use either method. In the example, we use A)

O) Place the DTC table at the absolute address using #pragma address direct directive.

```
#pragma address dtc_table=0x0F000
uint32_t dtc_table[256];
```

P) In section specification of the development environment, specify the section address of dtc_table.

```
#pragma section DTCTBL
uint32_t    dtc_table[256];        // caution alignment 0x000
#pragma section
```

In CS+ specify the address of the BDTCTBL
Project Tree
- ➔ Build tool
- ➔ Property
- ➔ Link Option
- ➔ Section
- ➔ BDTCTBL

DTC table address must be placed at 12 bits of low ranks are set to 0.

2) Call "ics2_init()" as following

Please put the initialization function "ics2_init((void*)dtc_table, ICS_SCI0_P32_P33, 6, 0, 1)"
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.
Forth parameter is "0".
Fifth parameter is normally "1".

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

```
#pragma address dtc_table=0x0F000
uint32_t dtc_table[256];

void main(void)
{
    ics2_init((void*)dtc_table, ICS_SCI0_P32_P33, 6, 0, 1);    /* Interrupt level 6          */
    while(1)
```

```
    {   nop();   }
}
```

3) Installation of ics2_watchpoint() function

In this sample software, ics2_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. (In the case of W1004). If the carrier interrupt period is below 250us, please decimate function call of ics2_watchpoint() as in the List 2.

---------- List 2 ics2_watchpoint() decimation ----------

```
int     deci = 0;

void    int_TM0(void)      /* 100us period */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics2_watchpoint();
        }
}
```

4) Modification of "intprg.c"

The case of SCI0

```
void Excep_SCI0_RXI0(void){ ics_int_sci_rxi(); }
```

The case of SCI1

```
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

The case of SCI2

```
void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); }
```

The case of SCI3

```
void Excep_SCI3_RXI3(void){ ics_int_sci_rxi(); }
```

*Caution

When using SmartConfigurator, the user also needs to add TXI functions according to the description of the interrupt function.

4.9. RX72T series (CC compiler)

4.9.1. RX72T resources

CPU name	RX72T series
Develop environment	CS+ Ver.8.01.00　CC-RX 3.01.00
Library version	Ver.3.60 (RX V3 core)
Communication rate	0.5Mbps – 7.5Mbps Transfer speed rate to be set SCI1, SCI5, SCI6, SCI8, SCI9, SCI12 PCLKB= 40MHz @CPU CLK=160MHz　(PCLKB max 60MHz) $Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)}[Mbps]$ [speed≧0] Exception settings: 1 [Mbps] @speed=250　and PCLK=50MHz , 1 [Mbps] @speed=251　and PCLK=60MHz , Standard Clock　7.5Mbps　speed = 0　@PCLKB = 60MHz *This library does not support SCI11
Support port	#define　ICS_SCI1_PD3_PD5　(0x10) #define　ICS_SCI5_PD7_PE0　(0x50) #define　ICS_SCI5_PB5_PB6　(0x51) #define　ICS_SCI6_PB0_PB1　(0x60) #define　ICS_SCI8_PD0_PD1　(0x80) #define　ICS_SCI8_PC1_PC0　(0x81) #define　ICS_SCI8_PA4_PA5　(0x82) #define　ICS_SCI8_P23_P22　(0x83) #define　ICS_SCI9_P01_P00　(0x90) #define　ICS_SCI9_PA3_PA2　(0x91) #define　ICS_SCI9_PG1_PG0　(0x92) #define　ICS_SCI11_PD3_PD5　(0xB1) #define　ICS_SCI11_PB5_PB6　(0xB2) #define　ICS_SCI11_PF0_PF1　(0xB4) #define　ICS_SCI12_PB5_PB6　(0xC0) #define　ICS_SCI12_P23_P22　(0xC1) #define　ICS_SCI12_P01_P00　(0xC2)
Library file name	ics2_RX72T.lib
Header file name	ics2_RX72T.h

Used CPU resources	Support variable type
・Used internal resources INT　SCIx RXI INT　SCIx TXI DTC　(TXIx) ICU.DTCER[xx].BIT.DTCE SCIx　(all resisters) DTC　(all resisters)	Numeric display 8bit unsigned char / signed char 16bit unsigned char / signed char 32bit unsigned char / signed char 32bit IEEE754 floating point 8bit　BOOL type 8bit　LOGIC type

ICU.IPR[xx].BYTE corresponding part ICU.IER[xx].BIT.IEN7 corresponding part SYSTEM.MSTPCRA.BIT.Bxx corresponding part SYSTEM.MSTPCRB.BIT.B31 MPC corresponding part PORTx. Corresponding part	Waveform display 8bit unsigned char / signed char 16bit unsigned short/ signed short 32bit unsigned int / signed int 32bit IEEE754 floating point

4.9.2. RX72T function library

Initialize function void ics2_init(void* addr, char port, char level, char speed, char mode);

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 ics2_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 ics2_<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.

Forth parameter:

Transfer speed rate to be used in the ICS++ system. The way to calculate the frequency is following

$$Transfer\_speed\_rate = \frac{PCLKB}{8 \times (speed + 1)}[Mbps]$$

Fifth parameter:

Definition of the transfer mode

0 : Do not use (Reserved for future use)

1: 32bit 8 channel two times transfer mode

2: 32bit 4 channel one time transfer mode

This mode is supported on W1004, W2001, W2002, T2001C and T2006A.

3 : 32bit 12channel three times transfer mode

This mode is supported on W2002, T2001C and T2006A.

4 : Do not use

5: Do not use

6: Do not use

Transfer function　　void　　ics2_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. Please keep and call the time defined by the following formula. Case of W2001, W2002, T2001C, T2006A $MinimumPeriod = 1/(Transfer\_speed\_rate[bps]) \times 180 + 30[us]$ Previous products Case of W1001, W1003, T2001A/B, T2002A/B, ICS++ W1004, $MinimumPeriod = 1/(Transdfer\_speed\_rate[bps]) \times 180 + 70[us]$ 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 ICS uses interrupt vector, please register the following functions into the interrupt vector of user software. *Case1: When using a project that is automatically generated by the CC-RL compiler running on CS+ Please add these functions to the file which indicated the interrupt processing “intprg.c”. The case of SCI1 void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); } *Case2: When using a project that is automatically generated by SmartConfigurator. Please add four interrupt functions. Since SmartConfigurator does not generated SCI interrupt functions used by ICS. Pay attention to the vector number, since vector number is changed according to the SCI number. #pragma interrupt (Excep_SCI1_RXI1(vect=60)) #pragma interrupt (Excep_SCI1_TXI1(vect=61)) void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); } void Excep_SCI1_TXI1(void){ /* no code */ }

4.9.3. RX72T function usage

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

7) Place the DTC table.
Please use either method. In the example, we use A)

Q) Place the DTC table at the absolute address using #pragma address direct directive.

```
#pragma address dtc_table=0x0F000
uint32_t dtc_table[256];
```

R) In section specification of the development environment, specify the section address of dtc_table.

```
#pragma section DTCTBL
uint32_t    dtc_table[256];        // caution alignment 0x000
#pragma section
```

In CS+ specify the address of the BDTCTBL
Project Tree
- ➔ Build tool
- ➔ Property
- ➔ Link Option
- ➔ Section
- ➔ BDTCTBL

DTC table address must be placed at 12 bits of low ranks are set to 0.

2) Call "ics2_init()" as following

Please put the initialization function "ics2_init((void*)dtc_table, ICS_SCI6_PB0_PB1, 6, 0, 1)"
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.
Forth parameter is "0".
Fifth parameter is normally "1".

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

```
#pragma address dtc_table=0x0F000
uint32_t dtc_table[256];

void main(void)
{
    ics2_init((void*)dtc_table, ICS_SCI6_PB0_PB1, 6, 0, 1);    /* Interrupt level 6          */
    while(1)
    {   nop();   }
}
```

3) Installation of ics2_watchpoint() function

In this sample software, ics2_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. (In the case of W1004). If the carrier interrupt period is below 250us, please decimate function call of ics2_watchpoint() as in the List 2.

---------- List 2 ics2_watchpoint() decimation ----------

```
int     deci = 0;

void    int_TM0(void)      /* 100us period */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics2_watchpoint();
        }
}
```

4) Modification of "intprg.c"

The case of SCI1

```
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

The case of SCI3

```
void Excep_SCI5_RXI5(void){ ics_int_sci_rxi(); }
```

*Caution

When using SmartConfigurator, the user also needs to add TXI functions according to the description of the interrupt function.

5. Revision history

Version	Date	Note
Ver.1.01	2017-11-16	・First English version release
Ver.1.03	2019-07-09	・Add RX66T, RX72T
Ver.1.04	2020-04-21	・Add RX13T
Ver1.05	2020-06-25	・Add description about interrupt function
Ver.1.08	2022-03-20	・Add RX72T ICS_SCI11_PF0_PF1 port support
Ver.1.09	2023-09-04	・Add RX26T

ICS++ Library Function manual

Issue date: Sep-4-2023 Ver.1.09EN.

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