NEC

# Application Note

# V850E/RS1™

## 32-Bit Single-Chip Microcontrollers

## CSI3 Serial Interface and DMA Application

---

## µPD70F3403, µPD70F3402

Document No. U17573EE1V0AN00
Date Published July 2005


Printed in Germany

## NOTES FOR CMOS DEVICES

### ① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

### ② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to $V_{DD}$ or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

### ③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

Power-on does not necessarily define initial status of MOS device. Production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings or contents of registers. Device is not initialized until the reset signal is received. Reset operation must be executed immediately after power-on for devices having reset function.

- **The information in this document is current as of July, 2005. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics 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 NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".
  The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.
  "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.
  "Special": 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).
  "Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)
(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02. 11-1

# Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.


**NEC Electronics America Inc.**
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

**NEC Electronics (Europe) GmbH**
Duesseldorf, Germany
Tel: 0211-65 03 1101
Fax: 0211-65 03 1327

**Sucursal en España**
Madrid, Spain
Tel: 091- 504 27 87
Fax: 091- 504 28 60

**Succursale Française**
Vélizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

**Filiale Italiana**
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

**Branch The Netherlands**
Eindhoven, The Netherlands
Tel: 040-244 58 45
Fax: 040-244 45 80

**Branch Sweden**
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

**United Kingdom Branch**
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

**NEC Electronics Hong Kong Ltd.**
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

**NEC Electronics Hong Kong Ltd.**
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

**NEC Electronics Singapore Pte. Ltd.**
Singapore
Tel: 65-6253-8311
Fax: 65-6250-3583

**NEC Electronics Taiwan Ltd.**
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

# Introduction

**Readers**

This application note is intended for users who want to understand the functions of the V850E/RS1.

**Purpose**

This application note presents the hardware application note of the V850E/RS1.

**Organization**

This system specification describes the following sections:

- Library handling
- Reprogramming the device
- Watchdog-handling
- Hardware requirements
- Used device resources

**Legend**

Symbols and notation are used as follows:

| | | |
|---|---|---|
| Weight in data notation | : | Left is high-order column, right is low order column |
| Active low notation | : | $\overline{xxx}$ (pin or signal name is over-scored) or /xxx (slash before signal name) |
| Memory map address: | : | High order at high stage and low order at low stage |
| **Note** | : | Explanation of (Note) in the text |
| **Caution** | : | Information requiring particular attention |
| **Remark** | : | Supplementary explanation to the text |
| Numeric notation | : | Binary... xxxx or xxxB<br>Decimal... xxxx<br>Hexadecimal... xxxxH or 0x xxxx |

Prefixes representing powers of 2 (address space, memory capacity)

K (kilo): $2^{10} = 1024$

M (mega): $2^{20} = 1024^2 = 1,048,576$

G (giga): $2^{30} = 1024^3 = 1,073,741,824$

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1 Overview

The V850E/RS1 single chip microcontroller is a member of NEC's V850 32-bit RISC family, which match the performance gains attainable with RISC-based controllers to the needs of embedded control applications.
The V850E CPU offers easy pipeline handling and programming, resulting in compact code size comparable to 16-bit CISC CPUs.
The V850E/RS1 offers an excellent combination of general purpose peripheral functions, like serial communication interfaces (UART, clocked SI, clocked SI with translation buffer), timers and measurement inputs (A/D converter), with dedicated CAN network support. The device offers power-saving modes to manage the power consumption effectively under varying conditions. Thus equipped, the V850E/RS1 is ideally suited for automotive applications.

This application note is written to describe a CSI3 transmit and receive operation procedure using the V850E/RS1 microcontroller. CSI3 interface is a 3-wire serial synchronous interface with a dedicated transmit and receive FIFO buffer of 16 elements. It also provides the chip select for each data to be transmitted and received.
The CSI3 communication is done using two different methods with and without DMA implementation:

- FIFO buffer transfer mode
- Single buffer transfer mode

A timer provides a cycle period of time to send the data frame of 16 elements every 300 µs.
Transfer data length is 8 bits.
Queued CSI serial clock is 4 MHz using below operation mode:

- Master Mode
- Transmit/Receive Mode

The timing measurement highlights:

- Setup Time: time in between chip select active to first rising edge of the serial clock ($\overline{SCK3}$)
- Hold Time: time in between the last falling edge of the serial clock ($\overline{SCK3}$) and the inactive level of the chip select
- Time to transmit the complete frame of 16 data of 8 bits.

**Remark:** Other configuration of the CSI3 and DMA interface could be adapted to dedicated customers needs.

# Chapter 2 Queued CSI3 Description (CSI30, CSI31)

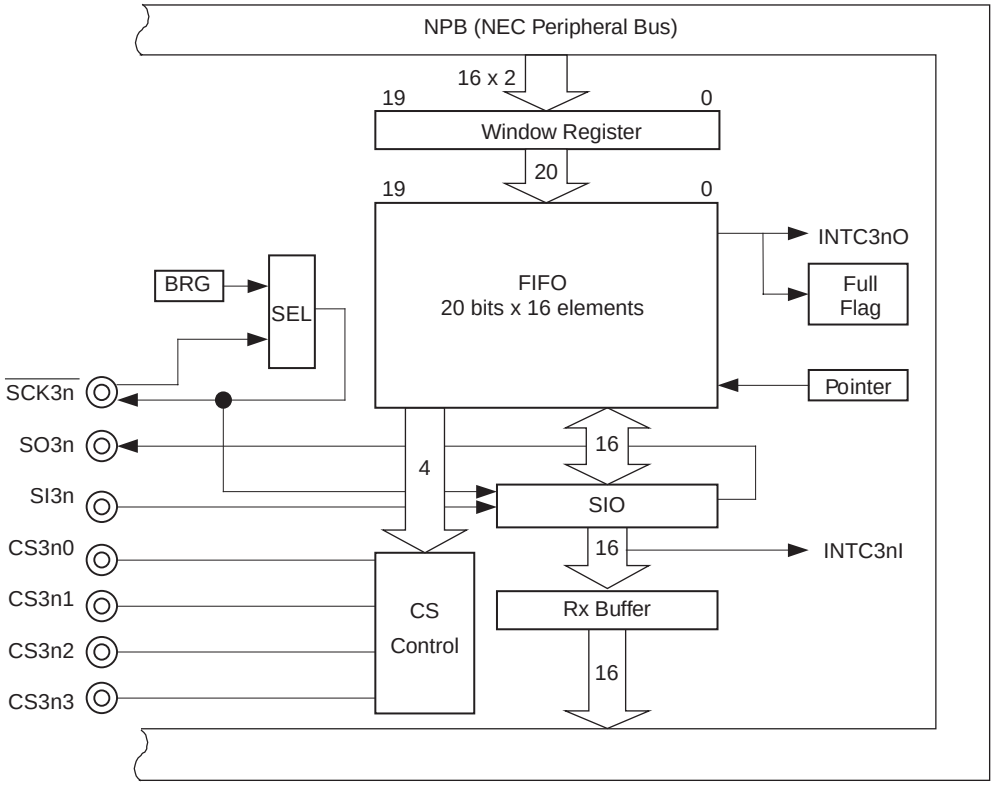
## 2.1 Features

- 3-wire serial synchronous transfers
- The following 7 pins are provided to enable a 3-wire serial interface:
  (a) SO3 (serial data output)
  (b) SI3 (serial data input)
  (c) $\overline{\text{SCK3}}$ (serial clock I/O)
  (d) CS30 - CS33 (Chip Select)
- Master mode and slave mode selectable
- Serial clock and data phase selectable
- Transfer date length selectable from 8 to 16 bits in 1-bit units
- Data transfer with MSB- for LSB-first selectable
- Three selectable transfer modes:
  (a) transmit only mode
  (b) receive only mode
  (c) transmit/receive mode
- Transmit and receive FIFO (16 elements)
- Selection between single buffer transfer mode and FIFO buffer transfer mode
- Internal baud rate generator
- Programmable baudrate through BRG output (master) or slave clock
- DMA transfer of received data to memory available

Maximum $\overline{\text{SCK3}}$ frequency: 10 MHz

### 2.1.1 CSI3 block diagram

***Figure 2-1: Queued CSI Block Diagram***




### 2.1.2 Input/output pins

The table below shows the input/output pins of the CSI3.

***Table 2-1: Input/Output Pins of the CSI3***

| Signal name | I/O | Active level | Disabled level | Function |
|---|---|---|---|---|
| $\overline{SCK3n}$ | I/O | – | H | Serial clock signal |
| SI3n | I | – | – | Input serial data signal |
| SO3n | O | – | – | Output serial data signal |
| CS3n0 | O | L**Note 2** | H**Note 2** | Serial peripheral chip select signal |
| CS3n1 | O | L**Note 2** | H**Note 2** | Serial peripheral chip select signal |
| CS3n2 | O | L**Note 2** | H**Note 2** | Serial peripheral chip select signal |
| CS3n2 | O | L**Note 2** | H**Note 2** | Serial peripheral chip select signal |

**Notes: 1.** n = 0, 1

**2.** The active level is programmable for each chip select.

## 2.2 Queued CSI Control Registers

### (1) Register map

The tables below show the Special Function Registers for the Queued CSI modules CSI30 and CSI31.

***Table 2-1: CSI30***

| Register name | Function | Address | Reset Value | Access | | | R/W |
|---|---|---|---|---|---|---|---|
| | | | | 1-bit | 8-bit | 16-bit | |
| CSIM0 | Queued CSI operation mode register | FFFFFD40H | 00H | O | O | - | R/W |
| CSIC0 | Queued CSI clock selection register | FFFFFD41H | 07H | O | O | - | R/W |
| SIRB0 | Receive data buffer | FFFFFD42H | 0000H | - | - | O | R |
| SIRB0L | Receive data buffer L | FFFFFD42H | 00H | - | O | - | R |
| SFCS0 | Chip Select FIFO buffer | FFFFFD44H | FFFFH | - | - | O | R/W |
| SFCS0L | Chip Select FIFO buffer L | FFFFFD44H | FFH | - | O | - | R/W |
| SFDB0 | Transmit data FIFO buffer | FFFFFD46H | 0000H | - | - | O | R/W |
| SFDB0L | Transmit data FIFO buffer L | FFFFFD46H | 00H | - | O | - | R/W |
| SFA0 | FIFO status register | FFFFFD48H | 20H | O | O | - | R/W |
| CSIL0 | Data length select register | FFFFFD49H | 00H | O | O | - | R/W |
| SFN0 | Transfer number select register | FFFFFD4CH | 00H | O | O | - | R/W |

***Table 2-2: CSI31***

| Register name | Function | Address | Reset Value | Access | | | R/W |
|---|---|---|---|---|---|---|---|
| | | | | 1-bit | 8-bit | 16-bit | |
| CSIM1 | Queued CSI operation mode register | FFFFFD60H | 00H | O | O | - | R/W |
| CSIC1 | Queued CSI clock selection register | FFFFFD61H | 07H | O | O | - | R/W |
| SIRB1 | Receive data buffer | FFFFFD62H | 0000H | - | - | O | R |
| SIRB1L | Receive data buffer L | FFFFFD62H | 00H | - | O | - | R |
| SFCS1 | Chip Select FIFO buffer | FFFFFD64H | FFFFH | - | - | O | R/W |
| SFCS1L | Chip Select FIFO buffer L | FFFFFD64H | FFH | - | O | - | R/W |
| SFDB1 | Transmit data FIFO buffer | FFFFFD66H | 0000H | - | - | O | R/W |
| SFDB1L | Transmit data FIFO buffer L | FFFFFD66H | 00H | - | O | - | R/W |
| SFA1 | Transmit FIFO status register | FFFFFD68H | 20H | O | O | - | R/W |
| CSIL1 | Data length select register | FFFFFD69H | 00H | O | O | - | R/W |
| SFN1 | Transmit data number select register | FFFFFD6CH | 00H | O | O | - | R/W |

### (2) Queued CSI operation mode registers (CSIM0, CSIM1)

The CSIM registers control the Queued CSI macro's operations.
These registers can be read or written in 1-bit and 8-bit units.
TRMD, DIR, CSIT, CSWE, CSMD bits can only be written when CTXE = 0 and CRXE = 0.
The registers are initialized to 00H at reset.

***Figure 2-2: Queued CSI Operation Mode Registers (CSIM0, CSIM1) Format (1/2)***

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Address | R/W | After reset |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CSIMn (n = 0, 1) | POWER | CTXE | CRXE | TRMD | DIR | CSIT | CSWE | CSMD | FFFFFD40H, FFFFFD60H | R/W | 00H |

| POWER | Queued CSI operation clock control |
|---|---|
| 0 | Stop macro operation clock (Reset internal control circuits) |
| 1 | Provide macro operation clock |
| Clearing POWER = "0" resets the internal circuits asynchronously, stops operation and sets the Queued CSI to standby state. Input clock is not provided to internal circuits.<br>Set POWER = "1" to activate the Queued CSI. | |

**Caution: When changing the POWER bit, do not change any other bit at the same time.**
**While POWER=”0”, the only registers that can be accessed are CSIM, SFDB, SFDBL, and SFA.**
**Set the POWER bit before writing any of the other bits of CSIMn.**

| CTXE | Transmission enable/disable |
|---|---|
| 0 | Transmission disabled |
| 1 | Transmission enabled |

| CRXE | Receive enable/disable |
|---|---|
| 0 | Receive disabled |
| 1 | Receive enabled |

| TRMD | Transfer mode select |
|---|---|
| 0 | Single buffer transfer mode |
| 1 | FIFO buffer transfer mode |

**Caution: Write is permitted only when CTXE = 0 and CRXE = 0.**

***Figure 2-3: Queued CSI Operation Mode Registers (CSIM0, CSIM1) Format (2/2)***

| DIR | Serial data direction selection |
|---|---|
| 0 | Data is sent/received with MSB first |
| 1 | Data is sent/received with LSB first |

**Caution: Write is permitted only when CTXE = 0 and CRXE = 0.**

See section **2.3.2 "Serial data direction select function" on page 29** for further details on the DIR bit setting.

| CSIT | Interrupt delay mode select (INTC3nI signal) |
|---|---|
| 0 | No delay |
| 1 | Half clock delay |

**Caution: Write is permitted only when CTXE = 0 and CRXE = 0.**
**This bit is only valid in master mode. In slave mode, no delay is generated.**

| CSWE | Transmission wait enable/disable select |
|---|---|
| 0 | Transmission wait disable. Not insert 1 clock ($\overline{SCK3}$) wait at transmission start. |
| 1 | Transmission wait enable. Insert 1 clock ($\overline{SCK3}$) wait at transmission start. |

**Caution: Write is permitted only when CTXE = 0 and CRXE = 0.**
**This bit is only valid in master mode. In slave mode, no wait is generated.**

| CSMD | Chip Select mode select |
|---|---|
| 0 | Chip select inactive level output disable. Do not force chip select inactive state after each transfer of a data element. |
| 1 | Chip select inactive level output enable. Hold all chip selects inactive for halt-length $\overline{SCK3}$ after each transfer of a data element. |

**Caution: Write is permitted only when CTXE = 0 and CRXE= 0.**
**This bit is only valid for CSWE=1.**
**This bit is only valid in master mode. In slave mode, CS signals are always held at inactive level.**

In combination:

| CSWE | CSMD | Transmission wait | Chip Select inactive level |
|---|---|---|---|
| 0 | 0 | None | Not output |
| 0 | 1 | None | Not output |
| 1 | 0 | One $\overline{SCK3}$ length clock wait | Not output |
| 1 | 1 | One $\overline{SCK3}$ length clock wait | Output inactive level of half-length $\overline{SCK3}$ during first half of transmission wait |

See section **2.3.10 "Additional timing and delay selections" on page 37** for further details on the timing selections by CSIT, CSWE, CSMD bits.

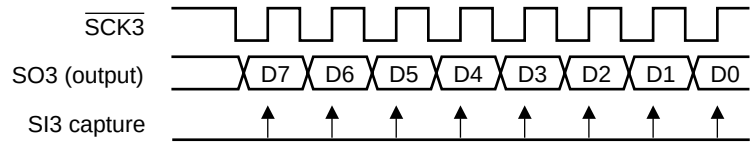
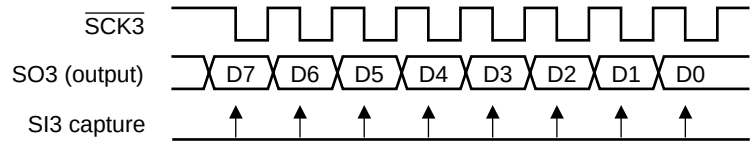
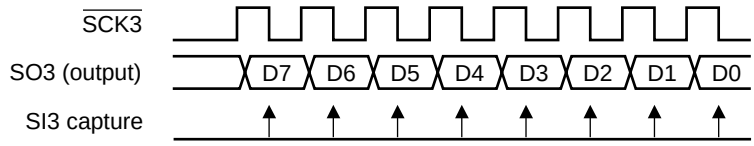
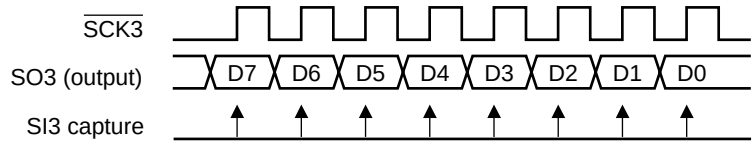
### (3) Queued CSI clock selection registers (CSIC0, CSIC1)

The CSIC register is an 8-bit register that is used to control the serial transfer operations. This register can be read or written in 1-bit and in 8-bit units.

**Caution: This register can be written only while CSIM register's CTXE = 0 and CRXE = 0.**

***Figure 2-4: Queued CSI Clock Selection Registers (CSIC0, CSIC1) Format (1/2)***

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Address | R/W | After reset |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CSICn (n = 0, 1) | MDL2 | MDL1 | MDL0 | CKP | DAP | CKS2 | CKS1 | CKS0 | FFFFFD41H, FFFFFD61H | R/W | 07H |

| CKP | DAP | Operation mode |
|---|---|---|
| 0 | 0 |   |
| 0 | 1 |   |
| 1 | 0 |   |
| 1 | 1 |   |

**Caution: Modification of these bits is permitted only when CTXE = 0 and CRXE = 0.**

**Remark:** CKP: Clock phase selection bit
DAP: Data phase selection bit

***Figure 2-5: Queued CSI Clock Selection Registers (CSIC0, CSIC1) Format (2/2)***

| CKS2 | CKS1 | CKS0 | Prescaler output (PRSOUT) | Mode | K |
|---|---|---|---|---|---|
| 0 | 0 | 0 | $f_{QCSI}$ | Master Mode | 0 |
| 0 | 0 | 1 | $f_{QCSI}/2$ | Master Mode | 1 |
| 0 | 1 | 0 | $f_{QCSI}/4$ | Master Mode | 2 |
| 0 | 1 | 1 | $f_{QCSI}/8$ | Master Mode | 3 |
| 1 | 0 | 0 | $f_{QCSI}/16$ | Master Mode | 4 |
| 1 | 0 | 1 | $f_{QCSI}/32$ | Master Mode | 5 |
| 1 | 1 | 0 | $f_{QCSI}/64$ | Master Mode | 6 |
| 1 | 1 | 1 | $\overline{SCK3}$ (input) **Note** | Slave Mode | – |
| These bits are used to select the input clock | | | | | |

**Note:** $\overline{SCK3}$ pin is configured as input mode for serial transfer clock.
$f_{QCSI}$ is the clock supply of the Queued-CSI macro.

**Caution: Rewriting these bits is only permitted when CSIE = 0.**

See sections **2.3.4 “Slave mode” on page 30** and **2.3.5 “Master mode” on page 31** for further explanation on slave mode and master mode.

| MDL2 | MDL1 | MDL0 | Operation clock | N | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | BRG disable | – | BRG stop for power saving |
| 0 | 0 | 1 | PRSOUT/2 | 1 | |
| 0 | 1 | 0 | PRSOUT/4 | 2 | |
| 0 | 1 | 1 | PRSOUT/6 | 3 | |
| 1 | 0 | 0 | PRSOUT/8 | 4 | |
| 1 | 0 | 1 | PRSOUT/10 | 5 | |
| 1 | 1 | 0 | PRSOUT/12 | 6 | |
| 1 | 1 | 1 | PRSOUT/14 | 7 | |
| These bits are used to modulate the selected clock | | | | | |

**Caution: Rewriting these bits is only permitted when CTXE = 0 and CRXE = 0.**
**MDL [2:0] = [0,0,1] is prohibited when CKS [2:0] = [0,0,0].**

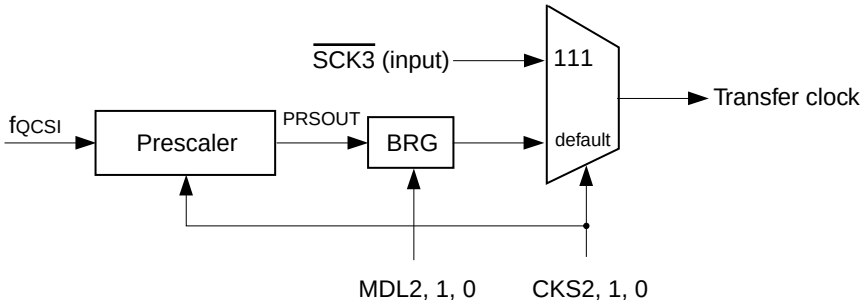
See section **2.3.6 “Transmission clock select function” on page 31** for further explanation on the transfer clock selection.

The baudrate for the transmission is calculated with the following formula:

Transmission Baud Rate = f/(N*2(K+1))

where: f: $f_{QCSI}$ frequency,
N = 1 - 7,
K = 0 - 6.

***Figure 2-6: Queued CSI Baud Rate Block Diagram***



**(4) Queued CSI macro clock selection ($f_{QCSI}$): Extension clock select register (EXCKSEL)**

The input clock of the CSI3 ($f_{QCSI}$) is provided either by the extension clock register or the system clock.

This is an 8-bit register that selects the internal clock. This register can be read or written in 8-bit or 1-bit units.

***Figure 2-7: Extension Clock Select Register (EXCKSEL) Format***

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Address | After reset |
|---|---|---|---|---|---|---|---|---|---|---|
| EXCKSEL | 0 | 0 | CB1CKSEL | CB0CKSEL | CS31CKSEL | CS30CKSEL | AF1CKSEL | AF0CKSEL | FFFFF30CH | 00H |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | | |

| CB1CKSEL | CSIB1 Clock selection |
|---|---|
| 0 | Normal clock selection ($f_{XX}$) |
| 1 | Extended clock selection ($f_{PLL\_PCKSEL}$) |

| CB0CKSEL | CSIB0 Clock selection |
|---|---|
| 0 | Normal clock selection ($f_{XX}$) |
| 1 | Extended clock selection ($f_{PLL\_PCKSEL}$) |

| CS31CKSEL | CSI31 Clock selection |
|---|---|
| 0 | Normal clock selection ($f_{XX}$) |
| 1 | Extended clock selection ($f_{PLL\_PCKSEL}$) |

| CS30CKSEL | CSI30 Clock selection |
|---|---|
| 0 | Normal clock selection ($f_{XX}$) |
| 1 | Extended clock selection ($f_{PLL\_PCKSEL}$) |

| AF1CKSEL | CAN1 Clock selection |
|---|---|
| 0 | Normal clock selection ($f_{XX}$) |
| 1 | Extended clock selection ($f_{PLL\_PCKSEL}$) |

| AF0CKSEL | CAN0 Clock selection |
|---|---|
| 0 | Normal clock selection ($f_{XX}$) |
| 1 | Extended clock selection ($f_{PLL\_PCKSEL}$) |

**Caution: Before modifying EXCKSEL register value, corresponding peripheral I/O function have to be stopped.**
**When set to 1 either bit 5 or 4 of EXCKSEL register, be sure to set bit 4 of OCKS2 register (OCKSEN2 bit) to 1.**

**Remark:** See RS1 user's manual for details description (UD16702EE1V0UD00)

**(5) Receive data buffer registers (SIRB0, SIRB1)**

The SIRB register is a 16-bit register or separated as upper 8 bits (SIRBH) and lower 8 bits (SIRBL), that is used to store receive data. This register can be read in 8-bit or 16-bit units and is initialized to 0000H by reset.

***Figure 2-8: Receive Data Buffer Registers (SIRB0, SIRB1) Format***

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Address | R/W | Initial value |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SIRBn (n=0, 1) | | | | | | | | | | | | | | | | | FFFF FD42H, FFFF FD62H | R/W | 0000H |

| | |
|---|---|
| SIRB15 - SIRB0 | Data received from the SIO serial shift register. |

**Caution: The receive data buffer register is considered as emptied by the application software whenever the lower 8 bits of the register are read. It is therefore necessary to read SIRBH before SIRBL when 8-bit access is used.**

**(6) Chip select data buffer registers (SFCS0, SFCS1)**

The SFCS register is a 16-bit register, or separated as upper 8 bits (SFCSH) and lower 8 bits (SFCSL), that stores Chip Select data. This register is read/write-enabled and is accessible in 8-bit or 16-bit units. Initial value is FFFFH by reset.

Following the FIFO write pointer, the value written to SFCS is stored in the FIFO data buffer as Chip Select bits. The value is stored to these bits when the transmit data is written to its register (SFDB or SFDBL).

SFCS write is prohibited when POWER = 1 and $f_{QCSI}$ is stopped.

***Figure 2-9: Chip Select Data Buffer Registers (SFCS0, SFCS1) Format***

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Address | R/W | Initial value |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SFCSn (n=0, 1) | - | - | - | - | - | - | - | - | - | - | - | - | SFCS3 | SFCS2 | SFCS1 | SFCS0 | FFFF FD44H, FFFF FD64H | R/W | FFFFH |

| SFCSm | Chip Select Output Selection (m=0 to 3) |
|---|---|
| 0 | Output an inactive level at the CS3nm pin during the data transfer |
| 1 | Output an active level at the CS3nm pin during the data transfer |

**Caution: The Chip Select register is stored in the FIFO buffer when the transmit data is written to its register. It is therefore necessary to write the SFCS register before the SFDB (SFDBL) register.**

**Remark:** The active level for each chip select is defined in the CSILn register (CSA[3:0] bits)

**(7) Transmission data buffer registers (SFDB0, SFDB1)**

The SFDB register is a 16-bit buffer register, or separated as upper 8 bits (SFDBH) and lower 8 bits (SFDBL), that stores transmission data. The SFDB register is read/write-enabled and is accessible in 8-bit or 16-bit units. Initial value is 0000H by reset.

Incrementing and following the FIFO buffer write pointer, the value written to SFDB is stored to the FIFO buffer as transmission data.

SFDB write is prohibited when POWER = 1 and $f_{QCSI}$ is stopped.

***Figure 2-10: Transmission Data Buffer Registers (SFDB0, SFDB1) Format***

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Address | R/W | Initial value |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SFDBn (n=0, 1) | | | | | | | | | | | | | | | | | FFFF FD46H, FFFF FD66H | R/W | 0000H |

| | |
|---|---|
| SFDB15 - SFDB0 | Write data is stored to the FIFO data buffer as transmission data.<br>A read operation reads out the last stored transmission data. |

**Caution: The transmit data buffer register is considered as written whenever the lower 8 bits of the register are written. It is therefore necessary to write SFDBH before SFDBL when 8-bit access is used.**

**(8) FIFO buffer status registers (SFA0, SFA1)**

The SFA register is an 8-bit register that shows the FIFO buffer status. The SFA register is read/write-enabled, accessible in 1-bit or 8-bit units. However, the bits SFFUL, SFEMP and CSOT are read only. Initial value is 20H by reset.

SFA read is prohibited when POWER = 1 and $f_{QCSI}$ is stopped.

***Figure 2-11: FIFO Buffer Status Registers (SFA0, SFA1) Format (1/2)***

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Address | R/W | After reset |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SFAn (n = 0, 1) | FPCLR | SFFUL | SFEMP | CSOT | SFP3 | SFP2 | SFP1 | SFP0 | FFFFFD48H, FFFFFD68H | R/W | 20H |

| FPCLR | FIFO buffer pointer clear command |
|---|---|
| 0 | No operation |
| 1 | Clear all FIFO pointers to "0" |

**Remark:** Read value is always “0”.

***Figure 2-12: FIFO Buffer Status Registers (SFA0, SFA1) Format (2/2)***

| SFFUL | FIFO buffer full status flag |
|---|---|
| 0 | FIFO buffer is not full |
| 1 | FIFO buffer is full |

**Remark:** Read only

| SFEMP | FIFO buffer empty status flag |
|---|---|
| 0 | FIFO buffer is not empty |
| 1 | FIFO buffer is empty |

**Remark:** Read only

| CSOT | Transmission status flag |
|---|---|
| 0 | Idle state |
| 1 | Transmission in on going or preparing |

**Remarks: 1.** Read only bit.

**2.** This bit is cleared to “0” by POWER = 0 or (CTXE = 0 and CRXE = 0).

**3.** In Single Buffer Transfer Mode, this bit holds “1” from transmission start to FIFO empty.

**4.** In FIFO Buffer Transfer Mode, this bit holds “1” from transmission start until finish transferring all data to be sent.

| SFP3 - SFP0 | Transmission data count |
|---|---|
| nnnnH | In Single transfer mode, SFP3 - SFP0 indicates the number of remaining transfers in the FIFO. This value can be understood as:<br>(Write FIFO pointer) - (SIO load pointer)<br>SFP3 - SFP0 is read only in Single transfer mode. |
| | In FIFO transfer mode,SPF3 - SFP0 indicates the number of data transfers completed.<br>In case of SFP3 - SFP0 = 0H:<br>- SFEMP = 0, SFP3 - SFP0 = 0H: Number of receptions completed = 0<br>- SFEMP = 1, SFP3 - SFP0 = 0H: Number of receptions completed = 16 |

**Remarks: 1.** Read only bits.

**2.** SFP3 - SFP0 holds its value until RESET or FPCLR = 1.

**Caution: SFFUL, SFEMP, CSOT and SFP3 - SFP0 are continuously updated with the current status of the Queued CSI. This means that a value read might be outdated shortly after the read was executed.**
**When writing accidentally a 17th data element in FIFO, an overflow interrupt (INTC3nO) will occur to indicate the error.**

**(9) Queued CSI data length selection registers (CSIL0, CSIL1)**

The CSIL register is an 8-bit register that specifies the active voltage level of the Chip Select pins and the Queued CSI data length.
This register can be read or written in 1-bit and 8-bit units.
This register can be overwritten only while CTXE = 0 and CRXE = 0 (CSIM register). Write operation during CTXE = 1 or CRXE = 1 is prohibited.
Initial value is 00H by reset.

***Figure 2-13: Queued CSI Data Length Selection Registers (CSIL0, CSIL1) Format***

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Address | R/W | After reset |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CSILn (n = 0, 1) | CSLV3 | CSLV2 | CSLV1 | CSLV0 | CCL3 | CCL2 | CCL1 | CCL0 | FFFFFD49H, FFFFFD69H | R/W | 00H |

| CSLV3 - CSLV0 | Chip Select active level selection |
|---|---|
| 0 | Chip Select signal is active low |
| 1 | Chip Select signal is active high |

| CCL3 | CCL2 | CCL1 | CCL0 | Transfer data length |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Data length is 16 bits |
| 0 | 0 | 0 | 1 | Setting prohibited |
| 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 0 | |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | Data length is 8 bits |
| 1 | 0 | 0 | 1 | Data length is 9 bits |
| 1 | 0 | 1 | 0 | Data length is 10 bits |
| 1 | 0 | 1 | 1 | Data length is 11 bits |
| 1 | 1 | 0 | 0 | Data length is 12 bits |
| 1 | 1 | 0 | 1 | Data length is 13 bits |
| 1 | 1 | 1 | 0 | Data length is 14 bits |
| 1 | 1 | 1 | 1 | Data length is 15 bits |

**Caution: Rewriting this register is only permitted when CTXE = 0 and CRXE = 0.**

See section **2.3.3 "Data length select function" on page 30** for further explanation on the data length selection.

### (10) Queued CSI transfer number selection registers (SFN0, SFN1)

The SFN register is an 8-bit register that specifies the number of data elements to be transferred in FIFO buffer transfer mode. It can be read or written in 1-bit and 8-bit units.
Initial value is 00H by reset.

***Figure 2-14: Queued CSI Transfer Number Selection Registers (SFN0, SFN1) Format***

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Address | R/W | After reset |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SFNn (n = 0, 1) | 0 Note | 0 Note | 0 Note | 0 Note | SFN3 | SFN2 | SFN1 | SFN0 | FFFFFD4CH, FFFFFD6CH | R/W | 00H |

**Note:** Unused bits must be written as 0.

| SFN3 | SFN2 | SFN1 | SFN0 | Transfer data number |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Transfer 16 data elements |
| 0 | 0 | 0 | 1 | Transfer 1 data element |
| 0 | 0 | 1 | 0 | Transfer 2 data elements |
| 0 | 0 | 1 | 1 | Transfer 3 data elements |
| 0 | 1 | 0 | 0 | Transfer 4 data elements |
| 0 | 1 | 0 | 1 | Transfer 5 data elements |
| 0 | 1 | 1 | 0 | Transfer 6 data elements |
| 0 | 1 | 1 | 1 | Transfer 7 data elements |
| 1 | 0 | 0 | 0 | Transfer 8 data elements |
| 1 | 0 | 0 | 1 | Transfer 9 data elements |
| 1 | 0 | 1 | 0 | Transfer 10 data elements |
| 1 | 0 | 1 | 1 | Transfer 11 data elements |
| 1 | 1 | 0 | 0 | Transfer 12 data elements |
| 1 | 1 | 0 | 1 | Transfer 13 data elements |
| 1 | 1 | 1 | 0 | Transfer 14 data elements |
| 1 | 1 | 1 | 1 | Transfer 15 data elements |

## 2.3 Explanation of Queued CSI Functions

### 2.3.1 Transmit buffer

Chip select data and transmission data can be stored to the transmit FIFO buffer continuously by writing to the SFCS register and SFDB register. The Writing FIFO pointer is automatically incremented when data is written to SFDB. The size of the transmit FIFO buffer is 20 bits × 16 entries.

In slave mode, chip select data does not need to be set.

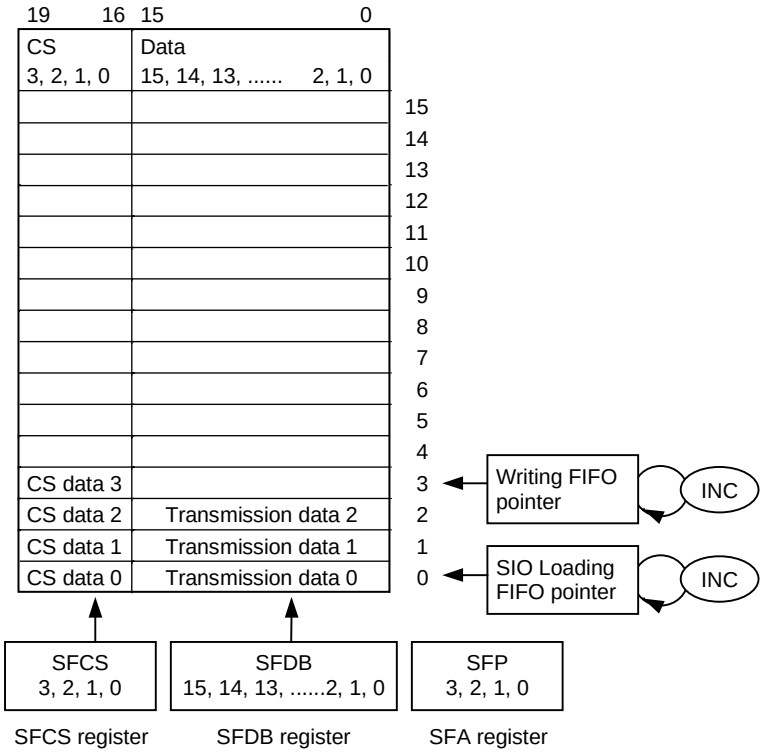
The transfer start condition (SFEMP = 0) is to write to the lower bits of SFDB register. If the transmission data length is 9 bits or more, data should be written by a 16-bit write to SFDB or by two 8-bit writes to first SFDBH, then SFDBL (in that order). When transmission data length is 8 bits, data should be set by one 8-bit write to SFDBL or by one 16-bit write to SFDB. For the 16-bit write the upper 8 bits are ignored in the 8-bit transmission.

The SFFUL bit in the SFA status register is set “1” after 16 writes have been made to the transmit buffer, assuming the Writing FIFO pointer was reset previously.

When a transmission write is attempted while the FIFO is full (SFFUL=1), the interrupt INTC3nO is generated to indicate an overflow. In that case, the transmission and chip select data are discarded and not stored.

When a transfer cycle is finished and the SIO Loading FIFO pointer is incremented, the FIFO buffer of the previous location is considered empty in case of single buffer transfer mode. Refer to section 2.3.8 on page 34 for more information on FIFO buffer transfer mode.

***Figure 2-15: Transmit Buffer***

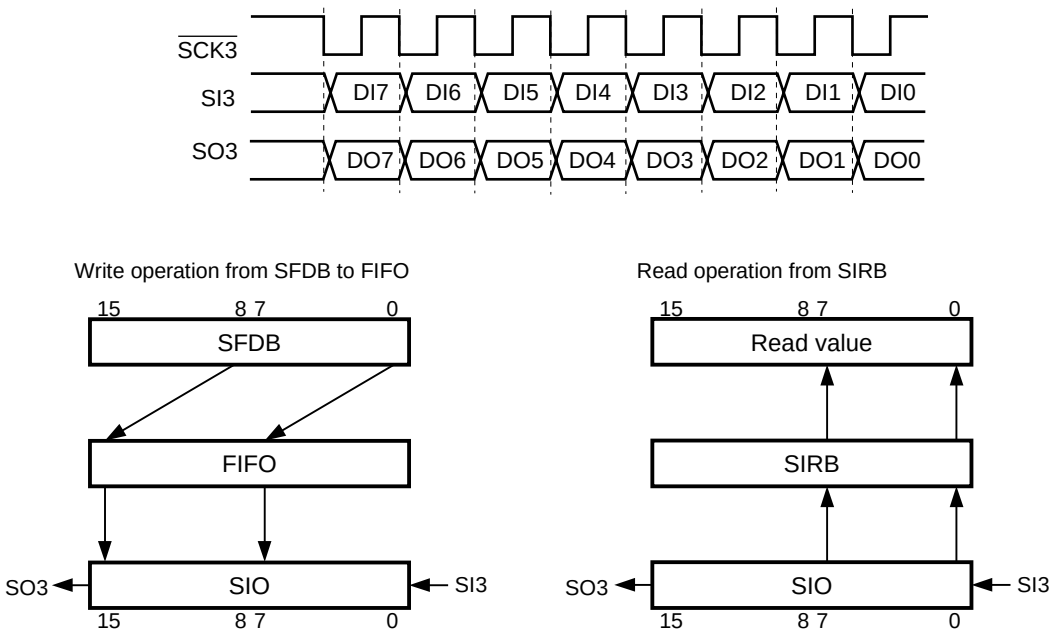


### 2.3.2 Serial data direction select function

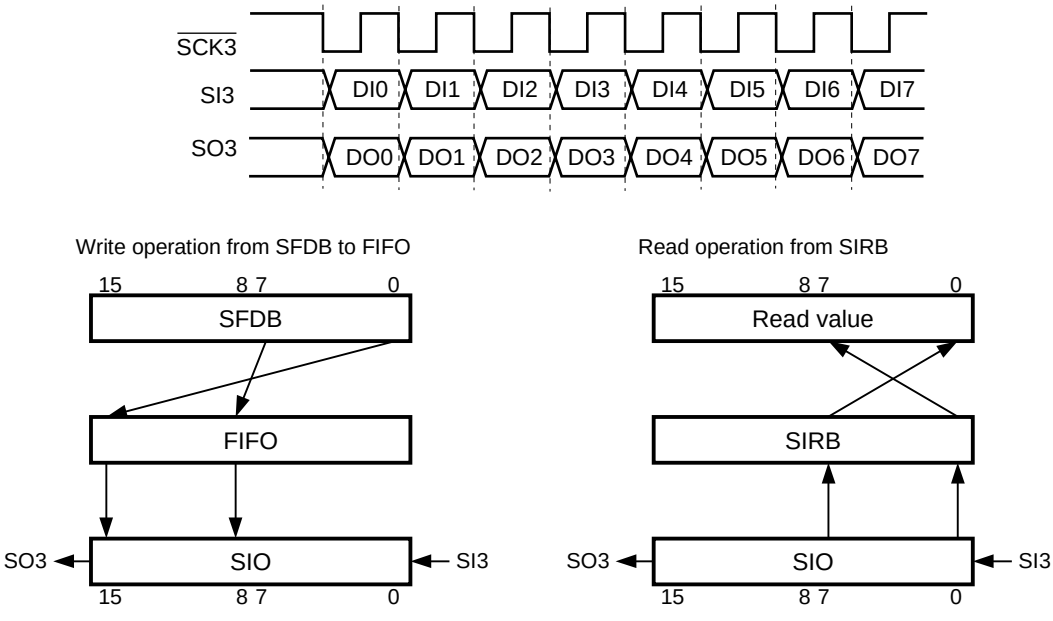
The serial data direction is selectable using the DIR bit in the CSIM register. The examples below show the communication for data length of 8 bit (CCL[3:0] = [1,0,0,0]):

***Figure 2-16: Serial Data Direction Select Function***

***(a) MSB first (DIR = 0)***



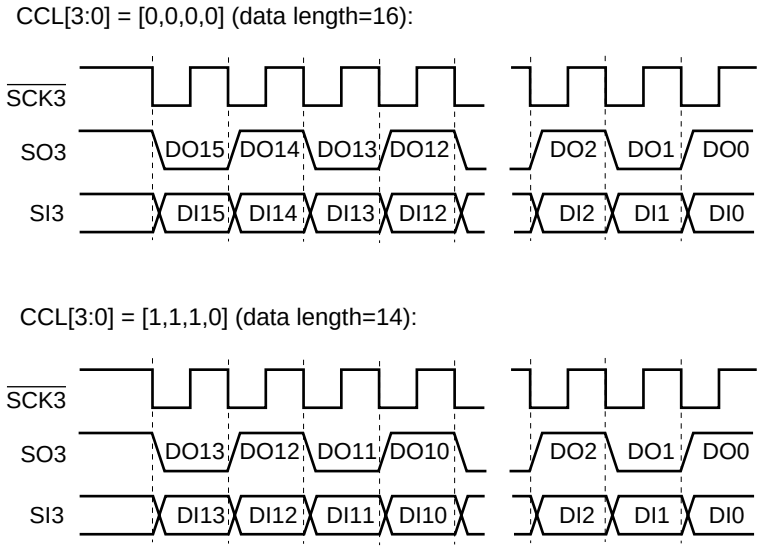

***(b) LSB first (DIR = 0)***



### 2.3.3 Data length select function

Transmission data length is selectable from 8 bits to 16 bits using the CCL[3:0] bits in CSIL register. The examples below show the communication with MSB first (DIR = 1):

***Figure 2-17: Data Length Select Function***

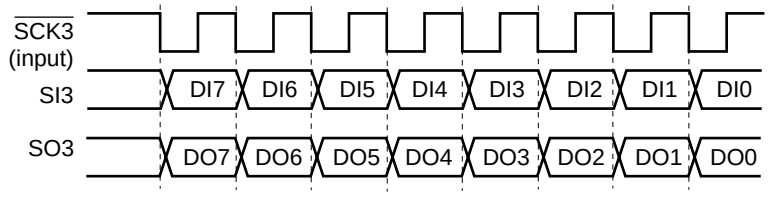



### 2.3.4 Slave mode

When the CKS[2:0] bits in CSIC are set to [1,1,1], the Queued CSI operates in slave mode. In slave mode, the SCK3 serial clock pin becomes input and another device is the CSI communication master. The baud rate generator “BRG” is recommended to be disabled by setting bits MDL[2:0] to [0,0,0] when using slave mode. Also, the chip select pin CS3n[3:0] outputs are not available in slave mode, as they are only available in master mode.

The example below shows the communication in slave mode for 8 data bits, CKP=0, DAP=0 and MSB first:

***Figure 2-18: Slave Mode***

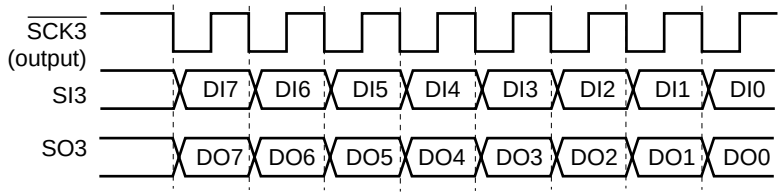


### 2.3.5 Master mode

When the CKS[2:0] bits in CSIC are not set to [1,1,1], the Queued CSI operates in master mode. In master mode, the $\overline{SCK3}$ pin is configured as output and the serial communication clock is generated by the Queued CSI module. The $\overline{SCK3}$ pin's default value is "1" when CKP = 1, and is default "0" when CKP = 0. The CS3n[3:0] pin outputs are available in master mode.

The example below shows the communication in master mode for 8 data bits, CKP=0, DAP=0 and MSB first:

***Figure 2-19: Master Mode***

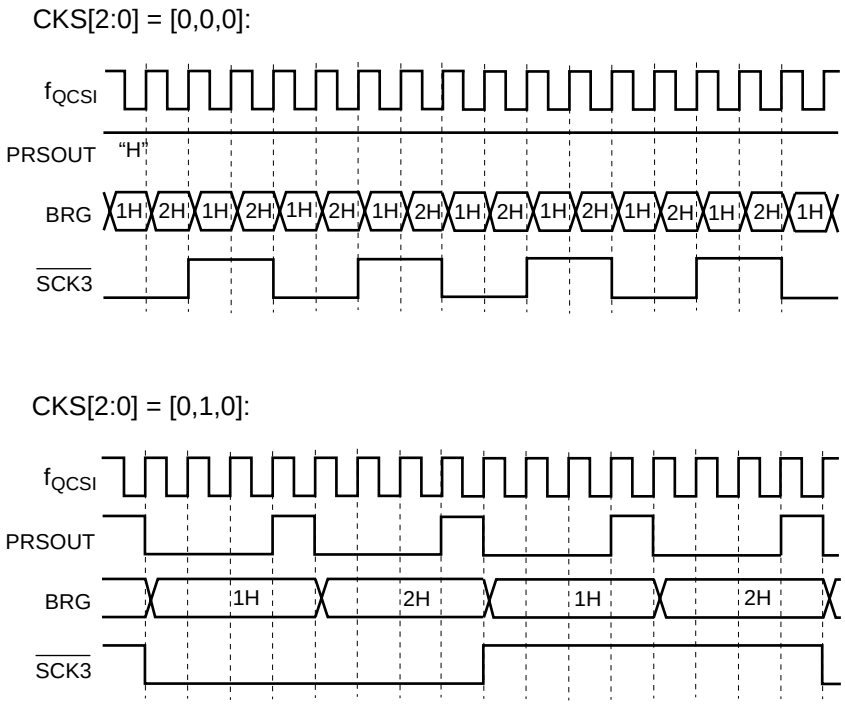



### 2.3.6 Transmission clock select function

In Master Mode, the transfer baud rate is selectable using CKS[2:0] bits and MDL[2:0] bits in CSIC register. The baud rate generator "BRG" counts up at each rising edge of $f_{QCSI}$.

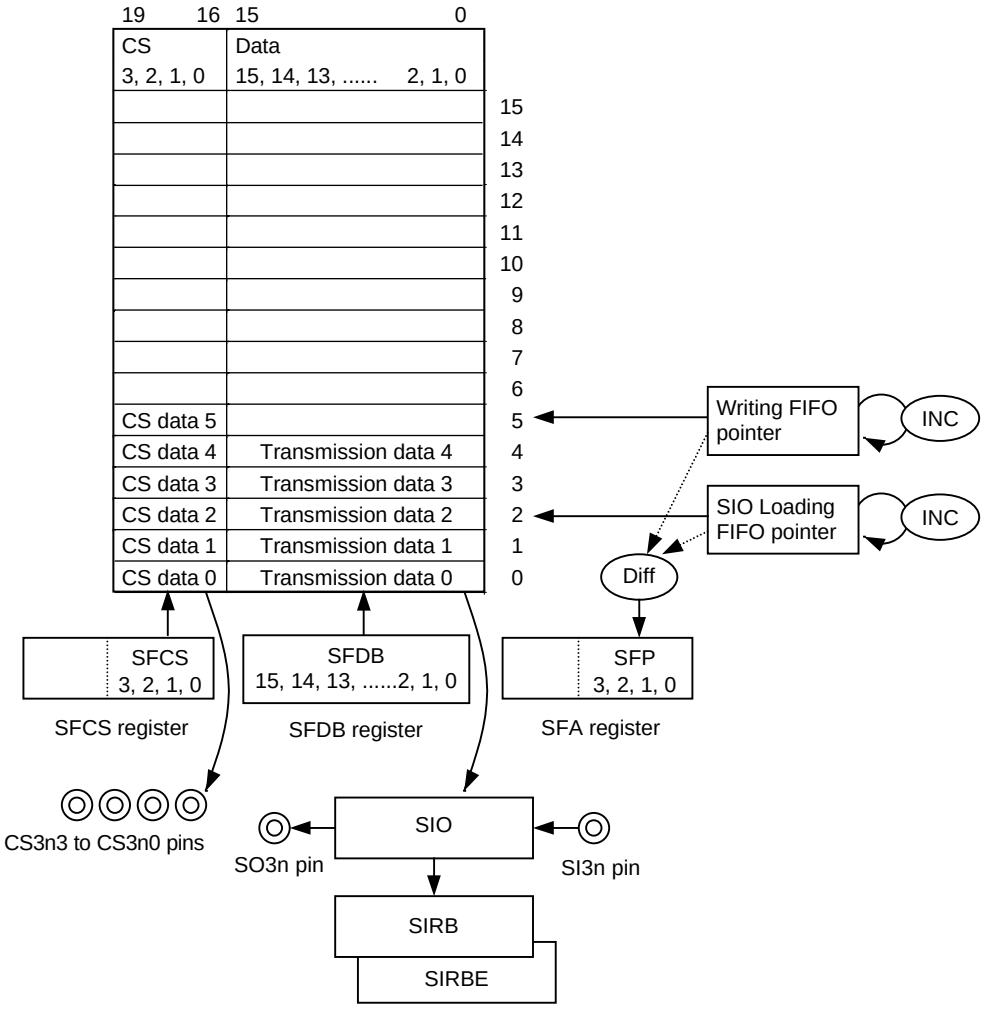
The example below illustrates the baud rate generation for MDL[2:0] = [0,1,0].

***Figure 2-20: Transfer Clock Select Function***



### 2.3.7 Description of the single buffer transfer mode

***Figure 2-21: Single Buffer Transfer Mode Data Handling***




Transfer start condition in single buffer transfer mode:

[CTXE = 1 or CRXE = 1] and
[Data exists in FIFO (SFEMP = 0)]

A transfer starts once the transmission data (pointed to by the SIO Loading FIFO pointer) is transferred from the FIFO buffer to the serial shift register SIO. At that time, the transfer status flag CSOT turns to “1”. The CS3n[3:0] pins output the chip select data from the FIFO buffer.

At the end of the transfer:

(A) If SIRB is empty, the received data is stored from SIO to SIRB, and transfer end interrupt signal INTC3nI is generated (in transmit only mode, INTC3nI will be generated only when the FIFO buffer becomes empty). Finally, the SIO loading FIFO pointer is incremented.

(B) If SIRB is not empty, the storing of receive data, INTC3nI generation and SIO Loading FIFO pointer incrementing wait for the SIRB to be emptied by a software read operation.

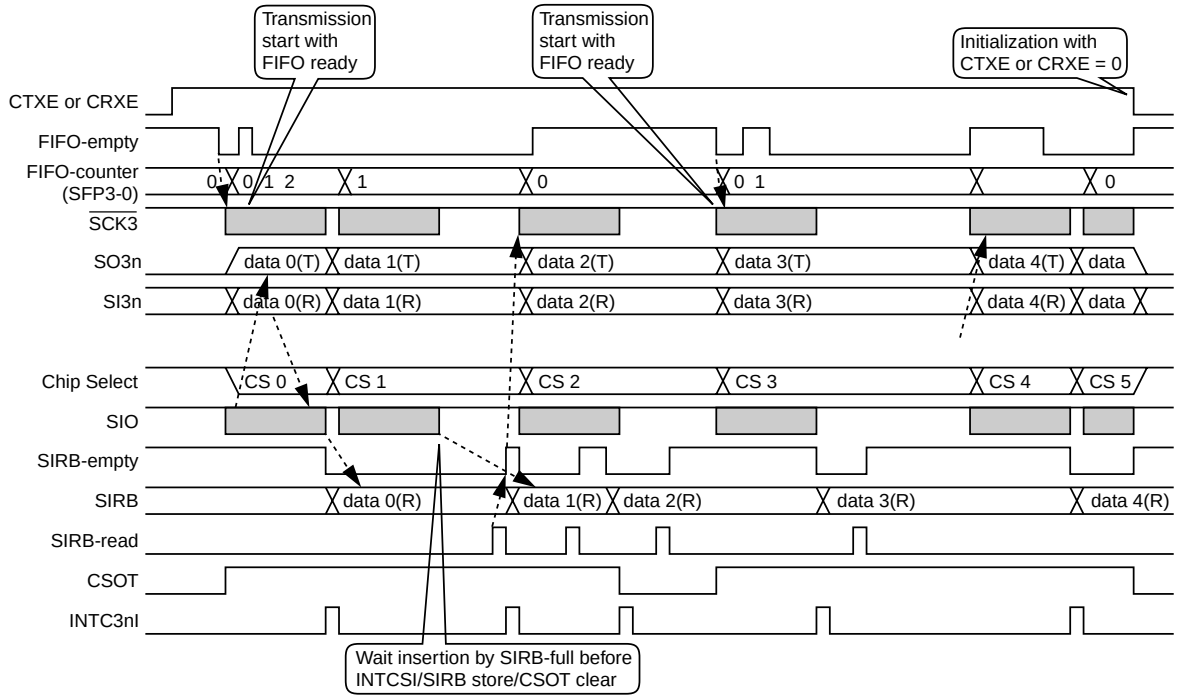
(C) In transmit only mode, if transmission data is available in the FIFO buffer, the next transfer will start immediately, regardless of the SIRB buffer condition.

When a transfer finishes and the FIFO buffer is empty (Writing FIFO pointer = SIO Loading FIFO pointer), CSOT is cleared “0”.

SFP[3:0] always show the current value of: (Writing FIFO pointer) - (SIO Loading FIFO pointer).

It is recommended to check that SFFUL = 0 just before data is written to the SFDB register.
If SFFUL = 1 when a write to SFDB is attempted, the overflow interrupt INT3nO is generated and the written data is ignored.

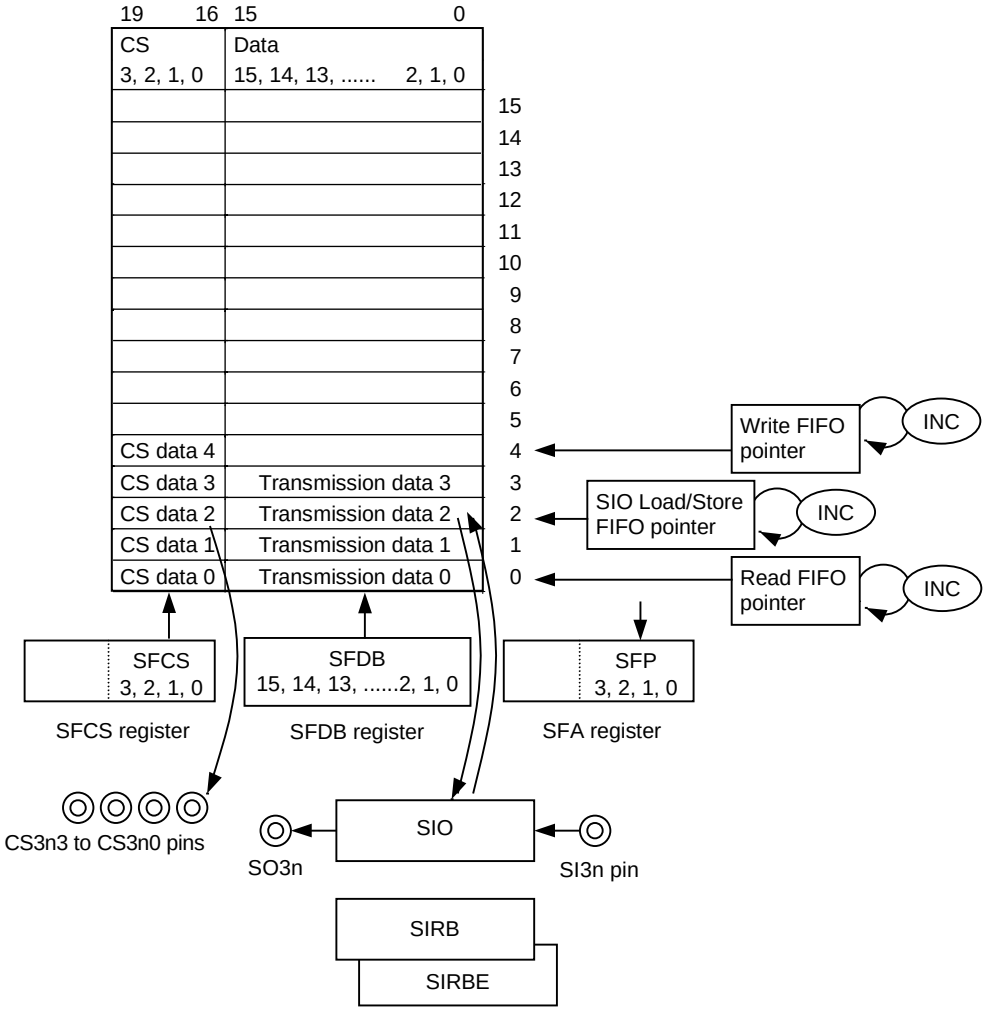
***Figure 2-22: Single Buffer Transfer Mode (Master, Transmit/Receive) Timing***



### 2.3.8 Description of the FIFO buffer transfer mode

When the TRMD bit in the CSIM register is set “1”, the Queued CSI operates in FIFO buffer transfer mode.

***Figure 2-23: FIFO Buffer Transfer Mode Data Handling***




Transfer start condition in FIFO buffer transfer mode:

[CTXE = 1 or CRXE = 1] and
[Data exists in FIFO (SFEMP = 0)]

The transmission data number must be set in SFN[3:0]. Note that writing a value greater than 16 to the SFN register is prohibited, as the FIFO buffer design can hold up to 16 elements only.

The transfer starts by copying the first data element - pointed to by SIO Load/Store FIFO pointer - to the SIO shift register. At that time the transmission status flag CSOT is set to “1”, and the CS3n[3:0] pins output the CS value from the FIFO.

When the transfer of the data element is finished, the received data overwrites the location in the FIFO using the SIO Load/Store FIFO pointer, and the SIO Load/Store FIFO pointer is then incremented.

When the transmission/reception counter reaches the value set by SFN[3:0], then CSOT is cleared “0” and the transmission/reception end interrupt signal INTC3nI is generated.

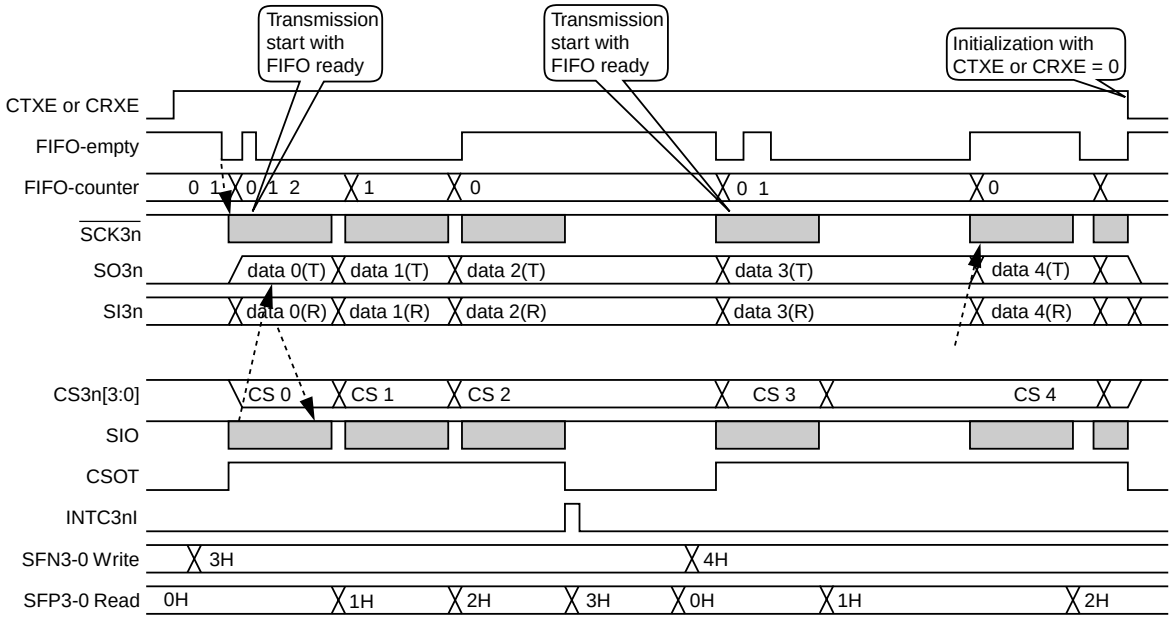
After the interrupt occurred, the received data can be read from SIRB. The Read FIFO pointer is automatically incremented by the SIRB read operation.

All FIFO pointers must be cleared by setting FPCLR = 1 before the next transmit/receive cycle can start.

SFP[3:0] represents the [SIO Load/Store FIFO pointer] and shows the number of transmission/receptions completed. In case of SFP[3:0]=0H, the numbers of transmissions/receptions depends on the setting of the SFEMP bit:

SFEMP=0: 0 transmissions/receptions completed
SFEMP=1: 16 transmissions/receptions completed

***Figure 2-24: FIFO Buffer Transfer Mode (Master. Transmit/Receive) Timing***



### 2.3.9 Description of the operation modes

**(1) Transmit only mode**

Setting the CSIM register's CTXE =1 and CRXE = 0 places the Queued CSI in transmit only mode. A transmission starts when transmit data is written in the SFDB register. The current condition of the SIRB buffer and SIO register no effect. The data in the SIRB and the SIO buffer is undefined after completion of the transmission.

**(2) Receive only mode**

Setting the CSIM register's CTXE = 0 and CRXE = 1 places the Queued CSI in receive only mode. A reception starts when dummy data is written in the SFDB register. It is mandatory, though, that the SIRB and SIO are empty. If a receive operation is terminated while the previous receive data remains unread in SIRB, the Queued CSI is placed on wait status until the previous data is completely read and SIRB becomes empty.

**(3) Transmit / receive mode**

Setting the CSIM register's CTXE = 1 and CRXE = 1 places the Queued CSI in transmit/receive mode. A transfer (meaning transmission and reception) starts when transmit data is written in the SFDB register. Note that an empty SIRB or SIO is mandatory. If a receive operation is terminated while the previous receive data remains unread in SIRB, the Queued CSI is placed on wait status until the previous data is completely read and SIRB becomes empty.

In FIFO buffer transfer mode with slave mode, only the first dummy data write operation is required. There is no need to write chip-select data, as these bits are ignored.

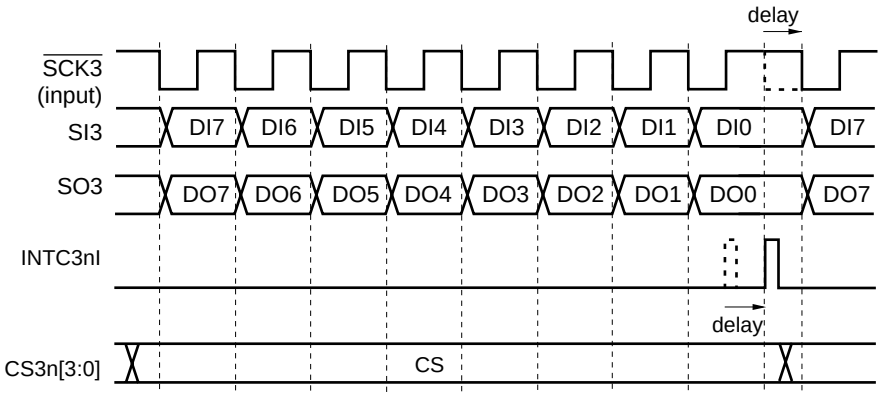
### 2.3.10 Additional timing and delay selections

**(1) Delay selection of receive termination interrupt signal (INTC3nI)**

In master mode, the CSIT bit of the CSIM register can be used to delay the generation of the receive termination interrupt signal (INTC3nI) by a half serial clock cycle ($\overline{SCK3}$). The CSIT bit takes effect only in the master mode and is ignored in slave mode.

Figure 2-25 below illustrates the CSIT function, assuming a setting of CSIT=1, CSWE=0, CKP=0, DAP=0 and CCL[3:0] = [1,0,0,0].

***Figure 2-25: Delay Selection of Receive Termination Interrupt (INTC3nI)***

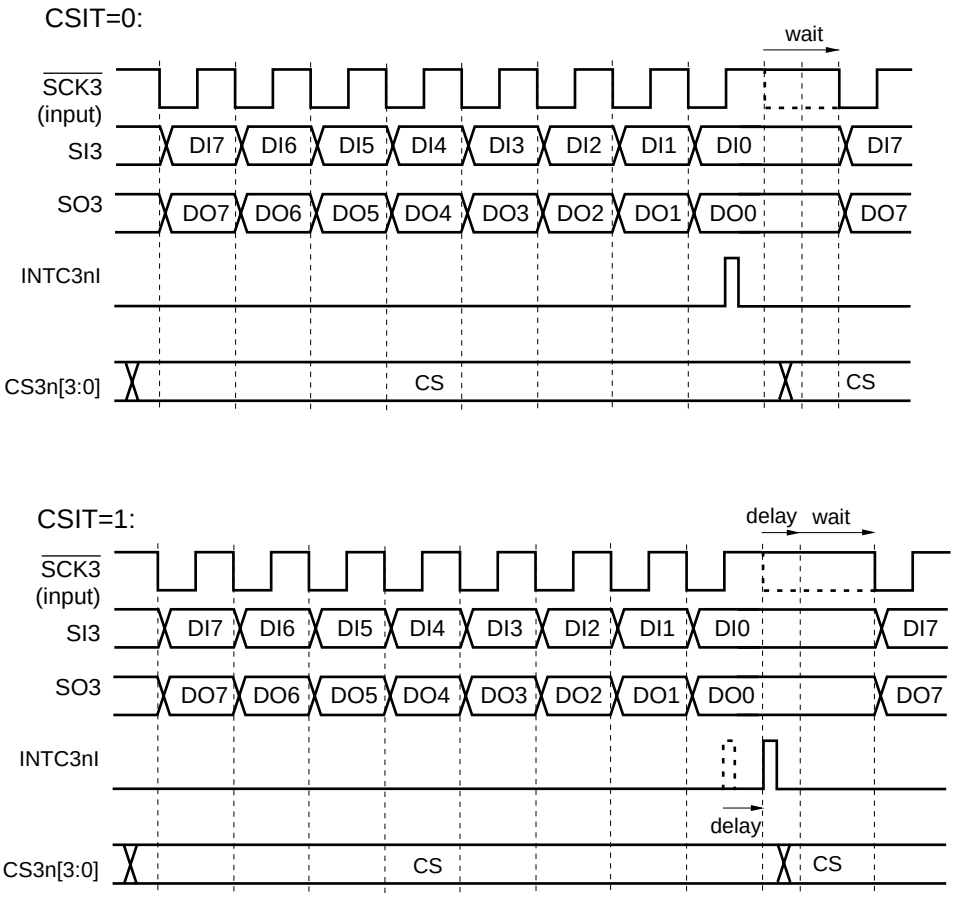


**(2) Selection of transmit wait enable/disable**

In master mode, the CSIM register's CSWE bit setting can be used to delay the start of transmission by one $\overline{SCK3}$ clock cycle. The CSWE bit takes effect only in the master mode and is ignored in slave mode.

Figure 2-26 below illustrates the CSWE function, assuming a setting of CSWE=1, CSMD=0, CKP=0, DAP=0 and CCL[3:0] = [1,0,0,0].

***Figure 2-26: Selection of Transmit Wait Enable/Disable***



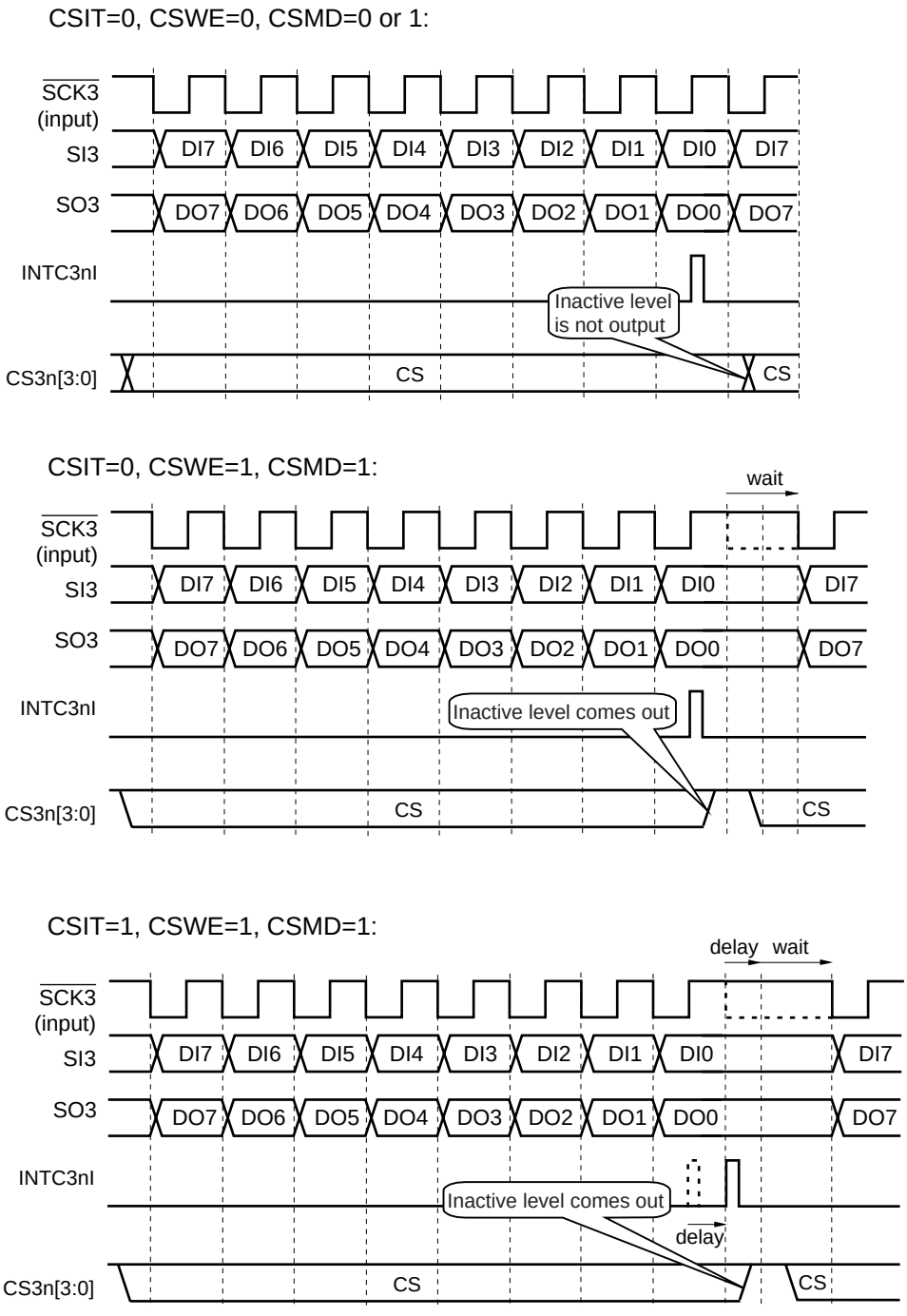
**(3) Selection of chip-select mode**

In master mode with CSWE = 1, the CSMD bit setting can be used to output an inactive level at the chip select pins CS3n[3:0] during the delay between data transmissions. The CSMD bit takes effect only in the master mode and is ignored in slave mode.

The CSMD bit has no effect while CSWE is set to 0.

Figure 2-27 below illustrates the CSMD function, assuming a setting of CKP=0, DAP=0 and CCL[3:0] = [1,0,0,0] and the active levels of all chip selects set to “active low”.

***Figure 2-27: Selection of Chip-Select Mode***



### 2.3.11 Default pin levels

**(1) $\overline{\text{SCK3}}$ pin's default level**

$\overline{\text{SCK3}}$ pin's default level with the CSIM register settings POWER = 0 or CTXE / CRXE = 0

| CKP | CKS2, CKS1, CKS0 | $\overline{\text{SCK3}}$ default level | |
|---|---|---|---|
| 0 | 1, 1, 1 (slave mode) | 1 | ← Initialization after reset |
| | Other than 1, 1, 1 (master mode) | 1 | |
| 1 | 1, 1, 1 (slave mode) | 1 | |
| | Other than 1, 1, 1 (master mode) | 0 | |

**Remark:** In slave mode, the $\overline{\text{SCK3}}$ pin is always set to “1.”

**(2) SO3 pin's default level**

SO3 pin's default level with the CSIM register settings POWER = 0 or CTXE / CRXE = 0

| SO3 pin's default level | |
|---|---|
| 0 | ←Initialization after reset |

**(3) CS3n0 to CS3n3 pins' default level**

CS3n0 to CS3n3 pins' default level with the CSIM register settings POWER = 0 or CTXE / CRXE = 0

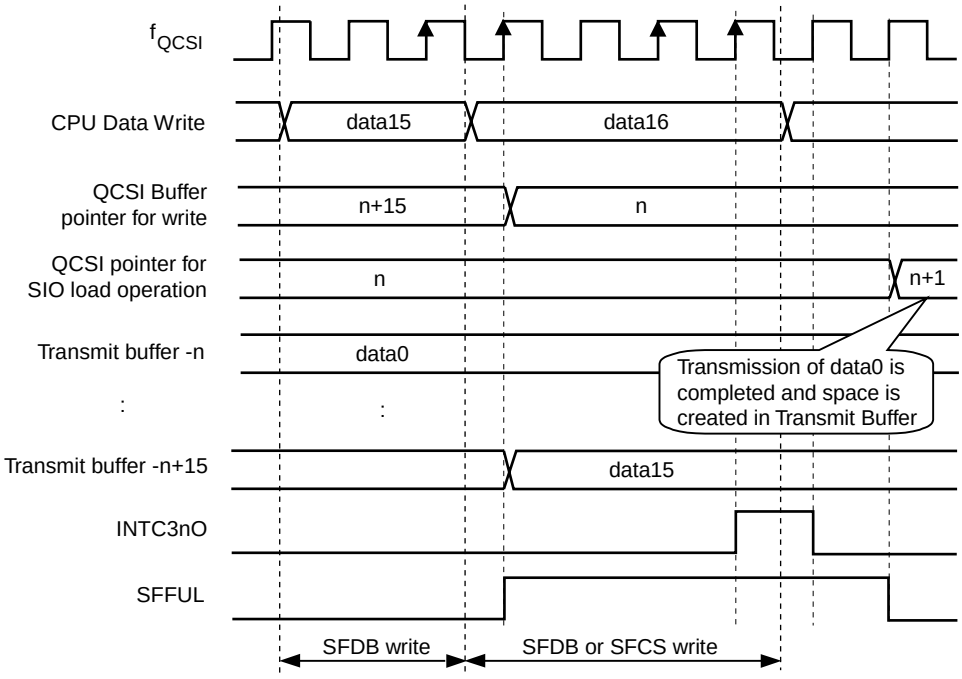
| CS3n0 to CS3n3 pins' default level | |
|---|---|
| inactive | ←Initialization after reset |

### 2.3.12 Transmit buffer overflow interrupt signal (INTC3nO)

When the transmit FIFO buffer contains 16 elements, writing a 17th chip-select data (SFCS write) or transfer data (SFDB write) results in the generation of the overflow interrupt INTC3nO. For the 17th item, both chip-select and transfer data values are discarded.

The transmit FIFO buffer contains 16 elements if the FIFO pointer value for the write operation equals the FIFO pointer value for the SIO load operation plus 15. When the transfer is completed and the FIFO buffer pointer for the SIO load operation is incremented, space for one element is available again in transmit buffer.

***Figure 2-28: Transmit Buffer Overflow Interrupt Signal (INTC3nO)***

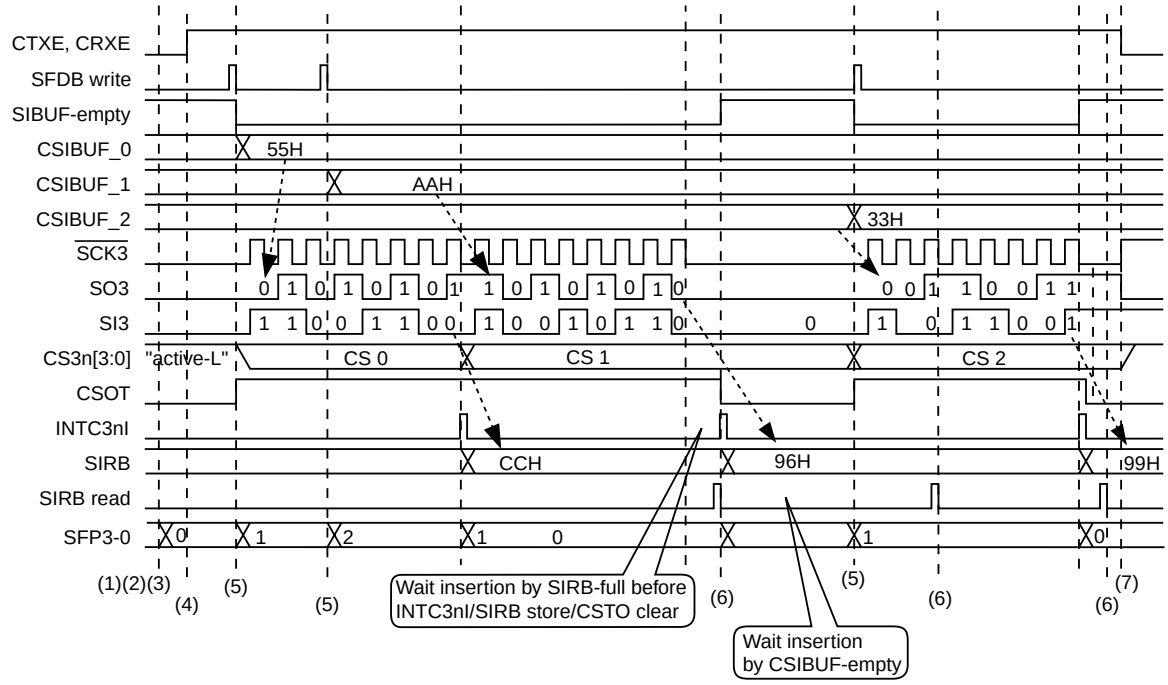


# Chapter 3 Operating Procedure

## 3.1 Single Buffer Transfer Mode (Master Mode, Transmit/Receive Mode)

MSB first (DIR = 0), no INTC3nI delay (CSIT = 0), transmission wait disabled (CSWE = 0), CS inactive disabled (CSMD = 0), CKP = 1, DAP = 0, transmission data length of 8 bits (CCL[3:0] = [1,0,0,0]), active levels of all chip selects set to “active low”:

***Figure 3-1: Single Buffer Transfer Mode (Master, Transmit/Receive) Timing***




1. Set the CSIM register's POWER bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CSIC and CSIL registers to specify the transfer mode.
3. Write “1” in the SFA register's FPCLR bit to clear all FIFO pointers.
4. Specify the transfer mode using the CSIM register's TRMD, DIR, and CSIT bits; at the same time, set the CTXE and CRXE bits to 1 to enable the transmit/receive operation.
5. Make sure that the SFA register's SFFUL bit is set to 0, then write chip-select data and transmission data in the SFCS and SFDB registers in this order.
6. Check for a transmission to be finished (e.g. by monitoring the INTC3nI interrupt). If so, read the SIRB register.

   Repeat steps (5) and (6) until the last element is send/received and read from the SIRB register.

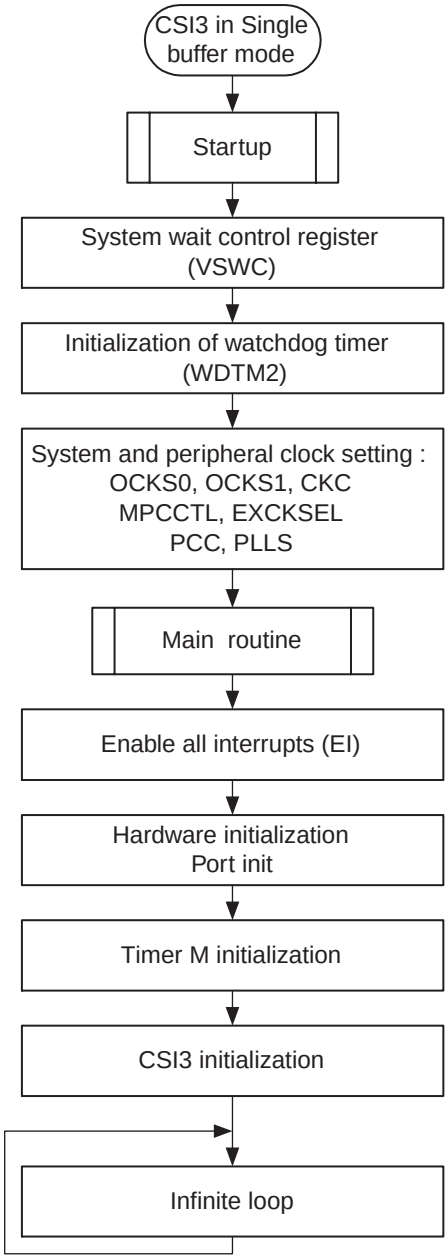
7. Set the CSIM register's CTXE and CRXE bits to 0 to disable the transmit/receive operation (end of transmit/receive operation).

### 3.1.1 Single buffer transfer mode application without DMA transfer

Only one cell of the FIFO buffer is used to transmit and receive the data.
Consequently it is useless to select in advance the number of data to be transmitted using SFN register (SFN register is only used is FIFO buffer mode).
Transmission is completed when one transmit/receive transfer of data (8 bits length) has been sent/ received.
Transmission completed status is detected by monitoring the INTCSI3n interrupt.
The reading of received data is done every interrupt INTCSI3n
DMA transfer is not used.

#### (1) Single buffer transfer flowchart

***Figure 3-2: Single Buffer Transfer Flowchart***



**(2) System clock and peripheral clock setting (Startup)**

This routine sets the clock using the PLLs mode for the CPU, CSI3 and Timer M.
PLL0 and PLL1 output clocks are respectively assigned to the system clock and CSI3.
System clock is used for CPU and timer.
The main resonator oscillates at 8 MHz.

Be sure to set the following registers first when using the V850E/RS1.

- System wait control register (VSWC)
- On-chip Debug Mode control register (OCDM)**Note**
- Watchdog timer mode register 2 (WDTM2)**Note**

**Note:** Refer to 3.4.7 “Caution” in V850E/RS1 user’s manual.

**(a) NEC peripheral bus wait control**

The system wait control register (VSWC) controls wait of bus access to the internal peripheral I/O registers.
Set the value to VSWC register in accordance with the frequency used.
The VSWC register can be read or written in 8-bit units.
In this application the value written to VSWC is 0x12H.
VSWC = 0x12H;

**(b) Clock selection register for PLL0 and PLL1 (OCKS0, OCKS1)**

Those 8-bit registers control the operation enable and clock input selection for PLL0 and PLL1.

Clock selection register 0: OCKS0 = 0x11H PLL0 input = $f_{XX}/3$
Clock selection register 1: OCKS1 = 0x13H PLL1 input = $f_{XX}/5$

**(c) Clock control register (CKC)**

This is an 8-bit register that controls system clock ($f_{XX}$) when it operates on PLL mode (SELPLL bit of MPCCTL register is set to 1). This register can be read or written in 8-bit or 1-bit units.
Be sure to clear bits 2 to 7 of the CKC register to 0. In other case the operation is not guaranteed.
Data can be written to this register only in combination of specific sequences (refer to 3.2.3 “Special registers” in V850E/RS1 user’s manual).
CKC = 0x03H $f_{PLL\_MCKSEL\_CKDIV} = f_{PLL}$(PLL internal clock)

**(d) Main peripheral clock control register (MPCCTL)**

This is an 8-bit register that selects the internal clock. Data can be written to this register only in combination of specific sequences (refer to 3.2.3 “Special registers” in V850E/RS1 user’s manual). The clock generator flexibility allows to use either simultaneously both PLL or to use only one PLL. Both PLLs can be assigned respectively to the CPU or to the peripherals.

Switch from oscillator clock to PLL clock:
MPCCTL = 0x84H PLL mode ($f_{XX} = f_{PLL\_MCKSEL\_CKDIV}$)
Main clock ($f_{PLL\_MCKSEL}$) = PLL0
Peripheral clock = PLL1

**(e) Extended clock selection (EXCKSEL)**

CSI3, CSIB, and AFCAN peripherals input clocks could be either from the system clock or directly form the PLL output using the extended clock selection.
In this application CSI3 uses the extended clock in order to get a 16 MHz input clock ($f_{QCSI}$) while the CPU clock is 32 MHz.

EXCKSEL = 0x0CH

CSI30 Clock = $f_{PLL\_PCKSEL}$ = 16 MHz
CSI31 Clock = $f_{PLL\_PCKSEL}$= 16 MHz

**(f) Processor clock control register (PCC)**

This 8-bit register controls the CPU clock selection using a prescaler.
Data can be written to this register only in combination of specific sequences (refer to 3.2.3 “Special registers” in V850E/RS1 user’s manual).
PCC = 0x00H

$f_{CPU} = f_{XX}$ = 32 MHz

**(3) Timer M Configuration in interval time mode**

Timer M (TMM0) supports only a clear & start mode. It does not support a free-running mode.
In the interval timer mode, a match interrupt signal (INTTM0EQ0) is output when the value of the 16-bit counter matches the value of TMM0 compare register 0 (TM0CMP0). At the same time, the counter is cleared to 0000H and starts counting up.

**(a) TMM0 timer control register (TM0CTL0)**

This register sets the count clocks of 16-bit timer.
In this application the count clock is set to $f_{XX}$.
TM0CTL0 = 0x80H

**(b) Timer M compare register**

In this application a timer interval of 300 µs is needed as time base. The value of compare register can be calculated by the following formula:

$$\text{Interval Time} = \frac{\text{Compare register}}{\text{Internal count clock}}$$

$$300\ \mu s = \frac{\text{Compare Register TM0CMP0}}{32MHz}$$

TM0CMP0 = 0x2580H

**(c) Timer M interrupt mask register is enabled with highest priority**

TM0EQIC0 register or IMR1 register could be used to setup the enable interrupt mode.
In this application TM0EQIC0 is set:

TM0EQIC0 = 0x00H

**(d) Flowchart**

***Figure 3-3: Timer M Initialization Routine***

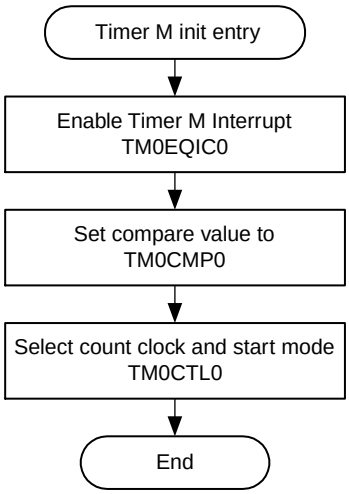
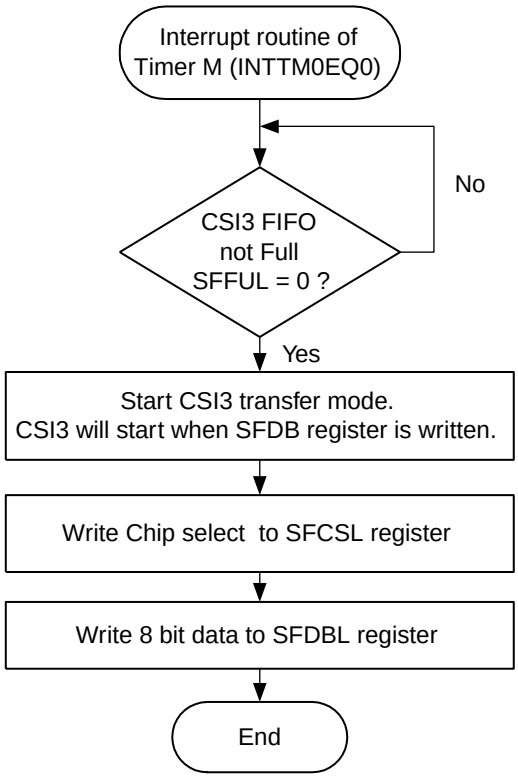



***Figure 3-4: Timer M Interrupt Routine***



**(4) CSIC3 configuration in single buffer mode**

3-wire serial I/O mode interface

**(a) Queued CSI Clock Selection Registers (CSIC)**

The CSIC register is an 8-bit register that is used to control the serial transfer operations (data phase, clock phase, master mode and clock selection).

**Caution: This register can be written only while CSIM register's CTXE = 0 and CRXE = 0.**

CSIC = 0x39H

| | |
|---|---|
| Master mode, $\overline{SCK3}$ = 4 MHz | CKS2, CKS1, CKS0 = 0, 0, 1<br>MDL2, MDL1, MDL0 = 0, 0, 1 |
| Sampling on first rising edge and idle level is low | CKP, DAP = 1, 1 |

**(b) Queued CSI Data Length Selection Registers (CSIL)**

The CSIL register is an 8-bit register that specifies the active voltage level of the Chip Select pins and the Queued CSI data length.
This register can be overwritten only while CTXE = 0 and CRXE = 0 (CSIM register).
Write operation during CTXE = 1 or CRXE = 1 is prohibited.

Transfer data length is 8bits and chip select signal is active low.
CSIL = 0x08H

**(c) Queued CSI Operation Mode Registers (CSIM)**

The CSIM registers control the Queued CSI macro's operations.
TRMD, DIR, CSIT, CSWE, CSMD bits can only be written when CTXE = 0 and CRXE = 0.

- Provide macro operation clock
  Power = 1
- interrupt delay mode select (INTC3nI signal)
  CSIT = 1; Half clock delay.
- Transmission wait disable. Not insert 1 clock ($\overline{SCK3}$) wait at transmission start.
  CSWE = 0.
- Chip select inactive level output disable. Do not force chip select inactive state after each transfer of a data element.
  CSMD = 0
- Single buffer transfer mode
  TRMD = 0
- Data is sent/received with MSB first
  DIR = 0
- Enable transmit and receive mode
  CTXE, CRXE = 1, 1

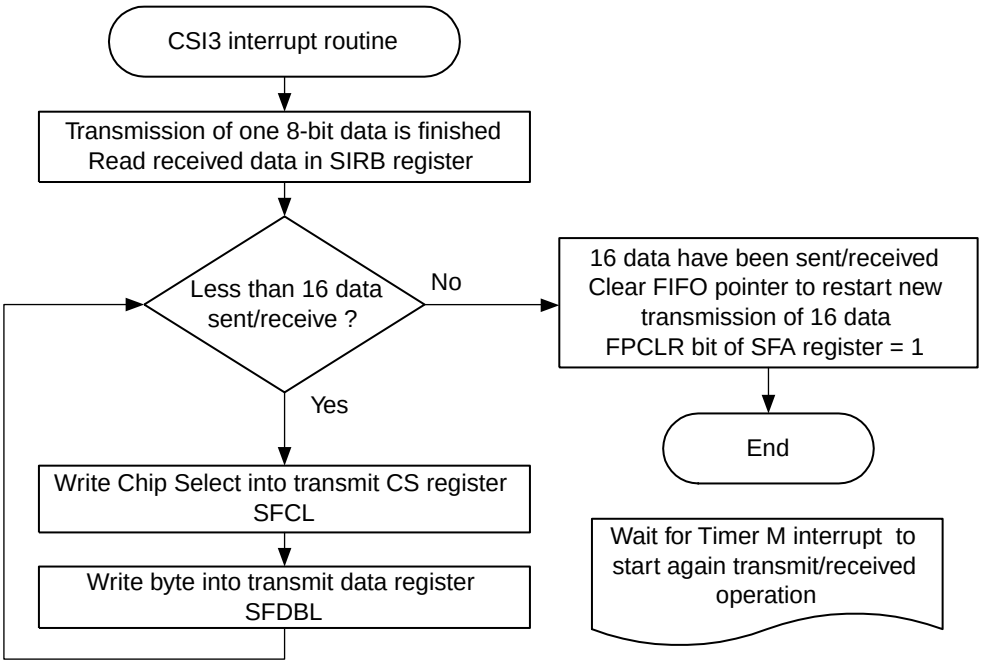
**(d) FIFO Buffer Status Register (SFA)**

The SFA register is an 8-bit register that shows the FIFO buffer status. The SFA register is read/write-enabled, accessible in 1-bit or 8-bit units. However, the bits SFFUL, SFEMP and CSOT are read only. Initial value is 20H by reset.
SFA read is prohibited when POWER = 1 and $f_{QCSI}$ is stopped.

- Clear all FIFO pointers
  FPCLR = 1
- FIFO buffer status flag
  SFFUL = 0: FIFO buffer is not full
  SFFUL = 1 FIFO buffer is full

**(e) CSI30 interrupt (INTC30I) flowchart**

***Figure 3-5: CSI3 Interrupt Routine***



### (5) Port mode initialisation

***Table 3-1: Non-Port Pins***

| Pin Name | I/O | Function | Alternate Function |
|---|---|---|---|
| SI30 | Input | Serial receive data input (CSI30) | P54 |
| SI31 | | Serial receive data input (CSI31) | P00 |
| SO30 | Output | Serial transmit data output (CSI30) | P55 |
| SO31 | | Serial transmit data output (CSI31) | P01 |
| $\overline{SCK30}$ | I/O | Serial clock I/O (CSI30) | P90/TIQ11/TOQ11, P95/RXDA1 |
| $\overline{SCK31}$ | | Serial clock I/O (CSI31) | P02 |
| CS300 | Output | Serial chip select output (CSI30) | P91/TIQ12/TOQ12, P96/TXDA1 |
| CS301 | | Serial chip select output (CSI30) | P92/TIQ13/TOQ13 |
| CS302 | | Serial chip select output (CSI30) | P93/TIQ10/TOQ10 |
| CS303 | | Serial chip select output (CSI30) | P94/$\overline{ASCKA1}$ |
| CS310 | | Serial chip select output (CSI31) | P03, P32 |
| CS311 | | Serial chip select output (CSI31) | P04 |
| CS312 | | Serial chip select output (CSI31) | P05/INTP2, P37/TIP00/TOP00/ INTP1 |
| CS313 | | Serial chip select output (CSI31) | P06/INTP3 |

#### (a) Port 0 initialisation

CSI31 is not used. All I/O could be set in output mode.
P0 = 0x00H; Reset output latches.
PM0 = 0x00;
PMC0 = 0x00H;
PFC0 = 0x00H;

#### (b) Port 5 initialisation

P5 = 0x00H; Reset output latches.
PM5 = 0xDFH; P54(SI30) in input mode, P55 (SO30) in output mode.
PMC5 = 0x30H; Select P55 and P54 respectively as serial output and serial input for CSI30.
PFC5 = 0x00; Default value after RESET.

#### (c) Port 9 initialisation

P9L = 0x00H; Reset output latches.
PM9L = 0x80H; P90 to P96 in output mode.
PMC9L = 0x70H; Specify CSI30 mode for the port.
PFC9L = 0x00H; Specify CSI30 function for the port, P96 = CS300, P95 = $\overline{SCK30}$.
PFCE9L = 0x00H; P90 = $\overline{SCK30}$, P91 = CS300, P92 = CS301, P93 = CS302, P94 = CS303.

P9H = 0x00H; Reset output latches.
PM9H = 0xCBH; Port mode register for P98 to P915.
PMC9H = 0x30H; I/O port mode.
PFC9H = 0x20H;
PFCE9H = 0x20H;

### (d) Port Function Swap Control Register (PSWAP)

This device has a special purpose register for swapping port function by setting control bit. The Swap function is implemented to allow flexible configuration.
PSWAP = 0x02H.

- P95 = $\overline{SCK30}$
- P96 = CS300
- P910 = CS301
- P912 = CS302

## (6) Measurement results

*Figure 3-6: Single Buffer Transfer Mode: Setup and Hold Time*

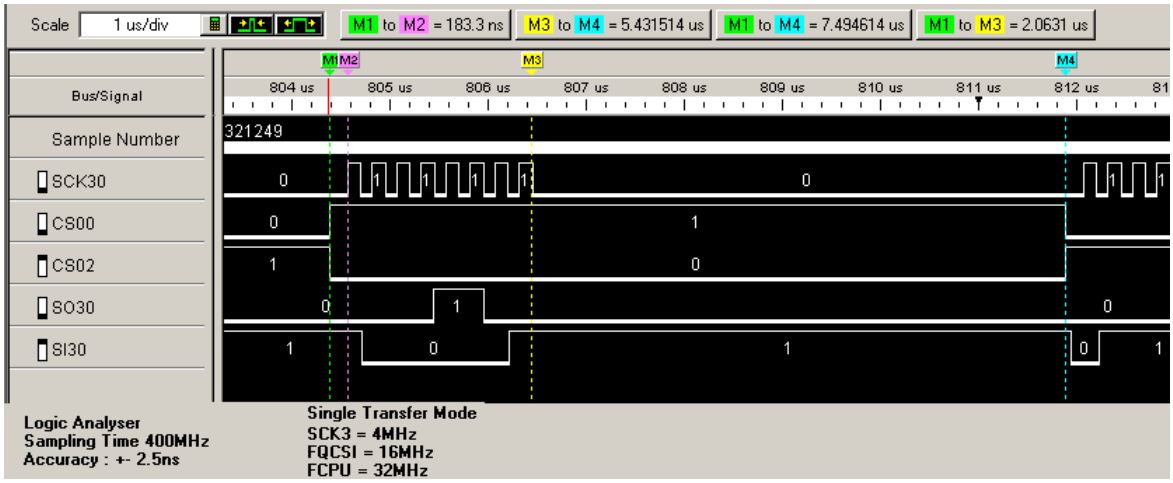
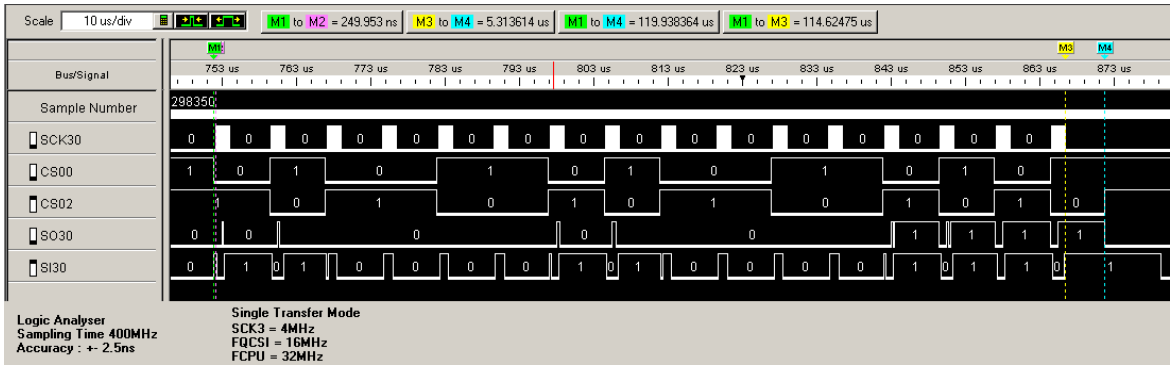


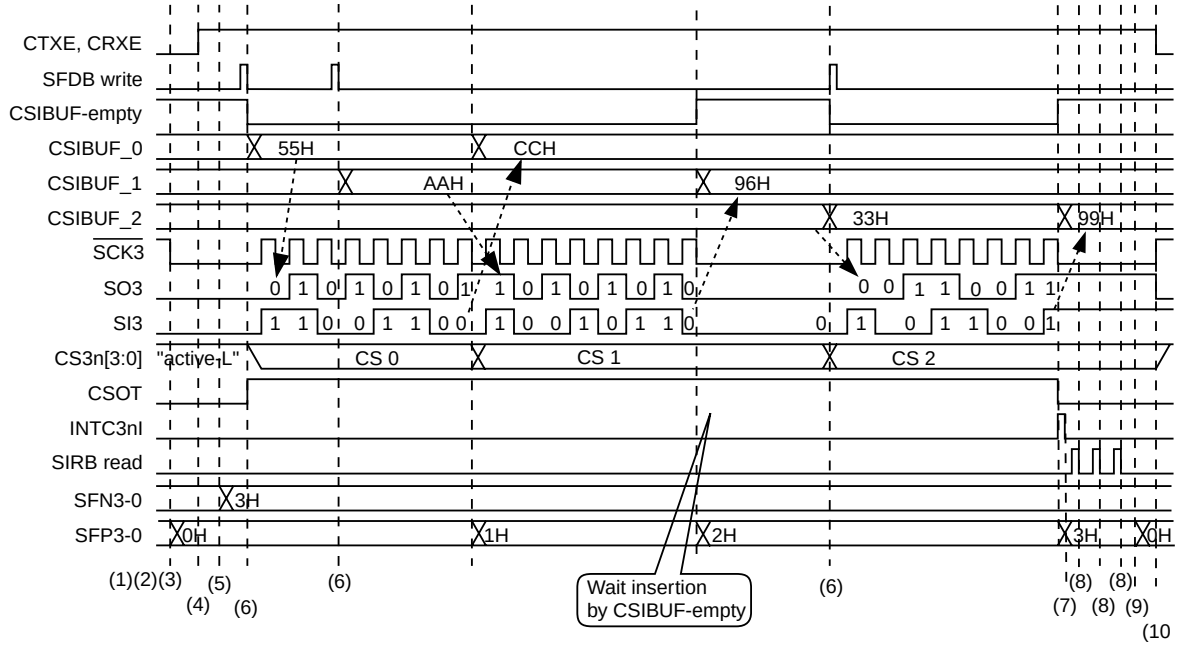

*Figure 3-7: Single Buffer Transfer Mode Timing*



## 3.2 FIFO Buffer Transfer Mode (Master Mode, Transmit/Receive Mode)

MSB first (DIR = 0), no INTC3nI delay (CSIT = 0), transmission wait disabled (CSWE = 0), CS inactive disabled (CSMD = 0), CKP = 1, DAP = 0, transmission data length of 8 bits (CCL[3:0] = [1,0,0,0]), active levels of all chip selects set to “active low”:

***Figure 3-8: FIFO Buffer Transfer Mode (Master, Transmit/Receive) Timing***




1. Set the CSIM register's POWER bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CSIC and CSIL registers to specify the transfer mode.
3. Write “1” in the SFA register's FPCLR bit to clear all FIFO pointers.
4. Specify the transfer mode using the CSIM register's TRMD, DIR, and CSIT bits; at the same time, set the CTXE and CRXE bits to 1 to enable the transmit/receive operation.
5. Set the number of transmit/receive data items in the SFN register's SFN[3:0] bits.
6. Make sure that the SFA register's SFFUL bit is set to 0, then write chip-select data and transmission data in the SFCS and SFDB registers in this order.
7. Wait for the transmissions/receptions to be completed (e.g. by monitoring the INT3C3nI interrupt).
8. Read the received data by multiple read of the SIRB register (= sequential read from the FIFO).
9. Write “1” in the SFA register's FPCLR bit and clear all FIFO pointers for the next transmission.

   To continue transmission/reception, repeat steps (5) - (9).

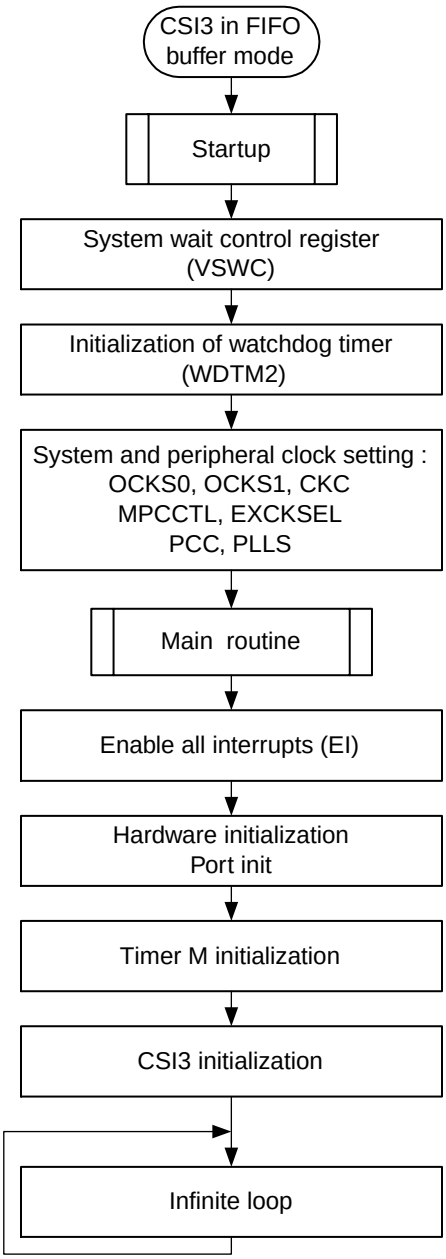
10. Set the CSIM register's CTXE and CRXE bits to 0 to disable the transmit/receive operation (end of transmit/receive operation).

### 3.2.1 FIFO buffer transfer mode application without DMA transfer

The FIFO buffer is used to transmit and receive the data.
The number of data to be transmitted is selected in advance using SFN register (SFN register is only used is FIFO buffer mode).
Transmission is completed when transmit/receive transfer of 16 data (8 bits length) has been sent/ received.
Transmission completed status is detected by monitoring the INTCSI3n interrupt.
The reading of the 16 received data is done every interrupt INTCSI3n
DMA transfer is not used.

#### (1) FIFO buffer transfer flowchart

***Figure 3-9: FIFO Buffer Transfer Flowchart***



**(2) System clock and peripheral clock setting (Startup)**

This routine sets the clock using the PLLs mode for the CPU, CSI3 and Timer M.
PLL0 and PLL1 output clocks are respectively assigned to the system clock and CSI3.
System clock is used for CPU and timer.
The main resonator oscillates at 8 MHz.

Be sure to set the following registers first when using the V850E/RS1.

- System wait control register (VSWC)
- On-chip Debug Mode control register (OCDM)**Note**
- Watchdog timer mode register 2 (WDTM2)**Note**

**Note:** Refer to 3.4.7 “Caution” in V850E/RS1 user’s manual.

**(a) NEC peripheral bus wait control**

The system wait control register (VSWC) controls wait of bus access to the internal peripheral I/O registers.
Set the value to VSWC register in accordance with the frequency used.
The VSWC register can be read or written in 8-bit units.
In this application the value written to VSWC is 0x12H.
VSWC = 0x12H;

**(b) Clock selection register for PLL0 and PLL1 (OCKS0, OCKS1)**

Those 8-bit registers control the operation enable and clock input selection for PLL0 and PLL1.

| | |
|---|---|
| Clock selection register 0: OCKS0 = 0x11H | PLL0 input = $f_{XX}/3$ |
| Clock selection register 1: OCKS1 = 0x13H | PLL1 input = $f_{XX}/5$ |

**(c) Clock control register (CKC)**

This is an 8-bit register that controls system clock ($f_{XX}$) when it operates on PLL mode (SELPLL bit of MPCCTL register is set to 1). This register can be read or written in 8-bit or 1-bit units.
Be sure to clear bits 2 to 7 of the CKC register to 0. In other case the operation is not guaranteed.
Data can be written to this register only in combination of specific sequences (refer to 3.2.3 “Special registers” in V850E/RS1 user’s manual).

| | |
|---|---|
| CKC = 0x03H | $f_{PLL\_MCKSEL\_CKDIV} = f_{PLL}$(PLL internal clock) |

**(d) Main peripheral clock control register (MPCCTL)**

This is an 8-bit register that selects the internal clock. Data can be written to this register only in combination of specific sequences (refer to 3.2.3 “Special registers” in V850E/RS1 user’s manual). The clock generator flexibility allows to use either simultaneously both PLL or to use only one PLL. Both PLLs can be assigned respectively to the CPU or to the peripherals.

Switch from oscillator clock to PLL clock:

| | |
|---|---|
| MPCCTL = 0x84H | PLL mode ($f_{XX} = f_{PLL\_MCKSEL\_CKDIV}$)<br>Main clock ($f_{PLL\_MCKSEL}$) = PLL0<br>Peripheral clock = PLL1 |

**(e) Extended clock selection (EXCKSEL)**

CSI3, CSIB, and AFCAN peripherals input clocks could be either from the system clock or directly form the PLL output using the extended clock selection.
In this application CSI3 uses the extended clock in order to get a 16 MHz input clock ($f_{QCSI}$) while the CPU clock is 32 MHz.

EXCKSEL = 0x0CH

CSI30 Clock = $f_{PLL\_PCKSEL}$ = 16 MHz
CSI31 Clock = $f_{PLL\_PCKSEL}$= 16 MHz

**(f) Processor clock control register (PCC)**

This 8-bit register controls the CPU clock selection using a prescaler.
Data can be written to this register only in combination of specific sequences (refer to 3.2.3 ”Special registers” in V850E/RS1 user’s manual).
PCC = 0x00H

$f_{CPU} = f_{XX}$ = 32 MHz

**(3) Timer M configuration in interval time mode**

Timer M (TMM0) supports only a clear & start mode. It does not support a free-running mode.
In the interval timer mode, a match interrupt signal (INTTM0EQ0) is output when the value of the 16-bit counter matches the value of TMM0 compare register 0 (TM0CMP0). At the same time, the counter is cleared to 0000H and starts counting up.

**(a) TMM0 timer control register (TM0CTL0)**

This register sets the count clocks of 16-bit timer.
In this application the count clock is set to $f_{XX}$.
TM0CTL0 = 0x80H

**(b) Timer M compare register**

In this application a timer interval of 300µs is needed as time base. The value of compare register can be calculated by the following formula:

$$\text{Interval Time} = \frac{\text{Compare register}}{\text{Internal count clock}}$$

$$300\ \mu s = \frac{\text{Compare Register TM0CMP0}}{32MHz}$$

TM0CMP0 = 0x2580H

**(c) Timer M interrupt mask register is enabled with highest priority**

TM0EQIC0 register or IMR1 register could be used to setup the enable interrupt mode.
In this application TM0EQIC0 is set:

TM0EQIC0 = 0x00H

**(d) Flowchart**

***Figure 3-10: Timer M Initialization Routine***

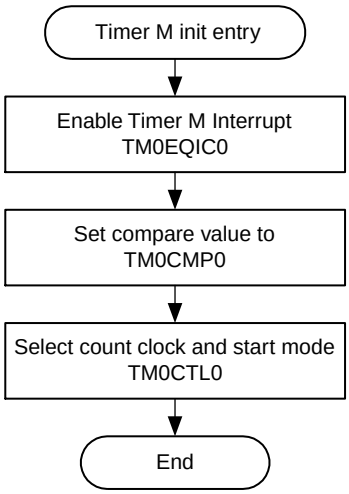
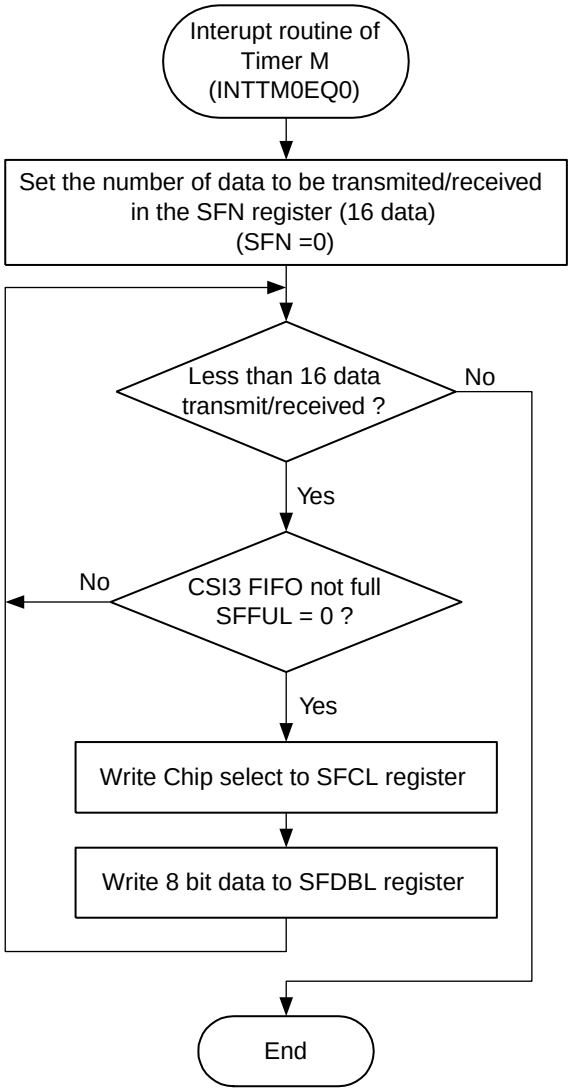



***Figure 3-11: Timer M Interrupt Routine***



**(4) CSIC3 configuration in FIFO buffer mode**

3-wire serial I/O mode interface

**(a) Queued CSI clock selection registers (CSIC)**

The CSIC register is an 8-bit register that is used to control the serial transfer operations (data phase, clock phase, master mode and clock selection).

**Caution: This register can be written only while CSIM register's CTXE = 0 and CRXE = 0.**

CSIC = 0x39H

| | |
|---|---|
| Master mode, $\overline{SCK3}$ = 4 MHz | CKS2, CKS1, CKS0 = 0, 0, 1<br>MDL2, MDL1, MDL0 = 0, 0, 1 |
| Sampling on first rising edge and idle level is low | CKP, DAP = 1, 1 |

**(b) Queued CSI data length selection registers (CSIL)**

The CSIL register is an 8-bit register that specifies the active voltage level of the Chip Select pins and the Queued CSI data length.
This register can be overwritten only while CTXE = 0 and CRXE = 0 (CSIM register).
Write operation during CTXE = 1 or CRXE = 1 is prohibited.

Transfer data length is 8bits and chip select signal is active low.
CSIL = 0x08H

**(c) Queued CSI operation mode registers (CSIM)**

The CSIM registers control the Queued CSI macro's operations.
TRMD, DIR, CSIT, CSWE, CSMD bits can only be written when CTXE = 0 and CRXE = 0.

- Provide macro operation clock
  Power = 1
- interrupt delay mode select (INTC3nI signal)
  CSIT = 1; Half clock delay.
- Transmission wait disable. Not insert 1 clock ($\overline{SCK3}$) wait at transmission start.
  CSWE = 0.
- Chip select inactive level output disable. Do not force chip select inactive state after each transfer of a data element.
  CSMD = 0
- FIFO buffer transfer mode
  TRMD = 1
- Data is sent/received with MSB first
  DIR = 0
- Enable transmit and receive mode
  CTXE, CRXE = 1, 1

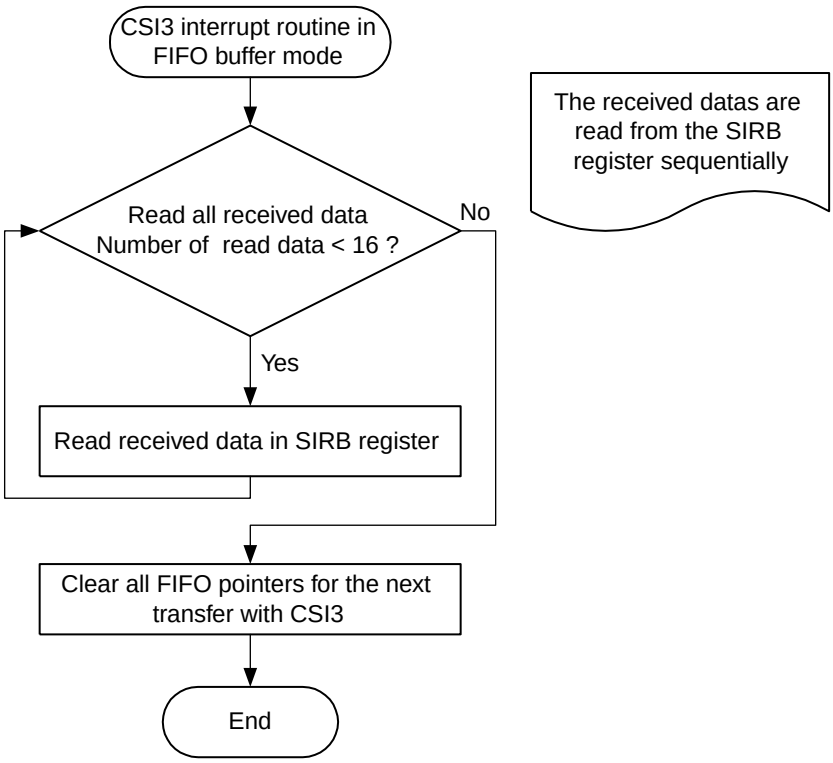
**(d) FIFO buffer status register (SFA)**

The SFA register is an 8-bit register that shows the FIFO buffer status. The SFA register is read/write-enabled, accessible in 1-bit or 8-bit units. However, the bits SFFUL, SFEMP and CSOT are read only. Initial value is 20H by reset.
SFA read is prohibited when POWER = 1 and $f_{QCSI}$ is stopped.

- Clear all FIFO pointers
  FPCLR = 1
- FIFO buffer status flag
  SFFUL = 0: FIFO buffer is not full
  SFFUL = 1 FIFO buffer is full

**(e) CSI30 interrupt (INTC30I) flowchart**

***Figure 3-12: CSI3 Interrupt Routine***



### (5) Port mode initialisation

***Table 3-2: Non-Port Pins***

| Pin Name | I/O | Function | Alternate Function |
|---|---|---|---|
| SI30 | Input | Serial receive data input (CSI30) | P54 |
| SI31 | | Serial receive data input (CSI31) | P00 |
| SO30 | Output | Serial transmit data output (CSI30) | P55 |
| SO31 | | Serial transmit data output (CSI31) | P01 |
| $\overline{SCK30}$ | I/O | Serial clock I/O (CSI30) | P90/TIQ11/TOQ11, P95/RXDA1 |
| $\overline{SCK31}$ | | Serial clock I/O (CSI31) | P02 |
| CS300 | Output | Serial chip select output (CSI30) | P91/TIQ12/TOQ12, P96/TXDA1 |
| CS301 | | Serial chip select output (CSI30) | P92/TIQ13/TOQ13 |
| CS302 | | Serial chip select output (CSI30) | P93/TIQ10/TOQ10 |
| CS303 | | Serial chip select output (CSI30) | P94/$\overline{ASCKA1}$ |
| CS310 | | Serial chip select output (CSI31) | P03, P32 |
| CS311 | | Serial chip select output (CSI31) | P04 |
| CS312 | | Serial chip select output (CSI31) | P05/INTP2, P37/TIP00/TOP00/ INTP1 |
| CS313 | | Serial chip select output (CSI31) | P06/INTP3 |

#### (a) Port 0 initialisation

CSI31 is not used. All I/O could be set in output mode.
P0 = 0x00H; Reset output latches.
PM0 = 0x00;
PMC0 = 0x00H;
PFC0 = 0x00H;

#### (b) Port 5 initialisation

P5 = 0x00H; Reset output latches.
PM5 = 0xDFH; P54(SI30) in input mode, P55 (SO30) in output mode.
PMC5 = 0x30H; Select P55 and P54 respectively as serial output and serial input for CSI30.
PFC5 = 0x00; Default value after RESET.

#### (c) Port 9 initialisation

P9L = 0x00H; Reset output latches.
PM9L = 0x80H; P90 to P96 in output mode.
PMC9L = 0x70H; Specify CSI30 mode for the port.
PFC9L = 0x00H; Specify CSI30 function for the port, P96 = CS300, P95 = $\overline{SCK30}$.
PFCE9L = 0x00H; P90 = $\overline{SCK30}$, P91 = CS300, P92 = CS301, P93 = CS302, P94 = CS303.

P9H = 0x00H; Reset output latches.
PM9H = 0xCBH; Port mode register for P98 to P915.
PMC9H = 0x30H; I/O port mode.
PFC9H = 0x20H;
PFCE9H = 0x20H;

**(d) Port function swap control register (PSWAP)**

This device has a special purpose register for swapping port function by setting control bit. The Swap function is implemented to allow flexible configuration.
PSWAP = 0x02H.

- P95 = $\overline{\text{SCK30}}$
- P96 = CS300
- P910 = CS301
- P912 = CS302

## (6) Measurement results

*Figure 3-13: FIFO Buffer Transfer Mode: Setup and Hold Time*

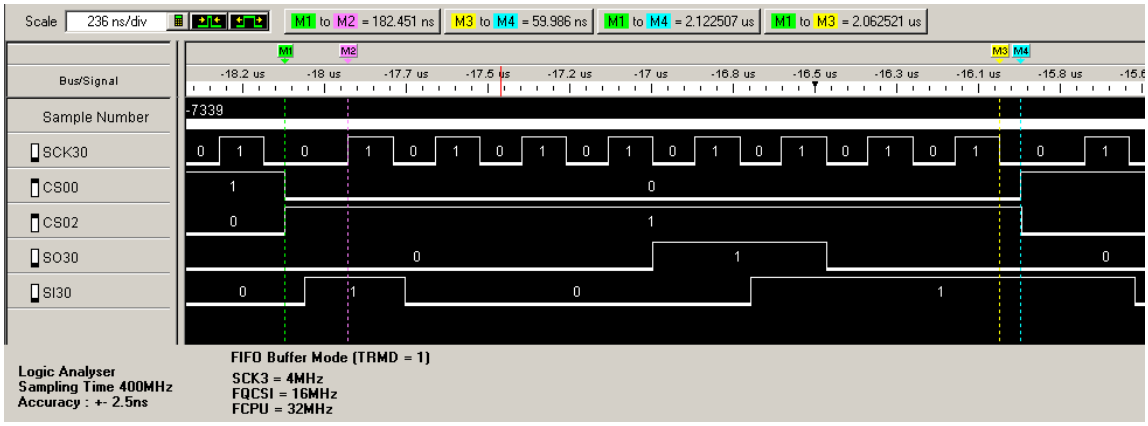
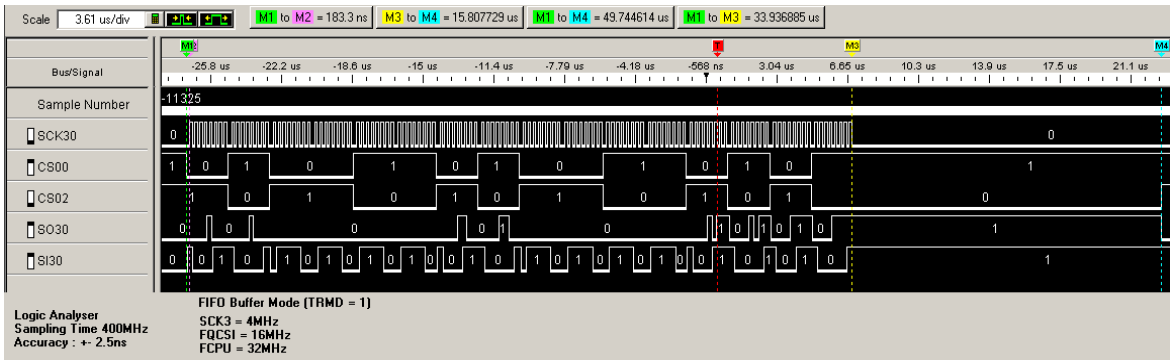



*Figure 3-14: FIFO Buffer Transfer Mode Timing*



## 3.3 Single Buffer Transfer Mode with DMA

This chapter describe the implementation of two DMA channels to transmit and to send the data in single buffer transfer mode.

For CSI3n configuration, please refer to chapter **3.1 "Single Buffer Transfer Mode (Master Mode, Transmit/Receive Mode)" on page 42**.
For system clock and peripheral clock setting, please refer to **"System clock and peripheral clock setting (Startup)" on page 44**
For DMA controller registers detail explanations, please refer to **V850E/RS1 User's manual (U16702EE1V0UD00), chapter 15 "DMA Functions"**.

### (1) DMA transfer mode

Single transfer mode is used for both DMA channels.

In Single transfer mode, the DMA releases the bus after each transfer. If there is a subsequent DMA transfer request, the transfer is performed again when the bus becomes available again. DMBCn triggers are required to transfer DMBCn elements. This operation continues until the transfer counter is cleared to 0 and the internal TC signal is generated.
When the DMA has released the bus, if another higher priority DMA transfer request is issued, the higher priority DMA request always takes precedence.

#### (a) DMA source/destination addresses, priority, and transfer size

DMA channel 1 and 2 are respectively used for receive operation (readout SIRBn register) and for transmit operation (write data into SFCSn and SFDBn registers).
DMA channel 1 transfer size is one byte (8-bit). DMA channel 2 transfer size is one word (32-bit).
According to CSI3n architecture, the receive register SIRBn must be readout before starting any following transfer. Otherwise the received data will be lost and the CSI3n will not operate correctly.
It is important to take care of the priority of each DMA channel to be used, giving the highest priority to the receive function.
The chip select FIFO buffer register (SFCSn) and the transmit data FIFO buffer register (SFDBn) are mapped in a continuous address and are 16bits long. The data to be sent is written to SFCSn register with 32bits access. Upper 16 bits are written to data register (SFDBn) and the lower 16 bits are written to chip select register (SFCSn).
DMA channel priority is fixed as follow:
DMA channel 0 > DMA channel 1 > DMA channel 2 > DMA channel 3 > DMA channel 4 > DMA channel 5.

#### (b) Interrupt source for DMA trigger

Each DMA could be triggered either by software or peripheral interruptions.
This program provides an example of the DMA and CSI3 operation using the end transmission/reception end interrupt signal INTC3nI to trig both DMA channels. So the first transmission/reception must be started by software every 300 µs.
CSI3 end transmission/reception end interrupt signal INTC3nI is only use to trig the DMA, so the interrupt routine is useless.

**(c) End of transmission/reception, Hold time setting for the last transmission/reception**

DMA channel 1 interrupt (INTDMA1) routine is used to clear all FIFO pointers. This operation turns the chip select signal to inactive level after the last 16th element has been successfully transmitted/received.
This operation could also be done later into the Timer M interrupt routine. This will provide a longer hold time signal and will save CPU load avoiding an additional interrupt handling procedure.

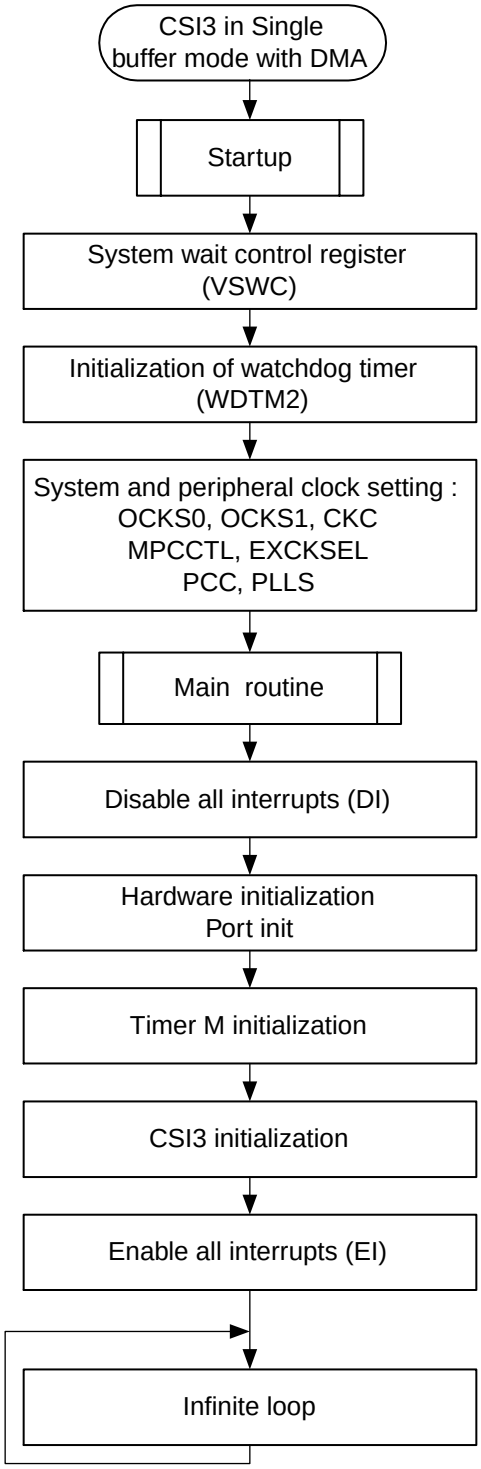
**(d) DMA initialization and CSI3 transfer start**

DMA initialization has to be done for each new transfer. Timer M interrupt routine provide such availability every 300 µs.
To initiate the transmission of CSI3 interface, chip select and data are respectively written to SFCS0 and SFDB0 register during the Timer M interrupt routine.

**Note:** n = 0, 1

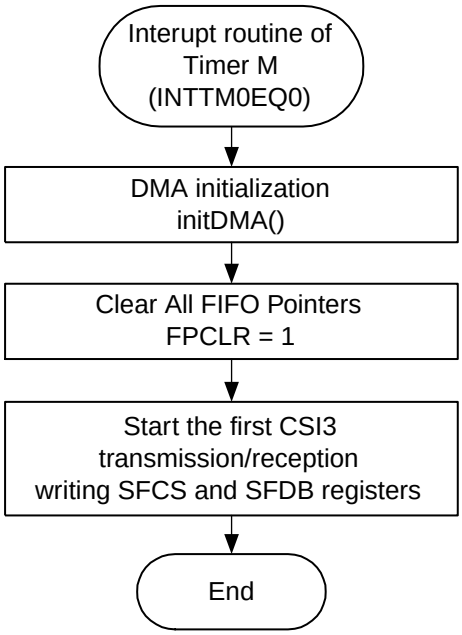
**(2) Single buffer transfer flowchart**

***Figure 3-15: Single Buffer Transfer Flowchart with DMA***



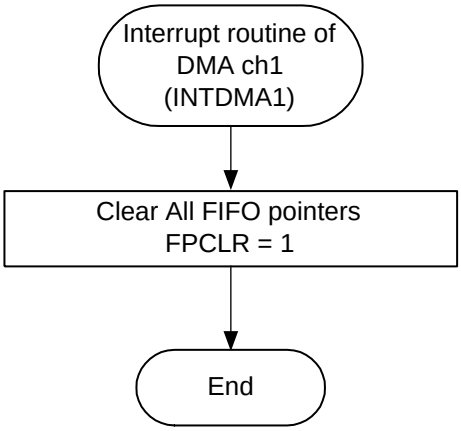
### (3) Timer M interrupt routine

*Figure 3-16: Timer M Interrupt Routine*




### (4) DMA channel 1 interrupt routine

*Figure 3-17: DMA channel 1 Interrupt Routine*



## (5) Measurements results

*Figure 3-18: Single Buffer Transfer Mode with DMA: Setup and Hold Time*

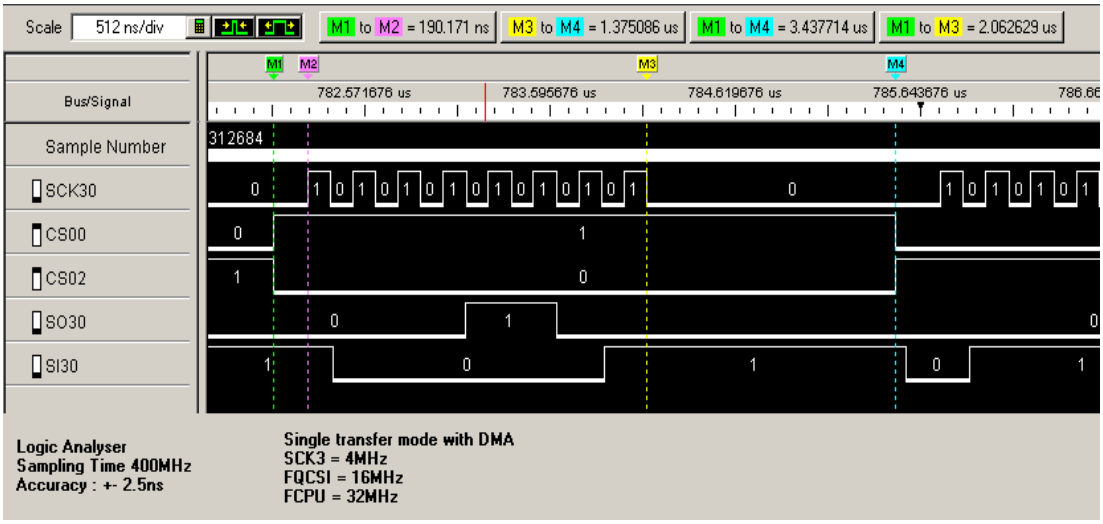
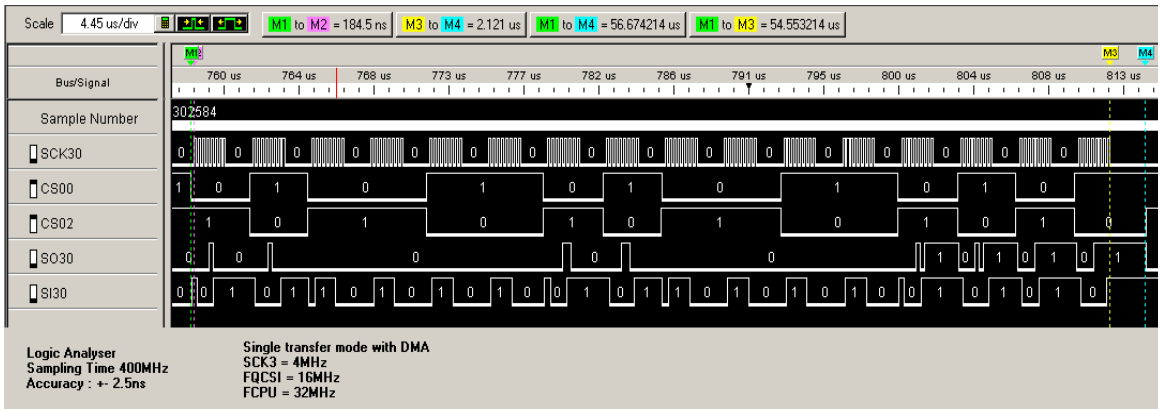



*Figure 3-19: Single Buffer Transfer Mode Timing with DMA*



## 3.4 FIFO Buffer Transfer Mode with DMA

This chapter describe the implementation of two DMA channels to transmit and to send the data in FIFO buffer transfer mode.

For CSI3n configuration, please refer to chapter **3.2 “FIFO Buffer Transfer Mode (Master Mode, Transmit/Receive Mode)” on page 51**.
For system clock and peripheral clock setting, please refer to **“System clock and peripheral clock setting (Startup)” on page 53**
For DMA controller registers detail explanations, please refer to **V850E/RS1 User’s manual (U16702EE1V0UD00), chapter 15 “DMA Functions”**.

### (1) DMA transfer mode

Fixed channel transfer mode is used for both DMA channels.

In “Fixed Channel” transfer mode, the DMA releases the bus after each transfer, but continues to repeat the DMA transfer until the transfer counter is cleared to 0 without requiring a new DMA transfer request.
Only one trigger is required to transfer DMBCn elements
When the DMA has released the bus, if another higher priority DMA transfer request is issued, the higher priority DMA request does not take precedence.

#### (a) DMA source/destination addresses, priority, and transfer size

DMA channel 1 and 2 are respectively used for receive operation (readout SIRBn register) and for transmit operation (write data into SFCSn and SFDbn registers).
DMA channel 1 transfer size is one byte (8-bit). DMA channel 2 transfer size is one word (32-bit).
According to CSI3n architecture, the receive register SIRBn must be readout before starting any following transfer. Otherwise the received data will be lost and the CSI3n will not operate correctly.
It is important to take care of the priority of each DMA channel to be used, giving the highest priority to the receive function.
The chip select FIFO buffer register (SFCSn) and the transmit data FIFO buffer register (SFDBn) are mapped in a continuous address and are 16bits long. The data to be sent is written to SFCSn register with 32bits access. Upper 16 bits are written to data register (SFDBn) and the lower 16 bits are written to chip select register (SFCSn).
DMA channel priority is fixed as follow:
DMA channel 0 > DMA channel 1 > DMA channel 2 > DMA channel 3 > DMA channel 4 > DMA channel 5.

#### (b) Interrupt source for DMA trigger

Each DMA could be triggered either by software or peripheral interruptions.
This program provides an example of the DMA and CSI3 operation using the end transmission/reception end interrupt signal INTC3nI to trig DMA channel 1 and a software trigger for DMA channel 2. So the first transmission/reception is started by software every 300 µs into the Timer M interrupt routine.
CSI3 end transmission/reception end interrupt signal INTC3nI is only use to trig the DMA channel 1, so the interrupt routine is useless.

**(c) End of transmission/reception, Hold time setting for the last transmission/reception**

DMA channel 1 interrupt (INTDMA1) routine is used to clear all FIFO pointers. This operation turns the chip select signal to inactive level after the last 16th element has been successfully transmitted/received.
This operation could also be done later into the Timer M interrupt routine. This will provide a longer hold time signal and will save CPU load avoiding an additional interrupt handling procedure.

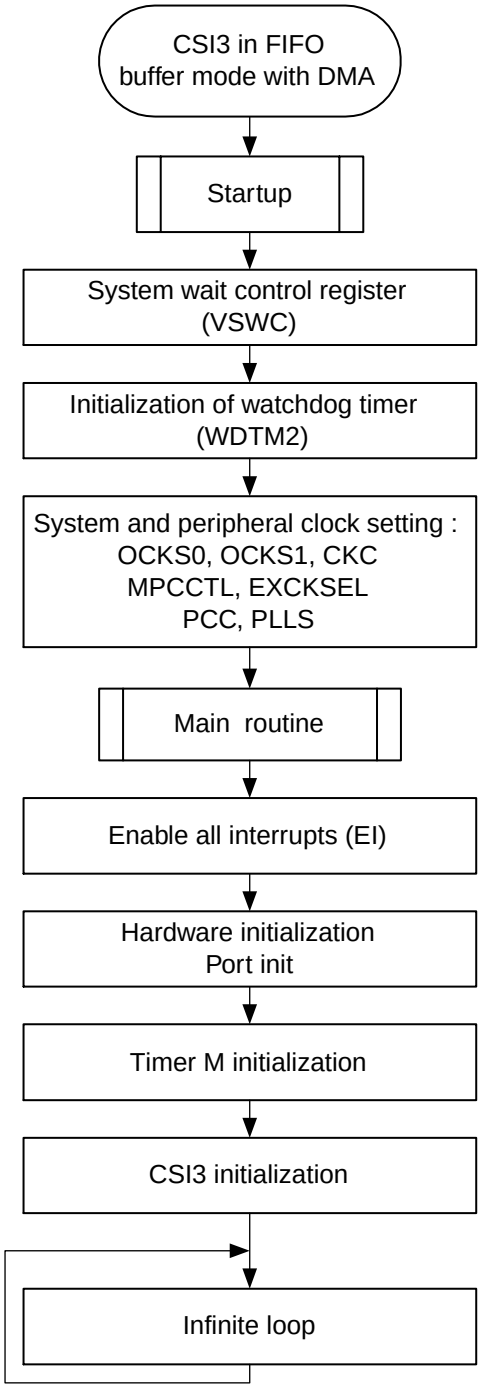
**(d) DMA initialization and CSI3 transfer start**

DMA initialization has to be done for each new transfer. Timer M interrupt routine provide such availability every 300 µs.
To initiate the first transmission of CSI3 interface, DMA channel 2 is triggered by software in the timer M interrupt routine.

**Note:** n = 0, 1

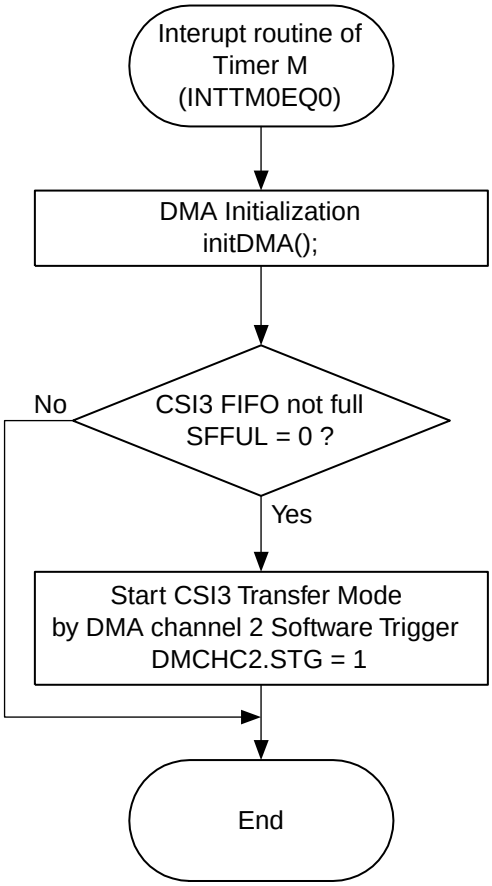
**(2) FIFO buffer transfer flowchart**

***Figure 3-20: FIFO Buffer Transfer Flowchart with DMA***



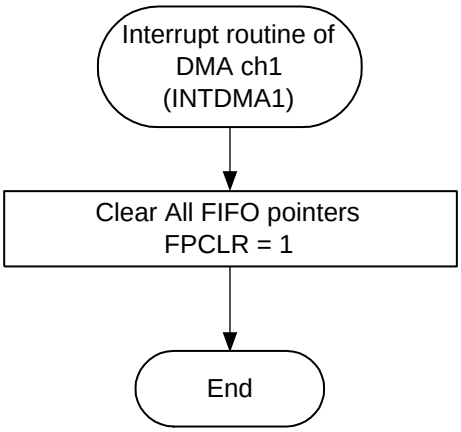
### (3) Timer M interrupt routine

*Figure 3-21: Timer M Interrupt Routine*




### (4) DMA channel 1 interrupt routine

*Figure 3-22: DMA Channel 1 Interrupt Routine*



## (5) Measurement results

*Figure 3-23: FIFO Buffer Transfer Mode with DMA: Setup and Hold Time*

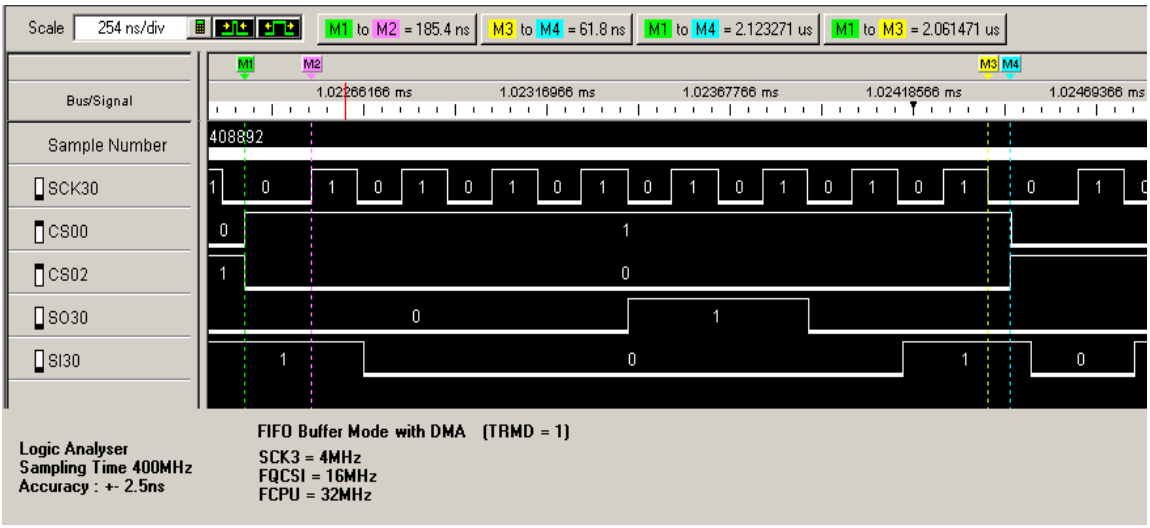
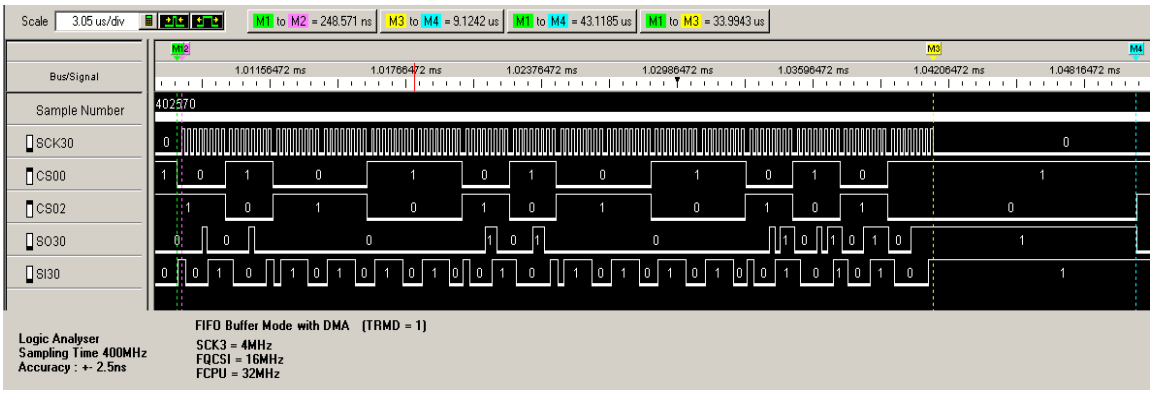



*Figure 3-24: FIFO Buffer Transfer Mode Timing with DMA*



## 3.5 Sample Programs

The 4 software examples projects are defined with below mentioned files:

- Startup file: Startup_df3402.850
- Header files: df3402.h, df3402extIO.h, df3402.s
- Source file: Main.c

**Note:** The header files df3402.h, df3402extIO.h and df3402.s are standard files provided by NEC. Please contact your NEC Electronics sales office representative to obtain it.

### 3.5.1 Single buffer mode without DMA

```
//=============================================================================
// PROJECT: V850E/RS1 CSI3 driver in single buffer mode without DMA
// MODULE : Main.c
// VERSION: 1.0
// DATE   : 22/06/05
//=============================================================================
//                           C O P Y R I G H T
//=============================================================================
// Copyright (c) 2002 by NEC Electronics (Europe) GmbH, Succursale Française.
// All rights reserved.
// 9, rue Paul Dautier
// 78142 Vélizy Cedex
// France
//=============================================================================
//
// Warranty Disclaimer:
// Because the Product(s) is licensed free of charge, there is no warranty of
// any kind whatsoever and expressly disclaimed and excluded by NEC, either
// expressed or implied, including but not limited to those for non-
// infringement of intellectual property, merchantability and/or fitness for
// the particular purpose. NEC shall not have any obligation to maintain,
// service or provide bug fixes for the supplied Product(s) and/or the
// Application.
//
// Each User is solely responsible for determining the appropriateness of
// using the Product(s) and assumes all risks associated with its exercise
// of rights under this Agreement, including, but not limited to the risks
// and costs of program errors, compliance with applicable laws, damage to
// or loss of data, programs or equipment, and unavailability or
// interruption of operations.
//
// Limitation of Liability:
// In no event shall NEC be liable to the User for any incidental,
// consequential, indirect, or punitive damage (including but not limited to
// lost profits) regardless of whether such liability is based on breach of
// contract, tort, strict liability, breach of warranties, failure of
// essential purpose or otherwise and even if advised of the possibility of
// such damages. NEC shall not be liable for any services or products provided
// by third party vendors, developers or consultants identified or referred
// to the User by NEC in connection with the Product(s) and/or the
// Application.
```

```
//
//=====================================================================================
// Environment: Devices   : V850E/RS1
//              Assembler : GHS AS-V850                    Version: 4.0.5
//              C-Compiler: GHS CCV850, Multi 2000, V800   Version: 4.0.5
//              Linker    : GHS lx, Multi 2000, V800       Version: 4.0.5
//              Debugger  : GHS Multi 2000, V800           Version: 4.0.5
//=====================================================================================
// Version    Author             Description
// 1.0        Ferry-Chappuis     Released
//=====================================================================================

/*************************************************************************
This software is used to develop the module for the CSI3 in single buffer mode
without DMA interface
*************************************************************************/

/*************************************************************************
Description of the CSI3 mode

CSI30 is using the single buffer transfer mode
in this mode, the CSI30 provides an interruption INTCSI30 each data transfer.
SFN register is not used.
*************************************************************************/


/*************************************************************************
 Includes
*************************************************************************/
#include “Inc\df3402.h”
#include “Inc\df3402extIO.h”


/*************************************************************************
** Declaration of local types **
*************************************************************************/
typedef unsigned char uchar;

/*************************************************************************
** Declaration of variables **
*************************************************************************/

int       mbytes = 0;
uchar     rxdata[16] ={1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
uchar     txdata[16] ={8,8,0,0,0,0,12,12,0,0,0,0,11,11,15,15};
uchar     csdata[16] ={0xE, 0XB, 0xE, 0xE, 0xB, 0xB, 0xE, 0XB, 0xE, 0xE, 0xB,
                       0xB, 0xE, 0XB, 0xE, 0XB};


/*************************************************************************
** CSI3 registers initialisation for single buffer mode operation
*************************************************************************/

void InitCSI3Single8bits(void)
{

   /* Configure serial transfer operation */
   /* This register must be initialised in several times because TRMD, DIR, CSIT, CSWE
      and CSMD bits can be written only when CTXE and CRXE are 0 */
   _POWER0 = 1;

   /* CSIL and CSIC registers must be written first because it is possible only if */
   /* CTXE and CRXE bits in CSIM register are 0 */
```

```
   /* Configure clock: - Idle level low
                       - sampling edge done on the first edge
                       - prescaler associated to clock, K= 1 (PRSOUT = FQCSI/2), N= 1
                         operation clock = PRSOUT/2)*/
   /* Clock speed = 4MHz*/

   CSIC0 = 0x39;


   /* Configure buffer number to store data and data length
   8-bit data length
   chip select signal is active low */
   CSIL0 = 0x08;

   /* clear FIFO pointer */
   _FPCLR0 = 1;

   _CSMD0 = 0; /* CS_INACTIVE_DISABLE */
   _CSWE0 = 0; /* WAIT_DISABLE */
   _CSIT0 = 1; /* HALF_CLK_DELAY */
   _DIR0 = 0;  /* Transfer direction: MSB first */
   _TRMD0 = 0; /* CSI3 in Single buffer transfer mode */

   /* enable the transmission */
   _CRXE0 = 1; /* RX_ENABLE */
   _CTXE0 = 1; /* TX_ENABLE */

}


/*****************************************************************************
PORTS MANAGEMENT                                                             *
*****************************************************************************/

void InitPorts(void)
{
/* Initialise port 0 */
   P0     = 0x00;
   PM0    = 0x00;
   PMC0   = 0x00;
   PFC0   = 0x00;

/* Initialise port 5 */
   P5     = 0x00;
   PM5    = 0xDF;
   PMC5   = 0x30;
   PFC5   = 0x00;

/* Initialise port 9 */
/* LSB */
   P9L     = 0x00;
   PM9L    = 0x80;
   PMC9L   = 0x70;
   PFC9L   = 0x00;
   PFCE9L  = 0x00;

/* MSB */
   P9H     = 0x17;
   PM9H    = 0xCB;
   PMC9H   = 0x30;
   PFC9H   = 0x20;
   PFCE9H  = 0x20;

/* Initialise PSWAP */
```

```
   PSWAP = 0x02;
}


/***********************************************************************
Timer M initialisation for compare operation
Compare value to obtain an interruption every 300 µs:
300 µs / (1/32 MHz) = 9600 = 0x2580
***********************************************************************/
void InitTimerMCompareOperation(void)
{
               TM0CTL0 = 0x80;

/* compare value for generating an interrupt every 300 µs */
               TM0CMP0 = 0x2580;

/* interrupt enable and highest level */
               TM0EQIC0 = 0;
}


/***********************************************************************
InitialiseRegisters()
This function initialises I/O ports and other peripherals of the MCU
***********************************************************************/
void InitialiseRegisters(void)
{
/* Configure IO ports */
               InitPorts();

/* Configure TIMER M compare operation to generate an interrupt every 300 µs*/
               InitTimerMCompareOperation();

               InitCSI3Single8bits();

}


/***********************************************************************
Interruption function IntTimerM_Compare()
This interrupt function allow to start the CSI3 transfer every 300 µs
***********************************************************************/
void IntTimerM_Compare(void)
#pragma ghs interrupt
{
/* every 300 µs a new transmission/reception starts */
                mbytes = 0;

/* Clear FIFO buffer full status flag: SFFUL = 0 */
         _SFFUL0 = 0;

/* Start the first transmission every 300 µs by writing the data to SFDB register */
/* Write CS into transmit CS register */
               SFCS0L = csdata[mbytes];

/* Write byte into transmit data register */
               SFDB0L = txdata[mbytes];
}
```

```
/************************************************************************
Interruption function INTC30I() of the CSI3
************************************************************************/
void INTC30I(void)
#pragma ghs interrupt
{
/* when transmission is finished, read SIRB register */
    rxdata[mbytes] = SIRB0L;

/* send the next data up mbytes value < 16 */
    mbytes++;

    if (mbytes < 16)
    {
               /* Write CS into transmit CS register */
               SFCS0L = csdata[mbytes];

               /* Write byte into transmit data register */
               SFDB0L = txdata[mbytes];
  }

    else
    {
               /* clear all FIFO pointers to “0” */
               _FPCLR0 = 1;
  }

}

/************************************************************************
Main Function
************************************************************************/
void main(void)
{
/* enable INTC30I interrupt of CSI30 */
    C30IC = 0x00;

    EI();
/* Configuration for:
 - port
 - CSI3 */
   InitialiseRegisters();

   while(1){};
}
```

### 3.5.2 Single buffer mode with DMA

```
//=========================================================================
// PROJECT: V850E/RS1 CSI3 driver single buffer mode with DMA to send and received data
// MODULE : Main.c
// VERSION: 1.0
// DATE   : 22/06/05
//=========================================================================
//                          C O P Y R I G H T
//=========================================================================
// Copyright (c) 2002 by NEC Electronics (Europe) GmbH, Succursale Française.
// All rights reserved.
// 9, rue Paul Dautier
// 78142 Vélizy Cedex
// France
//=========================================================================
//
// Warranty Disclaimer:
// Because the Product(s) is licensed free of charge, there is no warranty of
// any kind whatsoever and expressly disclaimed and excluded by NEC, either
// expressed or implied, including but not limited to those for non-
// infringement of intellectual property, merchantability and/or fitness for
// the particular purpose. NEC shall not have any obligation to maintain,
// service or provide bug fixes for the supplied Product(s) and/or the
// Application.
//
// Each User is solely responsible for determining the appropriateness of
// using the Product(s) and assumes all risks associated with its exercise
// of rights under this Agreement, including, but not limited to the risks
// and costs of program errors, compliance with applicable laws, damage to
// or loss of data, programs or equipment, and unavailability or
// interruption of operations.
//
// Limitation of Liability:
// In no event shall NEC be liable to the User for any incidental,
// consequential, indirect, or punitive damage (including but not limited to
// lost profits) regardless of whether such liability is based on breach of
// contract, tort, strict liability, breach of warranties, failure of
// essential purpose or otherwise and even if advised of the possibility of
// such damages. NEC shall not be liable for any services or products provided
// by third party vendors, developers or consultants identified or referred
// to the User by NEC in connection with the Product(s) and/or the
// Application.
//
//=========================================================================
// Environment: Devices   : V850E/RS1
//              Assembler : GHS AS-V850                   Version: 4.0.5
//              C-Compiler: GHS CCV850, Multi 2000, V800  Version: 4.0.5
//              Linker    : GHS lx, Multi 2000, V800      Version: 4.0.5
//              Debugger  : GHS Multi 2000, V800          Version: 4.0.5
//=========================================================================
// Version    Author            Description
// 1.0        Ferry-Chappuis    Released
//=========================================================================
```

```
/*****************************************************************************
Description of CSI3 mode

Single buffer Mode
16 data of 1 byte to be transfer and received.
DMA channel 1 is used for receiving
DMA channel 2 is used for transmitting
*****************************************************************************/


/*****************************************************************************
 Includes
*****************************************************************************/
#include “Inc\df3402.h”
#include “Inc\df3402extIO.h”

/*****************************************************************************
** Declaration of local types **
*****************************************************************************/
typedef unsigned char uchar;
typedef unsigned long ulong;

/*****************************************************************************
** Declaration of variables **
*****************************************************************************/
#define        TX_DATA_FIFO 0xFD44;// define address of SFCS0 as destination address
for DMA channel 2
#define        RX_DATA_FIFO 0xFD42;// define address of SIRB0 as source address for
DMA channel 1

int       mbytes = 0;
uchar     rxdata[16] ={1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
uchar     txdata[16] ={8,8,0,0,0,0,12,12,0,0,0,0,11,11,15,15};
uchar     csdata[16] ={0xE, 0XB, 0xE, 0xE, 0xB, 0xB, 0xE, 0XB, 0xE, 0xE, 0xB,
                       0xB, 0xE, 0XB, 0xE, 0XB};

/* Data + chip select to be sent in 16 data transfer */
/* 32 bit send to SFDB and SFCL registers*/

ulong  Txdata32[16] = {0x0008000E, 0x0008000B, 0x0000000E, 0x0000000E,
                       0x0000000B, 0x0000000B, 0x000C000E, 0x000C000B, 0x0000000E,
                       0x0000000E, 0x0000000B, 0x0000000B, 0x000B000E,
                       0x000B000B,0x000F000E, 0x000F000B};
```

```
/*********************************************************************
** CSI3 registers initialisation for single buffer mode operation
*********************************************************************/

void InitCSI3Single8bits(void)
{

   /* Configure serial transfer operation */
   /* This register must be initialised in several times because TRMD, DIR, CSIT, CSWE
      and CSMD bits can be written only when CTXE and CRXE are 0 */
   _POWER0 = 1;

   /* CSIL and CSIC registers must be written first because it is possible only if */
   /* CTXE and CRXE bits in CSIM register are 0 */

   /* Configure clock: - Idle level low
                       - sampling edge done on the first edge
                       - prescaler associated to clock, K= 1 (PRSOUT = FQCSI/2), N= 1
                         operation clock = PRSOUT/2)*/
   /* Clock speed = 4 MHz*/

   CSIC0 = 0x39;


   /* Configure buffer number to store data and data length
   8-bit data length
   chip select signal is active low */
   CSIL0 = 0x08;

   /* clear FIFO pointer */
   _FPCLR0 = 1;

   _CSMD0 = 0; /* CS_INACTIVE_DISABLE */
   _CSWE0 = 0; /* WAIT_DISABLE */
   _CSIT0 = 1; /* HALF_CLK_DELAY */
   _DIR0 = 0;  /* Transfer direction: MSB first */
   _TRMD0 = 0; /* CSI3 in Single buffer transfer mode */

   /* enable the transmission */
   _CRXE0 = 1; /* RX_ENABLE */
   _CTXE0 = 1; /* TX_ENABLE */

}
```

```
/************************************************************************
PORTS MANAGEMENT                                                        *
************************************************************************/

void InitPorts(void)
{
/* Initialise port 0 */
   P0     = 0x00;
   PM0    = 0x00;
   PMC0   = 0x00;
   PFC0   = 0x00;

/* Initialise port 5 */
   P5     = 0x00;
   PM5    = 0xDF;
   PMC5   = 0x30;
   PFC5   = 0x00;

/* Initialise port 9 */
/* LSB */
   P9L     = 0x00;
   PM9L    = 0x80;
   PMC9L   = 0x70;
   PFC9L   = 0x00;
   PFCE9L  = 0x00;

/* MSB */
   P9H     = 0x17;
   PM9H    = 0xCB;
   PMC9H   = 0x30;
   PFC9H   = 0x20;
   PFCE9H  = 0x20;

/* Initialise PSWAP */
   PSWAP = 0x02;

}


/*********************************************************************
Timer M initialisation for compare operation
Compare value to obtain an interruption every 300 µs:
300 µs / (1/32 MHz) = 9600 = 0x2580
*********************************************************************/
void InitTimerMCompareOperation(void)
{
               TM0CTL0 = 0x80;

/* compare value for generating an interrupt every 300 µs */
               TM0CMP0 = 0x2580;

/* interrupt enable and highest level */
               TM0EQIC0 = 0;
}
```

```
/************************************************************************
InitialiseRegisters()
This function initialises I/O ports and other peripherals of the MCU
************************************************************************/
void InitialiseRegisters(void)
{
/* Configure IO ports */
               InitPorts();

/* Configure TIMER M compare operation to generate an interrupt every 300 µs*/
               InitTimerMCompareOperation();

               InitCSI3Single8bits();
}

/*---------------------------------------------------------------------*/
/* Initialisation of DMA channel */
/*---------------------------------------------------------------------*/

void InitDMA(void)
{
/* ----- Set DMA Registers ----- */
/* DMA channel 1 used for received data */
/* DMA channel 2 is used to transmit the data: 32 bit into SFCS and SFDB register */
/* The first transfer is started manually into the timer M routine */
/* Both DMA are triggered by the CSI3 interrupt INTC30I (end of transmission)*/
/*---------------------------------------------------------------------*/

/* DMA control register (DMC)  */
//The DMC register controls the operation and the clock supply of the DMA controller.
//It can be read or written in 8-bit or 1-bit units. Initial value is 00H by reset.

//DMC |STPDIS|IDMEN|STPSET| STPCLR| 0 |0 |0 |POWER


// clear the status flag. All DMA transfer are operational: STPDIS = 0
// NMI interrupts and stops all on-going DMA transfers: IDMEN = 1
// DMA transfer interrupted by software: STPSET = 1
// The clock supply to the DMA controller is started and the DMA operation enabled.
// POWER = 1
    DMC = 0x71;

// INTDMAn timing is based on DMA execution: TCS = 1 is recommended for RS1
// Clear any pending DMA transfer request: FCLR = 1
// DMA channel 1 is disabled EN = 0
    DMCHC1 = 0x04;
    DMCHC2 = 0x04;

    DMSA1H = 0x02FF;
    DMSA2H = 0x02FF;

/* source adress */
    DMSA1L = RX_DATA_FIFO; // source - point to SIRB0 address buffer.
   DMSA2L = 0xffff & (unsigned int)(&Txdata32[0x01]); // source - point to transmit buffer

    DMDA1H = 0x02FF;
    DMDA2H = 0x02FF;

/* destination adress */
    DMDA1L = 0xffff & (unsigned int)(rxdata); // destination address is rxdata[]
    DMDA2L = TX_DATA_FIFO; /* SFCS0 destination address */
```

```
// Number of triggers required to transfer DMBC elements

    DMBC1 = 0x000F;/* 16 data to be received */
    DMBC2 = 0x000E;/* 15 data to be sent with DMA, because first
                      transmission is started manually */

// source address is internal RAM: TS1 TS0 = 1 0;
// Destination address is on chip IO TD1 TD0 = 1 1;
// Source address is incremented SAD = 0
// Destination address is fixed DAD= 1
// transmit 32 bit: DS1 DS0 = 1 0;
// Single transfer mode TM1 TM0 = 0 0;
// Transfer from source to destination TDIR = 0

    DMADC1 = 0xE800; // transfer size 8bit, single transfer mode
    // Transfer from source to destination
    DMADC2 = 0xB280; // 32bit length to transmit the data to SFCS0 register

    DMAIC1 = 0x00; // Highest priority for DMA channel 1

// DMA interrupt trigger
    DTFR1 = 0x29; // DMA channel 1 is triggered on CSI30 interrupt INTC30I
    DTFR2 = 0x29; // DMA channel 2 is triggered on CSI30 interrupt INTC30I
                  /* trigger will load SFCS and SFBD register to send the
                             next data (8bit)*/

    DMCHC1 = 0x21; // DMA transfer is enabled
    DMCHC2 = 0x21;
}


/***********************************************************************
Interruption function IntTimerM_Compare()
This interrupt function allow to start the CSI3 transfer every 300 µs
***********************************************************************/
void IntTimerM_Compare(void)
#pragma ghs interrupt
{
    InitDMA();

/* clear FIFO pointer */
    _FPCLR0 = 1;

    mbytes = 0;

/* Start the first transmission every 300 µs by writing the data to SFDB register */
/* Write CS into transmit CS register */
    SFCS0L = csdata[mbytes];

/* Write byte into transmit data register */
    SFDB0L = txdata[mbytes];

}

void INTDMA1(void)
#pragma ghs interrupt
{
/* clear FIFO pointer to make the last chip select inactive*/
    _FPCLR0 = 1;
}
```

```
/***********************************************************************
Main Function
***********************************************************************/

void main(void)
{
    DI();
/* Configuration for
               - port
               - QSPI   */
    InitialiseRegisters();
    EI();

    while(1){};
}
```

### 3.5.3 FIFO buffer mode without DMA

```
//============================================================================
// PROJECT: V850E/RS1 CSI3 driver in FIFO buffer mode without DMA
// MODULE : Main.c
// VERSION: 1.0
// DATE   : 22/06/05
//============================================================================
//                          C O P Y R I G H T
//============================================================================
// Copyright (c) 2002 by NEC Electronics (Europe) GmbH, Succursale Française.
// All rights reserved.
// 9, rue Paul Dautier
// 78142 Vélizy Cedex
// France
//============================================================================
//
// Warranty Disclaimer:
// Because the Product(s) is licensed free of charge, there is no warranty of
// any kind whatsoever and expressly disclaimed and excluded by NEC, either
// expressed or implied, including but not limited to those for non-
// infringement of intellectual property, merchantability and/or fitness for
// the particular purpose. NEC shall not have any obligation to maintain,
// service or provide bug fixes for the supplied Product(s) and/or the
// Application.
//
// Each User is solely responsible for determining the appropriateness of
// using the Product(s) and assumes all risks associated with its exercise
// of rights under this Agreement, including, but not limited to the risks
// and costs of program errors, compliance with applicable laws, damage to
// or loss of data, programs or equipment, and unavailability or
// interruption of operations.
//
// Limitation of Liability:
// In no event shall NEC be liable to the User for any incidental,
// consequential, indirect, or punitive damage (including but not limited to
// lost profits) regardless of whether such liability is based on breach of
// contract, tort, strict liability, breach of warranties, failure of
// essential purpose or otherwise and even if advised of the possibility of
// such damages. NEC shall not be liable for any services or products provided
// by third party vendors, developers or consultants identified or referred
// to the User by NEC in connection with the Product(s) and/or the
// Application.
//
//============================================================================
// Environment: Devices   : V850E/RS1
//              Assembler : GHS AS-V850                   Version: 4.0.5
//              C-Compiler: GHS CCV850, Multi 2000, V800  Version: 4.0.5
//              Linker    : GHS lx, Multi 2000, V800      Version: 4.0.5
//              Debugger  : GHS Multi 2000, V800          Version: 4.0.5
//============================================================================
// Version    Author             Description
// 1.0        Ferry-Chappuis     Released
//============================================================================

/*****************************************************************
This software is used to develop the module for the CSI3 in FIFO buffer mode
without DMA interface.

Notes:
In order to send the data 2 chip select are used alternatively.
The received data are read when the end of transmission interrupt INTC30I occurs.
*****************************************************************/
```

```
/*****************************************************************************
 Includes
*****************************************************************************/
#include “Inc\df3402.h”
#include “Inc\df3402extIO.h”

/*****************************************************************************
** Declaration of local types **
*****************************************************************************/
typedef unsigned char uchar;
typedef unsigned long ulong;

/*****************************************************************************
** Declaration of variables **
*****************************************************************************/
int            mbytes = 0;
// array for data received / transmitted and chip select
uchar     rxdata[16] ={1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
uchar     txdata[16] ={8,8,0,0,0,0,12,12,0,0,0,0,11,11,15,15};
uchar     csdata[16] ={0xE, 0XB, 0xE, 0xE, 0xB, 0xB, 0xE, 0XB, 0xE, 0xE, 0xB,
                       0xB, 0xE, 0XB, 0xE, 0XB};


/*****************************************************************************
 CSI3 registers initialisation for single buffer mode operation
*****************************************************************************
This function initialises the selected CSI3 interface
*****************************************************************************/
void InitCSI3FIFO8bits(void)
{

 /* Configure serial transfer operation */
 /* This register must be initialised in several times because TRMD, DIR, CSIT, CSWE
      and CSMD bits can be written only when CTXE and CRXE are 0 */
 _POWER0 = 1;


 /* CSIL and CSIC registers must be written first because it is possible only if */
 /* CTXE and CRXE bits in CSIM register are 0 */

 /* Configure clock:   - Idle level low
                       - sampling edge done on the first edge
                       - prescaler associated to clock, K= 1 (PRSOUT = FQCSI/2), N= 1
                         operation clock = PRSOUT/2)*/
 /* Clock speed = 4 MHz*/

 CSIC0 = 0x39;

 /* Configure buffer number to store data and data length
 8-bit data length
 chip select signal is active low */
 CSIL0 = 0x08;

 /* clear FIFO pointer */
 _FPCLR0 = 1;
```

```
 _CSMD0 = 0;    /* CS_INACTIVE_DISABLE */
 _CSWE0 = 0;    /* WAIT_DISABLE */
 _CSIT0 = 1;    /* HALF_CLK_DELAY */
 _DIR0 = 0;     /* Transfer direction: MSB first */
 _TRMD0 = 1;    /* CSI3 in FIFO buffer transfer mode */

   /* enable the transmission */
   _CRXE0 = 1;  /* RX_ENABLE */
   _CTXE0 = 1;  /* TX_ENABLE */

}

/************************************************************************
PORTS MANAGEMENT                                                       *
************************************************************************/

void InitPorts(void)
{
/* Initialise port 0 */
   P0     = 0x00;
   PM0    = 0x00;
   PMC0   = 0x00;
   PFC0   = 0x00;

/* Initialise port 5 */
   P5     = 0x00;
   PM5    = 0xDF;
   PMC5   = 0x30;
   PFC5   = 0x00;

/* Initialise port 9 */
/* LSB */
   P9L     = 0x00;
   PM9L    = 0x80;
   PMC9L   = 0x70;
   PFC9L   = 0x00;
   PFCE9L  = 0x00;

/* MSB */
   P9H     = 0x17;
   PM9H    = 0xCB;
   PMC9H   = 0x30;
   PFC9H   = 0x20;
   PFCE9H  = 0x20;

/* Initialise PSWAP */
   PSWAP = 0x02;

}
```

```
/***********************************************************************
Timer M initialisation for compare operation
Compare value to obtain an interruption every 300 µs:
300 µs / (1/32 MHz) = 9600 = 0x2580
***********************************************************************/
void InitTimerMCompareOperation(void)
{
               TM0CTL0 = 0x80;

               /* compare value for generating an interrupt every 300 µs */
               TM0CMP0 = 0x2580;

               /* interrupt enable and highest level */
               TM0EQIC0 = 0;
}


/*DC*******************************************************************
 Detailed Conception for the function InitialiseRegisters()
***********************************************************************
 Object   : This function initialises I/O ports and other peripherals
              of the MCU
 Parameters: none
 Return   : none
***********************************************************************
 Inputs validation conditions:
***********************************************************************EDC*/
void InitialiseRegisters(void)
{
/* Configure IO ports */
    InitPorts();
/* Configure TIMER M to generate an interrupt every 300 µs*/
    InitTimerMCompareOperation();
    InitCSI3FIFO8bits();
}


/***********************************************************************
Interruption function IntTimerM_Compare()
This interrupt function allow to start the CSI3 transfer every 300 µs
***********************************************************************/
void IntTimerM_Compare(void)
#pragma ghs interrupt
{

/* set the number of data in the SFN register */
/* we need to send 16 data */
    SFN0 = 0x00; /* 16 data to be transmitted/received */


    for (mbytes = 0; mbytes < 16;)
               {
               //test if the FIFO is not full
               if (_SFFUL0 == 0)
                {
                    /* Write CS into transmit CS register */
                    SFCS0L = csdata[mbytes];
                    /* Write byte into transmit data register */
                    SFDB0L = txdata[mbytes];
                    mbytes++; //increments variable to send next data
               }
               }
}
```

```
/*Interrupt of and of transmission of the CSI3 */
void INTC30I(void)
#pragma ghs interrupt
{
/* When the transmission of the 16 data is over, the INTC30I occurs
The received data are read from the SIRB register sequentially*/
    for (mbytes = 0; mbytes < 16; mbytes++)
    {
                rxdata[mbytes] = SIRB0L;
  }

    /* clear FIFO pointer for the next operation with CSI3*/
    SFA0 |= 0x80;
}


/************************************************************************
Main Function
************************************************************************/

void main(void)
{
/* Interrupt configuration */
    C30IC = 0x00;
    /* enable interrupt */
    EI();

/* Configuration for
   - port
   - QSPI */
   InitialiseRegisters();

/* infinite loop */
   while(1){};
}
```

### 3.5.4 FIFO buffer mode with DMA

```
//=========================================================================
// PROJECT: V850E/RS1 CSI3 driver in FIFO buffer mode with DMA
// MODULE : Main.c
// VERSION: 1.0
// DATE   : 22/06/05
//=========================================================================
//                          C O P Y R I G H T
//=========================================================================
// Copyright (c) 2002 by NEC Electronics (Europe) GmbH, Succursale Française.
// All rights reserved.
// 9, rue Paul Dautier
// 78142 Vélizy Cedex
// France
//=========================================================================
//
// Warranty Disclaimer:
// Because the Product(s) is licensed free of charge, there is no warranty of
// any kind whatsoever and expressly disclaimed and excluded by NEC, either
// expressed or implied, including but not limited to those for non-
// infringement of intellectual property, merchantability and/or fitness for
// the particular purpose. NEC shall not have any obligation to maintain,
// service or provide bug fixes for the supplied Product(s) and/or the
// Application.
//
// Each User is solely responsible for determining the appropriateness of
// using the Product(s) and assumes all risks associated with its exercise
// of rights under this Agreement, including, but not limited to the risks
// and costs of program errors, compliance with applicable laws, damage to
// or loss of data, programs or equipment, and unavailability or
// interruption of operations.
//
// Limitation of Liability:
// In no event shall NEC be liable to the User for any incidental,
// consequential, indirect, or punitive damage (including but not limited to
// lost profits) regardless of whether such liability is based on breach of
// contract, tort, strict liability, breach of warranties, failure of
// essential purpose or otherwise and even if advised of the possibility of
// such damages. NEC shall not be liable for any services or products provided
// by third party vendors, developers or consultants identified or referred
// to the User by NEC in connection with the Product(s) and/or the
// Application.
//
//=========================================================================
// Environment: Devices   : V850E/RS1
//              Assembler : GHS AS-V850                  Version: 4.0.5
//              C-Compiler: GHS CCV850, Multi 2000, V800 Version: 4.0.5
//              Linker    : GHS lx, Multi 2000, V800     Version: 4.0.5
//              Debugger  : GHS Multi 2000, V800         Version: 4.0.5
//=========================================================================
// Version     Author             Description
// 1.0         Ferry-Chappuis     Released
//=========================================================================

/*************************************************************************
This software is used to develop the module for the CSI3 in FIFO buffer mode
with DMA interface.

Notes:
In order to send the data 2 chip select are used alternatively.
The received data are read when the end of transmission interrupt INTC30I occurs.
*************************************************************************/
```

```
/*****************************************************************************
 Includes
*****************************************************************************/


#include “Inc\df3402.h”
#include “Inc\df3402extIO.h”


/*****************************************************************************
** Declaration of local types **
*****************************************************************************/
typedef unsigned char uchar;
typedef unsigned long ulong;

/*****************************************************************************
** Declaration of variables **
*****************************************************************************/

int            mbytes = 0;
// array for data received / transmitted and chip select
uchar      rxdata[16] ={1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
uchar      txdata[16] ={8,8,0,0,0,0,12,12,0,0,0,0,11,11,15,15};
uchar      csdata[16] = 0xE, 0XB, 0xE, 0xE, 0xB, 0xB, 0xE, 0XB, 0xE, 0xE, 0xB,
                        0xB, 0xE, 0XB, 0xE, 0XB};

/* Data + chip select to be sent in 16 data transfer */
/* 32 bit send to SFDB and SFCL registers     */

ulong    txdata32[16] ={0x0008000E, 0x0008000B, 0x0000000E, 0x0000000E,
                        0x0000000B, 0x0000000B, 0x000C000E, 0x000C000B,
                        0x0000000E, 0x0000000E, 0x0000000B, 0x0000000B,
                        0x000B000E, 0x000B000B, 0x000F000E, 0x000F000B};


#define TX_DATA_FIFO 0xFD44;// define address of SFCS0 as destination address for DMA
channel 2
#define RX_DATA_FIFO 0xFD42;// define address of SIRB0 as source address for DMA
channel 1
```

```
/*****************************************************************************
 CSI3 registers initialisation for single buffer mode operation
*****************************************************************************
This function initialises the selected CSI3 interface
*****************************************************************************/
void InitCSI3FIFO8bits(void)
{

 /* Configure serial transfer operation */
 /* This register must be initialised in several times because TRMD, DIR, CSIT, CSWE
      and CSMD bits can be written only when CTXE and CRXE are 0 */
 _POWER0 = 1;


 /* CSIL and CSIC registers must be written first because it is possible only if */
 /* CTXE and CRXE bits in CSIM register are 0 */

 /* Configure clock:   - Idle level low
                       - sampling edge done on the first edge
                       - prescaler associated to clock, K= 1 (PRSOUT = FQCSI/2), N= 1
                         operation clock = PRSOUT/2)*/
 /* Clock speed = 4 MHz*/

 CSIC0 = 0x39;

 /* Configure buffer number to store data and data length
 8-bit data length
 chip select signal is active low */
 CSIL0 = 0x08;

 /* clear FIFO pointer */
 _FPCLR0 = 1;


 _CSMD0 = 0;   /* CS_INACTIVE_DISABLE */
 _CSWE0 = 0;   /* WAIT_DISABLE */
 _CSIT0 = 1;   /* HALF_CLK_DELAY */
 _DIR0 = 0;    /* Transfer direction: MSB first */
 _TRMD0 = 1;   /* CSI3 in FIFO buffer transfer mode */

   /* enable the transmission */
   _CRXE0 = 1; /* RX_ENABLE */
   _CTXE0 = 1; /* TX_ENABLE */

   /* set the number of data in the SFN register */
   /* we need to send 16 data */
   SFN0 = 0x00; /* 16 data to be transmitted/received */

}
```

```
/************************************************************************
PORTS MANAGEMENT                                                       *
************************************************************************/

void InitPorts(void)
{
/* Initialise port 0 */
   P0     = 0x00;
   PM0    = 0x00;
   PMC0   = 0x00;
   PFC0   = 0x00;

/* Initialise port 5 */
   P5     = 0x00;
   PM5    = 0xDF;
   PMC5   = 0x30;
   PFC5   = 0x00;

/* Initialise port 9 */
/* LSB */
   P9L     = 0x00;
   PM9L    = 0x80;
   PMC9L   = 0x70;
   PFC9L   = 0x00;
   PFCE9L  = 0x00;

/* MSB */
   P9H     = 0x17;
   PM9H    = 0xCB;
   PMC9H   = 0x30;
   PFC9H   = 0x20;
   PFCE9H  = 0x20;

/* Initialise PSWAP */
   PSWAP = 0x02;

}


/*
************************************************************************
Timer M initialisation for compare operation
Compare value to obtain an interruption every 300 µs:
300 µs / (1/32 MHz) = 9600 = 0x2580
************************************************************************/
void InitTimerMCompareOperation(void)
{
               TM0CTL0 = 0x80;

               /* compare value for generating an interrupt every 300 µs */
               TM0CMP0 = 0x2580;

               /* interrupt enable and highest level */
               TM0EQIC0 = 0;
}
```

```
/***********************************************************************
InitialiseRegisters()
This function initialises I/O ports and other peripherals of the MCU
***********************************************************************/
void InitialiseRegisters(void)
{
/* Configure IO ports */
               InitPorts();

/* Configure TIMER M compare operation to generate an interrupt every 300 µs*/
               InitTimerMCompareOperation();

               InitCSI3FIFO8bits();

}


/* ----- Set DMA Registers ----- */
/* DMA channel 1 used for received data */
/* DMA channel 2 is used to transmit the data: 32 bit into SFCS and SFDB register */
/* The data transfer is started using a software trigger of DMA channel 2 in the */
/* timer M routine*/
/* DMA channel 1 is triggered by the CSI3 interrupt INTC30I (end of transmission)*/
/* Both DMA are in Fixed channel transfer mode */
/*---------------------------------------------------------------------*/

void InitDMA(void)
{

/* DMA control register (DMC)*/
// The DMC register controls the operation and the clock supply of the DMA controller.
// It can be read or written in 8-bit or 1-bit units. Initial value is 00H by reset.

//DMC |STPDIS|IDMEN|STPSET| STPCLR| 0 |0 |0 |POWER


// clear the status flag. All DMA transfer are operational: STPDIS = 0
// NMI interrupts and stops all on-going DMA transfers: IDMEN = 1
// DMA transfer interrupted by software: STPSET = 1
// The clock supply to the DMA controller is started and the DMA operation enabled.
// POWER = 1
    DMC = 0x71;

// INTDMAn timing is based on DMA execution: TCS = 1 is recommended for RS1
// Clear any pending DMA transfer request: FCLR = 1
// DMA channel 1 is disabled EN = 0
    DMCHC1 = 0x04;
    DMCHC2 = 0x04;

    DMSA1H = 0x02FF;
    DMSA2H = 0x02FF;

/* source adress */
    DMSA1L = RX_DATA_FIFO; // source - point to SIRB0 address buffer.
   DMSA2L = 0xffff & (unsigned int)(&Txdata32[0x00]); // source - point to transmit buffer

    DMDA1H = 0x02FF;
    DMDA2H = 0x02FF;
```

```
/* destination adress */
    DMDA1L = 0xffff & (unsigned int)(rxdata); // destination address is rxdata[]
    DMDA2L = TX_DATA_FIFO; /* SFCS0 destination address */

// One trigger is required to transfer DMBC elements

    DMBC1 = 0x000F;/* 16 data to be received with DMA ch1*/
    DMBC2 = 0x000F;/* 16 data to be sent with DMA ch2 */

// source address is internal RAM: TS1 TS0 = 1 0;
// Destination address is on chip IO TD1 TD0 = 1 1;
// Source address is incremented SAD = 0
// Destination address is fixed DAD= 1
// transmit 32 bit: DS1 DS0 = 1 0;
// Single transfer mode TM1 TM0 = 0 0;
// Transfer from source to destination TDIR = 0

    DMADC1 = 0xE804; // transfer size 8bit, Fixed channel transfer mode
// Transfer from source to destination

    DMADC2 = 0xB284; // 32bit length to transmit the data to SFCS0 register
    //Fixed channel transfer mode

// Unmasked DMA channel 1 interrupt
    DMAIC1 = 0x00; // Highest priority for DMA channel 1

// DMA interrupt trigger
    DTFR1 = 0x29; // DMA channel 1 is triggered on CSI30 interrupt INTC30I
    DTFR2 = 0x00; // DMA channel 2 is triggered by software
    // trigger will load SFCS and SFBD register to send the data (8bit)

    DMCHC1 = 0x21; // DMA transfer is enabled
    DMCHC2 = 0x21;
}

void INTDMA1(void)
#pragma ghs interrupt
{
/* clear all FIFO pointers to make the last chip select inactive*/
    _FPCLR0 = 1;
}


/***********************************************************************
Interruption function IntTimerM_Compare()
This interrupt function allow to start the CSI3 transfer every 300 µs
***********************************************************************/
void IntTimerM_Compare(void)
#pragma ghs interrupt
{
    InitDMA();

// test if the FIFO buffer is not full
    if (_SFFUL0 == 0)
    {
               // Start DMA channel 2 with software trigger
               DMCHC2 = 0x23; // STG = 1
  }
}
```

```
/***********************************************************************
Main Function
***********************************************************************/
void main(void)
{

/* enable interrupt */
               EI();

/* Configuration for
                        - port
                        - QSPI */
               InitialiseRegisters();

/* infinite loop */
               while(1){};

}
```

### 3.5.5 Startup program: Startup_df3402.850

**Remark:** Startup is a common program to every operating mode. In the section “Selection of external interrupt service handler”, please take care to enable only the required interrupt service handling used in the appropriate software example programs.

```
--*********************************************************************
--*           Startup_df3402.850 -
--*
--*   NEC V850 microcontroller device uPD70F3402
--*
--*   GHS Startup Module template
--*
--*
--* Copyright (C) NEC Corporation 2003
--* Startup_df3402.850 created from device file df3400.800 [E1.00e]
--* by DeFiX V1.06a
--*
--* This file is only intended as a sample supplement to NEC tools.
--* Feel free to adapt it to your own needs.
--* This File is provided 'as is' without warranty of any kind.
--* Neither NEC nor their sales representatives can be held liable
--* of any inconvenience or problem caused by its contents.
--*********************************************************************

------------------------------------------------------------------------
------------- Selection of external interrupt service handler
------------- User modifiable section
------------- Please unable the required interrupt service handler
------------------------------------------------------------------------

#define RESET_ENABLE          // Always used
#define INTTM0EQ0_ENABLE      // Always used
#define INTC30I_ENABLE        // Only used in single and FIFO buffer mode without DMA
--#define INTC30O_ENABLE      // Not used
#define INTDMA1_ENABLE        // Only used in single and FIFO buffer mode with DMA


.include “df3402.s”

------------------------------------------------------------------------
------------- Basic Initialisation of the controller
------------- User modifiable section
--
-- This startup initialize the PLL0 and PLL1
------------------------------------------------------------------------

.text

_RESET:
```

```
-- ============================================================================
-- == Initialisation of NPB wait states======
-- ============================================================================

----------------------------------------------------------------------------
-- NPB Wait Control: VSWC
-- Operation Frequency (fx)  VSWC setting
-- 4 MHz <=fx <= 25 MHz      11H
-- 25 MHz <=fx <= 33 MHz     12H
-- 33 MHz <=fx <= 50 MHz     14H
----------------------------------------------------------------------------

     movea   0x12,zero,r19
     st.b    r19,VSWC[zero]

----------------------------------------------------------------------------
-- initialisation of the watchdog timer
--
-- WDTM2     |0| WDM21| WDM20| WDCS24| WDCS23| WDCS22| WDCS21| WDCS20|
--
-- WDM21 WDM20 Selection of operation mode of watchdog timer 2
--    0     0   Stops operation
--    0     1   Non-maskable interrupt request mode (generation of INTWDT2)
--    1     -   Reset mode (generation of WDTRES2)
--
----------------------------------------------------------------------------

     st.b    r0,WDTM2[zero]

-- ============================================================================
-- == Initialisation of CLOCKTREE ======
-- ==                             ======
-- use PLL0 for CPU @ Fmax        ======
-- To use PLL                     ======
-- 1) setting OCKSn register      ======
-- 2) setting CKC register        ======
-- 1) setting PLLCTLn register    ======
-- 1) setting MPCCTL register     ======
-- ============================================================================

----------------------------------------------------------------------------
-- Clock selection register 0 (OCKS0)
-- OCKS0     | 0 | 0 | 0 |OCKSEN0| OCKSTH0| 0 |OCKS01| OCKS00|
--
-- OCKSEN0   Specified for execution enable
--    0      PLL operation Disable
--    1      PLL operation Enable
--
-- OCKSTH0   Specified for output clock through or divide
--    0      Output clock is divided clock by setting OCKS01 & OCKS00
--    1      Output clock is through
--
-- OCKS01    OCKS00    Specified for divider factor
--    0      0         fxx/2
--    0      1         fxx/3
--    1      0         fxx/4
--    1      1         fxx/5
----------------------------------------------------------------------------
```

```
      movea   0x11,zero,r19
      st.b    r19,OCKS0[zero]

-- To obtain 16 MHz with PLL1 we need to configure the following register to the
-- described value
-- OCKS1 divide value = 5 ==> OCKS1 = 0x13
-- CKC divide value = 1 ==> CKC = 0x03

      movea   0x13,zero,r19
      st.b    r19,OCKS1[zero]


------------------------------------------------------------------------
-- Clock control register (CKC)
-- CKC        |0|0|0|0|0|0|CKDIV1|CKDIV0|
--
-- CKDIV1 CKDIV0        System clock (fXX) select on PLL mode
--   0       0          Setting prohibit
--   0       1          fxx = fPLL(PLL internal clock) / 4
--   1       0          fxx = fPLL(PLL internal clock) / 2
--   1       1          fxx = fPLL(PLL internal clock)
--
------------------------------------------------------------------------
      movea   0x03,zero,r19
      -- Dummy write to PRCMD register before
      st.b    r19,PRCMD[zero]
      st.b    r19,CKC[zero]
      nop
      nop
      nop
      nop
      nop

      st.b    r0,PLLCTL0[zero]
      st.b    r0,PLLCTL1[zero]
------------------------------------------------------------------------
-- Switch from oscillator clock to PLL clock
-- MPCCTL |SELPLL| 0 | 0 | 0 |MCKSEL|PCKSEL| STPPLL1 |STPPLL0|
--
-- SELPLL selection of PLL operation
------------------------------------------------------------------------

-- need to write in two times because the SELPLL bit can not be configured
-- at the same time
      movea   0x04,zero,r19
      -- Dummy write to PRCMD register before
      st.b    r19,PRCMD[zero]
      st.b    r19,MPCCTL[zero]
      nop
      nop
      nop
      nop
      nop
```

```
      movea   0x84,zero,r19
     st.b    r19,PRCMD[zero]
     st.b    r19,MPCCTL[zero]
     nop
     nop
     nop
     nop
     nop

-- ------------------------------------------------------------------------
-- Extension clock select register (EXCKSEL)
-- This is an 8-bit register that selects the internal clock.
-- This register can be read or written in 8-bit or 1-bit units.

--EXCKSEL     |0|0|CB1CKSEL|CB0CKSEL|CS31CKSEL|CS30CKSEL|AF1CKSEL|AF0CKSEL|
--
-- CS3nCKSEL            CSI3n clock selection (n =0, 1)
--   0                  Normal clock selection (fxx)
--   1                  Extended clock selection (f_PLL_PCKSEL)
-------------------------------------------------------------------------------

     movea   0x0C,zero,r19
     st.b    r19,EXCKSEL[zero]
```

```
------------------------------------------------------------------------
-- Processor clock control register initialization (PCC):
--
-- PCC|      0 | 0 |MFRC| 0 | 0 |CK2| CK1| CK0|
--
-- MFRC internal main clock feedback resistor
--       0 = connected     // 1 = disconnected
--
--     CK2   CK1   CK0
--   0    0    0       -> fxx
--   0    0    1       -> fxx/2
--   0    1    0       -> fxx/4
--   0    1    1       -> fxx/8 (reset)
--   1    0    0       -> fxx/16
--   1    0    1       -> fxx/32
--   1    1    0       -> Setting prohibited
--   1    1    1       -> Setting prohibited
--
------------------------------------------------------------------------
     _Select_fxx:

      --clear sys register
     st.b   r0,SYS[zero]

     -- Dummy write to PRCMD register before
     st.b    r0,PRCMD[zero]
     st.b    r0,PCC[zero]

     --movea 0x02,zero,r19
     --st.b    r19,PRCMD[zero]
     --st.b    r19,PCC[zero]

     nop
     nop
     nop
     nop
     nop

      -- Check that no protection error occurred
      tst1 0,SYS[zero]

      -- Branch back if PRERR bit is 1
      bne _Select_fxx
------------------------------------------------------------------------
-- adjust PLL lockup time (PLLS)
--
-- 0x 0 0 0 0 0 0 PLLS1 PLLS0
--
-- PLLS1 PLLS0-         >  Lockup time (> 800 µs)
--   0           0     -> 2^10/fx
--   0           1     -> 2^11/fx
--   1           0     -> 2^12/fx (1024 µs at 4 MHz, 819 µs at 5 MHz)
--   1           1     -> 2^13/fx (value after reset)
--
------------------------------------------------------------------------
      movea   0x02,zero,r19
      st.b    r19,PLLS[zero]
```

```
-- ============================================================================
-- == Initialisation some SYSTEM PARAMETERS for GHS======
-- ============================================================================

      -- Initialisation of the global pointer
      movhi   hi(___ghsbegin_sdabase),zero,gp
      movea   lo(___ghsbegin_sdabase),gp,gp

      -- Initialisation of the text pointer
      movhi   hi(___ghsbegin_robase),zero,tp
      movea   lo(___ghsbegin_robase),tp,tp

      -- Initialisation of the stack pointer

      movhi   hi(___ghsend_stack-4),zero,sp

      movea   lo(___ghsend_stack-4),sp,sp

      -- Initialisation of the MM register
      --...

      -- Initialisation of the BCC register
      --...

      -- Initialisation of the DWC register
      --...

      -- Jump to the Initialisation functions of the library,
      -- from there to main()
      jr __start

------------------------------------------------------------------------------

------------------------------------------------------------------------------
------------- Add section ".intvect" to your linker command file
------------- at address 0x0000
-------------
------------- i.e..intvect 0x0000:
------------------------------------------------------------------------------

.section".intvect",.text

------------------------------------------------------------------------------
------------- Initializing of interrupt vector table
------------- Please don't modify this section
------------------------------------------------------------------------------

#ifdef RESET_ENABLE
     .offset 0x0000
     .extern _RESET
     jr _RESET
#endif
```

```
#ifdef INTWDT2_ENABLE
     .offset 0x0020
     .extern _INTWDT2
     jr _INTWDT2
#endif

#ifdef NMI2_ENABLE
     .offset 0x0030
     .extern _NMI2
     jr _NMI2
#endif

#ifdef TRAP0_ENABLE
     .offset 0x0040
     .extern _TRAP0
     jr _TRAP0
#endif

#ifdef TRAP1_ENABLE
     .offset 0x0050
     .extern _TRAP1
     jr _TRAP1
#endif

#ifdef DBG0_ENABLE
     .offset 0x0060
     .extern _DBG0
     jr _DBG0
#endif

#ifdef ILGOP_ENABLE
     .offset 0x0060
     .extern _ILGOP
     jr _ILGOP
#endif

#ifdef INTLVI_ENABLE
     .offset 0x0080
     .extern _INTLVI
     jr _INTLVI
#endif

#ifdef INTP0_ENABLE
     .offset 0x0090
     .extern _INTP0
     jr _INTP0
#endif

#ifdef INTP1_ENABLE
     .offset 0x00a0
     .extern _INTP1
     jr _INTP1
#endif
```

```
#ifdef INTP2_ENABLE
     .offset 0x00b0
     .extern _INTP2
     jr _INTP2
#endif

#ifdef INTP3_ENABLE
     .offset 0x00c0
     .extern _INTP3
     jr _INTP3
#endif

#ifdef INTP4_ENABLE
     .offset 0x00d0
     .extern _INTP4
     jr _INTP4
#endif

#ifdef INTP5_ENABLE
     .offset 0x00e0
     .extern _INTP5
     jr _INTP5
#endif

#ifdef INTP6_ENABLE
     .offset 0x00f0
     .extern _INTP6
     jr _INTP6
#endif

#ifdef INTP7_ENABLE
     .offset 0x0100
     .extern _INTP7
     jr _INTP7
#endif

#ifdef INTTQ0OV_ENABLE
     .offset 0x0110
     .extern _INTTQ0OV
     jr _INTTQ0OV
#endif

#ifdef INTTQ0CC0_ENABLE
     .offset 0x0120
     .extern _INTTQ0CC0
     jr _INTTQ0CC0
#endif

#ifdef INTTQ0CC1_ENABLE
     .offset 0x0130
     .extern _INTTQ0CC1
     jr _INTTQ0CC1
#endif
```

```
#ifdef INTTQ0CC2_ENABLE
     .offset 0x0140
     .extern _INTTQ0CC2
     jr _INTTQ0CC2
#endif

#ifdef INTTQ0CC3_ENABLE
     .offset 0x0150
     .extern _INTTQ0CC3
     jr _INTTQ0CC3
#endif

#ifdef INTTP0OV_ENABLE
     .offset 0x0160
     .extern _INTTP0OV
     jr _INTTP0OV
#endif

#ifdef INTTP0CC0_ENABLE
     .offset 0x0170
     .extern _INTTP0CC0
     jr _INTTP0CC0
#endif

#ifdef INTTP0CC1_ENABLE
     .offset 0x0180
     .extern _INTTP0CC1
     jr _INTTP0CC1
#endif

#ifdef INTTP1OV_ENABLE
     .offset 0x0190
     .extern _INTTP1OV
     jr _INTTP1OV
#endif

#ifdef INTTP1CC0_ENABLE
     .offset 0x01a0
     .extern _INTTP1CC0
     jr _INTTP1CC0
#endif

#ifdef INTTP1CC1_ENABLE
     .offset 0x01b0
     .extern _INTTP1CC1
     jr _INTTP1CC1
#endif

#ifdef INTTP2OV_ENABLE
     .offset 0x01c0
     .extern _INTTP2OV
     jr _INTTP2OV
#endif
```

```
#ifdef INTTP2CC0_ENABLE
     .offset 0x01d0
     .extern _INTTP2CC0
     jr _INTTP2CC0
#endif

#ifdef INTTP2CC1_ENABLE
     .offset 0x01e0
     .extern _INTTP2CC1
     jr _INTTP2CC1
#endif

#ifdef INTTP3OV_ENABLE
     .offset 0x01f0
     .extern _INTTP3OV
     jr _INTTP3OV
#endif

#ifdef INTTP3CC0_ENABLE
     .offset 0x0200
     .extern _INTTP3CC0
     jr _INTTP3CC0
#endif

#ifdef INTTP3CC1_ENABLE
     .offset 0x0210
     .extern _INTTP3CC1
     jr _INTTP3CC1
#endif

#ifdef INTTM0EQ0_ENABLE
     .offset 0x0220
     .extern _INTTM0EQ0
     jr _INTTM0EQ0
#endif

#ifdef INTCB0R_ENABLE
     .offset 0x0230
     .extern _INTCB0R
     jr _INTCB0R
#endif

#ifdef INTCB0T_ENABLE
     .offset 0x0240
     .extern _INTCB0T
     jr _INTCB0T
#endif

#ifdef INTCB1R_ENABLE
     .offset 0x0250
     .extern _INTCB1R
     jr _INTCB1R
#endif
```

```
#ifdef INTCB1T_ENABLE
     .offset 0x0260
     .extern _INTCB1T
     jr _INTCB1T
#endif

#ifdef INTUA0R_ENABLE
     .offset 0x0270
     .extern _INTUA0R
     jr _INTUA0R
#endif

#ifdef INTUA0T_ENABLE
     .offset 0x0280
     .extern _INTUA0T
     jr _INTUA0T
#endif

#ifdef INTUA1R_ENABLE
     .offset 0x0290
     .extern _INTUA1R
     jr _INTUA1R
#endif

#ifdef INTUA1T_ENABLE
     .offset 0x02a0
     .extern _INTUA1T
     jr _INTUA1T
#endif

#ifdef INTAD_ENABLE
     .offset 0x02b0
     .extern _INTAD
     jr _INTAD
#endif

#ifdef INTC0ERR_ENABLE
     .offset 0x02c0
     .extern _INTC0ERR
     jr _INTC0ERR
#endif

#ifdef INTC0WUP_ENABLE
     .offset 0x02d0
     .extern _INTC0WUP
     jr _INTC0WUP
#endif

#ifdef INTC0REC_ENABLE
     .offset 0x02e0
     .extern _INTC0REC
     jr _INTC0REC
#endif
```

```
#ifdef INTC0TRX_ENABLE
     .offset 0x02f0
     .extern _INTC0TRX
     jr _INTC0TRX
#endif

#ifdef INTC30I_ENABLE
     .offset 0x0300
     .extern _INTC30I
     jr _INTC30I
#endif

#ifdef INTC30O_ENABLE
     .offset 0x0310
     .extern _INTC30O
     jr _INTC30O
#endif

#ifdef INTC31I_ENABLE
     .offset 0x0320
     .extern _INTC31I
     jr _INTC31I
#endif

#ifdef INTC31O_ENABLE
     .offset 0x0330
     .extern _INTC31O
     jr _INTC31O
#endif

#ifdef INTCB2R_ENABLE
     .offset 0x0340
     .extern _INTCB2R
     jr _INTCB2R
#endif

#ifdef INTCB2T_ENABLE
     .offset 0x0350
     .extern _INTCB2T
     jr _INTCB2T
#endif

#ifdef INTTQ1OV_ENABLE
     .offset 0x0360
     .extern _INTTQ1OV
     jr _INTTQ1OV
#endif

#ifdef INTTQ1CC0_ENABLE
     .offset 0x0370
     .extern _INTTQ1CC0
     jr _INTTQ1CC0
#endif
```

```
#ifdef INTTQ1CC1_ENABLE
     .offset 0x0380
     .extern _INTTQ1CC1
     jr _INTTQ1CC1
#endif

#ifdef INTTM1EQ0_ENABLE
     .offset 0x0390
     .extern _INTTM1EQ0
     jr _INTTM1EQ0
#endif

#ifdef INTTQ1CC2_ENABLE
     .offset 0x0390
     .extern _INTTQ1CC2
     jr _INTTQ1CC2
#endif

#ifdef INTTM2EQ0_ENABLE
     .offset 0x03a0
     .extern _INTTM2EQ0
     jr _INTTM2EQ0
#endif

#ifdef INTTQ1CC3_ENABLE
     .offset 0x03a0
     .extern _INTTQ1CC3
     jr _INTTQ1CC3
#endif

#ifdef INTUA2R_ENABLE
     .offset 0x03b0
     .extern _INTUA2R
     jr _INTUA2R
#endif

#ifdef INTUA2T_ENABLE
     .offset 0x03c0
     .extern _INTUA2T
     jr _INTUA2T
#endif

#ifdef INTC1ERR_ENABLE
     .offset 0x03d0
     .extern _INTC1ERR
     jr _INTC1ERR
#endif

#ifdef INTC1WUP_ENABLE
     .offset 0x03e0
     .extern _INTC1WUP
     jr _INTC1WUP
#endif
```

```
#ifdef INTC1REC_ENABLE
     .offset 0x03f0
     .extern _INTC1REC
     jr _INTC1REC
#endif

#ifdef INTC1TRX_ENABLE
     .offset 0x0400
     .extern _INTC1TRX
     jr _INTC1TRX
#endif

#ifdef INTDMA0_ENABLE
     .offset 0x0410
     .extern _INTDMA0
     jr _INTDMA0
#endif

#ifdef INTDMA1_ENABLE
     .offset 0x0420
     .extern _INTDMA1
     jr _INTDMA1
#endif

#ifdef INTDMA2_ENABLE
     .offset 0x0430
     .extern _INTDMA2
     jr _INTDMA2
#endif

#ifdef INTDMA3_ENABLE
     .offset 0x0440
     .extern _INTDMA3
     jr _INTDMA3
#endif

#ifdef INTDMA4_ENABLE
     .offset 0x0450
     .extern _INTDMA4
     jr _INTDMA4
#endif

#ifdef INTDMA5_ENABLE
     .offset 0x0460
     .extern _INTDMA5
     jr _INTDMA5
#endif

#ifdef INTFLTMO_ENABLE
     .offset 0x0470
     .extern _INTFLTMO
     jr _INTFLTMO
#endif
```

# Appendix A Index

NEC

# Facsimile Message

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

From:

Name

Company

Tel. FAX

Address

*Thank you for your kind support.*

**North America**
NEC Electronics America Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

**Europe**
NEC Electronics (Europe) GmbH
Market Communication Dept.
Fax: +49(0)-211-6503-1344

**Hong Kong, Philippines, Oceania**
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

**Korea**
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

**Taiwan**
NEC Electronics Taiwan Ltd.
Fax: 02-2719-5951

**Asian Nations except Philippines**
NEC Electronics Singapore Pte. Ltd.
Fax: +65-6250-3583

**Japan**
NEC Semiconductor Technical Hotline
Fax: +81- 44-435-9608

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

| Document Rating | Excellent | Good | Acceptable | Poor |
|---|---|---|---|---|
| Clarity | ☐ | ☐ | ☐ | ☐ |
| Technical Accuracy | ☐ | ☐ | ☐ | ☐ |
| Organization | ☐ | ☐ | ☐ | ☐ |

**[MEMO]**