ssJ1708
User's Manual

Version 1.3
Revised February 2nd, 2009
Created by the J1708 Experts

ssJ1708 Protocol Stack License

READ THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT CAREFULLY BEFORE OPENING THE PACKAGE CONTAINING THE PROGRAM DISTRIBUTION MEDIA (DISKETTES, CD, ELECTRONIC MAIL), THE COMPUTER SOFTWARE THEREIN, AND THE ACCOMPANYING USER DOCUMENTATION. THIS SOURCE CODE IS COPYRIGHTED AND LICENSED (NOT SOLD). BY OPENING THE PACKAGE CONTAINING THE SOURCE CODE, YOU ARE ACCEPTING AND AGREEING TO THE TERMS OF THIS LICENSE AGREEMENT. IF YOU ARE NOT WILLING TO BE BOUND BY THE TERMS OF THIS LICENSE AGREEMENT, YOU SHOULD PROMPTLY RETURN THE PACKAGE IN UNOPENED FORM, AND YOU WILL RECEIVE A REFUND OF YOUR MONEY. THIS LICENSE AGREEMENT REPRESENTS THE ENTIRE AGREEMENT CONCERNING THE J1708 PROTOCOL STACK BETWEEN YOU AND SIMMA SOFTWARE, INC. (REFERRED TO AS "LICENSOR"), AND IT SUPERSEDES ANY PRIOR PROPOSAL, REPRESENTATION, OR UNDERSTANDING BETWEEN THE PARTIES.

1. Corporate License Grant. Simma Software hereby grants to the purchaser (herein referred to as the “Client”), a non-exclusive license to use the J1708 protocol stack source code software (collectively referred to as the "Software”) as part of Client’s proprietary software applications. This license is for unlimited client applications that can be distributed to an unlimited number of computers/devices/products. Except as provided above, Client agrees to not assign, sublicense, transfer, pledge, lease, rent, or share the Software Code under this License Agreement.

2. Simma Software's Rights. Client acknowledges and agrees that the Software and the documentation are proprietary products of Simma Software and are protected under U.S. copyright law. Client further acknowledges and agrees that all right, title, and interest in and to the Software, including associated intellectual property rights, are and shall remain with Simma Software. This License Agreement does not convey to Client an interest in or to the Software, but only a limited right of use revocable in accordance with the terms of this License Agreement.

3. License Fees. The Client in consideration of the licenses granted under this License Agreement will pay a one-time license fee.

4. Term. This License Agreement shall continue until terminated by either party. Client may terminate this License Agreement at any time. Simma Software may terminate this License Agreement only in the event of a material breach by Client of any term hereof, provided that such shall take effect 60 days after receipt of a written notice from Simma Software of such termination and further provided that such written notice allows 60 days for Client to cure such breach and thereby avoid termination. Upon termination of this License Agreement, all rights granted to Client will terminate and revert to Simma Software. Promptly upon termination of this Agreement for any reason or upon discontinuance or abandonment of Client’s possession or use of the Software, Client must return or destroy, as requested by Simma Software, all copies of the Software in Client’s possession, and all other materials pertaining to the Software (including all copies thereof). Client agrees to certify compliance with such restriction upon Simma Software’s request.

5. Limited Warranty. Simma Software warrants, for Client’s benefit alone, for a period of one year (called the “Warranty Period”) from the date of delivery of the software, that during this period the Software shall operate substantially in accordance with the functionality described in the User's Manual. If during the Warranty Period, a defect in the Software appears, Simma Software will make all reasonable efforts to cure the defect, at no cost to the Client. Client agrees that the foregoing constitutes Client’s sole and exclusive remedy for breach by Simma Software of any warranties made under this Agreement. Simma Software is not responsible for obsolescence of the Software that may result from changes in Client’s requirements. The foregoing warranty shall apply only to the most current version of the Software issued from time to time by Simma Software. Simma Software assumes no responsibility for the use of superseded, outdated, or uncorrected versions of the licensed software. EXCEPT FOR THE WARRANTIES SET FORTH ABOVE, THE SOFTWARE, AND THE SOFTWARE CONTAINED THEREIN, ARE LICENSED "AS IS," AND SIMMA SOFTWARE DISCLAIMS ANY AND ALL OTHER WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

6. Limitation of Liability. Simma Software's cumulative liability to Client or any other party for any loss or damages resulting from any claims, demands, or actions arising out of or relating to this License Agreement shall not exceed the license fee paid to Simma Software for the use of the Software. In no event shall Simma Software be liable for any indirect, incidental, consequential, special, or exemplary damages or lost profits, even if Simma Software has been advised of the possibility of such damages. SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO CLIENT.

7. Governing Law. This License Agreement shall be construed and governed in accordance with the laws of the State of Indiana.

8. Severability. Should any court of competent jurisdiction declare any term of this License Agreement void or unenforceable, such declaration shall have no effect on the remaining terms hereof.

9. No Waiver. The failure of either party to enforce any rights granted hereunder or to take action against the other party in the event of any breach hereunder shall not be deemed a waiver by that party as to subsequent enforcement of rights or subsequent actions in the event of future breaches.

TABLE OF CONTENTS

Chapter 1

Introduction

ssJ1708 is high performance J1708 protocol stack written in ANSI C. ssJ1708 adheres to both the SAE J1708 specification and to the software development best practices guidelines described in MISRA C.

ssJ1708 is a modularized design with an emphasis on software readability and performance. ssJ1708 is easy to understand and can be used on any CPU or DSP with a UART, and can run with or without an RTOS.

1.1 Source Files

Filenames	File Description
j1708.c	Core source file for ssJ1708
j1708.h	Core header file for ssJ1708

Table 1-1: ssJ1708 files

1.2 Code and RAM usage

Code size varies based on compiler and compiler settings, but should be approximately 1,200 bytes. RAM usage varies with the size of receive and transmit buffers. Assuming the architecture supports byte size memory locations, approximate RAM usage can be determined by multiplying the number of receive and transmit buffers by 23 bytes. An additional 15 bytes are used for miscellaneous purposes.

1.3 Requirements

The processor should have the following:

- A native UART.
- A free running timer with overflow period greater than 3.7 milliseconds.
- An external interrupt attached to the RX pin of UART.

Chapter 2

Integration of ssJ1708

This chapter describes how to integrate ssJ1708 into your application. After this is complete, you will be able to receive and transmit J1708 messages. For implementation details, please see the chapters covering the API and the HAL.

2.1 Integration Steps:

1. Implement the hardware specific firmware which is described in chapter 3.

2. Before using any of the ssJ1708 module features, make sure the module has been initialized by calling j1708_init. Typically j1708_init() is called shortly after power-on reset and before the application is started.

3. Call j1708_update() periodically. This function must be called more frequent than the overflow frequency of the free running timer. See the function's description in the API chapter for details.

Chapter 3

ssJ1708 Hardware Abstraction Layer

The hardware abstraction layer (HAL) minimizes and centralizes the CPU specific firmware. ssJ1708 uses a native UART, an external interrupt, and a free running timer of the target architecture. Since these features are hardware depended, it is the requirement of the module integrator to write a small amount of firmware. This chapter gives step by step instructions on what is required. Also, the source code is marked by "PORT:"s which help the integrator find the exact location for the firmware adaptation.

The HAL contains five macros that are responsible for reading the free running timer, transmitting a character, receiving a character, clearing the receive interrupt flag and clearing the external interrupt flag. These five macros are placed at the top of j1708.c.

Function Prototype	Macro Description
J1708_HAL_FRT_READ(val)	Reads the free running timer
J1708_HAL_RX_CHAR(val)	Reads a character from the UART
J1708_HAL_TX_CHAR(val)	Transmits a character out the UART
J1708_HAL_RX_ISR_CLR()	Clears the receive interrupt flag
J1708_HAL_EXT_ISR_CLR()	Clears the external interrupt flag
J1708_LOCK()	Disables RX and EXT interrupts
J1708_UNLOCK()	Enables RX and EXT interrupts

Table 3-1: HAL functions

3.1 UART Operation

The UART needs to be configured for 8 data bits, 1 stop bit, and no parity. The UART speed should be set to 9600 bps, with a maximum error of +/- 0.5%. This initialization may be placed in j1708_init.

3.2 External Interrupt Operation

The external interrupt should be configured to activate on a falling edge of the UART's receive pin. This initialization may be placed in j1708_init.

3.3 Free Running Timer

The free running timer should be configured to count up. When the timer overflows, it should continue counting automatically. The overflow period must be longer than the period between j1708_update() calls. The overflow period must also be larger than 3.7 ms.

3.4 Free Running Timer Size

The type "j1708_frt_t" needs to be set to uint8_t if the timer is 8 bits wide, uint16_t if 16 bits wide, or uint32_t if 32 bits wide. If the timer is neither 8, 16, or 32 bits wide, then additional firmware changes will be required. Please contact Simma Software for details.

3.5 Idle Line Timing

There are two different locations that contain timings that are dependent on the free running timer speed. The module integrator needs to set "J1708_RX_IDLE_TIME" to the number of clock cycles that will occur in 1.25 milliseconds. Also, the j1708_tx_idle_time_t array needs to be initialized with the cycle counts for times that vary between 2.19 and 3.65 milliseconds. See source code for full timing details.

3.6 Interrupt Service Routines

There are two interrupt service routines used by the J1708 module. One is for the UART's receive interrupt and is named j1708_rx_isr(). The other is for the external interrupt and is named j1708_bus_active_isr(). Since compilers define interrupts in different ways, it is the responsibility of the module integrator to define these two ISRs in accordance with the compiler's requirements. The integrator must also place these routines in the interrupt vector table or similar mechanism.

Chapter 4

ssJ1708 Application Program Interface

The ssJ1708 application program interface (API) describes the software interface for the ssJ1708 module. These functions initialize the module, provide a time base for the ssJ1708 module, transmit J1708 messages, and read messages from the J1708 receive buffer.

Function Prototype	Function Description
void j1708_init (void)	Initializes hardware
void j1708_update (void)	Provides required module updates
uint8_t j1708_rx (uint8_t *buf, uint8_t *size)	Reads a J1708 from receive buffer
uint8_t j1708_tx (uint8_t *buf, uint8_t size, uint8_t pri)	Places a J1708 message in the transmit buffer.

Table 4-1: API functions

j1708_init

Function Prototype:

void j1708_init (void);

Description:

This function initializes any J1708 specific hardware. By default, it is an empty function and is the responsibility of the module integrator to implement. See chapter 3 for details.

Parameters:

void

Return Value:

void

j1708_update

Function Prototype:

void j1708_update (void);

Description:

This function provides periodic execution for the ssJ1708 module. During this time ssJ1708 checks the transmit buffer. If a message is present and the bus is idle, then transmission will start. Since this function is responsible for starting transmission of every J1708 message, it should be called often enough to handle the application's transmit requirements. 5 or 10 milliseconds are common update rates. This routine can also be called continuously (e.g. from an idle task or idle portion of scheduler).

This function also tracks overflows of the free running timer. Therefore this function must be called at a higher frequency than the free running timer's overflow frequency. For example, if the free running timer overflows every X milliseconds, than this function must be called more often than once every X milliseconds.

NOTE: This function is also called by j1708_tx and j1708_rx and is not designed to be reentrant. Therefore, this function should not be called from an interrupt context unless it can be guaranteed that this function won't be reentered. For RTOS implementation j1708_update, j1708_tx , and j1708_rx are not thread safe and shouldn't be called from different tasks.

Parameters:

void

Return Value:

void

j1708_rx

Function Prototype:

uint8_t j1708_rx (uint8_t *msg, uint8_t *size);

Description:

If a message is waiting in the J1708 receive buffer, the entire message minus the checksum will be copied into the memory pointed to by msg. All messages returned by this function have a valid checksum.

NOTE: This function is not reentrant.

Parameters:

msg: Points to memory where the J1708 message will be copied. The first location contains the J1708 MID.

size: Size of J1708 message pointed to by msg.

Return Value

Non-zero: Message was not read from receive buffer.

0: Message was read from receive buffer.

j1708_tx

Function Prototype:

uint_8 j1708_tx (uint8_t *buf, uint8_t size, uint8_t pri);

Description:

This function first checks to see if there is memory available in the transmit buffer. If there is, the message checksum is calculated and stored at the end of the message. Then the message is copied into the transmit buffer. Lastly, j1708_update() is called which will check the state of the bus. If the bus is idle, then transmission will be started.

NOTE: This function is not reentrant.

Parameters:

msg: Points to the J1708 message. First location is the J1708 MID.
size: Size of J1708 message pointed to by msg.
pri: Priority of J1708 message

Return Value:

Non-zero: Message was not written to the transmit buffer.
0: Message was written to the transmit buffer.

Chapter 5

Configuration

This chapter describes all configurable items of the ssJ1708 module. These items are stored as defines at the top of j1708.c.

J1708 Receive Buffer Size

This defines the number of J1708 messages the receive buffer will hold. It is recommended to be three or more.

```
#define J1708_RX_BUF_SIZE          3
```

J1708 Transmit Buffer Size

This defines the number of J1708 messages the transmit buffer will hold. It is recommended to be two or more.

```
#define J1708_TX_BUF_SIZE          2
```