Systems Analysis INterface Tool 2(SAINT2)

Programming Reference

Version 2.21

7/26/2012

Table of Contents

1 Introduction 8

1.1 Scope 8

1.2 Precedence 8

1.3 Definitions and Nomenclature 8

2 Overview 9

3 Host to SAINT2 Connection 10

3.1 RS-232 10

3.1.1 RS-232 Set Up 10

3.2 USB 10

4 Message Format 11

4.1 Data Stream 11

4.2 SAINT Header Bytes 11

5 SAINT2 Configuration 13

5.1 SAINT2 Configuration Commands 13

5.1.1 Firmware Flash Programming (08 10h) 14

5.1.2 Embedded Emulation (08 50h) 14

5.1.3 Request an SD card file's CRC (08 51h) 14

5.1.4 Set Up Periodic Message (08 70h –08 7Fh) 15

5.1.5 SAINT2 Reset (08 80h) 15

5.1.6 Turn Time Stamp On and Off (08 86h and 08 87h) 16

5.1.7 Transmit Echo (08 88h and 08 89h) 16

5.1.8 Send Periodic Message (08 90h and 08 91h) 16

5.1.9 Software Version Request (08 92h) 16

5.1.10 SAINT2 Time Stamp Request (08 93) 16

5.1.11 Clear Operation Warning Code (08 A0h) 17

5.1.12 Retrieve and Report Operation Warning Code (08 A1h) 17

5.1.13 Select Host Communication Channel (08 A2h) 18
5.1.14 Select Serial Bus Protocols (08 A3h) 18
5.1.15 Retrieve Serial Number (08 A5h) 19
5.1.16 Trig Out (08 A8h) 19
5.1.17 Trig In - 1 (08 A9h) 20
5.1.18 Trig In – 2 (08 ABh) 20
5.1.19 Manufacturing Test Command(08 BEh) 22
5.1.20 Saint 2 USB Test Command(08 BFh) 22

5.2 SAINT2 Configuration Reports 23

5.3 SAINT2 Configuration through SD Card 25

5.3.1 Configuration File Requirements 25
5.3.2 Configuration Instructions 25
5.3.3 Config.txt Example 25
5.3.4 Group File Example 26

6 CAN Messages 27

6.1 CAN Commands 27

6.1.1 Configuring CAN Frequency 27
6.1.2 Single Wire Mode Control 28
6.1.3 Listen Only Mode Control 29
6.1.4 CAN Transceiver Control 29
6.1.5 CAN SAINT2 Options 29
6.1.6 Set CAN ID Include Filter 31

6.2 Constructing a CAN Transmit Frame 32

6.3 CAN Channel 1 Bus Flooding Function 33

6.4 Received CAN Frames 34

6.4.1 Header Description 36
6.4.2 CAN Frame Definition Bit Description 36
6.4.3 Message ID Bytes Description 36
6.4.4 Data Bytes Description 36
6.4.5 Completion Code 36
6.4.6 1 ms Resolution 2 Byte Time Stamp 37

7 ISO15765-2 Transport Protocol on CAN 38

7.1 Step 1 – Configure the SAINT2 38
7.1.1 Enable the protocols 38
7.1.2 Configure the CAN channel 38
7.1.3 Configure the SAINT2 for ISO15765-2 38
7.2 Step 2 – Send/Receive the ISO15765-2 message 41
7.3 Step 3 – Evaluate the response code 42
7.4 Clear the ISO15765-2 Configuration 43
7.5 Examples 43
7.5.1 Transmit Examples 43
7.5.2 Receive Examples 45
7.6 Important Notes 48
8 Class 2 49
8.1 Class 2 Messages 49
8.1.1 Class 2 Commands 49
8.1.2 Transmitted Class 2 Messages 49
8.1.3 Received Class 2 Messages 49
9 IIC 52
9.1 Overview 52
9.1.1 IIC Hardware 53
9.2 IIC Commands 54
9.2.1 Configure IIC Master Polling 54
9.2.2 Set Device Address 55
9.2.3 Get IIC Device Address 55
9.2.4 Set IIC Mode 55
9.2.5 Set IIC Baud Rate 55
9.3 IIC Data Messages 55
9.3.1 Request to transmit an IIC message as slave-transmitter 55
9.3.2 Receipt of IIC message as slave receiver 56
9.3.3 Request to transmit an IIC message as master-transmitter 56
9.3.4 Request to receive an IIC message as master-receiver 57

10 Keyword 2000 59

10.1 Keyword 2000 – Application Notes 60

10.1.1 General Operation 60

10.1.2 Initialization 60

10.1.3 Error Reporting 61

10.2 Keyword 2000 Commands 62

10.3 Keyword 2000 Messages 71

10.3.1 Transmitted Keyword 2000 Messages 71

10.3.2 Received Keyword 2000 Messages 72

10.3.3 Format Byte 72

10.3.4 Target Address Byte 73

10.3.5 Source Address Byte 73

10.3.6 Length Byte 73

10.3.7 Data Bytes 73

10.3.8 Checksum Byte 73

11 SPI 74

11.1 Serial Peripheral Interface (SPI) Overview 74

11.2 SPI Hardware Connection Diagram 75

12 LIN 76

12.1 LIN – Application Notes 77

12.2 LIN Commands 79

12.3 LIN Messages 82

12.3.1 Transmitted LIN Messages (Master Frames) 82

12.3.2 Received LIN Messages (frames) 83

12.3.3 PID Byte 83

12.3.4 Data Bytes 83

12.3.5 Checksum Byte 83

12.3.6 Completion Code (Frame Error) Byte 83

13 Block Transfer 85

13.1 Using Block Transfer with the SAINT2 Bus Engine 85

13.2 Using Block Transfer without the SAINT2 Bus Engine....85

14 SAINT2 Gateway Functions....87

14.1 CAN1/CAN2 Gateway....87

14.2 RS232/CAN Gateway....87

15 Connectors....89

15.1 USB Connector....89

15.2 RS-232 Connector....89

15.3 SAINT CABLE....89

16 LEDs....91

16.1 Reset Button - Manual Reset....91

17 Software Packages....92

Revision Log

Version	Revisions	Date Released to Web
1.1	1. Initial release of the document. 2. Document CAN interface 3. Document Class2 interface	March 30, 2006
1.2	1. Corrected and Updated Embedded Periodic SAINT2 Configuration Functions (08 70h – 08 7F)	April 4, 2006
1.3	1. Correct text for SAINT2 Config Command A3h in section 5.1.11 2. Changed CAN Baud Rate configuration values	May 2, 2006
1.4	1. Minor corrections 2. Corrected and updated configuration commands 3. Updated LED definition 4. Updated RS232 and USB interface 5. Updated coversheet of this document	June 27, 2006
1.5	1. Added document of configuration option on system reset 2. Changed documentation to reflect change of CAN High Resolution Time Stamp and Error Indicator bit to be optional.	July 28, 2006
1.6	1. Added documentation for configuration command 08h A4h 2. Changed documentation to reflect change of configuration command 08h A8h and 08h A9h	September 6, 2006
1.7	1. Changed configuration command assignment from 08h A4h to 08h A5h 2. Document IIC interface	September 21, 2006
1.8	1. Noted in the SAINT2 pin out that pins 18 – 21 should not be connected to any wires in a cable. 2. Added warning for bus traffic that exceeds RS232 capability.	October 23, 2006
1.9	1. Document Keyword 2000 interface 2. Modified Config Cmd table to correct omission from ver 1.7 change – assignment change from A4 to A5.	October 24, 2006
2.0	1. Modified sec 9.1 to add notes regarding supported SD card format (FAT-16) & extended init time w/ SD card installed. 2. Modified notes in Sec 9 (Keyword 2000) to indicated version 1.1 hardware required to use KW2K. 3. Added Class2 Error Commands 4. Added S2 CAN option command to disable continuous acknowledgment error retries	October 31, 2006
2.1	1. Document IIC Monitor Interface. 2. Inserted new sec 9.1: Keyword 2000 – Application Notes. a. Added KW2000 initialization sequence details. 3. Modified sec 9.2 (formerly sec 9.1) a. Add new configuration command to set the “Start Comms” message header type (format). b. Changed default for “Timing Error Reporting” command from disabled to enabled.	March 26, 2007
2.2	1. Added IIC Monitor reference for board modification 2. Necessary for external website posting	April 23, 2007
2.3	1. Added reference to “Low Speed CAN” as well as Fault Tolerant CAN 2. Added note that a user should wait 250ms before issuing any command after sending a SAINT2 reset.	June 19, 2007
2.4	1. Update Software Packages 2. Add SPI documentation	

Version	Revisions	Date Released to Web
2.5	Added 82 80 error message information	January 4, 2008
2.6	Extensive modifications to sec 9 (Keyword 2000) a. Added documentation for new "Direct" mode and related commands. b. Expanded the "Application Notes" documentation.	January 28, 2008
2.7	Editted IIC and SPI documentation on recommended practices.	February 29, 2008
2.8	1. Added remote frame information (supported in firmware version 3.6.9+) 2. Added Gateway function descriptions (supported in firmware 3.6.10+) 3. Additional Keyword 2000 changes/enhancements for J2534 Support. 4. Added Trig in command additions 5. Added periodic control messages 6. Added LIN documentation (Initial Release)	May 5, 2008
2.9	1. Updated LIN documentation – Slave Node emulation capability added. 2. Clarify CAN byte descriptions 3. Clarify Trigger In function commands (08 A9) 4. CAN operation warning codes $50 and $58 have been detailed	June 25, 2008
2.10	1. Clarify configuration commands in paragraph headings 2. Add and clarify CAN operational warning definitions 3. Add CAN Rx FIFO overflow error messages 4. Document Ack error retry function and limit changes	Aug 25, 2008
2.11	1. Document CAN bus flooding function 2. Document ISO15765-2 Transport Layer on CAN 3. Clarified group file description in config function section 4. Added information on block transfer ($F8) header 5. Added Operation Warning Code ($08 $A1) error codes 3-6	Feb 20, 2009
2.12	1. Updated/Clarified IIC Monitor Mode Documentation	Mar 10, 2009
2.13	1. Added note for IIC informing user that device must have pullups and provide acks 2. Added SAINT2 configuration commands for embedded emulation function	June 23, 2009
2.14	1. Corrected naming inconsistencies in the "Gateway" instructions 2. Corrected the "SAINT2 Configuration Reports table" in section 5.2 to reflect the proper values for "Retrieve serial number" command that got overlooked in the v1.6 & 1.9 updates (change A4 to A5)	Aug 27, 2009
2.15	1. Added Class2 example to block mode section 2. Corrections to Block Transfer size limits	August 28, 2009
2.16	1. Added clarifications to the KW Initialization operations sections.	September 11, 2009
2.17	1. Added documentation for ISO15765-2 RX function 2. Add the 08 93 request for a timestamp 3. Add the 08 51 command	February 12, 2010
2.18	1. Add configuration messages for HSCAN and FTCAN transceiver mode control	May 19, 2010
2.19	1. Add trigger in and trigger out schematics 2. Add CAN command to set a single CAN include filter 3. Add configurations for ISO15765-2 TP to filter out individual CAN frames	12/9/2010
2.20	1. Add TRIGOUT2 documentation (HW1.2 only) 2. Add SPI_INT – Trigger In (2) documentation 3. Add configuration command for number of ack retries	
2.21	1. Fix minor error in bit fields for ISO-15765 FC options	

1 Introduction

1.1 Scope

This document describes the use and operation of the SAINT2 in normal communication mode. This document cannot be copied in whole or in part without the express written consent of Delphi Automotive Systems Corporation.

1.2 Precedence

This document shall have precedence over any information in any other document. Between reference documents, the document with the later revision date shall have precedence.

1.3 Definitions and Nomenclature

SAINT2 - common nickname for the (<u>S</u>ystems <u>A</u>nalysis <u>IN</u>terface <u>T</u>ool 2).

Host - The computer which communicates to the SAINT2 via RS-232 or USB.

USB - Universal Serial Bus

CAN - Controller Area Network data bus

C2 - Class 2

2 Overview

This document describes the operation of the SAINT2.

The following is synopsis of the SAINT2 features:

- Allows host system to communicate on supported serial busses. Performs timing, access, arbitration, serialization, and error control for all protocols.
- Allows host system to configure the tool with SAINT2 Configuration Commands.
- Provides SAINT2 status with LEDs
- Software is easily updated via a RS232 or USB.

The following is a synopsis of requirements for use of the SAINT2:

- Connection to power, ground, and busses to be monitored
- RS-232 or USB connection to host

3 Host to SAINT2 Connection

3.1 RS-232

3.1.1 RS-232 Set Up

The SAINT2 uses the following RS-232 parameters:

- 8 Data Bits
- 1 Stop Bit
- No Parity

The SAINT2 supported baud rate is 57600. If the serial bus traffic is greater than can be handled by the 57600 RS232 baud rate, the message may be lost or corrupted.

The supported baud rate is sufficient for communicating with the Class2, Keyword, and slower CAN buses. However USB connection is recommended for high speed CAN (over TBD KHz) communications.

3.2 USB

The SAINT2 supports both USB 2.0 and USB 1.1.

4 Message Format

4.1 Data Stream

The host data stream is broken into messages. A message consists of a Message ID and one or more data bytes. The character FFh is used as an ESCAPE character to indicate the end of messages. There are three cases when an ESCAPE character is received.

If the ESCAPE character is followed immediately by a second ESCAPE character, the following are true:

- The message is not yet complete.
- The pair of ESCAPE characters represents a single byte of message data of value FFh.

If the ESCAPE character is followed immediately by a byte of value 00h, the following are true.

- The message is complete.
- Neither the FFh nor the 00h are part of the message.
- No more messages are ready to be sent.

If the ESCAPE character is followed immediately by of any value other than FFh or 00h, the following are true:

- The message is complete.
- Neither the ESCAPE character nor the character following the ESCAPE character are part of the message.
- The value following the ESCAPE character is the message ID for a new message.

This data stream format was chosen to allow arbitrary long messages (i.e. 4K Class 2 data blocks), to minimize the overhead to two bytes per message during peak traffic, and to immediately recognize the end of a message without having to waiting for the next message to start.

4.2 SAINT Header Bytes

The SAINT2 and the host use the SAINT Header byte to identify the type of message that is being sent or received. The header byte is 1 byte long and has the following general format:

SAINT Header Byte Format

bits 7 – 3	bit 2	bit 1	bit 0
Protocol ID	Command	Tx / Rx	Time Stamp Included

See SAINT Protocol Ids table below	= 1: message includes a protocol related SAINT2 Command = 0: message is to be sent to the serial bus	= 1: transmit message = 0: receive message	= 1: 2 byte ms time stamp is included = 0: 2 byte ms time stamp is not included

The Protocol ID, Bits 7 – 3 of the SAINT Header byte, identifies the serial bus protocol for the message being sent or received. The Protocol IDs are defined below:

SAINT Protocol IDs

Not Used (00h)	*UART (80h)
SAINT2 Configuration Command (08h)	*J1850 (88h)
*Keyword 82 (10h)	*BEAN1 (90h)
*Keyword 71 (18h)	*BEAN2 (98h)
IIC (20h)	Not Used (A0h)
Keyword 2000 (28h)	Not Used (A8h)
*IDB (30)	*IE Bus (B0h)
*ACP (38h)	LIN (B8h)
*E&C (40h)	ISO15765-2_CAN1 (C0h)
*J1708 (48h)	ISO15765-2_CAN2 (C8h)
CAN1 (50h)	Not Used (D0h)
CAN2 (58h)	Not Used (D8h)
Class2 (60h)	Not Used (E0h)
*AOS (68h)	Not Used (E8h)
*SPI (70h)	Not Used (F0h)
*Reserved (78h)	Block Transfer (F8h)

*These Ids are reserved for future use.

5 SAINT2 Configuration

5.1 SAINT2 Configuration Commands

The SAINT2 Configuration Commands are used to configure the general functions of the SAINT2.

SAINT2 Configuration Commands

Header	ID	Data	Description	Compatibility
08h	03h	5 Bytes	Send PWM data	Saint 1 Only
08h	04h	3 Bytes	Toggle PWM data	Saint 1 Only
08h	10h	1 Byte	Firmware Flash Programming	Saint 2
08	50h	1-TBD Bytes	Embedded Emulation	SAINT2
08h	50h	1 – 12 Bytes	Request an SD card file’s CRC	SAINT2
08h	70h – 7Fh	1-21 Bytes	Set Up Periodic Message	Saint 2 Only
08h	80h	None	SAINT2 Reset	Saint 1 & 2
08h	86h	None	Turn Time Stamp Information OFF	Saint 1 & 2
08h	87h	None	Turn Time Stamp Information ON	Saint 1 & 2
08h	88h	None	Turn Transmit Echo ON	TBD
08h	89h	None	Turn Transmit Echo OFF	TBD
08h	90h	1-20 Bytes	Send Periodic Message	Saint 1 & 2
08h	91h	None	End Periodic Message	Saint 1 & 2
08h	92h	0-1 Byte	Request Software Version	Saint 1 & 2
08h	93h	3 Bytes	Request A TimeStamp	Saint 2
08h	A0h	None	Clear Operation Warning Code	Saint 2 Only
08h	A1h	None	Retrieve/Report Operation Warning Code	Saint 2 Only
08h	A2h	1 Byte	Select Host Communication Channel	Saint 2 Only
08h	A3h	1-30 Bytes	Select Serial Bus Protocols	Saint 2 Only
08h	A5h	None	Retrieve Serial Number	Saint 2 Only(V2.5)
08h	A8h	1 Bytes	Trig Out function	Saint 2 (V2.5)
08h	A9h	3 Bytes	Trig In function – (1)	Saint 2 (V2.5)
08h	ABh	3 Bytes	Trig In function – (2)	Saint 2 (V3.18)
08h	BEh	1 Byte	Manufacturing Test Command	Saint 2
08h	BFh	1 Byte	Not A User Command	Saint 2
08h	F0h	D1-D12	Enable Saint 1 Trigger	TBD
08h	F1h	None	Disable Saint 1 Trigger	TBD

5.1.1 Firmware Flash Programming (08 10h)

This command places the SAINT2 into a mode in which the SAINT2 firmware can be reflashed. A reset command terminates this mode. For more information see TBD.

5.1.2 Embedded Emulation (08 50h)

These commands are used to interface with the SAINT2's embedded emulation function. See the SAINT2 Emulation User Guide for more information.

Host / Emulation Interface Messages

SAINT2 header	ID	Data	Description
08	50 00	ASCII file name	Start execution of emulation file
08	50 02	message data	Message sent from emulation to host
08	50 03	Message data	Message sent from host to emulation
08	50 04	emulation error information	Message indicating an error in emulation execution
08	50 05	Emulation status information	Message indicating the status of the emulation execution
08	50 06	Emulation variable value	Message indicating the value of a requested emulation variable

5.1.3 Request an SD card file's CRC (08 51h)

This command returns the 32bit CRC of a file residing on the SD card.

Host to SAINT2 request: 08 51 XX XX … XX

Data Byte	Description
XX XX	Emulation filename in ASCII (capital letters, 8.3 format)

SAINT2 response: 08 51 00 YY YY YY YY or 08 51 EE ZZ

<table>
<tr><td>Data Byte</td><td>Description</td></tr>
<tr><td>YY YY YY YY</td><td>32 Bit CRC of file</td></tr>
<tr><td>ZZ</td><td><table><tr><td>Error Value</td><td>Description</td></tr><tr><td>0x00</td><td>Emulation is executing or CRC calculation is executing</td></tr></table></td></tr>
</table>

	0x01	Undefined SD card error
	0x02	No such file or directory – SD card
	0x05	I/O Error – SD card
	0x09	Bad file number – SD card
	0x0D	Permission denied – SD card
	0x11	File Exists – SD card
	0x13	No such device – SD card
	0x16	Invalid Argument – SD card
	0x18	Too many files open – SD card
	0x1C	No space left on device – SD card
	0x1E	Read only file system (Sharing error) – SD card
	0x20	Buffer is busy

5.1.4 Set Up Periodic Message (08 70h –08 7Fh)

These commands allow the user to set up and transmit up to 16 periodic messages from the SAINT2 hardware. The following table describes the command formats.

Configuration Command	Description
$08 7X 00 T1 T2 HH YY YY YY …	Turn periodic message 7X ON T1 = MSB of 1ms/bit periodic time T2 = LSB of 1 ms/bit periodic time HH = SAINT2 Header (i.e. $60 for Class2, $50 for CAN) YY = Serial Message (1 to 15 bytes)
$08 7X 00	Turn message 7X ON after it has already been configured
$08 7X 01	Turn periodic message 7X OFF
$08 7X 10	Delete periodic message 7X
$08 7X 20	Delete all periodic messages that have been turned OFF
$08 7X 30	Delete all periodic messages
$08 7X 40	Request the configuration of periodic message 7X
$08 7X 50	Request the configuration of all periodic message that are OFF
$08 7X 60	Request the configuration of all periodic messages that are ON
$08 7X 70	Request both ONed and OFFed periodic message setting
$08 7X 80	Turn ON all periodic message that are OFF
$08 7X 90	Turn OFF all periodic messages that are ON

- The range of X in 08 7X is 0 to F.
- A Time Stamp of 0000h will cause the message to be sent one time and then turned OFF.
- If two messages align to be transmitted at the same time, the message with the lower ID will have priority.

5.1.5 SAINT2 Reset (08 80h)

This command will cause the Saint 2 to do a software reset. A user should wait approximately 250ms after issuing a SAINT2 Reset to send any other commands to the SAINT2.

5.1.6 Turn Time Stamp On and Off (08 86h and 08 87h)

These commands control whether or not a 16-bit time stamp is appended to the end of the bus message reports.

5.1.7 Transmit Echo (08 88h and 08 89h)

These commands control whether or not transmitted messages are echoed back to the host when they are sent. These commands are not implemented in Saint 2. *Echoed back messages will have their transmit bit set.*

5.1.8 Send Periodic Message (08 90h and 08 91h)

These commands allow the user to set up and transmit a single periodic messages. Use the following format:

08 90 T1 T2 HH XX XX XX XX XX XX XX XX XX XX XX

where

T1 = MSB of 1ms periodic time

T2 = LSB of 1ms periodic time

HH = SAINT2 Header

XX… = Serial Message

The 08 91 messages cancels the periodic message.

5.1.9 Software Version Request (08 92h)

Command 08h 92h requests the SAINT2 firmware functional block version (ASCII). If the RS232 host communication protocol is disabled, the command 08h 92h 01h requests the version of all blocks of the SAINT2 firmware (boot, re-flash, functional, and user block) (ASCII).

5.1.10 SAINT2 Time Stamp Request (08 93)

Command 08 93 requests the SAINT2's current 2 byte Time Stamp Value.

Request:

Header	ID	1 byte "marker"	Description
08	93	XX	Request 2 byte Time Stamp

Response:

Header	ID	1 byte "marker"	Data

08	93	XX	2 byte Time Stamp

5.1.11 Clear Operation Warning Code (08 A0h)

This command will clear the operation warning codes stored in the SAINT2.

5.1.12 Retrieve and Report Operation Warning Code (08 A1h)

ID A1h has two functions. As a command 08h A1h retrieves the operation warning codes stored in the SAINT2. The return message has 0 - 32 bytes of data representing the operation warning codes. The warning code is defined in the following table. As an unsolicited report message, it reports the operation warning code in the format of 08h A1h XXh. XXh is a one byte warning code defined also in the following table.

Header	ID	Data	Description
08h	A1h	01h	Loop overrun detected
08h	A1h	02h	USB buffer full detected
08h	A1h	03h	Escape sequence error
08h	A1h	04h	Message Too Long
08h	A1h	05h	Message Buffer Full
08h	A1h	06h	Message Truncated
08h	A1h	08h-0Fh	Configuration setting warning
08h	A1h	10h-17h	Keyword 82 operation warning
08h	A1h	18h-1Fh	Keyword 71 operation warning
08h	A1h	20h-27h	IIC operation warning
08h	A1h	28h-2Fh	Keyword 2000 operation warning
08h	A1h	30h-37h	IDB operation warning
08h	A1h	38h-3Fh	ACP operation warning
08h	A1h	40h-47h	E&C operation warning
08h	A1h	48h-4Fh	J1708 operation warning
08h	A1h	50h	The CAN1 TX buffer has overrun and CAN frames have been discarded
08h	A1h	51h	The CAN1 RX buffer has overrun and CAN frames have been discarded
08h	A1h	52h-57h	reserved for CAN 1
08h	A1h	58h	The CAN2 TX buffer has overrun and CAN frames have been discarded
08h	A1h	59h	The CAN2 RX buffer has overrun and CAN frames have been discarded
08h	A1h	5Ah-5Fh	reserved for CAN 2
08h	A1h	60h-67h	Class 2 operation warning

08h	A1h	68h-6Fh	AOS operation warning
08h	A1h	70h-77h	SPI operation warning
08h	A1h	78h-7Fh	Not Used
08h	A1h	80h-87h	UART operation warning
08h	A1h	88h-8Fh	J1850 operation warning
08h	A1h	90h-97h	BEAN1 operation warning
08h	A1h	98h-9Fh	BEAN 2 operation warning
08h	A1h	A0h-A7h	Not Used
08h	A1h	A8h-AFh	Not Used
08h	A1h	B0h-B7h	IE bus operation warning
08h	A1h	B8h-BFh	LIN operation warning
08h	A1h	C0h	CAN packet buffer busy – rx message dropped (ISO15765-2)
08h	A1h	C1h	TX FIFO full – CAN packet message dropped (can't send FC – ISO15765-2)
08h	A1h	C2h-C7h	reserved
08h	A1h	C8h-CFh	reserved
08h	A1h	D0h-D7h	Not Used
08h	A1h	D8h-DFh	Not Used
08h	A1h	E0h-E7h	Not Used
08h	A1h	E8h-EFh	Not Used
08h	A1h	F0h-F7h	Not Used
08h	A1h	F8h-FFh	reserved

5.1.13 Select Host Communication Channel (08 A2h)

This command selects the Saint 2 to host communication protocol to be used.

Select Host Communication Protocol Command

Header	ID	Data	Description
08h	A2h	03h	Select RS232 Only
08h	A2h	0Ch	Select USB Only
08h	A2h	0Fh	Select Both RS232 and USB

5.1.14 Select Serial Bus Protocols (08 A3h)

This command selects the serial bus protocols that will be enabled. Each byte in data field represents a protocol that is being enabled. Any protocol that is not in the list will be disabled.

Example: 08 A3 50 60 58 enables CAN1, Class 2, and CAN2 vehicle buses.

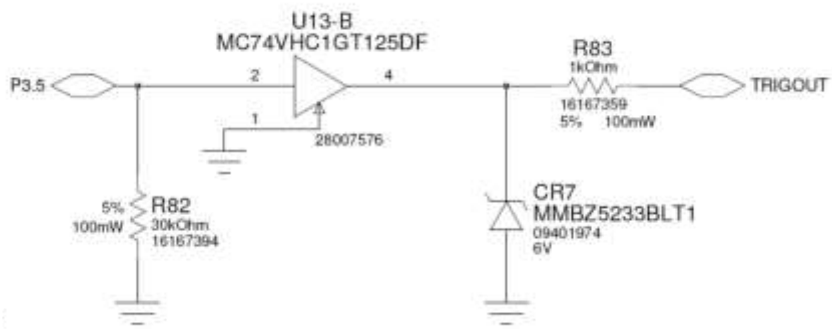
Note: The configuration command message is always enabled and not affected by this command.

5.1.15 Retrieve Serial Number (08 A5h)

This command retrieves serial number of Saint 2 unit. Return message with 6 data bytes – 6 charecter serial number in hexadecimal ASCII form.

5.1.16 Trig Out (08 A8h)

This command controls the TRIGOUT Pin (DB-25 connector pin number 24) operation. The command has a format of 08h A8h option_byte. Option byte has a bit by bit definition



Option Byte Bit Definition

Option byte	Value	Description
Bit 0 – Bit 1	00 01 10 11	No action Set TRIGOUT line low (0 volt) Set TRIGOUT line high (5 volt) Toggle TRIGOUT line
Bit 2	TBD	TBD
Bit 3	TBD	TBD
Bit 4 – Bit 5	00 01 10 11	No action Set TRIGOUT2 line low (0 volt) Set TRIGOUT2 line high (5 volt) Toggle TRIGOUT2 line (only in HW 1.2)
Bit 6	TBD	TBD
Bit 7	TBD	TBD

5.1.17 Trig In - 1 (08 A9h)

This command controls and reports TRIGIN Pin (DB-25 connector pin number 4) operation. A Trig In command must be sent to configure the SAINT2 to detect and report a change on the TrigIn pin.

5.1.17.1 Trigger In -- State Change

This command configures the Saint 2 firmware to send a trigger-in-marker message to the host when the state of the Trigger In pin changes:

Header	ID	Option byte	2 byte "marker"	Description
08	A9	01	XX XX	Configure SAINT2 to indicate trigger in pin state change

With this configuration, the SAINT2 firmware will send the following message to the host when it detects a state change on its trigger in pin:

Header	ID	Option byte	2 byte "marker"	Description
08	A9	01	XX XX	State change on trigger in pin

5.1.17.2 Trigger in -- edge detection marker

This command causes the Saint 2 firmware to send a trigger-in-marker message with edge detection information to the host when a trigger in signal transition is detected.

Header	ID	Option byte	2 byte "marker"	Description
08	A9	02	XX XX	Configure SAINT2 to indicate trigger in pin edge detected

With this configuration, the SAINT2 firmware will send one of the following messages to the host when it detects a state change on its trigger in pin

Header	ID	Option byte	Edge	2 byte "marker"	Description
08	A9	02	AA	XX XX	Rising edge detected on trigger in pin
08	A9	02	FF	XX XX	Falling edge detected on trigger in pin

5.1.18 Trig In – 2 (08 ABh)

This command controls and reports the state of the SPI_INT Pin (DB-25 connector pin number 22) operation. A Trig In command must be sent to configure the SAINT2 to detect and report a change on the SPI_INT pin.

5.1.18.1 Trigger In -- State Change

This command configures the Saint 2 firmware to send a trigger-in-marker message to the host when the state of the SPI_INT pin changes:

Header	ID	Option byte	2 byte "marker"	Description
08	AB	01	XX XX	Configure SAINT2 to indicate SPI_INT pin state change

With this configuration, the SAINT2 firmware will send the following message to the host when it detects a state change on its SPI_INT pin:

Header	ID	Option byte	2 byte "marker"	Description
08	AB	01	XX XX	State change on SPI_INT pin

5.1.18.2 Trigger in -- edge detection marker

This command causes the Saint 2 firmware to send a trigger-in-marker message with edge detection information to the host when a SPI_INT signal transition is detected.

Header	ID	Option byte	2 byte "marker"	Description
08	AB	02	XX XX	Configure SAINT2 to indicate SPI_INT pin edge detected

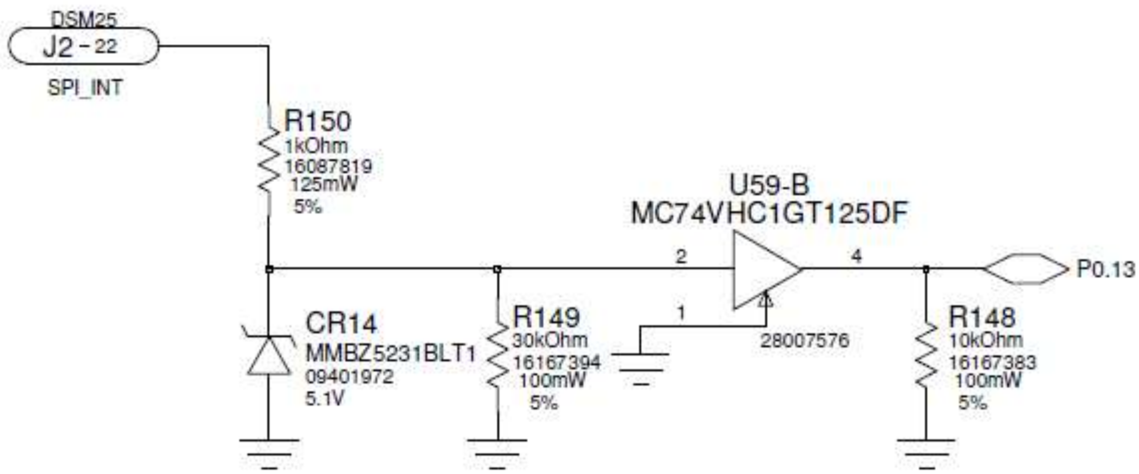
With this configuration, the SAINT2 firmware will send one of the following messages to the host when it detects a state change on its trigger in pin

Header	ID	Option byte	Edge	2 byte "marker"	Description
08	AB	02	AA	XX XX	Rising edge detected on SPI_INT pin
08	AB	02	FF	XX XX	Falling edge detected on SPI_INT pin

5.1.18.3 Trigger in – disable detection

This command causes the Saint 2 firmware to disable its trigger in pin detection:

Header	ID	Option byte	2 byte "marker"	Description
08	AB	00	00 00	Disable SPI_INT pin detection



5.1.19 Manufacturing Test Command(08 BEh)

This command controls the manufacturing test. The data byte value should match the pin number on the Saint 2 DB25 connector. Manufacturing test is only performed and valid when a special manufacturing test cable attached to the Saint2.

Manufacturing Test Command

Header	ID	Data
08h	BEh	SAINT2 pin #

Manufacturing Test Command Response

Header	ID	Data	Description
08h	BEh	FFh	Test Failed
08h	BEh	FEh	Test not defined
08h	BEh	FDh	Test in progress
08h	BEh	FCh	Illegal command
08h	BEh	Other #	Test passed on the # of the pin

5.1.20 Saint 2 USB Test Command(08 BFh)

This is not a user command.

5.2 SAINT2 Configuration Reports

The SAINT2 will report a Reset whenever one occurs. Most likely causes of reset are power on reset, host commanded reset, or SAINT2 micro watchdog error. The SAINT2 does a cold start on all resets (all information is lost). The SAINT2 will also report error conditions with the Report Error message below. See table T5-1 for error code definitions.

The SAINT2 will also echo back all system commands.

Header	ID	Description	Data Bytes
08h	70h – 7Fh	Report Set Up Periodic Message	1-21 bytes echoed
08h	80h	Reset Occurred	None
08h	82h	Report Error	1 byte Error Code (See T5-1)
08h	86h	Report Time Stamp Information OFF	None
08h	87h	Report Time Stamp Information ON	None
08h	88h	Report Transmit Echo ON	None
08h	89h	Report Transmit Echo OFF	None
08h	90h	Report Send Periodic Message ON	1-20 bytes echoed
08h	91h	Report Send Periodic Message OFF	None
08h	92h	Report SWID	1-N bytes of ASCII Data
08h	A0h	Report operation warning code cleared	None
08h	A1h	Report operation warning code	0-32 bytes warning code
08h	A2h	Report host communication channel	1 byte echoed
08h	A3h	Report Bus Protocol Selected	1-30 bytes protocol ID(See Section 4-2)
08h	A5h	Retrieve serial number	6 byte serial number ASCII code
08h	A8h	Report TriggerOut	None
08h	A9h	Report TriggerIn	2 byte marker
08h	BEh	Report Manufacturing Test Status	1 byte(See Section 5.1.15)
08h	BFh	N/A	None
08h	F0h	Report SAINT1 Trigger ON	1-12 bytes of Trigger Data
08h	F1h	Report SAINT1 Trigger OFF	None

T5-1 SAINT2 Configuration Command Error Codes

Code	Description
80h	System – Invalid message ID
81h	System – RS-232 Transmit Buffer Full
82h	System – UART Error (Overrun, etc)
83h	System – Received message length error
84h	System – Have not read previous message

5.3 SAINT2 Configuration through SD Card

The SAINT2 can be configured at reset by using an SD card. Following reset, the SAINT2 searches the SD card root directory for the file ***config.txt***. If ***config.txt*** is found, the SAINT2 executes the configuration instructions in ***config.txt***.

NOTES: 1) Currently, ONLY the FAT-16 format is supported (SD cards formatted as FAT-32 will not work.)
2) Initialization (reset/boot) time with an SD card inserted can take up to 30sec (nominally 15sec) due to additional integrity checks that are performed on the SD card media.

5.3.1 Configuration File Requirements

- Text format file
- Must be saved in root directory of SD card
- The first line of config.txt must be 'begin' and last line must be 'end'
- Configuration instructions are between 'begin' and 'end' with one instruction per line
- Comments are allowed at end of instruction and begin with a semicolon.
- The maximum length of each line is 56 characters.

5.3.2 Configuration Instructions

5.3.2.1 'can channel swap' Instruction

This instruction swaps the CAN hardware channel mapping between CAN 1 and CAN 2. Before swapping CAN_1 ID 0x50 is assigned to hardware channel 1 and CAN_2 ID 0x58 is assigned to hardware channel 2. After swapping, 0x50 is assigned to channel 2 and 0x58 assigned to channel 1.

5.3.2.2 'group file' Instruction

Group file instruction are followed by a colon and a group file name. After the SAINT2 reads the command, it executes the group file immediately if the group file exists in SD card root directory. Group file format is described in the Saint Document and has the following restrictions:

- The group file name cannot be longer than 8+3 characters.
- The maximum length of the group file is 16 lines with a maximum of 31 data bytes in each line.
- The time field in group file is ignored. Firmware executes group file messages sequentially and immediately.
- Comments are not allowed.
- System reset command 08 80 is not allowed.

5.3.3 Config.txt Example

begin

can channel swap

group file: playback.grp ; execute group file at system rest

end

5.3.4 Group File Example

A SAINT2 group file contains a list of messages and an associated transmit delay from the previous message. The delay must be a 4-digit decimal value in milliseconds from the previous message. The message must follow the SAINT2 Programmer's Reference format (i.e., the message must include the SAINT2 header byte). Group files may be used to send any message (within the allowable length) that may be sent from the host PC to the SAINT2. **IMPORTANT: Note that group files called from a config.txt file will be executed without timing control or guarantee of synchronous execution. These group files should only be used to configure the SAINT2 hardware. They should not be used to send messages onto a serial bus, send a SWCAN high voltage wakeup, or send any other function that requires synchronous execution or timing control.**

Group File Example:

0000 54 01 C9 39

0010 54 04 00

6 CAN Messages

The SAINT2 supports 2 independent CAN nodes. CAN_1 (SAINT2 header 50h) and CAN_2 (SAINT2 header 58h) may each be independently configured to support dual wire CAN, single wire CAN, and fault tolerant CAN at a user specified baud rate.

SAINT2 CAN Node Information

CAN Function	Connector Label	SAINT CABLE Pin #	Message Header	Command Header
Single Wire CAN 1	SWCAN1	1	50h	54h
Single Wire CAN 2	SWCAN2	5	58h	5Ch
Dual Wire, High Speed CAN 1	CAN_1_H CAN_1_L	8 9	50h	54h
Dual Wire, High Speed CAN 2	BArT_CANH BArT_CANL	10 11	58h	5Ch
Fault Tolerant CAN 1 (Low Speed CAN)	FT_1_H FT_1_L	Only on daughter board	50h	54h
Fault Tolerant CAN 2 (Low Speed CAN)	FT_2H FT_2L	2 13	58h	5Ch

6.1 CAN Commands

The CAN commands are used to configure the operation of the SAINT2's CAN nodes.

CAN CommandFormat

Header	ID	Description	Data Bytes
54h or 5Ch	01h	Set CAN Frequency	BTR0 BTR1
54h or 5Ch	02h	Set Single Wire CAN Mode Control	MODE
54h or 5Ch	03h	Set CAN Controller to listen only mode	RXONLY
54h or 5Ch	04h	Set CAN Transceiver	TXVR MODE
54h or 5Ch	05h	Set SAINT2 options	OPTION
54h or 5Ch	09h	Set CAN ID include filter	CAN ID MASK
54	FFh	Bus Flooding	MODE CAN MSG

6.1.1 Configuring CAN Frequency

By modifying BTR0 and BTR1, you can change the frequency, Synchronization Jump Width, and Sampling point, of the TC1130 MultiCAN controller.

Synchronization Jump Width = sjw = 1 + SJW, SJW = bit 7 – bit 6 of BTR0

Baud Rate Prescaler = brp = 1 + BRP, BRP = bit5 – bit0 of BTR0

tseg2 = 1 + TSEG2, TSEG2 = bit6 – bit4 of BTR1

tseg1 = 1 + TSEG1, TSEG1 = bit3 – bit0 of BTR1

DIV8 = bit7 of BTR1 (0: tq = BPR+1 clock cycles; 1: tq = 8*(BPR+1) clock cycles)

$$\textbf{Baud Rate} = \frac{75.0\text{+E6}}{(DIV8*7 + 1) * (BRP + 1) * (3 + TSEG1 + TSEG2)}$$

$$\textbf{Bit Time} = \frac{1}{\textbf{Baud Rate}}$$

$$\textbf{Sample Point} = \frac{2 + TSEG1}{3 + TSEG1 + TSEG2}$$

Example Set CAN Frequency Configuration Messages

Header	ID	Data Bytes	CAN Parameter Value	Baud Rate	Sample Point
54h or 5Ch	01h	8D AF	SJW=2, BRP=13, TSEG1 = 15, TSEG2=2, DIV8=1	33,482(33,333)	85%
54h or 5Ch	01h	B1 2D	SJW=2, BRP=49, TSEG1 = 13, TSEG2=2, DIV8=0	83,333	83.3%
54h or 5Ch	01h	F0 3A	SJW=3, BRP=48, TSEG1 = 10, TSEG2=3, DIV8=0	95,200	75%
54h or 5Ch	01h	F1 39	SJW=3, BRP=49, TSEG1 = 9, TSEG2=3, DIV8=0	100,000	73.3%
54h or 5Ch	01h	DD 3E	SJW=3, BRP=29, TSEG1 = 14, TSEG2=3, DIV8=0	125,000	80%
54h or 5Ch	01h	D8 39	SJW=3, BRP=24, TSEG1 = 9, TSEG2=3, DIV8=0	200,000	73.3%
54h or 5Ch	01h	CE 3E	SJW=3, BRP=14, TSEG1 = 14, TSEG2=3, DIV8=0	250,000	80%
54h or 5Ch	01h	C9 39	SJW=3, BRP=9, TSEG1 = 9, TSEG2=3, DIV8=0	500,000	73.3%

6.1.2 Single Wire Mode Control

The following commands allow the user to control the MODE0 and MODE1 pins of the Single Wire CAN transceiver (MC33897).

Set Single Wire Mode Command

Header	ID	Data Byte	Description
54h or 5Ch	02h	00	Sleep mode

54h or 5Ch	02h	01	High-speed transmission mode
54h or 5Ch	02h	02	Wake-up transmission mode
54h or 5Ch	02h	03	Normal transmission mode

6.1.3 Listen Only Mode Control

The following commands allow the user to control the Listen Only mode of the TC1130 MultiCAN controller. When the CAN Node is in the listen only mode, it will not transmit any frame onto the CAN bus, including active error frames and frame acknowledgments.

Set Listen Only Mode Command

Header	ID	Data Byte	Description
54h or 5Ch	03h	00	CAN Node will TX and RX
54h or 5Ch	03h	01	CAN Node will only RX

6.1.4 CAN Transceiver Control

The following commands allow the user to control the CAN transceiver connected to either CAN Node 1 or CAN Node 2 of the TC1130. The optional 2nd data byte also allows the user to configure any associated operational modes with the selected CAN transceiver.

Set CAN Transceiver Command

Header	ID	TXVR	MODE (optional)		Description
54h or 5Ch	04h	00	00	Normal Mode	High Speed/Dual Wire CAN (TJA1040)
			01	Standby Mode	
54h or 5Ch	04h	01	01	Sleep Mode	Fault Tolerant CAN (TJA1054A)
			03	Normal Mode	
54h or 5Ch	04h	02	00	Sleep Mode	Single Wire CAN (MC33897)
			01	High Speed Mode	
			02	High Voltage Mode	
			03	Normal Mode	
54h or 5Ch	04h	03	No optional mode supported		CAN transceiver on a daughter board

6.1.5 CAN SAINT2 Options

The following command allows the user to enable or disable SAINT2 options. Whether an option is set using a 54h or 5Ch header, the option will apply to both CAN1 and CAN2. There are currently two SAINT2 options available.

6.1.5.1 Error Indicator Option Bit

The SAINT2 can set a bit in the unused bits of the CAN message ID to indicate an error in a frame. The error information is always contained in the status byte that precedes the time stamp, but the additional error indicator bit allows for easier filtering of error frames. If this option is enabled, a frame that contains an error will have bit 6 set in the MSB of the Message ID.

Example: 51 40 00 01 E6 78

|

Error indicator bit set in Message ID of receive error

Example: 51 46 21 11 22 33 44 55 03 E6 78

|

Error indicator bit set in Message ID of transmit error

6.1.5.2 High Resolution Time Stamp Option Bit

The SAINT2 can provide a high resolution, relative time stamp. If this bit is set, the SAINT2 will insert a high resolution time stamp bit after the frame data and before the status byte of the frame. The high resolution time stamp can be used in conjunction with the 2 byte time stamp determine a high resolution time difference between two messages. The following equations can be used to calculate the high resolution delta time:

- Delta time = (((High Res TS2 – High Res TS1) * 2^5) + (X * 2^{12}))
- Where X = number of times the High Resolution Time Stamp has rolled over
- Rollover time = 2^{13} * bittime
- Resolution = 2^5 * bittime
- bittime = ((DIV8*7 + 1) * (BRP + 1) * (3 + TSEG1 + TSEG2)) / 75M - use actual values not general baud rate

Example: 51 06 21 11 22 33 44 55 A4 10 E6 78

51 06 21 11 22 33 44 55 F6 10 E6 78

|

High Resolution Time Stamp

6.1.5.3 Disable Continuous Ack Error Message Retries

In the normal operation of the CAN physical layer, a transmitted frame is not completed without error until a receiver has acknowledged the frame by setting the acknowledgment bit in the transmitted frame. By default, when an acknowledgment has not been received, the SAINT2 will attempt to resend the frame up to 4 times. There may be circumstances during product testing when this is not desirable. The SAINT2 can be configured in 3 different ways with regards to unacknowledged, transmitted CAN frames:

1) Default – The SAINT2 will attempt a CAN frame 5 times. (54 05 00)
2) The SAINT2 will only try a CAN frame one time. (54 05 01)
3) The user can configure the SAINT2 to retry a specified number of times. (54 05 00 XX XX XX XX)

If no ACK is received in the number of configured retries, the SAINT2 will flush and restart its transmit fifo. Note that this means that any frames that have been sent to the SAINT2 to be transmitted that have already been placed in the fifo will be lost.

6.1.5.4 Setting CAN SAINT2 Options

Use the following message to set the CAN SAINT2 Options.

Set CAN SAINT2 Options

Header	ID	Option	# of Ack Retries (optional)	Description
54h or 5Ch	05h	OPT	XX XX XX XX (1 retry/bit)	Set SAINT2 Options as described below

OPT bit definition

OPT	Description
b7 – b3	Reserved
b2	= 0: continous retries are enabled for ACK errors = 1: disable continous retries for ACK errors
b1	= 0: disable error indicator option bit = 1: enable error indicator option bit
b0	= 0: disable high resolution time stamp option bit = 1: enable high resolution time stamp bit

6.1.6 Set CAN ID Include Filter

The following command allows the user/plug-in to set the SAINT2's CAN controller's CAN ID acceptance filter. When this filter is specified, the SAINT2's CAN controller will only receive messages that match the CAN ID anded with the mask. Only the received messages will be sent from the SAINT2 to the host PC over USB/RS232. Using this command may be very helpful for PC applications that are overburned when trying to process all of the messages on a vehicle bus. This only filters messages received from the CAN bus. All messages transmitted by the SAINT2 onto the bus will be sent to the host.

Set CAN ID Include Filter

Header	ID	CAN ID	Mask	Description
54h or 5Ch	09h	XX XX	MM MM	Set an 11 bit CAN ID include filter
54h or 5Ch	09h	XX XX XX XX	MM MM MM MM	Set a 29 bit CAN ID include filter

54h or 5Ch	09h	00	n/a	Reset filter to include all CAN IDs

Examples:

1) To include only CAN ID 0x7E0 send a 54 09 07 E0 FF FF.
2) To include CAN IDs 0x7E0 to 0x7EF send a 54 09 07 E0 FF F0.

6.2 Constructing a CAN Transmit Frame

The following message formats are used to transmit a CAN frame onto the serial bus.

11 Bit Message ID CAN Data Frame Message Format

Byte 1	Byte 2: b7-b3	Byte 2: b2-b0 and Byte 3	Bytes 4-11
Header	CAN frame definition bits	11 bit Identifier (000h – 7FFh)	up to 8 Data Bytes
50h or 58h	00000b	$b_{10} – b_0$	D1 D2 D3 D4 D5 D6 D7 D8

29 Bit Message ID CAN Data Frame Message Format

Byte 1	Byte 2: b7-b5	Byte 2: b4-b0 and Byte 3	Bytes 6-13
Header	CAN frame definition bits	29 bit Identifier (00000000h – 1FFFFFFFh)	up to 8 Data Bytes
50h or 58h	100b	$b_{28} – b_0$	D1 D2 D3 D4 D5 D6 D7 D8

11 Bit Message ID CAN Remote Frame Message Format

Byte 1	Byte 2: b7-b3	Byte 2: b2-b0 and Byte 3	Byte 4
Header	CAN frame definition bits	11 bit Identifier (000h – 7FFh)	DLC (0 – 8)
50h or 58h	00100b	$b_{10} – b_0$	D1

29 Bit Message ID CAN Remote Frame Message Format

Byte 1	Byte 2: b7-b5	Byte 2: b4-b0 and Byte 3	Byte 6
Header	CAN frame definition bits	29 bit Identifier (00000000h – 1FFFFFFFh)	DLC (0 – 8)
50h or 58h	101b	$b_{28} – b_0$	D1

Note: For extended Identifiers, bit7 of byte2 must be set.

Note: For remote frames, bit 5 of byte 2 must be set.

6.3 CAN Channel 1 Bus Flooding Function

The SAINT2 can be used to flood the CAN bus using CAN channel 1. The SAINT2 will send the specified frame onto the CAN bus as quickly as it is able. The specified frame may have an incremented data byte allowing for 256 unique messages to be transmitted as the bus is being flooded.

Set CAN 1 Bus Flooding

Header	ID	Data Byte 1	Data Bytes 2 – up to Data Byte 13
54h	FFh	MODE	CAN message

MODE Byte Definition

MODE	Description
00	Disable Bus Flooding
01	Enable Bus Flooding with CAN message in data bytes 2 - 13
02	Enable Bus Flooding with CAN message in data bytes 2 – 13 and increment the last data byte value with each transmitted frame

Examples:

54 FF 01 01 22 11 22 33 44 00 will configure the SAINT2 to transmit the following:

$122 11 22 33 44 00

$122 11 22 33 44 00

$122 11 22 33 44 00

etc

54 FF 02 01 22 11 22 33 44 00 will configure the SAINT2 to transmit the following:

$122 11 22 33 44 00

$122 11 22 33 44 01

$122 11 22 33 44 02

etc

54 FF 01 81 22 33 44 11 22 00 will configure the SAINT2 to transmit the following:

$01223344 11 22 00

$01223344 11 22 00

$01223344 11 22 00

etc

54 FF 02 81 22 33 44 11 22 00 will configure the SAINT2 to transmit the following:

$01223344 11 22 00

$01223344 11 22 01

$01223344 11 22 02

etc

6.4 Received CAN Frames

The SAINT2 reports back to the host the CAN frames that it receives from the CAN bus. These reported frames include frames that the SAINT2 has transmitted onto the bus and frames that it has received from other devices on the CAN bus. The frames are reported back to the host in messages with the following formats.

11 Bit Message ID CAN Data Frame with 2 Byte Time Stamp

Byte 1	Byte 2:b7-b3	Byte 2:b2-b0, Byte 3	Bytes 4—(3+*N*)	Byte 4+*N*	Byte (5+*N*) – Byte (6+*N*)
Header	00000b	Message ID Bytes	*N* Data Bytes	Completion Code	1ms resolution Time Stamp

29 Bit Message ID CAN Data Frame with 2 Byte Time Stamp

Byte 1	Byte 2:b7-b5	Byte 2:b4-b0, Byte 5	Bytes 6—(5+*N*)	Byte 6+*N*	Byte (7+*N*) – Byte (8+*N*)
Header	100b	Message ID Bytes	*N* Data Bytes	Completion Code	1ms resolution Time Stamp

11 Bit Message ID CAN Data Frame without 2 Byte Time Stamp

Byte 1	Byte 2:b7-b3	Byte 2:b2-b0, Byte 3	Bytes 4—(3+*N*)	Byte 4+*N*
Header	00000b	Message ID Bytes	*N* Data Bytes	Completion Code

29 Bit Message ID CAN Data Frame without 2 Byte Time Stamp

Byte 1	Byte 2:b7-b5	Byte 2:b4-b0, Byte 5	Bytes 6—(5+*N*)	Byte 6+*N*
Header	100b	Message ID Bytes	*N* Data Bytes	Completion Code

11 Bit Message ID CAN Remote Frame with 2 Byte Time Stamp

Byte 1	Byte 2:b7-b3	Byte 2:b2-b0, Byte 3	Byte 4	Byte 5	Byte 5-6
Header	00100b	Message ID Bytes	DLC	Completion Code	1ms resolution Time Stamp

29 Bit Message ID CAN Remote Frame with 2 Byte Time Stamp

Byte 1	Byte 2:b7-b5	Byte 2:b4-b0, Byte 5	Byte 6	Byte 7	Byte 8-9
Header	101b	Message ID Bytes	DLC	Completion Code	1ms resolution Time Stamp

11 Bit Message ID CAN Remote Frame without 2 Byte Time Stamp

Byte 1	Byte 2:b7-b3	Byte 2:b2-b0, Byte 3	Byte 4	Byte 5
Header	00100b	Message ID Bytes	DLC	Completion Code

29 Bit Message ID CAN Remote Frame without 2 Byte Time Stamp

Byte 1	Byte 2:b7-b5	Byte 2:b4-b0, Byte 5	Byte 4	Byte 5
Header	101b	Message ID Bytes	DLC	Completion Code

6.4.1 Header Description

Header

Header	Description: See definition of SAINT2 header bits in TBD
50h or 58h	CAN Frame Received from bus, no 2 byte time stamp
51h or 59h	CAN Frame Received from bus with 2 byte time stamp
52h or 5Ah	CAN Frame Transmitted onto bus, no 2 byte time stamp
53h or 5Bh	CAN Frame Transmitted onto bus with 2 byte time stamp

6.4.2 CAN Frame Definition Bit Description

CAN Frame Definition Bits

CAN Frame Definition bits	**Description**
b7	=0 for 11 bit ID frame =1 for 29 bit ID frame
b6	=0 for no error in frame (optional) =1 for error in frame (optional)
b5	=0 for data frame =1 for remote frame

6.4.3 Message ID Bytes Description

Message ID Bytes

ID Type	**Allowed Values**
11 bit ID	000h – 7FFh
29 bit ID	00000000h – 1FFFFFFFh

6.4.4 Data Bytes Description

CAN allows 0 to 8 data bytes per frame.

6.4.5 Completion Code

The completion code contains information from the CAN controller on the TC1130. It is used to communicate whether or not the frame was successfully transmitted or received. It identifies the error if the communication was not successful.

Completion Code Definition

Completion Code Byte	Description
0x00	reserved
0x01	Stuff error – More than 5 equal bits in a sequence have occurred in a part of a received frame where this is not allowed.
0x02	Form error – A 'fixed format part' of a received frame has the wrong format.
0x03	Ack error – The transmitted frame was not acknowledged by another node.
0x04	Bit 1 error – During a frame transmission the CAN node tried to send a recessive level (1) outside the arbitration field and the acknowledge slot, but the monitored bus value was dominant.
0x05	Bit 0 error – 1) During transmission of a frame (or ack bit, active error flag, orverload flag) the CAN node tried to send a dominant level (0), but the monitored bus value was recessive. 2) During bus-off recovery this code is set each time a sequence of 11 recessive bits has been monitored. The CPU may use this code as indication that the bus is not continuously disturbed.
0x06	CRC error – The CRC checksum of the received frame was incorrect.
0x07	reserved
0x08	frame transmitted onto CAN bus successfully
0x10	frame received from CAN bus successfully
0x11-0x1F	reserved
0x20	CAN receive buffer has overrun – a CAN frame has been dropped
0x30	CAN receive FIFO has overrun multiple times, following frames may be out of order

6.4.6 1 ms Resolution 2 Byte Time Stamp

The 1ms resolution time stamp is the last two bytes of the message if this option is enabled by the SAINT2 configuration command (08 87h). Each bit count is 1 ms.

1 ms Resolution Time Stamp Range

2 Byte Value	Absolute Time in ms
0000h – FFFFh	0ms – 65535ms

7 ISO15765-2 Transport Protocol on CAN

The SAINT2 can be configured to transmit and receive messages using the ISO15765-2 transport protocol. To transmit an ISO15765-2 message, the SAINT2 is able to contruct the necessary CAN frames, receive and process the flow control frames from the receiving device, and handle consecutive frame timing. The SAINT2 is capable of processing 1 ISO15765-2 message (up to 4K) from the host / 1ms loop. To receive an ISO15765-2 message, the SAINT2 is able to send the flow control frame and reconstruct the message from the multiple CAN frames.

If your application does not use the SAINT2 Bus Engine to communicate with the SAINT2 please see the section on block transfer for additional information on how to construct messages that are greater than 60 bytes.

7.1 Step 1 – Configure the SAINT2

7.1.1 Enable the protocols

By default, CAN 1, CAN 2, ISO15675-2 1, and ISO15765-2 2 are all enabled. If you are working with some application that has specifically enabled only certain protocols, it may be necessary to re-enable the CAN and ISO15765-2 procols. Use the 08 A3 message as defined in the SAINT2 Configuration section of this document. To enable all of these protocols send an 08 A3 50 58 C0 C8 message.

7.1.2 Configure the CAN channel

You must configure the CAN channel that you intent to use. See the previous section on CAN for details on how to make each configuration.

- Configure the baud rate.
- Configure the CAN transceiver (single wire, dual wire or fault tolerant) being used.
- Configure any special CAN functions.

These configurations will remain in affect until the SAINT2 hardware is reset or a new configuration is set.

7.1.3 Configure the SAINT2 for ISO15765-2

You must configure the SAINT2 so that it knows the information necessary to transmit and receive a message using the ISO15765-2 transport layer. This configuration is done using the following commands:

for CAN1:

C4 06 XX YY ID1 ID2 [ID3 ID4] [XD]opt configure the transmit message options, receiver’s flow control ID

C4 07 XX ID1 ID2 [ID3 ID4] [XD] opt - configure receive message options, receive message ID

C4 08 XX YY ZZ ID1 ID2 [ID3 ID4] [XD]opt – configure the flow control options, flow control ID

for CAN 2:
CC 06 XX YY ID1 ID2 [ID3 ID4] [XD]opt - configure the transmit message options, receiver's flow control ID
CC 07 XX ID1 ID2 [ID3 ID4] [XD]opt - configure receive message options, receive message ID
CC 08 XX YY ZZ ID1 ID2 [ID3 ID4] [XD]opt – configure the flow control options, flow control ID

7.1.3.1 Configuration for Transmitting ISO15765-2 messages

To transmit a message using the ISO15765-2 transport protocol, the flowing items must be configured:

- Should transmit frames be padded with 0x00?
- Does the transmit message CAN ID include an ISO15765-2 extended ID byte?
- Should the STmin received in the flow control message be overridden? If so, use what value?
- What is the receiver's flow control CAN ID?

These configurations will remain in affect until the SAINT2 hardware is reset, a new configuration is set, or the configuration is reset.

The configuration message has the following format:
CAN 1: **C4 06 XX YY ID1 ID2 [ID3 ID4]opt [XD]opt**
CAN 2: **CC 06 XX YY ID1 ID2 [ID3 ID4]opt [XD]opt**

Where:
XX is the ISO15765-2 TX option byte
bit 0 = 0: no frame padding bit 0 = 1: frame padding with 0x00s
bit 1 = 0: use flow control STmin bit 1 = 1: override flow control STmin with YY value
bit 2 = 0: tx msg contains no ISO15765-2 extended ID bit 2 = 1: tx msg contains an ISO15765-2 extended ID
bit 3 = 0: display both individual CAN frames and the ISO TP message bit 3 = 1: display only the ISO TP message
bit 4-7 = 0

YY is the value used for STmin in ms if byte XX, bit 1 = 1

ID1 ID2 [ID3 ID4]opt [XD]opt is the flow control CAN message ID. This ID definition must be in one of the following formats:

- 11 bit CAN ID with no ISO15765-2 extended ID: ID1 ID2

- 11 bit CAN ID with an ISO15765-2 extended ID: ID1 ID2 XD
- 29 bit CAN ID with no ISO15765-2 extended ID: ID1 ID2 ID3 ID4 with bit7 of ID1 = 1
- 29 bit CAN ID with an ISO15765-2 extended ID: ID1 ID2 ID3 ID4 XD with bit7 of ID1 = 1

The CAN message ID definition ID1 ID2 ID3 ID4 is consistent with the SAINT2's normal CAN message ID definition.

7.1.3.2 Configuration for Receiving ISO15765-2 messages

To receive a message using the ISO15765-2 transport protocol, the flowing items must be configured:

- What CAN IDs should be interpreted as ISO15765-2 transport protocol frames?
- When an ISO15765-2 transport protocol first frame is received, should the SAINT2 send a flow control frame?
- If so, what is the flow control frame CAN ID?
- Should the flow control frame be padded with 0x00?
- What is the FS (flow status) value?
- What is the BS (block size) value?
- What is the STmin (separation time) value?

These configurations will remain in affect until the SAINT2 hardware is reset, a new configuration is set, or the configuration is reset.

The receive message configuration message has the following format:
CAN 1: **C4 07 XX ID1 ID2 [ID3 ID4]opt [XD]opt ID1m ID2m [ID3m ID4m]opt [XDm]opt**
CAN 2: **CC 07 XX ID1 ID2 [ID3 ID4]opt [XD]opt ID1m ID2m [ID3m ID4m]opt [XDm]opt**

Where:
XX = 0x00 : individual CAN frames will be displayed as well as the ISO TP message
XX = 0x01 : only the ISO TP message will be displayed

ID1 ID2 [ID3 ID4]opt [XD]opt is the flow control CAN message ID. This ID definition must be in one of the following formats:

- 11 bit CAN ID with no ISO15765-2 extended ID: ID1 ID2 (byte XX bit 2 = 0)
- 11 bit CAN ID with an ISO15765-2 extended ID: ID1 ID2 XD (byte XX bit 2 = 1)
- 29 bit CAN ID with no ISO15765-2 extended ID: ID1 ID2 ID3 ID4 with bit7 of ID1 = 1 (byte XX bit 2 = 0)
- 29 bit CAN ID with an ISO15765-2 extended ID: ID1 ID2 ID3 ID4 XD with bit7 of ID1 = 1 (byte XX bit 2 = 1)

The CAN message ID definition ID1 ID2 ID3 ID4 is consistent with the SAINT2's normal CAN message ID definition.

ID1m ID2m [ID3m ID4m]opt [XDm]opt is the bit mask for filtering the ID. A bit value of 1 means the associated bit in the RX CAN ID will be compared. A bit value of 0 means the associated bit in the RX CAN ID will not be compared.

If a flow control frame is to be sent by the SAINT2 in response to a received ISO15765-2 first frame, the flow control frame parameters must be configured. The flow control frame configuration has the following format:
CAN 1: **C4 08 XX YY ZZ ID1 ID2 [ID3 ID4] [XD]opt**
CAN 2: **CC 08 XX YY ZZ ID1 ID2 [ID3 ID4] [XD]opt**

Where:
XX is the ISO15765-2 option byte
bit 0 = 0: no frame padding bit 0 = 1: frame padding with 0x00s
bit 1 = 0:
bit 2 = 0: FS = 0 clear to send consecutive frames bit 2 = 1: FS = 0 wait to send CFs
bit 3-7 = 0:
YY is the value used for BS (block size)
ZZ is the value used for STmin (separation time)

ID1 ID2 [ID3 ID4]opt [XD]opt is the flow control CAN message ID. This ID definition must be in one of the following formats:

- 11 bit CAN ID with no ISO15765-2 extended ID: ID1 ID2
- 11 bit CAN ID with an ISO15765-2 extended ID: ID1 ID2 XD
- 29 bit CAN ID with no ISO15765-2 extended ID: ID1 ID2 ID3 ID4 with bit7 of ID1 = 1
- 29 bit CAN ID with an ISO15765-2 extended ID: ID1 ID2 ID3 ID4 XD with bit7 of ID1 = 1

The CAN message ID definition ID1 ID2 ID3 ID4 is consistent with the SAINT2's normal CAN message ID definition.

7.2 Step 2 – Send/Receive the ISO15765-2 message

The ISO15765-2 message sent to the SAINT2 should contain only the CAN message ID, the optional ISO15765-2 extended ID, and the data. The SAINT2 will determine the length of the message based on the number of data bytes it has been sent. The SAINT2 will format each frame based on how it has been configured and insert the PCI bytes before it transmits the frames onto the CAN bus. Likewise, when a ISO15765-2 message is received on the CAN bus, the

SAINT2 will reconstruct the message and remove the PCI bytes. The following is the format for an ISO15765-2 message:

CAN 1: C0 ID1 ID2 [ID3 ID4]opt [XD]opt D1 D2 D3 … D4095 (up to 4095 bytes)

CAN 2: C8 ID1 ID2 [ID3 ID4]opt [XD]opt D1 D2 D3 … D4095 (up to 4095 bytes)

Where:

ID1 ID2 [ID3 ID4]opt [XD]opt is the message ID. This ID definition must be in one of the following formats:

- 11 bit CAN ID with no ISO15765-2 extended ID: ID1 ID2 (byte XX bit 2 = 0)
- 11 bit CAN ID with an ISO15765-2 extended ID: ID1 ID2 XD (byte XX bit 2 = 1)
- 29 bit CAN ID with no ISO15765-2 extended ID: ID1 ID2 ID3 ID4 with bit7 of ID1 = 1 (byte XX bit 2 = 0)
- 29 bit CAN ID with an ISO15765-2 extended ID: ID1 ID2 ID3 ID4 XD with bit7 of ID1 = 1 (byte XX bit 2 = 1)

The CAN message ID definition ID1 ID2 ID3 ID4 is consistent with the SAINT2's normal CAN message ID definition.

D1 D2 D3 … D4095 (up to 4095 bytes) is the data.

7.3 Step 3 – Evaluate the response code

A user should pay attention to any operational warning code that get set during ISO15765-2 operation. These codes are defined in the 08 A1 SAINT2 Configuration section of this document.

Op Warning Code	Definition
C0	SAINT2 buffer used to store a received ISO15765-2 message is busy and the message has not been stored. (The individual 5X CAN frames will still be displayed but there will be no associated CX message.)
C1	The SAINT2 TX FIFO is full and the SAINT2 has not been able to transmit the required flow control frame.

The SAINT2 will provide either an error response or a successful transmission response. Following is the format for the response message:

CAN 1: C6/C7 EE RC

CAN 2: CE/CF EE RC

RC	Response Code Definition
00	Successful Transmission
01	SAINT2 could not write to transmit FIFO – message is discarded

02	SAINT2 did not receive the defined flow control response to the first frame or consecutive frame block – message is discarded
03	SAINT2 is unable to successfully transmit on the bus – message is discarded
04	SAINT2 has not been configured to transmit an ISO15765-2 message – message is discarded
05	The SAINT Bus Engine messages to the SAINT2 are out of order – message is discarded
06	The CAN packet function is busy – message is discarded

7.4 Clear the ISO15765-2 Configuration

To clear the ISO 15765-2 TX Configuration send the following message:

CAN 1: C4 06 FF

CAN 2: CC 06 FF

To clear the ISO 15765-2 RX Configuration send the following message:

CAN 1: C4 08 FF

CAN 2: CC 08 FF

7.5 Examples

7.5.1 Transmit Examples

7.5.1.1 Example 1 – 11 bit ID, no extended ID, no padding, no override

Send the following ISO15765-2 message on CAN1:

- Message ID: 0x7E0 Message Data: 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
- no padding
- use flow control STmin value
- expect flow control from message ID: 0x7E8

Perform the following steps to configure the SAINT2:

1. Configure the SAINT2 CAN1 baud rate and transceiver (see previous sections for instruction)
2. Configure the ISO15765-2 TX function: 0xC4 06 00 00 07 E8
3. Send the message: **0xC0 07 E0 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F**
4. Check the response from the SAINT2 to make sure the message transmitted successfully: 0xC7 EE 00

Message Monitor Example:

C7 06 00 00 07 E8 Command from PC to SAINT2 HW to config ISO15765-2 function

53 **07 E0** 10 0F **01 02 03 04 05 06** SAINT2 sends first frame to product

51 07 E8 30 00 00 Product responds with flow control frame

53 **07 E0** 21 **07 08 09 0A 0B 0C 0D** SAINT2 continues with consecutive frame

53 **07 E0** 22 **0E 0F** SAINT2 continues with consecutive frame

C3 07 E0 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Complete message in monitor

C7 EE 00 SAINT2 HW notifies PC that the ISO message has transmitted successfully

7.5.1.2 Example 2 – 29 bit ID, no extended ID, padding and override

Send the following ISO15765-2 message on CAN1:

- Message ID: 0x01001020 Message Data: 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
- padding
- use STmin of 10ms
- expect flow control from message ID: 0x01081020

Perform the following steps to configure the SAINT2:

1. Configure the SAINT2 CAN1 baud rate and transceiver (see previous sections for instruction)
2. Configure the ISO15765-2 TX function: 0xC4 06 03 0A 81 08 10 20
3. Send the message: 0xC0 **81 00 10 20 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F**
4. Check the response from the SAINT2 to make sure the message transmitted successfully: 0xC7 EE 00

Message Monitor Example:

C7 06 03 0A 81 08 10 20 Command from PC to SAINT2 HW to config ISO15765-2 function

53 **81 00 10 20** 10 0F **01 02 03 04 05 06** SAINT2 sends first frame to product

51 81 08 10 20 30 00 00 00 00 00 00 00 Product responds with flow control frame

53 **81 00 10 20** 21 **07 08 09 0A 0B 0C 0D** SAINT2 continues with consecutive frame

53 **81 00 10 20** 22 **0E 0F** 00 00 00 00 00 SAINT2 continues with consecutive frame

C3 **81 00 10 20 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F** Complete message in monitor

C7 EE 00 SAINT2 HW notifies PC that the ISO message has transmitted successfully

7.5.1.3 Example 3 – 11 bit ID, extended ID, padding, no override

Send the following ISO15765-2 message on CAN2:

- Message ID: 0x7DF, Extended ID: 0xFE Message Data: 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
- padding
- use flow control STmin value
- expect flow control from message ID: 0x7E8, Extended ID: 0xF0

Perform the following steps to configure the SAINT2:

1. Configure the SAINT2 CAN2 baud rate and transceiver (see previous sections for instruction)
2. Configure the ISO15765-2 TX function: 0xCC 06 05 00 07 E8 F0
3. Send the message: **0xC8 07 DF FE 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F**

4. Check the response from the SAINT2 to make sure the message transmitted successfully: 0xCF EE 00

Message Monitor Example:

CF 06 05 00 07 E8 F0 Command from PC to SAINT2 HW to config ISO15765-2 function

5C **07 DF FE** 10 0F **01 02 03 04 05** SAINT2 sends first frame to product

59 07 E8 F0 30 00 00 00 00 00 00 Product responds with flow control frame

5C **07 DF FE** 21 **06 07 08 09 0A 0B** SAINT2 continues with consecutive frame

5C **07 DF FE** 22 **0C 0D 0E 0F** 00 00 SAINT2 continues with consecutive frame

CB **07 DF FE 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F** Complete message in monitor

CF EE 00 SAINT2 HW notifies PC that the ISO message has transmitted successfully

7.5.1.4 Example 4 – 29 bit ID, no extended ID, padding, single frame

Send the following ISO15765-2 message on CAN1:

- Message ID: 0x01001020 Message Data: 01 02 03 04 05
- padding
- use flow control STmin value
- expect flow control from message ID: 0x01081020

Perform the following steps to configure the SAINT2:

1. Configure the SAINT2 CAN1 baud rate and transceiver (see previous sections for instruction)
2. Configure the ISO15765-2 TX function: 0xC4 06 01 00 81 08 10 20
3. Send the message: 0xC0 **81 00 10 20 01 02 03 04 05**
4. Check the response from the SAINT2 to make sure the message transmitted successfully: 0xC7 EE 00

Message Monitor Example:

C7 06 01 00 81 08 10 20 Command from PC to SAINT2 HW to config ISO15765-2 function

53 **81 00 10 20** 05 **01 02 03 04 05** 00 00 SAINT2 sends single frame to product

C3 **81 00 10 20 01 02 03 04 05** Complete message in monitor

C7 EE 00 SAINT2 HW notifies PC that the ISO message has transmitted successfully

7.5.2 Receive Examples

7.5.2.1 Example 1 – 11 bit ID, no extended ID, no padding, no override

Receive the following ISO15765-2 messages on CAN1:

- receive from Message ID: 0x7E8
- use flow control message ID: 0x7E0
- no FC padding
- STmin, BS, and FS = 0

Perform the following steps to configure the SAINT2:

1. Configure the SAINT2 CAN1 baud rate and transceiver (see previous sections for instruction)
2. Configure the ISO15765-2 RX function: 0xC4 07 00 07 E8 FF FF
3. Configure the ISO15765-2 FC function: 0xC4 08 00 00 00 07 E0
4. Receive the message: **0xC0 07 E8 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F**

Message Monitor Example:

C7 07 00 07 E8 FF FF — Command from PC to SAINT2 HW to config ISO15765-2 RX function

C7 08 00 00 00 07 E0 FF FF — Command from PC to SAINT2 HW to config ISO15765-2 RX function

51 **07 E8** 10 0F **01 02 03 04 05 06** — Product sends first frame to SAINT2

53 07 E0 30 00 00 — SAINT2 responds with flow control frame

51 **07 E8** 21 **07 08 09 0A 0B 0C 0D** — Product continues with consecutive frame

51 **07 E8** 22 **0E 0F** — Product continues with consecutive frame

C1 07 E8 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F — Complete message in monitor

7.5.2.2 Example 2 – 29 bit ID, no extended ID, padding

Receive the following ISO15765-2 messages on CAN 1:

- receive from Message ID: 0x01001020
- use flow control message ID: 0x01081020
- FC padding
- STmin = 10ms, BS = 0, and FS = 0

Perform the following steps to configure the SAINT2:

1. Configure the SAINT2 CAN1 baud rate and transceiver (see previous sections for instruction)
2. Configure the ISO15765-2 RX function: 0xC4 07 00 01 00 10 20 FF FF FF FF
3. Configure the ISO15765-2 FC function: 0xC4 08 01 00 0A 01 08 10 20
4. Receive the message: 0xC0 **81 00 10 20 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F**

Message Monitor Example:

C7 07 00 01 00 10 20 FF FF FF FF — Command from PC to SAINT2 HW to config ISO15765-2 RX function

C7 08 01 00 0A 01 08 10 20 — Command from PC to SAINT2 HW to config ISO15765-2 RX function

51 **81 00 10 20** 10 0F **01 02 03 04 05 06** — Product sends first frame to SAINT2

53 81 08 10 20 30 00 0A 00 00 00 00 00 — SAINT2 responds with flow control frame

51 **81 00 10 20** 21 **07 08 09 0A 0B 0C 0D** — Product continues with consecutive frame

51 **81 00 10 20** 22 **0E 0F** 00 00 00 00 00 — Product continues with consecutive frame

C1 **81 00 10 20 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F** — Complete message in monitor

7.5.2.3 Example 3 – 11 bit ID, extended ID, padding, no override

Receive the following ISO15765-2 messages on CAN 2:

- receive from Message ID: 0x7DF, Extended ID: 0xFE

- use flow control message ID: 0x7E8, Extended ID: 0xF0
- FC padding
- STmin, BS, and FS = 0

Perform the following steps to configure the SAINT2:

1. Configure the SAINT2 CAN2 baud rate and transceiver (see previous sections for instruction)
2. Configure the ISO15765-2 RX function: 0xCC 07 00 07 DF FE FF FF FF
3. Configure the ISO15765-2 FC function: 0xCC 08 01 00 00 07 E8 F0
4. Receive the message: 0xC8 **07 DF FE 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F**

Message Monitor Example:

CC 07 00 07 DF FE FF FF FF	Command from PC to SAINT2 HW to config ISO15765-2 RX function
CC 08 01 00 00 07 E8 F0	Command from PC to SAINT2 HW to config ISO15765-2 RX function
59 **07 DF FE** 10 0F **01 02 03 04 05**	Product sends first frame to SAINT2
5C 07 E8 F0 30 00 00 00 00 00 00	SAINT2 responds with flow control frame
59 **07 DF FE** 21 **06 07 08 09 0A 0B**	Product continues with consecutive frame
59 **07 DF FE** 22 **0C 0D 0E 0F** 00 00	Product continues with consecutive frame
C9 **07 DF FE 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F**	Complete message in monitor

7.5.2.4 Example 4 – 29 bit ID, no extended ID, padding, single frame

Receive the following ISO15765-2 messages on CAN 1:

- receive from Message ID: 0x01001020
- use flow control message ID: 0x01081020
- FC padding
- STmin, BS, and FS = 0

Perform the following steps to configure the SAINT2:

1. Configure the SAINT2 CAN1 baud rate and transceiver (see previous sections for instruction)
2. Configure the ISO15765-2 RX function: 0xC4 07 00 01 00 10 20 FF FF FF FF
3. Configure the ISO15765-2 FC function: 0xC4 08 01 00 00 01 08 10 20
4. Receive the message: 0xC0 **81 00 10 20 01 02 03 04 05**

Message Monitor Example:

C7 07 00 01 00 10 20 FF FF FF FF	Command from PC to SAINT2 HW to config ISO15765-2 RX function
C7 08 01 00 00 01 08 10 20	Command from PC to SAINT2 HW to config ISO15765-2 RX function
C1 **81 00 10 20 01 02 03 04 05**	Complete message in monitor

7.6 Important Notes

- Only a single CAN channel can be configured at a time for ISO15765-2. If you configure CAN1 and then configure CAN2, only CAN2 will be configured.
- As individual CAN frames are transmitted or received by the SAINT2, these frames will be sent back to the host just as any other transmitted/received frame would be sent back to the host. These frames will have the normal SAINT2 headers for CAN ($50/58 derivatives). The exception is that single frame ISO15765-2 received messages will only be displayed as a C0/C8 message.
- C0/C8 messages will not necessarily be transmitted in the order they are received from the host in relation to raw CAN frames ($50/$58) If this is a problem, you should always make sure that your raw messages have been transmitted successfully before you send your ISO15765-2 message to the SAINT2.
- Raw CAN messages and ISO15765-2 messages may be used together.
- There is a single buffer used for both transmitted and received ISO15765-2 messages on the SAINT2 hardware. An ISO15765-2 message may either be in the process of being transmitted or received, but the SAINT2 can not do both at the same time.

8 Class 2

8.1 Class 2 Messages

The following messages may be used to send a Class 2 commands and messages. Class 2 commands are used for configuring or commanding the DLC. Class 2 messages include functional, physical, and block messages. Note: the ESCAPE characters (section 4.1) must be instered at the end of every transmitted message to the SAINT.

8.1.1 Class 2 Commands

The following is the format for sending a Class 2 Command:

Header	ID	Description	Data Bytes
64h	01h	Report Error	1 byte Completion Code (See 8.1.3.1)
64h	02h	SPI Bus Error	1 byte Error Code (See 8.1.3.2)
64h	64h	Operate Bus at Normal Speed (10.4 KB/s) (default)	None
64h	65h	Operate Bus at High Speed (41.6 KB/s)	None
64h	66h	Generate a Class 2 Break Signal	None

8.1.2 Transmitted Class 2 Messages

The following is the format for sending a Class 2 message:

60h	Priority/Header	Target	Source	*ID*	*N Data Bytes*
Byte 1	Byte 2	Byte 3	Byte 4	*Byte 5*	*Bytes 6 —(5+N)*

The minimum accepted Class 2 message length is 3 bytes and the longest is 8000 bytes.

8.1.3 Received Class 2 Messages

The following is the format of a received Class 2 message without time stamp information:

60h	Priority Header	Target	Source	*ID*	*N Data Bytes*	Completion Code
Byte 1	Byte 2	Byte 3	Byte 4	*Byte 5*	*Bytes 6 —(5+N)*	Byte 6 + N

The following is the format of a received Class 2 message with time stamp information:

61h	Priority Header	Target	Source	*ID*	*N Data Bytes*	Completion Code	Time MSB	Time LSB
Byte 1	Byte 2	Byte 3	Byte 4	*Byte 5*	*Bytes 6 —(5+N)*	Byte 6 + N	7+N	8+N

The following is the format of a received Class 2 Error message with time stamp information:

65h	01	Completion Code	Time MSB	Time LSB
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5

8.1.3.1 Completion Code

The following is the definition of the completion code included with all reported Class 2 messages. The completion code in the received messages comes from the DLC chip. For more detail refer to the DLC Application document (XDE 3003.)

Bit 7 Errors in Received Message

0 No Error Detected.

1 Error(s) detected. When set, certain errors occurred in the reception of this message (see description for bits 1 and 0). If not set, then bits 1 and 0 are also 0.

Bit 6 RFIFO Overrun

0 No overrun.

1 A receiver FIFO overrun was detected and data was lost.

Bit 5 Bit 4 Transmitter Action

0 0 Transmitter not involved.

0 1 Transmitter under run. Should occur only when sending a Class 2 block message and the host does not keep up with the data requirements.

1 0 Transmitter lost arbitration on this message.

1 1 Transmit successful.

Bit 3 In-frame response

0 This message was not an in-frame response.

1 This message was an in-frame response.

Bit 2 In-frame response with/without CRC

0 This in-frame response does not contain a CRC.

1 This in-frame response contains a CRC.

Bit 1 Bit 0 Error Code

0 0 CRC error.

0 1 Incomplete byte received.

1 0 Bit timing error.

1 1 Break symbol received.

8.1.3.2 SPI Error Codes

The following is the definition of the SPI error code associated with all reported SPI error messages. Most of these errors are Class 2 errors that manifest themselves when the main micro is communicating with the DLC transceiver.

Bit 7 Unused

Bit 6 Rx of Tx

0 No Error.

1 A transmitted message was not received on the bus. This error typically indicates that the Class2 bus is shorted to another signal.

Bit 5 Other

0 No Error.

1 All other SPI errors

Bit 4 Unused

Bit 3 Tx Underrun

0 No Error.

1 DLC Tx Underrun. (Determined from completion code)

Bit 2 Message too big

0 No Error.

1 Too many bytes in Class 2 message

Bit 1 Unused

Bit 0 Receive

0 No Error.

1 Rx error on SPI bus. This error occurs when a new data byte is received, but a previously received data byte has not been read out of the receive buffer. The old data in the receive buffer will be overwritten and unretrievably lost.

8.1.3.3 Block Message Support

Please note the following differences when using Class2 firmware 3.0 or later. Firmware 3.0 and later has the support for Block messages. Therefore, the decoding for Rx/Tx (bit1 in header byte) is no longer valid in the header byte. It is recommended to use the Completion Code to check if the message was transmitted (30h) or received (00h). Also, note that “Echo Off” command (08 89) is not supported. So, all transmitted Class2 messages are received. The changes are due to the fact that the SAINT cannot determine if the incoming message was transmitted or received until completely receiving the entire message.

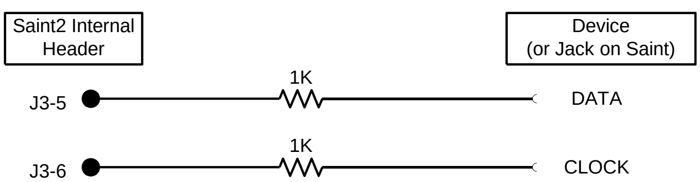
9 IIC

9.1 Overview

The Saint2 can function in either active participant mode or passive monitor mode. When the Saint2 is acting as a slave or master it is an **active participant** on the bus, but in the original Saint it functioned as a **passive participant** just "sniffing" the data.

Passive Bus Monitor Mode

This is the "sniffer" or passive monitor functionality for the Saint2, which is similar to the IIC support provided in the original Saint 1. To use this feature on the Saint 2, connection to the internal header pins is required. The IIC clock and data pins should be connected as follows:



The device connections can simply be "pigtailed" out of the Saint2 or, for more permanent use, connectors (such as Banana Jacks) can be installed in the end plate of the Saint2 box. (For permanent connections, it is recommended to mount any additional connectors to the end plate with the D-Type connectors. This allows the board, ,end plate, and any custom wiring to be removed as a single assembly).

Important Notes:

1) ***Failure to use the 1Kohm current limitings resistors can result in damage to the Saint2 or the attached device.***
2) ***Host (PC) commincation in this mode is USB ONLY! (RS-232 communicaiton is disabled upon entering this mode)***

The following command may be used to begin monitoring IIC traffic.

Header	ID	Description	Data Bytes
24h	01h	Begin IIC Monitor Mode	

To exit the IIC Monitor Mode use a hardware reset.

IIC Received Message Format with Time Stamp

21h	Address	N Data Bytes	Completion Code	Time Stamp Bytes
byte 1	byte 2	Byte 3 … Byte N	Byte N + 1	Byte N+2 Byte N+3

Completion Code

Bit 1	Bit 0	Error Code
0	1	Repeated Start Condition
1	0	NACK received in message

Active Participant Mode

The Saint2 also has an **active participant** mode in which the Saint2 acts as a slave or a master, and please note that this mode is completely different than monitor mode. The initial 7 bit IIC address is set to 0x55 when in master mode and 0x65 when in slave mode. The Saint2 IIC module can be used to test devices such as CD player mechanisms as they would work within a radio.

Note: In order to have successful IIC communication your device must meet the following conditions:

5. The device must provide the pull up resistors for SDA and SCL. If these pullups for SDA and SCL are not in the device you are testing , you must provide pull ups externally to the appropriate supply voltage for your device.
6. The device must be able to provides acknowlegments to the SAINT2 IIC bytes. The IIC bus will not operate properly unless there is both a master and a slave communicating on the bus.

9.1.1 IIC Hardware

Name	Saint 2 Pin Number	Description
SDA	J4.29	I^2C Data
SCL	J4.27	I^2C Clock

9.2 IIC Commands

IIC configuration command messages adhere to the following general format:

From PC:

IIC Configuration Command Message: 20 XX_2 XX_3 XX_4 … XX_n FF 00

Byte 1: Saint IIC Protocol Identifier for IIC (0x20)

Byte 2: IIC message type (see below for types)

Bytes 3-n: Command to be parsed and carried out by Saint 2

Last 2 Bytes: End of SAINT message

Note: If using the Saint Bus Monitor, the "End of SAINT message" bytes (FF 00) are automatically added to the end of the message. Therefore, they do not need to be manually specified.

9.2.1 Configure IIC Master Polling

Configure IIC Master Polling: 20 92 AA CC FF 00

Byte 1: Saint IIC Protocol Identifier for IIC (0x20)

Byte 2: IIC message type for polling (0x92)

Byte 3: 8 bit slave address to be polled (LSB must be 1)

Byte 4: Configuration byte. A value of 0 turns polling off. A value of 1 turns polling on.

Last 2 Bytes: End of SAINT message

Note: The slave address must be included since the master produces the address it wishes to address.

Note: The slave address is polled every 1 second by the master.

Note: If the slave does not acknowledge any of the data bytes, a message with the following format will be sent back:

20 91 F1 MM_1 … MM_N,

where MM_1 … MM_N are the data bytes of the failing IIC message.

9.2.2 Set Device Address

20 A0 XX FF 00 – sets the IIC Device Address to XX (7-bit address, XX <= 7F)

9.2.3 Get IIC Device Address

20 A1 FF 00 – returns current 7-bit address to Saint Bus Monitor as:

20 A1 F1 XX FF 00'. where XX is the address.

9.2.4 Set IIC Mode

20 DD 01 FF 00 – sets the IIC Device to Master Mode

20 DD 00 FF 00 – sets the IIC Device to Slave Mode

9.2.5 Set IIC Baud Rate

20 DE 01 FF 00 – sets the IIC Baud Rate to 100 Khz

20 DE 00 FF 00 – sets the IIC Baud Rate to 400 Khz

9.3 IIC Data Messages

Note: The Saint 2 can act as a slave or a master. When it is acting as a slave, it will use the slave-transmitter and slave-receiver message formats. When it is acting as a master, it will use the master-transmitter and master-receiver message formats.

Note: There is no status byte for receipt of IIC messages.

Note: The maximum length of the IIC data bytes is limited by the maximum Saint 2 message length.

9.3.1 Request to transmit an IIC message as slave-transmitter

Send IIC Slave Transmit Message: 20 90 XX_3 XX_4 XX_5 … XX_n FF 00

Byte 1: SAINT IIC Protocol Identifier (0x20)

Byte 2: Message type for IIC message (0x90)

Bytes 3-n: Data to be sent

Last 2 Bytes: End of SAINT message

Note: The data in bytes 3-n should not include the device address, since the IIC master (not the IIC slave) produces the address.

9.3.2 Receipt of IIC message as slave receiver

IIC Slave Received Message: 20 90 XX_3 XX_4 XX_5 … XX_n FF 00

Byte 1: SAINT IIC Protocol Identifier (0x20)

Byte 2: Message type for IIC Message (0x90)

Byte 3: 8-bit representation of the Saint 2 7-bit IIC slave device address (i.e., 7-bit address 001 1000 (18h) becomes 8-bit representation 0011 0000 (30h))

Bytes 4-n: Data received

Last 2 Bytes: End of SAINT message

9.3.3 Request to transmit an IIC message as master-transmitter

Send IIC Master Transmit Message: 20 90 AA XX_4 XX_5 … XX_n FF 00

Byte 1: SAINT IIC Protocol Identifier (0x20)

Byte 2: Message type for IIC message (0x90)

Byte 3: 8 bit slave address to send data to (LSB must be 0)

Bytes 4-n: Data to be sent

Last 2 Bytes: End of SAINT message

Note: The slave address must be included since the master produces the address it wishes to address.

Note: If the slave does not acknowledge any of the data bytes, a message with the following format will be sent back:

20 91 F1 MM_1 … MM_N,

where MM_1 … MM_N are the data bytes of the failing IIC message.

9.3.4 Request to receive an IIC message as master-receiver

Request to Read from Slave Message: 20 90 AA NN FF 00

Byte 1: SAINT IIC Protocol Identifier (0x20)

Byte 2: Message type for IIC Message (0x90)

Byte 3: 8 bit slave address to receive data from (LSB must be 1)

Byte 4: Number of data bytes to read from the slave

Last 2 Bytes: End of SAINT message

Note: If byte 4 is 0, instead of reading 0 bytes, the master will read 1 byte from the slave. The master will interrupt this byte as N (the number of remaining bytes to read). The master will then read N more data bytes from the slave.

Note: Since the master is in charge of driving the IIC bus, it must request to read data from the slave. The slave has no way of requesting it be addressed so that it can send.

Note: The above message requests a read be performed as a master-receiver. The actual data read will be sent back in the message below.

Note: If the slave does not acknowledge its address on the bus, a message with the following format will be sent back:

20 91 F1 MM_1 … MM_N,

where MM_1 … MM_N are the data bytes of the failing IIC message.

IIC Master-Received Data Message: 20 90 XX_3 XX_4 XX_5 … XX_n FF 00

Byte 1: SAINT IIC Protocol Identifier (0x20)

Byte 2: Message type for IIC Message (0x90)

Byte 3: 8-bit representation of the Saint 2 7-bit IIC slave device address (i.e., 7-bit address 001 1000 (18h) becomes 8-bit representation 0011 0000 (31h))

Bytes 4-n: Data received

Last 2 Bytes: End of SAINT message

10 Keyword 2000

Keyword 2000 for the Saint 2 is an enhanced version of the Keyword 2000 that was available on the Saint 1.
Some Key improvements are:

- Full length message support (up to the spec limit of 260 bytes)
- True 5 Baud initialization ("Active" baud rate detection)
- USB interface
- Expanded message corruption (forced error) capabilities

A completely new set of command IDs have been implemented for the Saint 2 to prevent potential conflicts from divergent development on each platform. While many of the command functions are the same, others have been modified to provide expanded capability and new commands have been added to accommodate new features.

To ensure compatability with users legacy Saint 1 applications and files, the Saint 2 KW2K implementation also supports the full Saint 1 Keyword 2000 command set (as of the Saint 1 Keyword 2000 Users Guide, Version A, Draft 7, 1/11/05). Either or both command sets may be used. This manual only includes documentation for the new Saint 2 Keyword 2000 command set. Users may reference the afore mentioned Saint 1 document to obtain the legacy command set information.

!!!!! IMPORTANT !!!!!

You MUST have a Saint 2 with hardware "Version 1.1" or higher to use it with Keyword 2000.

The hardware version number is listed on the bottom left corner of the front label. If the label does not indicate a hardware version, it is Version 1.0 hardware and will NOT work properly for Keyword 2000.

If you need to use Keyword 2000 and have an incompatible version of the Saint 2, contact the manufacturer or email support for upgrade/replacement options.

10.1 Keyword 2000 – Application Notes

10.1.1 General Operation

10.1.1.1 Message Construction

All modes, except Direct Mode (2C A0 02), employ automated message construction to simplify use of the protocol for manual/human operations (as was done in the Saint 1). The message header bytes and checksum are automatically built based on the current configuration settings. Once properly configured, users only need to provide the Saint 2 Header Byte (protocol ID) and the data portion of the message. The appropriate KW header and checksum bytes will be automatically constructed in the message frame before it is transmitted on the bus. The following is an example of the message construction performed using the default configuration settings (3 byte hdr, Tgt Addr=11, Src Addr=F1):

Saint2 Hdr (KW Protocol ID)

User/Host Msg: 28 AA — Msg data

Resulting Bus Msg: <u>81 11 F1</u> AA 2D

3 byte header — Checksum

Direct Mode does NOT employ any form of message construction. It is intended to provide users unrestricted control of the message frame content. Message frames are delivered exactly as supplied (unmodified) directly to the bus. The user is responsible for all message frame content including the header and checksum bytes.

10.1.2 Initialization

10.1.2.1 Client (Tester) Mode

Bus initialization is handled automatically by the Saint 2. Once the protocol configuration options are properly set, the user need only send the desired KW message. The Saint 2 tracks the bus initialization state and, if required, automatically performs the necessary initialization operations followed by the desired KW message. The user does NOT need to perform bus initialization manually (i.e. by sending a "Start Comms" message). If desired, the user may "manually" initialize the bus by sending the "Start Comms" (i.e. mode 81) message. This will initialize the bus without sending any additional messages. (Note that a user supplied "Start Comms" message is actually discarded by the Saint 2 and a new "Start Comms" message is rebuilt based on the configured "Start Comms Header Type" and other applicable configuration settings.) If the bus is already initialized and the “Start Comms” message is sent, the “Start Comms” message will be sent out on the bus as a “normal” bus bearing message and NOT as a Fast Init (i.e. **WITHOUT** the preceeding “wake up” pulse)

NOTE: The “Stop Comms” and “Tester Present” (Keep Alive) messages can NOT be used to either manually or automatically initialize the bus. These are treated as special messages by the Saint 2 and are prohibited from being used for starting communications. All other bus bearing messages will initiate the automatic bus initialization sequence. (Logically it does not make sense to send a “Stop Comms” message to start communications and the “Tester Present” message is ONLY intended to be used after communication has been established… not to start it.)

10.1.2.2 P2P Mode

Bus initialization is not required in this mode as the bus is considered always active (initialized). If the host/user sends a “start comms” message, it will be transmitted on the bus as a normal KW msg (i.e. NOT as an Fast Init sequence). Though not intended for use with this mode, the “Force Inititalization” command is allowed and will transmit the full initialization sequence (as configured) on the bus.

10.1.2.3 Direct Mode

Bus initialization can ONLY be initated by using the “Force Inititalization” command. All transmit messages are treated as normal “bus bearing” messages and are delivered to the bus unmodified. They are not intercepted, decoded, or checked and therefore, can not be used to trigger an initialization operation. In this mode, the host/user is completely responsible for intialiaztion operations and bus state tracking.

10.1.3 Error Reporting

If multiple error conditions are detected, generally, only the first (most significant) error is reported. For example, if a message is received with a “framing error”, this usually results in a checksum error as well. Only the “framing error” will be reported as this is considered the primary, or “root” error condition. This strategy is employed to prevent reporting of multiple related (cascading) error indications. Timing errors (P1-P4) are always report as appropriate for the operational mode selected.

10.2 Keyword 2000 Commands

The following commands may transmitted from the host interface to the SAINT 2:

Protocol Configuration Commands			
Hdr	ID	Description	Data Bytes

Protocol Configuration Commands			
2Ch	A0h	Set Operating Mode	1 byte: 0x00 = Client (Tester) mode **(default)** Behaves as a Tester device (i.e. scan tool) on a “regular” KW bus. Obeys the rules of a Client device per ISO 14230. 0x01 = "P2P" (Peer-to-Peer) mode (formerly “Dedicated” mode – renamed for clarity) This is a special mode for applications that employ a dedicated “always on” KW link between two devices (i.e. Peer to Peer). Behaves as an ECU and obeys the rules of a Server device except initialization operations are disabled and the bus is treated as always "initialized". 0x02 = “Direct” (raw) Client mode Similar to Client (Tester) mode above except: • Host/User (Tx) messages are delivered directly to the bus without modifidcation (i.e. no msg construction/checking). • Initializaiton is strictly manual (non-automatic) • Limited configuration settings apply. Received messages are processed normally (per KW protocol rules). Timing parameters (P1-4) are enforced for both Tx & Rx messages. Bus initialization can ONLY be performed using the dedicated “Force Initialization” command (2C B2). The host/user is entirely responsible for message construction (incl. headers and checksum), content, and initialization. Several commands are either not applicable or are limited in this mode due to redundant or non-applicable functionality (primarily those affecting msg content). These commands are denoted with an asteric “*” below the command code. Commands with functionality that can only be produced by the Saint 2 (i.e. by hardware/ firmware) are still applicable. Note: This mode essentially bypasses the automated message construction and is primarily intended for host applications that require unrestricted control of the KW message frame. Though added primarily for J2534 support, it is available for use in any application where direct conrol of the message frame content is desired.
2Ch	A1h	Set Initialization Mode	1 byte: 0x00 - Fast Init **(default)** 0x01 - 5 Baud 0x02 - Carb (not implemented)

Protocol Configuration Commands

2Ch	A2h	Set Baud Rate (Applies to Fast Init only).	1-2 bytes: Baud rate in hundreds units expressed in BCD . Leading zeros are disregarded. Range: 1k - 200k baud (in 100 baud steps) **Default = 10400 baud (spec compliant)** Examples: 2C A2 11 52 - set baud to 115200 2C A2 00 10 - set baud to 1000 (leading 0's ignored) 2C A2 or 2C A2 00 - set default baud rate
2Ch	A3h *	Set Header Type	1 byte: 0x00 = [Fmt] (len in Fmt byte) 0x01 = [Fmt][Len] 0x02 = [Fmt][Tgt][Src] (len in Fmt byte) **(default)** 0x03 = [Fmt][Tgt][Src][Len]
2Ch	A4h *	Set Tester Address	1 byte: Tester Address **(default = 0xF1)** Tester (Client) mode - used as source address. P2P/ECU (Server) mode - used as target address. Direct (Gateway) mode – used as source address (for initialization only)
2Ch	A5h *	Set ECU / 5 Baud Address	1 byte: ECU Address **(default = 0x11)** Tester (Client) mode - used as target address. P2P/ECU (Server) mode - used as source address. Direct (Gateway) mode - used as target address (for initialization only)
2Ch	A6h *	Set Addressing Mode	1 byte: 0x00 = Physical Addressing **(default)** 0x01 = Functional Addressing
2Ch	ACh	Set “Start Comms” Header Type Support for non-standard KW2K implementations. (The default (spec compliant) setting should be used unless specifically required.)	1 byte: 0x00 = [Fmt] (len in Fmt byte) 0x01 = [Fmt][Len] 0x02 = [Fmt][Tgt][Src] (len in Fmt byte) **(default)** 0x03 = [Fmt][Tgt][Src][Len] Note: When the “Force Initialization” command is used, this setting applies ONLY if the optional substitute Start Comms message is NOT supplied (otherwise the substitute message is used).

* - Not Applicable (ignored) in “Direct” Operational Mode (2C A0 02).

Protocol Timing Configuration Commands

(IMPORTANT - It is the users responsibility to abide by the timing relationship rules declared in the specifications. The Saint does NOT enforce inter-relational rules, it only enforces the absolute min/max limits for each parameter.)

Hdr	ID	Description	Data Bytes
2Ch	A7h	Set P1 min/max Server (ECU) interbyte time limits.	2 bytes - aa bb: aa - 0x00 = Set P1 min **(default=5ms, 0x05)** 0x01 = Set P1 max **(default=20ms, 0x14)** bb - time in ms. (0-20ms, 0x00-0x14)
2Ch	A8h	Set P2 min/max Client (Tester) to Server (ECU) or Server to Server inter message time limits.	2 bytes - aa bb: aa - 0x00=Set P2 min **(default=25ms, 0x19)** bb=time in ms. (0-127ms, 0x00-0x7F) - 0x01=Set P2 max **(default=50ms, 0x02)** bb=time in ms encoded as follows: 0x01-0xF0: time = value*25. 0xF1-0xFE: time = low nibble of value*256*25. 0xFF: n/a (invalid)
2Ch	A9h	Set P3 min/max Server (ECU) to Client (Tester) inter message time limits.	2 bytes - aa bb: aa - 0x00=Set P3 min **(default=55ms, 0x37)** bb=time in ms, (0-127, 0x00-0x7F) - 0x01=Set P3 max **(default=5120ms, 0x14)** bb=time*250 in ms. (0-255, 0x00-0xFF) where: 0-254, 0x00-0xFE = 0-63500ms 255, 0xFF=infinity
2Ch	AAh	Set P4 min/max Client (Tester) interbyte time limits.	2 bytes - aa bb: aa - 0x00=Set P4 min **(default=5ms, 0x05)** bb - time in ms. (0-19ms, 0x00-0x13) aa - 0x01=Set P4 max **(default=20ms, 0x14)** bb - time in ms. (0-20ms, 0x00-0x14)
2Ch	ABh	Set Fast Init Wakeup Pattern low time (T_{initL})	1 byte: time in msec (5ms multiples). **(Default=25ms, 0x19)** Values other than multiples of 5ms may be used but will be rounded to the closest multiple. The Wakeup Pattern time (T_{Wup}) is fixed at 50ms. Setting the low period adjusts the high period accordingly. $T_{IniH} = T_{Wup} - T_{IniL} = 50ms - T_{IniL}$

General/Misc Control Commands

Hdr	ID	Description	Data Bytes
2Ch	B0h	Enable/Disable Timing Error Reporting (P1-P4)	1 byte: 0x00 = Disable 0x01 = Enable **(default)**
2Ch	B1h	Enable Tester Present (Keep Alive) Periodic Message (Period= P3max-1ms, per specifications.)	1 byte: 0x00 - Disabled **(default)** 0x01 – Enabled
2Ch	B2h	Force Initialization Forces a bus initialization based on the current configuration settings. Optionally allows the user to provide a substitute message for the normal Fast Init “Start Comms” message.	(optional) 1 to 259 bytes xx…xx: xx… - Fast Init “Start Comms” Substitute msg Initialization will be performed without regard to the current bus initialization state. P3min timing is enforced before the initialization is allowed to begin. If provided, the optional message bytes will be substituted for the Fast Init “Start Comms” message. The configuration must be previously set to use Fast Init mode (default) before invoking this command or the message will be discarded as the “Start Comms” message does not apply to 5 Baud/CARB Inits. Note:This command was primarily added to support the “Direct” operating mode (2C A0 02) where no other means to invoke initialization exist. It is not required for the other “Operating” modes (since their exisiting initialization techniques still apply) but can be employed to invoke the unique features of this command that are not available with the normal initialization methods.
2Ch	B3h	Enable/Disable “Start of Message” Reporting (See “Commands Reported by Saint 2” for a description of this feature)	1 byte: 0x00 - Disabled **(default)** 0x01 – Enabled
2Ch	B4h	Enable/Disable 5 Baud Init Key Byte reporting. (See “Commands Reported by Saint 2” for a description of this feature)	1 byte: 0x00 – Disabled 0x01 – Enabled **(default)**

Message Corruption (Forced Error) Commands

Hdr	ID	Description	Data Bytes
2Ch	C0h	Disable Start Communications message Simulates a Fast Init error scenario where the wakeup pattern is sent without a subsequent Start Comms msg.	1 byte: 00 = Start Comm Msg Enabled **(default)** 01 = Disable Start Comm Msg Note:This command will be overridden by the "Force Initialization" command (2C B2) if the optional Start Comms substitute message is supplied. (i.e. assumes that if supplied, intent is to send regardless of this command state.)
2Ch	C1h *	Forced Format Byte Forces the use of a specified Format Byte value.	2 bytes - aa bb: aa - 0x00=Disabled **(default)** 0xnn=Enabled for nn Tx messages (nn=1-254, 01-FE) 0xFF=Enabled Continuously (until disabled) bb - New Format Byte value. **(default=0x00)**
2Ch	C2h *	Forced Length Value Forces the use of a specified msg length value. (Applied to either the Format byte or optional Length byte depending on header type.)	2 bytes - aa bb: aa - 0x00=Disabled **(default)** 0xnn=Enabled for nn Tx messages (nn=1-254, 01-FE) 0xFF=Enabled Continuously (until disabled) bb - New Length value. **(default=0x00)**
2Ch	C3h *	Forced Data Byte Value Forces the use of a specified value in the specified byte of the data payload portion of the message. (Applies to data payload bytes only, i.e. SID up to, but not including, the checksum. If specified byte number > last data byte, no data is modified.)	3 bytes - aa bb cc: aa - 0x00=Disabled **(default)** 0xnn=Enabled for nn Tx messages (nn=1-254, 01-FE) 0xFF=Enabled Continuously (until disabled) bb - Data payload byte number (base 0). **(default=0x00)** cc - New data byte value. **(default=0x00)**
2Ch	C4h *	Forced Data Bit Error Toggles the specified bit in the specified byte of the data payload portion of the message. (Applies to data payload bytes only, i.e. SID up to, but not including, the checksum. If the specified byte number > last data byte, no data is modified.)	3 bytes - aa bb cc: aa - 0x00=Disabled **(default)** 0xnn=Enabled for nn Tx messages (nn=1-254, 01-FE) 0xFF=Enabled Continuously (until disabled) bb - Data payload byte number (base 0). **(default=0x00)** cc - Bit to be corrupted 0-7 (0=lsb) **(default=0x00)**

Message Corruption (Forced Error) Commands

2Ch	C5h	Forced Byte Length Error Simulates a transmission error where a byte of the message is transmitted either short one bit or with an extra bit. (Applies to any byte of the message including header and checksum bytes.)	3-4 bytes aa bb cc dd: aa - 0x00 = Disabled **(default)** 0xnn = Enabled for nn Tx messages (nn=1-254, 01-FE) 0xFF = Enabled Continuously (until disabled) bb - 0x00 = Enable Short byte (7 bit) **(default)** 0x01 = Enable Long byte (9 bit) cc or ccdd - Byte number as a 1 or 2 byte unsigned integer (0-259, base 0). **(default=0)** Examples: 2C C5 01 00 0B - enabled for 1 msg, short byte error, corrupt 13th byte (byte #12 base 0). 2C C5 FF 01 01 03 - always enabled, long byte error, corrupt 260th byte (byte 259 base 0)
2Ch	C6h *	Omit Checksum Note: When enabled, this overrides commands C7 & C8 (below).	1 byte - aa: aa - 0x00=Disabled **(default)** 0xnn=Enabled for nn Tx messages (nn=1-254, 01-FE) 0xFF=Enabled Continuously (until disabled)
2Ch	C7h *	Forced Checksum Value Note: When enabled, this overrides command C8 (below) and is overriden by command C6 (above).	2 bytes - aa bb: aa - 0x00=Disabled **(default)** 0xnn=Enabled for nn Tx messages (nn=1-254, 01-FE) 0xFF=Enabled Continuously (until disabled) bb - New Checksum value. **(default=0x00)**
2Ch	C8h *	Force "Good" Checksum Allows messages containing "forced" errors to be sent with or without a checksum error. Note: This is overriden by commands C6 &C7 (above).	1 byte: 0x00 - Disabled **(default)** Original msg checksum is used. Checksum is not recalculated to include forced error values. (will yield a msg checksum error) 0x01 - Enabled Checksum recalculated to include the forced msg values/errors. (no checksum error)
2Ch	C9h *	Enable/Disable Extra Byte Transmission after checksum.	2 bytes aa bb: aa - 0x00 = Disabled **(default)** 0xnn = Enabled for nn Tx messages (nn=1-254, 01-FE) 0xFF = Enabled Continuously (until disabled) bb - Value of extra byte to be transmitted

Message Corruption (Forced Error) Commands			
2Ch	CFh	Global Forced Error Control "Globally" Enables, Disables, or clears all forced errors.	1 byte: 0x00 - Disabled Forced msg errors are globally disabled. (Individual Forced Error states are retianed) 0x01 - Enabled **(default)** Forced msg errors are globally enabled. 0x02 - Clear All Clears the individual enables for all forced errors. Forced errors must be individually re-enabled after sending this command. Global enable/disable is left in it's current state but will have no affect until errors are individually re-enabled.

***** - Not Applicable (ignored) in "Direct" Operational Mode (2C A0 02).

Commands Reported by the Saint 2			
Hdr	**ID**	**Description**	**Data Bytes**
2Ch	00h	Fast Wakeup Pattern Detected	None Note: **This ONLY reports Wakeup events SENT by the Saint 2.** It does NOT detect or report bus wakeup events generated by other devices/nodes on the bus.
2Ch	01h	Keyword 2000 Timeout/Error Note: P'x' timeouts are now always reported regardless of the bus init state. (previously reported only when the bus was initialized) As such, it is important to note that timeouts do not constitute a difinitive indication of the current or previous init state.	1 byte: Bitfiield Bit values=0: No error/timeout detected Bit values=1: (see detail below) **Bit 0 (0x01) - P1 Timeout** Inter byte timeout occurred for ECU (time between bytes > P1max, 20ms nominal). **Bit 1 (0x02) - P2 Timeout** Timeout occured between Tester and ECU or two ECU responses (> P2max, 50ms nominal). **Bit 2 (0x04) - P3 Timeout** Timeout occured between end of ECU response and start of Tester (> P3max, 5000ms nominal). **Bit 3 (0x08) - Checksum Error** Checksum error detected in received message. **Bit 4 (0x10) - Arbitration Failure/ Error** Arbitration error detected in transmitted message. **Bit 5 (0x20) - Receive Buffer Overflow Error** Receive buffer overflow error detected while receiving message. **Bit 6 (0x40) - Receive Framing Error** Framing errror detected while receiving message. **Bit 7 (0x80) - P4 Timeout** Inter byte timeout occurred for TESTER (time between bytes > P4max, 20ms nominal).

Commands Reported by the Saint 2

2Ch	F0h	"Start of Message" Report When enabled, this report indicates the start of a KW Tx/Rx message. **(Disabled as default)** Ref. command "2C B3", under "General/Misc. Commands" for control of this report.	1 Byte: (Message Type) 00 – Rx Msg Started 01 – Tx Msg Started Reported once per message after the first byte of a message has been sent or received. Return of a message to the host constitutes an "end of message" indication. This report provides an additional indication at the start of a message. This feature was added primarily for J2534 to support external protocol timing and control operations but is available for other uses as appropriate. Examples: 2D F0 00 - 1st byte of Rx msg received. 2D F0 01 - 1st byte of Tx msg transmitted.
2Ch	F1h	5 Baud Key Byte Report Reports the Key Byte values received during 5 Baud Init. **(Enabled as default)** Ref. command "2C B4", under "General/Misc. Commands" for control of this report.	2 Bytes: KB1 & KB2 Reported once for each 5 Baud Init event. This feature was added primarily for J2534 to support external protocol timing and control operations but is available for other uses as appropriate.

10.3 Keyword 2000 Messages

10.3.1 Transmitted Keyword 2000 Messages

The following is the format for sending a Keyword 2000 message:

Saint Header	**SID (Service ID)**	***N Data Bytes***
28h	Byte 2	*Bytes 3 —(2+N)*

The above information is used to form one of the following Keyword 2000 messages which is sent onto the Keyword 2000 bus. The message format depends on the header options configuration (user selected).

1 Byte Header			
Format	***SID***	***N Data Bytes***	**Checksum**
Byte 1	Byte 2	Bytes 3 - (2+N)	Byte 4 + N

2 Byte Header				
Format	**Length**	***SID***	***N Data Bytes***	**Checksum**
Byte 1	Byte 2	Byte 3	Bytes 4 - (3+N)	Byte 5 + N

3 Byte Header					
Format	**Target**	**Source**	***SID***	***N Data Bytes***	**Checksum**
Byte 1	Byte 2	Byte 3	Byte 4	Bytes 5 - (4+N)	Byte 5 + N

4 Byte Header						
Format	**Target**	**Source**	**Length**	***SID***	***N Data Bytes***	**Checksum**
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Bytes 6 - (5+N)	Byte 6 + N

Also, the SAINT will check to see if the bus is “initialized” and if necessary will send the Fast initialization sequence (transmit Wakeup Pattern of 25msec logic low and 25msec logic high followed by a Start Communications request message 81 11 F1 81 04). Refer to the “Keyword Protocol 2000 Data Link Layer Recommended Practice” section 5.1.5.3 for more information.

10.3.2 Received Keyword 2000 Messages

The following are the 4 possible formats of a received Keyword 2000 message shown with time stamp information included.

Keyword Header Format = 1 Byte						
Header	**Format**	***SID***	***N Data Bytes***	**Checksum**	**Time MSB**	**Time LSB**
29h	Byte 2	Byte 3	Bytes 4 - (3+N)	Byte 4+N	Byte 5+N	Byte 6+N

Keyword Header Format = 2 Byte							
Header	**Format**	**Length**	***SID***	***N Data Bytes***	**Checksum**	**Time MSB**	**Time LSB**
29h	Byte 2	Byte 3	Byte 4	Bytes 5 - (4+N)	Byte 5+N	Byte 6+N	Byte 7+N

Keyword Header Format = 3 Byte								
Header	**Format**	**Target**	**Source**	***SID***	***N Data Bytes***	**Checksum**	**Time MSB**	**Time LSB**
29h	Byte 2	Byte 3	Byte 4	Byte 5	Bytes 6 - (5+N)	Byte 6+N	Byte 7+N	Byte 8+N

Keyword Header Format = 4 Byte									
Header	**Format**	**Target**	**Source**	**Length**	***SID***	***N Data Bytes***	**Checksum**	**Time MSB**	**Time LSB**
29h	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Bytes 7 - (6+N)	Byte 7+N	Byte 8+N	Byte 9+N

10.3.3 Format Byte

The format byte contains a 6-bit length information and a 2-bit address mode information.

Bit 7	Bit 6	Address Information
0	0	no address informatio
0	1	exception mode (CARB)
1	0	with address information, physical addressing
1	1	with address information, functional addressing

Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Length
X	X	X	X	X	X	Define length of message from the Data to Checksum (not

included) bytes. A message length of 1 to 63 bytes is possible. If bits 5-0 are zero then the additional length byte is included. Refer to the “Keyword Protocol 2000 Data Link Layer Recommended Practice” section 4.1.1 for more information.

10.3.4 Target Address Byte

This is the target address for the message and is always used together with the source address byte. Refer to the “Keyword Protocol 2000 Data Link Layer Recommended Practice” section 4.1.2 for more information.

10.3.5 Source Address Byte

This is the address of the transmitting device which is a physical address. Refer to the “Keyword Protocol 2000 Data Link Layer Recommended Practice” section 4.1.3 for more information.

10.3.6 Length Byte

This byte is provided if the length in the header byte (bits 0-5) is set to 0. It allows the user to transmit messages with data fields longer than 63 bytes. With shorter messages it may be omitted. This byte defines the length of a message from the beginning of the data field (SID included) to the checksum byte (not included). Refer to the “Keyword Protocol 2000 Data Link Layer Recommended Practice” section 4.1.4 for more information.

10.3.7 Data Bytes

The data field may contain up to 255 bytes. The first byte of the data field is the Service Identification Byte. Refer to the “Keyword Protocol 2000 Data Link Layer Recommended Practice” section 4.2 for more information.

10.3.8 Checksum Byte

The checksum byte inserted at the end of the message block is defined as the simple 8-bit sum series of all bytes in the message, excluding the checksum. Refer to the “Keyword Protocol 2000 Data Link Layer Recommended Practice” section 4.3 for more information.

11 SPI

11.1 Serial Peripheral Interface (SPI) Overview

The SAINT2 supports SPI through a monitor program in which sending messages is not supported. All message traffic on the bus is “sniffed” and made available to the user. The SPI monitor is limited to baud rates of 400K or less.

The following command may be used to begin monitoring SPI traffic.

Header	ID	Description	Data Bytes
74h	01h	Begin SPI Monitor Mode	

To exit the SPI Monitor Mode use a hardware reset.

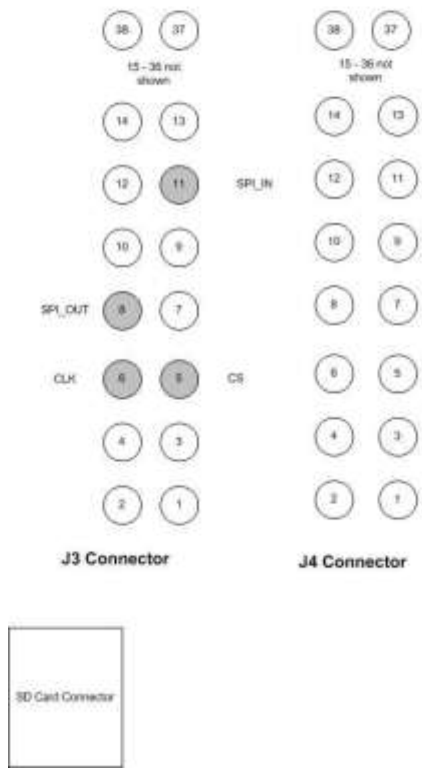
SPI Received Message Format with Time Stamp

71h (SPI IN)	N Data Bytes	Time Stamp Bytes
byte 1	Byte 2 … Byte N	Byte N+1 Byte N+1

73h (SPI OUT)	N Data Bytes	Time Stamp Bytes
byte 1	Byte 2 … Byte N	Byte N+1 Byte N+1

Note: SAINT2 SPI chip select (CS) is active low.

11.2 SPI Hardware Connection Diagram



12 LIN

The Saint 2 LIN implementation provides LIN functionality similar to that provided by the Saint 1. The Saint 2 release is an enhanced implementation that provides several significant improvements and benefits:

- Support for ALL existing LIN versions (up to 2.1) as well as SAE J2602 (US LIN).
- Simultaneous Master, Slave, and Bus Monitor support (no re-flash/mode switching).
- Stand alone simulation of a LIN sub bus network (up to 10 slave node/PID responses).

!!!!! IMPORTANT !!!!!

You MUST have a Saint 2 with hardware "Version 1.1" or higher to use it with LIN.

The hardware version number is listed on the bottom left corner of the front label. If the label does not indicate a hardware version, it is Version 1.0 hardware and will NOT work properly for LIN.

If you need to use LIN and have an incompatible version of the Saint 2, contact the manufacturer or email support for upgrade/replacement options.

12.1 LIN – Application Notes

1. **General Use Information**

<u>LIN & KW2000 share the same the same driver H/W and connector pin (PIN 7)</u>

<u>LIN & KW2000 and are MUTUALLY EXCLUSIVE protocols.</u>

LIN and KW2000 share the same physical interface and are therefore mutually exclusive protocols. Only one of these protocols may be enabled at a time. Enabling one automatically disables the other. All other supported protocols can be enabled simultaneously with either LIN or KW but not both. (i..e. CAN & LIN or CAN & KW can operate at the same time but not CAN & LIN & KW for example)
By default (or after reset) KW is enabled and LIN is disabled. The S2 Configuration Command “08 A3 B8 xx …” MUST be used to enable LIN (and any other desired protocols) before it can be used. Rrefer to the SAINT2 Configuration section.

2. **Receive Message (frame) Termination & Timestamp Accuracy**
LIN Basic does not have access to the PID/frame length information of the “LIN Definition File” (.ldf) to use for determining when a frame is complete. Therefore, frame completion is determined using timeout, break detection, and maximum frame length techniques (whichever occurs first). Due to this, the following conditions apply to timestamp values reported for LIN frames. This should be understood and accounted for by users before making timing critical judgements based on the timestamp information:

<u>Timestamp values MAY NOT accurately reflect the actual frame time.</u>

Timestamps reported with LIN Frames will normally be longer than the actual frame time (except when terminated by max frame length).

Timestamps due to timeout or break detection can be greater than actual frame times by up to 40% of the maximum frame time for a maximum length frame (at the given baud rate).

3. **Master Node Emulation Techniques (i.e. Master Schedule Table Emulation)**
An actual Master Node sends request/broadcast frames in a deterministic periodic fashion implemented as “Master Schedule Tables”. Users can, within limitations, emulate Master Schedule Table behavior on the Saint 2 using a couple of different techniques:

A) Saint Monitor Periodic Message Feature

The "Periodic Message" feature of the "Saint Monitor" plugin application can be configured to send multiple messages at individual periodic rates. Multiple periodic message files can be created and saved to allow a limited form of table switching. (Table switching would be manually controlled). PC to Saint communication latency can impose some timing inaccuracy which may or may not be of significance depending on the application. This technique is relatively flexible and would work well for all but the most demanding applications.

B) Saint Embedded Periodic Message Feature

Periodic message scheduling similar to that of the "Saint Monitor" can be configured and implemented to execute directly on the Saint 2 hardware (embedded firmware execution). Reference the "SAINT2 Configuration Commands" section, commands 0x90 & 0x91 for configuration information.
This method provides the benfit of better timing accuracy but limits the number of messages that can be configured for periodic transmission.

C) Custom Plugin Development

Users can develop their own custom plugins to allow emulation of their particular Master Node ECU. This would provide the most complete emulation including creation and full automatic control of schedule tables and switching. As with the "Saint Monitor" application, this method is also subject to potential timing inaccuracies due to the PC to Saint2 communication latencies.
Information and Plugin code templates can be found on the Saint 2 web site.

12.2 LIN Commands

The following commands may transmitted from the host interface to the SAINT 2:

Protocol Configuration Commands			
Hdr	**ID**	**Description**	**Data Bytes**
BCh	01h	Set LIN Version For “LIN Basic” this primarily affects checksum model used: <= 1.X - “Classic” >= 2.X - “Enhanced” (i.e. any value below 2.x should work for 1.x protocols, and any value above 1.x should work for 2.x or J2602 protocols.)	1 byte bitfield: Version as hex (no decimal point) Bits 0-3: Minor version number Bits 4-6: Major version number Bit 7: 0=LIN ver, 1= SAE J2602 ver Range: 0x10 – 0xFF **Default: 0x21 – LIN 2.1** Examples: BC 01 13 - set LIN ver 1.3 BC 01 20 - set LIN ver 2.0 BC 01 80 - set SAE J2602 ver 0.0 BC 01 92 - set SAE J2602 ver 1.2
BCh	02h	Set Baud Rate	1-2 bytes: Baud rate in hundreds units expressed in BCD . Leading zeros are disregarded. Range: 1k - 20k baud (in 100 baud steps) **Default = 19200 baud** Examples: BC 02 02 00 - set baud to 20,000(leading 0's ignored) BC 02 00 10 - set baud to 1000 (leading 0's ignored) BC 02 or 2C 02 00 - set default baud rate
BCh	03h	Enable/Disable Automatic Checksum Controls automatic calculation and appending of a checksum to Tx (Master) frames. Disabling this allows the user to send Tx frames with an omitted or custom checksum value.	1 byte: 0x00 = Disable 0x01 = Enable **(default)** Note: This does NOT affect Rx frames. The checksum Is always verified for received (and looped back Tx) frames. Examples: BC 03 00 - Disable Auto Checksum BC 03 01 - Enable Auto Checksum

Slave Signal Table (SST) Configuration Commands (Slave Emulation)

Hdr	ID	Description	Data Bytes
BCh	10h	Add/Modify Slave Emulation Response Adds or modifies a Slave emulation record in the SST which is used to generate simulated slave responses to Master "request" frames. When a Master "request" frame PID match is detected, the content of the matching record is transmitted as an "in frame response" to the Master frame. NOTE: User is responsible for avoiding PID conflicts between the SST and Master broadcast frames or bus collisions can result. (The Saint will transmitt a slave response over the top of a broadcast message -both nodes transmitting).	1-10 bytes: PID + Data (1-8) + Checksum (optional) Minimum content: PID + 1 Data (2 bytes) The SST holds 10 entries maximum. Commands that are invalid, illegal, or would violate table size limits will be discarded/ignored (without notificiation). PID: 0x01-0xFE (00&FF reserved for internal use) Any value within this range (incl. "illegal" PIDs) is allowed. Unique PIDs are enforced (no duplicates). If the specified PID already exists, the record is replaced (modified). Data: 0x00-0xFF (1-8 bytes) Checksum: 0x00-0xFF (optional) Checksums are NOT automatically generated for emulated slave responses. Inclusion/content is completely at the users discretion allowing full control of the slave response portion of the frame. Example: BC 10 C1 01 02 3D - add/modify "C1" PID Response Result: BC C1 ("C1" Master Request) 01 02 3D (emulated slave response) B9 C1 01 02 3D 00 (resulting LIN msg frame)
BCh	11h	Delete SST record	1 byte: PID 0x00 = Illegal, command will be discarded. 0x01-0xFE= Delete specified record (PID) 0xFF= Delete all records in the table. Examples: BC 11 C1 – Delete record w/ PID=C1 (if found) BC 11 FF – Delete ALL records in SST.

BCh	12h	Enable/Disable SST record This command allows the user to control emulated slave responses already stored in the SST.	2 bytes: PID xx PID: 0x00 = Illegal, command will be discarded. 0x01-0xFE= Apply setting to specified record (PID) 0xFF= Apply setting to all records. xx: 0x00 = Disable **(default)** 0x01 = Enable Examples: BC 12 C1 00 - Disable SST record w/ PID=C1 BC 12 C1 - (same as above – default=disable) BC 12 C1 01 - Enable SST record w/ PID=C1 BC 12 FF 01 - Enable all records in SST BC 12 FF - Disable all records in SST
BCh	13h	Report SST Config Reports the current configuration of the SST as a list of the populated records.	1 byte: PID PID: 0x00 = Illegal, command will be discarded. 0x01-0xFE= Report specified record (PID) 0xFF= Report all (populated) records. Report messages are returned first followed by an echo of the command message. If no populated entries are found, only the command echo will be returned (without report messages). Response: 1 message for each populated entry (10 max) BC 13 aa bb cc xx…xx aa = Record # (01-0A, 1-10dec) bb = Enable/Disable state (0=disabled,1=enabled) cc = PID xx.. = Record Data (including checksum) Example (assume 2 records populated): BC 13 FF – Request SST Config (all records). Result: BC 13 01 01 C1 01 02 3D BC 13 09 00 C2 01 02 3E BC 13 FF – Orig. Command Echo (Note: The initial command message will follow the record reports)

Commands Reported by the Saint 2			
Hdr	**ID**	**Description**	**Data Bytes**
BCh	E0h	LIN Error (Rx msg independent)	1 byte: Same as Rx msg Completion Code (See the Completion Code detail in the "LIN Received Messages section) This is a special error message used to report errors when an accompanying/associated Rx message is not available to report the errors (completion code) with.

12.3 LIN Messages

12.3.1 Transmitted LIN Messages (Master Frames)

The following is the format for sending a LIN Master message:

Saint Header	**PID (Protected ID)**	***N Data Bytes (N=0-8)***	Checksum (optional) (1)
B8h	Byte 2	*Bytes 3 —(2+N)*	Byte N+3

A LIN Master frame is a minimum of 1 byte (PID only = Master Request Frame) and a maxium of 10 bytes (PID + data + checksum = Master Broadcast Frame). A properly formatted S2 LIN message will therefore be between 2 – 11 bytes (including the S2 header byte).

Master frames are sent IMMEDIATELY and will be transmitted on the bus regardless of any current receive activity (i.e. potential collision conditions). This is intentional behaviour and in keeping with the LIN specifications and philosophy that the Master Node always has priority and can pre-empt any bus activity. It is the users responsibility to manage bus traffic and schedule message flow appropriately.

Notes:

(1) User provided checksum is optional. Inclusion is dependent on user intent AND the configured state of the "Automatic Checksum" feature. If Automatic Checksum is enabled, a user checksum should NOT be included (as one will be automatically calculated and appended by the firmware). If Automatic Checksum is disabled, inclusion of the user checksum is optional. This mode allows the user to send a Master frame with an omitted, corrupted, or valid checksum value as desired (useful for error/robustness testing.)

12.3.2 Received LIN Messages (frames)

The following is the format of a received LIN message (shown with time stamp information included):

Header	*PID*	*N Data Bytes*	Checksum	Completion Code	Time MSB	Time LSB
B9h	Byte 2	Bytes 3 - (2+N)	Byte 3+N	Byte 4+N	Byte 5+N	Byte 6+N

12.3.3 PID Byte

The “Protected Identifier” byte consists of the Frame ID (bits 0-5) and Parity fields (bits 6-7). Refer to the applicable LIN (LIN Consortium) or J12602 (SAE)specifications for more information.

12.3.4 Data Bytes

The data field may contain from 0 to 8 bytes. When supplied by the Master Node, the message constitutes a “Broadcast” message (for Slave Node consumption). When not supplied by the Master Node (PID only), the message constitutes a “Master Request” message (Slave Node response expected). Refer to the applicable LIN (LIN Consortium) or J12602 (SAE)specifications for more information

12.3.5 Checksum Byte

The checksum is calculated as an eight bit sum with carry. Two different checksum models are used depding on the LIN/J2602 version being used as follows:

Checksum Model	Checksumed Bytes	LIN Version
Classic	Data (only)	LIN 1.X
Enhanced	PID & Data	LIN 2.X & J2602

Refer to the applicable LIN (LIN Consortium) or J12602 (SAE)specifications for more information.

12.3.6 Completion Code (Frame Error) Byte

This byte is a bitfield indicating any errors detected by the Saint 2 firmware for the associated LIN message (frame). In certain cases, errors can occur without producing an Rx message to report the errors with. In this scenario, the errors (completion code) are reported to the user as an independent error message “BC E0 xx” (described earlier in the “LIN Commands” section).

Completion Code Bitfield Definition							
7 (MSb)	6	5	4	3	2	1	0 (LSb)
Overflow	Framing	LoopBack	Brk/Sync	Parity	Checksum	Tframe	SNR

The following provides additional detail regarding each of these errors:

LIN Completion Code/Error Bitfield Detail:

Bit 0 (0x01) – Slave Not Responding Timeout

0 No timeout detected.

1 Slave node(s) did not respond within the allotted time (40% of max frame time @ given baud).

Bit 1 (0x02) – Tframe Error

0 No error detected

1 LIN frame was not completed within the max allowable time (frame slot time exceeded).

Bit 2 (0x04) – Checksum Error

0 No error detected.

1 Checksum error detected in the LIN message.

Bit 3 (0x08) - Parity Error

0 No error detected.

1 PID parity error detected in the LIN message.

Bit 4 (0x10) – Break/Sync Error

0 No error detected.

1 Missing or inconsistent Break and/or Sync bytes detected.

Bit 5 (0x20) – LoopBack Error (AKA Data Error)

0 No error detected.

1 Received (loopback) byte did not match transmitted byte during LIN message transmission.

Bit 6 (0x40) - Receive Framing Error

0 No error detected.

1 Framing errror detected in the LIN message.

Bit 7 (0x80) - Receive Buffer Overflow Error

0 No error detected.

1 Receive buffer overflow error detected while receiving LIN message.

13 Block Transfer

The Block Transfer messages (SAINT2 header $F8) can be used to transfer large blocks of data between the host PC and certain protocols running in the SAINT2 firmware.

Maximum Block Transfer message size = 4K bytes (0xFFF)
Protocols that support block transfer:
Class 2 ($60) – block transmit and block receive supported
ISO15765-2 1($C0) – block transmit supported, block receive is not supported (see ISO15675-2 section for details)
ISO15765-2 2($C8) – block transmit supported, block receive is not supported (see ISO15675-2 section for details)

13.1 Using Block Transfer with the SAINT2 Bus Engine

If your plug-in is using the SAINT2 Bus Engine to communicate with the SAINT2 you do not need any special formatting to send your block message. You simply need to send the SAINT2 header byte for your protocol followed by the entire block message.

13.2 Using Block Transfer without the SAINT2 Bus Engine

If your application does not use the SAINT2 Bus Engine to communicate with the SAINT2 you must use the following special formatting to send or receive your block message. A single block message must be broken up into a series of $F8 messages. Also, the messages must be constructed using the SAINT2

Maximum $F8 message size = 56 bytes (including $F8 header byte, $HH and $ YY YY below)

$F8 **HH YY YY XX XX XX**….
where $**F8** is the SAINT2 header for block message transfers
$**HH** is the SAINT2 header for the protocol (i.e. $60 for Class 2, $C0 for ISO15765-2 1)
$**YY YY** is the incrementing message counter. Bits 0-14 are $0000 for the first message and increment by 1 for each subsequent message. Bit 15 = 1 in the last message of the series of messages.
$**XX XX XX**… is the serial message (up to 52 bytes)

Example: Transmit the following 170 data byte ISO15765-2 message $123 DB1 DB2 … DB170
$F8 C0 00 00 01 23 DB1 DB2 DB3 DB4 … DB54
$F8 C0 00 01 DB55 DB56 DB57 … DB110
$F8 C0 00 02 DB111 DB112 DB113 … DB166

$F8 C0 80 03 DB167 DB168 DB169 DB170

Also note that each message must be constructed using the SAINT2 FF and FF 00 escape characters. See the SAINT2 Message Format section for more details.

Class2 Example:

Users wants to transmit the following 140 byte Class2 block mode message:

```
60 6D 99 F0 36 00 00 80 00 F6 86 79 9F AC 00 84 05 AA 03 00
00 0E C7 14 C7 12 C7 80 81 C0 00 00 C0 00 00 FF CF 00 00 00
F6 00 00 FF FD 00 00 00 00 E0 00 FF 07 E0 00 00 40 C0 00 FF
45 C0 00 00 60 C0 00 FF FF C0 00 00 00 C1 00 FF FF C1 00 01
02 03 04 05 06 10 11 12 13 14 15 16 17 18 19 1A 1B 1C FF 9C
8C 9C 8C 9C 8C 9C 8C 9C 8C 88 8E C0 00 A0 8E C0 00 B4 8E C0
00 01 02 00 C7 C7 02 00 C7 01 C7 00 02 1A 90 C0 00 5A 8F 2F
1F
```

The Saint Bus Engine will actually break this message up into three different packets for USB transfer. Reminder: If you are bypassing the SBE you will need to account for the FF and FF 00 escape sequences.

```
F8 60 00 00 6D 99 F0 36 00 00 80 00 F6 86 79 9F AC 00 84 05
AA 03 00 00 0E C7 14 C7 12 C7 80 81 C0 00 00 C0 00 00 FF CF
00 00 00 F6 00 00 FF FD 00 00 00 00 E0 00 FF 07
```

```
F8 60 00 01 E0 00 00 40 C0 00 FF 45 C0 00 00 60 C0 00 FF FF
C0 00 00 00 C1 00 FF FF C1 00 01 02 03 04 05 06 10 11 12 13
14 15 16 17 18 19 1A 1B 1C FF 9C 8C 9C 8C 9C 8C
```

```
F8 60 80 02 9C 8C 9C 8C 88 8E C0 00 A0 8E C0 00 B4 8E C0 00
01 02 00 C7 C7 02 00 C7 01 C7 00 02 1A 90 C0 00 5A 8F 2F 1F
```

14 SAINT2 Gateway Functions

The SAINT2 supports several standalone gateway modes. In a gateway mode, the SAINT2 operates as a translator between two serial buses without any interface from a PC.

- An SD card is used to configure the SAINT2 to act as a stand alone gateway.
- The gateway configuration file must be named *CONFIG.TXT*.

14.1 CAN1/CAN2 Gateway

The CAN1/CAN2 gateway is a gateway between the CAN1 channel and the CAN2 channel. The SAINT2 can translate between supported baud rates and CAN transceivers. Any CAN frames received on CAN 1 are re-transmitted with the same message ID and data onto CAN 2. Also, any CAN frames received on CAN 2 are re-transmitted with the same message ID and data onto CAN 1. The CONFIG.TXT file should look like this:

```
begin
gateway
cangateway
header1: 50
header2: 58
group file: cancnfg.grp
end
```

where the goupfile *cancnfg.grp* configures the CAN channels' baud rates, CAN transceivers, and options. An example of *cancnfg.grp* might be

```
0010 54 04 00
0010 54 01 C9 39
0010 5C 04 01
0010 5C 01 F1 39
```

Where CAN 1 ($54) is configured to high speed CAN at 500K baud, and CAN 2 ($5C) is configured to fault tolerant CAN at 100K baud.

Important Notes:

- At some level of bus utilization, a slower baud rate channel is not going to be able to keep up with traffic on a higher baud rate channel. Messages may be lost. If this is critical, pay attention to the operation warning LED #5.

- If you are monitoring the bus with the SAINT2 being used as a gateway, you will not see the received CAN frames but you will the see the frames that are re-transmitted.

14.2 RS232/CAN Gateway

The RS232/CAN Gateway is a gateway between RS232 and the specified CAN channel. In this mode, the SAINT2 can be used to monitor CAN traffic using the PC application Hyperterminal. When any CAN frame (i.e. any CAN ID) is received on the specified CAN channel, the first data byte in the frame is retransmitted as a single byte over the RS232 connection. When a byte is received over RS232, a CAN frame is constructed with the specified message ID (0x001) and the RS232 byte of data and is re-transmitted over the specified CAN channel. The CONFIG.TXT should look like this

```
begin
gateway
can11bit                ; or "can29bit"
can_id: 0x001           ; can id from hyperterminal to CAN
header1: 0x50
header2: 0x80
group file: cancnfg.grp
end
```

where cancnfg.grp is the groupfile that configures the CAN channel.

Important Notes:

- CAN has the potential to be much faster than RS232. The CAN device connected to the gateway must regulate the CAN traffic so it does not overrun the RS232 channel.

- header1 must be the CAN channel and header 2 must be the RS232 channel ($80).

15 Connectors

The SAINT2 uses 3 connectors. The first connector is a USB connector. The second connector is for the RS-232 link and is DB9F. The third connector is a DB25F for the serial bus, power and ground connections. The following pin outs are used:

15.1 USB Connector

Uses USB Type A connector on host PC and Type B connector on Saint 2.

15.2 RS-232 Connector

A straight pass cable should be used to connect the SAINT2 to the host computer.

DB9F Pin #	Use
2	TX to Host
3	RX from Host
8	CTS
6	DSR
5	GND

15.3 SAINT CABLE

DE25F Pin #	Use
1	SWCAN1
2	FT_2_H
3	VBATT
4	TRIGIN
5	SWCAN2
6	BarTC2
7	KEYWORD
8	CAN_1_H
9	CAN_1_L
10	BArT_CANH
11	BArT_CANL
12	BArT_SW
13	FT_2_L
14	CLASS2

15	ACP_B
16	BEAN1
17	BEAN2
18	Do not connect a wire to this pin
19	Do not connect a wire to this pin
20	Do not connect a wire to this pin
21	Do not connect a wire to this pin
22	SPI_INT
23	GND
24	TRIGOUT
25	ACP_A

16 LEDs

Six LEDs are used by the SAINT2. These LEDs are:

LED #	Purpose
6	Not Used
5	Operation Warning Code Set In Firmware (Use 08h A1h to Retrieve)
4	Access SD Card
3	Not Used
2	Sending Message(s) To Host
1(PWR)	Toggle – Power On & System In Normal Operation

16.1 Reset Button - Manual Reset

Reset button is the manual reset switch. Pressing this switch will cause SAINT microprocessor software reset - start executing the embedded code from boot block.

17 Software Packages

The following table lists the latest firmware releases for the SAINT board.

Name	Software Version	Date	Description
Saint2 Team	001.000.000	11/29/05	Initial Firmware Release
Saint2 Team	001.011.000	03/03/06	Initial Autonomous Reflash
Saint2 Team	002.000.000	03/30/06	New boot and flash functionality
Saint2 Team	002.001.000	04/20/06	General Improvements
Saint2 Team	002.002.000	06/21/06	Class2 Autonomous Reflash / SD Card Config
Saint2 Team	002.004.000	07/28/06	GMLAN and Class2 updates / 08 A1 codes
Saint2 Team	002.005.000	09/20/06	IIC (master/slave) / new scripting commands
Saint2 Team	003.000.000	10/12/06	General Improvements
Saint2 Team	003.002.000	11/01/06	KW2K / Class2 improvements
Saint2 Team	003.003.000	02/01/07	IIC Monitor / CAN Cont. ACK disable
Saint2 Team	003.004.000	03/23/07	CAN to RS-232 gateway
Saint2 Team	003.006.000	12/05/07	SPI Monitor, LIN, Emulation, and Hyundai Reflash
SAINT2 Team	003.007.000	05/01/08	LIN, remote frame request, mods to C2 and KW for J2534
SAINT2 Team	003.008.000	6/25/08	LIN:Master/Slave/Monitor
SAINT2 Team	003.009.000	8/25/08	Stand-alone Message Monitor
SAINT2 Team	003.010.000	2/20/09	ISO15765-2 Transport Layer, CAN bus flooding
SAINT2 Team	003.011.000	6/23/09	Re-flash CRC verification
SAINT2 Team	003.012.000	8/26/09	
SAINT2 Team	003.013.000	2/12/10	
SAINT2 Team	003.014.000	5/31/10	
SAINT2 Team	003.015.000		
SAINT2 Team	003.016.000	12/9/10	