CIC61508

Safety Monitor

User's Manual

Release v2.2

Microcontrollers

Edition **Nov 2012**

Published by
Infineon Technologies AG
81726 München, Germany

Document Change History

Date	Version	Changed By	Change Description
2010-10-21	0.1	Viswanath.R	Initial Version
2010-11-08	0.2	Viswanath.R	Modules prepared Introduction, Error State Monitor, Voltage Monitor and Task Monitor
2010-11-09	0.3	Viswanath.R	Opcode test sequencer is added
2010-11-09	0.4	Viswanath.R, Bharatesh	Updated all the sections
2010-11-12	0.5	Viswanath.R, Bharatesh	Added acronyms and the abbreviations and edited all the sections.
2010-11-15	0.6	Viswanath.R, Bharatesh	updated as per Daryl's comments
2010-11-17	0.7	Viswanath.R, Bharatesh	Added the application use case
2010-11-17	0.8	Viswanath.R	Modified with the proper page breaks and with proper formats
2010-11-26	0.9	Ashish K	Incorporated review comments from Mike Beach and Christophe Bouquet
2010-12-10	0.94	M. Beach/A. Wenlock	Proofing and minor additions
2011-01-19	1.0	Ashish K	Modified Cover Page Template and updated the formula in Section 2.6.3 Added disclaimer for customization of DFLASH configuration
2011-03-22	1.1	Ashish K	Removed some confusing terms like "opcode test sequencer" and replaced them with standard terms
2011-03-23	1.2	Ashish K	Review comments incorporated. Update with respect to usage of TARDISS tool.
2011-03-24	1.3	M Beach	Review and minor reformatting
2011-03-25	1.4	M. Beach/A. Wenlock	Proofing
2011-04-11	1.5	Bharatesh	Corrected SYSDISA, SYSDISB parameters in section 2.7.1
2011-04-21	1.6	Bharatesh	Updated section 2.3.1 - SPI Communication Protocol
2012-01-18	1.7	Bharatesh	UTP AI00064054: Added section 5.1 - Selecting CIC61508 system clock frequency
2012-04-24	1.8	Bharatesh	Incorporated review comment of REV_003314. Added section 5.1 - Selecting CIC61508 system clock frequency. Updated sections 2.1.1.1 - ROM / PFLASH checksum check. 2.2.1 – Correction of CIC state. 2.3.2 - SPI Error Handling. 2.6.1 - Wake-up Timer Operation, 2.6.3 - Wake-up Timer calibration. 3 - Tuning the DFLASH NVM Configuration.
2012-05-28	1.9	Bharatesh	UTP AI00064054:Updated section 2.3 - SPI. Added 7 - Configuration guidelines
2012-05-29	2.0	Arjun Muddaiah	Updated the Table 7 in section 2.3.2

Document Change History			
Date	**Version**	**Changed By**	**Change Description**
			with worst case leading and trailing delay.
2012-07-10	2.1	Arjun Muddaiah	UTP AI00061900: Updated section 2.2.1 - Error Counters
2012-11-26	2.2	Arjun Muddaiah	UTP AI00127297: Updated the UM to follow the proper naming conventions for Error State Monitor module.

Table of Contents **Page**

Table of Contents **Page**

List of Figures **Page**

List of Tables **Page**

1 Introduction

1.1 Scope

The Safety Monitor CIC61508 Release is intended to support the CIC61508 with TriCore Architecture only. Hence all references to Safety Architecture will be with respect to TriCore Microcontroller Architecture.

1.2 Acronyms, Abbreviations and Definitions

1.2.1 Abbreviations

Abbreviation	Comment
ASIC	Application Specific Integrated Circuit
AUTOSAR	Automotive Open System Architecture
BIST	Built-in Self-Test
CIC	Companion IC
CPU	Central Processing Unit
CS	Chip Select
EPS	Electrical Powered Steering
MRST	Master Receive Slave Transmit
MTSR	Master Transmit Slave Receive
NVM	Non-Volatile Memory
PCP	Peripheral Controller Processor
PORST	Power-on Reset
RAM	Random Access Memory
ROM	Read Only Memory
SBST	Software Based Self Tests
SCLK	Serial Clock
SFR	Special Function Register
SPI	Serial Peripheral Interface
SW	Software
TARDISS	CIC61508 Test and Rapid Development for the Infineon Safety System
f_{sys}	CIC61508 System Clock Frequency

1.2.2 Definitions

Definition	Comment
Event	The condition(s) to be met to make a transition from a state to another state.
Heartbeat	All measurements are done in terms of heartbeat and this is the atomic unit of time for the CIC61508. One heartbeat is calibrated and is equal to 600µs.All the timing measurements in the CIC61508 are in terms of heartbeats.

Definition	Comment
Open window	In the Sequencer module, the open window is defined as the time period in which the test is initiated.
Closed window	In the Sequencer module, the closed window is defined as the Idle time when the CIC61508 does not expect the Sequencer trigger command (Write to OTRHH).
Maintain State	This state indicates that the specific monitor function has reached a safe state. This state is achieved if the pass counter of the respective monitor function has crossed the threshold value of 40_H.
Error State	This state indicates that the specific monitor function is not functioning properly to reach a safe state. This state is achieved if the pass counter of the respective monitor function is below the threshold value of 40_H.

1.3 References

[TARDISS] TARDISS_v2_9 User's Manual v1.6

1.4 Overview of Safety Architecture

In a safety-related system, safety integrity is based on a Challenge/Response Architecture controlled by a Safety Monitor independent of the microcontroller.

The Challenge/Response Architecture is built upon a system containing two processors. This allows it to have a layered hardware/software architecture that can be used to implement safety monitoring loops and fulfill the required hardware fault tolerance of the system. The cross-monitoring between the microcontroller and the safety monitor must be designed so that if a dangerous failure affects either the microcontroller or the safety monitor, then the safety-related system must enter the safe state, thus providing a hardware fault tolerance of one.

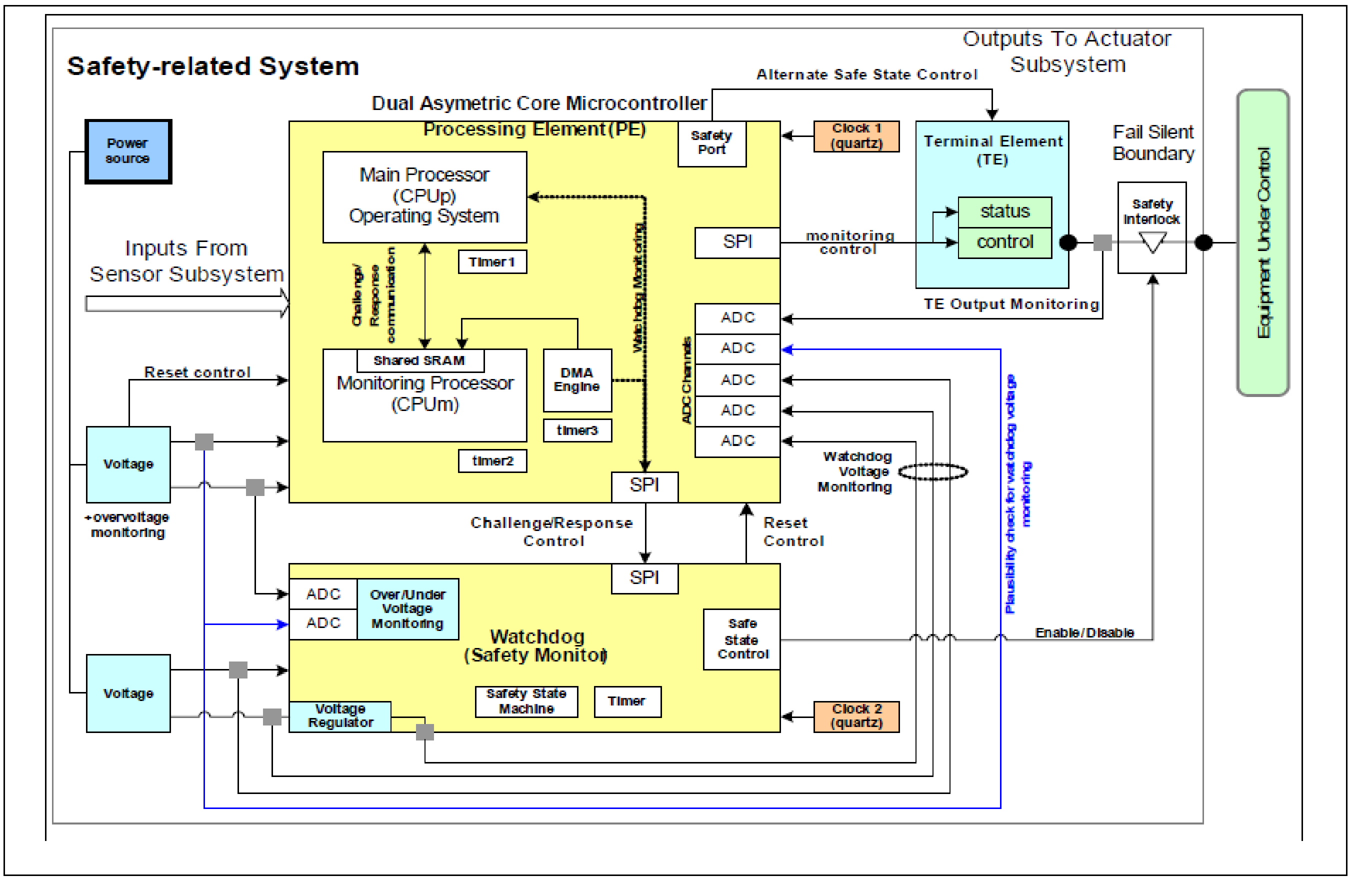


Figure 1 Block Diagram of the Safety System

The architecture presented in **Figure 1** **shows** the situation where the two processors (CPUp and CPUm), are inside the same microcontroller. This is similar to the TriCore microcontrollers where CPUp is the TriCore main CPU and CPUm is the Peripheral Controller Processor (PCP). The processor (CPUp) is responsible for the execution of all safety-related applications covering all the safety loops. The second processor (CPUm) acts as monitoring processor covering the execution integrity (mainly program sequence monitoring) of the main processor. Because both CPUp and CPUm are in the same silicon, some situations exist where the monitoring may fail because of common cause failures. Because of that possibility an external Safety Monitor is required to monitor the execution of CPUm. The Safety Monitor itself can be a microcontroller or an ASIC. The three components CPUp, CPUm and Safety Monitor participate in a closed monitoring loop.

1.5 Description of the CIC61508 Safety Monitor

The CIC61508 is a Companion Safety Monitor Chip to build up functional safety applications; examples include airbag, Electrical Powered Steering (EPS) and damping systems. The chip is responsible for monitoring the host microcontroller's behaviour. It can monitor the host microcontroller's power supply and verify the host microcontroller's requests. It therefore serves as a diagnostic monitoring device to allow the host microcontroller system to be SIL3 safety compliant.

The CIC61508 includes several modules such as a Sequencer, a Data Comparator, a Task Monitor, an Integrity Monitor, Built-in Self Test (BIST), 4 Voltage Monitors and Reset Path Control by Wake-up Timer. In addition to these, CIC61508 will monitor the communication between the CIC61508 and the Host.

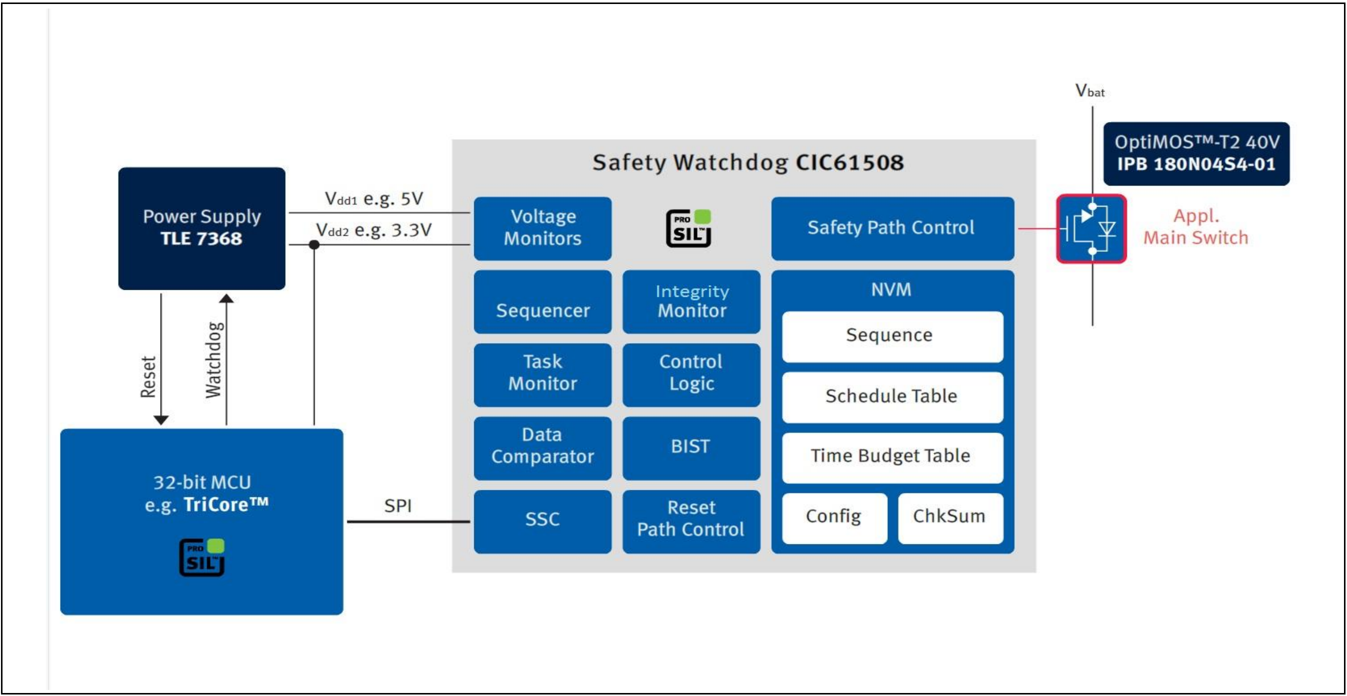


Figure 2 Block Diagram of CIC61508

The Sequencer is responsible for monitoring the sequence of answers generated by the Host. The answers generated by the Host are in response to the challenges initiated by CIC61508; these answers verify the Host Processor's integrity. The Host responds to the CIC61508 by sequentially sending a defined series of answers periodically within a defined timeframe. The Sequencer Monitor System will verify the answers against the static table stored in the CIC61508.

A Data Comparator compares two data variables delivered within a determined time period to check for an equal, greater or less than condition, based on a predefined mask value.

A Task Monitor uses a defined schedule table to check the dispatch of critical tasks running on the Host Microcontroller with predefined execution budgets. Such task deadline enforcements will allow, for example, the AutoSAR and OSEK operating systems to be used in safety applications.

Through the Voltage Monitors, the CIC61508 is also capable of detecting under- and over-voltage of the supply to the monitored microcontroller.

Communication between the Host and the CIC61508 is through the Serial Peripheral Interface (SPI). The CIC61508 screens for communication disturbances between the two.

To allow a low quiescent current for the Host Microcontroller System, the CIC61508 provides the function to wake up the Host at pre-defined intervals through a Wake-up Timer. The Wake-up Timer also provides a means to immediately reset the CIC61508 chip.

For added security, user-defined configuration parameters stored in the Non-Volatile Memory (NVM) of the CIC61508 are duplicated for redundancy. The CIC61508 also executes Built-In Self Tests (BIST) on start-up and during runtime, to ensure the correct operation of the CIC61508 chip.

The Integrity Monitor maintains the machine state of the CIC61508 based on all the other modules' functionality.

In the case of the TriCore's safety solution, the Task Monitor and Data Comparator Monitor are redundant, as the PCP controller in the Host Microcontroller (TriCore) is used instead. Hence, these 2 modules' monitoring needs to be disabled for the TriCore's safety solution. Please refer to Section 2.2.4.2 to disable monitoring of certain modules.

1.6 Feature Summary

The CIC61508 has the following features supported by software:
- Power Supply Monitor for over- and under-voltage
- Sequencer

- Task Monitor
- Data Comparison and Verification Functions
- SPI Communication Monitor
- Safety Path Control (enable/disable)
- Configurable Wake-Up Timer

1.7 Special Function Register (SFR) Mapping

CIC61508 will provide 8-bit SFRs to control and indicate the status of the CIC61508. The SFRs are mapped to 7-bit SFR addresses and accessed through SPI commands.

The SFR address mapping is as shown in **Table 1.**

Table 1 SFR Mapping

Address	SFR Name	SFR Group	Read Command	Write Command
0	OTRHH	**Sequencer Registers**	00_H	80_H
1	OTRHL		01_H	81_H
2	OTRLH		02_H	82_H
3	OTRLL		03_H	83_H
4	WINMAX		04_H	-
5	WINMIN		05_H	-
6	SEQ		06_H	-
7	SYSTEMINTEGRITY	**Integrity Monitor Registers**	07_H	-
8	PASSCNTSEQ		08_H	-
9	PASSCNTVA		09_H	-
10	PASSCNTVB		$0A_H$	-
11	PASSCNTVC		$0B_H$	-
12	PASSCNTVD		$0C_H$	-
13	PASSCNTTASK		$0D_H$	-
14	PASSCNTCOMPARE		$0E_H$	-
15	PASSCNTCOMM		$0F_H$	-
16	SUM0		10_H	-
17	SUM1		11_H	-
18	INT		12_H	-
19	MODE		13_H	93_H
20	VOLTMONAH	**Voltage Monitor Registers**	14_H	94_H
21	VOLTMONAL		15_H	95_H
22	VOLTMONBH		16_H	96_H
23	VOLTMONBL		17_H	97_H
24	VOLTMONCH		18_H	98_H
25	VOLTMONCL		19_H	99_H
26	VOLTMONDH		$1A_H$	$9A_H$
27	VOLTMONDL		$1B_H$	$9B_H$
28	TASKSTART	**Task Monitor Registers**	$1C_H$	$9C_H$
29	TASKEND		$1D_H$	$9D_H$
30	WAKERELOAD	**Wake-up Timer**	$1E_H$	$9E_H$

Address	SFR Name	SFR Group	Read Command	Write Command
31	WAKEPRESCALAR	**Registers**	$1F_H$	$9F_H$
32	DATAAHH	**Data Comparator Registers**	20_H	$A0_H$
33	DATAAHL		21_H	$A1_H$
34	DATAALH		22_H	$A2_H$
35	DATAALL		23_H	$A3_H$
36	COMPA		24_H	$A4_H$
37	DATABHH		25_H	$A5_H$
38	DATABHL		26_H	$A6_H$
39	DATABLH		27_H	$A7_H$
40	DATABLL		28_H	$A8_H$
41	COMPB		29_H	$A9_H$
42	Reserved	-	-	-
43	Reserved	-	-	-
44	SVER	Miscellaneous Registers	$2C_H$	-
45	HVER		$2D_H$	-

1.8 NVM (Non-Volatile Memory) Address Mapping

To configure the functionality of each CIC61508 monitor, the CIC61508 has 4-Kbytes of memory space (NVM). Of the 4-Kbytes memory 2-Kbytes ($A000_H$ – $A7FF_H$) is used as a main copy and the remaining 2-Kbytes ($A800_H$ - $AFFF_H$) is used as a redundant copy. Parameters used for the configuration of the CIC61508 are stored in the main copy of the NVM. The redundant copy is the inverted value of the main copy parameters. This NVM will be shared among the functions of the CIC61508. The user is required to configure the main copy of the NVM.

The 4-Kbyte memory space mapping is as shown in Table 2.

Table 2 NVM Address Mapping

Monitor Function	**Address range of main copy**	**Address range of Redundant copy**	**Number of Bytes**
Sequencer	$A000_H$ – $A142_H$	$A800_H$ – $A942_H$	323
Reserved	$A143_H$ – $A15F_H$	$A943_H$ – $A95F_H$	-
Data Comparator	$A160_H$ – $A461_H$	$A960_H$ – $AC61_H$	770
Reserved	$A462_H$ – $A47F_H$	$AC62_H$ – $AC7F_H$	-
Task Monitor	$A480_H$ – $A67E_H$	$AC80_H$ – $AE7E_H$	511
Reserved	$A67F_H$ – $A69F_H$	$AE7F_H$ – $AE9F_H$	-
Voltage Monitors	$A6A0_H$ – $A6AF_H$	$AEA0_H$ – $AEAF_H$	16
Reserved	$A6B0_H$ – $A6BF_H$	$AEB0_H$ – $AEBF_H$	-
Pass Counter Increment/Decrement Value	$A6C0_H$ – $A6CD_H$	$AEC0_H$ – $AECD_H$	14
Monitor Function Enable	$A6CE_H$ – $A6D3_H$	$AECE_H$ – $AED3_H$	6
Trip Time	$A6D4_H$ – $A6D6_H$	$AED4_H$ – $AED6_H$	3
Safety Path Control	$A6D7_H$ – $A6F6_H$	$AED7_H$ – $AEF6_H$	32
Reserved	$A6F7_H$ – $A7FF_H$	$AEF7_H$ – $AFFF_H$	-

2 Functional Description

2.1 Built-In Self-Tests (BIST)

Built-In Self-Tests are implemented in the CIC61508 to ensure system integrity at start-up (Start-up BIST) and also throughout its run-time (Background BIST). BIST ensures that the CIC61508 is fit to run and act as a safety monitor. It then performs continuous background tests to ensure that it remains operational.

2.1.1 Start-Up BIST

Start-up BIST is executed at Start-up when CIC61508 is in a RESET state.

The following tests are performed by Start-up BIST:

2.1.1.1 ROM / PFLASH checksum check

This check performs a CRC8 checksum which is calculated from the base of PFLASH/ROM address 0000_H till $2FFE_H$ ROM memory and compared against the checksum stored at $2FFF_H$.
The checksum value at $2FFF_H$ needs to be updated for any code changes in the PFLASH.

2.1.1.2 Opcode check

This check performs 8051 opcode integrity tests.

2.1.1.3 IRAM check

This check performs the MARCH C test from address 00_H till FF_H.

2.1.1.4 XRAM check

This check performs the MARCH C test from address $F000_H$ till $F1FF_H$.

2.1.1.5 DFLASH check

During Start-up BIST, the NVRAM parameters will be compared against the inverted copy.

2.1.1.6 DFLASH configuration check

This test checks for the plausibility of the NVRAM configurations.

- Valid Range of Sequencer table length (Min: 08_H, Max: 40_H).
- Sequencer Minimum Window (Min: 00_H, Max: 63_H), Maximum Window (Min: 01_H, Max: 64_H).
- Task Monitor table length should be of a maximum 255 monitored tasks.
- Data Comparator table length should be of a maximum 128 comparison tasks.
- Data Comparator, table length (Min: 0, Max: 128), Data Type (Min: 0, Max: 6) and Compare Type (Min: 0, Max: 2).
- Tripping Timeout range (Min: 00_H, Max: FF_H)
- Wakeup Prescalar Max: $0B_H$
- Voltage Monitor (Min: 0, Max: 1023)
- Checks for control bits corresponding to SYSDISA, SYSDISB (Only Port 3 bits 1 & 0 can be set), SYSDISC (Only Port 0 bit 2 can be set).
- Pass Increments (Min: 00_H, Max: $3F_H$) and Fail Decrements (Min: 00_H, Max: $3F_H$).

2.1.2 Runtime BIST (Background BIST)

Upon successful completion of Start-up BIST, the CIC61508 moves out of the RESET state. Henceforth, Runtime BIST is executed in the background whenever the CIC61508 is idle (after servicing its heartbeat service interrupt).

The following tests are performed by Runtime BIST:

2.1.2.1 DFLASH Runtime Slice Check

The Runtime BIST partitions the DFLASH main copy (lower 2K of DFLASH area) into 128 slices, where each slice is of 16 bytes. In each slice, the NVRAM parameters are compared against the corresponding inverted copy (upper 2K half of the DFLASH area). The comparison result, positive and negative, is reported to the Integrity Monitor. During every run of Runtime BIST, the incremented new slice is tested sequentially (wrap-around to the first slice at the end of the last slice).

2.1.2.2 Opcode Check

Refer to Section 2.1.1.2 for details.

2.1.2.3 System Heartbeat Check

If, for any reason, the main system heartbeat interrupt is delayed such that it becomes pending while a previous instance is still executing, a FATAL timing budget overrun event is flagged in INT SFR for the BIST. The CIC65108 then enters the DISABLED state. However, unlike other entry routes to this state, SPI communications become read-only and only a power-on reset can restart the device. Typically, the system heartbeat check is violated by SPI traffic that does not conform to the 8 messages per heartbeat limit.

2.1.3 BIST Failure

If any of the above Start-up/Runtime BIST tests detects any failure, it is a FATAL error and the system will be brought immediately into the Disabled State. A FATAL event will also be flagged in INT SFR. The pin states of SysDisA, SysDisB and SysDisC will be set to DISABLED start.

2.2 Integrity Monitor

2.2.1 Pass Counters (PASSCNTXX)

The Integrity Monitor is at the heart of the CIC61508. It will monitor all the CIC61508 functions.
It consists of eight pass counters which monitor the five main functions of the CIC61508:

- Sequencer
- Data Comparator
- Task Monitor
- Four Voltage Monitors
- SPI Communication Monitor

These counters will increment and decrement according to the pass or fail conditions of respective functions. The pass counters are initialized at 1 and run between counts of 1 and 128 (80_H), but they will never underflow nor overflow. Therefore, incrementing (or decrementing) an pass counter that has the value 80_H (or 01_H) will see the pass counter still retaining the value 80_H (or 01_H), since an overflow (or underflow) is not possible. These pass counters will be associated with the eight pass counter registers. The current Counter Value for each monitor function can be obtained from the respective PASSCNTXXX SFRs. These Counter SFRs are updated every 600µs (heartbeat).

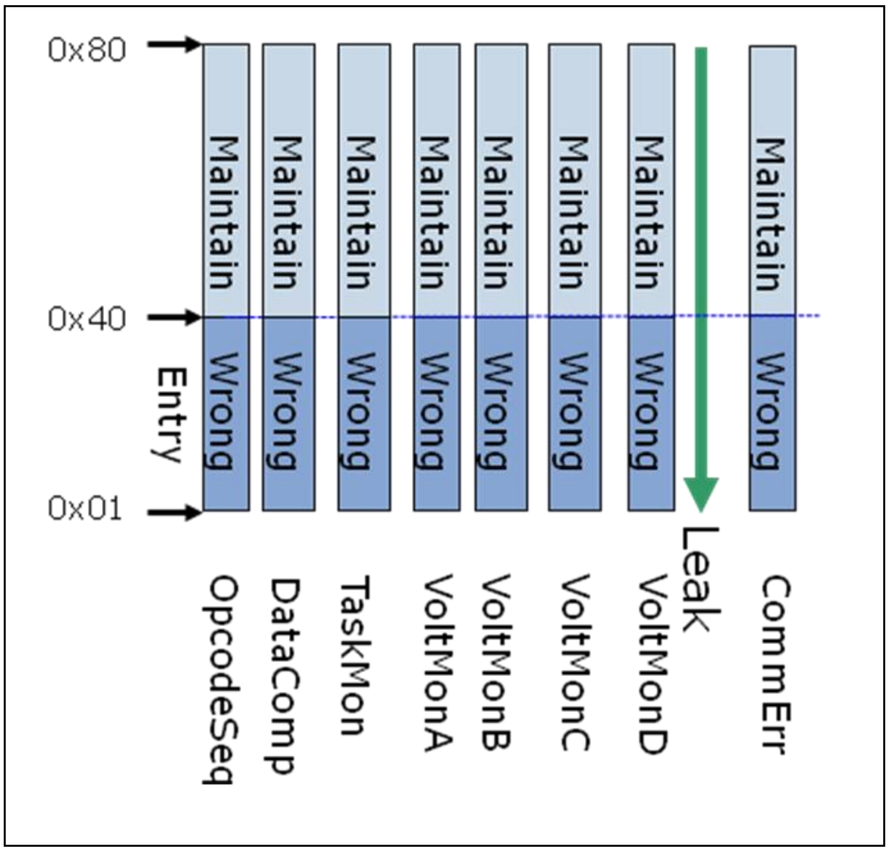


Figure 3 Integrity Monitor – The Eight Pass Counters

During the execution of the monitor functions, the pass counters are incremented / decremented by a predetermined configured value in the NVM, which may be different for each pass counter, when a pass or fail event for the respective function occurs. This happens irrespective of any state other than the RESET and Secure SPI state. The SPI Communication Monitor counter will never increment, but will be decremented by 01_H upon the SPI communication error. The SPI Communication counter value can be set to

80_H by the Host writing the SPI Reset Request to the Mode SFR. If this is not done, the Ready state can never be reached as the SPI communications pass counter will remain at 0x01.

In order to ensure that all the functions will happen periodically, the CIC61508 will provide an aging mechanism, so that pass counters will be decremented by 01_H regardless of pass or fail conditions. Auto-decay will happen in every heartbeat for Voltage Monitors. For the rest of the monitoring functions, it will happen for every four heartbeats. This auto-decay mechanism will not happen for the SPI Communication Monitor Counter.

If the value of the respective pass counters is equal to or above 64(40_H), the monitor function's state will be in Maintain. The status of the system can be detected by using the following SFRs:

- SystemIntegrity
- INT
- SUM0
- SUM1

For details refer to Section 2.2.5

2.2.2 System State Machine

An overview of the System State Machine is shown in **Figure 4**. The System State Machine consists of the following states:

- Reset state
- Not Ready state
- Ready state
- Active state
- Tripping states
  - Tripping state1
  - Tripping state 2
  - Tripping state 3
- Disabled state
- Reset Request state
- SPI Secure Mode state

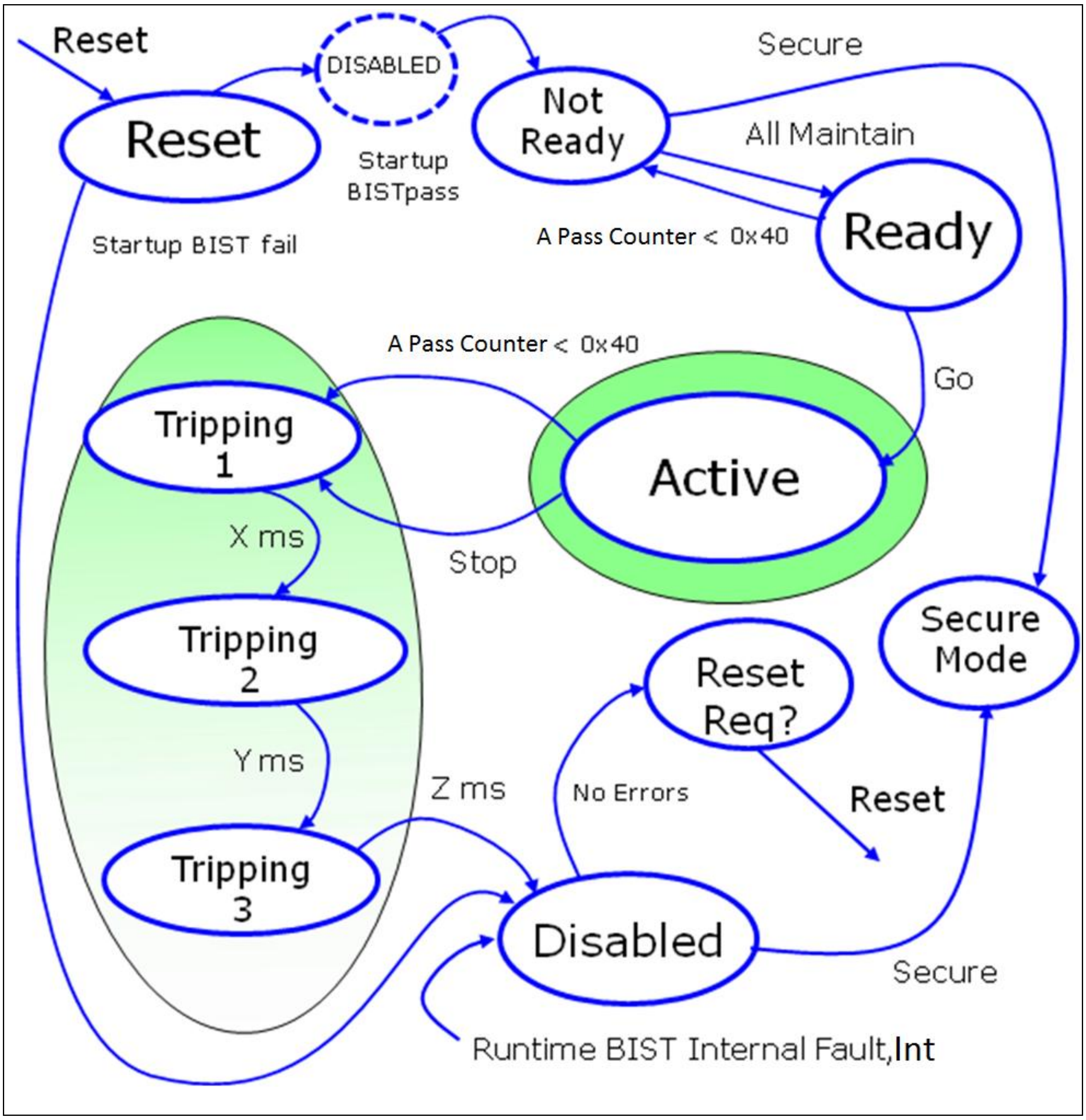


Figure 4 Integrity Monitor – System State Machine

2.2.3 State Transition

This section will describe the transition from one state to another. The transition of one state to another will mainly depend on Counter values and the Mode SFR. For more information on how these states relate to the SYSDISx safety path pins, including timings, please refer to section 2.7.2.

Note: *By writing the specific request to the MODE register, the state of the machine can be transferred to another state according to the Request written into the SFR (Refer to section 2.2.5) Mode SFR.*

2.2.3.1 RESET -> NOT READY

When the CIC61508 is powered on, the System enters the RESET state. In this state the CIC61508 will undergo Startup BIST (Built-in Self-Test). In this state, the CIC61508 does not communicate via SPI and so RESET is largely invisible. After successful completion of the BIST, the system will move to the NOT READY state. It should be noted that the SYSDISx pins will move to the DISABLED state for a short period of time before assuming the NOT READY configuration.

2.2.3.2 NOT READY -> READY

When the system is in NOT READY state, all the enabled monitor functions will be in Error state. For each test that passes, the corresponding pass counter will be incremented. Once all the pass counters of the enabled function are equal to or above 40_H, the system will move into READY state. As long as any of the pass counters are less than 40_H, the CIC61508 will remain in the NOT READY state.

2.2.3.3 NOT READY-> Secure SPI

The system in NOT READY State can move to secure SPI in two steps:

- By writing a Secure Request (94_H) to the Mode SFR
- By sending the magic numbers $AB02_H$ and $A5B6_H$ in two consecutive SPI messages. For details refer to section 2.8

2.2.3.4 READY -> NOT READY

After the system moves to the READY state, if any of the pass counters of the enabled functions fall below 40_H the system will move back to the NOT READY state.

2.2.3.5 READY -> ACTIVE

In the READY state, the Host has to send a Go Request by writing to the MODE SFR with the value $8A_H$ to trigger the state transition to the ACTIVE state.

2.2.3.6 ACTIVE -> TRIPPING 1

ACTIVE state is the working state of the CIC61508 where all the functions are in the Maintain state. To move the system into the ACTIVE state, we will provide you with a use case example in section 6.2.

The ACTIVE state can move to the TRIPPING 1 state in either of these two cases:

- The Host issues the Stop Request to make the CIC61508 move to the TRIPPING state.
- Or any one of the pass counter values falls to less than 40_H

2.2.3.7 TRIPPING 1 -> TRIPPING 2 -> TRIPPING 3 -> DISABLED

Once the TRIPPING1 state is entered, the CIC61508 waits for the defined trip time before proceeding to TRIPPING 2 and then to TRIPPING 3. The defined time for moving to the next TRIPPING state is configurable. The next state in the state machine is the DISABLED state. These three TRIPPING states provide additional states in the state machine to allow the host system to react in a timely and controlled manner.

2.2.3.8 DISABLED-> Secure SPI

The system in DISABLED state can move to secure SPI in two steps:

- By writing a Secure Request (94_H) to the MODE SFR
- And by sending the magic numbers $AB02_H$ and $A5B6_H$ in two consecutive SPI messages. For details refer to section 2.3

2.2.3.9 DISABLED-> RESET

The CIC61508 will move to the RESET state by writing to the MODE SFR with the value $C9_H$, which brings the state machine to the RESET state. It is entered if there is no error in the system. At this point, all modules should be in the Maintain state i.e. all tests are passing. This transition is also possible in response to a Wake-up Timer command.

2.2.3.10 <State Name> -> DISABLED

The Fatal error will be caused due to the:

- BIST failure (data corruption or the opcode check failure)
- System Heartbeat overrun check
- Or Task monitor/Data Comparators fatal error (over flow condition, data corruption or out of bounds access).

2.2.4 Integrity Monitor Configuration

The calibration of the Integrity Monitor requires the following four sets of user-defined parameters to be programmed into the NVM at 0xA000-0xAFFF:

- Pass counter increment/decrement value
- Monitor Function Enable
- Trip Time
- Safety Path Control

2.2.4.1 Integrity Monitor Increment and Decrement Value

The Pass Counter Increment/Decrement Value parameters determine the magnitude of the increment or decrement count value when the respective monitor function encounters a pass or fail event. The minimum count value will be 01_H and the maximum would be $3F_H$. The pass increment and fail decrement values allow the user to set the sensitivity of the CIC61508 to particular errors. For example, a very large sequence test pass increment (e.g. 0x20) and a small fail decrement (e.g. 0x02) would make the CIC61508 able to tolerate a large number of test failures before entering the DISABLED state. However it would also mean that the "failure reaction time" for this monitor would be greatly extended. If the increment and decrement values in this example were reversed, the CIC61508 would become very sensitive to test failures, requiring just two consecutive failures to cause a move to the DISABLED mode.

Table 3 Pass Counter Increment and Decrement value

Address of Main Copy	Address of Redundant Copy	Number of Bytes	Parameter
$A6C0_H$	$AEC0_H$	1	Sequencer Increment Value
$A6C1_H$	$AEC1_H$	1	Sequencer Decrement Value
$A6C2_H$	$AEC2_H$	1	Voltage Monitor A Increment Value
$A6C3_H$	$AEC3_H$	1	Voltage Monitor A Decrement Value
A6C4H	$AEC4_H$	1	Voltage Monitor B Increment Value
A6C5H	$AEC5_H$	1	Voltage Monitor B Decrement Value
$A6C6_H$	$AEC6_H$	1	Voltage Monitor C Increment Value
$A6C7_H$	$AEC7_H$	1	Voltage Monitor C Decrement Value
$A6C8_H$	$AEC8_H$	1	Voltage Monitor D Increment Value
$A6C9_H$	$AEC9_H$	1	Voltage Monitor D Decrement Value
$A6CA_H$	$AECA_H$	1	Task Monitor Increment Value
$A6CB_H$	$AECB_H$	1	Task Monitor Decrement Value

Address of Main Copy	Address of Redundant Copy	Number of Bytes	Parameter
A6CC$_H$	AECC$_H$	1	Data Comparator Increment Value
A6CD$_H$	AECD$_H$	1	Data Comparator Decrement Value

2.2.4.2 Monitor Function Enable

The Monitor Function Enable parameters control the enabling and disabling of the Voltage Monitors, Task Monitor and Data Comparator. To enable a monitor function, the corresponding parameter should have the value 00$_H$. To disable it, the value should be 40$_H$.

Table 4 Monitor Function Enable

Address of Main Copy	Address of Redundant Copy	Number of Bytes	Parameter
A6CE$_H$	AECE$_H$	1	Voltage Monitor A Enable
A6CF$_H$	AECF$_H$	1	Voltage Monitor B Enable
A6D0$_H$	AED0$_H$	1	Voltage Monitor C Enable
A6D1$_H$	AED1$_H$	1	Voltage Monitor D Enable
A6D2$_H$	AED2$_H$	1	Task Monitor Enable
A6D3$_H$	AED3$_H$	1	Data Comparator Enable

2.2.4.3 Trip Time

The Trip Time parameters define the time taken by the CIC61508 to move from the Tripping states to the Disabled state. The trip time will be the sum of time taken by the three intermediate states (Tripping states 1, 2, and 3). In the configuration, time taken for the each Tripping state in terms of the heartbeat is configured. The value of each Tripping state varies from 00$_H$ to FF$_H$ (to 153ms). The tripping states are intended to allow a sequence of SYSDISx pin states to be created that can be used to disable complex hardware in a controlled manner in 3 steps.

Table 5 Trip Time

Address of Main Copy	Address of Redundant Copy	Number of Bytes	Parameter
A6D4$_H$	AED4$_H$	1	Tripping 1 Time
A6D5$_H$	AED5$_H$	1	Tripping 2 Time
A6D6$_H$	AED6$_H$	1	Tripping 3 Time

2.2.5 Integrity Monitor Registers

The PASSCNTXX SFRs provide the current pass count value of a particular monitoring function. These SFRs will update for every heartbeat.

PASSCNTSEQ
Sequencer Pass Count Register **Reset Value: 00$_H$**
PASSCNTVA
Voltage Monitor A Pass Count Register **Reset Value: 00$_H$**
PASSCNTVB
Voltage Monitor B Count Register **Reset Value: 00$_H$**
PASSCNTVC
Voltage Monitor C Count Register **Reset Value: 00$_H$**
PASSCNTVD
Voltage Monitor D Count Register **Reset Value: 00$_H$**
PASSCNTTASK
Task Monitor Pass Count Register **Reset Value: 00$_H$**
PASSCNTCOMPARE
Data Comparator Pass Count Register **Reset Value: 00$_H$**
PASSCNTCOMM
SPI Communication Pass Count Register **Reset Value: 00$_H$**

7	6	5	4	3	2	1	0
PASS COUNT VALUE							
Rh	rh	rh	Rh	rh	rh	rh	rh

Field	Bits[1)]	Type	Description
PASS COUNT VALUE	[7:0]	rh	**These registers will give the pass counter value.**

SYSTEMINTEGRITY
System State Register **Reset Value: 69$_H$**

7	6	5	4	3	2	1	0
STATE CODE							
Rh	rh	rh	Rh	rh	rh	rh	rh

The SYSTEMINTEGRITY SFR provides the current state of the System State Machine. This register will update for every heartbeat.

Field	Bits[1)]	Type	Description
STATE CODE	[7:0]	rh	0F$_H$ Reset 1E$_H$ Active 2D$_H$ Disabled 3C$_H$ Ready 4B$_H$ Secure 69$_H$ Reset 78$_H$ Not Ready 96$_H$ Tripped1 B4$_H$ Tripped2 A5$_H$ Tripped3 **Others: Reserved**

SUM0
System State Summary 0 Register **Reset Value: 00_H**

7	6	5	4	3	2	1	0
SPICOMM	**DTACMP**	**TASKMON**	**VOLTD**	**VOLTC**	**VOLTB**	**VOLTA**	**SEQ**
Rh	rh	rh	rh	rh	rh	rh	rh

The SUM0 register will provide the state of each module. These registers will update for every heartbeat.

Field	Bits	Type	Description
SEQ	0	rh	**Sequencer** 0 Maintain state 1 Error State
VOLTA	1	rh	**Voltage A Monitor Status** 0 Maintain state 1 Error State
VOLTB	2	rh	**Voltage B Monitor Status** 0 Maintain state 1 Error State
VOLTC	3	rh	**Voltage C Monitor Status** 0 Maintain state 1 Error State
VOLTD	4	rh	**Voltage D Monitor Status** 0 Maintain state 1 Error State
TASKMON	5	rh	**Task Monitor Status** 0 Maintain state 1 Error State
DATACMP	6	rh	**Data Comparator Status** 0 Maintain state 1 Error State
SPICOMM	7	rh	**SPI Communication Status** 0 Maintain state 1 Error State

SUM1
System State Summary Register **Reset Value: 69_H**

7	6	5	4	3	2	1	0
	0		**WAKEUP**	**SPCON**	**CSFRH**	**BIST**	**ESMON**
Rh	rh	rh	rh	rh	rh	rh	rh

The SUM1 registers will provide the state of each module. These registers will update for every 600µs.

Field	Bits[1]	Type	Description
ESMON	0	rh	**Integrity Monitor Status** 0 Maintain state 1 Error State
BIST	1	rh	**Built-in Self Test Status** 0 Maintain state 1 Error State
CSFRH	2	rh	**CIC61508 SFR Handler Status** 0 Maintain state 1 Error State
SPCON	3	rh	**Safety Path Control Status** 0 Maintain state 1 Error State
WAKEUP	4	rh	**Wake-up Timer Status** 0 Maintain state 1 Error State
0	7:5	rh	**Reserved** **Return 0 if Read**

INT
System Integrity Status Register **Reset Value: 69_H**

7	6	5	4	3	2	1	0
ERROR CODE				ERROR ID			
rh	rh	rh	rh	rh	rh	rh	Rh

This register will update with the last occurrence failure condition of the CIC61508 caused by either a Fail or a Fatal response. This register will update for every heartbeat.

Field	Bits[1]	Type	Description
Error ID	[3:0]	rh	**ERROR ID** 0000 No error 0001 Sequencer Error 0010 Voltage Monitor A 0011 Voltage Monitor B 0100 Voltage Monitor C 0101 Voltage Monitor D 0110 Task Monitor 0111 Data Comparator 1000 SPI Communication 1010 Integrity Monitor 1011 Built in Self Test 1101 Safety Path Control 1110 Wake-Up Timer Others Reserved

ERROR CODE	[7:4]	rh	**ERROR CODE** 0000 No error 0001 Sequence error 0010 Time budget overrun 0011 Incorrect result 0100 Phase error 1000 Overflow condition; data corruption; out of bounds access 1001 Configuration error Others: Reserved

The list of possible INT SFR values encountered is shown in **Table 6**

Table 6 Monitor Function Enable

Int Value	**Monitor Function (ERROR ID)**	**ERROR CODE**	**Event**
00_h	-	-	Pass
21_H	Sequencer	Time budget overrun	Fail
31_H	Sequencer	Incorrect Result	Fail
32_H	Voltage Monitor A	Incorrect Result	Fail
33_H	Voltage Monitor B	Incorrect Result	Fail
34_H	Voltage Monitor C	Incorrect Result	Fail
35_H	Voltage Monitor D	Incorrect Result	Fail
16_H	Task Monitor	Sequence Error	Fail
26_H	Task Monitor	Time budget overrun	Fail
86_H	Task Monitor	Overflow condition; data corruption; out of bounds access	Fatal
17_H	Data Comparator	Sequence Error	Fail
27_H	Data Comparator	Time budget overrun	Fail
37_H	Data Comparator	Incorrect Result	Fail
87_H	Data Comparator	Overflow condition; data corruption; out of bounds access	Fatal
48_H	SPI Monitor	Phase Error	Fail
88_H	SPI Monitor	Overflow condition; data corruption; out of bounds access	Fail
$8A_H$	Integrity Monitor	Overflow condition; data corruption; out of bounds access	Fatal
$3B_H$	Built-in Self Test	Incorrect Result	Fatal
$8B_H$	Built-in Self Test	Overflow condition; data corruption; out of bounds access	Fatal
$9B_H$	Built-in Self Test	Configuration Error	Fatal
$8D_H$	Safety Path Control	Overflow condition; data corruption; out of bounds access	Fatal
$8E_H$	Wake-Up Timer	Overflow condition; data corruption; out of bounds access	Fatal
Others	Undefined		

MODE
Mode Change Request Register **Reset Value: 00_H**

7	6	5	4	3	2	1	0
MODE CR							
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

The MODE SFR can be written by a respective Request command to change the active running mode of the System State Machine. By using the Mode SFR only the following state transitions are possible:

- Active state-> Tripping 1 state
- Ready state -> Active state
- Not Ready state-> Secure SPI state
- Disabled state ->Secure SPI state
- Disabled state-> Reset state

By using the Mode SFR we can make the SPI Reset (Making SPI Pass Counter equal to 80_H).

This register will be updated with 00_H if the correct transition takes place by using the MODE SFR.

Field	Bits[1]	Type	Description
MODE CR	[7:0]	Rwh	**Mode Change Request** 85_H Stop Request (Active -> Tripping) $8A_H$ Go Request (Ready state -> Active State) 94_H Secure Request (Not Ready->Secure SPI and Disabled -> Secure SPI) $A9_H$ SPI Reset Request $C9_H$ Disabled->Reset Others Reserved

2.3 Serial Peripheral Interface

The Serial Peripheral Interface establishes a communication link between the CIC61508 and the host microcontroller. The CIC61508 is the SPI slave, whereas the host microcontroller is the master. The possible baud rates are 0.5Mbps, 1Mbps, 1.5Mbps and 2Mbps, subject to the host microcontroller being able to meet the chip select timing requirements. The MRST line must be fitted with a pull-up resistor as this is an open drain output.

By applying an active slave select signal (active low) at CS, the CIC61508 is selected by the SPI master. During the active (low) state of the select signal CS, the falling edge of the serial clock signal SCLK will be used to latch the input data at MTSR. Output data at MRST is driven with the rising edge of SCLK. LSB is always transmitted and received first.

2.3.1 SPI Communication Protocol

SPI transfers are 16-bit. The microcontroller host initiates the SPI communication to the CIC61508 by applying an active slave select signal at CS. The host then transmits the 16-bit command onto the MTSR line. Since the SPI is a full-duplex communication protocol, the CIC61508 receives the 16-bit command and at the same time returns a dummy data to the host. It will only respond with the expected 16-bit reply in the next transmission period, which is triggered by the host sending a second command or dummy data**.** If the CIC61508 receives an invalid command, it will reply with a No Acknowledge (NoACK) value of AAAA$_H$.

Note: *The first 16-bit message received from the CIC61508 (through a host-initiated SPI transfer) following a power-on reset is 5555$_H$*

Figure 5 shows the timing specification for the SPI communication at 1.5 Mbps for f_{sys} 80MHz.
After the CS signal (active low) is asserted, a minimum delay of 2µs is required before the start of the SCLK by the master. Following the 16-bit data transfer, which typically takes 10.67µs at 1.5 Mbps, a maximum hold time of 2µs, is also required before the de-assertion of the CS signal. In between consecutive transfers, a CS signal idle time of 57µs (and minimum idle time 52.7µs) is required. For every time tick of one heartbeat, the CIC61508 supports up to five 16-bit data transfers.

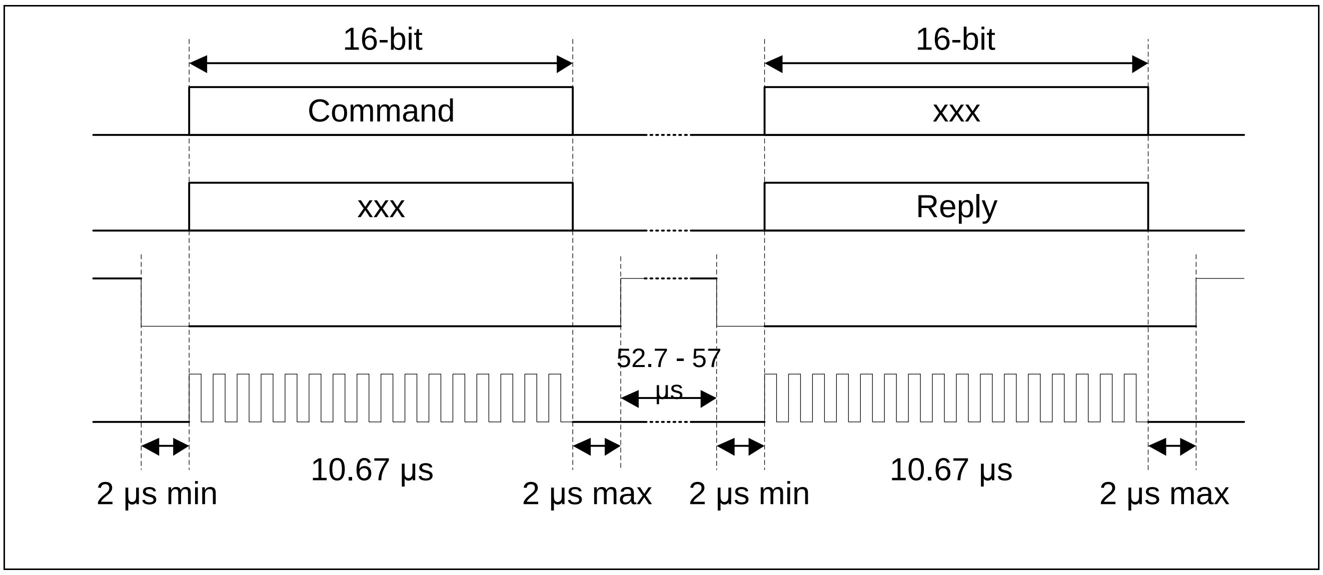


Figure 5 SPI communication Protocol

Table 7 shows the SPI timings specification for supported f_{sys}.

2.3.2 SPI Error Handling

The SPI handler is able to deal with some hardware-related errors. If the chip select trailing delay is too long, a chip select timing error is detected. In addition, if any noise occurs on the MTSR within 37.5ns before or 75ns after the falling edge of SCLK, a phase error will be detected. In both cases, the CIC61508 will return a value of 0xAAAA and the SPI pass counter will decrement by '1'.

The host microcontroller receives 0xFFFF for any SPI communication if CIC61508 is running Start-up BIST. In Start-up BIST transmit buffer of the CIC61508 had not been updated since the last transfer. To avoid slave shift out the 'old' contents of the shift register received during the last transfer which may lead to corruption of the data on the transmit/receive line, the CIC61508 transmit buffers are loaded with 'FFFFH' prior to any transfer.

Table 7 SPI Timing specification (Typical)

CIC61508	**f_{sys} 80MHz**	**f_{sys} 75MHz**
Bit Rate	1.5 mbps	1 mbps
SCLK period	0.67 µs	1.00 µs
Leading Delay	2 µs min	3 µs min
Leading Delay(Worst case)	1.98 µs	2.112 µs
Data Transfer	10.67 µs	16 µs
Trailing Delay	2 µs max	2 µs max
Trailing Delay(Worst case)	2.801 µs	2.988 µs
CS Signal Idle Time	52.7µs min - 57µs max	
Tolerance	Tolerance +5%, -6%	

Note: *The MRST pin goes low after the Chip Select (CS) goes low; this is caused by the CIC61508 re-enabling the SSC after the CS falling edge. The MRST goes '0' when SSC is re-enabled and this is about 1.2us after the CS falling edge .After this, the next byte to be transmitted is loaded into the SSC transmit buffer. However, nothing happens until the Host starts the SCLK at CS low + 2us, i.e. nothing happens before the first leading edge of the SCLK when the first bit of the new message is placed on the MRST pin. As SCLK does not start until 2us after CS goes low, this has no effect on the Host.*

2.3.3 SPI Command Format

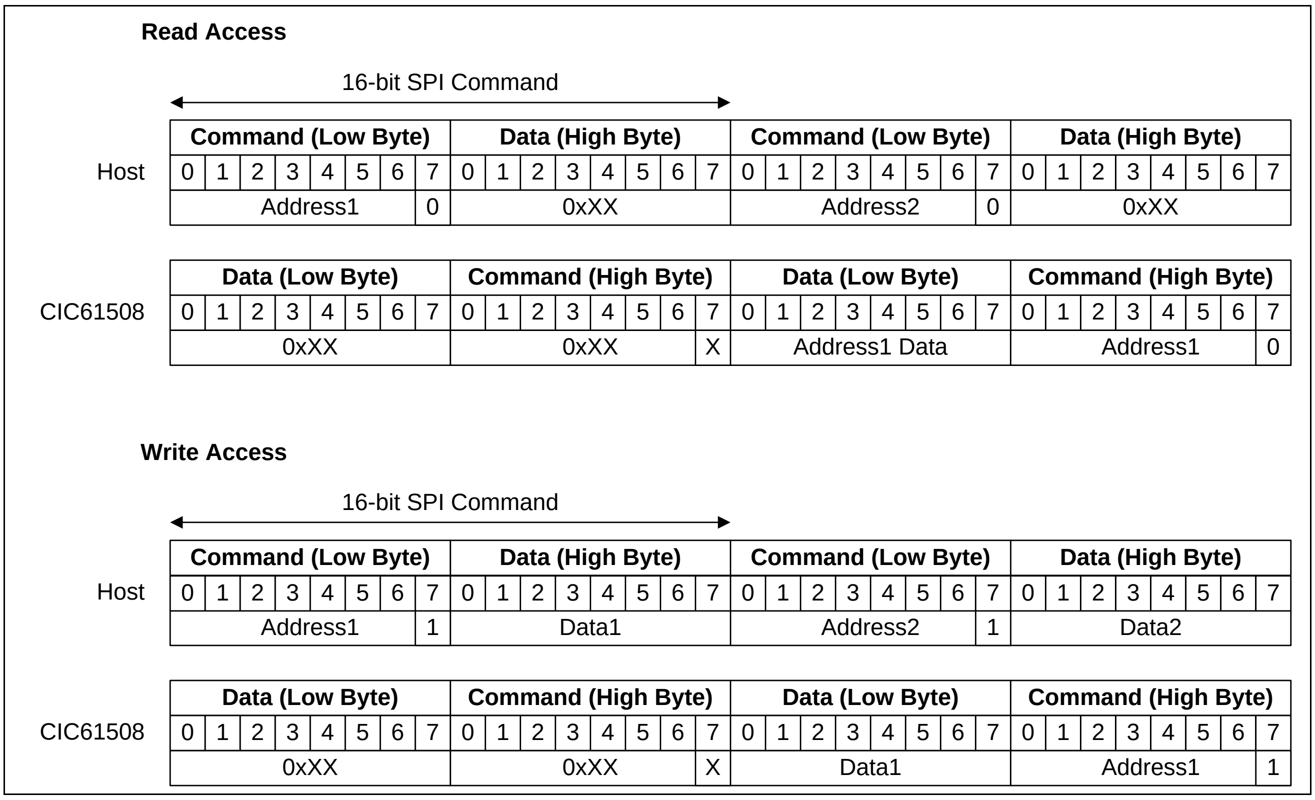
All communications between the host microcontroller and the CIC61508 are carried out by SFR accesses through the SPI. For both Read and Write access, the 16-bit SPI command consists of a command byte and a data byte. The command byte will be either Read command or Write command to the SFRs.

When receiving the 16-bit command, the CIC61508 gets the command byte first, followed by the data byte. When transmitting, it is the opposite; the CIC61508 transmits the data byte first, followed by the command byte.

Read and Write accesses on the SFRs are shown in **Figure 6**.

A Read command to the SFR will read the content in that particular SFR Read and output will be in the next CIC61508 SPI reply.

A Write command to the SFR, on the other hand, is buffered and the actual write to the SFR will take place only at the start of the next heartbeat. Therefore, if a Read on the same SFR is requested within the same heartbeat, the SFR Read data will contain the old value.



Read Access

	Command (Low Byte)								Data (High Byte)								Command (Low Byte)								Data (High Byte)							
Host	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
	Address1							0	0xXX								Address2							0	0xXX							

	Data (Low Byte)								Command (High Byte)								Data (Low Byte)								Command (High Byte)							
CIC61508	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
	0xXX								0xXX							X	Address1 Data								Address1							0

Write Access

	Command (Low Byte)								Data (High Byte)								Command (Low Byte)								Data (High Byte)							
Host	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
	Address1							1	Data1								Address2							1	Data2							

	Data (Low Byte)								Command (High Byte)								Data (Low Byte)								Command (High Byte)							
CIC61508	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
	0xXX								0xXX							X	Data1								Address1							1

Figure 6 SFR Read and Write access.

2.4 Sequencer

The Sequencer will test the series of answers generated by the Host controller at regular intervals of time. The Sequencer will update the request number (question) and will expect the Host to send the answer corresponding to the question. The result must be received at a specific time within the Window Watchdog. The result from the host is then compared with the expected result that is stored in the CIC61508 NVM. Depending on the result, the pass counter will be incremented or decremented.

Features

- It supports up to 64 test sequences (answers) of 4 bytes each.
- Configurable Window Watchdog time (Min and Max).

2.4.1 Sequencer Operation

The Sequencer has a SEQ SFR which defines the current request number (question). Upon a successful comparison of the current answer, the SEQ SFR is updated with the next request number. The request number and the corresponding 32-bit answer are configured in the NVM. The Sequencer will be provided with the two parameters Minimum Window Period and Maximum Window Period. The Maximum Window Period is the Window Watchdog time period, which is divided into the Open Window Period and the Closed Window Period. Minimum Window Period is the Closed Window Period. These two parameters are configurable in terms of heartbeats.

According to the Request number in the SEQ SFR, the CIC61508 will expect the 32-bit answer from the host controller. The answer is written through four separate SFRs (OTRHH, OTRHL, OTRLH, and OTRLL) by the host controller. Writing to OTRHL, OTRLH, and OTRLL can be in any order, but the final write to the OTRHH must happen in the Open Window Period which is defined by the equation (Maximum Window - Minimum Window). If the write to SFR OTRHH is done outside of the open window, the Sequencer pass counter will be decremented and a time budget overrun status will be flagged in INT SFR. Writing to SFR OTRHH resynchronizes the Window Watchdog to the next heartbeat and starts the Window Watchdog close window, which is defined by WinMin*heartbeat.

This 32-bit answer, which is received by writing to the OTRXX SFRs, is compared with the corresponding answer for the Request number in SEQ SFR. Depending on the result, the pass counter will be incremented if the answer is the same and the SEQ SFR is updated with the next Request number. The pass counter is decremented if the answer is not the same and the incorrect result is flagged in INT SFR. The SEQ SFR is not updated with the next Request number and it remains the same. After the comparison of the last answer, the SEQ SFR will be updated with the first Request number and the test will be carried out continuously. The minimum number of question-answer challenges to be carried out should be 8_H.

Figure 7 shows the sequence test carried out by the CIC61508.

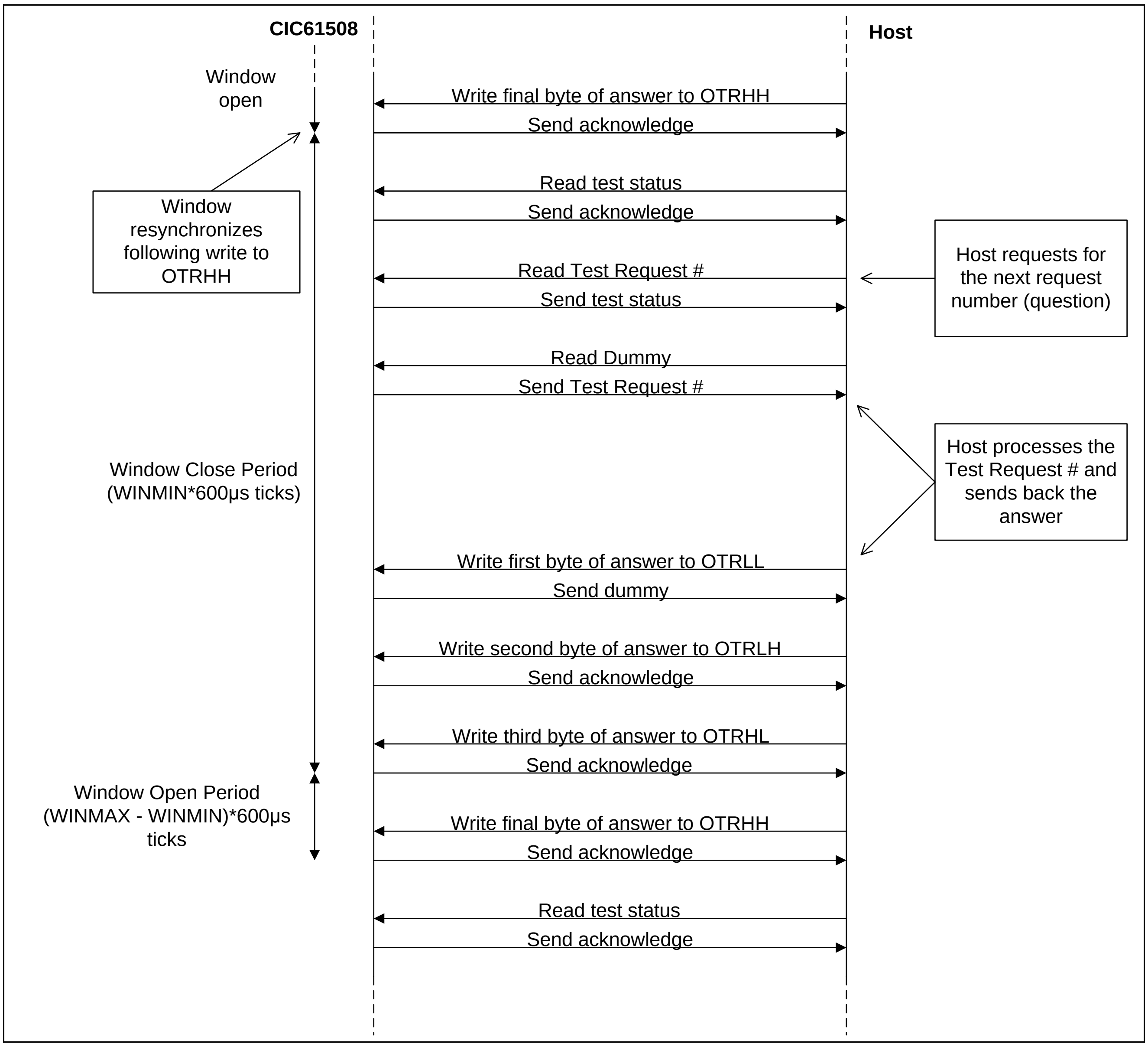

CIC61508
Host
Window open
Window resynchronizes following write to OTRHH
Write final byte of answer to OTRHH
Send acknowledge
Read test status
Send acknowledge
Read Test Request #
Send test status
Host requests for the next request number (question)
Read Dummy
Send Test Request #
Host processes the Test Request # and sends back the answer
Window Close Period (WINMIN*600µs ticks)
Write first byte of answer to OTRLL
Send dummy
Write second byte of answer to OTRLH
Send acknowledge
Write third byte of answer to OTRHL
Send acknowledge
Window Open Period (WINMAX - WINMIN)*600µs ticks
Write final byte of answer to OTRHH
Send acknowledge
Read test status
Send acknowledge

Figure 7 Sequencer's Operational Sequence

2.4.2 Sequencer Configuration

The Sequencer Configuration is defined by the following:

- Request number
- Answer for the Request number
- Minimum window parameter
- Maximum window parameter
- Table length parameter

The Request number is the 8-bit number. For each Request number it has the corresponding 32-bit answer which is stored in the four 8-bit NVM address locations.

The maximum window parameter defines the total Window Watchdog period where the test related to the request needs to be completed in terms of the number of heartbeats, ranging from one heartbeat (01_H) to 100 heartbeats (64_H). The minimum window parameter defines the window close period of the watchdog in terms of the number of heartbeats, ranging from 0_H to 63_H heartbeats. For example, a maximum window parameter value of 50 (32_H) equates to a total Window Watchdog period of 30 ms (50*600µs).

The table length parameter defines the length of the test sequence from 8 (08_H) to 64 (40_H). The sequence of tests will always start again from the beginning (sequence #0) after the last test of the sequence has passed.

All the parameters are configured in NVM through the Secure SPI or by using the TARDISS tool (Refer to **Section 3**).

Table 8 Sequencer Parameter Addresses

Address of Main Copy	**Address of Redundant Copy**	**Number of Bytes**	**Parameter**
$A000_H$	$A800_H$	1	Test Request #1
$A001_H$	$A801_H$	1	Answer to test Request #1 (High-High byte)
$A002_H$	$A802_H$	1	Answer to test Request #1 (High-Low byte)
$A003_H$	$A803_H$	1	Answer to test Request #1 (Low-High byte)
$A004_H$	$A804_H$	1	Answer to test Request #1 (Low-Low byte)
$A005_H$	$A805_H$	1	Test Request #2
$A006_H$	$A806_H$	1	Answer to test Request #2 (High-High byte)
$A007_H$	$A807_H$	1	Answer to test Request #2 (High-Low byte)
$A008_H$	$A808_H$	1	Answer to test Request #2 (Low-High byte)
$A009_H$	$A809_H$	1	Answer to test Request #2 (Low-Low byte)
$A00A_H$	$A80A_H$	1	Test Request #3
$A00B_H$	$A80B_H$	1	Answer to test Request #3 (High-High byte)
$A00C_H$	$A80C_H$	1	Answer to test Request #3 (High-Low byte)
$A00D_H$	$A80D_H$	1	Answer to test Request #3 (Low-High byte)
$A00E_H$	$A80E_H$	1	Answer to test Request #3 (Low-Low byte)
------	------	-----	------------
$A136_H$	$A936_H$	1	Test Request #63
$A137_H$	$A937_H$	1	Answer to test Request #63 (High-High byte)
$A138_H$	$A938_H$	1	Answer to test Request #63 (High-Low byte)
$A139_H$	$A939_H$	1	Answer to test Request #63 (Low-High byte)
$A13A_H$	$A93A_H$	1	Answer to test Request #63 (Low-Low byte)
$A13B_H$	$A93B_H$	1	Test Request #64
$A13C_H$	$A93C_H$	1	Answer to test Request #64 (High-High byte)
$A13D_H$	$A93D_H$	1	Answer to test Request #64 (High-Low byte)
$A13E_H$	$A93E_H$	1	Answer to test Request #64 (Low-High byte)
$A13F_H$	$A93F_H$	1	Answer to test Request #64 (Low-Low byte)
$A140_H$	$A940_H$	1	Minimum Window (00_H - 63_H)
$A141_H$	$A941_H$	1	Maximum Window (01_H - 64_H)
$A142_H$	$A942_H$	1	Table Length (08_H - 40_H)

2.4.3 Sequencer Registers

OTRLL
Opcode Test Result Register LOW-LOW Byte **Reset Value: 00_H**
OTRLH
Opcode Test Result Register LOW- HIGH Byte **Reset Value: 00_H**
OTRHL
Opcode Test Result Register HIGH-LOW Byte **Reset Value: 00_H**
OTRHH
Opcode Test Result Register HIGH- HIGH Byte **Reset Value: 00_H**

7	6	5	4	3	2	1	0
DATA							
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

The Result SFRs OTRLL, OTRLH and OTRHL can be written in any order. However, the final write to SFR OTRHH must be completed within the open watchdog window.

Field	Bits[1]	Type	Description
DATA	[7:0]	rwh	Test DATA (Answer)

WINMAX
Window Watchdog Maximum Value Register **Reset Value:10h**

7	6	5	4	3	2	1	0
WINDOWMAX							
rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits[1]	Type	Description
WINDOWMAX	[7:0]	rh	Defines the total watchdog period where the requested test needs to be completed in number of heartbeats.

WINMIN
Window Watchdog Minimum Value Register **Reset Value:05h**

7	6	5	4	3	2	1	0
WINDOWMIN							
rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits[1]	Type	Description
WINDOWMIN	[7:0]	rh	Defines the window close period of the watchdog after a refresh in number of heartbeats.

The values of the WinMax and WINMIN SFRs always take the programmed value of the maximum and minimum window parameters in the NVM.

SEQ
Test Sequence Register **Reset Value: First Request Number**

7	6	5	4	3	2	1	0
SEQ							
rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits[1)]	Type	Description
SEQ	[7:0]	rh	Defines the current request number for test sequence n.

2.5 Supply Voltage Monitor

The CIC61508 can monitor up to four voltages, sampled at every heartbeat. These voltages would typically be the power supplies to the Host CPU or other safety-critical hardware in the system. The user can program the range for each voltage via the NVM. The sampling of voltage will be initiated on reset of the CIC61508. The sampled voltages will be updated in the respective SFRs and the Host can read these voltages by using the Coherent Read mechanism.

The sampling of the voltage can be suspended for one heartbeat tick by invoking the Voltage injection feature (Refer to Section 2.5.3). The voltage count value has to be provided instead by a software write to the voltage monitor registers for that channel. The voltage threshold test will be carried out as before, but based on this software written value. This can be used to deliberately inject incorrect voltage readings to demonstrate that the pass counter system is correctly detecting voltage errors.

In all cases, the pass counter of the voltage monitor will be incremented if the result is valid (i.e. voltage in the range), or decremented if the result is invalid (voltage outside the range).

Features

- Monitors up to four Supply Voltages
- Programmable boundary limits for the voltage to be valid held in NVM.
- Allows software to provide the voltage count value for the threshold through voltage injection feature.
- Supports external precision reference for greater accuracy.
- The sampling of voltage will be carried out at 10-bit resolution.

2.5.1 Supply Voltage Monitored Operation

The CIC61508 can monitor up to four voltages (A, B, C and D). Each monitor voltage will be associated with the two SFRs namely VOLTMONXL and VOLTMONXH (X=A, B, C and D). Each of the monitored voltages is sampled every heartbeat and updated in the respective SFRs. These values in the SFRs are compared with minimum and maximum count values which are configured in the respective NVM. If the sampled voltage falls between the threshold voltages, the voltage is valid and will increment the Voltage Monitor Pass Counter for that particular channel.

If the sampled voltage falls outside the threshold voltage, the voltage is invalid and the respective Voltage Monitor Pass Counter will be decremented. An incorrect result status will also be flagged in INT SFR. Once in the Active state, if any of the channels' pass counters falls below the value 40_H, the Integrity Monitor will go to the Tripping states and subsequently bring the CIC61508 to the Disabled state. All these things will happen for every heartbeat. Thus the Voltage Monitor Pass Counter will be either decremented or incremented for every heartbeat.

2.5.2 Coherent Read

Since the monitored voltage will be sampled and the VOLTMONXX updated on every heartbeat, the values in the SFRs are not consistent over a period of time. To make the values in the SFRs consistent over a time period, the CIC61508 offers a mechanism called Coherent Read.

With this mechanism, the voltage monitor will sample the voltage but it will not update the particular VOLTMONXX SFRs over the next two heartbeats. To facilitate a Coherent Read, a Write targeting the VOLTMONXL SFR is required before the consecutive Reads to VOLTMONXH and VOLTMONXL must be carried out. The resolution of the sampled voltage is the 10 bits [9:0]; the upper 8 bits [9:2] can be read from the VOLMONXH [7:0] and the lower two bits [1:0] read from the VOLTMONXL [7:6].

2.5.3 Voltage Injection

Voltage Injection is a mechanism whereby the Host can inject a voltage value instead of the sampled voltage for a particular channel. By using this mechanism the sampling of the voltage will be suspended over the next heartbeat and it will use the injected voltage count value to compare against the threshold voltages. If the voltage is valid it will increment the voltage monitor pass counter, else it will decrement the pass counter for that particular channel. The normal Voltage sampling will resume in the next heartbeat.

The voltage injection is requested by writing the injected count value (upper 8 bits) to VoltMonXH SFR (where X represents the channel being sampled). The VoltMonXL SFR (containing the lower 2 bits of the voltage count value) has no relevance in voltage injection as this will be written with 00_H. Please refer to Section 2.5.5 for the calculation of injected voltage count value.

2.5.4 Supply Voltage Monitor Registers

VOLTMONXH (X=A,B,C,D)
Voltage Monitor X High Byte **Reset Value: Sampling Voltage High Byte Value**

7	6	5	4	3	2	1	0
VOLTX[9:2]							
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

This register will be updated with the higher bits of the sampled value. While reading using Coherent Read, this register will contain the higher bits.

Field	Bits[1]	Type	Description
VOLTX	[7:0]	rwh	During a Coherent Read these bits will contain the higher bits of the Sampled Voltage Value. For the injection method, the Host needs to write the higher bits of the injected value.

VOLTMONXL (X=A,B,C,D)
Voltage Monitor X Low Byte **Reset Value: Sampling Voltage Low Bits Value**

7	6	5	4	3	2	1	0
VOLTX[1:0]		**Reserved**					
rwh	rwh	rw	rw	rw	rw	rw	rw

This register will be updated with the lower bits of the sampled value. This register will be updated every 600µs.

Field	Bits[1]	Type	Description
VOLTX[1:0]	[7:6]	rwh	While in Coherent Read these bits will contain the lower bits of the Sampled Voltage Value. While at injection method the Host will need to write the lower bits of the injected value.
Reserved	[5:0]	rw	Writing into these bits has no effect on the monitor system. While reading we will always read 0

2.5.5 Supply Voltage Monitor Configuration

Each of the four voltage monitors is defined by a minimum and a maximum 10-bit count value, which determines the validity of the monitored voltage. The count value can be calculated using the following formula, where the monitored voltage must always be smaller or equal to the reference voltage:

$$\text{Count value} = \frac{\text{Monitored voltage}}{\text{Reference voltage}} \times 1024$$

Table 9 Voltage Monitor Configuration

Address of Main Copy	Address of Redundant Copy	Number of Bytes	Parameter
$A6A0_H$	$AEA0_H$	1	Voltage Monitor A Minimum Count (High Byte)
$A6A1_H$	$AEA1_H$	1	Voltage Monitor A Minimum Count (Low Byte)
$A6A2_H$	$AEA2_H$	1	Voltage Monitor A Maximum Count (High Byte)
$A6A3_H$	$AEA3_H$	1	Voltage Monitor A Maximum Count (Low Byte)
$A6A4_H$	$AEA4_H$	1	Voltage Monitor B Minimum Count (High Byte)
$A6A5_H$	$AEA5_H$	1	Voltage Monitor B Minimum Count (Low Byte)
$A6A6_H$	$AEA6_H$	1	Voltage Monitor B Maximum Count (High Byte)
$A6A7_H$	$AEA7_H$	1	Voltage Monitor B Maximum Count (Low Byte)
$A6A8_H$	$AEA8_H$	1	Voltage Monitor C Minimum Count (High Byte)
$A6A9_H$	$AEA9_H$	1	Voltage Monitor C Minimum Count (Low Byte)
$A6AA_H$	$AEAA_H$	1	Voltage Monitor C Maximum Count (High Byte)
$A6AB_H$	$AEAB_H$	1	Voltage Monitor C Maximum Count (Low Byte)
$A6AC_H$	$AEAC_H$	1	Voltage Monitor D Minimum Count (High Byte)
$A6AD_H$	$AEAD_H$	1	Voltage Monitor D Minimum Count (High Byte)
$A6AE_H$	$AEA0_H$	1	Voltage Monitor D Minimum Count (High Byte)
$A6AF_H$	$AEA1_H$	1	Voltage Monitor D Minimum Count (High Byte)

All the parameters are configured in NVM through the Secure SPI or by using the TARDISS tool (Refer to **Section 3**).

2.6 Wake-Up Timer

The Wake-up Timer performs the task of waking up the host system at pre-defined intervals, to enable the low quiescent current through a low-to-high transition on the SPI chip select pin. This enables the host to go into a Sleep state or a Low Power state, and can wake-up by monitoring the transition of the SPI chip select pin.

All CIC61508 functions will be stopped once the Wake-up Timer functionality is invoked by the host. The CIC61508 will also be put into a low current mode to enable a low quiescent current for the system. The Wake-up Timer waits for the pre-defined wake-up time before triggering a reset on the CIC61508 that generates the low-to-high transition on the chip select pin.

An additional function of the Wake-up Timer is to immediately reset the CIC61508.

Features

- Configurable wake-up time.
- Operate the CIC61508 in low current mode
- Can immediately reset the CIC61508.

2.6.1 Wake-up Timer Operation

The Wake-up function should be initialized in two steps:

1) First, WAKEPRESCALAR SFR must be written, else the default value will be taken.
2) Then the Wake-up Timer function is enabled by a SFR write command to the WAKERELOAD SFR.

If the SFR of the WAKEPRESCALAR is set to 80_H (CIC61508 Reset bit is set), then the Wake-up Timer will cause an immediate reset of the CIC61508.

The Wake-up Time, t_{WUT}, is determined by the SFRs WAKERELOAD and WAKEPRESCALAR using the following formulae:

$$t_{WUT} = \frac{(2^{WakeUpPrescalar} * (128(65536-256(WakeupReload))))}{F_{VCO}}$$

And

$$F_{VCO} = \frac{Fsys_{FreeRunning}}{3}$$

In the above formulae, F_{VCO} is the frequency value between 1.67 MHz and 13.3 MHz.

When the Wake-up Timer function is enabled, the SPI chip select pin will be driven low and all other CIC61508 functions will be stopped. The CIC61508 will also be put into a low current mode. The Wake-up Timer then waits for the Wake-up Time to elapse before triggering a reset on the CIC61508 to generate the low-to-high transition on the chip select pin. This low-to-high transition on the chip select pin can Wake-up the host controller if it is in a Sleep state.

2.6.2 CIC61508 Reset Operation

The CIC61508 will transition to a RESET state immediately by using a special Wake-up Timer mode. By setting a WAKEPRESCALAR SFR value of $8X_H$ and writing any value to WAKERELOAD SFR, the CIC61508 will reset immediately. The chip select pin is not actively driven though in this mode.

2.6.3 Wake-up Timer calibration

The frequency of the Wake-up Timer, f_{WUT}, is a value between 1.67 MHz and 13.3 MHz (maximum deviation of 10 %). Therefore, the host microcontroller is required to perform a calibration sequence to obtain the reload value corresponding to the targeted Wake-up Time interval.

The calibration sequence consists of the following steps:

- Select a suitable WAKEPRESCALAR based on the targeted Wake-up Time.
- Enable the Wake-up Timer by writing WAKERELOAD with 255.
- Measure the time between the high-to-low and low-to-high transitions on the CS pin.
- Derive the actual WAKERELOAD to be used for the targeted Wake-up time by using the formula below.

Note: *The host system is not put into any power-saving mode during the calibration sequence.*

After the time between the high-to-low and low-to-high transitions on the CS pin is measured, the actual WAKERELOAD value can be derived from the following formulae:

$$Actual\ WakeRelaod = 255 - \frac{Targeted\ Wakeup\ Time}{Measured\ Wakeup\ Time\ based\ on\ WakeRelaod\ of\ 255}$$

After calibrating the actual Wake-Up Reload value, the host can initiate the Wake-Up Timer by issuing the calibrated values.

Table 10 shows the Wake-up time interval range supported by each WAKEPRESCALAR for all values of f_{WUT}. As a general rule of thumb, the lower the WAKEPRESCALAR used, the higher the Wake-up time accuracy and current consumption, while the higher the WAKEPRESCALAR used, the lower the Wake-up time accuracy and current consumption.

Table 10 Wake-Up Time Interval per WAKEPRESCALAR value

Wake-up Prescalar		**Wake-Up Time t_{WUT} (Sec)**			
PRESCALAR	**2 ^ PRESCALAR**	**Reload=255 F_{VCO}=1.67 MHz**	**Reload=0 F_{VCO}=1.67 MHz**	**Reload=255 F_{VCO}=13.3 MHz**	**Reload=0 F_{VCO}=13.3 MHz**
1	1	0.0221	5.0231	0.0024	0.6307
2	2	0.0442	10.0462	0.0048	1.2614
3	4	0.0784	20.0924	0.0096	2.5228
4	8	0.1568	40.1848	0.0192	5.0456
5	16	0.3136	80.3696	0.0384	10.0912
6	32	0.6272	160.7392	0.0768	20.1824
7	64	1.2544	321.47844	0.1536	40.3648
8	128	2.5088	642.9768	0.3072	80.7296
9	256	5.0176	1285.9136	0.6144	161.4592
10	512	10.0352	2571.8272	1.2288	322.9184
11	1024	20.0704	5143.2544	2.4576	645.8368
12	2048	40.1408	10287.3088	4.9152	1291.6736

2.6.4 Wake-Up Timer Registers

WAKERELOAD
Wake-Up Timer Reload register **Reset Value:00h**

7	6	5	4	3	2	1	0
RELOAD							
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits[1)]	Type	Description
RELOAD	[7:0]	rwh	Wake-Up Timer Reload value

WAKEPRESCALAR
Wake-Up Timer Prescalar Register **Reset Value: 00_H**

7	6	5	4	3	2	1	0
CIC61508 RESET	**Reserved**			**PRESCALAR**			
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits[1]	Type	Description
PRESCALAR	[3:0]	rwh	**Wake-Up Timer Prescalar** 0000 1 0001 2 0010 4 0011 8 0100 16 0101 32 0110 64 0111 128 1000 256 1001 512 1010 1024 1011 2048 Others: Reserved
Reserved	[6:4]	rwh	**Reserved** Return 0 if read, should be written with 0
CIC61508 RESET	7	rwh	0: Wakeup according to WAKE_PRE settings 1: Triggers immediate Reset

Note: *Writing the Prescalar value with anything other than the above mentioned value will generate a Fatal error and flag an out of bounds access in INT.*

2.7 Safety Path Control

Instead of reading the status registers of the CIC61508, there is another mechanism to get the status of the CIC61508 through the **Safety Path Control (SPC)**. SPC has three pins named SYSDISA, SYSDISB, and SYSDISC.

2.7.1 Safety Path Control Configuration

The Safety Path Control parameters define the level (High: 1, Low: 0) of the SYSDISA, SYSDISB and SYSDISC pins for each individual state in the System State Machine. The level of each pin can be configured for every state. The configuration of SYSDISC will be done in separate NVM addresses while SYSDISA and SYSDISB will use the same set of NVM addresses for both.

Depending on the level of the pin required for the respective states in the System State Machine, the following values are to be written to the respective NVM location:

- For SYSDISC parameters:
  - 00_H to make the output 0
  - 04_H to make the output 1
- For SYSDISA, SYSDISB parameters
  - 00_H to make the output 0 on Both pins
  - 01_H to make the output 1 on SYSDISB and 0 on SYSDISA
  - 02_H to make the output 0 on SYSDISB and 1 on SYSDISA
  - 03_H to make the output 1 on Both pins

For example, if it is necessary to output 101_B on the three pins SYSDIS[C:A] in the event that the Tripping 2 state is entered, the SYSDISC parameter at address $A6DB_H$ has to be written with 04_H while the SYSDIS[B:A] parameter at address $A6EB_H$ has to be written with 01_H.

Table 11 Safety Path Control Configuration for SYSDISC

Address of Main Copy	Address of Redundant Copy	Number of Bytes	Parameter
$A6DB_H$	$AEDB_H$	1	Tripping 2 State
$A6DC_H$	$AEDC_H$	1	Tripping 3 State
$A6DD_H$	$AEDD_H$	1	Tripping 1 State
$A6DF_H$	$AEDF_H$	1	Not Ready State
$A6E0_H$	$AEE0_H$	1	Reset State
$A6E2_H$	$AEE2_H$	1	SPI Secure Mode State
$A6E3_H$	$AEE3_H$	1	Ready State
$A6E4_H$	$AEE4_H$	1	Disabled State
$A6E5_H$	$AEE5_H$	1	Active State
$A6E6_H$	$AEE6_H$	1	Reset Request State

Table 12 Safety Path Control Configuration for SYSDISA and SYSDISB

Address of Main Copy	Address of Redundant Copy	Number of Bytes	Parameter
A6EB$_H$	AEEB$_H$	1	Tripping 2 State
A6EC$_H$	AEEC$_H$	1	Tripping 3 State
A6ED$_H$	AEED$_H$	1	Tripping 1 State
A6EF$_H$	AEEF$_H$	1	Not Ready State
A6F0$_H$	AEF0$_H$	1	Reset State
A6F2$_H$	AEF2$_H$	1	SPI Secure Mode State
A6F3$_H$	AEF3$_H$	1	Ready State
A6F4$_H$	AEF4$_H$	1	Disabled State
A6F5$_H$	AEF5$_H$	1	Active State
A6F6$_H$	AEF6$_H$	1	Reset Request State

All the parameters are configured in NVM through the Secure SPI or by using the TARDISS tool (Refer to **Section 3**).

2.7.2 Real Time SYSDISx Pin Behaviour

The SYSDISx pins change directly in response to the internal state changes inside the CIC61508. However during the startup phase, the timings of the SYSDISx pin state changes are not directly linked to the SYSTEMINTEGRITY SFR. It should be noted that until the CIC61508 has fully initialized, the SYSDISx pins are floating and undriven. The pins then assume the configuration associated with the DISABLED state, before assuming the values for the NOTREADY state, around 600us later. Thus it is important to make sure that these pins are externally pulled-up to avoid undefined behaviour immediately after RESET. It is also recommended (but not mandatory) to make the SYSDISx pin states for the DISABLED mode programmed in the NVM equal to ‘1’, i.e. the floating state arising immediately after a CIC61508 power-up. At the very least, during system design the initial states of these pins and the devices they are connected to should be considered.

The timings of the possible SYSDIS pin states is set out the table below. These timings give an indication only and definitive figures can be found in the CIC61508 datasheet.

Table 13 Typical Safety Path Pin State Sequence (with timings)

SYSINT State	SYSINT Value	State Duration	Comment	SYSDISx Pin State	Notes
Reset	0x69	Zero	CIC61508 RESET pin goes high. SYSTEMINTEGRITY = 0x69 but this is not visible externally as the SPI is not initialised yet.	SYSDISx floats	CIC61508 RESET pin goes high.
Reset	0x69	196µs	Internal self-test (BIST) begins	SYSDISx floats	-
Reset	0x69	52ms	BIST ends	SYSDISx floats	-
Disabled	0x2D	600µs (max)	SYSTEMINTEGRITY = 0x2D. This is not visible externally as the SPI is not initialised yet.	SYSDISx driven to DISABLED pattern	If BIST fails, Disabled state is permanent. SPI interface now initialised.
Not Ready	0x78	Application-dependent	SYSTEMINTEGRITY = 0x78. This is visible by SPI.	SYSDISx driven to NOTREADY pattern	At least one pass counter < 0x40
Ready	0x3C	Application-dependent	SYSTEMINTEGRITY = 0x3C. This is visible by SPI.	SYSDISx driven to READY pattern.	All pass counters > 0x40
Active	0x1E	Application-dependent	SYSTEMINTEGRITY = 0x1E. This is visible by SPI.	SYSDISx driven to ACTIVE pattern.	All pass counters > 0x40 and GO written to MODE
Tripping 1	0x96	Tripping 1 timeout in NVM	SYSTEMINTEGRITY = 0x96. This is visible by SPI but may not be detected externally due to short duration.	SYSDISx driven to Tripping1 pattern.	At least one pass counter < 0x40 or STOP written to MODE. Max 153ms duration
Tripping 2	0xB4	Tripping 2 timeout in NVM	SYSTEMINTEGRITY = 0xB4. This is visible by SPI but may not be detected externally due to short duration.	SYSDISx driven to Tripping2 pattern.	Max 153ms duration
Tripping 3	0xA5	Tripping 3 timeout in NVM	SYSTEMINTEGRITY = 0xA5. This is visible by SPI but may not be detected externally due to short duration.	SYSDISx driven to Tripping3 pattern.	Max 153ms duration
Disabled	0x2D	Forever	SYSTEMINTEGRITY = 0x2D. This is visible by SPI.	SYSDISx driven to DISABLED pattern	This state can only be left via reset or Wake-Up command.
Reset Request	0x0F	600µs (max)	SYSTEMINTEGRITY = 0x0F. This is visible by SPI but may not be detected externally due to short duration. Device resets within 600µs.	SYSDISx driven to Reset Request pattern	Write 0xC9 to MODE SFR but SUM0 and SUM1 must equal 0x00.
Secure SPI Mode	0x4B	Application-dependent	SYSTEMINTEGRITY = 0x0F. This is visible via secure SPI by reading address I:0x07. However it is meaningless in secure SPI mode.	SYSDISx driven to Secure SPI Mode pattern	SYSTEMINTEGRITY must equal 0x78 (NOTREADY) or 0x2D (DSIABLED) to enter mode - see section "Secure SPI Mode" for detailed entry criteria.

2.8 Secure SPI Mode

The Secure SPI mode is provided to allow users to program/erase the DFLASH contents and to provide advanced diagnostics. The advanced diagnostics could be reading/writing to specific IRAM/XRAM memory locations, executing code from a specific memory address and causing a CIC61508 Reset. In addition, CIC61508 “applets” can be loaded into the XRAM and then executed to perform user-specific actions.

The Secure SPI mode can be entered from the NOT READY state or from the DISABLED state. Once the secure mode is entered, all normal SPI commands will no longer be recognized and all interrupts are disabled. Secure SPI mode can only be exited through a power-on reset (PORST) or by issuing a CIC61508 Reset command.

A set of predefined C functions for Infineon microcontrollers is available to allow the Secure SPI mode features to be accessed easily from user applications such as end-of-line test programs or diagnostic tools.

2.8.1 Secure Mode Entry

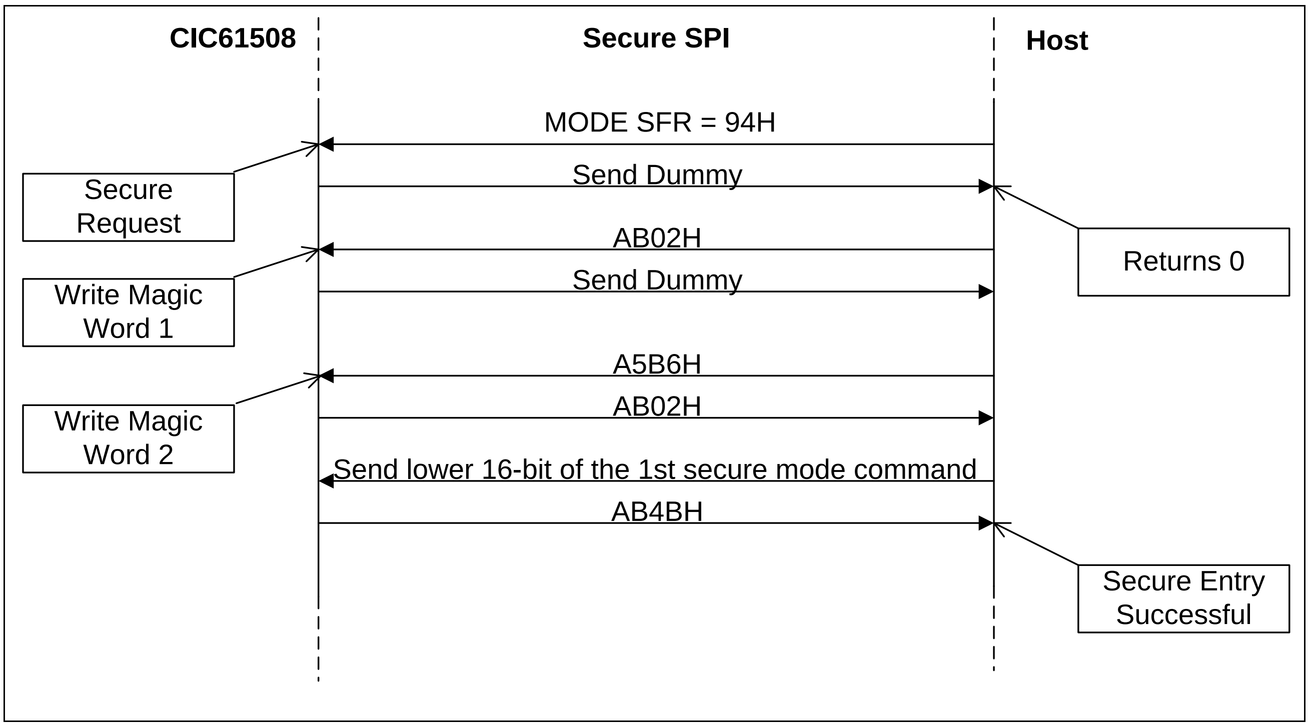


Figure 8 Entry to Secure SPI Operation

Step 1: To gain entry to Secure SPI Mode from Not Ready or Disabled state, 94_H should be set to MODE SFR.

Step 2: Access will be granted in Secure SPI Mode only if Magic Words $AB02_H$ & $A5B6_H$ are received through two consecutive 16-bit SPI transfers. Otherwise, an output of 1234_H, 5678_H is sent.

Step 3: Once Secure SPI Mode is entered with correct Magic Word, an output of $AB4B_H$ is sent.

2.8.2 Secure SPI Mode Operation

The Secure SPI mode uses a 32-bit command format as shown in Table 14. Bytes 0 and 1 contain the targeted NVM address, while Byte 2 defines the Read or Write operation. Byte 3 contains the data for a Write operation and for a Read operation, it can take any value.

The 32-bit command must be sent through two consecutive 16-bit SPI transfers. Therefore, the timing requirements described in Section 2.3.1 are also applicable for Secure SPI mode. Shift on Rising edge, Latch on Falling edge, LSB is sent first and the maximum speed is 2Mbps.

Table 14 Secure SPI mode Commands and operation spaces

Command	Byte 0	Byte 1	Byte 2	Byte 3
Secure SPI Read	Address Low	Address High	$7F_H$ & MEM	Don't Care
Secure SPI Write	Address Low	Address High	80_H | MEM	Data
Secure SPI Functions	Address Low	Address High	80_H | FUNC	Don't Care

MEM	Block	Range
2	IFX_IDATA	0000_H – $00FF_H$
4	IFX_XDATA	$F000_H$ – $F1FF_H$
8	IFX_CODE	0000_H – $2FFF_H$

FUNC	Function
3	Erase DFLASH
6	Jump to Address
7	Cause CIC61508 Reset

Operation	Value
Access IFX_CODE space	08_H
Access IFX_XDATA space	04_H
Access IFX_IDATA space	02_H
Write IFX_CODE space	88_H
Write IFX_XDATA space	84_H
Write IFX_IDATA space	82_H
Erase Complete DFLASH	83_H
Jump to an Absolute Address	86_H
Cause CIC61508 to Reset	87_H

Example: Reading IFX_CODE space content at $2900_H = 43_H$

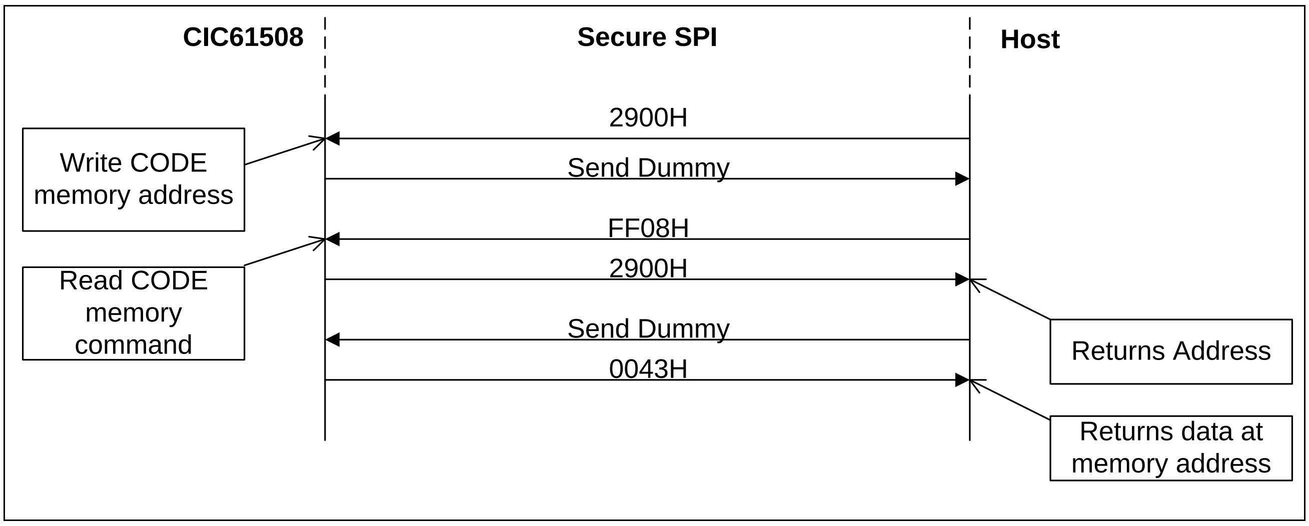


Figure 9 Secure SPI Read operation

Example: Writing of IFX_IDATA space contents at location $0080_H = AA_H$

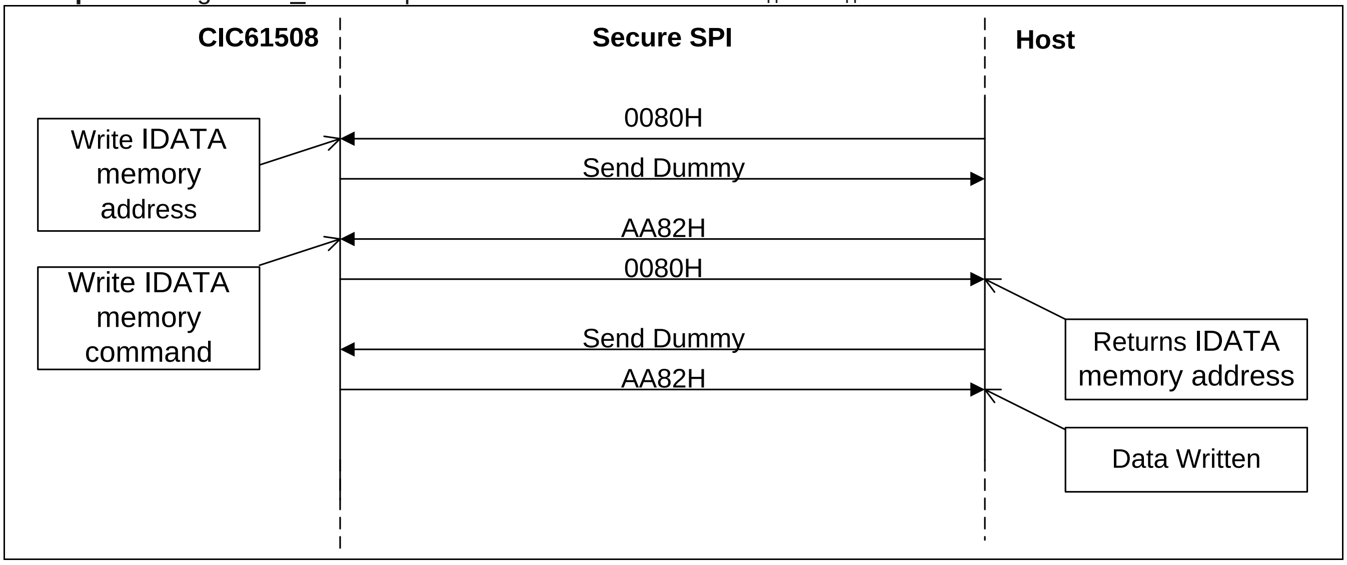

CIC61508
Secure SPI
Host
Write IDATA memory address
Write IDATA memory command
0080H
Send Dummy
AA82H
0080H
Send Dummy
AA82H
Returns IDATA memory address
Data Written

Figure 10 Secure SPI Write operation

2.8.3 Secure SPI Mode Error Handling

Secure SPI mode generally does not have advanced error handling, but the DFLASH NVM functions and READ/WRITE commands will return simple error codes in the event of a failure. These are set out below.

Table 15 Secure SPI mode error codes

Error Code	Meaning
0x0000	No Error Occurred.
0x0200	NVM FLASH did not erase properly.
0x0300	The base address supplied for erasing the DFLASH was incorrect.
0x0400	The base address supplied for programming the DFLASH was below 0xA000.
0x0400	The base address supplied for programming the DFLASH was above 0xAFFF.
0x0800	The DFLASH failed to program properly.
0xAAAA	Unknown command or action.

2.8.4 Secure SPI Mode Synchronization To Host

The secure SPI mode expects all message transactions to be sent by the Host CPU in pairs. If, due to noise or other factors, the CIC61508 misses one message, it become out of synchronization with the Host. This can be detected by the Host as the CIC61508 will not reply with the expected data. If this happens, the Host should send one dummy message and then send a message sequence with a predictable result i.e. READ CODE address 0x0000 and check that the value returned by the CIC61508 is ‘0x02’.

2.8.5 Secure SPI Mode Exit

Secure SPI mode can Exit by Power-on Reset or by issuing a CIC_RESET command.

2.9 Task Monitor

The Task Monitor monitors the flow of any sequential set of tasks, for example operating system (OS, Application) tasks, for the correct sequence and completion within an allocated time budget. The task monitor has 8 individual task timeout counters to allow up to 8 levels of task nesting.
A correct sequence and the task completion within the time budget will increment the Task Monitor Counter value. An incorrect sequence or task execution timeout will decrement the pass counter.

Features

- Task sequence monitoring
- Task execution time monitoring
- 8 individual task timeout counters to allow up to 8 levels of task nesting
- Up to 255 monitored tasks can be defined in the CIC61508.
- Configurable time budget ranging from 2 heartbeats to FE_H heartbeats.

2.9.1 Task Monitor Operation

The Task Monitors will monitor the tasks running in the host system. For each task to be monitored in the host system, they are assigned specific Task IDs and corresponding time budgets. These are configured in the respective addresses in the NVM in the sequence in which they are executed. The CIC61508 can monitor up to 255 tasks.

The CIC61508 provides two SFRs, TASKSTART and TASKEND, to execute the functions of the Task Monitor. The task monitoring is started by writing the Task ID of the first monitored task (Task #1) to the TASKSTART SFR. The Task ID is checked for the correct sequence and the corresponding time budget value is loaded into the next available internal CIC61508 timer, plus the Task Monitor pass counter increments. Eight timers are provided to support up to eight levels of task nesting. The timer is started to monitor the time budget for the corresponding task. When the monitored task completes execution, the TASKEND SFR must be written with the same Task ID to stop the timer. If the TASKEND SFR is written before the timer expires, the Task Monitor pass counter will be incremented, else the pass counter will be decremented and a time budget overrun status will be flagged in INT SFR.

Since only a linear flow of monitored tasks is allowed, the TASKSTART SFR has to be written in the correct sequence. A wrong sequence will also decrement the pass counter and flag a sequence error in INT SFR. The TASKEND SFR, on the other hand, can be written in any order.

Figure 11 shows an example of a task sequence. In the example, note that the monitoring of Task#3 is started before Task #2 is completed, resulting in two levels of task nesting.

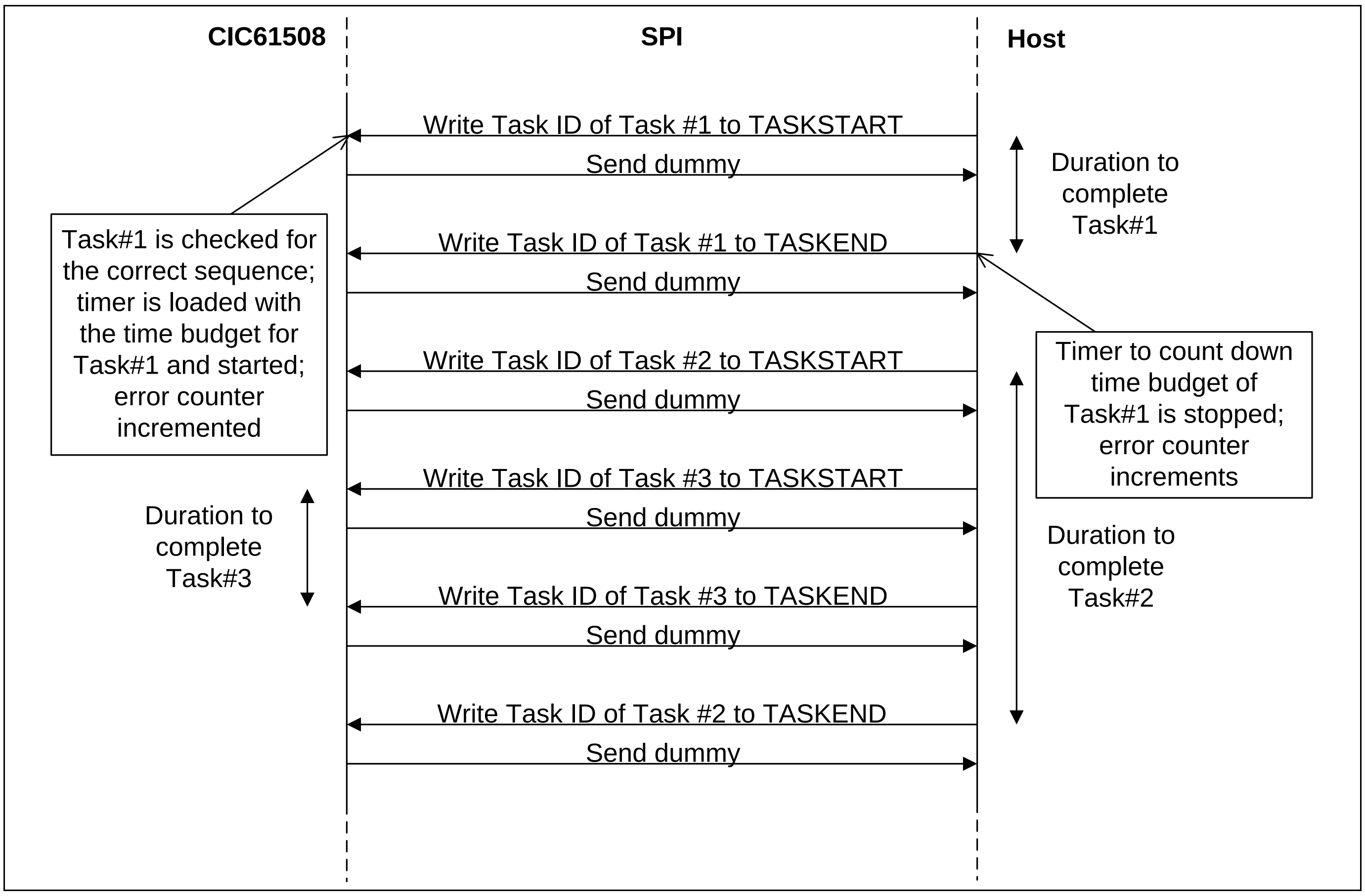

CIC61508
SPI
Host
Write Task ID of Task #1 to TASKSTART
Send dummy
Write Task ID of Task #1 to TASKEND
Send dummy
Write Task ID of Task #2 to TASKSTART
Send dummy
Write Task ID of Task #3 to TASKSTART
Send dummy
Write Task ID of Task #3 to TASKEND
Send dummy
Write Task ID of Task #2 to TASKEND
Send dummy
Task#1 is checked for the correct sequence; timer is loaded with the time budget for Task#1 and started; error counter incremented
Duration to complete Task#3
Duration to complete Task#1
Timer to count down time budget of Task#1 is stopped; error counter increments
Duration to complete Task#2

Figure 11 Example of a Task Sequence

2.9.2 Task Monitor Configuration

The Task Monitor is defined by the following:

- Time budget table
- Table length parameter

The time budget table defines the Task ID and its corresponding time budget for each task. The tasks are to be entered in running order sequence. It is possible to have more than one instance of the same Task ID in the task sequence provided they meet the sanity criteria (they are mutually exclusive).

The time budget can be configured to range from 2 heartbeats (02_H = 1200µs) to 254 heartbeats (152.4ms). The table length parameter defines the number of tasks that is to be monitored. A maximum of 255 (FF_H) tasks can be defined.

All the parameters are configured in NVM through the Secure SPI or by using the TARDISS tool (Refer to **Section 3**).

Table 16 shows an example of a time budget table for a task sequence consisting of eight tasks, four of which require a time budget of 1.2 ms, two require 1.8 ms and the another two require 3.6ms.

Table 16 Example of a Time Budget Table

Task No	Task ID	Time Budget
1	02_H	02_H (600µs*2 = 1.2 ms)
2	01_H	03_H (600µs*3 = 1.8 ms)
3	04_H	06_H (600µs*6 = 3.6 ms)
4	01_H	03_H (600µs*3 = 1.8 ms)

Task No	Task ID	Time Budget
5	05_H	02 (600µs*1 = 1.2 ms)
6	02_H	02_H (600µs*2 = 1.2 ms)
7	04_H	06_H (600µs*6 = 3.6 ms)
8	05_H	02_H (600µs*1 = 1.2 ms)

After the last task in the task sequence defined in the time budget table has been executed, the Task Monitor always expects the next task to start from task number 1 again.

Table 17 Task Monitor Parameter Addresses

Address of Main Copy	Address of Redundant Copy	Number of Bytes	Parameter
$A480_H$	$AC80_H$	1	Task #1 ID
$A481_H$	$AC81_H$	1	Time Budget for the Task #1
$A482_H$	$AC82_H$	1	Task #2 ID
$A483_H$	$AC83_H$	1	Time Budget for the Task #2
$A484_H$	$AC84_H$	1	Task #3 ID
$A485_H$	$AC85_H$	1	Time Budget for the Task #3
$A486_H$	$AC86_H$	1	Task #4 ID
$A487_H$	$AC87_H$	1	Time Budget for the Task #4
--------	--------	---	-----------
$A67A_H$	$AE7A_H$	1	Task #254 ID
$A67B_H$	$AE7B_H$	1	Time Budget for the Task #254
$A67C_H$	$AE7C_H$	1	Task #255 ID
$A67D_H$	$AE7D_H$	1	Time Budget for the Task #255
$A67E_H$	$AE7E_H$	1	Table Length

2.9.3 Task Monitor Registers

TASKSTART
Task Start Register **Reset Value:00h**

7	6	5	4	3	2	1	0
TASK ID							
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Writing the Task ID into the register, any one of the 8 available timers will start.

Field	Bits[1]	Type	Description
TASK ID	[7:0]	rwh	Writing the Task ID into the register will start the timer.

TASKEND
Task End Register **Reset Value:00h**

7	6	5	4	3	2	1	0
TASK ID							
Rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Writing the Task ID into this register will stop the timer which is triggered when the same ID is written to the TASKSTART. Writing the Task ID into this register before writing into the TASKSTART will generate the sequence error.

Field	Bits[1]	Type	Description
TASK ID	[7:0]	rwh	Writing the Task ID into the register will stop the timer.

2.10 Data Comparator

The Data Comparator allows two application threads to send algorithm results for comparison against a static pass or fail criterion. The Data Comparator has an 8 x 32-bit buffer to allow up to 8 comparisons to be made in parallel. All comparisons are allocated the same pre-defined time budget.

An incorrect comparison result, time budget or buffer overrun will cause the pass counter to be decremented.

Features

- 8 x 32-bit buffer to allow up to 8 comparisons to be made in parallel.
- Supports 8-/16-/32-bit signed/unsigned integers and 32-bit single precision float data types.
- Supports ‘greater than’, ‘less than’, and ‘equal to’ comparison criteria.
- Up to 128 comparison tasks could be defined.
- Configurable time budget ranging from 1_H to 80_H heartbeats (incremental time steps of 600µs).

2.10.1 Data Comparator Operation

A data comparison operation is started by writing the first set of data to the DATAAXX SFRs, followed by writing the Compare ID to the COMPA SFR. Here the Compare ID is the index to the compare buffer. Writing the index number to COMPA SFR selects the comparison criteria, data type and mask value for the data comparison. It also sets up the next available timer to start the timeout of the user-defined time budget.

The second set of data, to which the first set of data is compared, must be written to the DATABXX SFRs. The timer is stopped only when the same Compare ID is written to the COMPB SFR. The data comparison is always done with respect to DATAAXX, i.e. DATAAXX is greater than/less than/equal to DATABXX.

If comparison between the values in DATAAXX and DATABXX is in accordance with the Compare ID (ie true), the pass counter will decrement and an incorrect result status will be flagged in the INT SFR.

Both the writes to DATABXX and COMPB SFRs have to be completed before the time budget expires, else a time budget overrun status will be flagged in INT SFR. If the Compare ID that is written to COMPB SFR has not been previously written to COMPA SFR, i.e. is not a recognized comparison, a sequence error will be flagged in INT SFR. In both cases, the pass counter is also decremented.

If more than the eight comparisons happen simultaneously, the CIC61508 will generate the fatal error and the overflow condition is flagged in the INT SFR.

Figure 12 shows an example of a Data Comparator sequence. In this example, two data comparisons (of data1 and data2) are executed in parallel.

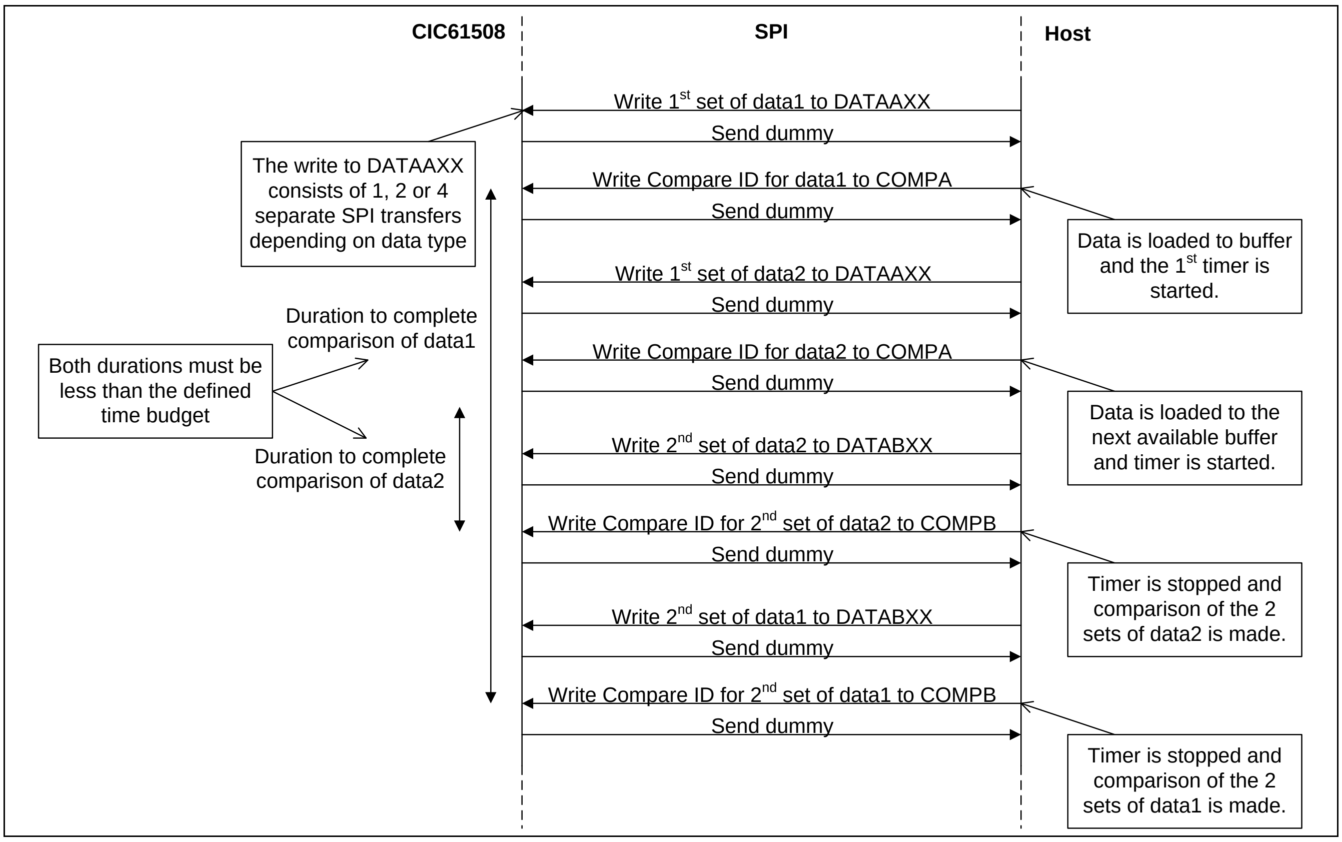

CIC61508
SPI
Host
Write 1st set of data1 to DATAAXX
Send dummy
Write Compare ID for data1 to COMPA
Send dummy
Write 1st set of data2 to DATAAXX
Send dummy
Write Compare ID for data2 to COMPA
Send dummy
Write 2nd set of data2 to DATABXX
Send dummy
Write Compare ID for 2nd set of data2 to COMPB
Send dummy
Write 2nd set of data1 to DATABXX
Send dummy
Write Compare ID for 2nd set of data1 to COMPB
Send dummy
The write to DATAAXX consists of 1, 2 or 4 separate SPI transfers depending on data type
Duration to complete comparison of data1
Both durations must be less than the defined time budget
Duration to complete comparison of data2
Data is loaded to buffer and the 1st timer is started.
Data is loaded to the next available buffer and timer is started.
Timer is stopped and comparison of the 2 sets of data2 is made.
Timer is stopped and comparison of the 2 sets of data1 is made.

Figure 12 Examples of Two Data Comparisons

2.10.2 Data Comparator Configuration

The Data Comparator is defined by the following:

- Comparison criteria
- Data Type
- 32-bit mask value
- Time budget parameter
- Table length parameter

The comparison criteria define the types of comparison to be carried out between the two buffers. The Data Comparator will support 'greater than', 'less than' and 'equal to', while for data type, 8-/16-/32-bit signed/unsigned integers and 32-bit single precision float data types are supported. A 32-bit mask value can be defined to adjust the precision of the comparison. The definition of the comparison criteria and data type is shown in **Table 18**.

Table 18 Comparison Criteria and Data Type Definition

Parameter	Definition
Comparison Criteria	00_H :> 01_H := 02_H: <
Data Type	00_H :8-bit signed integer 01_H :16-bit signed integer 02_H :32-bit signed integer 03_H :8-bit unsigned integer 04_H :16-bit unsigned integer 05_H :32-bit unsigned integer 06_H: 32-bit floating point.

The time budget parameter defines a single time budget value to be used for all data comparisons, ranging from 600µs (01_H) to 152.4ms (FE_H) in incremental steps of 600µs. The table length parameter defines the number of available Compare IDs and hence, the length of the comparison type table. The Data Comparator supports up to 128 (80_H) Compare IDs.

All the parameters are configured in NVM through the Secure SPI, or by using the TARDISS tool (Refer to **Section 3**).

Table 19 Data Comparator Parameter Addresses

Address of Main Copy	Address of Redundant Copy	Number of Bytes	Parameter
$A160_H$	$A800_H$	1	Data type for Compare ID 0
$A161_H$	$A801_H$	1	Compare Type for Compare ID 0
$A162_H$	$A802_H$	1	Mask For Compare ID0 (High-High byte)
$A163_H$	$A803_H$	1	Mask For Compare ID0 (High-Low byte)
$A164_H$	$A804_H$	1	Mask For Compare ID0 (Low-High byte)
$A165_H$	$A805_H$	1	Mask For Compare ID0 (Low-Low byte)
$A166_H$	$A806_H$	1	Data type for Compare ID 1
$A167_H$	$A807_H$	1	Compare Type for Compare ID 1
$A168_H$	$A808_H$	1	Mask For Compare ID1 (High-High byte)
$A169_H$	$A809_H$	1	Mask For Compare ID1 (High-Low byte)
$A16A_H$	$A80A_H$	1	Mask For Compare ID1 (Low-High byte)
$A16B_H$	$A80B_H$	1	Mask For Compare ID1 (Low-Low byte)
$A16C_H$	$A80C_H$	1	Data type for Compare ID 2
$A16D_H$	$A80D_H$	1	Compare Type for Compare ID 2
$A16E_H$	$A80E_H$	1	Mask For Compare ID2 (High-High byte)
$A16F_H$	------	1	Mask For Compare ID2 (High-Low byte)
$A170_H$	$A936_H$	1	Mask For Compare ID2 (Low-High byte)
$A171_H$	$A937_H$	1	Mask For Compare ID2 (Low-Low byte)
-----------	-----------	--	-----------
-------	---------	--	-----------
$A45A_H$	$A93A_H$	1	Data type for Compare ID 127
$A45B_H$	$A93B_H$	1	Compare Type for Compare ID 127
$A45C_H$	$A93C_H$	1	Mask For Compare ID 127 (High-High byte)

Address of Main Copy	Address of Redundant Copy	Number of Bytes	Parameter
A45D$_H$	A93D$_H$	1	Mask For Compare ID 127 (High-Low byte)
A45E$_H$	A93E$_H$	1	Mask For Compare ID 127 (Low-High byte)
A45F$_H$	A93F$_H$	1	Mask For Compare ID 127 (Low-Low byte)
A460$_H$	A940$_H$	1	Time Budget (01H - FEH)
A461$_H$	A941$_H$	1	Table length (00H - 80H)

2.10.3 Data Comparator Registers

The Data Registers allow two sets of data (Data A and Data B) to be written for comparison.

For 8-bit data type comparisons, only the Low-Low byte Data Registers (DATAALL and DATABLL) are used, while for 16-bit data type comparisons, both Low-High byte and Low-Low byte Data Registers (DATAALH, DATAALL, DATABLH and DATABLL) are used.

DATAALL
Data A Register LOW-LOW Byte **Reset Value: 00$_H$**
DATAALH
Data A Register LOW- HIGH Byte **Reset Value: 00$_H$**
DATAAHL
Data A Register HIGH-LOW Byte **Reset Value: 00$_H$**
DATAAHH
Data A Register HIGH- HIGH Byte **Reset Value: 00$_H$**

7	6	5	4	3	2	1	0
DATA A							
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits[1]	Type	Description
DATA A	[7:0]	rwh	DataA For Comparison

DATABLL
Data B Register LOW-LOW Byte **Reset Value: 00$_H$**
DATABLH
Data B Register LOW- HIGH Byte **Reset Value: 00$_H$**
DATABHL
Data B Register HIGH-LOW Byte **Reset Value: 00$_H$**
DATABHH
Data B Register HIGH- HIGH Byte **Reset Value: 00$_H$**

7	6	5	4	3	2	1	0
DATA B							
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits[1]	Type	Description
DATA B	[7:0]	rwh	DATAB For Comparison

COMPA
Compare Index A Register **Reset Value:00h**

7	6	5	4	3	2	1	0
COMPARE ID A							
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

When the SFR CompA is written, the timeout is started.

Field	Bits[1]	Type	Description
COMPARE ID A	[7:0]	rwh	**COMPARE ID A** Written with the Compare ID to select the width of the expected data vector, timeout timer and comparison criteria to be used.

COMPB
Compare Index A Register **Reset Value:00h**

7	6	5	4	3	2	1	0
COMPARE ID B							
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

When the SFR CompB is written, the timeout is stopped and the comparison is evaluated

Field	Bits[1]	Type	Description
COMPARE ID B	[7:0]	rwh	**COMPARE ID B** Written with the Compare ID to select the width of the expected data vector, timeout timer and comparison criteria to be used.

2.11 Scheduling Task Start Events

The Data Compare and Task Monitor systems have to be planned very carefully when both are being used. The Data Compare requires 5 SPI messages to start a compare and another 5 to stop a compare. The maximum number of SPI messages per 600us period is 8. If, for example, a TskM_ActivateTask(1) occurs in the same 600us period as a Data Compare start (and the Sequencer test trigger sequence is automatically scheduled by TriCore), the exact timing of the TskM_ActivateTask(1) may slip by one 600us period. Thus the resolution of any task event is 1200us. Therefore the task monitor is not really intended for monitoring tasks of less than 5ms duration or tasks that restart within this time.

The Task Monitor is best used for higher-level tasks that run every 5ms to 100ms and which have durations of 5ms to around 100ms. Tasks running every 2ms cannot realistically be monitored. (These figures are only a guide and every system will be different.)

It is necessary to establish at the system design stage the exact order in which monitored tasks will start under every operating condition. It is very easy to occasionally get a task running in an unexpected sequence in a real time system. Therefore it is recommended that you restrict monitored tasks to just a few critical, large tasks.

At all times it must be remembered that although task sequences can be up to 255 events long, no more than 8 can be actively monitored at any one time.

3 Tuning the DFLASH NVM Configuration

The CIC61508 firmware can be tuned according to specific requirements by updating the DFLASH configuration.

Users can use the following tools to undertake this tuning:

1) Infineon CIC61508 Test and Rapid Development for the Infineon Safety System[1] (TARDISS) – both ROM and FLASH based.

   Newer versions of (TARDISS) tool are released as “PRO-SIL SafeTkit Test Bench”
2) Keil uVision workspace tuned to generate binary code which will program the DFLASH area of the CIC61508 – For FLASH based only.
3) Infineon FLOAD tool to download the generated binary code and program the DFLASH memory – For FLASH based only.

3.1 TARDISS Installation[2]

Please refer to Section 4 of [TARDISS], for TARDISS software installation and configuring the supported microcontroller.

3.2 TARDISS Configuration (with microcontroller support)

The TARDISS tool provides the means to perform:

i. Live Monitoring of SFRs and update also.
ii. Reading of current DFLASH parameters into a local edit buffer.
iii. Programming of DFLASH.

3.2.1 Connection to CIC61508

Please refer to Section 5 of [TARDISS].

3.2.2 Edit and Program the DFLASH Configuration

Please refer to Section 6 of [TARDISS].

Relevant sections are

- Section 6.1 for Reading the current DFLASH content from CIC61508 into the Editor
- Section 6.3 for Updating the Editor with customized DFLASH settings
- Section 6.4 for Programming back into the DFLASH

The above mentioned functionality can be achieved only if TARDISS has support for the relevant microcontroller.

[1] TARDISS can also be used to program the DFLASH, but DFLASH programming requires TARDISS to connect to the respective TCXXX SafeTkit board. Currently, TARDISS supports only TC1782, TC1387 and TC1767 SafeTkit boards.

[2] Please note that this installation procedure is correct for version 2.8, but may be subject to change for future releases of TARDISS.

3.3 TARDISS Configuration (without microcontroller support)

This applies only to *FLASH based CIC61508 devices*. Irrespective of the microcontroller, the TARDISS tool also provides the means to:

i. Import the DFLASH configuration parameters from an Excel spreadsheet to the Editor.
ii. Update the DFLASH configuration parameters in a user-friendly manner.
iii. Export the Excel spreadsheet to a compliable C const array.
iv. Generate the binary code and program the DFLASH through JTAG.

3.3.1.1 Import DFLASH Contents from a Spreadsheet

An existing DFLASH calibration can be imported from the CIC61508 reference spreadsheet (CIC61508_BuildSheet_STC-I.xls). This reference DFLASH calibration data is tuned with respect to the SafeTcore-I production release. NVM Data Tables will be updated according to the imported spreadsheet.

Please refer to Section 6.2 and 6.3 of [TARDISS].

3.3.1.2 Export DFLASH Data to a C File

Please refer to Section 6.5 of [TARDISS].

3.4 TARDISS Troubleshooting

Table 20 TARDISS - Troubleshooting

Symptoms	*Cause/Workaround*
Please select a processor configuration file from the “Configuration and Live SFRs” tab before using this function!	Please follow the procedure mentioned in Section 5.1 of [TARDISS].

3.5 DFLASH Binary Generation (FLASH based CIC61508)

The DFLASH_Tune folder contains the following files:

a) *cic61508_tune.uv2* – This is a Keil uVision workspace which is responsible for generating a binary “cic61508_tune.hex” which will program only the DFLASH memory of CIC61508.
b) *CIC_DFLASH.c* – C source file exported by the TARDISS tool
c) *CIC_DFLASH.h* – Header file required by CIC_DFLASH.c
d) *cic61508_tune.lin* – Linker file which defines the DFLASH memory layout

Replace the CIC_DFLASH.c with the respective DFLASH configuration C file by following the procedure mentioned in Section 3.3.1.2.

Then do a “Re-Build All” from the workspace and the desired binary file will be created in the same folder as “cic61508_tune.hex”.

3.6 Programming DFLASH

Once the tuned DFLASH binary HEX has been generated, please follow the procedure mentioned in Section 4 – Flashing Procedure.

4 Flashing Procedure

4.1 FLOAD Tool

The FLOAD tool provides a means to download and FLASH the binary HEX code into Infineon XC800 microcontrollers with programmable non-volatile on-chip memory (PFLASH/DFLASH) or volatile memory (XRAM).

4.1.1 Installation

The FLOAD tool installation can be found in the FLOAD_Setup.zip file, which contains the following files:

Table 21 FLOAD Installation Files

File Name	Comment
Setup.exe	FLOAD Installer
Das_edition_v292.zip	Standalone installer for Device Access Server (DAS) version 2.92
Memtool.zip	Memtool Installer version 4.2 zip file (Contains DAS installer also)

The FLOAD tool can be installed on computers using Windows 2K, XP, Vista (32-bit) and Windows 7 (32-bit). There are no strict CPU or memory requirements.

The FLOAD Tool requires DAS 2.9.2 or later to support the JTAG/SPD protocol. To install DAS, please install either the standalone installer (Das_edition_v292.zip) or the Memtool installer, which installs DAS by default.

The following functions are available:

a) Open a binary file.
b) Connect to the CIC61508 (XC866-4F) microcontroller through a USB.
c) Download the binary FLASH, program and verify the FLASH contents.

4.1.2 Hardware connection between PC Host and Target

The hardware connection between the PC Host and the target device would be a USB mini-Wiggler cable. One end of the USB mini-Wiggler would be connected to a USB port on the PC Host and the other end would be connected to the CIC61508 JTAG connector.

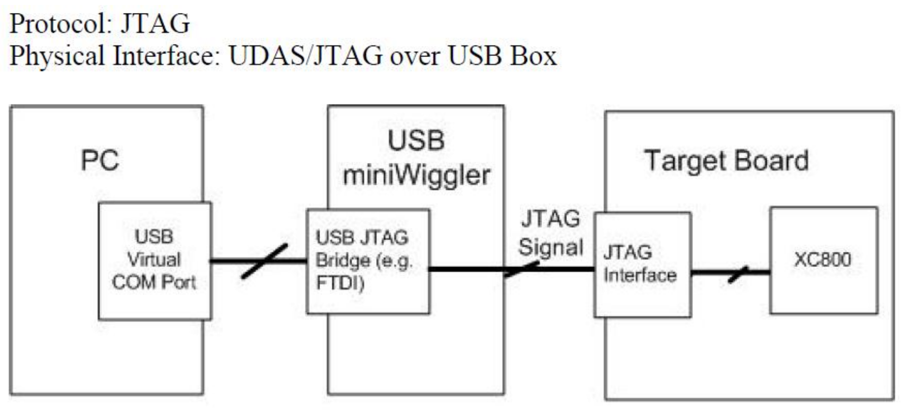


Figure 13 FLOAD – Hardware Connection between PC and Target

4.1.3 FLASH Settings and Commands

Please find the GUI interface of the FLOAD tool.

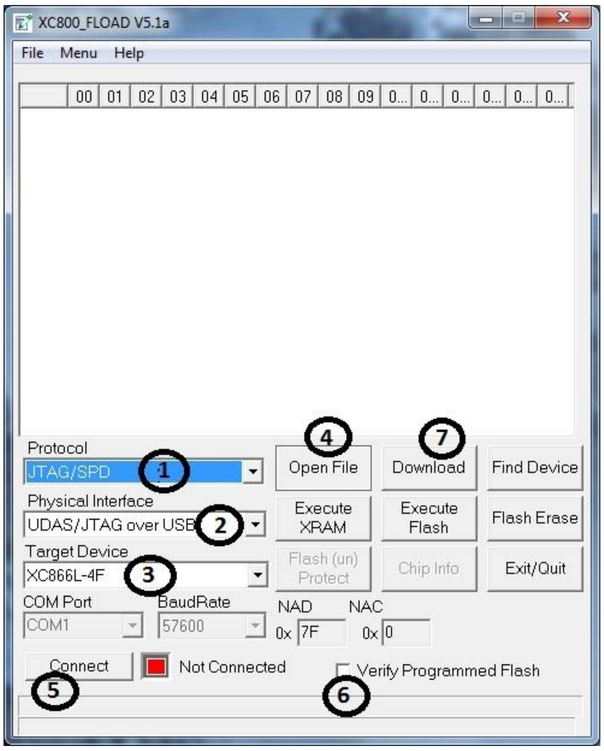


Figure 14 FLOAD – GUI Interface

Follow the commands/inputs given below to FLASH the desired binary HEX into the CIC61508 target. Refer to Figure 25 for the numberings as listed below:

1) Select the Protocol as “JTAG/SPD” in the *Protocol* Combo-box.
2) Select the Physical Interface as “UDAS/JTAG over USB” in the *Physical Interface* combo-box.
3) Select the Target Device as “XC866L-4F” in the *Target Device* combo-box.
4) Select the desired binary HEX to FLASH by using the button *Open File*.
5) Ensure that the hardware connection is established between the PC host and the target device as mentioned in Section 4.1.2 and that the target device is powered-up. Then the *COM Port* window will be populated automatically with the proper COM settings in the FLOAD GUI. Next select the *Connect* button to connect to the target device. The LED close to the *Connect* button should go from RED to GREEN.
6) Select the *Verify Programmed Flash* check box to ensure that the desired binary (HEX) has been properly flashed.
7) Select the *Download* button to download and FLASH the binary HEX to the Target Device. Once the progress bar completes, it pops open a message window “Download and Verification are successful”.

5 Software Build Environment

The CIC61508 workspace is located at:

<InstalledPath>\CIC61508\00_Source\CIC61508\sav\cic61508\cic61508_dev.uv2.

5.1 Selecting CIC61508 system clock frequency

A macro CIC61508_CONFIGCLK_75MHZ is defined in the Cic61508_Main.h to change the CIC61508 system clock frequency. Set CIC61508_CONFIGCLK_75MHZ to TRUE for 75 MHz and FALSE for 80 MHz.

Please do a “Clean Target” and “Rebuild all target files” to generate the hex file with a proper checksum.

6 Application Use Case

6.1 Description

This section will provide the detailed procedure to make the system move into the ACTIVE state. The CIC61508 should be in the ACTIVE state to ensure that the working condition of the host controller is normal. The following steps are required to get the CIC61508 into the ACTIVE state:

Note: *Since the CIC61508 for the TriCore safety solution will only support voltage monitoring and the sequencer, the other modules (Task Monitor and the Data Comparator) are disabled as their functions are already covered by SafeTcore.*

6.2 Sample Procedure to move the CIC61508 into the ACTIVE State

1. To make the CIC61508 work, the user has to configure all the available CIC61508 modules. Refer to Section 6.3 for the configurations. Please note that this is just an example and the configurations will change as per the project requirements.
2. If the any of the VoltageX (X=A, B, C, D) Monitoring functions are enabled, the user should make sure that all the monitored voltages should be in between or equal to the threshold values which are configured.
3. Make sure that for every heartbeat the Host only has to send between 5 and a maximum of 8 SPI messages and that timing settings should be appropriate for the respective speeds (Refer to Section 2.3.1 for the timings).
4. After the configuration has been completed and the necessary settings have been made on the Host microcontroller, force the CIC61508 to reset.
5. When the CIC61508 is in the RESET state, the BIST will execute and it will go to the DISABLED state if it fails. Refer to Section 2.1 for the BIST failure conditions. It will move to the NOT READY state if BIST passes.
6. The moment that the CIC61508 reaches the NOT READY state, all the monitoring functions will be initiated and the Counters will increment / decrement on the Pass or Fail condition of each function.
7. To set the SPI communication counter value to its maximum, the Host has to send the SPI Reset Request (by writing A9$_H$ into the MODE SFR).
8. To move the CIC61508 into the ACTIVE state, all the monitoring functions should first be in MAINTAIN state (all the respective counter values should be greater than or equal to 40$_H$).

6.2.1 Steps to move the Sequencer into the Maintain State

1. When the CIC61508 is in the NOT READY state, the window close period (= Minimum window period/WinMin) will be started and the SEQ SFR is updated with the first request number. Here, as per the example configuration, it will be updated with the value 00$_H$.
2. CIC61508 will expect the respective answer for the request number from the Host. The answer from the Host will be written into the following SFRs; OTRHH, OTRHL, OTRLH, OTRLL.
3. Writing into the SFRs OTRHL, OTRLH, and OTRLL can be in any order and can be in either the window close period or the window open period (Minimum Window- Maximum Window/WinMax). The final Write to the OTRHH should be carried out in the window open period. As per the example configuration the final Write should happen after the 1st heartbeat and before the 2nd heartbeat completes.
4. Writing the OTRHH before or after the window open period, or before Writing into the other Sequencer SFRs, will cause the INT SFR to be flagged with a Sequencer error and the counter value will be decremented. (In the Example Configuration, it will be decremented by 08$_H$).
5. Here we need to Write the following answer into the OTRXX SFRs as per the example configuration:
   OTRHH - FF$_H$
   OTRHL - FF$_H$
   OTRLH - FF$_H$
   OTRLL - 00$_H$
6. After Writing into the OTRHH, the CIC61508 will resynchronize the window period to the next heartbeat, and start the window close period. The Sequencer counter will be incremented if the correct answer is sent to the CIC61508 and the SEQ SFR will be updated with the next request number 01$_H$. It will be decremented if the incorrect answer has been sent and the SEQ SFR will be updated with the same request number 00$_H$.

7. Since the increment counter value in the example configuration is 32_H, it requires 2 consecutive correct answers to move the Sequencer into the MAINTAIN state.
8. After completion of the final Sequencer test, the SEQ SFR will be updated with the first request number.
9. It is necessary to follow the above steps repeatedly to keep the system continuously in the MAINTAIN state.

6.2.2 Steps to get the VoltageX Monitors into the MAINTAIN State

1. When the CIC61508 is in the NOT READY state, if the voltage monitoring functions are enabled, the monitored voltages will be sampled for every heartbeat. The respective VOLTMONXX SFRs will be updated with the sampled values. The respective counters will be incremented if the voltage falls under the respective threshold value and will be decremented if not.
2. It is not necessary for the Host to send any SPI messages to do this; it will be done by the CIC61508 itself. Make sure that all the monitored voltages are within the configured threshold values. As per our example configuration, all the counter values are equal to 20_H and it requires 3 heartbeats to reach the MAINTAIN state.
3. As per the example configuration, the threshold values are configured as below:
   - Volt A – 3.5 to 4.0
   - Volt B – 2.5 to 3.0
   - Volt C—3.0 to 3.5
   - Volt D—0.75 to 1.05
4. These sample voltages can be read at any time by using the Coherent Read method (Refer to Section 2.5.2) and it will not affect the count values.
5. The Host can monitor the Voltage Monitor by using the voltage injection method. It will inject the voltage count values into the VOLTMONSFRs and then compare them against the threshold values. It will then increment or decrement, according to the result.

After performing all the above-mentioned steps, the monitoring functions will be in the MAINTAIN state. When all functions are in the MAINTAIN state, issue a GO request (by writing $8A_H$ to the MODE SFR) and the system will move to the ACTIVE state.

The system will be in the ACTIVE state when all monitoring functions are in the MAINTAIN state, but will move to the TRIPPING State 1 if any one of the monitoring functions assumes the ERROR state.

Table 22 will show the set of SPI messages to be sent to move the system into the ACTIVE state, as per the example configuration.

Table 22 SPI Message Sequence from NOT_READY to ACTIVE state

Heart Beat	SPI MSG sent by Host	SPI MSG received by Host	Description of SPI MSG sent by the Host	Description of the SPI MSG received from the CIC61508 and the results
1	$A993_H$	$93A9_H$	Sending SPI reset request	
	0083_H	8300_H	writing answer for REQ #1 into OTRLL, OTRLH, OTRHL SFRs	
	$FF82_H$	$82FF_H$		
	$FF81_H$	$81FF_H$		
	0095_H	9500_H	Initiating Coherent Read for Volt A	
2	$FF80_H$	$80FF_H$	writing answer for REQ #1 into OTRHH	
	$DD14_H$	$14(VAL1)_H$	Reading the VOLTMONAH SFR	It will read the sampled voltage value and it should be equal to the respective tuned voltage value
	$DD15_H$	$15(VAL2)_H$	Reading the VOLTMONAL SFR	
	0097_H	9700_H	initiating Coherent Read for Volt B	
	$DD15_H$	$15(COUNT)_H$	Reading PASSCNTCOMM SFR	Since in the previous Heartbeat the Host will have sent the SPI request, that makes the PASSCNTCOMM value MAX (80_H).

Heart Beat	SPI MSG sent by Host	SPI MSG received by Host	Description of SPI MSG sent by the Host	Description of the SPI MSG received from the CIC61508 and the results
3	$DD09_H$	$09(COUNT)_H$	Reading PASSCNTVA SFR	We can read all the voltage monitor counter values. The expected count value as per the example configuration should be more than 40_H. Here COUNT represents the respective counter values
	$DD0A_H$	$0A(COUNT)_H$	Reading PASSCNTVB SFR	
	$DD0B_H$	$0B(COUNT)_H$	Reading PASSCNTVC SFR	
	$DD0C_H$	$0C(COUNT)_H$	Reading PASSCNTVD SFR	
	$DD08_H$	$08(COUNT)_H$	Reading PASSCNTSEQ SFR	The Sequencer counter will increment as per the example configuration and is equal to 32_H.
4	1183_H	8311_H	writing answer for REQ #2 into OTRLL, OTRLH, OTRHL and OTRHH SFRs	
	$FF82_H$	$82FF_H$		
	$FF81_H$	$81FF_H$		
	$FF80_H$	$80FF_H$		
	$DDDD_H$		Dummy message	
5	$DD08_H$	$08(COUNT)_H$	Reading PASSCNTSEQ SFR	The Sequencer counter will increment as per the example configuration and it is more than 40_H.
	$DD16_H$	16	Reading SUM0	By reading these two registers the Host can establish the state of all the modules. In the example configuration all the modules will reach the MAINTAIN state.
	$DD17_H$		Reading SUM1	
	DD07	$073C_H$	Reading SYSTEMINTEGRITY SFR	By reading this register the Host can establish the state of the CIC61508. In the example configuration the CIC61508 will reach the READY state.
	$DDDD_H$		Dummy message	
6	$8A93_H$	$938A_H$	Writing Go request in MODE SFR	
	$FF83_H$	$83FF_H$	writing respective answer for REQ #3 into OTRLL, OTRLH, OTRHL and OTRHH SFRs	
	$FF82_H$	$82FF_H$		
	0081_H	8100_H		
	0080_H	8000_H		
7	DD07	$071E_H$	Reading SYSTEMINTEGRITY SFR	Since the Host issues the GO request in the previous heartbeat the system will move to the ACTIVE state.
	$DD09_H$	$15(COUNT)_H$	Reading PASSCNTVA SFR	Reading the Counter values. Here COUNT represents the respective counter values.
	$DD0A_H$	$09(COUNT)_H$	Reading PASSCNTVB SFR	
	$DD0B_H$	$0A(COUNT)_H$	Reading PASSCNTVC SFR	
	$DD0C_H$	$0B(COUNT)_H$	Reading PASSCNTVD SFR	

The SPI format mentioned in Table 22 is defined in Section 2.3.3. The higher byte is the data part and the lower byte is the command part. While sending the Read command, the data part will not have much importance, hence the dummy data DD_H. The reply for the answer would be in reverse order (the command byte is in the higher byte and the data in the lower byte).

6.3 Example Configuration Settings

Here we provide an example configuration, with settings for the CIC61508 to monitor all the available functions (Sequencer, Voltage Monitor and the Integrity Monitor). The configuration can be updated in the DFLASH area by using the TARDISS tool (Refer to **Section 3**).

***Note:** The default DFLASH configuration provided by Infineon will work with SafeTcore-I releases (TriCore-based). The choice of updating the configuration parameters such as the Sequencer table is entirely up to the user and Infineon is not responsible for any unexpected results.*

6.3.1.1 Integrity Monitor Configuration

In this section we need to configure the following things:

- Pass Counter Increment and Decrement Value
- Monitor Function Enable
- Tripping Time Configuration

Table 23 Pass Counter Increment and Decrement Values

Address	Parameter	Value
$A6C0_H$	Sequencer Increment Value	32_H
$A6C1_H$	Sequencer Decrement Value	8_H
$A6C2_H$	Voltage Monitor A Increment Value	20_H
$A6C3_H$	Voltage Monitor A Decrement Value	8_H
$A6C4_H$	Voltage Monitor B Increment Value	20_H
$A6C5_H$	Voltage Monitor B Decrement Value	8_H
$A6C6_H$	Voltage Monitor C Increment Value	20_H
$A6C7_H$	Voltage Monitor C Decrement Value	8_H
$A6C8_H$	Voltage Monitor D Increment Value	20_H
$A6C9_H$	Voltage Monitor D Decrement Value	8_H
$A6CA_H$	Task Monitor Increment Value	01_H
$A6CB_H$	Task Monitor Decrement Value	01_H
$A6CC_H$	Data Comparator Increment Value	01_H
$A6CD_H$	Data Comparator Decrement Value	01_H

Table 24 Monitor Function Enable

Address	Value	Monitor
$A6CE_H$	00_H	Voltage Monitor channel A
$A6CF_H$	00_H	Voltage Monitor channel B
$A6D0_H$	00_H	Voltage Monitor channel C
$A6D1_H$	00_H	Voltage Monitor channel D
$A6D2_H$	40_H	Task Monitor
$A6D3_H$	40_H	Data Comparator

Table 25 Tripping Time

Address	Parameter	Value
$A6D4_H$	Tripping 1 time	01_H

A6D5$_H$	Tripping 2 time	01$_H$
A6D6$_H$	Tripping 3 time	01$_H$

6.3.1.2 Sequencer

Here we need to configure the following things:

- Test Request Number
- Answer for the respective request number
- Table length
- Window Maximum and Minimum period.

Here the table length should be a minimum of 08$_H$ and Maximum 40$_H$.

Table 26 Sequencer Configuration

Address	Parameter	Value
A000$_H$	Test Request #1	00$_H$
A001$_H$	Answer to Test Request #1 (High-High Byte)	FF$_H$
A002$_H$	Answer to Test Request #1 (High-Low Byte)	FF$_H$
A003$_H$	Answer to Test Request #1 (Low-High Byte)	FF$_H$
A004$_H$	Answer to Test Request #1 (Low-Low Byte)	00$_H$
A005$_H$	Test Request #2	01$_H$
A006$_H$	Answer to Test Request #2 (High-High Byte)	FF$_H$
A007$_H$	Answer to Test Request #2 (High-Low Byte)	FF$_H$
A008$_H$	Answer to Test Request #2 (Low-High Byte)	FF$_H$
A009$_H$	Answer to Test Request #2 (Low-Low Byte)	11$_H$
A00A$_H$	Test Request #3	02$_H$
A00B$_H$	Answer to Test Request #3 (High-High Byte)	00$_H$
A00C$_H$	Answer to Test Request #3 (High-Low Byte)	00$_H$
A00D$_H$	Answer to Test Request #3 (Low-High Byte)	FF$_H$
A00E$_H$	Answer to Test Request #3 (Low-Low Byte)	FF$_H$
A00A$_H$	Test Request #4	03$_H$
A00B$_H$	Answer to Test Request #4 (High-High Byte)	11$_H$
A00C$_H$	Answer to Test Request #4 (High-Low Byte)	11$_H$
A00D$_H$	Answer to Test Request #4 (Low-High Byte)	11$_H$
A00E$_H$	Answer to Test Request #4 (Low-Low Byte)	11$_H$
A00A$_H$	Test Request #5	04$_H$
A00B$_H$	Answer to Test Request #5 (High-High Byte)	FF$_H$
A00C$_H$	Answer to Test Request #5 (High-Low Byte)	FF$_H$
A00D$_H$	Answer to Test Request #5 (Low-High Byte)	00$_H$
A00E$_H$	Answer to Test Request #5 (Low-Low Byte)	00$_H$
A00A$_H$	Test Request #6	05$_H$
A00B$_H$	Answer to Test Request #6 (High-High Byte)	22$_H$
A00C$_H$	Answer to Test Request #6 (High-Low Byte)	22$_H$

A00D$_H$	Answer to Test Request #6 (Low-High Byte)	22$_H$
A00E$_H$	Answer to Test Request #6 (Low-Low Byte)	22$_H$
A00A$_H$	Test Request #7	06$_H$
A00B$_H$	Answer to Test Request #7 (High-High Byte)	33$_H$
A00C$_H$	Answer to Test Request #7 (High-Low Byte)	33$_H$
A00D$_H$	Answer to Test Request #7 (Low-High Byte)	33$_H$
A00E$_H$	Answer to Test Request #7 (Low-Low Byte)	33$_H$
A00A$_H$	Test Request #8	07$_H$
A00B$_H$	Answer to Test Request #8 (High-High Byte)	AA$_H$
A00C$_H$	Answer to Test Request #8 (High-Low Byte)	AA$_H$
A00D$_H$	Answer to Test Request #8 (Low-High Byte)	AA$_H$
A00E$_H$	Answer to Test Request #8 (Low-Low Byte)	AA$_H$
Address	**Min Window**	
A140$_H$	01$_H$	
Address	**Max Window**	
A141$_H$	03$_H$	
Address	**Length**	
A142$_H$	08$_H$	

6.3.1.3 Voltage Monitor Configuration

Table 27 Voltage Monitor Configuration

Address	Parameter	Value
A6A0$_H$	Voltage Monitor A Minimum Count (High Byte)	B3$_H$
A6A1$_H$	Voltage Monitor A Minimum Count (Low Byte)	40$_H$
A6A2$_H$	Voltage Monitor A Maximum Count (High Byte)	CC$_H$
A6A3$_H$	Voltage Monitor A Maximum Count (Low Byte)	C0$_H$
A6A4$_H$	Voltage Monitor B Minimum Count (High Byte)	80$_H$
A6A5$_H$	Voltage Monitor B Minimum Count (Low Byte)	00$_H$
A6A6$_H$	Voltage Monitor B Maximum Count (High Byte)	99$_H$
A6A7$_H$	Voltage Monitor B Maximum Count (Low Byte)	80$_H$
A6A8$_H$	Voltage Monitor C Minimum Count (High Byte)	99$_H$
A6A9$_H$	Voltage Monitor C Minimum Count (Low Byte)	80$_H$
A6AA$_H$	Voltage Monitor C Maximum Count (High Byte)	B3$_H$
A6AB$_H$	Voltage Monitor C Maximum Count (Low Byte)	40$_H$
A6AC$_H$	Voltage Monitor D Minimum Count (High Byte)	4C$_H$
A6AD$_H$	Voltage Monitor D Minimum Count (Low Byte)	C0$_H$
A6AE$_H$	Voltage Monitor D Maximum Count (High Byte)	66$_H$
A6AF$_H$	Voltage Monitor D Maximum Count (Low Byte)	40$_H$

7 Configuration Guidelines

For a safe system, it is mandatory for Host microcontroller to ensure that the correct configurations in the system in operation. The CIC61508 performs BIST on its internal configurations. However, this does not guarantee that the correct configuration is deployed in the system; hence it is assumed that correct configuration is ensured on the host side. The following sections are recommendations or shall serve as a checklist to enhance the system robustness.

Note: Since the CIC61508 for the TriCore safety solution will only support voltage monitoring and the sequencer, the other modules (Task Monitor and the Data Comparator) are disabled as their functions are already covered by SafeTcore.

7.1 Logical Monitoring

Table 1 provides cases of logical monitoring in the safety system.

Table 28 Logical monitoring description

State	Checks by the Integrator	Description
BIST Test has passed and NOT READY state is entered. The checks described shall be performed only once during start-up of the system	Correct user dflash configuration	Enter into secured SPI mode and read the Dflash release number, at the last 16 bytes of the Dflash. Upon confirmation that the correct user dflash configuration is used, the host shall issue perform a software reset on the CIC61508. The results of this test shall be stored in the Host. This operation shall be carried out only once.This is to ensure that the correct Dflash configuration is used before starting the system.
	Voltage Supply monitor integrity	Observe the changing of the voltage monitor pass counters through the injecting of the voltage monitor readings. The readings will comprise of testing for the minimum and maximum ranges; both within and outside these ranges. This is to ensure that the votage supply monitoring functionality is working before starting the system.
	Coherence of the Error state with the Fail-Safe path state	Check the error state with the pin state of the fail-safe path. This ensures for state coherence before starting the system.
	Correct ROM version used	Access the CIC61508 through the Host microcontroller for the SVER SFR. This is to ensure that the correct ROM version is used before starting the system.
Before transition into each state and throughout active operation	Communication integrity	It is recommended to check the SPI pass counter to acertain the communication integrity. Through the host microcontroller can choose to disable operation of the system from active state or issue a SPI Reset request to reset the SPI pass counter to maintain in active communication state.

7.2 Temporal Monitoring

Table 29 provides cases of temporal monitoring in the safety system. Ensure that the system is not disabled, as a result of this monitoring.

Table 29 **Temporal monitoring description**

State	Checks by the Integrator	Description
Before transition into each state and throughout active operation	Sequencer integrity	Inject errors in the sequencer test answers to the sequencer test requests to ensure correct monitor functioning. Observe a drop in the monitor pass counter as a result of the incorrect answer sent.

7.3 Configuring the Sequencer Table

The CIC61508 challenge and response system using the Sequencer can be configured in a very flexible way. However, this does not guarantee for the highest monitoring effectiveness.

The following guidelines are recommended to calibrate the device to increase monitoring effectiveness using the Sequencer. Table 30 shows an example sequencer table that fulfills the above recommendations.

1. Ensure at least 10 sets of different test requests.
2. Ensure that the period of the same test request changes.
3. Ensure that each byte in the test answer to be different.
4. Avoid having same test answers to different test requests.
5. Avoid having trivial test answers like 0x00000000 or 0xFFFFFFFF.

Table 30 **Sequencer Table example**

TEST REQUEST	TEST ANSWER
TEST0	0x1A2B3C4D
TEST1	0x5E6F7890
TEST2	0x12345678
TEST3	0x45678923
TEST4	0x98765432
TEST5	0x184263FD
TEST6	0x68402143
TEST7	0x09FEDCBA
TEST8	0x987312AB
TEST9	0xFEDCBA98
TEST5	0x184263FD
TEST6	0x68402143
TEST7	0x09FEDCBA
TEST8	0x987312AB
TEST9	0xFEDCBA98

TEST0	0x1A2B3C4D
TEST1	0x5E6F7890
TEST2	0x12345678
TEST3	0x45678923
TEST4	0x98765432