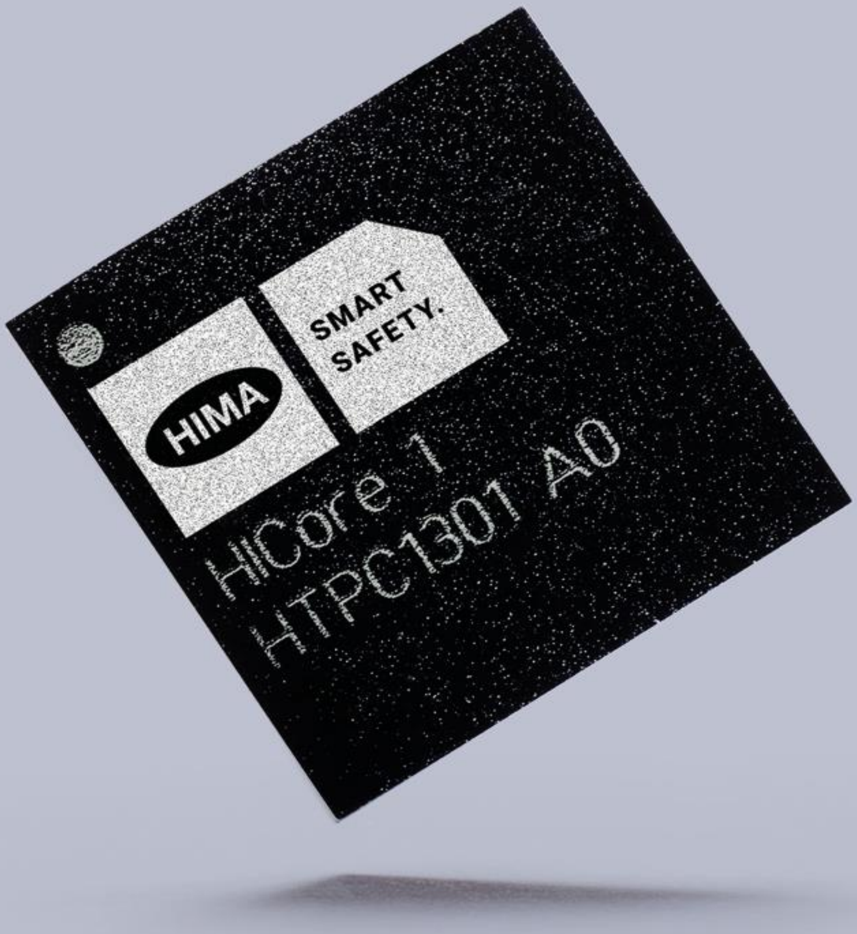
Manual

HICore®

Errata Sheet

HICore 1 Software Package



Contact

HIMA Paul Hildebrandt GmbH

P.O. Box 1261

68777 Brühl

Phone: +49 6202 709-0

Fax: +49 6202 709-107

E-mail: info@hima.com

Revision index	Revisions	Type of change	
		technical	editorial
1.00	First edition of the manual	X	X

Table of Contents

1 Introduction

The HIC1SP Errata Sheet is a collection of differences between the implementation of the HIC1SP and the description of the software package contained in the other HIC1SP related documents, HIC1SP System Manual and HIC1SP Safety Manual. Please check for the latest version of this document.

The copyright and other intellectual property rights in this document, including photographs and graphical images are owned by HIMA or its licensors.

1.1 Related Documents

Refer to the following documents for other HIC1SP related topics:

▪ HI_801_476_E_HICore_1_System_Manual	Description of the HIC1SP functions
▪ HI_801_472_E_HICore_1_Safety_Manual	Safety Manual of the HIC1SP
▪ HI_801_546_E_HICore_1_Errata_Sheet	Errata Sheet of the HIC1SP

Always use the latest available version of any technical documentation.

2 Known Errata SafeDomain

2.1 Output Parameter of CovershellServices::GetSystemTime() inconsistent at Interrupt Occurrence

Errata type	Data integrity
Classification	**potentially safety-critical**
Affected Version	HIC1SP V1.x
Affected Component	Covershell: `CovershellServices::GetSystemTime(Timestamp_type &out_rTTSystemTime)`
Description	If the SystemTimerInterrupt occurs during the copy process of the internal variable to the output parameter the returned system time might be inconsistent if an overflow occurred in the LSB (and higher) during the SystemTimerInterrupt. This results in an inconsistent value as some of the higher bytes have the new value and remaining bytes have the last value. The exact pattern depends on when exactly during the copy process the interrupt occurs. Examples (for inconsistent output parameter values): overflow of Byte 0: correct Value before interrupt: 12 34 56 78 FF correct Value after interrupt: 12 34 56 79 00 returned value: 12 34 56 79 FF overflow of Byte 0 … 1: correct Value before interrupt: 12 34 56 FF FF correct Value after interrupt: 12 34 57 00 00 returned value: 12 34 57 FF FF or: 12 34 57 00 FF overflow of Byte 0 … 2: correct Value before interrupt: 12 34 FF FF FF correct Value after interrupt: 12 35 00 00 00 returned value: 12 35 FF FF FF or: 12 35 00 FF FF or: 12 35 00 00 FF overflow of Byte 0 … 3: correct Value before interrupt: 12 FF FF FF FF correct Value after interrupt: 13 00 00 00 00 returned value: 13 FF FF FF FF or: 13 00 FF FF FF or: 13 00 00 FF FF or: 13 00 00 00 FF

Workaround	1. include header with hardware definitions **`"htpc1301_hw_defines.h"`**
	2. Disable interrupts during the call of CovershellServices:: GetSystemTime(Timestamp_type &out_rTTSystemTime) using the macros DISABLE_IRQ and ENABLE_IRQ.
	
	Coding example:
	`#include "htpc1301_hw_defines.h"`
	
	`(…)`
	`CovershellServices &services =`
	`CovershellServices::Instance();`
	`(…)`
	`{`
	`  /// disable interrupts`
	`  irqmask_t mask;`
	`  DISABLE_IRQ(mask)`
	`  /// call function`
	`  services.GetSystemTime(out_rTTSystemTime);`
	`  /// enable interrupts`
	`  ENABLE_IRQ(mask)`
	`}`
	
	Note: If you implement this workaround you shall make sure that the macros are used exactly as shown in the coding example and enclose no other code. In this case it is allowed to violate SRARQ_0400 (HI_801_472_E_HICore_1_Safety_Manual).

Workaround by avoidance	Make sure that the call of `CovershellServices::GetSystemTime()` occurs before the first SystemTimerInterrupt directly at the beginning of the cycle.
Fix plan	Will be fixed in HICore 1 Software Package V2.x.

2.2 Self-test CovershellServices::TestSFR() incorrectly fails at Interrupt Occurrence

Errata type	Data integrity
Classification	**Availability problem**
Affected Version	HIC1SP V1.x
Affected Component	Covershell: `CovershellServices::TestSFR(eDiagnosisState &out_reTestState)`
Description	Test method `CovershellServices::TestSFR()` executes the HIC1SP self-tests on the SFR registers. In the worst case a triggering of the SystemTimerInterrupt can lead to an incorrect test result and HICore 1 enters system error state. Possible system error codes are: • Error ID: 823 - REF0-SFR value check failed • Error ID: 824 - REF1-SFR value check failed • Error ID: 825 - REF2-SFR value check failed • Error ID: 826 - REF3-SFR value check failed

Workaround	The OS executes the HIC1SP self-tests, which also include the SFR tests, directly after SafeApp has been processed. With a system clock of 100MHz, the SFR tests should be completed after max. 300µs (at 50 MHz, the SFR tests should be completed after max. 600µs). In order to run the HIC1SP self-tests without interruption by the SystemTimerInterrupt, the HIC1SP self-tests must be executed immediately after the SystemTimerInterrupt.
	To achieve this, the system time must be read out several times at the end of the cyclically executed `SafeApp::Tick()` method until the lowest byte changes. From this point on, it can be assumed that a SystemTimerInterrupt will no longer be triggered for just under 1 ms. Then the `SafeApp::Tick()` method is ended so that the SFR test is executed without interruption.
	
	Coding example for HIC1SP SafeApp:
	`#include "htpc1301_hw_defines.h"`
	
	`(…)`
	`pr_type SafeApp::Tick(void)`
	`{`
	
	`   (…)`
	
	`// Start CovershellServices::TestSFR ClockGuard-Register workaround`
	
	`  CovershellServices &covershellServices = CovershellServices::Instance();`
	
	`  // disable interrupts`
	`  irqmask_t mask;`
	`  DISABLE_IRQ(mask)`
	`  // call function`
	`  Timestamp_type currentTime;`
	
	`  SAFETY_ASSERT(covershellServices.GetSystemTime(currentTime)  ==`

	`covershell::CovershellServices::s_prGETSYSTEMTIME, 10000U);`
	`  // enable interrupts`
	`  ENABLE_IRQ(mask)`
	`  // wait for next SystemTimerInterrupt that increment the system time`
	`  Timestamp_type nextTime;`
	`  nextTime = currentTime;`
	`  do {`
	`    // disable interrupts`
	`    DISABLE_IRQ(mask)`
	`    // call function`
	`    SAFETY_ASSERT(covershellServices.GetSystemTime(nextTime) == covershell::CovershellServices::s_prGETSYSTEMTIME, 10001U);`
	`    // enable interrupts`
	`    ENABLE_IRQ(mask)`
	`  } while (currentTime.lowUdword.ubyte[0] == nextTime.lowUdword.ubyte[0]);`
	
	`// end workaround`
	
	`  prCheckpoint++;`
	`  return prCheckpoint;`
	`}`
	
	Note: If you implement this workaround you shall make sure that the macros are used exactly as shown in the coding example and enclose no other code. In this case it is allowed to violate SRARQ_0400 (HI_801_472_E_HICore_1_Safety_Manual).

Workaround by avoidance	-
Fix plan	Will be fixed in HICore 1 Software Package V2.x.

2.3 Output Parameter of CovershellServices::GetUnusedCycleTime() Inconsistent at Interrupt Occurrence

Errata type	Data integrity
Classification	**Minor Error - Not safety critical and no influence to the availability**
Affected Version	HIC1SP V1.x
Affected Component	Covershell: `CovershellServices::GetUnusedCycleTime (uword & out_ru16UnusedCycleTime)`
Description	The method `CovershellServices::GetUnusedCycleTime (uword & out_ru16UnusedCycleTime)` returns the lowest remaining cycle time `out_ru16UnusedCycleTime` that has been measured since the system was started. In the worst case a triggering of the SystemTimerInterrupt can result in an inconsistent output value `out_ru16UnusedCycleTime`.
Workaround	1. include header with hardware definitions **`"htpc1301_hw_defines.h"`** 2. Disable interrupts during the call of `CovershellServices::GetUnusedCycleTime(uword &out_ru16UnusedCycleTime)` using the macros `DISABLE_IRQ` and `ENABLE_IRQ`. 3. End the SafeApp with the end of the last SystemTimerInterrupt, to reach the maximum time until the next SystemTimerInterrupt Voding example:

```
#include "htpc1301_hw_defines.h"

(...)

pr_type SafeApp::Tick(void)
{

(...)

  CovershellServices &covershellServices =
CovershellServices::Instance();

(...)

// Start CovershellServices::GetUnusedCycleTime
woraround
  // disable interrupts
  irqmask_t mask;
  DISABLE_IRQ(mask)
  // call function
  covershellServices.GetUnusedCycleTime(out_ru16UnusedCy
cleTime);
  // enable interrupts
  ENABLE_IRQ(mask)
// end CovershellServices::GetUnusedCycleTime workaround

(...)

// At the end of the SafeApp wait until the next
millisecond starts:
  // disable interrupts
  irqmask_t mask;
  DISABLE_IRQ(mask)
  // call function
  Timestamp_type currentTime;
  SAFETY_ASSERT(covershellServices.GetSystemTime(current
Time) ==
covershell::CovershellServices::s_prGETSYSTEMTIME,
```

	```
	10000U);
	  // enable interrupts
	  ENABLE_IRQ(mask)
	  // wait for next SystemTimerInterrupt that increment
	the system time
	  Timestamp_type nextTime;
	  nextTime = currentTime;
	  do {
	    // disable interrupts
	    DISABLE_IRQ(mask)
	    // call function
	    SAFETY_ASSERT(covershellServices.GetSystemTime(nextT
	ime) ==
	covershell::CovershellServices::s_prGETSYSTEMTIME,
	10001U);
	    // enable interrupts
	    ENABLE_IRQ(mask)
	  } while (currentTime.lowUdword.ubyte[0] ==
	nextTime.lowUdword.ubyte[0]);


	  prCheckpoint++;
	  return prCheckpoint;
	}
	```
	Note: If you implement this workaround you shall make sure that the macros are used exactly as shown in the coding example and enclose no other code. In this case it is allowed to violate SRARQ_0400 (HI_801_472_E_HICore_1_Safety_Manual).

Workaround by avoidance	-
Fix plan	Will be fixed in HICore 1 Software Package V2.x.

2.4 Output Parameter of CovershellServices::SetDiagnosisEntry() and Diagnosis::SetFaultEntry() Inconsistent at Interrupt Occurrence

Errata type	Data integrity
Classification	**Minor Error - Not safety critical and no influence to the availability**
Affected Version	HIC1SP V1.x
Affected Component	`CovershellServices::SetDiagnosisEntry(…)` **and** `Diagnosis::SetFaultEntry(…)`
Description	If the SystemTimerInterrupt occurs during the copy process of the internal variable the copied system time might be inconsistent if an overflow occurred in the LSB (and higher) during the SystemTimerInterrupt. This results in an inconsistent value as some of the higher bytes have the new value and remaining bytes have the last value. The exact pattern depends on when exactly during the copy process the interrupt occurs. Examples (for inconsistent output parameter values): overflow of Byte 0: correct Value before interrupt: 12 34 56 78 FF correct Value after interrupt: 12 34 56 79 00 returned value: 12 34 56 79 FF overflow of Byte 0 … 1: correct Value before interrupt: 12 34 56 FF FF correct Value after interrupt: 12 34 57 00 00 returned value: 12 34 57 FF FF or: 12 34 57 00 FF overflow of Byte 0 … 2: correct Value before interrupt: 12 34 FF FF FF correct Value after interrupt: 12 35 00 00 00 returned value: 12 35 FF FF FF or: 12 35 00 FF FF or: 12 35 00 00 FF overflow of Byte 0 … 3: correct Value before interrupt: 12 FF FF FF FF correct Value after interrupt: 13 00 00 00 00 returned value: 13 FF FF FF FF or: 13 00 FF FF FF or: 13 00 00 FF FF or: 13 00 00 00 FF
Workaround	No workaround is planned. The diagnostic and fault entries are written into the NVRAM on the ComDomain and output as a diagnostic message via the UART. The entries have no influence on the execution and only serve to inform the user. An incorrect time stamp in the diagnostic / fault entry has no effect on safety and availability of HICore 1.
Workaround by avoidance	Make sure that the call of `CovershellServices::SetDiagnosisEntry()` occurs before the first SystemTimerInterrupt directly at the beginning of the cycle.
Fix plan	Will be fixed in HICore 1 Software Package V2.x.

3 Known Errata ComDomain

3.1 Output Parameter of SystemServices::GetSystemTime() Inconsistent at Interrupt Occurrence

Errata type	Data integrity
Classification	**Availability problem**
Affected Version	HIC1SP V1.x
Affected Component	Comtools: `SystemServices::GetSystemTime(Timestamp_type &out_rTTSystemTime)`
Description	If the SystemTimerInterrupt occurs during the copy process of the internal variable to the output parameter the returned system time might be inconsistent if an overflow occurred in the LSB (and higher) during the SystemTimerInterrupt. This results in an inconsistent value as some of the higher bytes have the new value and remaining bytes have the last value. The exact pattern depends on when exactly during the copy process the interrupt occurs. Examples (for inconsistent output parameter values): overflow of Byte 0: correct Value before interrupt: 12 34 56 78 FF correct Value after interrupt: 12 34 56 79 00 returned value: 12 34 56 79 FF overflow of Byte 0 … 1: correct Value before interrupt: 12 34 56 FF FF correct Value after interrupt: 12 34 57 00 00 returned value: 12 34 57 FF FF or: 12 34 57 00 FF overflow of Byte 0 … 2: correct Value before interrupt: 12 34 FF FF FF correct Value after interrupt: 12 35 00 00 00 returned value: 12 35 FF FF FF or: 12 35 00 FF FF or: 12 35 00 00 FF overflow of Byte 0 … 3: correct Value before interrupt: 12 FF FF FF FF correct Value after interrupt: 13 00 00 00 00 returned value: 13 FF FF FF FF or: 13 00 FF FF FF or: 13 00 00 FF FF or: 13 00 00 00 FF This possible misconduct affects the following methods, which use `SystemServices::GetSystemTime(Timestamp_type &out_rTTSystemTime)` to call up the time stamp: 1. **`ComtoolsServices::SetDiagnosisEntry():`** The method `ComtoolsServices::SetDiagnosisEntry()` writes the current time stamp as part of the diagnostic entry. In the worst case a triggering of the SystemTimerInterrupt can lead to an inconsistency in the written time stamp.

	2. **`ComtoolsServices::GetSystemTime():`** The method `ComtoolsServices::GetSystemTime(Timestamp_type & out_rTTSystemTime)` returns the current time stamp as an output parameter. In the worst case a triggering of the SystemTimerInterrupt can lead to an inconsistency in the written time stamp. 3. **`Download::Tick():`** The method `Download::Tick(bool & out_rbDownloadStarted, bool & out_rbDownloadFinished, bool & out_rbDownloadAborted)` is used to download new firmware to the HICore 1. In the worst case a triggering of the SystemTimerInterrupt can lead to an inconsistency in the written time stamp. And the HICore 1 firmware upload must be restarted.

Workaround	Workaround for
	`ComtoolsServices::GetSystemTime(Timestamp_type &out_rTTSystemTime):`
	
	1. Possibility:
	To detect and avoid the error, the system time can be called up twice in succession and the small value can be used.
	2. Possibility:
	If faster performance is required, the SystemTimerInterrupt can be switched off for the duration of the `ComtoolsServices::GetSystemTime()` function call:
	Coding example:
	`#include "htpc1301_hw_com_defines.h"`
	
	`(…)`
	
	`pr_type ComApp::Tick(void)`
	`{`
	
	`(…)`
	
	`  comtools::ComtoolsServices &comtoolsServices = comtools::ComtoolsServices::Instance();`
	`  /// backup interrupt enable register (IE)`
	`  const ubyte u8OldIE = IE;`
	`  /// deactivate Timer 2 IRQ`
	`  IE &= static_cast<ubyte>(~static_cast<unsigned int>(INT_IE_ET2));`
	`  /// call GetSystemTime`
	`  comtoolsServices.GetSystemTime(out_rTTSystemTime);`
	`  /// restore interrupt enable register (IE)`
	`  IE = u8OldIE;`
	
	`(…)`
	
	`}`
	
	Note: If you implement this workaround you shall make sure that the macros are used exactly as shown in the coding example and enclose no other code.

	Workaround for **`Download::Tick():`** Restart the HICore 1 firmware upload. **There is no workaround for the other methods, the error will be fixed in HICore 1 Software Package V2.x.**
Workaround by avoidance	Make sure that the call of `ComtoolsServices::GetSystemTime()` occurs before the first SystemTimerInterrupt directly at the beginning of the cycle.
Fix plan	Will be fixed in HICore 1 Software Package V2.x.

Appendix

Appendix A - Glossary

Term	Description
Cycle	Unlimited repetition of calls to self-tests, functions and applications inside the SafeDomain
ComDomain	ComHW, ComSW and ComApp
ComApp	Non-safety related application of the HIC1SP
ComBL	Bootloader that starts the SW of the ComDomain. ComSW has four different variants of ComBL: ComBL_BASE, ComBL_UART2, ComBL_SILENT and ComBL_SPI_NVRAM
ComDomain	ComHW, ComSW and ComApp
ComHW	Collection of all non-safety related HW components of the ASI1, comprising of ComBaseHW, ComI/O-Hardware and BusCommunicationHW
ComOS	The non-safety related Operating System of the ComDomain
ComSW	Non-safety related components of the HICore 1 Software Package, comprising of ComBL, ComOS, Comtools, BusCommunication and ComI/O-drivers without ComApp
ComTools	ComHW drivers and the ComAPIs of the ComDomain
HIC1SP	HICore 1 Software Package
HICore	Generic term for the product family of all HICore, i.e. HICore 1, HICore 2, etc.
HICore 1	Generic term for the product family of all HICore 1, comprising of hardware (HTPC1301, HTPC1302 or others), software and documentation. Safety functions can be built with this product
HICore 1 Software Package	Software package comprising of software components for SafeDomain and ComDomain
HICore 1-System	Combination of SafeDomain and ComDomain
HTPC1301	The ASIC itself: HIMA Triple Processor Core 1=Product family, 3=3 Cores, 01=Variant
SafeApp	Safety related application on SafeDomain
SafeBL	Bootloader that starts the SW of the SafeDomain
SafeCode	SW comprising of SafeOS, Covershell and SafeApp
SafeCommunication	Communication protocols and drivers for communication within the SafeDomain
SafeDomain	SafeHW, SafeSW and SafeApp
SafeHW	Collection of all safety related HW components of the HICore 1
SafeIO	IODevice driver
SafeOS	The safety related Operating System of the SafeDomain
SafeSW	Safety related components of the HICore 1 Software Package, comprising of SafeBL, SafeOS, Covershell, SafeCommunication and SafeI/O-drivers without SafeApp
Safety-critical	Condition that can lead to a significant increase in the safety risk for the people or environment involved
System Time	The system time is the time elapsed since system startup, measured in milliseconds

Manual (Errata Sheet)
HICore 1HICore 1 Software Package

HI 801 546 E

For further information, please contact:

HIMA Embedded Solutions

Phone:	+49 6202 709-0
E-Mail:	hicore-support@hima.com

Learn more online about HIMA solutions for HICore 1:

www.hicore.info

www.hima.com