SafeTcore-II Frequently Asked Questions

Compiled by Dave Greenhill, 16-bit Applications Manager at Hitex UK

Organisations increasingly have to ensure that their products meet various safety standards such as IEC61508 or ISO26262 and as a result we are often asked questions relating to the PRO-SIL solution.

We have pulled these questions and their answers together as a useful guide. As time goes on, the list of questions will grow. Equally, if you have a question you need help with, do get in touch!

What is PRO-SIL, CIC61508 and SafeTcore?

PRO-SIL is an umbrella that covers the whole safety solution available from Infineon / Hitex. This solution comprises a programmed, windowed watchdog companion IC (the CIC61508) which connects to the main TriCore or XC2300 processor using an SPI interface. The TriCore or XC2300 runs the customer application code along with the SafeTcore library. Also included in the PRO-SIL package is the complete documentation required to help make the safety case. Using the PRO-SIL system means that the end user focus is on the main application code which hooks into the SafeTcore and in turn, communicates with the CIC61508 to provide a solution for ASIL-D applications.

What are the differences between latent and multiple point faults?

Latent faults arise from multiple point faults. In ISO26262 they are more correctly termed “Latent Dual Point Faults”. They are faults which could happen if two things failed. However these faults are not detected by the safety system or the driver of the car and cause no loss of operation or safety.

Can the XC2300 support ASIL-D?

As the SBST yields > 90% diagnostic coverage of single point faults it meets the requirements of ASIL-B. From the FMEDA, XC2300 could support ASIL-D applications but only if the software uses practices such as using diverse threads. It is really only ASIL-B(D) as the CIC61508 external safety monitor has to be present.

Is SafeTcore-II going to be supplied as source code?

Yes.

Is the CIC61508 Hex file going to be supplied FOC?

Yes!

How does the SBST work with different number of slices? Is the 'answer' is different for 12 slices than it is for 14 etc? How does the CIC61508 know, or does the corrector table in the XC change depending how many slices there are?

The corrector table has to be adapted for the number of slices used. This one is the 12 slice one:

```
SCII_uint32 SCII_CONST_DATA_SPACE Opc_ResultOffset[CIC_OpcTableLength]
=
{

                /* BARCELONA corrector table for 12 slices (90%
coverage) */
                /* Table = SBST Output + Corrector Table */

                0xA2CF1C3AU,
                0x5924750EU,
                0x18D8A615U,
                0xC2F10AD3U,
                0xAE8DDA72U,
                0x74EEB716U,
                0x210D57EDU,
                0xE33958BBU,
                0x5C708002U,
                0x26DED2D6U,
                0xCA69022FU,
                0x988C70C9U,
                0x71334DF2U,
                0x2B8D04A6U,
                0xDBB1C42FU,
                0xA1C8AF09U

} ;
```

The 14 slice would be different as the CRC accumulated in the MCHK would be different. The corrector table is obtained by capturing the actual SBST output in the MCHK into an array and then subtracting each value from the BARCELONA table. A spreadsheet can be used.

For a SIL3 system, what other measures would need to be combined with the SBST?

To get the SIL3 and (ASIL-D), the other measures include the diverse threads, redundant data copies, ECC on SRAM and FLASH etc. i.e. the facilities provided by PRO-SIL XC2000.

How and why might one detect too many or too few interrupts?

This measure is up to the user to implement. In simple terms, if you have a 1ms scheduler interrupt, you put a interrupt counter into the service routine and then in another interrupt - say every 10ms - so you make sure that the counter == 10. This could be added to the reference app. Alternatively, the task monitor could be used if the task sequence in the DFLASH table was:

1ms Task ID = '1'

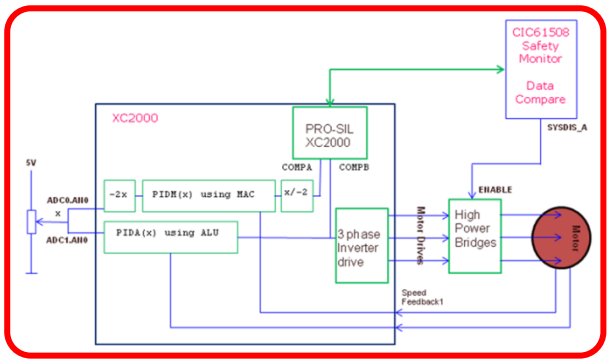
10ms Task ID = '2'

*1,1,1,1,1,1,1,1,1,1,**2**,1,1,1,1,1,1,1,1,1,1,**2**,1,1,1,1,1,1,1,1,1,1,.....*

Here, we must have 10 Task 1's between each Task2. To many or too few Task1's means the sequence fails and the CIC will shut down.

For the data compare, what kind of thing would qualify as a redundant calculation? Is running the exact same code at two different times sufficient? Running one calculation on the main CPU and one on the MAC? Running one in fixed and one in floating point? Using the DROPS concept?

An example of a diverse calculation would be to two similar routines, one using the ALU and one using the MAC. In addition, the input data for the redundant calculation could be multiplied by -2 (with the result being divided by -2) as shown in below. The two results would then be compared in the Data Comparator.



What happens when the TexM occurrence counter over flows? Does it just wrap round? Why is the occurrence counter needed anyway?

It rolls over. The TexM checks that each test occurs the expected number of times per 6ms. Each time a test runs, the occurrence counter increments. If a cyclic test fails to run a 6ms slot, the occurrence counter for that test will be zero so the error counter increase and a timeout error is reported.

How does the data compare and the task monitor differ in the TexM compared to the CIC; does the TexM just check that the requests are sent to the CIC?

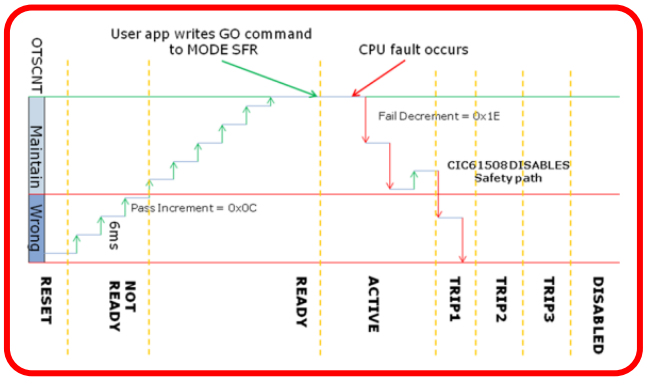
Yes, here is a manual extract:

> *1.1 Configuration*
>
> *The user can configure PRO-SIL XC2000 to monitor errors in the use of the Task monitor subsystem by the user application. The errors that can be detected are:*
>
> *(i) Invalid task ID i.e the task ID has a value > 254.*
>
> *This allows malfunctions of the application in the scheduling of task to be detected early. It should be noted that the CIC employs similar checks for increased security, especially with regard to SPI transmission errors.*

Also, why was ESR0 chosen as the enable signal rather than just a GPIO?
ESR0 can generate an output when the reset pin goes low.

Are there examples of what is needed to be done for ASIL-B or -D etc?
The PRO-SIL XC2000 demo app shows most of what needs to be done. We will add more complex examples when the revB board becomes available. This will interface to the BLDC motor controller.

Is the maintain value fixed at >= 0x40 - this is not configurable?
Yes it's fixed – see below.



Can the OTSCNT error counter start value be configured (so that the system could get to maintain more quickly)?
The OTSCNT etc. can be preloaded by the XC2000 theoretically if you modified the PowerOnHook to write to the counter registers late in the PowerOnHook.

Does the SafeTcore-II send a test SPI message during the startup?
No.

In the startup hook, the PLL is configured - does it matter if this is also configured elsewhere (Startup or AutoSAR etc)?
It could be elsewhere but it is best to let SafeTcore-II do it.

In the startup hook, there is a RAM test, but only for 0x80 words, why isn't all RAM tested at startup?
It would take too long. The original idea was that the startup hook would only test those RAM areas used for PRO-SIL XC2000 purposes. It would be up to the user to test the rest.

How does the SBST know which test to run?

The CIC61508 sets questions for the XC2300 to answer. The question number is held in the CIC61508 SEQ SFR. The read of the SEQ for the opcode test is in CicH.c:

```
STATIC void SCII_CODE_SPACE CicH_StartOpcCheck(void) {
    /* Send request to get Opcode test Id from CIC61508 */
    (void)CicH_GetSfr(SCII_SMSFR_OTS_SEQ, CicH_ServiceOpc);

    /* Send request to get system error state from CIC61508 */
    (void)CicH_GetSfr(SCII_SMSFR_ESTM_SYS, UpdateCicErrorState);

}   /* CicH_StartOpcCheck */
```

This is called from the SIL_Scheduler which is called from the 600us T6 int. The read of the SEQ takes place early in the 6ms system period to give enough time for all 14 slices of the SBST to run.

What is the easiest way to get started?

SafeTkits are available for both the TriCore (TC1767 and TC1782) and the XC2365A. These kits include an evaluation version of the development tools and a board. Also included are the SafeTcore library, demonstration application and the documentation. To find out more click here. Alternatively, contact Dave Greenhill or Richard Gledhill directly.