Introduction

A new topic is arising in the automotive world: how to avoid protection breaches and prevent security violations by eavesdroppers. Undetected tampering is one of these security threats. Tampering means improper alteration of sensible data inside the car ecosystem, for example unauthorized firmware modification, illegal calibration data adjustment, illicit odometer data gaining and so forth.

Undetected tampering can cause loss of income for vehicle makers (when a new firmware is installed without authorization) till even serious injuries (e.g. a software virus may disable the ABS or block the throttle device, the steering and so on).

The SPC57xx embeds different layers to protect from these security menaces.

The aim of this application note is to describe one of these layers to properly protect the flash from undetected tampering by using the "Tamper Detection Module", i.e. TDM.

The TDM module is used to associate an erase operation with a signature written into a "diary". Every time the flash is erased, a signature has to be saved in the log. The signature is user-dependent.

TDM is mainly configured by some writing DCF records in the UTest flash memory.

In order to speed up the usage of this module, a reference code has been used as example.
Contents

1 Overview ................................................................................................................. 5
  1.1 Diary ..................................................................................................................... 5
  1.2 DCF client ............................................................................................................ 6
    1.2.1 DCF client structure ....................................................................................... 7
  1.3 How to link Flash sector to a diary ................................................................. 10
  1.4 Lock/Unlock strategy via TDM ........................................................................ 12
  1.5 Life cycle configuration ...................................................................................... 13
  1.6 PASS module and life cycle .............................................................................. 14
  1.7 Override ............................................................................................................... 20

2 Summary .................................................................................................................. 21

Appendix A Reference code ....................................................................................... 22
  A.1 DCF_Utest_prog1.cmm ....................................................................................... 22
  A.2 K2_project_ram_vle.gpj ..................................................................................... 23

Appendix B Acronyms ................................................................................................. 24

Revision history .......................................................................................................... 25
List of tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>DCF of TDM in UTest Flash memory</td>
<td>7</td>
</tr>
<tr>
<td>Table 2</td>
<td>Acronyms</td>
<td>24</td>
</tr>
<tr>
<td>Table 3</td>
<td>Document revision history</td>
<td>25</td>
</tr>
</tbody>
</table>
## List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Diary position in Flash Memory</td>
<td>6</td>
</tr>
<tr>
<td>Figure 2</td>
<td>DCF structure</td>
<td>7</td>
</tr>
<tr>
<td>Figure 3</td>
<td>TDM DCF records involved in the application code</td>
<td>10</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Link between Flash memory sectors and diary (TDR Number=6)</td>
<td>11</td>
</tr>
<tr>
<td>Figure 5</td>
<td>TDRSR status flag Vs Diary and TDRxLOCKy DCF</td>
<td>11</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Lock/Unlock strategy via Diary writing</td>
<td>12</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Erase Flash Block strategy</td>
<td>13</td>
</tr>
<tr>
<td>Figure 8</td>
<td>SPC57xx life cycle progression</td>
<td>14</td>
</tr>
<tr>
<td>Figure 9</td>
<td>DCF record to advance the life cycle from “customer delivery” to “OEM production”</td>
<td>14</td>
</tr>
<tr>
<td>Figure 10</td>
<td>All PASS’s passwords are set to 0x5555_5555 (part 1)</td>
<td>16</td>
</tr>
<tr>
<td>Figure 11</td>
<td>All PASS’s passwords are set to 0x5555_5555 (part 2)</td>
<td>17</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Flow to unlock flash programming/erasing via PASS module</td>
<td>18</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Application code to remove the PASS lock</td>
<td>19</td>
</tr>
</tbody>
</table>
1 Overview

The TDM provides a protection mechanism of Flash memory that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased.

TDM requires a record to be written to a specified Flash area before the erase operation can be executed. Collectively, the records are referred to as a “diary”.

The content of the diary is defined by the user. It may serve as a simple erase counter or it may contain more advanced details for example a journal to log erase details like who is erasing, checksum of the Flash content and so forth.

Up to 6 TDRs can be defined via DCF records.

1.1 Diary

The diary is a region of Flash memory where records\(^\text{a}\) of block erases for each TDR are stored. The device implements 6 TDRs. Each TDR consists of 256 × 64-bit records to store until 256 attempts of erasing of Flash blocks assigned to each TDR.

Erasing of any block within a TDR is not protected by the TDM. To avoid malicious erasing of the diary, the diary should reside in a Flash block that is assigned as OTP\(^\text{b}\).

This means that at least one block of Flash memory, which is allocated to the TDR, shall be configured as OTP. Once done, such assigning can’t be removed. Customer can choose any Flash memory sector to be assigned to the TDR, but one of the low 16KB blocks is a good solution in terms of size.

The format and size of these records are defined by the user. The only hardware constraints are:

- each diary has a maximum size of 2 KB, and
- the minimum size of any programming operation is 8 bytes.

With 6 TDR the maximum overall size of the diary is 12 KB, as shown in Figure 1.

---

\(^\text{a}\) Records contain user-dependent details about each erase.

\(^\text{b}\) User shall use the DCF record called OTP\_ENx to configure a Flash block as OTP.
1.2 DCF client

It is important to understand the role of DCF records and DCF clients and how they affect TDM functions.

DCF records contain configuration data processed during device boot. Those records are permanently stored in a dedicated block of on-chip Flash memory, i.e. UTest sector. TDM-related DCF records configure the following parameters:

- Establish a permanent diary base address
- Define Tamper Detect Regions (TDRs) to monitor on-chip Flash memory program/erase activity
- Permanently disable one or more Tamper Regions
- Configure Flash memory as One Time Programmable (OTP) on a per-block basis

A DCF client is an internal register within a module that corresponds to a specific type of DCF record. Although multiple instances of a DCF record type may exist, but if they are considered valid depends on the strategy associated with that record type. For example:

- if it is defined "write once" and multiple instances are created, only the first one will be processed, or
- if it is defined "write 1 only", bits can be changed only from 0 to 1.
1.2.1 DCF client structure

All the DCF records have a common structure. They appear as contiguous double-word (64-bit) entries programmed in a reserved area of OTP UTEST Flash memory beginning at 0x0040_0200.

The Data[0:31] holds the value to write, CS[0:14] represents the Chip Select assigned during chip definition, Address[16:2] holds the value of the DCF address, relevant only to the address decoding within that module. Parity is not implemented in DCF record written in UTEST, while Stop bit indicates the end of the DCF records list.

The Tamper Detection Module implements the following DCF records that are saved in the UTest Flash memory:

Table 1. DCF of TDM in UTest Flash memory

<table>
<thead>
<tr>
<th>DCF client description</th>
<th>Reset value</th>
<th>DCF client Special Strategy</th>
<th>Value to write in the Utest (example)</th>
<th>Chip Select</th>
<th>DCF Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diary Base Address</td>
<td>0xffff_ffff</td>
<td>Write Once + Write ‘0’ only</td>
<td>0xxxxx_00020_0000</td>
<td>0x0020_0000</td>
<td>0x0</td>
</tr>
<tr>
<td>Tamper Region Override</td>
<td>0x00</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xxxxx_00020_0004</td>
<td>0x0020_0000</td>
<td>0x4</td>
</tr>
<tr>
<td>OTP_EN0</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xxxxx_00020_0020</td>
<td>0x0020_0000</td>
<td>0x20</td>
</tr>
<tr>
<td>OTP_EN1</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xxxxx_00020_0024</td>
<td>0x0020_0000</td>
<td>0x24</td>
</tr>
<tr>
<td>OTP_EN2</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xxxxx_00020_0028</td>
<td>0x0020_0000</td>
<td>0x28</td>
</tr>
<tr>
<td>OTP_EN3</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xxxxx_00020_002C</td>
<td>0x0020_0000</td>
<td>0x2C</td>
</tr>
<tr>
<td>TDR0_LOCK0</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xxxxx_00020_0050</td>
<td>0x0020_0000</td>
<td>0x50</td>
</tr>
<tr>
<td>TDR0_LOCK1</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xxxxx_00020_0054</td>
<td>0x0020_0000</td>
<td>0x54</td>
</tr>
</tbody>
</table>
Table 1. DCF of TDM in UTest Flash memory (continued)

<table>
<thead>
<tr>
<th>DCF client description</th>
<th>Reset value</th>
<th>DCF client Special Strategy</th>
<th>Value to write in the Utest (example) = Value to write(1)</th>
<th>Chip Select</th>
<th>DCF Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDR0_LOCK2</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0058</td>
<td>0x0020_0000</td>
<td>0x58</td>
</tr>
<tr>
<td>TDR0_LOCK3</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_005C</td>
<td>0x0020_0000</td>
<td>0x5C</td>
</tr>
<tr>
<td>TDR1_LOCK0</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0060</td>
<td>0x0020_0000</td>
<td>0x60</td>
</tr>
<tr>
<td>TDR1_LOCK1</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0064</td>
<td>0x0020_0000</td>
<td>0x64</td>
</tr>
<tr>
<td>TDR1_LOCK2</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0068</td>
<td>0x0020_0000</td>
<td>0x68</td>
</tr>
<tr>
<td>TDR1_LOCK3</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_006C</td>
<td>0x0020_0000</td>
<td>0x6C</td>
</tr>
<tr>
<td>TDR2_LOCK0</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0070</td>
<td>0x0020_0000</td>
<td>0x70</td>
</tr>
<tr>
<td>TDR2_LOCK1</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0074</td>
<td>0x0020_0000</td>
<td>0x74</td>
</tr>
<tr>
<td>TDR2_LOCK2</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0078</td>
<td>0x0020_0000</td>
<td>0x78</td>
</tr>
<tr>
<td>TDR2_LOCK3</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_007C</td>
<td>0x0020_0000</td>
<td>0x7C</td>
</tr>
<tr>
<td>TDR3_LOCK0</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0080</td>
<td>0x0020_0000</td>
<td>0x80</td>
</tr>
<tr>
<td>TDR3_LOCK1</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0084</td>
<td>0x0020_0000</td>
<td>0x84</td>
</tr>
<tr>
<td>TDR3_LOCK2</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0088</td>
<td>0x0020_0000</td>
<td>0x88</td>
</tr>
<tr>
<td>TDR3_LOCK3</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_008C</td>
<td>0x0020_0000</td>
<td>0x8C</td>
</tr>
<tr>
<td>TDR4_LOCK0</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0090</td>
<td>0x0020_0000</td>
<td>0x90</td>
</tr>
<tr>
<td>TDR4_LOCK1</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0094</td>
<td>0x0020_0000</td>
<td>0x94</td>
</tr>
<tr>
<td>TDR4_LOCK2</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_0098</td>
<td>0x0020_0000</td>
<td>0x98</td>
</tr>
<tr>
<td>TDR4_LOCK3</td>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_009C</td>
<td>0x0020_0000</td>
<td>0x9C</td>
</tr>
</tbody>
</table>
Hereafter the list of DCF records associated to the TDM, which are saved by the example Figure 10: All PASS’s passwords are set to 0x5555_5555 (part 1) and Figure 11: All PASS’s passwords are set to 0x5555_5555 (part 2) into the UTest:

- **SET_DCF_DBA** = (0x00FC_4000_0020_0000) in order to choose the Low Block 2 of Flash memory
- **SET_DCF_OTPEN0** = (0x0004_0000_0020_0020) in order to set OTP the Low Block 2
- **SET_DCF_TDR0_LOCK0** = (0x0008_0000_0020_0050) in order to assign the Low Block 3 to the diary
- **SET_DCF_TR0** = (0x0000_003F_0020_0004) in order to override all the TDR Locks DCF.

<table>
<thead>
<tr>
<th>TDR5 LOCK0</th>
<th>Description</th>
<th>Value to write in the Utest (example)</th>
<th>Chip Select</th>
<th>DCF Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000_0000</td>
<td>Write Once + Write ‘1’ only</td>
<td>0xXXXX_XXXX_0020_00A0</td>
<td>0x0020_0000</td>
<td>0xA0</td>
</tr>
</tbody>
</table>

**Table 1. DCF of TDM in UTest Flash memory (continued)**

1. Application depended. Here is left with 0xXXXX_XXXX.
1.3 How to link Flash sector to a diary

The mechanism to link a specific Flash memory sector to the diary is based on the TDRx_LOCKy DCF clients. Those DCF clients assign a Flash block to a specific Diary. See the Device-Specific Chapter” of the RM for the mapping between field bits of the TDRx_LOCKy DCF clients and Flash blocks. The same mapping also applies to similar functions in the embedded Flash memory module, for example password protection via PASS module.
Warning: No Flash block should be assigned to more than one TDR. Each Flash block should correspond to a single TDR. For example, if a single Flash block is assigned to TDR0 and TDR1, it will never be unlocked for erase operation even if the Diary is program.

TDM provides a read-only register to read the status of the Flash blocks linked to a specific TDR. As shown in Figure 5, this register reflects the status flag associated to the TDRx_LOCKy DCF to lock/unlock the chosen Flash Block.
Warning: EEPROM memory cannot be protected by TDM module.

1.4 Lock/Unlock strategy via TDM

Once the user has
1. programmed the right DCFs in order to set the Diary Base Address (DBA),
2. set the relative bit for TDRx_LOCKy to choose the Flash sector to be protected, and
3. trigger a reset specified Flash sectors are protected from tampering.

The only way to erase[c] this Flash sector is to write a double word in the TDR as described in Figure 6.

Figure 6. Lock/Unlock strategy via Diary writing

The Erase Flash Block strategy shows a high level schema of signals involved in TDM protection, which have to be set in order to unlock the erase Flash block protection. For sake of simplicity, OTP flags, lock password protection and region R/W protection from PASS module have not been considered.

c. TDM monitors erase operation only. User can over-program a Flash sector which is locked by the TDM (in this case, user shall take care of the potential injection of ECC error in the Flash array). Other security layers are available, e.g. PASS module, to protect from over-programming.
Figure 7. Erase Flash Block strategy

Warning: For SPC574K72xx device, user shall set the Program/Erase Complete Interrupt Enable (PECIE) bit in the Flash Module Configuration Register (MCR) before doing any Flash block erase operation which includes TDM diary update operation. The PECIE interrupt does not need to be processed by the Interrupt controller. To prevent the interrupt from being processed its priority should be left at the default of 0 within the Interrupt controller. See Errata ERR008089.

1.5 Life cycle configuration

Five states, called Life Cycles, are defined in SPC57xx devices. These working states represent a device maturity model (see Figure 8). The behavior of the device depends on its Life Cycle. The list of life cycles is:

- ST Production
  No security implemented
- Customer delivery\(^d\)
  Very limited security to guarantee the ECU development
- OEM production
  Limited security to guarantee the ECU development
- In Field
  Full security
- Failure analysis
  Limited security to guarantee the protection of critical data but also the testability. The unit can’t be run In Field any more.

\(^d\) Samples are shipped by ST with Life Cycle set as Customer Delivery.
Section "Life cycle truth table" of the Security Manual gives all needed details on security and lifecycle.

To advance the life cycle to OEM, the user shall write in the UTest sector starting from the address 0x0040_0208, four double words equal to 0x55AA_50AF_55AA_50AF (refer to section Life Cycle in SSCM chapter of the reference manual to have all additional details).

In the following example a debugger script has been used to program the UTest Flash in order to move the Life Cycle from Customer delivery to OEM production.

**Figure 9. DCF record to advance the life cycle from “customer delivery” to “OEM production”**

```plaintext
&UTEST_DCF_LIFE_CYCLE=0x00400208
&current_address=&UTEST_DCF_LIFE_CYCLE

:1 - UTEST_DCF_LIFE_CYCLE 55AA_50AF_55AA_50AF to pass to Customer delivery
  &current_address=&current_address
  GOSUB program_word &current_address 0x55AA50AF55AA50AF

:2 - UTEST_DCF_LIFE_CYCLE 55AA_50AF_55AA_50AF
  &current_address=&current_address+0x8
  GOSUB program_word &current_address 0x55AA50AF55AA50AF

:3 - UTEST_DCF_LIFE_CYCLE 55AA_50AF_55AA_50AF to pass to OEM production
  &current_address=&current_address+0x8
  GOSUB program_word &current_address 0x55AA50AF55AA50AF

:4 - UTEST_DCF_LIFE_CYCLE 55AA_50AF_55AA_50AF
  &current_address=&current_address+0x8
  GOSUB program_word &current_address 0x55AA50AF55AA50AF
```

**Warning:** For example the PASS module is activated in OEM production. This feature has a strong impact on the security. More details in the next section.

*Figure 9* shows an example of DCF records on which JTAG passwords and all PASS passwords are set to 0x5555_5555.

### 1.6 PASS module and life cycle

PASS module is automatically activated starting from the OEM live cycle. It provides password comparison to protect access via JTAG and program/erase operations on Flash sectors.
User shall define 5 passwords of 256 bit. If the user doesn't provide these passwords, PASS module can gate debug and Flash operations. These passwords are configured via DCF records. It's very important to set them before, or concurrently, moving the lifecycle to “OEM production”.

Have a look at the section “Password and Device Security Module (PASS)” of the reference manual to have all needed details on using the PASS module.

---

**Warning:** If the device is moved to the OEM lifecycle without knowing the PASS passwords, the specific samples can't be accessed anymore and Flash can't be programmed/erased.

---

Figure 9 shows an example of DCF records on which JTAG passwords and all PASS passwords are set to 0x5555_5555.
Figure 10. All PASS's passwords are set to 0x5555_5555 (part 1)

```
&UTEST_DCF_JTAG_PASS=0x00400120
&current_address=&UTEST_DCF_JTAG_PASS

IF &proInfoflash
{

;Lock0  - >TSLock enable UTEST memory
PER.S ANC:0xFFE0010 %LONG 0x3FFFFFF

;1 - SET_DCF_JTAG_PASS 0x55555555  0x55555555
GOSUB program_word &current_address 0x5555555555555555

;2 - SET_DCF_JTAG_PASS 0x55555555  0x55555555
&current_address=&current_address+0x8
GOSUB program_word &current_address 0x5555555555555555

;3 - SET_DCF_JTAG_PASS 0x55555555  0x55555555
&current_address=&current_address+0x8
GOSUB program_word &current_address 0x5555555555555555

;4 - SET_DCF_JTAG_PASS 0x55555555  0x55555555
&current_address=&current_address+0x8
GOSUB program_word &current_address 0x5555555555555555

;1 - SET_DCF_PASS_PASSWORD 0x55555555  0x55555555
&current_address=&current_address+0x8
GOSUB program_word &current_address 0x5555555555555555

;2 - SET_DCF_PASS_PASSWORD 0x55555555  0x55555555
&current_address=&current_address+0x8
GOSUB program_word &current_address 0x5555555555555555

;3 - SET_DCF_PASS_PASSWORD 0x55555555  0x55555555
&current_address=&current_address+0x8
GOSUB program_word &current_address 0x5555555555555555

;4 - SET_DCF_PASS_PASSWORD 0x55555555  0x55555555
&current_address=&current_address+0x8
GOSUB program_word &current_address 0x5555555555555555

;1 - SET_DCF_PASS_PASSWORD 0x55555555  0x55555555
&current_address=&current_address+0x8
GOSUB program_word &current_address 0x5555555555555555

;2 - SET_DCF_PASS_PASSWORD 0x55555555  0x55555555
&current_address=&current_address+0x8
GOSUB program_word &current_address 0x5555555555555555

;3 - SET_DCF_PASS_PASSWORD 0x55555555  0x55555555
&current_address=&current_address+0x8
GOSUB program_word &current_address 0x5555555555555555

;4 - SET_DCF_PASS_PASSWORD 0x55555555  0x55555555
&current_address=&current_address+0x8
GOSUB program_word &current_address 0x5555555555555555
```
Once the PASS module is activated, the debugger shall provide the right passwords before erasing/programming the Flash.

*Figure 12* shows the flow to implement the procedure to unlock Flash operations. It shall be done for all 4 password groups defined in the PASS module.

*Figure 13* shows an extract of code used to remove the lock for all PASS groups.
Figure 12. Flow to unlock Flash programming/erasing via PASS module

1. Select the password group CHSEL register
2. Write the 256bit password on 8 32bit registers CIN0..CIN7
3. Write lock registers related to the selected group LOCK1_PGx..LOCK3_PGx

This is the password related to the group selected by the CHSEL register
Figure 13. Application code to remove the PASS lock

```c
void Unlock_Flash()
{
    uint8_t i;

    PASS.CHSEL.B.GRP = 0; /* Unlock password group #0 */
    for (i = 0; i < 8; i++)
    {
        PASS.CIN[i].B.PW32 = 0x55555555; /* set the right password (same in DCF) */
    }

    PASS.TIMER[0].LOCK0_PG.R = 0x0; /* Unlock LOCK0_PG0 */
    PASS.TIMER[0].LOCK1_PG.R = 0x0; /* Unlock LOCK0_PG1 */
    PASS.TIMER[0].LOCK2_PG.R = 0x0; /* Unlock LOCK0_PG2 */
    PASS.TIMER[0].LOCK3_PG.R = 0x0; /* Unlock LOCK0_PG3 */

    PASS.CHSEL.B.GRP = 1; /* Unlock password group #1 */
    for (i = 0; i < 8; i++)
    {
        PASS.CIN[i].B.PW32 = 0x55555555; /* set the right password (same in DCF) */
    }

    PASS.TIMER[1].LOCK0_PG.R = 0x0; /* Unlock LOCK0_PG0 */
    PASS.TIMER[1].LOCK1_PG.R = 0x0; /* Unlock LOCK0_PG1 */
    PASS.TIMER[1].LOCK2_PG.R = 0x0; /* Unlock LOCK0_PG2 */
    PASS.TIMER[1].LOCK3_PG.R = 0x0; /* Unlock LOCK0_PG3 */

    PASS.CHSEL.B.GRP = 2; /* Unlock password group #2 */
    for (i = 0; i < 8; i++)
    {
        PASS.CIN[i].B.PW32 = 0x55555555; /* set the right password (same in DCF) */
    }

    PASS.TIMER[2].LOCK0_PG.R = 0x0; /* Unlock LOCK0_PG0 */
    PASS.TIMER[2].LOCK1_PG.R = 0x0; /* Unlock LOCK0_PG1 */
    PASS.TIMER[2].LOCK2_PG.R = 0x0; /* Unlock LOCK0_PG2 */
    PASS.TIMER[2].LOCK3_PG.R = 0x0; /* Unlock LOCK0_PG3 */

    PASS.CHSEL.B.GRP = 3; /* Unlock password group #3 */
    for (i = 0; i < 8; i++)
    {
        PASS.CIN[i].B.PW32 = 0x55555555; /* set the right password (same in DCF) */
    }

    PASS.TIMER[3].LOCK0_PG.R = 0x0; /* Unlock LOCK0_PG0 */
    PASS.TIMER[3].LOCK1_PG.R = 0x0; /* Unlock LOCK0_PG1 */
    PASS.TIMER[3].LOCK2_PG.R = 0x0; /* Unlock LOCK0_PG2 */
    PASS.TIMER[3].LOCK3_PG.R = 0x0; /* Unlock LOCK0_PG3 */

    FLASH.LOCK0.R = 0x0; /* unlock flash */
    FLASH.LOCK1.R = 0x0; /* unlock flash */
    FLASH.LOCK2.R = 0x0; /* unlock flash */
}
```
1.7 Override

Once the diary is full the user can decide to override the TDM protection to allow the erase operations without limitation given by the TDM or may choose to stop further erase operations.

Diary override mechanism is implemented with a DCF client consisting of 6 bits, one for each TDR.

In order to override the TDM protection the TO (Tamper Override) DCF client has to be programmed by the relative bitfield with 1. Any attempt to write bits from 1 to 0 is ignored, this means that the override process is permanent.
2 Summary

This overview of the TDM describes one of the security layers embedded in the SPC57xx family microcontroller. The TDM can disable the erase operation of Flash memory sectors until a “digital sign” is written in the “diary”.

Content of the diary is user-dependent, from a simple counter of erasing operation to an erase log.

This application note summarizes the key points to configure the TDM to start automatically after the reset.

To speed up the implementation of such configuration, a reference code is provided. Reference code is described on Section Appendix A.
Appendix A  Reference code

Reference code(e) associated to this document shows how to configure TDM automatically via DCF. If the device is configured correctly, once that debugger script is loaded, the following scenario is set:

- PASS password, JTAG passwords are set to 0x5555_5555 (see Figure 9: DCF record to advance the life cycle from “customer delivery” to “OEM production”)
- Device is uncensored writing in the censorship DCF record 0x0000_55AA_0010_00B0
- Diary Base Address (TDM_DBA) is set at the location 0x00FC4000, (Flash Low Block2)
- Flash Low Block2 is set as OTP, writing in the DCF record 0x0004_0000_0020_0020
- Flash Low Block3 (0x00FC8000) is assigned to the Diary, writing in the DCF record TDR0_Lock0 0x0008_0000_0020_0050
- Life cycle is moved from Customer delivery to OEM production

Code, which has been tested on K2 cut 2 using Lauterbach Trace32 as debugger and Multi GHS 6.1.4 as compiler/linker, can be adapted to be used with other development tool chains.

The TDM is enabled from Customer delivery Life Cycle.

The release contains 1 GHS projects:

- “DCF_Utest_prog1.cmm”
  - Debugger script to configure the UTest Flash memory
- "K2_project_ram_vle.gpj “
  - project to test the TDM module

A.1  DCF_Utest_prog1.cmm

This script is used to configure the UTEST:

- to move the lifecycle to OEM
- to set the relevant DCFs to configure the TDM after the reset
- to configure the PASS passwords to unlock the Flash
- to set the device uncensored

Pay attention that in OEM lifecycle, the PASS protects the Flash program/erase. Known passwords shall be saved into the UTEST.

e. Content on this appendix can be also found in the file README.txt included in the reference code
A.2  K2_project_ram_vle.gpj

Following steps shall be used to execute the reference code:
1. Run the debugger script in order to configure the UTest
2. Load the executable code into RAM
# Acronyms

## Appendix B  Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDM</td>
<td>Tamper Detection Module</td>
</tr>
<tr>
<td>DCF</td>
<td>Device Configuration Format Records</td>
</tr>
<tr>
<td>UTest</td>
<td>User Test Flash block</td>
</tr>
<tr>
<td>OTP</td>
<td>One Time Programming</td>
</tr>
<tr>
<td>TDR</td>
<td>Tamper Detection Region</td>
</tr>
<tr>
<td>DBA</td>
<td>Diary Base Address</td>
</tr>
<tr>
<td>SSCM</td>
<td>System Status and Configuration Module</td>
</tr>
<tr>
<td>TO</td>
<td>Tamper Override</td>
</tr>
<tr>
<td>RM</td>
<td>Reference Manual</td>
</tr>
</tbody>
</table>
Table 3. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>02-Oct-2014</td>
<td>1</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>
IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2014 STMicroelectronics – All rights reserved