SPC56APx, SPC560P54, SPC560P60 Errata sheet

SPC56APx, SPC560P54, SPC560P60 device errata JTAG_ID = 0x0AE2_4041

Introduction

This errata sheet describes all the functional and electrical problems of the SPC56APx and SPC560P54/60 devices, identified with the JTAG_ID = 0x0AE2_4041.

All the topics covered in this document refer to *RM0083* rev 2 and *SPC560P54x, SPC560P60x, SPC56AP54x, SPC56AP60x datasheet* rev 1 (see *Appendix A: Further information*).

Device identification for cut 1.0:

- JTAG_ID = 0x0AE2_4041
- MIDR1 register:
  - MAJOR_MASK[3:0]: 4’b0000
  - MINOR_MASK[3:0]: 4’b0000

Package device marking mask identifier: AA

Die mask ID: FP60X1

This errata sheet applies to SPC56APx and SPC560P54/60 devices in accordance with *Table 1*.

Table 1. Device summary

Part number	Package
SPC560P54L5	LQFP144
SPC56AP60L5	
SPC560P54L3	LQFP100
SPC56AP54L3	
SPC560P60L3	
SPC56AP60L3	

Contents

1 Functional problems

1.1 ERR000817: JTAGC: EVTI and RDY require TCK to toggle

Description:

The Nexus/JTAG read/write access control/status register (RWCS) write (to begin a read access) or the write to the read/write access data register (RWD)(to begin a write access) does not actually begin its action until 1 JTAG clock (TCK) after leaving the JTAG Update-DR state. This prevents the access from being performed and therefore do not signal its completion via the READY (RDY) output unless the JTAG controller receives an additional TCK. In addition, EVTI is not latched into the device unless there are clock transitions on TCK.

Workaround:

The tool/debugger must provide at least one TCK clock for the EVTI signal to be recognized by the MCU. When using the RDY signal to indicate the end of a Nexus read/write access, ensure that TCK continues to run for at least 1 TCK after leaving the Update-DR state. This can be just a TCK with TMS low while in the Run-Test/Idle state or by continuing with the next Nexus/JTAG command. Expect the affect of EVTI and RDY to be delayed by edges of TCK.

Note: RDY is not available in all packages of all devices.

1.2 ERR001388: FlexRay: Incomplete transmission of message frame in key slot

Description:

The FlexRay module will transmit an incomplete message in the key slot under the following circumstances:

1. The transmit message buffer n assigned to the key slot is located in the message buffer segment 2, that is, FR_MBSSUTR[MB_LAST_SEG1] < n.
2. The data size of the message buffer segment 1 is smaller than the static payload length, that is, FR_MBDSR[MBSEG1DS] < PCR19[payload_length_static].

In this case, the FlexRay module will transmit only FR_MBDSR[MBSEG1DS] payload words from message buffer n. The remaining words are padded with 0’s.

Workaround:

The transmit message buffer assigned to key slot must be located in message buffer segment 1.

1.3 ERR002302: FlexRay: Message buffer can not be disabled and not locked after CHI command FREEZE

Description:

If a complete message was transmitted from a transmit message buffer or received into a message buffer and the controller host interface (CHI) command FREEZE is issued by the

application before the end of the current slot, then this message buffer can not be disabled and locked until the module has entered the protocol state normal active.

Consequently, this message buffer can not be disabled and locked by the application in the protocol config state, which prevents the application from clearing the commit bit CMT and the module from clearing the status bits. The configuration bits in the message buffer configuration, control, status registers (MBCCSRn) and the message buffer configuration registers MBCCFRn, MBFIDRn, and MBIDXRn are not affected.

At most one message buffer per channel is affected.

Workaround:

There are two types of workaround.

1. The application should not send the CHI command FREEZE and use the CHI command HALT instead.
2. Before sending the CHI command FREEZE the application should repeatedly try to disable all message buffers until all message buffers are disabled. This maximum duration of this task is three static or three dynamic slots.

1.4 ERR002360: FlexCAN: Global masks misalignment

Description:

Convention: MSB = 0.

During CAN messages reception by FlexCAN, the RXGMASK (Rx Global Mask) is used as acceptance mask for most of the Rx message buffers (MB). When the FIFO Enable bit in the FlexCAN module configuration register (CANx_MCR[FEN], bit 2) is set, the RXGMASK also applies to most of the elements of the ID filter table. However there is a misalignment between the position of the ID field in the Rx MB and in RXIDA, RXIDB and RXIDC fields of the ID Tables. In fact RXIDA filter in the ID Tables is shifted one bit to the left from Rx MBs ID position as shown below:

- Rx MB ID = bits 3–31 of ID word corresponding to message ID bits 0–28
- RXIDA = bits 2–30 of ID Table corresponding to message ID bits 0–28

Note that the mask bits one-to-one correspondence occurs with the filters bits, not with the incoming message ID bits. This leads the RXGMASK to affect Rx MB and Rx FIFO filtering in different ways.

For example, if the user intends to mask out the bit 24 of the ID filter of message buffers then the RXGMASK is configured as 0xffff_ffef. As result, bit 24 of the ID field of the incoming message is ignored during filtering process for message buffers. This very same configuration of RXGMASK would lead bit 24 of RXIDA to be “don't care” and thus bit 25 of the ID field of the incoming message would be ignored during filtering process for Rx FIFO.

Similarly, both RXIDB and RXIDC filters have multiple misalignments with regards to position of ID field in Rx MBs, which can lead to erroneous masking during filtering process for either Rx FIFO or MBs.

RX14MASK (Rx 14 Mask) and RX15MASK (Rx 15 Mask) have the same structure as the RXGMASK. This includes the misalignment problem between the position of the ID field in the Rx MBs and in RXIDA, RXIDB and RXIDC fields of the ID Tables.

Workaround:

Therefore it is recommended that one of the following actions be taken to avoid problems:

- Do not enable the RxFIFO. If CANx_MCR[FEN] = 0 then the Rx FIFO is disabled and thus the masks RXGMASK, RX14MASK and RX15MASK do not affect it.
- Enable Rx individual mask registers. If the backwards compatibility configuration bit in the FlexCAN module configuration register (CANx_MCR[BCC], bit 15) is set then the Rx individual mask registers (RXIMR[0:63]) are enabled and thus the masks RXGMASK, RX14MASK and RX15MASK are not used.
- Do not use masks RXGMASK, RX14MASK and RX15MASK (that is, let them in reset value which is 0xffff_ffff) when CANx_MCR[FEN] =1 and CANx_MCR[BCC] = 0. In this case, filtering processes for both Rx MBs and Rx FIFO are not affected by those masks.
- Do not configure any MB as Rx (that is, let all MBs as either Tx or inactive) when CANx_MCR[FEN] = 1 and CANx_MCR[BCC] = 0. In this case, the masks RXGMASK, RX14MASK and RX15MASK can be used to affect ID tables without affecting filtering process for Rx MBs.

1.5 ERR002421: FLEXRAY : Message buffer can not be disabled in POC state INTEGRATION_LISTEN

Description:

If the communication controller is started as a non-coldstart node and configured and enabled message buffers in the POS config state for slot 1, then the message buffer can not be disabled in the INTEGRATION_LISTEN state, which is entered when no communication can be established.

Workaround:

A Software work-around is available, which is as follows: Run a freeze command just before running the message buffer disable for slot 1. This should enable the message buffer disable during the Listen States.

1.6 ERR002423: FlexRay: Transmission in a slot n of channel A in dynamic segment may be corrupted or duplicated on both the channels

Description:

If a transmit message buffer is assigned to a slot n in the dynamic segment and assigned to both channels A and B and if the slot ID n in the dynamic segment does not coincide for both channels, the FlexRay module transmits the message frame on both channels A and B, or may transmit a corrupted frame on channel A.

Workaround:

Assign message buffers in the dynamic segment to one channel only.

Note: The FlexRay specification (revision 2.1a) states that in the dynamic segment, transmit buffers should only be assigned channel A or channel B, and not to both.

1.7 ERR002656: FlexCAN: Abort request blocks the CODE field

Description:

An Abort request to a transmit message buffer (TxMB) can block any write operation into its CODE field. Therefore, the TxMB cannot be aborted or deactivated until it completes a valid transmission (by winning the CAN bus arbitration and transmitting the contents of the TxMB).

Workaround:

Instead of aborting the transmission, use deactivation instead.

Note: Note that there is a chance the the deactivated TxMB can be transmitted without setting IFLAG and updating the CODE field if it is deactivated.

1.8 ERR003022: SWT: Watchdog is disabled during BAM execution

Description:

The watchdog is disabled at the start of BAM execution. In the case of an unexpected issue during BAM execution the CPU may be stalled and it will be necessary to generate an external reset to recover.

Workaround:

No workaround

1.9 ERR003165: BAM: Code download via FlexCAN not functioning in a CAN network

Description:

When the serial download via FlexCAN is selected setting the FAB (force alternate boot) pin, and ABS (alternate boot selector) pins (ABS0 = 1 and ABS1 = 0) and the micro is part of a CAN network, the serial download protocol may unexpectedly stop in case of CAN traffic. After the code has been downloaded, the BAM tries to disable the FLexCAN module writing the MCR (module configuration register) without waiting for the acknowledge bit LPM_ACK (low power mode acknowledge) to be set. The FlexCAN cannot enter the low power mode until all current transmissions or receptions have finished, further writings into any FlexCAN register may cause the low power mode not to be entered and, as consequence, the BAM to stop.

Workaround:

Since the higher the traffic, the higher the chance for the BAM to try to disable the FlexCAN module during a CAN frame reception, make sure that no other CAN frame is sent until the code download protocol has been completed.

1.10 ERR003204: LINFlex: LDIV lower than 1.5 value are not valid when in UART mode.

Description:

Maximum baud rate is Fsys / 24 = Fsys/(16 × LDIV) with LDIV = LINIBRR + LINFBRR / 16.

Configuration with LINIBRR = 1 and LINFBRR < 8 are invalid when UART mode is selected.

Workaround:

No workaround

1.11 ERR003262: Register protection on full CMU_CSR

Description:

The register protection on CMU_CSR of CMU0 works only on the full 32 bit, while it should protect only the bits 24–31. As a consequence, when register protection is active on CMU_CSR the frequency meter cannot be used anymore.

Workaround:

In order to perform a frequency meter operation, the register protection of the relevant CMU must be disabled first; this workaround would work only when soft lock is active.

1.12 ERR003263: Serial boot and censorship: Flash read access

Description:

In a secured device, starting with a serial boot, it is possible to read the content of the four Flash locations where the RCHW is stored. For example if the RCHW is stored at address 0x00000000, the reads at address 0x00000000, 0x00000004, 0x00000008 and 0x0000000C will return a correct value. Any other Flash address is not readable.

Workaround:

No workaround

1.13 ERR003264: MCM: MRSR does not report power on reset event

Description:

The flag MRSR[POR] stays low after power on reset event on the device.

Workaround:

Do not use MRSR[POR] to determine power on reset cause. Use RGM_DES instead.

1.14 ERR003269: MC_ME: Peripheral clocks may get incorrectly disabled or enabled after entering debug mode

Description:

If ME_RUN_PCx, ME_LP_PCx, ME_PCTLx registers are changed to enable or disable a peripheral, and the device enters debug mode before a subsequent mode transition, the peripheral clock gets enabled or disabled according to the new configuration programmed. Also ME_PSx registers will report incorrect status as the peripheral clock status is not expected to change on debug mode entry.

Workaround:

After modifying any of the ME_RUN_PCx, ME_LP_PCx, ME_PCTLx registers, request a mode change and wait for the mode change to be completed before entering debug mode in order to have consistent behaviour on peripheral clock control process and clock status reporting in the ME_PSx registers.

1.15 ERR003324: FIRC - FIRC_CTL[TRIM] does not display correct trim value after reset

Description:

The FIRC is trimmed during reset using a factory programmed value stored in Flash. However after reset the trim value is not copied to FIRCRC_CTL[TRIM] as one would expect. Any read of FIRCRC_CTL[TRIM] reads 0 and in all likelihood this is not the factory programmed value. Therefore any read-modify-write on the 32-bit register set the FIRC to a trim value of 0 and not the factory programmed value.

Workaround:

As the lower 16-bits of FIRC_CTL register only contain the TRIM field it is recommended that if the user wishes to program any other field they should only access the upper 16-bits of this register.

If the user wishes to calibrate the FIRC this should be performed using the CMU.

1.16 ERR003407: FlexCAN: CAN transmitter stall in case of no remote frame in response to Tx packet with RTR = 1

Description:

FlexCAN does not transmit an expected message when the same node detects an incoming Remote Request message asking for any remote answer.

The issue happens when two specific conditions occur:

1. The Message Buffer (MB) configured for remote answer (with code “a”) is the last MB. The last MB is specified by Maximum MB field in the Module Configuration Register (MCR[MAXMB]).
2. The incoming Remote Request message does not match its ID against the last MB ID.

While an incoming Remote Request message is being received, the FlexCAN also scans the transmit (Tx) MBs to select the one with the higher priority for the next bus arbitration. It is expected that by the Intermission field it ends up with a selected candidate (winner). The

coincidence of conditions (1) and (2) above creates an internal corner case that cancels the Tx winner and therefore no message will be selected for transmission in the next frame. This gives the appearance that the FlexCAN transmitter is stalled or “stops transmitting”.

The problem can be detectable only if the message traffic ceases and the CAN bus enters into Idle state after the described sequence of events.

There is NO ISSUE if any of the conditions below holds:

1. The incoming message matches the remote answer MB with code “a”.
2. The MB configured as remote answer with code “a” is not the last one.
3. Any MB (despite of being Tx or Rx) is reconfigured (by writing its CS field) just after the Intermission field.
4. A new incoming message sent by any external node starts just after the Intermission field.

Workaround:

Do not configure the last MB as a Remote Answer (with code “a”).

1.17 ERR003452: FLASH: Programming just after reading a location of data Flash with double ECC error can trigger a functional reset.

Description:

If a double-bit ECC error is encountered when reading the data Flash, a functional reset will occur if the next data Flash access is a program (irrespective of the length of time between the data read causing the ECC error and the program attempt). If another location in the data Flash is read (which does not generate an ECC error) before attempting to program the data Flash, the reset does not occur. This only impacts programming operations - an erase after a double-bit ECC error will not generate a reset.

Workaround:

If a double-bit ECC error is encountered in the data Flash, the application software must read another byte in the data Flash without an ECC error prior to performing a program operation on the data Flash. There is a read only data Flash test Flash block at address 0x00C0_2000 and the recommendation is to read data from this location to provide the “non ECC” data read. This procedure could be automated by adding the test block read to the exception handler for a data Flash ECC event.

1.18 ERR003547: FLASH: Running ECC logic check just after reading a location of data Flash with double ECC error can trigger a functional reset.

Description:

If a double-bit ECC error is encountered when reading the data Flash, a functional reset will occur if an ECC logic check is then carried out on the data Flash at any point before the next reset.

Workaround:

Ensure that there have been no ECC double bit errors encountered in the data Flash before carrying out an ECC logic check.

1.19 ERR003583: MC_RGM: A non-monotonic ramp on the VDD_HV_REG supply can cause the RGM module to clear all flags in the DES register.

Description:

At system power-up a non-monotonic voltage ramp-up or a very slow voltage ramp-up (also known as 'soft start-up') can cause incorrect flag setting in the RGM_DES register. During monotonic power-up, F_POR flag is set when the high voltage regulator supply (VDD_HV_REG) goes above LVD27_VREG low voltage detector threshold and the 1.2V supply (VDD_LV_REGCOR) goes above LVD12_PD0 low voltage detector threshold. Expected behavior POR = 1 , LVD27 = 0, LVD12 = 0

During a non-monotonic power-up the VDD_HV_REG may show a non-linearity in the ramp up. When the VDD_HV_REG supply dips below LVD27_VREG threshold, LVD27_VREG low voltage detector is re-fired. If VDD_LV_REGCOR is already above LVD12_PD0 low voltage detector threshold, F_POR flag is reset and F_LVD27_VREG is set. Expected behavior POR = 0 , LVD27 = 1, LVD12 = 0

This errata reports behavior when the non-linearity on VDD_HV_REG coincides with the ramp-up of VDD_LV_REGCOR completion and LVD27_VREG is re-fired just after the LVD12_PD0 is released. In this case, neither F_POR flag nor F_LVD27_VREG flag will be set. In this case, application code cannot use the flags to tell if a power-on reset has occurred.

This errata only affects the flag circuit and not a the device initialization. Device initializes correctly under all conditions.

Workaround:

Hardware Workaround: Ensure that non-linearity on VDD_HV_REG is < 100 mV . This is the hysteresis limit of the device. Board regulator should be chosen accordingly.

Software Workaround: The software workaround need only be applied when neither the F_POR, LVD27 or LVD12 flags are set and involves checking SRAM contents and monitoring for ECC errors during this process. In all cases, an ECC error is assumed to signify a power-on reset (POR). Two suggestions are made for software workarounds. In both cases, if POR is detected all RAM should be initialized otherwise no power-on condition is detected and it is possible to initialize only needed parts of RAM while preserving required information.

Software workaround #1: An area of RAM can be reserved by the compiler into which a KEY, such as 0x3EC1_9678, is written. This area can be checked at boot and if the KEY is incorrect or an ECC error occurs, POR can be assumed and the KEY should be set. Use of a KEY increases detection rate to 31 bits(≤10e-9) or 23bits (≤5.10e-6) instead of 7-bit linked to ECC (≤10e-2)

Software workaround #2: When runtime data should be retained and RAM only re-initialized in the case of POR, the CRC module should be used to calculate and store a CRC signature when writing data that can be checked at boot time. If CRC signature is incorrect, POR can be assumed.

1.20 ERR003584: MC_ME: Possibility of machine check on low-power mode exit

Description:

When executing from the Flash and entering a low-power mode (LPM) where the Flash is in low-power or power-down mode, 2-4 clock cycles exist at the beginning of the RUNx to LPM transition during which a wakeup or interrupt will generate a machine check due to the Flash not being available on RUNx mode re-entry. This will cause either a checkstop reset or machine check interrupt.

Workaround:

This issue can be handled in one of the following ways. Workaround #1 configures the application to handle the machine check interrupt in RAM dealing with the problem if it occurs. Workaround #2 configures the MCU to avoid the machine check interrupt.

Workaround #1: The application can be configured to handle the machine check interrupt in RAM; when this occurs, the mode entry module can be used to bring up the Flash normally and resume execution. Before stop mode entry, ensure the following:

1. Enable the machine check interrupt at the core MSR[ME] - this prevents a machine check reset occurring
2. Copy IVOR vector table to RAM
3. Point IVPR to vector table in RAM
4. Implement machine check interrupt handler in RAM to power-cycle Flash to synchronise status of Flash between Mode Entry and Flash module

The interrupt handler should perform the following steps:

1. Test machine check at LPM exit due to wakeup/interrupt event
2. ME_RUNx_MC[CFLAON] = 0b01 (power-down)
3. Re-enter mode RUNx (x = 0,1,2,3) to power down Flash
4. Wait for transition to RUNx mode to complete (ME_GS[S_MTRANS] = 1)
5. ME_RUNx_MC[CFLAON] = 0b11 (normal)
6. Re-enter mode RUNx (x = previous x) to power up Flash
7. Wait for transition to RUNx mode to complete (ME_GS[S_MTRANS] = 1)
8. On completion, code execution will return to Flash (via se_rfci)

Workaround #2: The application can be configured to avoid the machine check interrupt; low-power mode can be entered from a RAM function and mode entry configured to have Flash off on return to the current RUNx mode. Flash can then be re-enabled by mode entry within the RAM function before returning to execution from Flash.

1. Prior to LPM mode entry request branch to code execution in RAM while Flash is still in normal mode
2. Set ME_RUNx_MC[CFLAON] = 0b01 (power-down) or 0b10 (low-power) for STOP0/HALT0
3. Set ME_STOP0/HALT0_MC[CFLAON] = ME_RUNx_MC[CFLAON]
4. Enter STOP0/HALT0 mode
5. At wakeup or interrupt from STOP0/HALT0, MCU enters RUNx mode executing from RAM with Flash in low-power or power-down as per the ME_RUNx_MC configuration from step 2.
6. After the STOP0/HALT0 request, set ME_RUNx_MC[CFLAON] = 0b11 (normal)
7. Enter RUNx mode
8. Wait for transition to RUNx mode to complete (ME_GS[S_MTRANS] = 0)
9. Return to code execution in Flash

1.21 ERR003609: CRC: Limitation of hardware comparison for CRC result via CRC_OUTP_CHK.

Description:

The comparison via CRC_OUTP_CHK register of the calculated CRC stored in CRC_OUTP register without CPU load doesn't work for Context 2 and Context 3. Moreover if the comparison via CRC_OUTP_CHK register is used for Context 1, then Context 2 and Context 3 cannot be used for CRC calculation If user needs to use more than one context, the comparison has to be done via SW with negligible CPU load.

Workaround:

Using the Context 1, do not use the Context 2 and Context 3 or to do the comparison via SW. Using the Context 2 and/or Context 3, always to do the comparison via SW.

1.22 ERR003610: FlexCAN: Wrong data transmission exiting from STOP mode in case EXTAL frequency is greater than IRC

Description:

The FlexCAN module has got a programmable clock source that can be either the system clock (SYS_CLK) or oscillator clock (XOSC_CLK) selected by CLK_SRC bit in the FlexCAN_CTRL register. In case FlexCAN module has selected the oscillator clock as clock source and XOSC_CLK is bigger than IRC frequency @16MHz and the system clock is PLL_CLK if the device enters STOP mode and the FlexCAN module is in transmission then when device exits from STOP mode the FlexCAN module can transmit wrong data. This behavior happens because during STOP mode exit, SYS_CLK will be IRC @16MHz till PLL gets locked and if a frame transmission happens during this time itself then there will be a CAN Spec violation. The FlexCAN module clock source should not be faster than SYS_CLK.

Workaround:

Just before entering/requesting the STOP mode, set the “FRZ” and “HALT” bit of CAN_MCR register to '1' to request for freeze mode. During the STOP mode exit, check for the mode transition completion. As mode transition will be over, only when all clock sources are on

and the PLL is selected as system clock, unfreeze the CAN by resetting the “FRZ” or “HALT” bit.

1.23 ERR003611: LINFlex: Lin_1 does not work as Slave.

Description:

LIN_1cannot be used as Slave. The two instances of LINFlex (LIN_0 and LIN_1) can both work as Master. Only the LIN_0 can be used as Slave.

Workaround:

No Workaround

1.24 ERR003612: MC_ME: No mode transition from RUNx (x = 0,1,2,3) mode to STOP0/HALT0 mode and possible wake up from STOP0/HALT0 mode when VDD_HV_IO is less than 4.2 V

Description:

The STOP0/HALT0 mode transition can be requested from one of the RUN0…3 modes programming the TARGET_MODE bit field of ME_MCTL register at the value “1010” for STOP0 mode and at the value “1000” for HALT0 mode. In case that the VDD_HV_IO voltage goes down below the LVD_MAIN5 threshold for this the device does not go into STOP0/HALT0 mode. Also if device is in HALT0/STOP0 mode it comes out of HALT0/STOP0 mode when VDD_HV_IO voltage goes down below 4.2 V.

Workaround:

No workaround.

1.25 ERR003702: Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge

Description:

The Nexus output pins (message data outputs 0:12 [MDO] and Message start/end outputs 0:1 [MSEO]) may drive an unknown value (high or low) immediately after power up but before the 1st clock edge propagates through the device (instead of being weakly pulled low). This may cause high currents if the pins are tied directly to a supply/ground or any low resistance driver (when used as a general purpose input [GPI] in the application).

Workaround:

1. Do not tie the Nexus output pins directly to ground or a power supply.
2. If these pins are used as GPI, limit the current to the ability of the regulator supply to guarantee correct start up of the power supply. Each pin may draw upto few hundred mA current.

If not used, the pins may be left unconnected.

1.26 ERR003713: ADC: ADC channels 16 -27 do not show full scale converted value.

Description:

The ADC channels 16 - 27 cannot be used in FULL RANGE, for a 5 V ADC power supply, the max conversion is 3.3 V, and for a 3.3 V supply the conversion is 1.3 V.

Workaround:

No Workaround.

Appendix A Further information

A.1 Reference document

1. *32-bit MCU family built on the Power Architecture® embedded category for automotive chassis and safety electronics applications* (RM0083, Doc ID 018714)
2. *32-bit Power Architecture® based MCU with 1088 KB Flash memory and 80 KB RAM for automotive chassis and safety applications* (SPC560P54x, SPC560P60x, SPC56AP54x, SPC56AP60x datasheet, Doc ID 18340)

Revision history

Table 2. Document revision history

Date	Revision	Changes
07-Sep-2011	1	Initial release.
18-Sep-2013	2	Updated Disclaimer.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER’S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR “AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL” INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.
Information in this document supersedes and replaces all information previously supplied.
The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2013 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com