STR91xFA
Errata sheet

STR91xFA limitations and corrections

Silicon identification

STR91x microcontrollers have two major salestype groups, as follows:

- The initial group consists of STR91xFxxxxx devices, for example, STR912FW44X6, with silicon revisions B and D. Detailed technical information for these devices can be found in the STR91xF datasheet, STR91xF reference manual, and STR91xF errata sheets.
- The second group consists of STR91xFAxxxxx devices, for example STR912FAW44X6, with silicon revision G and H (production devices). Detailed technical information for these devices can be found in the STR91xFA datasheet and STR91xFA reference manual.

This errata sheet covers only the second group.

Table 1 summarizes the marking to assist identification. *Figure 1* through *Figure 4* show where the physical marking can be found on the devices.

Table 1. Device identification

Salestype group, external marking	Internal Silicon revision	External marking	Note
STR91xFAx32xx STR91xFAx42xx STR91xFAx44xx	G	G	See *Figure 1* Production devices. LQFP128 and LQFP80 packages.
		G	See *Figure 2* Production devices. LFBGA144 packages.
	H	H	See *Figure 3* Production devices. LQFP128 and LQFP80 packages.
		H	See *Figure 4* Production devices. LFBGA144 packages.
STR91xFAx46xx STR91xFAx47xx	A	A	See *Figure 5* Production devices. LQFP128 and LQFP80 packages.
		A	See *Figure 6* Production devices. LFBGA144 packages.

Contents

1 Product evolution

Table 2 summarizes the fix status (● = fix).

Table 2. Product evolution summary

Section	Limitation	STR91xFAx32, STR91xFAx42, STR91xFAx44		STR91xFAx46 STR91xFAx47
		Revision G status	Revision H status	Revision A status
Section 2.1	*Flash memory status register bit 7*	Workaround	Workaround	Workaround
Section 2.2	*Flash memory sector protection*	Workaround	Workaround	Workaround
Section 2.3	*Flash memory remapping*	Workaround	●	●
Section 2.4	*Flash fetch/read across banks*	Workaround	Workaround	Workaround
Section 2.5	*Flash frequency limitation when programming*	Workaround	Workaround	Workaround
Section 2.6	*ETM (embedded trace module) configuration*	Workaround	Workaround	Workaround
Section 2.7	*EMI bus limitations*	Workaround	Workaround	Workaround
Section 2.8	*Sleep mode current (I_{SLEEP}) on V_{DDQ} pins*	Workaround	●	●
Section 2.9	*Waking up from sleep mode*	Workaround	Workaround	Workaround
Section 2.10	*Sleep and Idle mode requirements*	Workaround	Workaround	Workaround
Section 2.12	*High leakage on GPIO ports at 5.5 V*	Workaround	Workaround	Workaround
Section 2.13	*ADC interrupt generation*	Workaround	Workaround	Workaround
Section 2.14	*ADC conversion time and external trigger mode*	Workaround	●	●
Section 2.15	*ADC scan and continuous modes*	N/A	Workaround	Workaround
Section 2.16	*ADC single mode and channel selection*	Workaround	Workaround	Workaround
Section 2.17	*Daisy chained interrupt controller VIC1 hardware priority management limitation*	Workaround	Workaround	Workaround
Section 2.18	*Flash configuration register bits correction*. Refer to PM0020: STR91xFA Flash programming manual (section 1.13.6 Flash configuration register).	Workaround	Workaround	Workaround

2 Silicon limitations and fixes

2.1 Flash memory status register bit 7

Description of limitation

Status register bit 7 (ready bit) does not reflect the correct status when it is read immediately after the CPU issues a Flash memory program or erase command.

Workaround

This requires bit 18 (instruction TCM order bit) in the configuration control register of the ARM966E-S core to be set. This can be done by the following assembler code:

```
MOV R0, #0x40000
MCR P15,0x1,R0,C15,C1,0
```

When set, the write and read to the Flash bank are performed in the order generated by the ARM966-ES core.This ensures that writes are committed to the Flash memory before any subsequent read.

This will not be fixed in future silicon revisions.

2.2 Flash memory sector protection

Description of limitation

At power up, the Flash protection level 1 register is reset to 0FFFh (all Flash sectors are protected). A warm reset does not reset the register.

Workaround

Firmware must change the values if desired. This will not be fixed in future silicon revisions.

2.3 Flash memory remapping

Description of limitation in revision G

There are two independent Flash memory banks (primary and secondary) that allow read-while-write capability from one bank to the other, enabling in-application programming (IAP) of the primary bank while executing code from the secondary bank. After IAP has upgraded firmware in the primary bank, the bootloader code in the secondary bank should remap the banks by writing to FMI registers FMI_BBADR and FMI_NBBADR in order to have the primary bank start at address 0x0, which allows interrupt handlers to reside in the application code primary bank instead of interrupts being handled by bootloader code in the secondary bank.

There are two forms of reset in the STR91xFA: A global reset (from power-up or voltage drop-out) that clears all functions in the device, and a system reset (from the reset input pin, watchdog, or JTAG reset command) that clears all but a few configuration registers.

Silicon limitation

The remapping operation is possible. However, when a system reset occurs after the remapping, the application hangs because the FMI_BBADR and FMI_NBBADR registers which define bank locations are not cleared by the system reset, while the FMI_CR register which controls the chip select enable of the non boot bank are cleared by the system reset.

Workaround using revision G

Do not remap the banks after IAP (keep secondary bank at address 0x0), and in the secondary bank redirect ARM interrupts to be handled in the primary bank.

This limitation is fixed in revision H devices by clearing all FMI registers (including bank base address and bank size) after a system reset.

2.4 Flash fetch/read across banks

Description of limitation

In either of the following two conditions, the CPU may freeze or generate an exception:

1. When the two banks (Bank0 and Bank1) are configured at contiguous addresses in the STR91xFA address map and if a code sequence is stored and then fetched by the CPU across this boundary,
2. When data is read from a bank other than the bank fetched by the CPU for execution.

This will not be fixed in future silicon revisions.

Workarounds

For the first case, three workarounds are proposed:

1. Do not use the last 8 x 32-bit words of a Flash bank for storing code and use it instead to store data constants (this can be configured by the linker to specify the last zone).
2. If the application firmware is larger than one bank in size, ensure that before reaching the last 8 x 32-bit words of one bank, application code must jump (using LDR, or a branch) to an address forced by the linker to a location in the other bank.
3. Disable the PFQ block before executing code stored across a bank boundary.

For the second case, the workaround is as follows:

Disable the PFQ block before reading (data/constants) from a bank other than the bank from which code is fetched and enable it again after completing the read operations.

2.5 Flash frequency limitation when programming

Description of limitation

When programming the internal Flash or the Flash registers with an FMICLK frequency of 96 MHz, either using read-while-write operations or executing from the internal SRAM, for some STR91xFA slow parts that may be seen in the field, some data are not properly written. This issue is due to a limitation in the Flash command interface that affects Flash write operations (such as program or erase).

Workaround

To ensure that internal Flash programming and CUI registers (command user interface) programmings are performed correctly, configure the frequency of FMICLK to RCLK/2 during these operations.

2.6 ETM (embedded trace module) configuration

Description of limitation

By default, the ETM9 interface in the STR91xFA core is an 8-bit medium size model as defined by ARM Ltd. When an emulator tool boots up and performs an automatic configuration on the ETM (ETM sniffers), the configuration register always sends back the 8-bit model status.

However, in order to reduce the number of I/O pins required for debugging, the ETM trace data port is implemented as a 4-bit port. When polled, the 8-bit status provided by the configuration register is incorrect.

Workaround

When booting up an emulator, do not select the automatic configuration option. Instead, configure the ETM to be a 4-bit port manually.

This limitation will not be fixed in future silicon revisions.

2.7 EMI bus limitations

2.7.1 Impact of SCU_CLKCNTR register modification on EMI operation with clock ratio of 2

Description of limitation

When the EMI clock is configured with a ratio of 2, modifying SCU_CLKCNTR register may cause incorrect EMI write cycles to occur when the EMI is used. This is due to the de-synchronization of the two clock state machines feeding the EMI when the EMI clock ratio is configured with a ratio of 2.

Workaround

Whenever the SCU_CLKCNTR register is modified, the EMI must be subsequently reset by writing bit 6 (RST_EMI) of the SCU_PRR0 register to 0 and setting it again just before configuring and using the EMI.

2.7.2 Address boundary limitation in 16-bit asynchronous or synchronous modes

Description of limitation

When accessing the external memory in 16-bit mode, the address must always be aligned to half word (16 bit) or word (32 bit) boundary. The memory address must consequently be an even address, and reading or writing from/to an odd address location may result in incorrect data.

This limitation applies to data memory space only, as code fetch is always aligned to word boundary.

Workaround

All data must be stored at even address location.

2.7.3 Write signal configuration in 16-bit synchronous mode

Description of limitation (applies only to BGA144 devices)

There are two sets of write control signals for 16-bit write bus cycle:

1. EMI_WRLn, EMI_WRHn
2. EMI_WEn, EMI_UBn, EMI_LBn

The 2nd set of signals is activated by setting bit 2 in the SCU_GPIOEMI register. When configured in 16-bit synchronous mode, the EMI bus works only with the 2nd set of write signals. The 1st set of signal results in incorrect write signal timings.

Workaround

When using EMI in 16-bit synchronous mode, choose a memory device which accepts write enable (EMI_WEn), EMI_UBn and EMI_LBn signals.

If these signals are not available on the memory, it is recommended to operate in asynchronous mode.

2.8 Sleep mode current (I_{SLEEP}) on V_{DDQ} pins

Description of limitation

When the STR91xFA enters Sleep mode, the current drawn from the CPU core voltage (V_{DD}) and from the I/O supply voltage (V_{DDQ}) should drop to a very low value.

I_{SLEEP} current on V_{DD} pins correctly drops to as low as 50 µA at 25 °C.

However I_{SLEEP} current on the V_{DDQ} pins drops to around 500 µA at 25 °C while it should be less than 10 µA.

Workaround using revision G

A limited workaround may be implemented to save around 120 µA by using firmware to put the USB transceiver into Suspend mode (bit2, LP_MODE in the USB_CNTR register). This allows reducing Sleep mode current on V_{DDQ} pins to around 380 µA.

Sleep mode current consumption on V_{DDQ} pins is reduced to less than 10 µA on revision H silicon devices.

2.9 Waking up from sleep mode

Description of limitation

After the CPU enters sleep mode, it can be woken up by:

1. External interrupt
2. RTC/USB interrupt
3. External reset

When an oscillator chip is used as the clock source for the STR91xFA, the CPU wakes up from sleep mode following any of the above three input events. If a crystal is used as the clock source, the crystal is disabled in sleep mode to save power consumption. When a wakeup event occurs, the crystal does not recover fast enough and the CPU hangs.

Workarounds

Workaround solutions include:

1. Use the 32 kHz RTC clock as the clock source for sleep mode:
   a) Select the RTC clock as the CPU clock source prior entering sleep mode.
   b) The CPU wakes up following any of the three wakeup events and waits for the crystal to start oscillation. A crystal startup time is about 1.5 ms typical.
   c) After the crystal wakes up, the CPU waits for a t_{WAIT} time before the first code is fetched from Flash memory. The software can then change the CPU clock source back to the OSC or PLL clock. The duration of t_{WAIT} depends on the crystal frequency. t_{WAIT} equals 50 µs at 25 MHz and 312 µs at 4 MHz.
2. Instead of a crystal, use an oscillator as STR91xFA clock source.

This limitation will not be fixed in future silicon revisions

2.10 Sleep and Idle mode requirements

2.10.1 Code execution after setting the sleep or idle mode bit

Description of limitation

Once the idle or sleep mode are entered by writing the PWR_MODE[2:0] bits in the SCU_PWRMNG register, it takes about 12 f_{OSC} cycles for the device to stop the execution.

In addition, if a wakeup event or an interrupt (external or internal coming from peripherals) occurs during this period while entering idle, the internal low power state machine is frozen and the STR91xFA hangs. In this case, only a reset event can wake up the device.

Workarounds

In order to avoid executing any valid instructions after setting the idle or sleep bit and before entering the mode, it is mandatory to execute a certain number of dummy instructions after setting the SCU_PWRMNG register. The number of dummy instructions to be executed is given by the following formula:

$$\text{No\_dummy\_instr} = (f_{CPUCLK} / f_{OSC}) \times 12$$

Where f_{CPUCLK} is the CPU core clock frequency and f_{OSC} is the oscillator frequency.

The worst scenario is obtained when the core works from the PLL maximum frequency (96 MHz) with an 4 MHz crystal or oscillator connected to the X1_CPU input. In this case 288 dummy instructions are needed.

Note: If (f_{CPUCLK}/ f_{OSC}) is less than 1, the number of dummy instruction is always 3.

Random external/internal wakeup events or interrupts may freeze the STR91xFA when occurring during the execution of these dummy instructions. In this case, only a reset event can wake up the CPU.

This limitation will not be fixed in future silicon revisions

2.10.2 Time required to enter sleep mode

Description of limitation

After the mode bit is set in the SCU_PWRMNG register, the power management unit requires a period of time (t_{SLEEP}) to switch off all CPU and peripheral clocks safely before entering sleep mode. A very slow peripheral clock results in a long switch off time.

The t_{SLEEP} time required to enter sleep mode depends on the oscillator frequency, on the slowest peripheral clock frequency, and on the CPU clock frequency. If a wakeup event occurs during t_{SLEEP}, it is ignored and the STR91xFA does not exit from sleep mode. t_{SLEEP} is given by the following formula:

$$t_{SLEEP} = 17 \times \text{t\_OSC} + 14 \times \text{t\_Slowest\_IP\_CLK} + 6 \times \text{t\_CPUCLK}$$

Where t_OSC is the oscillator frequency, t_Slowest_IP_CLK the slowest peripheral clock frequency, and t_CPUCLK the CPU clock frequency.

Example

- CPU running on RTC clock before entering sleep mode (f_{CPUCLK} = 32 kHz) (see *Section 2.9*).
- t_OSC = 40 ns (f_{OSC} = 25 MHz)
- t_CPUCLK = t_RTC = 31,250 ns (f_{CPUCLK} = 32 kHz)
- t_Slowest_IP_CLK = 2*31,250 ns assuming all clock dividers are set to 1 (default state) except for APB clock divided which is set to 2, the slowest peripheral clock frequency is then f_{CPUCLK}/2.

Then, the value of t_{SLEEP} is:

$$t_{SLEEP} = 17 \times 40 + 14 \times 2 \times 31,\ 250 + 6 \times 31,\ 250 \sim 1.06\text{ms}$$

Workaround

To prevent random external wakeup events from occurring while the device is entering sleep mode (during t_{SLEEP}), the maximum time required by the application to enter sleep mode must be taken into account.

This limitation will not be fixed in future silicon revisions.

2.11 Noise sensitivity of GPIO ports

Description of limitation

None of the STR9 general inputs have hysteresis which means they have no Schmitt trigger. Simulation in typical conditions (3.3 V, ambient temperature, typical processing) shows that approximately 25 mV noise is enough to provide unexpected glitches in the core when the 1 kHz pad signaling crosses the input buffer threshold. The slower the input signaling frequency is, the greater this limitation.

Workarounds

Some possible software workarounds can be implemented depending on the application and the impact of the limitation.

A hardware workaround, feasible in all cases, is to implement an external Schmitt trigger to act as a noise filter.

2.12 High leakage on GPIO ports at 5.5 V

Description of limitation

A high leakage current is observed when applying 5.5 V on 5 V-tolerant I/O pins. This leakage does not dependent on V_{DDQ}, and increases with the temperature.

This high leakage is distributed between the different pins when using a 5.5 V supply voltage. If the application hardware has 10 pins connected to 5.5 V, the leakage current is not multiplied by 10, but distributed between the 10 pins.

This limitation will not be fixed in future silicon revisions.

Workaround

Limit the voltage applied to the 5 V-tolerant I/O pins to 5 V to limit the leakage current.

2.13 ADC interrupt generation

Description of limitation

The ADC generates an end of conversion (EVC) or an analog watchdog (AWD) interrupt when enabled. Before returning from serving the interrupt, the ISR typically clears the interrupt by setting the corresponding EVC or AWD flag bit in the ADC_CR register to ‘0’.

The ADC clock is used to clear the interrupt flags. The time taken to clear the flags is longer when the ADC runs on a slow clock. The CPU may return from ISR before the interrupt flag has been cleared. Since the interrupt controller input is level sensitive, the CPU sees immediately if another interrupt is pending.

Workaround

Instead of clearing the ADC interrupt flag at the end of the ISR, clear the flag when ISR is entered.

This limitation will not be fixed in future silicon revisions.

2.14 ADC conversion time and external trigger mode

Description of limitation

When the ADC unit is in single conversion mode, the time to complete the conversion varies between 36 and 48 ADC clock periods after the conversion is initiated by an external ADC trigger, a timer trigger, or a firmware command. This limits the maximum ADC conversion rate to 500,000 samples per second for a single channel using an external trigger. This situation also introduces a "jitter" of as many as 12 ADC clocks from one completed conversion to the next.

When the ADC unit is in continuous and scan conversion modes, the time to complete the first conversion varies between 36 and 48 ADC clocks like single conversion mode, but subsequent conversions complete every 16 ADC clocks producing a maximum ADC conversion rate of 1,500,000 samples per second for a single channel.

Workaround on revision G

In single conversion mode, there is no workaround to exceed 500 Ksps conversion rate with external trigger or timer trigger, and no workaround to reduce jitter from one sample to the next.

In continuous and scan conversion modes there is no workaround to reduce delay of the first conversion.

To eliminate jitter, a new conversion mode started by fast trigger has been added in silicon revision H (refer to STR91xFA datasheet and reference manual). It allows each conversion to be completed in 16 ADC clocks after it has been triggered by external ADC trigger or internal timer event. A minimum conversion rate of 1.2 Msps can be achieved on a single channel. In single and scan mode, the time to complete the first conversion is reduced to the range of 20 to 32 clocks (down from 36 to 48 clocks) after a trigger event allowing continuous conversion. In this case, the maximum rate on subsequent continuous conversions remains 1.5 Msps on a single channel.

2.15 ADC scan and continuous modes

Description of limitation only in revision H

ADC scan and continuous mode cannot be used together on revision H silicon. The first conversion is performed, but the next channel is not selected, so the end of conversion never occurs.

Workaround

Instead of using scan mode with continuous mode, the application has to select scan mode in single mode. Each conversion can be started by an internal trigger (PWM) for the required number of channels.

The conversion time is deterministic in fast trigger mode. This is a new feature available in revision H silicon (see STR91xFA datasheet and reference manual).

It can be computed using the following formula:

$$f_{TRIGGER} < 1 / (nb\_channels \times 16 / f_{ADC})$$

2.16 ADC single mode and channel selection

Description of limitation only in rev H

If both channel selection and start conversion bits (SC and STR bits) are written at the same time, it may occur that the ADC sampling/conversion is performed on the previous selected channel.

Workaround

A delay max. of 16 ADC clocks between the channel selection and the conversion start should be enough to prevent this effect. A generic dummy loop (equivalent to 16 ADC clocks) is recommended. This workaround is only needed in single mode with a channel selection change.

2.17 Daisy chained interrupt controller VIC1 hardware priority management limitation

Description of limitation

For daisy-chained interrupts, if only the VAR from one VIC is read, it does not update the hardware priority in the other VIC. This means the daisy chained interrupt controller doesn't realize the interrupt is being serviced and keeps the interrupt request active. For daisy-chained interrupts the processor must read the VAR register from both interrupt controllers and branch to the addresses provided. However, the address provided by reading the daisy-chained VAR register must be manipulated to skip the interrupt preamble. If two daisy chained interrupts occur soon after each other and the daisy chained VAR register address isn't branched to, the interrupt controller priority logic may be updated incorrectly. This may cause a low priority interrupt to be missed.

Workaround

Note: This workaround is taken from the ARM prime cell vectored interrupt controller (PL190) errata notice.

For already existing software that services daisy-chained nested interrupts, a possible software workaround is to branch to the address provided by the VIC1 VAR register. This ensures that the correct ISR is serviced.

```
0x18 LDR pc, [VIC0_VAR]        ; VIC0 VAR read
                               ; daisy_chained_vector_handler:

STMFD r13!, {r12-r14}

LDR r12, [VIC1_VAR]            ; Read VIC1 VAR and update PC,

                               ; VIC1 priority hardware is updated

LDR PC, [r12, #12]             ; Processor branches to the highest priority
                               ; daisy chained ISR and skips
                               ; the preamble(+12)
```

Note: The value, 12, used in instruction LDR PC, [r12, #12] is only specific to the example code given here. The offset #12 is dependent on the system's interrupt stacking preamble code size.

2.18 Flash configuration register bits correction

Description of limitation

Using wait state insertion bits and/or the Flash bus clock bit under some specific configurations and/or combinations with respect to the Flash programming manual, may cause the device to be used outside its operating conditions.

Workaround

To ensure that the application use of the above bits is well supported, a new Flash configuration specification has been put in place:

Bits 12:11 **WSTATES[1:0]:** Wait states

These bits define the number of wait states inserted in asynchronous read access.
00: 1 wait state (default)
01: 2 wait states
10: 3 wait states

Note: Wait states are inserted only for non-bursting Flash read bus cycles.
One wait state is required for a Flash memory interface (FMI) bus clock frequency < 66 MHz.
Two wait states are required for an FMI bus clock frequency ≥ 66 MHz.

Bit 4 **BUSCFG:** Flash bus clock configuration

This bit selects the FMI Flash bus clock configuration.
0: BUSCFG disabled (default)
1: BUSCFG enabled

Note: The BUSCFG bit must be set for a frequency of:
≥ 48 MHz (with 1 wait state)
≥ 66 MHz (with 2 wait state).

Appendix A Revision code on device marking

Figure 1 to *Figure 6* show the marking compositions for the LQFP and BGA packages.

Figure 1. Device marking for revision G LQFP80 and LQFP128 packages

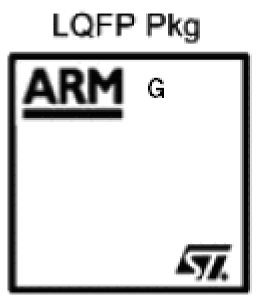


Figure 2. Device marking for revision G LFBGA144 packages

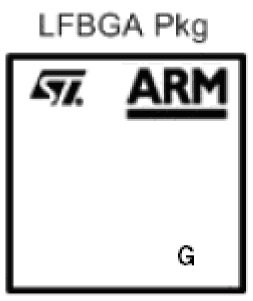


Figure 3. Device marking for revision H LQFP80 and LQFP128 packages

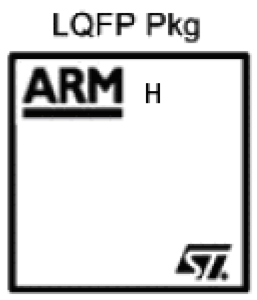


Figure 4. Device marking for revision H LFBGA144 packages

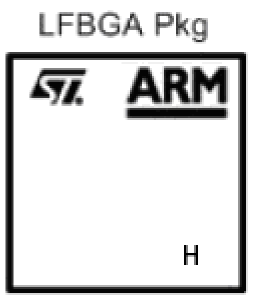


Figure 5. Device marking for revision A LQFP80 and LQFP128 packages

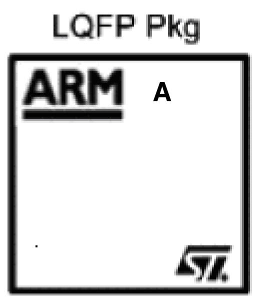
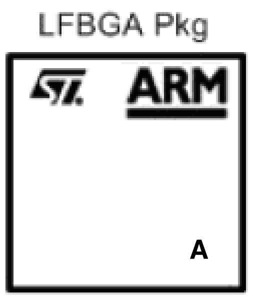


Figure 6. Device marking for revision A LFBGA144 packages



3 Revision history

Table 3. Document revision history

Date	Revision	Changes
10-May-2007	1	Initial release
18-Dec-2007	2	Added *Section 2.4: Flash fetch/read across banks*. Added *Section 2.10: Sleep and Idle mode requirements*. Added *Section 2.12: High leakage on GPIO ports at 5.5 V*. Added*Section 2.12: High leakage on GPIO ports at 5.5 V*. Added *Section 2.15: ADC scan and continuous modes*.
12-Nov-2008	3	Added *Section 2.5: Flash frequency limitation when programming*. Added *Section 2.7.1: Impact of SCU_CLKCNTR register modification on EMI operation with clock ratio of 2*. Added *Section 2.16: ADC single mode and channel selection*. Added *Section 2.17: Daisy chained interrupt controller VIC1 hardware priority management limitation*.
14-Dec-2008	4	Removed former section 2.11 “Pull-up effect on GPIO ports”. Updated *Section 2.4: Flash fetch/read across banks*.
18-May-2009	5	Added *Section 2.11: Noise sensitivity of GPIO ports*
02-Jul-2009	6	Updated section *Section 2.5: Flash frequency limitation when programming*
29-Jun-2010	7	Added *Section 2.18: Flash configuration register bits correction*