STM8AF622x/4x/66/68
STM8AF612x/4x/66/68 errata sheet

Device limitations for STM8AFxxxx automotive MCUs featuring up to 32 Kbytes of Flash program memory

Silicon identification

This errata sheet applies to the STMicroelectronics STM8AF622x/4x, STM8AF6266/68, STM8AF612x/4x, and STM8AF6166/68 devices. The full list of root part numbers is given in *Table 2*.

The products can be identified as shown in *Table 1*:

- By the revision code marked below the sales type on the device package
- By the last three digits of the Internal sales type printed on the box label

Table 1. Device identification

Sales type	Revision code marked on the device(1)
STM8AF622x/4x	X, W
STM8AF6266/68	X, W
STM8AF612x/4x	Y
STM8AF6166/68	Y

1. Refer to *Appendix A: Revision code on device marking* for details on how to identify the revision code according to the packages.

Table 2. Device summary

Part number	Part number
STM8AF622x/4x	STM8AF6226, STM8AF6246, STM8AF6248
STM8AF6266/68	STM8AF6266, STM8AF6268
STM8AF612x/4x	STM8AF6126, STM8AF6146, STM8AF6148
STM8AF6166/68	STM8AF6166, STM8AF6168

Contents

1 Product evolution

Table 3 gives a summary of the fix status.

Legend for *Table 3*: A = workaround available; N = no workaround available; P = partial workaround available, '-' and grayed = fixed.

Table 3. Product evolution summary

Section	Limitation	Rev Y	Rev X, W
Section 2.1: STM8A core	*Section 2.1.1: Activation level (AL) bit not functional in Halt mode*	N	N
	Section 2.1.2: JRIL and JRIH instructions not available	N	N
	Section 2.1.3: Main CPU execution is not resumed after an ISR resets the AL bit	A	A
	Section 2.1.4: Unexpected DIV/DIVW instruction result in ISR	A	A
	Section 2.1.5: Wait for event instruction (WFE) not available	N	N
Section 2.2: System limitations	*Section 2.2.1: HSI RC oscillator cannot be switched off in run mode*	N	N
	Section 2.2.2: LSI oscillator remains on in Active-halt mode when the AWU unit uses the HSE as input clock	N	N
Section 2.3: LINUART peripheral limitations	*Section 2.3.1: PE testing issue in UART mode*	N	N
	Section 2.3.2: LIN mode: LIN header error when automatic resynchronization is enabled	N	-
	Section 2.3.3: LIN mode: framing error with data byte 0x00	N	N
	Section 2.3.4: LIN mode: framing error when receiving an identifier (ID)	N	N
	Section 2.3.5: LIN mode: parity error when receiving an identifier (ID)	N	N
	Section 2.3.6: LIN mode: OR flag not correctly set in LIN Master mode	N	N
Section 2.4: I^2C peripheral limitations	*Section 2.4.1: I^2C event management*	A	A
	Section 2.4.2: Corrupted last received data in I^2C Master Receiver mode	A	A
	Section 2.4.3: Wrong behavior of I^2C peripheral in Master mode after misplaced STOP	A	A
	Section 2.4.4: Violation of I^2C "setup time for repeated START condition" parameter	A	A
	Section 2.4.5: In I^2C slave "NOSTRETCH" mode, underrun errors may not be detected and may generate bus errors	A	A
	Section 2.4.6: I^2C pulse missed	A	-

Table 3. Product evolution summary (continued)

Section	Limitation	Rev Y	Rev X, W
Section 2.5: SPI peripheral limitations	*Section 2.5.1: Last bit too short if SPI is disabled during communication*	A	A
	Section 2.5.2: Busy flag is not reliable when the SPI is a master simplex receiver	N	N
Section 2.6: ADC peripheral limitation	*Section 2.6.1: EOC interrupt triggered when AWDIE and EOCIE set to '1'*	N	N

2 Silicon limitations

2.1 STM8A core

2.1.1 Activation level (AL) bit not functional in Halt mode

Description

The AL bit is not supported in Halt mode. In particular, when the AL bit of the CFG_GCR register is set, the CPU does not return to Halt mode after exiting an interrupt service routine (ISR). It returns to the main program and executes the next instruction after the HALT instruction. The AL bit is supported correctly in WFI mode.

Workaround

No workaround available.

2.1.2 JRIL and JRIH instructions not available

Description

JRIL (jump if port INT pin = 0) and JRIH (jump if port INT pin = 1) are not supported by the devices covered by this datasheet. These instructions perform conditional jumps: JRIL and JRIH jump if one of the external interrupt lines is low and high, respectively. JRIL is equivalent to an unconditional jump and JRIH is equivalent to a NOP.

For further details on these instructions, refer to the STM8 CPU programming manual (PM0044) on *www.st.com*.

Workaround

No workaround available.

2.1.3 Main CPU execution is not resumed after an ISR resets the AL bit

Description

If the CPU is in wait for interrupt state and the AL bit is set, the CPU returns to wait for interrupt state after executing an ISR. To continue executing the main program, the AL bit must be reset by the ISR. When AL is reset just before exiting the ISR, the CPU may remain stalled.

Workaround

Reset the AL bit at least two instructions before the IRET instruction.

No fix is planned.

2.1.4 Unexpected DIV/DIVW instruction result in ISR

Description

In very specific conditions, a DIV/DIVW instruction may return a false result when executed inside an interrupt service routine (ISR). This error occurs when the DIV/DIVW instruction is interrupted and a second interrupt is generated during the execution of the IRET instruction of the first ISR. Under these conditions, the DIV/DIVW instruction executed inside the second ISR, including function calls, may return an unexpected result.

The applications that do not use the DIV/DIVW instruction within ISRs are not impacted.

Workaround 1

If an ISR or a function called by this routine contains a division operation, the following assembly code should be added inside the ISR before the DIV/DIVW instruction:

```
push cc
pop a
and a,#$BF
push a
pop cc
```

This sequence should be placed by C compilers at the beginning of the ISR using DIV/DIVW. Refer to your compiler documentation for details on the implementation and control of automatic or manual code insertion.

Workaround 2

To optimize the number of cycles added by workaround 1, you can use this workaround instead. Workaround 2 can be used in applications with fixed interrupt priorities, identified at the program compilation phase:

```
push #value
pop cc
```

where bits 5 and 3 of #value have to be configured according to interrupt priority given by I1 and I0, and bit 6 kept cleared.

In this case, compiler workaround 1 has to be disabled by using compiler directives.

No fix is planned for this limitation.

2.1.5 Wait for event instruction (WFE) not available

Description

The WFE instruction is not implemented in the devices covered by this datasheet. This instruction is used to synchronize the device with external computing resources. For further details on this instruction, refer to the STM8 CPU programming manual (PM0044) on *www.st.com*.

Workaround

No workaround available.

2.2 System limitations

2.2.1 HSI RC oscillator cannot be switched off in run mode

Description

The internal 16 MHz RC oscillator cannot be switched off in run mode, even if the HSIEN bit is programmed to 0.

Workaround

No workaround available. No fix planned.

2.2.2 LSI oscillator remains on in Active-halt mode when the AWU unit uses the HSE as input clock

Description

When the auto-wakeup unit (AWU) uses the high speed external clock (HSE) divided by the prescaler (clock source enabled by setting the CKAWUSEL option bit), the LSI RC oscillator is not switched off when the device operates in Active-halt mode with the main voltage regulator (MVR) on. This causes negligible extra power consumption compared to the total consumption of the MCU in Active-halt mode with the MVR on.

Workaround

No workaround available. No fix planned.

2.3 LINUART peripheral limitations

2.3.1 PE testing issue in UART mode

Description

When the RXNE flag is not polled, the device is in overrun condition and the PE flag does not rise in case of a parity error. The flag rises only for the last data which have been correctly received.

Workaround

No workaround available. No fix planned.

2.3.2 LIN mode: LIN header error when automatic resynchronization is enabled

Description

If UART2 is configured in LIN slave mode (LSLV bit set in UART2_CR6 register) and the automatic resynchronization is enabled (LASE bit set in UART2_CR6), the LHE flag may be set instead of LHDF flag when receiving a valid header.

This limitation is fixed in silicon revision X.

Workaround

No workaround available.

2.3.3 LIN mode: framing error with data byte 0x00

Description

If the UART2 interface is configured in LIN slave mode, and the active mode with break detection length is set to 11 (LBDL bit of UART2_CR4 register set to 1), FE and RXNE flags are not set when receiving a 0x00 data byte with a framing error, followed by a recessive state. This occurs only if the dominant state length is between 9.56 and 10.56 times the baud rate.

Workaround

The LIN software driver can handle this exceptional case by implementing frame timeouts to comply with the LIN standard. This method has been implemented in ST LIN 2.1 driver package which passed the LIN compliance tests.

2.3.4 LIN mode: framing error when receiving an identifier (ID)

Description

If an ID framing error occurs when the UART2, configured in LIN mode, is in active mode, both the LHE and LHDF flags are set at the end of the LIN header with an ID framing error.

Workaround

The LIN software driver can handle this case by checking both LHE and LHDF flags upon header reception.

2.3.5 LIN mode: parity error when receiving an identifier (ID)

Description

If an ID parity error occurs, the UART2, configured in LIN mode, wakes up from mute mode and both LHE and LHDF are set at the end of the LIN header with parity error. The PE flag is also set.

Workaround

The LIN software driver can handle this case by checking all flags upon header reception.

No fix planned.

2.3.6 LIN mode: OR flag not correctly set in LIN Master mode

Description

When the UART operates in LIN Master mode, the OR flag is not set if an overrun condition occurs.

Workaround

The LIN software driver can detect this case through a LIN protocol error.

No fix planned.

2.4 I²C peripheral limitations

2.4.1 I²C event management

Description

As described in the I²C section of the STM8S and STM8A microcontroller reference manual (RM0016), the application firmware has to manage several software events before the current byte is transferred. If the EV7, EV7_1, EV6_1, EV6_3, EV2, EV8, and EV3 events are not managed before the current byte is transferred, problems may occur such as receiving an extra byte, reading the same data twice, or missing data.

Workaround

When the EV7, EV7_1, EV6_1, EV6_3, EV2, EV8, and EV3 events cannot be managed before the current byte transfer, and before the acknowledge pulse when the ACK control bit changes, it is recommended to use I²C interrupts in nested mode and to make them uninterruptible by increasing their priority to the highest priority in the application.

No fix is planned for this limitation.

2.4.2 Corrupted last received data in I²C Master Receiver mode

Conditions

In Master Receiver mode, when the communication is closed using method 2, the content of the last read data may be corrupted. The following two sequences are concerned by the limitation:

- Sequence 1: transfer sequence for master receiver when N = 2
  a) BTF = 1 (Data N-1 in DR and Data N in shift register)
  b) Program STOP = 1
  c) Read DR twice (Read Data N-1 and Data N) just after programming the STOP bit.
- Sequence 2: transfer sequence for master receiver when N > 2
  a) BTF = 1 (Data N-2 in DR and Data N-1 in shift register)
  b) Program ACK = 0
  c) Read Data N-2 in DR
  d) Program STOP bit to 1
  e) Read Data N-1.

Description

The content of the shift register (data N) is corrupted (data N is shifted 1 bit to the left) if the user software is not able to read data N-1 before the STOP condition is generated on the bus. In this case, reading data N returns a wrong value.

Workarounds

- Workaround 1
  - Sequence 1

    When sequence 1 is used to close communication using method 2, mask all active interrupts between STOP bit programming and Read data N-1.
  - Sequence 2

    When sequence 2 is used to close communication using method 2, mask all active interrupts between Read data N-2, STOP bit programming and Read data N-1.
- Workaround 2

  Manage I2C RxNE and TxE events with interrupts of the highest priority level, so that the condition BTF = 1 never occurs.

2.4.3 Wrong behavior of I²C peripheral in Master mode after misplaced STOP

Description

The I²C peripheral does not enter Master mode properly if a misplaced STOP is generated on the bus. This can happen in the following conditions:

- If a void message is received (START condition immediately followed by a STOP): the BERR (bus error) flag is not set, and the I²C peripheral is not able to send a START condition on the bus after writing to the START bit in the I2C_CR2 register.
- In the other cases of a misplaced STOP, the BERR flag is set in the IC2_CR2 register. If the START bit is already set in I2C_CR2, the START condition is not correctly generated on the bus and can create bus errors.

Workaround

In the I²C standard, it is not allowed to send a STOP before the full byte is transmitted (8 bits + acknowledge). Other derived protocols like CBUS allow it, but they are not supported by the I²C peripheral.

In case of noisy environment in which unwanted bus errors can occur, it is recommended to implement a timeout to ensure that the SB (start bit) flag is set after the START control bit is set. In case the timeout has elapsed, the peripheral must be reset by setting the SWRST bit in the I2C_CR2 control register. The I²C peripheral should be reset in the same way if a BERR is detected while the START bit is set in I2C_CR2.

No fix is planned for this limitation.

2.4.4 Violation of I²C “setup time for repeated START condition” parameter

Description

In case of a repeated Start, the “setup time for repeated START condition” parameter (named $t_{SU(STA)}$ in the datasheet and Tsu:sta in the I²C specifications) may be slightly violated when the I²C operates in Master Standard mode at a frequency ranging from 88 to 100 kHz. $t_{SU(STA)}$ minimum value may be 4 µs instead of 4.7 µs.

The issue occurs under the following conditions:

1. The I^2C peripheral operates in Master Standard mode at a frequency ranging from 88 to 100 kHz (no issue in Fast mode)
2. and the SCL rise time meets one of the following conditions:
   – The slave does not stretch the clock and the SCL rise time is more than 300 ns (the issue cannot occur when the SCL rise time is less than 300 ns).
   – or the slave stretches the clock.

Workaround

Reduce the frequency down to 88 kHz or use the I^2C Fast mode if it is supported by the slave.

2.4.5 In I^2C slave "NOSTRETCH" mode, underrun errors may not be detected and may generate bus errors

Description

The data valid time ($t_{VD;DAT}$, $t_{VD;ACK}$) described by the I^2C specifications may be violated as well as the maximum current data hold time ($t_{HD;DAT}$) under the conditions described below. In addition, if the data register is written too late and close to the SCL rising edge, an error may be generated on the bus: SDA toggles while SCL is high. These violations cannot be detected because the OVR flag is not set (no transmit buffer underrun is detected).

This issue occurs under the following conditions:

1. The I^2C peripheral operates In Slave transmit mode with clock stretching disabled (NOSTRETCH=1)
2. and the application is late to write the DR data register, but not late enough to set the OVR flag (the data register is written before the SCL rising edge).

Workaround

If the master device supports it, use the clock stretching mechanism by programming the bit NOSTRETCH=0 in the I2C_CR1 register.

If the master device does not support it, ensure that the write operation to the data register is performed just after TXE or ADDR events. You can use an interrupt on the TXE or ADDR flag and boost its priority to the higher level.

Using the "NOSTRETCH" mode with a slow I^2C bus speed can prevent the application from being late to write the DR register (second condition).

Note: The first data to be transmitted must be written into the data register after the ADDR flag is cleared, and before the next SCL rising edge, so that the time window to write the first data into the data register is less than t_{LOW}.

If this is not possible, a possible workaround can be the following:

1. Clear the ADDR flag
2. Wait for the OVR flag to be set
3. Clear OVR and write the first data.

The time window for writing the next data is then the time to transfer one byte. In that case, the master must discard the first received data.

2.4.6 I²C pulse missed

Description

When the I²C interface is used for long transmit/receive transactions, the MCU may return a NACK somewhere during the transaction instead of returning an ACK for all data. The received data may also be corrupted. In Master mode the I²C may not detect an incoming ACK. This is due to a weakness in the noise filter of the I/O pad which in certain conditions may cause the STM8 I²C to miss a pulse.

This limitation is fixed in revision X.

Workaround

Since data corruption is caused by noise generated by the CPU, CPU activity should be minimized during data reception and/or transmission. This is done by performing physical data transmission (Master mode) and reception (slave mode) in WFI state (wait for interrupt).

To allow the device to be woken up from WFI, I²C transmission and reception routines must be implemented through interrupt routines instead of polling mechanisms. Receive and transmit interrupts (received data processing) must be triggered only by the BTF bit flag (byte transfer finished) in the I2C_SR1 register. This flag indicates that the I²C is in stretched state (data transfers are stretched on the bus).

Clock stretching must be enabled to allow data transfers from the slave to be stopped and to allow the CPU to be woken up to read the received byte.

To recover from possible errors, periodically check if the I²C does not remain in busy state for too long (BUSY bit set in I2C_SR3 register). If so, it should be reinitialized.

Example of I²C slave code:

```
//...
//---------------------------------------------------------------
void main()
{
  Init_I2C(); // init I2C to use interrupts: ITBUFEN=0, ITEVTEN=1,
ITERREN=1
  while(1)
```

A fix is planned for the next silicon revision.

2.5 SPI peripheral limitations

2.5.1 Last bit too short if SPI is disabled during communication

Description

When the SPI interface operates in Master mode and the baud rate generator prescaler is equal to 2, the SPI is disabled during ongoing communications, and the data and clock output signals are switched off at the last strobing edge of the SPI clock.

As a consequence the length of the last bit is out of range and its reception on the bus is not ensured.

Workaround

Check if a communication is ongoing before disabling the SPI interface. This can be done by monitoring the BSY bit in the SPI_SR register.

2.5.2 Busy flag is not reliable when the SPI is a master simplex receiver

Description

When the master is receiver only, it provides the clock immediately after setting the SPE bit in the SPI_CR1 register. In this case, the clock is provided until the SPE bit is disabled, meaning that the SPI is always busy because it is in receiver mode only and continuously receives data from the clock. There is no need to read the BUSY bit to know the SPI status because as soon the SPI is enabled, it is BUSY.

Note: The SPE bit has no meaning when the SPI is in master receiver only mode.

Workaround

No workaround available. No fix planned.

2.6 ADC peripheral limitation

2.6.1 EOC interrupt triggered when AWDIE and EOCIE set to ‘1’

Description

When the analog watchdog is enabled and AWDIE and EOCIE are both set to ‘1’, the ADC interrupt should only be triggered when the conversion result exceeds one of the analog watchdog thresholds (see table 79 in the reference manual RM0016).

However, for the devices covered by this datasheet, the interrupt is triggered after each conversion, thus leading to a high interrupt load.

Workaround

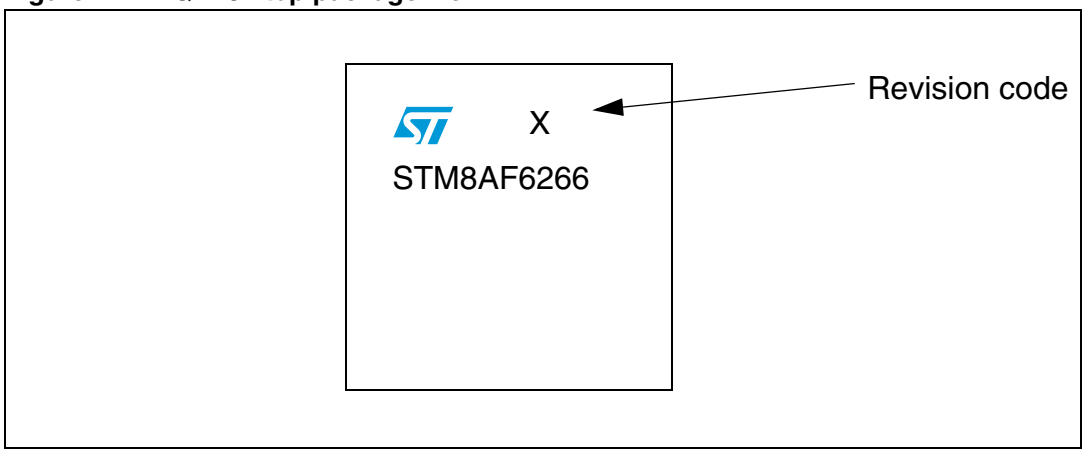
Set AWDIE to ‘1’ and EOCIE to ‘0’ instead and stop the conversions inside the ISR by resetting the CONT bit. However the latest conversion result having triggered the watchdog may be overwritten.

Appendix A Revision code on device marking

The revision code is marked on the package.

Figure 1 shows the marking for the LQFP32 package.

Figure 1. LQFP32 top package view



3 Revision history

Table 4. Document revision history

Date	Revision	Changes
31-Jan-2011	1	Initial release.
16-Dec-2011	2	Added product revision W. Updated disclaimer on last page.

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.

UNLESS EXPRESSLY APPROVED IN WRITING BY TWO AUTHORIZED ST REPRESENTATIVES, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

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.

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