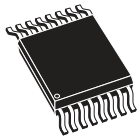
TDA7333N

RDS/RBDS processor

Features

- 3rd order high resolution sigma delta converter for MPX sampling
- Digital decimation and filtering stages
- Demodulation of european radio data system (RDS)
- Demodulation of USA radio broadcast data system (RBDS)
- Automatic group and block synchronization with flywheel mechanism
- Error detection and correction
- RAM buffer with a storage capacity of 24 RDS blocks and related status information
- Programmable interrupt source (RDS block A, B, or D, TA, TA EON)
- I^2C/SPI bus interface
- Input frequency range 4-21 MHz
- Power down mode
- 3.3 V power supply, 0.35 µm CMOS technology



TSSOP16

Description

The TDA7333N circuit is a RDS/RDBS signal processor, intended for recovering the inaudible RDS/RBDS informations which are transmitted on most FM radio broadcasting stations..

Table 1. Device summary

Order code	Operating temp. range, °C	Package	Packing
TDA7333N	-40 to +85	TSSOP16	Tube
TDA7333NTR	-40 to +85	TSSOP16	Tape & reel

Contents

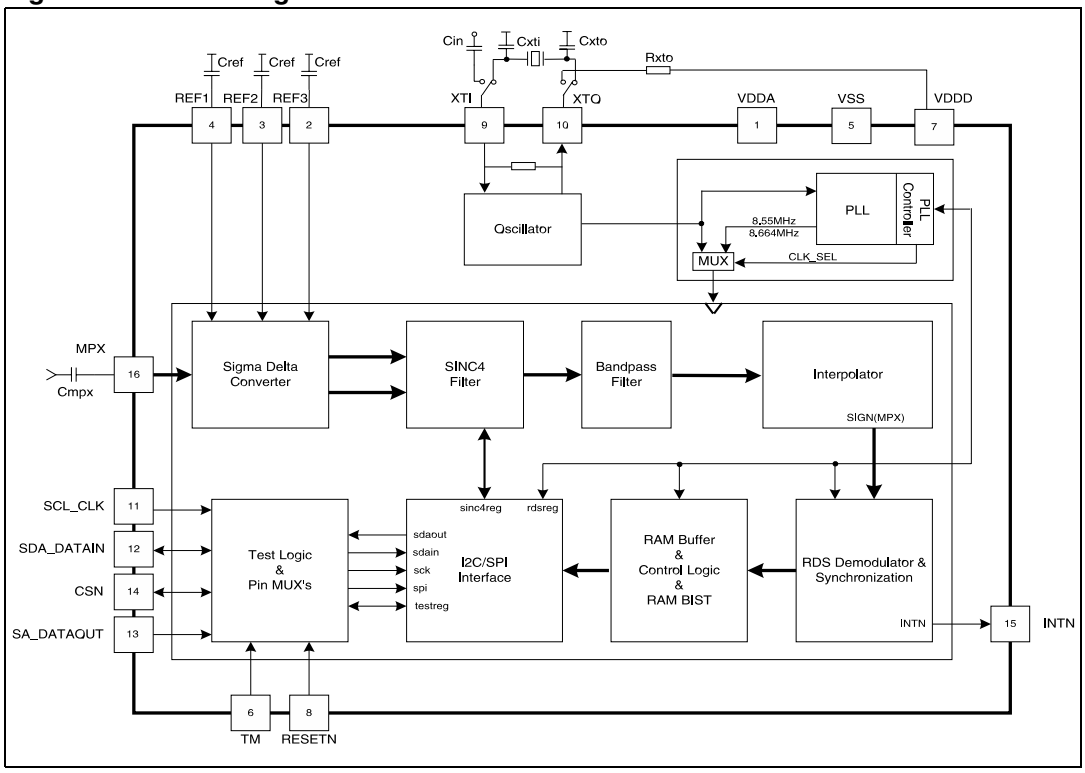
List of tables

List of figures

1 Block diagram and pin description

1.1 Block diagram

Figure 1. Block diagram



1.2 Pin description

Figure 2. Pin connection (top view)

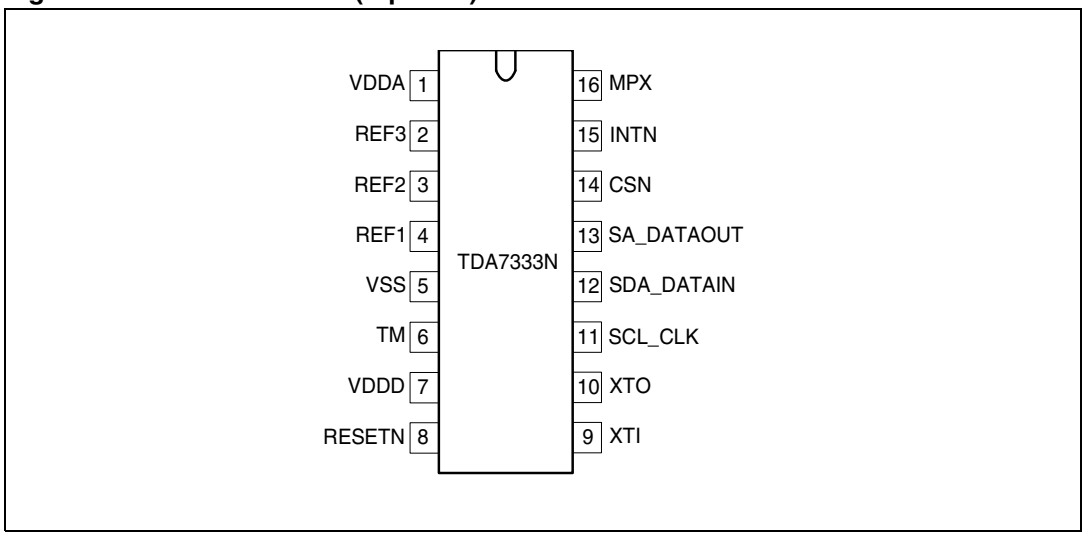


Table 2. Pin description

Pin #	Pin name	Function
1	VDDA	Analog supply voltage
2	REF3	Reference voltage 3 of A/D converter (2.65 V)
3	REF2	Reference voltage 2 of A/D converter (1.65 V)
4	REF1	Reference voltage 1 of A/D converter (0.65 V)
5	VSS	Common ground
6	TM	Testmode selection (scan test). Normal mode must be connected to gnd.
7	VDDD	Digital supply voltage
8	RESETN	External reset input (active low)
9	XTI	Oscillator input
10	XTO	Oscillator output
11	SCL_CLK	Clock signal for I^2C and SPI modes
12	SDA_DATAIN	Data line in I^2C mode, data input in SPI mode
13	SA_DATAOUT	Slave address in I^2C mode, data output in SPI mode
14	CSN	Chip select (1 = I^2C mode, 0=SPI mode)
15	INTN	Interrupt output (active low), prog. at buff.not empty,buff. full, block A,B,D ,TA, TA EON
16	MPX	Multiplex input signal

2 Electrical specifications

2.1 Absolute maximum ratings

Table 3. Absolute maximum ratings

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
V_{DD}	3.3 V power supply voltages	-	-0.5	-	4	V
V_{in}	Input voltage	5 V tolerant inputs	-0.5	-	5.5	V
V_{out}	Output voltage	5 V tolerant output buffers in tri-state	-0.5	-	5.5	V
T_{stg}	Storage temperature	-	-55	-	150	°C
V_{ESD}	ESD withstand voltage	Human body model	≥ ±2000			V
		Machine model	≥ ±200			V
		Charged device model, corner pins	≥ ±1000			V

2.2 General interface electrical characteristics

Table 4. General interface electrical characteristics

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
I_{il}	Low level input current	V_i = 0 V	-	-	1	µA
I_{ih}	High level input current	$V_i = V_{DD}$	-	-	1	µA
I_{ozFT}	Five volt tolerant tri-state output leakage without pull up/down device	V_o = 0 V or V_{DD}	-	-	1	µA
		V_o = 5.5 V	-	1	3	µA

2.3 Electrical characteristics

T_{amb} = -40 to +85 °C, V_{DDA}/V_{DDD} = 3.0 to 3.6 V, f_{osc} = 8.55 MHz, unless otherwise specified
V_{DDD} and V_{DDA} must not differ more than 0.15 V

Table 5. Electrical characteristics

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
Supply (pin 1,5,7)						
V_{DDD}	Digital supply voltage	-	3.0	3.3	3.6	V
V_{DDA}	Analog supply voltage	-	3.0	3.3	3.6	V
I_{DDD}	Digital supply current	Normal mode	-	14	-	mA
		Power down mode	-	< 1	-	µA
I_{DDA}	Analog supply current	Normal mode	-	11.7	-	mA
		Power down mode	-	< 1	-	mA

Table 5. Electrical characteristics (continued)

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
Digital inputs(pin 6,8,11,12,13,14)						
V_{il}	Low level input voltage	-	-	-	0.8	V
V_{ih}	High level input voltage	-	2.0	-	-	V
V_{ilhyst}	Low level threshold input falling	-	1.0	-	1.15	V
V_{ihhyst}	High level threshold input rising	-	1.5	-	1.7	V
V_{hst}	Schmitt trigger hysteresis	-	0.4	-	0.7	V
Digital outputs (pin 12,13,15) are open drains						
V_{oh}	High level output Voltage	Open drain, depends on external circuitry	-	n/a	-	V
V_{ol}	Low level output Voltage	I_{ol} =4 mA, takes into account 200 mV drop in the supply voltage	-	-	0.4	V
Analog inputs (pin 16)						
V_{MPX}	Input Range of MPX Signal	-	-	-	0.75	Vrms
-	Input Impedance of MPX pin	-	-	55k	-	Ohm
C_{ref}	Blocking Capac. of REF Pins	Electrolyte capacitor parallel to ceramic capacitor	-	2.2	-	µF
			-	100	-	nF
Crystal/oscillator parameters						
f_{osc}	Quartz frequency	-	4	10.25	21	MHz
f_{oto}	Total quartz frequency tolerance	T_{amb} = -40 to 85 °C	-	-	100	ppm
t_{su}	Start up time	-	-	-	10	ms
g_m	Oscillator transconductance	-	0.0006	-	-	A/V
C_{xti},C_{xto}	Load capacitance	With crystal between XTI and XTO	-	16	-	pF
External XTI input frequency mode (pin 9)						
f_{exti}	Externaly applied XTI frequency	-	4	10.25	21	MHz
V_{xti}	XTI input voltage	With R_{xto} = 3.3 kOhm, and f_{exti} = 10.25 MHz	220	-	-	mVpp
C_{in}	Coupling capacitor for external clock frequency	-	-	100	-	pF
R_{xto}	XTO pull up to VDDD	-	-	3.3	-	kΩ

Table 5. Electrical characteristics (continued)

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
PLL parameters						
f_{vco}	VCO range	-	150	-	250	MHz
f_{vin}	VCO input range	-	4	-	21	MHz
t_{lock}	PLL lock time	-	-	-	500	µs
I_{DF}	Input divide factor	-	1	-	32	-
O_{DF}	Output divide factor	-	2	-	32	-
M_F	Integer multiplication factor	-	10	-	128	-
FRA	Fractional multiplication factor	$FRA/2^{14}$	0	-	2^{14}	-
Bandpass filter						
f_p	Pass-band frequencies	-	55.6	-	58.4	kHz
R_p	Pass-band ripple	-	-0.5	-	+0.5	dB
f_{stop}	Stop-band corner frequencies	-	53	-	61	kHz
R_s	Stop-band attenuation	-	-	-43	-	dB
I^2C (@ fsys = 8.55/8.664 MHz)						
f_{I2C}	Clock frequency in I^2C mode	-	-	-	400	kHz
ts_{udat}	Data setup time	-	250	-	-	ns
SPI (@ fsys = 8.55/8.664 MHz)						
f_{SPI}	Clock frequency in SPI mode	-	-	-	1	MHz
t_{ch}	Clock high time	-	450	-	-	ns
t_{cl}	Clock low time	-	450	-	-	ns
t_{csu}	Chip select setup time	-	500	-	-	ns
t_{csh}	Chip select hold	-	500	-	-	ns
t_{odv}	Output data valid	-	-	-	250	ns
t_{oh}	Output hold	-	0	-	-	ns
t_d	Deselect time	-	1000	-	-	ns
t_{su}	Data setup time	-	200	-	-	ns
t_h	Data hold time	-	200	-	-	ns

3 Functional description

3.1 Overview

The new RDS/RBDS processor contains all RDS/RBDS relevant functions on a single chip. It recovers the inaudible RDS/RBDS information which are transmitted on most FM radio broadcasting stations.

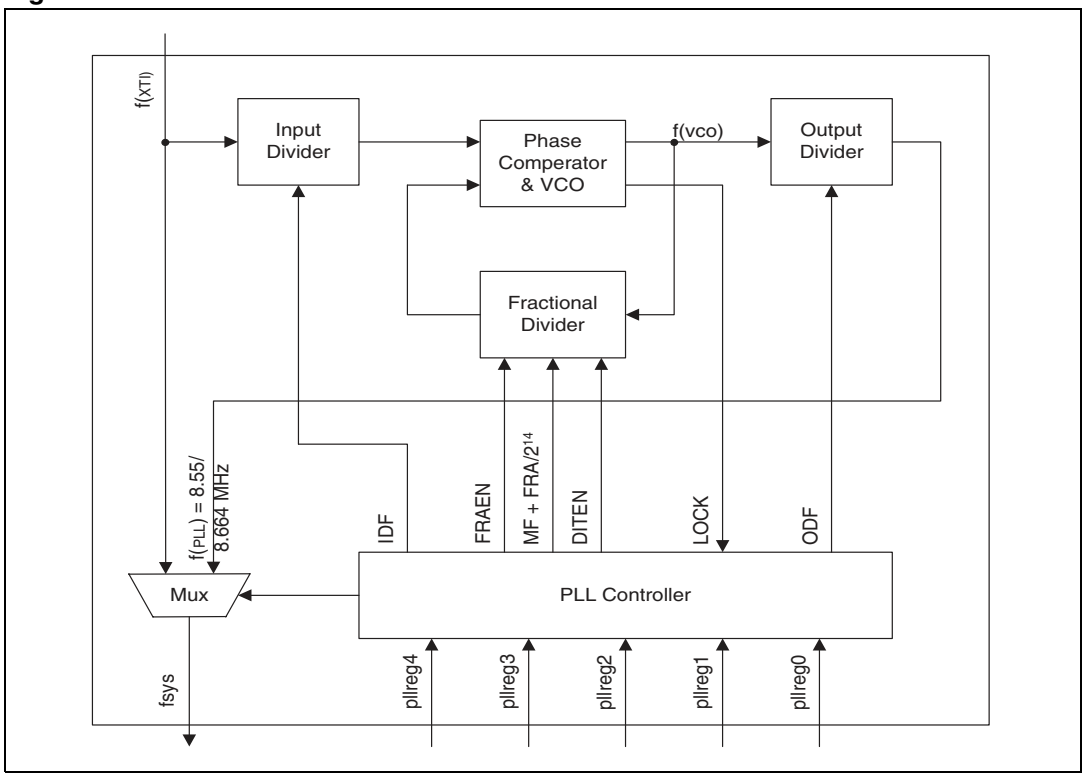
The oscillator frequency can be derived from the tuner with typical value of 10.25 MHz . The device can operate with frequencies in the range of 4-21 MHz. Therefor the fractional PLL must be initialized through I^2C/SPI interface to generate the internal 8.55 MHz or 8.664 MHz reference clock with a freq. tolerance of ±0.7 kHz.

Due to an integrated 3rd order sigma delta converter, which samples the MPX signal, all further processing is done in the digital. After filtering the highly over sampled output of the A/D converter, the RDS/RBDS demodulator extracts the RDS data clock, RDS data signal and the quality information. A next RDS/RBDS decoder will synchronize the bit wise RDS stream to a group and block wise information. This processing includes an error detection and error correction algorithm. In addition, an automatic flywheel control avoids overheads in the data exchange between the RDS/RBDS processor and the host.

The device operates in accordance with the CENELEC Radio Data System (RDS) specification EN50067.

3.2 Fractional PLL

Figure 3. Fractional PLL



The fractional PLL (*Figure 3*) is used to generate from the XTI input clock one of the two possible system clocks (fsys) 8.55 MHz or 8.664 MHz. For this a setting for the input diver factor (IDF), output divider factor (ODF), multiplication factor (MF) and fractional factor (FRA) must be found (max. fsys tolerance ±0.7 kHz). For fractional mode an additional dither can be enabled (DITEN) to eliminate tones in the PLL output clock. The fractional mode can be disabled (FRAEN) if not needed.

The system clock (fsys) is equal to the XTI input clock after reset. After the PLL is locked, the system clock will switch automatically to the PLL output clock. Then the SPI/I^2C can be used at the maximum speed of 400 kbits/s.

The initialization of the PLL must be done only once after hardware reset. After PLL locking the RDS functionality can be used regardless of the PLL.

All clocks can be disabled in power down mode, which can be exited only by a hardware reset (pin RESETN).

3.3 Sigma delta converter

The sigma delta modulator is a 3rd order (second order-first order cascade) structure. Therefore a multi bit output (2 bit streams) represents the analog input signal. A next digital noise canceller will take the 2 bit streams and calculates a combined stream which is then fed to the decimation filter. The modulator works at a sampling frequency of fsys/2. The over sampling factor in relation to the band of interest (57 kHz ± 2.4 kHz) is 38.

3.4 Demodulator

The demodulator includes:

- RDS quality indicator with selectable sensitivity
- Selectable time constant of 57 kHz PLL
- Selectable time constant of bit PLL
- Time constant selection done automatically or by software

The demodulator is fed by the 57 kHz bandpass filter and interpolated multiplex signal. The input signal passes a digital filter extracting the sinus and cosinus components, to be used for further processing.

The sign of both channels are used as input for the ARI indicator and for the 57 kHz PLL.

A fast ARI indicator determines the presence of an ARI carrier. If an ARI carrier is present, the 57 kHz PLL is operating as a normal PLL, else it is operating as a Costas loop.

One part of the PLL is compensating the integral offset (frequency deviation between oscillator and input signal).

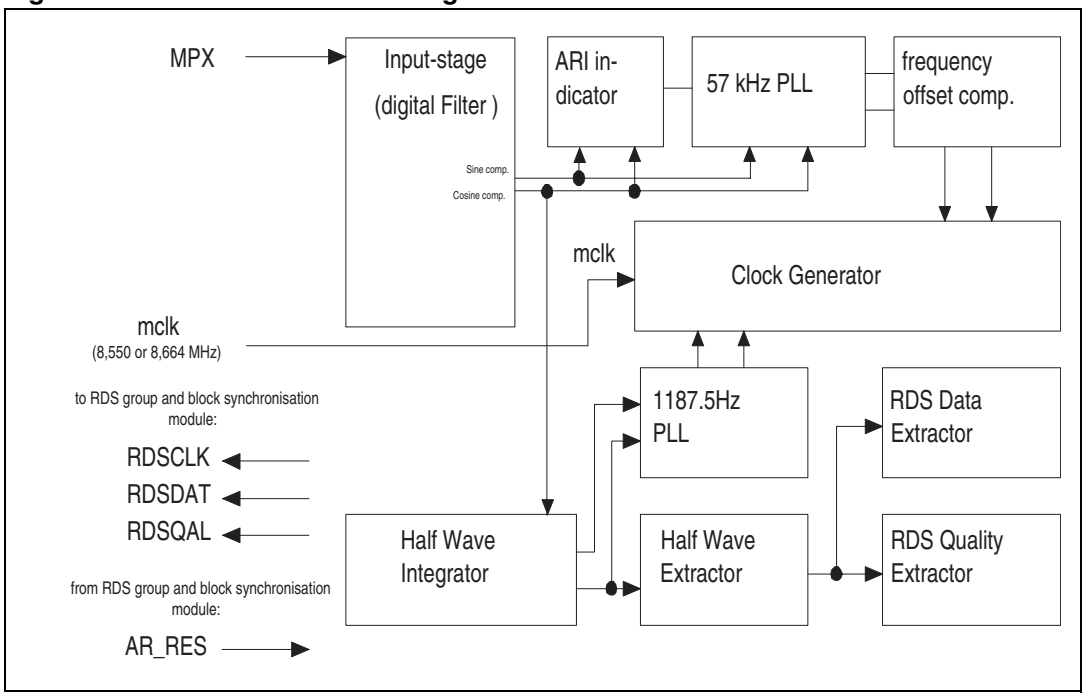
One channel of the filter is fed into the half wave integrator. Two half waves are created, with a phase deviation of 90 degrees. One wave represents the RDS component, whereas the other wave represents the ARI component.

The sign of both waves are used as reference for the bit PLL (1187.5 Hz).

The RDS wave is then fed into the half wave extractor. This leads into an RDS signal, which after integration and differential decoding represents the RDS data.

In a similar way a quality bit can be calculated. This is useful to optimize error correction.

Figure 4. Demodulator block diagram



The module needs a fixed clock of 8.55 MHz. Optionally an 8.664 MHz clock may be used by setting the corresponding bit in rds_bd_ctrl register (refer to *Section 3.8.6*).

In order to optimize the error correction in the group and block synchronization module, the sensitivity level of the quality bit can be adjusted in four steps with “qsens” bits rds_bd_ctrl[5:4]. Only bits marked as bad by the quality bit are allowed to be corrected in the group and block synchronization module. If an error correction is done on a good marked RDS bit, the “data_ok” bit rds_corrp[1] will not be set (refer to *Section 3.8.3*).

The RDS bit demodulator can be controlled by the bits 1-6 of rds_bd_ctrl register for example to select 57 kHz PLL and 1187.5 Hz PLL time constant. This is useful in order to achieve a fast synchronization after a program resp. frequency change (fast time constant) and to get a maximum of noise immunity after synchronization (slow time constant).

The user may choose between 2 possibilities via bit rds_bd_ctrl[1]:

a) Hardware selected time constant - In this case both pll time constants are reset to the fastest one, with a reset from the group and block synchronization module, or if the software decides to resynchronize by setting “ar_res” rds_int[5] (refer to page 18). Then both PLLs increase automatically to the slowest time constant. This is done in four steps within a total time of 215.6 ms (256 RDS clocks).

b) Software selected time constant - In this case the time constant of both PLL can be selected individually by software (rds_bd_ctrl[4:2]). Four time constants (5 ms, 15 ms, 35 ms, 76 ms) can be set independently for 1187.5 Hz PLL and two time constants (2 ms, 10 ms) for the 57 kHz PLL.

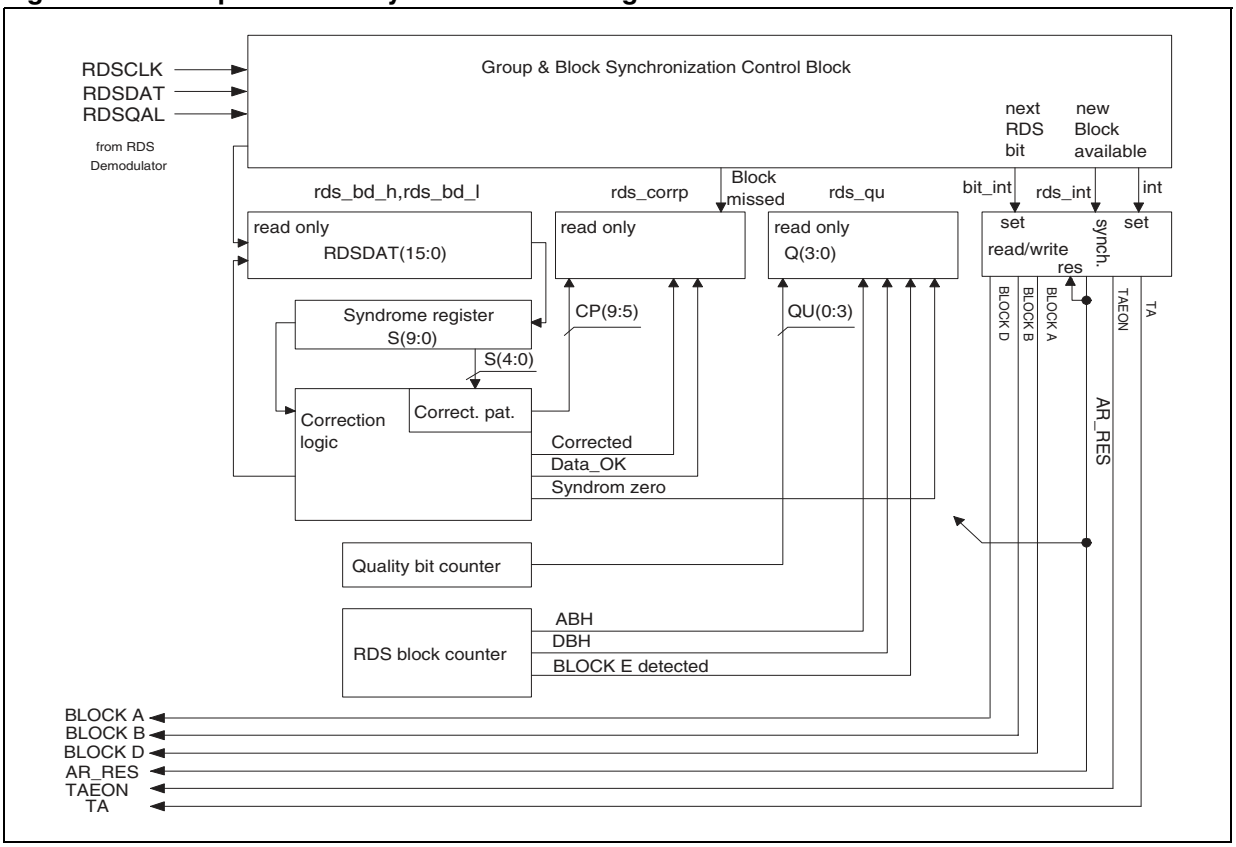
The sensitivity of the quality bit can be adjusted to four levels with the “qsens1” and “qsens0” rds_bd_ctrl[6:5] bits. “qsens1 = 0” and “qsens0 = 0” means minimum sensitivity, “qsens1 = 1” and “qsens0 = 1” maximum sensitivity.

3.5 Group and block synchronization module

The group and block synchronization module has the following features:

- Hardware group and block synchronization
- Hardware error detection
- Hardware error correction, using quality bit information to indicate bad corrections
- Hardware synchronization flywheel
- TA, TAEON information extraction
- Reset by software “ar_res”, which resets also RAM buffer addresses and RDS demodulator

Figure 5. Group and block synchronization diagram



This module is used to acquire group and block synchronization of the received RDS data stream, which is provided in a modified shortened cyclic code. For theory and implementation of modified shortened cyclic code and error correction, please refer to CENELEC Radio Data System (RDS) specification EN50067.

Group and block synchronization module can detect and correct five bit error burst in the data stream. If an error correction is done on a good quality marked RDS bit, the “data_ok” bit rds_corrp[1] won’t be set (refer to page 22). Before error correction, the five MSBs of the syndrome register are stored in the “cp” bits rds_corrp[7:3].

If the five LSBs of the syndrome register are zero, the “cp” pattern is used for error correction. After that operation the syndrome must become zero for valid RDS data. The

type of error can be measured with the five “cp” bits in order to classify the reliability of the correction. Each bit set within “cp” means that one bit was corrected.

The two RDS data bytes rds_bd_h[7:0] and rds_bd_l[7:0] are available at the I^2C/SPI interface together with status bits rds_corrp[7:0] and rds_qu[7:0] giving reliability information of the data (refer to *Figure 5*). rds_int[7:0] bits are used for interrupt and group and block synchronization control. A software reset “ar_res” rds_int[5] can be used to force resynchronization.

An endless 2 bit block counter (A, B, C or C’, D, A, B...) increments one step if a new RDS block was received. During synchronization the block counter is set to the first identified valid RDS block. Then every next RDS block must be of that type which is indicated by the block counter “blk” rds_qu[3:2]. If this is not true, then the syndrome becomes not zero (indicated by “synz” bit rds_qu[0]) and the “data_ok” bit rds_corrp[1] is not set. In case of USA BRDS, four consecutive E blocks can be received which are indicated by the “e” bit rds_qu[1].

The quality bit counter rds_qu[7:4] counts the bad quality marked RDS bits within a RDS block.

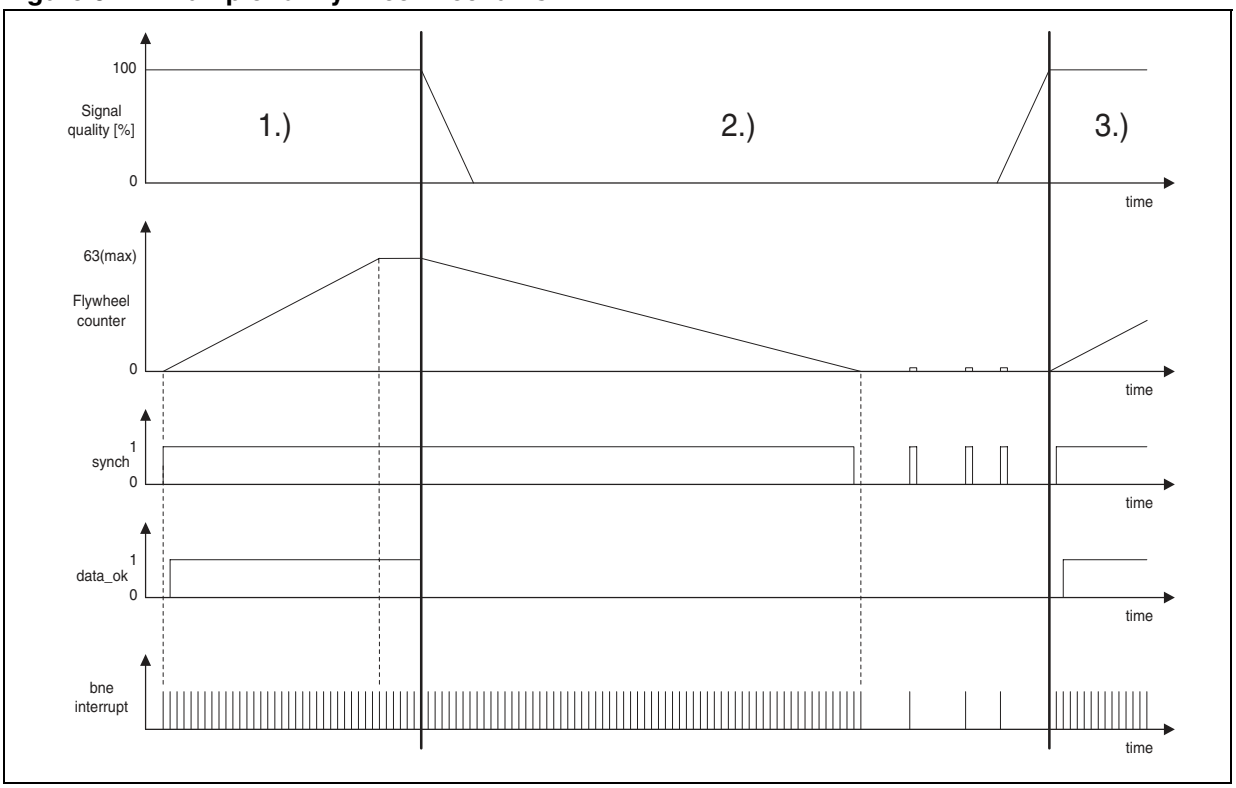
The group and block synchronization module extracts also TA, TAEON information and detects blocks types A, B, D (refer to page 21) which can be used as interrupt sources.

The TA interrupt is performed in two cases: If within block B the group 0A or 0B is indicated and the TA bit is set or if within block B group 15B is indicated and the TA bit is set. The TAEON interrupt is performed, if within block B group 14B is indicated and the TA bit is set.

The interrupts can be recognized on the interrupt flag “int” rds_int[0] (refer to *Section 3.8.1*). The external open drain pin INTN (15) is the inverted version of the “int” flag.

3.6 Flywheel mechanism

Figure 6. Example for flywheel mechanism



Within group and block synchronization control block a 6 bit (64 states) flywheel counter is implemented to control RDS synchronization. After reset or a forced resynchronization by setting “ar_res” bit rds_int[5], this counter increments from zero to one, if a valid RDS block was detected. Valid means the syndrome has to be zero (“synz” = 1 rds_qu[0]) without any error corrections done on good quality marked RDS bits. Then the RDS module is synchronized. This is indicated by “synch” bit rds_int[4] which is set if the flywheel counter is greater than zero. Every valid consecutive RDS block (A, B, C or C’, D, A, B...) increments the flywheel counter by two.

If the next consecutive RDS block has its syndrome not zero, or corrections are done on good quality marked RDS bits, then the flywheel counter decrements by one. If the flywheel counter becomes zero, then a new RDS block synchronization will be performed. If blocks of type E are detected (indicated by “e” bit rds_qu[1]), then the flywheel counter will be not modified, because in case of European RDS, block E is an error but not in case of USA BRDS. This means E blocks are treated as neutral in this RDS/BRDS implementation.

The “data_ok” bit rds_corrp[1] is set only, if the flywheel counter is greater than two, the syndrome of the detected RDS block is zero and if no error corrections are done on good quality marked RDS bits.

Figure 6 shows an example for the flywheel mechanism.

The first diagram shows the relative signal quality of 26 received RDS bits. 100 % means that the last received 26 RDS bits are all marked as good by the demodulator and 0% that all are marked as bad.

The second diagram gives information about the flywheel counter status. The counter value could be between 0 and 63.

The next two charts showing the bits “synch” rds_int[4] and “data_ok” rds_corrp[1] (refer to *Section 3.8.1* and *Section 3.8.3*).

The last graph indicates every generated buffer not empty (bne) interrupt. After each interrupt the RDS data will be read out from the RAM buffer (within 22 ms), before next RDS block is written into. This is done to reset the interrupt flag “int” rds_int[0] each time. Further the “syncw” bit rds_bd_ctrl[0] is set to one, to store only synchronized RDS blocks (refer to *Section 3.8.6*).

The following case is considered now: First the receiving condition is good (section 1), then it is going to be worse (section 2) because of entering a tunnel, after leaving it is going to be better again (section 3).

Section 1: After power up or resynchronization (“ar_res”, rds_int[5]), the first recognized RDS block is stored in the RAM buffer and generates an “bne” interrupt. At the same time “synch” bit rds_int[4] is set to one. With the next stored RDS block the “data_ok” bit rds_corrp[1] is set, because the flywheel counter becomes greater than two. With every next RDS block the flywheel counter increments by two, until the upper margin of 63 is reached.

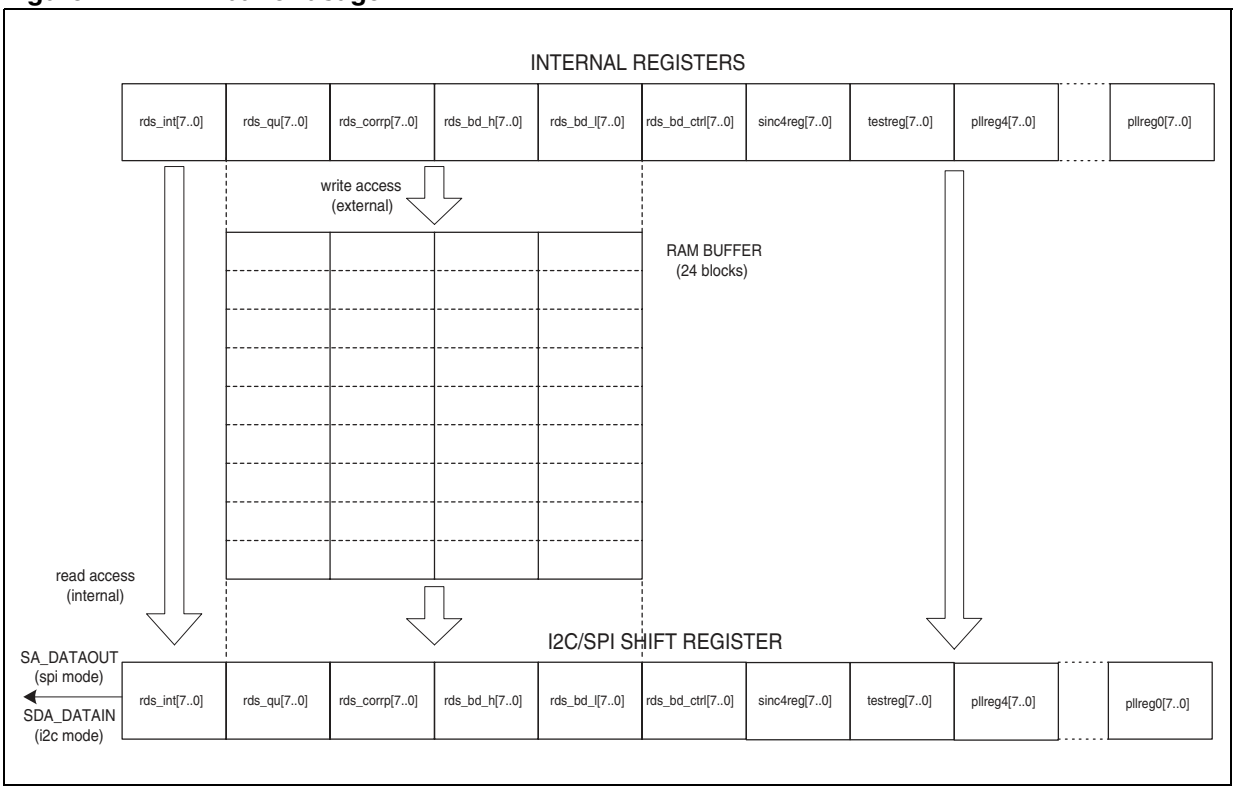
Section 2: Because of entering a tunnel, the demodulator increases bad marked RDS bits until all are marked as bad. The flywheel counter decrements by one after each new RDS block because of error corrections done on good marked RDS bits or because the syndrome of the expected block was not zero after error correction. The “data_ok” bit rds_corrp[1] is set to zero whenever the flywheel counter decrements. Note that the synchronization flag “synch” rds_int[4] is set and the interrupt is performed after every expected RDS block, until the flywheel counter is zero. Then the RDS is desynchronized. Now spurious interrupts could occur because of random RDS blocks detected during resynchronization process. If the time of receiving bad signal is shorter than the decreasing time of the flywheel counter, then the RDS will keep its synchronization and stores RDS data every 22 ms.

Section 3: After leaving the tunnel, the signal is getting better and the RDS will be synchronized again as described in section 1.

3.7 RAM Buffer

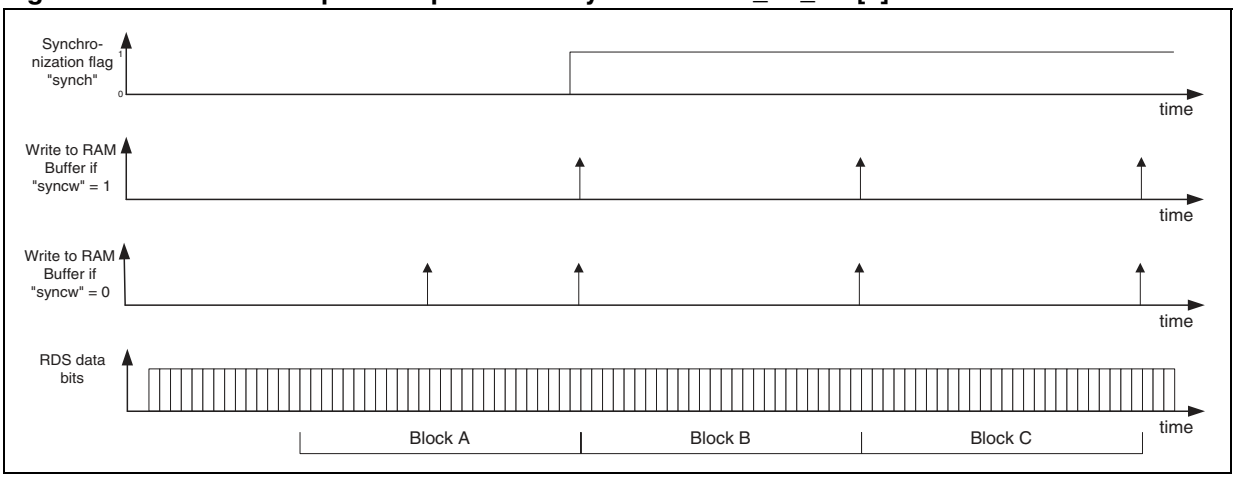
The RAM buffer can store up to 24 RDS blocks (rds_bd_h[7:0] and rds_bd_l[7:0]) with their related information (rds_qu[7:0] and rds_corrp[7:0]) (*Figure 7*):

Figure 7. RAM buffer usage



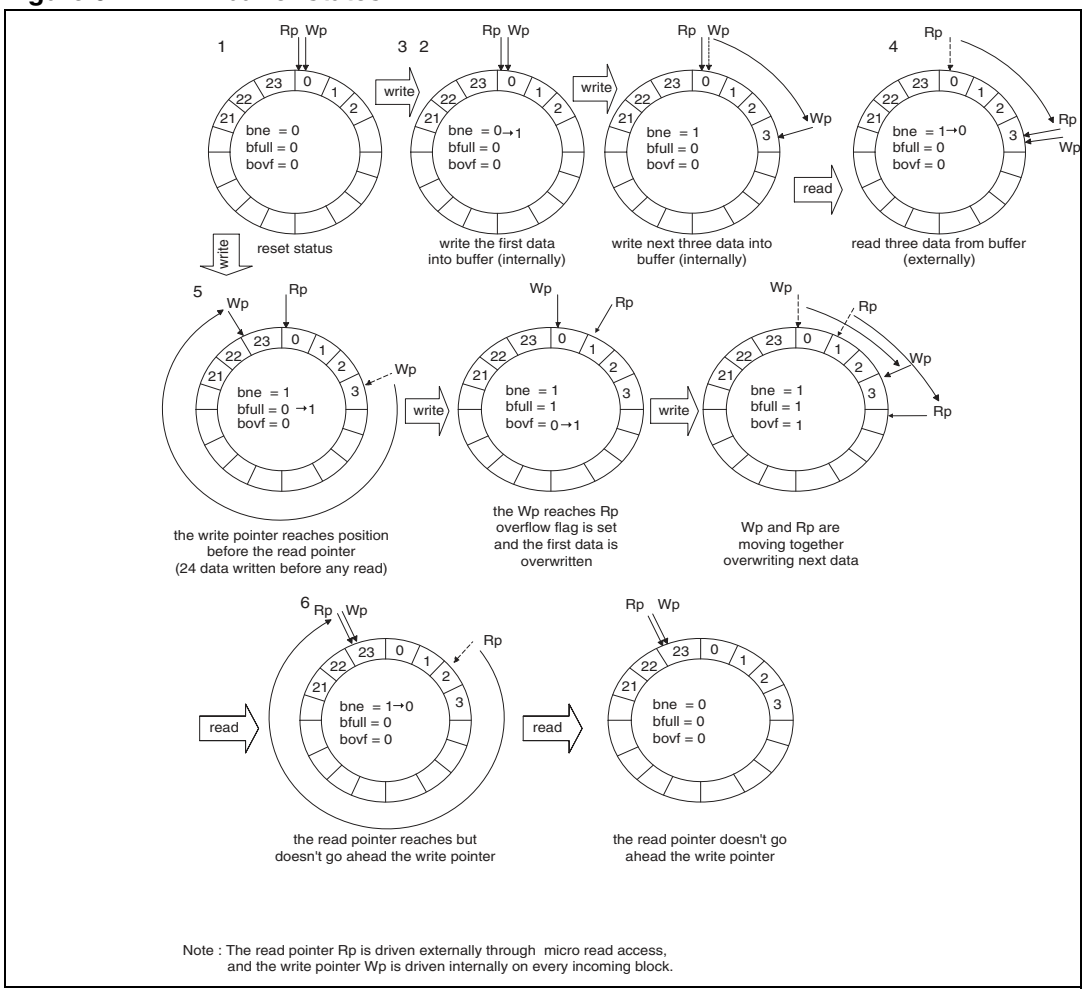
After power up, or after resynchronization by setting “ar_res” rds_int[5] to one, incoming RDS blocks are stored in the RAM buffer when synchronization has been established (*Figure 8*). But if the bit “syncw” rds_bd_ctrl[0] (refer to *Section 3.8.6*) is cleared, every received RDS block is stored, also without synchronization. This means if the RDS is not synchronized, every received consecutive 26 RDS data bits are treated as a RDS block.

Figure 8. RAM buffer update depends on “syncw” bit rds_bd_ctrl[0]



The RAM buffer is used as a circular FIFO (*Figure 9*). If more than 24 blocks are written, the oldest data will be overwritten. One level of the buffer consists of 4 bytes (2 information bytes, 2 RDS data bytes). If less than 4 bytes of the RAM buffer are read out from the master via the SPI or I^2C interface, the buffer address will not be incremented.

Figure 9. RAM buffer states



The different states of the buffer are indicated with the help of following flags:

- "bne", buffer not empty. It is set as soon as one RDS block is written in the buffer, and reset when reading rds_int register. This flag is a bit of rds_int register, it is also an interrupt source (refer to *Section 3.8.1*).
- "bfull", buffer full. It is set when 24 RDS blocks have been written, that is to say that there is about 20 ms to read out the buffer content before an overflow occurs. This flag is an interrupt source.
- "bovf", buffer overflow. It is set if more than 24 RDS blocks are written. This flag is a bit of register rds_corrp (refer to *Section 3.8.3*) and is cleared only by reading the whole buffer (24 blocks).

An address reset of the RAM buffer can be performed by writing a 1 to "ar_res" bit in rds_int register, it also forces a resynchronization.

Figure 9 describes the different states of the buffer with corresponding flags values:

1. This is the reset state, read (Rp) and write pointer (Wp) pointing at the same location 0. The buffer is empty.
2. After the first buffer write operation, Wp points to the last written data (0, it is not incremented) and the flag “bne” (buffer not empty) is set.
3. After next buffer write operation, Wp points to the last written data (3, incremented address).
4. After buffer read operation, Rp points to incremented address (data to be read on the next read cycle), following the Wp. As soon as Rp reaches the Wp (of value 3), it is not incremented to 4 and flag “bne” is reset. Rp never goes ahead the Wp.
5. If the buffer is full (i.e. 24 blocks have been written before any read), flag “bfull” is set. If no read operation is performed, on next write operation “bovf” (buffer overflow) is set, and each subsequent write operation will overwrite the oldest data of the RAM buffer. Rp is moved in front of the Wp.
6. If the whole content of the buffer has already been read, subsequent read operation will always read the last written location - Rp never goes ahead the Wp.

3.8 Programming through serial bus interface

The serial bus interface is used to access the different registers of the chip. It is able to handle both I^2C and

SPI transfer protocols, the selection between the two modes is done thanks to the pin CSN:

– if the pin CSN is high, the interface operates as an I^2C bus.
– if the pin CSN is asserted low, the interface operates as a SPI bus.

In both modes, the device is a slave, i.e the clock pin SCL_CLK is only an input for the chip.

Depending on the transfer mode, external pins have alternate functions as following:

Table 6. External pins alternate functions

Pin	Function in SPI mode (CSN=0)	Function in I^2C mode (CSN=1)
SCL_CLK	CLK (serial clock)	SCL (serial clock)
SDA_DATAIN	DATAIN (data input)	SDA (data line)
SA_DATAOUT	DATAOUT (data output)	SA (slave address)

13 registers are available with read or read/write access:

Table 7. Registers description

Register	Access rights	Function
rds_int[7:0] (see *3.8.1*)	read/write	interrupt source setting, synch., bne information
rds_qu[7:0] (see *3.8.2*)	read	quality counter, actual block name
rds_corrp[7:0] (see *3.8.3*)	read	error correction status, buffer ovf information
rds_bd_h[7:0] (see *3.8.4*)	read	high byte of current RDS block
rds_bd_l[7:0] (see *3.8.5*)	read	low byte of current RDS block

Table 7. Registers description (continued)

Register	Access rights	Function
rds_bd_ctrl[7:0] (see *3.8.6*)	read/write	frequency, quality sensitivity, demodulator pll settings
sinc4reg[7:0] (see *3.8.7*)	read/write	sinc4 filter settings (for internal use only)
testreg[7:0] (see *3.8.8*)	read/write	test modes (for internal use only)
pllreg4[7:0] (see *3.8.9*)	read/write	PLL control register 4
pllreg3[7:0] (see *3.8.10*)	read/write	PLL control register 3
pllreg2[7:0] (see *3.8.11*)	read/write	PLL control register 2
pllreg1[7:0] (see *3.8.12*)	read/write	PLL control register 1
pllreg0[7:0] (see *3.8.13*)	read/write	PLL control register 0

The meaning of each bit is described below:

3.8.1 rds_int register

Figure 10. rds_int registe

rds_int	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	0	0	0	0	0	0	0	0
bit name	write	bne	ar_res	synch	itsrc2	itsrc1	itsrc0	int
access	r/w	r	r/w	r	r/w	r/w	r/w	r

int	Interrupt bit. It is set to one on every programmed interrupt. It is reset by reading rds_int register. The inverted version is also externally available on RDSINT pin.
itsrc0 itsrc1 itsrc2	itsrc[2:0] selects interrupt source (1). Block A, B, D and TA, TA EON interrupts only if "synch" =1.
synch	Synchronization information (refer to pages 13-15). 1: The module is already synchronized. 0: The module is synchronizing.
ar_res	It is used to force a resynchronization. If it is set to one, the RDS modules are forced to resynchronization state and the RAM buffer address is reset. This bit is reset automatically. It is read always as zero.
bne	Buffer not empty. 1: At least one block is present in the RAM buffer. 0: The RAM buffer is empty.
bne	rds_int, rds_bd_ctrl and pllreg4-0 write order. **This bit is only used in SPI mode and is read always as zero**. 1: Update of rds_int, rds_bd_ctrl and pllreg4-0 with data shifted in. 0: No update of rds_int, rds_bd_ctrl and pllreg4-0.

(1)

interrupt source	itsrc2	itsrc1	itsrc0
no interrupt	0	0	0
buffer not empty	0	0 1	
buffer full	0	1 0	
block A	0	1 1	
block B	1	0 0	
block D	1	0 1	
TA	1	1 0	
TA EON	1	1 1	

(1) If the interrupt source is changed form block A, B, D, TA, TA EON to another one "no interrupt" must be set before to clear the previous interrupt acknowledge.

3.8.2 rds_qu register

Figure 11. rds_qu register

rds_qu	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	0	0	0	0	0	0	0	0
bit name	qu3	qu2	qu1	qu0	blk1	blk0	e	synz
access	r	r	r	r	r	r	r	r

It indicates if error correction was successful. 1: The syndrome was zero after error correction. 0: The syndrome did not become zero and therefore the error correction was not successful.
1: Block E is detected. This indicates a paging block which is deÞned in the RBDS speciÞcation used in the United States of America. 0: An ordinary RDS block A, B, C, C« or D is detected, or no valid syndrome was found.
Bit 0 of block counter (2).
bit 1 of block counter (2).
bit 0 of quality counter (3).
bit 1 of quality counter (3).
bit 2 of quality counter (3).
bit 3 of quality counter (3).

(2)

block name	blk1	blk0
block A	0	0
block B	0	1
block C,C'	1	0
block D	1	1

(2) If "syncw" =1 of rds_bd_ctrl register, the block counter indicates the expected RDS block.
(3) qu[3...0] counts the number of bits (max.16) which are marked as bad by the demodulator within each RDS block. It could be used as a quality information, indicating the maximum number of bits which are allowed to be corrected.

3.8.3 rds_corrp register

Figure 12. rds_corrp register

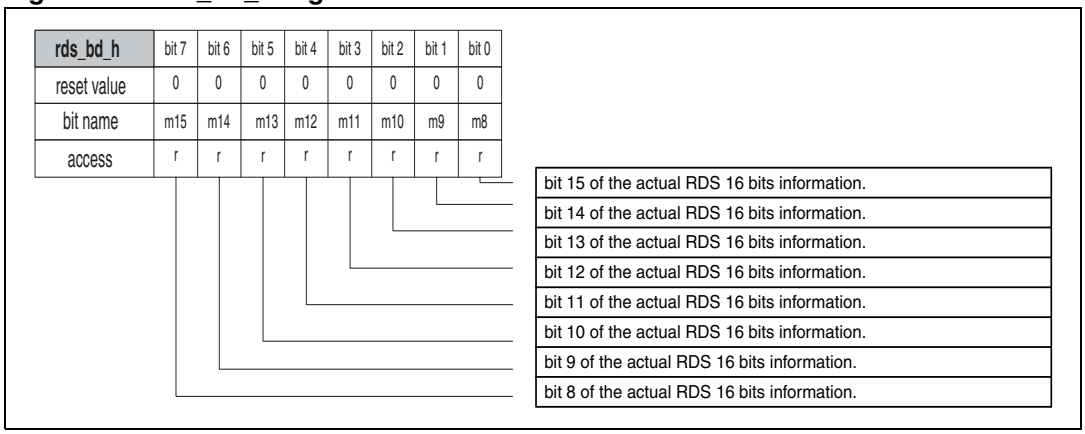
rds_corrp	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	0	0	0	0	0	0	0	0
bit name	cp9	cp8	cp7	cp6	cp5	correct	dat_ok	bovf
access	r	r	r	r	r	r	r	r

Buffer overflow 1: More than 24 blocks have been written into the buffer. 0: No buffer data has been overwritten.
Information if the current RDS data could be used. 1: A correct syndrome was detected and no error correction was done on a good quality marked RDS bit and the flywheel counter is greater than 2 (RDS data is OK). 0: The syndrome was wrong, or an error correction was done on a good quality marked RDS bit, or the flywheel counter is lower than 3 (RDS data is not OK).
It is an information about error correction. 1: An error correction was done. 0: The actual RDS block is detected as error free.
bit 5 of the syndrome register(4).
bit 6 of the syndrome register(4).
bit 7 of the syndrome register(4).
bit 8 of the syndrome register(4).
bit 9 of the syndrome register(4).

(4) (Refer to CENELEC Radio Data System speciÞcation EN50067, ANNEX B). When bits 0...4 of the syndrome register are zero, a possible error burst is detected. With help of the correction pattern (bits 5...9 of the syndrome register), the type of error can be measured, in order to classify the reliability of the correction.

3.8.4 rds_bd_h register

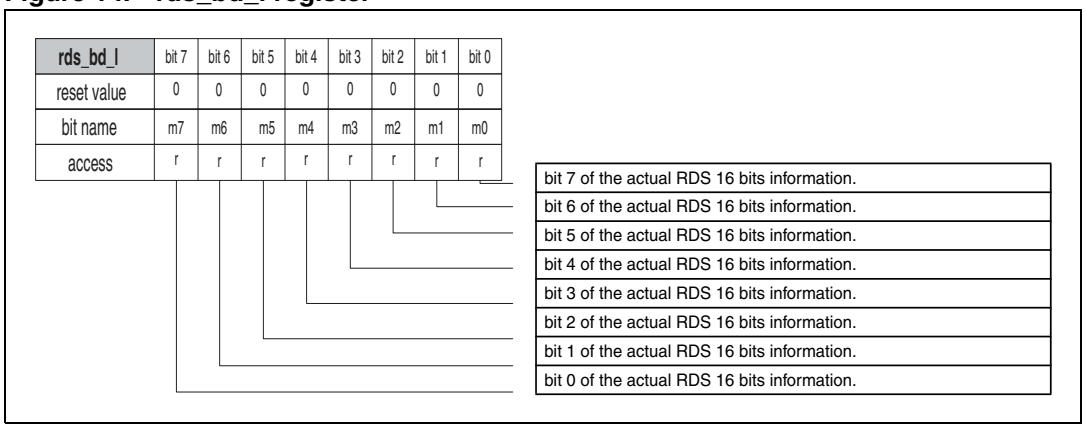
Figure 13. rds_bd_h registe



rds_bd_h	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	0	0	0	0	0	0	0	0
bit name	m15	m14	m13	m12	m11	m10	m9	m8
access	r	r	r	r	r	r	r	r

3.8.5 rds_bd_l register

Figure 14. rds_bd_l register



rds_bd_l	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	0	0	0	0	0	0	0	0
bit name	m7	m6	m5	m4	m3	m2	m1	m0
access	r	r	r	r	r	r	r	r

3.8.6 rds_bd_ctrl register

Figure 15. rds_bd_ctrl register

rds_bd_ctrl	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	0	0	0	0	0	0	0	1
bit name	freq	qsens1	qsens0	pllb1	pllb0	pllf	shw	syncw
access	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Write into buffer if synchronized (refer to page 10-12) (8) 1: Write into buffer only if synchronized (reset value). 0: Write into buffer any incoming RDS block.
Select PLL time constants by software or hardware (8) 1: Software. Time constants are selected by pllb[1:0] respectively pllf. 0: Hardware (reset value). Time constants automatically increase after reset or resynchronization.
Set the 57 kHz pll time constant (5) (8).
Bit 0 of 1187.5 Hz pll time constant (6) (8).
Bit 1 of 1187.5 Hz pll time constant (6) (8).
Bit 0 of quality sensitivity (7) (8).
Bit 1 of quality sensitivity (7) (8).
Select internal master clock frequency (fsys): 1: 8.664 MHz. 0: 8.55 MHz (reset value).

(5)

pllf	lock time needed for 90 deg deviation
0	2 ms
1	10 ms

(6)

pllb1	pllb0	lock time needed for 90 deg deviation
0	0	5 ms (reset status)
0	1	15 ms
1	0	35 ms
1	1	76 ms

(7) Select sensitivity of quality bit.
00: minimum (reset value)
11: maximum

(8) Bit 5 "ar_res" of rds_int register will clear the bits 0-6 of the rds_bd_ctrl register.

3.8.7 sinc4reg register

Figure 16. sinc4reg register

sinc4reg	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	0	0	0	0	0	0	0	0
bit name	-	-	-	-	-	-	-	-
access	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

sinc4reg register is for internal use only. For application this register must be always Þlled with æros.

3.8.8 testreg register

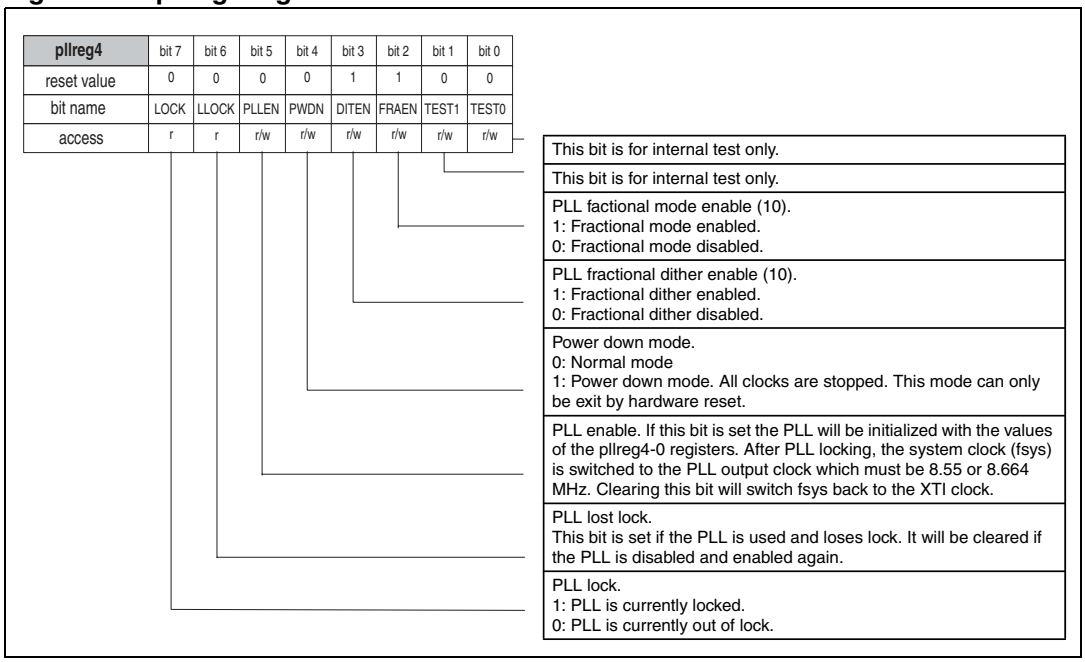
Figure 17. testreg register

testreg	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	0	0	0	0	0	0	0	0
bit name	-	-	-	-	-	-	-	-
access	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

testreg register is for internal use only. For application this register must be always Þlled with æros.

3.8.9 pllreg4 register

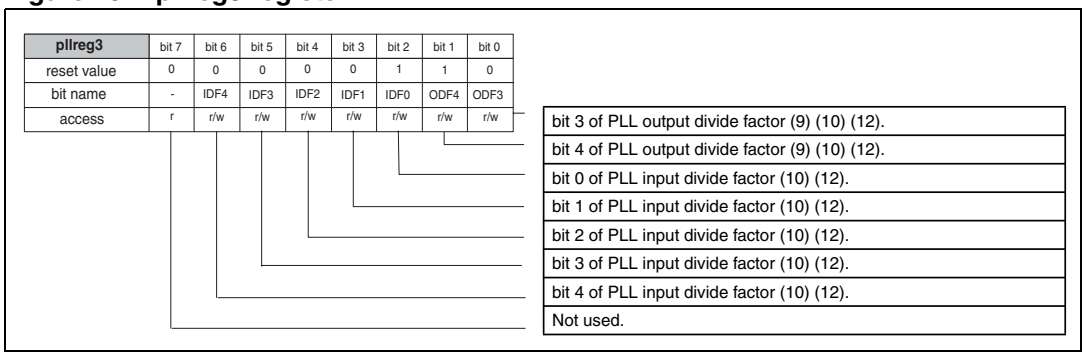
Figure 18. pllreg4 register



pllreg4	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	0	0	0	0	1	1	0	0
bit name	LOCK	LLOCK	PLLEN	PWDN	DITEN	FRAEN	TEST1	TEST0
access	r	r	r/w	r/w	r/w	r/w	r/w	r/w

3.8.10 pllreg3 register

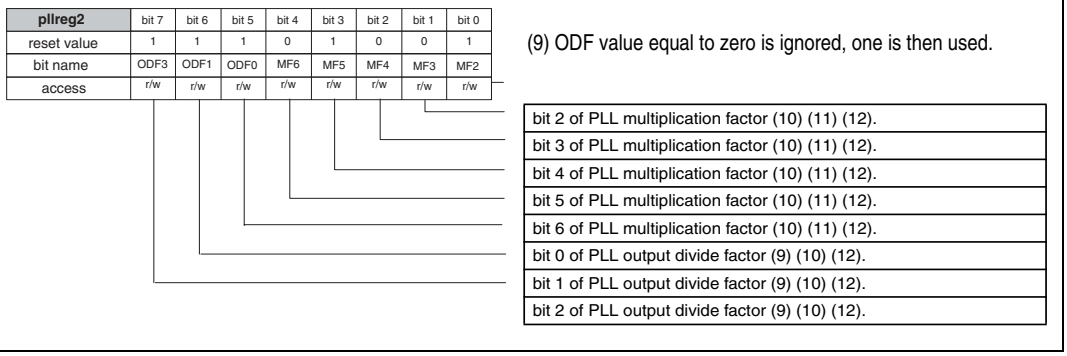
Figure 19. pllreg3 register



pllreg3	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	0	0	0	0	0	1	1	0
bit name	-	IDF4	IDF3	IDF2	IDF1	IDF0	ODF4	ODF3
access	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w

3.8.11 pllreg2 register

Figure 20. pllreg2 register



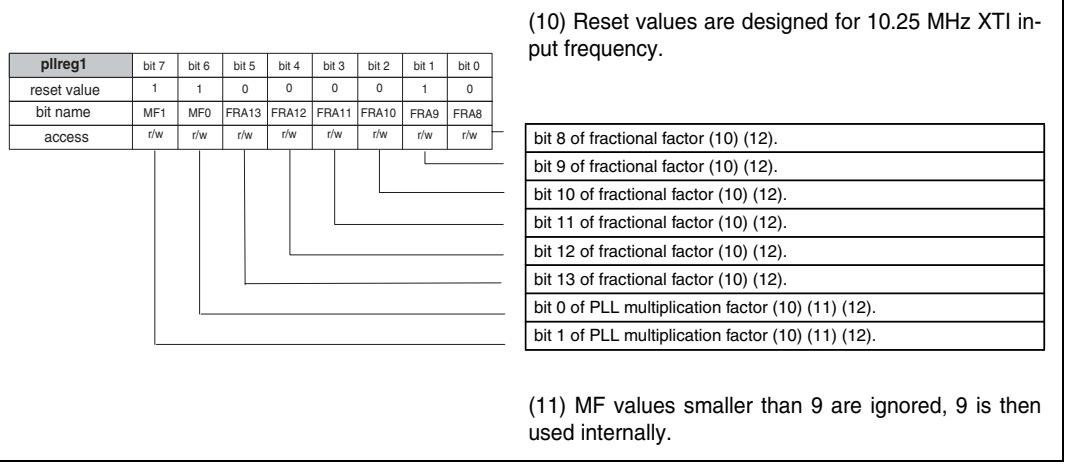
pllreg2	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	1	1	1	0	1	0	0	1
bit name	ODF3	ODF1	ODF0	MF6	MF5	MF4	MF3	MF2
access	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

(9) ODF value equal to zero is ignored, one is then used.

bit 2 of PLL multiplication factor (10) (11) (12).
bit 3 of PLL multiplication factor (10) (11) (12).
bit 4 of PLL multiplication factor (10) (11) (12).
bit 5 of PLL multiplication factor (10) (11) (12).
bit 6 of PLL multiplication factor (10) (11) (12).
bit 0 of PLL output divide factor (9) (10) (12).
bit 1 of PLL output divide factor (9) (10) (12).
bit 2 of PLL output divide factor (9) (10) (12).

3.8.12 pllreg1 register

Figure 21. pllreg1 register



pllreg1	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	1	1	0	0	0	0	1	0
bit name	MF1	MF0	FRA13	FRA12	FRA11	FRA10	FRA9	FRA8
access	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

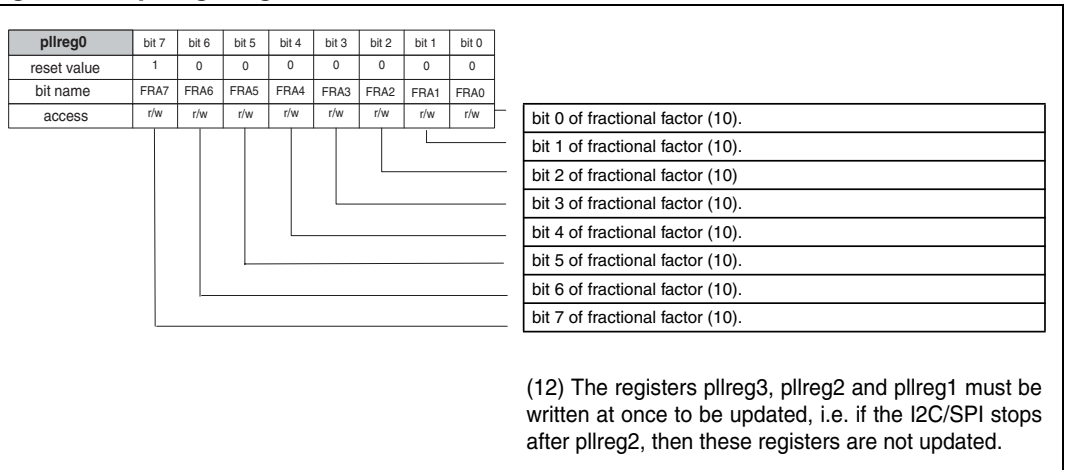
(10) Reset values are designed for 10.25 MHz XTI input frequency.

bit 8 of fractional factor (10) (12).
bit 9 of fractional factor (10) (12).
bit 10 of fractional factor (10) (12).
bit 11 of fractional factor (10) (12).
bit 12 of fractional factor (10) (12).
bit 13 of fractional factor (10) (12).
bit 0 of PLL multiplication factor (10) (11) (12).
bit 1 of PLL multiplication factor (10) (11) (12).

(11) MF values smaller than 9 are ignored, 9 is then used internally.

3.8.13 pllreg0 register

Figure 22. pllreg0 register



pllreg0	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
reset value	1	0	0	0	0	0	0	0
bit name	FRA7	FRA6	FRA5	FRA4	FRA3	FRA2	FRA1	FRA0
access	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

bit 0 of fractional factor (10).
bit 1 of fractional factor (10).
bit 2 of fractional factor (10)
bit 3 of fractional factor (10).
bit 4 of fractional factor (10).
bit 5 of fractional factor (10).
bit 6 of fractional factor (10).
bit 7 of fractional factor (10).

(12) The registers pllreg3, pllreg2 and pllreg1 must be written at once to be updated, i.e. if the I2C/SPI stops after pllreg2, then these registers are not updated.

Note: sinc4reg and testreg registers are dedicated for testing and are not described in this specification.
Reset values of rds_qu, rds_corrp, rds_bd_h and rds_bd_l registers are not visible for the programmer, because he can see only the copy of this registers in the RAM buffer after a new RDS block was received.

The pllreg4-0 registers must be initialized first, before the RDS functionality can be used. If the “PLLEN” bit of pllreg4 is set from zero to one, then the PLL will be initialized after I^2C/SPI transfer with the actual values of pllreg4-0. After the lock time the PLL switches automatically over to the PLL output clock. The next I^2C/SPI transfer is only allowed after the lock time (500 µs) and additional 25 XTI input clock cycles. If the “PLLEN” bit is set from one to zero, the PLL will be stopped and the system clock is switched back to the XTI input clock (after the I^2C/SPI transfer). The next I^2C/SPI transfer is then only allowed after 25 XTI input clock cycles. This is to avoid any I^2C/SPI communication during clock switching.

The registers pllreg3-1 can be only changed at once. If there are less then all three pllreg3-1 registers written during a I^2C/SPI transfer, then they will be not updated.

If the XTI input frequency is 10.25 MHz, then only register pllreg4 must be programmed, because the pllreg3-0 register reset values can be used without any modification.

3.9 I^2C transfer mode

This interface consists of three lines: a serial data line (SDA), a bit clock (SCL), and a slave address select (SA).

The interface is capable of operating up to 400 kbits/s. If during the setup the system clock fsys is smaller then 8.55 MHz, then the max. I^2C speed decreases linear (e.i. if fsys = 4.275 MHz then the maximum I^2C speed is 200 kbits/s for setup).

Data transfers follow the format shown in *Figure 23*. After the START condition (S), a slave address is sent. The address is 7 bits long followed by an eighth bit which is a data direction bit (R/_W).

A zero indicates a transmission (WRITE), a one indicates a request for data (READ).

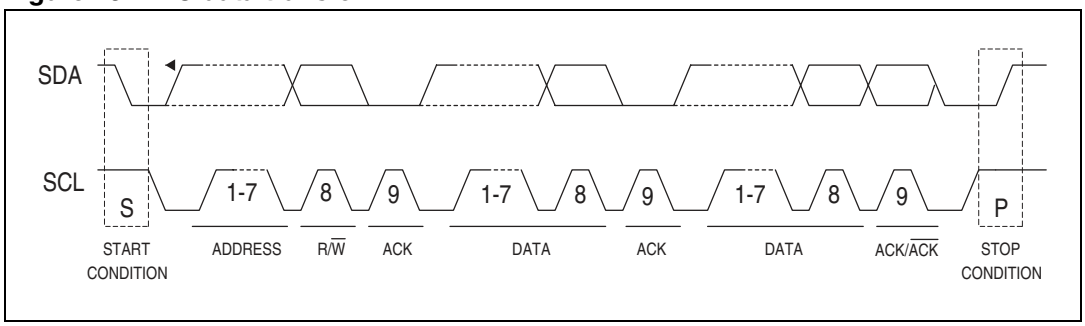
The slave address of the chip is set to 001000S, where S is the least significant bit of the slave address set externally via the pin SA_DATAOUT. This allows to choose between two addresses in case of conflict with another device of the radio set.

Each byte has to be followed by an acknowledge bit (SDA low).

Data is transferred with the most significant (MSB) bit first.

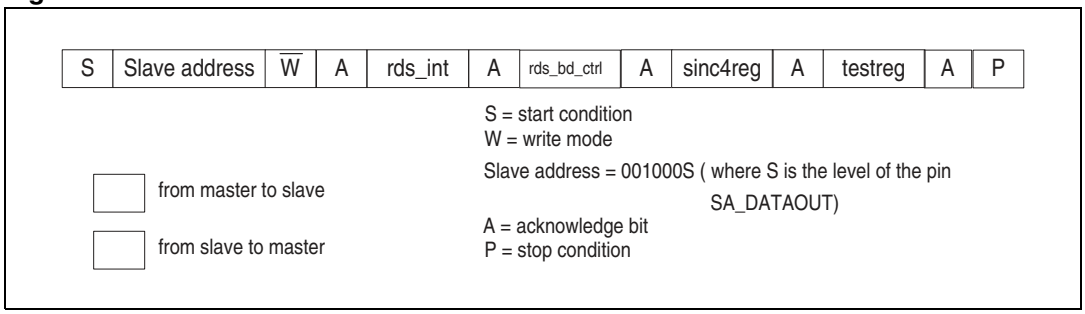
A data transfer is always terminated by a stop condition (P) generated by the master.

Figure 23. I^2C data transfer



3.9.1 Write transfer

Figure 24. I²C write transfer



9 registers are available with write access (please refer to *Section 3.8* for the meaning of each bit).

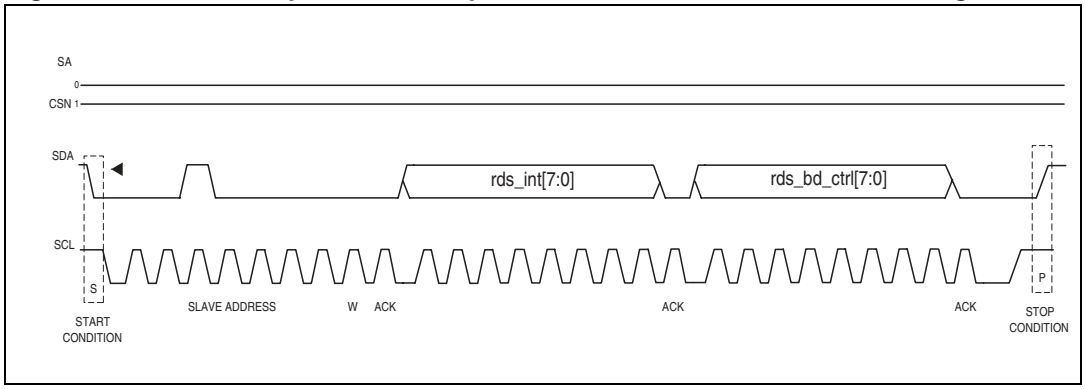
To write registers, the external master must initiate the write transfer as described above, then send the data to be written, and terminate the transfer by generating a stop condition. The transfer can be terminated after having written one, two, three, four (*Figure 24*), or five bytes.

The registers are written in the following order:

rds_int[7:0], rds_bd_ctrl[7:0], sinc4reg[7:0], testreg[7:0], pllreg4[7:0], pllreg3[7:0], pllreg2[7:0], pllreg1[7:0], pllreg0[7:0].

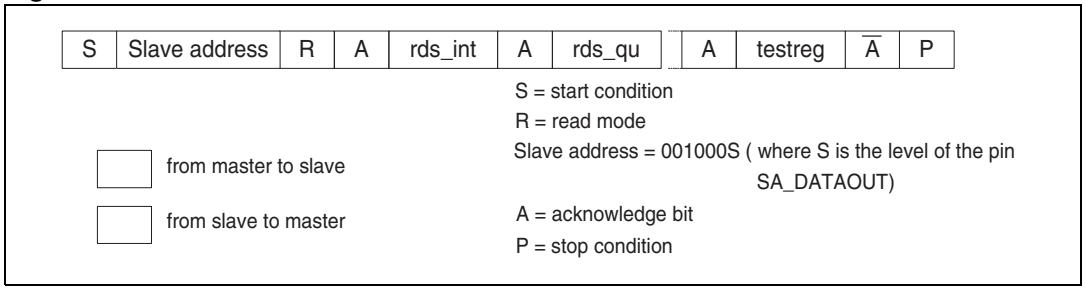
sinc4reg[7:0] and testreg[7:0] are dedicated for test and have to keep zero filled for application.

Figure 25. I²C write operation example: write of rds_int and rds_bd_ctrl registers



3.9.2 Read transfer

Figure 26. I²C read transfer



13 bytes can be read at a time (please refer to *Section 3.8* for the meaning of each bit).

The master has the possibility to read less than 13 registers by not sending the acknowledge bit and then generating a stop condition after having read the needed amount of registers.

There are two typical read access:

- read only the first register rds_int to check the interrupt bit.
- read the first five registers rds_int, rds_qu, rds_corrp, rds_bd_h and rds_bd_l to get the RDS data.

The registers are read in the following order:

rds_int[7:0], rds_qu[7:0], rds_corrp[7:0], rds_bd_h[7:0], rds_bd_l[7:0], rds_bd_ctrl[7:0], sinc4reg[7:0], testreg[7:0], pllreg4[7:0], pllreg3[7:0], pllreg2[7:0], pllreg1[7:0], pllreg0[7:0].

Only the “bne” flag can be used for polling mode. There are two different ways to use this mode, while the first one causes less bus traffic than the second:

1. Read only the first register rds_int to check the “bne” bit.
   If “bne” bit is not set, the stop condition can be set, as shown in (*Figure 28*).
   If “bne” bit is set, the transfer must be continued by the i2c master, until at least the four register rds_qu, rds_corrp, rds_bd_h and rds_bd_l are read out, then the i2c master is allowed to set the stop condition (*Figure 27*). Then the whole Buffer must be read out, by reading each time at least the five registers rds_int, rds_qu, rds_corrp, rds_bd_h and rds_bd_l without interruption. This must be done until the “bne” bit is set to zero (last RDS block).
2. If the I^2C master is not able to handle the above protocol, it must read always at least the first five registers rds_int, rds_qu, rds_corrp, rds_bd_h, rds_bd_l out independent if “bne” is set or not (*Figure 27*). If the “bne” flag is set the whole RAM buffer must be read out, by reading each time at least the five registers rds_int, rds_qu, rds_corrp, rds_bd_h and rds_bd_l without interruption. This must be done until the “bne” bit is set to zero (last RDS block).

Note: In polling mode the interrupt flag “int” is just a indication that the wanted information is stored within the RAM Buffer.
In polling mode it is possible that the last RDS data (rds_qu, rds_corrp, rds_bd_h and rds_bd_l), which was read out as the “bne” flag was set to zero, is identical to the RDS data before. This must be checked by the external micro controller by comparing the last received 2 RDS blocks. If they are identical, one of them can be skipped. (This is the case if just one RDS block is stored in the RAM buffer).

Figure 27. I^2C read access example 1: read of 5 bytes

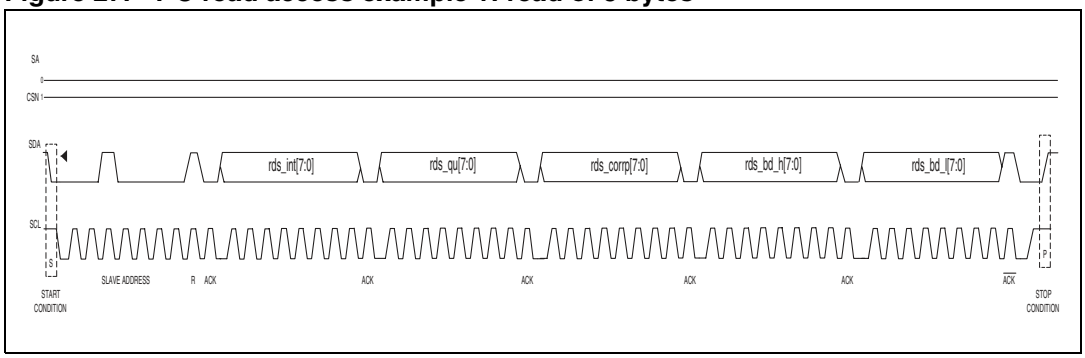
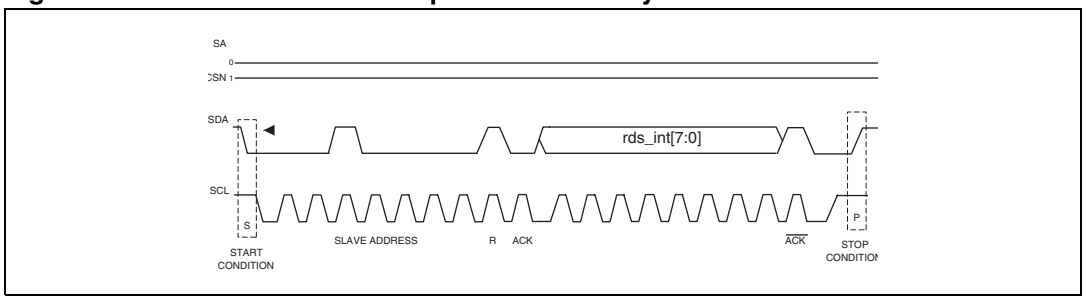
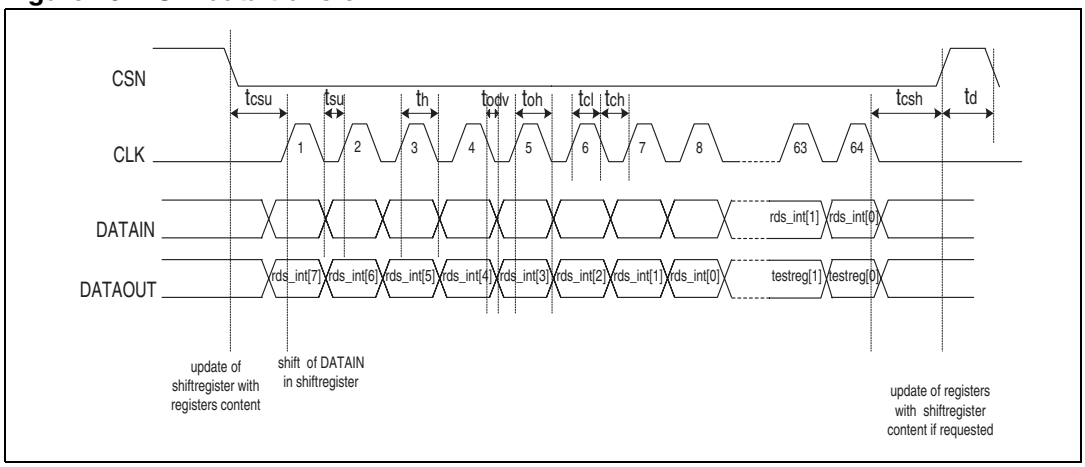


Figure 28. I²C read access example 2: read of 1 byte



3.10 SPI Mode

Figure 29. SPI data transfer



This interface consists of four lines (*Figure 29*). A serial data input (DATAIN), a serial data output (DATAOUT), a chip select input (CSN) and a bit clock input (CLK).

The interface is capable of operating up to 1 MHz. If during the setup the system clock fsys is smaller then 8.55 MHz, then the max. SPI speed decreases linear (e.i. if fsys = 4.275 MHz then the maximum SPI speed is 500 kHz for setup).

CSN starts and stops the data transfer. After starting data transfer, one bit is shifted out (DATAOUT) with the active bit clock edge (CLK) and at the same time one bit in (DATAIN). When CSN stops the data transfer, the pllreg0[7:0], pllreg1[7:0] pllreg2[7:0], pllreg3[7:0], pllreg4[7:0], rdstest[7:0], sinc4reg[7:0], rds_bd_ctrl[7:0], rds_int[7:0] registers can be updated with the last bytes which have been shifted in.

The last byte shifted in on DATAIN must be always rds_int[7:0] and the last but one is rds_bd_ctrl[7:0], and so on, as listed above. In other words, the master has take into account the number of bytes to transfer before starting, to be sure that the last byte shifted in at DATAIN is rds_int[7:0].

If the pllreg0[7:0], pllreg1[7:0] pllreg2[7:0], pllreg3[7:0], pllreg4[7:0], rdstest[7:0], sinc4reg[7:0], rds_bd_ctrl[7:0], rds_int[7:0] registers will be updated depends on the MSB of rds_int. If rds_int[7] = 1 all registers listed above are updated (refer to page 18). The registers pllreg3-1 are only updated if they are shifted completely into the SPI.

sinc4reg[7:0] and testreg[7:0] are dedicated for test **and have to be kept zero filled** in the application, independent if rds_int[7] bit is set or not.

Only the “bne” flag can be used for polling mode. There are two different ways to use **polling mode**, while the first one causes less bus traffic than the second:

1. Read only the first register rds_int to check the “bne” bit.
   If “bne” bit is not set, the CSN can be set, as shown in (*Figure 32*).
   If “bne” bit is set, the transfer must be continued by the SPI master, until at least the four register rds_qu, rds_corrp, rds_bd_h and rds_bd_l are read out, then the SPI master is allowed to stop the transfer by pulling CSN up. Then the whole Buffer must be read out, by reading each time at least the five registers rds_int, rds_qu, rds_corrp, rds_bd_h and rds_bd_l without interruption. This must be done until the “bne” bit is set to zero (last RDS block).
2. If the SPI master is not able to handle the above protocol, it must read always at least the first five registers rds_int, rds_qu, rds_corrp, rds_bd_h, rds_bd_l out independent if “bne” is set or not. If the “bne” flag is set the whole RAM Buffer must be read out, by reading each time at least the five registers rds_int, rds_qu, rds_corrp, rds_bd_h and rds_bd_l without interruption. This must be done until the “bne” bit is set to zero (last RDS block).

Note: In polling mode the interrupt flag “int” is just a indication that the wanted information is stored within the RAM buffer.
In polling mode it is possible that the last RDS data (rds_qu, rds_corrp, rds_bd_h and rds_bd_l), which was read out as the “bne” flag was set to zero, is identical to the RDS data before. This must be checked by the external micro controller by comparing the last received 2 RDS blocks. If they are identical, one of them can be skipped (This is the case if just one RDS block is stored within the RAM buffer).
Hereafter you can find typical read/write access in SPI mode:

Figure 30. Write rds_int, rds_bd_ctrl and pll_reg4 registers in SPI mode, reading RDS data and related flags

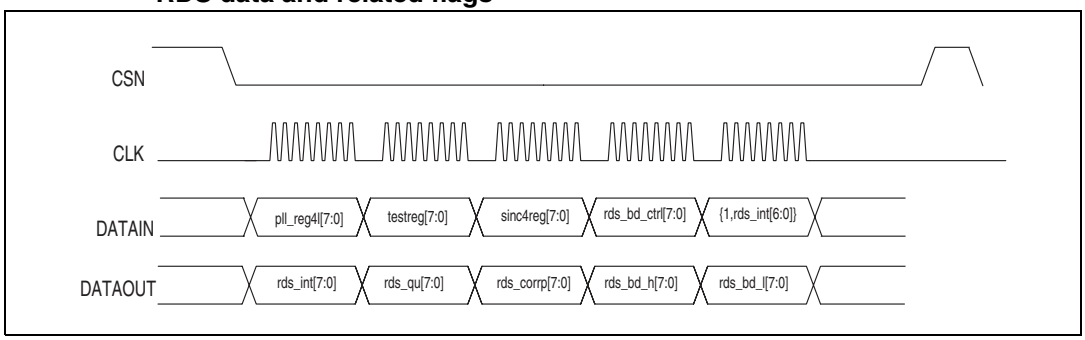
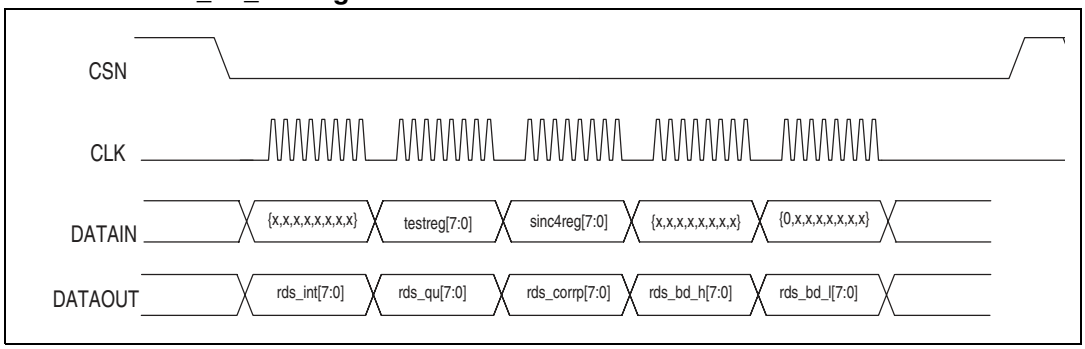
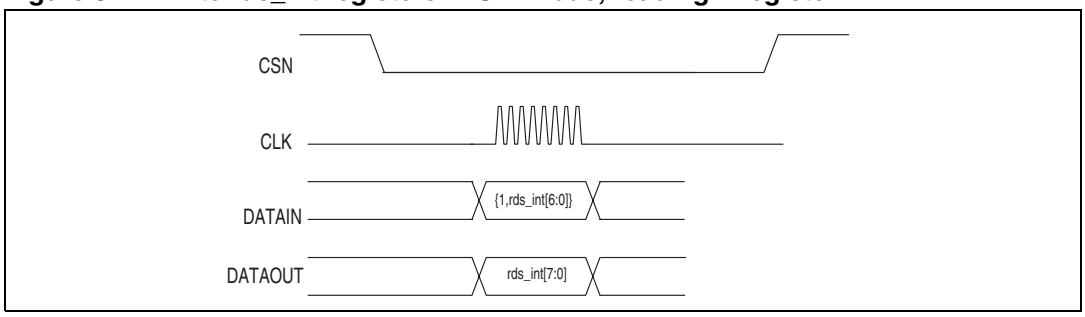


Figure 31. Read out RDS data and related flags, no update of rds_int and rds_bd_ctrl registers



Note: sinc4reg and testreg must be zero filled for application.

Figure 32. Write rds_int registers in SPI mode, reading 1 register



The content of the RDS registers is clocked out on DATAOUT pin in the following order:

rds_int[7:0], rds_qu[7:0], rds_corrp[7:0], rds_bd_l[7:0], rds_bd_h[7:0], rds_ctrl[7:0], sinc4reg[7:0], testreg[7:0], pllreg4[7:0], pllreg3[7:0], pllreg2[7:0], pllreg1[7:0], pllreg0[7:0].

For the meaning of each bit please refer to *Section 3.8*.

Note: 1 After 40 bit clocks the whole RDS data and flags are clocked out.

2 In SPI mode with applications having 2 or more SPI peripherals, it is necessary to inhibit the clock line going to the TDA7333N when the CE line is kept high (not active).

4 Application notes

4.1 Typical RDS data transfer

1. After power up the device, the PLL must be initialized and enabled to generate the 8.55 MHz or 8.664 MHz system clock (fsys). If the XTI frequency is already 8.55 MHz or 8.664 MHz, this point can be skipped. If not, the pllreg4-0 register must be programmed via I2C/SPI. If the XTI frequency is smaller then 8.55 MHz, the reduced maximum I2C/SPI speed must be considered. After the pllreg4-0 register has been programmed, 500 us and additional 25 XTI input clock cycles must be waited until the PLL is locked and the system clock fsys is switched over to the PLL output clock. Then the next I^2C/SPI transfer is allowed with its maximum speed specified for the 8.55/8.664 MHz system clock (fsys).
2. In the next I2C/SPI transfer the interrupt source will be set to “buffer not empty” (itsrc[2:0] = 001) and a resynchronization should be forced (rds_int[5] = 1), to be sure that the buffer is empty and not filled with spurious RDS data. To do this only an write access to the first register rds_int is needed.
3. Now the pin INTN must be continuously checked for an interrupt (active low). If there is an interrupt the five registers rds_int, rds_qu, rds_corrp, rds_bd_h and rds_bd_l must be read out to get the RDS data. The next interrupt can not be expected before 22 ms.
4. If it is not possible to service the interrupt in time, then the RDS buffer can store up to 24 RDS bocks. If the buffer is full and the data could not be read before the next RDS block, the “buffer overflow” flag (rds_corrp[0] = 1) will be set. In this case at least one RDS block is missed. The “buffer overflow” flag is only cleared, if the whole RDS buffer is read out.

If there is no pin available for checking the INTN pin, then it is possible to read out the RDS data by I^2C/SPI polling. Only the “buffer not empty” flag (rds_int[6]) can be used for that. If rds_int[6] bit is set, the 2C/SPI transfer must be continued, until at least the four register rds_qu, rds_corrp, rds_bd_h and rds_bd_l are read out.

This must be done until rds_int[6] bit is set to zero (last RDS block). It is possible that the last RDS block is the same as the last but one RDS block. This is the case if just one RDS block was stored in the RAM buffer. If they are identical, one of them can be skipped.

If another interrupt source is used instead of “buffer not empty” for the INTN pin, also the polling mode must be used for reading out the whole RDS buffer, as described above.

5 Package information

In order to meet environmental requirements, ST offers these devices in different grades of ECOPACK® packages, depending on their level of environmental compliance. ECOPACK® specifications, grade definitions and product status are available at: *www.st.com*.

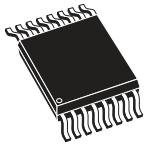
ECOPACK® is an ST trademark.

Figure 33. TSSOP16 mechanical data and package dimensions

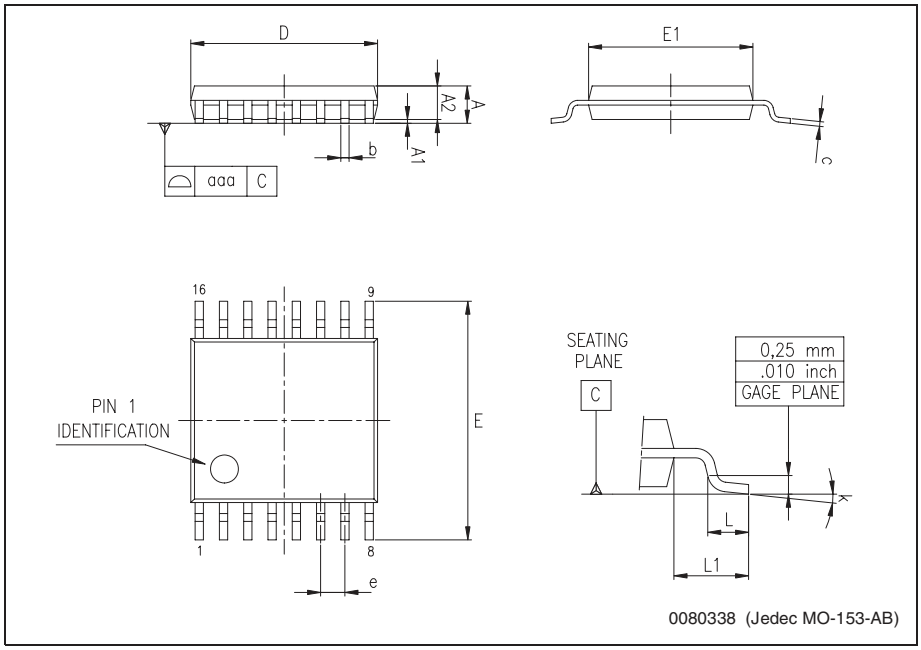
DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			1.200			0.047
A1	0.050		0.150	0.002		0.006
A2	0.800	1.000	1.050	0.031	0.039	0.041
b	0.190		0.300	0.007		0.012
c	0.090		0.200	0.005		0.009
D (1)	4.900	5.000	5.100	0.114	0.118	0.122
E	6.200	6.400	6.600	0.244	0.252	0.260
E1 (1)	4.300	4.400	4.500	0.170	0.173	0.177
e		0.650			0.026	
L	0.450	0.600	0.750	0.018	0.024	0.030
L1		1.000			0.039	
k	0° (min.) 8° (max.)					
aaa			0.100			0.004

Note: 1. D and E1 does not include mold flash or protrusions. Mold flash or potrusions shall not exceed 0.15mm (.006inch) per side.

OUTLINE AND MECHANICAL DATA



TSSOP16 (Body 4.4mm)



6 Revision history

Table 8. Document revision history

Date	Revision	Changes
06-Feb-2006	1	Initial release.
24-Jul-2006	2	Document reformatted. Modified function of the pin 6 on *Table 2*.
09-Jun-2008	3	Added *Note 2 on page 32*.
23-Sep-2009	4	Updated *Figure 1: Block diagram* and *Figure 2: Pin connection (top view)*. *Section 3.8: Programming through serial bus interface* reformatted, no content change.