Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	ST10
Core Size	16-Bit
Speed	40MHz
Connectivity	CANbus, EBI/EMI, SSC, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	111
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	12K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	144-BQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st10f269z2t3

TABLE OF CONTENTS PAGE

5.5.3 - Programming Examples

Most of the microcontroller programs are written in the C language where the data page pointers are automatically set by the compiler. But because the C compiler may use the not allowed direct addressing mode for Flash write addresses, it is necessary to program the organizational Flash accesses (command sequences) with assembler in-line routines which use indirect addressing.

Example 1 Performing the command Read/Reset

We assume that in the initialization phase the lowest 32K Bytes of Flash memory (sector 0) have been mapped to segment 1.

According to the usual way of ST10 data addressing with data page pointers, address bit A15 and A14 of a 16-bit command write address select the data page pointer (DPP) which contains the upper 10-bit for building the 24-bit physical data address. Address bit A13...A0 represent the address offset. As the bit A14...A17 are "don't care" when written a Flash command in the Command Interface (CI), we can choose the most convenient DPPx register for address handling.

The following examples are making usage of DPP0. We just have to make sure, that DPP0 points to active Flash memory space.

To be independent of mapping of sector 0 we choose for all DPPs which are used for Flash address handling, to point to segment 2.

For this reason we load DPP0 with value 08h (00 0000 l000b).

```
MOV   R5, #01554h          ;load auxilary register R5 with command address
                           ;(used in command cycle 1)
MOV   R6, #02AA8h          ;load auxilary register R6 with command address
                           ;(used in command cycle 2)
SCXT  DPPO, #08h           ;push data page pointer 0 and load it to point to
                           ;segment 2
MOV   R7, #0A8h            ;load register R7 with 1st CI enable command
MOV   [R5], R7             ;command cycle 1
MOV   R7, #054h            ;load register R7 with 2cd CI enable command
MOV   [R6], R7             ;command cycle 2
MOV   R7, #0F0h            ;load register R7 with Read/Reset command
MOV   [R5], R7             ;command cycle 3. Address is don't care
POP   DPP0                 ;restore DPP0 value
```

In the example above the 16-bit registers R5 and R6 are used as auxiliary registers for indirect addressing.

Example 2 Performing a Program Word command

We assume that in the initialization phase the lowest 32K Bytes of Flash memory (sector 0) have been mapped to segment 1.The data to be written is loaded in register R13, the address to be programmed is loaded in register R11/R12 (segment number in R11, segment offset in R12).

```
MOV   R5, #01554h          ;load auxilary register R5 with command address
                           ;(used in command cycle 1)
MOV   R6, #02AA8h          ;load auxilary register R6 with command address
                           ;(used in command cycle 2)
SXCT  DPPO, #08h           ;push data page pointer 0 and load it to point to
                           ;segment 2
MOV   R7, #0A8h            ;load register R7 with 1st CI enable command
MOV   [R5], R7             ;command cycle 1
MOV   R7, #054h            ;load register R7 with 2cd CI enable command
MOV   [R6], R7             ;command cycle 2
MOV   R7, #0A0h            ;load register R7 with Program Word command
MOV   [R5], R7             ;command cycle 3
POP   DPP0                 ;restore DPP0: following addressing to the Flash
                           ;will use EXTended instructions
                           ;R11 contains the segment to be programmed
```

Example 3 Performing the Block Erase command

We assume that in the initialization phase the lowest 32K Bytes of Flash memory (sector 0) have been mapped to segment 1.The registers R11/R12 contain an address related to the block to be erased (segment number in R11, segment offset in R12, for example R11 = 01h, R12= 4000h will erase the block 1 - first 8K byte block).

```
MOV   R5, #01554h          ;load auxilary register R5 with command address
                           ;(used in command cycle 1)
MOV   R6, #02AA8h          ;load auxilary register R6 with command address
                           ;(used in command cycle 2)
SXCT  DPPO, #08h           ;push data page pointer 0 and load it to point ;to
                           ;segment 2
MOV   R7, #0A8h            ;load register R7 with 1st CI enable command
MOV   [R5], R7             ;command cycle 1
MOV   R7, #054h            ;load register R7 with 2cd CI enable command
MOV   [R6], R7             ;command cycle 2
MOV   R7, #080h            ;load register R7 with Block Erase command
MOV   [R5], R7             ;command cycle 3
MOV   R7, #0A8h            ;load register R7 with 1st CI enable command
MOV   [R5], R7             ;command cycle 4
MOV   R7, #054h            ;load register R7 with 2cd CI enable command
MOV   [R6], R7             ;command cycle 5
POP   DPP0                 ;restore DPP0: following addressing to the Flash
                           ;will use EXTended instructions
                           ;R11 contains the segment of the block to be erased
                           ;R12 contains the segment offset address of the
                           ;block to be erased
MOV   R7, #030h            ;load register R7 with erase confirm code
EXTS  R11, #1              ;use EXTended addressing for next MOV instruction
MOV   [R12], R7            ;command cycle 6: the EPC starts execution of
                           ;Erasing Command
Erase_Polling:
EXTS  R11, #1              ;use EXTended addressing for next MOV instruction
MOV   R7, [R12]            ;read Flash Status register (FSB) in R7
                           ;Check if FSB.7 = ‘1’ (i.e. R7.7 = ‘1’)
JB    R7.7, Erase_OK

                           ;Check if FSB.5 = 1 (Erasing Error)
JNB   R7.5, Erase_Polling
                           ;Programming failed: Flash remains in Write
                           ;Operation.
                           ;To go back to normal Read operations, a Read/Reset
                           ;command
                           ;must be performed
Erase_Error:
MOV   R7, #0F0h            ;load register R7 with Read/Reset command
EXTS  R11, #1              ;use EXTended addressing for next MOV instruction
MOV   [R12], R7            ;address is don’t care for Read/Reset command
...                        ;here place specific Error handling code
...
...

                           ;When erasing operation finished succesfully,
                           ;Flash is set back automatically to normal Read Mode
Erase_OK:
....
```

. . . .

5.6 - Bootstrap Loader

The built-in bootstrap loader (BSL) of the ST10F269 provides a mechanism to load the startup program through the serial interface after reset. In this case, no external memory or internal Flash memory is required for the initialization code starting at location 00'0000h (see Figure 5).

The bootstrap loader moves code/data into the internal RAM, but can also transfer data via the serial interface into an external RAM using a second level loader routine. Flash Memory (internal or external) is not necessary, but it may be used to provide lookup tables or "core-code" like a set of general purpose subroutines for I/O operations, number crunching, system initialization, etc.

The bootstrap loader can be used to load the complete application software into ROMless systems, to load temporary software into complete systems for testing or calibration, or to load a programming routine for Flash devices.

The BSL mechanism can be used for standard system startup as well as for special occasions like system maintenance (firmer update) or end-of-line programming or testing.

5.6.1 - Entering the Bootstrap Loader

The ST10F269 enters BSL mode when pin P0L.4 is sampled low at the end of a hardware reset. In this case the built-in bootstrap loader is activated independent of the selected bus mode.

The bootstrap loader code is stored in a special Boot-ROM. No part of the standard mask Memory or Flash Memory area is required for this.

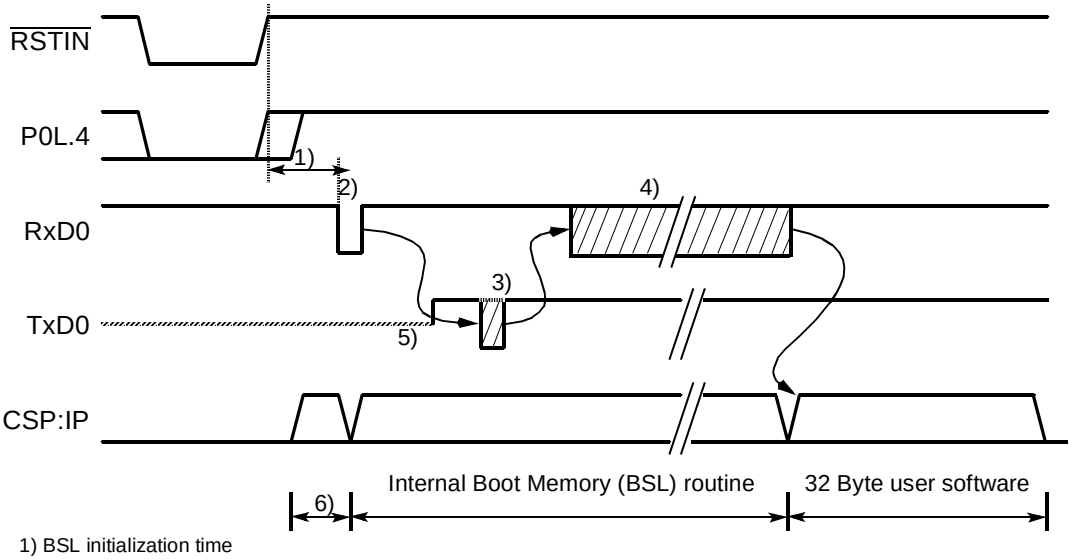
After entering BSL mode and the respective initialization the ST10F269 scans the RXD0 line to receive a zero Byte, one start bit, eight '0' data bits and one stop bit.

From the duration of this zero Byte it calculates the corresponding Baud rate factor with respect to the current CPU clock, initializes the serial interface ASC0 accordingly and switches pin TxD0 to output.

Using this Baud rate, an identification Byte is returned to the host that provides the loaded data.

This identification Byte identifies the device to be booted. The identification byte is D5h for ST10F269.

Figure 5 : Bootstrap Loader Sequence



1) BSL initialization time

2) Zero Byte (1 start bit, eight '0' data bits, 1 stop bit), sent by host.

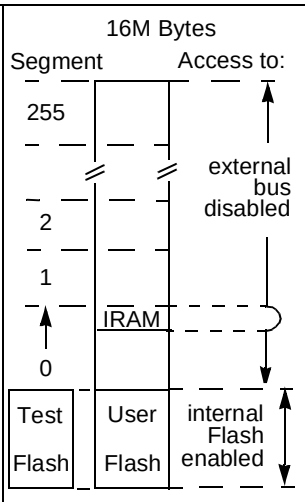
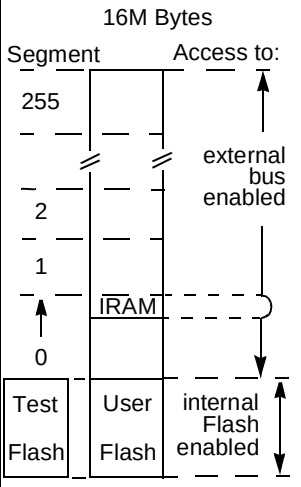
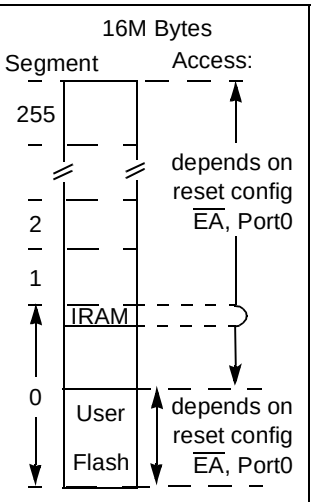
3) Identification Byte (D5h), sent by ST10F269.

4) 32 Bytes of code / data, sent by host.

5) Caution: TxD0 is only driven a certain time after reception of the zero Byte.

6) Internal Boot ROM.

Figure 7 : Memory Configuration after Reset

			
BSL mode active	Yes (P0L.4='0')	Yes (P0L.4='0')	No (P0L.4='1')
$\overline{EA}$ pin	High	Low	Access to application
Code fetch from internal Flash area	Test-Flash access	Test-Flash access	User Flash access
Data fetch from internal Flash area	User Flash access	User Flash access	User Flash access

5.6.3 - Loading the Startup Code

After sending the identification Byte the BSL enters a loop to receive 32 Bytes via ASC0. These Byte are stored sequentially into locations 00'FA40h through 00'FA5Fh of the internal RAM. So up to 16 instructions may be placed into the RAM area. To execute the loaded code the BSL then jumps to location 00'FA40h, which is the first loaded instruction.

The bootstrap loading sequence is now terminated, the ST10F269 remains in BSL mode, however. Most probably the initially loaded routine will load additional code or data, as an average application is likely to require substantially more than 16 instructions. This second receive loop may directly use the pre-initialized interface ASC0 to receive data and store it to arbitrary user-defined locations.

This second level of loaded code may be the final application code. It may also be another, more sophisticated, loader routine that adds a transmission protocol to enhance the integrity of the loaded code or data. It may also contain a code sequence to change the system configuration and enable the bus interface to store the received data into external memory.

This process may go through several iterations or may directly execute the final application. In all cases the ST10F269 will still run in BSL mode, that means with the watchdog timer disabled and limited access to the internal Flash area.

All code fetches from the internal Flash area (00'0000h...00'7FFFh or 01'0000h...01'7FFFh, if mapped to segment 1) are redirected to the special Boot-ROM. Data fetches access will access the internal Boot-ROM of the ST10F269, if any is available, but will return undefined data on ROMless devices.

5.6.4 - Exiting Bootstrap Loader Mode

In order to execute a program in normal mode, the BSL mode must be terminated first. The ST10F269 exits BSL mode upon a software reset (ignores the level on P0L.4) or a hardware reset (P0L.4 must be high). After a reset the ST10F269 will start executing from location 00'0000h of the internal Flash or the external memory, as programmed via pin $\overline{EA}$.

Mnemonic	Addressing Modes	Repeatability
CoMUL CoMULu CoMULus CoMULsu CoMUL- CoMULu- CoMULus- CoMULsu- CoMUL, rnd CoMULu, rnd CoMULus, rnd CoMULsu, rnd	Rw_n, Rw_m [IDX_i⊗], [Rw_m⊗] Rw_n, [Rw_m⊗]	No No No
CoMAC CoMACu CoMACus CoMACsu CoMAC- CoMACu- CoMACus- CoMACsu- CoMAC, rnd CoMACu, rnd CoMACus, rnd CoMACsu, rnd CoMACR CoMACRu	Rw_n, Rw_m [IDX_i⊗], [Rw_m⊗] Rw_n, [Rw_m⊗]	No Yes Yes
CoMACRus CoMACRsu CoMACR, rnd CoMACRu, rnd CoMACRus, rnd CoMACRsu, rnd	Rw_n, Rw_m [IDX_i⊗], [Rw_n⊗] Rw_n, [RW_m⊗]	No No No
CoNOP	[Rw_m⊗] [IDX_i⊗] [IDX_i⊗], [Rw_m⊗]	Yes Yes Yes
CoNEG CoNEG, rnd CoRND	-	No
CoSTORE	Rw_n, CoReg [Rw_n⊗], Coreg	No Yes
CoMOV	[IDX_i⊗], [Rw_m⊗]	Yes

Mnemonic	Addressing Modes	Repeatability
CoMACM CoMACMu CoMACMus CoMACMsu CoMACM- CoMACMu- CoMACMus- CoMACMsu- CoMACM, rnd CoMACMu, rnd CoMACMus, rnd CoMACMsu, rnd CoMACMR CoMACMRu CoMACMRus CoMACMRsu CoMACMR, rnd CoMACMRu, rnd CoMACMRus, rnd CoMACMRsu, rnd	[$IDX_i\otimes$], [$Rw_m\otimes$]	Yes
CoADD CoADD2 CoSUB CoSUB2 CoSUBR CoSUB2R CoMAX CoMIN	Rw_n, Rw_m [$IDX_i\otimes$], [$Rw_m\otimes$] Rw_n, [$Rw_m\otimes$]	No Yes Yes
CoLOAD CoLOAD- CoLOAD2 CoLOAD2- CoCMP	Rw_n, Rw_m [$IDX_i\otimes$], [$Rw_m\otimes$] Rw_n, [$Rw_m\otimes$]	No No No
CoSHL CoSHR CoASHR CoASHR, rnd	Rw_m #data4 [$Rw_m\otimes$]	Yes No Yes
CoABS	- Rw_n, Rw_m [$IDX_i\otimes$], [$Rw_m\otimes$] Rw_n, [$Rw_m\otimes$]	No No No

7 - EXTERNAL BUS CONTROLLER

All of the external memory accesses are performed by the on-chip external bus controller.

The EBC can be programmed to single chip mode when no external memory is required, or to one of four different external memory access modes:

– 16- / 18- / 20- / 24-bit addresses and 16-bit data, demultiplexed

– 16- / 18- / 20- / 24-bit addresses and 16-bit data, multiplexed

– 16- / 18- / 20- / 24-bit addresses and 8-bit data, multiplexed

– 16- / 18- / 20- / 24-bit addresses and 8-bit data, demultiplexed

In demultiplexed bus modes addresses are output on PORT1 and data is input / output on PORT0 or P0L, respectively. In the multiplexed bus modes both addresses and data use PORT0 for input / output.

Timing characteristics of the external bus interface (memory cycle time, memory tri-state time, length of ALE and read / write delay) are programmable giving the choice of a wide range of memories and external peripherals.

Up to 4 independent address windows may be defined (using register pairs ADDRSELx / BUSCONx) to access different resources and bus characteristics.

These address windows are arranged hierarchically where BUSCON4 overrides BUSCON3 and BUSCON2 overrides BUSCON1.

All accesses to locations not covered by these 4 address windows are controlled by BUSCON0. Up to 5 external $\overline{CS}$ signals (4 windows plus default) can be generated in order to save external glue logic. Access to very slow memories is supported by a 'Ready' function.

A $\overline{HOLD}$ / $\overline{HLDA}$ protocol is available for bus arbitration which shares external resources with other bus masters.

The bus arbitration is enabled by setting bit HLDEN in register PSW. After setting HLDEN once, pins P6.7...P6.5 ($\overline{BREQ}$, $\overline{HLDA}$, $\overline{HOLD}$) are automatically controlled by the EBC. In master mode (default after reset) the $\overline{HLDA}$ pin is an output. By setting bit DP6.7 to'1' the slave mode is selected where pin $\overline{HLDA}$ is switched to input. This directly connects the slave controller to another master controller without glue logic.

For applications which require less external memory space, the address space can be restricted to 1M Byte, 256K Bytes or to 64K Bytes. Port 4 outputs all 8 address lines if an address space of 16M Bytes is used, otherwise four, two or no address lines.

Chip select timing can be made programmable. By default (after reset), the CSx lines change half a CPU clock cycle after the rising edge of ALE. With the CSCFG bit set in the SYSCON register the CSx lines change with the rising edge of ALE.

The active level of the READY pin can be set by bit RDYPOL in the BUSCONx registers. When the READY function is enabled for a specific address window, each bus cycle within the window must be terminated with the active level defined by bit RDYPOL in the associated BUSCON register.

7.1 - Programmable Chip Select Timing Control

The ST10F269 allows the user to adjust the position of the CSx line changes. By default (after reset), the CSx lines change half a CPU clock cycle (12.5ns at 40MHz of CPU clock on PQFP144 devices and 31.25ns at 32MHz of CPU clock on TQFP144 devices) after the rising edge of ALE. With the CSCFG bit set in the SYSCON register the CSx lines change with the rising edge of ALE, thus the CSx lines and the address lines change at the same time (see Figure 11).

7.2 - READY Programmable Polarity

The active level of the READY pin can be selected by software via the RDYPOL bit in the BUSCONx registers.

When the READY function is enabled for a specific address window, each bus cycle within this window must be terminated with the active level defined by this RDYPOL bit in the associated BUSCON register.

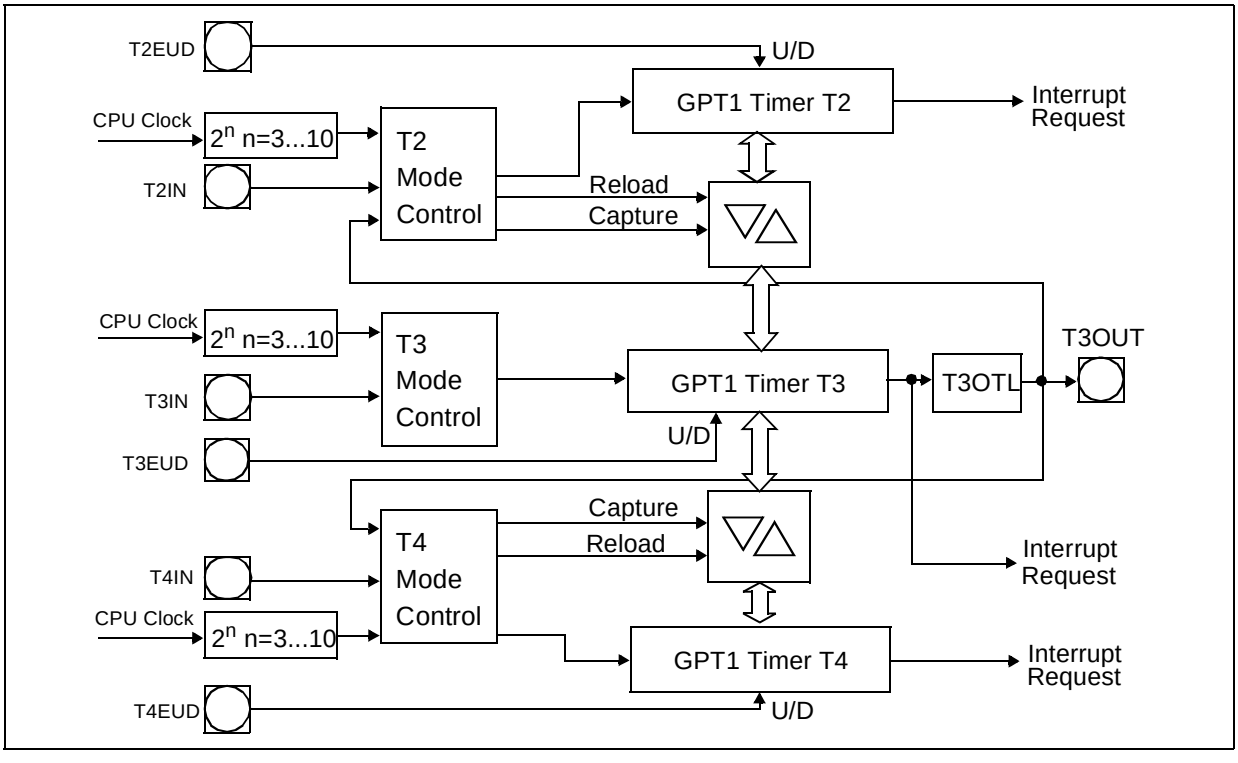
BUSCONx registers are described in Section 20.2 -: *System Configuration Registers*.

Note ST10F269 as no internal pull-up resistor on READY pin.

Table 13 : GPT1 Timer Input Frequencies, Resolution and Periods (TQFP144 devices)

f_{CPU} = 32MHz	Timer Input Selection T2I / T3I / T4I							
	000b	**001b**	**010b**	**011b**	**100b**	**101b**	**110b**	**111b**
Pre-scaler factor	8	16	32	64	128	256	512	1024
Input Freq	4MHz	2MHz	1MHz	500KHz	250KHz	125KHz	62.5KHz	31.125KHz
Resolution	250ns	500ns	1µs	2µs	4µs	8µs	16µs	32µs
Period maximum	16.4ms	32.8ms	65.5ms	131ms	262.1ms	524.3ms	1.05s	2.1s

Figure 15 : Block Diagram of GPT1



10.2 - GPT2

The GPT2 module provides precise event control and time measurement. It includes two timers (T5, T6) and a capture/reload register (CAPREL). Both timers can be clocked with an input clock which is derived from the CPU clock via a programmable prescaler or with external signals. The count direction (up/down) for each timer is programmable by software or may additionally be altered dynamically by an external signal on a port pin (TxEUD). Concatenation of the timers is supported via the output toggle latch (T6OTL) of timer T6 which changes its state on each timer overflow/underflow.

The state of this latch may be used to clock timer T5, or it may be output on a port pin (T6OUT). The overflow / underflow of timer T6 can additionally be used to clock the CAPCOM timers T0 or T1, and to cause a reload from the CAPREL register. The CAPREL register may capture the contents of timer T5 based on an external signal transition on the corresponding port pin (CAPIN), and timer T5 may optionally be cleared after the capture procedure. This allows absolute time differences to be measured or pulse multiplication to be performed without software overhead.

The capture trigger (timer T5 to CAPREL) may also be generated upon transitions of GPT1 timer T3 inputs T3IN and/or T3EUD. This is advantageous when T3 operates in Incremental Interface Mode.

Table 14 GPT2 Timer Input Frequencies, Resolution and Period (PQFP144 devices) and Table 15 GPT2 Timer Input Frequencies,

Resolution and Period (TQFP144 devices) list the timer input frequencies, resolution and periods for each pre-scaler option at 40MHz (or 32MHz) CPU clock. This also applies to the Gated Timer Mode of T6 and to the auxiliary timer T5 in Timer and Gated Timer Mode.

Table 14 : GPT2 Timer Input Frequencies, Resolution and Period (PQFP144 devices)

f_{CPU} = 40MHz	Timer Input Selection T5I / T6I							
	000b	**001b**	**010b**	**011b**	**100b**	**101b**	**110b**	**111b**
Pre-scaler factor	4	8	16	32	64	128	256	512
Input Freq	10MHz	5MHz	2.5MHz	1.25MHz	625kHz	312.5kHz	156.25kHz	78.125kHz
Resolution	100ns	200ns	400ns	0.8µs	1.6µs	3.2µs	6.4µs	12.8µs
Period maximum	6.55ms	13.1ms	26.2ms	52.4ms	104.8ms	209.7ms	419.4ms	838.9ms

Table 15 : GPT2 Timer Input Frequencies, Resolution and Period (TQFP144 devices)

f_{CPU} = 32MHz	Timer Input Selection T5I / T6I							
	000b	**001b**	**010b**	**011b**	**100b**	**101b**	**110b**	**111b**
Pre-scaler factor	4	8	16	32	64	128	256	512
Input Freq	8MHz	4MHz	2MHz	1MHz	500KHz	250KHz	125KHz	62.5KHz
Resolution	125ns	250ns	500ns	1µs	2µs	4µs	8µs	16µs
Period maximum	8.19ms	16.4ms	32.8ms	65.5ms	131ms	262.1ms	524.3ms	1.05s

12.7.1 - Alternate Functions of Port 4

During external bus cycles that use segmentation (address space above 64K Bytes) a number of Port 4 pins may output the segment address lines. The number of pins that is used for segment address output determines the external address space which is directly accessible. The other pins of Port 4 may be used for general purpose I/O. If segment address lines are selected, the alternate function of Port 4 may be necessary to access external memory directly after reset. For this reason Port 4 will be switched to this alternate function automatically.

The number of segment address lines is selected via PORT0 during reset. The selected value can be read from bitfield SALSEL in register RP0H (read only) in order to check the configuration during run time.

The CAN interfaces use 2 or 4 pins of Port 4 to interface each CAN Modules to an external CAN transceiver. In this case the number of possible segment address lines is reduced.

The Table 21 summarizes the alternate functions of Port 4 depending on the number of selected segment address lines (coded via bitfield SALSEL)

Table 21 : Port 4 Alternate Functions

Port 4	Standard Function SALSEL = 01 64K Bytes	Alternate Function SALSEL = 11 256K Bytes	Alternate Function SALSEL = 00 1M Byte	Alternate Function SALSEL = 10 16M Bytes
P4.0	GPIO	Segment Address A16	Segment. Address A16	Segment Address A16
P4.1	GPIO	Segment Address A17	Segment Address A17	Segment Address A17
P4.2	GPIO	GPIO	Segment Address A18	Segment Address A18
P4.3	GPIO	GPIO	Segment Address A19	Segment Address A19
P4.4	GPIO/CAN2_RxD	GPIO/CAN2_RxD	GPIO/CAN2_RxD	Segment Address A20
P4.5	GPIO/CAN1_RxD	GPIO/CAN1_RxD	GPIO/CAN1_RxD	Segment Address A21
P4.6	GPIO/CAN1_TxD	GPIO/CAN1_TxD	GPIO/CAN1_TxD	Segment Address A22
P4.7	GPIO/CAN2_TxD	GPIO/CAN2_TxD	GPIO/CAN2_TxD	Segment Address A23

Figure 30 : Port 4 I/O and Alternate Functions

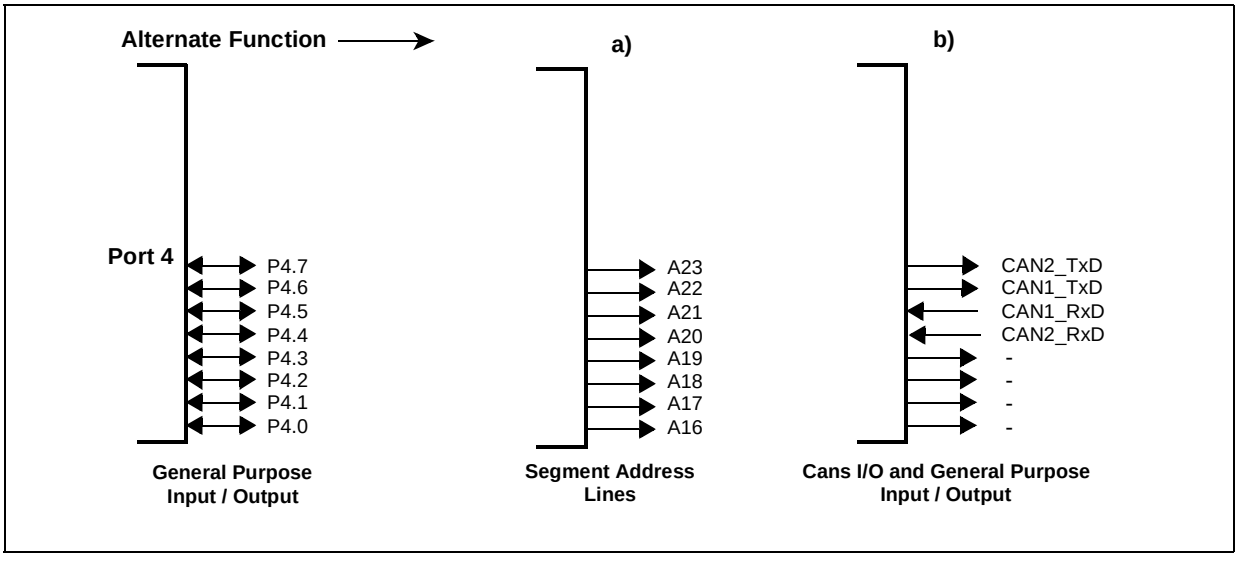
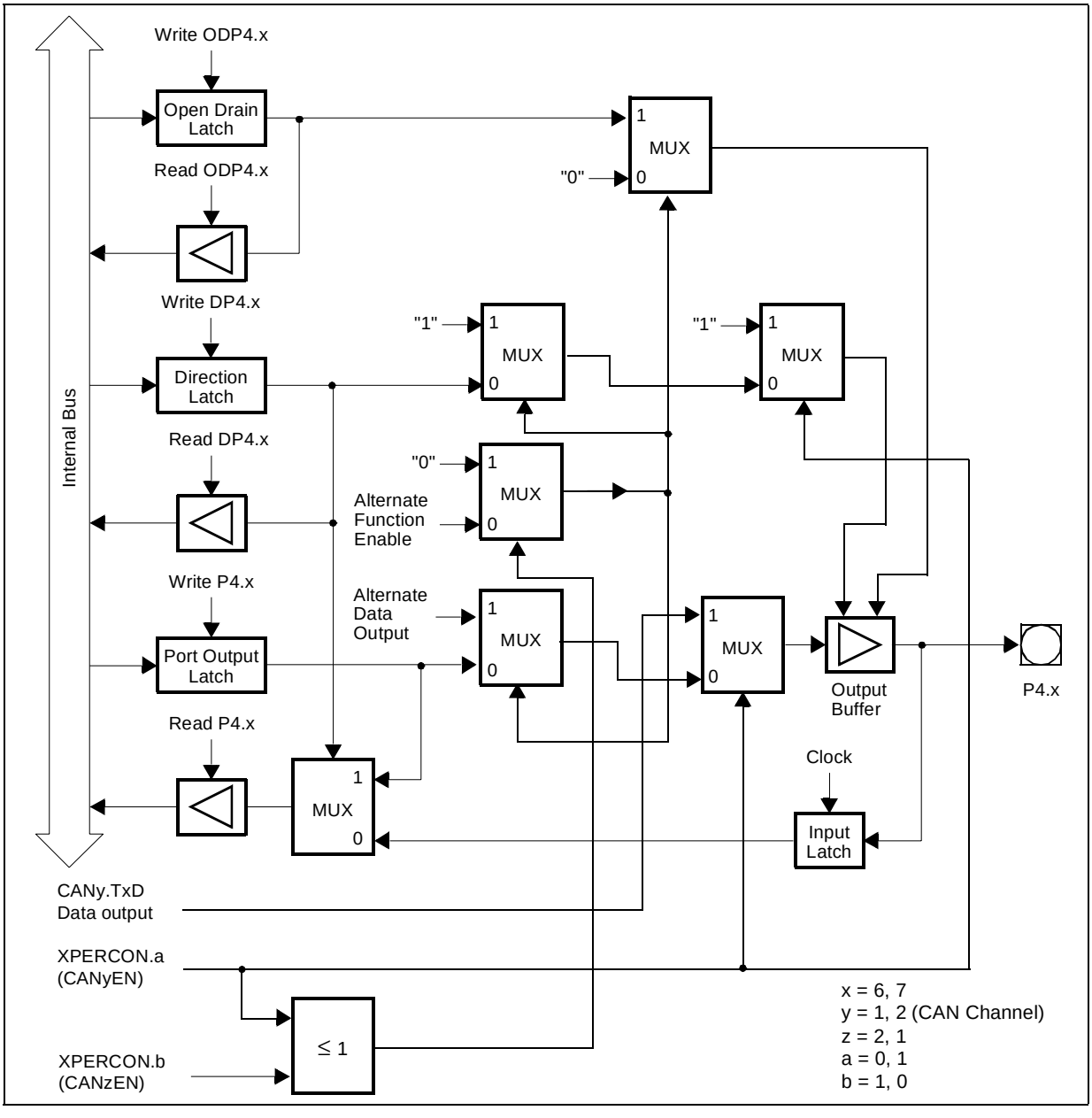


Figure 33 : Block Diagram of P4.6 and P4.7 Pins



12.8 - Port 5

This 16-bit input port can only read data. There is no output latch and no direction register. Data written to P5 will be lost.

P5 (FFA2h / D1h) SFR Reset Value: XXXXh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P5.15	P5.14	P5.13	P5.12	P5.11	P5.10	P5.9	P5.8	P5.7	P5.6	P5.5	P5.4	P5.3	P5.2	P5.1	P5.0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

P5.y	**Port Data Register P5 Bit y (Read only)**

The chip select lines of Port 6 have an internal weak pull-up device. This device is switched on during reset. This feature is implemented to drive the chip select lines high during reset in order to avoid multiple chip selection.
After reset the $\overline{CS}$ function must be used, if selected so. In this case there is no possibility to program any port latches before. Thus the alternate function ($\overline{CS}$) is selected automatically in this case.

Note: The open drain output option can only be selected via software earliest during the initialization routine; at least signal $\overline{CS0}$ will be in push/pull output driver mode directly after reset.

Figure 37 : Block Diagram of Port 6 Pins with an Alternate Output Function

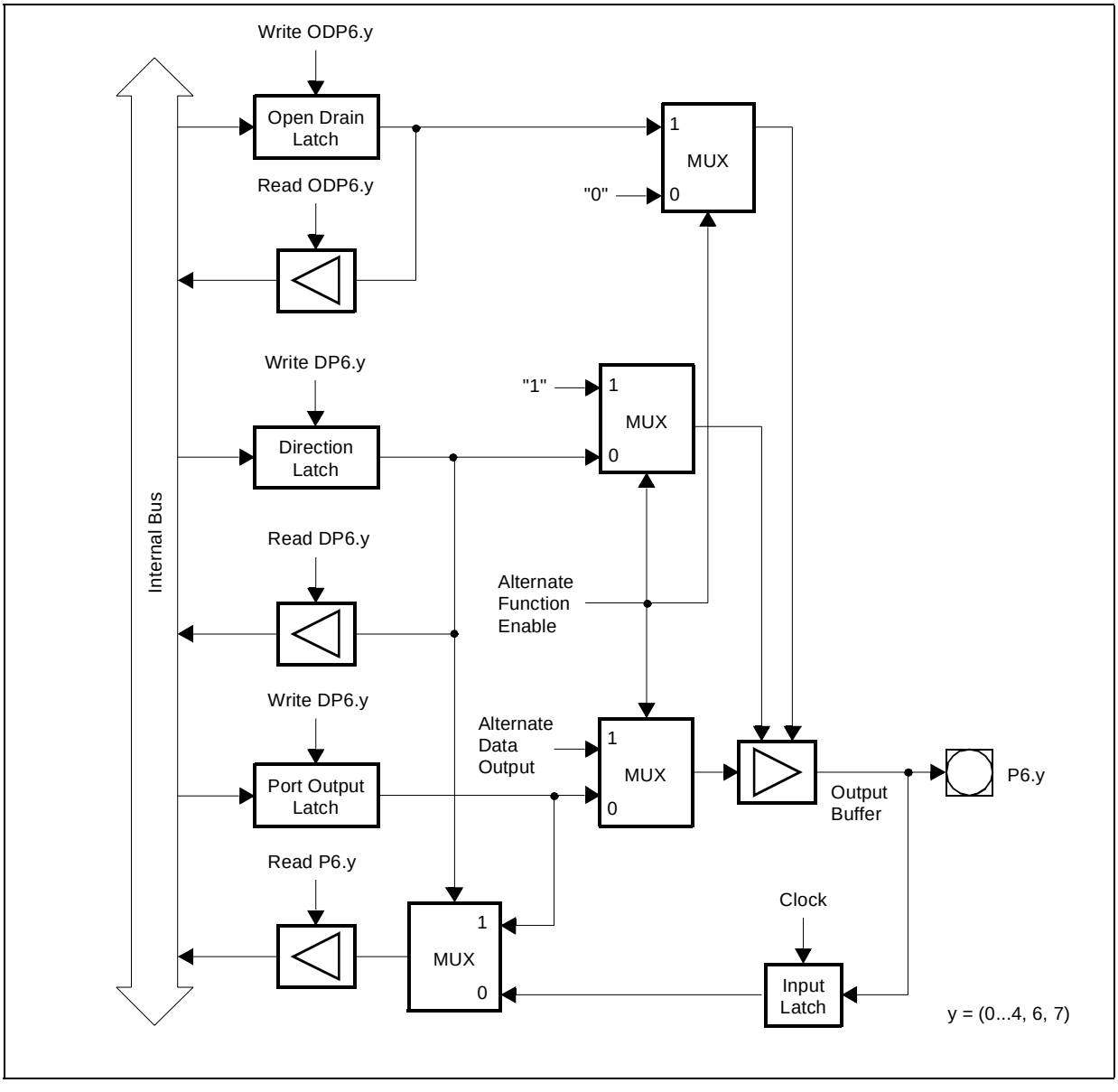
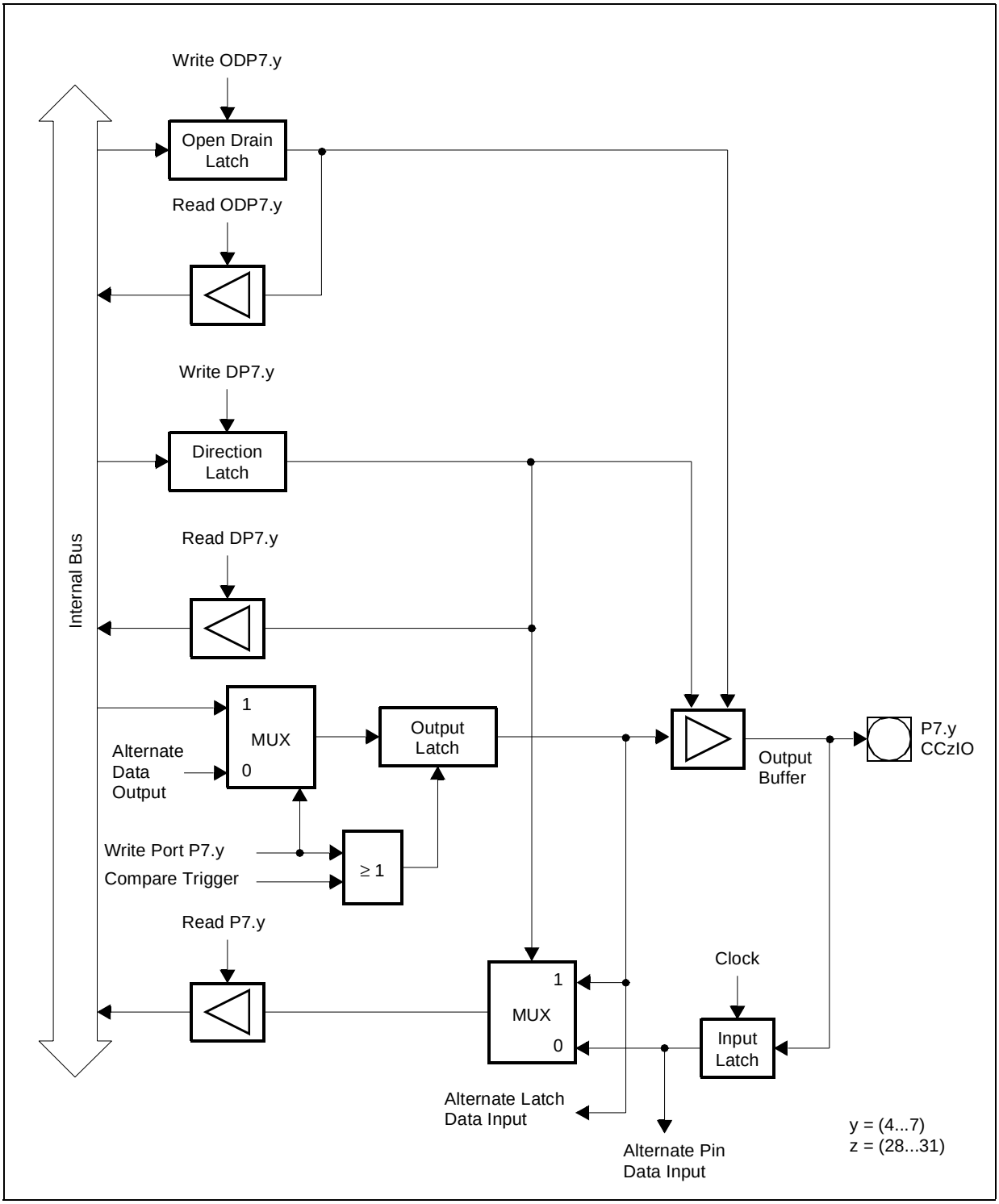


Figure 41 : Block Diagram of Port 7 Pins P7.7...P7.4



13 - A/D CONVERTER

A 10-bit A/D converter with 16 multiplexed input channels and a sample and hold circuit is integrated on-chip. The sample time (for loading the capacitors) and the conversion time is programmable and can be adjusted to the external circuitry.

To remove high frequency components from the analog input signal, a low-pass filter must be connected at the ADC input.

Overrun error detection / protection is controlled by the ADDAT register. Either an interrupt request is generated when the result of a previous conversion has not been read from the result register at the time the next conversion is complete, or the next conversion is suspended until the previous result has been read. For applications which require less than 16 analog input channels, the remaining channel inputs can be used as digital input port pins. The A/D converter of the ST10F269 supports different conversion modes:

– **Single channel single conversion**: the analog level of the selected channel is sampled once and converted. The result of the conversion is stored in the ADDAT register.

– **Single channel continuous conversion**: the analog level of the selected channel is repeatedly sampled and converted. The result of the conversion is stored in the ADDAT register.

– **Auto scan single conversion**: the analog level of the selected channels are sampled once and converted. After each conversion the result is stored in the ADDAT register. The data can be transferred to the RAM by interrupt software management or using the powerful Peripheral Event Controller (PEC) data transfer.

– **Auto scan continuous conversion**: the analog level of the selected channels are repeatedly sampled and converted. The result of the conversion is stored in the ADDAT register. The data can be transferred to the RAM by interrupt software management or using the PEC data transfer.

– **Wait for ADDAT read mode**: when using continuous modes, in order to avoid to overwrite the result of the current conversion by the next one, the ADWR bit of ADCON control register must be activated. Then, until the ADDAT register is read, the new result is stored in a temporary buffer and the conversion is on hold.

– **Channel injection mode**: when using continuous modes, a selected channel can be converted in between without changing the current operating mode. The 10-bit data of the conversion are stored in ADRES field of ADDAT2. The current continuous mode remains active after the single conversion is completed

Table 26 : ADC Sample Clock and Conversion Clock (PQFP144 devices)

ADCTC	Conversion Clock t_{CC}		ADSTC	Sample Clock t_{SC}	
	TCL[1] = 1/2 x f_{XTAL}	At f_{CPU} = 40MHz		t_{SC} =	At f_{CPU} = 40MHz
00	TCL x 24	0.3µs	00	t_{CC}	0.3µs [2]
01	Reserved, do not use	Reserved	01	t_{CC} x 2	0.6µs [2]
10	TCL x 96	1.2 µs	10	t_{CC} x 4	1.2µs [2]
11	TCL x 48	0.6 µs	11	t_{CC} x 8	2.4µs [2]

Notes: 1. Section 21.4.5 -: Direct Drive for TCL definition.
2. t_{CC} = TCL x 24

Baud Rate Generation

The Baud rate generator is clocked by $f_{CPU}/2$. The timer is counting downwards and can be started or stopped through the global enable bit SSCEN in register SSCCON. Register SSCBR is the dual-function Baud Rate Generator/Reload register. Reading SSCBR, while the SSC is enabled, returns the content of the timer. Reading SSCBR, while the SSC is disabled, returns the programmed reload value. In this mode the desired reload value can be written to SSCBR.

Note Never write to SSCBR, while the SSC is enabled.

The formulas below calculate the resulting Baud rate for a given reload value and the required reload value for a given Baud rate:

$$\text{Baud rate}_{SSC} = \frac{f_{CPU}}{2 \times [(SSCBR) + 1]}$$

$$SSCBR = \left(\frac{f_{CPU}}{2 \times \text{Baud rate}_{SSC}} \right) - 1$$

(SSCBR) represents the content of the reload register, taken as unsigned 16-bit integer.

Table 32 lists some possible Baud rates against the required reload values and the resulting bit times for a 40MHz CPU clock.

Table 32 : Synchronous Baud Rate and Reload Values (PQFP144 devices)

Baud Rate	Bit Time	Reload Value
Reserved use a reload value > 0.	---	---
10M Baud	100ns	0001h
5M Baud	200ns	0003h
2.5M Baud	400ns	0007h
1M Baud	1µs	0013h
100K Baud	10µs	00C7h
10K Baud	100µs	07CFh
1K Baud	1ms	4E1Fh
306 Baud	3.26ms	FF4Eh

Table 33 lists some possible Baud rates against the required reload values and the resulting bit times for a 32MHz CPU clock.

.

Table 33 : Synchronous Baud Rate and Reload Values (TQFP144 devices)

Baud Rate	Bit Time	Reload Value
Reserved use a reload value > 0.	---	---
8MBaud	125ns	0001h
4MBaud	250ns	0003h
2MBaud	500ns	0007h
1MBaud	1µs	000Fh
500KBaud	2µs	001Fh
100KBaud	10µs	009Fh
10KBaud	100µs	030Ch
1K Baud	1ms	3E7Fh
244.14 Baud	5.24ms	FFFFh

BUSCON4 (FF1Ah / 8Dh) SFR Reset Value: 0000h

15	14	13	12	11	10	9	8	7 6	5	4	3 2 1 0
CSWEN4	CSREN4	RDYPOL4	RDYEN4	-	BUSACT4	ALECTL4	-	BTYP	MTTC4	RWDC4	MCTC
RW	RW	RW	RW		RW	RW		RW	RW	RW	RW

Notes: 1. BTYP (bit 6 and 7) are set according to the configuration of the bit I6 and I7 of PORT0 latched at the end of the reset sequence.
2. BUSCON0 is initialized with 0000h, if $\overline{EA}$ pin is high during reset. If $\overline{EA}$ pin is low during reset, bit BUSACT0 and ALECTRL0 are set ('1') and bit field BTYP is loaded with the bus configuration selected via PORT0.

MCTC	**Memory Cycle Time Control (Number of memory cycle time wait states)** 0 0 0 0: 15 wait states (Nber = 15 - [MCTC]) . . . 1 1 1 1: No wait state
RWDCx	**Read/Write Delay Control for BUSCONx** '0': With read/write delay: activate command 1 TCL after falling edge of ALE '1': No read/write delay: activate command with falling edge of ALE
MTTCx	**Memory Tristate Time Control** '0': 1 wait state '1': No wait state
BTYP	**External Bus Configuration** 0 0: 8-bit Demultiplexed Bus 0 1: 8-bit Multiplexed Bus 1 0: 16-bit Demultiplexed Bus 1 1: 16-bit Multiplexed Bus Note: For BUSCON0, BTYP bit-field is defined via PORT0 during reset.
ALECTLx	**ALE Lengthening Control** '0': Normal ALE signal '1': Lengthened ALE signal
BUSACTx	**Bus Active Control** '0': External bus disabled '1': External bus enabled (within the respective address window, see ADDRSEL)
RDYENx	**$\overline{READY}$ Input Enable** '0': External bus cycle is controlled by bit field MCTC only '1': External bus cycle is controlled by the $\overline{READY}$ input signal
RDYPOLx	**Ready Active Level Control** '0': Active level on the $\overline{READY}$ pin is low, bus cycle terminates with a '0' on READY pin, '1': Active level on the $\overline{READY}$ pin is high, bus cycle terminates with a '1' on $\overline{READY}$ pin.
CSRENx	**Read Chip Select Enable** '0': The CS signal is independent of the read command ($\overline{RD}$) '1': The CS signal is generated for the duration of the read command
CSWENx	**Write Chip Select Enable** '0': The $\overline{CS}$ signal is independent of the write command ($\overline{WR}$,$\overline{WRL}$,$\overline{WRH}$) '1': The $\overline{CS}$ signal is generated for the duration of the write command

EXICON (F1C0h / E0h ESFR Reset Value: 0000h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXI7ES		EXI6ES		EXI5ES		EXI4ES		EXI3ES		EXI2ES		EXI1ES		EXI0ES	
RW		RW		RW		RW		RW		RW		RW		RW	

EXIxES(x=7...0)	**External Interrupt x Edge Selection Field (x=7...0)** 0 0: Fast external interrupts disabled: standard mode EXxIN pin not taken in account for entering/exiting Power Down mode. 0 1: Interrupt on positive edge (rising) Enter Power Down mode if EXiIN = '0', exit if EXxIN = '1' (referred as 'high' active level) 1 0: Interrupt on negative edge (falling) Enter Power Down mode if EXiIN = '1', exit if EXxIN = '0' (referred as 'low' active level) 1 1: Interrupt on any edge (rising or falling) Always enter Power Down mode, exit if EXxIN level changed.

EXISEL (F1DAh / EDh) ESFR Reset Value: 0000h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXI7SS		EXI6SS		EXI5SS		EXI4SS		EXI3SS		EXI2SS		EXI1SS		EXI0SS	
RW		RW		RW		RW		RW		RW		RW		RW	

EXIxSS	**External Interrupt x Source Selection (x=7...0)** '00': Input from associated Port 2 pin. '01': Input from "alternate source". '10': Input from Port 2 pin ORed with "alternate source". '11': Input from Port 2 pin ANDed with "alternate source".

EXIxSS	Port 2 pin	Alternate Source
0	P2.8	CAN1_RxD
1	P2.9	CAN2_RxD
2	P2.10	RTCSI
3	P2.11	RTCAI
4...7	P2.12...15	Not used (zero)

XP3IC (F19Eh / CFh) [1] ESFR Reset Value: --00h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	**XP3IR**	XP3IE	XP3ILVL				GLVL	
								RW	RW	RW				RW	

Note: 1. XP3IC register has the same bit field as xxIC interrupt registers

Symbol	Parameter	Maximum CPU Clock = 32MHz		Variable CPU Clock 1/2 TCL = 1 to 32MHz		Unit
		Minimum	Maximum	Minimum	Maximum	
t_{44} CC	Address float after $\overline{RdCS}$, $\overline{WrCS}$ (with RW delay) 1	–	0	–	0	ns
t_{45} CC	Address float after $\overline{RdCS}$, $\overline{WrCS}$ (no RW delay) 1	–	15.625	–	TCL	ns
t_{46} SR	$\overline{RdCS}$ to Valid Data In (with RW delay)	–	7.25 + t_C	–	2TCL - 24 + t_C	ns
t_{47} SR	$\overline{RdCS}$ to Valid Data In (no RW delay)	–	22.875 + t_C	–	3TCL - 24 + t_C	ns
t_{48} CC	$\overline{RdCS}$, $\overline{WrCS}$ Low Time (with RW delay)	21.25 + t_C	–	2TCL - 10 + t_C	–	ns
t_{49} CC	$\overline{RdCS}$, $\overline{WrCS}$ Low Time (no RW delay)	36.875 + t_C	–	3TCL - 10 + t_C	–	ns
t_{50} CC	Data valid to $\overline{WrCS}$	17.25 + t_C	–	2TCL - 14+ t_C	–	ns
t_{51} SR	Data hold after $\overline{RdCS}$	0	–	0	–	ns
t_{52} SR	Data float after $\overline{RdCS}$ 1	–	11.25 + t_F	–	2TCL - 20 + t_F	ns
t_{54} CC	Address hold after $\overline{RdCS}$, $\overline{WrCS}$	11.25 + t_F	–	2TCL - 20 + t_F	–	ns
t_{56} CC	Data hold after $\overline{WrCS}$	11.25 + t_F	–	2TCL - 20 + t_F	–	ns

1. Partially tested, guaranted by design characterization.

ERRATA SHEET

ST10F269Zxxx-D
LIMITATIONS AND CORRECTIONS

1 - DESCRIPTION

This Errata sheet describes the functional and electrical problems known in the D revision of the ST10F269Zxxx.

The revision number can be found in the third line on the ST10F269 package. It looks like: 'xxxxxxxxx D' where "D" identifies the revision number.

2 - FUNCTIONAL PROBLEMS

The following malfunctions are known in this step:

2.1 - PWRDN.1 - Execution of PWRDN Instruction

When instruction PWRDN is executed while pin $\overline{\text{NMI}}$ is at a high level (if PWRDCFG bit is clear in SYSCON register) or while at least one of the port 2 pins used to exit from power-down mode (if PWRD-CFG bit is set in SYSCON register) is at the active level, power down mode is not entered, and the PWRDN instruction is ignored.

However, under the conditions described below, the PWRDN instruction is not ignored, and no further instructions are fetched from external memory, i.e. the CPU is in a quasi-idle state.

This problem only occurs in the following situations:

a) The instructions following the PWRDN instruction are located in an external memory, and a **multiplexed bus** configuration **with memory tristate waitstate** (bit MT-TCx = 0) is used.

   Or

b) The instruction preceeding the PWRDN instruction **writes** to external memory or an XPeripheral (XRAM,CAN), and the instructions following the PWRDN instruction are located in external memory. In this case, the problem occurs for any bus configuration.

Note: The on-chip peripherals are still working correctly, in particular the Watchdog Timer, if not disabled, resets the device upon an overflow. Interrupts and PEC transfers, however, cannot be processed. In case NMI is asserted low while the device is in this quasi-idle state, power-down mode is entered.

No problem occurs if the $\overline{\text{NMI}}$ pin is low (if PWRDCFG = 0) or if all P2 pins used to exit from power-down mode are at inactive level (if PWRDCFG = 1): the chip normally enters powerdown mode.

Workaround:

Ensure that no instruction that writes to external memory or an XPeripheral preceeds the PWRDN instruction, otherwise insert a NOP instruction in front of PWRDN. When a multiplexed bus with memory tristate wait state is used, the PWRDN instruction must be executed from internal RAM or XRAM.