! "

$ %% % #&

' (( ) # $ %% %
#

- ,	.
$ -	, &/
$, #	&' 0
,	1/ 2#
$	$! 3 0 , , $ 4! 5 4, ! 5
-	- . 5 - 6 6 7
3 .	&&&
- , #	) 8' 90 :) 8' 9 * %
-	<=! , 2
- 5 . , #	
5! , #	&) 9 * %
> , :> > ;	1 8> ? 8 8>
7 $	! 7 &' * &/
.	
.	1/ @ ? &) 8@ : ! ;
	,
- $	&11 =A<-
, 7 -	&11 =A<- :) / *) / ;
- 45=	B * &/) ' C#) " (

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:
....
```

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

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.

RP0H (F108h / 84h) ESFR Reset Value: --XXH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	CLKSEL			SALSEL		CSSEL		WRC
								R [1 - 2]			R [2]		R [2]		R [2]

WRC [2]	Write Configuration Control '0': Pin $\overline{WR}$ acts as $\overline{WRL}$, pin $\overline{BHE}$ acts as $\overline{WRH}$ '1': Pins $\overline{WR}$ and $\overline{BHE}$ retain their normal function
CSSEL [2]	Chip Select Line Selection (Number of active $\overline{CS}$ outputs) 0 0: 3 $\overline{CS}$ lines: $\overline{CS}2$...$\overline{CS}0$ 0 1: 2 $\overline{CS}$ lines: $\overline{CS}1$...$\overline{CS}0$ 1 0: No $\overline{CS}$ line at all 1 1: 5 $\overline{CS}$ lines: $\overline{CS}4$...$\overline{CS}0$ (Default without pull-downs)
SALSEL [2]	Segment Address Line Selection (Number of active segment address outputs) 0 0: 4-bit segment address: A19...A16 0 1: No segment address lines at all 1 0: 8-bit segment address: A23...A16 1 1: 2-bit segment address: A17...A16 (Default without pull-downs)
CLKSEL [1 - 2]	System Clock Selection 000: $f_{CPU} = 2.5 \times f_{OSC}$ 001: $f_{CPU} = 0.5 \times f_{OSC}$ 010: $f_{CPU} = 1.5 \times f_{OSC}$ 011: $f_{CPU} = f_{OSC}$ 100: $f_{CPU} = 5 \times f_{OSC}$ 101: $f_{CPU} = 2 \times f_{OSC}$ 110: $f_{CPU} = 3 \times f_{OSC}$ 111: $f_{CPU} = 4 \times f_{OSC}$

Notes: 1. RP0H.7 to RP0H.5 bits are loaded only during a long hardware reset. As pull-up resistors are active on each Port P0H pins during reset, RP0H default value is "FFh".

2. These bits are set according to Port 0 configuration during any reset sequence.

3. RP0H is a read only register.

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

	'1': The on-chip Real Time Clock is enabled and can be accessed.

When both CAN are disabled via XPERCON setting, then any access in the address range 00'EE00h - 00'EFFFh will be directed to external memory interface, using the BUSCONx register corresponding to address matching ADDRSELx register. P4.4 and P4.7 can be used as General Purpose I/O when CAN2 is not enabled, and P4.5 and P4.6 can be used as General Purpose I/O when CAN1 is not enabled.

The default XPER selection after Reset is identical to XBUS configuration of ST10C167: XCAN1 is enabled, XCAN2 is disabled, XRAM1 (2K Byte compatible XRAM) is enabled, XRAM2 (new 8K Byte XRAM) is disabled.

Register XPERCON cannot be changed after the global enabling of XPeripherals, i.e. after setting of bit XPEN in SYSCON register.

In EMUlation mode, all the XPERipherals are enabled (XPERCON bit are all set).

When the Real Time Clock is disabled (RTCEN = 0), the clock oscillator is switch off if ST10 enters in power-down mode. Otherwise, when the Real Time Clock is enabled, the bit RTCOFF of the RTCCON register allows to choose the power-down mode of the clock oscillator.

21.4.10 - Multiplexed Bus

V_{DD} = 5V ± 10%, V_{SS} = 0V, T_A = -40 to +125°C, C_L = 50pF,
ALE cycle time = 6 TCL + $2t_A$ + t_C + t_F (75ns at 40MHz CPU clock without wait states, PQFP144 devices).

Table 46 : Multiplexed Bus Characteristics (PQFP144 devices)

Symbol		Parameter	Max. CPU Clock = 40MHz		Variable CPU Clock 1/2 TCL = 1 to 40MHz		Unit
			min.	max.	min.	max.	
t_5	CC	ALE high time	4 + t_A	–	TCL - 8.5 + t_A	–	ns
t_6	CC	Address setup to ALE	2 + t_A	–	TCL - 10.5 + t_A	–	ns
t_7	CC	Address hold after ALE 1	4 + t_A	–	TCL - 8.5 + t_A	–	ns
t_8	CC	ALE falling edge to $\overline{RD}$, $\overline{WR}$ (with RW-delay)	4 + t_A	–	TCL - 8.5 + t_A	–	ns
t_9	CC	ALE falling edge to $\overline{RD}$, $\overline{WR}$ (no RW-delay)	-8.5 + t_A	–	-8.5 + t_A	–	ns
t_{10}	CC	Address float after $\overline{RD}$, $\overline{WR}$ (with RW-delay) 1	–	6	–	6	ns
t_{11}	CC	Address float after $\overline{RD}$, $\overline{WR}$ (no RW-delay) 1	–	18.5	–	TCL + 6	ns
t_{12}	CC	$\overline{RD}$, $\overline{WR}$ low time (with RW-delay)	15.5 + t_C	–	2 TCL -9.5 + t_C	–	ns
t_{13}	CC	$\overline{RD}$, $\overline{WR}$ low time (no RW-delay)	28 + t_C	–	3 TCL -9.5 + t_C	–	ns
t_{14}	SR	$\overline{RD}$ to valid data in (with RW-delay)	–	6 + t_C	–	2 TCL - 19 + t_C	ns
t_{15}	SR	$\overline{RD}$ to valid data in (no RW-delay)	–	18.5 + t_C	–	3 TCL - 19 + t_C	ns
t_{16}	SR	ALE low to valid data in	–	18.5 + t_A + t_C	–	3 TCL - 19 + t_A + t_C	ns
t_{17}	SR	Address/Unlatched $\overline{CS}$ to valid data in	–	22 + $2t_A$ + t_C	–	4 TCL - 28 + $2t_A$ + t_C	ns
t_{18}	SR	Data hold after $\overline{RD}$ rising edge	0	–	0	–	ns
t_{19}	SR	Data float after $\overline{RD}$ 1	–	16.5 + t_F	–	2 TCL - 8.5 + t_F	ns
t_{22}	CC	Data valid to $\overline{WR}$	10 + t_C	–	2 TCL -15 + t_C	–	ns
t_{23}	CC	Data hold after $\overline{WR}$	4 + t_F	–	2 TCL - 8.5 + t_F	–	ns
t_{25}	CC	ALE rising edge after $\overline{RD}$, $\overline{WR}$	15 + t_F	–	2 TCL -10 + t_F	–	ns
t_{27}	CC	Address/Unlatched $\overline{CS}$ hold after $\overline{RD}$, $\overline{WR}$	10 + t_F	–	2 TCL -15 + t_F	–	ns
t_{38}	CC	ALE falling edge to Latched $\overline{CS}$	-4 - t_A	10 - t_A	-4 - t_A	10 - t_A	ns
t_{39}	SR	Latched $\overline{CS}$ low to Valid Data In	–	18.5 + t_C + $2t_A$	–	3 TCL - 19 + t_C + $2t_A$	ns
t_{40}	CC	Latched $\overline{CS}$ hold after $\overline{RD}$, $\overline{WR}$	27 + t_F	–	3 TCL - 10.5 + t_F	–	ns

Figure 71 : External Memory Cycle: Multiplexed Bus, With / Without Read / Write Delay, Normal ALE

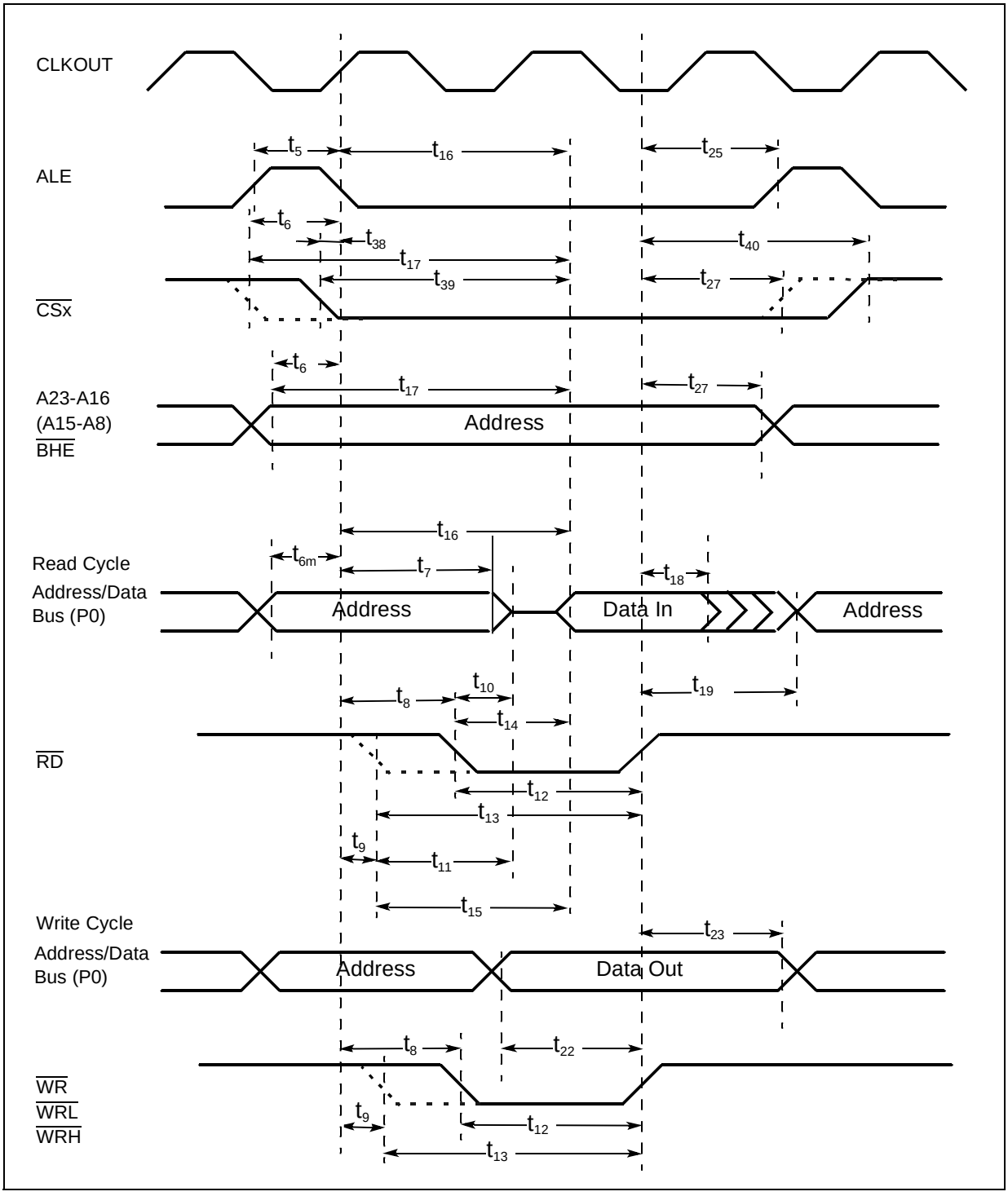


Figure 74 : External Memory Cycle: Multiplexed Bus, With / Without Read / Write Delay, Extended ALE, Read / Write Chip Select

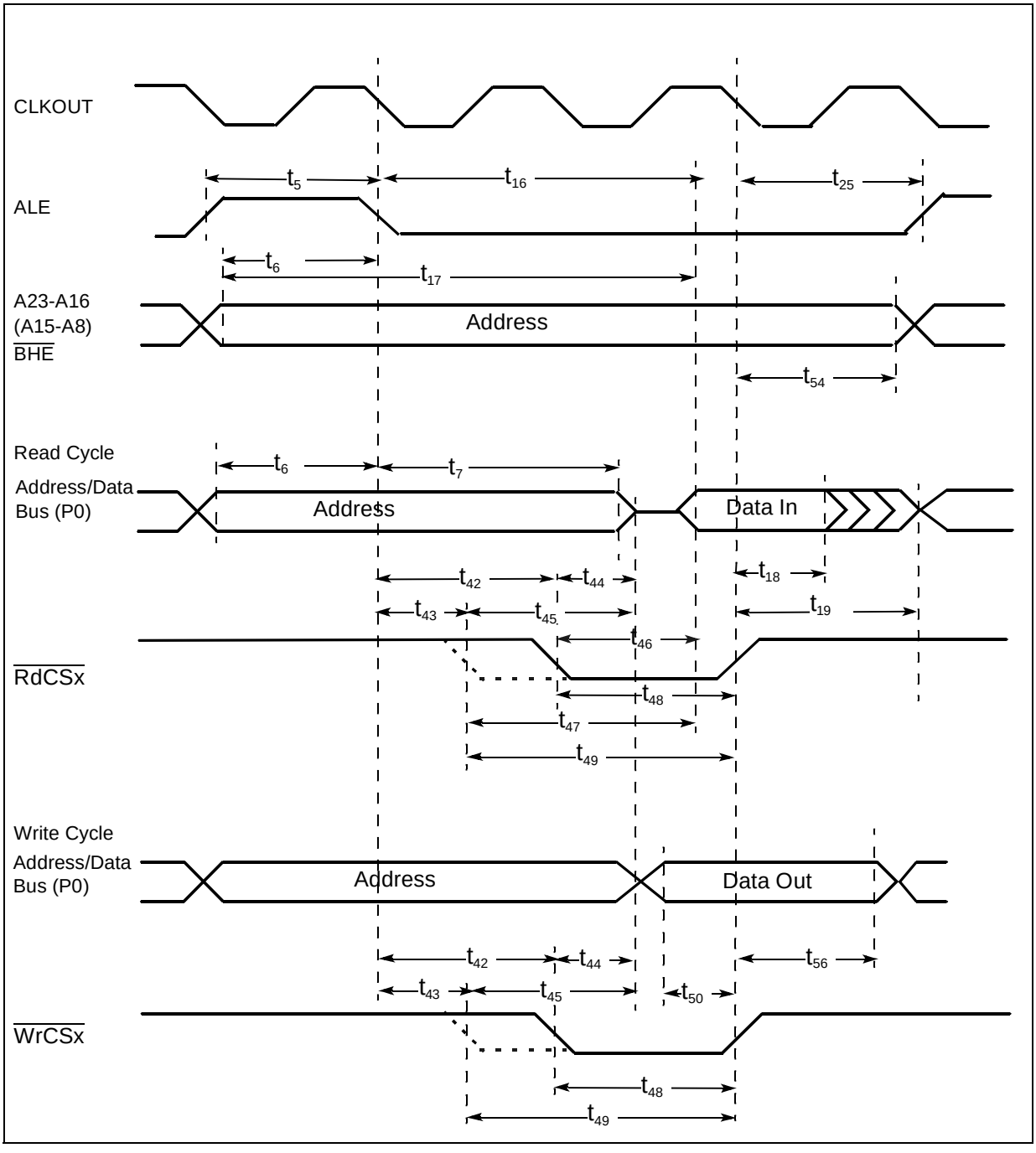
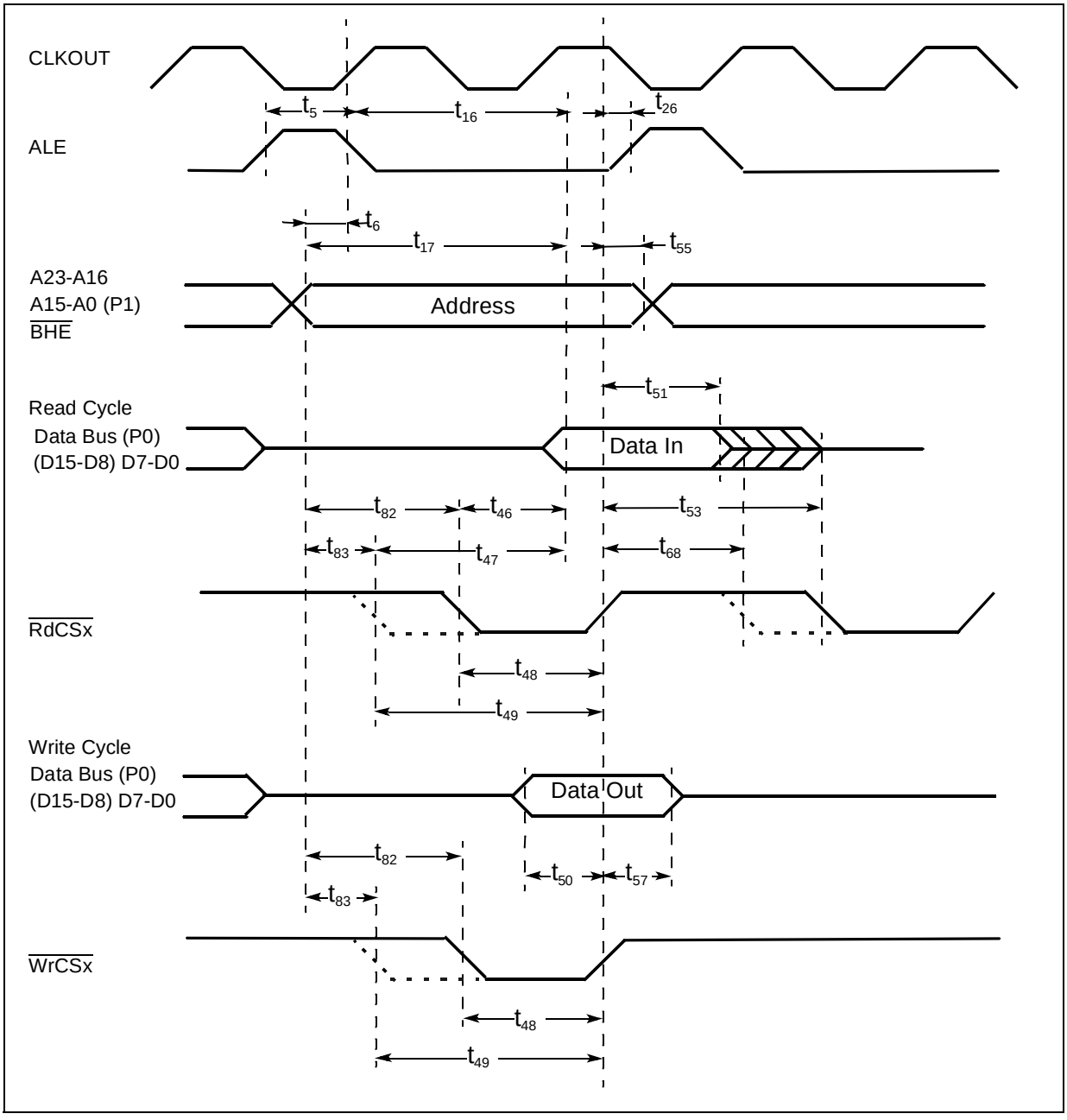


Figure 77 : External Memory Cycle: Demultiplexed Bus, With / Without Read / Write Delay, Normal ALE, Read / Write Chip Select



2.2 - MAC.9 - CoCMP Instruction Inverted Operands

The ST10 Family Programming Manual describes the CoCMP instruction as: subtracts a 40-bit signed operand from th 40-bit accumulator content (acc - op2\op1), and updates the N, Z and C flags in the MSW register, leaving the accumulator unchanged. On the device the reverse operation (op2\op1 - acc) has been implemented in the Mac Unit. Therefore, the N and C flags are set according to the reverse operation (Z flag is not affected).

Workaround:

Change interpretation of the N and C flags in the MSW register.

Example:

```
MOV     R12, #07h
MOV     R13, #06h
MOV     R14, #0
CoLOAD  R14, R12        ; Accumulator = 70000h
CoCMP   R14, R13        ; Compares 70000h to 60000h
```

Here the content of MSW is 0500h, i.e. C = 1, Z = 0 and N = 1.

To test if the Accumulator was greater than or equal the compared value, the "normal" test, according to the description in the ST10 Programming Manual, would be:

```
JNB     MSW.10, Greater ; If C flag cleared, then greater than or equal
```

With the implementation, this test does not provide the expected result.

To obtain the correct comparison, use instead:

```
JB      MSW.10, Greater ; C flag set: 60000h < 70000h (60000h-70000h implemented)
                        ; i.e. the accumulator is greater than or equal compared value
```

2.3 - MAC.10 - E Flag Evaluation for CoSHR and CoASHR Instructions when Saturation Mode is Enabled

The Logical and the Arithmetic Right Shift instructions (CoSHR/CoASHR) are specififed not to be affected by the saturation mode (MS bit of the MCW register): the shift operation is always made on the 40 bits of the accumulator. The result shifted in the accumulator is never saturated. Only when the saturation mode is enabled, the evaluation of the E Flag (in the MSW register) is erroneous.

Comment to the example:

In this example below (Table 1), the E Flag is kept cleared however MAE is used: bit 0 of MAE has been shifted into bit 15 of MAH. The MAE part has been used and it contents significant bits but the E Flag has not been set.

The content of the flags is given after the execution of the instruction.

Table 1 : MAC.10 Example

MS Bit is Set, Saturation Mode is Enabled		Status of Flags After Instruction Execution								
Code		Accumulator Value (Hexa.)			SL	E	SV	C	Z	N	Remark
MOV	R5, #5555h	--	----	----	-	-	-	-	-	-	-
CoLOAD	R5, R5	00	5555	5555	0	0	0	0	0	0	Right
NOP		00	5555	5555	0	0	0	0	0	0	Right
MOV	MSW, #007Fh	7F	5555	5555	0	0	0	0	0	0	Right
NOP		7F	5555	5555	0	0	0	0	0	0	Right
CoSHR	#1	3F	AAAA	AAAA	0	0*	0	0	0	0	*E is wrong

Workaround:

If the MAE flag is used, the saturation mode must be disabled before running Logical and/or Arihmetic Right Shift instructions and re-enable just after.