80C51 External Memory Interfacing AN457

INTRODUCTION

The ’51 family is arguably the most popular 8-bit embedded controller lineup thanks to efficient yet powerful architecture, multi-sourcing by the world’s top semiconductor companies and unprecedented third-party tool support. A byproduct of the chip’s popularity is a lot of technical know-how embodied in legions of experienced system designers and software programmers.

However, since the chips continue to attract new applications and customers, it shouldn’t be taken for granted that ‘everyone’ already knows how to design it in. Evidence to the contrary is the fact that many of the FAQs (Frequently Asked Questions) continue to be frequently asked.

This application note is written to assist those designers new to the ’51 family who typically fit into one or more of the following categories:

– Designers of extremely cost conscious systems that could previously only afford discrete logic or 4-bit solutions now upgrading their designs with superior price/performance and easier to program 8-bit microprocessors.
– Designers that previously used different micros and are now switching to the ’51 family.
– New designers, most of whom were quite young at the time of the ’51 introduction when design techniques were actively disseminated.

In addition, even experienced designers might find it useful to review this application note, since many of the traditional design techniques and ‘conventional wisdom’ have been superceded by faster CPUs and memories, changing timing specifications and more sophisticated application requirements. Indeed, those who have simply done it the way somebody else did it before are advised to confirm the validity of their design assumptions.

The application note starts with a basic description of the ’51 family memory organization and expansion bus characteristics. Since all the technical details are completely documented in the data sheet and other application notes (see the references section at the end), this section simply highlights the basic operation and timing considerations.

Next, the operation and characteristics of the most often used memories – specifically JEDEC standard byte-wide EPROMs and SRAMs – are described. Specifications are presented for a variety of actual memories.

Having described the CPU and memories, timing analysis is performed for a typical system configuration. The goal of this section is to illustrate how to answer one of the most frequently asked FAQs – “What speed memory should I use with my xxMHz CPU?”.

’51 FAMILY MEMORY ORGANIZATION

Though dozens of derivatives, are offered, all ’51 family members share a common memory architecture and similar expansion bus. For this application note, the 80C31/8XC51 (Figure 1) will be used. The only difference between these two devices is that the 80C31 has no on-chip instruction memory while the 8XC51 has 4KB of ROM (80C51) or EPROM (87C51) instruction memory on-chip. The entire range of ’51 family derivatives encompasses devices with 0KB (80C31) to 64KB (8XCE560) of on-chip instruction memory of various types including ROM, EPROM (window package), OTP (One Time Programmable), and even EEPROM.

A primary characteristic is that the ’51 is logically a ‘Harvard’ machine referring to an organization consisting of separate instruction and data buses. One key byproduct is that the ’51 family offers twice the memory expandability (64KB each of code and data) compared to most other 8-bit micros (typically 64KB total).

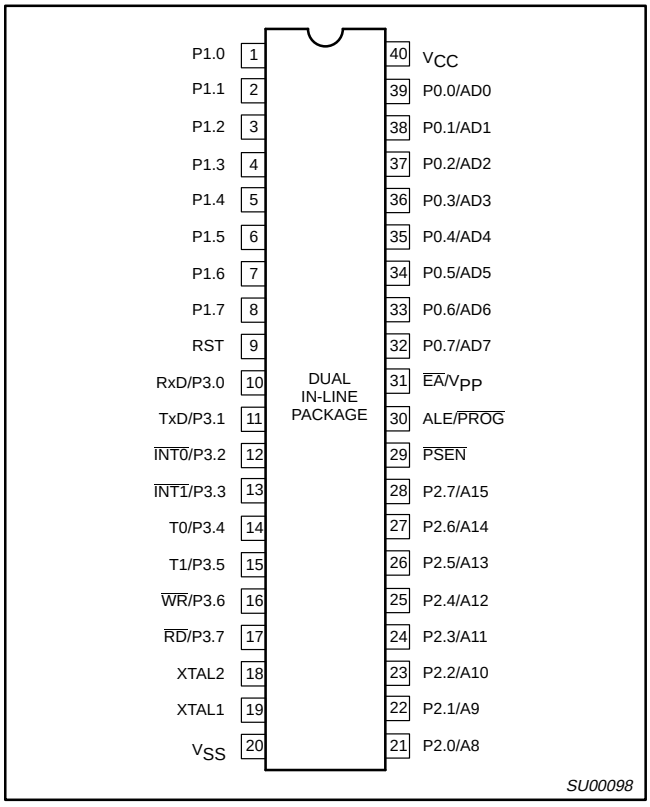

Figure 1. 80C31/8XC51 Pinout

As shown in Figure 2, the CPU operates in one of two modes determined at reset (RST pin) by the state of the $\overline{EA}$ (External Access) pin. If $\overline{EA}$ is asserted, on-chip instruction (but not data) memory is disabled and the entire 64KB of instruction space is accessed externally (this is the only option for the 80C31). Otherwise ($\overline{EA}$ deasserted), on-chip instruction memory is enabled and only addresses beyond the end of on-chip instruction memory (i.e., ≥ 1000H for the 8XC51) are accessible externally.

The situation for on-chip data memory (i.e., RAM) is somewhat different. First, all '51 family devices include a basic complement of on-chip RAM comprised of CPU register banks, SFRs (Special Function Registers, i.e., built-in I/O functions such as UART, timer, etc.) and general purpose RAM. The on-chip data memory for the 80C31/8XC51 is shown in Figures 3 and 4. By convention, which is used in this application note, a CPU said to have 'xx' bytes of RAM (which varies from 64 bytes to 1.5 KB across the '51 family) actually contains 'xx' bytes of 'general purpose RAM' in addition to the space (128 bytes) allocated to SFRs.

Unlike instruction memory, the state of the $\overline{EA}$ pin at reset doesn't affect on-chip data RAM which is always enabled and accessible. Another difference is related to the way the presence of on-chip RAM affects the external data memory space. For CPUs with up to 256 bytes of on-chip RAM, the full 64KB external data space is available. Devices with more than 256 bytes of RAM map the excess portion (i.e., 768 bytes for a CPU with 1K bytes of RAM) to the bottom of the external address space (Figure 5). In this case, an SFR bit (ARD – Auxillary RAM Disable) determines whether the accesses to the lower space are on- or off-chip.

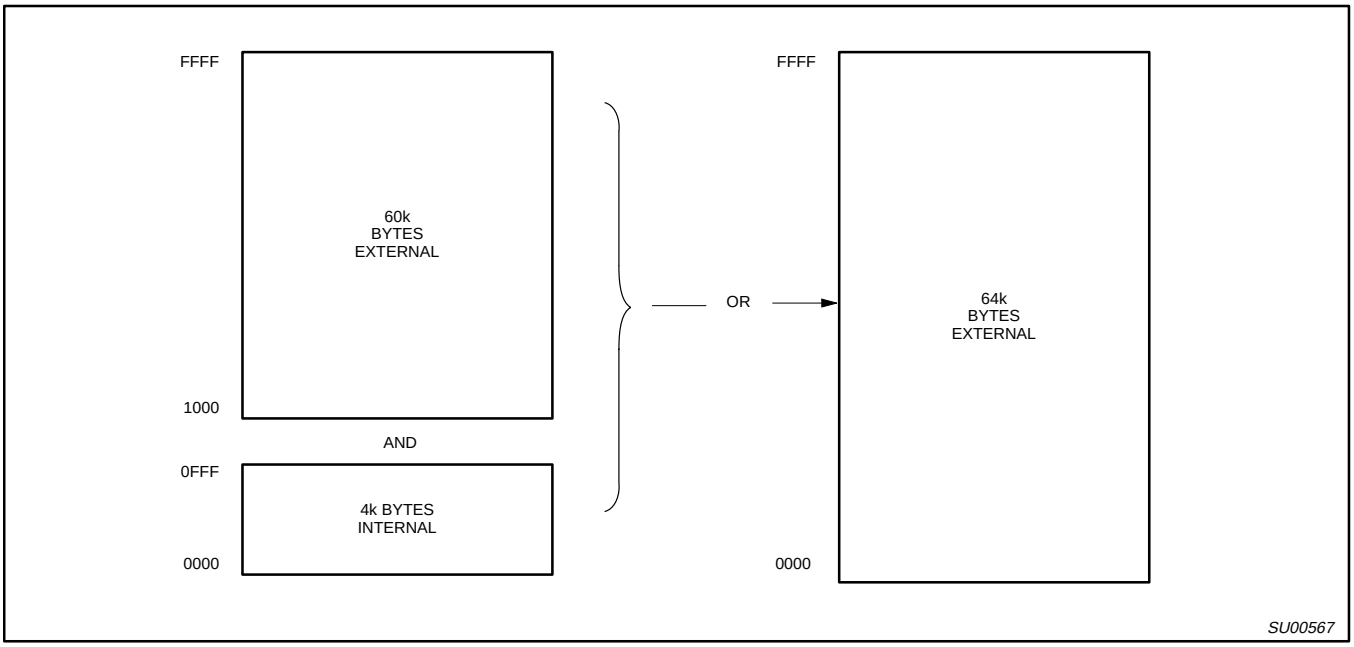

Figure 2. Program/Data Memory Map

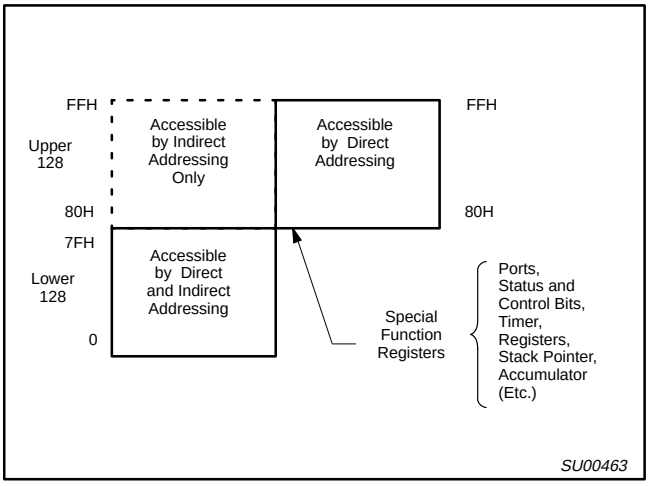

Figure 3. On-chip RAM Memory Map ≥ 256 Bytes Internal Data Memory

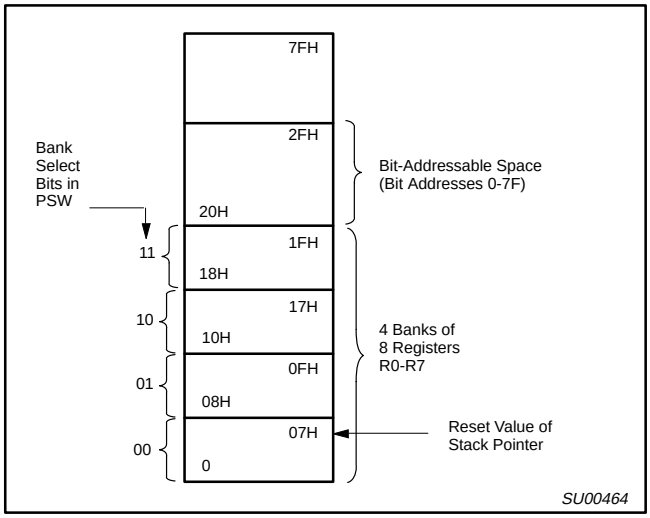

Figure 4. On-chip RAM Memory Map ≥ 256 Bytes Lower 128 Bytes of Internal RAM

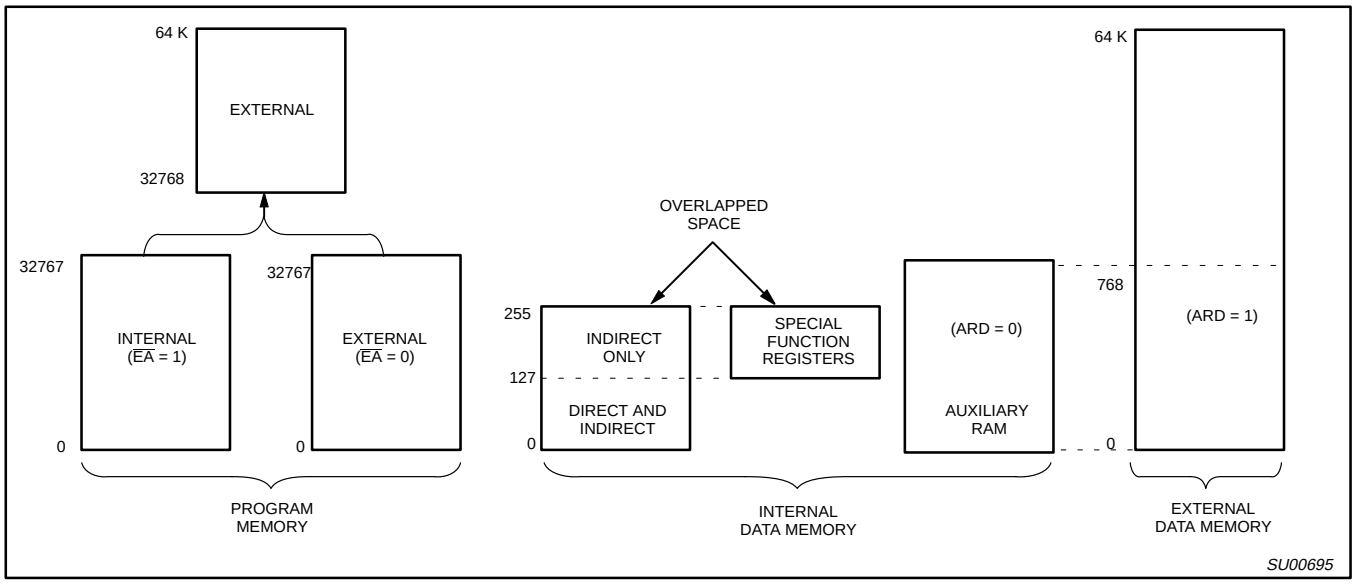

Figure 5. On-chip RAM Memory Map > 256 Bytes

80C31/8XC51 EXPANSION BUS INTERFACE

'51 family devices with 40 or more pins generally feature four I/O ports (P0–P3). P0, P2 and a portion of P3 (two pins, $\overline{RD}$ and $\overline{WR}$), along with dedicated ALE (Address Latch Enable) and $\overline{PSEN}$ (Program Store Enable) pins, comprise the expansion bus interface.

P0.0–P0.7

For both program and data access, P0 is used as a multiplexed address/data bus (AD0–7) that outputs the low order address bits (A0–A7) and inputs/outputs the 8-bit data (D0–D7). Note that P0 is open collector, so pull-up resistors are typically required when interfacing to external memory or I/O chips.

P2.0–P2.7

For all external program accesses, P2 outputs the high order address bits (A8–A15). The same is true for external data accesses with 16-bit addresses (MOVX A,@DPTR and MOVX @DPTR,A). However, external data accesses with 8-bit addresses (MOVX A,@Ri and MOVX @Ri,A) do not affect P2.

ALE (Address Latch Enable)

ALE is used to demultiplex the AD0–7 bus. At the beginning of the external cycle ALE is high and the CPU emits A0–A7 which should be externally latched when ALE goes low. Note that ALE is always active, even during internal program and data accesses.

$\overline{PSEN}$ (Program Store Enable)

$\overline{PSEN}$ is the read strobe for external instruction access. Unlike ALE, $\overline{PSEN}$ is not asserted during internal accesses.

$\overline{RD}$ (Data Read)

$\overline{RD}$ is the read strobe for external data access and (like $\overline{PSEN}$) is not asserted during internal accesses.

$\overline{WR}$ (Data Write)

$\overline{WR}$ is the write strobe for external data access and (like $\overline{PSEN}$ and $\overline{RD}$) is not asserted during internal accesses.

Thus, there are three types of external access – instruction read ($\overline{PSEN}$), data read ($\overline{RD}$) and data write ($\overline{WR}$) as shown in Figures 6, 7, and 8.

From this brief description, a number of system design implications can be drawn.

ALE is essentially a continuous clock that runs at 1/6 the oscillator frequency regardless of the mix of internal and external accesses. However, note that one ALE cycle is skipped during external data access.

The skipping of the ALE cycle and assertion of $\overline{RD}$ or $\overline{WR}$ only occur during external data access. The MOVX instruction, and only the MOVX instruction, performs external data access. Thus, if the program contains no MOVX instructions, ALE can be used as a timebase (i.e., no skipping) and $\overline{RD}$ and $\overline{WR}$ as general purpose outputs.

External program reads ($\overline{PSEN}$), whatever the address or cause (i.e., fetch beyond the end of internal instruction memory or $\overline{EA}$ pin asserted at reset) always use 16-bit addresses and thus require all pins of P0 and P2. [P0 & P2 can be used for general purpose I/O during non-$\overline{PSEN}$ times? P0 SFR is overwritten by $\overline{PSEN}$ but what about P2?]

External data accesses affect on P0 and P2 is a little more complicated. 16-bit address accesses (MOVX using DPTR) always drive P0 and P2 with A0–A15. However, 8-bit address accesses (MOVX using Ri) instead drive P2 with the value programmed into the P2 SFR, i.e., P2 is essentially general purpose I/O during an 8-bit address external data access.

However, while the P0 SFR is overwritten by any external data access, the P2 SFR is only modified temporarily for the duration of a 16-bit address (DPTR) access. Subsequently, the P2 SFR is restored to whatever value it contained prior to the external data access.

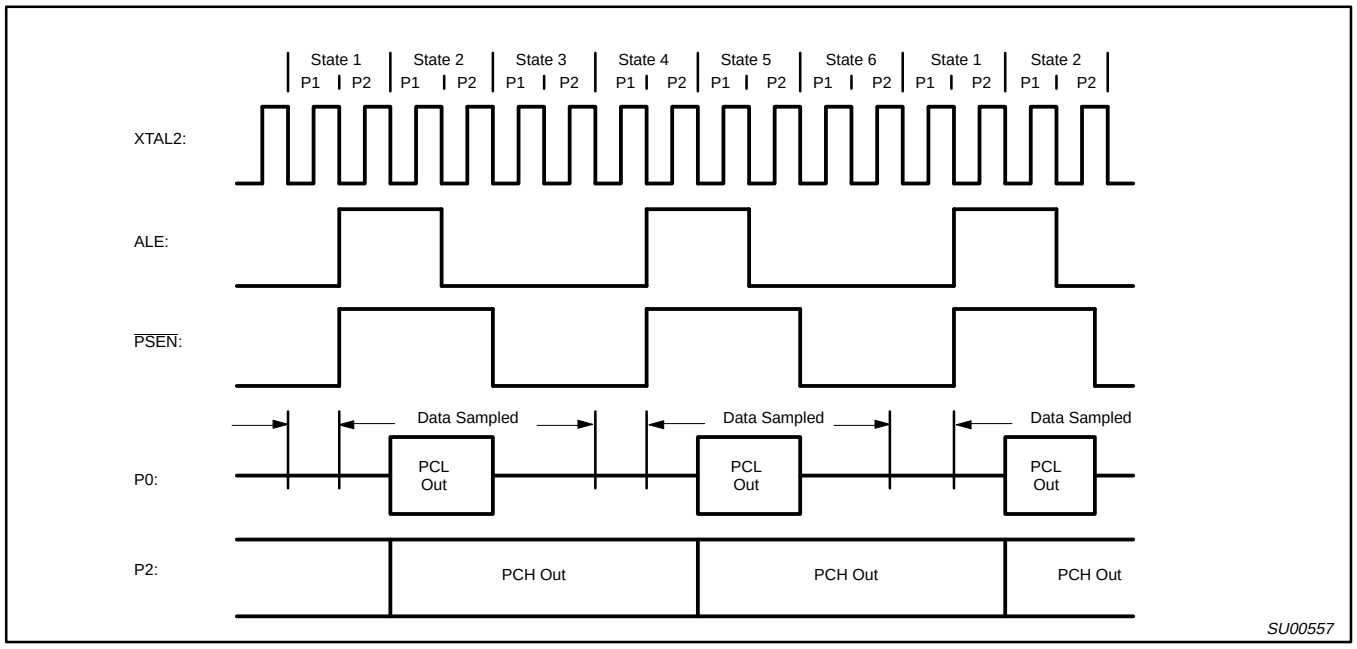

Figure 6. External Program Memory Fetches

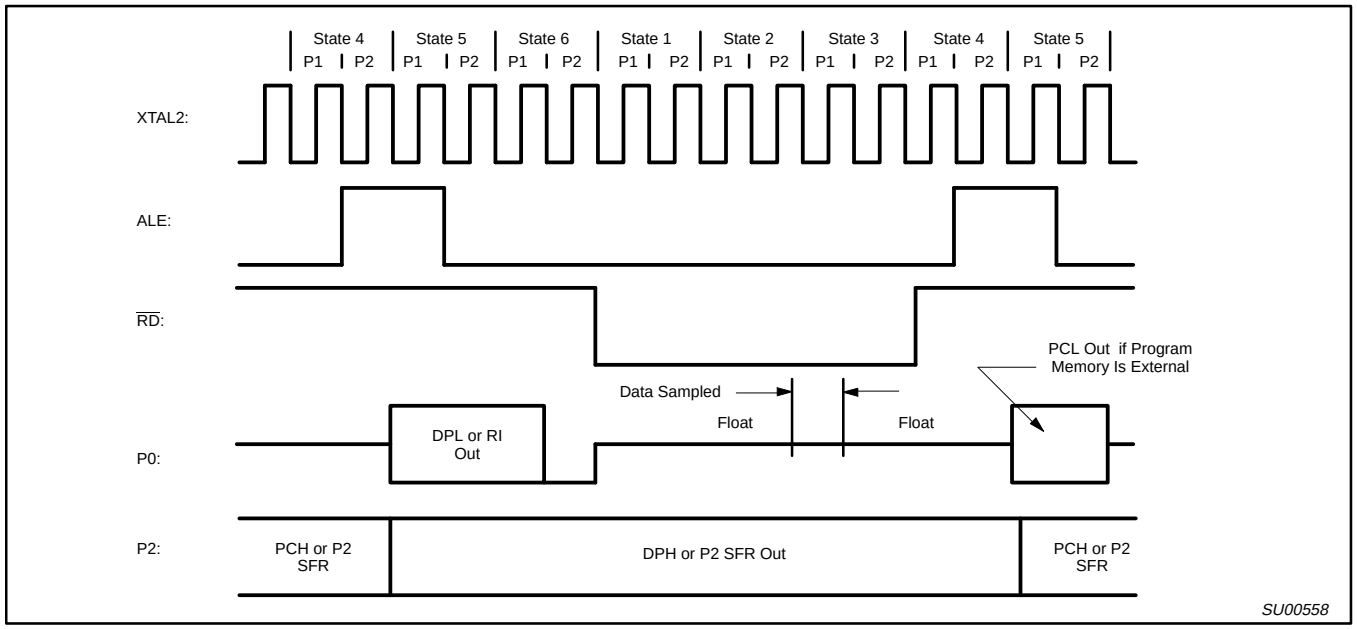

Figure 7. External Data Memory Read Cycle

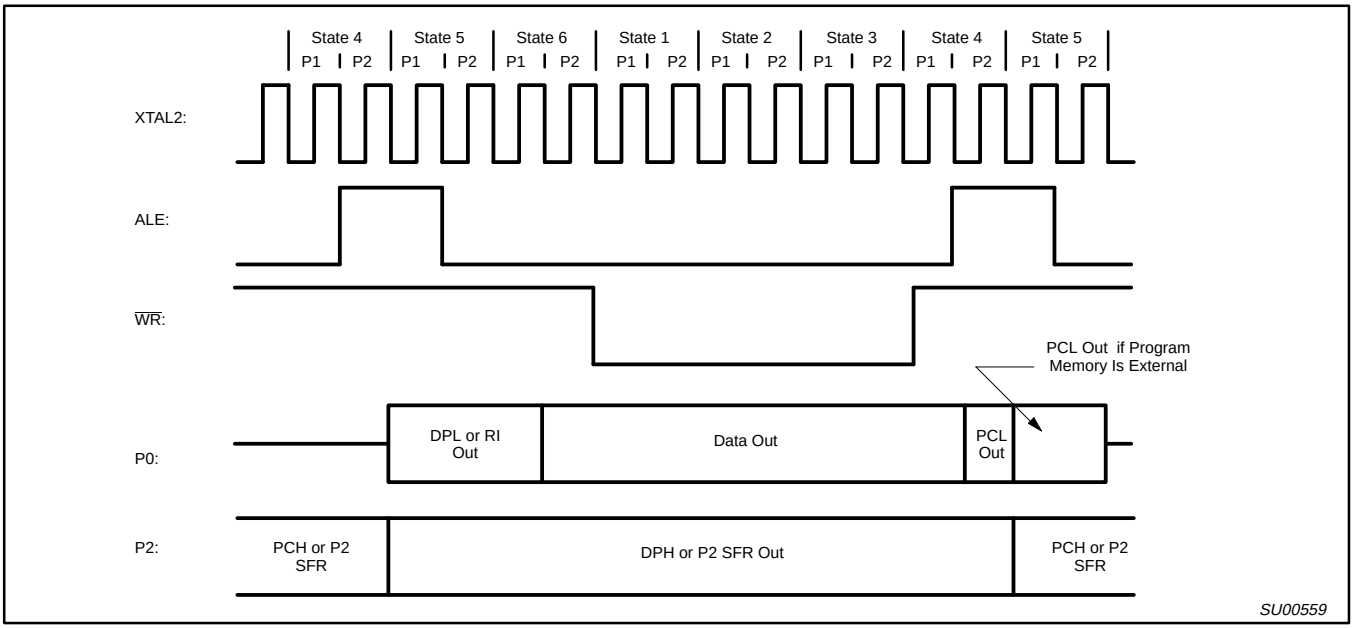

Figure 8. External Data Memory Write Cycle

Table 1 summarizes the usage and modification of the P0 and P2 SFRs depending on the cycle type.

Exploiting this information, system designers can make optimal use of all pins depending on the system configuration. For instance, systems that make no external program accesses and only 8-bit address external data accesses are free to use P2 for general purpose I/O. Just remember that CPUs with more than 256 bytes of on-chip RAM must use 16-bit addresses (DPTR) to perform external data access.

Table 1. P0 and P2 SFR Usage During External Memory Access

EXTERNAL ACCESS TYPE	DURING ACCESS		AFTER ACCESS	
	P0 SFR	P2 SFR	P0 SFR	P2 SFR
Instruction Access	0FFH	0FFH	0FFH	0FFH
Data Access – 8-bit Address (MOVX using Ri)	0FFH	Prev. Value	0FFH	Prev. Value
Data Access – 16-bit Address (MOVX using DPTR)	0FFH	0FFH	0FFH	Prev. Value

EPROMs

Before continuing, it should be pointed out that the original reason the '51 supported external code – limited ROM-only on-chip space – is hardly applicable today. Windowed EPROM-based CPUs appeared long ago and recently low-cost no-window OTP EPROM devices have become quite popular. Now, even EEPROM-based devices are starting to appear. As for capacity, Philips currently offers derivatives with up to 64KB of on-chip memory including EPROM (windowed and OTP) and ROM.

That being said, there are still situations calling for external code memory. Systems requiring relatively high capacity, say 32KB–64KB and beyond, may still be best served by external EPROM, though the advantage will continue to wane. Conversely, due to the dynamics of the memory market in which 'old' devices quickly become non-price competitive, low-capacity applications (say 1KB–16KB) are, even without considering the intrinsic advantages of single-chip form factor, arguably best served with on-chip memory.

Using external EPROM can be appropriate when an existing design is simply being upgraded to higher speed without PCB change. Also, certain products such as 'general-purpose' single-board computers need external memory since different code will be programmed by each customer. Along the same lines, 'Rev.0' products may use external EPROM until the code stabilizes, after which the design can migrate to a single-chip. Finally, some applications, such as handheld translators or dictionaries, call for large capacity (EP)ROMs.

The JEDEC (Joint Electron Devices Engineering Council – an international standards body) standard defines the pinout applied to a number of memory technologies including ROM, EPROM, bulk- and byte-erasable EEPROM, etc. However, though the general function is well-defined, detailed timing specifications are left to each manufacturer and may differ slightly. Figure 9 shows the pinout, Figure 10 shows the block diagram, and Table 2 shows the truth table of a 512k bit (64KB) EPROM organized as 64Kx8.

The bulk of the functionality is defined by the sixteen address lines A0–A15 that select a byte for output on the eight data lines O0–O7. Only two control lines are necessary – Chip Enable ($\overline{CE}$) and Output Enable ($\overline{OE}$). Notice that deasserting $\overline{CE}$ places the EPROM in 'standby' mode, consuming typically 1/2 to 1/3 the active ($\overline{CE}$ asserted) power. Also, observe that both $\overline{CE}$ and OE must be asserted to enable the data output drivers.

Referring to the timing diagram (Figure 11) shows that operation of the EPROM is quite simple, typically characterized by only a few specifications.

The first, t_{AA}, defines the maximum time after the address stabilizes that the EPROM will return valid data. This parameter is commonly referred to as the 'access time' of the device.

Similarly, t_{CE} defines the maximum time after the $\overline{CE}$ input is asserted that the EPROM will return data. This spec is typically, though not always, the same as t_{AA}.

t_{OE} defines the maximum time from $\overline{OE}$ assertion to valid data output much like t_{CE}. However, because $\overline{OE}$ only enables the output drivers as opposed to powering up the device, t_{OE} is typically much less than t_{CE}.

Having delivered the data, the remaining two specs deal with what happens at the end of the cycle, i.e., following the deassertion of either $\overline{CE}$ or $\overline{OE}$ (remember, both are required to enable the output). t_{OH} indicates the minimum time the data is guaranteed to remain valid after $\overline{CE}/\overline{OE}$ deassertion. The other side of the coin, t_{DF}, defines the maximum time after which the output is guaranteed to completely float.

Table 3 shows the specifications for 70ns EPROMs from two different suppliers. Comparing the difference in specs highlights the point that not all 'xx' nanosecond EPROMs are created equal. Notice the significant difference in output enable (t_{OE}) and float (t_{DF}) times as well as $t_{\overline{CE}}/t_{\overline{OE}}$ ratios.

Further, notable differences can be observed even within a single suppliers product line. Table 4 shows a spectrum of EPROMs comprised of high-speed selections from one supplier and low-speed selections from another.

For the high-speed supplier, notice how t_{AA} and t_{CE} are equal (as expected) down to 35ns but then start to diverge. This points out the need for careful spec review since many designers might presume a t_{CE} of 35ns implies a 35ns, not 30ns, 'access time'. Also, except for t_{AA} itself, none of the timings change in a 'linear' way across speed ranges. For instance t_{CE} gets 'stuck' at 35ns for both the –35 and –30 parts while both t_{OE} and t_{DF} plateau at 18ns for the –45, –35 and –30 selections. A byproduct is that key ratios (ex: t_{AA}/t_{OE}, t_{AA}/t_{CE}, t_{OE}/t_{CE}, etc.) vary for practically every part. For example, the $t_{AA}/t_{\overline{OE}}$ ratio for the –70 part is almost 3:1 (70ns/25ns) while it's less than 2:1 (30ns/18ns) for the –30.

On the other hand, the low-speed supplier specs extrapolate from speed-grade to speed-grade in a more consistent and intuitive way. For instance t_{AA} always equals t_{CE} and the variance of timing ratios between speed-grades is reduced.

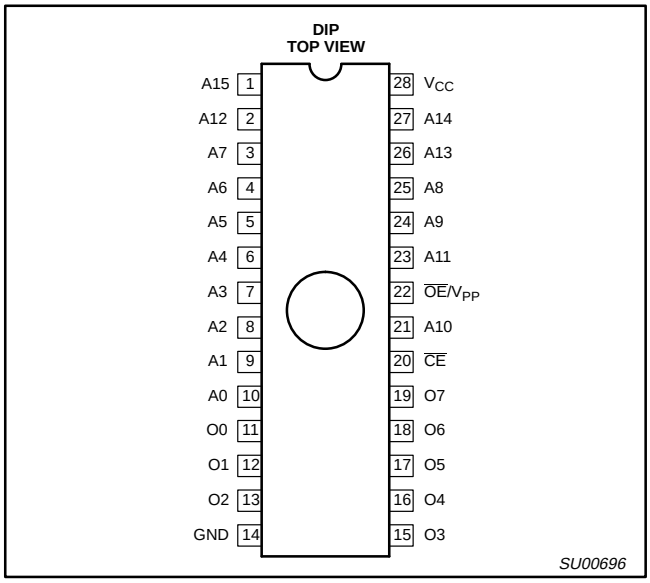

Figure 9. 512KB EPROM Pinout

Table 2. 512KB EPROM Truth Table

MODE	PIN FUNCTION				
	$\overline{CE}$	$\overline{OE}/V_{PP}$	A0	A9	DATA
Read	V_{IL}	V_{IL}	A0	A9	O7 – O0
Output Disable	X	V_{IH}	A0	A9	Hi-Z
Stand-by	V_{IH}	X	X	X	Hi-Z

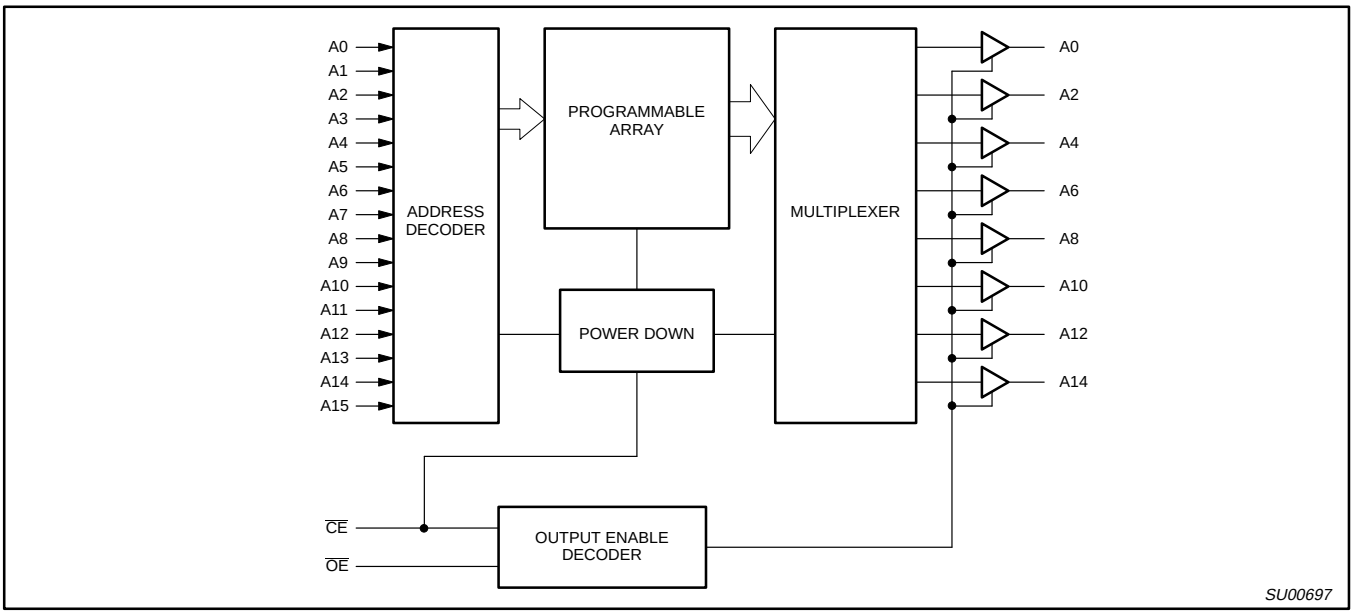

Figure 10. 512KB EPROM Block Diagram

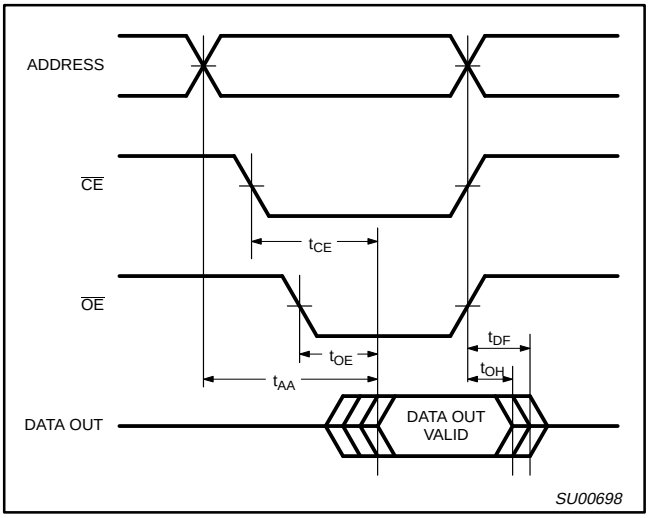

Figure 11. 512KB EPROM Timing Diagram

Table 3. Comparing Two '70ns' 27512 EPROMs

PARAMETER	SUPPLIER A	SUPPLIER B
t_{AA} (max)	70	70
t_{CE} (max)	70	70
t_{OE} (max)	25	40
t_{DF} (max)	25	30
t_{OH} (min)	0	5

Table 4. Spectrum of 27512 EPROM Specs

PARAMETER	SUPPLIER A						SUPPLIER C				
	–25	–30	–35	–45	–55	–70	–90	–120	–150	–170	–200
t_{AA} (max)	25	30	35	45	55	70	90	120	150	170	200
t_{CE} (max)	30	35	35	45	55	70	90	120	150	170	200
t_{OE} (max)	12	18	18	18	20	25	40	60	65	70	75
t_{DF} (max)	12	18	18	18	20	25	25	30	45	50	55
t_{OH} (min)	0	0	0	0	0	0	0	0	0	0	0

SRAMs

Compared to EPROMs, SRAMs are less subject to competition from ever higher integrated CPUs. Across the entire Philips '51 family the largest on-chip RAM offered is 2KB, with the majority of CPUs featuring only 64–512 bytes. Thus an application that requires 2KB–64KB or more of data definitely calls for external SRAM. SRAM is also useful as instruction memory in those applications (such as a PC-plug in) that exploit 'downloadable' code to vary their functionality at runtime.

Figure 12 shows the pinout, Figure 13 shows the block diagram, and Table 5 shows the truth table for a 256k bit (32K bytes) SRAM organized as 32Kx8. Reflecting JEDEC standardization, the pinout and functionality is nearly the same as EPROMs, the major difference being addition of a $\overline{WE}$ (Write Enable) line. Once again, $\overline{CE}$ is responsible for controlling power consumption and $\overline{OE}$ simply enables the output reflected in the t_{CE}/t_{OE} differential. For all practical purposes, an SRAM is little distinguishable from an EPROM as far as reads are concerned.

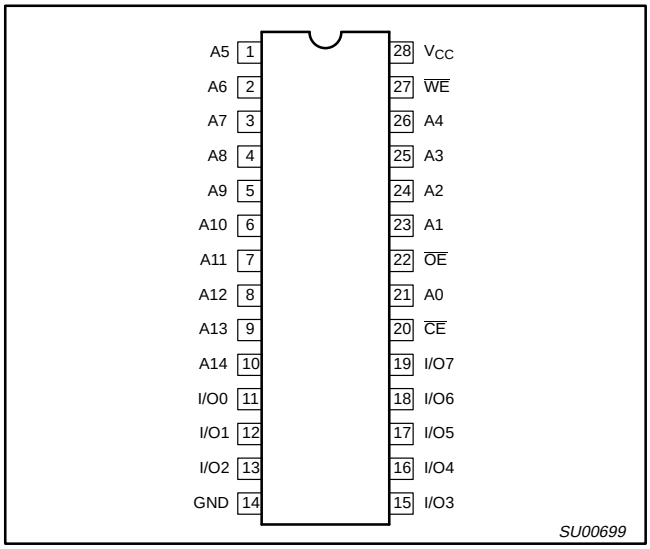

Figure 12. 256KB SRAM Pinout

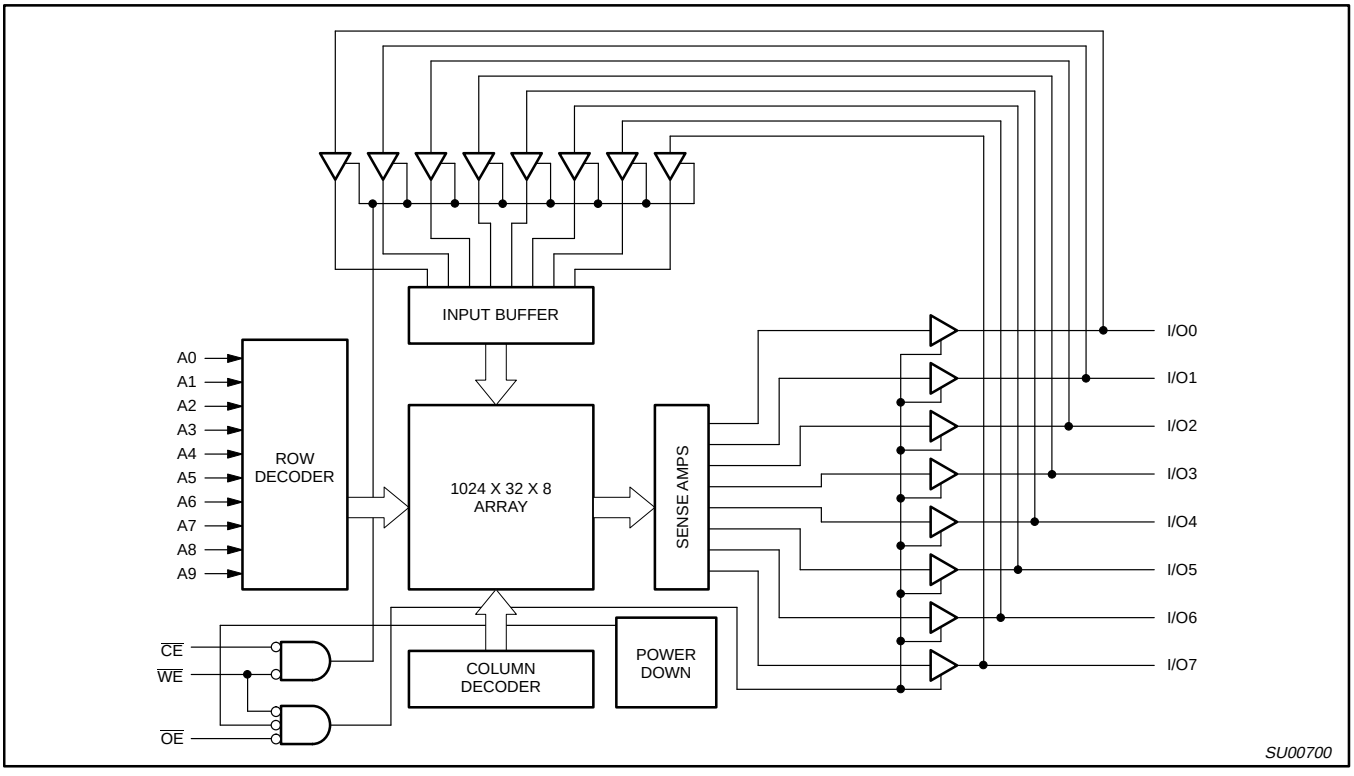

Figure 13. 256KB SRAM Block Diagram

Table 5. 256KB SRAM Truth Table

$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	MODE	V_{CC} CURRENT	I/O PIN
H	X	X	Not Selected	I_{SB}, I_{SB1}	Hi-Z
L	L	H	Read	I_{CC}	D_{OUT}
L	H	L	Write	I_{CC}	D_{IN}
L	L	L	Write	I_{CC}	D_{IN}

Figure 14 shows the read cycle timing for an 85ns SRAM which is quite similar to that of an EPROM. Depending on the supplier, the naming of certain specs may be slightly different, but for ease of comparison this application note translates to a common nomenclature. This SRAM (as do some EPROMS), defines separate $\overline{CE}$ and $\overline{OE}$ data float specs (t_{CHZ}, t_{OHZ}) instead of a single t_{DF}. Since the SRAM (unlike the EPROM) can be written, it also specifies the other side of the data float coin (i.e., enable to output driven) with t_{CLZ} and t_{OLZ}.

Now let's examine how $\overline{WE}$ factors into the SRAM operation. When $\overline{WE}$ is asserted, the SRAMs output buffers are disabled while the input buffers are enabled to receive data.

Figure 15 shows the write cycle timing which is slightly more complicated than reads. Write time (t_{WP}) is defined as the time during which both $\overline{CE}$ and $\overline{WE}$ are asserted. Thus, various setup and hold timings that are specified relative to the 'end' of the write cycle (such as t_{DW}, t_{DH}, t_{WR}, etc.) should refer to whichever signal ($\overline{CE}$ or $\overline{WE}$) terminates first.

t_{WC} simply defines the write cycle time which, along with t_{RC}, is the same as the 'access time', i.e., 85ns. This is in contrast to other types of memory (notably DRAMs) in which the cycle time may be longer than the access time due to 'precharge' delay.

t_{CW} and t_{AW} specify the minimum time from valid $\overline{CE}$ and address inputs to the end of the write cycle ($\overline{CE}$ or $\overline{WE}$ high, whichever comes first). They are the write cycle corollary to the read cycles t_{CE} and t_{AA} specs and in this (the usual, but not always as we saw for EPROMs) case are the same. The SRAM also defines an address setup to the beginning of the write cycle (t_{AS}).

t_{WP} simply specifies the minimum write pulse (the overlap of $\overline{CE}$ and $\overline{WE}$) width. t_{WR} specifies a minimum write 'recovery' time, essentially an address hold time after the end of write. This SRAM specs t_{WR} at 10ns minimum, but many SRAMs need no address hold (i.e., t_{WR}=0ns minimum) should the spec prove troublesome. Remember, the 'end' of the write cycle is defined as the earlier of $\overline{CE}$ or $\overline{WE}$ deassertion.

t_{DW} and t_{DH} specify the input data setup and hold times relative to the end of write. Finally, t_{OHZ} reappears as a reminder that a write cycle shouldn't drive the bus until previous read data disappears to avoid bus contention.

There's no need to repeat the previous EPROM spec comparison exercise to emphasize that various suppliers of SRAMs may exhibit subtle, but possibly critical, timing differences.

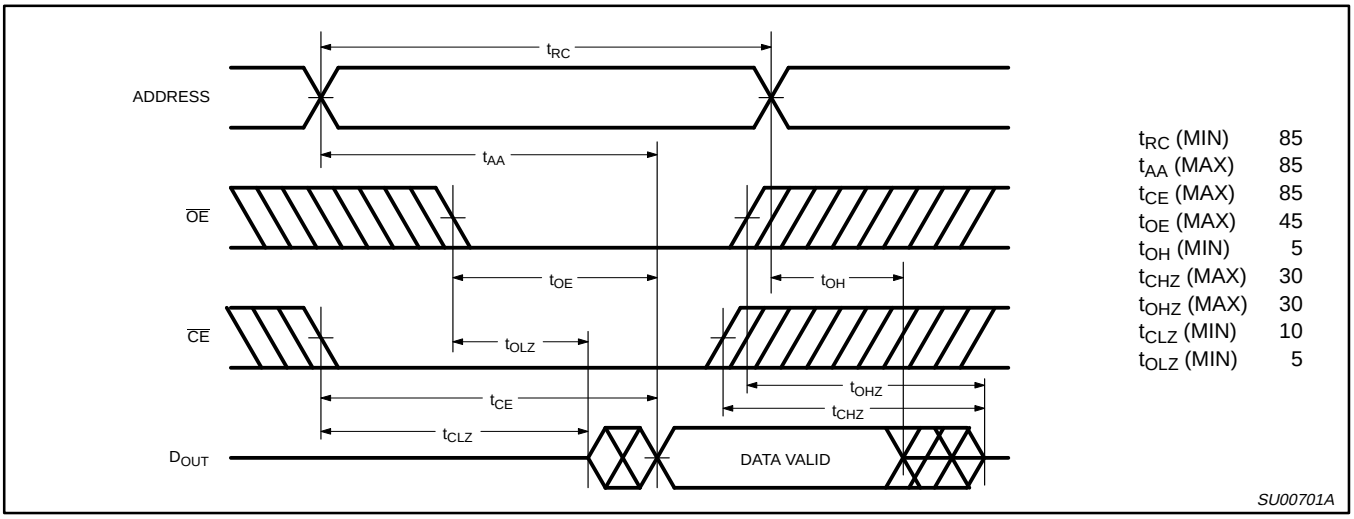

Figure 14. 256KB SRAM Read Cycle Timing Diagram

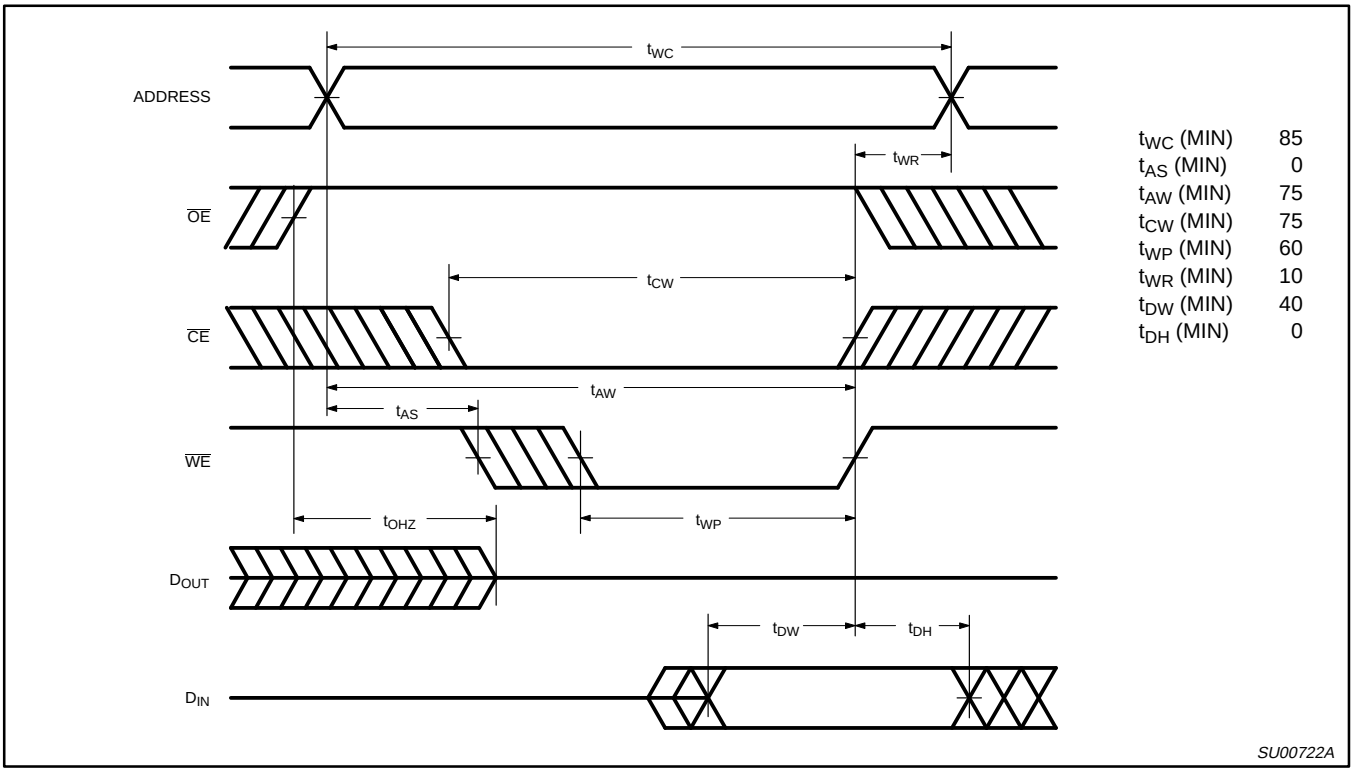

Figure 15. 256KB SRAM Write Cycle Timing Diagram

80C31/8XC51 – 33MHz INTERFACE EXAMPLE

One major contributor to the longevity of the '51 family is that Philips has continued to upgrade the maximum clock rate. In the case of the 80C31/8XC51, speed selections up to 33MHz are available. Boosting performance to nearly 3 MIPS (≈360ns instruction cycle) offers an easy upgrade path for existing designs as well as enabling new ones.

Of course, higher performance is only possible with faster memories. Fortunately, thanks largely to the 'need for speed' on desktop PCs, memory suppliers have responded to the call.

Let's perform a detailed analysis of an example 33MHz design, pointing out the useful tips, and possible traps, along the way. Note that, like memories, individual members of the '51 family may exhibit slightly different specs. However, the principles and techniques explained in this application note are generally applicable for any CPU at any clock rate.

Table 6, reproduced from the data sheet for convenience, details the timing specification for each of the three external bus cycle types – instruction read ($\overline{PSEN}$), data read ($\overline{RD}$) and data write ($\overline{WR}$). Rather than explain every spec up front, let's simply propose and evaluate a design to focus on the relevant, and possibly critical, timing parameters.

First of all, since the CPU is available in many speed grades, notice that most specs are stated in terms of t_{CLCL}, i.e., the clock cycle. For a 33MHz part, t_{CLCL} is 30.3ns. To ease calculations, this can be rounded to 30ns with the caveat final timing should be confirmed, especially if a marginal situation arises. Observing that the maximum t_{CLCL} multiplication factor shown on the datasheet is 9 (t_{AVDV}), watch out for margins of ≈3ns (i.e., 9×0.3ns = 2.7ns) or less when approximating.

Table 6. AC Electrical Characteristics for 12–33MHz Philips North America Devices

T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±20%, V_{SS} = 0V (80C31/51)[1, 2, 4] (12, 16, and 24MHz versions)
T_{amb} = 0°C to +70°C or –40°C to +85°C, V_{CC} = 5V ±10%, V_{SS} = 0V (87C51 12, 16 AND 24MHz versions) (80C31/80C51 33MHz version);
For 87C51 (33MHz only) T_{amb} = = 0°C to +70°C, V_{CC} = 5V ±5%

SYMBOL	FIGURE	PARAMETER	VARIABLE CLOCK[3]		UNIT
			MIN	MAX	
$1/t_{CLCL}$		Oscillator frequency: **Speed Versions** SC80C31/51 C G P Y	 3.5 3.5 3.5 3.5	 12 16 24 33	 MHz MHz MHz MHz
t_{LHLL}	16	ALE pulse width	$2t_{CLCL}$–40		ns
t_{AVLL}	16	Address valid to ALE low	t_{CLCL}–13		ns
t_{LLAX}	16	Address hold after ALE low	t_{CLCL}–20		ns
t_{LLIV}	16	ALE low to valid instruction in		$4t_{CLCL}$–65	ns
t_{LLPL}	16	ALE low to $\overline{PSEN}$ low	t_{CLCL}–13		ns
t_{PLPH}	16	$\overline{PSEN}$ pulse width	$3t_{CLCL}$–20		ns
t_{PLIV}	16	$\overline{PSEN}$ low to valid instruction in		$3t_{CLCL}$–45	ns
t_{PXIX}	16	Input instruction hold after $\overline{PSEN}$	0		ns
t_{PXIZ}	16	Input instruction float after $\overline{PSEN}$		t_{CLCL}–10	ns
t_{AVIV}	16	Address to valid instruction in		$5t_{CLCL}$–55	ns
t_{PLAZ}	16	$\overline{PSEN}$ low to address float		10	ns
Data Memory					
t_{RLRH}	17,18	$\overline{RD}$ pulse width	$6t_{CLCL}$–100		ns
t_{WLWH}	17,18	$\overline{WR}$ pulse width	$6t_{CLCL}$–100		ns
t_{RLDV}	17,18	$\overline{RD}$ low to valid data in		$5t_{CLCL}$–90	ns
t_{RHDX}	17,18	Data hold after $\overline{RD}$	0		ns
t_{RHDZ}	17,18	Data float after $\overline{RD}$		$2t_{CLCL}$–28	ns
t_{LLDV}	17,18	ALE low to valid data in		$8t_{CLCL}$–150	ns
t_{AVDV}	17,18	Address to valid data in		$9t_{CLCL}$–165	ns
t_{LLWL}	17,18	ALE low to $\overline{RD}$ or $\overline{WR}$ low	$3t_{CLCL}$–50	$3t_{CLCL}$+50	ns
t_{AVWL}	17,18	Address valid to $\overline{WR}$ low or $\overline{RD}$ low	$4t_{CLCL}$–75		ns
t_{QVWX}	17,18	Data valid to $\overline{WR}$ transition	t_{CLCL}–20		ns
t_{WHQX}	17,18	Data hold after $\overline{WR}$	t_{CLCL}–20		ns
t_{RLAZ}	17,18	$\overline{RD}$ low to address float		0	ns
t_{WHLH}	17,18	$\overline{RD}$ or $\overline{WR}$ high to ALE high	t_{CLCL}–20	t_{CLCL}+25	ns
External Clock					
t_{CHCX}	19	High time	12		ns
t_{CLCX}	19	Low time	12		ns
t_{CLCH}	19	Rise time		20	ns
t_{CHCL}	19	Fall time		20	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. Load capacitance for port 0, ALE, and $\overline{PSEN}$ = 100pF, load capacitance for all other outputs = 80pF.
3. For all Philips North America speed versions only.
4. Interfacing the 80C31/51 to devices with float times up to 50ns is permitted. This limited bus contention will not cause damage to port 0 drivers.

EXPLANATION OF THE AC SYMBOLS

Each timing symbol has five characters. The first character is always 't' (= time). The other characters, depending on their positions, indicate the name of a signal or the logical status of that signal. The designations are:

A – Address
C – Clock
D – Input data
H – Logic level high
I – Instruction (program memory contents)
L – Logic level low, or ALE
P – $\overline{\text{PSEN}}$
Q – Output data
R – $\overline{\text{RD}}$ signal
t – Time
V – Valid
W – $\overline{\text{WR}}$ signal
X – No longer a valid logic level
Z – Float

Examples: t_{AVLL} = Time for address valid to ALE low.
t_{LLPL} = Time for ALE low to $\overline{\text{PSEN}}$ low.

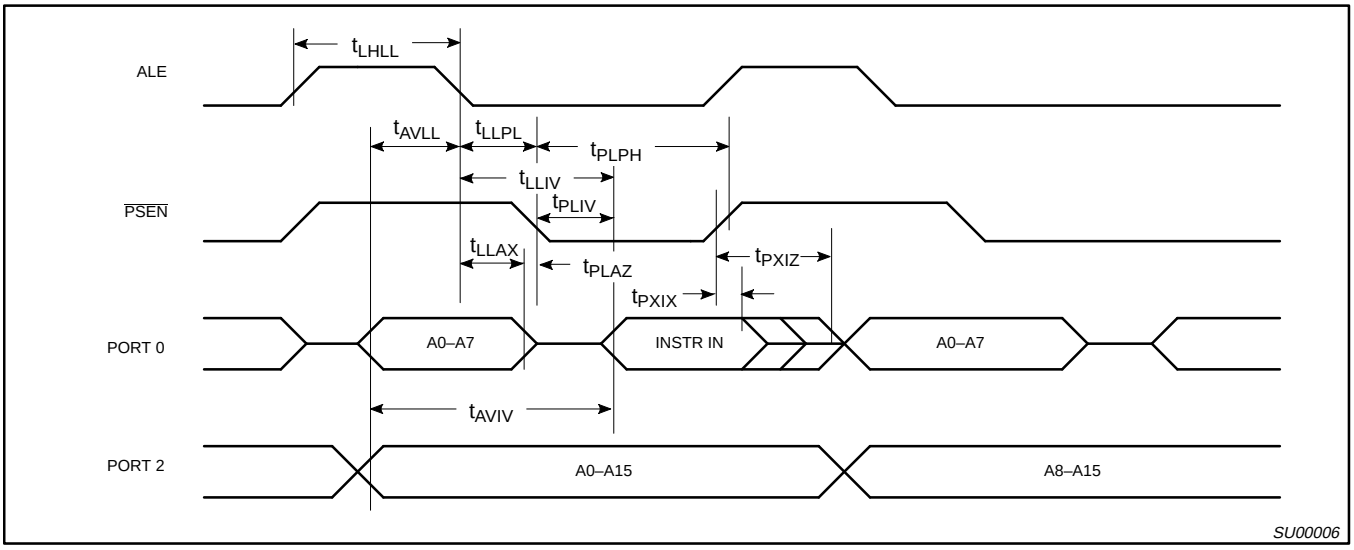

Figure 16. External Program Memory Read Cycle

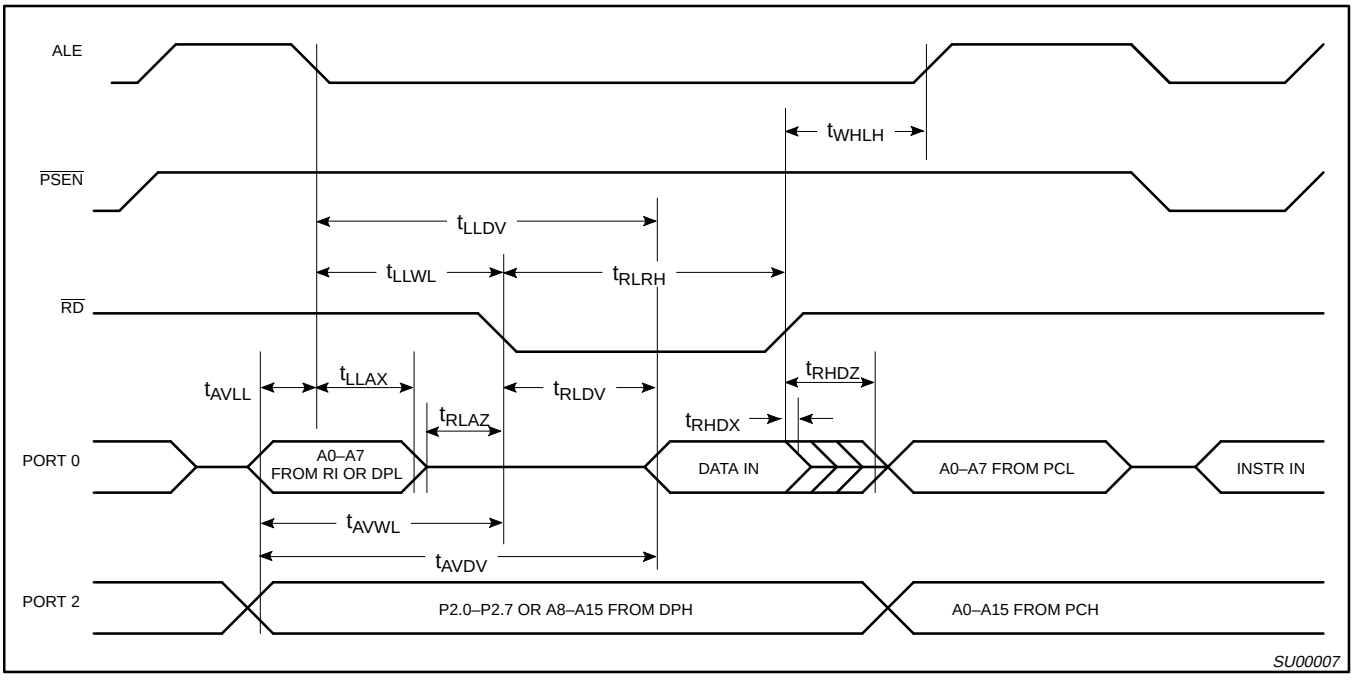

Figure 17. External Data Memory Read Cycle

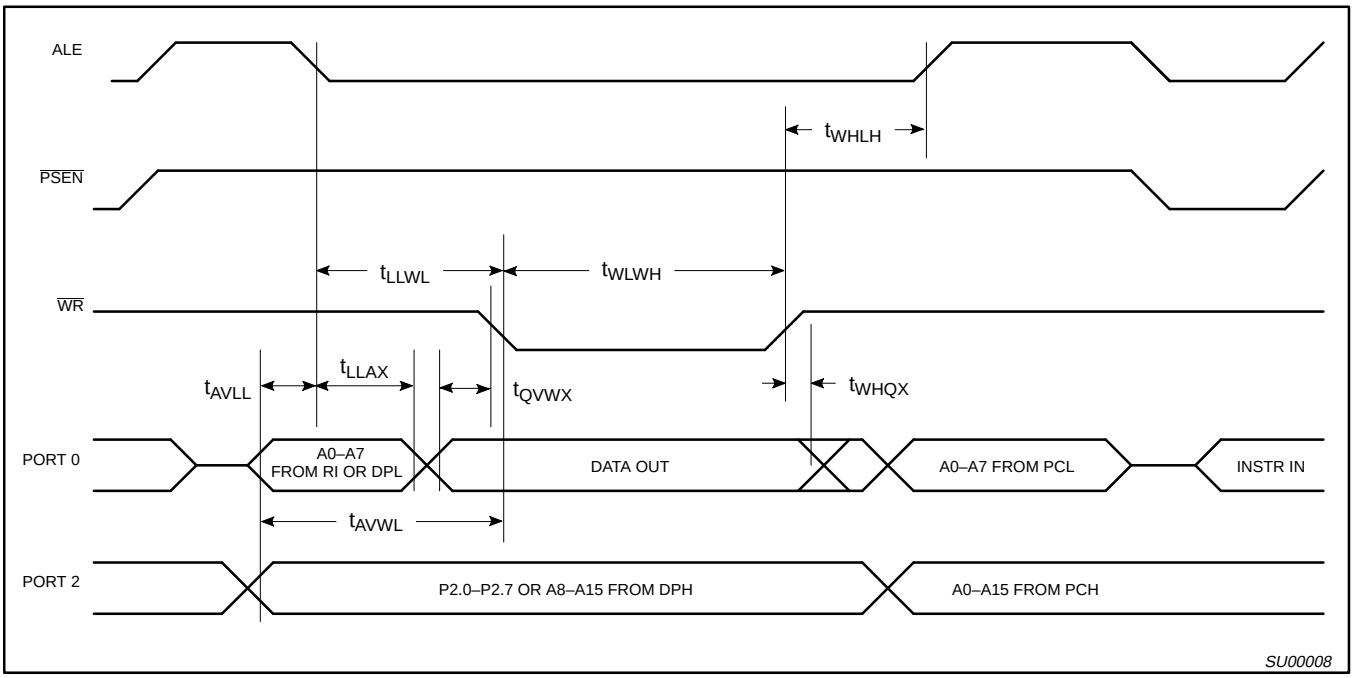

Figure 18. External Data Memory Write Cycle

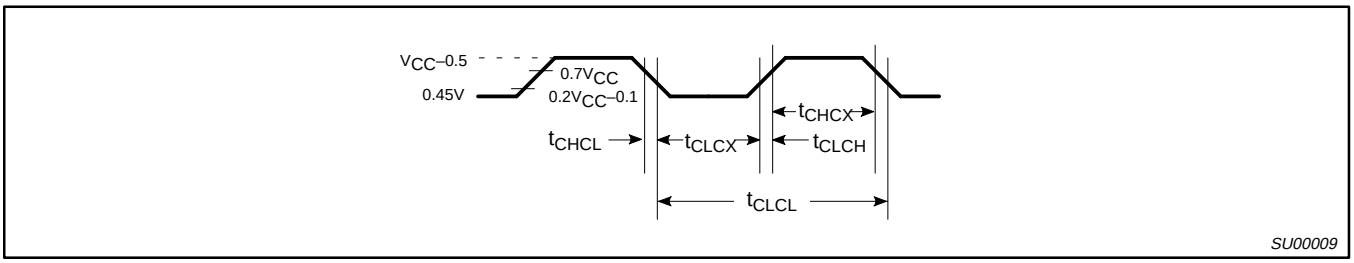

Figure 19. External Clock Drive

EPROM INTERFACE

Let's start with the 'classic' circuit shown in Figure 20 that uses a transparent latch to demultiplex the address/data bus AD0–7 (P0). The latch, (a '573 which combines the function of the well-known '373 with an easy to layout 'broadside' pinout) shown in Figure 21, is controlled with two pins – E (Enable) and $\overline{OE}$ (Output Enable). It's called transparent because as soon as E is asserted, the inputs will begin to flow to the outputs and are subsequently latched on the trailing edge of E. This is exactly the behavior called for to demultiplex the CPU AD0–7 bus (P0) with the ALE output from the CPU. Usually, $\overline{OE}$ is simply connected to ground enabling the output at all times.

Now is a good time to say a few words about 'glue' logic like the '573. Based on the previous memory spec comparison, it should be no surprise that all glue logic, including latches, buffers, decoders, etc. not to mention PLDs, is not created equal.

For instance, as shown in Table 7, there are a myriad of 'TTL' variants – HC, HCT, LS, ALS, AS, FAST, etc. – covering a rather broad (≈5:1) range of speed and power. The important point to note is that these absolutely small differences and delays were easier to ignore in yesterdays slow speed designs, but can become critical as the CPU clock rate increases.

The first step is to confirm the CPU meets the setup and hold times for the chosen latch. Referring back to the CPU timing (Table 6, Figures 16, 17, 18, and 19)...

$t_S < t_{AVLL}$ (address valid to ALE low) = t_{CLCL}–13 = 17ns

$t_H < t_{LLAX}$ (address hold after ALE low) = t_{CLCL}–20 = 10ns

In this case, the choice of TTL technology proves non-critical. For reference, t_S & t_H cover a spectrum from 15 & 5ns for an HCT '574 to 3 & 0ns for a FAST '574.

The next decision is how to connect the EPROM $\overline{CE}$ and $\overline{OE}$ control lines. A simple 'no glue' solution is to simply ground $\overline{CE}$. This has the benefit of eliminating the t_{CE}/t_{OE} differential as a component of access time, though at the expense of higher power consumption. With $\overline{CE}$ grounded, access time depends only on $\overline{OE}$ which is connected to the CPU $\overline{PSEN}$ line.

Note that this scheme (grounding $\overline{CE}$) limits code expansion to a single EPROM, otherwise decoding is required. However, this is not an unreasonable restriction given the typical code size of '51 applications relative to the high density of modern EPROMs.

Finally, it is important to remember that applications exploiting the low power modes of the CPU (IDLE and POWERDOWN) must accommodate the behavior of ALE and $\overline{PSEN}$. Specifically, during IDLE, ALE and $\overline{PSEN}$ = 0 and during POWERDOWN, ALE and $\overline{PSEN}$ = 1. In general, this requires externally gating $\overline{CE}$, $\overline{ALE}$, $\overline{OE}$, etc. Since the optimal solution is very dependent on the particulars of the configuration and application, the issue is not addressed further in this application note.

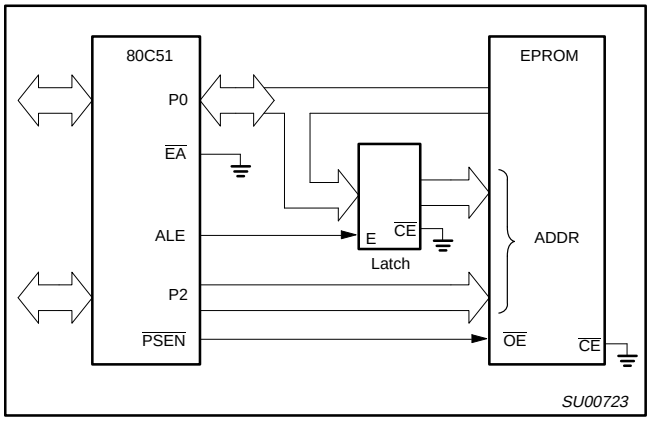

Figure 20. CPU + Latch + ERPOM Diagram

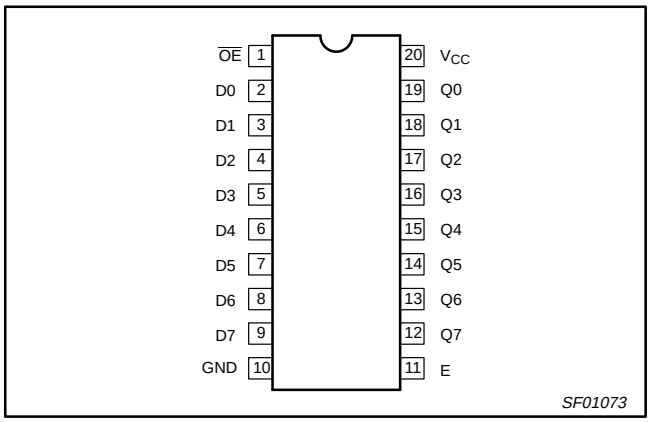

Figure 21. 74F573 Pinout

Table 7. Spectrum of TTL Performance

V_{CC} = 5V; T_{amb} = 25°C; C_L = 15pF

Parameters	Technology	HCMOS	Metal Gate CMOS		Standard TTL	Low-Power Schottky TTL	Schottky TTL	Advanced Low-Power Schottky TTL	Advanced Schottky TTL	Fairchild Advanced Schottky TTL
	Family	74HC	4000		74	74LS	74S	74ALS	74AS	74F
			CD	HE						
Power dissipation, typ. (mW)										
Gate	static	0.0000025	0.001		10	2	19	1.2	8.5	5.5
	dynamic @100kHz	0.075	0.1		10	2	19	1.2	8.5	5.5
Counter	static	0.000005	0.001		300	100	500	60	–	190
	dynamic @100kHz	0.125	0.120		300	100	500	60	–	190
Propagation delay (ns)										
Gate	typical	8	94	40	10	9.5	3	4	1.5	3
	maximum	14	190	80	20	15	5	7	2.5	4
Delay/power product (pJ)										
Gate	@100kHz	0.52	9	4	100	19	57	4.8	13	16.5
Maximum clock frequency (MHz)										
D-type flip-flop	typical	55	4	12	25	33	100	60	160	125
	minimum	30	2	6	15	25	75	40	–	100
Counter	typical	45	2	6	32	32	70	45	–	125
	minimum	25	1	3	25	25	40	–	–	100
Output Drive (mA)										
	standard outputs	4	0.51	0.8	16	8	20	8	20	20
	bus outputs	6	1.6		48	24	64	24	48	64
Fan-out (LS-loads)										
	standard outputs	10	1	2	40	20	50	20	50	50
	bus outputs	15	4		120	60	160	60	120	160

EPROM TIMING ANALYSIS

With a design established, it is now possible to perform an initial timing evaluation. The procedure is simply to step through each EPROM spec one by one to identify a speed grade that meets all the relevant CPU timing requirements.

Starting with t_{AA}, it is apparent that address access time for the EPROM must be less than the CPU t_{AVIV} (address to valid instruction in). However, don't forget that the '573 propagation delay must also be factored in. The conclusion is expressed in the form of an equation as...

$$t_{AA}\ (\text{EPROM}) < t_{AVIV}\ (\text{CPU}) - t_{PROP}\ (\text{TTL})$$

t_{PROP}, depending on the choice of TTL technology, varies from approximately 10ns (FAST) to 40ns (HCT). Throughout this application note, t_{PROP} for the latch is assumed to be 10ns.

Substituting the CPU data sheet value for t_{AVIV} ($5tC_{LCL}$–55) and solving yields a required t_{AA} of...

$$t_{AA} < ((5tC_{LCL})–55)–t_{PROP} = ((5\times30)–55)–10 = 85\text{ns}$$

According to the EPROM spec chart (refer back to Table 4), this calls for a '–70' (70ns) EPROM.

Since $\overline{CE}$ is grounded, it meets the t_{CE} spec of any EPROM and need not be considered in this design.

The only 'access-time' related spec remaining to check is $t_{\overline{OE}}$. Connected directly to the CPU $\overline{PSEN}$ with no intervening TTL, the equation is simply...

$$t_{OE} < t_{PLIV} = 3t_{CLCL}–45 = (3\times30)–45 = 45\text{ns}$$

...a spec easily met by the –70 EPROM (t_{OE} max. = 25ns).

Able to access the EPROM successfully, all that's left is to verify the EPROM data hold (t_{OH}) and float (t_{DF}) specs.

The EPROM t_{OH} spec is 0ns (whatever the speed), i.e., the EPROM output is not held after deselection. On the CPU side, since $\overline{PSEN}$ is controlling the EPROM selection (via $\overline{OE}$), the corresponding spec is t_{PXIX} (input instruction hold after $\overline{PSEN}$) which is also 0ns. This may sound tight but actually isn't a problem. First, there's the simple fact that the CPU will 'see' $\overline{PSEN}$ go high before the EPROM by virtue of the package and PCB wiring delays. Second, though it's convenient for the EPROM manufacturer to spec 0ns, it's clear that the outputs can't shut off in zero time.

Finally, t_{DF} can be checked against the CPU t_{PXIZ} (input instruction float after $\overline{PSEN}$) spec.

$$t_{DF} < t_{PXIZ} = t_{CLCL}–10 = (30–10) = 20\text{ns}$$

Oops, unfortunately a –70 EPROM, with t_{DF} max. 25ns, can't meet this spec. This is a bus contention situation in which the next cycle address driven by the CPU will collide with the remnants of the previous cycle EPROM output. Though access time is certainly important, the lesson is that 'unaccess' (i.e., float) time can't be overlooked.

So, what are the options to resolve the problem?

The straightforward solution is to select an EPROM with better t_{DF} spec by moving to a faster speed grade (for instance, the –55 which cuts t_{DF} to an acceptable 20ns). Alternatively, another vendor may offer a –70 with better t_{DF} spec (note the difference between supplier A & B –90 t_{DF} spec as an example).

Another option is to insert a buffer or transceiver with a fast shutoff in the datapath as shown in Figure 22. Take care to choose a TTL technology is actually faster. For instance, the output disable time for a FAST '244 is a speedy 6ns, but that for an HCT '244 a leisurely 31ns which doesn't help at all.

Also, make sure solving the problem at the 'back-end' of the cycle doesn't just push it to the 'front-end'. Remember that propagation delay through the buffer or transceiver must now be subtracted from available access time (t_{AA}, t_{OE}). Re-evaluating the access times shows that t_{PROP} must be less than 15ns, or a faster EPROM has to be selected anyway. For reference, t_{PROP} for a FAST '244 is only 7ns but an unhelpful 28ns for HCT.

A final, and rather widely used, option is to simply ignore the problem. Referring back to Table 6, note 4 at the bottom of the page points out that the Philips CPU P0 drivers are designed to tolerate contention (i.e., t_{DF} up to 50ns). Determining whether the memory is similarly robust requires confirmation by the particular EPROM supplier.

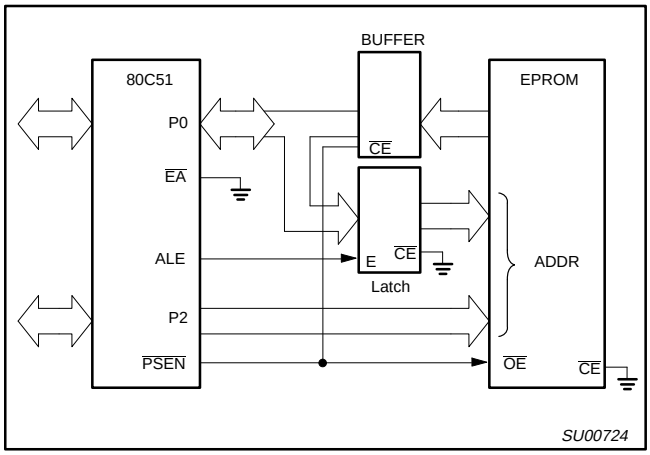

Figure 22. CPU + Latch + EPROM +Xcvr Diagram

SRAM INTERFACE

Though other configurations are possible, this example illustrates the most common one in which the SRAM is used for data-only (as opposed to instruction or instruction & data) storage. Should the SRAM be used for instruction storage, the previous CPU instruction access timings used in evaluating the EPROM interface apply.

Instead of grounding the SRAM $\overline{CE}$ line, since it requires only 15 address lines (i.e., 32Kx8), it is connected to the CPU A15 (P2.7). Thus, the SRAM occupies the lower 32K bytes of the data space, leaving the upper 32KB for I/O expansion. The SRAM $\overline{OE}$ and $\overline{WE}$ lines are connected to the CPU $\overline{RD}$ and $\overline{WR}$ lines respectively as shown in Figure 23.

This configuration can take advantage of the previously mentioned operating characteristics of P2 (A8–A15), specifically the fact that P2 (and thus A15 which is connected to the SRAM $\overline{CE}$ line) reverts to prior levels programmed into the SFR when not performing an external data access. By programming P2.7 (A15) to output a 1, the SRAM will be deselected when it is not being accessed, saving power. Further, unlike the EPROM case, no extra control line gating logic is needed to accommodate the CPU low power (IDLE and POWERDOWN) modes.

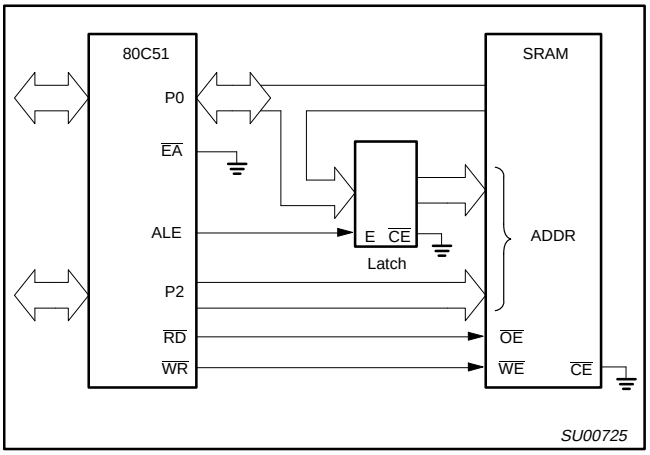

Figure 23. CPU + Latch + SRAM Diagram

SRAM Timing Analysis

The process of evaluating the 32Kx8 SRAM interface is similar to that for the EPROM, recognizing that the CPU external data access ($\overline{RD}$, $\overline{WR}$) timing is different than instruction access ($\overline{PSEN}$) and both read and write cycles must be checked.

Starting with the read cycle, t_{RC} is clearly not an issue since the CPU data access bus cycle of 360ns is well beyond the 85ns SRAM spec.

For a data read, the SRAM t_{AA} is compared with the CPU t_{AVDV} (address to valid data in). Once again, as for the EPROM, propagation delay through the address latch is considered (10ns assumed)...

$t_{AA} < t_{AVDV} – t_{PROP} = ((9t_{CLCL})–165)–t_{PROP} = ((9\times30)–165)–10 = 95ns$

...which meets the –85 SRAM t_{AA} spec of 85ns.

Since $\overline{CE}$ is no longer grounded, it must be evaluated as well. Connected to A15 without intervening TTL, the equation is simply the same as that for t_{AA} without t_{PROP} derating...

$t_{CE} < t_{AVDV} = (9t_{CLCL})–165 = (9\times30)–165 = 105ns$

...meeting the SRAM t_{CE} spec of 85ns with even more margin.

The SRAM $\overline{OE}$ is connected to the CPU $\overline{RD}$ line so t_{OE} is evaluated as...

$t_{OE} < t_{RLDV} = (5t_{CLCL})–90 = (5\times30)–90 = 60ns$

...meeting the SRAM t_{OE} spec of 45ns.

Unlike the EPROM, the SRAM guarantees some output data hold time (t_{OH} = 5ns min) while the CPU still only requires 0ns (t_{RHDX}).

Though the EPROM had a problem with float time (t_{DF}) note that for data cycles...

$t_{DF} < t_{RHDX} = (2t_{CLCL})–28 = (2\times30)–28 = 32ns$

...showing that the SRAM t_{DF} spec (30ns), though worse than the EPROM spec (25ns for –70), is actually met. This highlights the more relaxed timing of CPU data access compared to instruction access.

On the other side of the data float equation, is it possible for the SRAM to start to return data before the address has been removed from the CPU AD0–7 bus thus causing bus contention? The SRAM will start to drive the bus within 5ns (t_{OLZ}) of $\overline{OE}$ assertion. Fortunately, the CPU t_{RLAZ} spec of 0ns guarantees that the address is off the bus at the time $\overline{RD}$ (connected to the SRAM $\overline{OE}$) is asserted.

This completes the SRAM read cycle evaluation and shows there is plenty of margin due to the relaxed nature of data access timing. Now, let's move on to the write cycle evaluation.

Once again, the SRAM t_{WC} (as was t_{RC}) spec of 85ns is easily met since the CPU data access bus cycle is 360ns (12×30ns).

The SRAM t_{AW} (address valid to end of write) is compared against the sum of the CPU t_{AVWL} (address valid to write low) and t_{WLWH} (write low to write high) specs again considering the latch propagation delay...

$t_{AW} < t_{AVWL} + t_{WLWH} – t_{PROP}$

or...

$t_{AW} < ((4t_{CLCL})–75)+ ((6t_{CLCL})–100) – t_{PROP}$

thus...

$t_{AW} < ((4\times30)–75)+ ((6\times30)–100) – 10 = 115ns$

...which easily meets the SRAM t_{AW} spec of 75ns.

Meanwhile, the SRAM t_{AS} spec defines the time addresses must be setup prior to the assertion of $\overline{WE}$ which is connected to the CPU $\overline{WR}$ line so...

$t_{AS} < t_{AVWL} – t_{PROP} = ((4t_{CLCL})–75) – t_{PROP} = ((4\times30)–75)–10 = 35ns$

...again easily met by the t_{AS} spec of 0ns.

Since the SRAM $\overline{CE}$ pin is simply connected to the CPU A15 without intervening TTL, calculation of t_{CW} ($\overline{CE}$ to write end) is the same as for t_{AW} without derating for t_{PROP}...

$t_{CW} < t_{AVWL} + t_{WLWH} = 125ns$

...meeting the t_{CW} spec (75ns, same as t_{AW}) with an additional t_{PROP} (10ns) margin.

Remembering that the SRAM write pulse is defined as the overlap of $\overline{CE}$ and $\overline{WE}$, t_{WP} (write pulse width) is simply defined by t_{WLWH}...

$t_{WP} < t_{WLWH} = (6t_{CLCL})–100 = (6\times30)–100 = 80ns$

...which meets the SRAM t_{WP} spec of 60ns.

The SRAM t_{WR} (write recovery) spec defines how long the addresses must be held after the end of write, which is the end of CPU $\overline{WR}$ in this design. Since addresses are guaranteed to remain stable while ALE is low, this becomes t_{WHLH} ($\overline{RD}$ or $\overline{WR}$ high to ALE high)...

$t_{WR} < t_{CLCL}–20 = 30–20 = 10ns$

...which just meets the SRAM spec of 10ns.

As for t_{DS} (data setup to end of write), the corresponding CPU timing is derived by summing t_{QVWX} (data valid to $\overline{WR}$ transition) and t_{WLWH} ($\overline{WR}$ pulse width) so...

$t_{DS} < t_{QVWX} + t_{WLWH} = (t_{CLCL}–20) + ((6t_{CLCL})–100)$

thus...

$t_{DS} < (30–20) + ((6\times30)–100) = 90ns$

...easily meeting the SRAM spec of 45ns.

The SRAM hold spec t_{DH} is simply compared with t_{WHQX} (data hold after write)...

$t_{DH} < t_{CLCL}–20 = 30–20 = 10ns$

...which meets the SRAM 0ns hold spec.

As for possible bus contention with previous SRAM read data, the SRAM spec indicates the data bus shouldn't be driven until t_{OHZ} after the end of a previous read cycle to the same SRAM. Whether this presents a problem depends on the system configuration.

In this example, the SRAM is being used for data access only (i.e., connected to $\overline{RD}$ and $\overline{WR}$). Noting that external data cycles are always separated by an (internal or external) instruction fetch, it is impossible for back-to-back accesses to occur with a data-only memory.

Though not a factor in this example, meeting the t_{OHZ} spec could be of concern in a system that 1) overlaps code and data into a single 64K space (i.e., by ANDing $\overline{PSEN}$ and $\overline{RD}$) and 2) fetches a data write instruction (ex: MOVX @DPTR,A) from the **same** SRAM to which the write is targeted. In this case, the tail end of the opcode (MOVX) fetch may collide with the beginning of the subsequent address (DPTR) output if the t_{OHZ} spec isn't met.

In this (and only this) rather rare situation, the SRAM t_{OHZ} spec must meet the tighter CPU instruction float ($t_{PXIZ} = t_{CLCL}–10$, i.e., 20ns @ 33MHz) rather than the more relaxed data float ($t_{RHDZ} = 2t_{CLCL}–28$, i.e., 32ns @ 33MHz) timing. As stated in the earlier EPROM analysis, this can be accomplished by selecting a faster SRAM chip, isolating the SRAM data bus with a fast shutoff transceiver or confirming the SRAM can operate reliably and correctly despite the bus contention.

SUMMARY AND CONCLUSIONS

This application note examined the detailed timing of a 33MHz 80C31/8XC51 system based on actual EPROM and SRAM specifications. Beyond the particulars of this example, the techniques shown are applicable to any design.

One important point illuminated was the degree to which CPU, memory and glue logic specs can vary and thus must be considered on a supplier-by-supplier, speed grade-by-speed grade basis.

For instance, ostensibly equivalent parts from different suppliers may vary in one or more parameters, an example being '–70' EPROMs from two different suppliers which differ in a number of important specs.

Furthermore, even parts from a single supplier may exhibit non-intuitive spec variations across derivatives, speed-grades or process technology. In particular, a part that is 'twice as fast' in terms of one spec does not imply that other specs are similarly improved.

These caveats apply to memories, TTL & glue logic and even CPUs themselves, whether from different suppliers or derivatives within a single suppliers catalog.

The development of very fast memories has largely removed 'access time' as a barrier for high-speed '51 family-based designs. However, despite the natural tendency to focus on 'access time', other specifications prove equally, if not more, critical.

Most notably 'float time', as it relates to bus contention, cannot be overlooked. Though in practice some degree of bus contention may be tolerable, no recommendation other than to meet the specs can be officially made. Philips explicitly guarantees acceptable behavior of their 80C31/8XC51 in this regard (i.e., note #4) but only the other chip suppliers can vouch for the integrity of their parts.