

마이크로 프로세서의 이해



교육목표

- 마이크로프로세서의 학습에 반드시 필요한 버스에 대하여 이해한다.
-



목 차

□ 프로그램의 실행과 버스동작과의 관계

버스 인터페이스의 사례

버스의 동작원리



프로그램의 실행과 버스동작과의 관계

`int A, B, C`

`C = A+B`

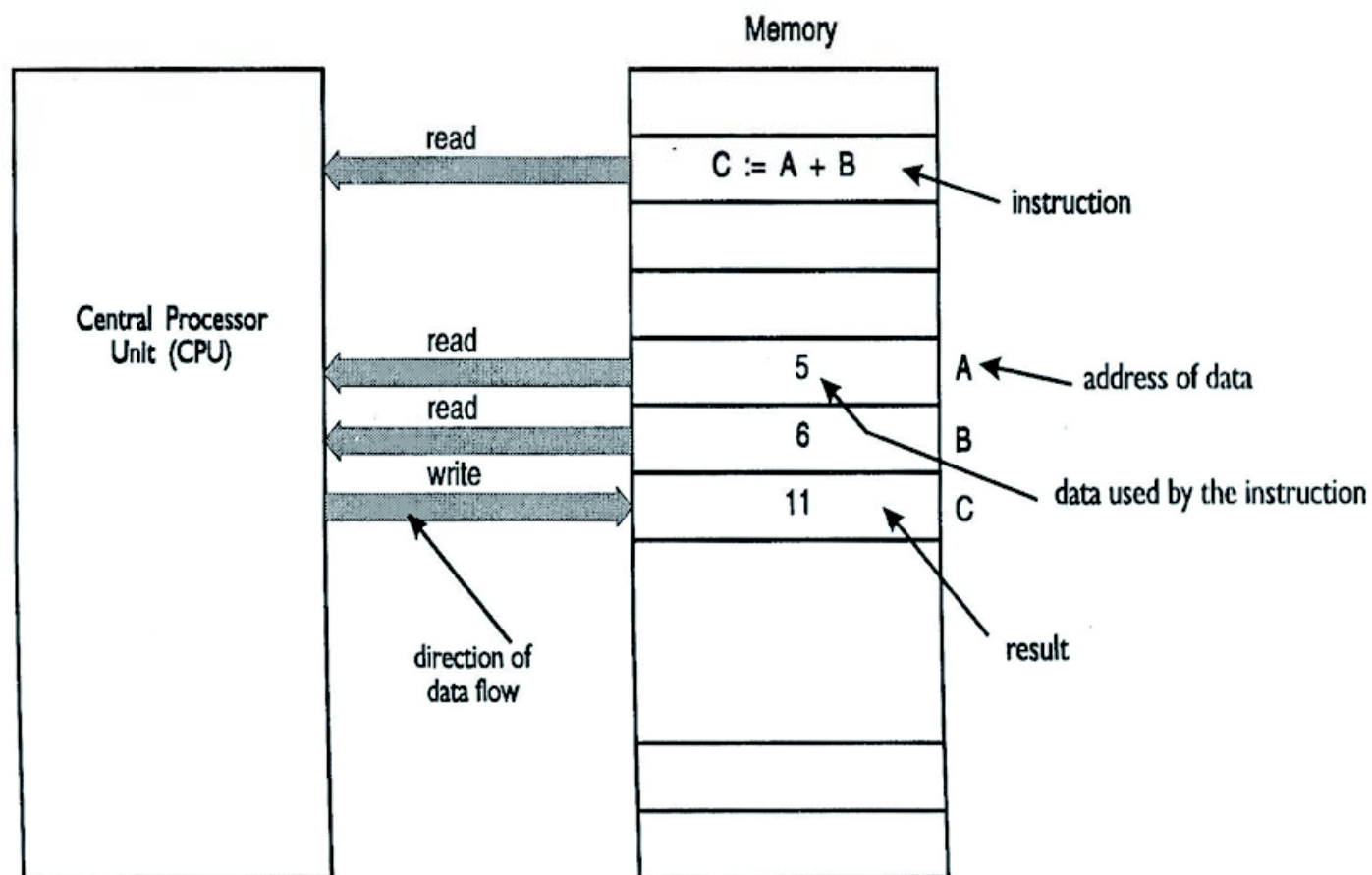


C=A+B 프로그램의 실행

- ❑ STEP1) C/C++ 컴파일러와 링커에 의해 기계어코드로 변환됨
 - ❑ STEP2) 기계어코드를 소정의 메모리(코드메모리라 칭함)에 저장시킴
 - ❑ STEP3) 프로세서는 PRE-FETCH라는 방법으로 버스를 통해 기계어코드를 코드메모리로부터 읽어들이м
 - ❑ STEP4) 프로세서는 해당 코드를 해석하고 실행함
 - ❑ STEP5) 해석된 코드가 버스 동작을 요구하는 것일 경우에 버스신호가 발생되며 일련의 동작을 수행함
-

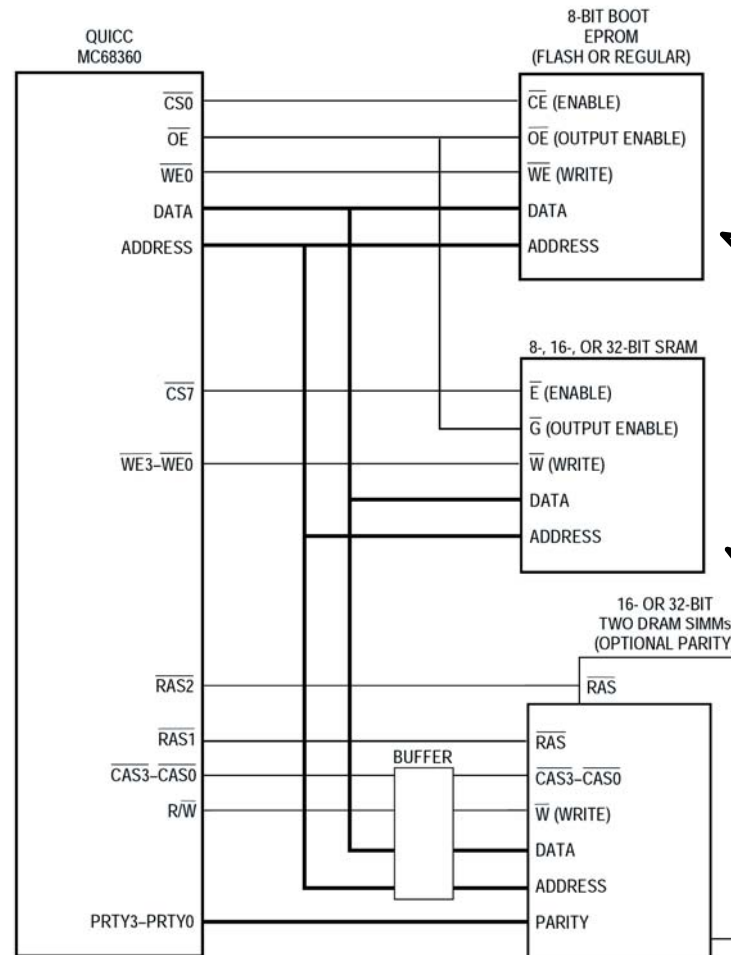


프로그램의 실행 프로세스





Programmers Memory Model



코드메모리(TEXT)

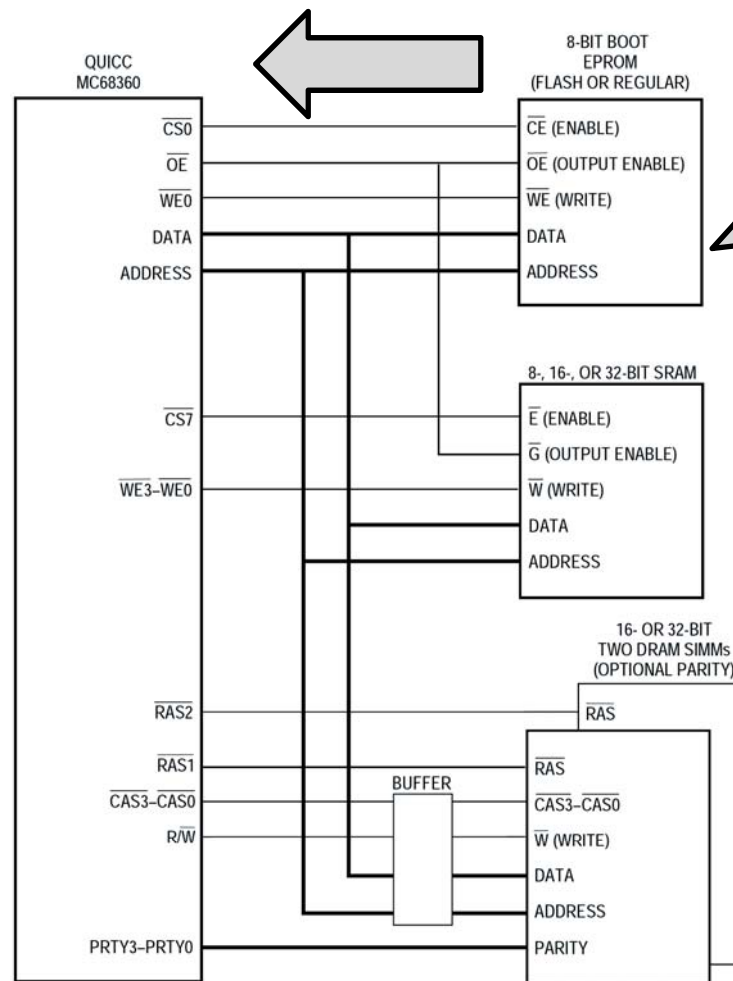
C=A+B 문장에 해당하는 기계어 코드가 이 영역에 상주한다

데이터메모리(RW,BSS)

변수 A, B, C 는 각각 이 영역에 위치한다



코드 메모리에서 기계어 코드를 읽어 들인다 (PRE-FETCH)

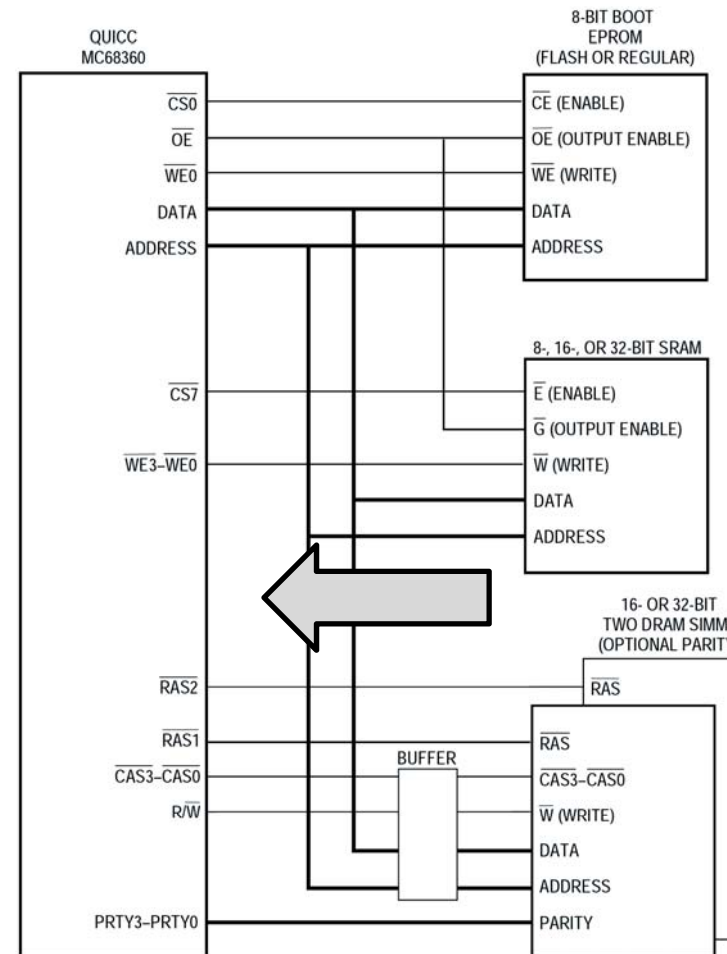


코드메모리(TEXT)

C=A+B 문장에 해당하는 기계어 코드가 이 영역에 상주한다



데이터 메모리에서 오퍼랜드를 읽어 들인다

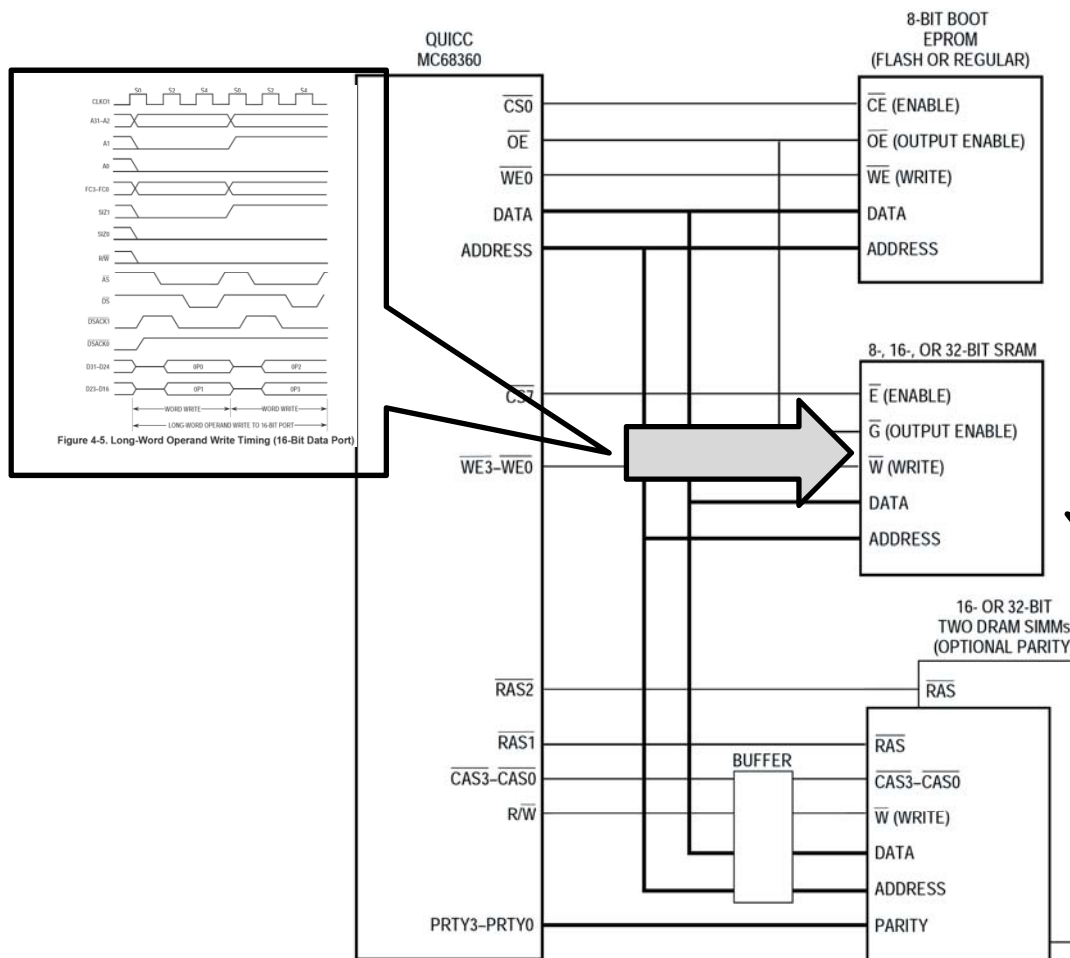


데이터메모리(RW,BSS)

변수 A와 B를 읽어들인다



연산 결과를 데이터 메모리에 기록한다



데이터메모리(RW,BSS)

변수 C 의 위치에 A+B 결과
를 기록한다



목 차

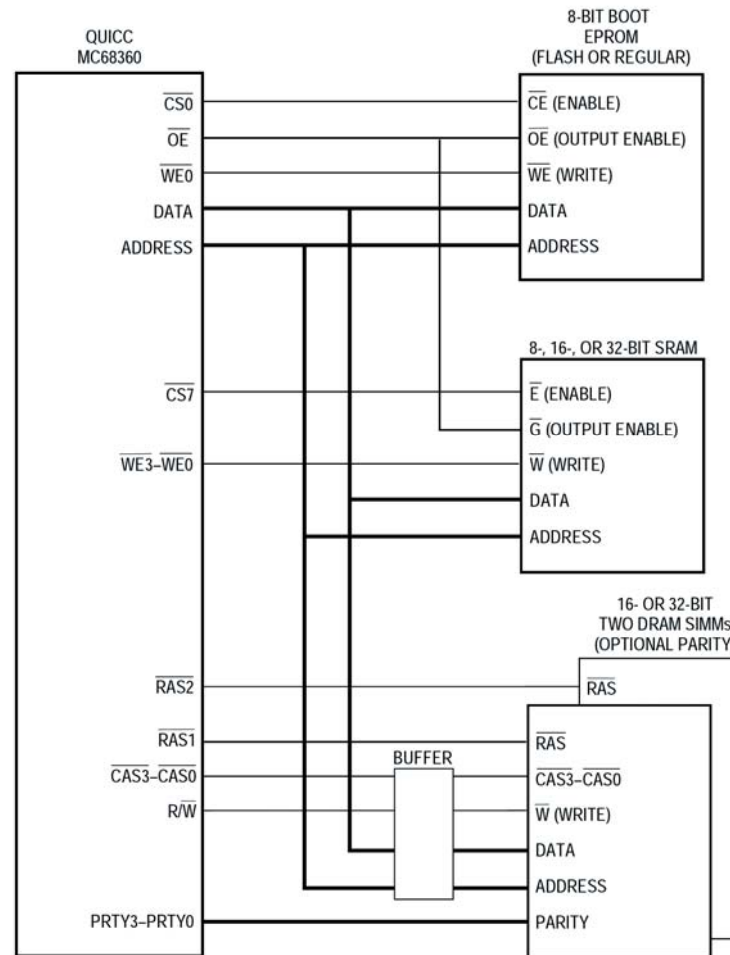
프로그램의 실행과 버스동작과의 관계

□ 버스 인터페이스의 사례

버스의 동작원리

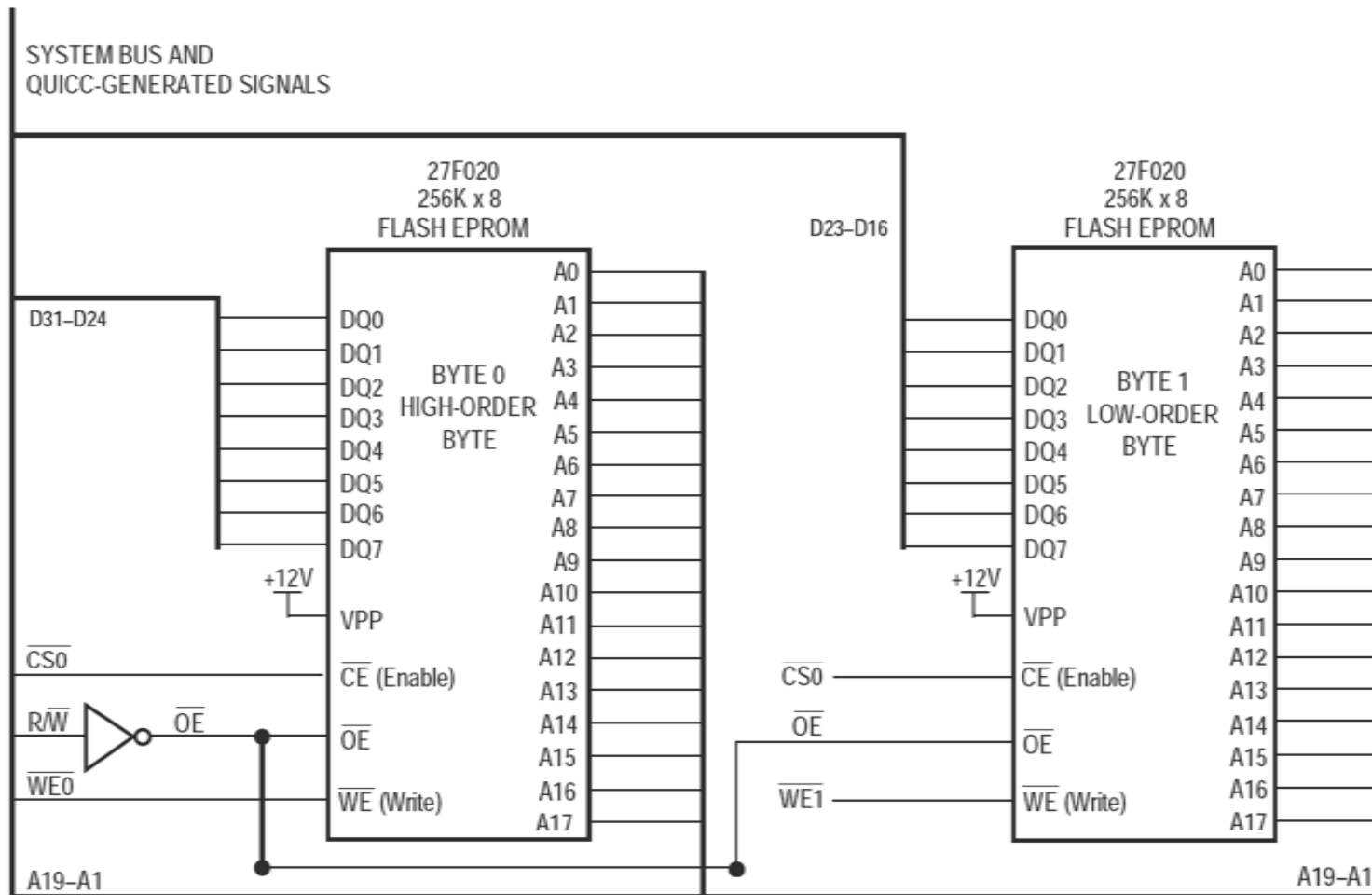


메모리(EPROM/SRAM/DRAM)의 인터페이스 예



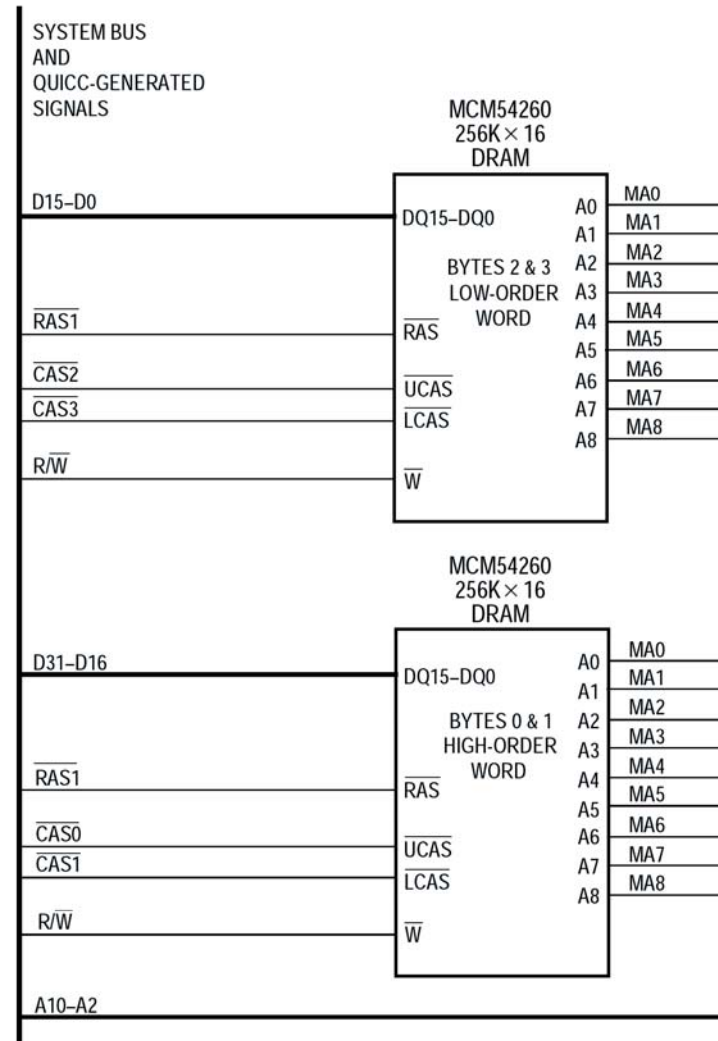


EPROM 과의 인터페이스 예





DRAM 과의 인터페이스 예





Ethernet controller과의 인터페이스 예

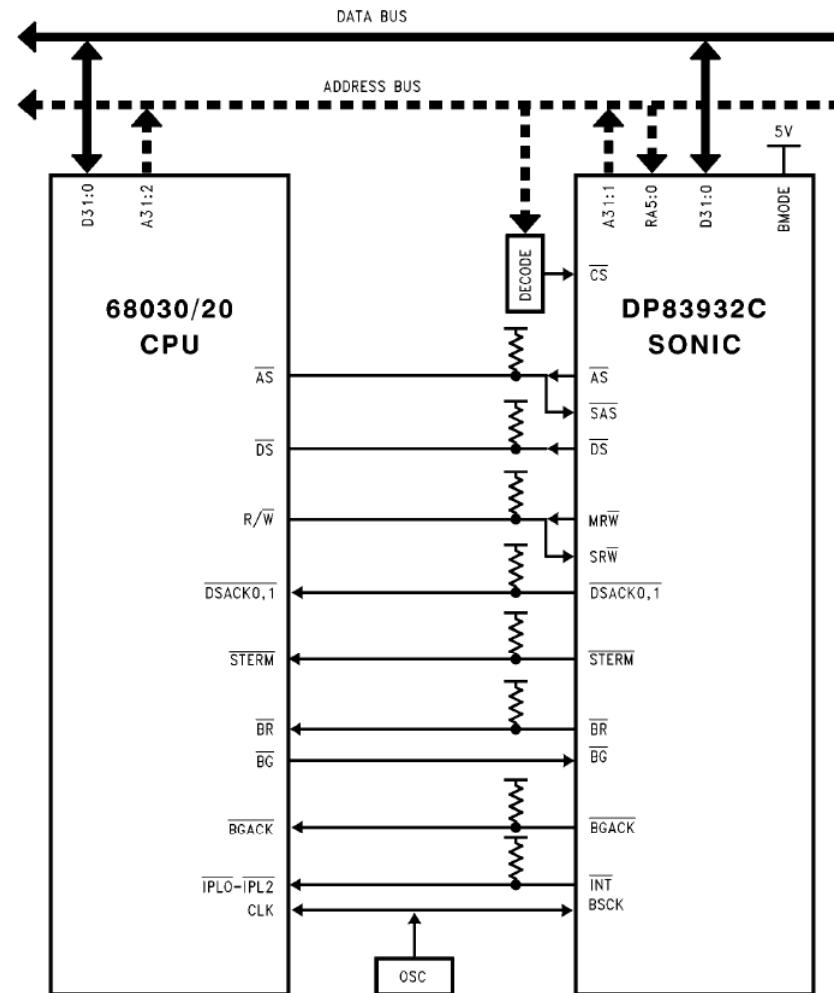


그림. Bus configuration of DP83932c Ethernet controller



LCD controller과의 인터페이스 예

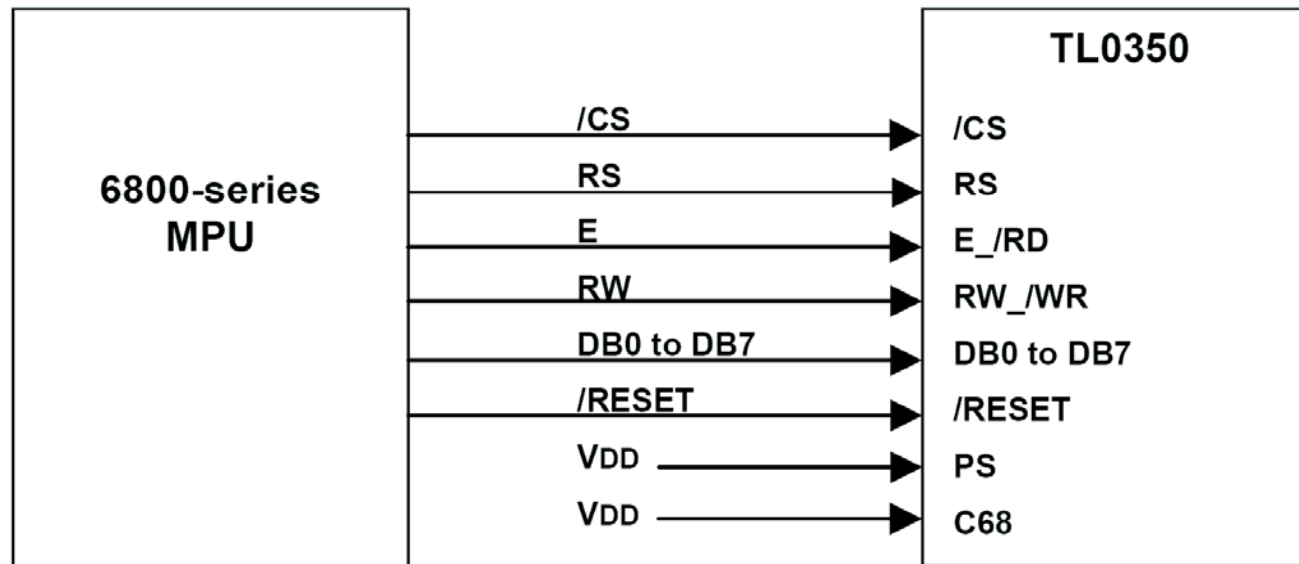


그림. Interfacing with TL0350 LCD controller



DSP 과의 인터페이스 예

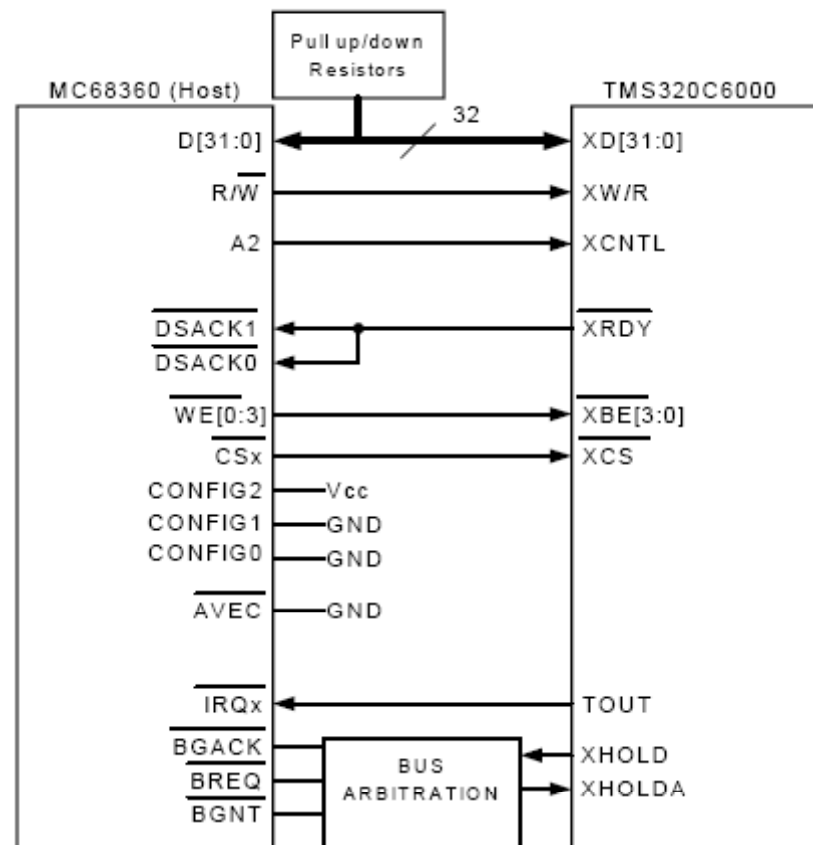


그림. Block Diagram of Interface Between MC68360 (Host) and Expansion Bus



목 차

프로그램의 실행과 버스동작과의 관계

버스 인터페이스의 사례

□ 버스의 동작원리



INPUT SAMPLE WINDOW

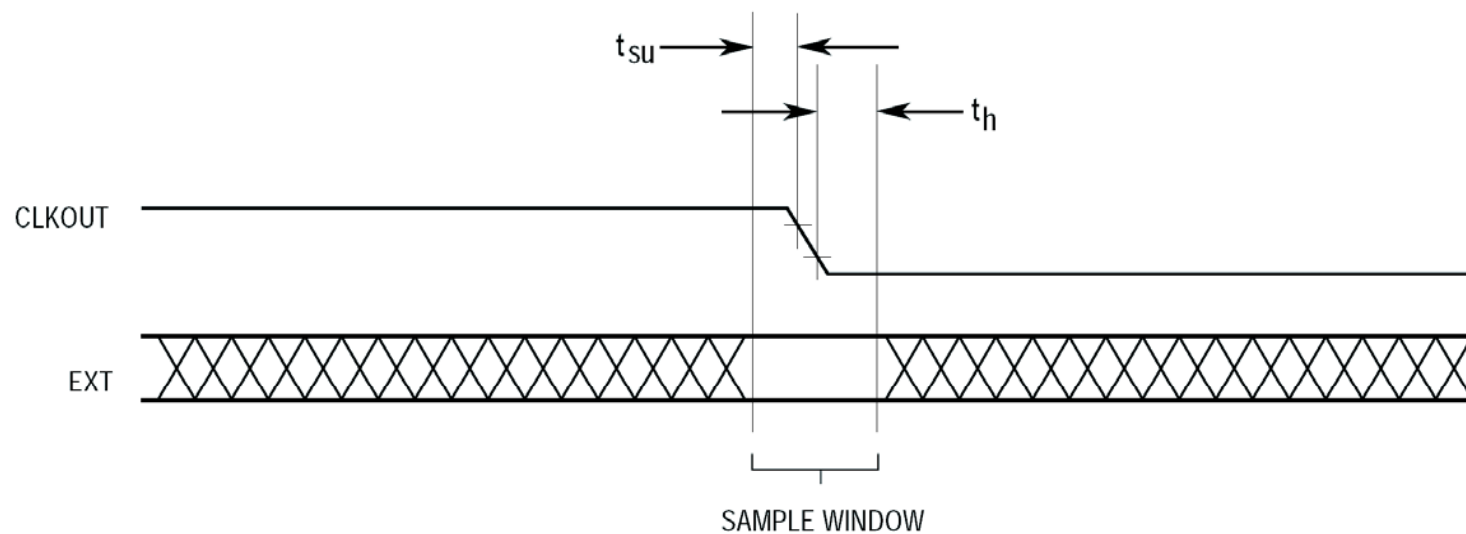


그림. Input Sample Window



SIGNAL CONVENTION

□ /SIGNAL 의 의미

❖ HIGH ACTIVE 신호

정의: HIGH일때 의미가 있는 신호

예) SIZ0, FC3

❖ LOW ACTIVE 신호

정의: LOW일때 의미가 있는 신호

특기 표기법: 신호명의 앞에 '/'를 붙인다

예) /CS, /WR, /AS, INT*, nRESET, $\overline{\text{ACK}}$, BREQ#

❖ ASSERT 의 의미

의미가 있는 경우의 신호되었을때 "ASSERT" 라 부르며 그 반대의 경우를 "NEGATE"라 부른다



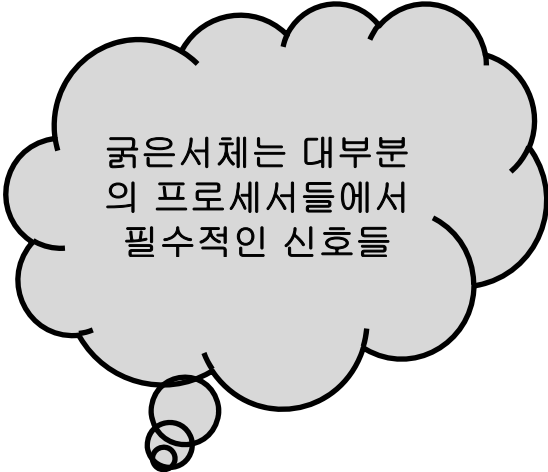
BUS 의 핵심 신호들

- ❖ 어드레스 버스 신호
 - ❖ 데이터 버스 신호
 - ❖ 읽고/쓰기 제어 신호
 - ❖ 칩셀렉트 제어 신호
 - ❖ 버스 클럭 신호
-



BUS의 구성 신호

- ❖ CLKOUT 신호: 버스 클럭(기준신호)
- ❖ SIZx 신호: 전송의 크기를 나타냄
- ❖ R/W 신호: 읽기/쓰기 제어
- ❖ FC3-0 신호: 어드레스 스페이스
- ❖ A31-0(어드레스 버스): 주소 버스
- ❖ AS 신호: 어드레스 스트로브
- ❖ D31-0(데이터 버스): 데이터 버스
- ❖ DS 신호: 데이터 스트로브
- ❖ DSACK1-0 신호: 버스사이클 터미네이터
- ❖ BERR 신호: 버스사이클 오류 발생
- ❖ /CS 신호: 칩선택



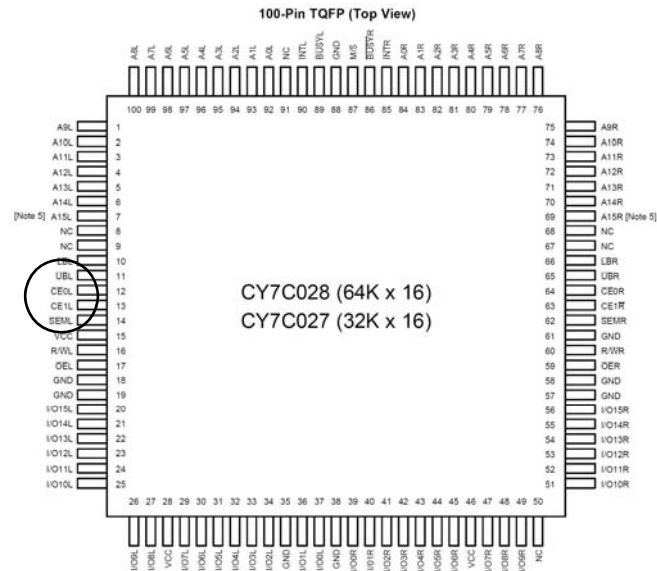
굵은서체는 대부분
의 프로세서들에서
필수적인 신호들



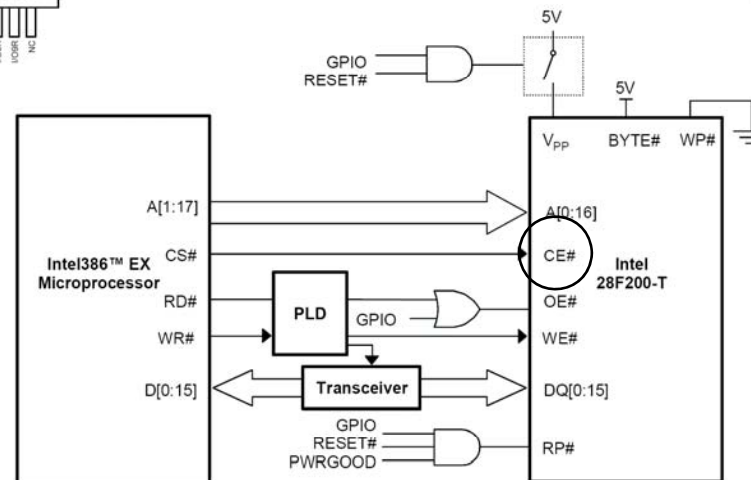
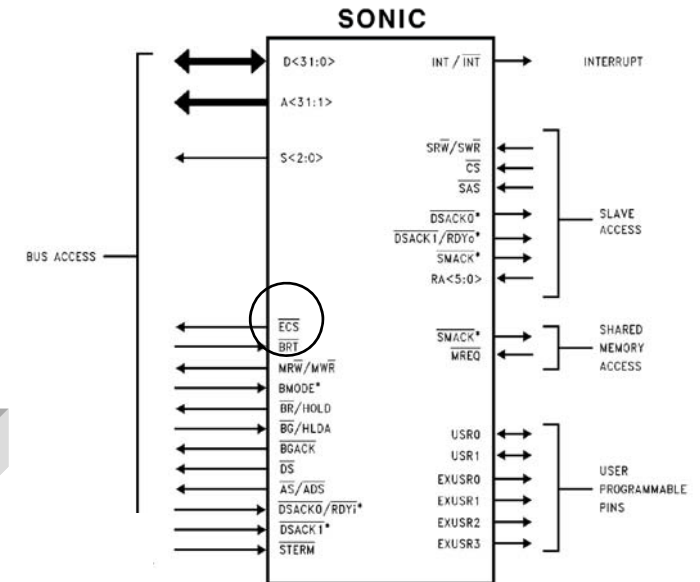
CHIP SELECT

Dual Port Ram

Ethernet Controller



Flash Memory





버스 동작 타이밍의 예

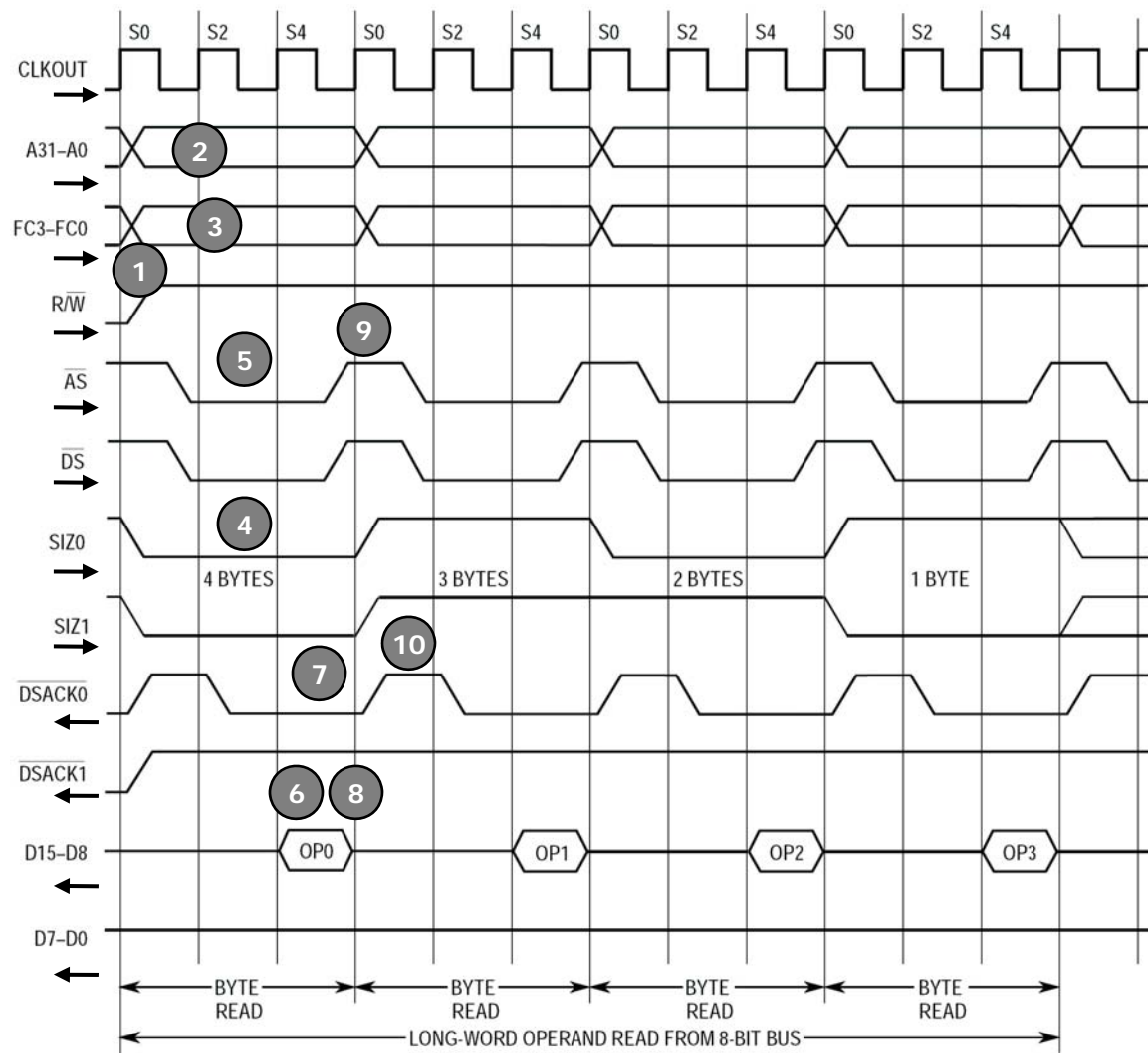
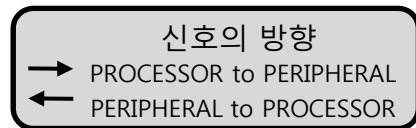


그림. Long-Word Operand Read Timing from 8-Bit Port



버스 동작 타이밍의 예

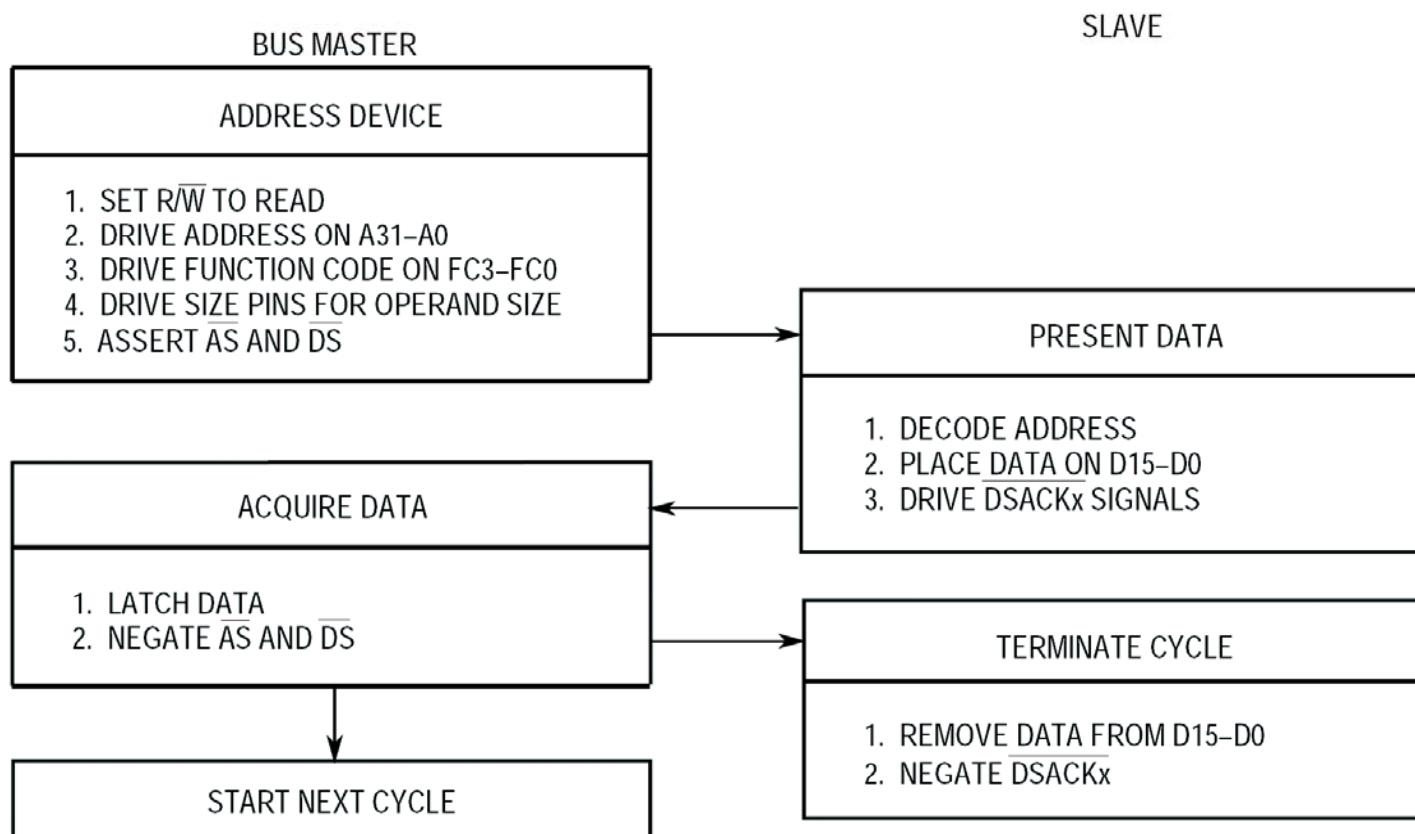


그림. Word Read Cycle Flowchart



오퍼랜드의 크기 종류(1/2/4)

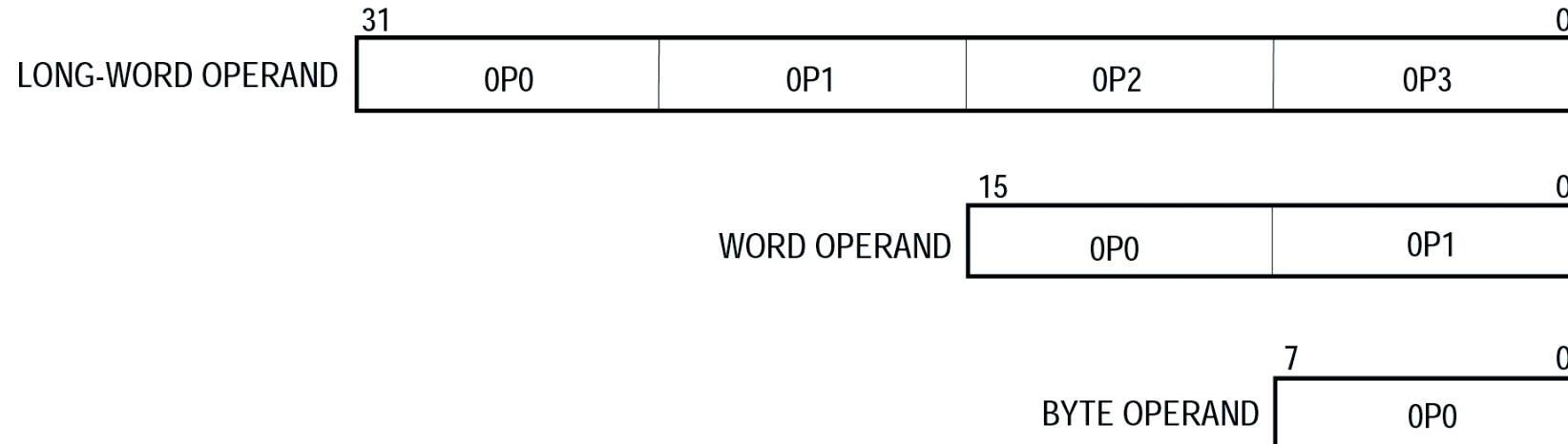


그림. Internal Operand Representation



임베디드에서 사용되는 자주 사용되는 연산 '<<, >>'

□ << , >> 쉬프트

❖ 로케이터

비트의 위치 선정

예1) `rGPECON = (0x1<<3);`

예2) `rGPECON >>= 16; // select upper byte 16`

❖ 곱하기/나누기

정수배의 산술 곱하기/나눗셈

예1) `SUM <=< 2; // multiply by '4'`

예2) `SUM = value >> 1; // divide by '2'`



임베디드에서 사용되는 자주 사용되는 연산 '&= 와 |=

□ '&= '와 '|=' 마스크

❖ &= [~]

선택된 비트 패턴 '클리어(zero fill 마스크)'

예1) `rGPECON &= ~((0x3<<22) | (0x3<<24));`

예2) `rGPECON &= 0xFFFF0000;`

❖ |=

선택된 비트 패턴 '세트(기록)'

예1) `rGPEDAT |= (0x1<<11) | (0x1<<12);`

예2) `rGPEDAT |= 0x00ff0000;` ('1' fill 마스크)



임베디드에서 사용되는 자주 사용되는 연산 '&= 와 |=

□ '&'와 '|' 의 종합 사용 예

❖ 임의의 연속된 2비트 이상을 변경코저 할때

=>연속된 비트 패턴을 우선 '클리어(zero clear)'
한 후 이어 새로운 값을 '세트'

예1) `rGPECON &= ~((0x3<<22) | (0x3<<24));`
`rGPECON |= ((0x1<<22) | (0x1<<24));`



논리 회로(식) 'AND'

□ C 연산자
' & '

□ 특징

❖ ZERO(MASK '0')

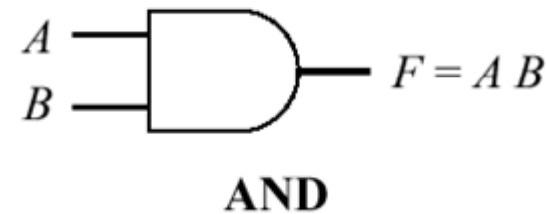
임의의 값과 '0' 을 AND 하면 결과는 항상 '0'

❖ COMPARATOR

임의의 값을 비교하는 비교기

eg. `if(status_flag & (1<<3)) // status_flag[3]이 1인가 ?`

<i>A</i>	<i>B</i>	<i>F</i>
0	0	0
0	1	0
1	0	0
1	1	1





논리 회로(식) 'OR'

□ C 연산자

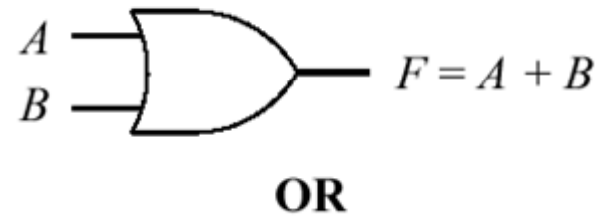
'|'

□ 특징

❖ ONE(MASK '1')

임의의 값과 '1' 을 OR 하면 결과는 항상 '1'

<i>A</i>	<i>B</i>	<i>F</i>
0	0	0
0	1	1
1	0	1
1	1	1





논리 회로(식) 'XOR'

□ C 연산자

' ^ '

□ 특징

❖ TOGGLE

임의의 값과 '1' 을 XOR 하면 토글의 효과를 얻음

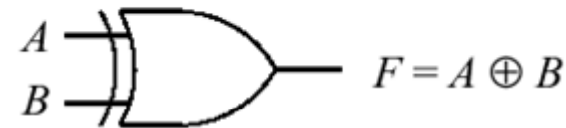
❖ SET to ZERO('0')

eg. `result = value ^ value` // `result = 0` 와 동일한 결과

❖ COMPARATOR

eg. `if(!(status ^ 0x1234))` // status 이 0x1234인가 ?

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



Exclusive-OR (XOR)



질의 응답
