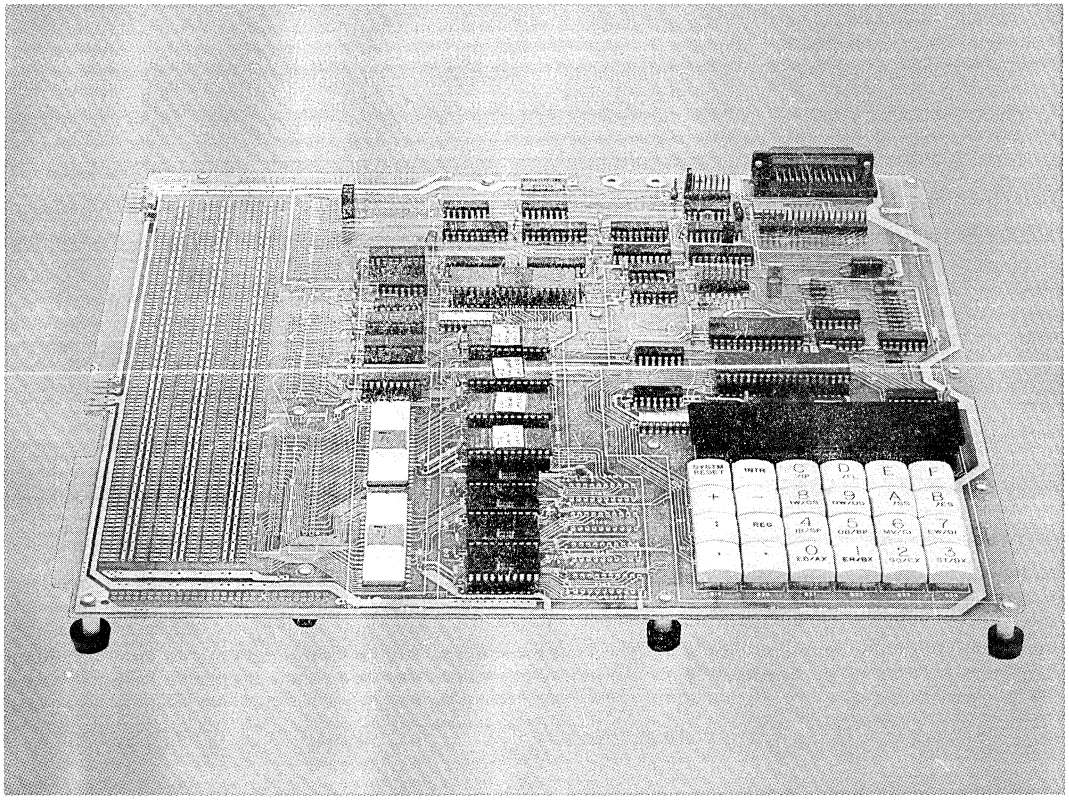
SDK-86
MCS-86™ SYSTEM DESIGN KIT

- **Complete Single Board Microcomputer System Including CPU, Memory, and I/O**
- **Easy to Assemble Kit Form**
- **High Performance 8086 16-Bit CPU**
- **Interfaces Directly with TTY or CRT**
- **Interactive LED Display and Keyboard**
- **Wire Wrap Area for Custom Interfaces**
- **Extensive System Monitor Software in ROM**
- **Comprehensive Design Library Included**

The SDK-86 MCS-86 System Design Kit is a complete single board 8086 microcomputer system in kit form. It contains all necessary components to complete construction of the kit, including LED display, keyboard, resistors, caps, crystal, and miscellaneous hardware. Included are preprogrammed ROMs containing a system monitor for general software utilities and system diagnostics. The complete kit includes an 8-digit LED display and a mnemonic 24-key keyboard for direct insertion, examination, and execution of a user's program. In addition, it can be directly interfaced with a teletype terminal, CRT terminal, or the serial port of an Intellec system. The SDK-86 is a high performance prototype system with designed-in flexibility for simple interface to the user's application.



FUNCTIONAL DESCRIPTION

The SDK-86 is a complete MCS-86 microcomputer system on a single board, in kit form. It contains all necessary components to build a useful, functional system. Such items as resistors, caps, and sockets are included. Assembly time varies from 4 to 10 hours, depending on the skill of the user. The SDK-86 functional block diagram is shown in Figure 1.

8086 Processor

The SDK-86 is designed around Intel's 8086 microprocessor. The Intel 8086 is a new generation, high performance microprocessor implemented in N-channel, depletion load, silicon gate technology (HMOS), and packaged in a 40-pin CerDIP package. The processor features attributes of both 8-bit and 16-bit microprocessors in that it addresses memory as a sequence of 8-bit bytes, but has a 16-bit wide physical path to memory for high performance. Additional features of the 8086 include the following:

- Direct addressing capability to one megabyte of memory
- Assembly language compatibility with 8080/8085
- 14 word x 16-bit register set with symmetrical operations
- 24 operand addressing modes
- Bit, byte, word, and block operations
- 8 and 16-byte signed and unsigned arithmetic in binary or decimal mode, including multiply and divide
- 4 or 5 or 8 MHz clock rate

A block diagram of the 8086 microprocessor is shown in Figure 2.

System Monitor

A compact but powerful system monitor is supplied with the SDK-86 to provide general software utilities and system diagnostics. It comes in preprogrammed read only memories (ROMs).

Communications Interface

The SDK-86 communicates with the outside world through either the on-board light emitting diode (LED) display/keyboard combination or the user's TTY or CRT terminal (jumper selectable), or by means of a special mode in which an Intellec development system transports finished programs to and from the SDK-86. Memory may be easily expanded by simply soldering in additional devices in locations provided for this purpose. A large area of the board (22 square inches) is laid out as general purpose wire-wrap for the user's custom interfaces.

Assembly

Only a few simple tools are required for assembly: soldering iron, cutters, screwdriver, etc. The SDK-86 assembly manual contains step-by-step instructions for easy assembly with a minimum of mistakes. Once construction is complete, the user connects his kit to a power supply and the SDK-86 is ready to go. The monitor starts immediately upon power-on or reset.

Commands — Keyboard mode commands, serial port commands, and Intellec slave mode commands are summarized in Table 1, Table 2, and Table 3, respectively. The SDK-86 keyboard is shown in Figure 3.

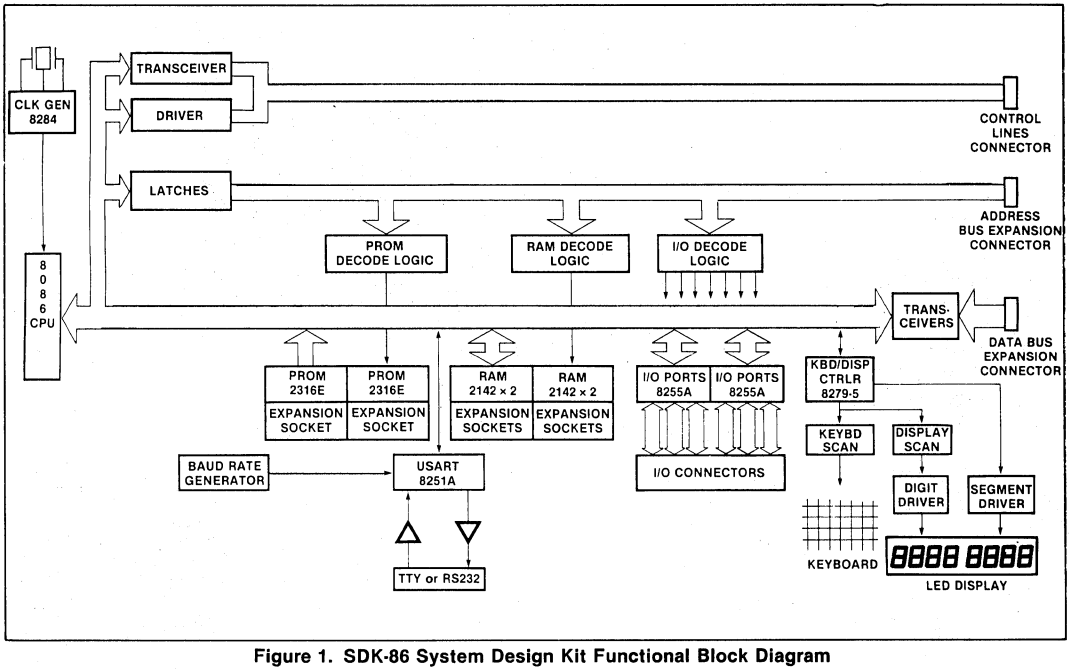


Figure 1. SDK-86 System Design Kit Functional Block Diagram

Documentation

In addition to detailed information on using the monitors, the SDK-86 user's manual provides circuit diagrams, a monitor listing, and a description of how the system works. The complete design library for the SDK-86 is shown in Figure 4 and listed in the specifications section under Reference Manuals.

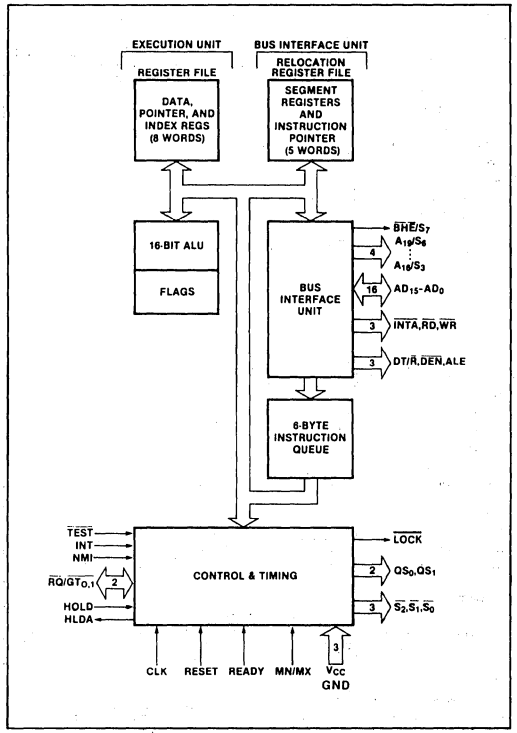


Figure 2. 8086 Microprocessor Block Diagram

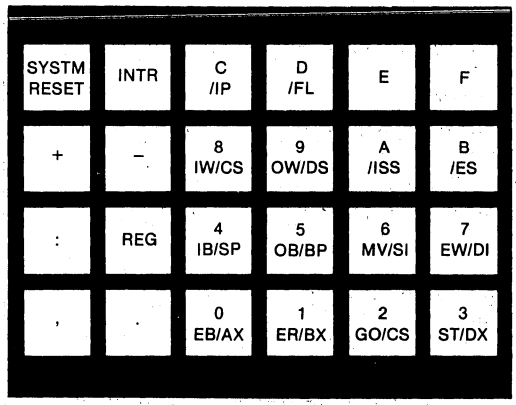


Figure 3. SDK-86 Keyboard

Figure 4. SDK-86 Design Library

Table 1. Keyboard Mode Commands

Command	Operation
Reset	Starts monitor.
Go	Allows user to execute user program, and causes it to halt at predetermined program stop. Useful for debugging.
Single step	Allows user to execute user program one instruction at a time. Useful for debugging.
Substitute memory	Allows user to examine and modify memory locations in byte or word mode.
Examine register	Allows user to examine and modify 8086 register contents.
Block move	Allows user to relocate program and data portions in memory.
Input or output	Allows direct control of SDK-86 I/O facilities in byte or mode.

Table 2. Serial Mode Commands

Command	Operation
Dump memory	Allows user to print or display large blocks of memory information in hex format than amount visible on terminal's CRT display.
Start/continue display	Allows user to display blocks of memory information larger than amount visible on terminal's CRT display.
Punch/read paper tape	Allows user to transmit finished programs into and out of SDK-86 via TTY paper tape punch.

8086 INSTRUCTION SET

Table 4 contains a summary of processor instructions used for the 8086 microprocessor.

Table 4. 8086 Instruction Set Summary

Mnemonic and Description	76543210	76543210	76543210	76543210
Data Transfer				
MOV = Move:				
Register/memory to/from register	100010dw	mod reg r/m		
Immediate to register/memory	1100011w	mod000 r/m	data	data if w=1
Immediate to register	1011w reg	data	data if w=1	
Memory to accumulator	1010000w	addr-low	addr-high	
Accumulator to memory	1010001w	addr-low	addr-high	
Register/memory to segment register	10001110	mod0reg r/m		
Segment register to register/memory	10001100	mod0reg r/m		
PUSH = Push:				
Register/memory	11111111	mod110 r/m		
Register	01010 reg			
Segment register	000reg110			
POP = Pop:				
Register/memory	10001111	mod000 r/m		
Register	01011 reg			
Segment register	000reg111			
XCHG = Exchange:				
Register/memory with register	1000011w	mod reg r/m		
Register with accumulator	10010 reg			
IN = Input				
Fixed port	1110010w	port		
Variable port	1110110w			
OUT = Output				
Fixed port	1110011w	port		
Variable port	1110111w			
XLAT=Translate byte to AL	11010111			
LEA=Load EA to register	10001101	mod reg r/m		
LDS=Load pointer to DS	11000101	mod reg r/m		
LES=Load pointer to ES	11000100	mod reg r/m		
LAHF=Load AH with flags	10011111			
SAHF=Store AH into flags	10011110			
PUSHF=Push flags	10011100			
POPF=Pop flags	10011101			
Arithmetic				
ADD = Add:				
Reg./memory with register to either	000000dw	mod reg r/m		
Immediate to register/memory	100000sw	mod000 r/m	data	data if s:w=01
Immediate to accumulator	0000010w	data	data if w=1	
ADC = Add with carry:				
Reg./memory with register to either	000100dw	mod reg r/m		
Immediate to register/memory	100000sw	mod010 r/m	data	data if s:w=01
Immediate to accumulator	0001010w	data	data if w=1	
INC = Increment:				
Register/memory	1111111w	mod000 r/m		
Register	01000 reg			
AAA=ASCII adjust for add	00110111			
DAA=Decimal adjust for add	00100111			
SUB = Subtract:				
Reg./memory and register to either	001010dw	mod reg r/m		
Immediate from register/memory	100000sw	mod101 r/m	data	data if s:w=01
Immediate from accumulator	0010110w	data	data if w=1	
SBB = Subtract with borrow				
Reg./memory and register to either	000110dw	mod reg r/m		
Immediate from register/memory	100000sw	mod011 r/m	data	data if s:w=01
Immediate from accumulator	0001110w	data	data if w=1	
DEC = Decrement:				
Register/memory	1111111w	mod001 r/m		
Register	01001 reg			
NEG=Change sign	1111011w	mod011 r/m		

Mnemonic and Description	76543210	76543210	76543210	76543210
CMP = Compare:				
Register/memory and register	001110dw	mod reg r/m		
Immediate with register/memory	100000sw	mod111 r/m	data	data if s:w=01
Immediate with accumulator	0011110w	data	data if w=1	
AAS=ASCII adjust for subtract	00111111			
DAS=Decimal adjust for subtract	00101111			
MUL=Multiply (unsigned)	1111011w	mod100 r/m		
IMUL=Integer multiply (signed)	1111011w	mod101 r/m		
AAM=ASCII adjust for multiply	11010100	00001010		
DIV=Divide (unsigned)	1111011w	mod110 r/m		
IDIV=Integer divide (signed)	1111011w	mod111 r/m		
AAD=ASCII adjust for divide	11010101	00001010		
CBW=Convert byte to word	10011000			
CWD=Convert word to double word	10011001			
Logic				
NOT=Invert	1111011w	mod010 r/m		
SHL/SAL=Shift logical/arithmetic left	110100vw	mod100 r/m		
SHR=Shift logical right	110100vw	mod101 r/m		
SAR=Shift arithmetic right	110100vw	mod111 r/m		
ROL=Rotate left	110100vw	mod000 r/m		
ROR=Rotate right	110100vw	mod001 r/m		
RCL=Rotate through carry flag left	110100vw	mod010 r/m		
RCR=Rotate through carry right	110100vw	mod011 r/m		
AND = And:				
Reg./memory and register to either	001000dw	mod reg r/m		
Immediate to register/memory	1000000w	mod100 r/m	data	data if w=1
Immediate to accumulator	0010010w	data	data if w=1	
TEST = And function to flags, no result:				
Register/memory and register	1000010w	mod reg r/m		
Immediate data and register/memory	1111011w	mod000 r/m	data	data if w=1
Immediate data and accumulator	1010100w	data	data if w=1	
OR = Or:				
Reg./memory and register to either	000010dw	mod reg r/m		
Immediate to register/memory	1000000w	mod001 r/m	data	data if w=1
Immediate to accumulator	0000110w	data	data if w=1	
XOR = Exclusive or:				
Reg./memory and register to either	001100dw	mod reg r/m		
Immediate to register/memory	1000000w	mod110 r/m	data	data if w=1
Immediate to accumulator	0011010w	data	data if w=1	
String Manipulation				
REP=Repeat	1111001z			
MOVS=Move byte/word	1010010w			
CMPS=Compare byte/word	1010011w			
SCAS=Scan byte/word	1010111w			
LODS=Load byte/wd to AL/AX	1010110w			
STOS=Stor byte/wd frm AL/A	1010101w			
Control Transfer				
CALL = Call:				
Direct within segment	11101000	disp-low	disp-high	
Indirect within segment	11111111	mod010 r/m		
Direct intersegment	10011010	offset-low	offset-high	
		seg-low	seg-high	
Indirect intersegment	11111111	mod011 r/m		

continued

Table 4. 8086 Instruction Set Summary (Continued)

Mnemonic and Description	76543210	76543210	76543210
JMP - Unconditional Jump:			
Direct within segment	11101001	disp-low	disp-high
Direct within segment-short	11101011	disp	
Indirect within segment	11111111	mod 100 r/m	
Direct intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	11111111	mod 101 r/m	
RET - Return from CALL:			
Within segment	11000011		
Within seg. adding immed to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment, adding immediate to SP	11001010	data-low	data-high
JE/JZ-Jump on equal/zero	01110100	disp	
JL/JNGE-Jump on less/not greater or equal	01111100	disp	
JLE/JNG-Jump on less or equal/not greater	01111110	disp	
JB/JNAE-Jump on below/not above or equal	01110010	disp	
JBE/JNA-Jump on below or equal/ not above	01110110	disp	
JP/JPE-Jump on parity/parity even	01111010	disp	
JO-Jump on overflow	01110000	disp	
JS-Jump on sign	01111000	disp	
JNE/JNZ-Jump on not equal/not zero	01110101	disp	
JNL/JGE-Jump on not less/greater or equal	01111101	disp	
JNLE/JG-Jump on not less or equal/ greater	01111111	disp	
JNB/JAE-Jump on not below/above or equal	01110011	disp	
JNBE/JA-Jump on not below or equal/above	01110111	disp	
JNP/JPO-Jump on not par/par odd	01111011	disp	
JNO-Jump on not overflow	01110001	disp	

Mnemonic and Description	76543210	76543210
JNS Jump on not sign	01111001	disp
LOOP Loop CX times	11100010	disp
LOOPZ/LOOPE-Loop while zero/equal	11100001	disp
LOOPNZ/LOOPNE-Loop while not zero/equal	11100000	disp
JCXZ-Jump on CX zero	11100011	disp
INT Interrupt		
Type specified	11001101	type
Type 3	11001100	
INTO-Interrupt on overflow	11001110	
IRET Interrupt return	11001111	
Processor Control		
CLC Clear carry	11111000	
CMC Complement carry	11110101	
STC Set carry	11111001	
CLD Clear direction	11111100	
STD-Set direction	11111101	
CLI Clear interrupt	11111010	
STI-Set interrupt	11111011	
HLT Halt	11110100	
WAIT Wait	10011011	
ESC Escape (to external device)	11011xxx	mod x x x r/m
LOCK Bus lock prefix	11110000	

Notes

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

if s:w = 01 then 16 bits of immediate data form the operand.
if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand.
if v = 0 then "count" = 1; if v = 0 then "count" in (CL)
x = don't care.
if v = 0 then "count" = 1; if v = 1 then "count" in (CL) register.
z is used for string primitives for comparison with ZF FLAG.

SEGMENT OVERRIDE PREFIX

001 reg 110

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

SPECIFICATIONS

Central Processor

CPU — 8086 (5 MHz clock rate)

Note
May be operated at 2.5 MHz or 5 MHz, jumper selectable, for use with 8086.

Memory

ROM — 8K bytes 2316/2716
RAM — 2K bytes (expandable to 4K bytes) 2142

Addressing

ROM — FE000-FFFFF
RAM — 0-7FF (800-FFF available with additional 2142's)

Note
The wire-wrap area of the SDK-86 PC board may be used for additional custom memory expansion.

Input/Output

Parallel — 48 lines (two 8255A's)
Serial — RS232 or current loop (8251A)
Baud Rate — selectable from 110 to 4800 baud

Interfaces

Bus — All signals TTL compatible

Parallel I/O — All signals TTL compatible

Serial I/O — 20 mA current loop TTY or RS232

Note
The user has access to all bus signals which enable him to design custom system expansions into the kit's wire-wrap area.

Interrupts (256 vectored)

Maskable
Non-maskable
TRAP

DMA

Hold Request — Jumper selectable. TTL compatible input.

Software

System Monitor — Preprogrammed 2716 or 2316 ROMs

Addresses — FE000-FFFFF

Monitor I/O — Keyboard/display or TTY or CRT (serial I/O)

Physical Characteristics

Width — 13.5 in. (34.3 cm)

Height — 12 in. (30.5 cm)

Depth — 1.75 in. (4.45 cm)

Weight — approx. 24 oz. (3.3 kg)

Electrical Characteristics

DC Power Requirement
(Power supply not included in kit)

Voltage	Current
V_{CC} 5V ± 5%	3.5A
V_{TTY} – 12V ± 10%	0.3A
	(V_{TTY} required only if teletype is connected)

Environmental Characteristics

Operating Temperature — 0–50°C

Reference Manuals

9800697A — SDK-86 MCS-86 System Design Kit Assembly Manual

9800722 — MCS-86 User's Manual

9800640A — 8086 Assembly Language Programming Manual

8086 Assembly Language Reference Card

Reference manuals are shipped with each product only if designated SUPPLIED (see above). Manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.

ORDERING INFORMATION

Part Number	Description
SDK-86	MCS-86 system design kit