CME-562

Development Board

xiom
anufacturing
™

2813 Industrial Ln. • Garland, TX 75041 • 972-926-9303 FAX 972-926-6063
email: sales@axman.com, support@axman.com • web: http://www.axman.com

TABLE OF CONTENTS

TABLE OF CONTENTS 2

GETTING STARTED 3

DEVELOPMENT PHILOSOPHY 3

OPERATING MODES 4

DEVELOPMENT MODE 4
AUTOSTART RAM MODE 5
AUTOSTART EEPROM MODE 5
AUTOSTART U5 MODE 5

QUICK TUTORIAL 6

HARDWARE 7

SERIAL PORT COM1 7
SS : KEYPAD INTERFACE 8
LCD INTERFACE 8
MCU_PORT CONNECTOR 9
A/D REFERENCE 9
EXPANSION BUS 10
MEMORY AND ADDRESSING 11
MEMORY SELECTION JUMPERS 11

TROUBLESHOOTING 12

SOFTWARE 12

8051 MONITOR PROGRAM NOTES 13

MONITOR COMMANDS 13

8051 TINY BASIC NOTES 14

80562 INSTRUCTION SET SUMMARY 14

GETTING STARTED

The Axiom CME-562 single board computer is a fully assembled, fully functional development system complete with wall plug style power supply and serial cable. Primary interface to the system is through the serial port identified as COM1 on the development board. Connect this serial port to a serial port on your PC using the supplied cable.

Communications settings are 9600 baud, 1 start, 1 stop, 8 data, no parity. Most any communications program that can be set to these parameters will work with this system. Even the Terminal feature in Windows 3.1 will do an adequate job. We have supplied a simple terminal program on the software disk called AX52 that is already configured correctly for the CME-562.

After the serial cable is connected, apply power to the board by plugging in the wall plug power supply that came with the system. If everything is working properly, the Monitor or Basic prompt will appear on the PC display.

If you do not see the Monitor or Basic prompt, see the section titled **TROUBLESHOOTING**.

DEVELOPMENT PHILOSOPHY

Project development is performed by using the Monitor or Basic software installed in U5 to create or assist in creation of a program that is stored in U7 (Assembly Code Origination of $8000 hex) for testing purposes. After satisfactory operation of the program under the monitor or basic environment, the program can be autostarted by removing the autostart jumper. Final permanent program completion is performed relocating Assembly Code to Originate at $0000 hex and performing an Offset Load (not necessary with Basic), then performing an ECOPY8 or 32 command to move the code and program an EPROM located in U6. This EPROM is then moved to the U5 position for firmware operation.

OPERATING MODES

There are basically four different operating modes that the CME-562 can be set for. You should become familiar with the different modes of operation available, though most of the time you will probably be using **DEVELOPMENT** mode. For a complete listing of all the jumpers and other hardware specifications, see the **HARDWARE** section.

DEVELOPMENT MODE

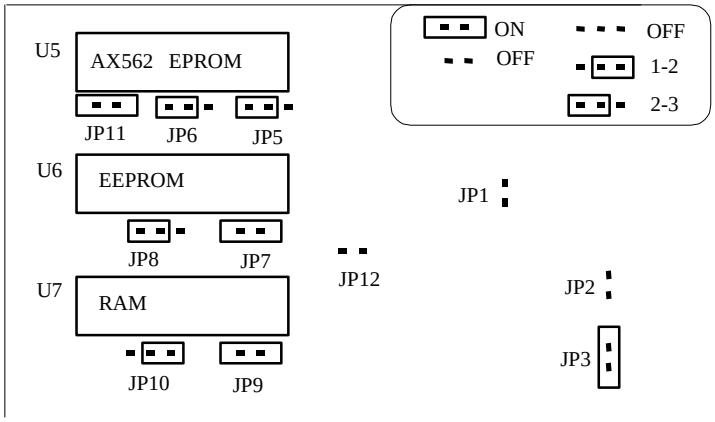
This is the default mode and is the mode which the CME-562 was set to when you received it. **DEVELOPMENT** mode is the best setting to use when developing your software since in this mode all of your code is located in RAM and you can quickly modify it.

For **DEVELOPMENT** mode, you should assemble or compile your software to start at 8000h. The first 7B hex bytes in your software should be interrupt vector addresses. You can make them all point to the start of your program if you want, as long as this space is reserved. The first instruction in your program will therefore be at address 807C hex. Neither the Monitor nor the Tiny Basic interpreter have their own interrupt service routines, they will use yours.

The AX562 EPROM, which contains the software development Monitor tool as well as the Tiny Basic interpreter, should be in the U5 socket. U6 should contain the EEPROM chip, either an 8K or 32K size. U7 should contain the RAM chip, either 8K or 32K size. **NOTE**: for the rest of this tutorial we will assume that you're using 8K to make this simpler. If you have the 32K upgrade simply use 32K options where 8K is specified.

For **DEVELOPMENT** mode, jumpers on the board should be set as follows:

Jumper	Setting
3	ON
5	2-3 *
6	2-3
7	ON
8	2-3
9	ON
10	1-2
11	ON
12	OFF - 8k
	ON - 32K



* If running Basic - JP5 should be OFF

AUTOSTART RAM MODE

In this mode your program will execute from RAM automatically whenever RESET is applied. The CME-562 should be configured identical to **DEVELOPMENT** mode, except JP3 should be OFF.

AUTOSTART EEPROM MODE

This is the recommended mode for starting your program automatically on powerup. Your code will execute from EEPROM whenever RESET or Power is applied.

Before using AUTOSTART EEPROM mode, you should first configure the board to **DEVELOPMENT** mode and get your code running from RAM. When everything is working from RAM, follow these steps:

1. Copy the contents of RAM, where your program is stored, into the EEPROM. Do this by executing either the 8K or 32K EEPROM copy command using the Monitor utility or the Basic command line.
2. Remove power.
3. Remove Jumpers 3, 7 and 9.
4. Remove the RAM chip from the U7 socket.
5. Remove the EEPROM chip from U6 and install it in the empty U7 socket.
6. Install the RAM chip into the empty U6 socket.
7. Re-Apply power.

Your software should now start automatically, just like it did from RAM.

AUTOSTART U5 MODE

It is possible to execute your code from an EEPROM in the U5 socket. However, since you must remove the Basic interpreter EPROM to do this, you cannot run a Tiny Basic program this way. This method does give you complete access to the full memory map addressed to U5, U6 and U7 however.

As in AUTOSTART EEPROM mode, it is recommended that you first configure the board to **DEVELOPMENT** mode and get your code running from RAM. When you're satisfied that everything is working properly from RAM, follow this procedure:

1. Re-compile or assemble your software to start at 0000h. Remember to reserve the first 34h bytes as your interrupt vectors.
2. Apply power to the board to get the Monitor utility prompt.
3. Set Offset mode to ON.
4. Upload your software hex file.
5. Execute the EEPROM copy command.
6. Remove Jumpers 3, 5 and 7 and change JP4 to 2-3.
7. Remove power.
8. Remove the AX562 EPROM from U5 and set it aside.
9. Remove the EEPROM from U6 and install it in the empty U5 socket.
10. Apply power.

Your program should start. You can add your own RAM or EEPROM to the U6 socket if you want.

QUICK TUTORIAL

As with any new development system there are several details you must learn to develop software and hardware on the CME-562. To help reduce this learning time, we highly recommend you complete the following brief tutorial. This will walk you thru the complete development cycle of a simple program. After completing this exercise and experimenting with your code in the different development modes, the remaining advanced software and hardware details should be easier to understand.

For this tutorial, we will use an assembly program on the CME-562 software disk called HELLO.ASM. This is a simple program that echoes a text string to a terminal on your PC serial port. You can substitute your own program here if you wish but, to verify everything is working properly, it's a good idea to start with something simple.

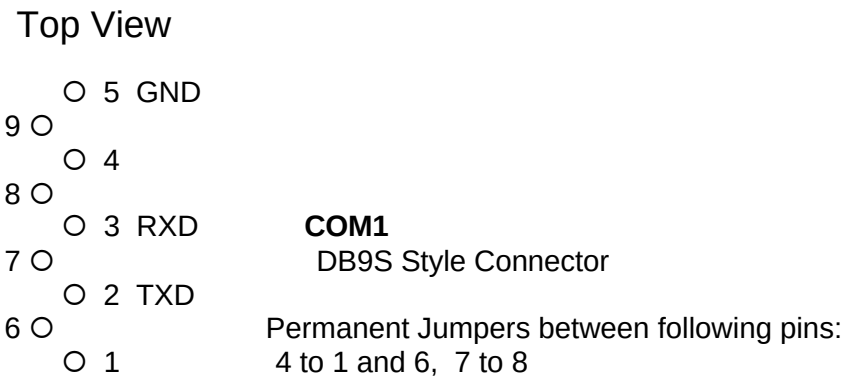
1. Make sure the board is configured in **DEVELOPMENT** mode as described earlier.
2. Connect the 9 pin serial cable from your PC to the CME-562 development board.
3. Connect the 9 volt power adapter to the barrel jack on the CME-562 development board.
4. At your PC's MSDOS command line prompt, change to your CME-562 software directory.
5. Execute the command: **`ASEM HELLO.ASM`** ↵
   This will assemble our test source code.
6. If any errors were found, they would be displayed on the screen, otherwise, you should have the files HELLO.LST (the source listing file) and HELLO.HEX (the Intel Hex object file).
7. Execute the command: **`AX52`** ↵
   This will launch the Axiom terminal program for the CME-562. The first time it is run you must select the PC serial port you are using. You can later change this value in the Options menu.
8. Select Terminal from the menu to see the terminal window.
9. Press then release the RESET button on the CME-562 board. You should see the Monitor command menu.
10. Press the Page Up key, then when prompted for a file name, type: **`HELLO.HEX`**
    then select [OK].
11. Make sure Echo is turned on (default) then select [OK].
12. Type: **`S 8000`** this sets the address pointer to the start of our program.
13. Type: **`G`** this executes our program.
14. If everything is working properly you should see the message ìHello Worldî echoed back to your terminal screen.
15. If you do not get this message, try going thru this tutorial once more, then if still no go, see the section titled **TROUBLESHOOTING**.
16. You can also test the AutoStart feature now by simply removing JP3 (also labeled AutoStart) and pressing RESET. The HELLO program should start first now then the Monitor. To remain in the HELLO program, we could have simply not returned from hello (an endless loop for example). To get the monitor back, reinstall JP3.

That's all there is to it. Try modifying HELLO to do other interesting things. You can also develop your own programs using this same procedure.

HARDWARE

SERIAL PORT COM1

COM1 is a simple, three wire asynchronous serial interface with hard wired Clear to Send (CTS) and Data Terminal Ready (DTR). It is driven by the 80C562 internal UART port using I/O pins P30 and P31. These two logic level signals are coupled thru a RS232 level shifter to the COM1 connector. The connector pin configuration provides direct connection to a compatible PC COM Port with a straight thru Male to Female 9 pin cable.



Note:

COM1 is setup to connect directly to a PC Com port with a straight thru type of cable.
*OPTREX is a registered trademark of OPTREX, INC.

Serial Communication Baud Rates

The Oscillator is 11.059MHz to provide high speed with standard baud rate selections. The typical application of the serial port requires communication with another device such as a PC. Standard configuration for this purpose is MODE1 with TIMER1 providing the baud clock. Following is an example of Register settings and values to generate common baud rates:

PCON register SMOD bit = 1, high speed operation for highest possible baud rate.

SCON register = $70hex = MODE 1 and Receiver ON, Use timer1 to generate baud clock.

TMOD register = $20hex = Timer1 is auto-reload mode to provide serial clock.
T1H register = baud rate select value = $FD hex - 19.2Kbaud
$FA hex - 9600 baud
$EE hex - 4800 baud
$DC hex - 2400 baud
$B8 hex - 1200 baud

SS : KEYPAD INTERFACE

The keypad interface can be accomplished with software drivers through the I/O ports of the 80C562. The SS (Simple Serial) configuration is used with a serially encoded keyboard available from the manufacturer. Use of the keypad port with a passive keypad is performed by using I/O port P40 - P43 as output column lines with P50 - P53 as input row lines. Note that P50 - P53 are terminated with 100K pull-down resistors on the board. The keypad should be installed with pin 8 of the keypad connector aligned with pin 10 of the SS:KEYPAD connector to avoid the Power pins. The Simple Serial interface is implemented by using P40 - 42 as serial I/O lines with P43 and P50 - P53 as peripheral select lines. Simple Serial peripherals are available from Axiom also.

SS : KEYPAD Connector

1	□	+5
2	○	GND
3	○	P40
4	○	P41
5	○	P42
6	○	P43
7	○	P50
8	○	P51
9	○	P52
10	○	P53

Note that SEL0-4 of Axiom SS boards map directly across to P43, P50 - P53.

LCD INTERFACE

The LCD interface is connected to the data buss and memory mapped to locations FFF0 and FFF1. It supports all OPTREX* DMC series displays up to 80 characters. Power, ground, and Vee are available at the LCD_PORT connector on the card. Vee contrast voltage is connected to ground thru R31 which is suitable for standard temperature range LCD modules. If contrast adjustment is necessary, R31 can be removed and a Vee source applied. Software operations to access the LCD should use DPTR MOVX instructions for proper timing delays.

LCD_PORT CONNECTOR

FUNCTION		PIN		FUNCTION
GND	1	□ ○	2	+5
-VEE	3	○ ○	4	A0
R/W	5	○ ○	6	LCDCS
D0	7	○ ○	8	D1
D2	9	○ ○	10	D3
D4	11	○ ○	12	D5
D6	13	○ ○	14	D7

MCU_PORT CONNECTOR

The MCU_PORT supports off-board input and output using the 80C562 I/O lines. Pin assignments are shown on page 1 of the schematic and are listed below for convenience.

FUNCTION	PIN		PIN	FUNCTION
GND	1	□	2	+5V
/INT0	3	○	4	/INT1
P30/RXD0	5	○	6	P31/TXD0
P34	7	○	8	P35
P16	9	○	10	P17
P14	11	○	12	P15
P12	13	○	14	P13
P10	15	○	16	P11
P46	17	○	18	P47
P44	19	○	20	P45
P42	21	○	22	P43
P40	23	○	24	P41
PWM0	25	○	26	PWM1
GND	27	○	28	STADC
P50/AI0	29	○	30	P51/AI1
P52/AI2	31	○	32	P53/AI3
P54/AI4	33	○	34	P55/AI5
P56/AI6	35	○	36	P57/AI7
VR+	37	○	38	VR-
+5V	39	○	40	GND

A/D REFERENCE

The VR+ and VR- lines from the 80562 are connected to +5v through R3 and to ground through R2 respectively. These two resistors are located between JP1 and the SS:KEYPAD connector on the component side of the circuit board. The resistors are identified on the silk screen by their reference designators. The appropriate resistor(s) need to be removed in order to apply an external reference to the VR+ and/or VR- inputs via the MCU_PORT connector.

EXPANSION BUS

The BUS_PORT supports off-board memory mapped devices. Power (+5V), ground, address lines, data lines, and control lines are brought out to this 40 pin connector. Pin assignments are shown on page 2 of the schematic and are listed below for convenience.

GND	1	□ ○	2	D3
D2	3	○ ○	4	D4
D1	5	○ ○	6	D5
D0	7	○ ○	8	D6
A0	9	○ ○	10	D7
A1	11	○ ○	12	A2
A10	13	○ ○	14	A3
/RD	15	○ ○	16	A4
A11	17	○ ○	18	A5
A9	19	○ ○	20	A6
A8	21	○ ○	22	A7
A12	23	○ ○	24	A13
/WR	25	○ ○	26	/CS0
/CS1	27	○ ○	28	/CS2
/CS3	29	○ ○	30	/CS4
/CS5	31	○ ○	32	/INT0
+5	33	○ ○	34	/M3
/PRD	35	○ ○	36	/CS6
E	37	○ ○	38	/CS7
GND	39	○ ○	40	/RESET

SIGNAL DEFINITIONS

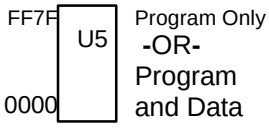
D0 - D7:	Data BUS
A0 - A13:	Address BUS
/CS0 - /CS7:	External Chip Selects - located at $FF80 - $FFFF hex. 16 bytes each, active low.
/WR:	Write data control (MCU output data on BUS)
/RD:	Read data control (MCU input data on BUS)
/PRD:	Read program or data control (MCU input)
E:	Data valid signal (Motorola emulation)
/INT0:	Interrupt input
/RESET:	Reset output or input
/M3:	Memory select 3, same as U7 select (use caution when using /M3 for external memory access)

MEMORY and ADDRESSING

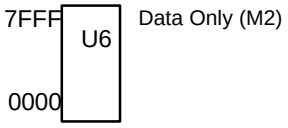
The 80562 has separate address spaces for program and data memory. The program memory can be up to 64k bytes long. Additionally, the 80C562 can address up to 64k bytes of external data memory. The three memory sockets provide can be configured in two basic memory maps. Default memory map is U5 up to 32K Byte Program, U6 up to 32K Byte Data space, and U7 up to 32K Program or Data space. Optional memory maps provide U5 with 64K Byte Program space with 32K Data Space in U6 and 32K Data Space in U7.

Memory configuration depends on the type/size memory devices installed on the board (U5 - U7) and jumper settings. Below is the addresses and memory types of the chips available:

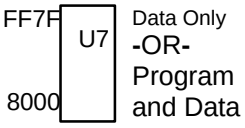
EPROM / EPROM



RAM / EPROM



RAM / EPROM



MEMORY SELECTION JUMPERS

	Open	Closed	Notes
JP1	External Program Memory	Internal Program Memory	Only installed if OTP is in U1.
JP2	Disable Watchdog	Enable Watchdog	
JP3	AutoStart On	AutoStart Off	Use with Monitor or Basic
JP4	32K EPROM in U5	64K EPROM in U5	Used in conjunction with JP11.
JP7	Write Protect U6	U6 Write Enabled	
JP9	Write Protect U7	U7 Write Enabled	
JP11	32K EPROM in U5	64K EPROM in U5	Used in conjunction with JP4
JP12	8K RAM in U7	32K RAM in U7	

	1 - 2	2 - 3	
JP5	64K EPROM in U5	Monitor Enable for U5	Open is Basic Enable for U5
JP6	Program and Data in U5, U6 not installed or JP8 = 1-2	Program Only in U5, U6 is data memory	JP6 and JP8 must be in the same positions if U6 is installed.
JP8	U6 not in memory map	U6 is data memory	
JP10	Program and Data in U7	Data only in U7	Open if U5 is 64K program and data memory

TROUBLESHOOTING

The board is fully tested and operational before shipping. If it fails to function properly, inspect the board for obvious physical damage first. Ensure that all socketed IC devices are properly seated in the their sockets.

The most common problems are improperly configured communications parameters, and attempting to use the wrong COM port (on the PC AND on the development board). Verify that your communications port is working by substituting a known good serial device, or by doing a loop back diagnostic.

Re-check ALL the hardware configuration jumpers. Verify the power source. If no voltage is found, verify wall plug connections to 115VAC outlet and power connector. Disconnect all external connections to the board except for COM1 to the PC and the wall plug. Follow these steps in the order given below:

1. Visual Inspection
2. Verify that all jumpers are properly installed
3. Verify power by checking for +9 volts between GND and +9V test point pads.
4. Verify the presence of +5 volts between GND test point and pin 1 of the SS : KEYBOARD connector.
5. Verify U5 EPROM for proper installation (no bent pins) and proper jumper settings for the device used.
6. Re-Check the serial communications parameters.
7. Disconnect any peripheral devices including display, and keyboard.
8. Make sure that the RESET line is not being held low. Check for this by measuring across S1.
9. Please check off these steps and any others you may have performed before calling so we can better help you.

SOFTWARE

The CME-562 package comes with all the software you need to develop and run your own embedded applications. Included on the software disk FREE is a copy of ASEM-51, a very powerful assembler and bootloader by W.W. Heinz, a simple 8051 MONITOR program and a version of Tiny Basic 51. Software versions supplied by Axiom may have been modified to improve support for our product. You may also download the latest versions of original software from the Internet.

ASEM-51 will produce Intel HEX records from standard 8051 assembly code. The HEX records can then be uploaded to the CME-562 development board using the BOOT51 bootloader or the 8051 MONITOR program and executed.

Before writing software for the CME-562, you should be familiar with the Philips 8051 instruction set and software methods. For your convenience, the supported software instructions are listed below. For further information on writing software for the 8051 Microcontroller, see the 8051 DATA HANDBOOK or similar reference book from the manufacturer.

8051 Monitor Program Notes

- Communication is 9600,n,8,1
- Monitor operates via polling the serial port. Programs will not run while you are accessing the monitor.
- Incoming hex digits get shifted into MPARAM. Any valid non-hex digit is interpreted as a command. The following commands are implemented here:
- To scan through memory, type the address and hold down the L or R key for auto-repeat. MPARAM is automatically offset after each line to give a continuous list.
- MPARAM is also modified by L. After the L command, the next address is already in MPARAM. To modify a range of registers, type the 8-bit address, data0, L, data1, L, data2, ...
- To modify external memory, use the : (upload) command. The format is
  `:nnAAAA00ddddddddddd...`

  where nn is a byte count, AAAA is the address and ddd... is a block of data. Entering any non-hex digit aborts the upload.
- The G (run) command calls a program at MPARAM. To reset the monitor, type 0000G. Loading a program into an area that already contains a running program can cause a crash. When in doubt, reset before you upload.
- The Auto-Start Jumper will allow you to execute your code starting at $8000 hex from RESET. The monitor will LCALL your RESET vector installed at $8000 hex. If you place a return in you code for this lcall, the monitor will restart and enable you to check registers for debugging.

MONITOR COMMANDS

R	dump eight registers, starting at MPARAM.
S	store low byte of MPARAM in register (hi byte of) MPARAM.
L	dump 16 external memory locations, starting at MPARAM.
G	call program at location MPARAM
M	dump MPARAM
:	accept incoming hex file data (should be .ORG at $8000 hex)
P	copy 8K RAM contents to 8K EEPROM (U7 to U6)
X	copy 32K RAM contents to 32K EEPROM (U7 to U6)
O	Offset hex mode. Stores incoming hex file data at address + 8000h (Use prior to P or X command. Hex file should be .ORG at $0000 hex)
H	Help menu

8051 Tiny Basic Notes

The Tiny Basic provided has been modified to store programs in U7 from $8000 hex up to $FF7F hex (almost 32K). The interrupt vectors are relocated beginning at $8000 hex with a fixed offset of $8000 from the normal location in ROM Page 0. After Basic development, an autostart may be performed by removing the AutoStart Jumper (INT1). Programs residing in U7 may be permanently saved in EPROM by placing an EEPROM in U6 and performing the Basic ESAVE8 or ESAVE32 command. See the Tiny Basic manual on the Disk for more information on Tiny Basic.

80562 Instruction Set Summary

`A`	Accumulator
`Rn`	Register R7-R0 of the currently selected Register Bank.
`Direct`	8-bit internal data location's address. This could be an internal data RAM location (0-127) or a SFR (128-255).
`@Ri`	8-bit internal data RAM location(0-255) addressed indirectly through register R1 or R0.
`#data`	8-bit constant included in the instruction.
`#data16`	16-bit constant included in the instruction.
`Addr16`	16-bit destination address. A branch can be anywhere within the 64k-byte Program Memory address space.
`Addr11`	11-bit destination address. The branch must be within the same 2k-byte page of program memory as the first byte of the following instruction.
`Rel`	signed 8-bit offset byte. Range is -128 to +127 bytes relative to first byte of the following instruction.
`Bit`	Direct Addressed bit in Internal Data RAM or Special Function Register.

ACALL	Addr11	absolute subroutine call
ADD	A,Rn	add register to A
	A,direct	add direct byte to A
	A,@Ri	add indirect RAM to A
	A,#data	add immediate data to A
ADDC	A,Rn	add register to A with carry
	A,direct	add direct byte to A with carry
	A,@Ri	add indirect RAM to A with carry
	A,#data	add immediate data to A with carry
AJMP	addr11	absolute jump
ANL	A,Rn	AND register to A
	A,@Ri	AND indirect RAM to A
	A,#data	AND immediate data to A
	A,direct	AND direct byte to A
	direct,A	AND A to direct byte
	direct,#data	AND immediate data to direct byte
	C,bit	AND direct bit to carry
	C,/bit	AND complement of direct bit to carry
CJNE	A,direct,rel	compare direct byte to A and jump if not equal
	A,#data,rel	compare immediate to A and jump if not equal
	Rn,#data,rel	compare immediate to register and jump if not equal
	@Ri,#data,rel	compare immediate to indirect and jump if not equal
CLR	A	clear A
	C	clear carry
	bit	clear direct bit
CPL	A	complement A
	C	complement carry
	bit	complement direct bit
DA	A	decimal adjust A
DEC	A	decrement A
	Rn	decrement Register
	direct	decrement direct byte
	@Ri	decrement indirect RAM
DIV	AB	divide A by B
DJNZ	Rn,rel	decrement register and jump if not zero
	direct,rel	decrement direct byte and jump if not zero
INC	A	increment A
	Rn	increment register
	direct	increment direct byte
	@Ri	increment indirect RAM
	DPTR	increment Data Pointer
JB	rel	jump if direct bit is set
JBC	bit,rel	jump if direct bit is set and clear bit
JC	rel	jump if carry is set
JMP	@A+DPTR	jump indirect relative to the DPTR
JNB	rel	jump if direct bit is not set
JNC	rel	jump if carry not set
JNZ	rel	jump if A is not zero
JZ	rel	jump if A is zero
LCALL	addr16	long subroutine call
LJMP	addr16	long jump
MOV	A,Rn	move register to A
	A,direct	move direct byte to A
	A,@Ri	move indirect RAM to A

	A,#data	move immediate data to A
	Rn,A	move A to register
	Rn,direct	move direct byte to register
	Rn,#data	move immediate data to register
	direct,A	move A to direct byte
	direct,Rn	move register to direct byte
	direct,direct	move direct byte to direct byte
	@Ri,A	move A to indirect RAM
	@Ri,direct	move direct byte to indirect RAM
	@Ri,#data	move immediate data to indirect RAM
	DPTR,#data16	load Data Pointer with a 16-bit constant
	C,bit	move direct bit to carry
	bit,C	move carry to direct bit
MOVC	A,@A+DPTR	move code byte relative to DPTR to A
	A,@A+PC	move code byte relative to PC to A
MOVX	A,@Ri	move external RAM (8-bit addr) to A
	A,@DPTR	move external RAM (16-bit addr) to A
	A,@Ri,A	movA to external RAM (8-bit addr)
	@DPTR,A	move A to external RAM (16-bit addr)
MUL	AB	multiply A and B
NOP		no operation
ORL	A,Rn	OR register to A
	A,@Ri	OR indirect RAM to A
	A,#data	OR immediate data to A
	A,direct	OR direct byte to A
	direct,A	OR A to direct byte
	direct,#data	OR immediate data to direct byte
	C,bit	OR direct bit to carry
	C,/bit	OR complement of direct bit to carry
POP	direct	pop direct byte from stack
PUSH	direct	push direct byte onto stack
RET		return from subroutine
RETI		return from interrupt
RL	A	rotate A left
RLC	A	rotate A left through carry
RR	A	rotate A right
RRC	A	rotate A right through carry
SETB	C	set carry
	bit	set direct bit
SJMP	rel	short jump (relative addr)
SUBB	A,Rn	subtract register from A with borrow
	A,direct	subtract direct byte from A with borrow
	A,@Ri	subtract indirect RAM from A with borrow
	A,#data	subtract immediate data from A with borrow
SWAP	A	swap nibbles in A
XCH	A,Rn	exchange register with A
	A,direct	exchange direct byte with A
	A,@Ri	exchange indirect RAM with A
XCHD	A,@Ri	exchange low-order digit indirect RAM with A
XRL	A,Rn	Exclusive-OR register to A
	A,@Ri	Exclusive-OR indirect RAM to A
	A,#data	Exclusive-OR immediate data to A
	A,direct	Exclusive-OR direct byte to A