BAB 6

KESIMPULAN DAN SARAN

6.1. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan, maka kesimpulan dari penelitian ini adalah dengan implementasi teknologi mikrokontroler *AT89S51* dihasilkan model kunci elektronik yang dapat membatasi akses pengguna ke dalam ruangan. Kunci elektronik yang dihasilkan memiliki tingkat keamanan, kemudahan dan kenyaman yang lebih baik dari kunci konvensional. Kunci elektronik juga dilengkapi dengan *display LCD* dan sirine.

6.2. Saran

Dari penelitian yang telah dilakukan, ada beberapa saran yang dapat diberikan penulis untuk lebih menyempurnakan hasil penelitian ini di masa mendatang. Adapun saran penulis sebagai berikut:

1. Sumber tegangan listrik kunci elektronik dapat menggunakan baterai basah agar tidak mudah di-*reset*.
2. Keamanan kunci elektronik dapat ditingkatkan dengan menambah jumlah kode pin.

DAFTAR PUSTAKA

Budiharto, W., 2005, *Perancangan Sistem dan Aplikasi Mikrokontroler*, PT. Elex Media Komputindo, Jakarta.

Budioko, T., 2005, *Belajar dengan mudah dan cepat Pemograman Bahasa C dengan SDCC (Small Device C Compiler) pada Mikrokontroler AT89X051/AT89C51/52 Teori, Simulasi dan Aplikasi*, ed. 1, cet. 1, Gava Media, Yogyakarta.

Fransiska, H., 2006, *Implementasi Mikrokontroler AT89S51 pada Sistem Parkir*, Skripsi di Program Studi Teknik Industri, Fakultas Teknologi Industri Universitas Atma Jaya, Yogyakarta.

Groover, M., 1987, *Automation, Production Systems, And Computer-Integrated Manufacturing*, Prentice-Hall International, Inc., Englewood Cliffs, New Jersey., U.S.A.

Gunawan, M. U., 2004, *Sistem Pengontrol Lampu dan Pendeteksi Orang Dalam Ruangan Berbasis Mikrokontroelr AT89C51*, Skripsi di Jurusan Elektronika Peminatan Elektronika Industri, Fakultas Teknik, Universitas Kristen Krida Wacana, Jakarta.

Halim, R., 2005, *Pengendalian Peralatan Listrik dengan Infra-Red Remote Control Berbasis Mikrokontroler AT89C51*, Skripsi di Program Studi Teknik Industri, Fakultas Teknologi Industri Universitas Atma Jaya, Yogyakarta.

Hartono, J., 2003, *Konsep Dasar Pemograman Bahasa C*, ed. 3, ANDI, Yogyakarta.

Nalwan, P.A., 2003, *Panduan Praktis Teknik Antar Muka dan Pemograman Mikrokontroler AT89C51*, cet. 2, PT.Elex Media Komputindo, Jakarta.

LAMPIRAN

Lampiran 1 :

Data Sheet Mikrokontroler *AT89S51*

'eatures

- Compatible with MCS®-51 Products
- 4K Bytes of In-System Programmable (ISP) Flash Memory
  – Endurance: 1000 Write/Erase Cycles
- 4.0V to 5.5V Operating Range
- Fully Static Operation: 0 Hz to 33 MHz
- Three-level Program Memory Lock
- 128 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-bit Timer/Counters
- Six Interrupt Sources
- Full Duplex UART Serial Channel
- Low-power Idle and Power-down Modes
- Interrupt Recovery from Power-down Mode
- Watchdog Timer
- Dual Data Pointer
- Power-off Flag
- Fast Programming Time
- Flexible ISP Programming (Byte and Page Mode)
- Green (Pb/Halide-free) Packaging Option

8-bit Microcontroller with 4K Bytes In-System Programmable Flash

AT89S51

. Description

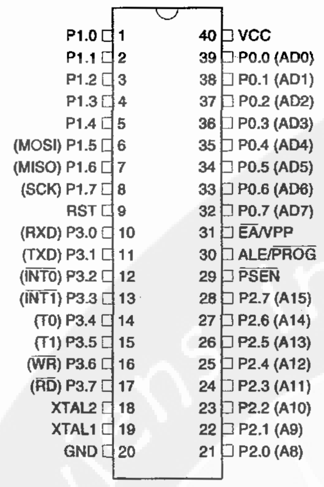
he AT89S51 is a low-power, high-performance CMOS 8-bit microcontroller with 4K ytes of In-System Programmable Flash memory. The device is manufactured using tmel's high-density nonvolatile memory technology and is compatible with the indus-y-standard 80C51 instruction set and pinout. The on-chip Flash allows the program ıemory to be reprogrammed in-system or by a conventional nonvolatile memory pro-rammer. By combining a versatile 8-bit CPU with In-System Programmable Flash on monolithic chip, the Atmel AT89S51 is a powerful microcontroller which provides a ighly-flexible and cost-effective solution to many embedded control applications.

he AT89S51 provides the following standard features: 4K bytes of Flash, 128 bytes f RAM, 32 I/O lines, Watchdog timer, two data pointers, two 16-bit timer/counters, a ve-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, nd clock circuitry. In addition, the AT89S51 is designed with static logic for operation own to zero frequency and supports two software selectable power saving modes. he Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and ıterrupt system to continue functioning. The Power-down mode saves the RAM con-ents but freezes the oscillator, disabling all other chip functions until the next external ıterrupt or hardware reset.

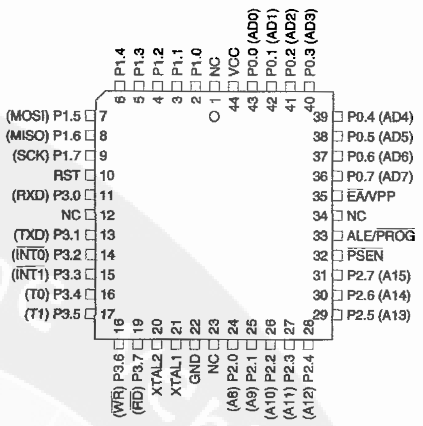
2487C–MICRO–03/05

. Pin Configurations

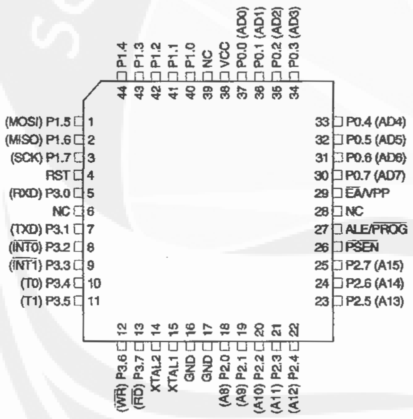
.1 40-lead PDIP



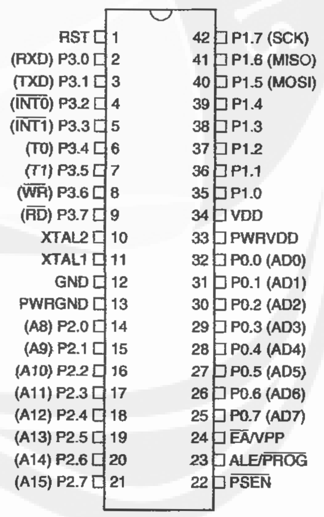
2.3 44-lead PLCC



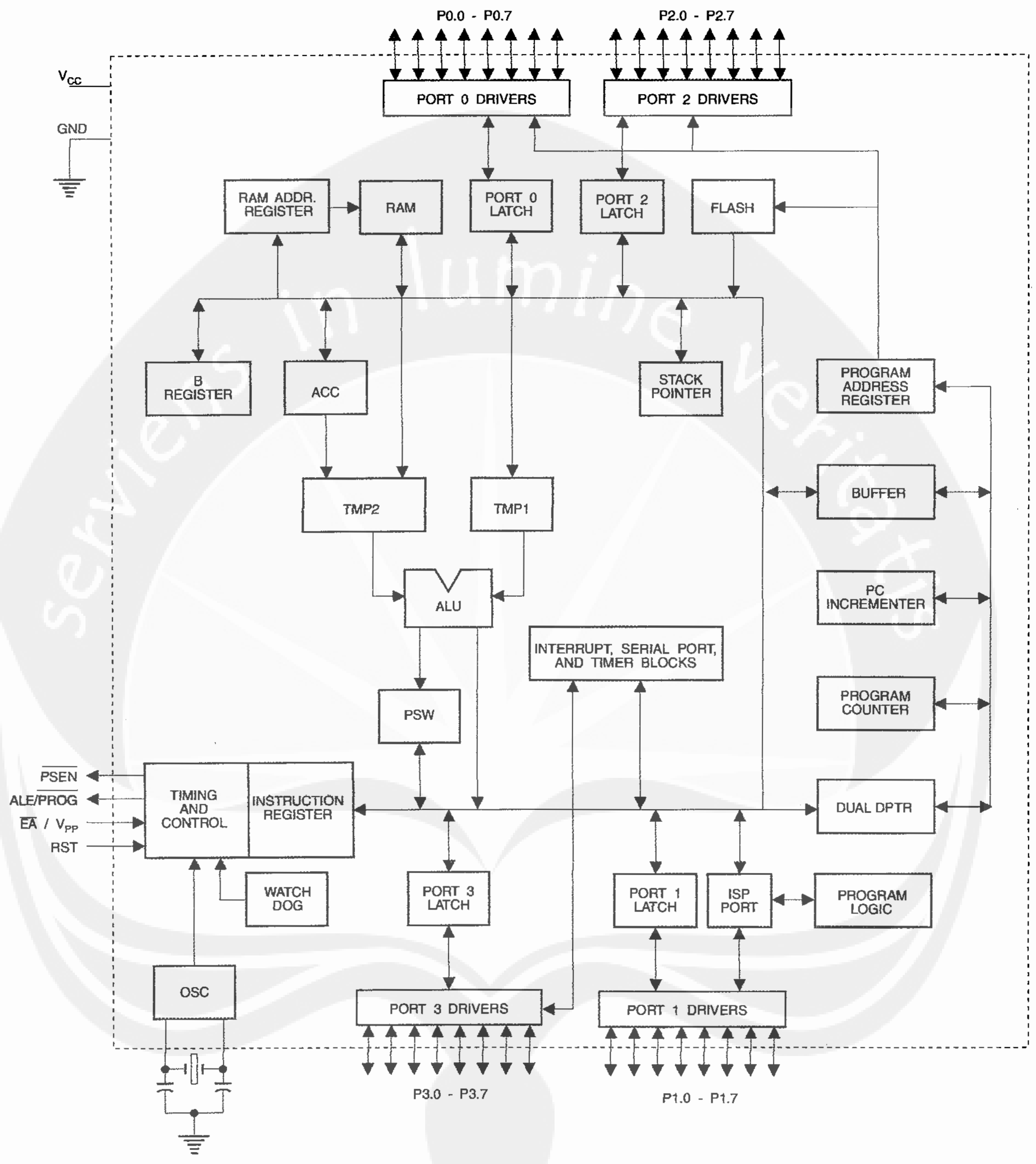
.2 44-lead TQFP



2.4 42-lead PDIP



2. Block Diagram



. Pin Description

.1 VCC

Supply voltage (all packages except 42-PDIP).

.2 GND

Ground (all packages except 42-PDIP; for 42-PDIP GND connects only the logic core and the embedded program memory).

.3 VDD

Supply voltage for the 42-PDIP which connects only the logic core and the embedded program memory.

.4 PWRVDD

Supply voltage for the 42-PDIP which connects only the I/O Pad Drivers. The application board **MUST** connect both VDD and PWRVDD to the board supply voltage.

.5 PWRGND

Ground for the 42-PDIP which connects only the I/O Pad Drivers. PWRGND and GND are weakly connected through the common silicon substrate, but not through any metal link. The application board **MUST** connect both GND and PWRGND to the board ground.

.6 Port 0

Port 0 is an 8-bit open drain bi-directional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pull-ups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. **External pull-ups are required during program verification.**

.7 Port 1

Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pull-ups.

Port 1 also receives the low-order address bytes during Flash programming and verification.

Port Pin	Alternate Functions
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

.8 Port 2

Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pull-ups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

.9 Port 3

Port 3 is an 8-bit bi-directional I/O port with internal pull-ups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pull-ups.

Port 3 receives some control signals for Flash programming and verification.

Port 3 also serves the functions of various special features of the AT89S51, as shown in the following table.

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interrupt 0)
P3.3	$\overline{\text{INT1}}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

.10 RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. This pin drives High for 98 oscillator periods after the Watchdog times out. The DISRTO bit in SFR AUXR (address 8EH) can be used to disable this feature. In the default state of bit DISRTO, the RESET HIGH out feature is enabled.

.11 ALE/$\overline{\text{PROG}}$

Address Latch Enable (ALE) is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{\text{PROG}}$) during Flash programming.

In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

.12 $\overline{\text{PSEN}}$

Program Store Enable ($\overline{\text{PSEN}}$) is the read strobe to external program memory.

When the AT89S51 is executing code from external program memory, $\overline{\text{PSEN}}$ is activated twice each machine cycle, except that two $\overline{\text{PSEN}}$ activations are skipped during each access to external data memory.

.13 $\overline{\text{EA}}$/VPP

External Access Enable. $\overline{\text{EA}}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{\text{EA}}$ will be internally latched on reset.

$\overline{\text{EA}}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming.

.14 XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

.15 XTAL2

Output from the inverting oscillator amplifier

. Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 5-1.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

able 5-1. AT89S51 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000								0D7H
0C8H									0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000								0AFH
0A0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX		0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0		8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000		PCON 0XXX0000	87H

ser software should not write 1s to these unlisted locations, since they may be used in future products to invoke new fea-ıres. In that case, the reset or inactive values of the new bits will always be 0.

ıterrupt Registers: The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the five ıterrupt sources in the IP register.

Table 5-2. AUXR: Auxiliary Register

AUXR	Address = 8EH							Reset Value = XXX00XX0B
Not Bit Addressable								
	–	–	–	WDIDLE	DISRTO	–	–	DISALE
Bit	7	6	5	4	3	2	1	0

–	Reserved for future expansion	
DISALE	Disable/Enable ALE	
	DISALE Operating Mode	
	0	ALE is emitted at a constant rate of 1/6 the oscillator frequency
	1	ALE is active only during a MOVX or MOVC instruction
DISRTO	Disable/Enable Reset-out	
	DISRTO	
	0	Reset pin is driven High after WDT times out
	1	Reset pin is input only
WDIDLE	Disable/Enable WDT in IDLE mode	
WDIDLE		
0	WDT continues to count in IDLE mode	
1	WDT halts counting in IDLE mode	

Dual Data Pointer Registers: To facilitate accessing both internal and external data memory, two banks of 16-bit Data Pointer Registers are provided: DP0 at SFR address locations 82H-83H and DP1 at 84H-85H. Bit DPS = 0 in SFR AUXR1 selects DP0 and DPS = 1 selects DP1. The user should **ALWAYS** initialize the DPS bit to the appropriate value before accessing the respective Data Pointer Register.

Power Off Flag: The Power Off Flag (POF) is located at bit 4 (PCON.4) in the PCON SFR. POF is set to “1” during power up. It can be set and rest under software control and is not affected by reset.

Table 5-3. AUXR1: Auxiliary Register 1

AUXR1 Address = A2H Reset Value = XXXXXXX0B

Not Bit Addressable

	–	–	–	–	–	–	–	DPS
Bit	7	6	5	4	3	2	1	0

–	Reserved for future expansion	
DPS	Data Pointer Register Select	
	DPS	
	0	Selects DPTR Registers DP0L, DP0H
	1	Selects DPTR Registers DP1L, DP1H

. Memory Organization

MCS-51 devices have a separate address space for Program and Data Memory. Up to 64K bytes each of external Program and Data Memory can be addressed.

.1 Program Memory

If the $\overline{EA}$ pin is connected to GND, all program fetches are directed to external memory.

On the AT89S51, if $\overline{EA}$ is connected to V_{CC}, program fetches to addresses 0000H through FFFH are directed to internal memory and fetches to addresses 1000H through FFFFH are directed to external memory.

.2 Data Memory

The AT89S51 implements 128 bytes of on-chip RAM. The 128 bytes are accessible via direct and indirect addressing modes. Stack operations are examples of indirect addressing, so the 128 bytes of data RAM are available as stack space.

. Watchdog Timer (One-time Enabled with Reset-out)

The WDT is intended as a recovery method in situations where the CPU may be subjected to software upsets. The WDT consists of a 14-bit counter and the Watchdog Timer Reset (WDTRST) SFR. The WDT is defaulted to disable from exiting reset. To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, it will increment every machine cycle while the oscillator is running. The WDT timeout period is dependent on the external clock frequency. There is no way to disable the WDT except through reset (either hardware reset or WDT overflow reset). When WDT overflows, it will drive an output RESET HIGH pulse at the RST pin.

.1 Using the WDT

To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, the user needs to service it by writing 01EH and 0E1H to WDTRST to avoid a WDT overflow. The 14-bit counter overflows when it reaches 16383 (3FFFH), and this will reset the device. When the WDT is enabled, it will increment every machine cycle while the oscillator is running. This means the user must reset the WDT at least every 16383 machine cycles. To reset the WDT the user must write 01EH and 0E1H to WDTRST. WDTRST is a write-only register. The WDT counter cannot be read or written. When WDT overflows, it will generate an output RESET pulse at the RST pin. The RESET pulse duration is 98xTOSC, where TOSC = 1/FOSC. To make the best use of the WDT, it should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset.

.2 WDT During Power-down and Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode, the user does not need to service the WDT. There are two methods of exiting Power-down mode: by a hardware reset or via a level-activated external interrupt, which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, servicing the WDT should occur as it normally does whenever the AT89S51 is reset. Exiting Power-down with an interrupt is significantly different. The interrupt is held low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service for the interrupt used to exit Power-down mode.

To ensure that the WDT does not overflow within a few states of exiting Power-down, it is best to reset the WDT just before entering Power-down mode.

Before going into the IDLE mode, the WDIDLE bit in SFR AUXR is used to determine whether the WDT continues to count if enabled. The WDT keeps counting during IDLE (WDIDLE bit = 0) as the default state. To prevent the WDT from resetting the AT89S51 while in IDLE mode, the user should always set up a timer that will periodically exit IDLE, service the WDT, and reenter IDLE mode.

With WDIDLE bit enabled, the WDT will stop to count in IDLE mode and resumes the count upon exit from IDLE.

. UART

The UART in the AT89S51 operates the same way as the UART in the AT89C51. For further information on the UART operation, please click on the document link below:

http://www.atmel.com/dyn/resources/prod_documents/DOC4316.PDF

. Timer 0 and 1

Timer 0 and Timer 1 in the AT89S51 operate the same way as Timer 0 and Timer 1 in the AT89C51. For further information on the timers' operation, please click on the document link below:

http://www.atmel.com/dyn/resources/prod_documents/DOC4316.PDF

0. Interrupts

The AT89S51 has a total of five interrupt vectors: two external interrupts ($\overline{\text{INT0}}$ and $\overline{\text{INT1}}$), two timer interrupts (Timers 0 and 1), and the serial port interrupt. These interrupts are all shown in Figure 10-1.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE. IE also contains a global disable bit, EA, which disables all interrupts at once.

Note that Table 10-1 shows that bit positions IE.6 and IE.5 are unimplemented. User software should not write 1s to these bit positions, since they may be used in future AT89 products.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle.

Table 10-1. Interrupt Enable (IE) Register

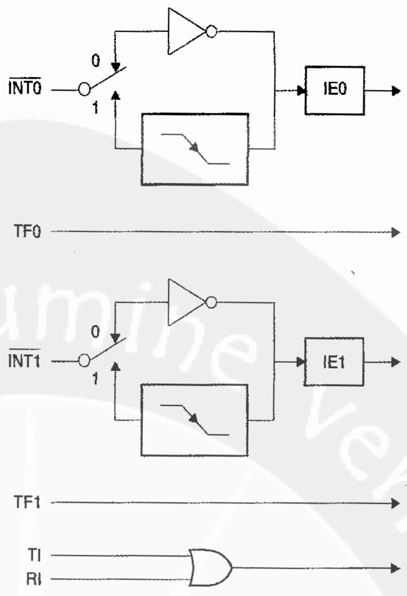
(MSB)							(LSB)
EA	–	–	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved
–	IE.5	Reserved
ES	IE.4	Serial Port interrupt enable bit
ET1	IE.3	Timer 1 interrupt enable bit
EX1	IE.2	External interrupt 1 enable bit
ET0	IE.1	Timer 0 interrupt enable bit
EX0	IE.0	External interrupt 0 enable bit
User software should never write 1s to reserved bits, because they may be used in future AT89 products.		

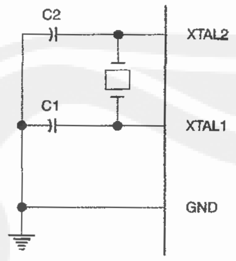
Figure 10-1. Interrupt Sources



1. Oscillator Characteristics

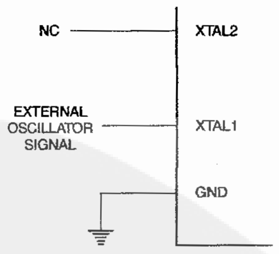
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 11-1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 11-2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Figure 11-1. Oscillator Connections



Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 11-2. External Clock Drive Configuration



2. Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special function registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

Note that when idle mode is terminated by a hardware reset, the device normally resumes program execution from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when idle mode is terminated by a reset, the instruction following the one that invokes idle mode should not write to a port pin or to external memory.

3. Power-down Mode

In the Power-down mode, the oscillator is stopped, and the instruction that invokes Power-down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the Power-down mode is terminated. Exit from Power-down mode can be initiated either by a hardware reset or by activation of an enabled external interrupt ($\overline{INT0}$ or $\overline{INT1}$). Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Table 13-1. Status of External Pins During Idle and Power-down Modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

4. Program Memory Lock Bits

The AT89S51 has three lock bits that can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in Table 14-1.

Table 14-1. Lock Bit Protection Modes

	Program Lock Bits			
	LB1	**LB2**	**LB3**	**Protection Type**
1	U	U	U	No program lock features
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the Flash memory is disabled
3	P	P	U	Same as mode 2, but verify is also disabled
4	P	P	P	Same as mode 3, but external execution is also disabled

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value and holds that value until reset is activated. The latched value of $\overline{EA}$ must agree with the current logic level at that pin in order for the device to function properly.

5. Programming the Flash – Parallel Mode

The AT89S51 is shipped with the on-chip Flash memory array ready to be programmed. The programming interface needs a high-voltage (12-volt) program enable signal and is compatible with conventional third-party Flash or EPROM programmers.

The AT89S51 code memory array is programmed byte-by-byte.

Programming Algorithm: Before programming the AT89S51, the address, data, and control signals should be set up according to the Flash Programming Modes table (Table 17-1) and Figure 17-1 and Figure 17-2. To program the AT89S51, take the following steps:

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}/V_{PP}$ to 12V.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 50 μs. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

$\overline{\text{Data}}$ Polling: The AT89S51 features $\overline{\text{Data}}$ Polling to indicate the end of a byte write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P0.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. $\overline{\text{Data}}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{\text{Busy}}$: The progress of byte programming can also be monitored by the RDY/$\overline{BSY}$ output signal. P3.0 is pulled low after ALE goes high during programming to indicate $\overline{BUSY}$. P3.0 is pulled high again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. **The status of the individual lock bits can be verified directly by reading them back.**

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 000H, 100H, and 200H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(000H) = 1EH indicates manufactured by Atmel
(100H) = 51H indicates AT89S51
(200H) = 06H

Chip Erase: In the parallel programming mode, a chip erase operation is initiated by using the proper combination of control signals and by pulsing ALE/$\overline{\text{PROG}}$ low for a duration of 200 ns - 500 ns.

In the serial programming mode, a chip erase operation is initiated by issuing the Chip Erase instruction. In this mode, chip erase is self-timed and takes about 500 ms.

During chip erase, a serial read from any address location will return 00H at the data output.

6. Programming the Flash – Serial Mode

The Code memory array can be programmed using the serial ISP interface while RST is pulled to V_{CC}. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After RST is set high, the Programming Enable instruction needs to be executed first before other operations can be executed. Before a reprogramming sequence can occur, a Chip Erase operation is required.

The Chip Erase operation turns the content of every memory location in the Code array into FFH.

Either an external system clock can be supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2. The maximum serial clock (SCK) frequency should be less than 1/16 of the crystal frequency. With a 33 MHz oscillator clock, the maximum SCK frequency is 2 MHz.

6.1 Serial Programming Algorithm

To program and verify the AT89S51 in the serial programming mode, the following sequence is recommended:

1. Power-up sequence:
   a. Apply power between VCC and GND pins.
   b. Set RST pin to "H".

If a crystal is not connected across pins XTAL1 and XTAL2, apply a 3 MHz to 33 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.

2. Enable serial programming by sending the Programming Enable serial instruction to pin MOSI/P1.5. The frequency of the shift clock supplied at pin SCK/P1.7 needs to be less than the CPU clock at XTAL1 divided by 16.
3. The Code array is programmed one byte at a time in either the Byte or Page mode. The write cycle is self-timed and typically takes less than 0.5 ms at 5V.
4. Any memory location can be verified by using the Read instruction that returns the content at the selected address at serial output MISO/P1.6.

5. At the end of a programming session, RST can be set low to commence normal device operation.

Power-off sequence (if needed):

1. Set XTAL1 to "L" (if a crystal is not used).
2. Set RST to "L".
3. Turn V_{CC} power off.

Data Polling: The $\overline{\text{Data}}$ Polling feature is also available in the serial mode. In this mode, during a write cycle an attempted read of the last byte written will result in the complement of the MSB of the serial output byte on MISO.

6.2 Serial Programming Instruction Set

The Instruction Set for Serial Programming follows a 4-byte protocol and is shown in the "Serial Programming Instruction Set" on page 20.

7. Programming Interface – Parallel Mode

Every code byte in the Flash array can be programmed by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

Most major worldwide programming vendors offer worldwide support for the Atmel AT89 microcontroller series. Please contact your local programming vendor for the appropriate software revision.

ıble 17-1. Flash Programming Modes

Mode	V_{CC}	RST	$\overline{\text{PSEN}}$	ALE/ $\overline{\text{PROG}}$	$\overline{\text{EA}}$/ V_{PP}	P2.6	P2.7	P3.3	P3.6	P3.7	P0.7-0 Data	P2.3-0	P1.7-0
												Address	
Vrite Code Data	5V	H	L	⎺╲_╱⎺ (2)	12V	L	H	H	H	H	D_{IN}	A11-8	A7-0
Read Code Data	5V	H	L	H	H	L	L	L	H	H	D_{OUT}	A11-8	A7-0
Vrite Lock Bit 1	5V	H	L	⎺╲_╱⎺ (3)	12V	H	H	H	H	H	X	X	X
Vrite Lock Bit 2	5V	H	L	⎺╲_╱⎺ (3)	12V	H	H	H	L	L	X	X	X
Vrite Lock Bit 3	5V	H	L	⎺╲_╱⎺ (3)	12V	H	L	H	H	L	X	X	X
Read Lock Bits 1, 2, 3	5V	H	L	H	H	H	H	L	H	L	P0.2, P0.3, P0.4	X	X
Chip Erase	5V	H	L	⎺╲_╱⎺ (1)	12V	H	L	H	L	L	X	X	X
Read Atmel ID	5V	H	L	H	H	L	L	L	L	L	1EH	0000	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	51H	0001	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	06H	0010	00H

otes:
1. Each $\overline{\text{PROG}}$ pulse is 200 ns - 500 ns for Chip Erase.
2. Each $\overline{\text{PROG}}$ pulse is 200 ns - 500 ns for Write Code Data.
3. Each $\overline{\text{PROG}}$ pulse is 200 ns - 500 ns for Write Lock Bits.
4. RDY/$\overline{\text{BSY}}$ signal is output on P3.0 during programming.
5. X = don't care.

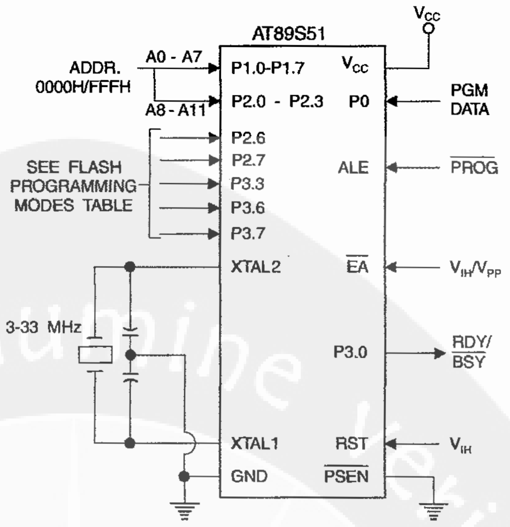


Figure 17-1. Programming the Flash Memory (Parallel Mode)

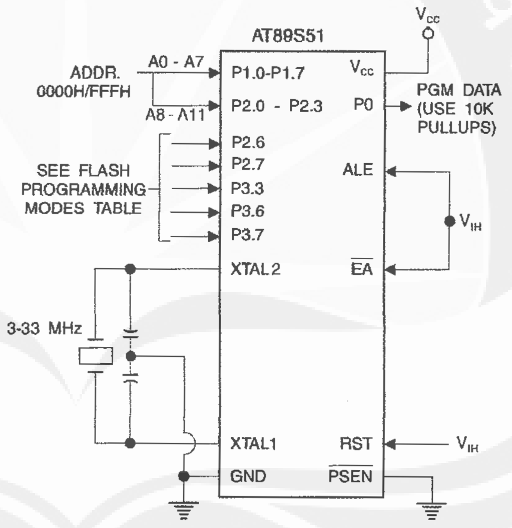


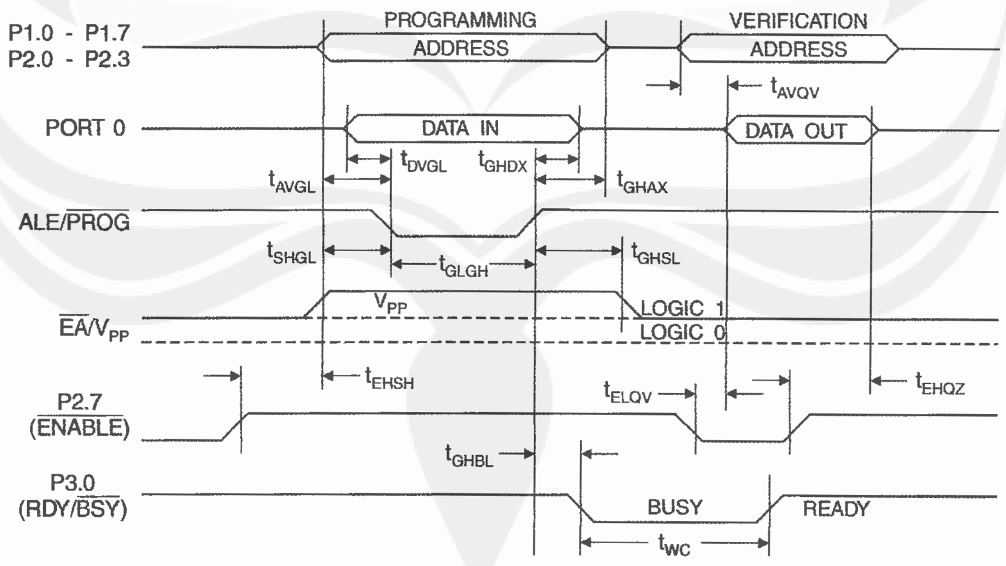
Figure 17-2. Verifying the Flash Memory (Parallel Mode)

8. Flash Programming and Verification Characteristics (Parallel Mode)

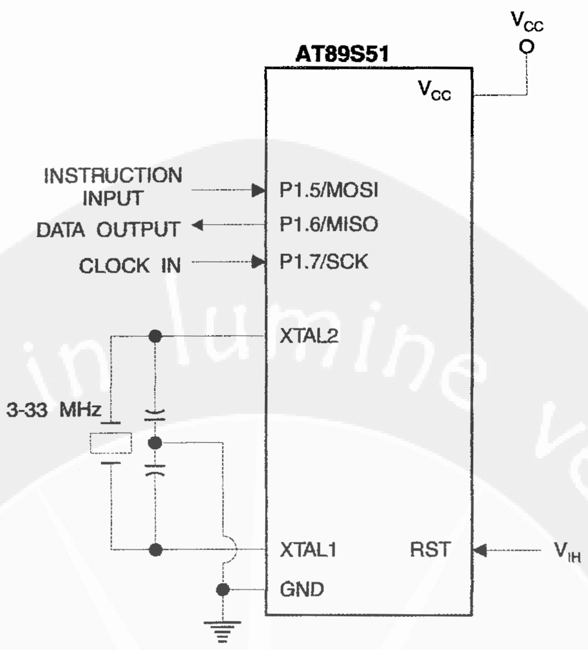
$_A$ = 20°C to 30°C, V_{CC} = 4.5 to 5.5V

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	11.5	12.5	V
$_{PP}$	Programming Supply Current		10	mA
$_{CC}$	V_{CC} Supply Current		30	mA
$1/t_{CLCL}$	Oscillator Frequency	3	33	MHz
$_{AVGL}$	Address Setup to $\overline{PROG}$ Low	48 t_{CLCL}		
$_{GHAX}$	Address Hold After $\overline{PROG}$	48 t_{CLCL}		
$_{DVGL}$	Data Setup to $\overline{PROG}$ Low	48 t_{CLCL}		
$_{GHDX}$	Data Hold After $\overline{PROG}$	48 t_{CLCL}		
$_{EHSH}$	P2.7 ($\overline{ENABLE}$) High to V_{PP}	48 t_{CLCL}		
$_{SHGL}$	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
$_{GHSL}$	V_{PP} Hold After $\overline{PROG}$	10		µs
$_{GLGH}$	$\overline{PROG}$ Width	0.2	1	µs
$_{AVQV}$	Address to Data Valid		48t_{CLCL}	
$_{ELQV}$	$\overline{ENABLE}$ Low to Data Valid		48t_{CLCL}	
$_{EHQZ}$	Data Float After $\overline{ENABLE}$	0	48t_{CLCL}	
$_{GHBL}$	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
$_{WC}$	Byte Write Cycle Time		50	µs

igure 18-1. Flash Programming and Verification Waveforms – Parallel Mode

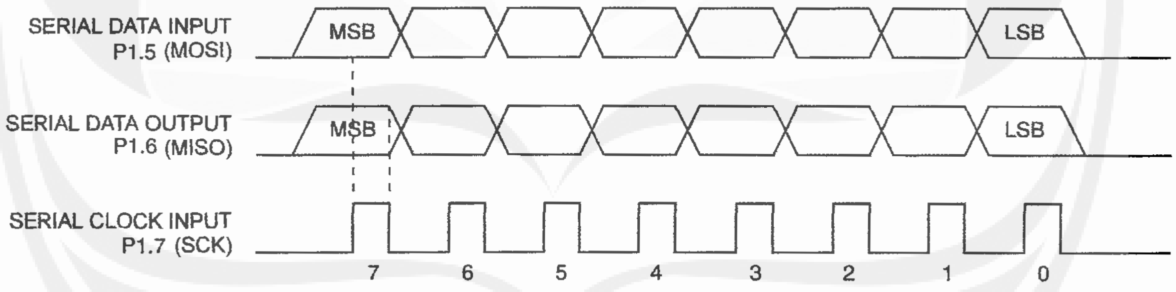


igure 18-2. Flash Memory Serial Downloading



9. Flash Programming and Verification Waveforms – Serial Mode

igure 19-1. Serial Programming Waveforms


SERIAL DATA INPUT P1.5 (MOSI)
MSB
LSB
SERIAL DATA OUTPUT P1.6 (MISO)
MSB
LSB
SERIAL CLOCK INPUT P1.7 (SCK)
7
6
5
4
3
2
1
0

0. Serial Programming Instruction Set

	Instruction Format				
nstruction	Byte 1	Byte 2	Byte 3	Byte 4	Operation
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx 0110 1001 (Output on MISO)	Enable Serial Programming while RST is high
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase Flash memory array
Read Program Memory (Byte Mode)	0010 0000	xxxx A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	D7 D6 D5 D4 D3 D2 D1 D0	Read data from Program memory in the byte mode
Write Program Memory (Byte Mode)	0100 0000	xxxx A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	D7 D6 D5 D4 D3 D2 D1 D0	Write data to Program memory in the byte mode
Write Lock Bits[1]	1010 1100	1110 00B1 B2	xxxx xxxx	xxxx xxxx	Write Lock bits. See Note (1).
Read Lock Bits	0010 0100	xxxx xxxx	xxxx xxxx	xxx LB3 LB2 LB1 xx	Read back current status of the lock bits (a programmed lock bit reads back as a "1")
Read Signature Bytes	0010 1000	xxxx A11 A10 A9 A8	A7 xxx xxx0	Signature Byte	Read Signature Byte
Read Program Memory (Page Mode)	0011 0000	xxxx A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Read data from Program memory in the Page Mode (256 bytes)
Write Program Memory (Page Mode)	0101 0000	xxxx A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Write data to Program memory in the Page Mode (256 bytes)

ote: 1. B1 = 0, B2 = 0 → Mode 1, no lock protection
B1 = 0, B2 = 1 → Mode 2, lock bit 1 activated
B1 = 1, B2 = 0 → Mode 3, lock bit 2 activated
B1 = 1, B2 = 1 → Mode 4, lock bit 3 activated

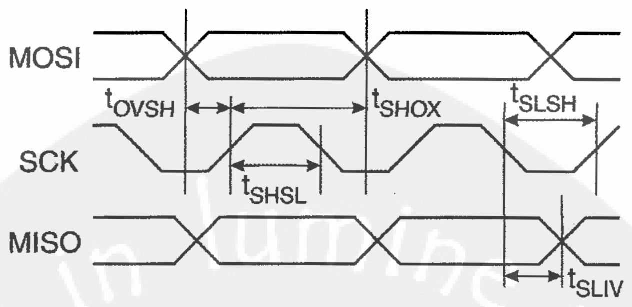
} Each of the lock bit modes need to be activated sequentially before Mode 4 can be executed.

fter Reset signal is high, SCK should be low for at least 64 system clocks before it goes high to clock in the enable data ytes. No pulsing of Reset signal is necessary. SCK should be no faster than 1/16 of the system clock at XTAL1.

or Page Read/Write, the data always starts from byte 0 to 255. After the command byte and upper address byte are tched, each byte thereafter is treated as data until all 256 bytes are shifted in/out. Then the next instruction will be ready be decoded.

1. Serial Programming Characteristics

gure 21-1. Serial Programming Timing



ble 21-1. Serial Programming Characteristics, T_A = -40°C to 85°C, V_{CC} = 4.0 - 5.5V (Unless Otherwise Noted)

ymbol	Parameter	Min	Typ	Max	Units
1/t_{CLCL}	Oscillator Frequency	3		33	MHz
CLCL	Oscillator Period	30			ns
SHSL	SCK Pulse Width High	8 t_{CLCL}			ns
SLSH	SCK Pulse Width Low	8 t_{CLCL}			ns
OVSH	MOSI Setup to SCK High	t_{CLCL}			ns
SHOX	MOSI Hold after SCK High	2 t_{CLCL}			ns
SLIV	SCK Low to MISO Valid	10	16	32	ns
ERASE	Chip Erase Instruction Cycle Time			500	ms
SWC	Serial Byte Write Cycle Time			64 t_{CLCL} + 400	μs

2. Absolute Maximum Ratings*

)perating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
/oltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
)C Output Current	15.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

3. DC Characteristics

ıe values shown in this table are valid for T_A = -40°C to 85°C and V_{CC} = 4.0V to 5.5V, unless otherwise noted.

ymbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC}-0.1	V
V_{IL1}	Input Low Voltage ($\overline{EA}$)		-0.5	0.2 V_{CC}-0.3	V
V_{IH}	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC}+0.9	V_{CC}+0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC}+0.5	V
V_{OL}	Output Low Voltage(1) (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
V_{OL1}	Output Low Voltage(1) (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
V_{OH}	Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 μA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 μA	0.75 V_{CC}		V
		I_{OH} = -10 μA	0.9 V_{CC}		V
V_{OH1}	Output High Voltage *(Port 0 in External Bus Mode)*	I_{OH} = -800 μA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 μA	0.75 V_{CC}		V
		I_{OH} = -80 μA	0.9 V_{CC}		V
IL	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	μA
TL	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-300	μA
LI	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	μA
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
CC	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power-down Mode(2)	V_{CC} = 5.5V		50	μA

otes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin: 10 mA
Maximum I_{OL} per 8-bit port:
Port 0: 26 mA Ports 1, 2, 3: 15 mA
Maximum total I_{OL} for all output pins: 71 mA
If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power-down is 2V.

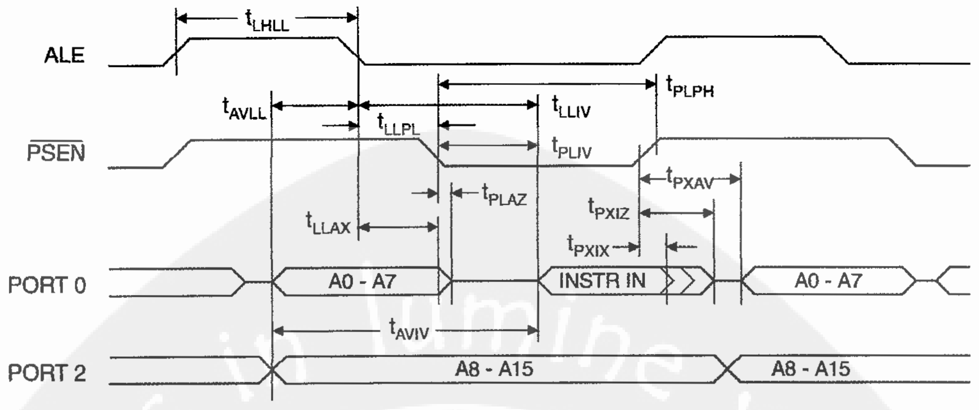
4. AC Characteristics

nder operating conditions, load capacitance for Port 0, ALE/$\overline{\text{PROG}}$, and $\overline{\text{PSEN}}$ = 100 pF; load capacitance for all other ıtputs = 80 pF.

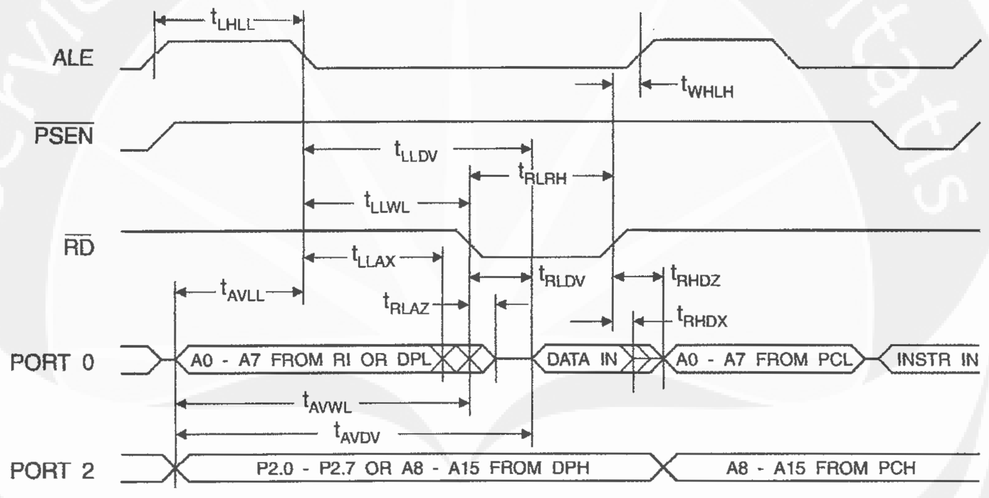
4.1 External Program and Data Memory Characteristics

ymbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/t_{CLCL}	Oscillator Frequency			0	33	MHz
LHLL	ALE Pulse Width	127		2 t_{CLCL}-40		ns
AVLL	Address Valid to ALE Low	43		t_{CLCL}-25		ns
LLAX	Address Hold After ALE Low	48		t_{CLCL}-25		ns
LLIV	ALE Low to Valid Instruction In		233		4 t_{CLCL}-65	ns
LLPL	ALE Low to $\overline{\text{PSEN}}$ Low	43		t_{CLCL}-25		ns
PLPH	$\overline{\text{PSEN}}$ Pulse Width	205		3 t_{CLCL}-45		ns
PLIV	$\overline{\text{PSEN}}$ Low to Valid Instruction In		145		3 t_{CLCL}-60	ns
PXIX	Input Instruction Hold After $\overline{\text{PSEN}}$	0		0		ns
PXIZ	Input Instruction Float After $\overline{\text{PSEN}}$		59		t_{CLCL}-25	ns
PXAV	$\overline{\text{PSEN}}$ to Address Valid	75		t_{CLCL}-8		ns
AVIV	Address to Valid Instruction In		312		5 t_{CLCL}-80	ns
PLAZ	$\overline{\text{PSEN}}$ Low to Address Float		10		10	ns
RLRH	$\overline{\text{RD}}$ Pulse Width	400		6 t_{CLCL}-100		ns
WLWH	$\overline{\text{WR}}$ Pulse Width	400		6 t_{CLCL}-100		ns
RLDV	$\overline{\text{RD}}$ Low to Valid Data In		252		5 t_{CLCL}-90	ns
RHDX	Data Hold After $\overline{\text{RD}}$	0		0		ns
RHDZ	Data Float After $\overline{\text{RD}}$		97		2 t_{CLCL}-28	ns
LLDV	ALE Low to Valid Data In		517		8 t_{CLCL}-150	ns
AVDV	Address to Valid Data In		585		9 t_{CLCL}-165	ns
LLWL	ALE Low to $\overline{\text{RD}}$ or $\overline{\text{WR}}$ Low	200	300	3 t_{CLCL}-50	3 t_{CLCL}+50	ns
AVWL	Address to $\overline{\text{RD}}$ or $\overline{\text{WR}}$ Low	203		4 t_{CLCL}-75		ns
QVWX	Data Valid to $\overline{\text{WR}}$ Transition	23		t_{CLCL}-30		ns
QVWH	Data Valid to $\overline{\text{WR}}$ High	433		7 t_{CLCL}-130		ns
WHQX	Data Hold After $\overline{\text{WR}}$	33		t_{CLCL}-25		ns
RLAZ	$\overline{\text{RD}}$ Low to Address Float		0		0	ns
WHLH	$\overline{\text{RD}}$ or $\overline{\text{WR}}$ High to ALE High	43	123	t_{CLCL}-25	t_{CLCL}+25	ns

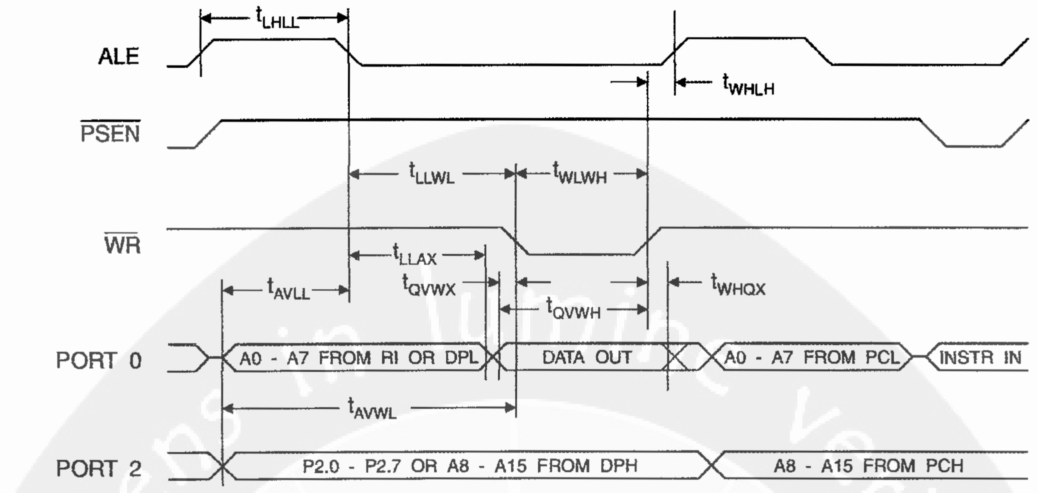
5. External Program Memory Read Cycle



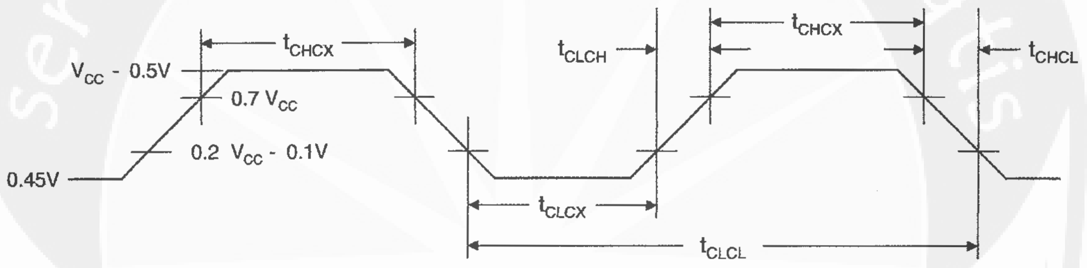
6. External Data Memory Read Cycle



7. External Data Memory Write Cycle



8. External Clock Drive Waveforms



9. External Clock Drive

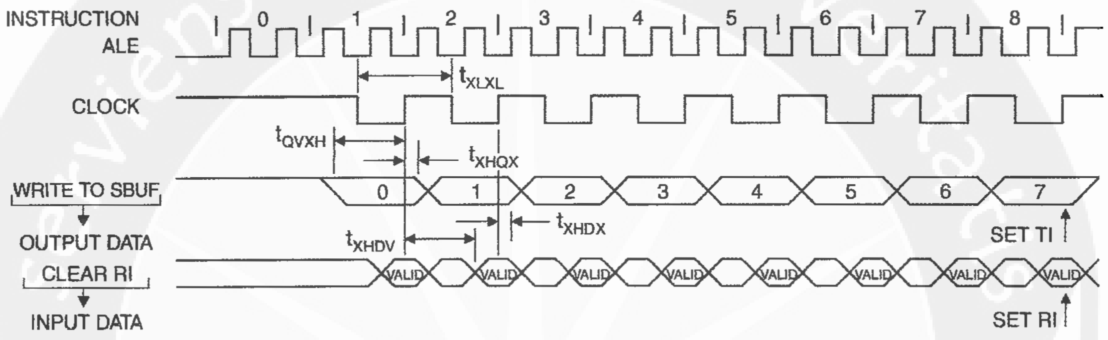
ymbol	Parameter	Min	Max	Units
/t_{CLCL}	Oscillator Frequency	0	33	MHz
CLCL	Clock Period	30		ns
CHCX	High Time	12		ns
CLCX	Low Time	12		ns
CLCH	Rise Time		5	ns
CHCL	Fall Time		5	ns

0. Serial Port Timing: Shift Register Mode Test Conditions

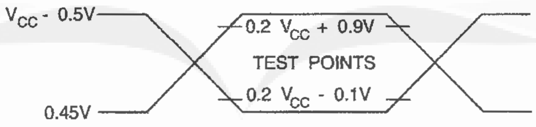
ıe values in this table are valid for V_{CC} = 4.0V to 5.5V and Load Capacitance = 80 pF.

ʒymbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
XLXL	Serial Port Clock Cycle Time	1.0		12 t_{CLCL}		μs
QVXH	Output Data Setup to Clock Rising Edge	700		10 t_{CLCL}-133		ns
XHQX	Output Data Hold After Clock Rising Edge	50		2 t_{CLCL}-80		ns
XHDX	Input Data Hold After Clock Rising Edge	0		0		ns
XHDV	Clock Rising Edge to Input Data Valid		700		10 t_{CLCL}-133	ns

1. Shift Register Mode Timing Waveforms

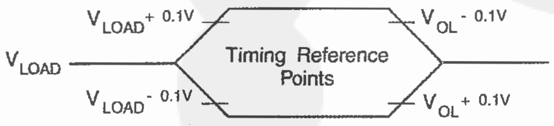


2. AC Testing Input/Output Waveforms[1]



ɔte: 1. AC Inputs during testing are driven at V_{CC} - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

3. Float Waveforms[1]



ɔte: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

4. Ordering Information

4.1 Standard Package

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
24	4.0V to 5.5V	AT89S51-24AC	44A	Commercial (0°C to 70°C)
		AT89S51-24JC	44J	
		AT89S51-24PC	40P6	
		AT89S51-24SC	42PS6	
		AT89S51-24AI	44A	Industrial (-40°C to 85°C)
		AT89S51-24JI	44J	
		AT89S51-24PI	40P6	
		AT89S51-24SI	42PS6	
33	4.5V to 5.5V	AT89S51-33AC	44A	Commercial (0°C to 70°C)
		AT89S51-33JC	44J	
		AT89S51-33PC	40P6	
		AT89S51-33SC	42PS6	

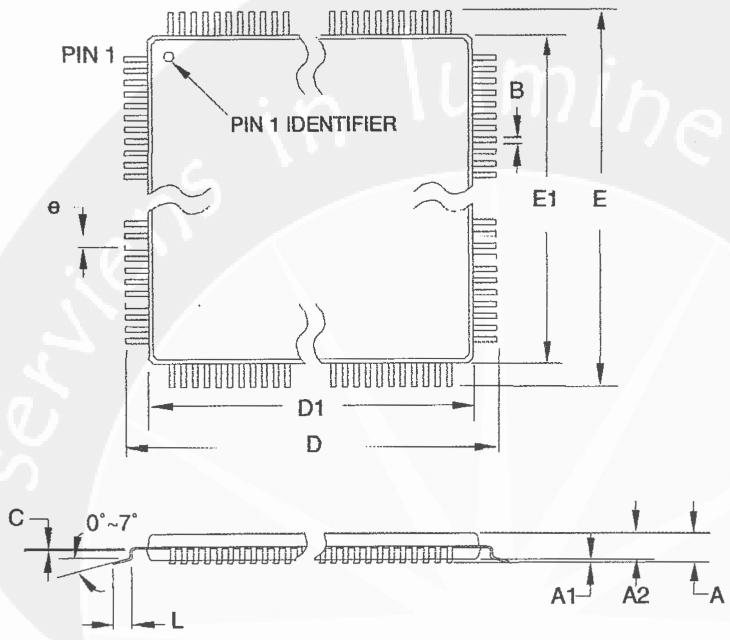
4.2 Green Package Option (Pb/Halide-free)

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
24	4.0V to 5.5V	AT89S51-24AU	44A	Industrial (-40°C to 85°C)
		AT89S51-24JU	44J	
		AT89S51-24PU	40P6	

Package Type	
44A	44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
44J	44-lead, Plastic J-leaded Chip Carrier (PLCC)
40P6	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)
42PS6	42-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)

5. Packaging Information

5.1 44A – TQFP



COMMON DIMENSIONS
(Unit of Measure = mm)

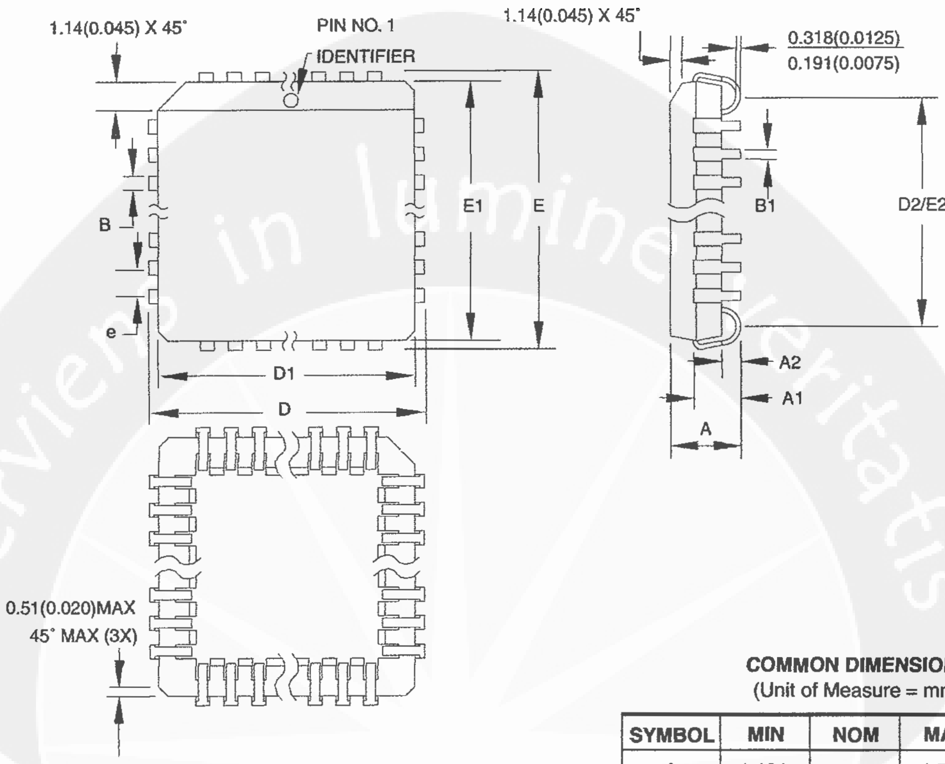
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	11.75	12.00	12.25	
D1	9.90	10.00	10.10	Note 2
E	11.75	12.00	12.25	
E1	9.90	10.00	10.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

Notes: 1. This package conforms to JEDEC reference MS-026, Variation ACB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

10/5/2001

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44A**, 44-lead, 10 x 10 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	44A	B

5.2 44J – PLCC



COMMON DIMENSIONS
(Unit of Measure = mm)

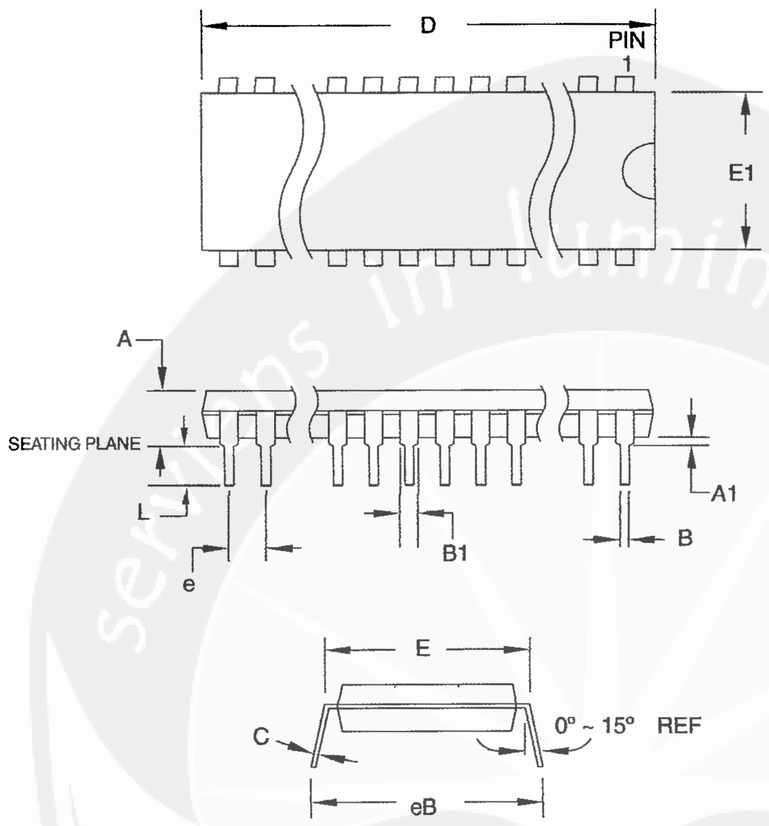
SYMBOL	MIN	NOM	MAX	NOTE
A	4.191	–	4.572	
A1	2.286	–	3.048	
A2	0.508	–	–	
D	17.399	–	17.653	
D1	16.510	–	16.662	Note 2
E	17.399	–	17.653	
E1	16.510	–	16.662	Note 2
D2/E2	14.986	–	16.002	
B	0.660	–	0.813	
B1	0.330	–	0.533	
e	1.270 TYP			

Notes:
1. This package conforms to JEDEC reference MS-018, Variation AC.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is .010"(0.254 mm) per side. Dimension D1 and E1 include mold mismatch and are measured at the extreme material condition at the upper or lower parting line.
3. Lead coplanarity is 0.004" (0.102 mm) maximum.

10/04/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44J**, 44-lead, Plastic J-leaded Chip Carrier (PLCC)	44J	B

.3 40P6 – PDIP



COMMON DIMENSIONS
(Unit of Measure = mm)

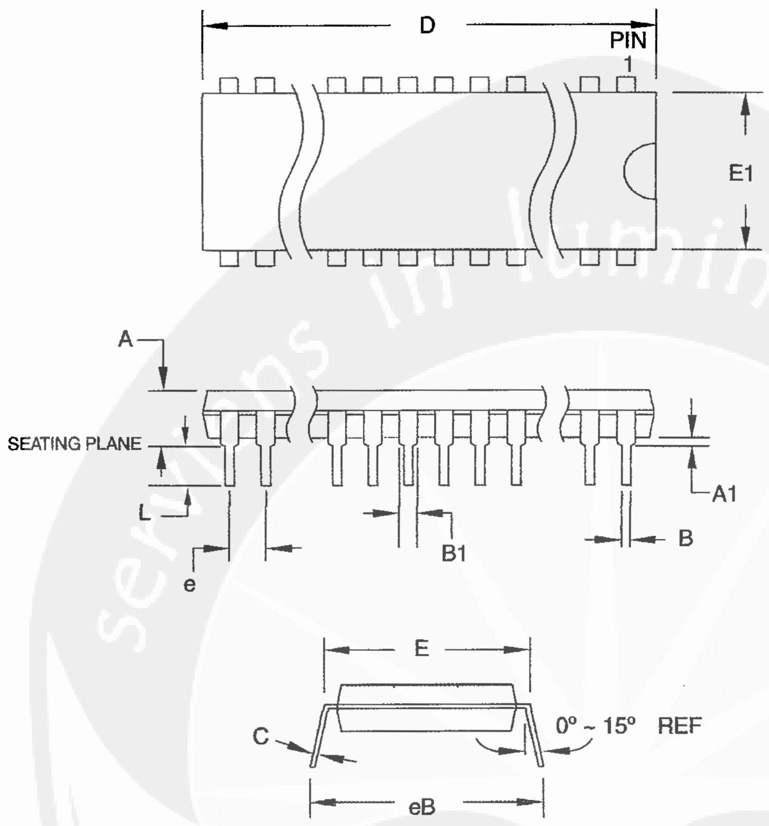
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.826	
A1	0.381	–	–	
D	52.070	–	52.578	Note 2
E	15.240	–	15.875	
E1	13.462	–	13.970	Note 2
B	0.356	–	0.559	
B1	1.041	–	1.651	
L	3.048	–	3.556	
C	0.203	–	0.381	
eB	15.494	–	17.526	
e	2.540 TYP			

Notes: 1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

09/28/01

	TITLE	DRAWING NO.	REV.
ATMEL 2325 Orchard Parkway San Jose, CA 95131	**40P6**, 40-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	40P6	B

5.4 42PS6 – PDIP



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.83	
A1	0.51	–	–	
D	36.70	–	36.96	Note 2
E	15.24	–	15.88	
E1	13.46	–	13.97	Note 2
B	0.38	–	0.56	
B1	0.76	–	1.27	
L	3.05	–	3.43	
C	0.20	–	0.30	
eB	–	–	18.55	
e	1.78 TYP			

Notes: 1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

11/6/03

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**42PS6**, 42-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	42PS6	A

tmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

egional Headquarters

urope
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

sia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

apan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

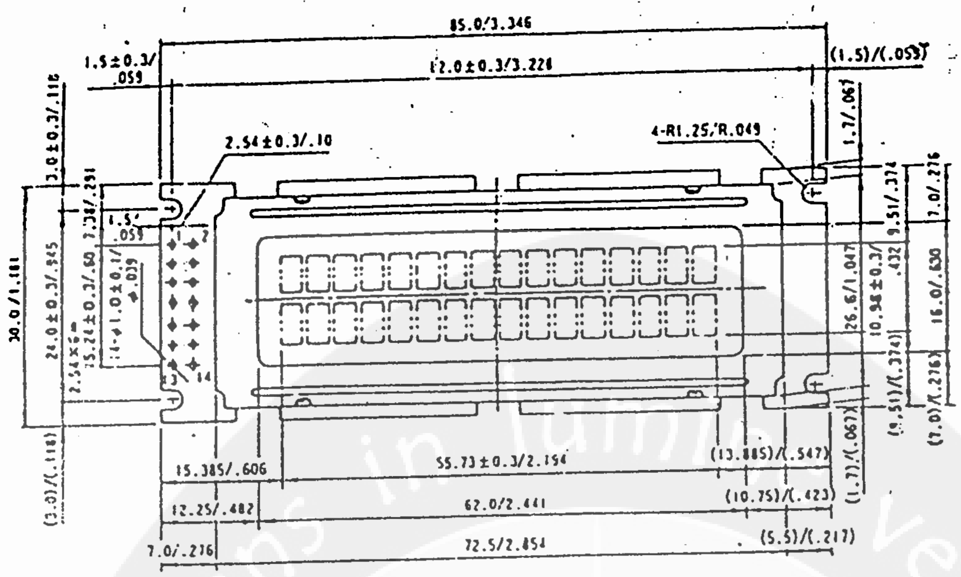
Literature Requests
www.atmel.com/literature

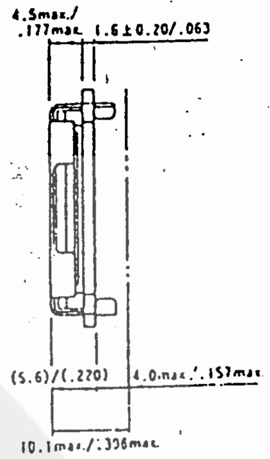
Printed on recycled paper.

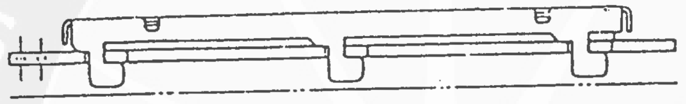
Lampiran 2 :

Data Sheet LCD

Dimensions Diagram







Unit : mm/inch
General tolerance : ±0.5 mm

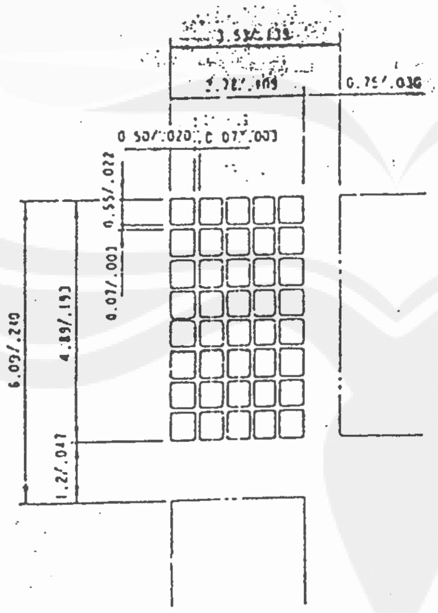


Figure 1 Dimensions diagram

No.	Symbol	Level	Function	
1	Vss	-	Power Supply	0V (GND)
2	Vcc	-	Power Supply	5V ±10%
3	Vcc	-	Power Supply	0V for LCD Drive
4	RS	H/L	H: Data Input L: Instruction Input	
5	R/W	H/L	H:READ L:WRITE	
6	E	H, ⌞	Enable Signal	
7	DB0	H/L	Data Bus	
8	DB1	H/L	Data Bus	
9	DB2	H/L	Data Bus	
10	DB3	H/L	Data Bus	
11	DB4	H/L	Data Bus	
12	DB5	H/L	Data Bus	
13	DB6	H/L	Data Bus	
14	DB7	H/L	Data Bus	
15	V+BL	-	Back Light Supply	4 - 4.2V 50-200mA
16	V-BL	-	Back Light Supply	0V (GND)

OPERATING INSTRUCTIONS

NTRODUCTION

Seiko Instruments intelligent dot matrix liquid crystal lisplay modules have on-board controller and LSI drivers. vhich display alpha numerics, Japanese KATA KANA chai-icters and a wide variety of other symbols in either 5 x 7 dot natrix.

The internal operation in the KS0066 controller chip is letermined by signals sent from the MPU. The signals include: 1) Register select RS input consisting of instruction register (IR) when RS = 0 and data register (DR) when RS = 1; 2) Read/write (R/W); 3) Data bus (DB7~ DB0); and 4) Enable strobe (E) depending on the MPU or through an external parallel I/O port. Details on instructions data entry, execution times, etc. are explained in the following sectio.s.

READ AND WRITE TIMING DIAGRAMS AND TABLES

he following timing characteristics are applicable for all of Seiko's LCD dot matrix character modules.

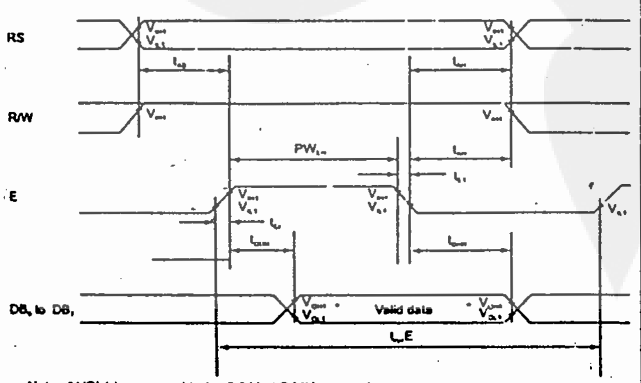
READ TIMING CHARACTERISTICS
V_{DD}=5.0V±5%, V_{SS}=0V, T_a=0°C TO 50°C

Item	Symbol	Standard Min.	Standard Max.	Unit
Enable cycle time	$t_{CYC}E$	500	—	ns
Enable pulse width High Level	PW_{EH}	230	—	ns
Enable rise and fall time	t_{ER}, t_{EF}	—	20	ns
Address setup time RS,R/W—E	t_{AS}	40	—	ns
Address hold time	t_{AH}	10	—	ns
Data delay time	t_{DDR}	—	160	ns
Data hold time	t_H	5	—	ns

WRITE TIMING CHARACTERISTICS
V_{DD}=5.0V±5%, V_{SS}=0V, T_a=0°C TO 50°C

Item	Symbol	Standard Min.	Standard Max.	Unit
Enable cycle time	$t_{CYC}E$	500	—	ns
Enable pulse width High Level	PW_{EH}	230	—	ns
Enable rise and fall time	t_{ER}, t_{EF}	—	20	ns
Address setup time RS,R/W—E	t_{AS}	40	—	ns
Address hold time	t_{AH}	10	—	ns
Data setup time	t_{DSW}	80	—	ns
Data hold time	t_H	10	—	ns

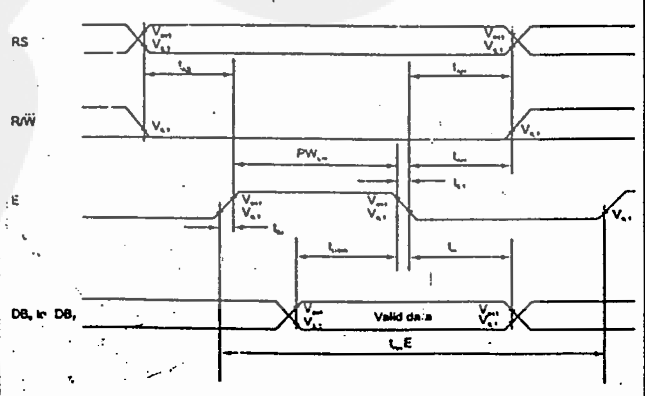
READ OPERATION



Note: * VOL1 is assumed to be 0.8 V at 2 MHz operation.

DATA READ FROM MODULE TO MPU

WRITE OPERATION



DATA WRITE FROM MPU TO MODULE

Operating Instructions

Introduction Codes

Instruction	Set		Instruction Code								Description	Execution Time (when f_{cp} or f_{osc} is 250 kHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display memory and returns the cursor to the home position (Address 0).	82 μs - 1.64ms
Return Home	0	0	0	0	0	0	0	0	1	•	Returns the cursor to the home position (Address 0) shifted to the original position. DD RAM contents remain unchanged.	40 μs - 1.6ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies to or not to shift the display. These operations write and read.	40 μs - 1.64ms
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	(D) is display ON/OFF control; memory remains unchanged in OFF condition. (C) cursor ON/OFF (B) blinking cursor.	40 μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	•	•	Moves the cursor and shifts the display without changing DD RAM contents.	40 μs
Function Set	0	0	0	0	1	DL	N	F	•	•	Sets interface data length (DL), number of display lines (N), and character font (F).	40 μs
Set CG RAM Address	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40 μs
Set DD RAM Address	0	0	1	A_{DD}							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40 μs
Read Busy Flag & Address	0	1	BF	AC							Reads Busy Flag (BF) indicating internal operation is being performed and reads address counter contents.	1 μs
Write Data to CG or to DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40 μs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40 μs

• Doesn't matter

DD RAM:	Display data RAM
CG RAM:	Character generator RAM
A_{CG}:	CG RAM address
A_{DD}:	DD RAM address corresponds to cursor address
A_C:	Address counter used for both DD RAM and CG RAM address

I/D = 1:	Increment
I/D = 0:	Decrement
S = 1:	Display shift
S = 0:	No display shift
D = 1:	Display ON
D = 0:	Display OFF

C = 1:	Cursor ON
C = 0:	Cursor OFF
B = 1:	Blink ON
B = 0:	Blink OFF
S/C = 1:	Display shift
S/C = 0:	Cursor movement
BF = 1:	Internal operation in progress
BF = 0:	Instruction can be accepted

R/L = 1:	Right shift
R/L = 0:	Left shift
DL = 1:	8 bits
DL = 0:	4 bits
N = 1:	2 lines (L1671)
F = 0:	5 x 7 dot matrix

Execution times in the above table indicate the minimum values when operating frequency is 250 kHz.

When f_{osc} is 270 kHz: 40μs x 250/250 = 37μs

OPERATING INSTRUCTIONS

NTRODUCTION CODE EXPLANATIONS

The two registers 1) Instruction Register (IR) and the) Data Register (DR) in the KS0066 controller chip are directly ontrolled by the MPU. Control information is temporarily stored these registers prior to internal operation start. This allows terface to various types of MPUs which operate at different speeds from that of the KS0066, and allows interface from peripheral control ICs. Internal operations of the KS0066 are determined from the signals sent from the MPU. These signals, including register selection signals (RS), Read/Write (R/W) and data bus signals (DB0 - DB7) are polled instructions.

REGISTER SELECTION

RS	R/W	Operation
0	0	IR selection, IR write. Internal operation: Display clear
0	1	Busy flag (DB7) and address counter (DB0 to DB6) read
1	0	DR selection, DR write. Internal operation: DR to DD RAM or CG RAM
1	1	DR selection, DR read. Internal operation: DD RAM or CG RAM to DR

DDRESS COUNTER (AC)

The counter specifies an address when data is written to DD RAM or CG RAM and the data stored in DD RAM or G RAM is read out. If an Address Set instruction (for DD AM or CG RAM) is written in the IR, the address information transferred from the IR to the AC. When display data is written into or read from DD RAM or CG RAM, the AC is automatically incremented or decremented by one according to the Entry Mode Set. The contents of the AC are output to DB0 to DB6; refer to above "Register Selection Table" when RS = 0 and R/W = 1.

LEAR DISPLAY

	RS	R/W	DB7							DB0
Code	0	0	0	0	0	0	0	0	0	1

Clear all display memory and return the cursor to the home position. In other words, the cursor returns to the first character block on the first line on all 1, 2, and 4 line character modules except L4044. I the above is entered on E2 (the second controller for lines 3 and 4), the cursor will return to the first character on the third line.

URSOR HOME

	RS	R/W	DB7							DB0
Code	0	0	0	0	0	0	0	0	1	*

*Doesn't matter

Returns cursor to home position. First line first character blocks on all 1, 2 and 4 line display; except L4044 refer "clear display": (Address 0; A₀₀ "80"). The contents of the DD RAM remain unchanged.

RESTRICTIONS ON EXECUTION OF DISPLAY CLEAR AND CURSOR HOME INSTRUCTIONS

Conditions of use	Restrictions
When executing the Display Clear or Cursor Home instruction when the display is shifted (after execution of Display Shift instruction).	The Cursor Home instruction should be executed again immediately after the Display Clear or Cursor Home instruction is executed. Do not leave an interval of a multiple of 400 f_{osc}* second after the first execution. • L4052: f_{osc} = 250 kHz • The other modules: f_{osc} = 270 kHz *f_{osc}: Oscillation frequency
When 23_H, 27_H, 63_H, or 67_H is used as a DD.RAM address to execute Cursor Home instruction.	Before executing the Cursor Home instruction, the data of the four DD RAM addresses given at the left should be read and saved. After execution, write the data again in DD RAM. (This restriction is necessary to prevent the contents of the DD RAM addresses from being destroyed after the Cursor Home instruction has been executed.)

Operating Instructions

Entry Mode Set

	RS	R/W	DB7							DB0
Code	0	0	0	0	0	0	0	1	I/D	S

I/D: Increments (I/D = 1) or decrements (I/D = 0) the DD RAM address by one block when writing or reading a character code from DD RAM or CG RAM. The cursor automatically moves to the right when incremented by one or to the left if decremented by one.

S: Shifts the entire display to either the right or left when S = 1 (high). When S = 1 and I/D = 1 the display shifts one position to the left. When S = 1 and 1/D = 0 the display shifts one position to the right. This right or left shift occurs after each data write to DD RAM. Display is not shifted when reading from DD RAM. Display is not shifted when S = 0.

Display and Cursor ON/OFF Control

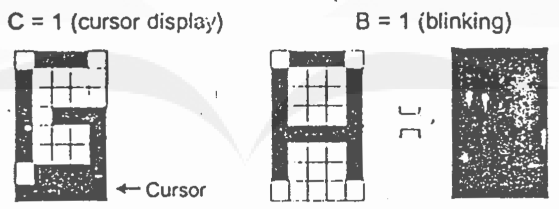
	RS	R/W	DB7							DB0
Code	0	0	0	0	0	0	1	D	C	B

D: Display is turned ON when D = 1 and OFF when D = 0. When display is OFF, display data in DD RAM remains unchanged. Information comes back immediately when D = 1 is entered.

C: Cursor is displayed when C = 1 and not displayed when C = 0. If the cursor disappears, function of I/D etc. does not change during display data write. In a 5 x 7 dot matrix there is an eighth line which functions as the cursor.

B: When B = 1, the character at the cursor position starts blinking. When B = 0 the cursor does not blink. The blink is done by stiching between the all black dot matrix and displayed character at 0.4 seconds intervals. The cursor and the blink can be set at the same time (fosc = 250 kHz).

5 x 7 dot matrix



Cursor or Display Shift

	RS	R/W	DB7							DB0
Code	0	0	0	0	0	1	S/C	R/L	•	•

• Doesn't Matter

Cursor/Display Shift moves the cursor or shifts the display without changing the DD RAM contents.

The cursor position and the AC contents match. This instruction is available for display correction and retrieval because the cursor position or display can be shifted without writing or reading display data. In case of a 2-line display, the cursor is shifted from character block 40 of line 1 to character block 1 of line 2. Displays of lines 1 and 2 are shifted at the same time. In case of a 4-line display, the cursor does not move continuously from line 2 to line 3. The cursor is shifted from character block 40 of line 3 to character block 1 of line 4. Displays of lines 3 and 4 are shifted at the same time. The display pattern of line 2 or 4 is not shifted to line 1 or 3.

S/C	R/L	Operation
0	0	The cursor position is shifted to the left (the AC decrements one)
0	1	The cursor position is shifted to the right (the AC increments one)
1	0	The entire display is shifted to the left with the cursor
1	1	The entire display is shifted to the right with the cursor

Operating Instructions

unction Set

	RS	R/W	DB7							DB0
Code	0	0	0	0	1	DL	N	F	*	*

* Doesn't Matter

Function Set sets the interface data length, the number f display lines and the character font.

DL: Interface data length

When DL = 1, the data length is set at 8 bits (DB7 to DB0).

When DL = 0, the data length is set at 4 bits (DB7 to DB4).

The upper 4 bits are transferred first, then the lower 4 bits follow.

N: Number of display lines

F: Sets character font

N	F	Number of display lines	Character font	Duty Cycle	LCD Module
1	0	2	5 x 7 dot matrix	1/16	All character LCD modules

The Function Set instruction must be executed prior to all other instructions except for Busy Flag/Address Read. If another instruction is executed first, no function instruction except changing the interface data lenght can be executed.

G RAM Address Set

	RS	R/W	DB7							DB0
Code	0	0	0	1	A	A	A	A	A	A

← Upper bit Lower bit →

CG RAM addresses, expressed as binary AAAAAA, are et to the AC. Then data in CG RAM is written from or read to ne MPU.

D RAM Address Set

	RS	R/W	DB7							DB0
Code	0	0	1	A	A	A	A	A	A	A

← Upper bit Lower bit →

DD RAM addresses expressed as binary AAAAAA are et to the AC. Then data in DD RAM is written from or read to ne MPU.

usy Flag/Address Read

	RS	R/W	DB7							DB0
Code	0	0	BF	A	A	A	A	A	A	A

← Upper bit Lower bit →

The BF signal can be read to verify if the controller is indiating that the module is working on a current instruction.

When BF = 1, the module is working internally and the next instruction cannot be accepted until the BF value becomes 0.

When BF = 0, the next instruction can be accepted.

Therefore, make sure that BF = 0 before writing the next struction. The AC values of binary AAAAAA are read out at e same time as reading the busy flag. The AC addresses e used for both CG RAM and DD RAM but the address set efore execution of the instruction determines which address to be used.

Data Write to CG RAM or DD RAM

	RS	R/W	DB7							DB0
Code	1	0	D	D	D	D	D	D	D	D

← Upper bit Lower bit →

Binary eight-bit data DDDDDDDD is written into CG RAM or DD RAM. The CG RAM Address Set instruction or the DD RAM Address Set instruction before this instruction selects either RAM. After the write operation, the address and display shift are determined by the entry mode setting.

Data Read to CG RAM or DD RAM

	RS	R/W	DB7							DB0
Code	1	1	D	D	D	D	D	D	D	D

← Upper bit Lower bit →

Binary eight-bit data DDDDDDDD is read from CG RAM or DD RAM. The CG RAM Address Set instruction or the DD RAM Address Set instruction before this instruction selects either RAM. In addition, either instruction is executed immediately before this instruction. If no Address Set instruction is executed before a read instruction, the first data read becomes invalid. If read instructions are executed consecutively, data is normally read from the second time. However, if the cursor is shifted by the Cursor Shift instruction when reading DD RAM, there is no need to execute an address set instruction because the Cursor Shift instruction does this.

After the read operation, the address is automatically incremented or decremented by one according to the entry mode, but the display is not shifted.

Note: The AC is automatically incremented or decremented by one according to the entry mode after a write instruction is excecuted to write data in CG RAM or DD RAM. However, the data of the RAM selected by the AC are not read out even if a read instruction is executed immediately afterwards.

5 x 7 + CURSOR

Relationships between CG RAM addresses and character codes (DD RAM) and character patterns (CG RAM data). (5 x 7 dot matrix).5 X 7 Table

Character code (DD RAM data)	CG RAM address		Character pattern (CG RAM data)		
7 6 5 4 3 2 1 0 Upper bit Lower bit	5 4 3 Upper bit	2 1 0 Lower bit	7 6 5 Upper bit	4 3 2 1 0 Lower bit	
		0 0 0	• • •	1 1 1 1 0	Example of character pattern (R)
		0 0 1		1 0 0 0 1	
		0 1 0		1 0 0 0 1	
		0 1 1		1 1 1 1 0	
0 0 0 0 * 0 0 0	0 0 0	1 0 0		1 0 1 0 0	
		1 0 1		1 0 0 1 0	
		1 1 0		1 0 0 0 1	
		1 1 1	• • •	0 0 0 0 0	Cursor position
		0 0 0	• • •	1 0 0 0 1	Example of character pattern (¥)
		0 0 1		0 1 0 1 0	
		0 1 0		1 1 1 1 1	
0 0 0 0 * 0 0 1	0 0 1	0 1 1		0 0 1 0 0	
		1 0 0		1 1 1 1 1	
		1 0 1		0 0 1 0 0	
		1 1 0		0 0 1 0 0	
		1 1 1	• • •	0 0 0 0 0	
		0 0 0	• • •		
		0 0 1			
0 0 0 0 * 1 1 1	1 1 1				
		1 0 0			
		1 0 1			
		1 1 0			
		1 1 1	• • •		

NOTES:

- In CG RAM data, 1 corresponds to Selection and 0 to Non-selection on the display.
- Character code bits 0 to 2 and CG RAM address bits 3 to 5 correspond with each other (three bits, eight types).
- CG RAM address bits 0 to 2 specify a line position for a character pattern. Line 8 of a character pattern is the cursor position where the logical sum of the cursor and CG RAM data is displayed. Set the data of line 8 to 0 to display the cursor. If the data is charged to 1, one bit lights, regardless of the cursor.
- The character pattern column position corresponds to CG RAM data bits 0 to 4 and bit 4 comes to the left end. CG RAM data bits 5 to 7 are not displayed but can be used as general data RAM.
- When reading a character pattern from CG RAM, set to 0 all of character code bits 4 to 7. Bits 0 to 2 determine which pattern will be read out. Since bit 3 is not valid, 00H and 08H select the same character.

OPERATING INSTRUCTIONS

ROGRAMMING THE CHARACTER GENERATOR RAM (CG RAM)

The character generator RAM (CG RAM) allows the user create up to eight custom 5 x 7 characters + cursor (5 x 8). nce programmed, the custom characters or symbols are ccessed exactly as if they were in ROM. However since the AM is a volatile memory, power must be continually aintained. Otherwise, the custom characters/symbols must e programmed into non-volatile external ROM and sent to e display after each display initialization. All dots in the 5 x 8 ot matrix can be programmed, which includes the cursor osition.

The modules RAM are divided into two parts: data splay RAM (DD RAM) and custom character generator RAM G RAM). This is not to be confused programming the stom character generator RAM with the 192 character nerator ROM. The CG RAM is located between hex 40 and and is contiguous. Locations 40 thru 47 hold the first stom character (5 x 8), 48 thru 4F hold the second custom aracter, 50 thru 57 hold the third CG, and so forth to 78 u 7F for the eighth CG character/symbol.

If during initialization the display was programmed to automatically increment, then only the single initial address, 40, need be sent. Consecutive row data will automatically appear at 41, 42, etc. until the completed character is formed. All eight custom CG characters can be programmed in 64 consecutive "writes" after sending the single initial 40 address.

The CG RAM is 8 bits wide, although only the right-most 5-bits are used for a custom CG character row. The left-most dot of programming the CG RAM character corresponds to D4 in the most significant nibble (XXXD4) of the data bus code, with the remaining 4 dots in the row corresponding to the least significant nibble (D3 thru D0), D0 being the right-most dot. Thus, hex 1F equals all dots on and hex 00 equals all dots off. Examples include hex 15 (10101) equal to 3 dots on the hex 0A (01010) equal 2 dots on. In each case the key 5-bits of the 8-bit code program one row of a custom CG character. When all 7 or 8 rows are programmed, the character is complete. A graphic example is shown below:

RS	R/W	Data	Display	Description
0	0	40	—	addresses 1st row, 1st CG character
1	0	11	• •	result of 11, 1st row
1	0	0A	• •	result of 0A, 2nd row
1	0	1F	•••••	result of 1F, 3rd row
1	0	04	•	result of 04, 4th row
1	0	1F	•••••	result of 1F, 5th row
1	0	04	•	result of 04, 6th row
1	0	04	•	result of 04, 7th row
1	0	00	—	result of 00, 8th row (cursor position)
1	0	15	•••	1st row, 2nd CG character. Note: Addressing not now required; hex 48 is next in the sequence.

Operating Instructions

Address Locations

1) **L1671-Series** (16 characters x 1 line)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	80	81	82	83	84	85	86	87	C0	C1	C2	C3	C4	C5	C6	C7

NOTE: L1671 series is initialized as a 2 line display, because of the absence of an LCD driver. You must address character no. 9 as you would the first position on the 2nd line which is (C0).

2) **L1672-Series** (16 characters x 2 lines)
L1682-Series
L1692-Series

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

3) **L1634-Series** (16 characters x 4 lines)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
Line 3	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
Line 4	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF

4) **L2032-Series** (20 characters x 2 lines)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3

5) **L2034-Series** (20 characters x 4 lines)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3
Line 3	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
Line 4	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7

6) **L2462-Series** (24 characters x 2 lines)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7

7) **L4052-Series** (40 characters x 2 lines)
L4044-Series (40 characters x 4 lines)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7
Line 3	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
Line 4	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7

Note: Address locations on lines 1 & 2 are controlled by enabling E1. Address locations on lines 3 & 4 are controlled by enabling E2.

CHARACTER FONT CODES (5 x 7 DOT MATRIX)

Upper 4 Bit Hexadecimal

Lower 4 Bit Hexadecimal

Upper 4 bits / Lower 4 bits	0000 (0)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1010 (A)	1011 (B)	1100 (C)	1101 (D)	1110 (E)	1111 (F)
× × × × 0000 (0)	CG RAM (1)		0	@	P	`	p		ー	タ	ミ	α	p
× × × × 0001 (1)	(2)	!	1	A	Q	a	q	。	ア	チ	ム	ä	q
× × × × 0010 (2)	(3)	"	2	B	R	b	r	「	イ	ツ	メ	β	θ
× × × × 0011 (3)	(4)	#	3	C	S	c	s	」	ウ	テ	モ	ε	∞
× × × × 0100 (4)	(5)	$	4	D	T	d	t	、	エ	ト	ヤ	μ	Ω
× × × × 0101 (5)	(6)	%	5	E	U	e	u	・	オ	ナ	ユ	σ	ü
× × × × 0110 (6)	(7)	&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
× × × × 0111 (7)	(8)	'	7	G	W	g	w	ァ	キ	ヌ	ラ	g	π
× × × × 1000 (8)	(1)	(	8	H	X	h	x	ィ	ク	ネ	リ	√	x̄
× × × × 1001 (9)	(2)	)	9	I	Y	i	y	ゥ	ケ	ノ	ル	⁻¹	y
× × × × 1010 (A)	(3)	*	:	J	Z	j	z	ェ	コ	ハ	レ	j	千
× × × × 1011 (B)	(4)	+	;	K	[	k	{	ォ	サ	ヒ	ロ	ˣ	万
× × × × 1100 (C)	(5)	,	<	L	¥	l	|	ャ	シ	フ	ワ	¢	円
× × × × 1101 (D)	(6)	-	=	M	]	m	}	ュ	ス	ヘ	ン	£	÷
× × × × 1110 (E)	(7)	.	>	N	^	n	→	ョ	セ	ホ	゛	ñ	
× × × × 1111 (F)	(8)	/	?	O	_	o	←	ッ	ソ	マ	゜	ö	█

Operating Instructions

Examples of 8-Bit and 4-Bit Data Transfer Operation

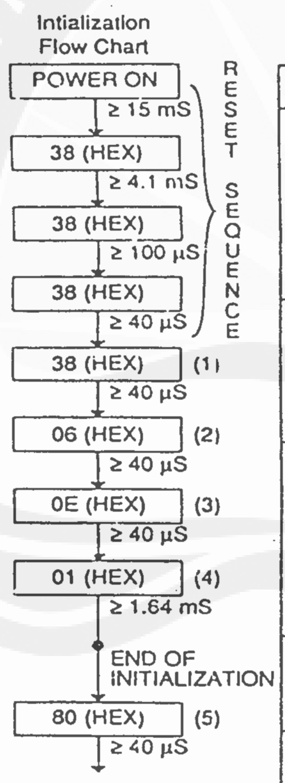
Display initialization

Each time the module is turned on or reset, an initialization procedure must be executed. The procedure consists of sending a sequence of hex codes from the microprocessor or parallel I/O port. The initialization sequence turns on the cursor, clears the display, and sets the module onto an auto-increment mode.

The initial hex code 30, 34, or 38 is sent two or more times to ensure the module enters the 8-bit or 4-bit data mode. All the initialization sequences are performed under the condition of Register Select (RS) = 0 (low) and Read/Write (R/W) = 0 (low).

The 4-bit data bus microcontroller may operate the display module by sending the initialization sequence in 4-bit format. Since 4-bit operation requires the data to be sent twice over the higher 4-bit bus lines (D4-D7), memory requirements are doubled.

A. Example for the module with 5 x 7 Character Format Under 8-Bit Data Transfer

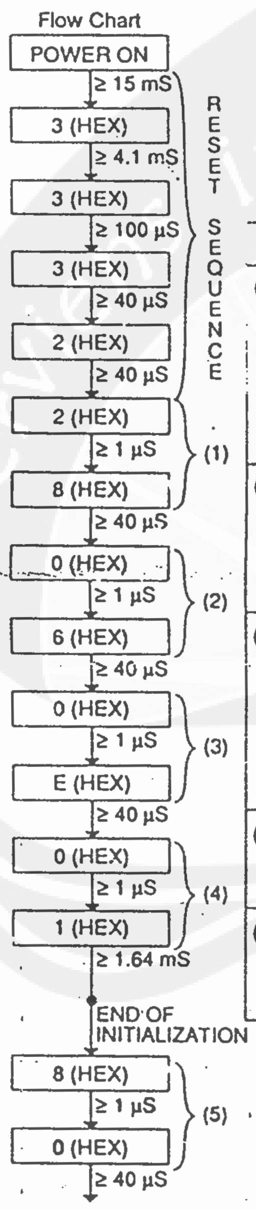


	Hex Code	D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Instructions
(1)	38 (Hex)	0 0 1 1 1 0 0 0	Function Set 8 bit Data Length 2 Line 5 x 7 Dot Format
(2)	06 (Hex)	0 0 0 0 0 1 1 0	Entry Mode Set Increment one No Shift
(3)	0E (Hex)	0 0 0 0 1 1 1 0	Display ON/OFF Control Display ON Cursor ON Blink OFF
(4)	01 (Hex)	0 0 0 0 0 0 0 1	Display Clear
(5)	80 (Hex)	1 0 0 0 0 0 0 0	DD RAM Address Set 1st Digit

Note:
1) Both RS and R/W terminals shall be "0" in this sequence.
2) RS, R/W and Data are latched at the falling edge of the Enable signal, (falling edge is typically 10nSec; Max: 20nSec).
3) L4044 has t obe initialized on E1 and E2 respectively.

EXAMPLES OF 8-BIT AND 4-BIT DATA TRANSFER OPERATION

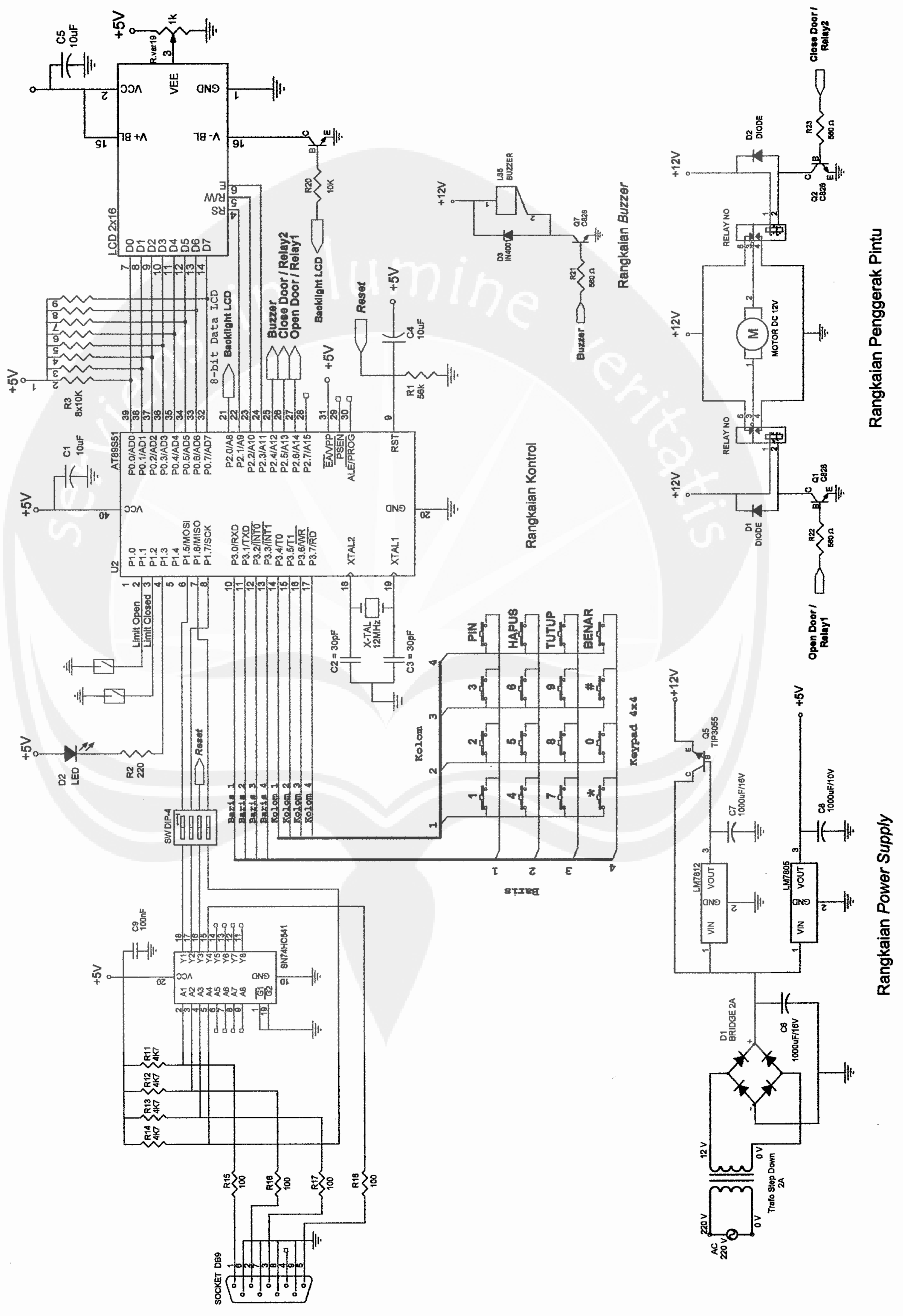
B. EXAMPLE FOR THE MODULE WITH 5 x 7 Character Format Under 4-Bit Data Transfer



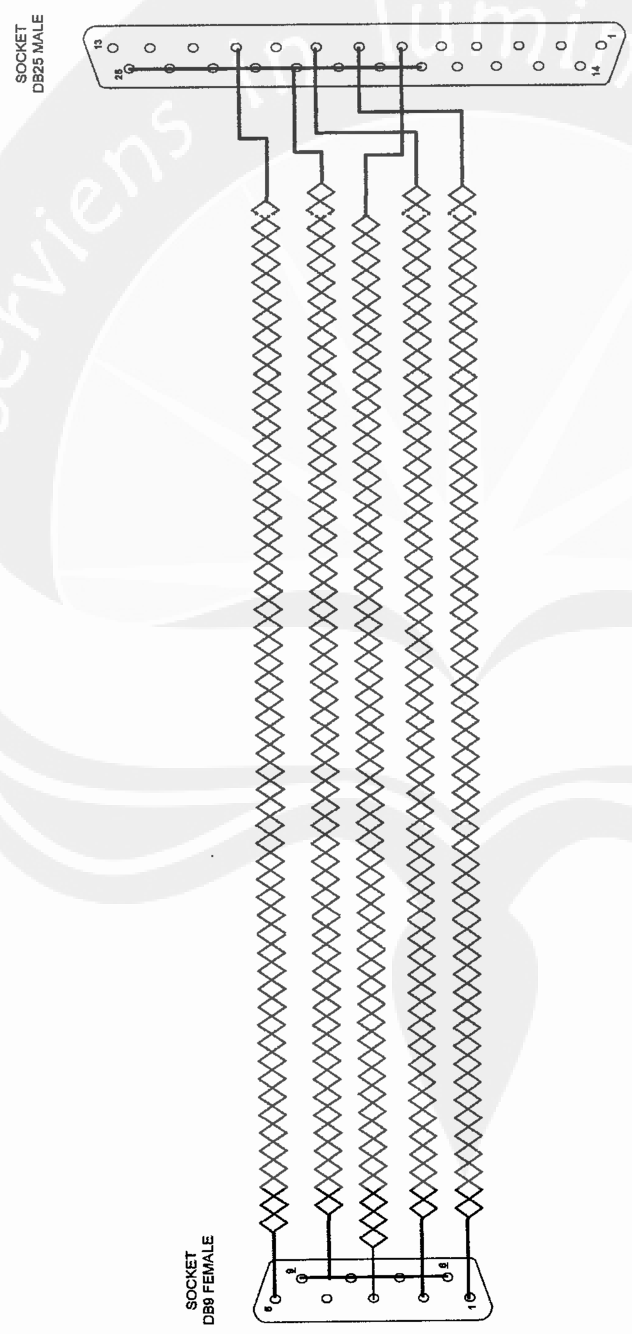
	Hex Code	1st D_7 D_6 D_5 D_4	2nd D_7 D_6 D_5 D_4	Instructions
(1)	2 (Hex) 8 (Hex)	0 0 1 0	1 0 0 0	→ Function Set → 4 bit Data Length → 2 Line → 5 x 7 Dot Format
(2)	0 (Hex) 6 (Hex)	0 0 0 0	0 1 1 0	→ Entry Mode Set → Increment → No Shift
(3)	0 (Hex) E (Hex)	0 0 0 0	1 1 1 0	→ Display ON/OFF Control → Display ON → Cursor ON → Blink OFF
(4)	0 (Hex) 1 (Hex)	0 0 0 0	0 0 0 1	→ Display Clear
(5)	8 (Hex) 0 (Hex)	1 0 0 0	0 0 0 0	→ DD RAM Address Set → 1st Digit

Note:
1) Both RS and R/W terminals shall be "0" in this sequence.
2) RS, R/W and Data are latched at the falling edge of the Enable signal.
3) Enable signal has to be sent after every 4-bit Data transfer.

Lampiran 3 : Gambar Rangkaian Kunci Elektronik



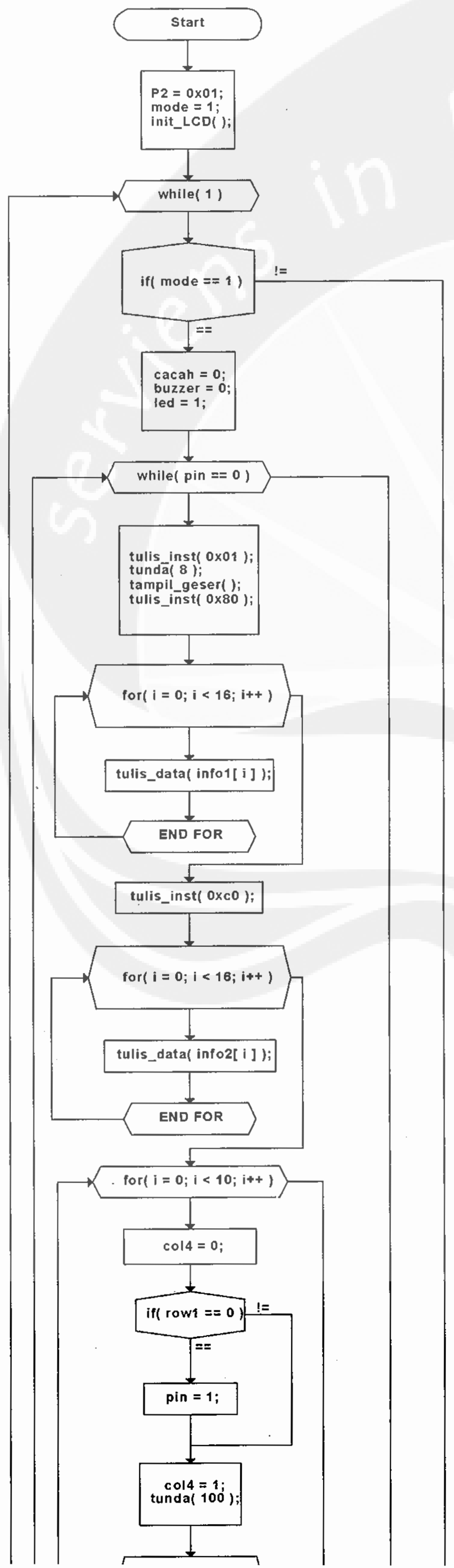
Lampiran 4 : Gambar *ISP* (*In-System Programming*) kabel

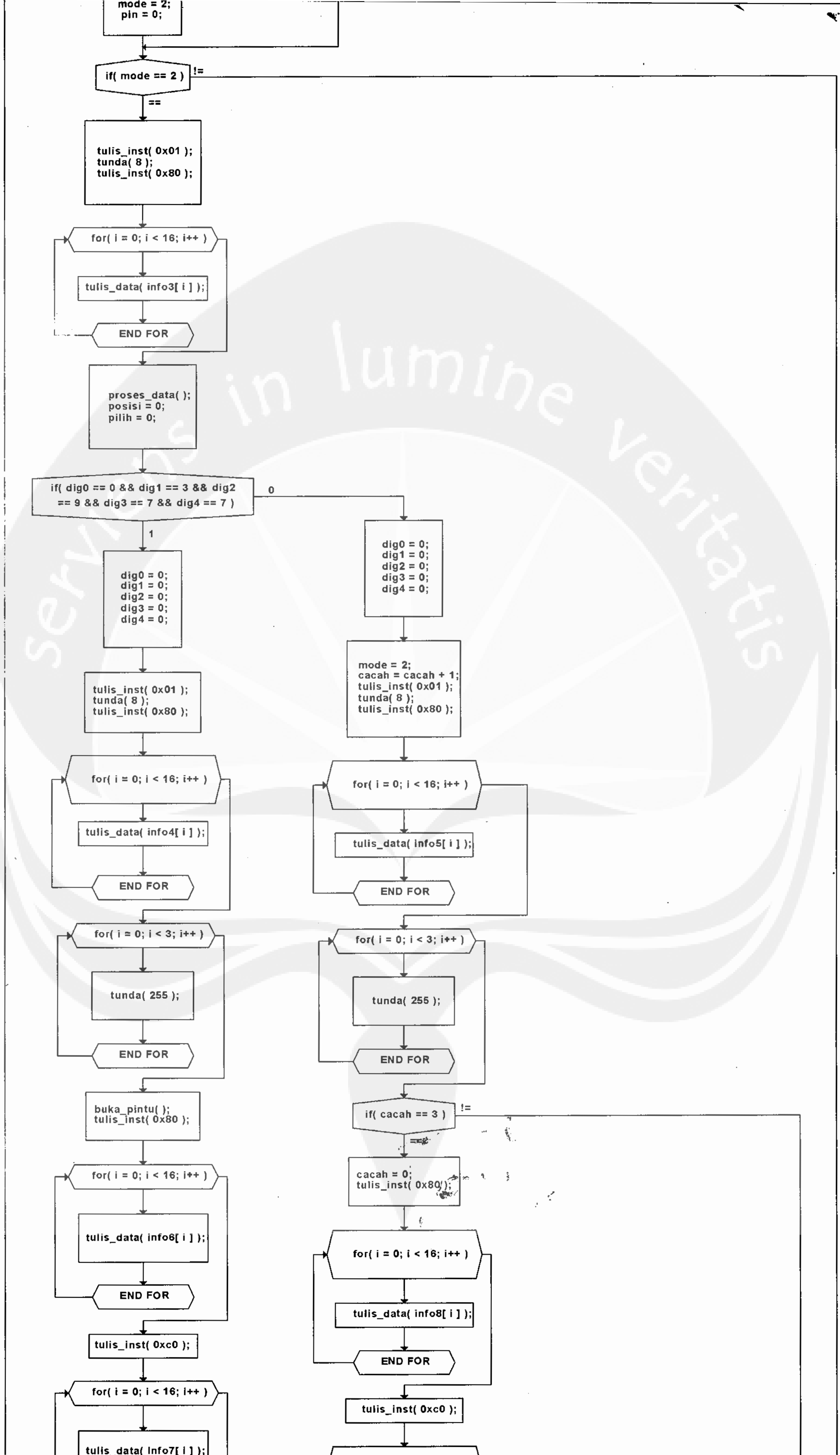


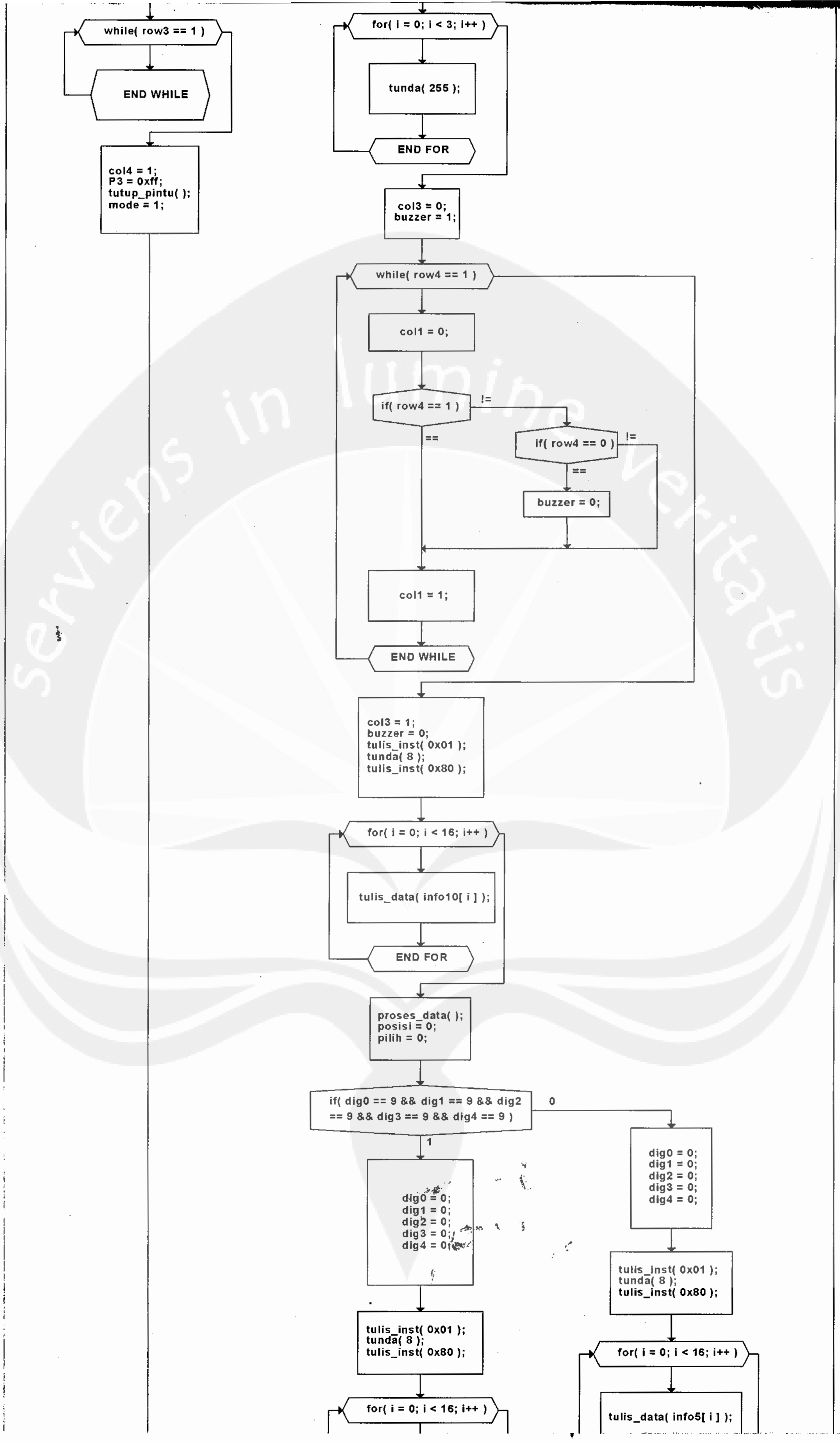
Konfigurasi Pin ISP kabel

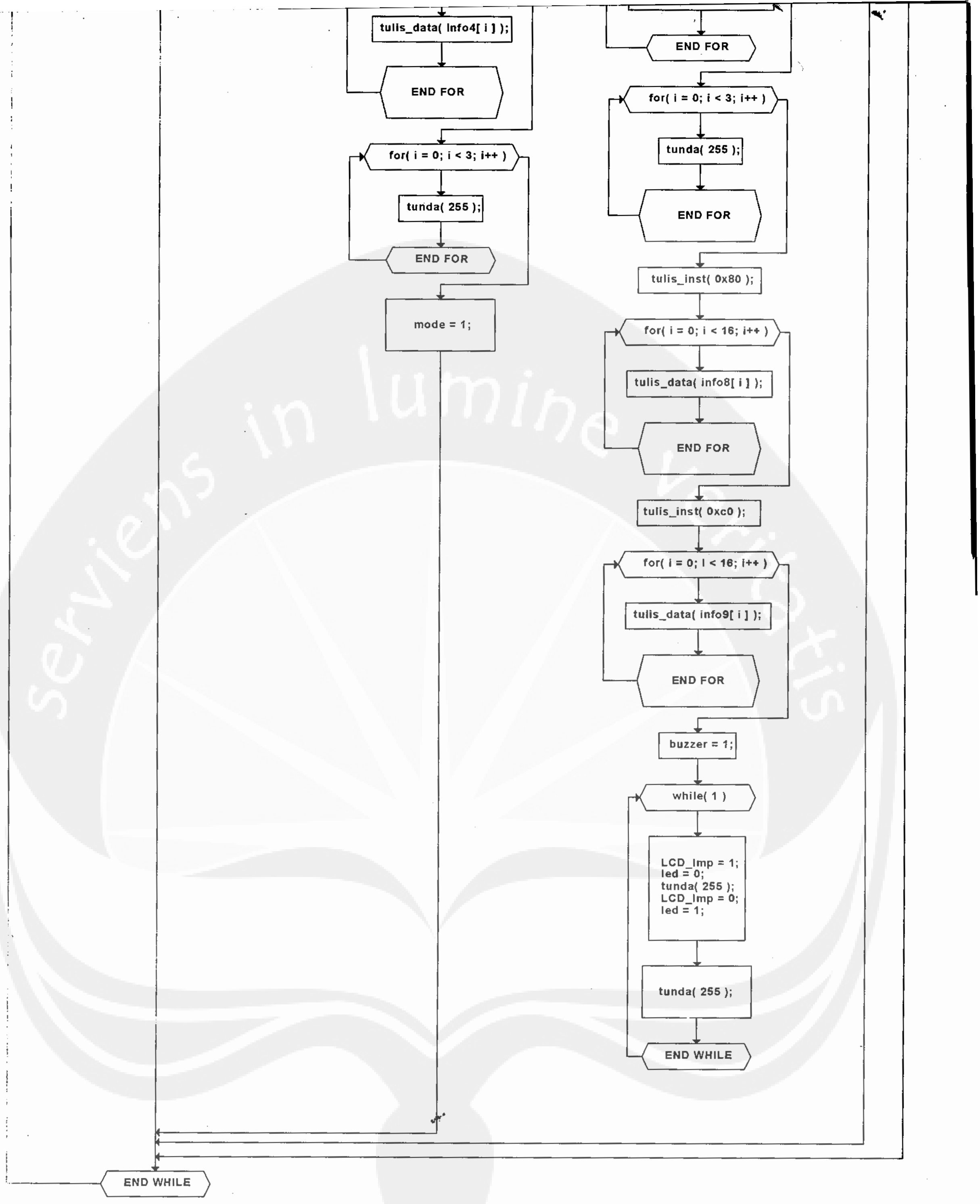
DB 9	DB 25
Pin 1	Pin 7
Pin 2	Pin 8
Pin 3	Pin 6
Pin 5	Pin 10
Pin 6-9	Pin 18-25

Lampiran 5: *Flowchart of Program List*




mode = 2;
pin = 0;
if(mode == 2)
!=
==
tulis_inst(0x01);
tunda(8);
tulis_inst(0x80);
for(i = 0; i < 16; i++)
tulis_data(info3[i]);
END FOR
proses_data( );
posisi = 0;
pilih = 0;
if(dig0 == 0 && dig1 == 3 && dig2 == 9 && dig3 == 7 && dig4 == 7)
0
1
dig0 = 0;
dig1 = 0;
dig2 = 0;
dig3 = 0;
dig4 = 0;
dig0 = 0;
dig1 = 0;
dig2 = 0;
dig3 = 0;
dig4 = 0;
tulis_inst(0x01);
tunda(8);
tulis_inst(0x80);
mode = 2;
cacah = cacah + 1;
tulis_inst(0x01);
tunda(8);
tulis_inst(0x80);
for(i = 0; i < 16; i++)
tulis_data(info4[i]);
END FOR
for(i = 0; i < 16; i++)
tulis_data(info5[i]);
END FOR
for(i = 0; i < 3; i++)
tunda(255);
END FOR
for(i = 0; i < 3; i++)
tunda(255);
END FOR
buka_pintu( );
tulis_inst(0x80);
if(cacah == 3)
!=
==
cacah = 0;
tulis_inst(0x80);
for(i = 0; i < 16; i++)
tulis_data(info6[i]);
END FOR
for(i = 0; i < 16; i++)
tulis_data(info8[i]);
END FOR
tulis_inst(0xc0);
tulis_inst(0xc0);
for(i = 0; i < 16; i++)
tulis_data(info7[i]);


while(row3 == 1)
END WHILE
col4 = 1;
P3 = 0xff;
tutup_pintu( );
mode = 1;
for(i = 0; i < 3; i++)
tunda(255);
END FOR
col3 = 0;
buzzer = 1;
while(row4 == 1)
col1 = 0;
if(row4 == 1)
!=
==
if(row4 == 0)
!=
==
buzzer = 0;
col1 = 1;
END WHILE
col3 = 1;
buzzer = 0;
tulis_inst(0x01);
tunda(8);
tulis_inst(0x80);
for(i = 0; i < 16; i++)
tulis_data(info10[i]);
END FOR
proses_data( );
posisi = 0;
pilih = 0;
if(dig0 == 9 && dig1 == 9 && dig2 == 9 && dig3 == 9 && dig4 == 9)
0
1
dig0 = 0;
dig1 = 0;
dig2 = 0;
dig3 = 0;
dig4 = 0;
dig0 = 0;
dig1 = 0;
dig2 = 0;
dig3 = 0;
dig4 = 0;
tulis_inst(0x01);
tunda(8);
tulis_inst(0x80);
tulis_inst(0x01);
tunda(8);
tulis_inst(0x80);
for(i = 0; i < 16; i++)
for(i = 0; i < 16; i++)
tulis_data(info5[i]);



tulis_data(Info4[i]);
END FOR
for(i = 0; i < 3; i++)
tunda(255);
END FOR
mode = 1;
END FOR
for(i = 0; i < 3; i++)
tunda(255);
END FOR
tulis_inst(0x80);
for(i = 0; i < 16; i++)
tulis_data(info8[i]);
END FOR
tulis_inst(0xc0);
for(i = 0; i < 16; i++)
tulis_data(info9[i]);
END FOR
buzzer = 1;
while(1)
LCD_lmp = 1;
led = 0;
tunda(255);
LCD_lmp = 0;
led = 1;
tunda(255);
END WHILE
END WHILE

Lampiran 6 :

Program List Kunci Elektronik

```
/*
            Program pengendali pintu dengan kode PIN
       menggunakan keypad 4x4 dan LCD 16x2 mode data 8-bit
Bahasa pemograman = bahasa C, Compiler = Small Device C Compiler
*/

#include <at89x51.h>         //header jenis mikrokontroler AT89x51
//-----------------------------------------------------
#define  LCD_data     P0          //inisialisasi nama (alias) register
#define  LCD_lmp      P2_0
#define  LCD_Rs       P2_1
#define  LCD_RW       P2_2
#define  LCD_E        P2_3

#define  row1         P3_0
#define  row2         P3_1
#define  row3         P3_2
#define  row4         P3_3
#define  col1         P3_4
#define  col2         P3_5
#define  col3         P3_6
#define  col4         P3_7

#define  buzzer       P2_4
#define  close_door   P2_5
#define  open_door    P2_6

#define  limit_open   P1_1
#define  limit_close  P1_2
#define  led          P1_3
//-----------------------------------------------------
bit at   0x20       pilih;        //alokasi register RAM 1bit
bit at   0x21       batal;
bit at   0x22       pin;
bit at   0x23       puk;
bit at   0x24       buzz;
//-----------------------------------------------------
data unsigned char mode,nilai,posisi,cacah,   //alokasi register RAM 8bit
                   dig0,dig1,dig2,dig3,dig4;
//-----------------------------------------------------
code unsigned char tampil0[16]  ={"                "};  //alokasi memori
untuk database tampilan
code unsigned char tampil1[16]  ={"               S"};  //pada memori FLASH
code unsigned char tampil2[16]  ={"              SE"};
code unsigned char tampil3[16]  ={"             SEL"};
code unsigned char tampil4[16]  ={"            SELA"};
code unsigned char tampil5[16]  ={"           SELAM"};
code unsigned char tampil6[16]  ={"          SELAMA"};
code unsigned char tampil7[16]  ={"         SELAMAT"};
code unsigned char tampil8[16]  ={"        SELAMAT "};
code unsigned char tampil9[16]  ={"       SELAMAT D"};
code unsigned char tampil10[16] ={"      SELAMAT DA"};
code unsigned char tampil11[16] ={"     SELAMAT DAT"};
code unsigned char tampil12[16] ={"    SELAMAT DATA"};
code unsigned char tampil13[16] ={"   SELAMAT DATAN"};
code unsigned char tampil14[16] ={"  SELAMAT DATANG"};
code unsigned char tampil15[16] ={" SELAMAT DATANG "};
code unsigned char tampil16[16] ={"SELAMAT DATANG  "};
code unsigned char tampil17[16] ={"ELAMAT DATANG   "};
code unsigned char tampil18[16] ={"LAMAT DATANG    "};
code unsigned char tampil19[16] ={"AMAT DATANG     "};
code unsigned char tampil20[16] ={"MAT DATANG      "};
code unsigned char tampil21[16] ={"AT DATANG       "};
code unsigned char tampil22[16] ={"T DATANG        "};
code unsigned char tampil23[16] ={" DATANG         "};
code unsigned char tampil24[16] ={"DATANG          "};
code unsigned char tampil25[16] ={"ATANG           "};
code unsigned char tampil26[16] ={"TANG            "};
code unsigned char tampil27[16] ={"ANG             "};
code unsigned char tampil28[16] ={"NG              "};
code unsigned char tampil29[16] ={"G               "};
code unsigned char tampil30[16] ={"                "};

code unsigned char info1[16]    ={"  MASUKKAN PIN  "};
code unsigned char info2[16]    ={"   TEKAN PIN    "};
code unsigned char info3[16]    ={"  PIN  5 DIGIT  "};
code unsigned char info4[16]    ={"   PIN BENAR    "};
code unsigned char info5[16]    ={"   PIN SALAH    "};
code unsigned char info6[16]    ={"  TUTUP PINTU   "};
code unsigned char info7[16]    ={"  TEKAN TUTUP   "};
code unsigned char info8[16]    ={"     SISTEM     "};
code unsigned char info9[16]    ={"    TERKUNCI    "};
code unsigned char info10[16]   ={"  MASUKKAN PIN2 "};
//-------------------------------------------------------
void tunda(unsigned char val)       //sub rutin tundaan
{
  unsigned char i,j;
  for(i=0;i<255;i++)                //tundaan selama 255 x val(variabel)
siklus
  {for(j=0;j<val;j++);}
}
//-------------------------------------------------------
void tulis_inst(unsigned char cmd)  //sub rutin menuliskan instruksi pada
LCD 16x2
{
  unsigned int i;
  LCD_Rs=0;LCD_E=1;LCD_data=cmd;    //pin kendali LCD pada mode tulis
instruksi
  for(i=0;i<10;i++);                //lebar pulsa handshake kendali LCD
  LCD_E=0;
}
//-------------------------------------------------------
void tulis_data(unsigned char chr)  //sub rutin menuliskan instruksi pada
LCD 16x2
{
  unsigned int i;
  LCD_Rs=1;LCD_E=1;LCD_data=chr;    //pin kendali LCD pada mode tulis data
  for(i=0;i<10;i++);                //lebar pulsa handshake kendali LCD
  LCD_E=0;
}
//-------------------------------------------------------
void init_LCD()          //sub rutin inisialisasi LCD 16x2
{
  tulis_inst(0x38);tunda(20);     //reset LCD step-1
  tulis_inst(0x38);tunda(1);      //reset LCD step-2
  tulis_inst(0x38);tunda(1);      //reset LCD step-3
  tulis_inst(0x38);tunda(1);      //mode LCD: lebar data 8-bit, 2-row dan
5x7 dot
  tulis_inst(0x06);tunda(1);      //mode LCD: penambahan 1 & tidak
bergerser
  tulis_inst(0x0c);tunda(1);      //mode LCD: display hidup, kursor dan
kedip padam
  tulis_inst(0x01);tunda(8);      //layar bersih
}
```

```
//-------------------------------------------------------
void beep()
{
  buzzer=1;tunda(4);buzzer=0;
}
//-------------------------------------------------------
void load()          //sub rutin pengolah nilai dari masukan keypad 4x4
{
  if(posisi==0)          //alokasi register untuk digit kode ke-1
  {
    tulis_inst(0xc5);       //tampil pada row-2 col-6
    tulis_data('*');        //menampilkan tanda '*' pada LCD
    dig0=nilai;posisi=posisi+1;     //simpan nilai
  }
  else if(posisi==1)        //alokasi register untuk digit kode ke-2
  {
    tulis_inst(0xc6);       //tampil pada row-2 col-7
    tulis_data('*');        //menampilkan tanda '*' pada LCD
    dig1=nilai;posisi=posisi+1;     //simpan nilai
  }
  else if(posisi==2)        //alokasi register untuk digit kode ke-
  {
    tulis_inst(0xc7);       //tampil pada row-2 col-8
    tulis_data('*');        //menampilkan tanda '*' pada LCD
    dig2=nilai;posisi=posisi+1;     //simpan nilai
  }
  else if(posisi==3)        //alokasi register untuk digit kode ke-
  {
    tulis_inst(0xc8);       //tampil pada row-2 col-9
    tulis_data('*');        //menampilkan tanda '*' pada LCD
    dig3=nilai;posisi=posisi+1;     //simpan nilai
  }
  else if(posisi==4)        //alokasi register untuk digit kode ke-
  {
    tulis_inst(0xc9);       //tampil pada row-2 col-10
    tulis_data('*');        //menampilkan tanda '*' pada LCD
    dig4=nilai;posisi=posisi+1;     //simpan nilai
  }
  else{;}
}
//-------------------------------------------------------
void scan_keypad()        //sub rutin scanning keypad 4x4
{
  col1 = 0;          //scanning pada col1
  if(row1==0)
  {beep();nilai=1;load();while(row1==0){}}    //angka '1'
  else if(row2==0)
  {beep();nilai=4;load();while(row2==0){}}    //angka '4'
  else if(row3==0)
  {beep();nilai=7;load();while(row3==0){}}    //angka '7'
  else if(row4==0)
  {beep();buzz=1;while(row4==0){}}            //kode '*'
  else{;}P3 = 0xff;

  col2 = 0;          //scanning pada col2
  if(row1==0)
  {beep();nilai=2;load();while(row1==0){}}    //angka '2'
  else if(row2==0)
  {beep();nilai=5;load();while(row2==0){}}    //angka '5'
  else if(row3==0)
  {beep();nilai=8;load();while(row3==0){}}    //angka '8'
  else if(row4==0)
  {beep();nilai=0;load();while(row4==0){}}     //angka '0'
  else{;}P3 = 0xff;

  col3 = 0;          //scanning pada col3
  if(row1==0)
  {beep();nilai=3;load();while(row1==0){}}     //angka '3'
  else if(row2==0)
  {beep();nilai=6;load();while(row2==0){}}     //angka '6'
  else if(row3==0)
  {beep();nilai=9;load();while(row3==0){}}     //angka '9'
  else if(row4==0)
  {beep();puk=1;while(row4==0){}}              //kode '#'
  else{;}P3 = 0xff;

  col4 = 0;          //scanning pada col4
  if(row1==0)
  {beep();while(row1==0){}}                   //huruf 'PIN'
  else if(row2==0)
  {beep();batal=1;while(row2==0){}}           //huruf 'HAPUS'
  else if(row3==0)
  {beep();while(row3==0){}}                   //huruf 'TUTUP'
  else if(row4==0)
  {beep();pilih=1;while(row4==0){}}           //huruf 'BENAR'
  else{;}P3 = 0xff;
}
//-----------------------------------------------------
void proses_data()        //sub rutin pengendali masukkan nilai keypad 4x4
{
  unsigned char i;
  while(pilih==0)                             //mendeteksi penekanan tombol
'BENAR'
  {
    scan_keypad();                            //sub rutin scanning keypad 4x4
    if(batal==1)                              //pendeteksian tombol 'HAPUS'
    {
      dig0=0;dig1=0;dig2=0;dig3=0;dig4=0; //membersihkan register password
      posisi=0;batal=0;                       //mengembalikan pd kondisi
default
      tulis_inst(0xc0);
      for(i=0;i<16;i++)                      //menghapus tampilan pada row ke-
2 LCD
      {tulis_data(' ');}
    }
    tunda(1);
  }
}
//-----------------------------------------------------
void buka_pintu()       //sub rutin membuka pintu
{
  close_door=0;open_door=1;led=0;   //motor pengendali pintu aktif membuka
dan led menyala
  while(limit_open==1){}            //apakah sudah membuka maksimal???
  close_door=0;open_door=0;led=1;   //jika ya, motor pengendali pintu
dipadamkan dan led padam
}
//-----------------------------------------------------
void tutup_pintu()        //sub rutin menutup pintu
{
  close_door=1;open_door=0;led=0;   //motor pengendali pintu aktif menutup
dan led menyala
  while(limit_close==1){}           //apakah sudah menutup minimal???
  close_door=0;open_door=0;led=1;   //jika ya, motor pengendali pintu
dipadamkan dan led padam
```

```
}
//-----------------------------------------------------
void tampil_geser()       //sub rutin menampilkan tampilan bergeser
{
  unsigned char i;
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil0[i]);}      //tampilkan frame ke-1
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil1[i]);}      //tampilkan frame ke-2
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil2[i]);}      //tampilkan frame ke-3
  tunda(100)
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil3[i]);}      //tampilkan frame ke-4
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil4[i]);}      //tampilkan frame ke-5
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil5[i]);}      //tampilkan frame ke-6
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil6[i]);}      //tampilkan frame ke-7
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil7[i]);}      //tampilkan frame ke-8
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil8[i]);}      //tampilkan frame ke-9
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil9[i]);}      //tampilkan frame ke-10
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil10[i]);}     //tampilkan frame ke-11
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil11[i]);}     //tampilkan frame ke-12
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil12[i]);}     //tampilkan frame ke-13
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil13[i]);}     //tampilkan frame ke-14
  tunda(100);
  tulis_inst(0x80);
```

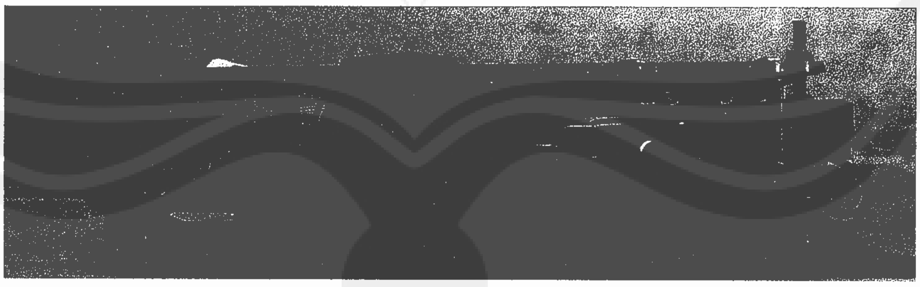
```
for(i=0;i<16;i++)
{tulis_data(tampil14[i]);}     //tampilkan frame ke-15
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil15[i]);}     //tampilkan frame ke-16
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil16[i]);}     //tampilkan frame ke-17
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil17[i]);}     //tampilkan frame ke-18
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil18[i]);}     //tampilkan frame ke-19
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil19[i]);}     //tampilkan frame ke-20
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil20[i]);}     //tampilkan frame ke-21
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil21[i]);}     //tampilkan frame ke-22
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil22[i]);}     //tampilkan frame ke-23
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil23[i]);}     //tampilkan frame ke-24
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil24[i]);}     //tampilkan frame ke-25
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil25[i]);}     //tampilkan frame ke-26
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil26[i]);}     //tampilkan frame ke-27
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil27[i]);}     //tampilkan frame ke-28
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil28[i]);}     //tampilkan frame ke-29
tunda(100);
tulis_inst(0x80);
for(i=0;i<16;i++)
{tulis_data(tampil29[i]);}     //tampilkan frame ke-30
```

```
  tunda(100);
  tulis_inst(0x80);
  for(i=0;i<16;i++)
  {tulis_data(tampil30[i]);}    //tampilkan frame ke-31
}
//-----------------------------------------------------
void main()           //program utama pengendali sistem
{
  unsigned char i;
  P2=0x01;mode=1;        //pengkondisian awal
  init_LCD();            //melakukan inisialisasi LCD 16x2
  while(1)
  {
    if(mode==1)
    {
      cacah=0;buzzer=0;led=1;    //mengembalikan kondisi register fungsi
menuju awal (0)
      while(pin==0)
      {
        tulis_inst(0x01);tunda(8);   //bersihkan layar
        tampil_geser();              //menampilkan tampilan bergeser
'SELAMAT DATANG'

        tulis_inst(0x80);
        for(i=0;i<16;i++)            //mengambil 16 karakter tampilan
        {tulis_data(info1[i]);}      //menampilkan informasi 'MASUKKAN PIN'
        tulis_inst(0xc0);
        for(i=0;i<16;i++)            //mengambil 16 karakter tampilan
        {tulis_data(info2[i]);}      //menampilkan informasi 'TEKAN PIN'
        for(i=0;i<10;i++)
        {
          col4=0;                    //memilih larik tombol pada col4
          if(row1==0){pin=1;}        //mendeteksi penekanan tombol 'PIN'
          col4=1;                    //mengembalikan pada kondisi default
          tunda(100);
        }
      }
      mode=2;pin=0;                  //mengubah dari mode 1 menuju mode 2
    }
    if(mode==2)
    {
      tulis_inst(0x01);tunda(8);     //bersihkan layar
      tulis_inst(0x80);
      for(i=0;i<16;i++)              //mengambil 16 karakter tampilan
      {tulis_data(info3[i]);}        //menampilkan informasi 'PIN 5 DIGIT'

      proses_data();                 //mengambil nilai dari masukkan
keypad 4x4

      posisi=0;pilih=0;
      if(dig0==0 && dig1==3 && dig2==9 && dig3==7 && dig4==7) //jika kode
benar?? maka...
      {
        dig0=0;dig1=0;dig2=0;dig3=0;dig4=0;

        tulis_inst(0x01);tunda(8);     //bersihkan layar
        tulis_inst(0x80);
        for(i=0;i<16;i++)              //mengambil 16 karakter tampilan
        {tulis_data(info4[i]);}        //menampilkan informasi 'PIN BENAR!'
        for(i=0;i<3;i++){tunda(255);}

        buka_pintu();                  //sub rutin membuka pintu
        tulis_inst(0x80);
        for(i=0;i<16;i++)                   //mengambil 16 karakter tampilan
        {tulis_data(info6[i]);}             //menampilkan informasi 'TUTUP PINTU'
        tulis_inst(0xc0);
        for(i=0;i<16;i++)                   //mengambil 16 karakter tampilan
        {tulis_data(info7[i]);}             //menampilkan informasi 'TEKAN TUTUP'


        P3=0xff;col4=0;                     //memilih larik tombol pada col4
        while(row3==1){}                    //mendeteksi penekanan tombol 'TUTUP'
        col4=1;P3=0xff;

        tutup_pintu();                      //sub rutin menutup pintu
        mode=1;                             //mengubah dari mode 2 kembali
menuju mode 1
      }
      else
      {
        dig0=0;dig1=0;dig2=0;dig3=0;dig4=0;

        mode=2;
        cacah=cacah+1;
        tulis_inst(0x01);tunda(8);          //bersihkan layar
        tulis_inst(0x80);
        for(i=0;i<16;i++)                   //mengambil 16 karakter tampilan
        {tulis_data(info5[i]);}             //menampilkan informasi 'PIN SALAH!'
        for(i=0;i<3;i++){tunda(255);}

        if(cacah==3)                        //mendeteksi kesalahan selama 3x
        {
          cacah=0;
          tulis_inst(0x80);
          for(i=0;i<16;i++)                 //mengambil 16 karakter tampilan
          {tulis_data(info8[i]);}           //menampilkan informasi 'SISTEM'
          tulis_inst(0xc0);
          for(i=0;i<16;i++)                 //mengambil 16 karakter tampilan
          {tulis_data(info9[i]);}           //menampilkan informasi 'TERKUNCI'
          for(i=0;i<3;i++){tunda(255);}

          col3=0;buzzer=1;                 //kesalahan 3x mengakibatkan buzzer
nyala
          while(row4==1)
          {
            col1=0;
      if(row4==1){}       //mendeteksi penekanan tombol '*'
      else if(row4==0){buzzer=0;}   //jika ditekan, buzzer padam
      col1=1;
    }
    col3=1;buzzer=0;                 //jika tombol '#' ditekan, buzzer ikut
padam

          tulis_inst(0x01);tunda(8);        //bersihkan layar
          tulis_inst(0x80);
          for(i=0;i<16;i++)                 //mengambil 16 karakter tampilan
          {tulis_data(info10[i]);}          //menampilkan informasi 'MASUKKAN
PIN2'

    proses_data();                          //mengambil nilai dari masukkan
keypad 4x4

    posisi=0;pilih=0;
```

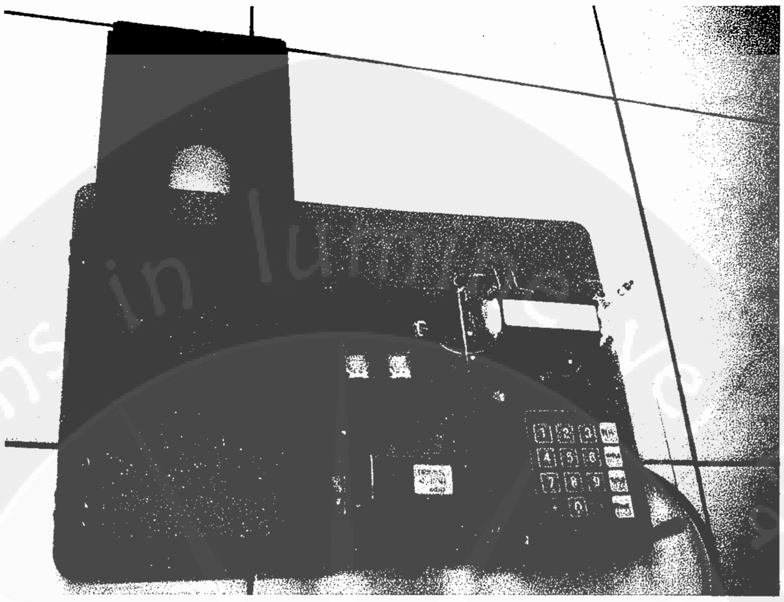
```
          if(dig0==9 && dig1==9 && dig2==9 && dig3==9 && dig4==9) //jika
kode benar??? maka...
          {
            dig0=0;dig1=0;dig2=0;dig3=0;dig4=0;

            tulis_inst(0x01);tunda(8);    //bersihkan layar
            tulis_inst(0x80);
            for(i=0;i<16;i++)             //mengambil 16 karakter tampilan
            {tulis_data(info4[i]);}       //menampilkan informasi 'PIN
BENAR!'
            for(i=0;i<3;i++){tunda(255);}

            mode=1;                       //mengubah dari mode 2 kembali
menuju mode 1
      }
      else
      {
        dig0=0;dig1=0;dig2=0;dig3=0;dig4=0;

        tulis_inst(0x01);tunda(8);      //bersihkan layar
            tulis_inst(0x80);
            for(i=0;i<16;i++)           //mengambil 16 karakter tampilan
            {tulis_data(info5[i]);}     //menampilkan informasi 'PIN SALAH!'
            for(i=0;i<3;i++){tunda(255);}

            tulis_inst(0x80);
            for(i=0;i<16;i++)           //mengambil 16 karakter tampilan
            {tulis_data(info8[i]);}     //menampilkan informasi 'SISTEM'
            tulis_inst(0xc0);
            for(i=0;i<16;i++)           //mengambil 16 karakter tampilan
            {tulis_data(info9[i]);}     //menampilkan informasi 'TERKUNCI'

            buzzer=1;                   //buzzer menyala terus hingga sistem
di-reset

        while(1)                        //terjebak di sini!!!
        {
          LCD_lmp=1;led=0;tunda(255);   //dengan led & lampu LCD berkedip-
kedip
          LCD_lmp=0;led=1;tunda(255);
        }
      }
    }
        }
      }
    }
}
```

Lampiran 7 : Gambar Kunci Elektronik

Gambar 1. Kunci Elektronik Tampak Depan



Gambar 2. Kunci Elektronik Tampak Samping


Gambar 3. Membuka Pintu

Gambar 4. Menutup Pintu