TOSHIBA CMOS Digital Integrated Circuit Silicon Monolithic

T6C96A

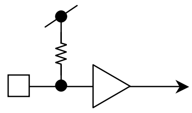
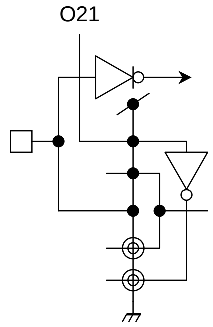
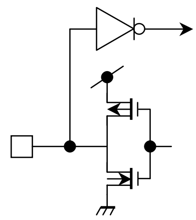
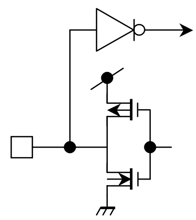
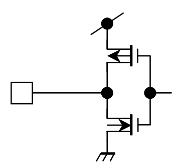
Overview

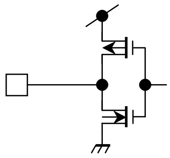
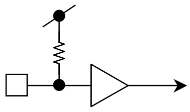
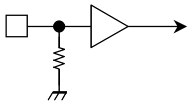
The T6C96A is an advanced-function 4-bit single-chip microcontroller that integrates ROM, work RAM, I/O ports, timers and LCD driver, etc., into one chip. The T6C96A features low power consumption.

Features

- ROM: 4096 words (1 words = 14 bit)
- Work RAM: 1024 bits (16 × 16 × 4 bit)
- Subroutine nesting level: 4 levels
- Timers: 500 ms·125 ms·31.25 ms (× 1ch each)
- IN ports: 4 bit
- IO ports: 8 bit
- IO ports for data: 1 bit (tristate I/O)
- O ports: 4 bit
- Output ports for clock: 1 bit (high-speed clock output pins)
- Output port
- External device control pin: None
- Display types: 24-segment × 4 common
  1/4 duty, 1/3 pre-bias
- Minimum instruction execution time: 2.23 μs (High-speed clock 3.58 MHz)
  244 μs (Low-speed clock 32.768 kHz)
- Power-saving functions: High-speed operation mode
  Low-speed operation mode
  STOP mode
  OFF mode

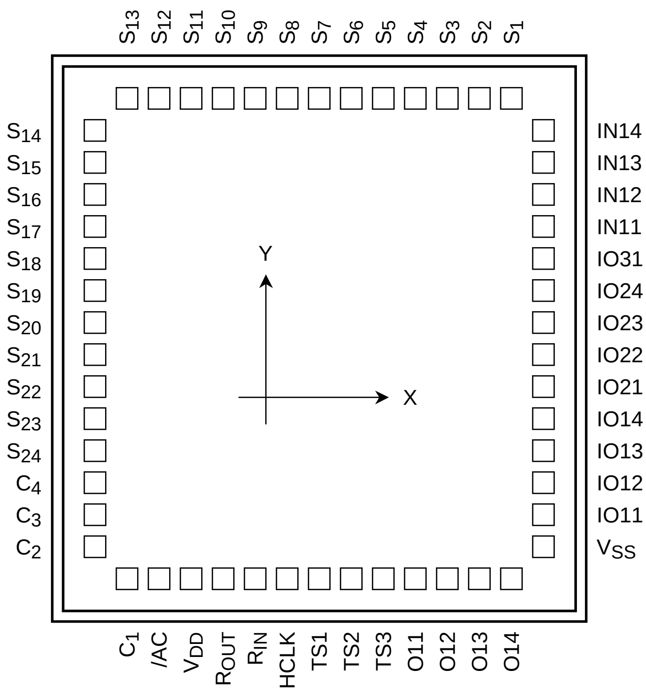
Pin Functions

Pin name	I/O	Function	Circuit configuration	Initial
IN11~IN14	IN	4-bit input ports		H
IO11	IO	I/O ports with output latch 3-state output buffer	 3-state output	L
IO12	IO	I/O ports with output port	 P < N	H
IO13~IO14 IO21~IO24 IO31	IO	I/O ports with output latch	 P < N	L
O11~O14	O	Output port with latch	 push pull output	L

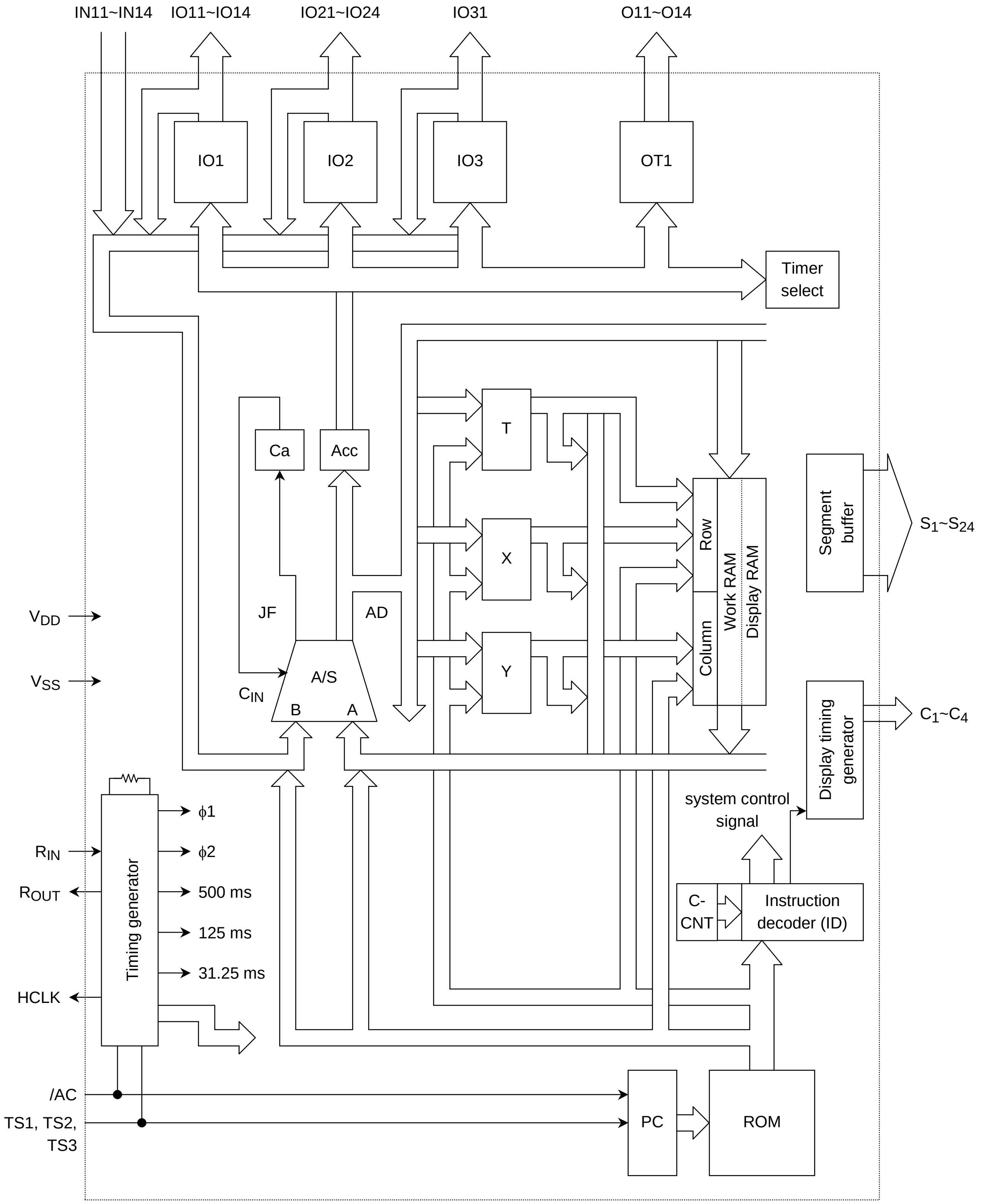
Pin name	I/O	Function	Circuit configuration	Initial
HCLK	O	Output the inverted waveform of the high-speed clock or the waveform of the high-speed clock divided in harf.	 Push-pull output	L
R_{IN} R_{OUT}	IN O	Terminals for connecting oscillation resistor for high-speed clock oscillator	—	—
/AC	IN	System reset terminal		H
TS1~TS3	IN	Used by Toshiba for shipping test. Fix the level of this terminal “L”.		L
V_{DD} V_{SS}	Power supply	4.5 V~5.5 V 0 V (GND)	—	—
S_1~S_{24}	O	Segment signal output terminal	—	(*)
C_1~C_4	O	Common signal output terminal	—	(*)

Total 54 pin
P: P-ch Tr output ability
N: N-ch Tr output ability
(*) High level when display is OFF

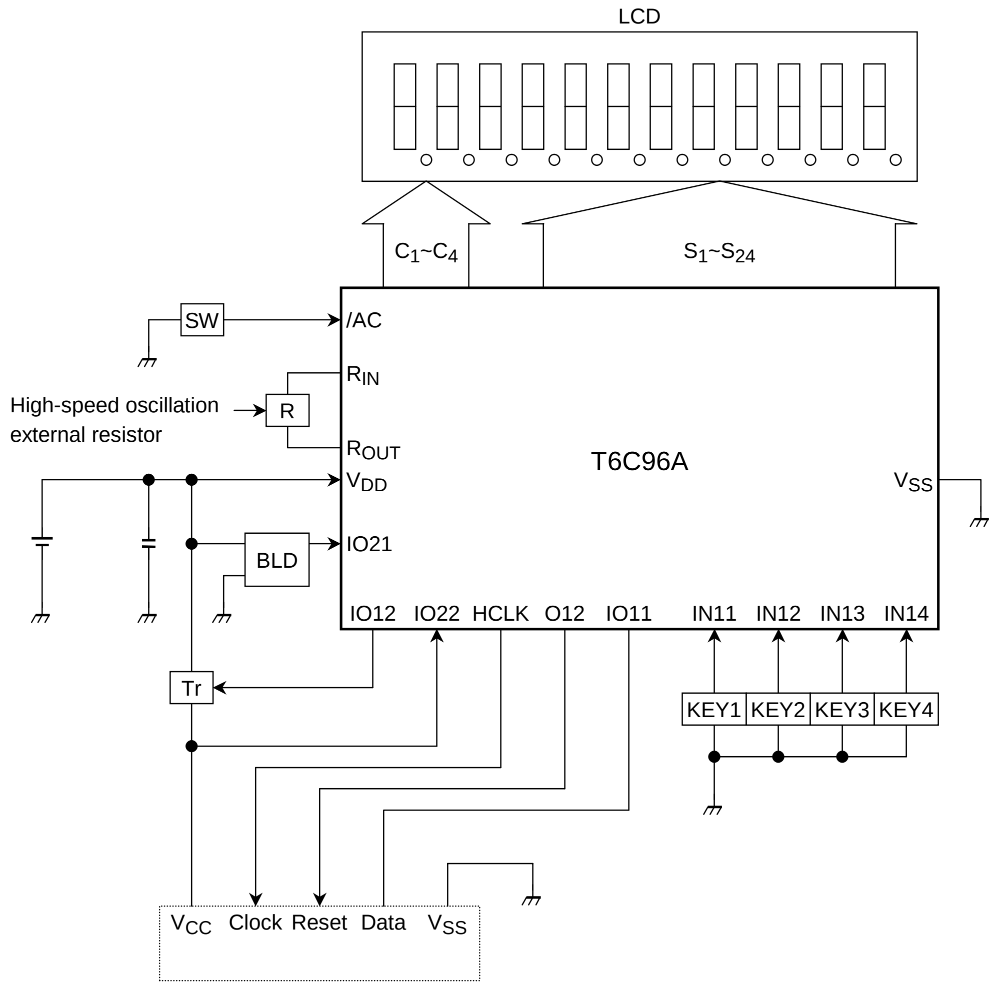
Pad Assignment



Block Diagram



Example of T6C96A IC Card Reader System



Use of an external BLD allows you to monitor the IC card's supply voltage of T6C96A.
IO11 is 3-state type so as to use it as data line.
HCLK is a supply control terminal from the driver to the IC card.

Internal CPU Functions

Program Memory (ROM)

Instruction codes are stored in the program memory. The instruction next in line to be executed is read from the address indicated by the content of the program counter. Each word in program memory consists of 14 bits, and each instruction consists of one word.

Figure 1 shows the program memory map.

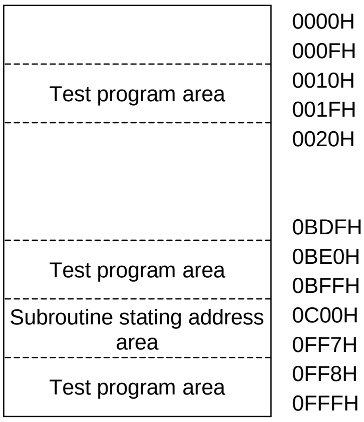


Figure 1 Program Memory Map

When creating a user program, note that a specific area is occupied by the test program used by Toshiba for its shipping tests.

Test Program

The T6C96A requires the following test programs to be included with user programs. Note that the following test program areas are not available to users.

address	object			
			ORG	0BE0H
0BE0	091E		OIB1	14
0BE1	092F		OIB2	15
0BE2	093C		OIB3	12
0BE3	094D		OTB1	13
0BE4	095A		OTB2	10
0BE5	096B		OTB3	11
0BE6	0030	ROM1	ROM	
0BE7	0000		NOP	
0BE8	0001		CAR	
0BE9	3BF6		JUMP	ROM2
0BEA	0914		OIB1	4
0BEB	0925		OIB2	5
0BEC	0932		OIB3	2
0BED	0943		OTB1	3
0BEE	0950		OTB2	0
0BEF	0961		OTB3	1
0BF0	095F		OTB2	15
0BF1	093D		OIB3	13
0BF2	096E		OTB3	14
0BF3	091B		OIB1	11
0BF4	094C		OTB1	12
0BF5	0955		OTB2	5
0BF6	0030	ROM2	ROM	
0BF7	0000		NOP	
0BF8	0011		CAS	
0BF9	3BE6		JUMP	ROM1
0BFA	092A		OIB2	10
0BFB	0933		OIB3	3
0BFC	0964		OTB3	4
0BFD	0911		OIB1	1
0BFE	0942		OTB1	2
0BFF	0920		OIB2	0

address	object		
		ORG	0010H
0010	0000	NOP	
0011	0000	NOP	
0012	0000	NOP	
0013	0000	NOP	
0014	0000	NOP	
0015	0000	NOP	
0016	0000	NOP	
0017	0000	NOP	
0018	0000	NOP	
0019	0000	NOP	
001A	0000	NOP	
001B	0000	NOP	
001C	0000	NOP	
001D	0000	NOP	
001E	0000	NOP	
001F	0000	NOP	
		ORG	0FF8H
0FF8	0000	NOP	
0FF9	0000	NOP	
0FFA	0000	NOP	
0FFB	0000	NOP	
0FFC	0000	NOP	
0FFD	0000	NOP	
0FFE	0000	NOP	
0FFF	0000	NOP	
		;	
		END	

Program Counter

The program counter, which is a 12-bit binary counter (Figure 2), indicates the address in program memory of the next instruction to be executed.

The program counter is normally incremented each time an instruction is executed. When the count reaches 0FFFH, it is reset to 000H.

When the CPU is reset, the program counter and page register are both initialized to "1".

Figure 2 shows the structure of the program counter.

Program counter

PC 11	PC 10	PC 9	PC 8	PC 7	PC 6	PC 5	PC 4	PC 3	PC 2	PC 1	PC 0

Figure 2 Program Counter

Table 1 is a table of change in the program counter.

Table 1 Table of Change in Program Counter

Instruction and operation		Condition	PC 11	PC 10	PC 9	PC 8	PC 7	PC 6	PC 5	PC 4	PC 3	PC 2	PC 1	PC 0
Executed instruction	JUMP adr	JF = 0	Address (adr) specified in instruction											
	CALLn adr		1	1	Address (adr) specified in instruction									
	RETn		Address stored in work RAM by corresponding CALLn instruction											
Address following 0FFFH			0	0	0	0	0	0	0	0	0	0	0	0
Reset			1	1	1	1	1	1	1	1	1	1	1	1

The program counter can directly specify addresses throughout the whole address space of the program memory. However, care is required when using the following branching instructions.

(1) Jump Instruction [JUMP adr]

When JUMP is executed, the value (adr) specified in the instruction is set in the program counter (PC11 to PC0) if the jump flag (JF) is "0".

(2) Subroutine Instructions [CALLn adr] and [RETn]

When [CALLn adr] is executed, the value of [program counter content +1] are stored in a predetermined area of work RAM as the subroutine return address, and the value (adr) specified in the instruction is set in the lower 10 bits (PC9 to PC0) of the program counter. At this point, the upper 2 bits of the program counter (PC11 and PC10) are each set to "1". That is, the starting address of the subroutine must be between 0C00H and 0FFFH of the last page.

When [RETn] is executed, execution returns to the address stored using the [CALLn adr] instruction corresponding to n.

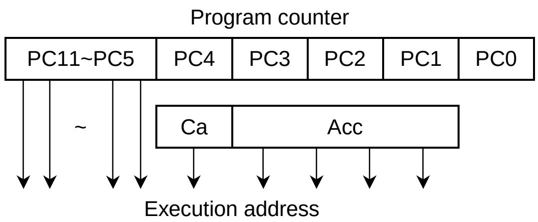
Subroutines can be nested up to 4 levels.

(3) One-Step Branch Instruction [ROM]

This instruction temporarily changes the output of the program counter. After [ROM] instruction is executed, the instruction following by one is executed. The instruction doesn't access the program memory directly by the contents of the program counter, but temporarily replaces the 5 th bit from the LSB of the program counter with the contents of the Ca register, and the 4 bits with the contents of the accumulator with the program counter output the address following [ROM] instruction. After executing one step at the branch destination, execution returns to the address of the ROM instruction + 2. However, if a second branching is executed from the first destination, execution is transferred to the address specified in that branch instruction.

Example	Address	Instruction	
①	02E0H	MVBA 5	Acc ← 5
②	02E1H	CAS	Ca ← 1
③	02E2H	ROM	
④	02E3H	NOP	
⑥	02E4H	*****	
⑦	02E5H	*****	
⑧	02E6H	*****	
⑨	02E7H	*****	
⑩	02E8H	*****	
·	02E9H	*****	
·	02EAH	*****	
·	02EBH	*****	
·	02ECH	*****	
·	02EDH	*****	
	02EEH	*****	
	02EFH	*****	
	02F0H	*****	
	02F1H	*****	
	02F2H	*****	
	02F3H	*****	
	02F4H	*****	
⑤	02F5H	*****	
	02F6H	*****	
	02F7H	*****	
	02F8H	*****	

Instructions are executed in the order of the circled numbers. At ①, the accumulator is set to "5" and, at ②, the Ca register is set to "1". If [ROM] is now executed, the instruction ④ in the step after [ROM] is executed and then the address in ④ is changed into the address in ⑤ replaced with the contents of Ca register and accumulator, and the instruction at the address in ⑤ is executed.



The instruction at ⑤ is executed and execution then return to the normal address ⑥.

Note 1: Do not use branch instructions such as [PAGE], [JUMP], [CALL] or [RET] in position ④.
When a branch instruction is executed at position ⑤, execution branches to the destination specified at position ⑤.

X, Y, and T Registers

X and Y Registers

Both X and Y registers are 4-bit registers. They are mainly used as address registers for work RAM, but can also be used as general-purpose registers.

X register			
X8	X4	X2	X1

* Specifies X address of work RAM
* General-purpose register

Y register			
Y8	Y4	Y2	Y1

* Specifies Y address of work RAM
* General-purpose register

T Register

The T register is a 4-bit general-purpose register. It can also be used to specify the X address in work RAM when using the [ADD b] and [SUB b] instructions. It can also be used to set the delay time in the [DLY b] instruction.

T register			
T8	T4	T2	T1

* General-purpose register
* Specifies X address of work RAM
* Sets delay time

Work RAM

The work RAM is a 16 × 16 matrix with each cell having a 4-bit capacity. The work RAM stores user data for working, LCD display data, and subroutine return addresses.

LCD display data is stored in the work RAM such that it corresponds to the LCD panel matrix. “1” lights the corresponding dot, which “0” turns it off. The position and capacity of the LCD display area is different depending on products.

When making a subroutine call, the value of the address of the [CALLn adr] instruction + 1 (program counter +1) can be stored in row X7. Up to 4 values can be stored in Y = 0 to 3 for [CALL1 adr], Y = 4 to 7 for [CALL2 adr], Y = 8 to 11 for [CALL3 adr], and Y = 12 to 15 for [CALL4 adr]. When returning from a subroutine, the value stored in the work RAM by the [CALLn adr] instruction (where n corresponds to n in [RETn]) is sent to the program counter. Other areas are available to the user, but because the user can also access the subroutine return address area, care must be taken not to corrupt the contents at that location.

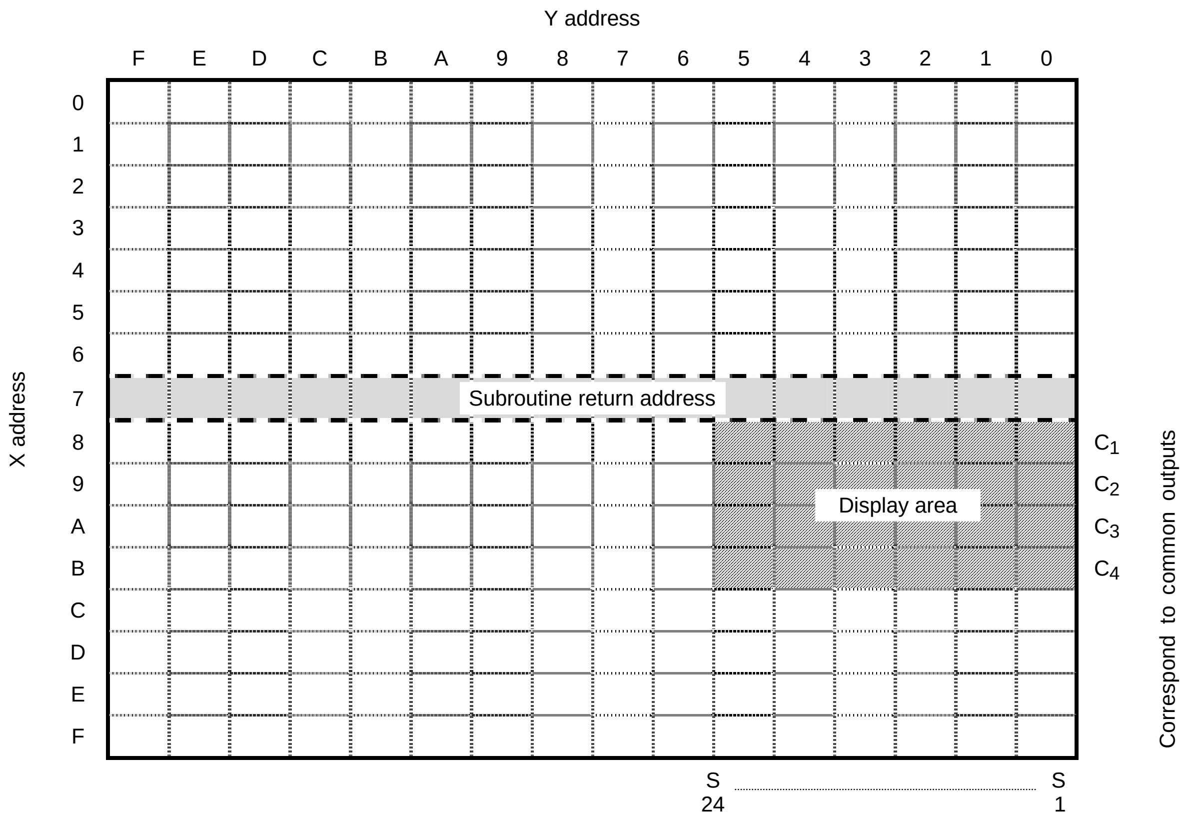


Figure 3 Work RAM Map

The work RAM is undefined when the power is turned on, and must therefore be initialized.

A/S, Accumulator (Acc), Carry Register (Ca) and Jump Flag (JF)

A/S

The A/S (adder/subtracter unit) is a circuit for calculations on 4-bit binary data. The A/S performs calculations as specified in instructions and outputs the result (4 bits) with carry data.

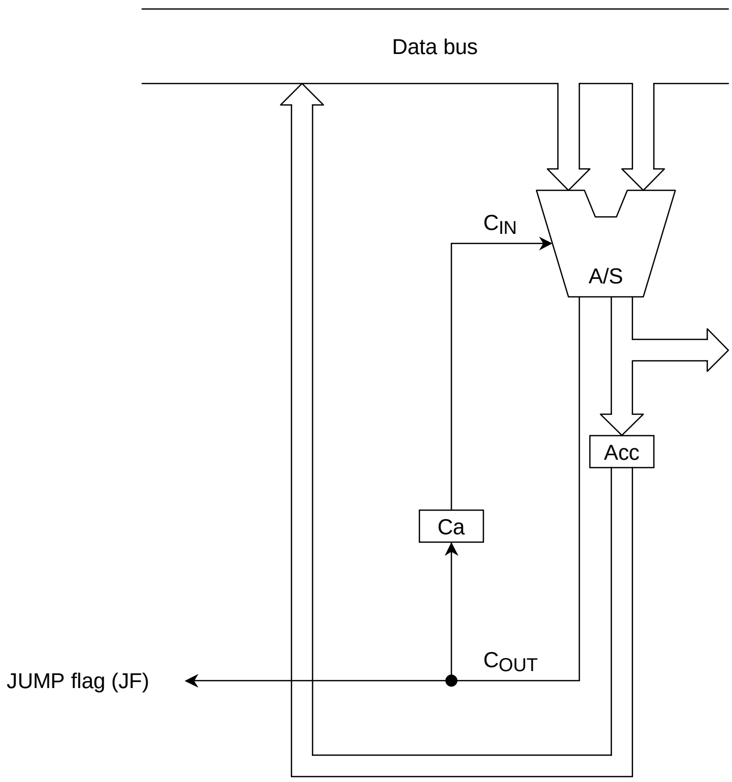


Figure 4 A/S

Accumulator (Acc)

The accumulator is a 4-bit register that stores the results of calculations.

Carry Register (Ca)

The carry register is a 1-bit register that stores the carry data output from the A/S. It can be set to "0" or "1" by instructions. Whichever value remains unchanged until the next instruction is executed that has an effect on the carry register setting.

As shown in Figure 5, the carry register can also be used in conjunction with the accumulator as a 5-bit register.

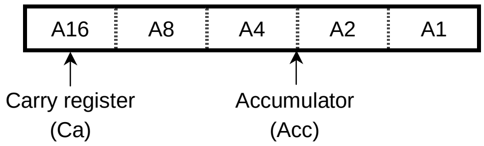


Figure 5 Ca Register Used in Conjunction with Accumulator

Jump Flag (JF)

The jump flag is a 1-bit flag consisting of the carry information output from the A/S and used as the branch condition for a jump instruction [JUMP]. The jump flag is set to "1" when an instruction that sets the jump flag is executed and all conditions for the instruction are satisfied. It is kept at "1" until the next instruction has been executed, at which point it is reset to "0". An unconditional jump is performed when an instruction that does not affect the jump flag is inserted before the jump instruction.

Note that setting the jump flag (JF) and the carry register (Ca) are different processes.

Oscillation and Operating Modes

Clock Generator and Timing Generator

The T6C96A clock generator consists of two oscillation circuits, allowing a high-speed clock (frequency fCR) and low-speed clock (frequency fXT) to be generated. By using each of these clocks in appropriate circumstances, it is possible to save power consumption. Both the high-speed and the low-speed clock oscillation circuit are CR oscillation circuits.

The system clock supplied to the CPU is created using the timing generator. Either the high-speed or low-speed clock can be selected, using instructions, for sending to the timing generator. Executing the [FCR] instruction sends the high-speed clock to the timing generator. Conversely, executing [FXT] sends the low-speed clock to the timing generator.

On a system reset, the high-speed clock is sent to the timing generator.

Either the high-speed or low-speed clock can be selected using instructions for sending to the LCD driver. Executing the [DXT] instruction sends the low-speed clock to the LCD driver.

Clock Generator

The high-speed CR oscillation clock generator oscillates by the external mounting of a resistor (Rosc).

The high-speed CR oscillation clock generator frequency is defined as “fCR”. The oscillation frequency of the low-speed CR clock generator is defined as “fXT”.

Note the following:

Note 2: The value of the oscillation resistor must be selected such that fCR>fXT.

Note 3: Due to variations in the oscillation resistor and the respective LSI characteristics, the values shown for fCR in the electrical characteristics table are only for reference. For debugging software, for example, it may be necessary to check operation with a ±50% frequency variation.

With reference to the above and the electrical characteristics tables, select the optimum oscillation resistance to ensure power savings.

Figure 6 shows the high-speed clock oscillation circuit.

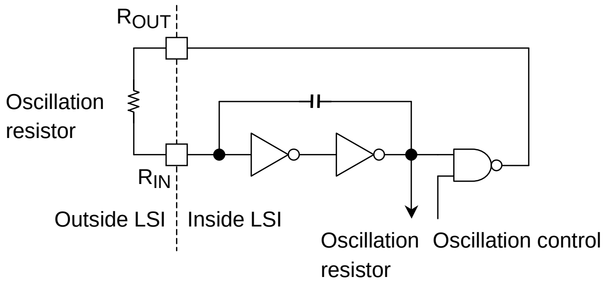


Figure 6 High-Speed Clock Oscillation Circuit

Low-speed CR oscillation clock generator is built in resistor and capacitor so that external parts are not required.

Figure 7 shows the CR oscillation circuit for low-speed CR.

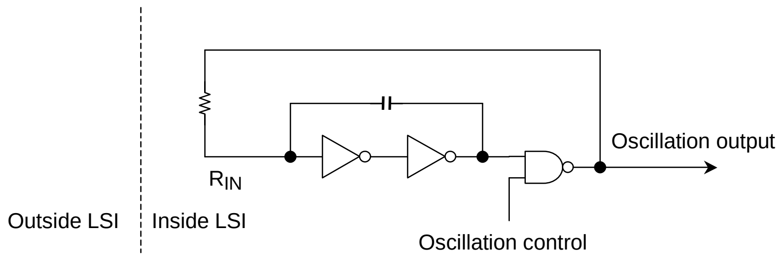


Figure 7 Low-Speed CR Oscillation Circuit

Timing Generator

The timing generator is a circuit that generators the system clock supplied to the CPU from the source clock generated by the clock generator.

The execution of instructions and CPU operations are performed in sync with the system clock. Instruction executions are executed as combinations of basic instructions called “microinstructions”. The time required to execute one microinstruction is called a machine cycle, which is one cycle of the system clock. The T6C96A instructions include 2-cycle instructions, which are executed in 2 machine cycles, 4-cycle instructions, which are executed in 4 machine cycles, 6-cycle instructions, which are executed in 6 machine cycles, and 8-cycle instructions, which are executed in 8 machine cycles.

Figure 8 shows the relationship between the source clock and system clock.

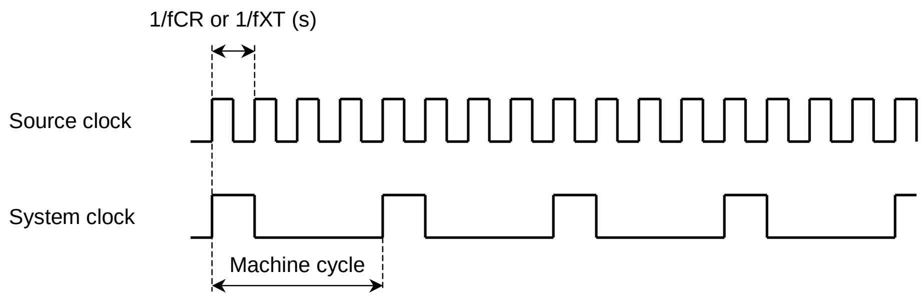


Figure 8 Source Clock and System Clock

Operating Modes

To achieve greater power savings, the T6C96A microcontrollers have four CPU operating modes: high-speed, low-speed, STOP, and OFF.

Statuses in Respective Operating Modes

- High-Speed Operating Mode
  In high-speed operating mode, both high-speed and low-speed clocks oscillate, the system clock being generated from the high-speed clock so that instructions are executed at high speed.
  The system is initialized to high-speed operating mode after a system reset.
- Low-Speed Operating Mode
  In low-speed operating mode, the high-speed clock is stopped such that only the low-speed clock oscillates, the system clock being generated from the low-speed clock so that instructions are executed at low speed.
- STOP Mode
  In STOP mode, the high-speed clock is stopped such that only the low-speed clock oscillates. The system clock is stopped and the CPU is therefore also stopped. The internal status of the CPU is maintained.
- OFF Mode
  In OFF mode, both high-speed and low-speed clocks are stopped such that both the CPU and display system are also stopped. The internal status of the CPU is maintained.
  Table 2 shows the oscillation circuits, CPU, and LCD driver status in the respective operating modes.

Table 2 Oscillation Circuit, CPU, and LCD Driver Statuses in Operating Modes

	High-speed operating mode	Low-speed operating mode	STOP mode	OFF mode
High-speed clock	Oscillating	Stopped	Stopped	Stopped
Low-speed clock	Oscillating	Oscillating	Oscillating	Stopped
CPU	High-speed clock	Low-speed clock	Stopped	Stopped
LCD display	Display available	Display available	Display available	No display

Transition between Operating Modes

- At Power-ON and at Resetting
  The system defaults to high-speed operating mode when the power is turned ON or after a system reset.

- Transition from High-speed Operating Mode
  The [FXT] instruction transfers from high-speed to low-speed operating mode.
  STOP mode is selected by executing the [STOP] instruction. OFF mode is selected by executing the [OFF] instruction. When STOP mode or OFF mode have been selected from high-speed operating mode, high-speed operating mode resumes when those modes are exited.
  After resuming operation, instructions start to be executed from the step following execution of the [STOP] or [OFF] instruction.

Note 4: Neither [STOP] or [OFF] are executed when an “L” level signal is input to the IN terminal, or when the timer F/F corresponding to the content of the TIM register is set.

- Transition from Low-speed Operating Mode
  The [FCR] instruction transfers from low-speed to high-speed operating mode.
  STOP mode is selected by executing the [STOP]” instruction. OFF mode is selected by executing the [OFF] instruction. When STOP mode has been selected from low-speed operating mode, low-speed operating mode resumes when that mode is exited.
  When OFF mode has been selected form low-speed operation mode, high-speed operation mode resumes when that mode is exited.
  After resuming operation, instructions start to be executed from the step following execution of the [STOP] or [OFF] instruction.

Note 5: Neither [STOP] or [OFF] are executed when an “L” level signal is input to the IN terminal, or when the timer F/F corresponding to the content of the TIM register is set.

- Resumption from STOP Mode
  There are two methods of resuming operation from STOP mode, as follows:

  (1) Resumption Using IN Terminal
  When an “L” level signal is input to one of the normally pulled up IN terminals (IN11 to IN14)

  (2) Resumption by Timer
  When the timer F/F corresponding to the content of the TIM register is set

  When STOP mode has been selected from high-speed operating mode, the system resumes high-speed operation when the STOP mode is exited.
  When STOP mode has been selected from low-speed operating mode, the system resumes low-speed operation when the STOP mode is exited.

Note 6: [STOP] is not executed while the conditions for resuming operation from STOP mode are satisfied.

Note 7: See timer description for details.

- Resumption from OFF Mode
  There is only one method of resuming operation from OFF mode, as follows:

  (1) Resumption from IN Terminal
  When an “L” level signal is input to one of the normally pulled up IN terminals (IN11 to IN14)

  Note that the system is initialized to high-speed operating mode on resuming operation whether OFF mode was selected from high-speed or low-speed operating modes.

Note 8: [OFF] is not executed when the conditions for resuming operation from OFF mode are satisfied or when the timer F/F corresponding to the content of the TIM register is set.

Figure 9 shows the transition between operating modes.

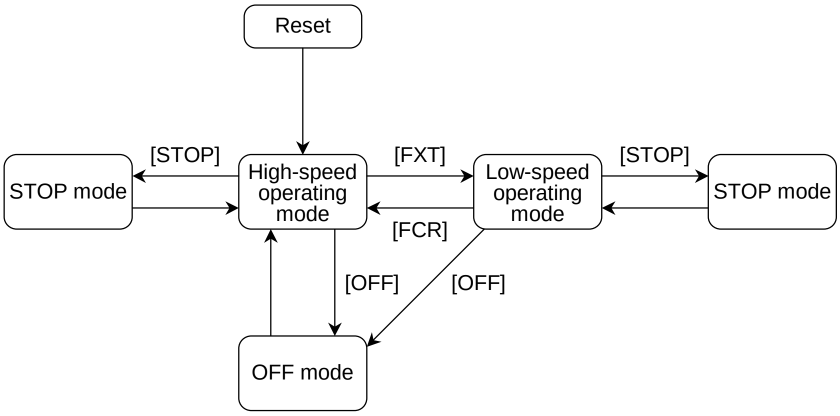


Figure 9 Operating Mode Status Transition Diagram

Timer

The timer is created by splitting the low-speed clock source oscillation (TYP. 32.8 kHz) using a 16-stage binary counter. The 10 th stage outputs a 31.25 ms signal, the 12 th stage a 125 ms signal, the 14 th stage a 500 ms signal, and the 16 th stage a 2 s signal. These four signals set specially provided timer F/Fs. The timer functions are implemented by checking the statuses of these F/Fs.

When a CR oscillation is used for the low-speed clock, the timer signals are not output with precision timing.

Figure 10 shows the timer structure.

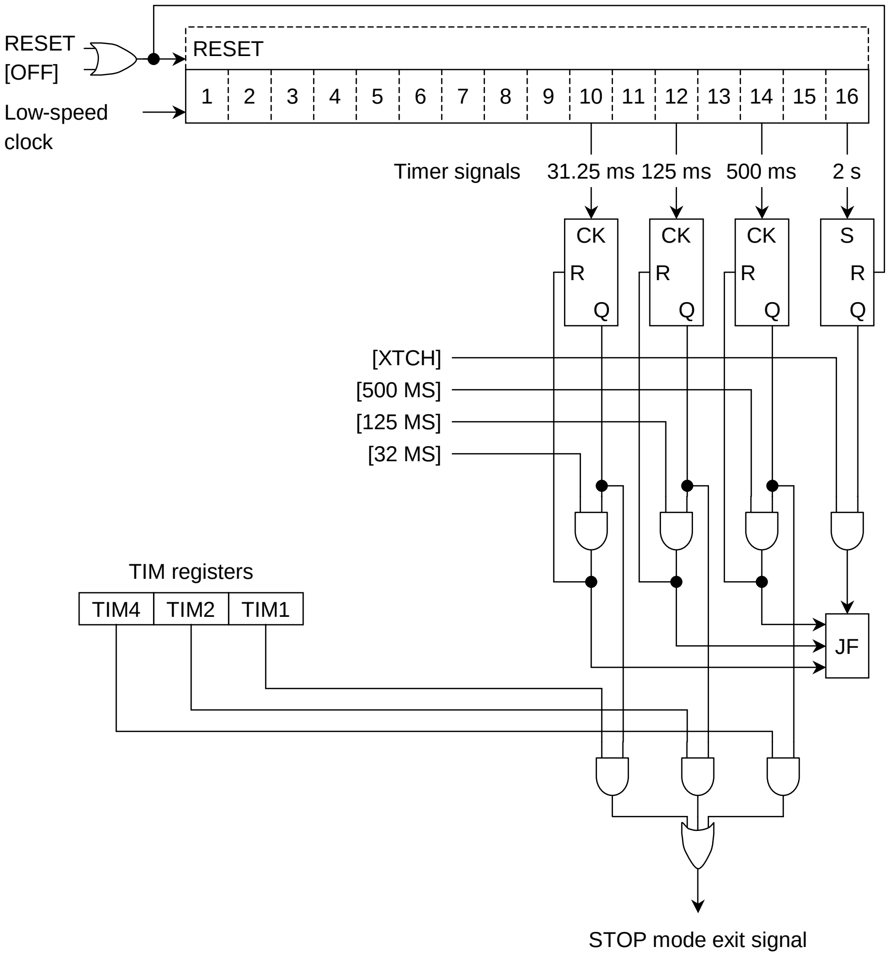


Figure 10 Timer Structure

The timer has the following 3 functions:

(1) Detection of 31.25 ms, 125 ms and 500 ms signals by instructions

(2) Exiting STOP mode

(3) Checking the oscillation status of the quartz oscillation circuit

Detection of 31.25 ms, 125 ms and 500 ms Signals by Instructions

The statuses of F/Fs of the 31.25 ms, 125 ms and 500 ms signals can be checked using instructions [32 MS], [125 MS] and [500 MS]. If the timer F/F is set, the jump flag is set (the Ca register is not set).

At the same time, the timer F/F is reset and again set at the next timing pulse.

Figure 11 shows the timer signals output from the 16-stage timer F/F and the respective timer F/F timing.

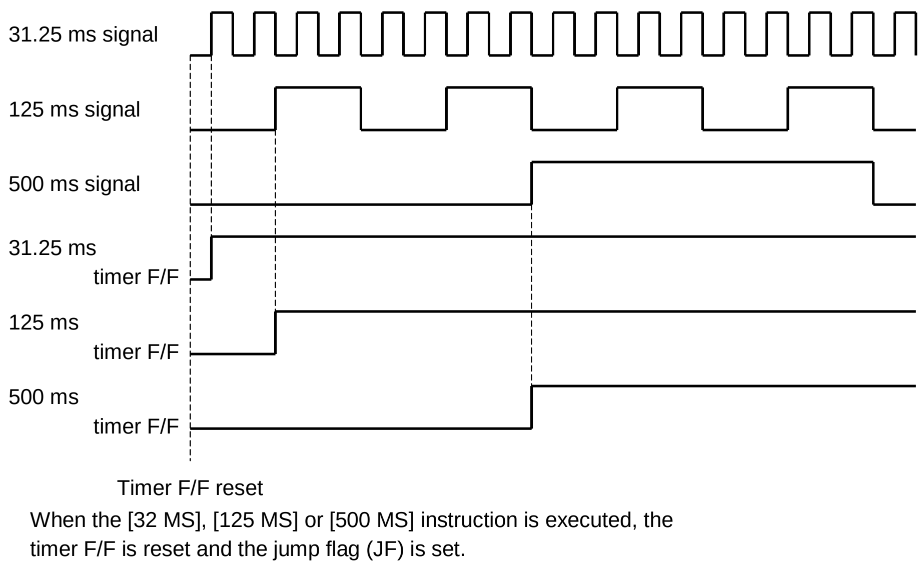


Figure 11 Timing for Setting Timer F/Fs

Exiting STOP Mode

Executing [TIM] sends the lower 3 bits of Acc to the TIM register. TIM1 corresponds to the 31.25 ms timer signal, TIM2 to 125 ms, and TIM4 to 500 ms, and check the timer F/Fs. If, when checked, the timer F/F corresponding to the bit of the set TIM register is set, STOP mode is exited. If two or more bits are set to “1”, the STOP mode exit signal is generated at the respective timings.

The timer F/F is not reset.

Figure 12 shows the timer signal output from the 16-stage timer F/F and the timing of the STOP mode exit signal according to the content of the TIM register.

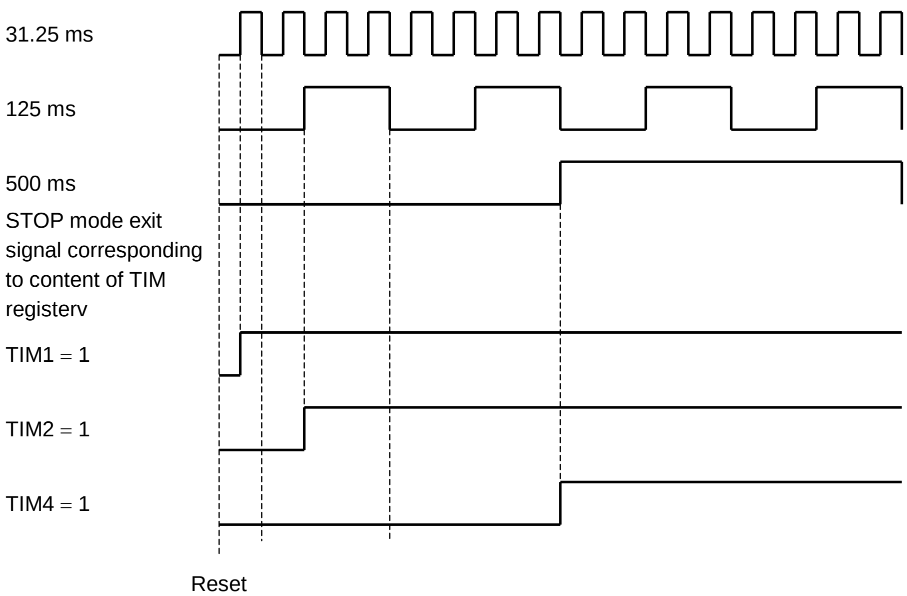


Figure 12 STOP Mode Exit Signal

External Interface

The T6C96A has one 4-bit input port, input-output port, and output port (IN port · I/O port · O port). The ports are used to "exchange data with peripheral circuits" by executing instructions.

First of all, the following is an explanation on port descriptions. In the case of IOmn (n = 1 to 4) being used as the description, m represents the port number and n represents the port's internal terminal number. For example, IO2 indicates the second IO port, and IO23 indicates the third terminal of the second IO port.

Input Ports (IN Ports)

IN ports are dedicated four terminal, four bit input ports, and the data input through the IN ports is stored in the accumulator.

As the In terminals are constantly subject to pull-up within the LSI, simple input circuits can be created with a minimum of external components.

Figure 13 shows the configuration of the IN terminals.

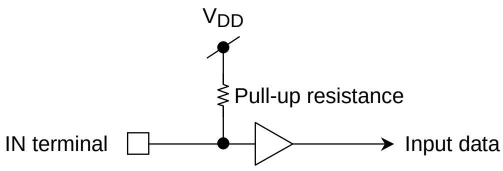


Figure 13 IN Terminal Configuration in the Normal Mode

Figure 14 shows an example of timing using the input instruction [IIN] to fetch IN port data into the accumulator.

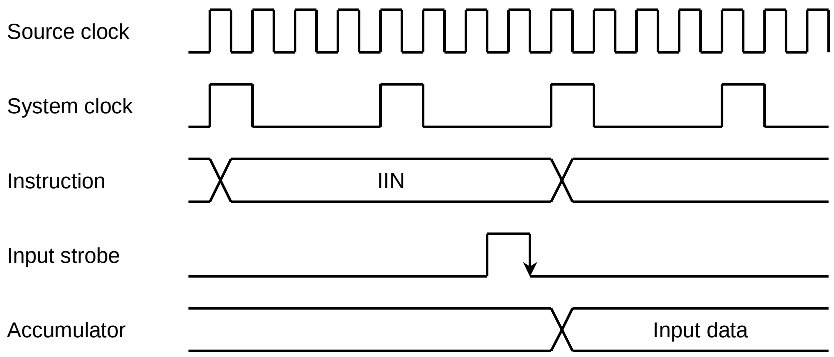


Figure 14 IN Port Input Timing

The IN ports also function for exiting the STOP and OFF modes. When the MCU is in STOP or OFF mode, inputting an “L” level signal to one of the normally pulled-up IN terminals exits the STOP or OFF mode.

Figure 15 shows the structure of a circuit for creating the signals for exiting the STOP and OFF modes. Please also refer to the timer in Figure 10.

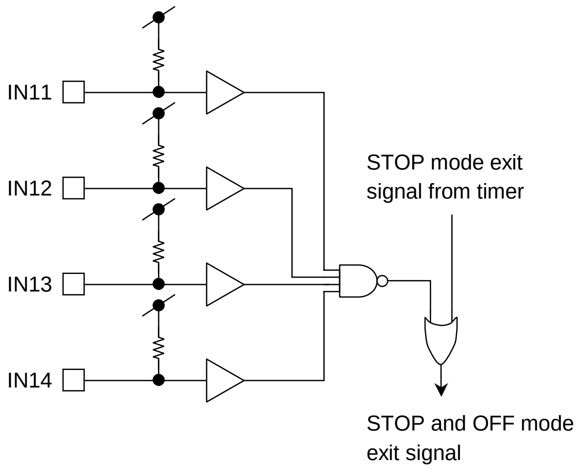


Figure 15　STOP and OFF Mode Exit Signal from IN Port

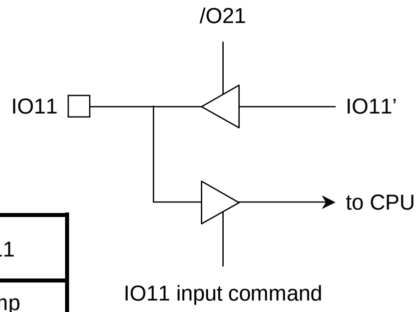
IO Ports

Each T6C96A port consists of two four-bit IO ports (IO11 to IO14, IO21 to IO24) and one single-bit IO port (IO31).

<IO11 Terminals>

Port IO11 has a 3-state structure equipped with output latching which enables HI-Z when the O21 control signal (single-bit) is set with software. The O21 control signal for [OTB2 1] is initialized with "L" when reset.

Figure 16 shows the configuration of the IO11 terminal.



O21	IO11
1	H-imp
0	IO11'

Figure 16 IO11 Terminal Configuration

<IO12 to IO14, IO21 to IO24 and IO31>

Ports IO12 to IO14, IO21 to IO24 and IO31 employ a method that does not utilize 3-state buffers. This method creates large "H" level output resistance for the inverter's P channel transistors used for the output.

Figure 17 shows the configuration of these terminals.

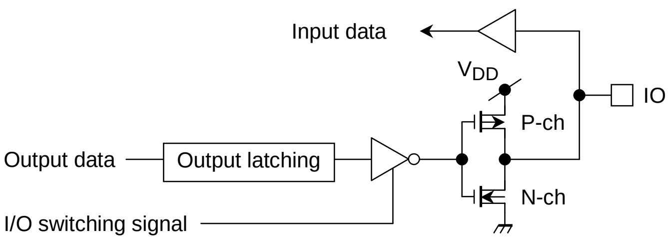


Figure 17 IO12 to IO14, IO21 to IO24 and IO31 Terminal Configuration

As the performance of the N-ch transistor used for output is large when the latch data in the output latch is "L", there will be cases where the "H" level in Acc cannot be loaded despite the [INIOm] input instruction being executed.

It is therefore necessary to execute the [OIBm 15] instruction and set the inverter's P channel transistor at ON prior to executing the input instruction.

IO Port Input Timing

Figure 18 shows an example of the input instruction [IIOm] for reading data via an IO port with nothing connected to the IO port.

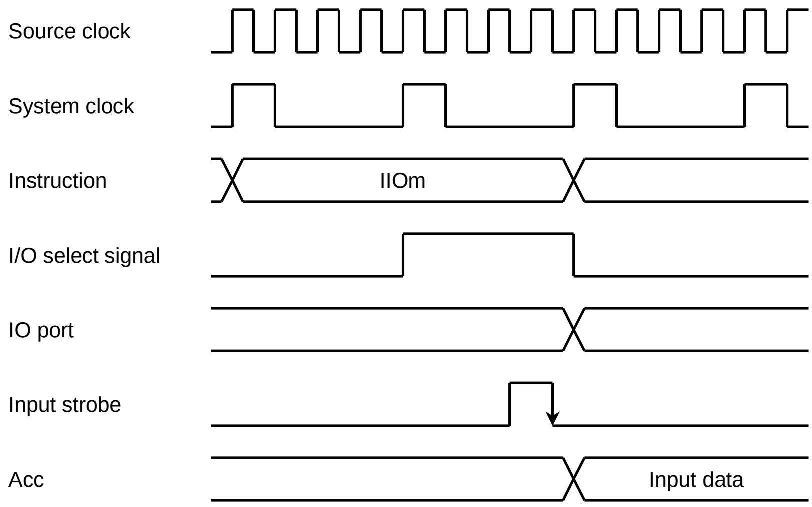


Figure 18 Input Timing of IO Port

IO Port Output Timing

Figure 19 shows an example of using the 2-cycle IO port output instruction [OIAm]. This instruction outputs the content of the accumulator (Acc) to port IOm.

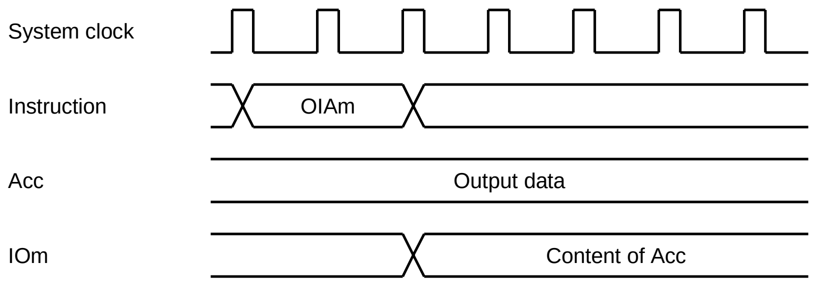


Figure 19 IO Port Output Timing for [OIAm] Instruction

Figure 20 shows an example of using the 4-cycle IO port output instruction [OIBm b].
This instruction writes immediate data b (4 bits) to the accumulator (Acc) and outputs that data to port IOm.

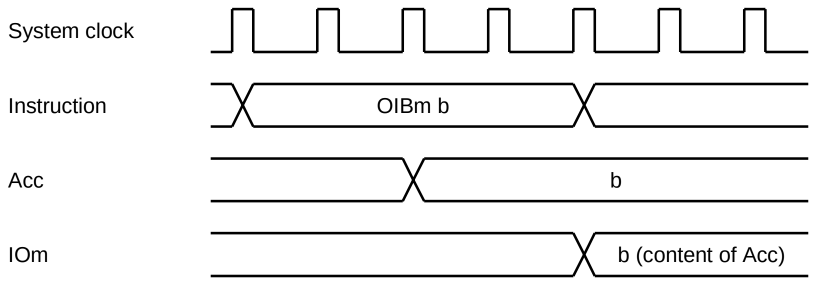


Figure 20 IO Port Output Timing for [OIBm b] Instruction

Figure 21 shows an example of using the 8-cycle IO port output instruction [OIR].
This instruction writes the content of work RAM specified by the contents of the X and Y registers to the accumulator (Acc), then outputs the data to port IO1. It then writes the contents of the work RAM specified by the contents of the X and Y registers + 1 to the Acc, then outputs that data to port IO2.

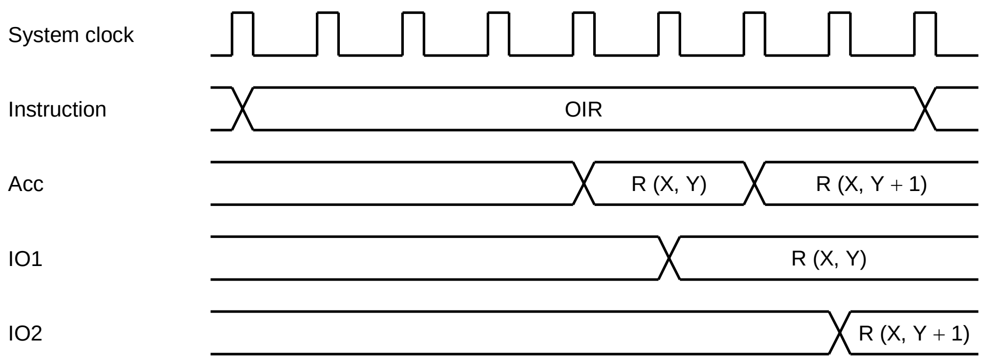


Figure 21 IO Port Output Timing for [OIR] Instruction

Output Ports (O)

O ports are 4-bit output only ports with 4 terminals. All are push-pull structures with latches. When the output instruction is executed, the output data is latched before being output. The latch is initialized to the "L" level when the MCU is reset.

Figure 22 shows the structure of an O terminal.

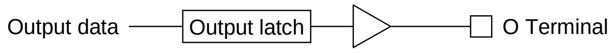


Figure 22 O Terminal Structure

The following shows O port output timing.

Figure 23 shows an example of the 2-cycle O port output instruction [OTAm].

This instruction outputs the contents of the accumulator (Acc) to port Om.

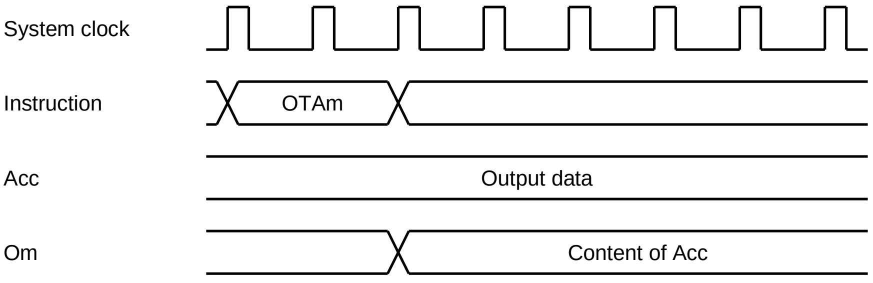


Figure 23 O Port Output Timing for [OTAm] Instruction

Figure 24 shows an example of the 4-cycle O port output instruction [OTBm b].

This instruction writes immediate data b (4 bits) to the accumulator (Acc) and outputs that data to port Om.

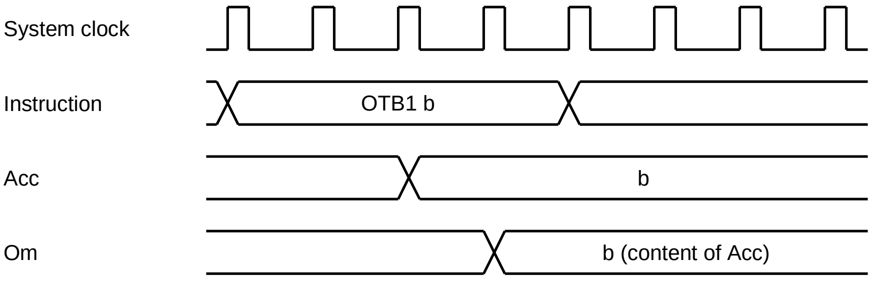


Figure 24 O Port Output Timing for [OTBm b] Instruction

HCLK

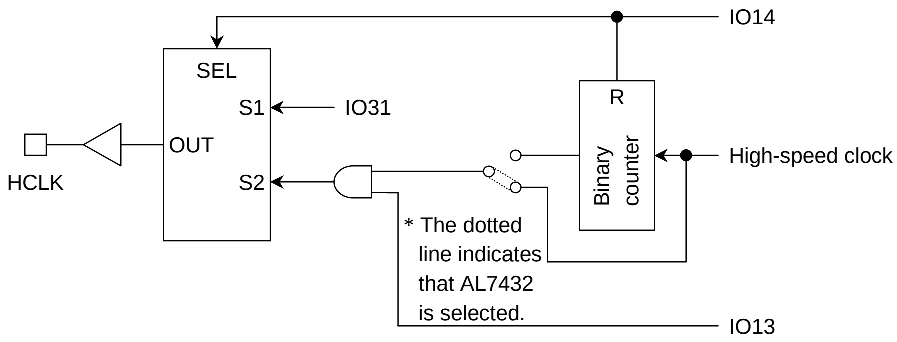
The HCLK is mainly used for outputting a high-speed CR oscillation (3.58/1.79 MHz) clock. The HCLK pin is controlled using the output instruction [IOBm b] by which output data to the IO11 to IO14 and IO31 pins are determined.

Either the high-speed CR oscillation frequency (3.58 MHz) or half of that frequency (1.79 MHz) output from the HCLK pin can be selected as shown in the mask option table below. Please request the desired mask option at sampling.

Table 3 Mask Option

AL CODE	HCLK output frequency
AL7432	High-speed CR oscillation frequency (3.58 MHz)
AL7434	High-speed CR oscillation frequency × 1/2 (1.79 MHz)

A structural diagram of the HCLK circuit is shown below.



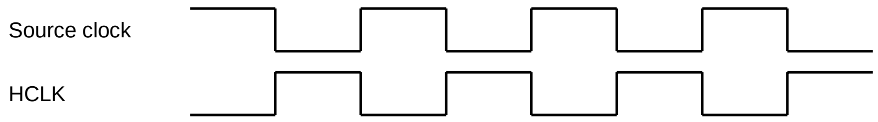
IO14	IO13	IO31 (= S1)	S2	HCLK
1	*	0	*	0
1	*	1	*	1
0	1	*	CLK*1	CLK*1

* : Don’t care

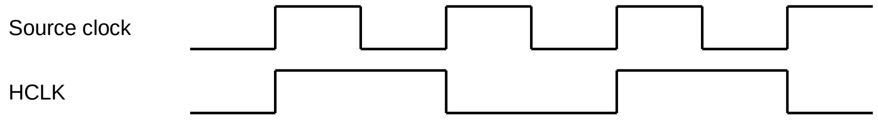
*1: When mask option AL7434 is selected, CLK × 1/2 is output.

Figure 25 HCLK Structural Diagram

The relation between the source clock and HCLK is as follows:
When mask option AL7432 is selected (HCLK = high-speed CR oscillation frequency output is selected)



When mask option AL7434 is selected (HCLK = high-speed CR oscillation frequency × 1/2 output is selected)



Note 9: When the high-speed CR oscillation frequency divided in half is selected (AL7434 selected), to start the signal output from the HCLK pin at Low level, initialize the binary counter using the following procedure when starting the LSI and when stopping the HCLK output.

OIB3 0000b ; outputs Low to IO31.
OIB1 11**b ; changes HCLK output to S1 (IO31 = Low) and simultaneously resets binary counter Because the asterisks (**) are the output values of IO12 and IO11, make settings accordingly.

LCD Driver

The T6C96A microcontrollers incorporate an LCD driver for driving an LCD and the necessary control circuit. It has 96 LCD pixels (segements × common).

LCD Driver Structure

Figure 26 shows the structure of the LCD driver.

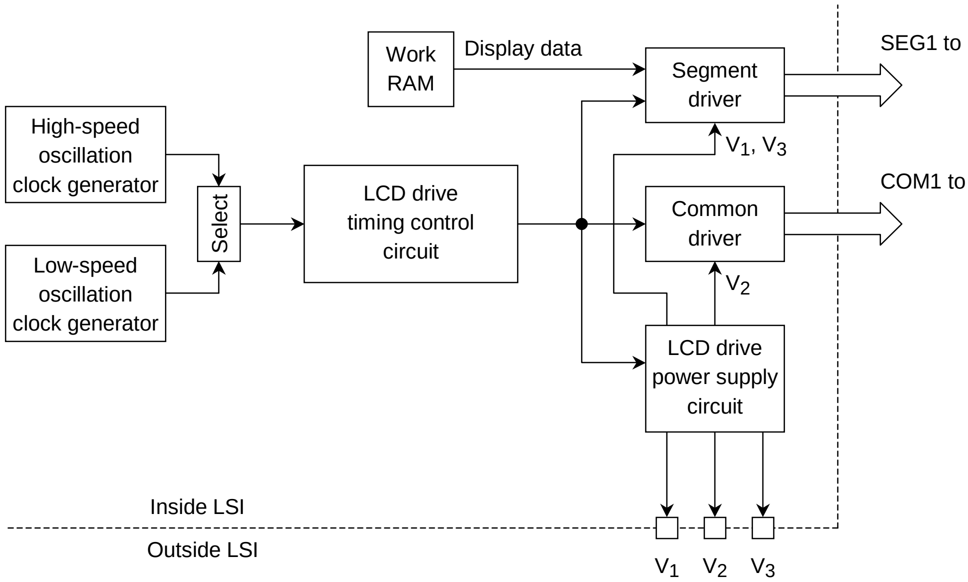


Figure 26 Structure of LCD Driver

The LCD drive timing is derived from either the high-speed or low-speed clock. At a system reset, the system is initialized to the high-speed clock, but the low-speed clock can be selected by executing an instruction. When in low-speed operating mode, the LCD driver should also use the low-speed clock. When the LCD driver uses the low-speed clock, the LCD display can be continued even in STOP mode.

Method of LCD Drive

The T6C96A microcontrollers have a display area in the work RAM. By writing data to this display area, a display waveform is automatically output from the segment terminals. The display area is a bitmap that corresponds one-to-one with the LCD display pixels.

If an area as indicated in Figure 27 is included within the work RAM, a 24 × 4 bit map as shown in Figure 29 exists owing to the fact that each cell consists of four bits, as indicated in Figure 28.

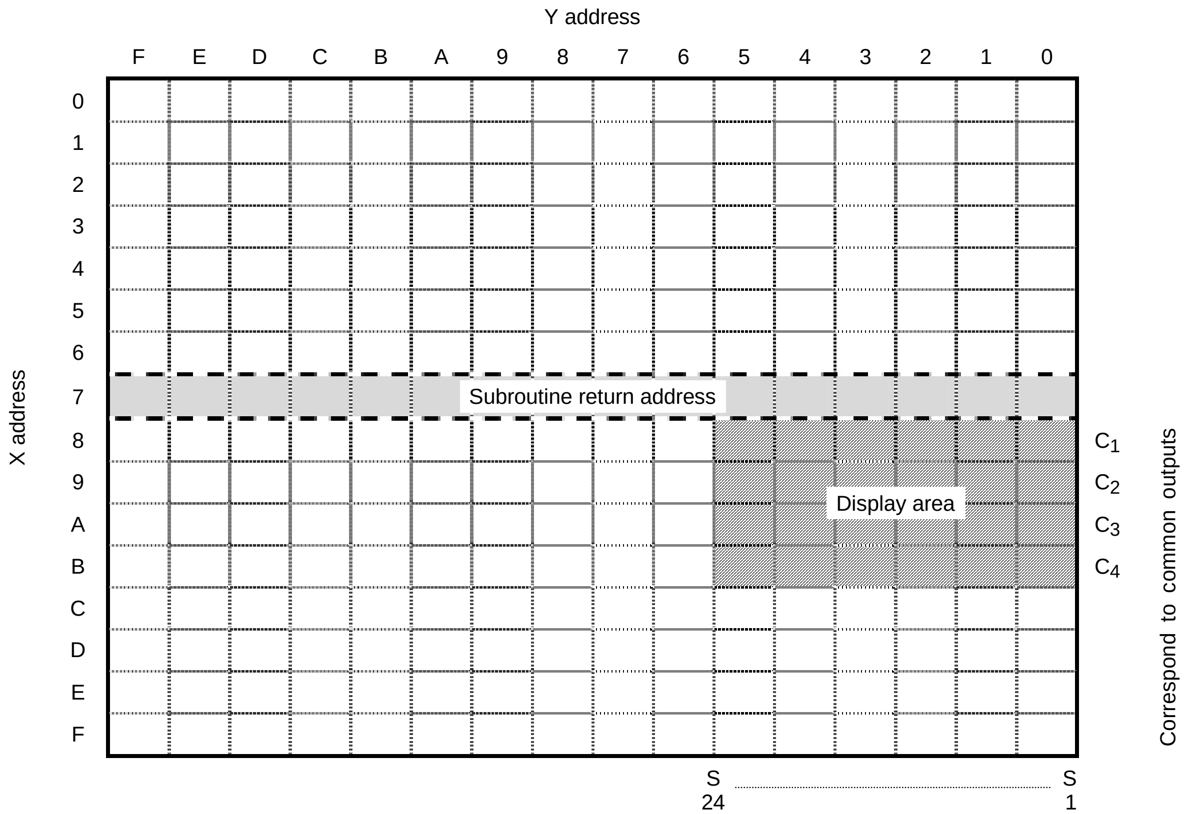


Figure 27 Map of Work RAM

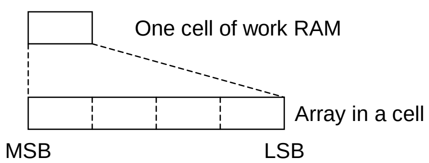


Figure 28 Cell Structure of Work RAM

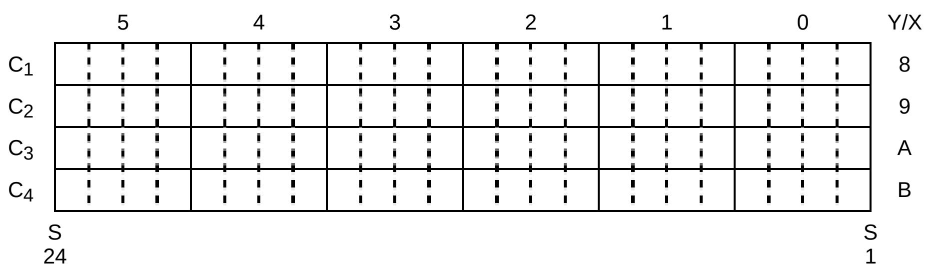


Figure 29 Bitmap of Display Area

LCD Drive Power Supply Circuit

The LCD drive power supply circuit generates the V_1, V_2 and V_3 voltages supplied to the common and segment drivers.

V_1, V_2 and V_3 are generated by using the frame pulse to control the resistance dividing circuit between V_{DD} and V_{SS}.

Figure 30 shows the structure of the LCD drive power supply circuit.

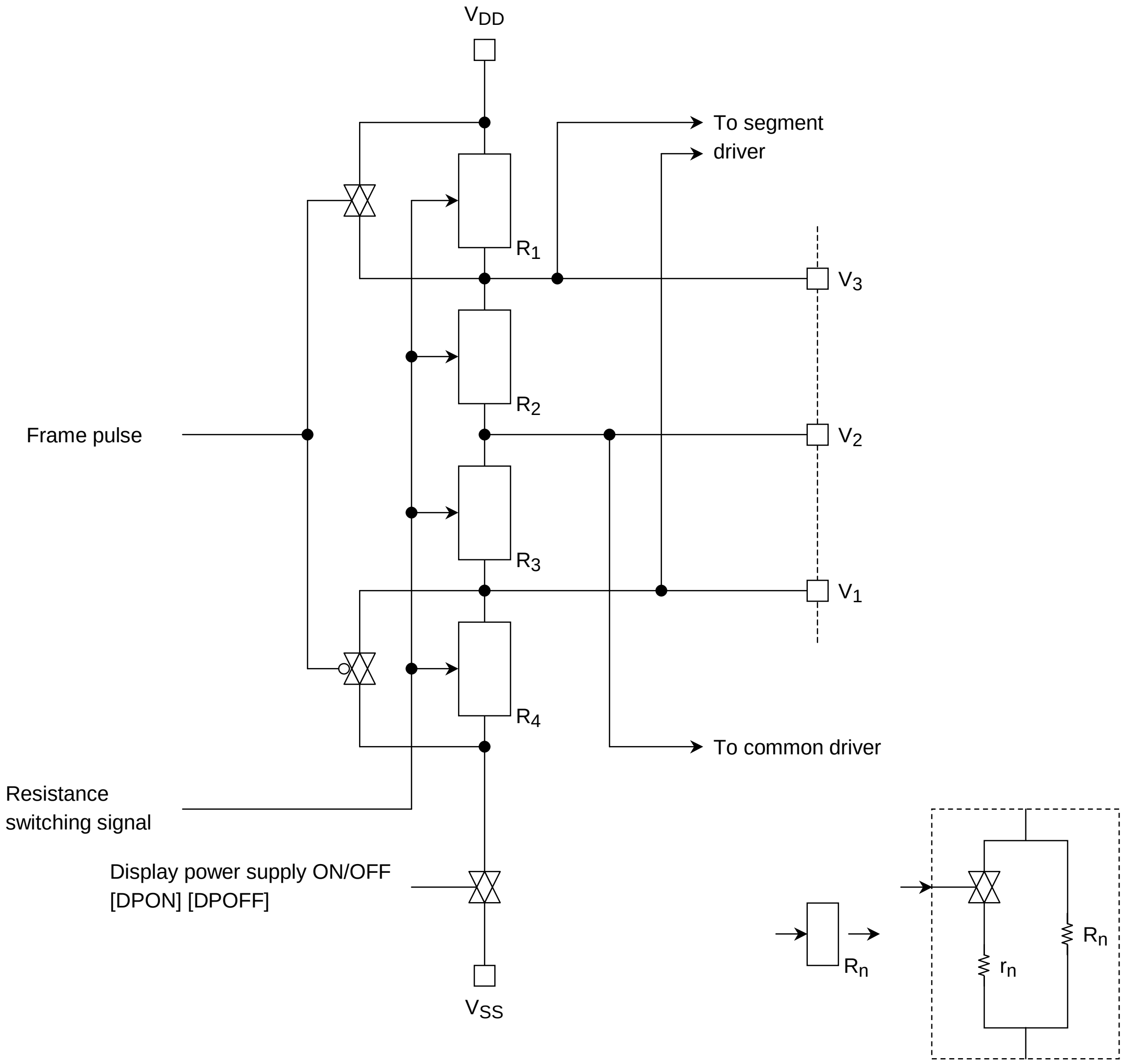


Figure 30 Structure of LCD Drive Power Supply Circuit

Note 10: $R_1:R_2:R_3:R_4 = r_1:r_2:r_3:r_4$

If the ratio between R_1, R_2, R_3 and R_4 (the dividing resistance) is 1:1:1:1, we achieve a 1/3 prebias.

To improve the through rate when switching the LCD elements ON/OFF, a large current is caused to flow at the instant the common is switched. Concretely, the large current flows only when the LCD element turns ON (current in both r_n and R_n), after that, the current is reduced (current only in R_n) for power saving.

The T6C96A microcontrollers have a power saving function that operates by turning the display power supply ON/OFF. When [DPON] is executed, the display power supply turns ON. When [DPOFF] is executed, it turns OFF.

Common Driver

The common driver circuit creates common signals with a predetermined timing.

Executing [DON] changes the common level to ON. Conversely, [DOFF] turns it OFF (same as V_2).

Figure 31 shows the common signal when the [DON] and [DOFF] instructions are executed. Signal COM1 is used in this example.

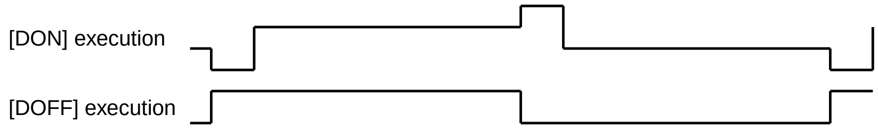


Figure 31 Common Signal at Execution of [DON] and [DOFF]

Figure 32 shows the structure of the common driver.

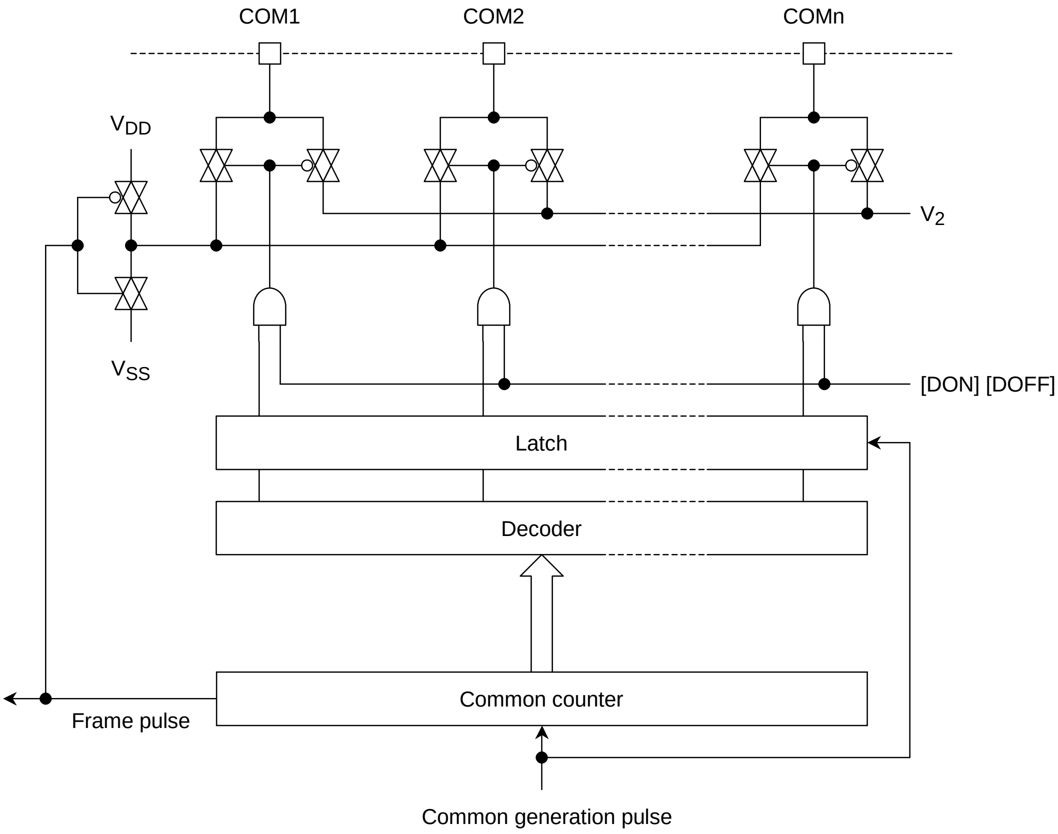


Figure 32 Structure of Common Driver

Segment Driver

The segment driver circuit generates segment signals using the timing determined from the content of the work RAM.

Figure 33 shows the structure of the segment driver.

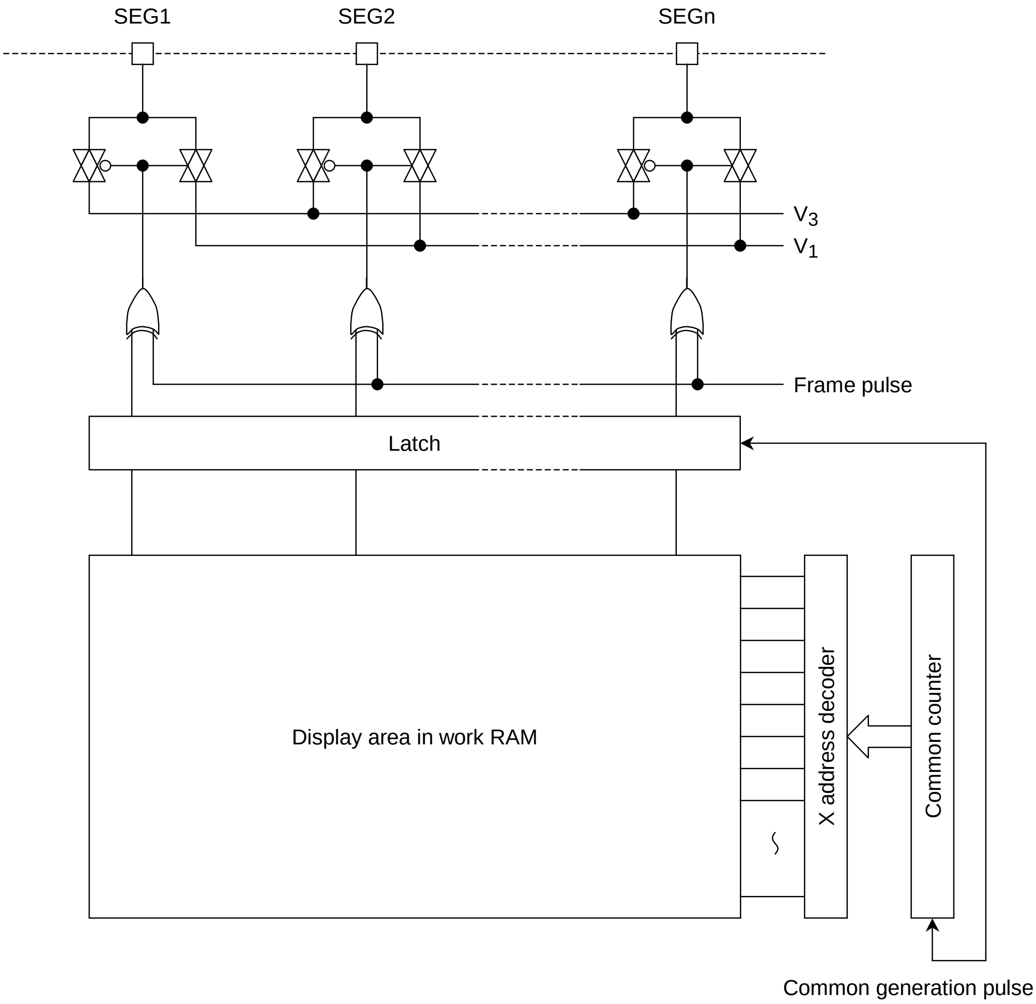


Figure 33 Structure of Segment Driver

Using Display Control Instructions

This section explains how to use the display control instructions described in the previous pages.

The instructions for selecting the clock used to generate the LCD display timing are placed at the beginning of a user program. Then, after data has been written to the work RAM, [DPON] is executed to turn ON the display power supply, followed by [DON] to set the common signal level to “ON” and effect the LCD display.

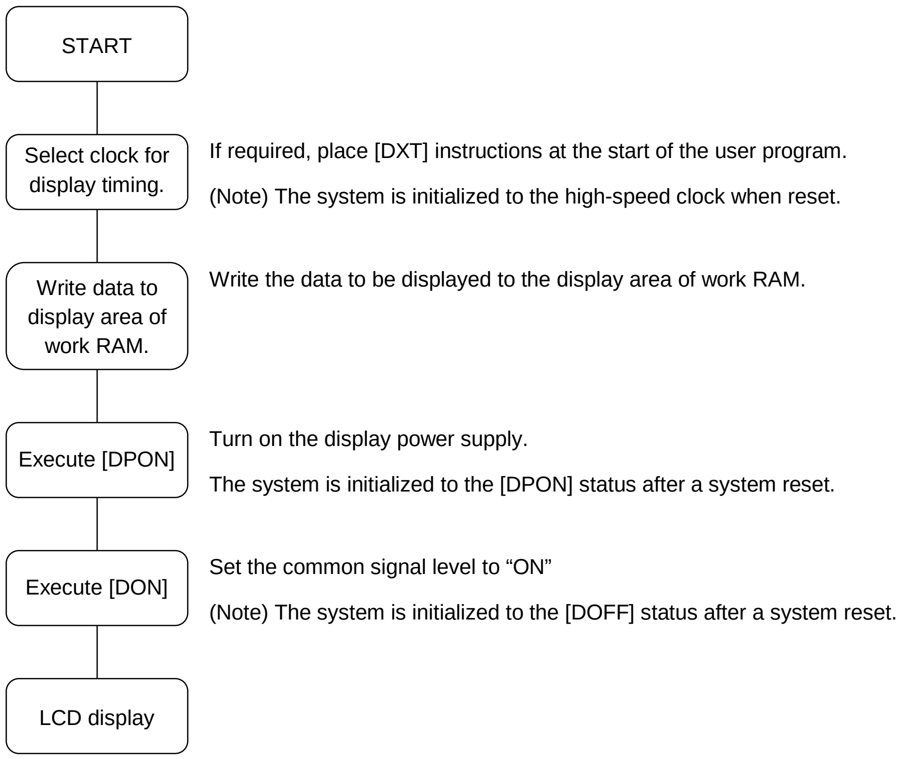


Figure 34 Flow of Operations to Displaying Data on LCD

Special T6C96A Instructions

[ADD b]

This instruction performs addition operations on decimal numbers. The result is also stored as a decimal.

Executing [ADD b] adds the content of work RAM specified by the X and Y registers and the content of work RAM specified by the T and Y registers to Ca. The result is stored in work RAM as specified by the X and Y registers.

This operation is repeated incrementing the Y register by 1 each time until an overflow occurs at Y register + b.

Example:

When the content of work RAM is as shown in Figure 35, executing the following instructions causes the content to change to that shown in Figure 36.

```
MVBW   2, 2
MVBT4
ADD    7
```

Y address

X address	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0																
1																
2							7	6	5	4	3	2	1	0		
3																
4							0	1	2	3	4	5	6	7		
5																
6																
7																

Figure 35 Content of Work RAM before Execution of [ADD b]

Y address

X address	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0																
1																
2							7	7	7	7	7	7	7	7		
3																
4							0	1	2	3	4	5	6	7		
5																
6																
7																

Figure 36 Content of Work RAM after Execution of [ADD b]

[SUB b]

This instruction performs subtraction operations on decimal numbers. The result is also stored as a decimal.

Executing [SUB b] subtracts the content of work RAM specified by the T and Y registers with Ca from the content of work RAM specified by the X and Y registers. The result is stored in work RAM as specified by the X and Y registers.

This operation is repeated incrementing the Y register by 1 each time until an overflow occurs at Y register + b.

Example:

When the content of work RAM is as shown in Figure 37, executing the following instructions causes the content to change to that shown in Figure 38.

```
MVBW    2, 2
MVBT4
SUB     7
```

Y address

X address	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0																
1																
2							7	6	5	4	3	2	1	0		
3																
4							0	1	2	3	4	5	6	7		
5																
6																
7																

Figure 37 Content of Work RAM before Execution of [SUB b]

Y address

X address	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0																
1																
2							7	5	3	0	8	6	4	3		
3																
4							0	1	2	3	4	5	6	7		
5																
6																
7																

Figure 38 Content of Work RAM after Execution of [SUB b]

[DLY b]

Executing [DLY b] generates a delay of {(16 − T) × 2 + 2} machine cycles according to the content of the T register.

The content of the T register is incremented by 1 each two machine cycles. Two machine cycles after an overflow occurs, immediate data b is sent to the T register and normal operation then resumes.

Figure 39 shows the delay that is generated and the resulting content of the T register when [DLY b] is executed when the content of the T register is 9.

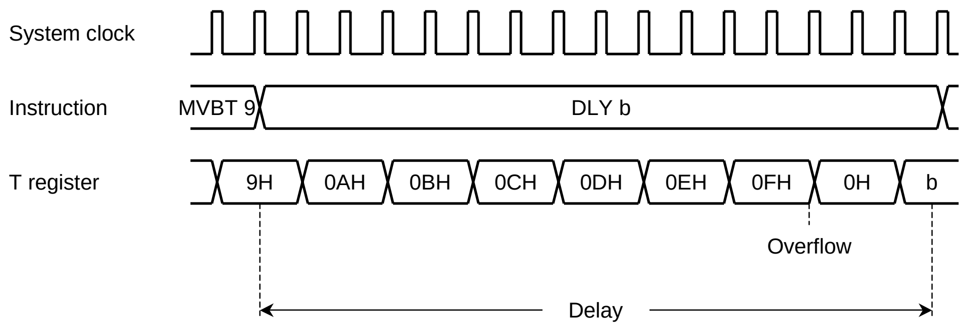


Figure 39 Timing of [DLY b] Execution

Configuration

This section contains notes on mounting T6C96A microcontrollers.

/AC (Reset)

When the power supply voltage is within the operating voltage range, keeping the level of the /AC terminal "L" for a minimum of 100 µs initializes the entire contents of the LSI. When the level of the /AC terminal changes to "H", the reset operation is canceled and program execution starts. The /AC terminal is internally pulled up, and it is therefore possible to create a simple reset circuit using only one switch.

Figure 40 shows a simple reset circuit.

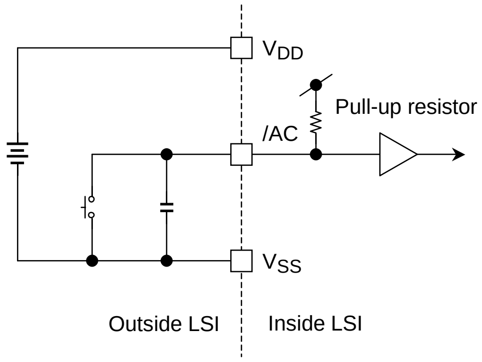


Figure 40 Example Reset Circuit

TS (Test)

The TS terminal is used for the shipping test by TOSHIBA and has a built-in pull-down resistor. For normal use, the TS terminal level should be fixed "L" as shown in Figure 41.

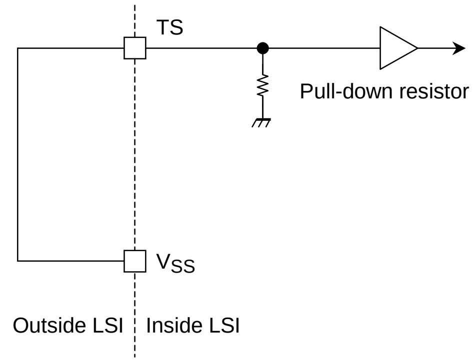


Figure 41 Treatment of TS terminal

R_{IN} and R_{OUT}

The R_{IN} and R_{OUT} terminals are connected inside the LSI to the clock generators. Connect a resistor between R_{IN} and R_{OUT} as shown in Figure 42.

Setup the frequency form 42.6 kHz upward.

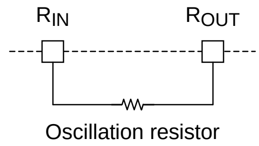


Figure 42 Method of Connecting Oscillator

Instruction Tables

In addition to the normal move, calculate, bit manipulation, and input-output instructions, the T6C96A also has special instructions for driving a LCD.

The following shows the conventions used in the instruction tables.

Mnemonic	Machine language	Microinstruction	Acc	Ca	JF	T	X	Y	Cy-cles
(*1)	(*2)	(*3)	(*4)						(*5)

(*1)

The instruction mnemonic

The operation code and operand are separated by one space. The x and y operands are 4-bit data specifying the cell in work RAM.

Operands a and b are, unless specifically mentioned in the text, 4-bit data. (An exception is the move instruction MVBA (b = 5 bits).)

Operand n is 4-bit data in move instruction TRNS, or indicates a specific bit 8, 4, 2, or 1 from the MSB in bit manipulation instructions.

Operand adr is 12 bits in the branch instruction JUMP, and 10 bits in the CALL instruction.

(*2)

The instruction in machine language

a, b, x, y, and n represent immediate data. The left of the string is the MSB, the right is the LSB.

(*3)

Microinstruction

The functions of the instruction are written as microinstructions.

One line corresponds to one machine cycle.

RTN indicates the end of a microinstruction.

When two or more microinstructions are shown in one machine cycle, they are delimited by "/".

(*4)

Contents of registers after instruction execution

0 for those that become "0"

1 for those that become "1"

* for those that change according to conditions

— for those that do not change

JF for those storing the JF in the Ca register

Others according to *1 and *2

(*5)

Number of machine cycles required to execute the instruction

Move Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cy-clesv
MVAT	00 0000 0101 0001	T ← Acc RTN	—	—	—	Acc	—	—	2
MVAX	00 0000 0101 0010	X ← Acc RTN	—	—	—	—	Acc	—	2
MVAY	00 0000 0101 0011	Y ← Acc RTN	—	—	—	—	—	Acc	2
MVAR	00 0001 0000 0000	R (X, Y) ← Acc RTN	—	—	—	—	—	—	2
MVAI x, y	00 0101 xxxx yyyy	R (x, y) ← Acc RTN	—	—	—	—	—	—	2
MVTA	00 0000 0101 1001	Acc ← T RTN	T	0	—	—	—	—	2
MVTX	00 0000 0101 0110	X ← T RTN	—	—	—	—	T	—	2
MVTY	00 0000 0101 0111	Y ← T RTN	—	—	—	—	—	T	2
MVTR	00 0001 0000 0010	R (X, Y) ← T RTN	—	—	—	—	—	—	2
MVXA	00 0000 0101 1010	Acc ← X RTN	X	0	—	—	—	—	2
MVXT	00 0000 0101 0100	T ← X RTN	—	—	—	X	—	—	2
MVYA	00 0000 0101 1011	Acc ← Y RTN	Y	0	—	—	—	—	2
MVYT	00 0000 0101 0101	T ← Y RTN	—	—	—	Y	—	—	2
MVRA	00 0001 0000 0001	Acc ← R (X, Y) /Ca Hold RTN	R (X, Y)	—	—	—	—	—	2
MVRT	00 0001 0000 0011	T ← R (X, Y) RTN	—	—	—	R (X, Y)	—	—	2
MVIA x, y	00 0100 xxxx yyyy	Acc ← R (x, y) RTN	R (x, y)	0	—	—	—	—	2
MVBA b	00 0001 010b bbbb	Acc ← b8~b1, Ca ← b16 RTN	b8~b1	b16	—	—	—	—	2
MVBAH b	00 0001 1000 bbbb	Acc ← b/Ca Hold RTN	b	—	—	—	—	—	2
MVBT b	00 0001 1001 bbbb	T ← b RTN	—	—	—	b	—	—	2
MVBX b	00 0001 1100 bbbb	X ← b RTN	—	—	—	—	b	—	2

Move Instructions

Mnemonic	Function
MVAT	Moves content of Acc to T register.
MVAX	Moves content of Acc to X register.
MVAY	Moves content of Acc to Y register.
MVAR	Moves content of Acc to work RAM specified by contents of X and Y registers.
MVAI x, y	Moves content of Acc to work RAM specified by instruction codes x and y.
MVTA	Moves content of T register to Acc. Ca is set to “0”.
MVTX	Moves content of T register to X register.
MVTY	Moves content of T register to Y register.
MVTR	Moves content of T register to work RAM specified by X and Y registers.
MVXA	Moves content of X register to Acc. Ca is set to “0”.
MVXT	Moves content of X register to T register.
MVYA	Moves content of Y register to Acc. Ca is set to “0”.
MVYT	Moves content of Y register to T register.
MVRA	Moves content of work RAM specified by X and Y registers to Acc.
MVRT	Moves content of work RAM specified by X and Y registers to T register.
MVIA x, y	Moves content of work RAM specified by instruction codes x and y to Acc. Ca is set to “0”.
MVBA b	Moves MSB of immediate data b (5 bits) to Ca and 4 bits on LSB side to Acc.
MVBAH b	Moves immediate data b to Acc.
MVBT b	Moves immediate data b to T register.
MVBX b	Moves immediate data b to X register.

Move Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cycles
MVBY b	00 0001 1101 bbbb	Y ← b RTN	—	—	—	—	—	b	2
MVBW a, b	00 0010 aaaa bbbb	X ← a Y ← b RTN	—	—	—	—	a	b	2
MBRX b	00 0001 0110 bbbb	R (X, Y) ← b NOP X ← X + 1 RTN	—	—	*	—	X + 1	—	4
MBRY b	00 0001 0111 bbbb	R (X, Y) ← b NOP Y ← Y + 1 RTN	—	—	*	—	—	Y + 1	4
TRNS x, n	00 0011 xxxx nnnn	Acc ← R (X, Y) Y ← Y − n R (x, Y) ← Acc /JF Y ← Y + n + 1 T ← T − 1 JF Check NOP JF RTN	R (X, Y + T)	0	—	15	—	Y + T + 1	6T + 8

Move Instructions

Mnemonic	Function
MVBY b	Moves immediate data b to Y register.
MVBW a, b	Moves immediate data a to X register and b to Y register.
MBRX b	Stores immediate data b in work RAM specified by X and Y registers, then increments X register.
MBRY b	Stores immediate data b in work RAM specified by X and Y registers, then increments Y register.
TRNS x, n	Moves content of work RAM specified by X and Y registers to work RAM specified by instruction code x and (Y register-n), then increments Y register. This is repeated by (T register + 1) times.

Operation Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cy-cles
ADTA	00 0001 0001 0101	Acc, Ca ← T + Acc + Ca RTN	T + Acc + Ca	JF	*	—	—	—	2
SBTA	00 0001 0001 0111	Acc, Ca ← T − Acc − Ca RTN	T − Acc − Ca	JF	*	—	—	—	2
ADTR	00 0001 0000 0100	T ← T + R (X, Y) RTN	—	—	*	T + R (X, Y)	—	—	2
SBTR	00 0001 0000 0110	T ← T − R (X, Y) RTN	—	—	*	T − R (X, Y)	—	—	2
ADRA	00 0001 0000 0101	Acc, Ca ← R (X, Y) + Acc + Ca RTN	R (X, Y) + Acc + Ca	JF	*	—	—	—	2
SBRA	00 0001 0000 0111	Acc, Ca ← R (X, Y) − Acc − Ca RTN	R (X, Y) − Acc − Ca	JF	*	—	—	—	2
ADIA x, y	00 0110 xxxx yyyy	Acc, Ca ← R (x, y) + Acc + Ca RTN	R (x, y) + Acc + Ca	JF	*	—	—	—	2
SBIA x, y	00 0111 xxxx yyyy	Acc, Ca ← R (x, y) − Acc − Ca RTN	R (x, y) − Acc − Ca	JF	*	—	—	—	2
ADBA b	00 0000 1000 bbbb	Acc, Ca ← b + Acc + Ca RTN	b + Acc + Ca	JF	*	—	—	—	2
SBBA b	00 0000 1001 bbbb	Acc, Ca ← b − Acc − Ca RTN	b − Acc − Ca	JF	*	—	—	—	2
ADTB b	00 0000 1010 bbbb	T ← T + b RTN	—	—	*	T + b	—	—	2
SBTB b	00 0000 1011 bbbb	T ← T − b RTN	—	—	*	T − b	—	—	2
ADXB b	00 0000 1100 bbbb	X ← X + b RTN	—	—	*	—	X + b	—	2
SBXB b	00 0000 1101 bbbb	X ← X − b RTN	—	—	*	—	X − b	—	2
ADYB b	00 0000 1110 bbbb	Y ← Y + b RTN	—	—	*	—	—	Y + b	2
SBYB b	00 0000 1111 bbbb	Y ← Y − b RTN	—	—	*	—	—	Y − b	2
INC x, y	00 1110 xxxx yyyy	Acc, Ca ← R (x, y) + 1 R (x, y) ← Acc R (x, y) − Acc − Ca RTN	R (x, y) + 1	JF	*	—	—	—	4
DEC x, y	00 1111 xxxx yyyy	Acc, Ca ← R (x, y) − 1 R (x, y) ← Acc R (x, y) − Acc − Ca RTN	R (x, y) − 1	JF	*	—	—	—	4

Operation Instructions

Mnemonic	Function
ADTA	Adds the content of the T register to contents of Acc and Ca and stores the result in Acc. An overflow sets Ca to “1”.
SBTA	Subtracts the contents of Acc and Ca from the content of the T register and stores the result in Acc. A borrow sets Ca to “1”.
ADTR	Adds the content of the T register to the content of work RAM specified by the X and Y registers, then stores the result in the T register.
SBTR	Subtracts the content of work RAM specified by the X and Y registers from the content of the T register, then stores the result in the T register.
ADRA	Adds the content of the work RAM specified by the X and Y registers to the contents of Acc and Ca, then stores the result in Acc. An overflow sets Ca to “1”.
SBRA	Subtracts the contents of Acc and Ca from the content of the work RAM specified by the X and Y registers, then stores the result in Acc. A borrow sets Ca to “1”.
ADIA x, y	Adds the content of work RAM specified by instruction codes x and y to the contents of Acc and Ca, then stores the result in Acc. An overflow sets Ca to “1”.
SBIA x, y	Subtracts the contents of Acc and Ca from the content of the work RAM specified by instruction codes x and y, then stores the result in Acc. A borrow sets Ca to “1”.
ADBA b	Adds immediate data b to the contents of Acc and Ca, then stores the result in Acc. An overflow sets Ca to “1”.
SBBA b	Subtracts the contents of Acc and Ca from immediate data b, then stores the result in Acc. A borrow sets Ca to “1”.
ADTB b	Adds the content of the T register to immediate data b, then stores the result in the T register.
SBTB b	Subtracts immediate data b from the content of the T register, then stores the result in the T register.
ADXB b	Adds the content of the X register to immediate data b, then stores the result in the X register.
SBXB b	Subtracts immediate data b from the content of the X register, then stores the result in the X register.
ADYB b	Adds the content of the Y register to immediate data b, then stores the result in the Y register.
SBYB b	Subtracts immediate data b from the content of the Y register, then stores the result in the Y register.
INC x, y	Increments the content of work RAM specified by instruction codes x and y. An overflow sets Ca to “1”. The content of work RAM after incrementing is stored in Acc.
DEC x, y	Decrements the content of work RAM specified by instruction codes x and y. A borrow sets Ca to “1”. The content of work RAM after decrementing is stored in Acc.

Operation Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cy-cles
TINC	00 0000 0011 1000	Acc ← R (X, Y) + 1 R (X, Y) ← Acc converted to decimal Y ← Y + 1 Acc ← 0/Ca Hold Acc, Ca ← R (X, Y) + Acc + Ca R (X, Y) ← Acc/RTN	Same as R (X, Y + 1)	JF	*	—	—	Y + 1	6
TDEC	00 0000 0011 1001	Acc ← R (X, Y) − 1 R (X, Y) ← Acc converted to decimal Y ← Y + 1 Acc ← 0/Ca Hold Acc, Ca ← R (X, Y) − Acc − Ca R (X, Y) ← Acc/RTN	Same as R (X, Y + 1)	JF	*	—	—	Y + 1	6
ADD b	00 1011 0000 bbbb	Acc ← R (T, Y) / Ca Hold Acc, Ca ← R (X, Y) + Acc + Ca R (X, Y) ← Acc converted to decimal NOP Y + b Y ← Y + 1/JF Check /JF NOP RTN JF	Result of operation		—	—	—	16 − b + 1	*
SUB b	00 1011 0001 bbbb	Acc ← R (T, Y) / Ca Hold Acc, Ca ← R (X, Y) − Acc − Ca R (X, Y) ← Acc converted to decimal NOP Y + b Y ← Y + 1/JF Check /JF NOP RTN JF	Result of operation		—	—	—	16 − b + 1	*
CMBA b	00 0001 0010 bbbb	NOP Acc, Ca ← b − Acc 0 − Acc RTN	b − Acc	*	*	—	—	—	4
CMBR b	00 0001 0011 bbbb	Acc ← R (X, Y) Acc, Ca ← b − Acc 0 − Acc RTN	b − R (X, Y)	*	*	—	—	—	4
CMBT b	00 0001 1010 bbbb	NOP T ← T − b T − 1 RTN	—	—	*	T − b	—	—	4

Operation Instructions

<table>
<tr><th>Mnemonic</th><th>Function</th></tr>
<tr><td>TINC</td><td>Decimally increments the content of work RAM specified by X and Y registers. If an overflow occurs, this instruction hexadecimally increments the content of work RAM specified by the X register and (Y register + 1).</td></tr>
<tr><td>TDEC</td><td>Decimally decrements the content of work RAM specified by X and Y registers. If a borrow occurs, this instruction hexadecimally decrements the content of work RAM specified by the X register and (Y register + 1).</td></tr>
<tr><td>ADD b</td><td>Adds the contents of work RAM specified by X and Y registers to the content of work RAM specified by T and Y registers, and the Ca register. If an overflow occurs, the jump flag (JF) is set and stored in the Ca register. The content of the Y register is then incremented by 1 and the above operations repeated until the value of the Y register is (16-b).

Example: MVBW 0, 1
MVBT 1
ADD 13
Repeated until (16 − 13 = 3)
→ Y = 3

Work RAM before execution
<table>
<tr><th>3</th><th>2</th><th>1</th><th>0</th><th>y/x</th></tr>
<tr><td>A_3</td><td>A_2</td><td>A_1</td><td></td><td>0</td></tr>
<tr><td>B_3</td><td>B_2</td><td>B_1</td><td></td><td>1</td></tr>
</table>
Work RAM after execution
$C_n = (A_n + B_n)$
<table>
<tr><th>3</th><th>2</th><th>1</th><th>0</th><th>y/x</th></tr>
<tr><td>C_3</td><td>C_2</td><td>C_1</td><td></td><td>0</td></tr>
<tr><td>B_3</td><td>B_2</td><td>B_1</td><td></td><td>1</td></tr>
</table></td></tr>
<tr><td>SUB b</td><td>Subtracts the content of work RAM specified by T and Y registers, and the Ca register, from the contents of work RAM specified by X and Y registers. If a borrow occurs, the jump flag (JF) is set and stored in the Ca register. The content of the Y register is then incremented by 1 and the above operations repeated until the value of the Y register is (16-b).

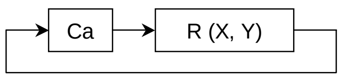
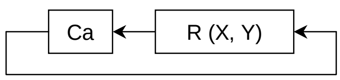
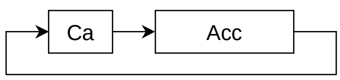
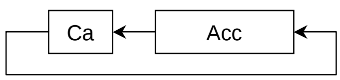
Example: MVBW 2, 1
MVBT 3
SUB 13
Repeated until (16 − 13 = 3)
→ Y = 3

Work RAM before execution
<table>
<tr><th>3</th><th>2</th><th>1</th><th>0</th><th>y/x</th></tr>
<tr><td>A_3</td><td>A_2</td><td>A_1</td><td></td><td>2</td></tr>
<tr><td>B_3</td><td>B_2</td><td>B_1</td><td></td><td>3</td></tr>
</table>
Work RAM before execution
$Cn = (A_n - B_n)$
<table>
<tr><th>3</th><th>2</th><th>1</th><th>0</th><th>y/x</th></tr>
<tr><td>C_3</td><td>C_2</td><td>C_1</td><td></td><td>2</td></tr>
<tr><td>B_3</td><td>B_2</td><td>B_1</td><td></td><td>3</td></tr>
</table></td></tr>
<tr><td>CMBA b</td><td>Compares Acc with immediate data b and sets the jump flag (JF) if they are not the same</td></tr>
<tr><td>CMBR b</td><td>Compares the content of work RAM specified by the X and Y registers with immediate data b and sets the jump flag (JF) if they are not the same</td></tr>
<tr><td>CMBT b</td><td>Compares T register with immediate data b and sets the jump flag (JF) if they are the same</td></tr>
</table>

Bit Manipulation Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cy-cles
SET n, x, y	01 00nn xxxx yyyy	NOP Acc ← R (x, y) or n R (x, y) ← Acc RTN	R (x, y) or n	0	—	—	—	—	4
RESET n, x, y	01 01nn xxxx yyyy	Acc ← n Acc ← {R (x, y) or n} – Acc R (x, y) ← Acc RTN	R (x, y) and /n	0	—	—	—	—	4
CHECK n, x, y	01 10nn xxxx yyyy	R (x, y) or n – R (x, y) – 1 RTN	—	—	*	—	—	—	2
BTS b	00 0001 1110 bbbb	T ← R (X, Y) NOP R (X, Y) ← b or T RTN	—	—	*	R (X, Y) before execution	—	—	4
BTR b	00 0001 1111 bbbb	T ← R (X, Y) NOP R (X, Y) ← b or T – b RTN	—	—	*	R (X, Y) before execution	—	—	4
BTC b	00 0001 1011 bbbb	T ← b NOP {R (X, Y) or T} – R (X, Y) – 1 RTN	—	—	*	b	—	—	4
CAS	00 0000 0001 0001	Ca ← 1 RTN	—	1	—	—	—	—	2
CAR	00 0000 0000 0001	Ca ← 0 RTN	—	0	—	—	—	—	2
RTRR	00 0001 0000 1000	Acc ← R (X, Y) /Ca Hold R (X, Y) ← Acc, Ca Rotate RIGHT/RTN	R (X, Y) before execution	*	—	—	—	—	2
RTRL	00 0001 0000 1001	Acc ← R (X, Y) /Ca Hold R (X, Y) ← Acc, Ca Rotate LEFT/RTN	R (X, Y) before execution	*	—	—	—	—	2
RTAR	00 0001 0000 1010	NOP Acc, Ca Rotate RIGHT/ RTN	ROTATED DATA		—	—	—	—	2
RTAL	00 0001 0000 1011	NOP Acc, Ca Rotate LEFT/ RTN	ROTATED DATA		—	—	—	—	2

Bit Manipulation Instructions

<table>
<tr><th>Mnemonic</th><th colspan="2">Function</th></tr>
<tr><td>SET n, x, y</td><td>Sets only the bit specified in immediate data n of the work RAM specified in instruction codes x and y to “1”.</td><td rowspan="3">The following table shows the relationship between immediate data n and the instruction codes.
<table><tr><td>Immediate data n</td><td>8</td><td>4</td><td>2</td><td>1</td></tr><tr><td>Machine language nn</td><td>11</td><td>10</td><td>01</td><td>00</td></tr></table></td></tr>
<tr><td>RESET n, x, y</td><td>Sets only the bit specified in immediate data n of the work RAM specified in instruction codes x and y to “0”.</td></tr>
<tr><td>CHECK n, x, y</td><td>Sets the JF if the bit specified in immediate data n of the work RAM specified in instruction codes x and y is “1”.</td></tr>
<tr><td>BTS b</td><td>Calculates the logical sum of the content of work RAM specified by the X and Y registers and immediate data b. The result is stored in the work RAM specified in the X and Y registers.</td><td>Example:
<table><tr><td>R (X, Y)</td><td>0101</td><td>0101</td></tr><tr><td>b</td><td>1100</td><td>1010</td></tr><tr><td>Result of execution</td><td>1101</td><td>1111</td></tr></table></td></tr>
<tr><td>BTR b</td><td>Calculates the logical product of the content of work RAM specified by the X and Y registers and the inverted value of immediate data b. The result is stored in the work RAM specified in the X and Y registers.</td><td>Example:
<table><tr><td>R (X, Y)</td><td>0101</td><td>0101</td></tr><tr><td>b</td><td>1100</td><td>1010</td></tr><tr><td>Result of execution</td><td>0001</td><td>0101</td></tr></table></td></tr>
<tr><td>BTC b</td><td>The JF is set when the logical sum of the content of work RAM specified by the X and Y registers and the inverted value of immediate data b is “1111”.</td><td>Example:
<table><tr><td>R (X, Y)</td><td>0101</td><td>0101</td></tr><tr><td>b</td><td>0100</td><td>1010</td></tr><tr><td>Result of execution</td><td>1111</td><td>0101</td></tr><tr><td></td><td>JF</td><td>NO JF</td></tr></table></td></tr>
<tr><td>CAS</td><td colspan="2">Sets the Ca register to “1”</td></tr>
<tr><td>CAR</td><td colspan="2">Sets the Ca register to “0”</td></tr>
<tr><td>RTRR</td><td></td><td>Rotates the contents of work RAM specified by the X and Y registers and the Ca register</td></tr>
<tr><td>RTRL</td><td></td><td>Rotates the contents of work RAM specified by the X and Y registers and the Ca register</td></tr>
<tr><td>RTAR</td><td></td><td>Rotates Acc with the Ca register</td></tr>
<tr><td>RTAL</td><td></td><td>Rotates Acc with the Ca register</td></tr>
</table>

I/O Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cycles
IIN	00 1000 1011 0000	Acc ← IN1 RTN	IN1	0	—	—	—	—	2
INIO1	00 1000 1011 0001	Acc ← IO1 RTN	IO1	0	—	—	—	—	2
INIO2	00 1000 1011 0010	Acc ← IO2 RTN	IO2	0	—	—	—	—	2
INIO3	00 1000 1011 0011	Acc ← IO3 RTN	IO3	0	—	—	—	—	2
OIA1	00 1000 1010 0001	NOP IO1 ← Acc/RTN	—	—	—	—	—	—	2
OIA2	00 1000 1010 0010	NOP IO2 ← Acc/RTN	—	—	—	—	—	—	2
OIA3	00 1000 1010 0011	NOP IO3 ← Acc/RTN	—	—	—	—	—	—	2
OTA1	00 1000 1110 0000	NOP OT1 ← Acc/RTN	—	—	—	—	—	—	2
OTA2	00 1000 1110 0001	NOP OT2 ← Acc/RTN	—	—	—	—	—	—	2

I/O Instructions

Mnemonic	Function
IIN	Fetches data to the Acc from the IN port
INIO1	Fetches data to the Acc from port IO1 (Note) Execute the output instruction [OIB1 15] before the input instruction.
INIO2	Fetches data to the Acc from port IO2 (Note) Execute the output instruction [OIB2 15] before the input instruction.
INIO3	Fetches data to the Acc from port IO3 (Note) Execute the output instruction [OIB3 15] before the input instruction.
OIA1	Outputs the content of the Acc to port IO1 Depending on the higher 2 bits of the Acc, the HCLK terminal is set.
OIA2	Outputs the content of the Acc to port IO2
OIA3	Outputs the lower 1 bit of the content of the Acc to port IO3 Depending on the lower 1 bit of the Acc, the HCLK terminal is set.
OTA1	Outputs the content of the Acc to port O1.
OTA2	The lower 1 bit of the Acc sets IO11 terminal to be 3-state controlled.

I/O Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cy-cles
OIB1 b	00 1001 0001 bbbb	Acc ← b NOP IO1 ← Acc RTN	b	0	—	—	—	—	4
OIB2 b	00 1001 0010 bbbb	Acc ← b NOP IO2 ← Acc RTN	b	0	—	—	—	—	4
OIB3 b	00 1001 0011 bbbb	Acc ← b NOP IO3 ← Acc RTN	b	0	—	—	—	—	4
OTB1 b	00 1001 0100 bbbb	Acc ← b NOP OT1 ← Acc RTN	b	0	—	—	—	—	4
OTB2 b	00 1001 0101 bbbb	Acc ← b NOP OT2 ← Acc RTN	b	0	—	—	—	—	4
OIR	00 1011 1010 0000	NOP NOP NOP Acc ← R (X, Y) Y ← Y + 1 IO1 ← Acc/Acc ← R (X, Y) NOP IO2 ← Acc/RTN	R (X, Y + 1)	0	—	—	—	Y + 1	8

Timer Instruction

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cy-cles
500 MS	00 0000 0000 0100	500 ms TIMER F/F Check RTN	—	—	*	—	—	—	2
125 MS	00 0000 0000 0101	125 ms TIMER F/F Check RTN	—	—	*	—	—	—	2
32 MS	00 0000 0000 0110	31.25 ms TIMER F/F Check RTN	—	—	*	—	—	—	2
TIM	00 0000 0001 0000	TIM ← Acc RTN	—	—	—	—	—	—	2

I/O Instructions

Mnemonic	Function
OIB1 b	Outputs immediate data b to port IO1 via the Acc Depending on the higher 2 bits of immediate data, the HCLK terminal is set.
OIB2 b	Outputs immediate data b to port IO2 via the Acc
OIB3 b	Outputs the lower 1 bit of immediate data b to port IO3 via the Acc Depending on the lower 1 bit of immediate data b, the HCLK terminal is set.
OTB1 b	Outputs immediate data b to port O1 via the Acc
OTB2 b	The lower 1 bit of immediate data b sets IO11 terminal to be 3-state controlled.
OIR	Outputs the contents of work RAM specified by the X and Y registers to port IO1, then increments the Y register and outputs the content of work RAM specified by the X and Y registers to port IO2.

Timer Instruction

Mnemonic	Function
500 MS	Sets the JF when the 500 ms timer F/F is “1”, then resets the timer F/F
125 MS	Sets the JF when the 125 ms timer F/F is “1”, then resets the timer F/F
32 MS	Sets the JF when the 31.25 ms timer F/F is “1”, then resets the timer F/F
TIM	Sends the lower 3 bits of the Acc to the TIM register. STOP mode is exited when the timer F/F corresponding to the TIM register bits is set.

Branch Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cycles
ROM	00 0000 0011 0000	NOP PC modification ← Acc, Ca NOP NOP NOP RTN	—	—	—	—	—	—	6
JUMP adr	11 adr	IF JF = 0 PC0~PC11 ← adr RTN	—	—	—	—	—	—	2
CALL1 adr	10 00 adr	NOP Y ← 0 R (7, Y) ← 15/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← 0 NOP RTN	—	—	—	—	—	0	8
CALL2 adr	10 01 adr	NOP Y ← 4 R (7, Y) ← 15/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← 0 NOP RTN	—	—	—	—	—	0	8
CALL3 adr	10 10 adr	NOP Y ← 8 R (7, Y) ← 15/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← 0 NOP RTN	—	—	—	—	—	0	8
CALL4 adr	10 11 adr	NOP Y ← 12 R (7, Y) ← 15/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← Y + 1 R (7, Y) ← PC + 1/Y ← 0 NOP RTN	—	—	—	—	—	0	8
RET1	00 1011 1000 0000	NOP Y ← 0 Y ← Y + 1 PC ← R (7, Y) /Y ← Y + 1 PC ← R (7, Y) /Y ← Y + 1 PC ← R (7, Y) /Y ← 0 NOP RTN	—	—	—	—	—	0	8

Branch Instructions

Mnemonic	Function
ROM	The instruction to be executed after that in the step following the [ROM] instruction is at the address indicated by replacing the lower 5 bits of the address following the [ROM] instruction with the contents of the Ca and Acc. Then returns to the address at which the ROM instruction was executed, plus 2. However, if a branch instruction such as [JUMP], [CALL] or [RET] is executed at the branch destination, execution is transferred to the address specified in that branch instruction.
JUMP adr	If JF was not set by the instruction executed before “JUMP adr”, execution jumps to the address specified in adr.
CALL1 adr	The address of this instruction + 1 is stored in work RAM at line X = 7, Y = 0 to 3, and execution branches to the address specified in the instruction code (lower 10 bits of address = adr, all others are “1”).
CALL2 adr	The address of this instruction + 1 is stored in work RAM at line X = 7, Y = 4 to 7, and execution branches to the address specified in the instruction code (lower 10 bits of address = adr, all others are “1”).
CALL3 adr	The address of this instruction + 1 is stored in work RAM at line X = 7, Y= 8 to 11, and execution branches to the address specified in the instruction code (lower 10 bits of address = adr, all others are “1”).
CALL4 adr	The address of this instruction + 1 is stored in work RAM at line X = 7, Y = 12 to 15, and execution branches to the address specified in the instruction code (lower 10 bits of address = adr, all others are “1”).
RET1	Returns to address stored in work RAM by “CALL1”

Branch Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cycles
RET2	00 1011 1000 0001	NOP Y ← 4 PC ← R (7, Y) /Y ← Y + 1 PC ← R (7, Y) /Y ← Y + 1 PC ← R (7, Y) /Y ← 0 NOP RTN	—	—	—	—	—	0	8
RET3	00 1011 1000 0010	NOP Y ← 8 Y ← Y + 1 PC ← R (7, Y) /Y ← Y + 1 PC ← R (7, Y) /Y ← Y + 1 PC ← R (7, Y) /Y ← 0 NOP RTN	—	—	—	—	—	0	8
RET4	00 1011 1000 0011	NOP Y ← 12 PC ← R (7, Y) /Y ← Y + 1 PC ← R (7, Y) /Y ← Y + 1 PC ← R (7, Y) /Y ← 0 NOP RTN	—	—	—	—	—	0	8

Operating Mode Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cycles
FCR	00 0000 0000 1001	System clock ← High-speed clock RTN	—	—	—	—	—	—	2
FXT	00 0000 0000 1000	System clock ← Low-speed clock RTN	—	—	—	—	—	—	2
STOP	00 0000 0011 0011	NOP NOP STOP NOP NOP RTN	—	—	—	—	—	—	6
OFF	00 0000 0011 0001	NOP NOP OFF NOP NOP RET	—	—	—	—	—	—	6

Branch Instructions

Mnemonic	Function
RET2	Returns to address stored in work RAM by “CALL2”.
RET3	Returns to address stored in work RAM by “CALL3”.
RET4	Returns to address stored in work RAM by “CALL4”.

Operating Mode Instructions

Mnemonic	Function
FCR	Generates the system clock from the high-speed clock. The CPU operates at high speed.
FXT	Generates the system clock from the low-speed clock. The CPU operates at low speed.
STOP	Transfers to STOP mode
OFF	Transfers to OFF mode

Display Control Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cy-cles
DON	00 0000 0000 1011	Display ON RTN	—	—	—	—	—	—	2
DOFF	00 0000 0000 1010	Display OFF RTN	—	—	—	—	—	—	2
DPON	00 0000 0001 1011	Display power supply ON RTN	—	—	—	—	—	—	2
DPOFF	00 0000 0001 1010	Display power supply OFF RTN	—	—	—	—	—	—	2
DXT	00 0000 0001 1000	Display timing ← Low-speed clock RTN	—	—	—	—	—	—	2

Miscellaneous Instructions

Mnemonic	Machine language	Microinstructions	Acc	Ca	JF	T	X	Y	Cy-cles
DLY b	00 0000 0111 bbbb	T ← T + 1 JF Check (/JF: back to T ← T + 1; JF: to T ← b) T ← b RTN	—	—	—	b	—	—	*
NOP	00 0000 0000 0000	NOP RTN	—	—	—	—	—	—	2

Display Control Instructions

Mnemonic	Function
DON	Sets the common pulse level to ON
DOFF	Sets the common pulse level to OFF. Common pulse = V_2
DPON	Turns ON the display power supply and generates V_1, V_2, and V_3
DPOFF	Turns OFF the display power supply and stops generating V_1, V_2, and V_3
DXT	Generates the LCD drive timing from the low-speed clock

Miscellaneous Instructions

Mnemonic	Function
DLY b	Makes a delay of $\{(16 - T) \times 2 + 2\} \times$ machine cycles
NOP	No operation

T6C96A Electrical Characteristics Tables

Absolute Maximum Ratings (V_{SS} = 0 V)

Item	Symbol	Rating	Unit
Power Supply Voltage	V_{DD}	−0.3~6.0	V
Input Voltage	V_{IN}	−0.3~V_{DD} + 0.3	V
Operating Temperature	T_{opr}	0~40	°C
Storage Temperature	T_{stg}	−55~125	°C

Recommended Operating Conditions (V_{SS} = 0 V, Ta = 25°C)

Item	Symbol	Applicable terminal	Test circuit	Conditions	Min	Typ.	Max	Unit
Power Supply Voltage	V_{DD}	V_{DD}	—	f_{CR} = 3.58 MHz	4.5	5.0	5.5	V
"H" Input Voltage	V_{IH}	/AC, IN, IO	—	—	V_{DD} × 0.75	—	V_{DD}	V
"L" Input Voltage	V_{IL}	/AC, IN, IO	—	—	0	—	V_{DD} × 0.25	V
High-Speed Clock CR Oscillation Frequency	f_{CR}	R_{IN}, R_{OUT}	—	V_{DD} = 5.0 V R = 9.1 kΩ	3.22	3.58	3.93	MHz
Low-Speed Clock CR Oscillation Frequency (Low-Speed CR)	f_{CRL}	—	—	V_{DD} = 5.0 V Built in CR	24.6	32.8	42.6	kHz
Frame Frequency	F_R	—	—	f_{CRL} = 32.8 kHz	48	64	83.2	Hz

DC Characteristics (Current Consumption) (V_{SS} = 0 V, Ta = 25°C)

MCU Block

Item	Symbol	Applicable terminal	Test circuit	Conditions	Min	Typ.	Max	Unit
Current Consumption High-Speed Operating Mode	$I_{DD\ (OPH)}$	—	—	V_{DD} = 5.0 V f_{CR} = 3.58 MHz	—	800	1600	μA
Current Consumption Low-Speed Operating Mode	$I_{DD\ (OPL)}$	—	—	V_{DD} = 5.0 V f_{CRL} = 32.8 kHz (typ.) Built in CR	—	33	66	μA
Current Consumption STOP Mode	$I_{DD\ (STOP)}$	—	—	V_{DD} = 5.0 V f_{CRL} = 32.8 kHz (typ.) Built in CR	—	30	60	μA
Current Consumption OFF Mode	$I_{DD\ (OFF)}$	—	—	V_{DD} = 5.0 V	—	0.01	3	μA

Note 11: The overall consumption of the device is the current consumed by the MCU plus that consumed by the LCD driver.

DC Characteristics (Current Consumption) (V_{SS} = 0 V, Ta = 25°C)

LCD Driver Block

Item	Symbol	Applicable terminal	Test circuit	Conditions	Min	Typ.	Max	Unit
Current Consumption Display Power Supply ON	ILCD (ON)	—	—	V_{DD} = 5.0 V	—	14	26	μA
Current Consumption Display Power Supply OFF	ILCD (OFF)	—	—	V_{DD} = 5.0 V	—	0.01	3	μA

DC Characteristics (Terminal Ability) (V_{SS} = 0 V, V_{DD} = 5.0 V, Ta = 25°C)

Item	Symbol	Applicable terminal	Test circuit	Conditions	Min	Typ.	Max	Unit
"H" Output Voltage	V_{OH}	O11, O12, O13, O14, IO11, IO12, IO13, IO14, HCLK	—	—	V_{DD} − 0.2	—	V_{DD}	V
"L" Output Voltage	V_{OL}	O11, O12, O13, O14, IO11, IO12, IO13, IO14, HCLK	—	—	0	—	0.2	V
"H" Output Current	I_{OH}	HCLK	—	V_{DD} = 5.0 V V_{OH} = 4.5 V	—	—	−2	mA
		IO11, O11, O12, O13, O14			—	—	−800	μA
"L" Output Current	I_{OL}	HCLK	—	V_{DD} = 5.0 V V_{OL} = 0.5 V	2	—	—	mA
		IO11, O11, O12, O13, O14			800	—	—	μA
"H" Output Current	I_{OH}	IO12, IO13, IO14, IO21, IO22, IO23, IO24, IO31	—	V_{DD} = 5.0 V V_{OH} = 4.5 V	—	—	−11	μA
"L" Output Current	I_{OL}	IO12, IO13, IO14, IO21, IO22, IO23, IO24, IO31	—	V_{DD} = 5.0 V V_{OL} = 0.5 V	800	—	—	μA
Pull-Up Resistance	RPU	/AC, IN, IN11, IN12, IN13, IN14	—	V_{DD} = 5.0 V	42	60	78	kΩ
Input Leakage Current	I_{IH} I_{IL}	IO11	—	V_{DD} = 5.0 V V_{IN} = 0 V or 5.0 V	−1	—	1	μA

DC Characteristics (Terminal Ability) (V_{SS} = 0 V, Ta = 25°C)

LCD Driver Block

Item	Symbol	Applicable terminal	Test circuit	Conditions	Min	Typ.	Max	Unit
"H" Output Voltage	V_{OH}	SEG, COM	—	—	V_{DD} − 0.2	—	V_{DD}	V
"L" Output Voltage	V_{OL}	SEG, COM	—	—	0	—	0.2	V
"H" Output Current	I_{OH}	SEG, COM	—	V_{DD} = 5.0 V V_{OH} = V_{DD} − 0.5 V	—	—	−300	μA
"L" Output Current	I_{OL}	SEG, COM	—	V_{DD} = 5.0 V V_{OL} = V_{SS} + 0.5 V	300	—	—	μA

Addendum

Pad coordinates

Chip size : 2.82 × 2.82 (mm)
Chip thickness : 450 (μm)
Pad size : 100 × 100 (μm)
Number of pads : 54
Substrate voltage : V_{SS}

(Unit: μm)

PAD NAME	X POINT	Y POINT
C_1	−901	−1167
/AC	−751	−1167
V_{DD}	−600	−1167
R_{OUT}	−450	−1167
R_{IN}	−300	−1167
HCLK	−150	−1167
TS1	32	−1167
TS2	182	−1167
TS3	333	−1167
O11	483	−1167
O12	633	−1167
O13	783	−1167
O14	933	−1167
V_{SS}	1163	−975
IO11	1163	−825
IO12	1163	−675
IO13	1163	−525
IO14	1163	−375
IO21	1163	−225
IO22	1163	−75
IO23	1163	75
IO24	1163	225
IO31	1163	375
IN11	1163	526
IN12	1163	676
IN13	1163	826
IN14	1163	976

PAD NAME	X POINT	Y POINT
S_1	901	1167
S_2	751	1167
S_3	600	1167
S_4	450	1167
S_5	300	1167
S_6	150	1167
S_7	0	1167
S_8	−150	1167
S_9	−300	1167
S_{10}	−450	1167
S_{11}	−600	1167
S_{12}	−751	1167
S_{13}	−901	1167
S_{14}	−1163	976
S_{15}	−1163	826
S_{16}	−1163	676
S_{17}	−1163	526
S_{18}	−1163	375
S_{19}	−1163	225
S_{20}	−1163	75
S_{21}	−1163	−75
S_{22}	−1163	−225
S_{23}	−1163	−375
S_{24}	−1163	−525
C_4	−1163	−675
C_3	−1163	−825
C_2	−1163	−975

RESTRICTIONS ON PRODUCT USE

000707EBA