1. DESCRIPTION

DMC6830 is a remote control transmitter, consists of the optimized 4-bit CPU with ROM and RAM. It contains power-on reset, watchdog timer and carrier frequency generator. The DMC6830 provide a various carrier frequency for encoding output of key matrix and has built-in transistor to drive infrared LED. The DMC6830 is supported with a software development tool, which allows code development in a PC environment. It allows the user to simulate the DMC6830 on an instruction level.

2. FEATURES

- ❑ Number of basic instructions---------- 45
- ❑ Instruction cycle time (one word instruction)
  - At Fsys=480KHz---------- 16.67uS
  - At Fsys=455kHz---------- 17.58uS
- ❑ Memory size
  - ROM---------- 1024 x 8 Bits
  - RAM---------- 32 x 4 Bits
- ❑ Input ports (D0 ~ D3, E0 ~ E3 : with pull-up resistor)
- ❑ Output ports (C, G, K, F0 ~ F7)
- ❑ Carrier frequency generator

  Fsys/12 (1/2 duty), Fsys/12 (1/3 duty), Fsys/12 (1/4 duty),
  Fsys/8 (1/2 duty), Fsys/8 (1/4 duty), Fsys/11 (4/11 duty), No carrier
- ❑ Watchdog Timer
- ❑ Built-in power on reset
- ❑ Single power supply ---------- 1.8V ~ 3.6V
- ❑ Power dissipation (stop mode , VDD = 3V)---------- Less than 3uW
- ❑ Package---------- 20/24 DIP, 20/24 SOP
- ❑ Low-power system applications such as an infrared remote controller

❑ MASK OPTION

1. Divide ratio of the oscillator frequency
2. Whether connected infrared LED driver or not

※ Descriptions of this spec sheet assume that the DMC6830 include driver for infrared LED.

3. BLOCK DIAGRAM

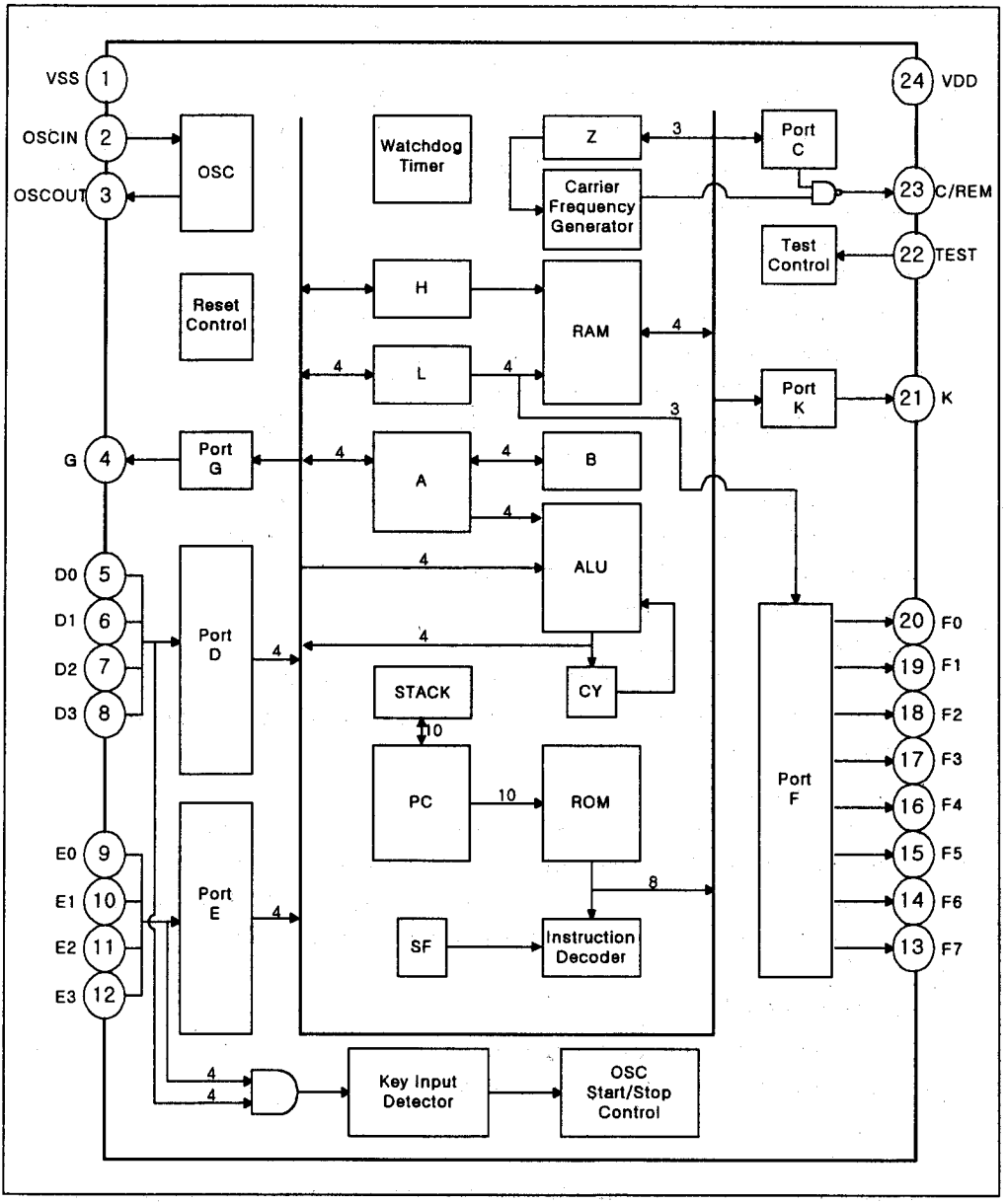


Figure 3.1 Block Diagram of the DMC6830

4. PIN ASSIGNMENT AND DESCRIPTION

4.1 PIN ASSIGNMENT FOR 24PINS (24 DIP, 24 SOP)

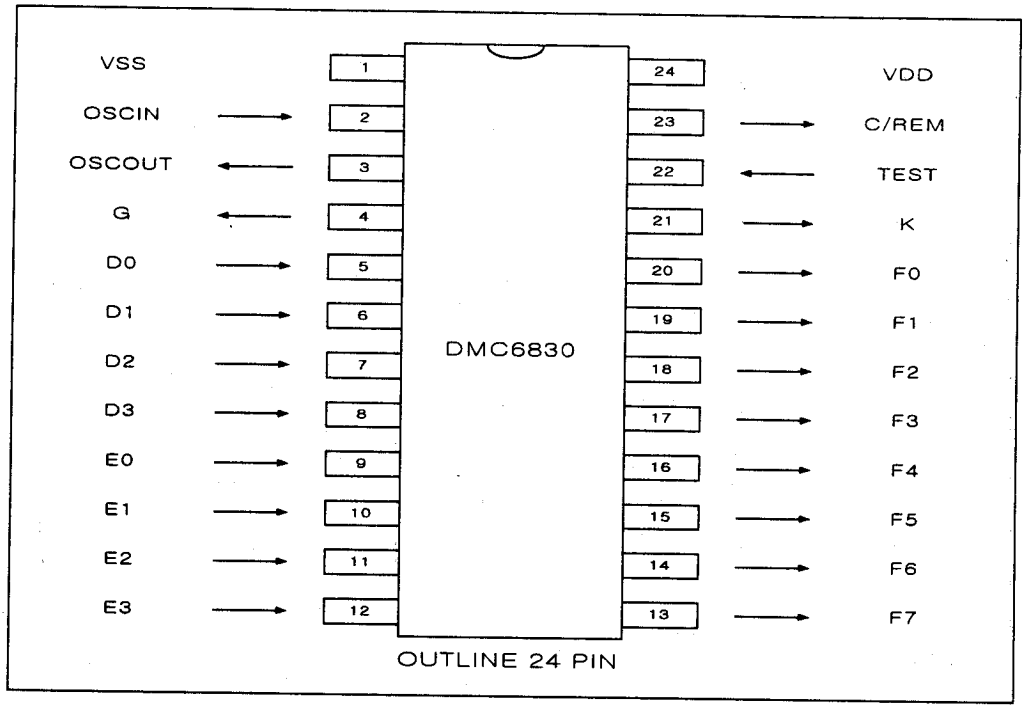


Figure 3-1. Pin Assignment of 24Pins

4.2 PIN DESCRIPTION FOR 24 PINS

Symbol	Pin No.	I /O	Functions	I/O Type
VDD	24	-	Power Supply	
VSS	1	-	Ground	
TEST	22	INPUT	Input for test (Normally connected to VSS)	
OSCIN	2	INPUT	Input for oscillating	
OSCOUT	3	OUTPUT	Output for oscillating	
C/REM	23	OUTPUT	1-Bit output for remote transmission	B
D0 - D3	5 - 8	INPUT	4-Bit input for key sense (with pull-up resistor)	A
E0 - E3	9 – 12	INPUT	4-Bit input for key sense (with pull-up resistor)	A
F0 - F7	20 - 13	OUTPUT	1-Bit individual output for key scan	C
G	4	OUTPUT	1-Bit output	D
K	21	OUTPUT	1-Bit output	D

4.3 PIN ASSIGNMENT (20 DIP, 20 SOP)

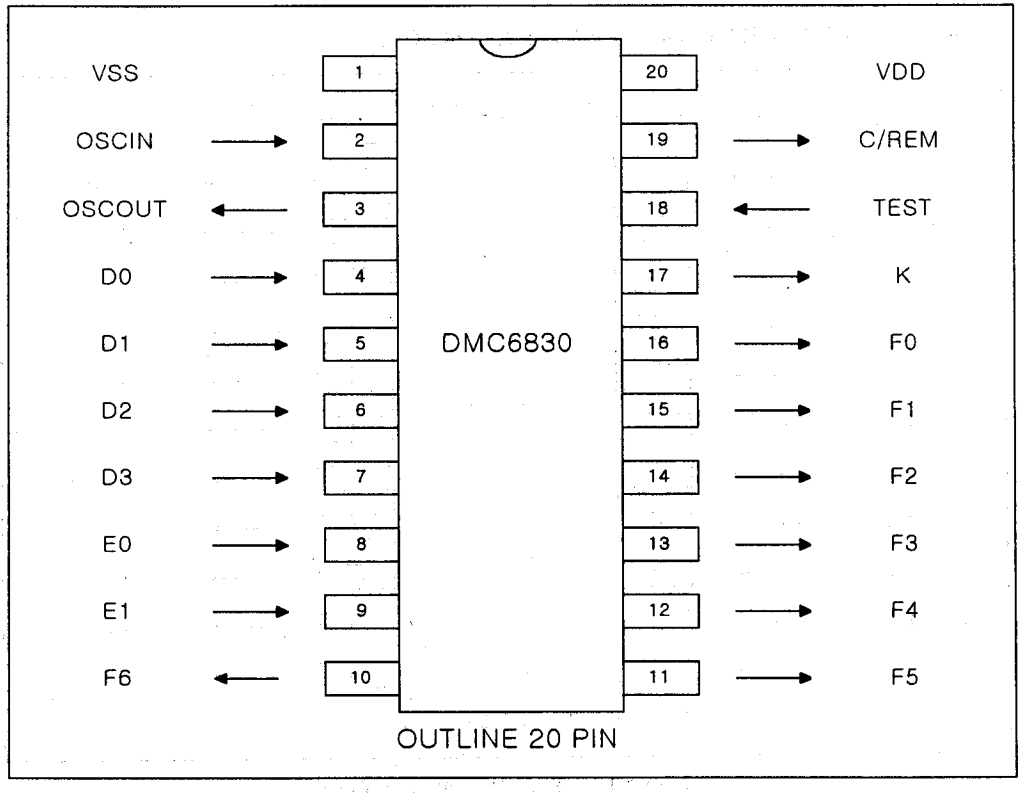


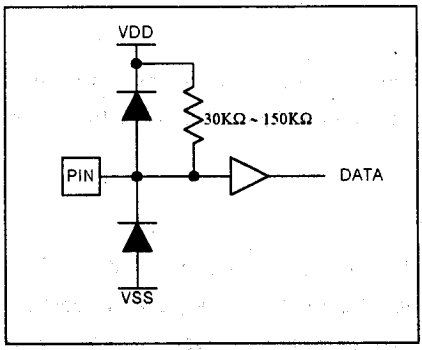
Figure 3-1. Pin Assignment of 20Pin

4.4 PIN DESCRIPTION FOR 20 PINS

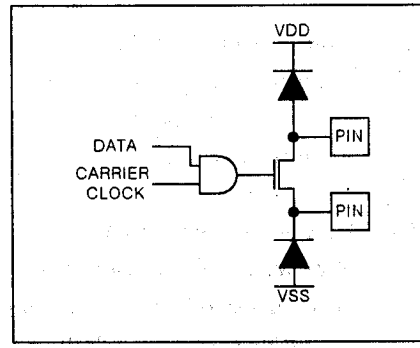
Symbol	Pin No.	I /O	Functions	I/O Type
VDD	20	-	Power Supply	
VSS	1	-	Ground	
TEST	18	INPUT	Input for test (Normally connected to VSS)	
OSCIN	2	INPUT	Input for oscillating	
OSCOUT	3	OUTPUT	Output for oscillating	
C/REM	19	OUTPUT	1-Bit output for remote transmission	B
D0 - D3	4 - 7	INPUT	4-Bit input for key scan (with pull-up resistor)	A
E0 - E1	8 - 9	INPUT	2-Bit input for key scan (with pull-up resistor)	A
F0 - F6	16 - 10	OUTPUT	1-Bit individual output for key scan	C
K	17	OUTPUT	1-Bit output	D

4.5 I/O CIRCUIT SCHEMATICS

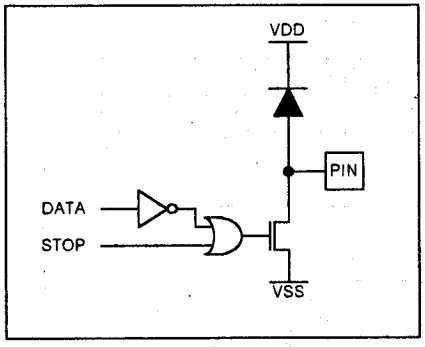
TYPE A



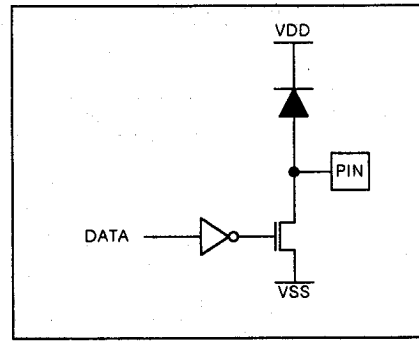
TYPE B



TYPE C



TYPE D



NOTE : If STOP mode is specified, the TYPE C output becomes "L" state and the TYPE B output becomes floating state, the TYPE D output maintains previous state.

Figure 3-1. I/O Circuit Schematics

5. BASIC FUNCTION BLOCK

5.1 Program Counter (PC)

Program counter is used to indicate the address of the next instruction to be executed. The 10-bit program counter consists of two registers, PC_H(4-bit) and PC_L(6-bit). This is a polynomial counter.

5.2 Program Memory (ROM)

Program memory is used to store user-specified program. This consists of a 1024 × 8-bit. It is organized in 16 pages and each page is 64 bytes long. For page-in addressing, all instructions excluding JMPL and CALL can be executed by page. In order to execute jump or call in page, JMP or CAL is suitable. For page-to-page addressing, JMPL or CALL must be used.

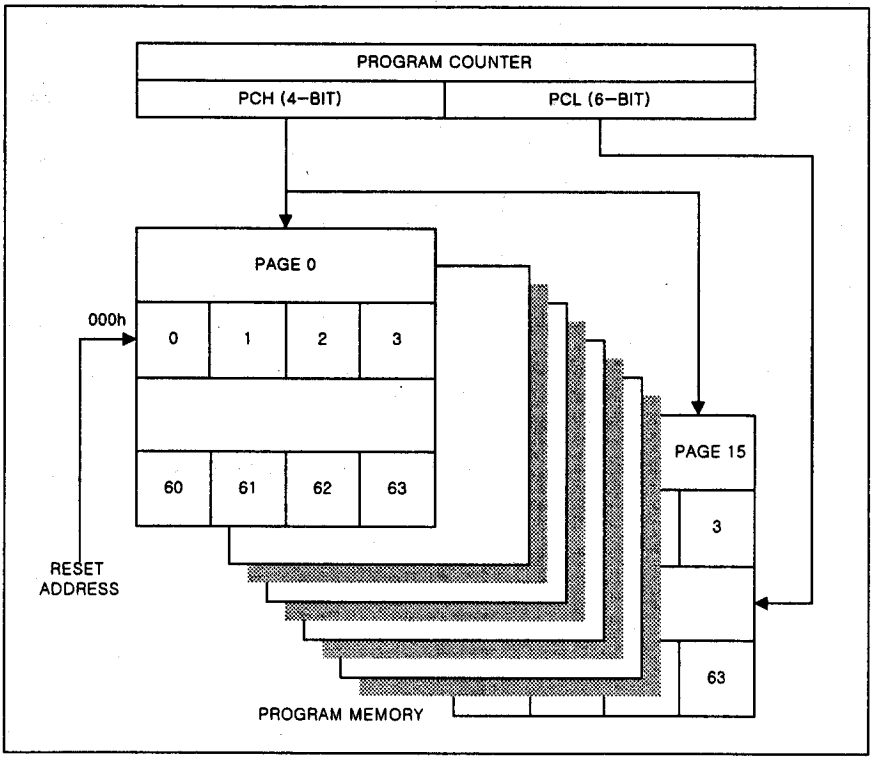


Figure 5-1. Program Memory Map

5.3 Data Memory (RAM)

Data memory is used to store various type of processing data. This consists of a 32-nibble, which is organized into two files of 16 nibbles each. RAM addressing is indirectly implemented by a two registers; H, L. It's upper 1-bit register (H) selects one of two files and its lower 4-bit register (L) selects one of 16 nibbles in the selected file.

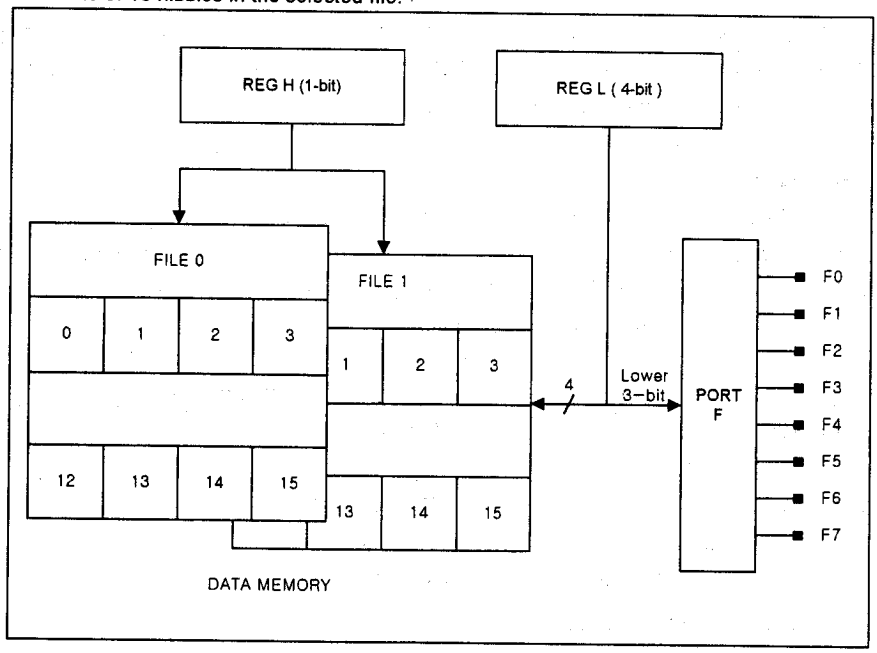


Figure 5-2. Data Memory Map

5.4 Stack Register (SK)

Stack register is used to store return address and provide a particularly mechanism for transferring control between programs. Two level hardware push/pop stacks are manipulated by CAL, CALL, and RET instructions. CAL/CALL instructions push the current program counter value, incremented by "1", into stack level 1. Stack level 1 is automatically pushed to level 2. If more than two subsequent CAL/CALL are executed, only the most recent two return addresses are stored. RET instruction load the contents of stack level 1 into the program counter while stack level 2 gets copied into level 1. If more than two subsequent RET are executed, the stack will be filled with the address previously stored in level 2.

5.5 Arithmetic and Logic Unit (ALU)

This unit is used to perform arithmetic and logical operations such as addition, comparison, and bit manipulation.

5.6 Carry Flag (CY)

The carry flag contains the carry generated by the arithmetic and logical unit immediately after an operation. The set carry (SETB CY) and clear carry (CLRB CY) instructions allow direct access for setting and clearing this flag.

5.7 Skip Flag (SF)

The skip flag is a 1-bit register, which enables programs to conditionally skip an instruction. All instructions are executed when this flag is "0". But if SF is "1", the program executes NOP instruction and resets SF to "0". Then program execution proceeds.
The following instructions affect the skip flag.

Instructions		Set conditions of SF
Arithmetic	ADD n	If carry occurs
	INC L	(L) = 0
Compare	IF0 @HL.b	M[HL].b = 0
	IF0 CY	(CY) = 0
	IFEQU @HL	(A) = M[HL].b
	IFEQU n	(A) = n
Data Transfer	STA @HL+	(L) = 0
	XCH @HL+	(L) = 0

The instructions, which doesn't affect the skip flag but have a skip condition, are as follows.

Instructions		Skip conditions
Data Transfer	LDA n	If it is continuous, skip next same instruction.
	LDL n	If it is continuous, skip next same instruction.
Bit Manipulate	SETB H	If SETB H or CLRB H are continuous, skip next
	CLRB H	SETB H or CLRB H instruction.

5.8 Registers

Register A

Register A, called the accumulator, plays a central role, is used to store an input or an output operand (result) in the execution of most instructions. It consists of 4-bit.

Register B

Register B is used to store a temporary data in CPU. It consists of 4-bit.

Register H

Register H is used to indicate an address of the data memory in conjunction with register L. It consists of 1-bit, which is related with the bit 0 of accumulator.

Register L

Register L is used to indicate an address of the data memory in conjunction with register H, Also lower 3-bit can be used to indicate the bit position of the port F. It consists of 4-bit.

Register Z

Register Z is used to select a carrier frequency. The carrier frequency must be selected before Port C data write operation. It consists of 3-bit.

Register Z			Carrier frequency
Bit 2	Bit 1	Bit 0	
0	0	0	F_{SYS}/12, 1/2 duty
0	0	1	F_{SYS}/12, 1/3 duty
0	1	0	F_{SYS}/12, 1/4 duty
0	1	1	F_{SYS}/8, 1/2 duty
1	0	0	F_{SYS}/8, 1/4 duty
1	0	1	F_{SYS}/11, 4/11 duty
1	1	0	No carrier
1	1	1	No carrier

5.9 I/O Ports

Port C/REM

Port C/REM is a 1-bit output port, which is related with the bit 3 of accumulator, with CMOS N-channel open drain, which have large current sink capability, for I.R.LED drive. This output can be configured as carrier frequency by programming the register Z and port C data. This pin is put into the high-impedance state in stop mode.

Port D

Port D is a 4-bit input port with pull-up resistor. Forcing any input pins to "L" state, system reset occurs and it starts to operate from the reset address.

Port E

Port E is a 4-bit input port with pull-up resistor. Forcing any input pins to "L" state, system reset occurs and it starts to operate from the reset address.

Port F

Port F is an 8-bit output port with N-channel open drain. Each output which specified by the lower 3-bit of register L can be set and reset individually. All F pins are put into the low state in stop mode.

Port G

Port G is a 1-bit output port with N-channel open drain. When stop mode is specified, this pin still remains in the previous state. Set this pin to appropriate state before entering stop mode for visible LED or key scan application.

Port K

Port K is a 1-bit output port with N-channel open drain. When stop mode is specified, this pin still remains in the previous state. Set this pin to appropriate state before entering stop mode for visible LED or key scan application.

5.10 Carrier frequency generator

One of seven carrier frequencies can be selected and transmitted through the C/REM pin by programming the register Z and port C.

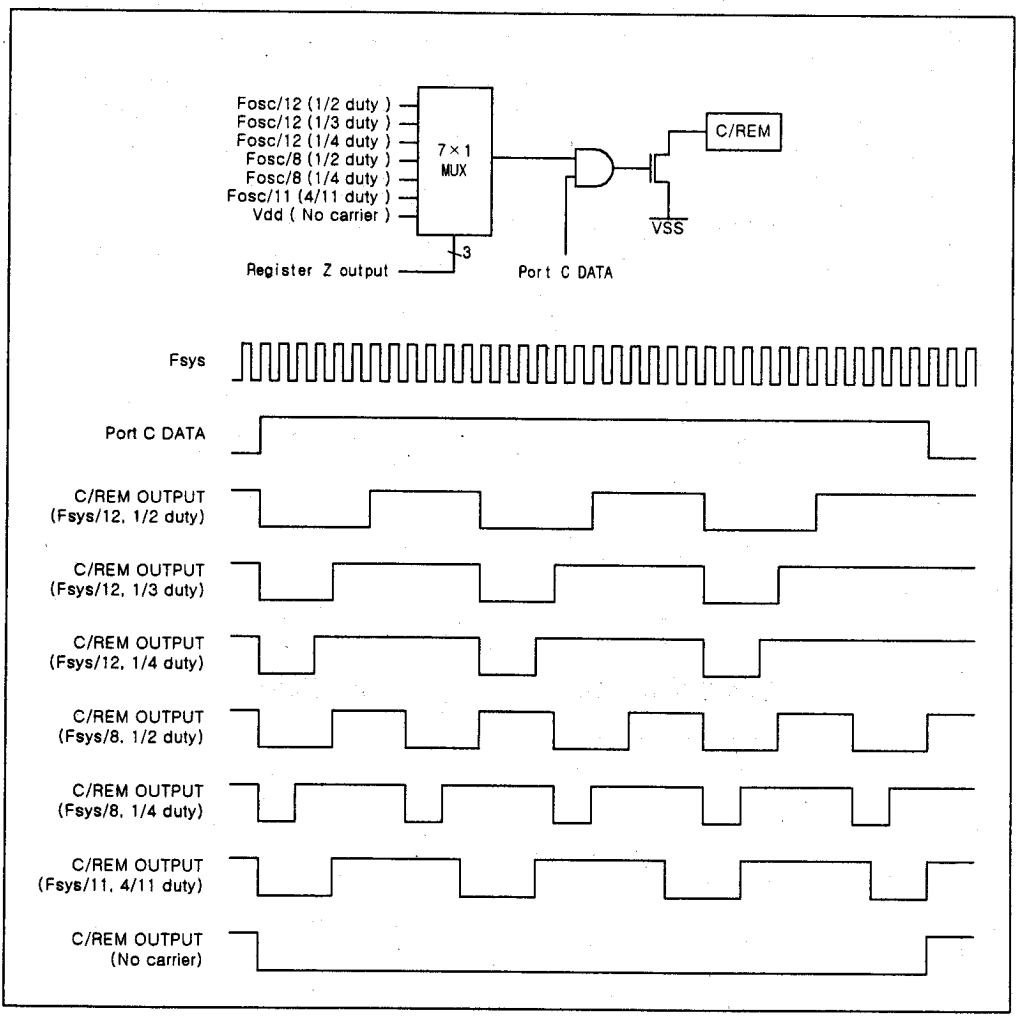


Figure 5-3 PORT C/REM and Carrier Output

5.11 Watchdog timer (WDT)

The watchdog timer provides the means to return to a reset condition when a system malfunction occurs and the program enters an infinite loop caused by noise or any abnormal state. Also this timer have a function of oscillation stabilization timer. This is a 13-bit counter, counts the clock which is divided twelve ($F_{SYS}/12$). In the stop mode the oscillation circuit stops but when a key input is detected (Port D, Port E) oscillation starts. When 12288 clock cycles have been counted, the program will be executed from reset address (000H). If the port C data register's value does not change from "L" to "H" before the timer counts 98304 clock cycles, a device reset condition is generated.

The oscillator stabilization time : $12/F_{SYS} * 2^{10} = 1/F_{SYS} * 12288 = 27mS$ (@455KHz)

The time-out period : $12/F_{SYS} * 2^{13} = 1/F_{SYS} * 98304 = 216mS$ (@455KHz)

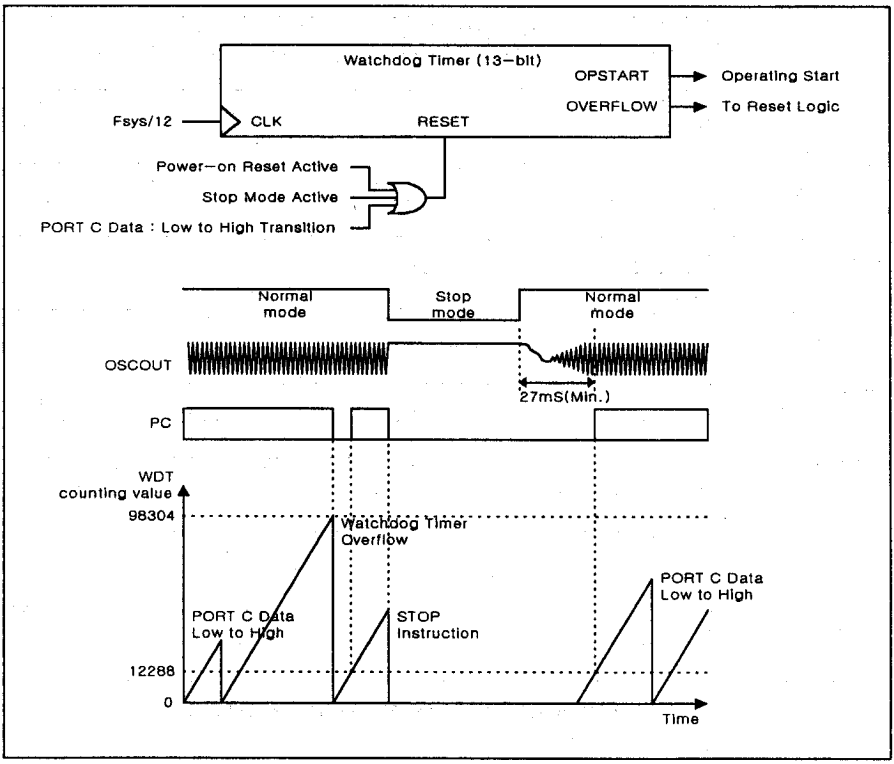


Figure 3-1. Function of Watchdog Timer

5.12 Power-on reset

The DMC6830 incorporates an on-chip power-on reset circuitry which provides internal chip reset for most power-up situations. The power-on reset circuit and the watchdog timer are closely related. On power-up the power-on reset circuit is active and watchdog timer is reset. After the reset time, which is in proportion to the rate of rise of VDD, watchdog timer begins counting. After the oscillator stabilization time, which is typically 27mS in F_{SYS}=455KHz, program execution proceeds from reset address (000H).

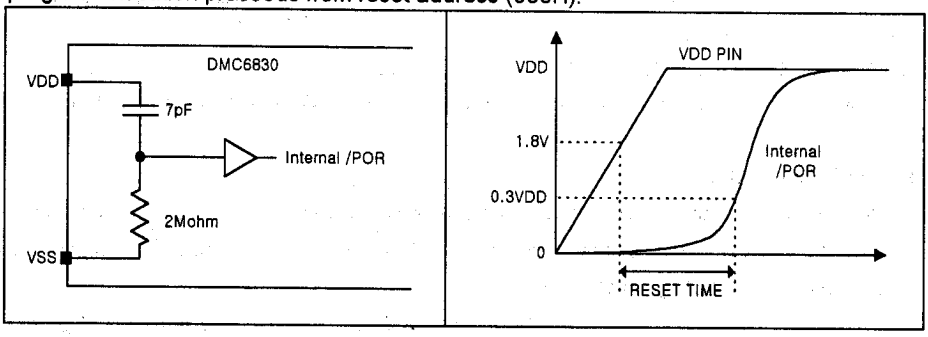


Figure 5-5. Built-in Power-on Reset

5.13 Stop mode

The DMC6830 support the stop mode to reduce power consumption. This mode is entered when the STOP instruction is executed during key inputs are not active. Activating any key inputs (Port D, Port E) the device is awakened from stop mode and restarts to operate from reset address. When the device is released from stop mode, following module's data must set to appropriate value in reset routine: PORT G and PORT K.

In stop mode, the oscillator is stopped and the each port state is as follows.

Port C/REM become inactive state. ("floating" for including I.R.LED driver, "L" otherwise)

Port F become "L" state ("floating" after the reset release)

Port G and Port K retain previous state.

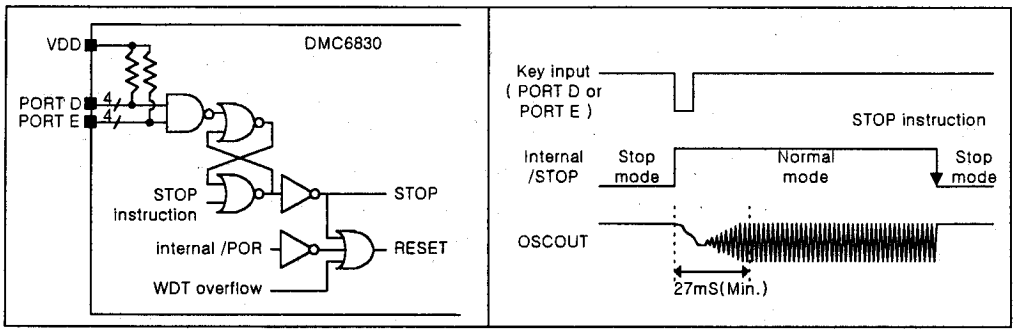


Figure 5-6. Rest structure and Release Timing for STOP Mode to Normal Mode

5.14 OSC Divide Option

The OSC divide option provides a maximum 1MHz system clock (F_{SYS}). F_{OSC} which is generated in oscillation circuit is divided eight or non-divide to produce F_{SYS}. This dividing ratio will be selected by mask option.

F_{OSC} : Oscillator clock, F_{SYS} : System clock (F_{OSC} or $F_{OSC}/8$)

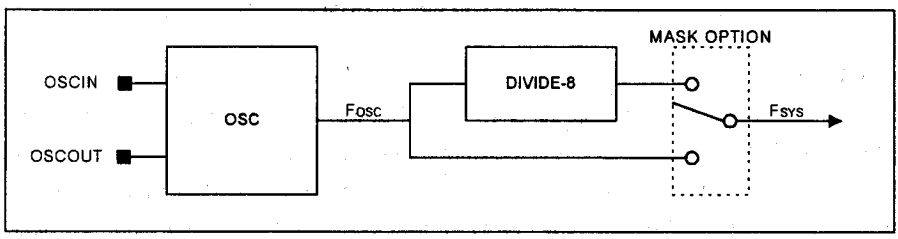


Figure 5-7 OSC Divide Option

6. ELECTRICAL SPECIFICATIONS

6.1 Absolute maximum ratings

Symbols	Parameters	Conditions	Ratings	Units
V_{DD}	Supply Voltage	Ta=25℃	-0.3 ~ 6.0	V
V_I	Input Voltage		-0.3 ~ V_{DD} + 0.3	V
V_O	Output Voltage		-0.3 ~ V_{DD} + 0.3	V
T_{OPR}	Operating temperature	—	-20 ~ 85	℃
T_{STG}	Storage Temperature	—	-40 ~ 125	℃

6.2 Recommended operating conditions

(V_{DD} =3V ± 10%, Ta= -20 ~ 70℃, unless otherwise noted)

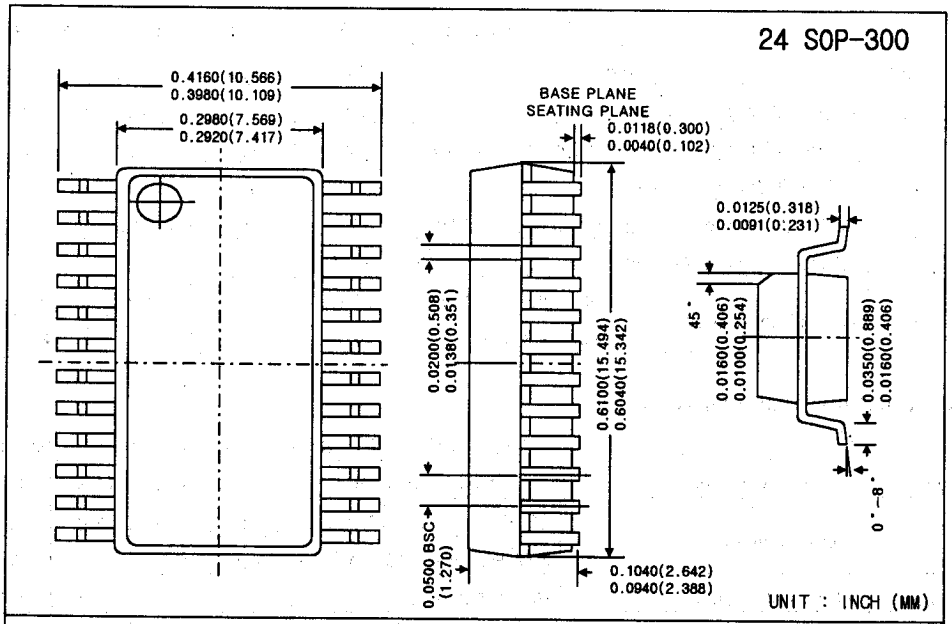
Symbols	Parameters		Min.	Typ.	Max.	Units
V_{DD}	Supply Voltage		1.8		3.6	V
V_{IH1}	"H" input Voltage, all input pins except OSCIN		0.7V_{DD}	V_{DD}	V_{DD}	V
V_{IH2}	"H" input Voltage, OSCIN		V_{DD} - 0.3	V_{DD}	V_{DD}	V
V_{IL1}	"L" input Voltage, all input pins except OSCIN		0	0	0.3 V_{DD}	V
V_{IL2}	"L" input Voltage, OSCIN		0	0	0.3	V
F_{OSC}	Oscillating frequency	Non-divide option	250		1000	KHz
		Divide-8 option	2		6	MHz

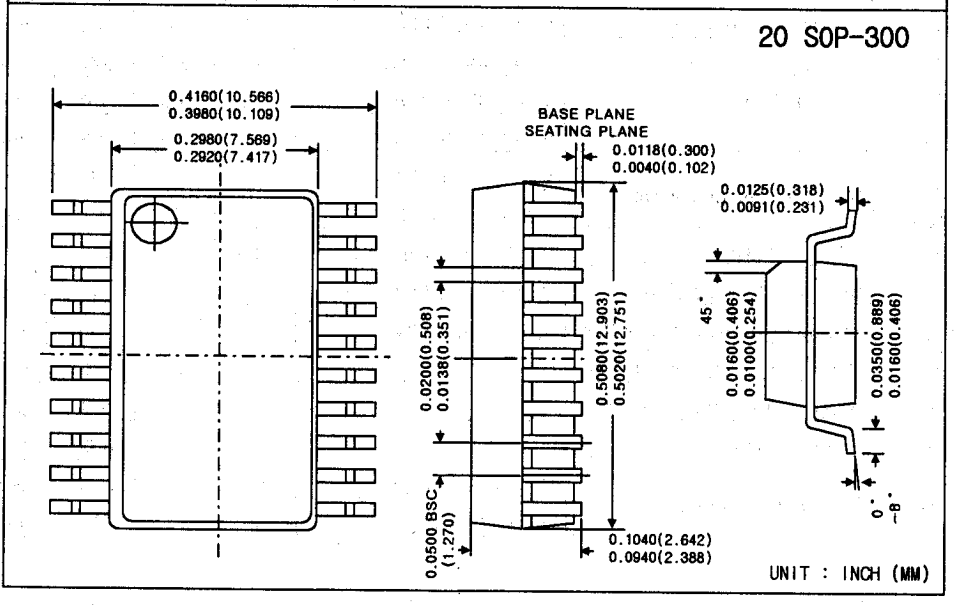
6.3 Electrical characteristics

(V_{DD} =3V ± 10%, Ta= 25℃, unless otherwise noted)

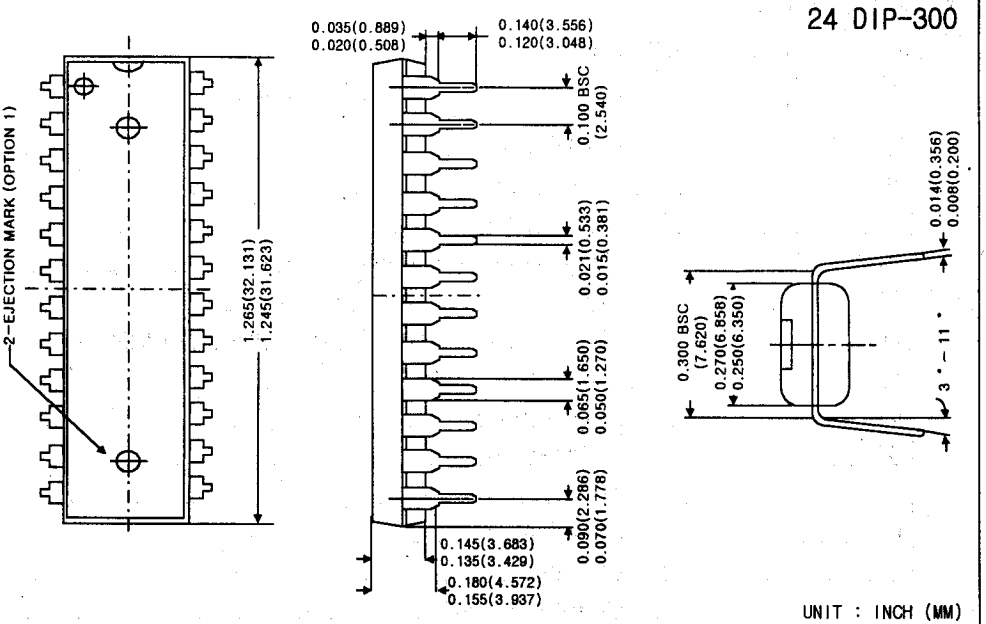
Symbols	Parameters	Test Conditions	Min.	Typ.	Max.	Units
V_{DD}	Supply Voltage	250KHz ≤ F_{OSC} ≤ 3.9MHz	1.8	3.0	3.6	V
		3.9MHz < F_{OSC} < 6.0MHz	2.2	3.0	3.6	V
I_{OH}	"H" output current	V_O = 2.0V, Port C	-6	-9	-14	mA
I_{OL0}	"L" output current	V_O = 0.4V, Port C	1.5	3	4.5	mA
I_{OL1}	"L" output current	V_O = 0.4V, Port C	180	210	240	mA
I_{OL2}		V_O = 0.4V, Port F	0.5	1.0	2.0	mA
I_{OL3}		V_O = 0.4V, Port G/K	1.5	3.0	4.5	mA
I_{LIH1}	"H" input leakage current	V_I = V_{DD}, Port D/E	—	—	3	μA
I_{LIH2}		V_I = V_{DD}, OSCIN	—	3	10	μA
I_{LIL}	"L" input leakage current	V_I = V_{SS}, OSCIN	-0.6	-3	-10	μA
I_{LOH}	"H" output leakage current	V_O = V_{DD}, Port C/F/G/K	—	—	1	μA
$R_{PULL-UP}$	Pull-up resistance of input Port	V_I = 0V, V_{DD} = 3V	30	70	150	KΩ
I_{DD}	Supply current at normal mode			0.5	1.0	mA
I_{DDS}	Supply current at stop mode				1.0	μA
F_{SYS}	Clock frequency		250		1000	KHz
F_{OSC}	Oscillator frequency	Non-divide option	250		1000	KHz
		Divide-8 option	2		6	MHz

7. PACKAGING OUTLINES and DIMENSIONS



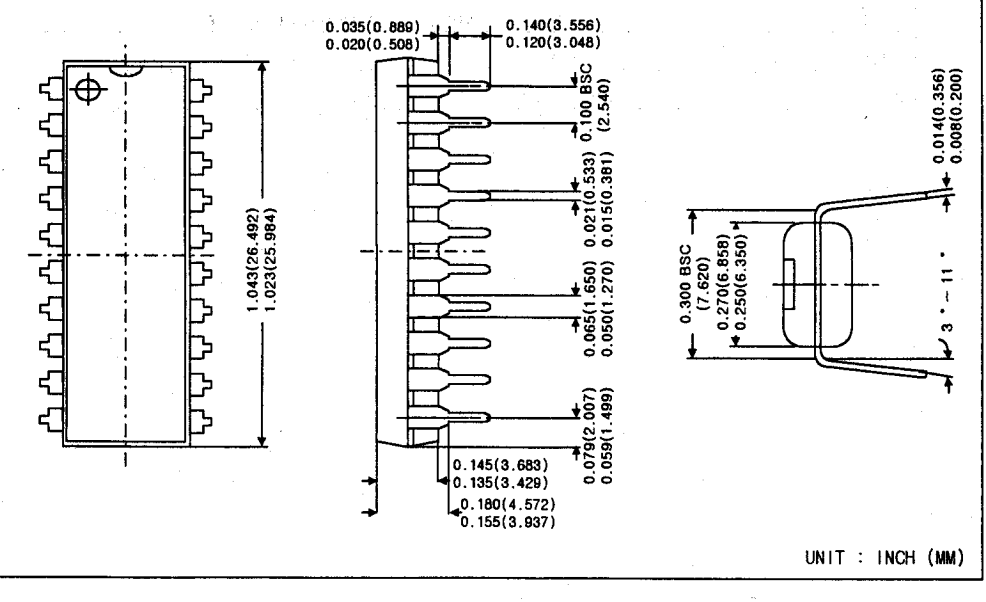


24 DIP-300



UNIT : INCH (MM)

20 DIP-300



UNIT : INCH (MM)

8. INSTRUCTIONS

8.1 SYMBOL DESCRIPTION

SYMBOL	DESCRIPTIONS
A , B , L	4 Bit Register
H	1-Bit Register
Z	3-Bit Register
PCH	The Higher 4-Bit of the Program Counter
PCL	The Lower 6-Bit of the Program Counter
PC	10-Bit Program Counter (Consisting of the PCH and PCL)
SK	10-Bit Stack Register
CY	1-Bit Carry Flag
SF	1-Bit Skip Flag
C, G, K	1-Bit Port
D, E	4-Bit Port
F	8-Bit Port
←	Direction of Data Flow
M[(HL)] or @HL	The Contents of Data Memory Addressed by Reg HL
M[(HL)].b or @HL.b	The Specified Bit's Content of Data Memory Addressed by Reg HL
@HL+	As a result of execution, increment L by one
addr	Address
n	immediate data

8.2 OPCODE MAP

LSB \ MSB		0000b	0001b	0010b	0011b	0100b	0101b	0110b	0111b	1000b~1011b	1100b~1111b
		0h	1h	2h	3h	4h	5h	6h	7h	8h~Bh	Ch~Fh
0000b	0h	NOP	ADDC @HL	XCH @HL+	LDZ n	LDL n	CALL addr	ADD n	LDA n	JMP addr	CAL addr
0001b	1h	STOP	LDA H	XCH @HL							
0010b	2h		LDA E	INC L							
0011b	3h	STA H	RRC	LDA @HL							
0100b	4h	IF0 @HL.b	LDA D	CLRB H			JMPL addr				
0101b	5h		LDA B	SETB H							
0110b	6h		LDA L								
0111b	7h		NOT								
1000b	8h	CLRB CY		STA @HL+			CLRB @HL.b				
1001b	9h	SETB CY		STA @HL							
1010b	Ah	CLRB F									
1011b	Bh	SETB F									
1100b	Ch	STA C	IF0 CY	CLRB G			SETB @HL.b				
1101b	Dh		RET	SETB G							
1110b	Eh	IFEQU n	STA B	CLRB K							
1111b	Fh	IFEQU @HL	STA L	SETB K							

8.3 INSTRUCTION DESCRIPTIONS

■ ADD n

Binary code : 0110xxxx
Syntax : [<label>] ADD n
Operation : (A) ← (A) + n, n=0~15 (n must be decimal number)
Flags : CY: Unaffected.
SF: Set to one if carry occurs, cleared otherwise.
Words/Cycles: 1/1
Description : Adds an immediate data to the accumulator and stores the result in the accumulator.
Example :

```
ADD   8    ;  Add 8 to A.
JMP   035  ;  Jump to 035 if 0≤A≤7
JMP   05F  ;  Jump to 05F if 8≤A≤15
```

■ ADDC @HL

Binary code : 00010000
Syntax : [<label>] ADDC @HL
Operation : (A) ← (A) + M[(HL)] + (CY), (CY) ← Carry
Flags : CY: Set on carry-out of (A) + M[(HL)] + (CY)
SF: Unaffected
Words/Cycles: 1/1
Description : Adds the contents of the accumulator, the contents of data memory addressed by registers H and L, and the carry bit. It stores the result in the accumulator and the carry flag.
Example :

```
CLRB  CY   ;  Clear CY to zero
LDA   5    ;  Load 5 to A
CLRB  H    ;  Clear H to zero
LDL   6    ;  Load 6 to L
ADDC  @HL  ;  Add the content of A, M[(06)], and the content of CY
```

■ CAL addr

Binary code : 11xxxxxx
Syntax : [<label>] CAL addr
Operation : (SK1) ← (SK0), (SK0) ← (PC) + 1, (PCL) ← addr, addr = 000 ~ 03F
(addr must be hexadecimal number)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Calls a subroutine located at the indicated address and pushes the current contents of the program counter to the top of stack. The indicated address must be within the current page.
Example :

```
CAL   100  ;  Call subroutine located at the 100. The 100 must be logical address
              and within the current page.
```

■ **CALL addr**

Binary code : 010100xx xxxxxxxx
Syntax : [<label>] CALL addr
Operation : (SK1) ← (SK0), (SK0) ← (PC) + 1, (PC) ← addr,
addr = 000 ~ 3FF (addr must be hexadecimal number)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles : 2/2
Description : Calls a subroutine located at the indicated address and pushes the current contents of the program counter to the top of stack. The indicated address can be anywhere in the full 1K-byte memory space.
Example :
```
CALL    2FF     ;  Call subroutine located at the 2FF. The 2FF must be logical address.
```

■ **CLRB @HL.b**

Binary code : 010110xx
Syntax : [<label>] CLRB @HL.b
Operation : M[(HL)].b ← 0
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles : 1/1
Description : Clears the specified bit of data memory addressed by registers H and L to zero.
Example :
```
CLRB    H       ;  Clear H to 0
LDL     10      ;  Load 10 to L. The 10 must be decimal number.
CLRB    @HL.0   ;  Clear the bit 0 of M[(0A)] to 0.
```

■ **CLRB CY**

Binary code : 00001000
Syntax : [<label>] CLRB CY
Operation : (CY) ← 0
Flags : CY: Set to zero
SF: Unaffected
Words/Cycles : 1/1
Description : Clears the carry flag to zero.
Example :
```
CLRB    CY      ;  Clear CY to zero
```

■ **CLRB F**

Binary code : 00001010
Syntax : [<label>] CLRB F
Operation : F.(L) ← 0
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Clears the specified bit of port F addressed by the lower 3-bit of register L to zero.
Example :
```
LDL   13   ; Load 13 to L
CLRB  F    ; Clears the bit 5 of F to zero
```

■ **CLRB G**

Binary code : 00101100
Syntax : [<label>] CLRB G
Operation : (G) ← 0
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Clears the port G to zero.
Example :
```
CLRB  G    ; Clear G to zero
```

■ **CLRB H**

Binary code : 00100100
Syntax : [<label>] CLRB H
Operation : (H) ← 0
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Clears the contents of register H to zero. Skip this instruction if it or SETB H was used just before.
Example :
```
IFEQU 1
CLRB  H    ; Clear H to zero and skip continuous SETB H/CLRB H, if (A)≠1
SETB  H    ; Sets H to one and skip continuous SETB H/CLRB H, if (A)=1
```

■ **CLRB K**

Binary code : 00101110
Syntax : [<label>] CLRB K
Operation : (K) ← 0
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles : 1/1
Description : Clears the port K to zero.
Example :
```
CLRB    K       ;  Clear K to zero.
```

■ **IF0 @HL.b**

Binary code : 000001xx
Syntax : [<label>] IF0 @HL.b
Operation : M[(HL)b] = 0
Flags : CY: Unaffected
SF: Set to one if equal, cleared otherwise
Words/Cycles : 1/1
Description : Compares the specified bit of data memory addressed by registers H and L with zero.
Example :
```
SETB    H       ;  Set H to one
LDL     4       ;  Load 4 to L
IF0     @HL.3   ;  Compare the bit 3 of M[(14)] with zero
JMP     020     ;  Jump to 020 if not equal
JMP     030     ;  Jump to 030 if equal
```

■ **IF0 CY**

Binary code : 00011100
Syntax : [<label>] IF0 CY
Operation : (CY) = 0
Flags : CY: Unaffected
SF: Set to one if equal, cleared otherwise
Words/Cycles : 1/1
Description : Compares the carry flag with zero.
Example :
```
IF0     CY      ;  Compare the content of CY to zero
JMP     030     ;  Jump to 030 if not equal
JMP     040     ;  Jump to 040 if equal
```

■ **IFEQU @HL**

Binary code : 00001111
Syntax : [<label>] IFEQU @HL
Operation : (A) = M[(HL)]
Flags : CY: Unaffected
SF: Set to one if equal, cleared otherwise
Words/Cycles: 1/1
Description : Compares the contents of accumulator with the contents of data memory addressed by registers H and L.
Example :

```
LDA   14   ;  Load 14 to A, and 14 must be decimal number
SETB  H    ;  Sets H to one
LDL   4    ;  Loads 4 to L
IFEQU @HL  ;  Compares 14 with M[(14)]
JMP   050  ;  Jump to 050 if not equal
JMP   060  ;  Jump to 060 if equal
```

■ **IFEQU n**

Binary code : 00001110 0111xxxx
Syntax : [<label>] IFEQU n
Operation : (A) = n, n = 0 ~15 (n must be decimal number)
Flags : CY: Unaffected
SF: Set to one if equal, cleared otherwise
Words/Cycles: 2/2
Description : Compares the contents of accumulator with an immediate data.
Example :

```
IFEQU 15   ;  Compare the contents of accumulator with 15
JMP   070  ;  Jump to 070 if not equal
JMP   080  ;  Jump to 080 if equal
```

■ **INC L**

Binary code : 00100010
Syntax : [<label>] INC L
Operation : (L) ← (L) + 1
Flags : CY: Unaffected
SF: As a result of execution, set to one if the contents of register L are zero, cleared otherwise.
Words/Cycles: 1/1
Description : The contents of register L are incremented by one.
Example :

```
LDL   14   ;  Load 14 to L
INC   L    ;  The contents of L are incremented by one
INC   L    ;  The contents of L are incremented by one
JMP   090  ;  It is skipped because the contents of L is "0"
JMP   0A0  ;  Jump to 0A0
```

■ **JMP addr**

Binary code : 10xxxxxx
Syntax : [<label>] JMP addr
Operation : (PCL) ← addr, addr = 00 ~ 3F (addr must be hexadecimal number)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Jumps unconditionally to the indicated address. The indicated address must be within the current page.
Example : JMP 2EF ; Jump unconditionally to the 2EF. The 2EF address must be within the current page.

■ **JMPL addr**

Binary code : 010101xx xxxxxxxx
Syntax : [<label>] JMPL addr
Operation : (PC) ← addr, addr = 000 ~ 3FF (addr must be hexadecimal number.)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 2/2
Description : Jumps unconditionally to the indicated address. The indicated address can be anywhere in the full 1K-byte memory space.
Example : JMPL 100 ; Jump unconditionally to 100

■ **LDA @HL**

Binary code : 00100011
Syntax : [<label>] LDA @HL
Operation : (A) ← M[(HL)]
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Loads the contents of memory addressed by registers H and L into the accumulator.
Example : SETB H ; Set H to 1
LDL 0 ; Load 0 to L
LDA @HL ; Load M[(10)] into A

■ **LDA n**

Binary code : 0111xxxx
Syntax : [<label>] LDA n
Operation : (A) ← n, n=0~15 (n must be decimal number.)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles : 1/1
Description : Loads an immediate data into the accumulator. Skip this instruction if it was used just before.
Example :

```
STA   B
LDA   15    ;  Load 15 into A.
LDA   4     ;  It is skipped because this instruction was used just before
LDA   7     ;  It is skipped because this instruction was used just before
JMP   0B0   ;  Jump to 0B0
```

■ **LDA B**

Binary code : 00010101
Syntax : [<label>] LDA B
Operation : (A) ← (B)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles : 1/1
Description : Loads the contents of register B into the accumulator.
Example :

```
LDA   B     ;  Load the contents of B into A
```

■ **LDA D**

Binary code : 00010100
Syntax : [<label>] LDA D
Operation : (A) ← (D)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles : 1/1
Description : Loads the contents of port D into the accumulator.
Example :

```
LDA   D     ;  Load the contents of D into A
```

■ **LDA E**

Binary code : 00010010
Syntax : [<label>] LDA E
Operation : (A) ← (E)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Loads the contents of port E into the accumulator.
Example : LDA E ; Load the contents of E into A

■ **LDA H**

Binary code : 00010001
Syntax : [<label>] LDA H
Operation : (A) ← (H)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Loads the contents of register H into the bit 0 of accumulator.
Example : LDA H ; Load the content of H into the bit 0 of A

■ **LDA L**

Binary code : 00010110
Syntax : [<label>] LDA L
Operation : (A) ← (L)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Loads the contents of register L into the accumulator.
Example : LDA L ; Load the contents of L into A

■ **LDL n**

Binary code : 0100xxxx
Syntax : [<label>] LDL n
Operation : (A) ← n, n = 0 ~ 15 (n must be decimal number)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles : 1/1
Description : Loads an immediate data to the register L. Skip this instruction if it was used just before.
Example :

```
LDA   3
LDL   8     ;  Load 8 to L
LDL   4     ;  It is skipped because this instruction was used just before
JMP   0C0   ;  Jump to 0C0
```

■ **LDZ n**

Binary code : 00110xxx
Syntax : [<label>] LDZ n
Operation : (A) ← n, n = 0 ~ 7 (n must be decimal number)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles : 1/1
Description : Load an immediate data into the register Z.
Example :

```
LDZ   0     ;  Load 0 into Z. The 0 must be decimal number
```

■ **NOP**

Binary code : 00000000
Syntax : [<label>] NOP
Operation : (PC) ← (PC) + 1
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles : 1/1
Description : No operation.
Example :

```
NOP         ;  No operation
```

■ NOT

Binary code : 00010111
Syntax : [<label>] NOT
Operation : (A) ← /(A)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles : 1/1
Description : The contents of accumulator are 1's complemented.
Example :
```
LDA     7
NOT             ;  1's complement 7, then leaves 8 in A
```

■ RET

Binary code : 00011101
Syntax : [<label>] RET
Operation : (PC) ← (SK0), (SK0) ← (SK1)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles : 1/1
Description : Returns from the subroutine to main routine.
Example :
```
RET             ;  Returns from the subroutine to main routine
```

■ RRC

Binary code : 00010011
Syntax : [<label>] RRC
Operation : (A.b) ← (A.b+1) (A.3) ← (CY) (CY) ← (A.0)
Flags : CY: Set to bit 0 of the accumulator
SF: Unaffected
Words/Cycles : 1/1
Description : Shifts the contents of accumulator 1-bit to the right through the carry. The carry bit content shifts into the bit 3 of accumulator, and the bit 0 of accumulator is shifted into the carry bit.
Example :
```
SETB    CY      ;  Set CY to one.
LDA     5       ;  Load 5 to A
RRC             ;  CY becomes zero, and the contents of A is 11
```

■ **SETB @HL.b**

Binary code : 010111xx
Syntax : [<label>] SETB @HL.b
Operation : M[(HL)].b ← 1
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Sets the specified bit of memory addressed by registers H and L to one.
Example :
```
CLRB   H      ; Clear H to zero
LDL    5      ; Load 5 to L
SETB   @HL.2  ; Set the bit 2 of M[(05)] to one
```

■ **SETB CY**

Binary code : 00001001
Syntax : [<label>] SETB CY
Operation : (CY) ← 1
Flags : CY: Set to one
SF: Unaffected
Words/Cycles: 1/1
Description : Sets the contents of carry flag to one.
Example :
```
SETB   CY     ; Sets the content of CY to one
```

■ **SETB F**

Binary code : 00001011
Syntax : [<label>] SETB F
Operation : F.(L) ← 1
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Sets the specified bit of the port F addressed by register L to one.
Example :
```
LDL    4      ; Loads 4 to L
SETB   F      ; Sets the bit 4 of F to one
```

■ **SETB G**

Binary code : 00101101
Syntax : [<label>] SETB G
Operation : (G) ← 1
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Sets the port G to one.
Example : SETB G ; Sets the port G to one

■ **SETB H**

Binary code : 00100101
Syntax : [<label>] SETB H
Operation : (H) ← 1
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Sets the contents of register H to one. Skip this instruction if it or SETB H was used just before.
Example : IFEQU 1
SETB H ; Sets H to one and skip continuous CLRB H/SETB H, if (A)≠1
CLRB H ; Clear H to zero and skip continuous CLRB H/SETB H, if (A)=1

■ **SETB K**

Binary code : 00101111
Syntax : [<label>] SETB K
Operation : (K) ← 1
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Sets the port K to one.
Example : SETB K ; Sets the port K to one

■ **STA @HL**

Binary code : 00101001
Syntax : [<label>] STA @HL
Operation : M[(HL)] ← (A)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Stores the contents of accumulator in memory addressed by registers H and L.
Example :
```
LDL   0    ; Load 0 to L
SETB  H    ; Set H to one
STA   @HL  ; Stores the contents of A in M[(10)]
```

■ **STA @HL+**

Binary code : 00101000
Syntax : [<label>] STA @HL+
Operation : M[(HL)] ← (A), (L) ← (L) + 1
Flags : CY: Unaffected
SF: As a result of execution, set to one if the contents of register L are zero, cleared otherwise
Words/Cycles: 1/1
Description : Stores the contents of accumulator in memory addressed by registers H and L. And then the contents of register L are incremented by one.
Example :
```
LDL   15   ; Load 15 to L
SETB  H    ; Set H to one
STA   @HL+ ; Stores the contents of A in M[(1F)]. L becomes "0"
JMP   035  ; It is skipped because L is "0"
JMP   045  ; Jump to 045
```

■ **STA B**

Binary code : 00011110
Syntax : [<label>] STA B
Operation : (B) ← (A)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Stores the contents of accumulator in the register B.
Example :
```
STA   B    ; Stores the contents of A in B
```

■ **STA C**

Binary code : 00001100
Syntax : [<label>] STA C
Operation : (C) ← $(A)_3$
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Stores the bit 3 of accumulator in the port C.
Example : STA C ; Stores the bit 3 of A in C

■ **STA H**

Binary code : 00000011
Syntax : [<label>] STA H
Operation : (H) ← $(A)_0$
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Stores the bit 0 of accumulator in the register H.
Example : STA H ; Store the bit 0 of A in H

■ **STA L**

Binary code : 00011111
Syntax : [<label>] STA L
Operation : (L) ← (A)
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Stores the contents of accumulator in the register L.
Example : STA L ; Stores the contents of A in L

■ **STOP**

Binary code : 00000001
Syntax : [<label>] STOP
Operation : Stop the oscillation of the oscillator, and reset PORT F to zero
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Stops the oscillation of the oscillator.
Example : STOP

■ **XCH @HL**

Binary code : 00100001
Syntax : [<label>] XCH @HL
Operation : (A) ↔ M[(H,L)]
Flags : CY: Unaffected
SF: Unaffected
Words/Cycles: 1/1
Description : Exchanges the accumulator with the contents of the data memory addressed by registers H and L without going through an intermediate location.
Example :

```
LDL   3    ; Load 3 to L
SETB  H    ; Set H to one
XCH   @HL  ; Exchanges the contents of A with M[(13)] without going
             through an intermediate location
```

■ **XCH @HL+**

Binary code : 00100000
Syntax : [<label>] XCH @HL+
Operation : (A) ↔ M[(H,L)], (L) ← (L) + 1
Flags : CY: Unaffected
SF: As a result of execution, set to one if the contents of register L are zero, cleared otherwise
Words/Cycles: 1/1
Description : Exchanges the accumulator with the contents of the data memory addressed by registers H and L without going through an intermediate location. As a result of execution, the contents of register L are incremented by one.
Example :

```
SETB  H    ; Set H to one
LDL   15   ; Load 15 into L
XCH   @HL+ ; Exchanges A with M[(1F)] without going through an
             intermediate location. As a result of execution, the contents
             of L are "0"
JMP   055  ; It is skipped because L is "0"
JMP   065  ; Jump to 065
```