GMS87C1102 / GMS87C1202

CMOS SINGLE-CHIP 8-BIT MICROCONTROLLER

1. OVERVIEW

1.1 Description

The GMS87C1102 and GMS87C1202 are an advanced CMOS 8-bit microcontroller with 2K bytes of ROM. The HYUNDAI MicroElectronics GMS87C1102 and GMS87C1202 are a powerful microcontroller which provides a highly flexible and cost effective solution to many small applications. The GMS87C1102 and GMS87C1202 provide the following standard features: 2K bytes of ROM(OTP), 128 bytes of RAM, 8-bit timer/counter, 8-bit A/D converter, 10-bit High Speed PWM Output, Programmable Buzzer Driving Port (GMS87C1202 only), on-chip oscillator and clock circuitry. In addition, the GMS87C1102 and GMS87C1202 support power saving modes to reduce power consumption.

This document is only explained for the base of GMS87C1202, the eliminated functions are same as below.

Device name	OTP Size	RAM Size	I/O	BUZ	INT1	Package	Operating Temperature
GMS87C1102	2K bytes	128 bytes	11	NO	NO	16DIP/SOP	-20°C~+85°C
GMS87C1202	2K bytes	128 bytes	15	YES	YES	20DIP/SOP	-20°C~+85°C
GMS87C1102E	2K bytes	128 bytes	11	NO	NO	16DIP/SOP	-40°C~+125°C
GMS87C1202E	2K bytes	128 bytes	15	YES	YES	20DIP/SOP	-40°C~+125°C

1.2 Features

- **2K bytes On-chip Program Memory**
- **128 Bytes of On-Chip Data RAM**
- **Minimum Instruction execution time:**
  - 500ns at 8MHz (2cycle NOP Instruction)
- **2.7V to 6.0V Wide Operating Range**
- **Basic Interval Timer**
- **Two 8-Bit Timer/ Counters**
- **10-Bit High Speed PWM Output**
- **Two external interrupt ports**
  (GMS87C1102 has one external interrupt port)
- **One Programmable Buzzer Driving port**
  (GMS87C1202 only)
- **15 Programmable I/O Lines**
  (GMS87C1102 has 11 programmable I/O lines)
- **Seven Interrupt Sources**
  (GMS87C1102 has Six interrupt sources)
- **8-Channel 8-Bit On-Chip Analog to Digital Converter**
- **Watch dog timer**
- **Oscillation :**
  - Crystal
  - Ceramic Resonator
  - External Oscillator
  - RC Oscillation
- **Power Down Mode**
  - STOP mode
  - Wake-up Timer mode
  - RC-WDT mode
- **Power Fail Processor**
  (Noise Immunity Circuit)

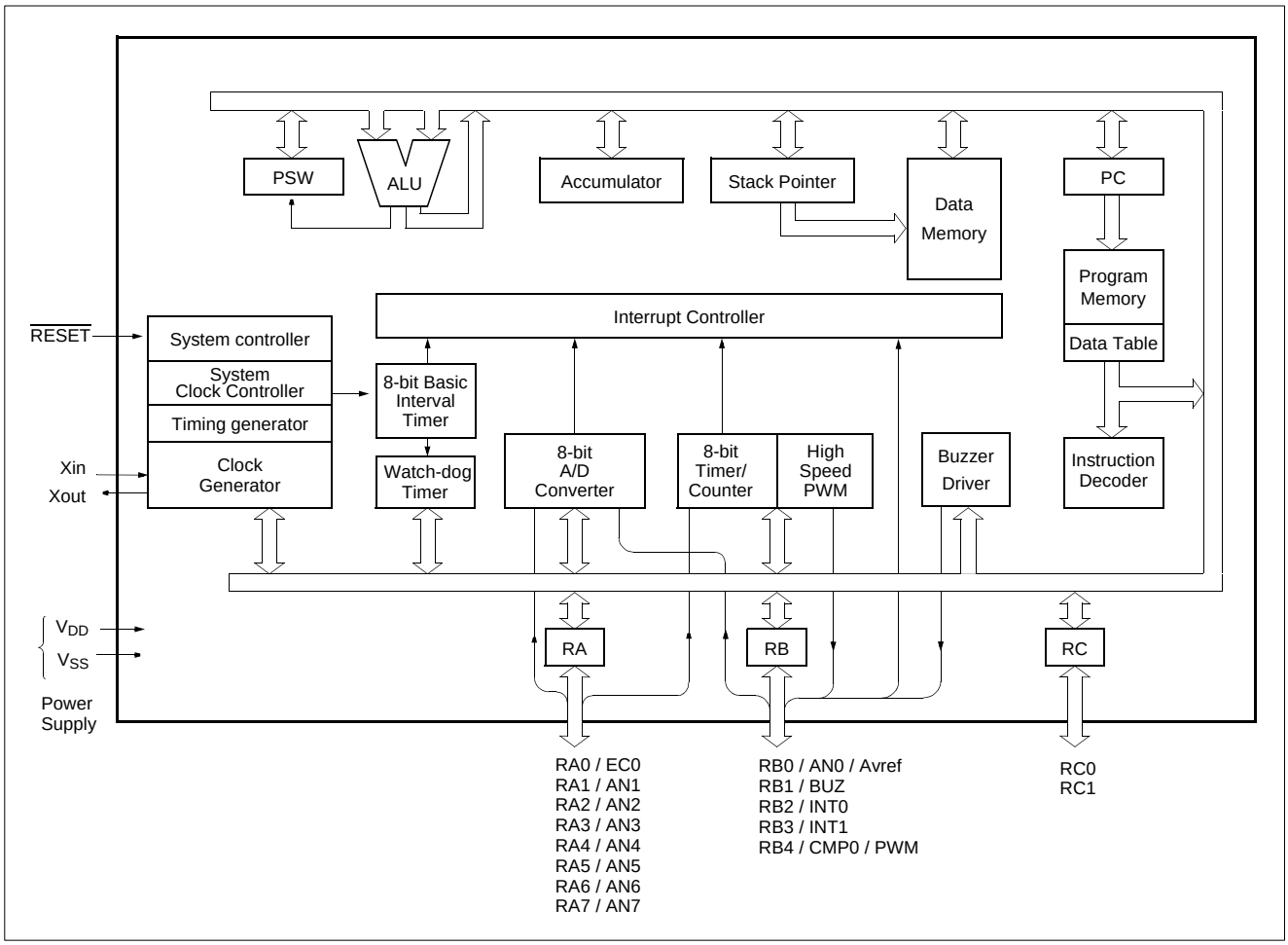
1.3 Development Tools

The GMS800 family is supported by a full-featured macro assembler, an in-circuit emulators CHOICE-Dr.™, and add-on board type OTP writer Dr.Writer™ .

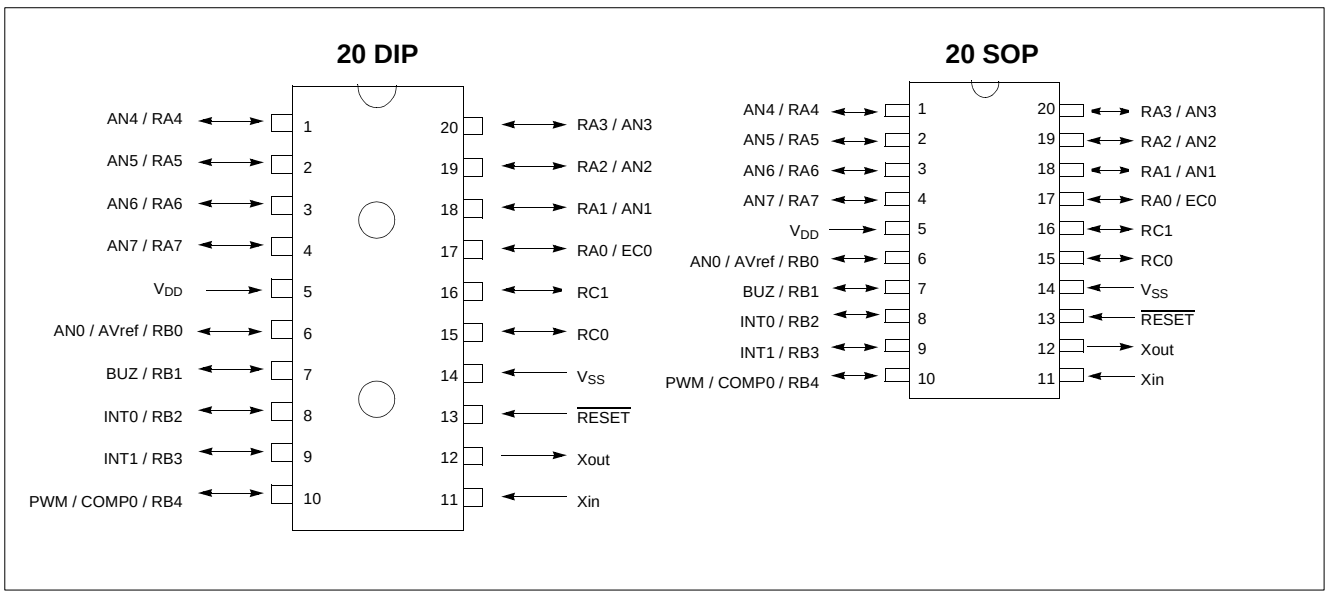
The availability of OTP devices is especially useful for customers expecting frequent code changes and updates. The OTP devices, packaged in plastic packages, permit the user to program them once.

In Circuit Emulator	CHOICE-Dr.
Assembler	HYUNDAI MicroElectronics Macro Assembler
OTP Writer	Dr.Writer

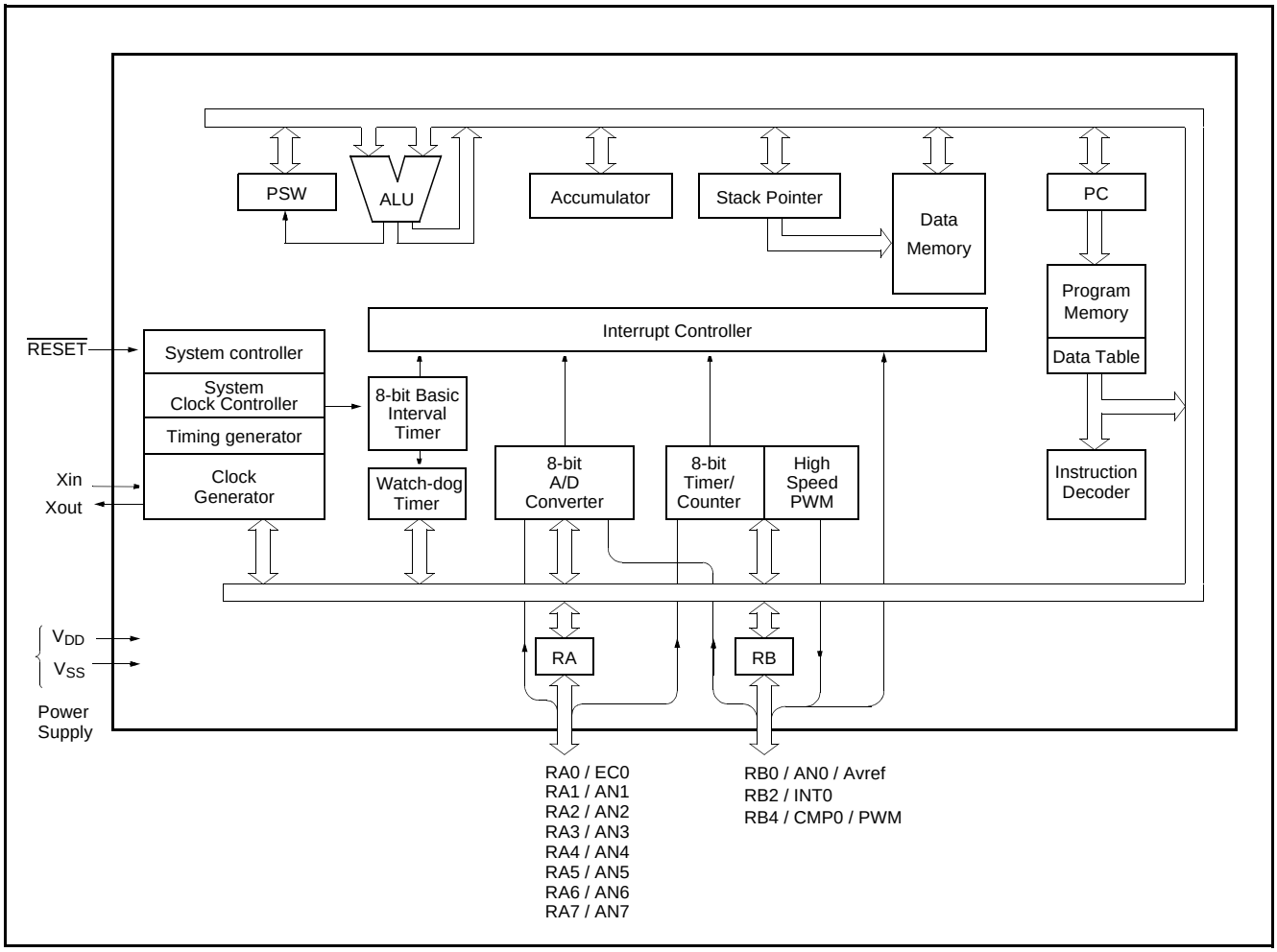
2. BLOCK DIAGRAM(GMS87C1202)



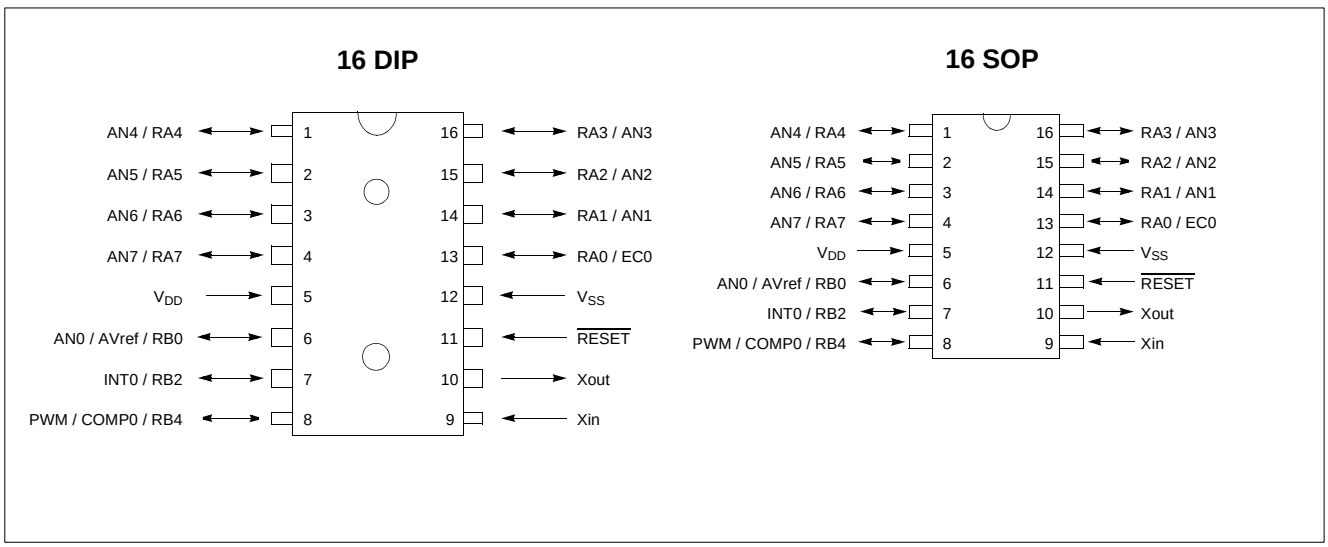
3. PIN ASSIGNMENT(GMS87C1202)



4. BLOCK DIAGRAM(GMS87C1102)



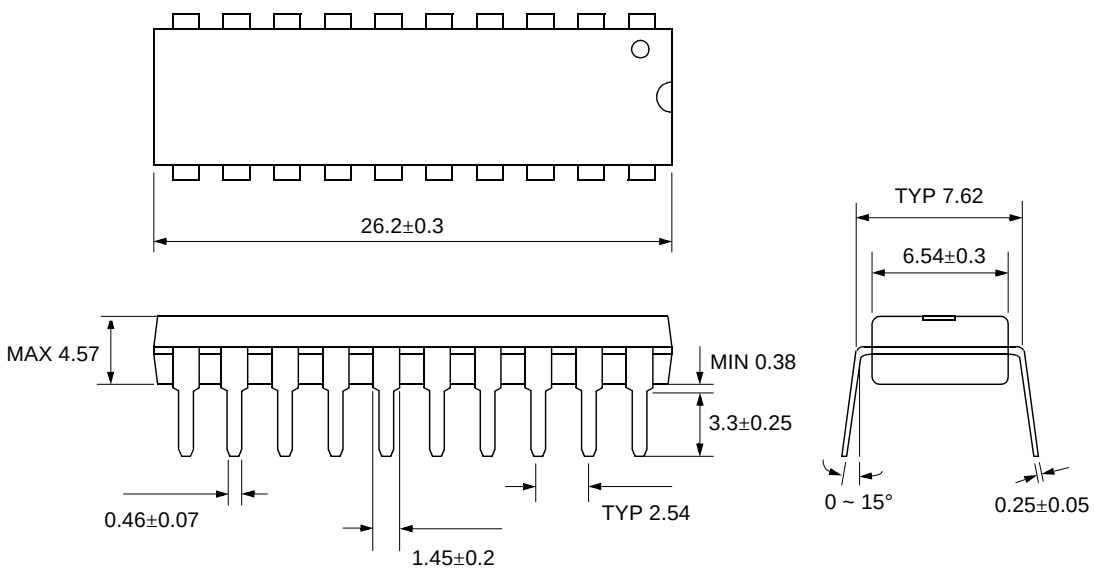
5. PIN ASSIGNMENT(GMS87C1102)



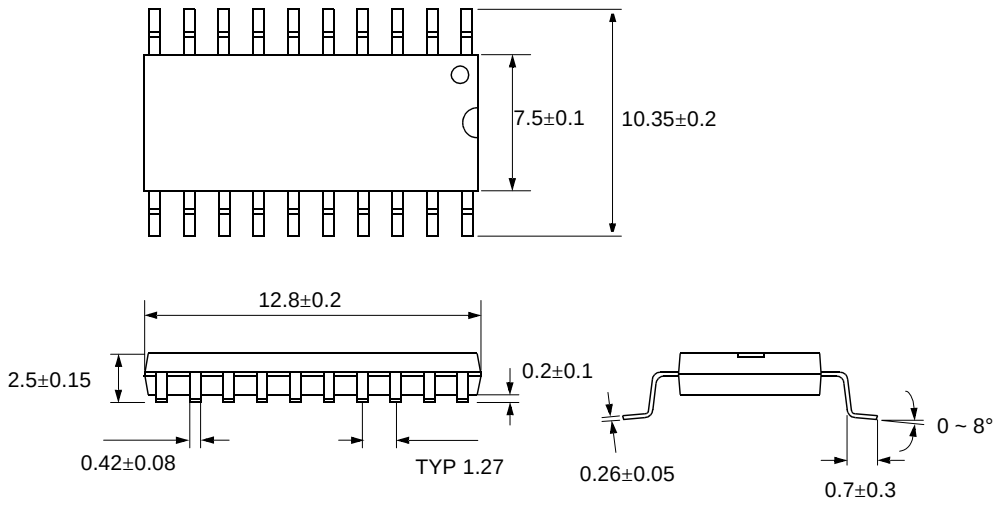
6. PACKAGE DIMENSION(GMS87C1202)

20 DIP

unit : mm



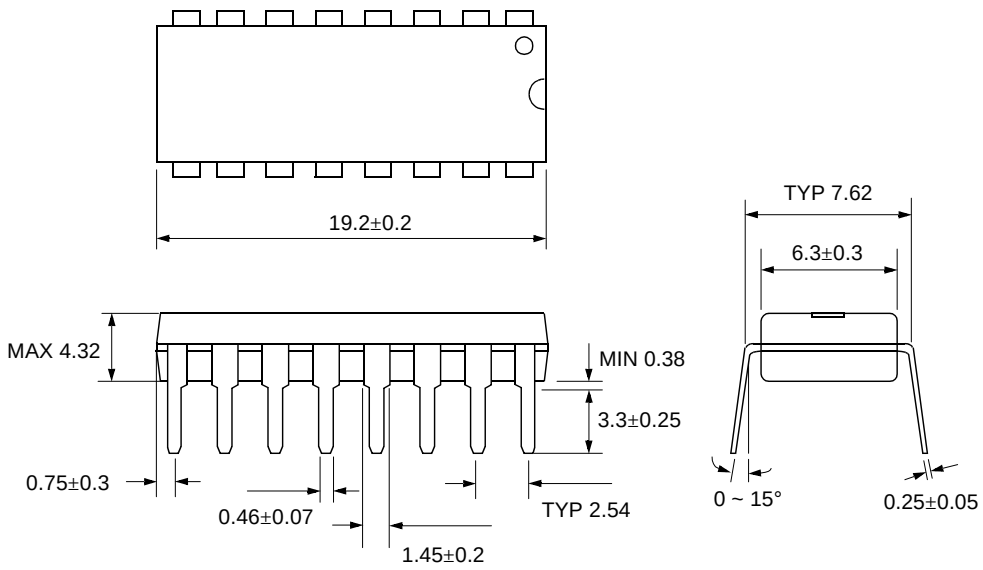
20 SOP



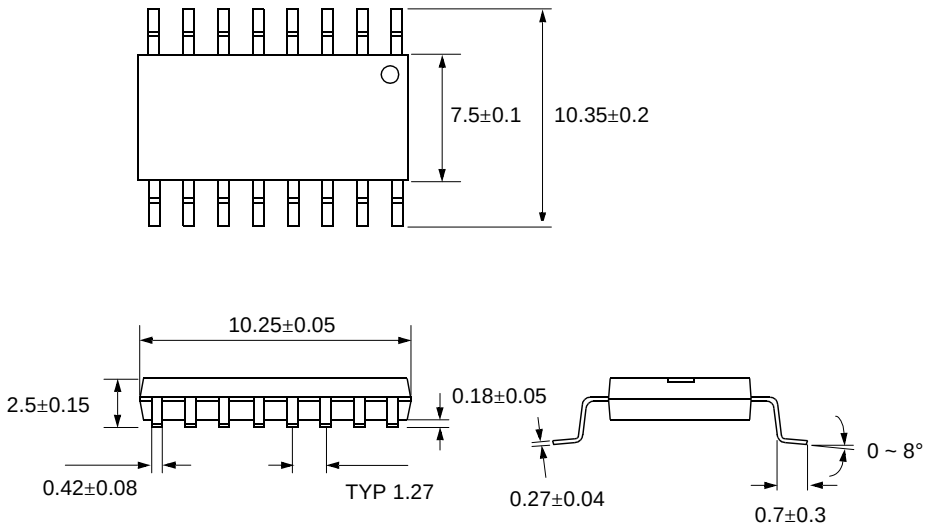
7. PACKAGE DIMENSION(GMS87C1102)

16 DIP

unit : mm



16 SOP



8. PIN FUNCTION

V_{DD}: Supply voltage.

V_{SS}: Circuit ground.

$\overline{\text{RESET}}$: Reset the MCU.

X_{IN}: Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

X_{OUT}: Output from the inverting oscillator amplifier. If RC Option is used, the oscillator frequency divided by 4 (Xin/4) comes out from Xout pin.

RA0~RA7: RA is an 8-bit, CMOS, bidirectional I/O port. RA pins can be used as outputs or inputs according to "1" or "0" written in the Port Direction Register(RAIO).

Port pin	Alternate function
RA0	EC0 (Event Counter Input Source)
RA1	AN1 (Analog Input Port 1)
RA2	AN2 (Analog Input Port 2)
RA3	AN3 (Analog Input Port 3)
RA4	AN4 (Analog Input Port 4)
RA5	AN5 (Analog Input Port 5)
RA6	AN6 (Analog Input Port 6)
RA7	AN7 (Analog Input Port 7)

Table 8-1 RA Port

In addition, RA serves the functions of the various special features in Table 8-1.

RB0~RB4: RB is a 5-bit, CMOS, bidirectional I/O port. RB pins can be used as outputs or inputs according to "1" or "0" written in the Port Direction Register(RBIO).

RB serves the functions of the various following special features.

Port pin	Alternate function
RB0	AN0 (Analog Input Port 0) AVref (External Analog Reference Pin)
RB1	BUZ (Buzzer Driving Output Port)
RB2	INT0 (External Interrupt Input Port 0)
RB3	INT1 (External Interrupt Input Port 1)
RB4	PWM (PWM Output) COMP0 (Timer0 Compare Output)

Table 8-2 RB Port

RC0~RC1: RC is a 2-bit, CMOS, bidirectional I/O port. RC pins can be used as outputs or inputs according to "1" or "0" written in the Port Direction Register(RCIO)

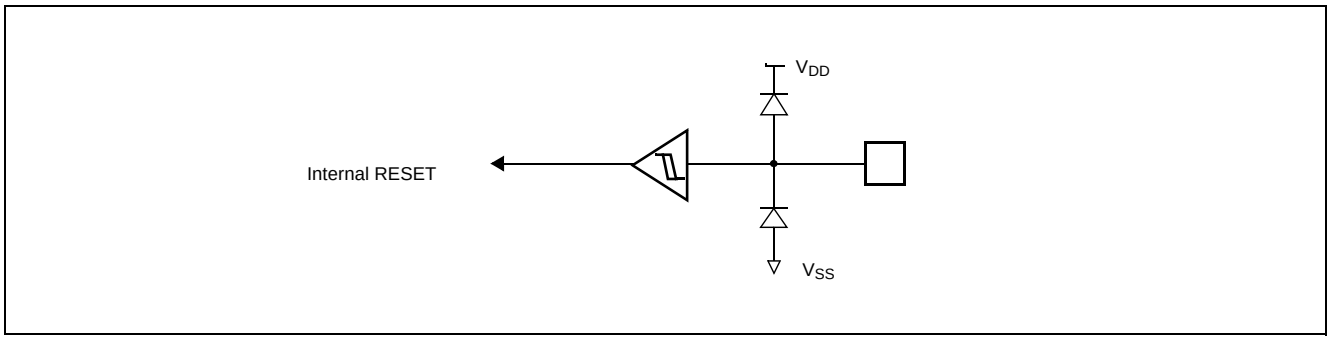
.

PIN NAME	Pin No.	In/Out	Function	
V_{DD}	5	-	Supply voltage	
V_{SS}	14	-	Circuit ground	
$\overline{\text{RESET}}$	13	I	Reset signal input	
X_{IN}	11	I		
X_{OUT}	12	O		
RA0 (EC0)	17	I/O (Input)	8-bit general I/O ports	External Event Counter input
RA1 (AN1)	18	I/O (Input)		Analog Input Port 1
RA2 (AN2)	19	I/O (Input)		Analog Input Port 2
RA3 (AN3)	20	I/O (Input)		Analog Input Port 3
RA4 (AN4)	1	I/O (Input)		Analog Input Port 4
RA5 (AN1)	2	I/O (Input)		Analog Input Port 5
RA6 (AN1)	3	I/O (Input)		Analog Input Port 6
RA7 (AN7)	4	I/O (Input)		Analog Input Port 7
RB0 (AVref/AN0)	6	I/O (Input)	5-bit general I/O ports	Analog Input Port 0 / Analog Reference
RB1 (INT0)	7	I/O (Input)		External Interrupt Input 0
RB2 (INT1)	8	I/O (Input)		External Interrupt Input 1
RB3 (BUZ)	9	I/O (Output)		Buzzer Driving Output
RB4 (PWM/COMP0)	10	I/O (Output/Output)		PWM Output or Timer Compare Output
RC0	15	I/O	2-bit general I/O ports	
RC1	16	I/O		

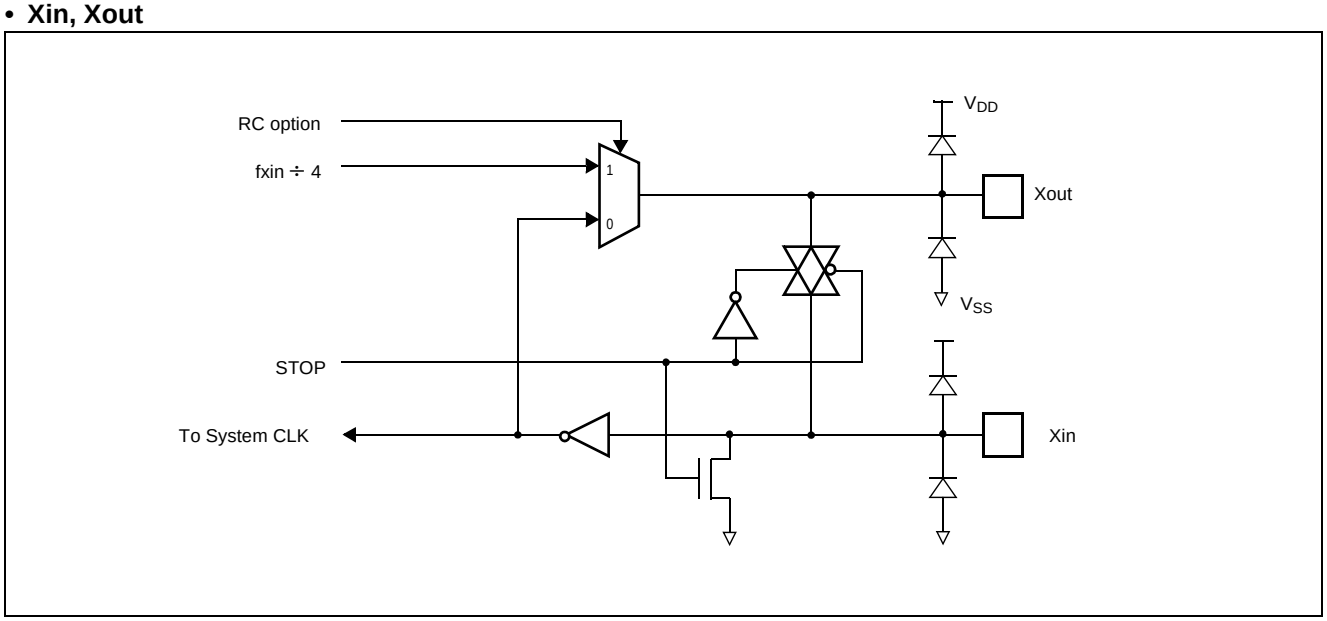
Table 8-3 Pin Description

9. PORT STRUCTURES

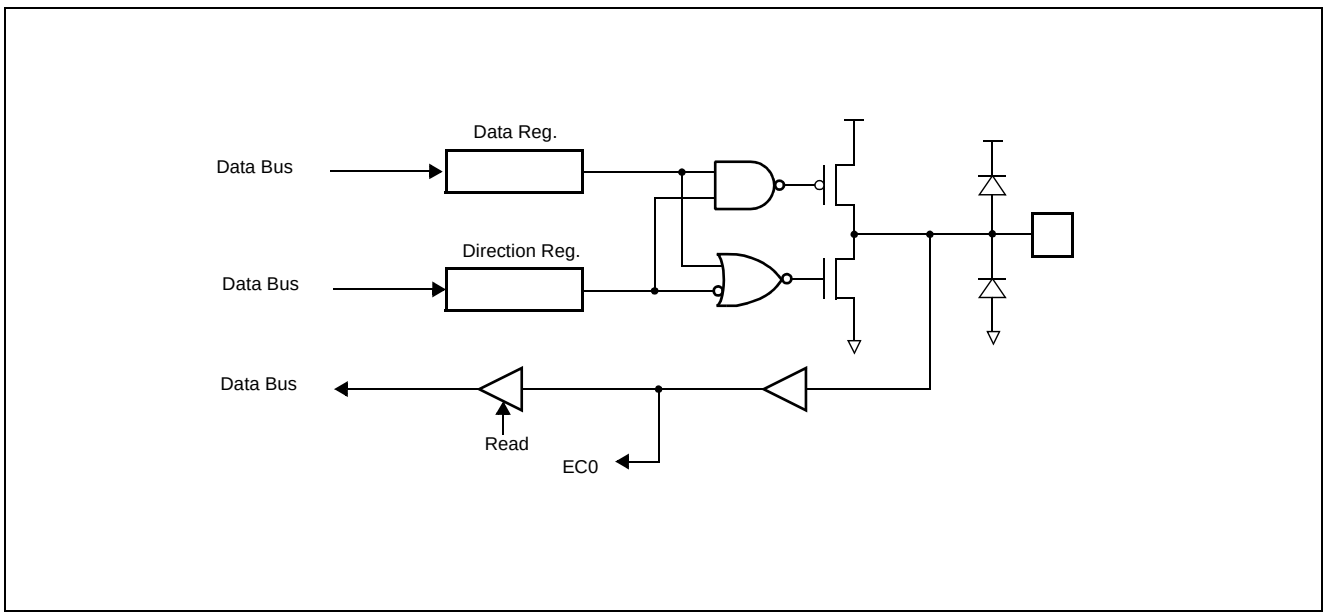
• **$\overline{\text{RESET}}$**



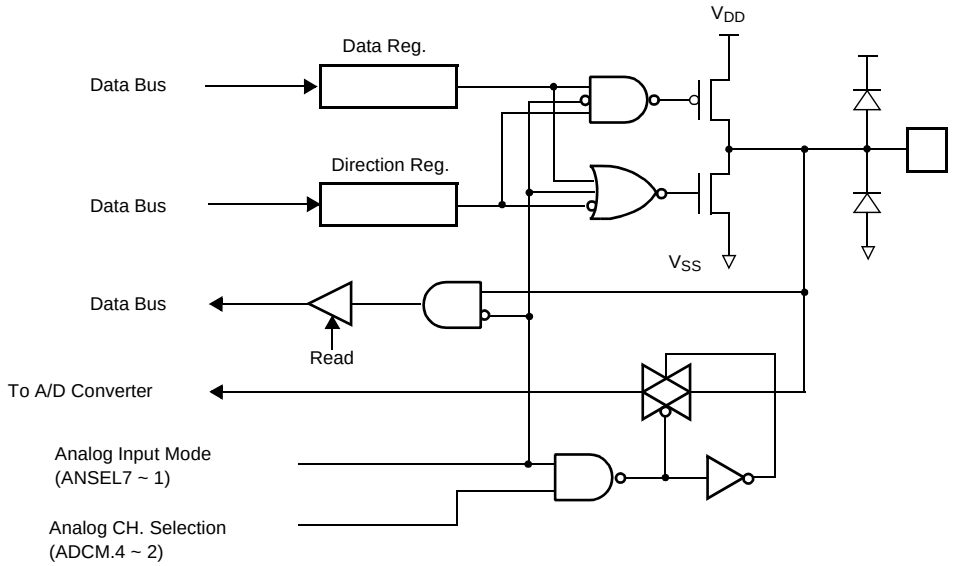
• **Xin, Xout**



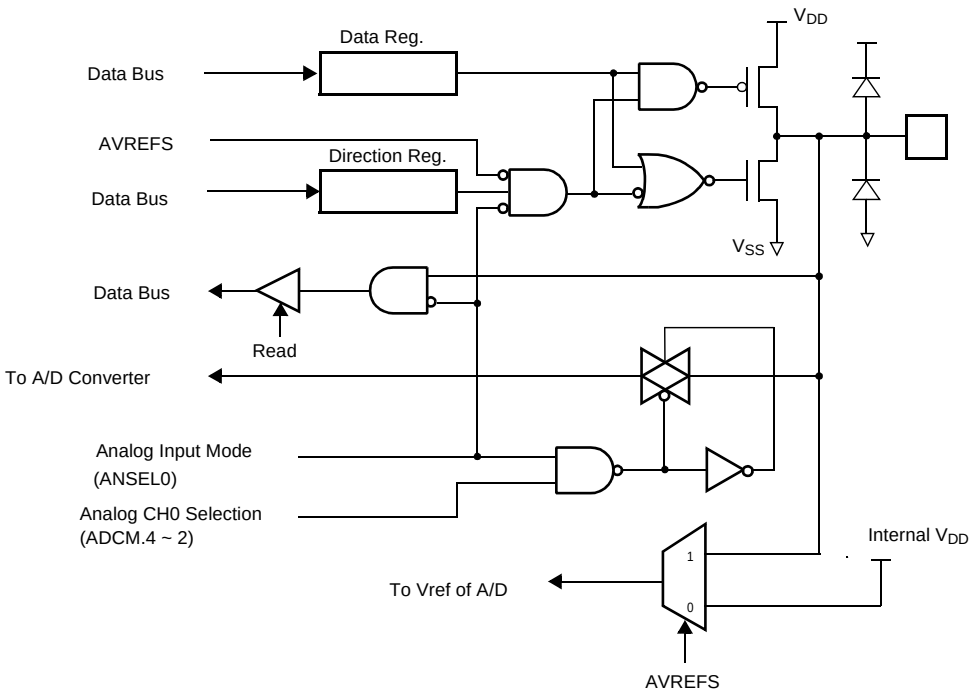
• **RA0/EC0**



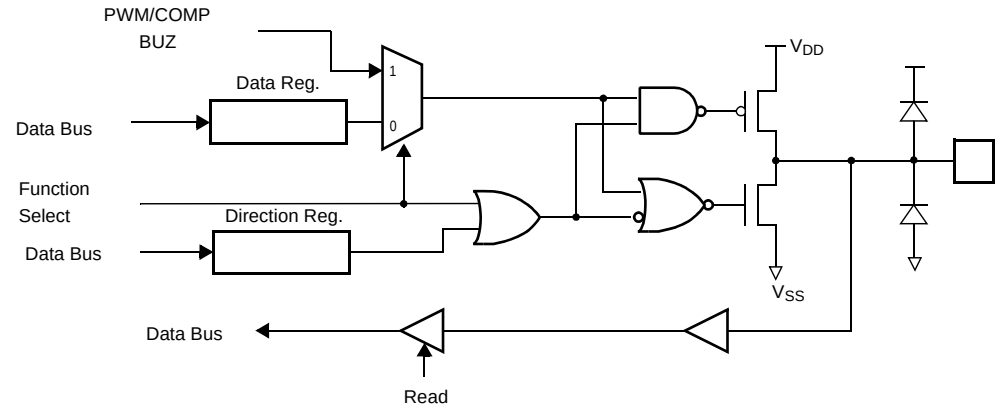
- **RA1/AN1 ~ RA7/AN7**



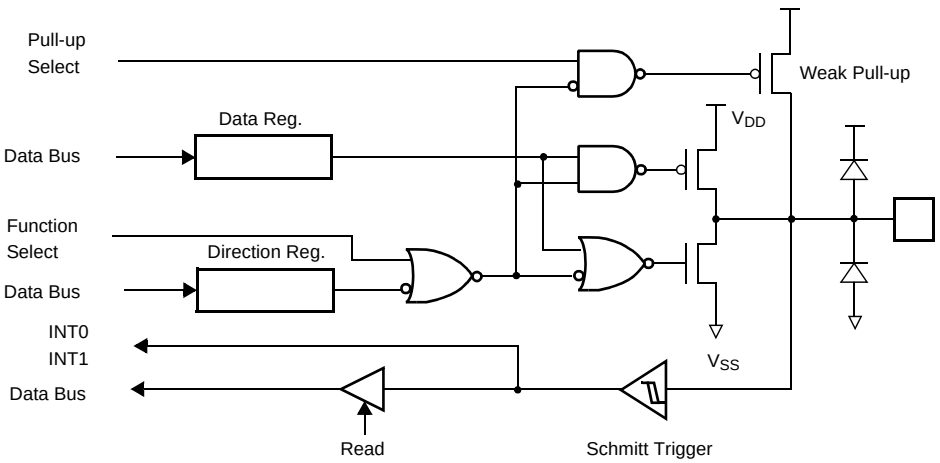
- **RB0 / AN0 / AVref**



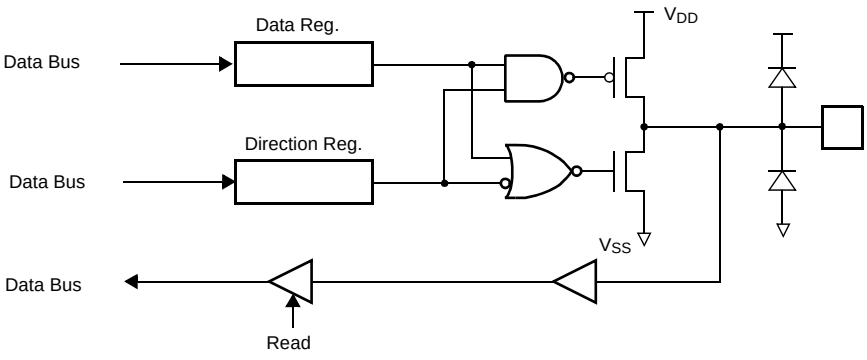
- **RB1/BUZ, RB4/PWM0/COMP**



- **RB2/INT0, RB3/INT1**



- **RC0, RC1**



10. ELECTRICAL CHARACTERISTICS

10.1 Absolute Maximum Ratings

Supply voltage ..... -0.3 to +6.5 V

Storage Temperature ..... -40 to +125 °C

Voltage on any pin with respect to Ground (V_{SS}) ..... -0.3 to V_{DD}+0.3

Maximum current out of V_{SS} pin ..... 200 mA

Maximum current into V_{DD} pin ..... 150 mA

Maximum current sunk by (I_{OL} per I/O Pin) ..... 25 mA

Maximum output current sourced by (I_{OH} per I/O Pin) ..... 15 mA

Maximum current (ΣI_{OL}) ..... 150 mA

Maximum current (ΣI_{OH}) ..... 100 mA

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

10.2 Recommended Operating Conditions

Parameter	Symbol	Condition	Specifications		Unit
			Min.	Max.	
Supply Voltage	V_{DD}	f_{XIN}=12MHz	4.5	6.0	V
		f_{XIN}=4.2MHz	2.7	6.0	
Operating Frequency	f_{XIN}	V_{DD}=4.5~6.0V	1	12	MHz
		V_{DD}=2.7~6.0V	1	4.2	
Operating Temperature	T_{OPR}	GMS87C1102/GMS87C1202	-20	85	°C
		GMS87C1102E/GMS87C1202E	-40	125	

10.3 DC Electrical Characteristics - GMS87C1102, GMS87C1202

- **(V_{DD}=4.5~6.0V, V_{SS}=0V, f_{XIN} =1MHz~12MHz, T_A=-20°C~+85°C)**

Parameter	Symbol	Pin[1]	Test Condition	Specification			Unit
				Min	Typ[2]	Max	
Input High Voltage	V_{IH1}	X_{IN}, $\overline{RESET}$	V_{DD}=4.5~6.0V	0.8V_{DD}		V_{DD}	V
	V_{IH2}	RB2, RB3		0.8V_{DD}		V_{DD}	
	V_{IH3}	RA,RB0,RB1,RB4,RC		0.7V_{DD}		V_{DD}	
Input Low Voltage	V_{IL1}	X_{IN}, $\overline{RESET}$	V_{DD}=4.5~6.0V	0		0.2V_{DD}	V
	V_{IL2}	RB2, RB3		0		0.2V_{DD}	
	V_{IL3}	RA,RB0,RB1,RB4,RC		0		0.3V_{DD}	
Output High Voltage	V_{OH}	RA, RB, RC	V_{DD} = 5V, I_{OH} = -2mA	V_{DD}-1			V
Output Low Voltage	V_{OL}	RA, RB, RC	V_{DD} = 5V, I_{OL}= 10mA			1	V
Input Leakage Current	I_{IL}	$\overline{RESET}$,RA,RB,RC	V_{IN} = V_{SS}~V_{DD}			±5	uA
	I_{IL}	X_{IN}				±15	
Input Pull-up Current	I_{PU}	RB2, RB3[3]	V_{DD} = 5V, V_{IN} = V_{SS}	-350	-280	-200	uA
Power Fail Detect Voltage	V_{PFD}	V_{DD}		2	3.5	4	V
Normal Operating Current	I_{DD}	V_{DD}	V_{DD}=5.5V, f_{XIN}=8MHz		4	6	mA
			V_{DD}=5.5V, f_{XIN}=12MHz		6	10	
Wake-up Timer Mode Current	I_{WKUP}	V_{DD}	V_{DD}=5.5V, f_{XIN}=8MHz		1	1.8	
			V_{DD}=5.5V, f_{XIN}=12MHz		2	3	mA
RC-oscillated Watchdog Timer Mode Current	I_{RCWDT}	V_{DD}	V_{DD} = 5.5V, f_{XIN} = 8MHz/12MHz		0.3	0.8	mA
STOP Mode Current	I_{STOP}	V_{DD}	V_{DD} = 5.5V		0.2	2	uA
Hysteresis	V_{T+} ~ V_{T-}	$\overline{RESET}$, RB2, RB3		0.5			V
Internal RC Oscillation Period (RC-WDT CLK)	T_{RCWDT}	X_{OUT}	V_{DD} = 5V, f_{XIN} = 8MHz/12MHz	100		200	uS
RC Oscillation Frequency (System CLK)	f_{RCOSC}	X_{OUT}	R = 20KΩ, C=24pF	300		550	kHz

1. *RC0, RC1, RB1 and RB3 pins are applied for GMS87C1202 only.*
2. *Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.*
3. *This parameter is valid when the bit PUPSELx is selected and set the Input mode or Interrupt Input Function.*

• **(V_{DD}=2.7~6.0V, V_{SS}=0V, f_{XIN} =1MHz~4.2MHz, T_A=-20°C~+85°C)**

Parameter	Symbol	Pin[1]	Test Condition	Specification			Unit
				Min	Typ[2]	Max	
Input High Voltage	V_{IH1}	X_{IN}	V_{DD}=2.7~6.0V	0.8V_{DD}		V_{DD}	V
	V_{IH2}	$\overline{RESET}$		0.9V_{DD}		V_{DD}	
	V_{IH3}	RB2, RB3		0.8V_{DD}		V_{DD}	
	V_{IH4}	RA,RB0,RB1,RB4,RC		0.7V_{DD}		V_{DD}	
Input Low Voltage	V_{IL1}	X_{IN}	V_{DD}=2.7~6.0V	0		0.2V_{DD}	V
	V_{IL2}	$\overline{RESET}$		0		0.1V_{DD}	
	V_{IL3}	RB2, RB3		0		0.2V_{DD}	
	V_{IL4}	RA,RB0,RB1,RB4,RC		0		0.3V_{DD}	
Output High Voltage	V_{OH}	RA, RB, RC	V_{DD} = 3V, I_{OH} = -2mA	V_{DD}-0.7			V
Output Low Voltage	V_{OL}	RA, RB, RC	V_{DD} = 3V, I_{OL}= 7mA			0.8	V
Input Leakage Current	I_{IL}	$\overline{RESET}$,RA,RB,RC	V_{IN} = V_{SS}~V_{DD}			±5	uA
	I_{IL}	X_{IN}				±15	
Input Pull-up Current	I_{PU}	RB2, RB3[3]	V_{DD} = 3V, V_{IN} = V_{SS}	-100	-80	-50	uA
Power Fail Detect Voltage	V_{PFD}	V_{DD}		2	3.5	4	V
Normal Operating Current	I_{DD}	V_{DD}	V_{DD} = 3V, f_{XIN} = 4MHz		1	3	mA
Wake-up Timer Mode Current	I_{WKUP}	V_{DD}	V_{DD} = 3V, f_{XIN} = 4MHz		0.3	1	mA
RC-oscillated Watchdog Timer Mode Current	I_{RCWDT}	V_{DD}	V_{DD} = 3V, f_{XIN} = 4MHz		0.1	0.6	mA
STOP Mode Current	I_{STOP}	V_{DD}	V_{DD} = 3V		0.01	1	uA
Hysteresis	V_{T+} ~ V_{T-}	$\overline{RESET}$, RB2, RB3		0.5			V
Internal RC Oscillation Period (RC-WDT CLK)	T_{RCWDT}	X_{OUT}	V_{DD} = 3V, f_{XIN} = 4MHz	200		400	uS
RC Oscillation Frequency (System CLK)	f_{RCOSC}	X_{OUT}	R = 20KΩ, C=39pF	200		400	kHz

1. *RC0, RC1, RB1 and RB3 pins are applied for GMS87C1202 only.*
2. *Data in "Typ" column is at 3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.*
3. *This parameter is valid when the bit PUPSELx is selected and set the Input mode or Interrupt Input Function.*

10.4 DC Electrical Characteristics - GMS87C1102E, GMS87C1202E(Extended version)

- **(V_{DD}=4.5~6.0V, V_{SS}=0V, f_{XIN} =1MHz~12MHz, T_A=-40°C~+125°C)**

Parameter	Symbol	Pin[1]	Test Condition	Specification			Unit
				Min	Typ[2]	Max	
Input High Voltage	V_{IH1}	X_{IN}, $\overline{RESET}$	V_{DD}=4.5~6.0V	0.8V_{DD}		V_{DD}	V
	V_{IH2}	RB2, RB3		0.8V_{DD}		V_{DD}	
	V_{IH3}	RA,RB0,RB1,RB4,RC		0.7V_{DD}		V_{DD}	
Input Low Voltage	V_{IL1}	X_{IN}, $\overline{RESET}$	V_{DD}=4.5~6.0V	0		0.2V_{DD}	V
	V_{IL2}	RB2, RB3		0		0.2V_{DD}	
	V_{IL3}	RA,RB0,RB1,RB4,RC		0		0.3V_{DD}	
Output High Voltage	V_{OH}	RA, RB, RC	V_{DD} = 5V, I_{OH} = -2mA	V_{DD}-1			V
Output Low Voltage	V_{OL}	RA, RB, RC	V_{DD} = 5V, I_{OL}= 10mA			1	V
Input Leakage Current	I_{IL}	$\overline{RESET}$,RA,RB,RC	V_{IN} = V_{SS}~V_{DD}			±5	uA
	I_{IL}	X_{IN}				±15	
Input Pull-up Current	I_{PU}	RB2, RB3[3]	V_{DD} = 5V, V_{IN} = V_{SS}	-350	-250	-100	uA
Power Fail Detect Voltage	V_{PFD}	V_{DD}		2	3.5	4.2	V
Normal Operating Current	I_{DD}	V_{DD}	V_{DD}=5.5V, f_{XIN}=8MHz		4	6	mA
			V_{DD}=5.5V, f_{XIN}=12MHz		6	10	
Wake-up Timer Mode Current	I_{WKUP}	V_{DD}	V_{DD}=5.5V, f_{XIN}=8MHz		1	1.8	
			V_{DD}=5.5V, f_{XIN}=12MHz		2	3	mA
RC-oscillated Watchdog Timer Mode Current	I_{RCWDT}	V_{DD}	V_{DD} = 5.5V, f_{XIN} = 8Mhz/12MHz		0.3	0.8	mA
STOP Mode Current	I_{STOP}	V_{DD}	V_{DD} = 5.5V		0.2	2	uA
Hysteresis	V_{T+} ~ V_{T-}	$\overline{RESET}$, RB2, RB3		0.5			V
Internal RC Oscillation Period (RC-WDT CLK)	T_{RCWDT}	X_{OUT}	V_{DD} = 5V, f_{XIN} = 8MHz/12MHz	80		250	uS
RC Oscillation Frequency (System CLK)	f_{RCOSC}	X_{OUT}	R = 20KΩ, C=24pF	300		550	kHz

1. RC0, RC1, RB1 and RB3 pins are applied for GMS87C1202 only.
2. Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.
3. This parameter is valid when the bit PUPSELx is selected and set the Input mode or Interrupt Input Function.

• **(V_{DD}=2.7~6.0V, V_{SS}=0V, f_{XIN} =1MHz~4.2MHz, T_A=-40°C~+125°C)**

Parameter	Symbol	Pin[1]	Test Condition	Specification			Unit
				Min	Typ[2]	Max	
Input High Voltage	V_{IH1}	X_{IN}	V_{DD}=2.7~6.0V	0.8V_{DD}		V_{DD}	V
	V_{IH2}	$\overline{RESET}$		0.9V_{DD}		V_{DD}	
	V_{IH3}	RB2, RB3		0.8V_{DD}		V_{DD}	
	V_{IH4}	RA,RB0,RB1,RB4,RC		0.7V_{DD}		V_{DD}	
Input Low Voltage	V_{IL1}	X_{IN}	V_{DD}=2.7~6.0V	0		0.2V_{DD}	V
	V_{IL2}	$\overline{RESET}$		0		0.1V_{DD}	
	V_{IL3}	RB2, RB3		0		0.2V_{DD}	
	V_{IL4}	RA,RB0,RB1,RB4,RC		0		0.3V_{DD}	
Output High Voltage	V_{OH}	RA, RB, RC	V_{DD} = 3V, I_{OH} = -2mA	V_{DD}-0.7			V
Output Low Voltage	V_{OL}	RA, RB, RC	V_{DD} = 3V, I_{OL}= 7mA			0.8	V
Input Leakage Current	I_{IL}	$\overline{RESET}$,RA,RB,RC	V_{IN} = V_{SS}~V_{DD}			±5	uA
	I_{IL}	X_{IN}				±15	
Input Pull-up Current	I_{PU}	RB2, RB3[3]	V_{DD} = 3V, V_{IN} = V_{SS}	-120	-80	-50	uA
Power Fail Detect Voltage	V_{PFD}	V_{DD}		2	3.5	4	V
Normal Operating Current	I_{DD}	V_{DD}	V_{DD} = 3V, f_{XIN} = 4MHz		1	3	mA
Wake-up Timer Mode Current	I_{WKUP}	V_{DD}	V_{DD} = 3V, f_{XIN} = 4MHz		0.3	1	mA
RC-oscillated Watchdog Timer Mode Current	I_{RCWDT}	V_{DD}	V_{DD} = 3V, f_{XIN} = 4MHz		0.1	0.6	mA
STOP Mode Current	I_{STOP}	V_{DD}	V_{DD} = 3V		0.01	1	uA
Hysteresis	V_{T+} ~ V_{T-}	$\overline{RESET}$, RB2, RB3		0.5			V
Internal RC Oscillation Period (RC-WDT CLK)	T_{RCWDT}	X_{OUT}	V_{DD} = 3V, f_{XIN} = 4MHz	200		450	uS
RC Oscillation Frequency (System CLK)	f_{RCOSC}	X_{OUT}	R = 20KΩ, C=39pF	200		400	kHz

1. *RC0, RC1, RB1 and RB3 pins are applied for GMS87C1202 only.*
2. *Data in "Typ" column is at 3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.*
3. *This parameter is valid when the bit PUPSELx is selected and set the Input mode or Interrupt Input Function.*

10.5 A/D Converter Characteristics - GMS87C1102, GMS87C1202

- **(V_{SS}=0V, V_{DD}=3.072V/@f_{XIN} =4MHz, V_{DD}=5.12V/@f_{XIN} =8/12MHz, T_A=-20°C~+85°C)**

Parameter	Symbol	Condition	Specifications			Unit
			Min.	Typ.	Max.	
Analog Input Voltage Range	V_{AIN}	AVREFS=0	V_{SS}	-	V_{DD}	V
		AVREFS=1	V_{SS}	-	V_{REF}	
Analog Power Supply Input Voltage Range	V_{REF}	AVREFS=1	3	-	V_{DD}	V
Overall Accuracy	N_{ACC}		-	±1.3	±1.5	LSB
Non-Linearity Error	N_{NLE}		-	–	±1.2	LSB
Differential Non-Linearity Error	N_{DNLE}		-	±1.0	±1.5	LSB
Zero Offset Error	N_{ZOE}		-	±1.0	±1.5	LSB
Full Scale Error	N_{FSE}		-	±0.25	±0.5	LSB
Gain Error	N_{NLE}		-	±1.0	±1.5	LSB
Conversion Time	T_{CONV}	f_{XIN}=4MHz	-	-	20	µS
		f_{XIN}=8MHz	-	-	10	
		f_{XIN}=12MHz	-	-	7	
AV_{REF} Input Current	I_{REF}	f_{XIN}=4MHz	-	0.4	0.6	mA
		f_{XIN}=8MHz	-	0.5	0.8	
		f_{XIN}=12MHz	-	0.6	1	

10.6 A/D Converter Characteristics - GMS87C1102E, GMS87C1202E(Extended version)

• **(V_{SS}=0V, V_{DD}=5.12V, @f_{XIN} =8MHz/12MHz, T_A=-40°C~+125°C)**

Parameter	Symbol	Condition	Specifications			Unit
			Min.	Typ.	Max.	
Analog Input Voltage Range	V_{AIN}	AVREFS=0	V_{SS}	-	V_{DD}	V
		AVREFS=1	V_{SS}	-	V_{REF}	
Analog Power Supply Input Voltage Range	V_{REF}	AVREFS=1	3	-	V_{DD}	V
Overall Accuracy	N_{ACC}		-	±1.3	±2.0	LSB
Non-Linearity Error	N_{NLE}		-	–	±2.0	LSB
Differential Non-Linearity Error	N_{DNLE}		-	±1.0	±2.0	LSB
Zero Offset Error	N_{ZOE}		-	±1.0	±2.5	LSB
Full Scale Error	N_{FSE}		-	±0.25	±1.0	LSB
Gain Error	N_{NLE}		-	±1.0	±2.0	LSB
Conversion Time	T_{CONV}	f_{XIN}=8MHz	-	-	10	µS
		f_{XIN}=12MHz	-	-	7	
AV_{REF} Input Current	I_{REF}	f_{XIN}=8MHz	-	0.5	0.8	mA
		f_{XIN}=12MHz	-	0.6	1	

• **(V_{SS}=0V, V_{DD}=3.072V, @f_{XIN} =4MHz, T_A=-40°C~+125°C)**

Parameter	Symbol	Condition	Specifications			Unit
			Min.	Typ.	Max.	
Analog Input Voltage Range	V_{AIN}	AVREFS=0	V_{SS}	-	V_{DD}	V
		AVREFS=1	V_{SS}	-	V_{REF}	
Analog Power Supply Input Voltage Range	V_{REF}	AVREFS=1	3	-	V_{DD}	V
Overall Accuracy	N_{ACC}		-	±1.3	±1.5	LSB
Non-Linearity Error	N_{NLE}		-	–	±1.2	LSB
Differential Non-Linearity Error	N_{DNLE}		-	±1.0	±1.5	LSB
Zero Offset Error	N_{ZOE}		-	±1.0	±1.5	LSB
Full Scale Error	N_{FSE}		-	±0.25	±0.5	LSB
Gain Error	N_{NLE}		-	±1.0	±1.5	LSB
Conversion Time	T_{CONV}	f_{XIN}=4MHz	-	-	20	µS
AV_{REF} Input Current	I_{REF}	f_{XIN}=4MHz	-	0.4	0.6	mA

10.7 AC Characteristics

(T_A=-20~+85°C, V_{DD}=5V±10%, V_{SS}=0V)

Parameter	Symbol	Pins	Specifications			Unit
			Min.	Typ.	Max.	
Operating Frequency	f_{CP}	X_{IN}	1	-	12	MHz
External Clock Pulse Width	t_{CPW}	X_{IN}	80	-	-	nS
External Clock Transition Time	t_{RCP},t_{FCP}	X_{IN}	-	-	20	nS
External Input Pulse Width	t_{EPW}	INT0, INT1, EC0	2	-	-	t_{SYS}
External Input Pulse Transiton Time	t_{REP},t_{FEP}	INT0, INT1, EC0	-	-	20	nS
$\overline{\text{RESET}}$ Input Width	t_{RST}	$\overline{\text{RESET}}$	8	-	-	t_{SYS}

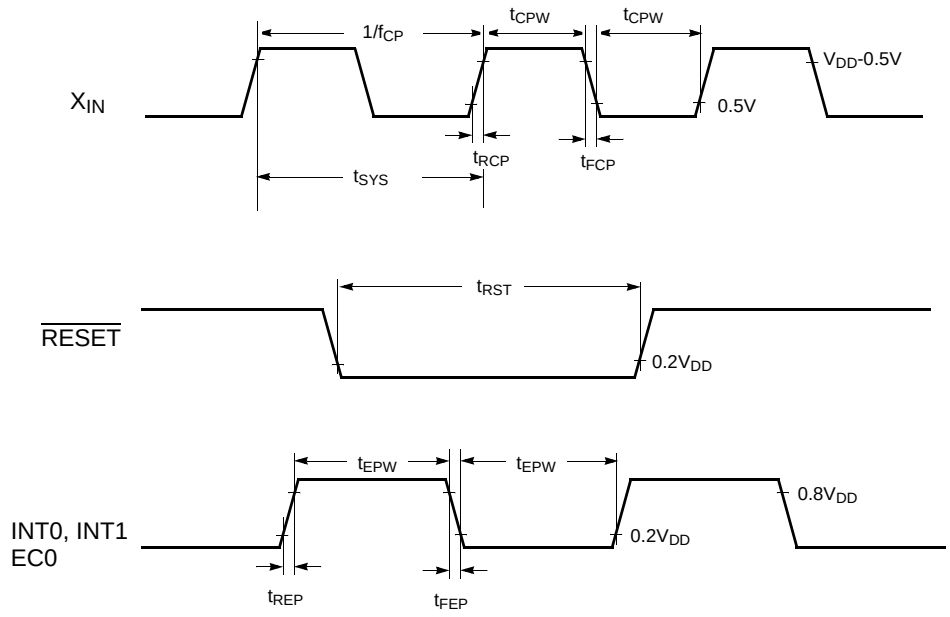


Figure 10-1 Timing Chart

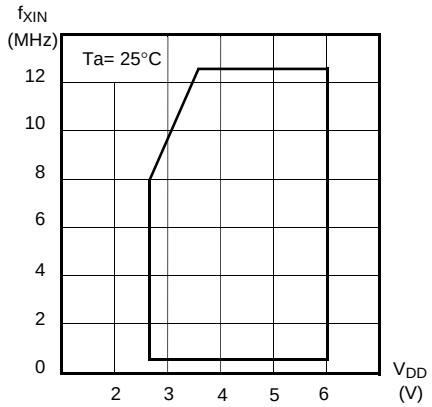
10.8 Typical Characteristics

This graphs and tables provided in this section are for design guidance only and are not tested or guranteed.

In some graphs or tables the data presented are outside specified operating range (e.g. outside specified V_{DD} range). This is for imformation only and divices are guranteed to operate properly only within the specified range.

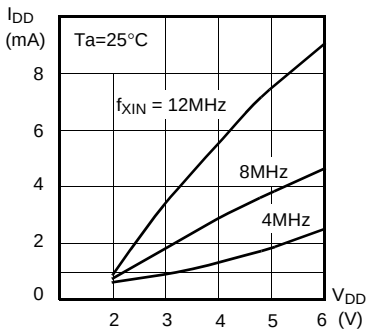
The data presented in this section is a statistical summary of data collected on units from different lots over a period of time. "Typical" represents the mean of the distribution while "max" or "min" represents (mean + 3σ) and (mean – 3σ) respectively where σ is standard deviation

Operating Area



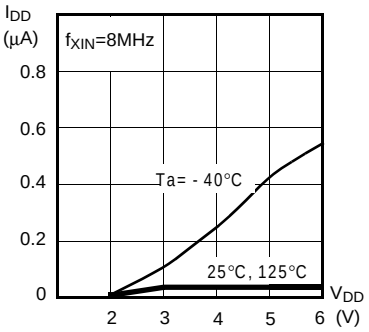
Normal Operation

I_{DD}–V_{DD}



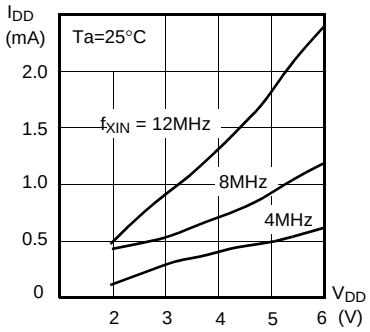
Stop Mode

I_{STOP}–V_{DD}



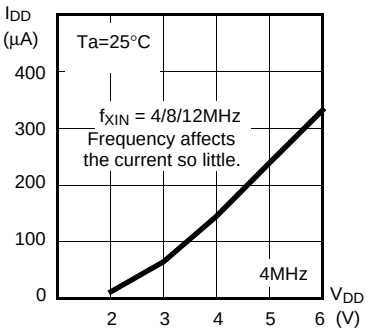
Wake-up Timer Mode

I_{WKUP}–V_{DD}

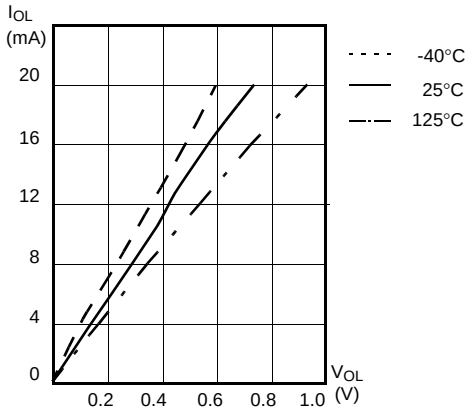


RC-WDT in Stop Mode

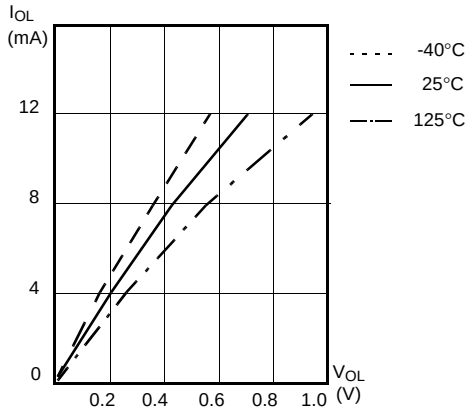
I_{RCWDT}–V_{DD}



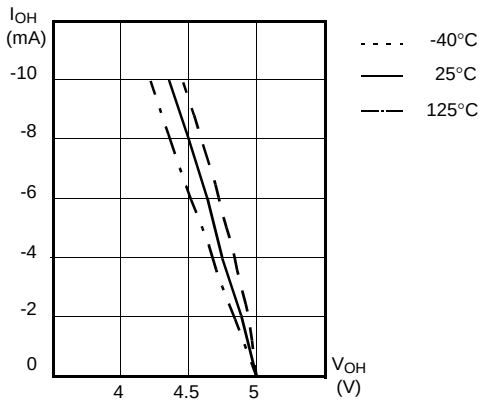
I_{OL}–V_{OL}, V_{DD}=5V



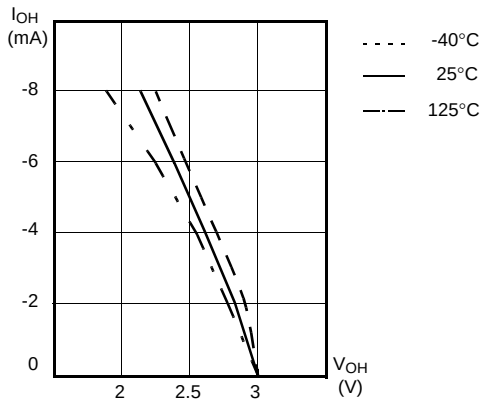
I_{OL}–V_{OL}, V_{DD}=3V



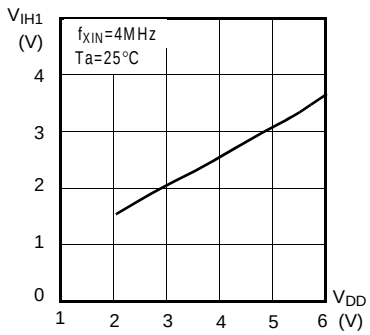
I_{OH}–V_{OH}, V_{DD}=5V



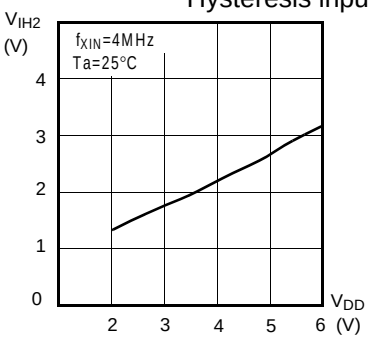
I_{OH}–V_{OH}, V_{DD}=5V



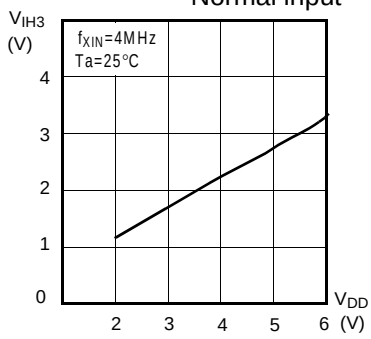
V_{DD}–V_{IH1} X_{IN}



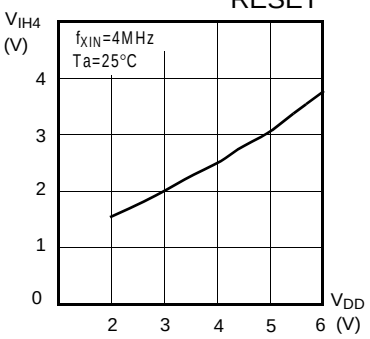
V_{DD}–V_{IH2} Hysteresis input



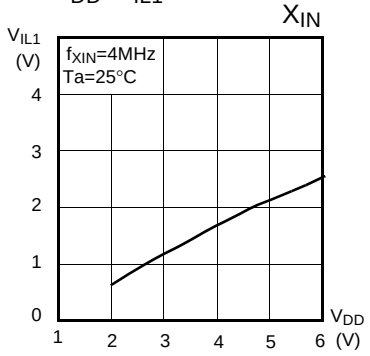
V_{DD}–V_{IH3} Normal input



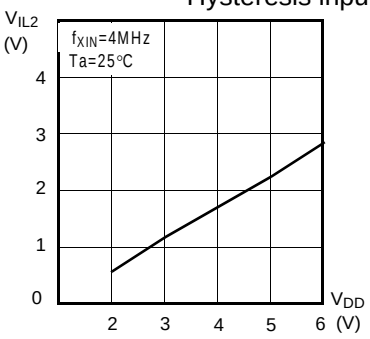
V_{DD}–V_{IH4} $\overline{\text{RESET}}$



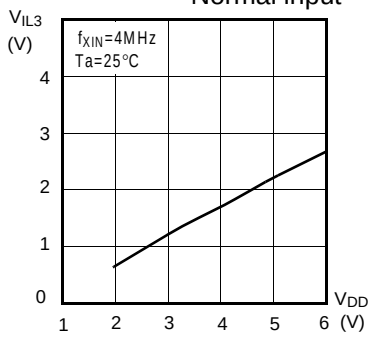
V_{DD}–V_{IL1} X_{IN}



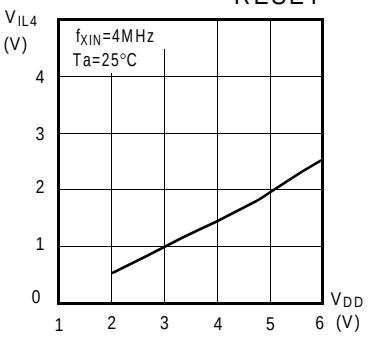
V_{DD}–V_{IL2} Hysteresis input

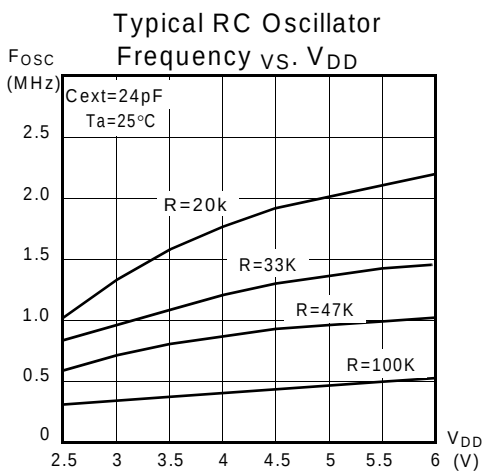


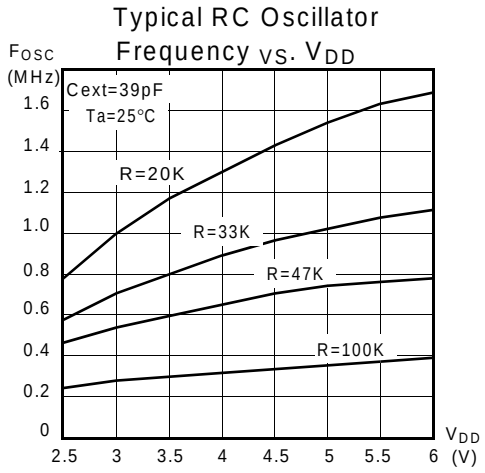
V_{DD}–V_{IL3} Normal input

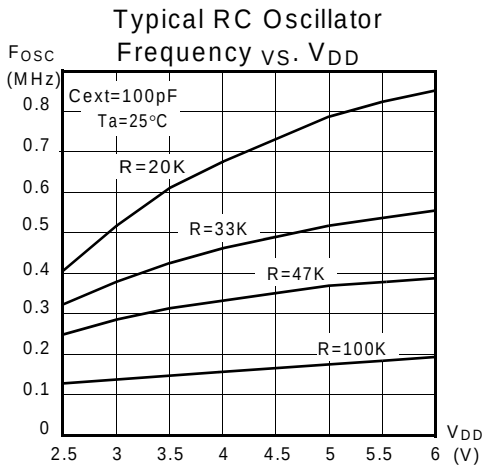


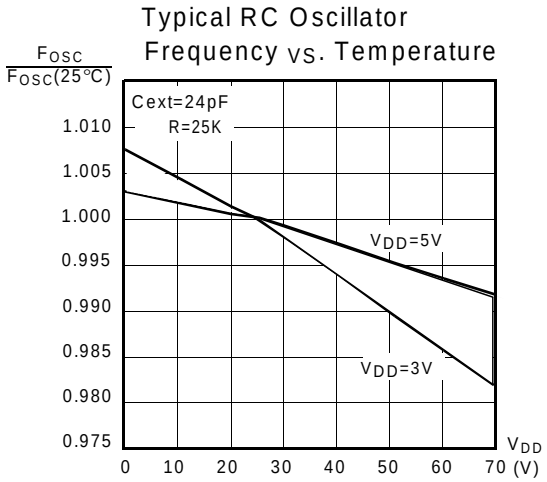
V_{DD}–V_{IL4} $\overline{\text{RESET}}$











Cext	Rext	Average	
		Fosc @ 5V,25°C	
24pF	20K	2.02MHz	±14.11%
	33K	1.34MHz	±11.50%
	47K	0.952MHz	±10.30%
	100K	0.48MHz	±9.07%
39pF	20K	1.536MHz	±14.79%
	33K	1.012MHz	±11.67%
	47K	0.72MHz	±10.42%
	100K	0.364MHz	±9.75%
100pF	20K	0.78MHz	±13.53%
	33K	0.512MHz	±10.35%
	47K	0.364MHz	±9.48%
	100K	0.18MHz	±7.34%

Table 10-1 RC Oscillator Frequencies

11. MEMORY ORGANIZATION

The GMS87C1202 has separated address spaces for Program memory and Data Memory. Program memory can only be read, not written to. It can be up to 2K bytes of Program memory. Data memory can be read and written to up to 128 bytes including the stack area.

11.1 Registers

This device has six registers that are the Program Counter (PC), a Accumulator (A), two index registers (X, Y), the Stack Pointer (SP), and the Program Status Word (PSW). The Program Counter consists of 16-bit register.

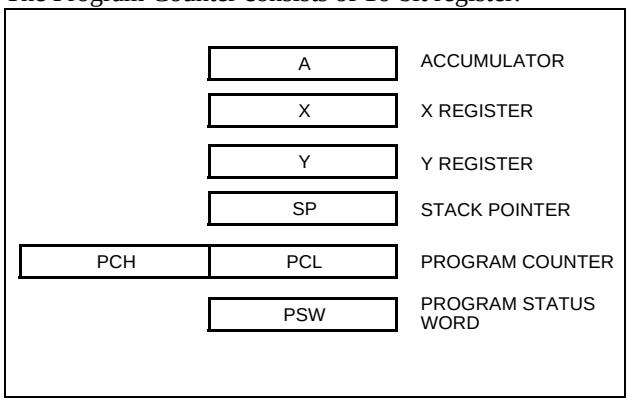


Figure 11-1 Configuration of Registers

• Accumulator

The Accumulator is the 8-bit general purpose register, used for data operation such as transfer, temporary saving, and conditional judgement, etc.

The Accumulator can be used as a 16-bit register with Y Register as shown below

.

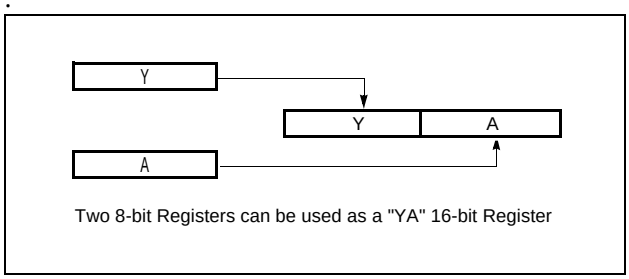


Figure 11-2 Configuration of YA 16-bit Register

• X, Y Registers

In the addressing mode which uses these index registers, the register contents are added to the specified address, which becomes the actual address. These modes are extremely effective for referencing subroutine tables and memory tables. The index registers also have increment, decrement, comparison and data transfer functions, and they can be used as simple accumulators.

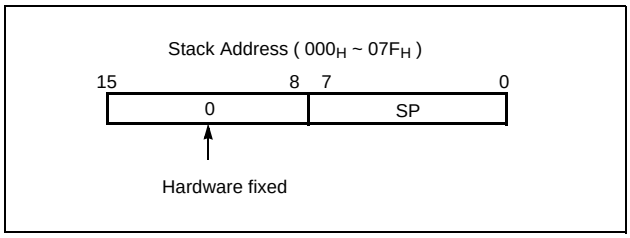
• Stack Pointer

The Stack Pointer is an 8-bit register used for occurrence interrupts and calling out subroutines. Stack Pointer identifies the location in the stack to be accessed (save or restore).

Generally, SP is automatically updated when a subroutine call is executed or an interrupt is accepted. However, if it is used in excess of the stack area permitted by the data memory allocating configuration, the user-processed data may be lost.

The stack can be located at any position within 00_H to$7F_H$ of the internal data memory. The SP is not initialized by hardware, requiring to write the initial value (the location with which the use of the stack starts) by using the initialization routine. Normally, the initial value of "$7F_H$" is used

.



Note: *The Stack Pointer must be initialized by software because its value is undefined after RESET.*
Example: To initialize the SP
LDX #07FH
TXSP ; SP ← 7F$_H$

• Program Counter

The Program Counter is a 16-bit wide which consists of two 8-bit registers, PCH and PCL. This counter indicates the address of the next instruction to be executed. In reset state, the program counter has reset routine address (PC_H:$0FF_H$, PC_L:$0FE_H$).

• Program Status Word

The Program Status Word (PSW) contains several bits that reflect the current state of the CPU. The PSW is described in Figure 11-3 . It contains the Negative flag, the Overflow flag, the Break flag the Half Carry (for BCD operation), the Interrupt enable flag, the Zero flag, and the Carry flag.

[Carry flag C]

This flag stores any carry or borrow from the ALU of CPU after an arithmetic operation and is also changed by the Shift Instruction or Rotate Instruction.

[Zero flag Z]

This flag is set when the result of an arithmetic operation or data transfer is "0" and is cleared by any other result.

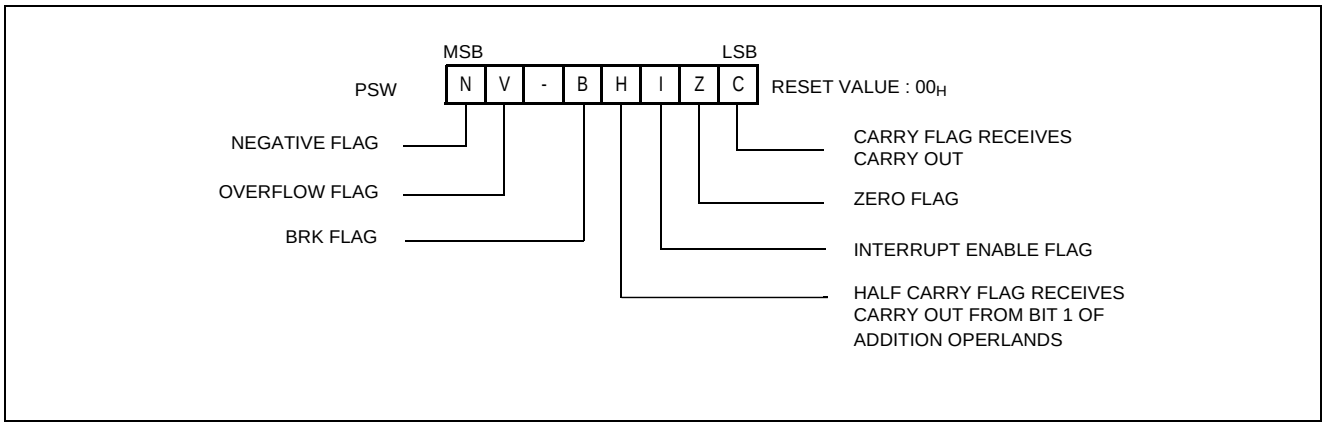


Figure 11-3 PSW (Program Status Word) Register

[Interrupt disable flag I]

This flag enables/disables all interrupts except interrupt caused by Reset or software BRK instruction. All interrupts are disabled when cleared to "0". This flag immediately becomes "0" when an interrupt is served. It is set by the EI instruction and cleared by the DI instruction.

[Half carry flag H]

After operation, this is set when there is a carry from bit 3 of ALU or there is no borrow from bit 4 of ALU. This bit can not be set or cleared except CLRV instruction with Overflow flag (V).

[Break flag B]

This flag is set by software BRK instruction to distinguish BRK from TCALL instruction with the same vector address.

[Overflow flag V]

This flag is set to "1" when an overflow occurs as the result of an arithmetic operation involving signs. An overflow occurs when the result of an addition or subtraction exceeds +127($7F_H$) or -128(80_H). The CLRV instruction clears the overflow flag. There is no set instruction. When the BIT instruction is executed, bit 6 of memory is copied to this flag.

[Negative flag N]

This flag is set to match the sign bit (bit 7) status of the result of a data or arithmetic operation. When the BIT instruction is executed, bit 7 of memory is copied to this flag.

11.2 Program Memory

A 16-bit program counter is capable of addressing up to 64K bytes, but this device has 2K bytes program memory space only the physically implemented. Accessing a location above $FFFF_H$ will cause a wrap-around to 0000_H.

Figure 11-5 shows a map of the upper part of the Program Memory. After reset, the CPU begins execution from reset vector which is stored in address $FFFE_H$, $FFFF_H$.

As shown in Figure 11-5, each area is assigned a fixed location in Program Memory. Program Memory area contains the user program, Page Call (PCALL) area contains subroutine program, to reduce program byte length because of using by 2 bytes PCALL instead of 3 bytes CALL instruction. If it is frequently called, more useful to save program byte length.

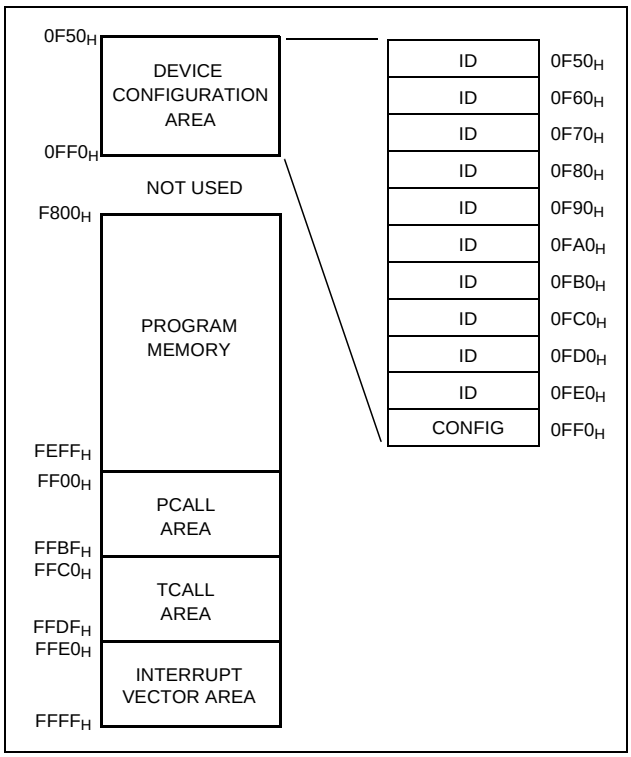


Figure 11-4 Program Memory Map

The Device Configuration Area can be programmed or left unprogrammed to select device configuration such as RC oscillation option. This area is not accessible during normal execution but is readable and writable during program / verify.

More detail informations are explained in device configuration area section.

Table Call (TCALL) causes the CPU to jump to each TCALL address, where it commences execution of the service routine. The Table Call service locations are spaced at 2-byte interval : $FFC0_H$ for TCALL15, $FFC2_H$ for TCALL14, etc.

The interrupt causes the CPU to jump to specific location, where it commences execution of the service routine. The External interrupt 0, for example, is assigned to location $FFFA_H$. The interrupt service locations are spaced at 2-byte interval : $FFF8_H$ for External Interrupt 1, $FFFA_H$ for External Interrupt 0, etc.

Address	TCALL Name
FFC0H	TCALL15
FFC2H	TCALL14
FFC4H	TCALL13
FFC6H	TCALL12
FFC8H	TCALL11
FFCAH	TCALL10
FFCCH	TCALL9
FFCEH	TCALL8
FFD0H	TCALL7
FFD2H	TCALL6
FFD4H	TCALL5
FFD6H	TCALL4
FFD8H	TCALL3
FFDAH	TCALL2
FFDCH	TCALL1
FFDEH	TCALL0 / BRK [1]

Table 11-1 TCALL Vectors

1. The BRK software interrupt is using same address with TCALL0.

As for the area from $FF00_H$ to $FFFF_H$, if any area of them is not going to be used, its service location is available as general purpose Program Memory.

Address	Vector Name
FFE0H	Not Used
FFE2H	Not Used
FFE4H	Not Used
FFE6H	Basic Interval Timer
FFE8H	Watchdog Timer
FFEAH	A/D Converter
FFECH	Not Used
FFEEH	Not Used
FFF0H	Not Used
FFF2H	Not Used
FFF4H	Timer / Counter 1
FFF6H	Timer / Counter 0
FFF8H	External Interrupt 1
FFFAH	External Interrupt 0
FFFCH	Not Used
FFFEH	RESET

Table 11-2 Interrupt Vectors

Page Call (PCALL) area contains subroutine program to

reduce program byte length by using 2 bytes PCALL instead of 3 bytes CALL instruction. If it is frequently called, it is more useful to save program byte length.

Table Call (TCALL) causes the CPU to jump to each TCALL address, where it commences the execution of the service routine. The Table Call service area spaces 2-byte for every TCALL: $0FFC0_H$ for TCALL15, $0FFC2_H$ for TCALL14, etc., as shown in Figure 11-5 .

Example: Usage of TCALL

```
LDA       #5
          TCALL  0FH          ;1BYTE INSTRUCTION
          :                   ;INSTEAD OF 3 BYTES
          :                   ;NORMAL CALL
;
;TABLE CALL ROUTINE
;
FUNC_A:   LDA    LRG0
          RET
;
FUNC_B:   LDA    LRG1     ②  ①
          RET
;
;TABLE CALL ADD. AREA
;
          ORG    0FFC0H       ;TCALL ADDRESS AREA
          DW     FUNC_A
          DW     FUNC_B
```

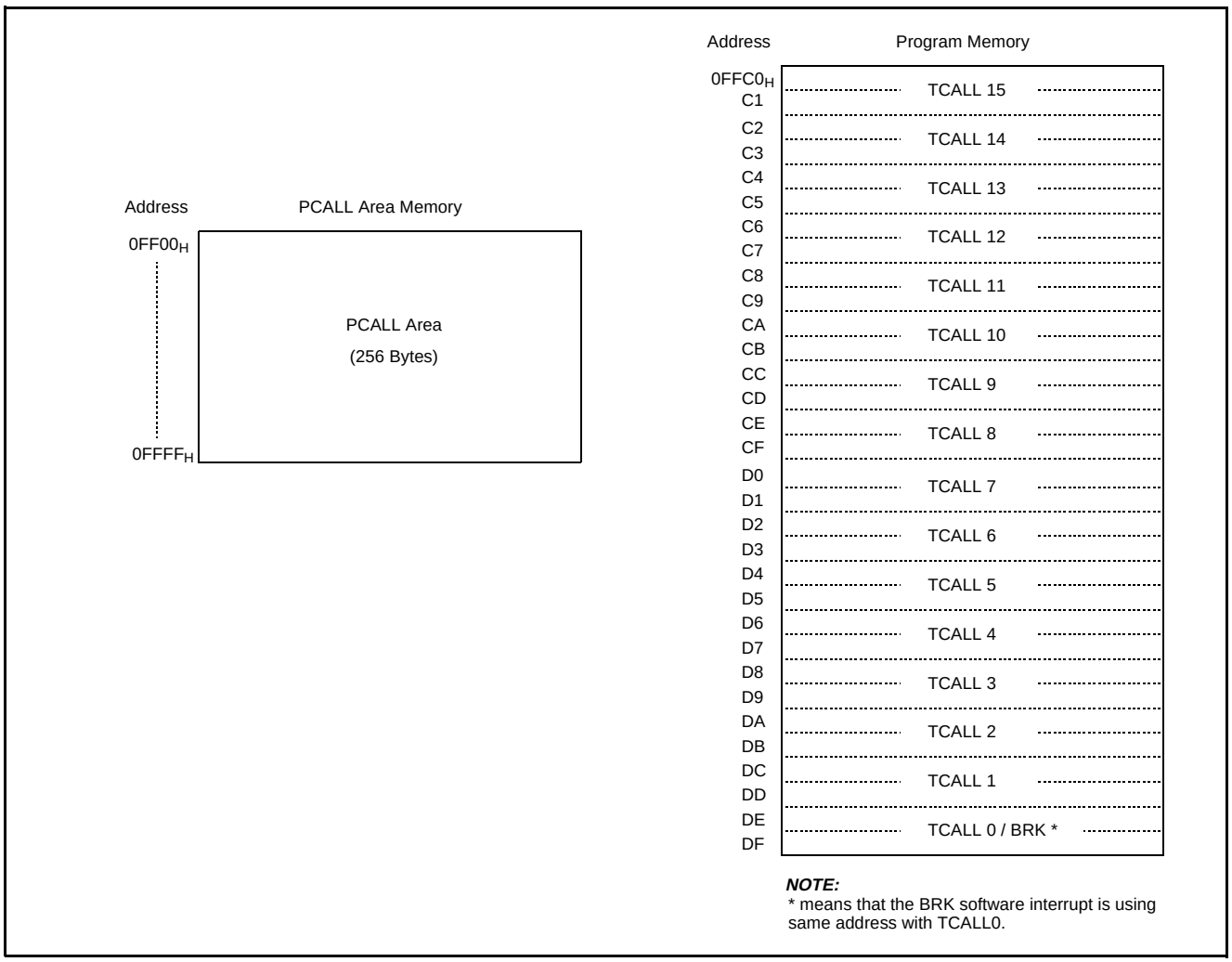
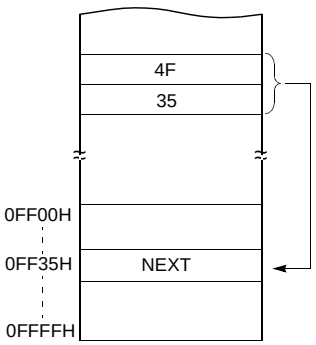


Figure 11-5 PCALL and TCALL Memory Area

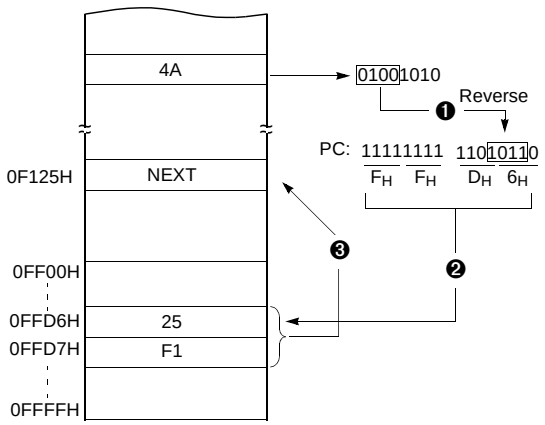
PCALL→ rel

4F35 PCALL 35H



TCALL→ n

4A TCALL 4



Example: The usage software example of Vector address and the initialize part.

```
        ORG     0FFE0H

        DW      NOT_USED            ; (0FFEO)
        DW      NOT_USED            ; (0FFE2)
        DW      NOT_USED            ; (0FFE4)
        DW      BIT_INT             ; (0FFE6) Basic Interval Timer
        DW      WDT_INT             ; (0FFE8) Watchdog Timer
        DW      AD_INT              ; (0FFEA) A/D
        DW      NOT_USED            ; (0FFEC)
        DW      NOT_USED            ; (0FFEE)
        DW      NOT_USED            ; (0FFF0)
        DW      NOT_USED            ; (0FFF2)
        DW      TMR1_INT            ; (0FFF4) Timer-1
        DW      TMR0_INT            ; (0FFF6) Timer-0
        DW      INT1                ; (0FFF8) Int.1
        DW      INT0                ; (0FFFA) Int.0
        DW      NOT_USED            ; (0FFFC)
        DW      RESET               ; (0FFFE) Reset

        ORG     0F800H

;*****************************************
;           MAIN     PROGRAM             *
;*****************************************
;
RESET:  DI                          ;Disable All Interrupts
        LDX     #0
RAM_CLR: LDA    #0                  ;RAM Clear(!0000H->!007FH)
        STA     {X}+
        CMPX    #080H
        BNE     RAM_CLR
;
        LDX     #07FH               ;Stack Pointer Initialize
        TXSP
;
        CALL    INITIAL             ;
;
        LDM     RA, #0              ;Normal Port A
        LDM     RAIO,#1000_0010B    ;Normal Port Direction
        LDM     RB, #0              ;Normal Port B
        LDM     RBIO,#1000_0010B    ;Normal Port Direction
        :
        :
        LDM     PFDR,#0             ;Enable Power Fail Detector
        :
```

11.3 Data Memory

Figure 11-6 shows the internal Data Memory space available. Data Memory is divided into two groups, a user RAM(including Stack) and control registers.

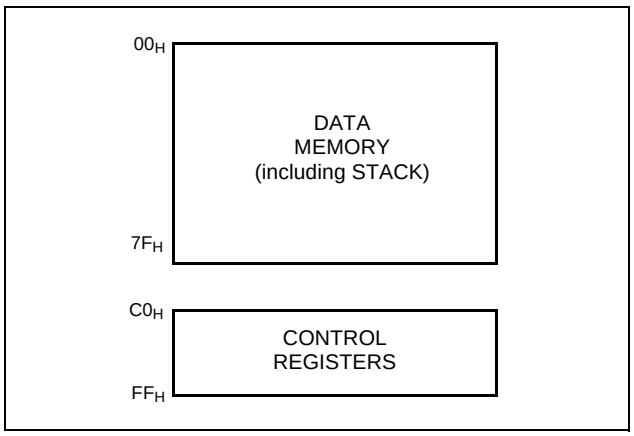


Figure 11-6 Data Memory Map

Internal Data Memory addresses are always one byte wide, which implies an address space of 128 bytes including the stack area.

The stack pointer should be initialized within 00_H to $7F_H$ by software because its value is undefined after RESET.

The Stack area is defined at the Data Memory area, so the stack should not be overlapped by manipulating RAM Data. For example, we assumed the Stack pointer is 6F. If this address is accessed by program, the stack value is changed. So the malfunction is occurred.

The control registers are used by CPU and Peripheral functions for controlling the desired operation of the device. Therefore these registers contain control and status bits for the interrupt system, the timer/ counters, analog to digital converters, I/O ports. The control registers are in address $C0_H$ to FF_H.

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

More detail informations of each register are explained in each peripheral sections.

Note: *Write only registers can not be accessed by bit manipulation instruction. Do not use read-modify-write instruction. Use byte manipulation instruction.*

Example; To write at CKCTLR

```
LDM   CKCTLR,#09H ;Divide ratio ÷16
```

Address	Symbol	R/W	RESET Value
C0H	RA	R/W	Undefined
C1H	RAIO	W	0000_0000
C2H	RB	R/W	Undefined
C3H	RBIO	W	---0_0000
C4H	RC	R/W	Undefined
C5H	RCIO	W	----_--00
CAH	RAFUNC	W	0000_0000
CBH	RBFUNC	W	---0_0000
CCH	PUPSEL	W	----_--00
D0H	TM0	R/W	--00_0000
D1H	T0	R	0000_0000
D1H	TDR0	W	1111_1111
D1H	CDR0	R	0000_0000
D2H	TM1	R/W	0000_0000
D3H	TDR1	W	1111_1111
D3H	T1PPR	W	1111_1111
D4H	T1	R	0000_0000
D4H	CDR1	R	0000_0000
D4H	T1PDR	R/W	0000_0000
D5H	PWMHR	W	----_0000
DEH	BUR	W	1111_1111
E2H	IENH	R/W	0000_----
E3H	IENL	R/W	000-_----
E4H	IRQH	R/W	0000_----
E5H	IRQL	R/W	000-_----
E6H	IEDS	R/W	----_0000
EAH	ADCM	R/W	--00_0001
EBH	ADCR	R	Undefined
ECH	BITR	R	0000_0000
ECH	CKCTLR	W	-001_0111
EDH	WDTR	R/W	0111_1111
EFH	PFDR	R/W	----_-100

Table 11-3 RESET Value of Control Registers

Note: *Several names are given at same address. Refer to below table.*

Addr.	When read			When write	
	Timer Mode	Capture Mode	PWM Mode	Timer Mode	PWM Mode
D1H	T0	CDR0	-	TDR0	-
D3H	-			TDR1	T1PPR
D4H	T1	CDR1	T1PDR	-	T1PDR
ECH	BITR			CKCTLR	

Table 11-4 Various Register Name in Same Adress

Stack Area

The stack provides the area where the return address is saved before a jump is performed during the processing routine at the execution of a subroutine call instruction or the acceptance of an interrupt.

When returning from the processing routine, executing the subroutine return instruction [RET] restores the contents of the program counter from the stack; executing the interrupt return instruction [RETI] restores the contents of the program counter and flags.

The save/restore locations in the stack are determined by the stack pointed (SP). The SP is automatically decreased after the saving, and increased before the restoring. This means the value of the SP indicates the stack location number for the next save.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
C0H	RA	RA Port Data Register							
C1H	RAIO	RA Port Direction Register							
C2H	RB	RB Port Data Register							
C3H	RBIO	RB Port Direction Register							
C4H	RC	RC Port Data Register							
C5H	RCIO	RC Port Direction Register							
CAH	RAFUNC	ANSEL7	ANSEL6	ANSEL5	ANSEL4	ANSEL3	ANSEL2	ANSEL1	ANSEL0
CBH	RBFUNC	-	-	-	PWMO	INT1I	INT0I	BUZO	AVREFS
CCH	PUPSEL	-	-	-	-	-	-	PUPSEL1	PUPSEL0
D0H	TM0	-	-	CAP0	T0CK2	T0CK1	T0CK0	T0CN	T0ST
D1H	T0/TDR0/ CDR0	Timer0 Register / Timer Data Register 0 / Capture Data Register 0							
D2H	TM1	POL	16BIT	PWME	CAP1	T1CK1	T1CK0	T1CN	T1ST
D3H	TDR1/ T1PPR	Timer Data Register 1/ PWM Period Register 1							
D4H	T1/CDR1/ T1PDR	Timer1 Register / Capture Data Register 1 / PWM Duty Register 1							
D5H	PWMHR	PWM High Register							
DEH	BUR	BUCK1	BUCK0	BUR5	BUR4	BUR3	BUR2	BUR1	BUR0
E2H	IENH	INT0E	INT1E	T0E	T1E	-	-	-	-
E3H	IENL	ADE	WDTE	BITE	-	-	-	-	-
E4H	IRQH	INT0IF	INT1IF	T0IF	T1IF	-	-	-	-
E5H	IRQL	ADIF	WDTIF	BITIF	-	-	-	-	-
E6H	IEDS	-	-	-	-	IED1H	IED1L	IED0H	IED0L
EAH	ADCM	-	-	ADEN	ADS2	ADS1	ADS0	ADST	ADSF
EBH	ADCR	ADC Result Data Register							
ECH	BITR[1]	Basic Interval Timer Data Register							
ECH	CKCTLR Note1	-	WAKEUP	RCWDT	WDTON	BTCL	BTS2	BTS1	BTS0
EDH	WDTR	WDTCL	7-bit Watchdog Counter Register						
EFH	PFDR[2]	-	-	-	-	-	PFDIS	PFDM	PFDS

Table 11-5 Control Registers of GMS87C1202

These registers of shaded area can not be accessed by bit manipulation instruction as "SET1, CLR1", so should be accessed by register operation instruction as "LDM dp,#imm".

1. *The register BITR and CKCTLR are located at same address. Address ECH is read as BITR, written to CKCTLR.*
2. *The register PFDR only be implemented on devices, not on In-circuit Emulator.*

11.4 Addressing Mode

The GMS87C1201 and GMS87C1202 use six addressing modes.

- **Register addressing**
- **Immediate addressing**
- **Direct page addressing**
- **Absolute addressing**
- **Indexed addressing**
- **Register-indirect addressing**

Below example is shown for GMS87C1202.

(1) Register Addressing

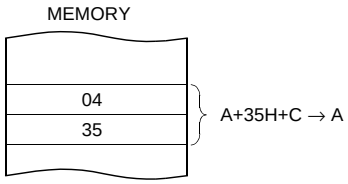
Register addressing accesses the A, X, Y, C and PSW.

(2) Immediate Addressing → #imm

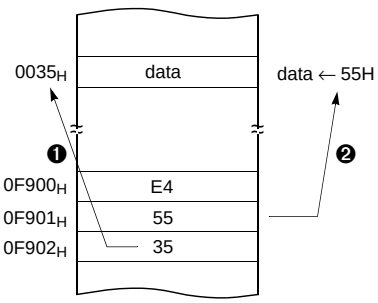
In this mode, second byte (operand) is accessed as a data immediately.

Example:

```
0435     ADC    #35H
```




```
E45535   LDM    35H,#55H
```

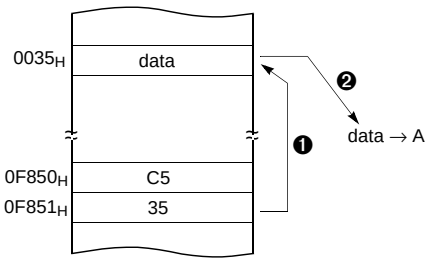


(3) Direct Page Addressing → dp

In this mode, a address is specified within direct page.

Example;

```
C535     LDA    35H          ;A ←RAM[35H]
```



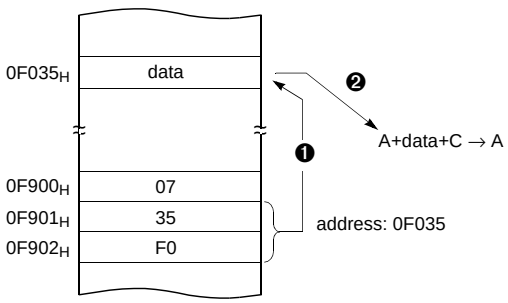
(4) Absolute Addressing → !abs

Absolute addressing sets corresponding memory data to Data , i.e. second byte(Operand I) of command becomes lower level address and third byte (Operand II) becomes upper level address.
With 3 bytes command, it is possible to access to whole memory area.

ADC, AND, CMP, CMPX, CMPY, EOR, LDA, LDX, LDY, OR, SBC, STA, STX, STY

Example;

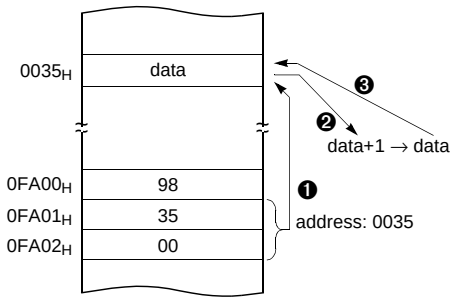
```
0735F0   ADC    !0F035H      ;A ←ROM[0F035H]
```



The operation within data memory (RAM)
ASL, BIT, DEC, INC, LSR, ROL, ROR

Example; Addressing accesses the address 0035_H .

```
983500   INC    !0035H        ;A ←RAM[035H]
```



(5) Indexed Addressing

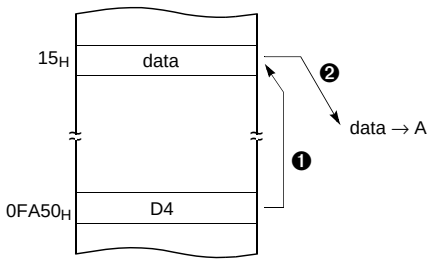
X indexed direct page (no offset) → {X}

In this mode, a address is specified by the X register.

ADC, AND, CMP, EOR, LDA, OR, SBC, STA, XMA

Example; $X=15_H$

```
D4       LDA    {X}           ;ACC←RAM[X].
```



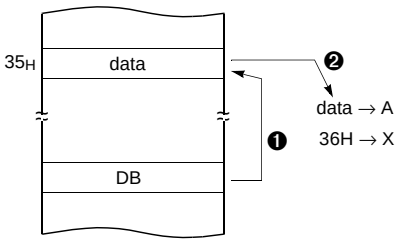
X indexed direct page, auto increment→ {X}+

In this mode, a address is specified within direct page by the X register and the content of X is increased by 1.

LDA, STA

Example; $X=35_H$

```
DB       LDA    {X}+
```



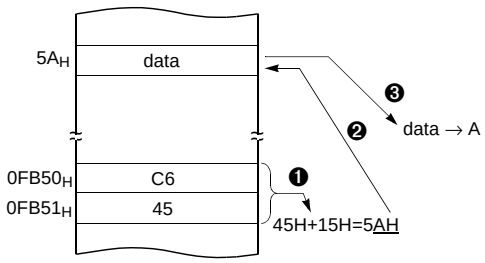
X indexed direct page (8 bit offset) → dp+X

This address value is the second byte (Operand) of command plus the data of X-register. And it assigns the memory in Direct page.

ADC, AND, CMP, EOR, LDA, LDY, OR, SBC, STA STY, XMA, ASL, DEC, INC, LSR, ROL, ROR

Example; $X=015_H$

```
C645     LDA    45H+X
```



Y indexed direct page (8 bit offset) → dp+Y

This address value is the second byte (Operand) of command plus the data of Y-register, which assigns Memory in Direct page.

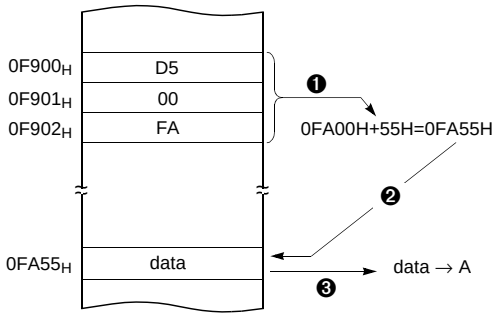
This is same with above (2). Use Y register instead of X.

Y indexed absolute → !abs+Y

Sets the value of 16-bit absolute address plus Y-register data as Memory. This addressing mode can specify memory in whole area.

Example; Y=55_H

```
D500FA   LDA    !0FA00H+Y
```



(6) Indirect Addressing

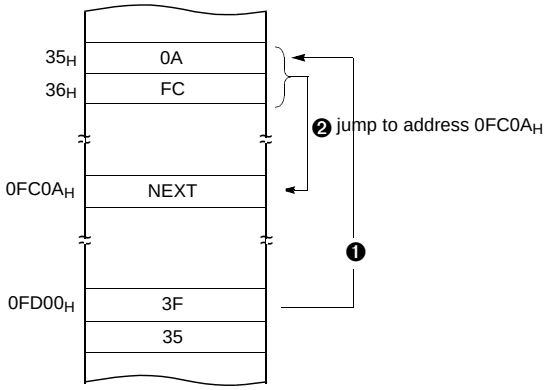
Direct page indirect → [dp]

Assigns data address to use for accomplishing command which sets memory data(or pair memory) by Operand. Also index can be used with Index register X,Y.

JMP, CALL

Example;

```
3F35     JMP    [35H]
```



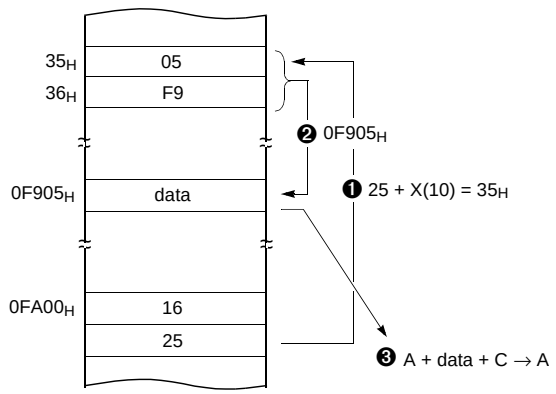
X indexed indirect → [dp+X]

Processes memory data as Data, assigned by 16-bit pair memory which is determined by pair data [dp+X+1][dp+X] Operand plus X-register data in Direct page.

ADC, AND, CMP, EOR, LDA, OR, SBC, STA

Example; X=10_H

```
1625     ADC    [25H+X]
```



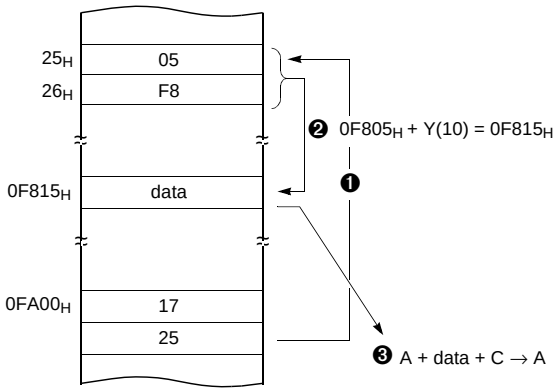
Y indexed indirect → [dp]+Y

Processes momory data as Data, assigned by the data [dp+1][dp] of 16-bit pair memory paired by Operand in Direct page plus Y-register data.

ADC, AND, CMP, EOR, LDA, OR, SBC, STA

Example; $Y=10_H$

```
1725      ADC    [25H]+Y
```



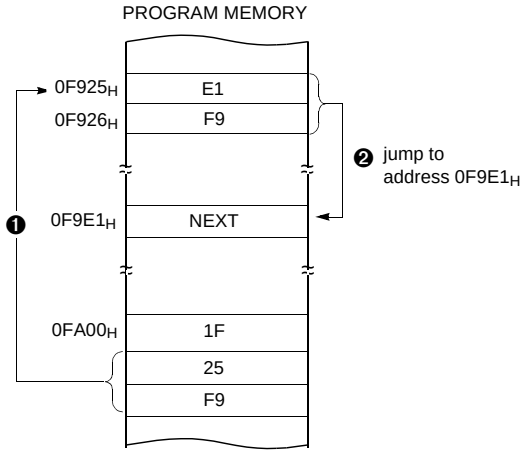
Absolute indirect → [!abs]

The program jumps to address specified by 16-bit absolute address.

JMP

Example;

```
1F25F9   JMP    [!0F925H]
```



12. I/O PORTS

The GMS87C1202 has three ports, RA, RB and RC. These ports pins may be multiplexed with an alternate function for the peripheral features on the device. In general, when a initial reset state, all ports are used as a general purpose input port.

All pins have data direction registers which can set these ports as output or input. A “1” in the port direction register defines the corresponding port pin as output. Conversely, write “0” to the corresponding bit to specify as an input pin. For example, to use the even numbered bit of RA as output ports and the odd numbered bits as input ports, write “55_H” to address $C1_H$ (RA direction register) during initial setting as shown in Figure 12-1.

Reading data register reads the status of the pins whereas writing to it will write to the port latch.

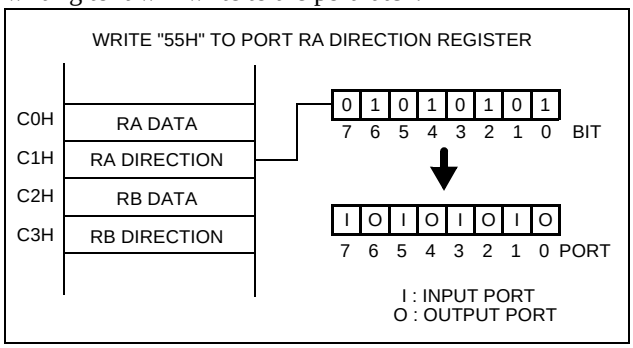


Figure 12-1 Example of port I/O assignment

12.1 RA and RAIO registers

RA is an 8-bit bidirectional I/O port (address $C0_H$). Each port can be set individually as input and output through the RAIO register (address $C1_H$).

RA7~RA1 ports are multiplexed with Analog Input Port (AN7~AN1) and RA0 port is multiplexed with Event Counter Input Port (EC0).

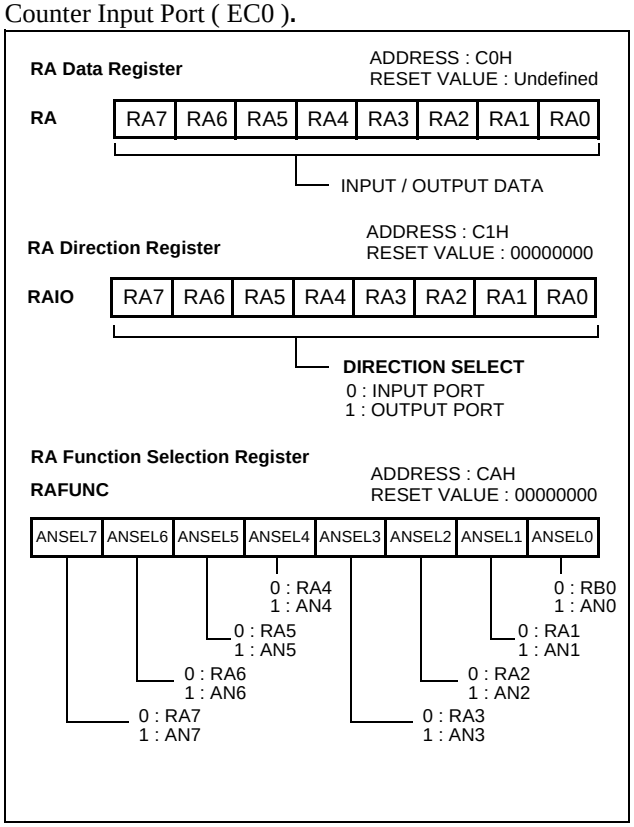


Figure 12-2 Registers of Port RA

The control register RAFUNC (address CA_H) controls to select alternate function. After reset, this value is "0", port may be used as general I/O ports. To select alternate function such as Analog Input or External Event Counter Input, write "1" to the corresponding bit of RAFUNC.Regardless of the direction register RAIO, RAFUNC is selected to use as alternate functions, port pin can be used as a corresponding alternate features (RA0/EC0 is controlled by RBFUNC)

PORT	RAFUNC.7~0	Description
RA7/AN7	0	RA7 (Normal I/O Port)
	1	AN7 (ADS2~0=111)
RA6/AN6	0	RA6 (Normal I/O Port)
	1	AN6 (ADS2~0=110)
RA5/AN5	0	RA5 (Normal I/O Port)
	1	AN5 (ADS2~0=101)
RA4/AN4	0	RA4 (Normal I/O Port)
	1	AN4 (ADS2~0=100)
RA3/AN3	0	RA3 (Normal I/O Port)
	1	AN3 (ADS2~0=011)
RA2/AN2	0	RA2 (Normal I/O Port)
	1	AN2 (ADS2~0=010)
RA1/AN1	0	RA1 (Normal I/O Port)
	1	AN1 (ADS2~0=001)
RA0/EC0[1]		RA0 (Normal I/O Port)
		EC0 (T0CK2~0=111)

*1. This port is **not an Analog Input port**, but Event Counter clock source input port. ECO is controlled by setting TOCK2~0 = 111.*
***The bit RAFUNC.0 (ANSEL0) controls the RB0/AN0/AVref port (Refer to Port RB)**.*

12.2 RB and RBIO registers

RB is a 5-bit bidirectional I/O port (address $C2_H$). Each pin can be set individually as input and output through the RBIO register (address $C3_H$).

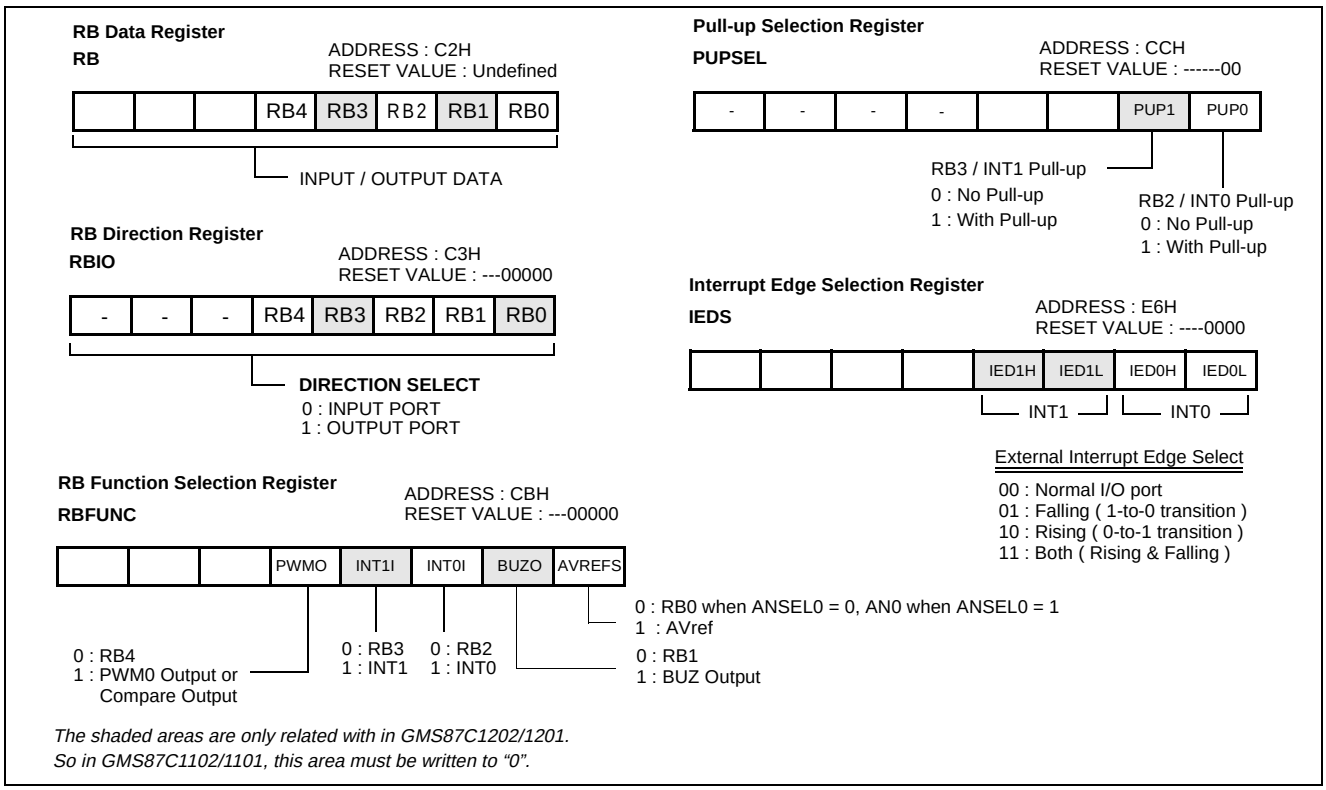


Figure 12-3 Registers of Port RB

In addition, Port RB is multiplexed with various special features. The control register RBFUNC (address CB_H) controls to select alternate function. After reset, this value is "0", port may be used as general I/O ports. To select alternate function such as External interrupt or Timer compare output, write "1" to the corresponding bit of RBFUNC.

Regardless of the direction register RBIO, RBFUNC is selected to use as alternate functions, port pin can be used as a corresponding alternate features.

PORT	RBFUNC.4~0	Description
RB4/ PWM0/ COMP0	0	RB4 (Normal I/O Port)
	1	PWM0 Output / Timer1 Compare Output
RB3/INT1	0	RB3 (Normal I/O Port)
	1	External Interrupt Input 1
RB2/INT0	0	RB2 (Normal I/O Port)
	1	External Interrupt Input 0
RB1/BUZ	0	RB1 (Normal I/O Port)
	1	Buzzer Output
RB0/AN0/ AVref	0[1]	RB0 (Normal I/O Port) / AN0 (ANSEL0=1)
	1[2]	External Analog Reference Voltage

1. When ANSEL0 = "0", this port is defined for normal I/O port (RB0).
When ANSEL0 = "1" and ADS2~0 = " 000", this port can be used Analog Input Port (AN0).
2. When this bit set to "1", this port defined for AVref , so it can not be used Analog Input Port AN0 and Normal I/O Port RB0.

12.3 RC and RCIO registers

RC is an 4-bit bidirectional I/O port (address $C4_H$). Each pin can be set individually as input and output through the RCIO register (address $C5_H$).

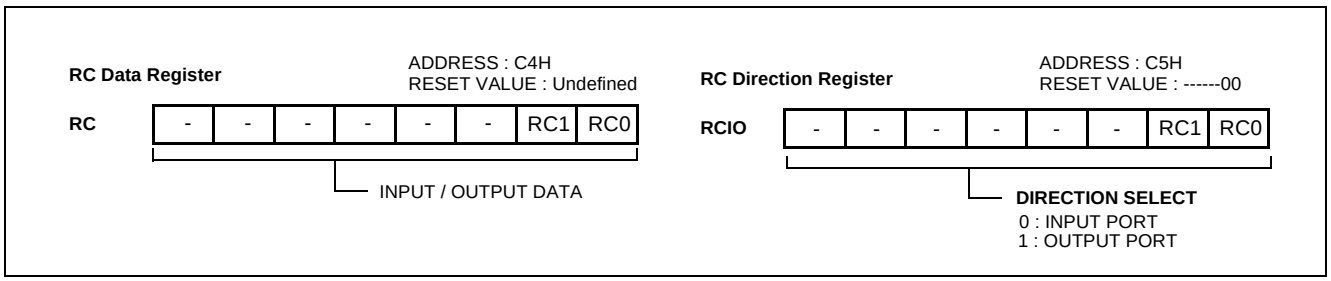


Figure 12-4 Registers of Port RC

13. CLOCK GENERATOR

The clock generator produces the basic clock pulses which provide the system clock to be supplied to the CPU and peripheral hardware. The main system clock oscillator oscillates with a crystal resonator or a ceramic resonator connected to the Xin and Xout pins. External clocks can be input to the main system clock oscillator. In this case, input a clock signal to the Xin pin and open the Xout pin

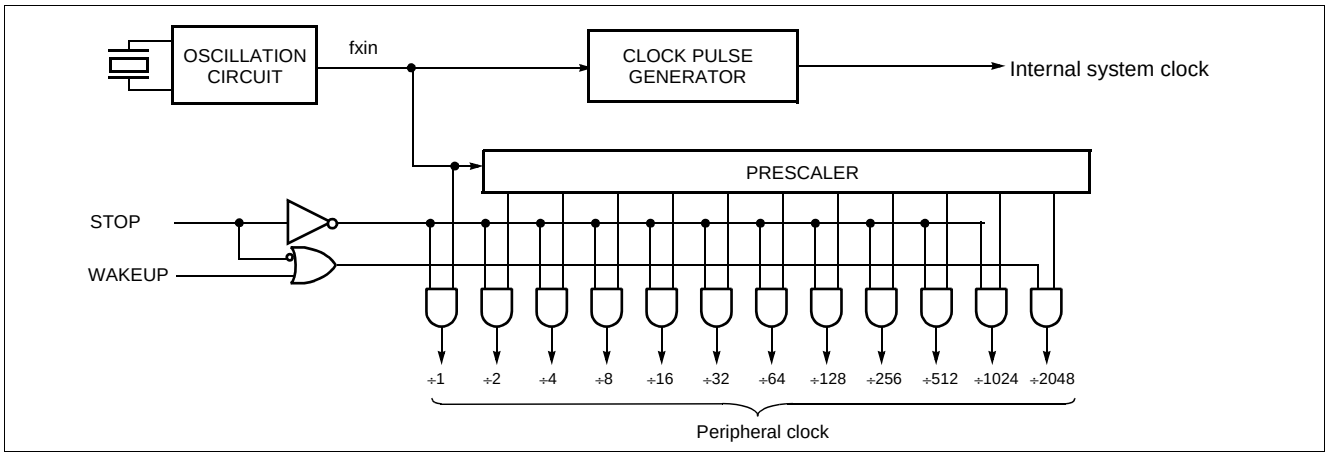


Figure 13-1 Block Diagram of Clock Pulse Generator

13.1 Oscillation Circuit

X_{IN} and X_{OUT} are the input and output, respectively, a inverting amplifier which can be set for use as an on-chip oscillator, as shown in Figure 13-2 .

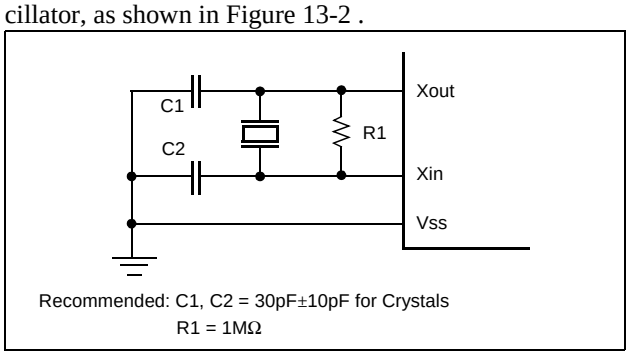


Figure 13-2 Oscillator Connections

To drive the device from an external clock source, Xout should be left unconnected while Xin is driven as shown in Figure 13-3. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the data sheet must be observed.

Oscillation circuit is designed to be used either with a ceramic resonator or crystal oscillator. Since each crystal and ceramic resonator have their own characteristics, the user should consult the crystal manufacturer for appropriate values of external components

In addition, the GMS87C1202 has an ability for the external RC oscillated operation. It offers additional cost savings for **timing insensitive applications**. The RC oscillator frequency is a function of the supply voltage, the external resistor (Rext) and capacitor (Cext) values, and the operating temperature.

The user needs to take into account variation due to tolerance of external R and C components used. Figure 13-4 shows how the RC combination is connected to the GMS87C1202.

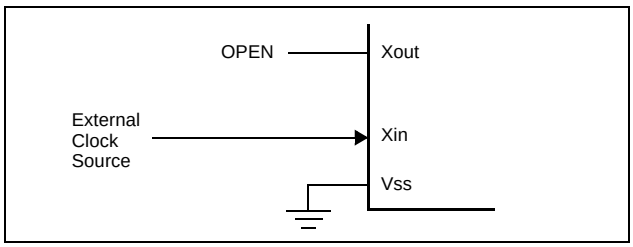


Figure 13-3 External Clock Connections

Note: *When using a system clock oscillator, carry out wiring in the broken line area in Figure 13-2 to prevent any effects from wiring capacities.*
- Minimize the wiring length.
- Do not allow wiring to intersect with other signal conductors.
- Do not allow wiring to come near changing high current.
- Set the potential of the grounding position of the oscillator capacitor to that of V_{SS}. Do not ground to any ground pattern where high current is present.
- Do not fetch signals from the oscillator.

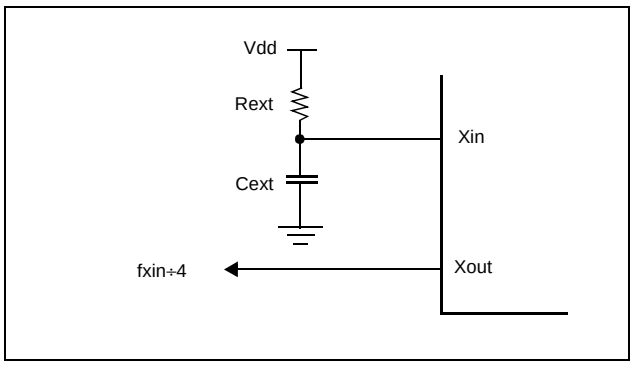


Figure 13-4 RC Oscillator Connections

The oscillator frequency, divided by 4, is output from the Xout pin, and can be used for test purpose or to synchroze other logic.

To set the RC oscillation, it should be programmed RCOPT bit to "1" to CONFIG ($0FF0_H$). (Refer to DEVICE CONFIGURATION AREA)

Note: *When using a system clock oscillator, carry out wiring in the broken line area in Figure 13-2 to prevent any effects from wiring capacities.*

- Minimize the wiring length.

- Do not allow wiring to intersect with other signal conductors.

- Do not allow wiring to come near changing high current.

- Set the potential of the grounding position of the oscillator capacitor to that of V_{SS}. Do not ground to any ground pattern where high current is present.

- Do not fetch signals from the oscillator.

14. Basic Interval Timer

The GMS87C1202 has one 8-bit Basic Interval Timer that is free-run, can not stop. Block diagram is shown in Figure 14-1 .The 8-bit Basic interval timer register (BITR) is increased every internal count pulse which is divided by prescaler. Since prescaler has divided ratio by 8 to 1024, the count rate is 1/8 to 1/1024 of the oscillator frequency. As the count overflows from FF_H to 00_H, this overflow causes to generate the Basic interval timer interrupt. The BITF is interrupt request flag of Basic interval timer.

When write "1" to bit BTCL of CKCTLR, BITR register is cleared to "0" and restart to count-up. The bit BTCL becomes "0" after one machine cycle by hardware.

If the STOP instruction executed after writing "1" to bit WAKEUP of CKCTLR, it goes into the wake-up timer mode. In this mode, all of the block is halted except the oscillator, prescaler (only fxin÷2048) and Timer0.

If the STOP instruction executed after writing "1" to bit RCWDT of CKCTLR, it goes into the internal RC oscillated watchdog timer mode. In this mode, all of the block is halted except the internal RC oscillator, Basic Interval Timer and Watchdog Timer. More detail informations are explained in Power Saving Function. The bit WDTON decides Watchdog Timer or the normal 7-bit timer

Note: *All control bits of Basic interval timer are in CKCTLR register which is located at same address of BITR (address EC_H). Address EC_H is read as BITR, written to CKCTLR. Therefore, the CKCTLR can not be accessed by bit manipulation instruction.*

.

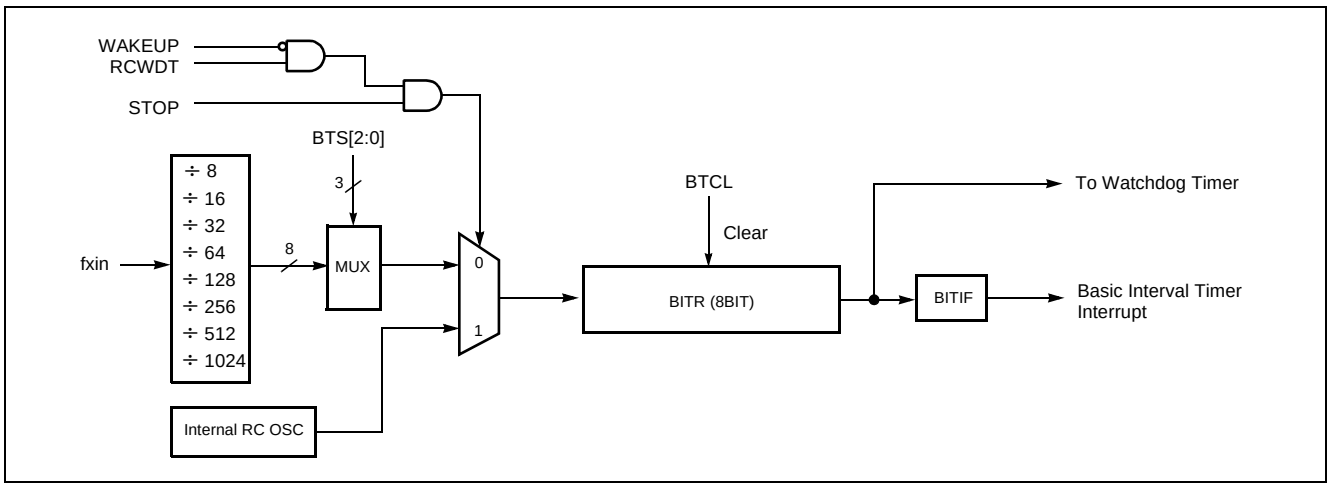


Figure 14-1 Block Diagram of Basic Interval Timer

Clock Control Register

CKCTLR

-	WAKEUP	RCWDT	WDTON	BTCL	BTS2	BTS1	BTS0

ADDRESS : ECH
RESET VALUE : -0010111
Bit Manipulation Not Available

Basic Interval Timer Clock Selection

Symbol	Function Description
WAKEUP	1: Enables Wake-up Timer 0: Disables Wake-up Timer
RCWDT	1: Enables Internal RC Watchdog Timer 0: Disables Internal RC Watchdog Time
WDTON	1: Enables Watchdog Timer 0: Operates as a 7-bit Timer
BTCL	1: BITR is cleared and BTCL becomes "0" automatically after one machine cycle, and BITR continue to count-up

000 : fxin ÷ 8
001 : fxin ÷ 16
010 : fxin ÷ 32
011 : fxin ÷ 64
100 : fxin ÷ 128
101 : fxin ÷ 256
110 : fxin ÷ 512
111 : fxin ÷ 1024

Figure 14-2 CKCTLR : Clock Control Register

15. TIMER / COUNTER

The GMS87C1202 has two Timer/Counter registers. Each module can generate an interrupt to indicate that an event has occurred (i.e. timer match).

Timer 0 and Timer 1 can be used either the two 8-bit Timer/Counter or one 16-bit Timer/Counter by combining them.

In the "timer" function, the register is increased every internal clock input. Thus, one can think of it as counting internal clock input. Since a least clock consists of 2 and most clock consists of 2048 oscillator periods, the count rate is 1/2 to 1/2048 of the oscillator frequency in Timer0. And Timer1 can use the same clock source too. In addition, Timer1 has more fast clock source (1/1 to 1/8).

In the "counter" function, the register is increased in response to a 0-to-1 (rising edge) transition at its corresponding external input pin, EC0.

And in the "capture" function, the register is increased in response external interrupt same with timer function. When external interrupt edge input, the count register is captured into capture data register CDRx.

Timer1 is shared with "PWM" function and "Compare output" function

It has seven operating modes: "8-bit timer/counter", "16-bit timer/counter", "8-bit capture", "16-bit capture", "8-bit compare output", "16-bit compare output" and "10-bit PWM" which are selected by bit in Timer mode register TM0 and TM1 as shown in Figure 15-1 and Table 15-1.

Timer 0 Mode Register

TM0

-	-	CAP0	T0CK2	T0CK1	T0CK0	T0CN	T0ST

ADDRESS : D0H
RESET VALUE : --000000

CAP0 — Capture mode selection bit.
0 : Disables Capture
1 : Enables Capture

T0CK[2:0] — Input clock selection

000 : fxin ÷ 2	100 : fxin ÷ 128
001 : fxin ÷ 4	101 : fxin ÷ 512
010 : fxin ÷ 8	110 : fxin ÷ 2048
011 : fxin ÷ 32	111 : External Event (EC0)

T0CN — Continue control bit
0 : Stop counting
1 : Start counting continuously

T0ST — Start control bit
0 : Stop counting
1 : Counter register is cleared and started again

Timer 1 Mode Register

TM1

POL	16BIT	PWME	CAP1	T1CK1	T1CK0	T1CN	T1ST

ADDRESS : D2H
RESET VALUE : 00000000

POL — PWM Output Polarity
0 : Duty active low
1 : Duty active high

16BIT — 16-bit mode selection
0 : 8-bit mode
1 : 16-bit mode

PWME — PWM enable bit
0 : Disables PWM
1 : Enables PWM

CAP1 — Capture mode selection bit.
0 : Disables Capture
1 : Enables Capture

T1CK[2:0] — Input clock selection

00 : fxin	10 : fxin ÷ 8
01 : fxin ÷ 2	11 : using the Timer 0 clock

T1CN — Continue control bit
0 : Stop counting
1 : Start counting continuously

T1ST — Start control bit
0 : Stop counting
1 : Counter register is cleared and started again

Figure 15-1 Timer 0 and Timer 1 Mode Register

16BIT	CAP0	CAP1	PWME	T0CK[2:0]	T1CK[1:0]	PWMO[1]	TIMER 0	TIMER1
0	0	0	0	XXX	XX	0	8-bit Timer	8-bit Timer
0	0	1	0	111	XX	0	8-bit Event Counter	8-bit Capture
0	1	0	0	XXX	XX	1	8-bit Capture	8-bit Compare output
0	0	0	1	XXX	XX	1	8-bit Timer/Counter	10-bit PWM
1	0	0	0	XXX	11	0	16-bit Timer	
1	0	0	0	111	11	0	16-bit Event Counter	
1	1	X[2]	0	XXX	11	0	16-bit Capture	
1	0	0	0	XXX	11	1	16-bit Compare output	

Table 15-1 Operating Modes of Timer 0 and Timer 1

1. *This bit is the bit4 of RB Function register(RBFUNC).*
2. *X : The value is “0” or “1” corresponding your operation.*

15.1 8-bit Timer/Counter Mode

The GMS87C1202 has two 8-bit Timer/Counters, Timer 0 and Timer 1, as shown in Figure 15-2 .

The "timer" or "counter" function is selected by mode registers TM0, TM1 as shown in Figure 15-1 and Table 15-1 . To use as an 8-bit timer/counter mode, bit CAP0 of TM0 is cleared to "0" and bits 16BIT of TM1 should be cleared to “0”(Table 15-1).

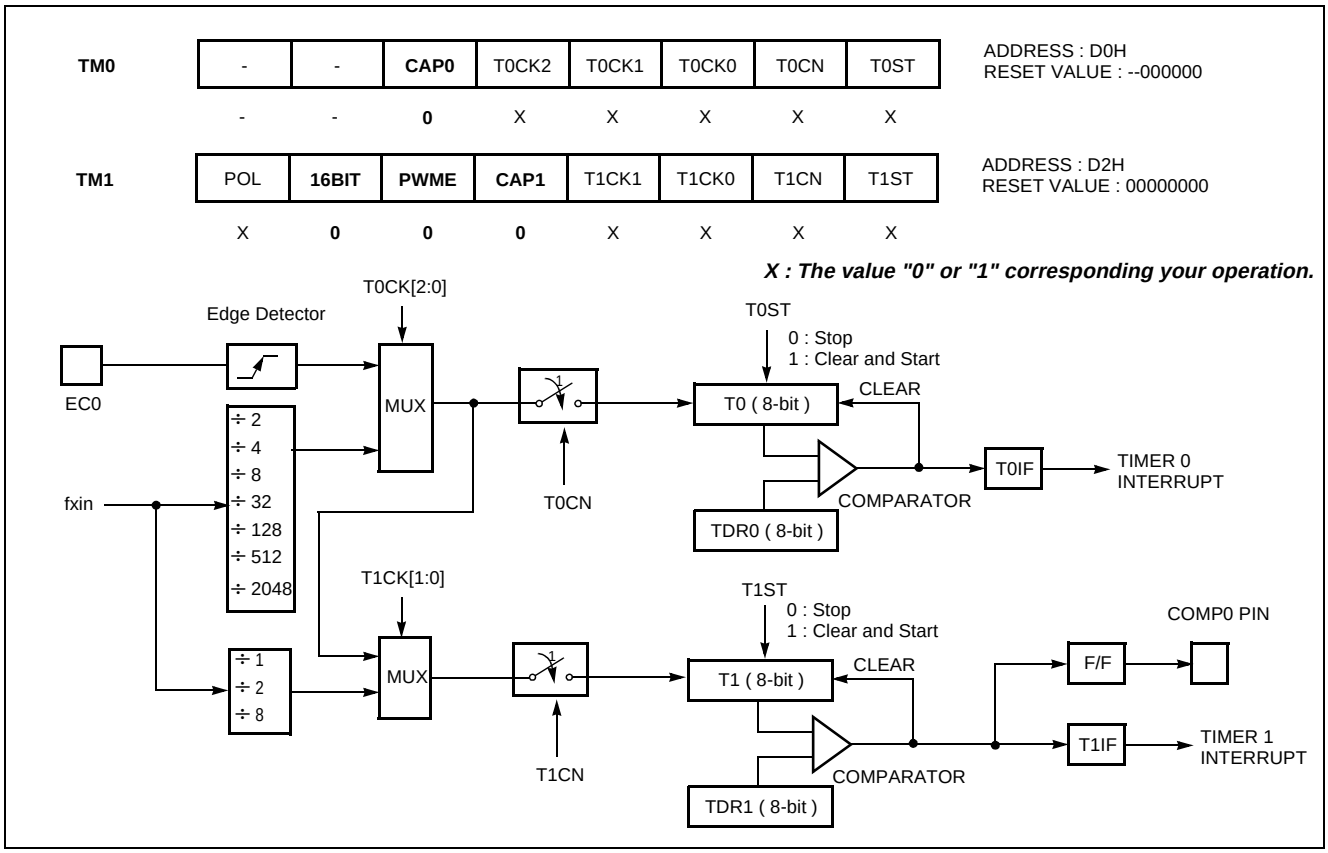


Figure 15-2 8-bit Timer / Counter Mode

These timers have each 8-bit count register and data register. The count register is increased by every internal or external clock input. The internal clock has a prescaler divide ratio option of 2, 4, 8, 32,128, 512, 2048 (selected by control bits T0CK2, T0CK1 and T0CK0 of register TM0) and 1, 2, 8 (selected by control bits T1CK1 and T1CK0 of register TM1). In the Timer 0, timer register T0 increases from 00_H until it matches TDR0 and then reset to 00_H. The match output of Timer 0 generates Timer 0 interrupt (latched in T0F bit). As TDRx and Tx register are in same address, when reading it as a Tx, written to TDRx.

In counter function, the counter is increased every 0-to 1 (rising edge) transition of EC0 pin. In order to use counter function, the bit RA0 of the RA Direction Register RAIO is set to "0". The Timer 0 can be used as a counter by pin EC0 input, but Timer 1 can not.

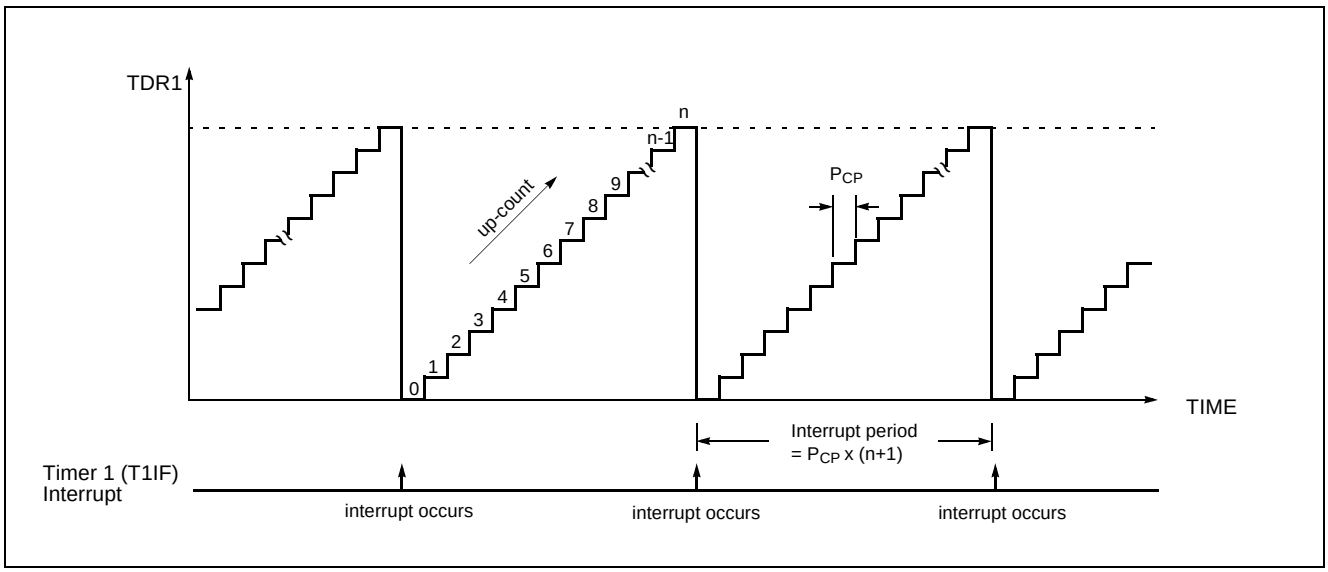


Figure 15-3 Counting Example of Timer Data Registers

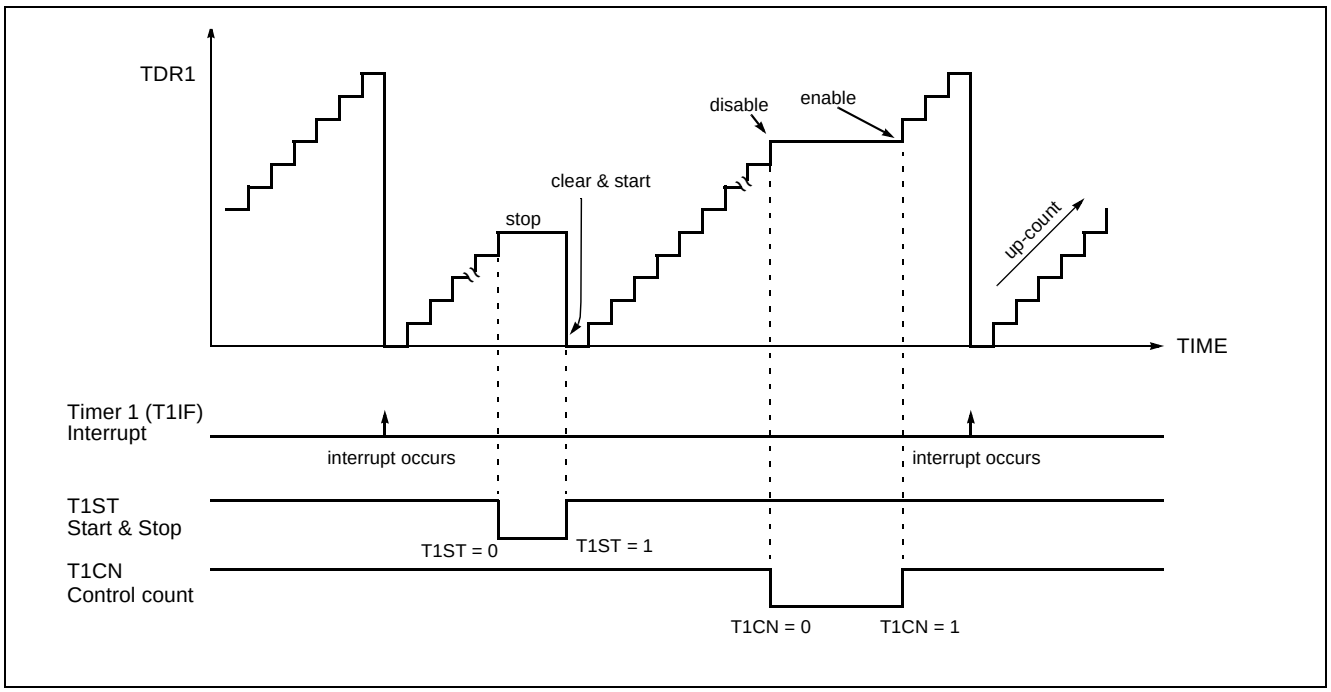


Figure 15-4 Timer Count Operation

15.2 16-bit Timer/Counter Mode

The Timer register is being run with 16 bits. A 16-bit timer/counter register T0, T1 are increased from 0000_H until it matches TDR0, TDR1 and then resets to 0000_H. The match output generates Timer 0 interrupt not Timer 1 interrupt.

The clock source of the Timer 0 is selected either internal or external clock by bit T0CK2, T0CK1 and T0SL0.

In 16-bit mode, the bits T1CK1,T1CK0 and 16BIT of TM1 should be set to "1" respectively.

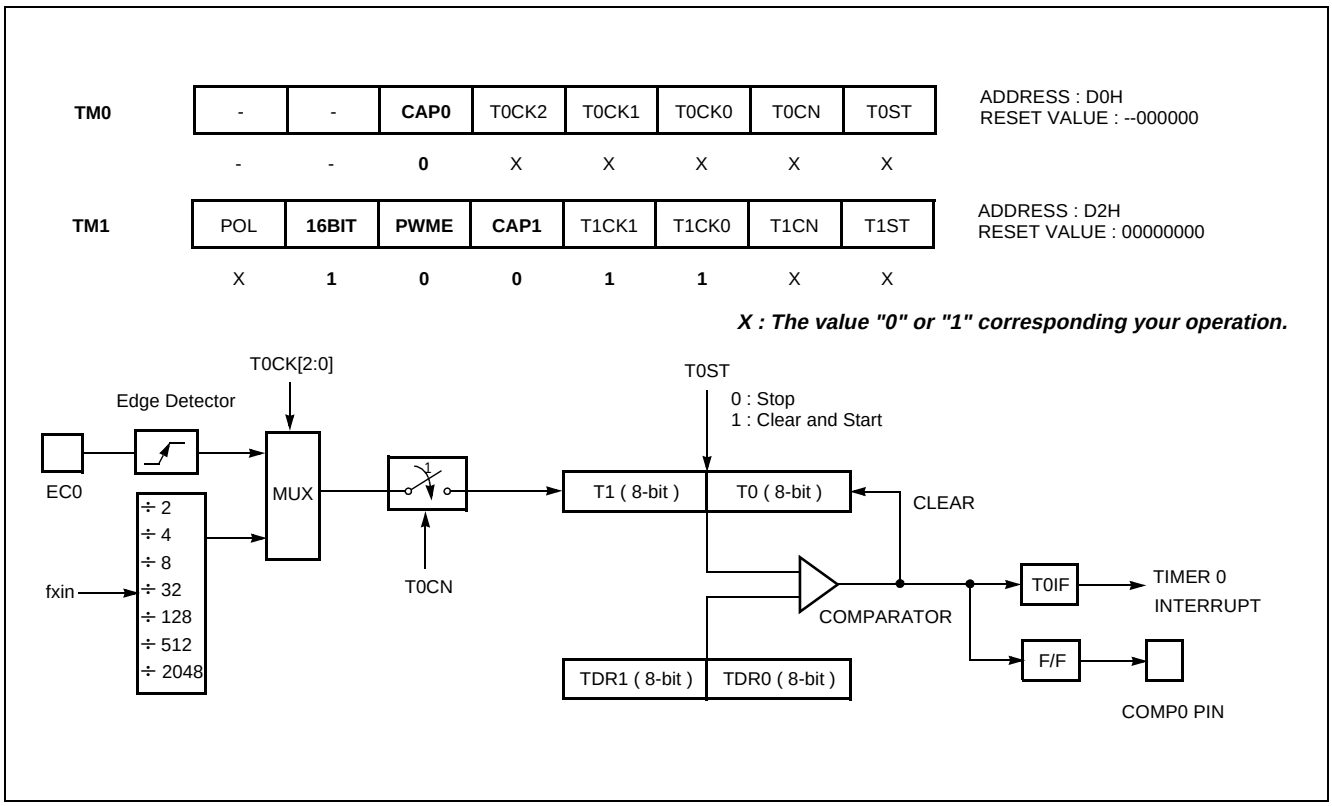


Figure 15-5 16-bit Timer / Counter Mode

15.3 8-bit Compare Output (16-bit)

The GMS87C1201 and GMS87C1202 has a function of Timer Compare Output. To pulse out, the timer match can goes to port pin(COMP0) as shown in Figure 15-2 and Figure 15-5 . Thus, pulse out is generated by the timer match. These operation is implemented to pin, RB4/COMP0/PWM.

This pin output the signal having a 50 : 50 duty square wave, and output frequency is same as below equation.

$$f_{COMP} = \frac{\text{Oscillation Frequency}}{2 \times \text{Prescaler Value} \times (TDR + 1)}$$

In this mode, the bit PWMO of RB function register (RBFUNC) should be set to "1", and the bit PWME of timer1 mode register (TM1) should be set to "0".

In addition, 16-bit Compare output mode is also available.

15.4 8-bit Capture Mode

The Timer 0 capture mode is set by bit CAP0 of timer mode register TM0 (bit CAP1 of timer mode register TM1 for Timer 1) as shown in Figure 15-6.

As mentioned above, not only Timer 0 but Timer 1 can also be used as a capture mode.

The Timer/Counter register is increased in response internal or external input. This counting function is same with normal timer mode, and Timer interrupt is generated when timer register T0 (T1) increases and matches TDR0 (TDR1).

This timer interrupt in capture mode is very useful when the pulse width of captured signal is more wider than the maximum period of Timer.

For example, in Figure 15-8 , the pulse width of captured signal is wider than the timer data value (FF_H) over 2 times. When external interrupt is occured, the captured value (13_H) is more little than wanted value. It can be obtained correct value by counting the number of timer overflow occurence.

Timer/Counter still does the above, but with the added feature that a edge transition at external input INTx pin causes the current value in the Timer x register (T0,T1), to be captured into registers CDRx (CDR0, CDR1), respectively. After captured, Timer x register is cleared and restarts by hardware.

It has three transition modes: "falling edge", "rising edge", "both edge" which are selected by interrupt edge selection register IEDS (Refer to External interrupt section). In addition, the transition at INTx pin generate an interrupt.

Note: *The CDRx, TDRx and Tx are in same address. In the capture mode, reading operation is read the CDRx, not Tx because path is opened to the CDRx, and TDRx is only for writing operation.*

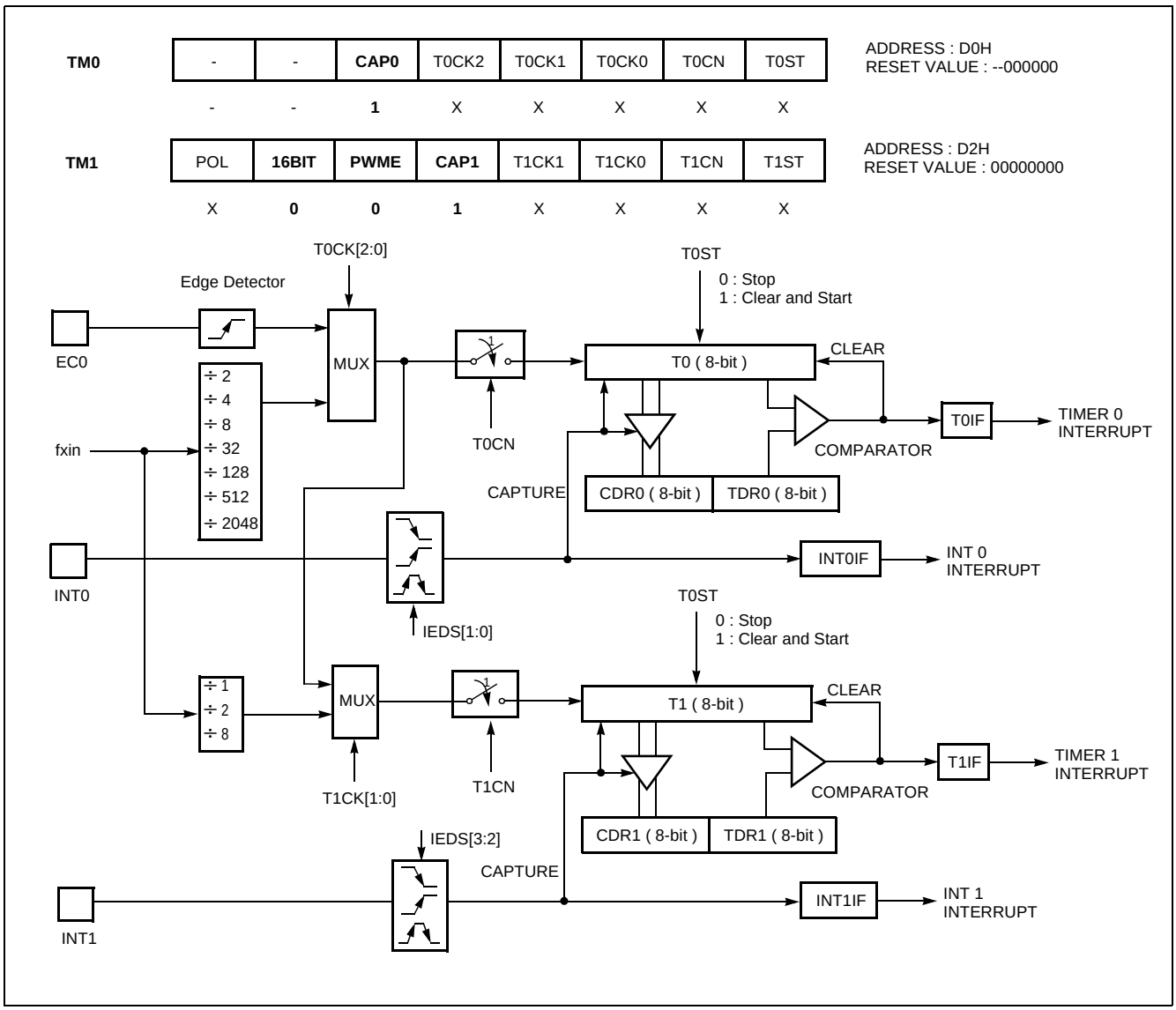


Figure 15-6 8-bit Capture Mode

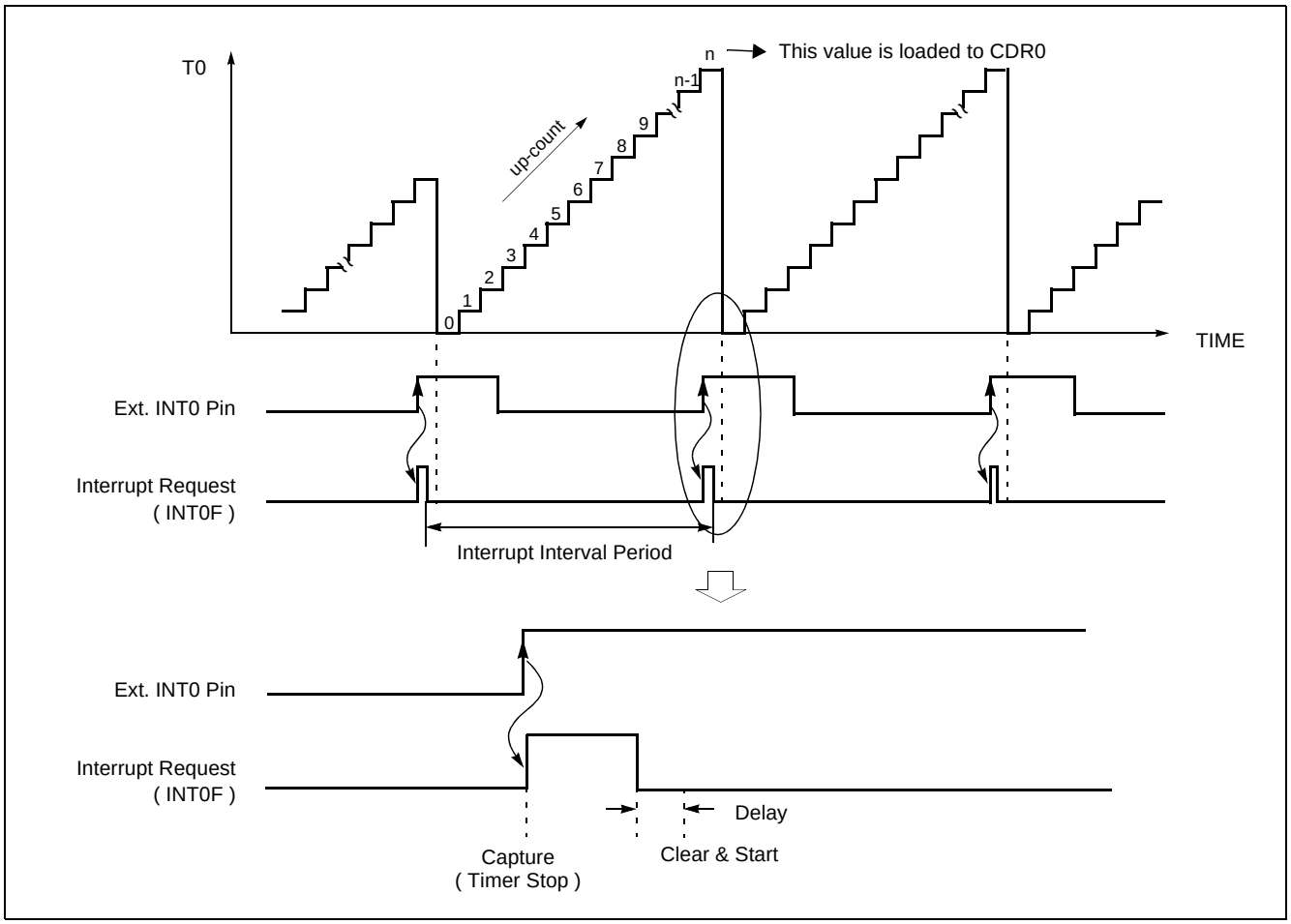


Figure 15-7 Input Capture Operation

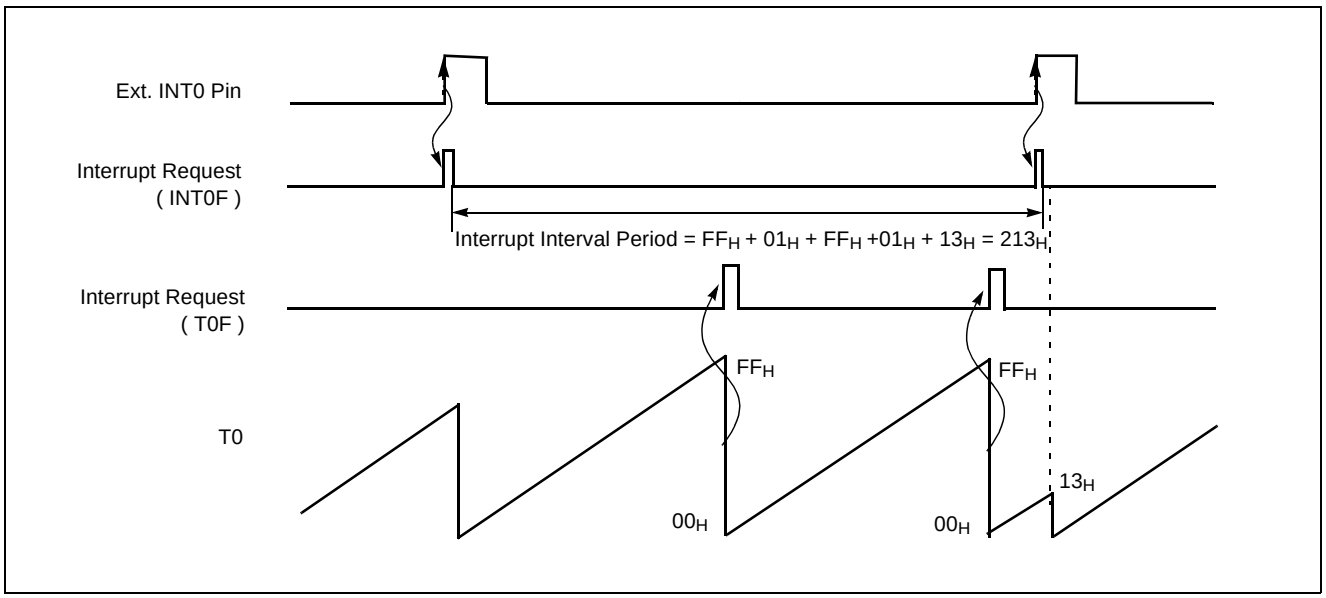


Figure 15-8 Excess Timer Overflow in Capture Mode

15.5 16-bit Capture Mode

16-bit capture mode is the same as 8-bit capture, except that the Timer register is being run will 16 bits.

The clock source of the Timer 0 is selected either internal or external clock by bit T0CK2, T0CK1 and T0CK0.

In 16-bit mode, the bits T1CK1,T1CK0 and 16BIT of TM1 should be set to "1" respectively.

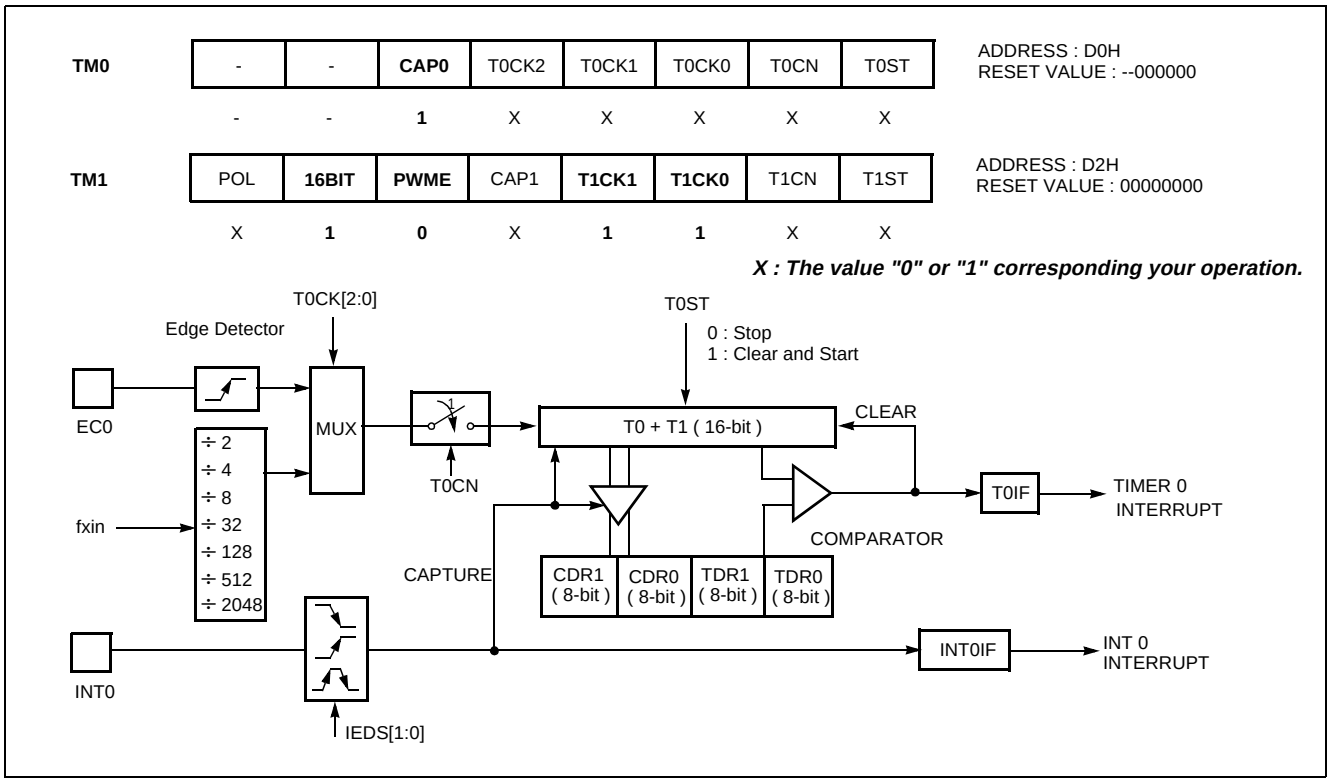


Figure 15-9 16-bit Capture Mode

15.6 PWM Mode

The GMS87C1202 has a two high speed PWM (Pulse Width Modulation) functions which shared with Timer1.

In PWM mode, pin RB4/COMP0/PWM0 outputs up to a 10-bit resolution PWM output. This pin should be defined as a PWM output by setting "1" bit PWMO in RBFUNC register.

The period of the PWM output is determined by the T1PPR (PWM0 Period Register) and PWM0HR[3:2] (bit3,2 of PWM0 High Register) and the duty of the PWM output is determined by the T1PDR (PWM0 Duty Register) and PWM0HR[1:0] (bit1,0 of PWM0 High Register).

The user writes the lower 8-bit period value to the T1PPR and the higher 2-bit period value to the PWM0HR[3:2]. And writes duty value to the T1PDR and the PWM0HR[1:0] same way.

The T1PDR is configured as a double buffering for glitch-less PWM output. In Figure 15-10 , the duty data is transfered from the master to the slave when the period data matched to the counted value. (i.e. at the beginning of next duty cycle)

PWM Period = [PWM0HR[3:2]T1PPR] X Source Clock

PWM Duty = [PWM0HR[1:0]T1PDR] X Source Clock

The relation of frequency and resolution is in inverse proportion. Table 15-2 shows the relation of PWM frequency vs. resolution.

If it needed more higher frequency of PWM, it should be reduced resolution.

Resolution	Frequency		
	T1CK[1:0] = 00(125nS)	T1CK[1:0] = 01(250nS)	T1CK[1:0] = 10(1uS)
10-bit	7.8KHz	3.9KHz	0.98KHZ
9-bit	15.6KHz	7.8KHz	1.95KHz
8-bit	31.2KHz	15.6KHz	3.90KHz
7-bit	62.5KHz	31.2KHz	7.81KHz

Table 15-2 PWM Frequency vs. Resolution at 8MHz

The bit POL of TM1 decides the polarity of duty cycle.

If the duty value is set same to the period value, the PWM output is determined by the bit POL (1: High, 0: Low). And if the duty value is set to "00_H", the PWM output is determined by the bit POL (1: Low, 0: High).

It can be changed duty value when the PWM output. Howerver the changed duty value is output after the current period is over. And it can be maintained the duty value at present output when changed only period value shown as Figure 15-12 . As it were, the absolute duty time is not changed in varying frequency. But the changed period value must greater than the duty value

Note: *At PWM output start command, one first pulse would be output abnormally. Because if user writes register values while timer is in operaiton, these register could be set with certain values at first. To prevent this operation, user must stop PWM timer clock and then set the duty and the period register values.*

.

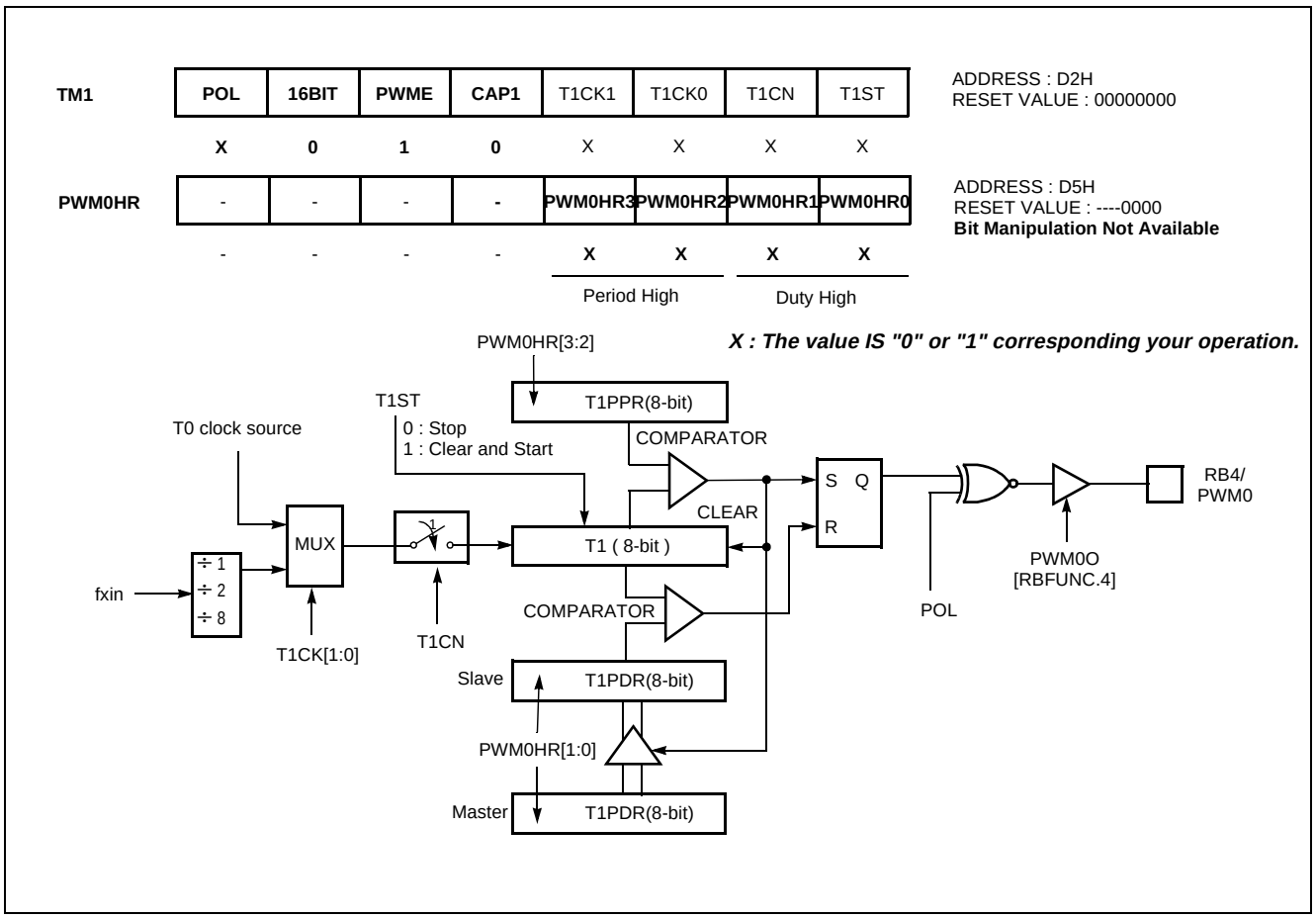


Figure 15-10 PWM Mode

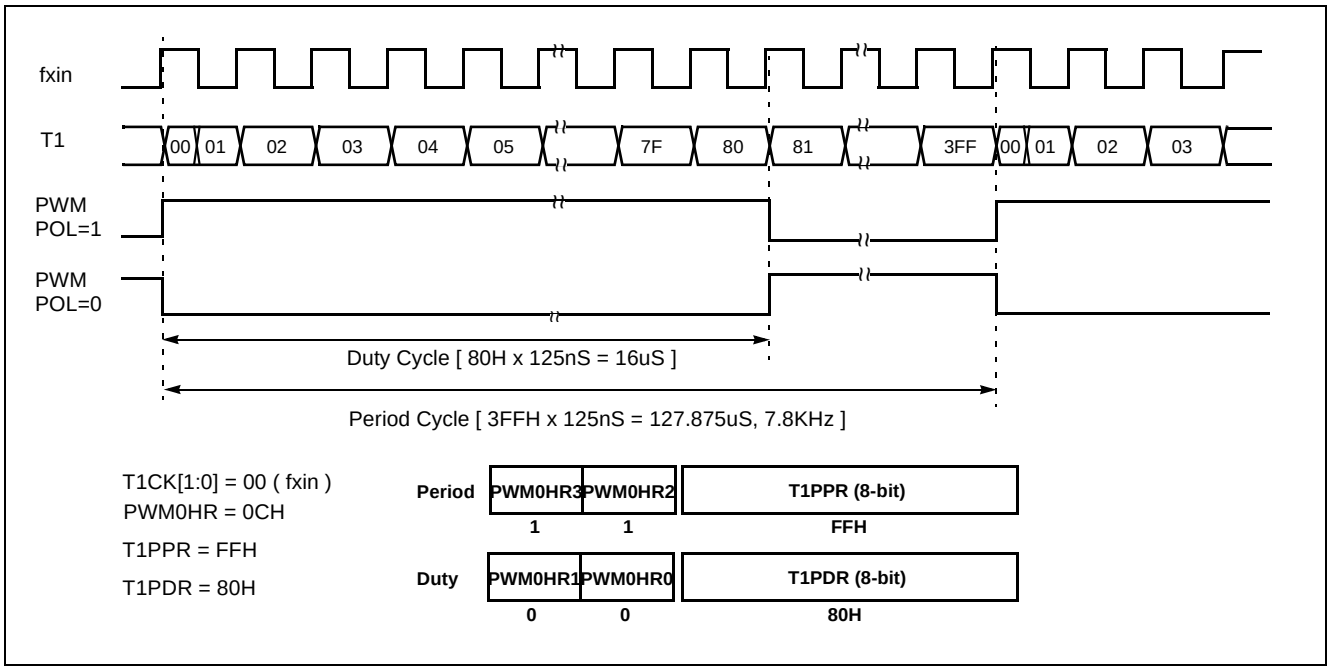


Figure 15-11 Example of PWM at 8MHz

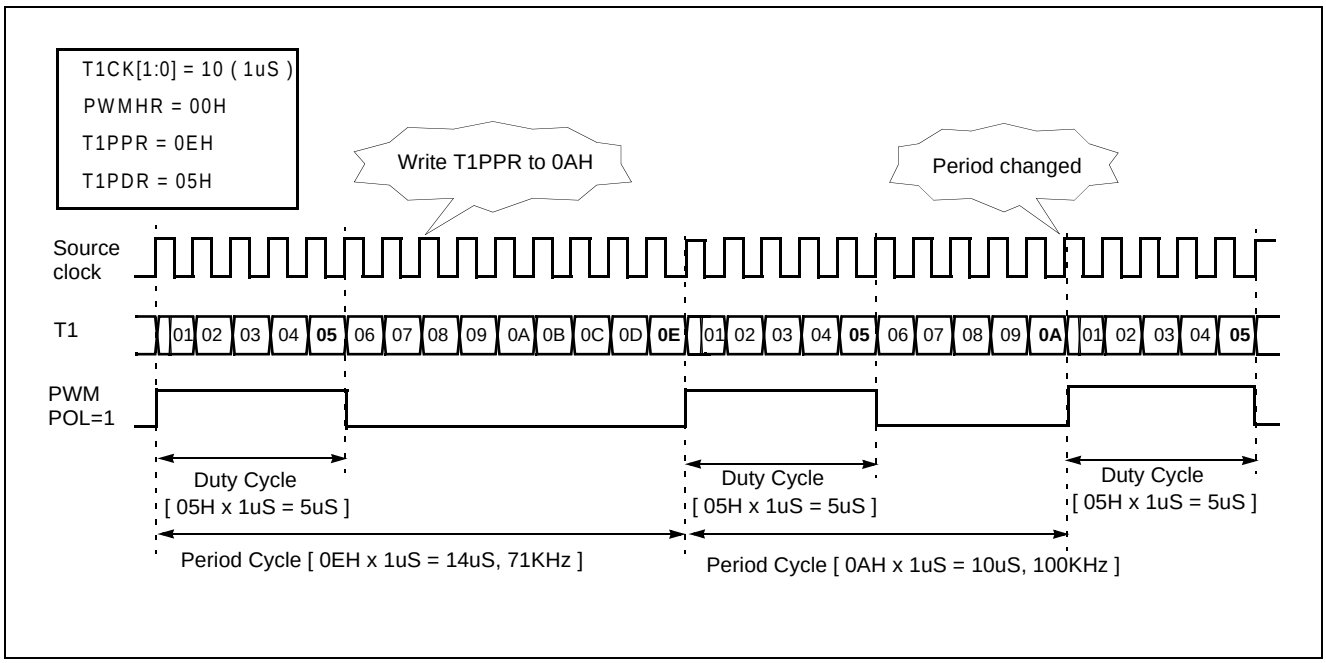


Figure 15-12 Example of Changing the Period in Absolute Duty Cycle (@8MHz)

16. Buzzer Output Function

The buzzer driver consists of 6-bit binary counter, the buzzer register BUR and the clock selector. It generates square-wave which is very wide range frequency (480 Hz~250 KHz at fxin = 4 MHz) by user programmable counter.

Pin RB1 is assigned for output port of Buzzer driver by setting the bit BUZO of RBFUNC to "1".

The 6-bit buzzer counter is cleared and start the counting by writing signal to the register BUR. It is increased from 00H until it matches 6-bit register BUR.

Also, it is cleared by counter overflow and count up to output the square wave pulse of duty 50%.

The bit 0 to 5 of BUR determines output frequency for buzzer driving. Frequency calculation is following as shown below.

$$f_{BUZ}(Hz) = \frac{\text{Oscillator Frequency}}{2 \times \text{Prescaler Ratio} \times (BUR + 1)}$$

The bits BUCK1, BUCK0 of BUR selects the source clock from prescaler output.

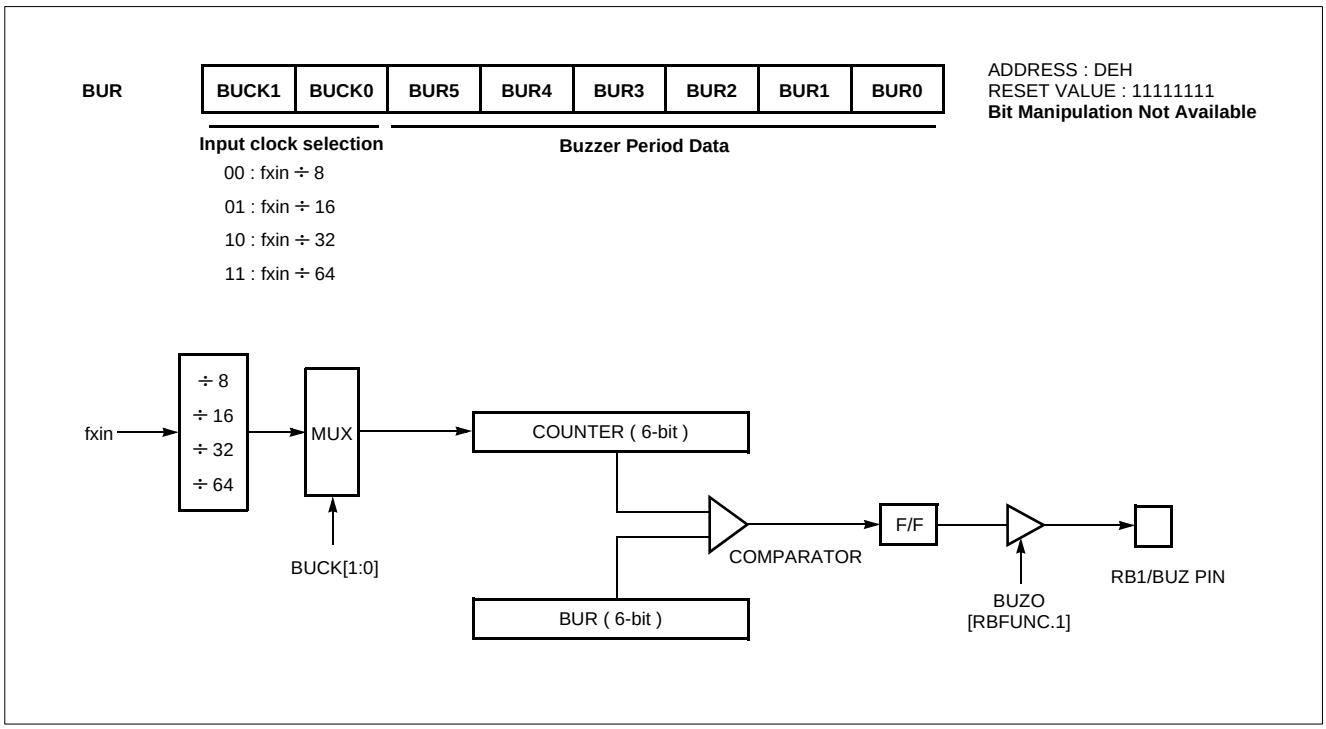


Figure 16-1 Buzzer Driver

17. ANALOG TO DIGITAL CONVERTER

The analog-to-digital converter (A/D) allows conversion of an analog input signal to a corresponding 8-bit digital value. The A/D module has eight analog inputs, which are multiplexed into one sample and hold. The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

The analog reference voltage is selected to V_{DD} or AVref by setting of the bit AVREFS in RBFUNC register. If external analog reference AVref is selected, the bit ANSEL0 should not be set to "1", because this pin is used to an analog reference of A/D converter.

The A/D module has two registers which are the control register ADCM and A/D result register ADCR. The ADCM register, shown in Figure 17-2 , controls the operation of the A/D converter module. The port pins can be configured as analog inputs or digital I/O.

To use analog inputs, each port is assigned analog input port by setting the bit ANSEL[7:0] in RAFUNC register. And selected the corresponding channel to be converted by setting ADS[2:0].

The processing of conversion is start when the start bit ADST is set to "1". After one cycle, it is cleared by hardware. The register ADCR contains the results of the A/D conversion. When the conversion is completed, the result is loaded into the ADCR, the A/D conversion status bit ADSF is set to "1", and the A/D interrupt flag ADIF is set. The block diagram of the A/D module is shown in Figure 17-1. The A/D status bit ADSF is set automatically when A/D conversion is completed, cleared when A/D conversion is in process. The conversion time takes maximum 10 uS (at fxin=8 MHz).

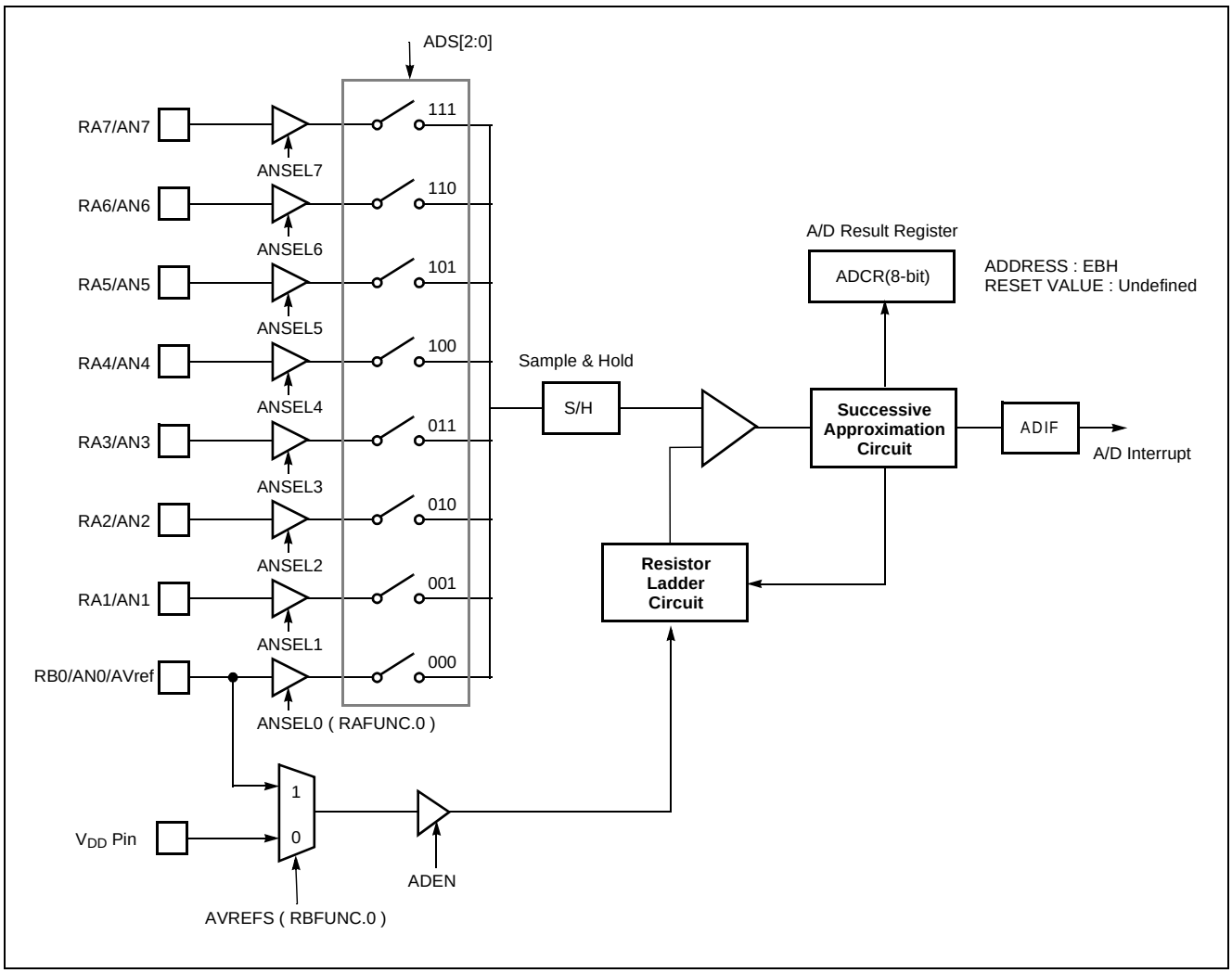


Figure 17-1 A/D Converter Block Diagram

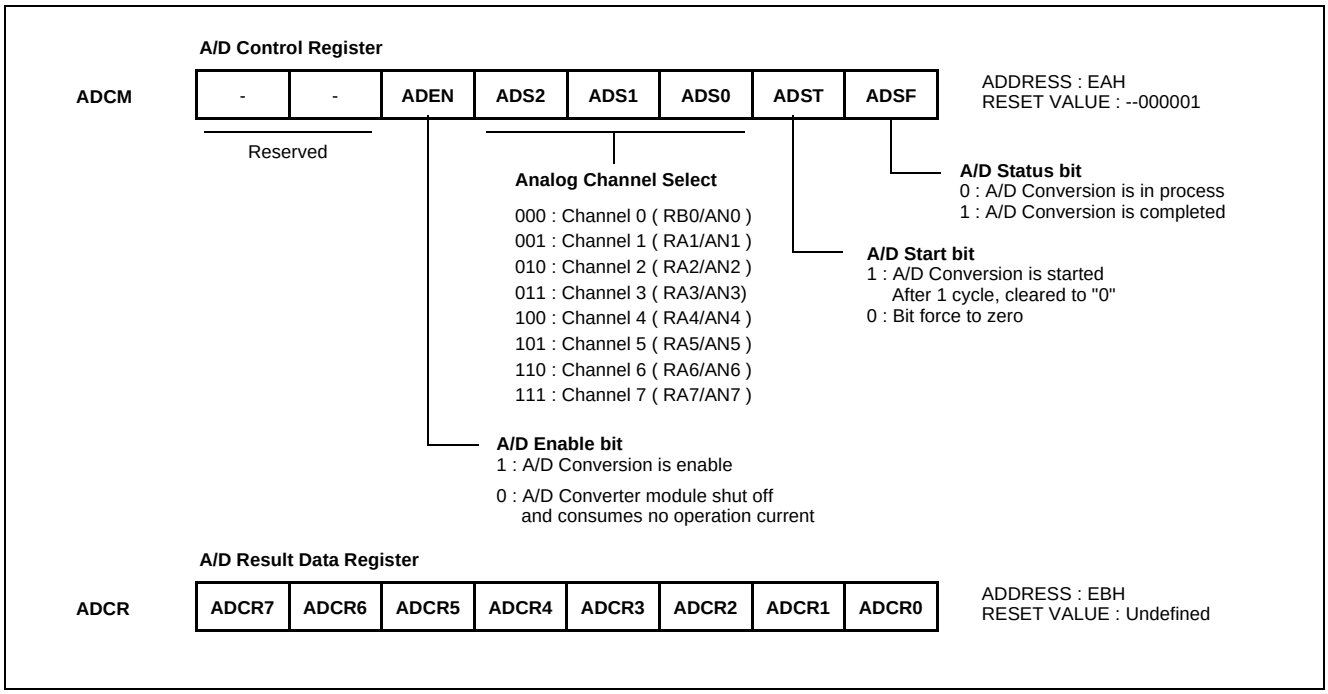


Figure 17-2 A/D Converter Registers

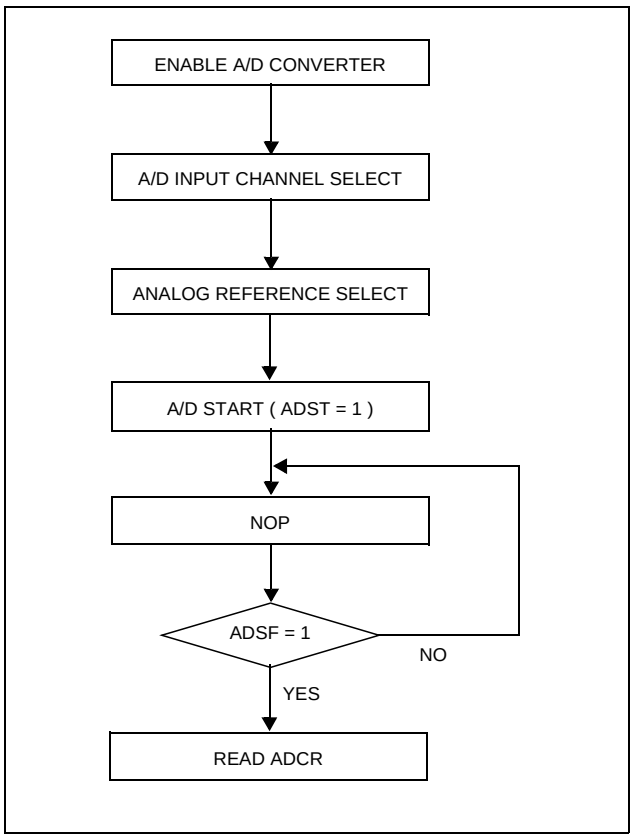


Figure 17-3 A/D Converter Operation Flow

A/D Converter Cautions

(1) Input range of AN0 to AN7

The input voltages of AN0 to AN7 should be within the specification range. In particular, if a voltage above VDD (or AVref) or below Vss is input (even if within the absolute maximum rating range), the conversion value for that channel can not be indeterminate. The conversion values of the other channels may also be affected.

(2) Noise countermeasures

In order to maintain 8-bit resolution, attention must be paid to noise on pins AVref(or VDD)and AN0 to AN7. Since the effect increases in proportion to the output impedance of the analog input source, it is recommended that a capacitor be connected externally as shown in Figure 17-4 in order to reduce noise

.

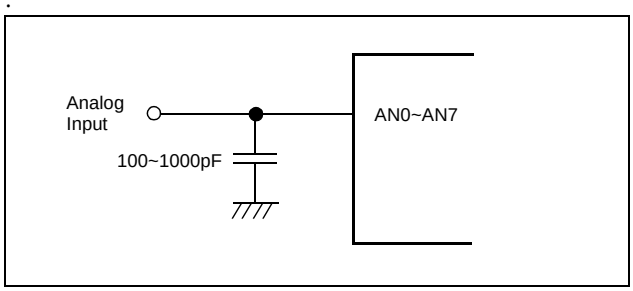


Figure 17-4 Analog Input Pin Connecting Capacitor

(3) Pins AN0/RB0 and AN1/RA1 to AN7/RA7

The analog input pins AN0 to AN7 also function as input/output port (PORT RA and RB0) pins. When A/D conversion is performed with any of pins AN0 to AN7 selected, be sure not to execute a PORT input instruction while conversion is in progress, as this may reduce the conversion resolution.

Also, if digital pulses are applied to a pin adjacent to the pin in the process of A/D conversion, the expected A/D conversion value may not be obtainable due to coupling noise. Therefore, avoid applying pulses to pins adjacent to the pin undergoing A/D conversion.

(4) AV_{REF} pin input impedance

A series resistor string of approximately 10KΩ is connected between the AV_{REF}pin and the V_{SS} pin.

Therefore, if the output impedance of the reference voltage source is high, this will result in parallel connection to the series resistor string between the AV_{REF} pin and the V_{SS} pin, and there will be a large reference voltage error.

18. INTERRUPTS

The GMS87C1202 interrupt circuits consist of Interrupt enable register (IENH, IENL), Interrupt request flags of IRQH, IRQL, Interrupt Edge Selection Register (IEDS), priority circuit and Master enable flag("I" flag of PSW). The configuration of interrupt circuit is shown in Figure 18-1 and Interrupt priority is shown in Table 18-1 .

The External Interrupts INT0 and INT1 can each be transition-activated (1-to-0, 0-to-1 and both transiton).

The flags that actually generate these interrupts are bit INT0IFand INT1IF in Register IRQH. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to only if the interrupt was transition-activated.

The Timer 0 and Timer 1 Interrupts are generated by T0IF, T1IF, which are set by a match in their respective timer/ counter register. The AD converter Interrupt is generated by ADIF which is set by finishing the analog to digital conversion. The Watch dog timer Interrupt is generated by WDTIF which set by a match in Watch dog timer register (when the bit WDTON is set to "0"). The Basic Interval Timer Interrupt is generated by BITIF which is set by a overflowing of the Basic Interval Timer Register(BITR).

.

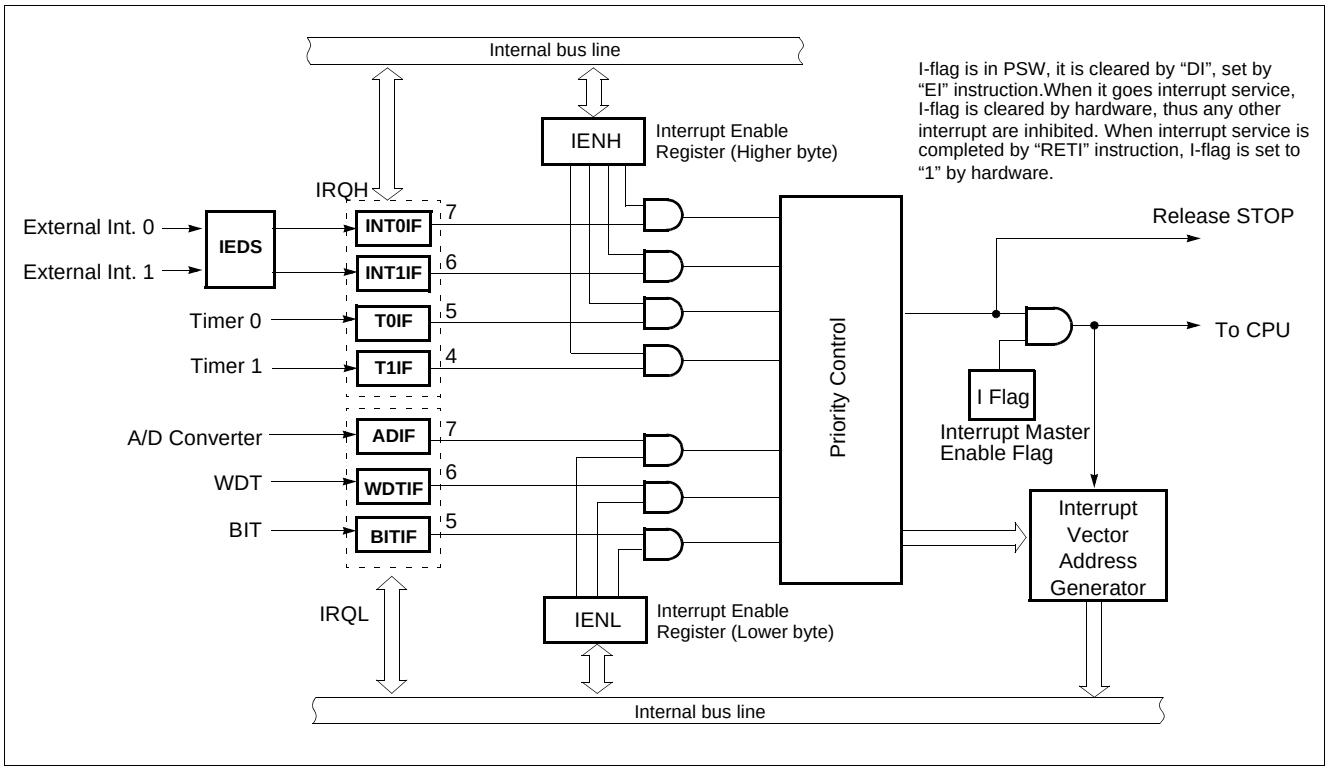


Figure 18-1 Block Diagram of Interrupt Function

The interrupts are controlled by the interrupt master enable flag I-flag (bit 2 of PSW), the interrupt enable register (IENH, IENL) and the interrupt request flags (in IRQH, IRQL) except Power-on reset and software BRK interrupt.

Interrupt enable registers are shown in Figure 18-2 . These registers are composed of interrupt enable flags of each interrupt source, these flags determines whether an interrupt will be accepted or not. When enable flag is "0", a corresponding interrupt source is prohibited. Note that PSW contains also a master enable bit, I-flag, which disables all interrupts at once.

Reset/Interrupt	Symbol	Priority	Vector Addr.
Hardware Reset	RESET	-	$FFFE_H$
External Interrupt 0	INT0	1	$FFFA_H$
External Interrupt 1	INT1	2	$FFF8_H$
Timer 0	Timer 0	3	$FFF6_H$
Timer 1	Timer 1	4	$FFF4_H$
A/D Converter	A/D C	5	$FFEA_H$
Watch Dog Timer	WDT	6	$FFE8_H$
Basic Interval Timer	BIT	7	$FFE6_H$

Table 18-1 Interrupt Priority

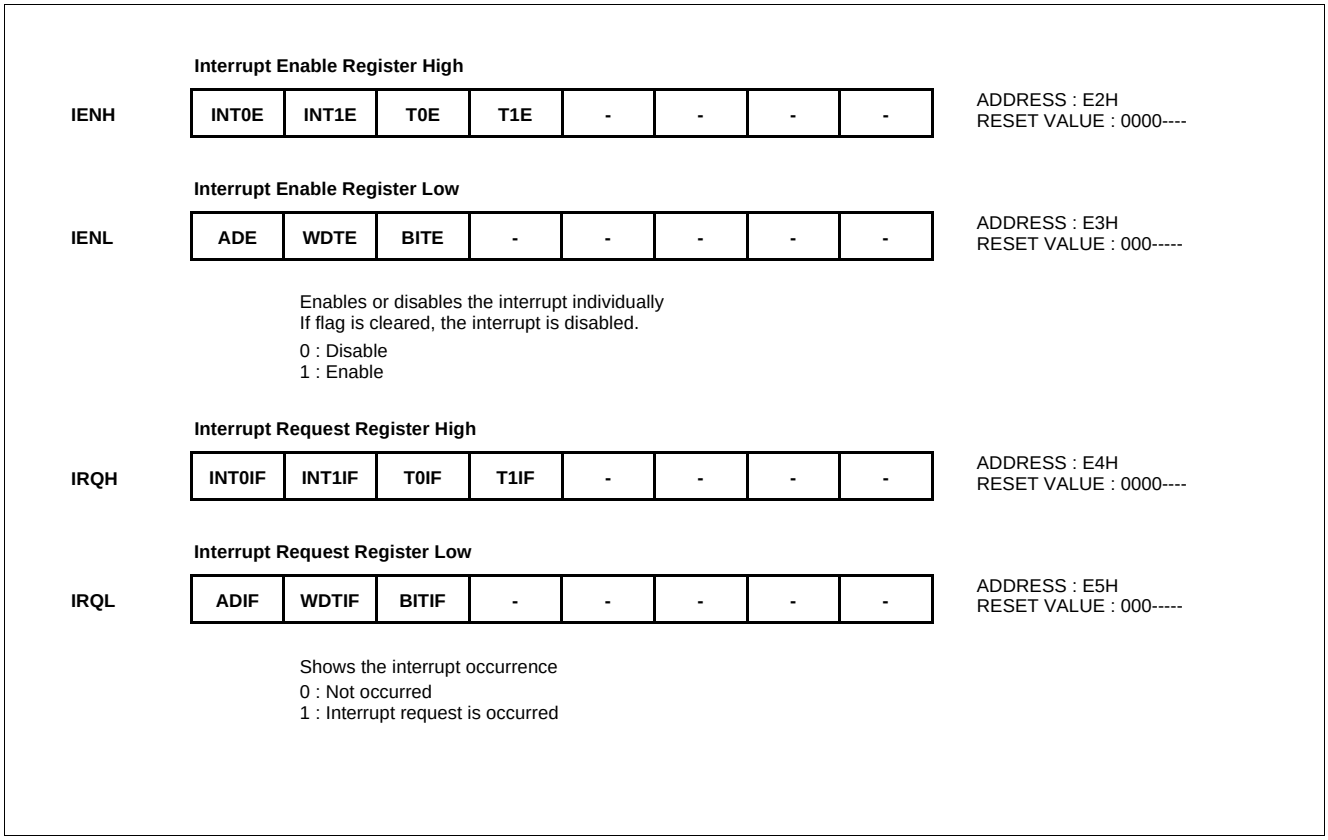


Figure 18-2 Interrupt Enable Registers and Interrupt Request Registers

When an interrupt is occured, the I-flag is cleared and disable any further interrupt, the return address and PSW are pushed into the stack and the PC is vectored to. Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt request flag bits. The interrupt request flag bit(s) must be cleared by software before re-enabling interrupts to avoid recursive interrupts. The Interrupt Request flags are able to be read and written.

18.1 Interrupt Sequence

An interrupt request is held until the interrupt is accepted or the interrupt latch is cleared to "0" by a reset or an instruction. Interrupt acceptance sequence requires 8 f_{OSC} (2 µs at f_{XIN}=4MHz) after the completion of the current instruction execution. The interrupt service task is terminated upon execution of an interrupt return instruction [RETI].

Interrupt acceptance

1. The interrupt master enable flag (I-flag) is cleared to "0" to temporarily disable the acceptance of any following maskable interrupts. When a non-maskable interrupt is accepted, the acceptance of any following interrupts is temporarily disabled.
2. Interrupt request flag for the interrupt source accepted is cleared to "0".
3. The contents of the program counter (return address) and the program status word are saved (pushed) onto the stack area. The stack pointer decreases 3 times.
4. The entry address of the interrupt service program is read from the vector table address and the entry address is loaded to the program counter.
5. The instruction stored at the entry address of the interrupt service program is executed.

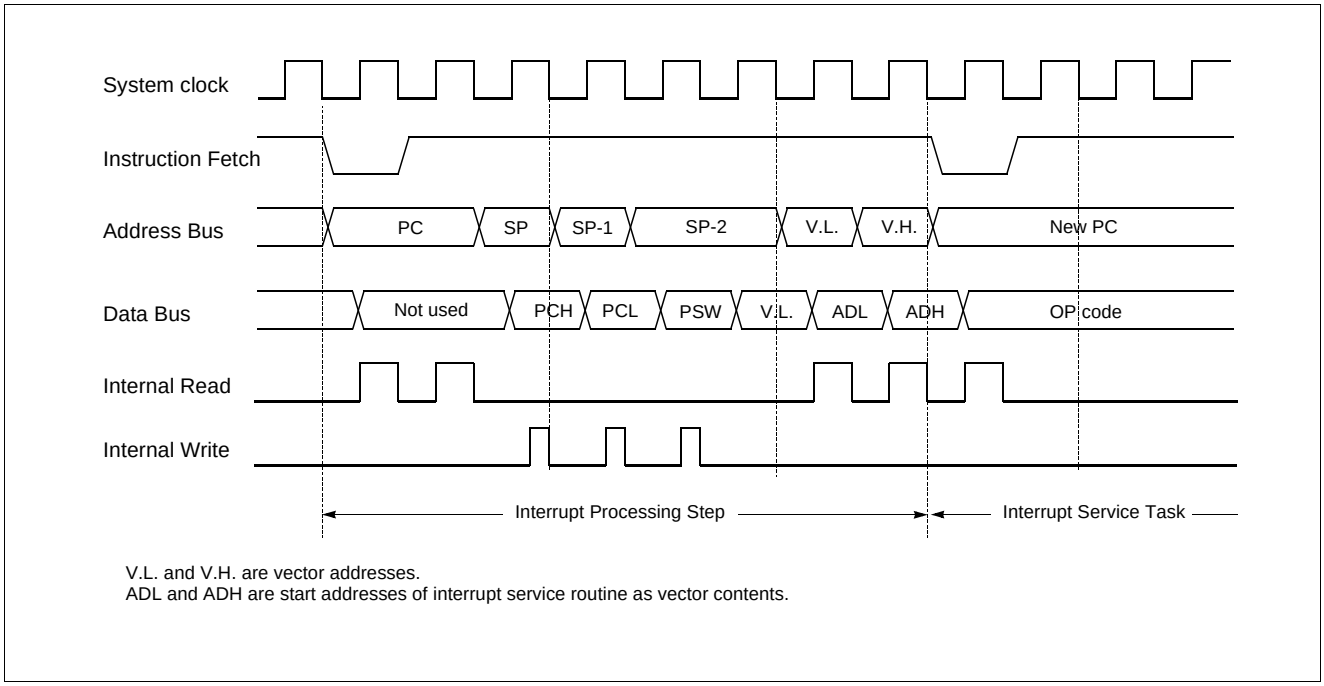
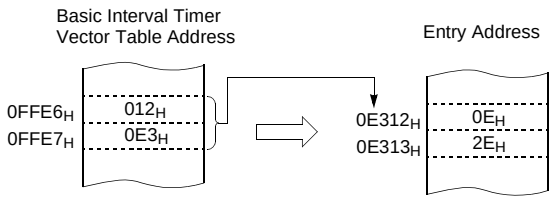


Figure 18-3 Timing chart of Interrupt Acceptance and Interrupt Return Instruction



Correspondence between vector table address for BIT interrupt and the entry address of the interrupt service program.

A interrupt request is not accepted until the I-flag is set to "1" even if a requested interrupt has higher priority than that of the current interrupt being serviced.

When nested interrupt service is required, the I-flag should be set to "1" by “EI” instruction in the interrupt service program. In this case, acceptable interrupt sources are selectively enabled by the individual interrupt enable flags.

Saving/Restoring General-purpose Register

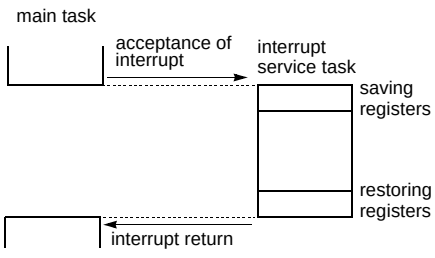
During interrupt acceptance processing, the program counter and the program status word are automatically saved on the stack, but accumulator and other registers are not saved itself. These registers are saved by the software if necessary. Also, when multiple interrupt services are nested, it is necessary to avoid using the same data memory area for saving registers.

The following method is used to save/restore the general-purpose registers.

Example: Register save using push and pop instructions

```
INTxx:  PUSH    A       ;SAVE ACC.
        PUSH    X       ;SAVE X REG.
        PUSH    Y       ;SAVE Y REG.
        interrupt processing
        POP     Y       ;RESTORE Y REG.
        POP     X       ;RESTORE X REG.
        POP     A       ;RESTORE ACC.
        RETI            ;RETURN
```

General-purpose register save/restore using push and pop instructions;



18.2 BRK Interrupt

Software interrupt can be invoked by BRK instruction, which has the lowest priority order.

Interrupt vector address of BRK is shared with the vector of TCALL 0 (Refer to Program Memory Section). When BRK interrupt is generated, B-flag of PSW is set to distinguish BRK from TCALL 0.

Each processing step is determined by B-flag as shown in Figure 18-4.

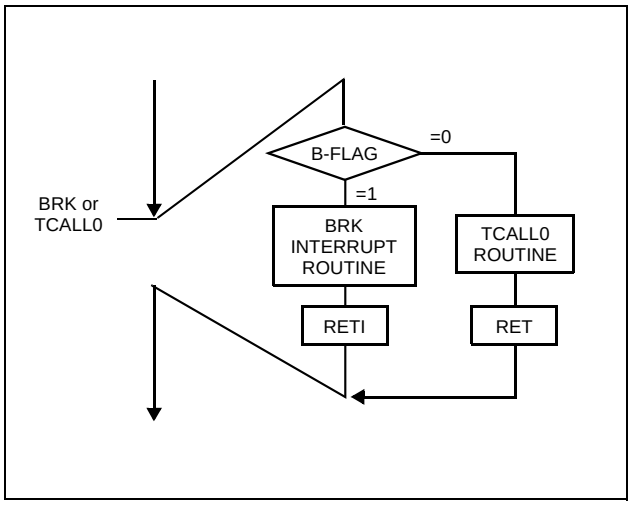


Figure 18-4 Execution of BRK/TCALL0

18.3 Multi Interrupt

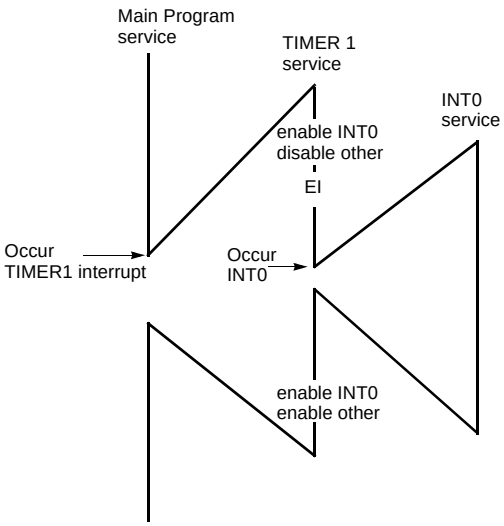
If two requests of different priority levels are received simultaneously, the request of higher priority level is serviced. If requests of the interrupt are received at the same time simultaneously, an internal polling sequence determines by hardware which request is serviced.

However, multiple processing through software for special features is possible. Generally when an interrupt is accepted, the I-flag is cleared to disable any further interrupt. But as user sets I-flag in interrupt routine, some further interrupt can be serviced even if certain interrupt is in progress.

Example: Even though Timer1 interrupt is in progress, INT0 interrupt serviced without any suspend.

```
TIMER1: PUSH   A
        PUSH   X
        PUSH   Y
        LDM    IENH,#80H    ;Enable INT0 only
        LDM    IENL,#0      ;Disable other
        EI                  ;Enable Interrupt
        :
        :
        :


        :
        :
        :
        LDM    IENH,#0FFH   ;Enable all interrupts
        LDM    IENL,#0F0H
        POP    Y
        POP    X
        POP    A
        RETI
```



In this example, the INT0 interrupt can be serviced without any pending, even TIMER1 is in progress.
Because of re-setting the interrupt enable registers IENH,IENL and master enable "EI" in the TIMER1 routine.

Figure 18-5 Execution of Multi Interrupt

18.4 External Interrupt

The external interrupt on INT0 and INT1 pins are edge triggered depending on the edge selection register IEDS (address $0E6_H$) as shown in Figure 18-6 .

The edge detection of external interrupt has three transition activated mode: rising edge, falling edge, and both edge

.

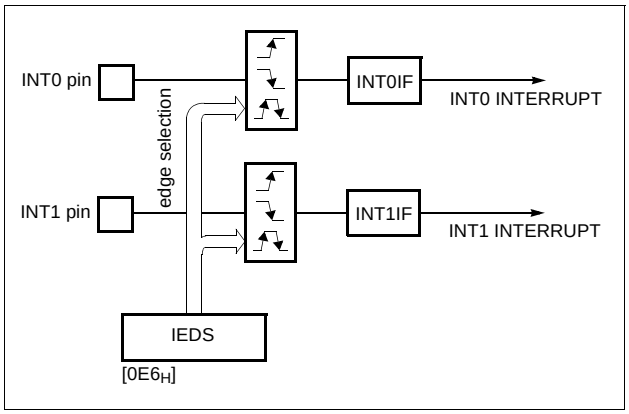
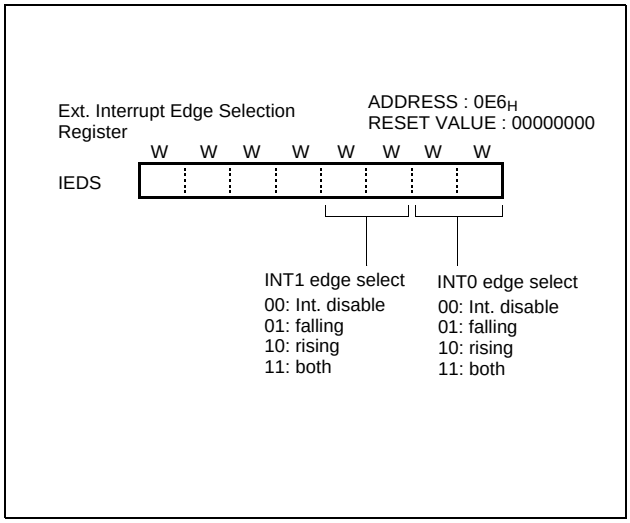


Figure 18-6 External Interrupt Block Diagram

Example: To use as an INT0 and INT1

```
            :
            :
;**** Set port as an input port RB2,RB3
            LDM     RBIO,#1111_0011B
            ;
            ;
;**** Set port as an interrupt port
            LDM     RBFUNC,#0C0H
            ;
            ;
;**** Set Falling-edge Detection
            LDM     IEDS,#0000_0101B
            :
            :
```



Response Time

The INT0 and INT1 edge are latched into INT0IF and INT1IF at every machine cycle. The values are not actually polled by the circuitry until the next machine cycle. If a request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The DIV itself takes twelve cycles. Thus, a minimum of twelve complete machine cycles elapse between activation of an external interrupt request and the beginning of execution of the first instruction of the service routine.

Below shows interrupt response timings.

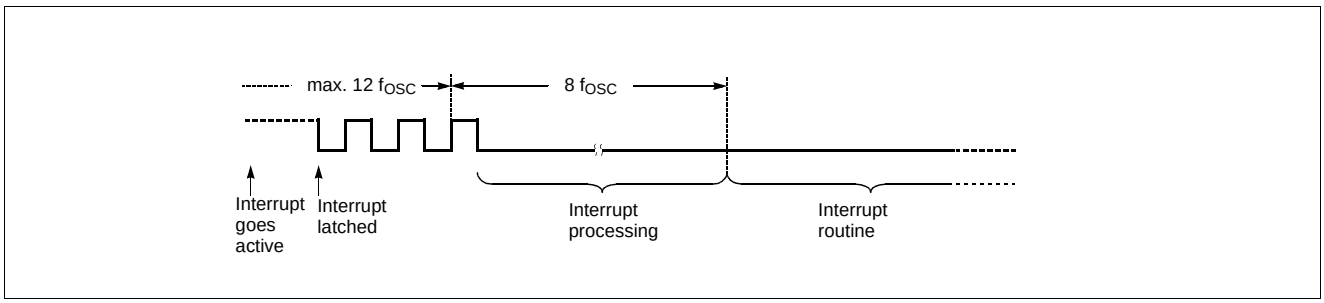


Figure 18-7 Interrupt Response Timing Diagram

19. WATCHDOG TIMER

The purpose of the watchdog timer is to detect the malfunction (runaway) of program due to external noise or other causes and return the operation to the normal condition.

The watchdog timer has two types of clock source.

The first type is an on-chip RC oscillator which does not require any external components. This RC oscillator is separate from the external oscillator of the Xin pin. It means that the watchdog timer will run, even if the clock on the Xin pin of the device has been stopped, for example, by entering the STOP mode.

The other type is a prescaled system clock.

The watchdog timer consists of 7-bit binary counter and the watchdog timer data register. The source clock of WDT is overflow of Basic Interval Timer. When the value of 7-bit binary counter is equal to the lower 7 bits of WDTR, the interrupt request flag is generated. This can be used as WDT interrupt or CPU reset signal in accordance with the bit WDTON .

Note: *Because the watchdog timer counter is enabled after clearing Basic Interval Timer, after the bit WDTON set to "1", maximum error of timer is depend on prescaler ratio of Basic Interval Timer.*

The 7-bit binary counter is cleared by setting WDTCL(bit7 of WDTR) and the WDTCL is cleared automatically after 1 machine cycle.

The RC oscillated watchdog timer is activated by setting the bit RCWDT of CKCTLR and executing the STOP instruction as shown below.

```
  :
LDM     CKCTLR,#3FH   ; enable the RC-osc WDT
LDM     WDTR,#0FFH    ; set the WDT period
STOP                  ; enter the STOP mode
NOP
NOP                   ; RC-osc WDT running
  :
```

The RC oscillation period is variable according to the temperature, V_{DD} and process variations from part to part (approximately, 120~180uS). The following equation shows the RC oscillated watchdog timer time-out.

$$T_{RCWDT} = CLK_{RC} \times 2^8 \times [WDTR.6\sim0] + (CLK_{RC} \times 2^8)/2$$

$$where,\ CLK_{RC} = 120\sim180uS$$

In addition, this watchdog timer can be used as a simple 7-bit timer by interrupt WDTIF. The interval of watchdog timer interrupt is decided by Basic Interval Timer. Interval equation is as below.

$$\boldsymbol{T_{WDT} = [WDTR.6\sim0] \times Interval\ of\ BIT}$$

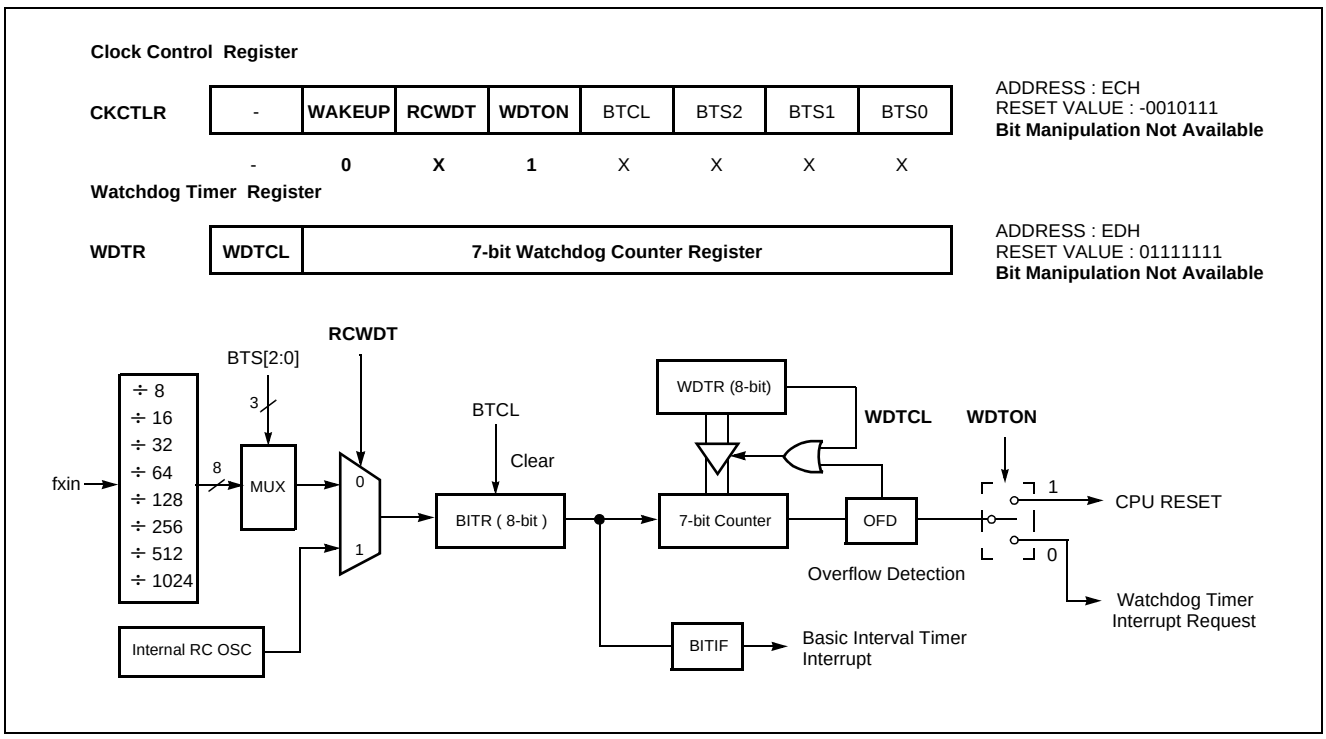


Figure 19-1 Block Diagram of Watchdog Timer

20. Power Saving Mode

For applications where power consumption is a critical factor, device provides three kinds of power saving functions, STOP mode, Wake-up Timer mode and internal RC-oscillated watchdog timer mode.

The power saving function is activated by execution of STOP instruction after setting the corresponding bit (WAKEUP, RCWDT) of CKCTLR.

Table 20-1 shows the status of each Power Saving Mode

Note: ***Before executing STOP instruction, clear all interrupt request flag.*** *Because if the interrupt request flag is set before STOP instruction, the MCU runs as if it doesn't perform STOP instruction, even though the STOP instruction is completed. So insert two lines to clear all interrupt request flags (IRQH, IRQL) before STOP instruction as shown each example.*

Peripheral	STOP	Wake-up Timer	Internal RC-WDT
RAM	Retain	Retain	Retain
Control Registers	Retain	Retain	Retain
I/O Ports	Retain	Retain	Retain
CPU	Stop	Stop	Stop
Timer0	Stop	Operation	Stop
Oscillation	Stop	Oscillation	Stop
Prescaler	Stop	÷ 2048 only	Stop
Internal RC oscillator	Stop	Stop	Oscillation
Entering Condition CKCTLR[6,5]	00	1X	01
Power Saving Release Source	RESET, INT0, INT1	RESET, INT0, INT1, Timer0	RESET, INT0, INT1, RC-WDT

Table 20-1 Power Saving Mode

20.1 Stop Mode

In the Stop mode, the on-chip oscillator is stopped. With the clock frozen, all functions are stopped, but the on-chip RAM and Control registers are held. The port pins out the values held by their respective port data register, port direction registers. Oscillator stops and the systems internal operations are all held up.

- The states of the RAM, registers, and latches valid immediately before the system is put in the STOP state are all held.
- The program counter stop the address of the instruction to be executed after the instruction "STOP" which starts the STOP operating mode.

The Stop mode is activated by execution of STOP instruction after setting the bit WAKEUP and RCWDT of CKCTLR to "00". (This register should be written by byte operation. If this register is set by bit manipulation instruction, for example "set1" or "clr1" instruction, it may be undesired operation)

In the Stop mode of operation, V_{DD} can be reduced to minimize power consumption. Care must be taken, however, to ensure that V_{DD} is not reduced before the Stop mode is invoked, and that V_{DD} is restored to its normal operating level, before the Stop mode is terminated.

The reset should not be activated before V_{DD} is restored to its normal operating level, and must be held active long enough to allow the oscillator to restart and stabilize.

Note: *After STOP instruction, at least two or more NOP instruction should be written*

```
Ex)   LDM CKCTLR,#0000_1110B
      LDM IRQH,#0
      LDM IRQL,#0
      STOP
      NOP
      NOP
```

In the STOP operation, the dissipation of the power associated with the oscillator and the internal hardware is lowered; however, the power dissipation associated with the pin interface (depending on the external circuitry and program) is not directly determined by the hardware operation of the STOP feature. This point should be little current flows when the input level is stable at the power voltage

level (V_{DD}/V_{SS}), however, when the input level gets higher than the power voltage level (by approximately 0.3 to 0.5V), a current begins to flow. Therefore, if cutting off the output transistor at an I/O port puts the pin signal into the high-impedance state, a current flow across the ports input transistor, requiring to fix the level by pull-up or other means.

Release the STOP mode

The exit from STOP mode is hardware reset or external interrupt. Reset re-defines all the Control registers but does not change the on-chip RAM. External interrupts allow both on-chip RAM and Control registers to retain their values.

After releasing STOP mode, instruction execution is divided into two ways by I-flag(bit2 of PSW).
If I-flag = 1, the normal interrupt response takes place. If I-flag = 0, the chip will resume execution starting with the instruction following the STOP instruction. It will not vector to interrupt service routine. (refer to Figure 20-1)

When exit from Stop mode by external interrupt, enough oscillation stabilization time is required to normal operation. Figure 20-2 shows the timing diagram. When release the Stop mode, the Basic interval timer is activated on wake-up. It is increased from 00_H until FF_H. The count overflow is set to start normal operation. Therefore, before STOP instruction, user must be set its relevant prescaler divide ratio to have long enough time (more than 20msec). This guarantees that oscillator has started and stabilized.

By reset, exit from Stop mode is shown in Figure 20-3.

Minimizing Current Consumption in Stop Mode

The Stop mode is designed to reduce power consumption. To minimize the current consumption during Stop mode, the user should turn-off output drivers that are sourcing or sinking current, if it is practical. Weak pull-ups on port pins should be turned off, if possible. All inputs should be either as V_{SS} or at V_{DD} (or as close to rail as possible).

An intermediate voltage on an input pin causes the input buffer to draw a significant amount of current.

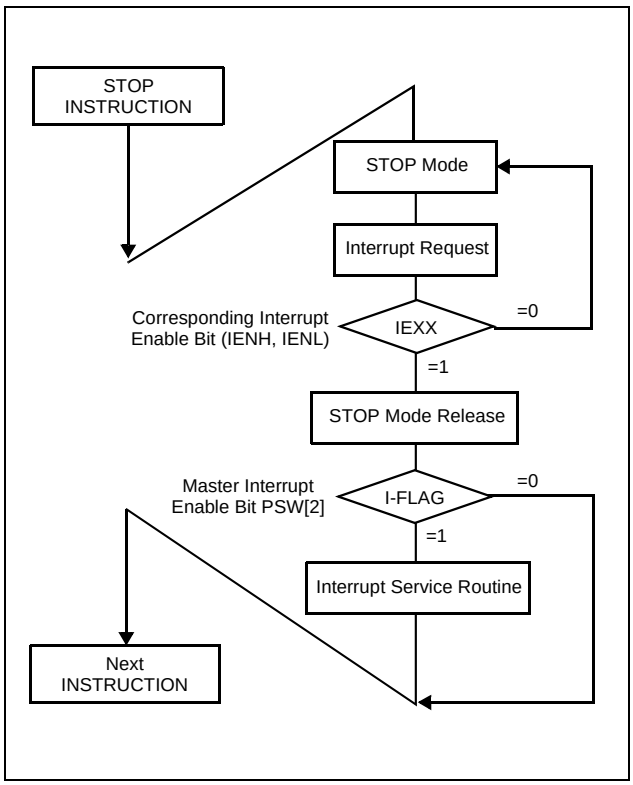


Figure 20-1 STOP Releasing Flow by Interrupts

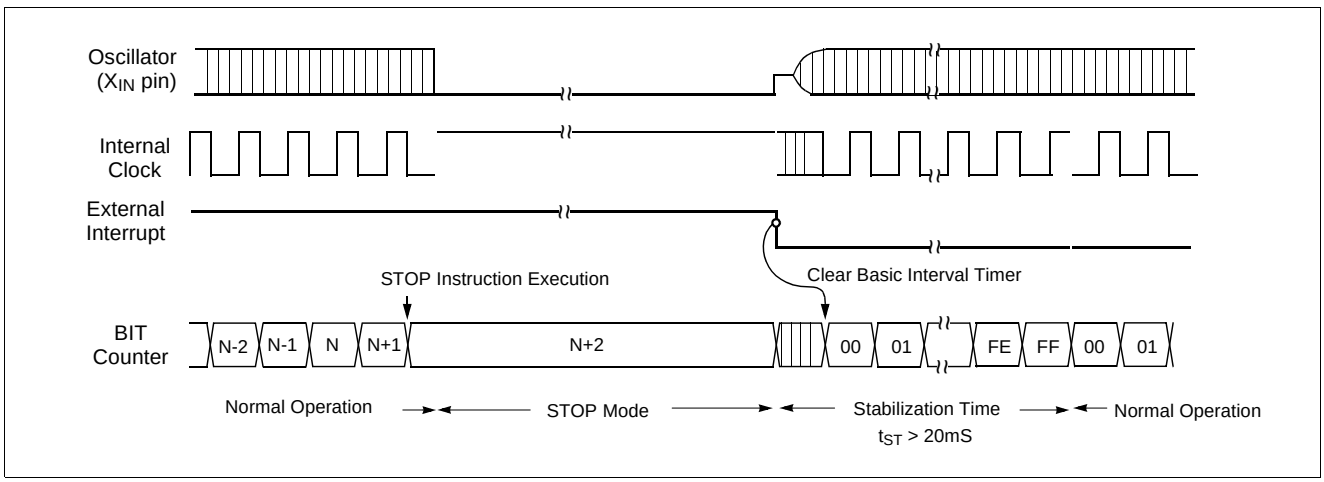


Figure 20-2 Timing of STOP Mode Release by External Interrupt

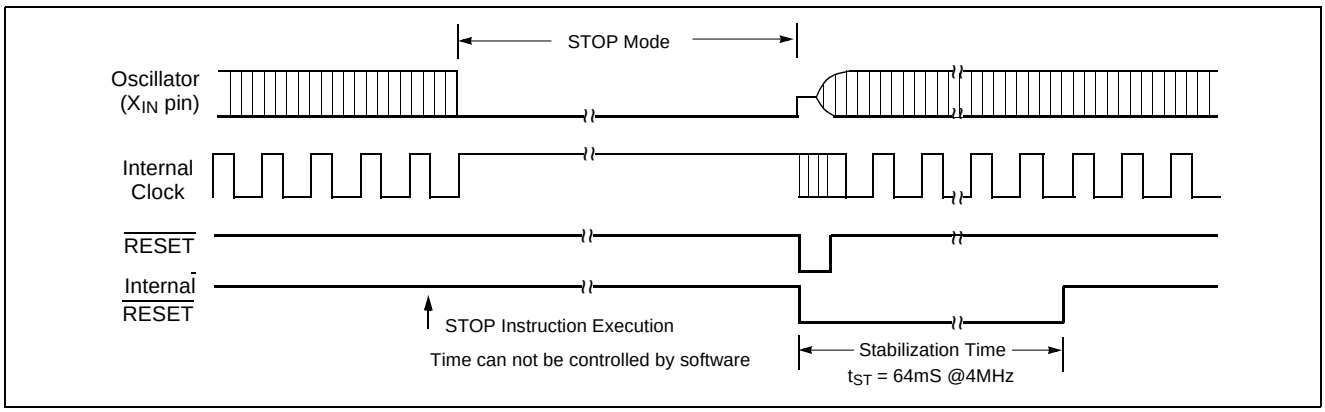


Figure 20-3 Timing of STOP Mode Release by RESET

20.2 Wake-up Timer Mode

In the Wake-up Timer mode, the on-chip oscillator is not stopped. Except the Prescaler (only 2048 divided ratio) and Timer0, all functions are stopped, but the on-chip RAM and Control registers are held. The port pins out the values held by their respective port data register, port direction registers.

The Wake-up Timer mode is activated by execution of STOP instruction after setting the bit WAKEUP of CKCTLR to "1". (This register should be written by byte operation. If this register is set by bit manipulation instruction, for example "set1" or "clr1" instruction, it may be undesired operation)

Note: *After STOP instruction, at least two or more NOP instruction should be written*

```
Ex)  LDM   TDR0,#0FFH
     LDM   TM0,#0001_1011B
     LDM   CKCTLR,#0100_1110B
     LDM   IRQH,#0
     LDM   IRQL,#0
     STOP
     NOP
     NOP
```

In addition, the clock source of timer0 should be selected to 2048 divided ratio. Otherwise, the wake-up function can not work. And the timer0 can be operated as 16-bit timer with timer1 (refer to timer function). The period of wake-up function is varied by setting the timer data register 0, TDR0.

Release the Wake-up Timer mode

The exit from Wake-up Timer mode is hardware reset, Timer0 overflow or external interrupt. Reset re-defines all the Control registers but does not change the on-chip RAM. External interrupts and Timer0 overflow allow both on-chip RAM and Control registers to retain their values.

If I-flag = 1, the normal interrupt response takes place. If I-flag = 0, the chip will resume execution starting with the instruction following the STOP instruction. It will not vector to interrupt service routine (refer to Figure 20-1).

When exit from Wake-up Timer mode by external interrupt or timer0 overflow, the oscillation stabilization time is not required to normal operation. Because this mode do not stop the on-chip oscillator shown as Figure 20-4.

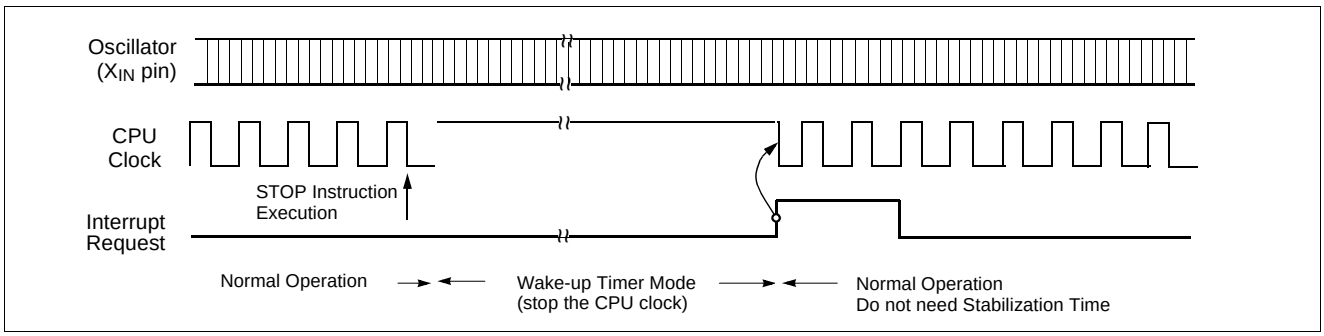


Figure 20-4 Wake-up Timer Mode Releasing by External Interrupt or Timer0 Interrupt

20.3 Internal RC-Oscillated Watchdog Timer Mode

In the Internal RC-Oscillated Watchdog Timer mode, the on-chip oscillator is stopped. But internal RC oscillation circuit is oscillated in this mode. The on-chip RAM and Control registers are held. The port pins out the values held by their respective port data register, port direction registers.

The Internal RC-Oscillated Watchdog Timer mode is activated by execution of STOP instruction after setting the bit WAKEUP and RCWDT of CKCTLR to "01". (This register should be written by byte operation. If this register is set by bit manipulation instruction, for example "set1" or "clr1" instruction, it may be undesired operation)

Note: *After STOP instruction, at least two or more NOP instruction should be written*

```
Ex)  LDM   WDTR,#1111_1111B
     LDM   CKCTLR,#0010_1110B
     LDM   IRQH,#0
     LDM   IRQL,#0
     STOP
     NOP
     NOP
```

Release the Internal RC-Oscillated Watchdog Timer mode

The exit from Internal RC-Oscillated Watchdog Timer mode is hardware reset or external interrupt. Reset re-defines all the Control registers but does not change the on-chip RAM. External interrupts allow both on-chip RAM and Control registers to retain their values.

If I-flag = 1, the normal interrupt response takes place. In this case, if the bit WDTON of CKCTLR is set to "0" and the bit WDTE of IENH is set to "1", the device will execute the watchdog timer interrupt service routine.(Figure 20-5) However, if the bit WDTON of CKCTLR is set to "1", the device will generate the internal RESET signal and execute the reset processing. (Figure 20-6)

If I-flag = 0, the chip will resume execution starting with the instruction following the STOP instruction. It will not vector to interrupt service routine (refer to Figure 20-1).

When exit from Internal RC-Oscillated Watchdog Timer mode by external interrupt, the oscillation stabilization time is required for normal operation. Figure 20-5 shows the timing diagram. When release the Internal RC-Oscillated Watchdog Timer mode, the basic interval timer is activated on wake-up. It is increased from 00_H until FF_H. The count overflow is set to start normal operation. Therefore, before STOP instruction, user must be set its relevant prescaler divide ratio to have long enough time (more than 20msec). This guarantees that oscillator has started and stabilized.

By reset, exit from internal RC-Oscillated Watchdog Timer mode is shown in Figure 20-6.

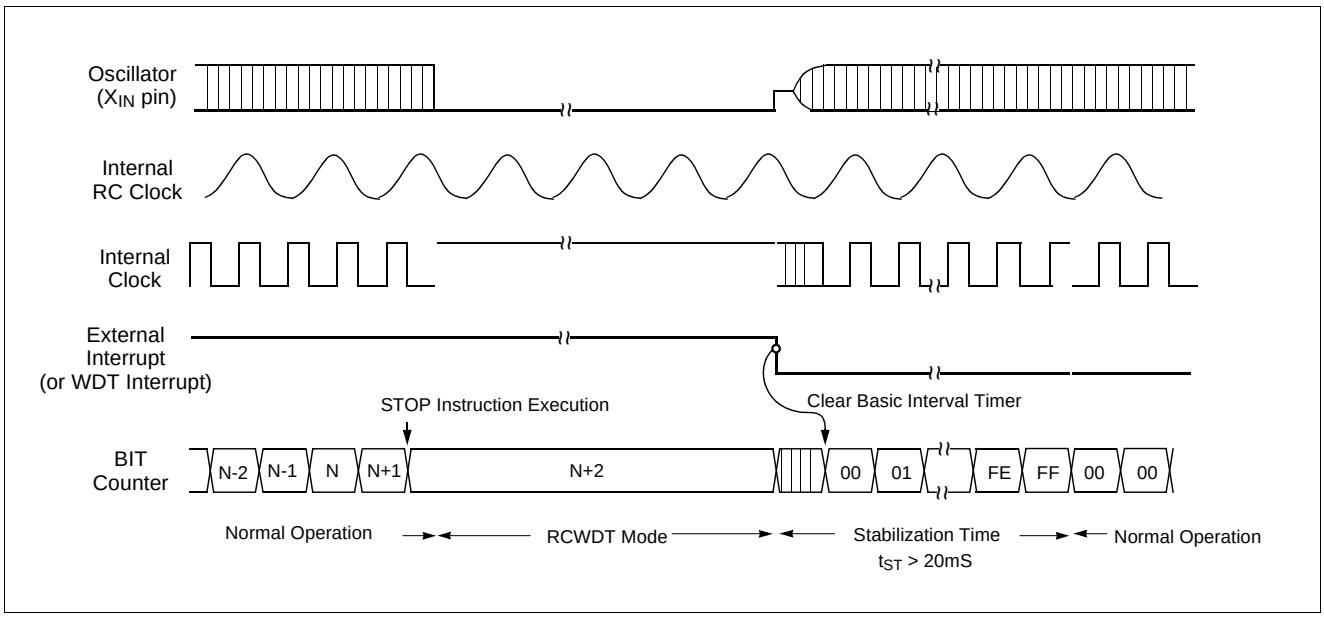


Figure 20-5 Internal RCWDT Mode Releasing by External Interrupt or WDT Interrupt

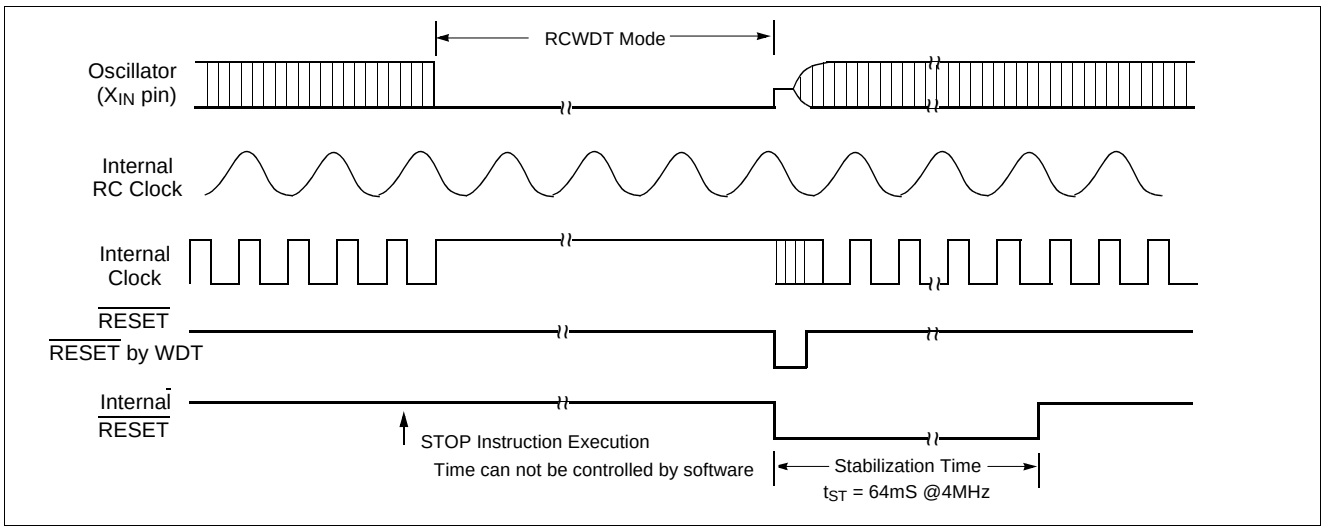
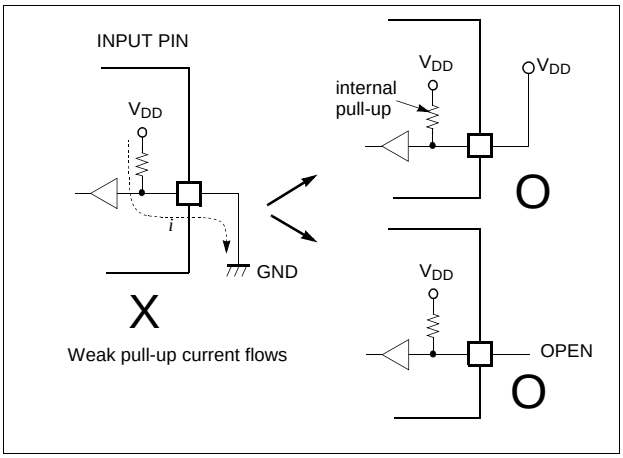


Figure 20-6 Internal RCWDT Mode Releasing by RESET.



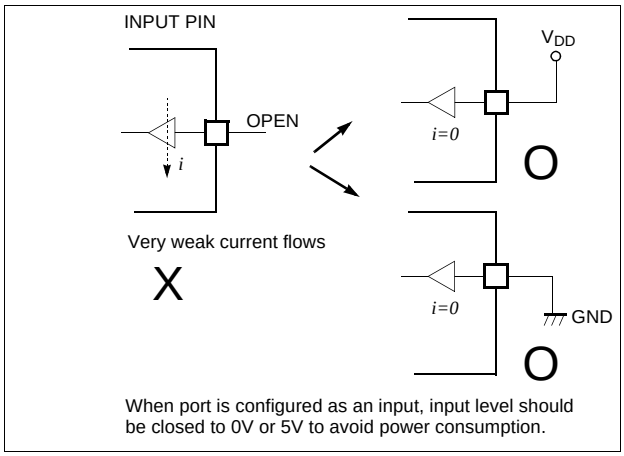
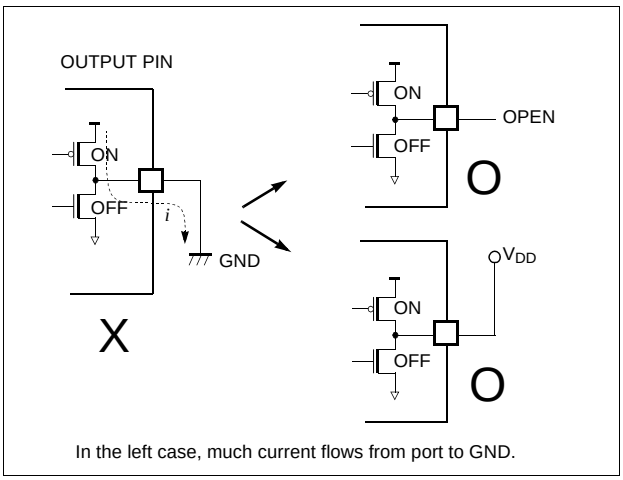


Figure 20-7 Application Example of Unused Input Por t



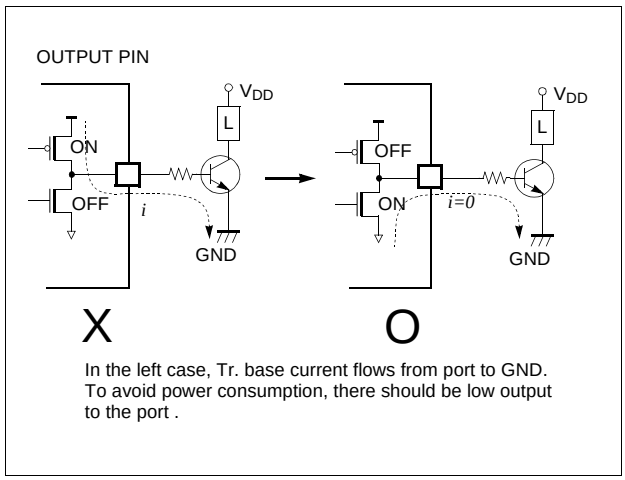


Figure 20-8 Application Example of Unused input Port

21. RESET

The reset input is the RESET pin, which is the input to a Schmitt Trigger. A reset in accomplished by holding the RESET pin low for at least 8 oscillator periods, while the oscillator running. After reset, 64ms (at 4 MHz) add with 7 oscillator periods are required to start execution as shown in Figure 21-1 .

Internal RAM is not affected by reset. When V_{DD} is turned on, the RAM content is indeterminate. Therefore, this RAM should be initialized before reading or testing it.

Initial state of each register is shown as Table 11-3 .

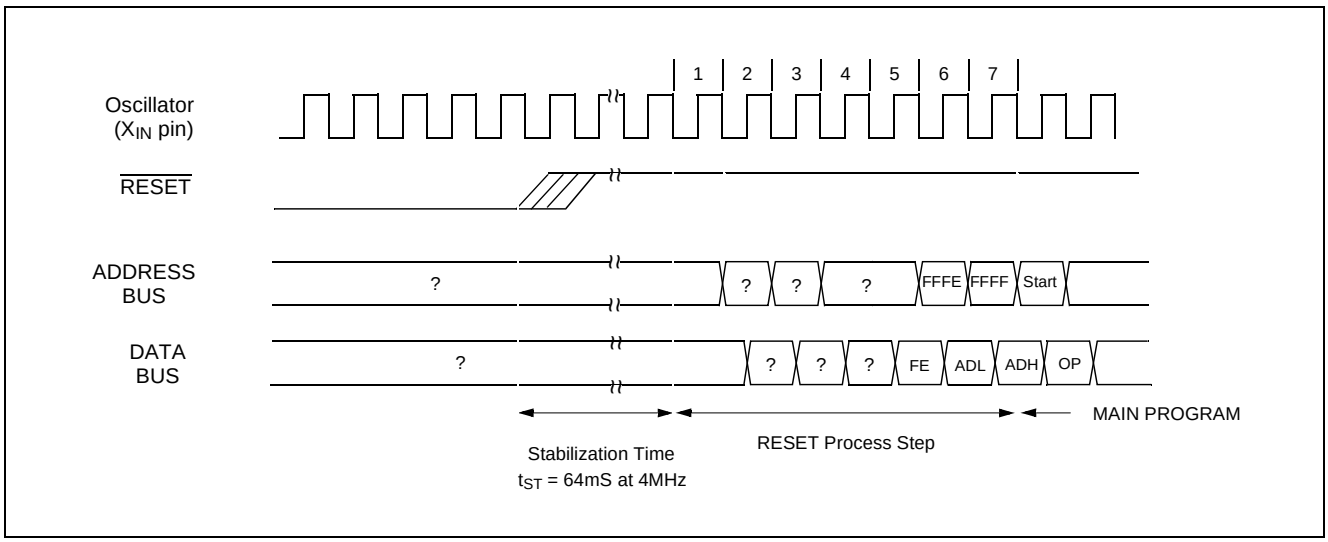


Figure 21-1 Timing Diagram after RESET

22. POWER FAIL PROCESSOR

The GMS87C1202 has an on-chip power fail detection circuitry to immunize against power noise. A configuration register, PFDR, can enable (if clear/programmed) or disable (if set) the Power-fail Detect circuitry. If V_{DD} falls below 3.0~4.0V range for longer than 50 nS, the Power fail situation may reset MCU according to PFR bit of PFDR.

As below PFDR register is not implemented on the in-circuit emulator, user can not experiment with it. Therefore, after final development of user program, this function may be experimented.

Note: *Power fail processor function is not available on 3V operation, because this function will detect power fail all the time.*

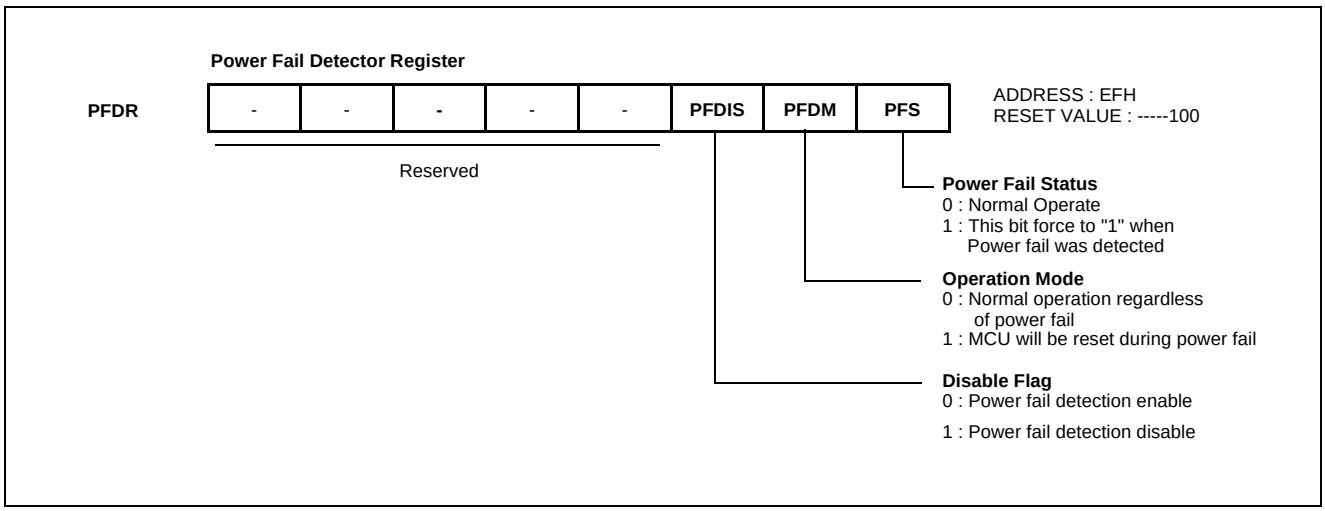


Figure 22-1 Power Fail Detector Register

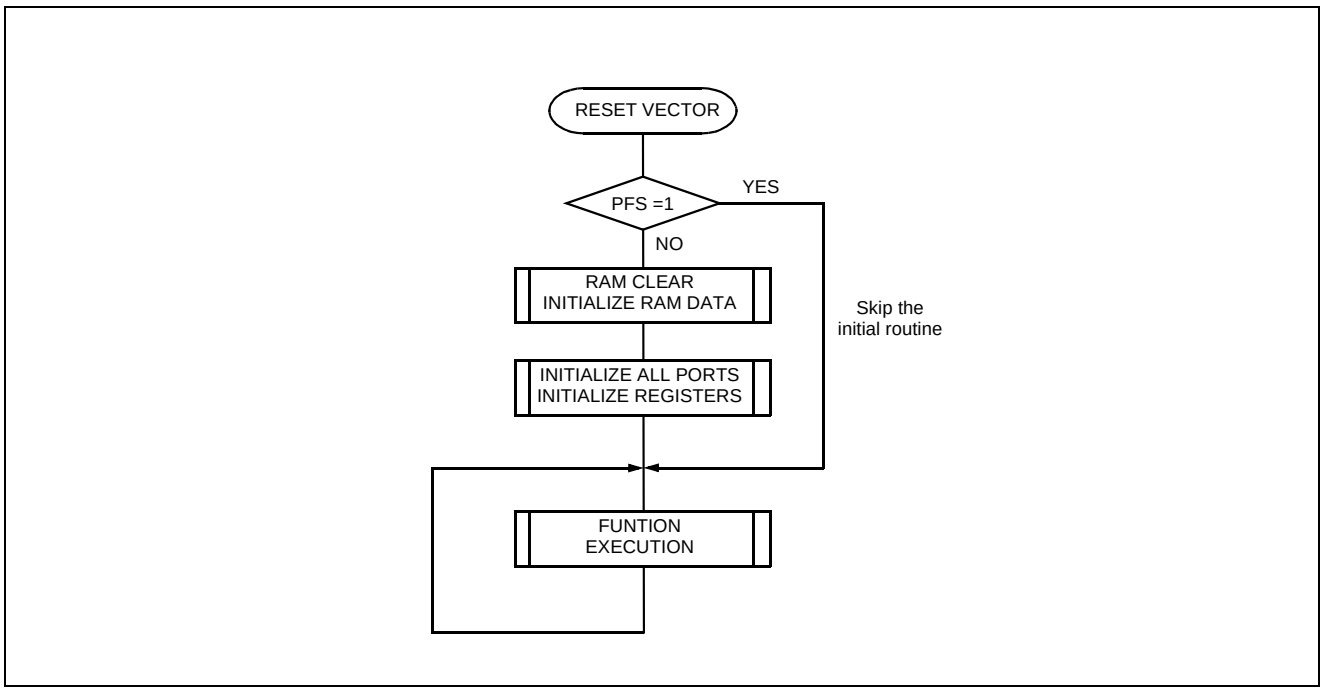


Figure 22-2 Example S/W of RESET by Power fail

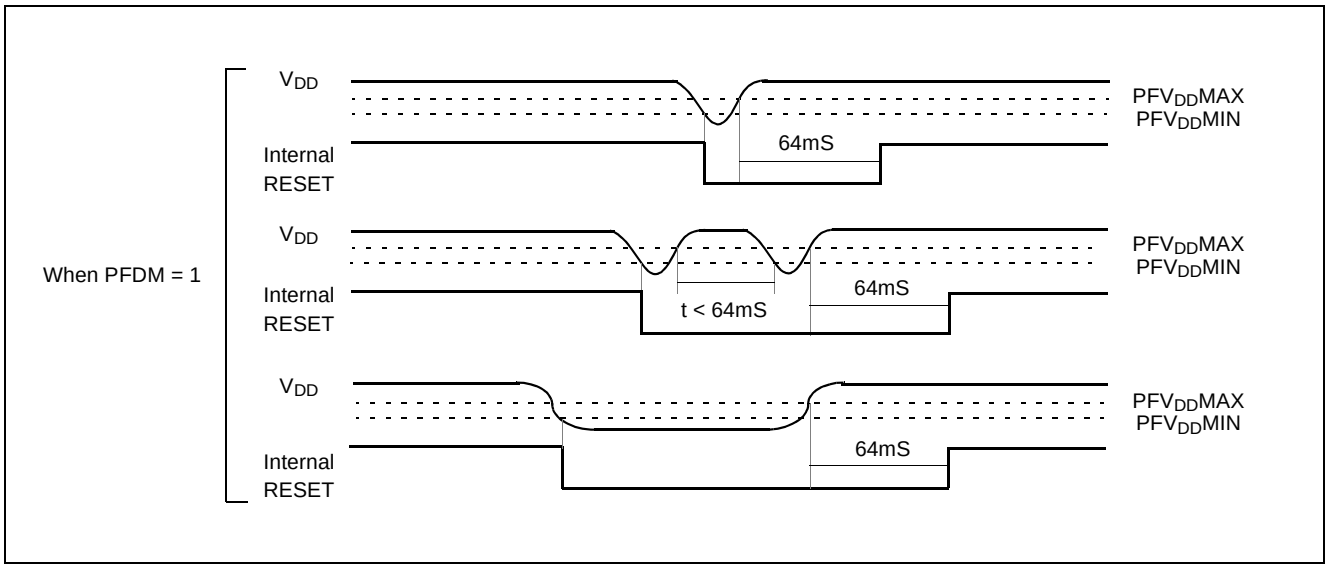


Figure 22-3 Power Fail Processor Situations

23. OTP PROGRAMMING

The GMS87C1102/1202T is one-time PROM(OTP) microcontroller with 2K bytes electrically programmable read only memory for the GMS87C1102/1202 system evaluation, first production and fast mass production.

To programming the OTP device, user must use the universal programmer which is support HYUNDAI MicroElectronics microcontrollers.

23.1 Program Memory MAP

Program Memory consists of configuration area and user program memory area. The configuration memory area has two parts (User ID & System Configuration Bits), the areas are shown below in Figure 1-1.

The Device Configuration Area can be programmed or left unprogrammed to select device configuration such as security bit. Ten memory locations ($0F50_H$ ~ $0FE0_H$) are designated as Customer ID recording locations where the user can store check-sum or other customer identification numbers.
This area is not accessible during normal execution but is readable and writable during program / verify.

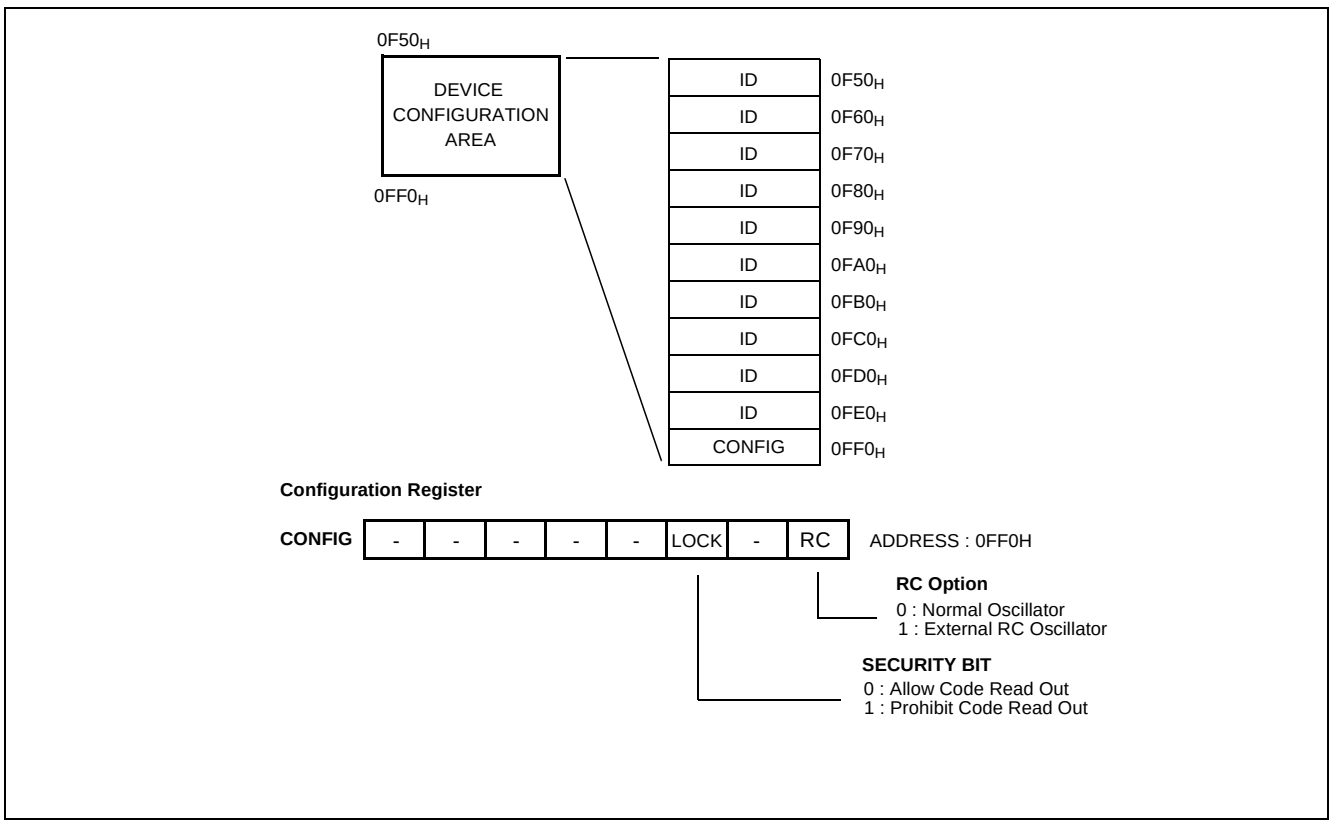


Figure 23-1 Device Configuration Area

The Security Definition Method is explained below.

1) After writing “H” to code protect bit in Write & Verify Mode and getting out of Write & Verify Mode, user cannot read out the program code. But if not getting out of Write & Verify Mode (maintaining Programming Power VPP = 12.75V), user can verify Program code.
2) Regardless of Code protect, user can read out configuration Memory (User ID and Configuration Bits)
3) If user knows Security (Lock) state, user can read code protect bit in the System Configuration Bits.

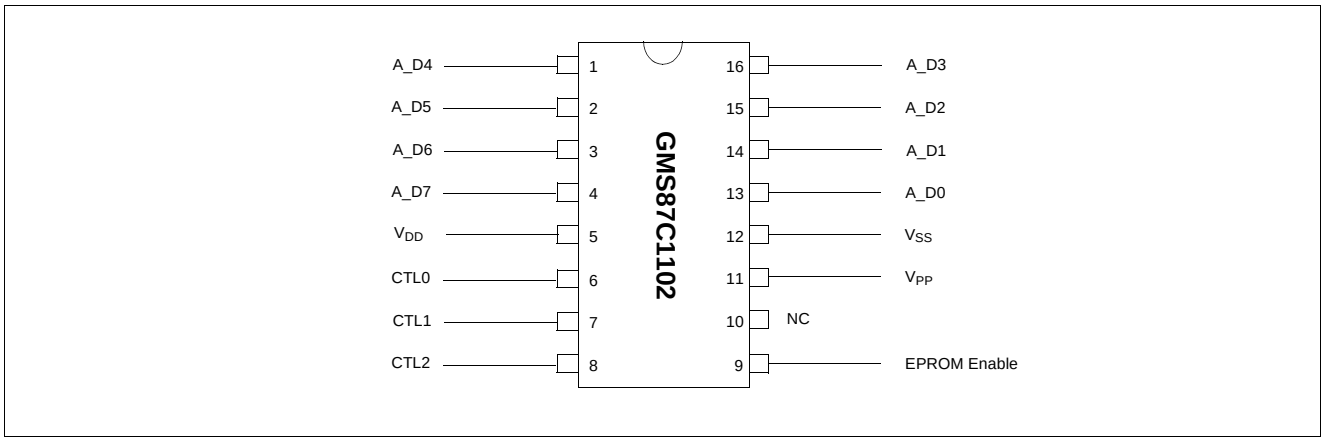


Figure 23-2 Pin Assignment

Pin No.	User Mode	EPROM MODE				
	Pin Name	Pin Name	Description			
1	RA4 (AN4)	A_D4	Address Input Data Input/Output	A12	A4	D4
2	RA5 (AN5)	A_D5		A13	A5	D5
3	RA6 (AN6)	A_D6		A14	A6	D6
4	RA7 (AN7)	A_D7		A15	A7	D7
5	V_{DD}	V_{DD}	Connect to V_{DD} (6.0V)			
6	RB0 (AVref/AN0)	CTL0	Read/Write Control Address/Data Control			
7	RB2 (INT0)	CTL1				
8	RB4 (PWM/COMP)	CTL2				
9	X_{IN}	EPROM Enable	High Active, Latch Address in falling edge			
10	X_{OUT}	NC	No connection			
11	$\overline{RESET}$	V_{PP}	Programming Power (0V, 12.75V)			
12	V_{SS}	V_{SS}	Connect to V_{SS} (0V)			
13	RA0 (EC0)	A_D0	Address Input Data Input/Output	A8	A0	D0
14	RA1 (AN1)	A_D1		A9	A1	D1
15	RA2 (AN2)	A_D2		A10	A2	D2
16	RA3 (AN3)	A_D3		A11	A3	D3

Table 23-1 Pin Description in EPROM Mode

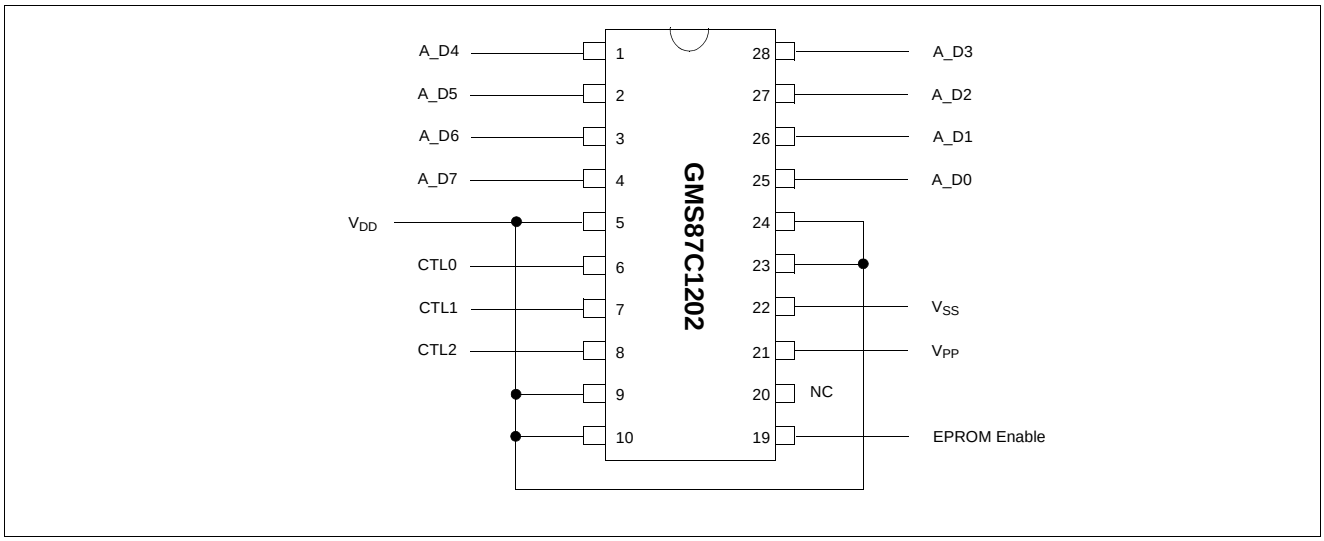


Figure 23-3 Pin Assignment

Pin No.	User Mode	EPROM MODE				
	Pin Name	Pin Name	Description			
1	RA4 (AN4)	A_D4	Address Input Data Input/Output	A12	A4	D4
2	RA5 (AN5)	A_D5		A13	A5	D5
3	RA6 (AN6)	A_D6		A14	A6	D6
4	RA7 (AN7)	A_D7		A15	A7	D7
5	V_{DD}	V_{DD}	Connect to V_{DD} (6.0V)			
6	RB0 (AVref/AN0)	CTL0	Read/Write Control Address/Data Control			
7	RB1 (BUZ)	CTL1				
8	RB2 (INT0)	CTL2				
9	RB3 (INT1)	V_{DD}	Connect to V_{DD} (6.0V)			
10	RB4 (PWM/COMP)	V_{DD}	Connect to V_{DD} (6.0V)			
11	X_{IN}	EPROM Enable	High Active, Latch Address in falling edge			
12	X_{OUT}	NC	No connection			
13	$\overline{RESET}$	V_{PP}	Programming Power (0V, 12.75V)			
14	V_{SS}	V_{SS}	Connect to V_{SS} (0V)			
15,16	RC0, 1	V_{DD}	Connect to V_{DD} (6.0V)			
17	RA0 (EC0)	A_D0	Address Input Data Input/Output	A8	A0	D0
18	RA1 (AN1)	A_D1		A9	A1	D1
19	RA2 (AN2)	A_D2		A10	A2	D2
20	RA3 (AN3)	A_D3		A11	A3	D3

Table 23-2 Pin Description in EPROM Mode

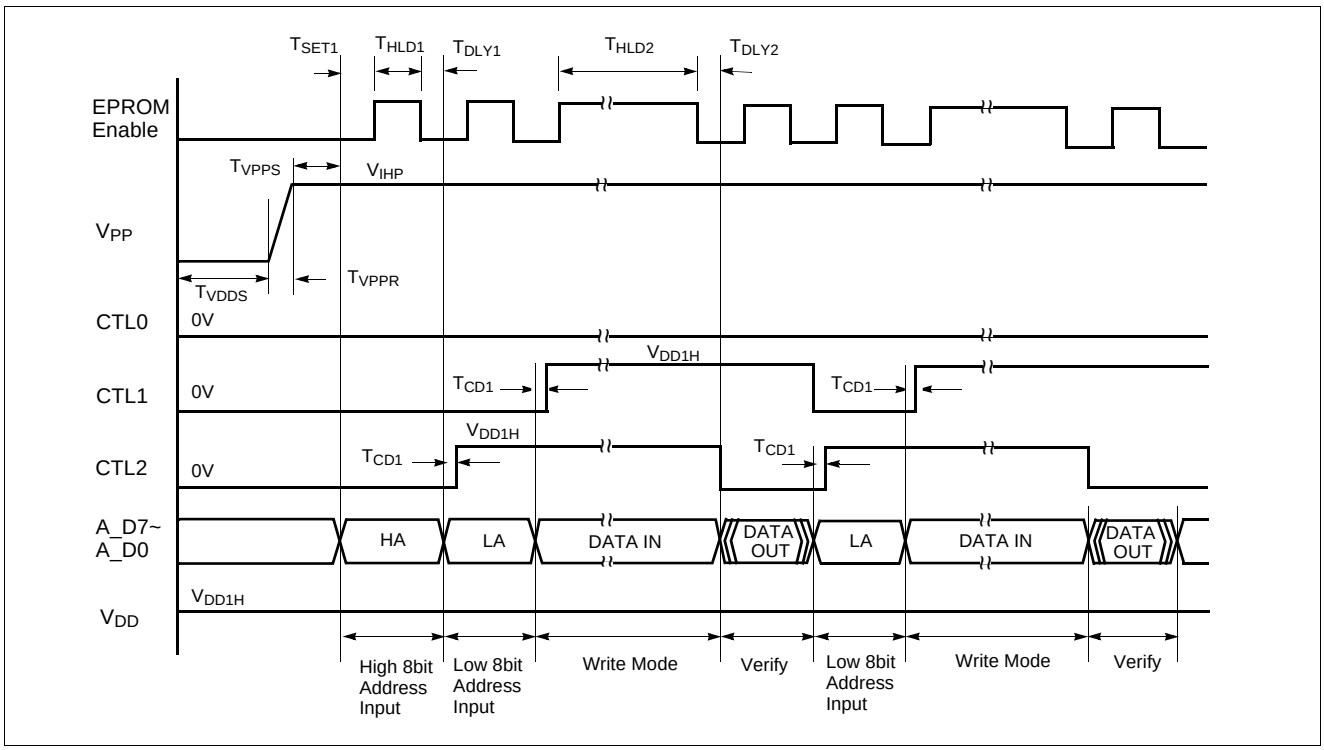


Figure 23-4 Timing Diagram in Program (Write & Verify) Mode

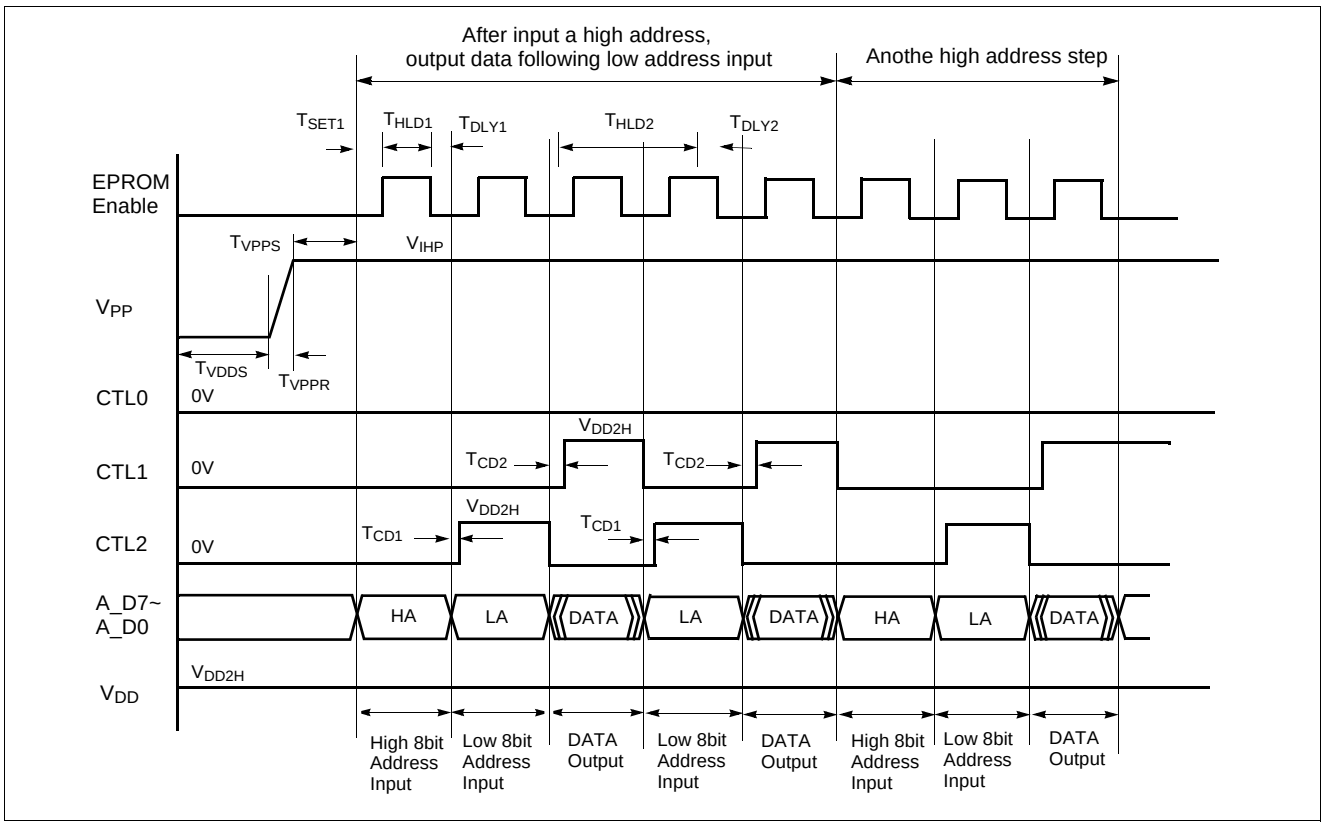


Figure 23-5 Timing Diagram in READ Mode

Parameter	Symbol	MIN	TYP	MAX	Unit
Programming Supply Current	I_{VPP}	-	-	50	mA
Supply Current in EPROM Mode	I_{VDDP}	-	-	20	mA
V_{PP} Level during Programming	V_{IHP}	12.0	12.5	13.0	V
V_{DD} Level in Program Mode	V_{DD1H}	5	6	6.5	V
V_{DD} Level in Read Mode	V_{DD2H}	-	2.7	-	V
CTL2~0 High Level in EPROM Mode	V_{IHC}	$0.8V_{DD}$	-	-	V
CTL2~0 Low Level in EPROM Mode	V_{ILC}	-	-	$0.2V_{DD}$	V
A_D7~A_D0 High Level in EPROM Mode	V_{IHAD}	$0.9V_{DD}$	-	-	V
A_D7~A_D0 Low Level in EPROM Mode	V_{ILAD}	-	-	$0.1V_{DD}$	V
V_{DD} Saturation Time	T_{VDDS}	1	-	-	mS
V_{PP} Setup Time	T_{VPPR}	-	-	1	mS
V_{PP} Saturation Time	T_{VPPS}	1	-	-	mS
EPROM Enable Setup Time after Data Input	T_{SET1}		200		nS
EPROM Enable Hold Time after T_{SET1}	T_{HLD1}		500		nS
EPROM Enable Delay Time after T_{HLD1}	T_{DLY1}		200		nS
EPROM Enable Hold Time in Write Mode	T_{HLD2}		100		nS
EPROM Enable Delay Time after T_{HLD2}	T_{DLY2}		200		nS
CTL2,1 Setup Time after Low Address input and Data input	T_{CD1}		100		nS
CTL1 Setup Time before Data output in Read and Verify Mode	T_{CD2}		100		nS

Table 23-3 AC/DC Requirements for Program/Read Mode

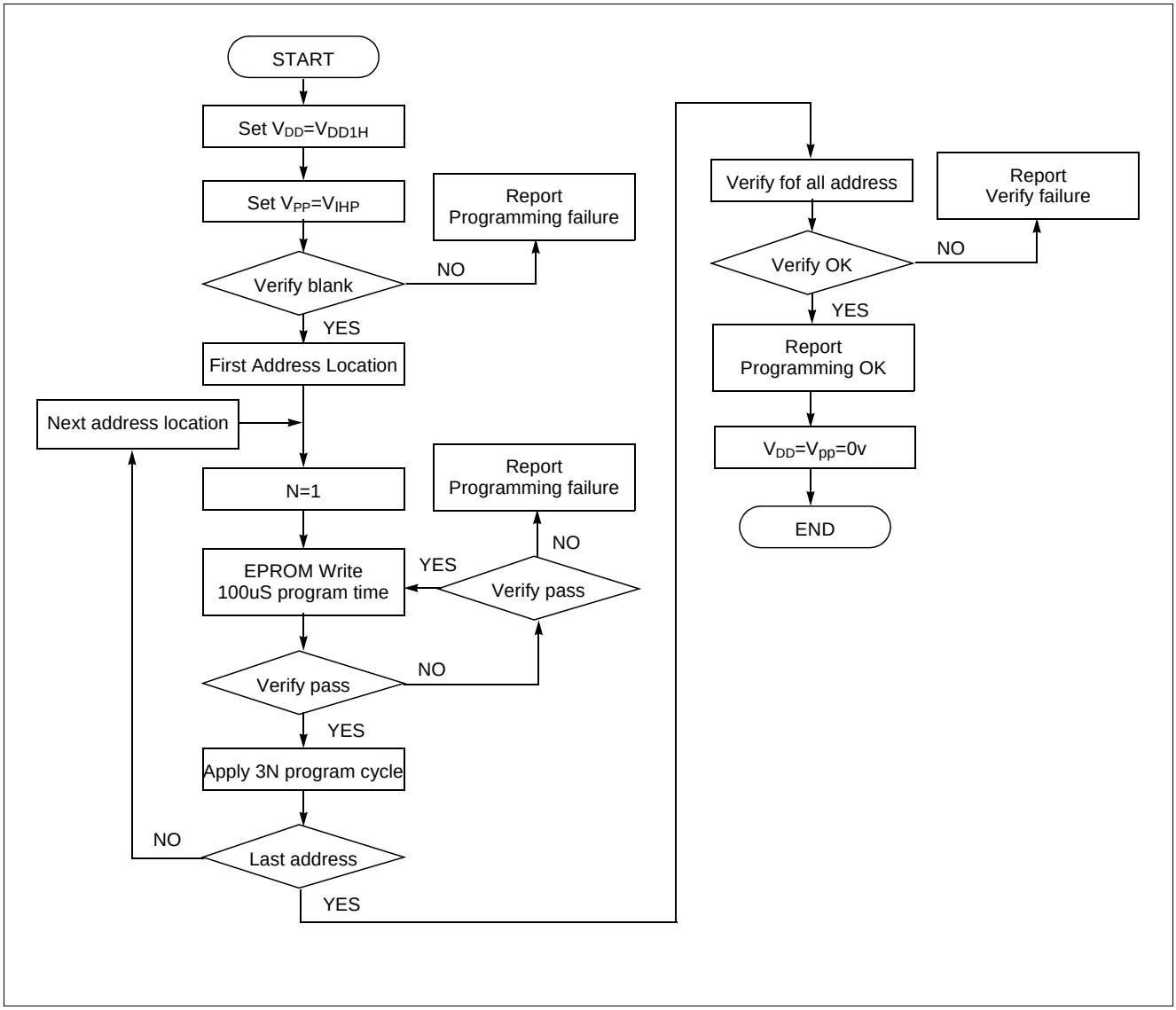


Figure 23-6 Programming Flow Chart

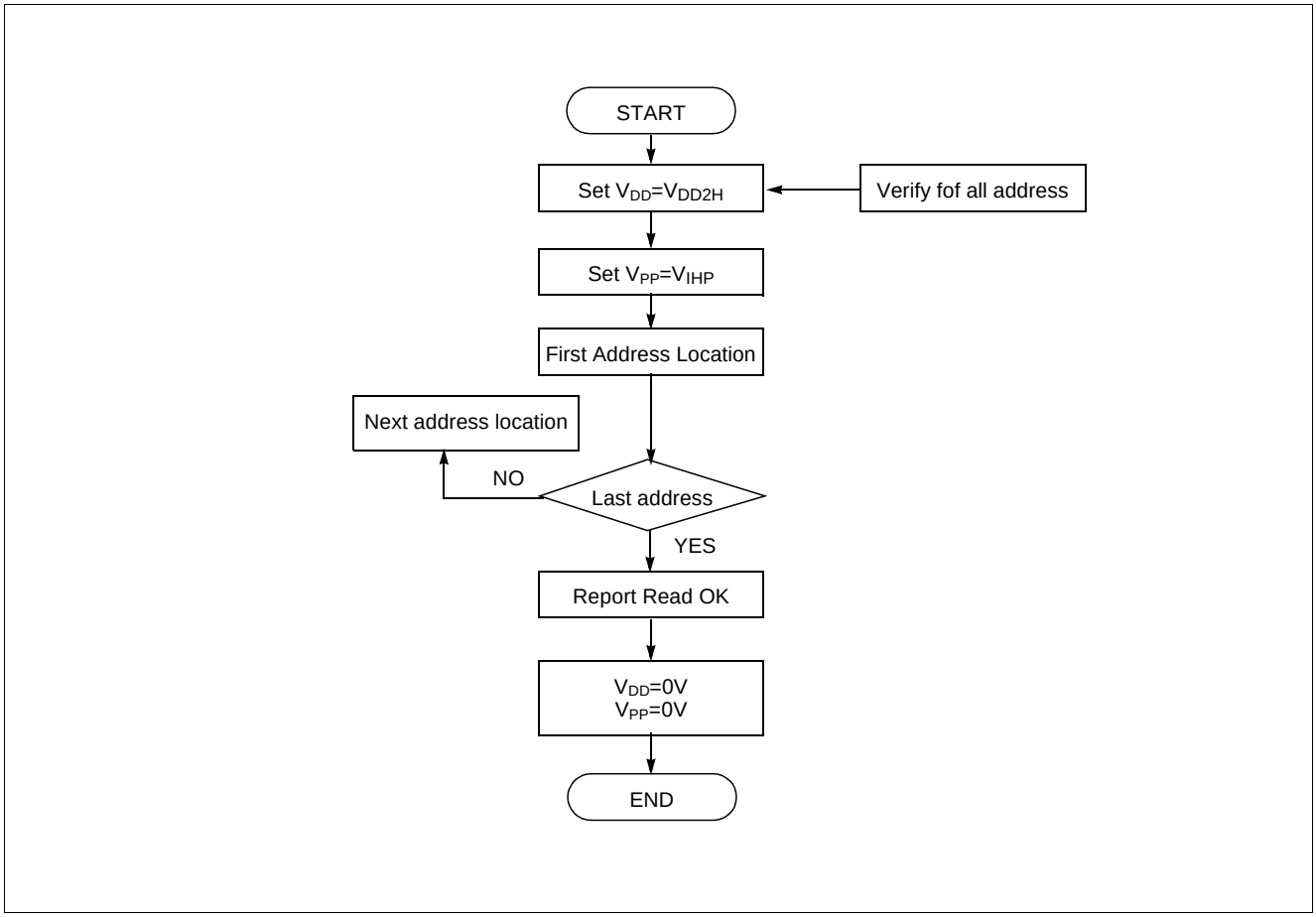


Figure 23-7 Reading Flow Chart

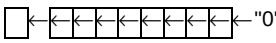
A. INSTRUCTION MAP

LOW / HIGH	00000 00	00001 01	00010 02	00011 03	00100 04	00101 05	00110 06	00111 07	01000 08	01001 09	01010 0A	01011 0B	01100 0C	01101 0D	01110 0E	01111 0F
000	-	SET1 dp.bit	BBS A.bit,rel	BBS dp.bit,rel	ADC #imm	ADC dp	ADC dp+X	ADC !abs	ASL A	ASL dp	TCALL 0	SETA1 .bit	BIT dp	POP A	PUSH A	BRK
001	CLRC				SBC #imm	SBC dp	SBC dp+X	SBC !abs	ROL A	ROL dp	TCALL 2	CLRA1 .bit	COM dp	POP X	PUSH X	BRA rel
010	CLRG				CMP #imm	CMP dp	CMP dp+X	CMP !abs	LSR A	LSR dp	TCALL 4	NOT1 M.bit	TST dp	POP Y	PUSH Y	PCALL Upage
011	DI				OR #imm	OR dp	OR dp+X	OR !abs	ROR A	ROR dp	TCALL 6	OR1 OR1B	CMPX dp	POP PSW	PUSH PSW	RET
100	CLRV				AND #imm	AND dp	AND dp+X	AND !abs	INC A	INC dp	TCALL 8	AND1 AND1B	CMPY dp	CBNE dp+X	TXSP	INC X
101	SETC				EOR #imm	EOR dp	EOR dp+X	EOR !abs	DEC A	DEC dp	TCALL 10	EOR1 EOR1B	DBNE dp	XMA dp+X	TSPX	DEC X
110	SETG				LDA #imm	LDA dp	LDA dp+X	LDA !abs	TXA	LDY dp	TCALL 12	LDC LDCB	LDX dp	LDX dp+Y	XCN	DAS
111	EI				LDM dp,#imm	STA dp	STA dp+X	STA !abs	TAX	STY dp	TCALL 14	STC M.bit	STX dp	STX dp+Y	XAX	STOP

LOW / HIGH	10000 10	10001 11	10010 12	10011 13	10100 14	10101 15	10110 16	10111 17	11000 18	11001 19	11010 1A	11011 1B	11100 1C	11101 1D	11110 1E	11111 1F
000	BPL rel	CLR1 dp.bit	BBC A.bit,rel	BBC dp.bit,rel	ADC {X}	ADC !abs+Y	ADC [dp+X]	ADC [dp]+Y	ASL !abs	ASL dp+X	TCALL 1	JMP !abs	BIT !abs	ADDW dp	LDX #imm	JMP [!abs]
001	BVC rel				SBC {X}	SBC !abs+Y	SBC [dp+X]	SBC [dp]+Y	ROL !abs	ROL dp+X	TCALL 3	CALL !abs	TEST !abs	SUBW dp	LDY #imm	JMP [dp]
010	BCC rel				CMP {X}	CMP !abs+Y	CMP [dp+X]	CMP [dp]+Y	LSR !abs	LSR dp+X	TCALL 5	MUL	TCLR1 !abs	CMPW dp	CMPX #imm	CALL [dp]
011	BNE rel				OR {X}	OR !abs+Y	OR [dp+X]	OR [dp]+Y	ROR !abs	ROR dp+X	TCALL 7	DBNE Y	CMPX !abs	LDYA dp	CMPY #imm	RETI
100	BMI rel				AND {X}	AND !abs+Y	AND [dp+X]	AND [dp]+Y	INC !abs	INC dp+X	TCALL 9	DIV	CMPY !abs	INCW dp	INC Y	TAY
101	BVS rel				EOR {X}	EOR !abs+Y	EOR [dp+X]	EOR [dp]+Y	DEC !abs	DEC dp+X	TCALL 11	XMA {X}	XMA dp	DECW dp	DEC Y	TYA
110	BCS rel				LDA {X}	LDA !abs+Y	LDA [dp+X]	LDA [dp]+Y	LDY !abs	LDY dp+X	TCALL 13	LDA {X}+	LDX !abs	STYA dp	XAY	DAA
111	BEQ rel				STA {X}	STA !abs+Y	STA [dp+X]	STA [dp]+Y	STY !abs	STY dp+X	TCALL 15	STA {X}+	STX !abs	CBNE dp	XYX	NOP

B. INSTRUCTION SET

1. ARITHMETIC/ LOGIC OPERATION

NO.	MNEMONIC	OP CODE	BYTE NO	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	ADC #imm	04	2	2	Add with carry.	
2	ADC dp	05	2	3	A ← (A) + (M) + C	
3	ADC dp + X	06	2	4		
4	ADC !abs	07	3	4		NV--H-ZC
5	ADC !abs + Y	15	3	5		
6	ADC [dp + X]	16	2	6		
7	ADC [dp] + Y	17	2	6		
8	ADC { X }	14	1	3		
9	AND #imm	84	2	2	Logical AND	
10	AND dp	85	2	3	A ← (A) ∧ (M)	
11	AND dp + X	86	2	4		
12	AND !abs	87	3	4		N-----Z-
13	AND !abs + Y	95	3	5		
14	AND [dp + X]	96	2	6		
15	AND [dp] + Y	97	2	6		
16	AND { X }	94	1	3		
17	ASL A	08	1	2	Arithmetic shift left	
18	ASL dp	09	2	4	C 7 6 5 4 3 2 1 0	N-----ZC
19	ASL dp + X	19	2	5		
20	ASL !abs	18	3	5		
21	CMP #imm	44	2	2	Compare accumulator contents with memory contents	
22	CMP dp	45	2	3	(A) - (M)	
23	CMP dp + X	46	2	4		
24	CMP !abs	47	3	4		N-----ZC
25	CMP !abs + Y	55	3	5		
26	CMP [dp + X]	56	2	6		
27	CMP [dp] + Y	57	2	6		
28	CMP { X }	54	1	3		
29	CMPX #imm	5E	2	2	Compare X contents with memory contents	
30	CMPX dp	6C	2	3	(X) - (M)	N-----ZC
31	CMPX !abs	7C	3	4		
32	CMPY #imm	7E	2	2	Compare Y contents with memory contents	
33	CMPY dp	8C	2	3	(Y) - (M)	N-----ZC
34	CMPY !abs	9C	3	4		
35	COM dp	2C	2	4	1'S Complement : (dp) ← ~(dp)	N-----Z-
36	DAA	DF	1	3	Decimal adjust for addition	N-----ZC
37	DAS	CF	1	3	Decimal adjust for subtraction	N-----ZC
38	DEC A	A8	1	2	Decrement	N-----Z-
39	DEC dp	A9	2	4	M ← (M) - 1	
40	DEC dp + X	B9	2	5		N-----Z-
41	DEC !abs	B8	3	5		
42	DEC X	AF	1	2		
43	DEC Y	BE	1	2		
44	DIV	9B	1	12	Divide : YA / X Q: A, R: Y	NV--H-Z-

NO.	MNEMONIC	OP CODE	BYTE NO	CYCLE NO	OPERATION	FLAG NVGBHIZC
45	EOR #imm	A4	2	2	Exclusive OR	
46	EOR dp	A5	2	3	A ← (A) ⊕ (M)	
47	EOR dp + X	A6	2	4		
48	EOR !abs	A7	3	4		N-----Z-
49	EOR !abs + Y	B5	3	5		
50	EOR [dp + X]	B6	2	6		
51	EOR [dp] + Y	B7	2	6		
52	EOR { X }	B4	1	3		
53	INC A	88	1	2	Increment	N-----Z-
54	INC dp	89	2	4	M ← (M) + 1	
55	INC dp + X	99	2	5		N-----Z-
56	INC !abs	98	3	5		
57	INC X	8F	1	2		
58	INC Y	9E	1	2		
59	LSR A	48	1	2	Logical shift right	
60	LSR dp	49	2	4	7 6 5 4 3 2 1 0 C	N-----ZC
61	LSR dp + X	59	2	5	"0" →	
62	LSR !abs	58	3	5		
63	MUL	5B	1	9	Multiply : YA ← Y × A	N-----Z-
64	OR #imm	64	2	2	Logical OR	
65	OR dp	65	2	3	A ← (A) ∨ (M)	
66	OR dp + X	66	2	4		
67	OR !abs	67	3	4		N-----Z-
68	OR !abs + Y	75	3	5		
69	OR [dp + X]	76	2	6		
70	OR [dp] + Y	77	2	6		
71	OR { X }	74	1	3		
72	ROL A	28	1	2	Rotate left through carry	
73	ROL dp	29	2	4	C 7 6 5 4 3 2 1 0	N-----ZC
74	ROL dp + X	39	2	5		
75	ROL !abs	38	3	5		
76	ROR A	68	1	2	Rotate right through carry	
77	ROR dp	69	2	4	7 6 5 4 3 2 1 0 C	N-----ZC
78	ROR dp + X	79	2	5		
79	ROR !abs	78	3	5		
80	SBC #imm	24	2	2	Subtract with carry	
81	SBC dp	25	2	3	A ← (A) - (M) - ~(C)	
82	SBC dp + X	26	2	4		
83	SBC !abs	27	3	4		NV--HZC
84	SBC !abs + Y	35	3	5		
85	SBC [dp + X]	36	2	6		
86	SBC [dp] + Y	37	2	6		
87	SBC { X }	34	1	3		
88	TST dp	4C	2	3	Test memory contents for negative or zero (dp) - 00_H	N-----Z-
89	XCN	CE	1	5	Exchange nibbles within the accumulator A_7~A_4 ↔ A_3~A_0	N-----Z-

2. REGISTER / MEMORY OPERATION

NO.	MNEMONIC	OP CODE	BYTE NO	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	LDA #imm	C4	2	2	Load accumulator	
2	LDA dp	C5	2	3	A ← (M)	
3	LDA dp + X	C6	2	4		
4	LDA !abs	C7	3	4		
5	LDA !abs + Y	D5	3	5		N-----Z-
6	LDA [dp + X]	D6	2	6		
7	LDA [dp] + Y	D7	2	6		
8	LDA { X }	D4	1	3		
9	LDA { X }+	DB	1	4	X- register auto-increment : A ← (M) , X ← X + 1	
10	LDM dp,#imm	E4	3	5	Load memory with immediate data : (M) ← imm	--------
11	LDX #imm	1E	2	2	Load X-register	
12	LDX dp	CC	2	3	X ← (M)	N-----Z-
13	LDX dp + Y	CD	2	4		
14	LDX !abs	DC	3	4		
15	LDY #imm	3E	2	2	Load Y-register	
16	LDY dp	C9	2	3	Y ← (M)	N-----Z-
17	LDY dp + X	D9	2	4		
18	LDY !abs	D8	3	4		
19	STA dp	E5	2	4	Store accumulator contents in memory	
20	STA dp + X	E6	2	5	(M) ← A	
21	STA !abs	E7	3	5		
22	STA !abs + Y	F5	3	6		--------
23	STA [dp + X]	F6	2	7		
24	STA [dp] + Y	F7	2	7		
25	STA { X }	F4	1	4		
26	STA { X }+	FB	1	4	X- register auto-increment : (M) ← A, X ← X + 1	
27	STX dp	EC	2	4	Store X-register contents in memory	
28	STX dp + Y	ED	2	5	(M) ← X	--------
29	STX !abs	FC	3	5		
30	STY dp	E9	2	4	Store Y-register contents in memory	
31	STY dp + X	F9	2	5	(M) ← Y	--------
32	STY !abs	F8	3	5		
33	TAX	E8	1	2	Transfer accumulator contents to X-register : X ← A	N-----Z-
34	TAY	9F	1	2	Transfer accumulator contents to Y-register : Y ← A	N-----Z-
35	TSPX	AE	1	2	Transfer stack-pointer contents to X-register : X ← sp	N-----Z-
36	TXA	C8	1	2	Transfer X-register contents to accumulator: A ← X	N-----Z-
37	TXSP	8E	1	2	Transfer X-register contents to stack-pointer: sp ← X	N-----Z-
38	TYA	BF	1	2	Transfer Y-register contents to accumulator: A ← Y	N-----Z-
39	XAX	EE	1	4	Exchange X-register contents with accumulator :X ↔ A	--------
40	XAY	DE	1	4	Exchange Y-register contents with accumulator :Y ↔ A	--------
41	XMA dp	BC	2	5	Exchange memory contents with accumulator	
42	XMA dp+X	AD	2	6	(M) ↔ A	N-----Z-
43	XMA {X}	BB	1	5		
44	XYX	FE	1	4	Exchange X-register contents with Y-register : X ↔ Y	--------

3. 16-BIT OPERATION

NO.	MNEMONIC	OP CODE	BYTE NO	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	ADDW dp	1D	2	5	16-Bits add without carry YA ← (YA) + (dp +1) (dp)	NV--H-ZC
2	CMPW dp	5D	2	4	Compare YA contents with memory pair contents : (YA) – (dp+1)(dp)	N-----ZC
3	DECW dp	BD	2	6	Decrement memory pair (dp+1)(dp) ← (dp+1) (dp) - 1	N-----Z-
4	INCW dp	9D	2	6	Increment memory pair (dp+1) (dp) ← (dp+1) (dp) + 1	N-----Z-
5	LDYA dp	7D	2	5	Load YA YA ← (dp +1) (dp)	N-----Z-
6	STYA dp	DD	2	5	Store YA (dp +1) (dp) ← YA	--------
7	SUBW dp	3D	2	5	16-Bits substact without carry YA ← (YA) - (dp +1) (dp)	NV--H-ZC

4. BIT MANIPULATION

NO.	MNEMONIC	OP CODE	BYTE NO	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	AND1 M.bit	8B	3	4	Bit AND C-flag : C ← (C) ∧ (M .bit)	-------C
2	AND1B M.bit	8B	3	4	Bit AND C-flag and NOT : C ← (C) ∧ ~(M .bit)	-------C
3	BIT dp	0C	2	4	Bit test A with memory : Z ← (A) ∧ (M) , N ← (M_7) , V ← (M_6)	MM----Z-
4	BIT !abs	1C	3	5		
5	CLR1 dp.bit	y1	2	4	Clear bit : (M.bit) ← “0”	--------
6	CLRA1 A.bit	2B	2	2	Clear A bit : (A.bit)← “0”	--------
7	CLRC	20	1	2	Clear C-flag : C ← “0”	-------0
8	CLRG	40	1	2	Clear G-flag : G ← “0”	--0-----
9	CLRV	80	1	2	Clear V-flag : V ← “0”	-0--0---
10	EOR1 M.bit	AB	3	5	Bit exclusive-OR C-flag : C ← (C) ⊕ (M .bit)	-------C
11	EOR1B M.bit	AB	3	5	Bit exclusive-OR C-flag and NOT : C ← (C) ⊕ ~(M .bit)	-------C
12	LDC M.bit	CB	3	4	Load C-flag : C ← (M .bit)	-------C
13	LDCB M.bit	CB	3	4	Load C-flag with NOT : C ← ~(M .bit)	-------C
14	NOT1 M.bit	4B	3	5	Bit complement : (M .bit) ← ~(M .bit)	--------
15	OR1 M.bit	6B	3	5	Bit OR C-flag : C ← (C) ∨ (M .bit)	-------C
16	OR1B M.bit	6B	3	5	Bit OR C-flag and NOT : C ← (C) ∨ ~(M .bit)	-------C
17	SET1 dp.bit	x1	2	4	Set bit : (M.bit) ← “1”	--------
18	SETA1 A.bit	0B	2	2	Set A bit : (A.bit) ← “1”	--------
19	SETC	A0	1	2	Set C-flag : C ← “1”	-------1
20	SETG	C0	1	2	Set G-flag : G ← “1”	--1-----
21	STC M.bit	EB	3	6	Store C-flag : (M .bit) ← C	--------
22	TCLR1 !abs	5C	3	6	Test and clear bits with A : A - (M) , (M) ← (M) ∧ ~(A)	N-----Z-
23	TSET1 !abs	3C	3	6	Test and set bits with A : A - (M) , (M) ← (M) ∨ (A)	N-----Z-

5. BRANCH / JUMP OPERATION

NO.	MNEMONIC	OP CODE	BYTE NO	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	BBC A.bit,rel	y2	2	4/6	Branch if bit clear :	--------
2	BBC dp.bit,rel	y3	3	5/7	if (bit) = 0 , then pc ← (pc) + rel	
3	BBS A.bit,rel	x2	2	4/6	Branch if bit set :	--------
4	BBS dp.bit,rel	x3	3	5/7	if (bit) = 1 , then pc ← (pc) + rel	
5	BCC rel	50	2	2/4	Branch if carry bit clear if (C) = 0 , then pc ← (pc) + rel	--------
6	BCS rel	D0	2	2/4	Branch if carry bit set if (C) = 1 , then pc ← (pc) + rel	--------
7	BEQ rel	F0	2	2/4	Branch if equal if (Z) = 1 , then pc ← (pc) + rel	--------
8	BMI rel	90	2	2/4	Branch if minus if (N) = 1 , then pc ← (pc) + rel	--------
9	BNE rel	70	2	2/4	Branch if not equal if (Z) = 0 , then pc ← (pc) + rel	--------
10	BPL rel	10	2	2/4	Branch if minus if (N) = 0 , then pc ← (pc) + rel	--------
11	BRA rel	2F	2	4	Branch always pc ← (pc) + rel	--------
12	BVC rel	30	2	2/4	Branch if overflow bit clear if (V) = 0 , then pc ← (pc) + rel	--------
13	BVS rel	B0	2	2/4	Branch if overflow bit set if (V) = 1 , then pc ← (pc) + rel	--------
14	CALL !abs	3B	3	8	Subroutine call	
15	CALL [dp]	5F	2	8	M(sp)←(pc_H), sp←sp - 1, M(sp)← (pc_L), sp ←sp - 1, if !abs, pc← abs ; if [dp], pc_L← (dp), pc_H← (dp+1) .	--------
16	CBNE dp,rel	FD	3	5/7	Compare and branch if not equal :	--------
17	CBNE dp+X,rel	8D	3	6/8	if (A) ≠ (M) , then pc ← (pc) + rel.	
18	DBNE dp,rel	AC	3	5/7	Decrement and branch if not equal :	--------
19	DBNE Y,rel	7B	2	4/6	if (M) ≠ 0 , then pc ← (pc) + rel.	
20	JMP !abs	1B	3	3	Unconditional jump	
21	JMP [!abs]	1F	3	5	pc ← jump address	--------
22	JMP [dp]	3F	2	4		
23	PCALL upage	4F	2	6	U-page call M(sp) ←(pc_H), sp ←sp - 1, M(sp) ← (pc_L), sp ← sp - 1, pc_L ← (upage), pc_H ← ”$0FF_H$” .	--------
24	TCALL n	nA	1	8	Table call : (sp) ←(pc_H), sp ← sp - 1, M(sp) ← (pc_L),sp ← sp - 1, pc_L ← (Table vector L), pc_H ← (Table vector H)	--------

6. CONTROL OPERATION & etc.

NO.	MNEMONIC	OP CODE	BYTE NO	CYCLE NO	OPERATION	FLAG NVGBHIZC
1	BRK	0F	1	8	Software interrupt : B ← "1", M(sp) ← (pc_H), sp ←sp-1, M(s) ← (pc_L), sp ← sp - 1, M(sp) ← (PSW), sp ← sp -1, pc_L ← ($0FFDE_H$) , pc_H ← ($0FFDF_H$) .	---1-0--
2	DI	60	1	3	Disable interrupts : I ← "0"	-----0--
3	EI	E0	1	3	Enable interrupts : I ← "1"	-----1--
4	NOP	FF	1	2	No operation	--------
5	POP A	0D	1	4	sp ← sp + 1, A ← M(sp)	--------
6	POP X	2D	1	4	sp ← sp + 1, X ← M(sp)	
7	POP Y	4D	1	4	sp ← sp + 1, Y ← M(sp)	
8	POP PSW	6D	1	4	sp ← sp + 1, PSW ← M(sp)	restored
9	PUSH A	0E	1	4	M(sp) ← A , sp ← sp - 1	--------
10	PUSH X	2E	1	4	M(sp) ← X , sp ← sp - 1	
11	PUSH Y	4E	1	4	M(sp) ← Y , sp ← sp - 1	
12	PUSH PSW	6E	1	4	M(sp) ← PSW , sp ← sp - 1	
13	RET	6F	1	5	Return from subroutine sp ← sp +1, pc_L ← M(sp), sp ← sp +1, pc_H ← M(sp)	--------
14	RETI	7F	1	6	Return from interrupt sp ← sp +1, PSW ← M(sp), sp ← sp + 1, pc_L ← M(sp), sp ← sp + 1, pc_H ← M(sp)	restored
15	STOP	EF	1	3	Stop mode (halt CPU, stop oscillator)	--------