Ver 1.0

4-BIT SINGLE CHIP MICROCOMPUTERS

ADAM27PXX

USER`S MANUAL

- ADAM27P08
- ADAM27P16

ETACHIPS Co., Ltd.

1. OVERVIEW

The ADAM27PXX is remote control transmitter which uses CMOS technology.
The ADAM27PXX is suitable for remote control of TV, VCR, FANS, Air-conditioners, Audio Equipments, Toys, Games etc. The ADAM27PXX is MTP version.

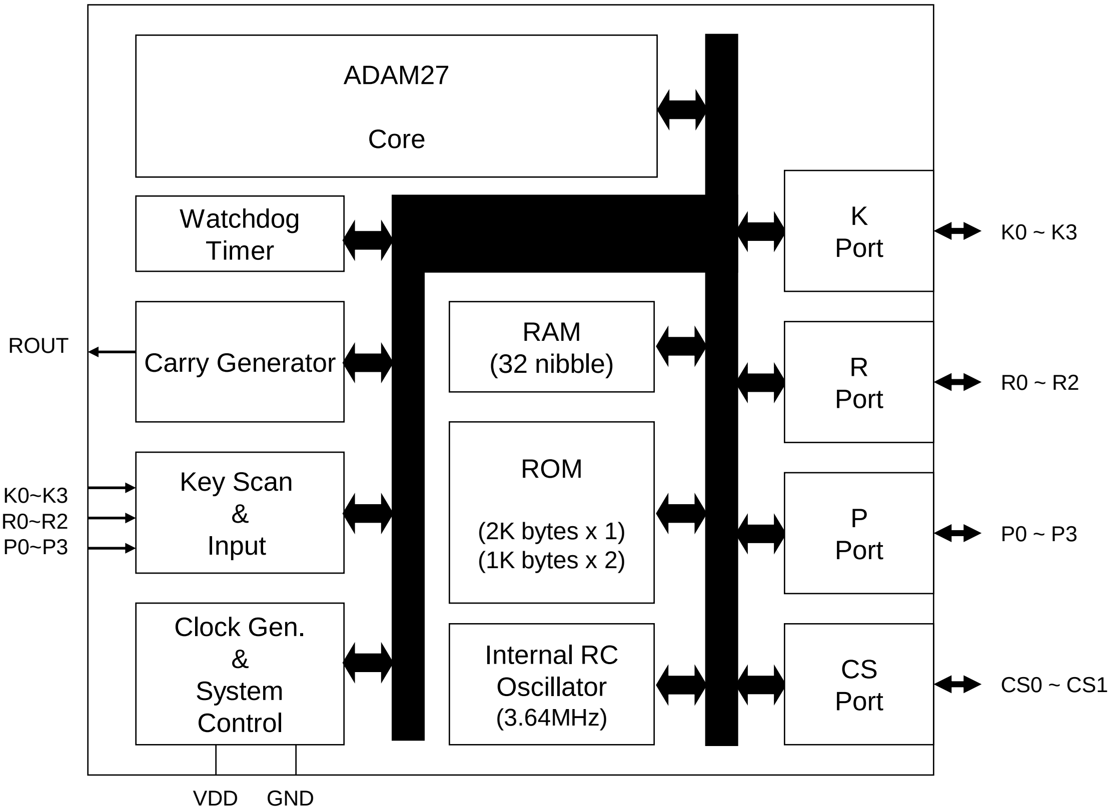
1.1. Features

- Program memory
  - 2,048 bytes (2,048 x 8bit)
  - MTP(Multi Time Programming) : 1K * 2, 2K * 1
- Data memory (RAM)
  - 32 nibble (32 x 4bit)
- 3 levels of subroutine nesting
- 8-bit Table Read Instruction
- Oscillator Type (Operating frequency)
  - Internal RC Oscillator (typically 3.64MHz)
- Instruction cycle
  - f_{OSC}/48
- Stop mode
- Released stop mode by key input
- Built in Power-on Reset circuit
- Built in Transistor for I.R LED Drive
  - I_{OL}=250mA at V_{DD}=3V and V_O=0.3V
- Built in Low Voltage reset circuit
- Built in a watch dog timer (WDT)
- Low operating voltage
  - 1.8 ~ 3.6V
- 8/16-SOP Package.

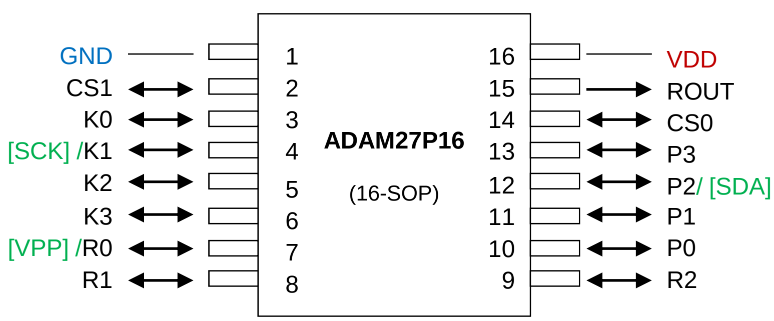
Series	ADAM27P16	ADAM27P08
Program memory	2,048 x 8	2,048 x 8
Data memory	32 x 4	32 x 4
I/O ports	13	5
Output ports	1	1
Package	16SOP(150mil)	8SOP(150mil)

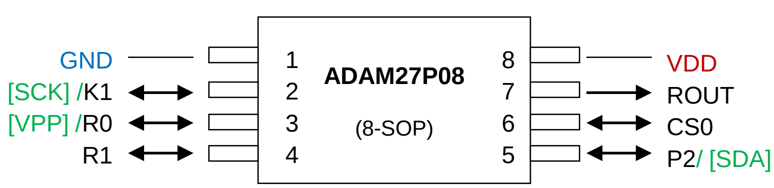
Table 1.1 ADAM27PXX series members

1.2. Block Diagram

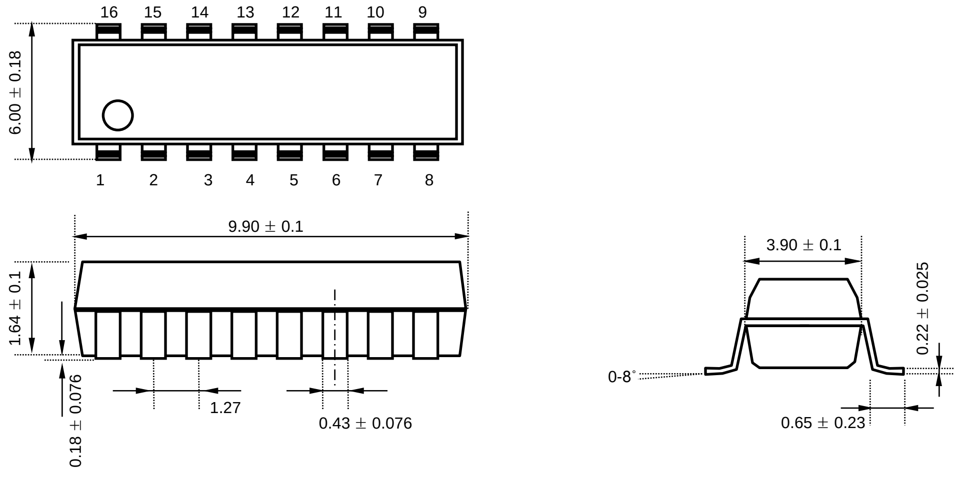


1.3. Pin Assignments (top view)



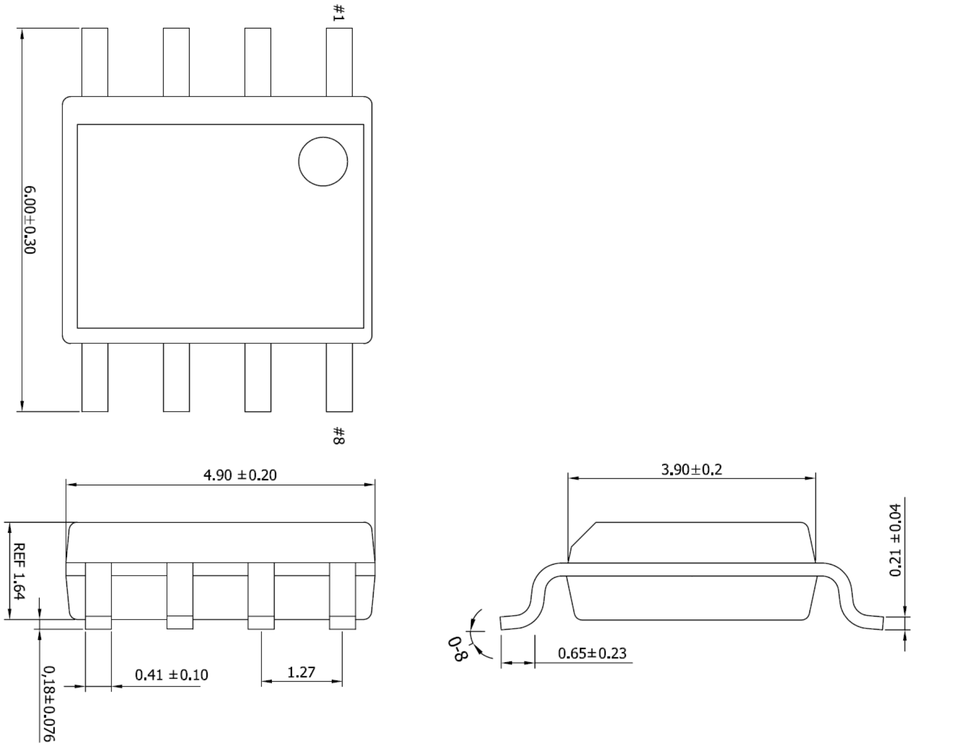


1.4. Package Dimension



Outline (Unit : mm)

16 SOP(150MIL) Pin Dimension (dimensions in millimeters)

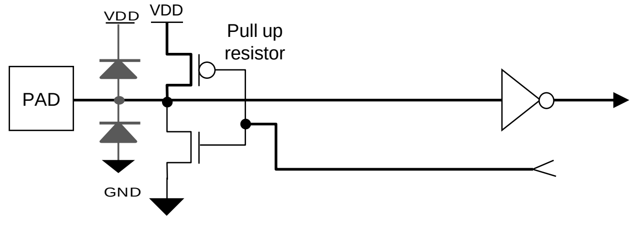
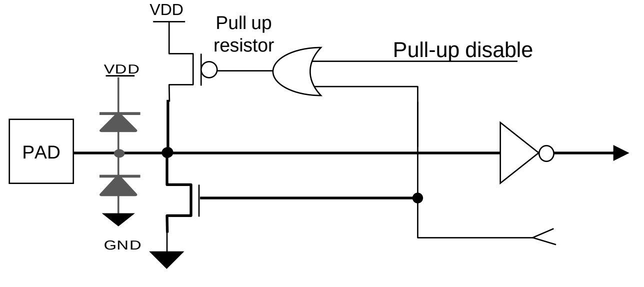
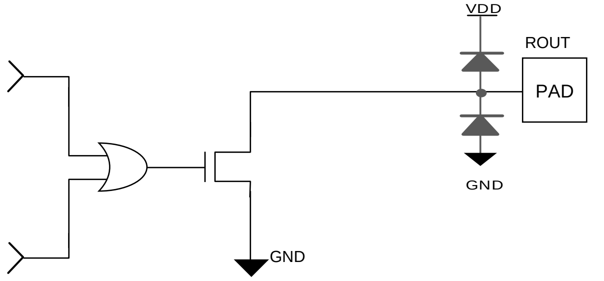


8 SOP (150MIL) Pin Dimension (dimensions in millimeters)

1.5. Pin Function

PIN NAME	INPUT OUTPUT	FUNCTION	@RESET	@STOP
K0 ~ K3 R0 ~ R2	I/O	-. 4-bit I/O port. (Input mode is set only when each of them output \`H\`) -. Each pin has STOP mode release function in input mode. -. Output mode is set when each of them output \`L\`. -. When used as \`output\`, each pin can be set and reset independently. -. When set as the input mode, input state of pin is read. At output mode, if port is read, data register is read instead of the state of pin.	Input (with Pull-up)	Key-Strobe (at T-key Scan) or Keep status Before STOP (at M-key Scan)
P0 ~ P3	I/O	-. 4-bit I/O port. (Input mode is set only when each of them output \`H\`) -. Each pin has STOP mode release function in input mode. -. Output mode is set when each of them output \`L\`. -. When used as \`output\`, each pin can be set and reset independently. -. When T-key Scan is disabled, P0~P3 are forcibly Low output at STOP mode. -. When set as the input mode, input state of pin is read. At output mode, if port is read, data register is read instead of the state of pin.	Input (with Pull-up)	Key-Strobe (at T-key Scan) or Low (at M-key Scan)
CS0~CS1	I/O	-. 2-bit I/O port. (Input mode is set only when each of them output \`H\` and pull-up is enabled.) -. Pull-ups can be enabled by user program. -. Output mode is set when each of them output \`L\`, or when it's pull-up is disabled. -. When used as \`output\`, each pin can be set and reset independently. -. When set as the input mode, input state of pin is read. At output mode, if port is read, data register is read instead of the state of pin.	Hi-Z	Keep status before STOP
ROUT	Output	-. High Current Pulse Output. -. N-ch open drain output.	Hi-Z	Hi-Z
VDD	Power	-. Positive power supply.	-	-
GND	Power	-. Ground	-	-

1.6. Pin Circuit

Pin Name	I/O	I/O circuit	Note
K0 ~ K3 R0 ~ R2	I/O		- CMOS output. - Input mode with pull-up at reset. - Built in MOS Tr. for pull-up. - In M-key scan mode, they keep the status before STOP at Stop Mode. - In T-key scan mode, they do key-strobe at STOP Mode.
P0 ~ P3	I/O		- CMOS output. - Input mode with pull-up at reset. - Built in MOS Tr. for pull-up. - In M-key scan mode, they are \`L\` output at Stop Mode. - In T-key scan mode, they do key-strobe at STOP Mode.
CS0 CS1	I/O		- CMOS output. - Open drain output at reset. - Built in MOS Tr. for pull-up. It can be enabled by user program. - Keep the status before STOP at STOP Mode.
ROUT	O		- Open drain output - Output Tr. Disable at reset and Stop Mode.

1.7. Electrical Characteristics

1.7.1. Absolute Maximum Ratings (Ta = 25℃)

Parameter	Symbol	Max. rating	Unit
Supply Voltage	V_{DD}	-0.3 ~ 5.0	V
Power dissipation	P_D	700 *	㎽
Input voltage	V_{IN}	-0.3 ~ V_{DD}+0.3	V
Output voltage	V_{OUT}	-0.3 ~ V_{DD}+0.3	V
Storage Temperature	T_{STG}	-65 ~ 150	℃

* Thermal derating above 25℃ : 6mW per degree ℃ rise in temperature.

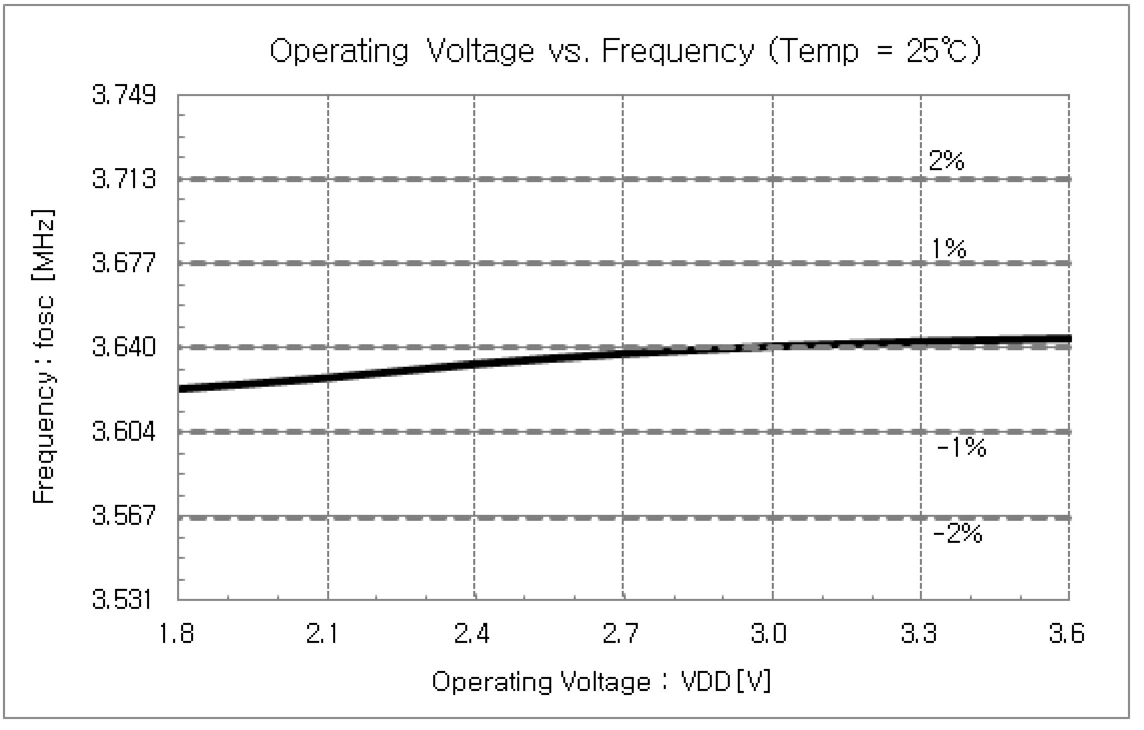
1.7.2. Recommended operating condition

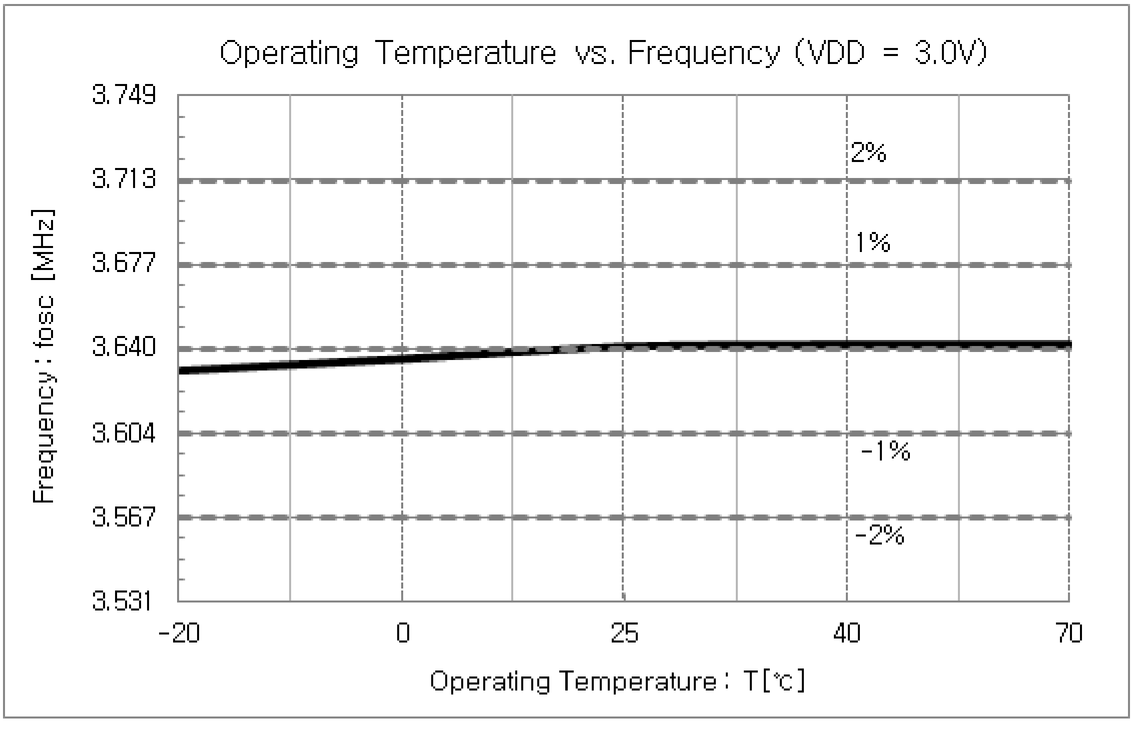
Parameter	Symbol	Condition	MIN.	TYP.	MAX.	Unit
Supply Voltage	V_{DD}	f_{OSC} = 3.64MHz	1.8	-	3.6	V
Oscillation Frequency	f_{OSC}	VDD=2.0 ~ 3.6V Temp. = 0 ~ 40℃	3.604 (-1%)	3.640	3.676 (+1%)	MHz
		VDD=2.0 ~ 3.6V Temp. = -20 ~ 70℃	3.585 (-1.5%)	3.640	3.695 (+1.5%)	MHz
		VDD=1.8 ~ 3.6V Temp. = -20 ~ 70℃	3.567 (-2.0%)	3.640	3.713 (+2.0%)	MHz
Operating temperature	T_{opr}	-	-20	-	70	℃

1.7.3. DC Characteristics (Ta = 25℃, V_{DD}=3V)

Parameter	Symbol	Limits			Unit	Condition
		Min.	Typ.	Max.		
Input H current	I_{IH}	-	-	1	㎂	V_I=V_{DD}
Input Pull-up Resistance	R_{PU}	90	150	210	㏀	V_I=GND
Input H voltage	V_{IH}	2.1	-	-	V	-
Input L voltage	V_{IL}	-	-	0.9	V	-
Output L Current	I_{OL2}	-	10	-	㎃	V_{OL}=0.6V
ROUT output L current	I_{OL1}	-	250	-	㎃	V_{OL}=0.3V
ROUT leakage current	I_{OLK1}	-	-	1	㎂	V_{OUT}=V_{DD}, Output off
Output leakage current	I_{OLK2}	-	-	1	㎂	V_{OUT}=V_{DD}, Output off
Current on STOP mode	I_{STP}	-	-	1.0	㎂	At STOP mode
Operating supply current	I_{DD}	-	0.5	1.0	㎃	f_{OSC} = 3.64MHz

※ **Internal RC Oscillator Characteristics Graphs (for reference only)**



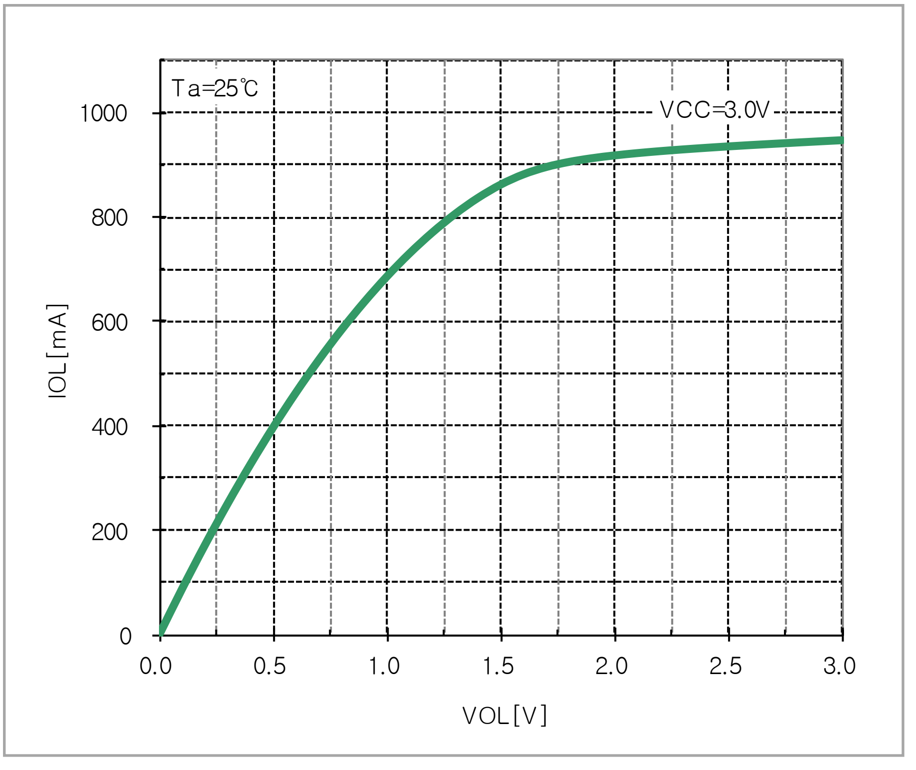


※ Typical Characteristics

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

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.

▶ **IOL vs. VOL (at T=25℃) for ROUT Port with built in Transistor.**



2. ARCHITECTURE

2.1. Program Memory

The ADAM27PXX can incorporate maximum 2,048 words (2 Block × 16 pages × 64 words × 8bits) for program memory. Program counter PC (A0~A5) , page address register PA(A6~A9) and Block address register BA(A10) are used to address the whole area of program memory having an instruction (8bits) to be next executed.
The program memory consists of 64 words on each page, and thus each page can hold up to 64 steps of instructions.
The program memory is composed as shown below.

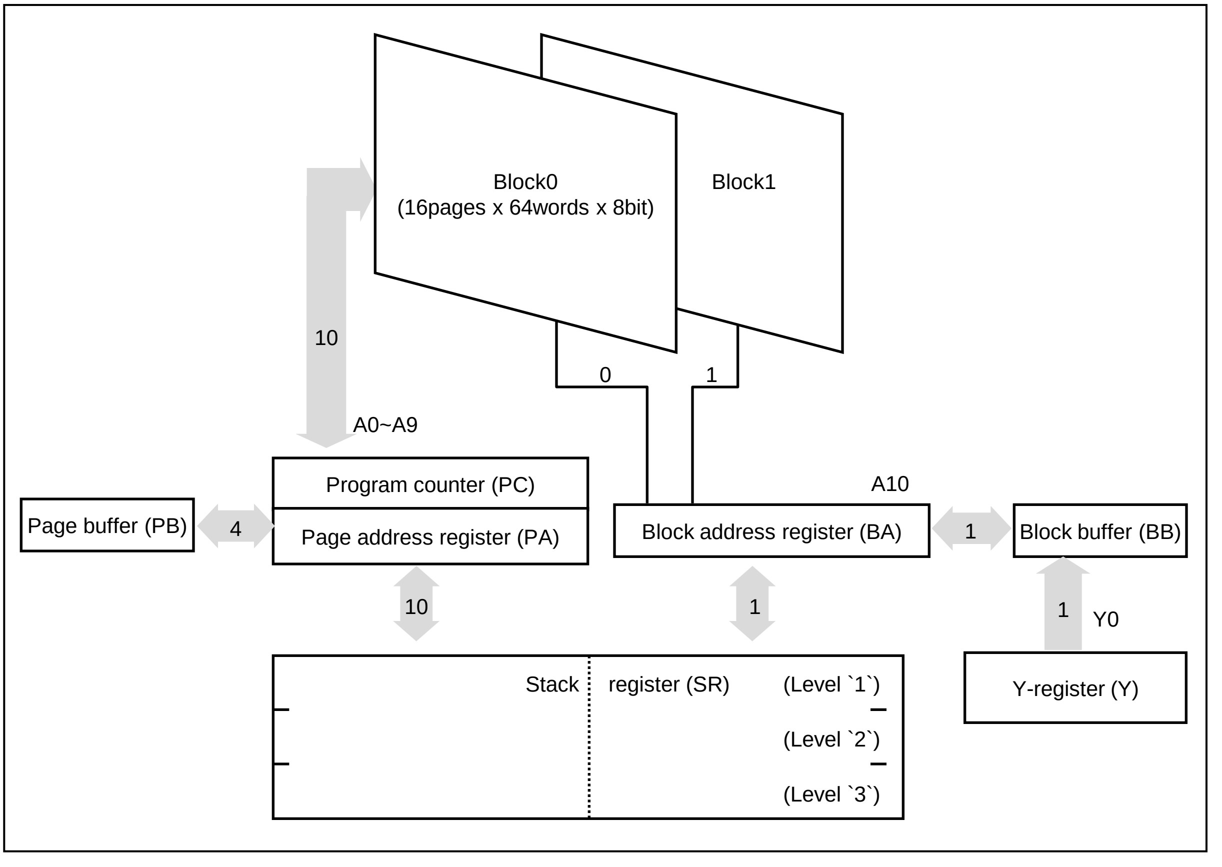


Fig 2-1 Configuration of Program Memory

2.2. Address Register

The following registers are used to address the ROM.

- Block address register (BA) :
  Holds ROM's Block number (0~1h) to be addressed.
- Block buffer register (BB) :
  Value of BB is loaded by an LBBY command when newly addressing a block.
  Then it is shifted into the BA when rightly executing a branch instruction (BR) and a subroutine call (CAL).
- Page address register (PA) :
  Holds ROM's page number (0~Fh) to be addressed.
- Page buffer register (PB) :
  Value of PB is loaded by an LPBI command when newly addressing a page.
  Then it is shifted into the PA when rightly executing a branch instruction (BR) and a subroutine call (CAL).
- Program counter (PC) :
  Available for addressing word on each page.
- Stack register (SR) :
  Stores returned-word address in the subroutine call mode.

2.2.1. Block address register and Block buffer register :

Address one of block #0 to #1 in the ROM by the 1-bit register.
Unlike the program counter, the block address register is not changed automatically.
To change the block address, take two steps such as
(1) writing in the block buffer what block to jump (execution of LBBY) and
(2) execution of BR or CAL, because instruction code is of eight bits so that block can not be specified at the same time.

In case a return instruction (RTN) is executed within the subroutine that has been called in the other block, the block address will be changed at the same time.

2.2.2. Page address register and page buffer register :

Address one of pages #0 to #15 in the ROM by the 4-bit binary counter. Unlike the program counter, the page address register is usually unchanged so that the program will repeat on the same page unless a page changing command is issued. To change the page address, take two steps such as

(1) writing in the page buffer what page to jump (execution of LPBI) and

(2) execution of BR or CAL, because instruction code is of eight bits so that page and word can not be specified at the same time.

In case a return instruction (RTN) is executed within the subroutine that has been called in the other page, the page address will be changed at the same time.

2.2.3. Program counter :

This 6-bit binary counter increments for each fetch to address a word in the currently addressed page having an instruction to be next executed. For easier programming, at turning on the power, the program counter is reset to the zero location. The PA is also set to `0`. Then the program counter specifies the next address in random sequence. When BR, CAL or RTN instructions are decoded, the switches on each step are turned off not to update the address. Then, for BR or CAL, address data are taken in from the instruction operands (a_0 to a_5), or for RTN, and address is fetched from stack register No. 1.

2.2.4. Stack register :

This stack register provides three stages each for the program counter (6bits), the page address register (4bits) and block address (1bit) so that subroutine nesting can be made on three levels.

2.3. Data Memory (RAM)

Up to 32 nibbles (16 words × 2pages × 4bits) is incorporated for storing data. The whole data memory area is indirectly specified by a data pointer (X,Y). Page number is specified by zero bit of X register, and words in the page by 4 bits in Y-register. Data memory is composed in 16 nibbles/page. Figure 2-2 shows the configuration.

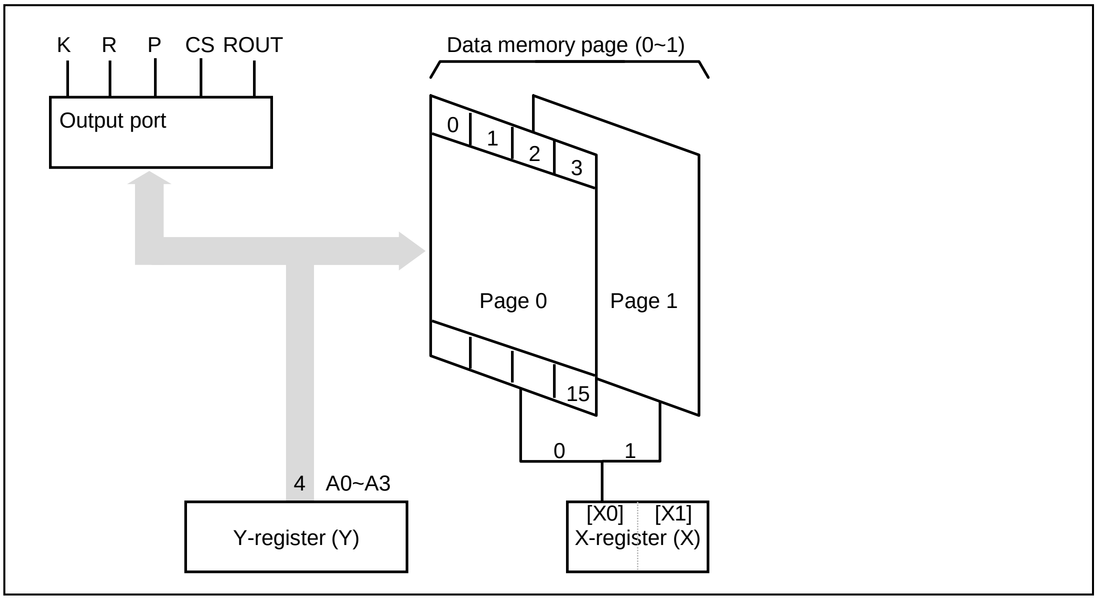


Fig 2-2 Composition of Data Memory

2.4. X-register (X)

X-register is consist of 2bit, X0 is a data pointer of page in the RAM, X1 is used for selecting the input/output of K, R, P, CS Ports with value of Y-register.

		X1 = 0	**X1 = 1**
Input Data	LAK (Instruction)	A ← K0~K3	A ← P0~P3
	LAR (Instruction)	A ← R0~R2	A ← CS0~CS1
Output Data	Y=0h~3h	K0~K3	P0~P3
	Y=4h~7h	R0~R2	CS0~CS1

Table2-1 Mapping table between X and Y register

2.5. Y-register (Y)

Y-register has 4 bits. It operates as a data pointer or a general-purpose register. Y-register specifies an address (A_0~A_3) in a page of data memory, as well as it is used to specify an output port. Further it is used to specify a mode of carrier signal outputted from the ROUT port. It can also be treated as a general-purpose register on a program.

2.6. Accumulator (A_{CC})

The 4-bit register for holding data and calculation results.

2.7. Arithmetic and Logic Unit (ALU)

In this unit, 4bits of adder/comparator are connected in parallel as it's main components and they are combined with status latch and status logic (flag.)

2.7.1. Operation circuit (ALU) :

The adder/comparator serves fundamentally for full addition and data comparison. It executes subtraction by making a complement by processing an inversed output of A_{CC} (A_{CC}+1)

2.7.2. Status logic :

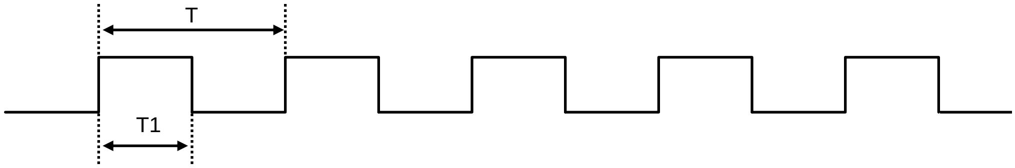
This is to bring an ST, or flag to control the flow of a program. It occurs when a specified instruction is executed in three cases such as overflow or underflow in operation and two inputs unequal.

2.8. Clock Generator

The ADAM27PXX has an internal RC oscillator which has 3.64MHz frequency only. The oscillator circuit is designed to operate without an external ceramic resonator. The Internal Oscillator is calibrate in Factory. In STOP mode, Internal oscillator is stopped.

2.9. Pulse Generator

The following frequency and duty ratio are selected for carrier signal outputted from the ROUT port depending on a PMR (Pulse Mode Register) value set in a program.



PMR	ROUT Signal		Carrier Frequency (fosc = 3.64MHz)
0	$T = 1/f_{PUL}$ = [$96/f_{OSC}$],	T1/T = 1/2	37.92 kHz
1	$T = 1/f_{PUL}$ = [$96/f_{OSC}$],	T1/T = 1/3	37.92 kHz
2	$T = 1/f_{PUL}$ = [$64/f_{OSC}$],	T1/T = 1/2	56.88 kHz
3	$T = 1/f_{PUL}$ = [$64/f_{OSC}$],	T1/T = 1/4	56.88 kHz
4	$T = 1/f_{PUL}$ = [$88/f_{OSC}$],	T1/T = 4/11	41.36 kHz
5	No Pulse (same to P0~P3)		-
6	$T = 1/f_{PUL}$ = [$101/f_{OSC}$],	T1/T = 34/101	36.04 kHz
7	$T = 1/f_{PUL}$ = [$91/f_{OSC}$],	T1/T = 31/91	40.00 kHz

* Default value is `0`

Table 2-2 PMR selection table

2.10. Reset Operation

ADAM27PXX has three reset sources. One is a built-in Low VDD Detection circuit, another is the overflow of Watch Dog Timer (WDT), the other is the overflow of Stack. All reset operations are internal in the ADAM27PXX.

2.11. Built-in Low VDD Reset Circuit

ADAM27PXX has a Low VDD detection circuit.

If VDD becomes Reset Voltage of Low VDD detection circuit in a active status, system reset occur and WDT is cleared.

When VDD is increased over Reset Voltage again, WDT is re-counted until WDT overflow, system reset is released.

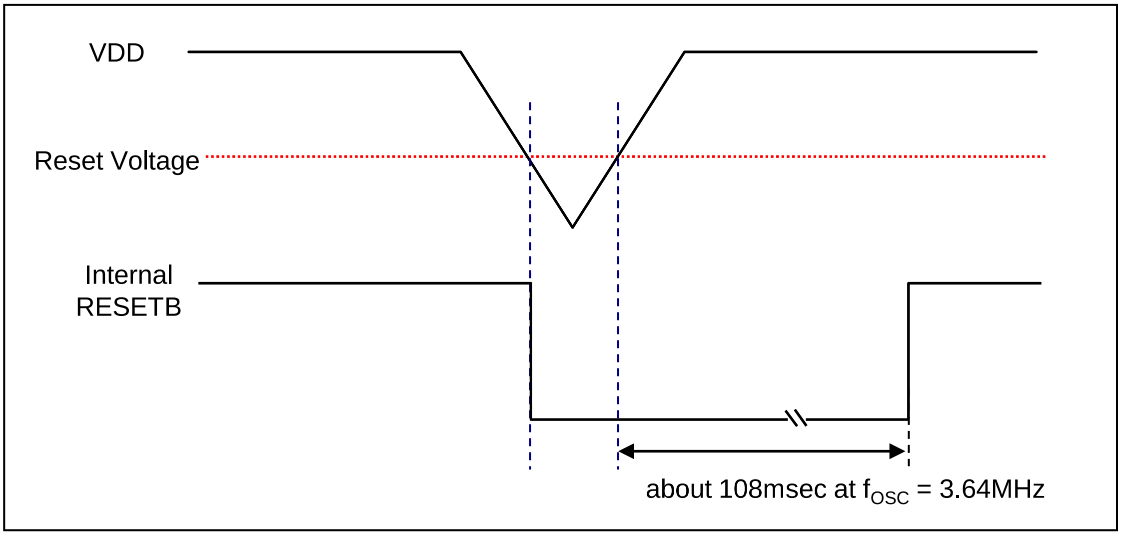


Fig 2-3 Low Voltage Detection Timing Chart.

2.12. Watch Dog Timer (WDT)

Watch dog timer is organized binary of 14 steps. The signal of $f_{OSC}/48$ cycle comes in the first step of WDT after WDT reset. If this counter was overflowed, reset signal automatically comes out so that internal circuit is initialized.

The overflow time is $8 \times 6 \times 2^{13}/f_{OSC}$ (108.026ms at f_{OSC} = 3.64MHz)

Normally, the binary counter must be reset before the overflow by using reset instruction (WDTR), Power-on reset pulse or Low VDD detection pulse.

* It is constantly reset in STOP mode. When STOP is released, counting is restarted. (Refer to 2.14. STOP Operation)

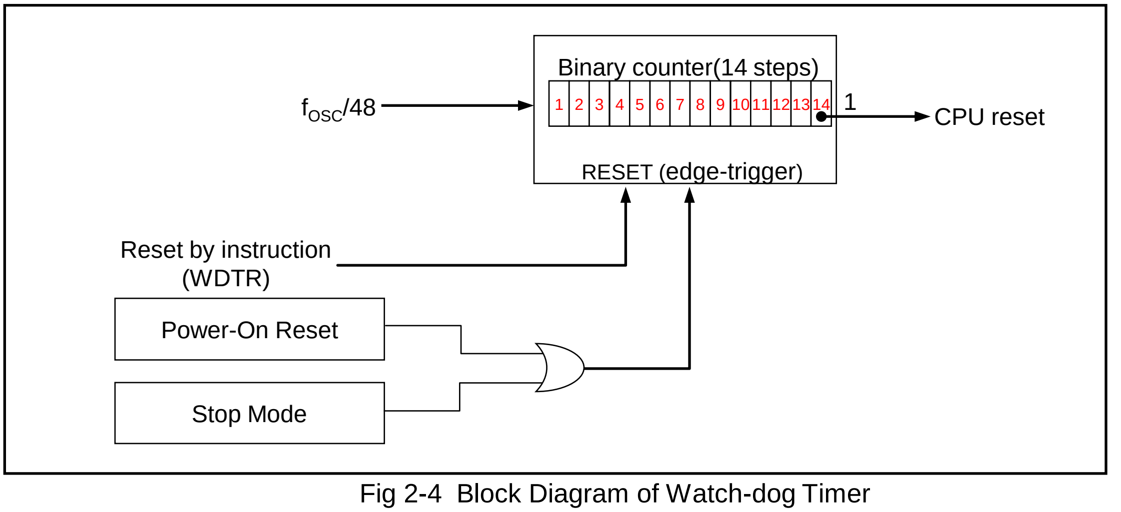


Fig 2-4 Block Diagram of Watch-dog Timer

2.13. STOP Operation

Stop mode can be achieved by STOP instructions.
In stop mode :

1. Oscillator is stopped, the operating current is low.
2. Watch dog timer is reset and ROUT output is `High-Z` .
3. Part other than WDT and ROUT output have a value before come into stop mode.
4. P0~P3 are outputted successively T-Key Scan when T-Key Scan mode is enabled, but when M-Key Scan mode is enabled, they output Low.
5. All of K, R is outputted successively T-Key Scan when T-Key Scan mode is enabled, but when M-Key Scan mode is enabled, It keeps the status before STOP. .
6. At T-Key Scan mode, before entering the STOP mode, All of K, R and P must be set the input mode with pull-up.

Stop mode is released when one of K or R or P input is going to `Low`.
When stop mode released :

1. State of K, R, P output and ROUT output is return to state of before stop mode is achieved.
2. After $8 \times 6 \times 2^{10}/f_{OSC}$ time for stable oscillating, first instruction start to operate.
3. In return to normal operation, WDT is counted from zero.

When executing stop instruction, if any one of K,R,P input is `Low` state, stop instruction is same to NOP instruction.

2.14. Port Operation

Value of X-reg	Value of Y-reg	Operation	
0 or 1	0h~3h	SO : K[Y] ← 1 (Pull-up)	RO : K[Y] ← 0
	4h~7h	SO : R[Y-4] ← 1 (Pull-up)	RO : R[Y-4] ← 0
2 or 3	0h~3h	SO : P[Y] ← 1 (Pull-up)	RO : P[Y] ← 0
	4h~7h	SO : CS[Y-4] ← 1 (Pull-up or Hi-Z)	RO : CS[Y-4] ← 0
0 or 1 or 2 or 3	8h	SO : ROUT(PMR) ← 0	RO : ROUT ← 1 (High-Z)
	9h	SO : All of P, CS ← 1	RO : All of P, CS ← 0
	Ah~Bh	SO : CS[Y-10] ← Pull-up disable	RO : CS[Y-10] ← Pull-up enable
	Eh	SO : T-Key Scan enable	RO : M-Key Scan enable
	Fh	SO : All of K,R,P,CS ← 1	RO : All of K,R,P,CS ← 0

3. INSTRUCTION

3.1. INSTRUCTION FORMAT

All of the 43 instruction in ADAM27PXX is format in two fields of OP code and operand which consist of eight bits. The following formats are available with different types of operands.

*Format Ⅰ

All eight bits are for OP code without operand.

*Format Ⅱ

Two bits are for operand and six bits for OP code.
Two bits of operand are used for specifying bits of RAM and X-register (bit 1 and bit 7 are fixed at ″0″)

*Format Ⅲ

Four bits are for operand and the others are OP code.
Four bits of operand are used for specifying a constant loaded in RAM or Y-register, a comparison value of compare command, or page addressing in ROM.

*Format Ⅳ

Six bits are for operand and the others are OP code.
Six bits of operand are used for word addressing in the ROM.

3.2. INSTRUCTION TABLE

The ADAM27PXX provides the following 43 basic instructions.

	Category	Mnemonic	Function	ST*1
1	Register to Register	LAY	A ← Y	S
2		LYA	Y ← A	S
3		LAZ	A ← 0	S
4	RAM to Register	LMA	M(X,Y) ← A	S
5		LMAIY	M(X,Y) ← A, Y ← Y+1	S
6		LYM	Y ← M(X,Y)	S
7		LAM	A ← M(X,Y)	S
8		XMA	A ↔ M(X,Y)	S
9	Immediate	LYI i	Y ← i	S
10		LMIIY i	M(X,Y) ← i, Y ← Y+1	S
11		LXI n	X ← n	S
12	RAM Bit Manipulation	SEM n	M(n) ← 1	S
13		REM n	M(n) ← 0	S
14		TM n	TEST M(n) = 1	**E**
15	ROM Address	BR a	if ST = 1 then Branch	S
16		CAL a	if ST = 1 then Subroutine call	S
17		RTN	Return from Subroutine	S
18		LPBI i	PB ← i	S
19		LBBY	BB ←Y	S
20		LDWAY	AY ← [@XAY]	S
21	Arithmetic	AM	A ← M(X,Y) + A	**C**
22		SM	A ← M(X,Y) - A	**B**
23		IM	A ← M(X,Y) + 1	**C**
24		DM	A ← M(X,Y) - 1	**B**
25		IA	A ← A + 1	S
26		IY	Y ← Y + 1	**C**
27		DA	A ← A - 1	**B**

	Category	Mnemonic	Function	ST*1
28	Arithmetic	DY	Y ← Y - 1	**B**
29		EORM	A ← A ⊕ M (X,Y)	S
30		NEGA	A ← $\overline{A}$ + 1	**Z**
31	Comparison	ALEM	TEST A ≤ M(X,Y)	**E**
32		ALEI i	TEST A ≤ i	**E**
33		MNEZ	TEST M(X,Y) ≠ 0	**N**
34		YNEA	TEST Y ≠ A	**N**
35		YNEI i	TEST Y ≠ i	**N**
36	Input / Output	LAK	A ← K (if X1=0), A ← P (if X1=1)	S
37		LAR	A ← R (if X1=0), A ← CS (if X1=1)	S
38		SO	Output(Y) ← 1*2	S
39		RO	Output(Y) ← 0*2	S
40	Control	WDTR	Watch Dog Timer Reset	S
41		STOP	Stop operation	S
42		LPY	PMR ← Y	S
43		NOP	No operation	S

Note) i = 0~f, n = 0~3, a = 6bit PC Address

*1 Column ST indicates conditions for changing status. Symbols have the following meanings

S : On executing an instruction, status is unconditionally set.
C : Status is only set when carry or borrow has occurred in operation.
B : Status is only set when borrow has not occurred in operation.
E : Status is only set when equality is found in comparison.
N : Status is only set when equality is not found in comparison.
Z : Status is only set when the result is zero.

*2 Refer to 2.14. Port Operation.

3.3. DETAILS OF INSTRUCTION SYSTEM

All 43 basic instructions of the ADAM27PXX are one by one described in detail below.

Description Form

Each instruction is headlined with its mnemonic symbol according to the instructions table given earlier.
Then, for quick reference, it is described with basic items as shown below. After that, detailed comment follows.

• Items :

- Naming : Full spelling of mnemonic symbol
- Status : Check of status function
- Format : Categorized into Ⅰ to Ⅳ
- Operand : Omitted for Format Ⅰ
- Function

(1) LAY

Naming : Load Accumulator from Y-Register
Status : Set
Format : I
Function : A ← Y
<Comment> Data of four bits in the Y-register is unconditionally transferred to the accumulator. Data in the Y-register is left unchanged.

(2) LYA

Naming : Load Y-register from Accumulator
Status : Set
Format : I
Function : Y ← A
<Comment> Load Y-register from Accumulator

(3) LAZ

Naming : Clear Accumulator
Status : Set
Format : I
Function : A ← 0
<Comment> Data in the accumulator is unconditionally reset to zero.

(4) LMA

Naming : Load Memory from Accumulator
Status : Set
Format : I
Function : M(X,Y) ← A
<Comment> Data of four bits from the accumulator is stored in the RAM location addressed by the X-register and Y-register. Such data is left unchanged.

(5) LMAIY

Naming : Load Memory from Accumulator and Increment Y-Register
Status : Set
Format : I
Function : M(X,Y) ← A, Y ← Y+1
<Comment> Data of four bits from the accumulator is stored in the RAM location addressed by the X-register and Y-register. Such data is left unchanged.

(6) LYM

Naming : Load Y-Register form Memory
Status : Set
Format : I
Function : Y ← M(X,Y)
<Comment> Data from the RAM location addressed by the X-register and Y-register is loaded into the Y-register. Data in the memory is left unchanged.

(7) LAM

Naming : Load Accumulator from Memory
Status : Set
Format : I
Function : A ← M(X,Y)
<Comment> Data from the RAM location addressed by the X-register and Y-register is loaded into the Y-register. Data in the memory is left unchanged.

(8) XMA

Naming : Exchanged Memory and Accumulator
Status : Set
Format : I
Function : M(X,Y) ↔ A
<Comment> Data from the memory addressed by X-register and Y-register is exchanged with data from the accumulator. For example, this instruction is useful to fetch a memory word into the accumulator for operation and store current data from the accumulator into the RAM. The accumulator can be restored by another XMA instruction.

(9) LYI i

Naming : Load Y-Register from Immediate
Status : Set
Format : III
Operand : Constant $0 \le i \le 15$
Function : Y ← i
<Purpose> To load a constant in Y-register. It is typically used to specify Y-register in a particular RAM word address, to specify the address of a selected output line, to set Y-register for specifying a carrier signal outputted from OUT port, and to initialize Y-register for loop control. The accumulator can be restored by another XMA instruction.
<Comment> Data of four bits from operand of instruction is transferred to the Y-register.

(10) LMIIY i

Naming : Load Memory from Immediate and Increment Y-Register
Status : Set
Format : III
Operand : Constant 0 ≤ i ≤ 15
Function : M(X,Y) ← i, Y ← Y + 1
<Comment> Data of four bits from operand of instruction is stored into the RAM location addressed by the X-register and Y-register. Then data in the Y-register is incremented by one.

(11) LXI n

Naming : Load X-Register from Immediate
Status : Set
Format : II
Operand : X file address 0 ≤ n ≤ 3
Function : X ← n
<Comment> A constant is loaded in X-register. It is used to set X-register in an index of desired RAM page. Operand of 1 bit of command is loaded in X-register.

(12) SEM n

Naming : Set Memory Bit
Status : Set
Format : II
Operand : Bit address 0 ≤ n ≤ 3
Function : M(X,Y,n) ← 1
<Comment> Depending on the selection in operand of operand, one of four bits is set as logic 1 in the RAM memory addressed in accordance with the data of the X-register and Y-register.

(13) REM n

Naming : Reset Memory Bit
Status : Set
Format : II
Operand : Bit address 0 ≤ n ≤ 3
Function : M(X,Y,n) ← 0
<Comment> Depending on the selection in operand of operand, one of four bits is set as logic 0 in the RAM memory addressed in accordance with the data of the X-register and Y-register.

(14) TM n

Naming : Test Memory Bit
Status : Comparison results to status
Format : Ⅱ
Operand : Bit address $0 \le n \le 3$
Function : M(X,Y,n) ← 1?
ST ← 1 when M(X,Y,n)=1, ST ← 0 when M(X,Y,n)=0
<Purpose> A test is made to find if the selected memory bit is logic. 1
Status is set depending on the result.

(15) BR a

Naming : Branch on status 1
Status : Conditional depending on the status
Format : Ⅳ
Operand : Branch address a (Addr)
Function : When ST =1 : BA ← BB, PA ← PB, PC ← a (Addr)
When ST = 0 : PC ← PC + 1, ST ← 1
Note : PC indicates the next address in a fixed sequence that is actually pseudo-random count.
<Purpose> For some programs, normal sequential program execution can be change.
A branch is conditionally implemented depending on the status of results obtained by executing the previous instruction.
<Comment> Branch instruction is always conditional depending on the status.
a. If the status is reset (logic 0), a branch instruction is not rightly executed but the next instruction of the sequence is executed.
b. If the status is set (logic 1), a branch instruction is executed as follows.
Branch is available in two types - short and long. The former is for addressing in the current page and the latter for addressing in other block/page.
Which type of branch to execute is decided according to the BB and PB register. To execute a long branch, data of the BB or PB register should in advance be modified to a desired block/page address through the LBBY or LPBI instruction.

(16) CAL a

Naming : Subroutine Call on status 1
Status : Conditional depending on the status
Format : Ⅳ
Operand : Subroutine code address a (Addr)
Function : When ST =1 :

PC ← a (Addr)	PA ← PB	BA ← BB
SR1 ← PC + 1	PSR1 ← PA	BSR1 ← BA
SR2 ← SR1	PSR2 ← PSR1	BSR2 ← BSR1
SR3 ← SR2	PSR3 ← PSR2	BSR3 ← BSR2

When ST = 0 :
PC ← PC + 1 PA ← PA BA ← BA ST ← 1
Note : PC actually has pseudo-random count against the next instruction.

<Comment> In a program, control is allowed to be transferred to a mutual subroutine. Since a call instruction preserves the return address, it is possible to call the subroutine from different locations in a program, and the subroutine can return control accurately to the address that is preserved by the use of the call return instruction (RTN).
Such calling is always conditional depending on the status.

a. If the status is reset, call is not executed.
b. If the status is set, call is rightly executed.

The subroutine stack (SR) of three levels enables a subroutine to be manipulated on three levels. Besides, a long call (to call another page) can be executed on any level.

For a long call, LBBY or LPBI instruction should be executed before the CAL. When LBBY or LPBI is omitted (and when BA=BB and PA=PB), a short call (calling in the same page) is executed.

(17) RTN

Naming : Return from Subroutine
Status : Set
Format : Ⅰ
Function :

PC ← SR1	PA, PB ← PSR1	BA, BB ← BSR1
SR1 ← SR2	PSR1 ← PSR2	BSR1 ← BSR2
SR2 ← SR3	PSR2 ← PSR3	BSR2 ← BSR3
SR3 ← SR3	PSR3 ← PSR3	BSR3 ← BSR3
		ST ← 1

<Purpose> Control is returned from the called subroutine to the calling program.

<Comment> Control is returned to its home routine by transferring to the PC the data of the return address that has been saved in the stack register (SR1).
At the same time, data of the page stack register (PSR1) is transferred to the PA and PB, and data of the block stack register(BSR1) is transferred to the BA and BB.

(18) LPBI i

Naming : Load Page Buffer Register from Immediate
Status : Set
Format : III
Operand : ROM page address 0 ≤ i ≤ 15
Function : PB ← i
<Purpose> A new ROM page address is loaded into the page buffer register (PB).
This loading is necessary for a long branch or call instruction.
<Comment> The PB register is loaded together with three bits from 4 bit operand.

(19) LBBY

Naming : Load Block Buffer Register from Y-register.
Status : Set
Format : I
Function : BB ← Y
<Purpose> A new ROM page address is loaded into the block buffer register (BB).
This loading is necessary for a long branch or call instruction.
<Comment> The BB register is loaded two bits(Y[1:0]) in the Y-register.
Data in the Y-register is left unchanged.

(20) LDWAY

Naming : Load Word from ROM addressed by XAY-register.
Status : Set
Format : I
Function :

SR1 ← PC + 1　　PSR1 ← PA　　BSR1 ← BA
SR2 ← SR1　　PSR2 ← PSR1　　BSR2 ← BSR1
SR3 ← SR2　　PSR3 ← PSR2　　BSR3 ← BSR2
PA,PC ← XAY(Addr)

AY ← [@XAY]
A ← MSB 4-Bit of [@XAY]
Y ← LSB 4-Bit of [@XAY]

PC ← SR1　　PA, PB ← PSR1　　BA ← BSR1
SR1 ← SR2　　PSR1 ← PSR2　　BSR1 ← BSR2
SR2 ← SR3　　PSR2 ← PSR3　　BSR2 ← BSR3
SR3 ← SR3　　PSR3 ← PSR3　　BSR3 ← BSR3

<Purpose> Data transfer from ROM to AY-register.
<Comment> The A register is loaded higher four bits in the ROM,
and the Y register is loaded lower four bits in the ROM.

(21) AM

Naming : Add Accumulator to Memory and Status 1 on Carry
Status : Carry to status
Format : I
Function : A ← M(X,Y) + A ST ← 1(when total>15),
ST ← 0 (when total ≤15)
<Comment> Data in the memory location addressed by the X and Y-register is added to data of the accumulator. Results are stored in the accumulator. Carry data as results is transferred to status. When the total is more than 15, a carry is caused to put ″1″ in the status. Data in the memory is not changed.

(22) SM

Naming : Subtract Accumulator to Memory and Status 1 Not Borrow
Status : Carry to status
Format : I
Function : A ← M(X,Y) - A ST ← 1(when A ≤ M(X,Y))
ST ← 0(when A > M(X,Y))
<Comment> Data of the accumulator is, through a 2`s complement addition, subtracted from the memory word addressed by the Y-register. Results are stored in the accumulator. If data of the accumulator is less than or equal to the memory word, the status is set to indicate that a borrow is not caused.
If more than the memory word, a borrow occurs to reset the status to ″0″.

(23) IM

Naming : Increment Memory and Status 1 on Carry
Status : Carry to status
Format : I
Function : A ← M(X,Y) + 1 ST ← 1(when M(X,Y) ≥ 15)
ST ← 0(when M(X,Y) < 15)
<Comment> Data of the memory addressed by the X and Y-register is fetched. Adding 1 to this word, results are stored in the accumulator. Carry data as results is transferred to the status. When the total is more than 15, the status is set. The memory is left unchanged.

(24) DM

Naming : Decrement Memory and Status 1 on Not Borrow
Status : Carry to status
Format : I
Function : A ← M(X,Y) - 1 ST ← 1(when M(X,Y) ≥1)
ST ← 0 (when M(X,Y) = 0)
<Comment> Data of the memory addressed by the X and Y-register is fetched, and one is subtracted from this word (addition of Fh). Results are stored in the accumulator. Carry data as results is transferred to the status. If the data is more than or equal to one, the status is set to indicate that no borrow is caused. The memory is left unchanged.

(25) IA

Naming : Increment Accumulator
Status : Set
Format : Ⅰ
Function : A ← A+1
<Comment> Data of the accumulator is incremented by one. Results are returned to the accumulator.
A carry is not allowed to have effect upon the status.

(26) IY

Naming : Increment Y-Register and Status 1 on Carry
Status : Carry to status
Format : Ⅰ
Function : Y ← Y + 1 ST ← 1 (when Y = 15)
ST ← 0 (when Y < 15)
<Comment> Data of the Y-register is incremented by one and results are returned to the Y-register.
Carry data as results is transferred to the status. When the total is more than 15, the status is set.

(27) DA

Naming : Decrement Accumulator and Status 1 on Borrow
Status : Carry to status
Format : Ⅰ
Function : A ← A - 1 ST ← 1(when A ≥1)
ST ← 0 (when A = 0)
<Comment> Data of the accumulator is decremented by one. As a result (by addition of Fh), if a borrow is caused, the status is reset to ″0″ by logic. If the data is more than one, no borrow occurs and thus the status is set to ″1″.

(28) DY

Naming : Decrement Y-Register and Status 1 on Not Borrow
Status : Carry to status
Format : I
Function : Y ← Y -1 ST ← 1 (when Y ≥ 1)
ST ← 0 (when Y = 0)
<Purpose> Data of the Y-register is decremented by one.
<Comment> Data of the Y-register is decremented by one by addition of minus 1 (Fh).
Carry data as results is transferred to the status. When the results is equal to 15, the status is set to indicate that no borrow has not occurred.

(29) EORM

Naming : Exclusive or Memory and Accumulator
Status : Set
Format : I
Function : A ← M(X,Y) ⊕ A
<Comment> Data of the accumulator is, through a Exclusive OR, subtracted from the memory word addressed by X and Y-register. Results are stored into the accumulator.

(30) NEGA

Naming : Negate Accumulator and Status 1 on Zero
Status : Carry to status
Format : I
Function : A ← $\overline{A}$ + 1 ST ← 1(when A = 0)
ST ← 0 (when A != 0)
<Purpose> The 2`s complement of a word in the accumulator is obtained.
<Comment> The 2`s complement in the accumulator is calculated by adding one to the 1`s complement in the accumulator. Results are stored into the accumulator. Carry data is transferred to the status. When data of the accumulator is zero, a carry is caused to set the status to ″1″.

(31) ALEM

Naming : Accumulator Less Equal Memory

Status : Carry to status

Format : Ⅰ

Function : A ≤ M(X,Y) ST ← 1 (when A ≤ M(X,Y))
ST ← 0 (when A > M(X,Y))

<Comment> Data of the accumulator is, through a complement addition, subtracted from data in the memory location addressed by the X and Y-register. Carry data obtained is transferred to the status. When the status is ″1″, it indicates that the data of the accumulator is less than or equal to the data of the memory word. Neither of those data is not changed.

(32) ALEI

Naming : Accumulator Less Equal Immediate

Status : Carry to status

Format : Ⅲ

Function : A ≤ i ST ← 1 (when A ≤ i)
ST ← 0 (when A > i)

<Purpose> Data of the accumulator and the constant are arithmetically compared.

<Comment> Data of the accumulator is, through a complement addition, subtracted from the constant that exists in 4bit operand. Carry data obtained is transferred to the status. The status is set when the accumulator value is less than or equal to the constant. Data of the accumulator is left unchanged.

(33) MNEZ

Naming : Memory Not Equal Zero

Status : Comparison results to status

Format : Ⅰ

Function : M(X,Y) ≠ 0 ST ← 1(when M(X,Y) ≠ 0)
ST ← 0 (when M(X,Y) = 0)

<Purpose> A memory word is compared with zero.

<Comment> Data in the memory addressed by the X and Y-register is logically compared with zero. Comparison data is transferred to the status. Unless it is zero, the status is set.

(34) YNEA

Naming :	Y-Register Not Equal Accumulator	
Status :	Comparison results to status	
Format :	I	
Function :	Y ≠ A	ST ← 1 (when Y ≠ A)
		ST ← 0 (when Y = A)
<Purpose>	Data of Y-register and accumulator are compared to check if they are not equal.	
<Comment>	Data of the Y-register and accumulator are logically compared. Results are transferred to the status. Unless they are equal, the status is set.	

(35) YNEI

Naming :	Y-Register Not Equal Immediate	
Status :	Comparison results to status	
Format :	III	
Operand :	Constant 0 ≤ i ≤ 15	
Function :	Y ≠ i	ST ← 1 (when Y ≠ i)
		ST ← 0 (when Y = i)
<Comment>	The constant of the Y-register is logically compared with 4bit operand. Results are transferred to the status. Unless the operand is equal to the constant, the status is set.	

(36) LAK

Naming :	Load Accumulator from K or P
Status :	Set
Format :	I
Function :	A ← K (when X-reg = 0 or 1) A ← P (when X-reg = 2 or 3)
<Comment>	Data on K or P are transferred to the accumulator

(37) LAR

Naming :	Load Accumulator from R or CS
Status :	Set
Format :	I
Function :	A ← R (when X-reg = 0 or 1) A ← CS (when X-reg = 2 or 3)
<Comment>	Data on R or CS are transferred to the accumulator

(38) SO

Naming :	Set Output Register Latch	
Status :	Set	
Format :	I	
Function :	K(Y) ← 1 (Pull-up)	if 0 ≤ Y ≤ 3 , X=0 or 1
	P(Y) ← 1 (Pull-up)	if 0 ≤ Y ≤ 3 , X=2 or 3
	R(Y-4) ← 1 (Pull-up)	if 4 ≤ Y ≤ 7 , X=0 or 1
	CS(Y-4) ← 1 (Pull-up or Hi-Z)	if 4 ≤ Y ≤ 7 , X=2 or 3
	ROUT ← 0 (PMR=5)	if Y = 8
	All of P, CS ← 1	if Y = 9
	Pull-up disable of CS(Y-10)	if Ah ≤ Y ≤ Bh
	T-Key Scan Enable	if Y = Eh
	All of K, R, P, CS ← 1	if Y = Fh

(43) RO

Naming :	Set Output Register Latch	
Status :	Set	
Format :	I	
Function :	K(Y) ← 0	if 0 ≤ Y ≤ 3 , X=0 or 1
	P(Y) ← 0	if 0 ≤ Y ≤ 3 , X=2 or 3
	R(Y-4) ← 0	if 4 ≤ Y ≤ 7 , X=0 or 1
	CS(Y-4) ← 0	if 4 ≤ Y ≤ 7 , X=2 or 3
	ROUT ← 1 (Hi-Z)	if Y = 8
	All of P, CS← 0	if Y = 9
	Pull-up enable of CS(Y-10)	if Ah ≤ Y ≤ Bh
	M-Key Scan Enable	if Y = Eh
	All of K, R, P, CS ← 0	if Y = Fh

(40) WDTR

Naming :	Watch Dog Timer Reset
Status :	Set
Format :	I
Function :	Reset Watch Dog Timer (WDT)
<Purpose>	Normally, you should reset this counter before overflowed counter for dc watch dog timer. this instruction controls this reset signal.

(41) STOP

Naming :	STOP
Status :	Set
Format :	I
Function :	Operate the stop function
<Purpose>	Stopped oscillator, and little current.

(42) LPY

Naming :	Pulse Mode Set
Status :	Set
Format :	I
Function :	PMR ← Y
<Comment>	Selects a pulse signal outputted from ROUT port.

(43) NOP

Naming :	No Operation
Status :	Set
Format :	I
Function :	No operation

3.4. Guideline for S/W

(1) All rams need to be initialized to any value in reset address for proper design.

(2) Make the output ports `High` after reset.

(3) Do not use WDTR instruction in subroutine.

(4) When you try to read input port changed from external condition, you must secure chattering time more than 200uS.

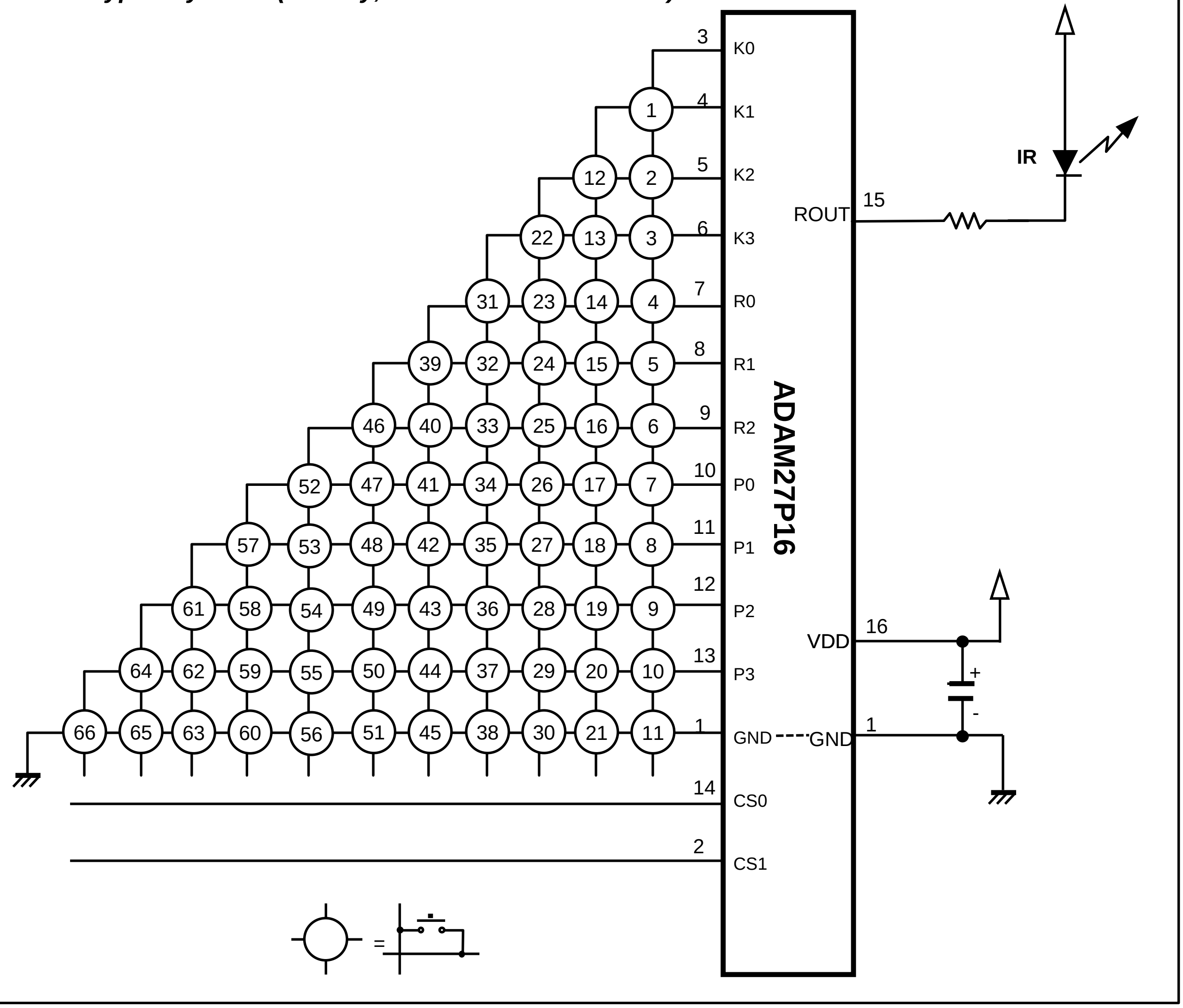
(5) To decrease current consumption, make the output port as high in normal routine except for key scan strobe and STOP mode in the M-KEY Scan mode

(6) We recommend you do not use all 64 ROM bytes in a page.
It's recommend to add `BR $` at first and last address of each page.
Do not add `BR $` at reset address which is first address of `00` page of `0` bank.

(7) `NOP` instruction should be follows STOP instruction for pre-charge time of Data Bus line.
ex) STOP : STOP instruction execution
NOP : NOP instruction

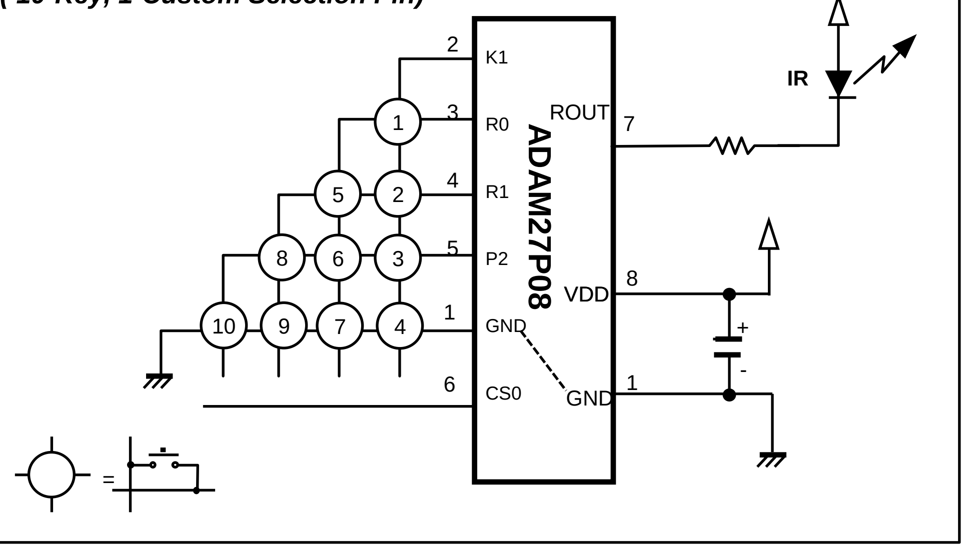
Application Circuit of ADAM27P16

T-Type Keyboard (66-Key, 2-Custom Selection Pin)



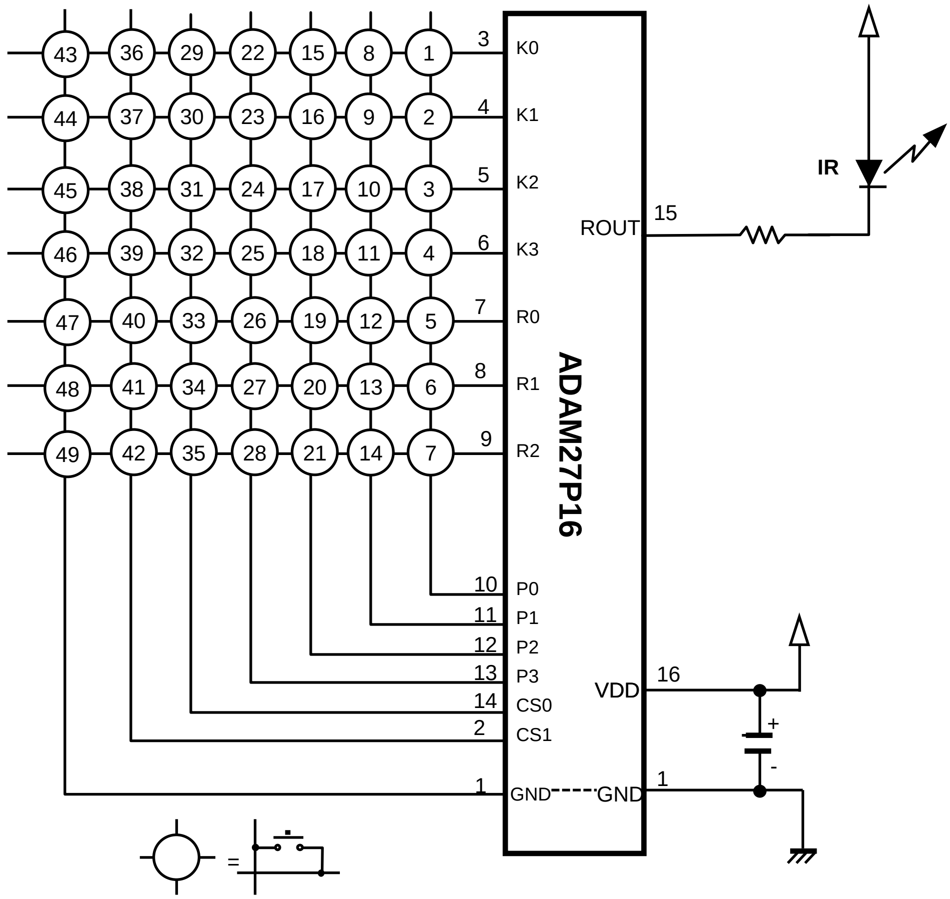
Application Circuit of ADAM27P08

T-Type Keyboard (10-Key, 1-Custom Selection Pin)



Application Circuit of ADAM27P16

M-Type Keyboard (49-Key)



Application Circuit of ADAM27P08

M-Type Keyboard (9-Key)