ADAM27PXX
USER`S MANUAL

• ADAM27P08
• ADAM27P16
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.

<table>
<thead>
<tr>
<th>Series</th>
<th>ADAM27P16</th>
<th>ADAM27P08</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program memory</td>
<td>2,048 x 8</td>
<td>2,048 x 8</td>
</tr>
<tr>
<td>Data memory</td>
<td>32 x 4</td>
<td>32 x 4</td>
</tr>
<tr>
<td>I/O ports</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>Output ports</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Package</td>
<td>16SOP(150mil)</td>
<td>8SOP(150mil)</td>
</tr>
</tbody>
</table>

Table 1.1 ADAM27PXX series members
1.2. Block Diagram

ADAM27

Core

Watchdog Timer

Carry Generator

Key Scan & Input

Clock Gen. & System Control

RAM (32 nibble)

ROM (2K bytes x 1) (1K bytes x 2)

Internal RC Oscillator (3.64MHz)

K Port

R Port

P Port

CS Port

VDD

GND

K0 ~ K3

R0 ~ R2

P0 ~ P3

CS0 ~ CS1

1.3. Pin Assignments (top view)

ADAM27P16

(16-SOP)

[CS1]

[CS0]

[R0]

[P3]

[SDA]

[VPP]

GND

1

2

3

4

5

6

7

8

16

15

14

13

12

11

10

9

VDD

ROUT

P0

P1

R1

[R1]

[R2]

[P2]

P2

[SDA]

[VPP]

ADAM27P08

(8-SOP)

GND

[SCK] / K1

K0

K2

K3

R1

[CS0]

[VPP]

R0

1

2

3

4

5

6

7

8

VDD

ROUT

P2

[SDA]
1. Overview

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

<table>
<thead>
<tr>
<th>PIN NAME</th>
<th>INPUT OUTPUT</th>
<th>FUNCTION</th>
<th>@RESET</th>
<th>@STOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>K0 ~ K3</td>
<td>I/O</td>
<td>- 4-bit I/O port. (Input mode is set only when each of them output <code>H</code>). - Each pin has STOP mode release function in input mode. - Output mode is set when each of them output <code>L</code>. - When used as <code>output</code>, 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.</td>
<td>Input (with Pull-up)</td>
<td>Key-Strobe (at T-key Scan) or Keep status before STOP (at M-key Scan)</td>
</tr>
<tr>
<td>R0 ~ R2</td>
<td>I/O</td>
<td>- 4-bit I/O port. (Input mode is set only when each of them output <code>H</code>). - Each pin has STOP mode release function in input mode. - Output mode is set when each of them output <code>L</code>. - When used as <code>output</code>, 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.</td>
<td>Input (with Pull-up)</td>
<td>Key-Strobe (at T-key Scan) or Low (at M-key Scan)</td>
</tr>
<tr>
<td>P0 ~ P3</td>
<td>I/O</td>
<td>- 2-bit I/O port. (Input mode is set only when each of them output <code>H</code> and pull-up is enabled.) - Pull-ups can be enabled by user program. - Output mode is set when each of them output <code>L</code>, or when it's pull-up is disabled. - When used as <code>output</code>, 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.</td>
<td>Hi-Z</td>
<td>Keep status before STOP</td>
</tr>
<tr>
<td>CS0~CS1</td>
<td>I/O</td>
<td>- High Current Pulse Output. - N-ch open drain output.</td>
<td>Hi-Z</td>
<td>Hi-Z</td>
</tr>
<tr>
<td>ROUT</td>
<td>Output</td>
<td>- Hi-Z</td>
<td>Hi-Z</td>
<td></td>
</tr>
<tr>
<td>VDD</td>
<td>Power</td>
<td>- Positive power supply.</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GND</td>
<td>Power</td>
<td>- Ground</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
1.6. Pin Circuit

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>I/O circuit</th>
<th>Note</th>
</tr>
</thead>
</table>
| CS0 ~ CS1 | I/O | ![Pull up resistor](#) | - 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 | ![Pull up resistor](#) | - 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. |
| ROUT | O | ![Pull up disable](#) | - Open drain output  
- Output Tr. Disable at reset and Stop Mode. |
1.7. Electrical Characteristics

1.7.1. Absolute Maximum Ratings (Ta = 25 ℃)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Max. rating</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>VDD</td>
<td>-0.3 ~ 5.0</td>
<td>V</td>
</tr>
<tr>
<td>Power dissipation</td>
<td>Pd</td>
<td>700 *</td>
<td>mW</td>
</tr>
<tr>
<td>Input voltage</td>
<td>VIN</td>
<td>-0.3 ~ VDD+0.3</td>
<td>V</td>
</tr>
<tr>
<td>Output voltage</td>
<td>VOUT</td>
<td>-0.3 ~ VDD+0.3</td>
<td>V</td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>TSTG</td>
<td>-65 ~ 150</td>
<td>℃</td>
</tr>
</tbody>
</table>

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

1.7.2. Recommended operating condition

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Condition</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>VDD</td>
<td>fosc = 3.64MHz</td>
<td>1.8</td>
<td>-</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>Oscillation Frequency</td>
<td>fosc</td>
<td>VDD=2.0 ~ 3.6V</td>
<td>3.604</td>
<td>3.640</td>
<td>3.676</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Temp. = 0 ~ 40 ℃</td>
<td>3.802</td>
<td>3.840</td>
<td>3.878</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-1%)</td>
<td>3.421</td>
<td>3.456</td>
<td>3.491</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VDD=2.0 ~ 3.6V</td>
<td>3.585</td>
<td>3.640</td>
<td>3.695</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Temp. = -20 ~ 70 ℃</td>
<td>3.782</td>
<td>3.840</td>
<td>3.898</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-1.5%)</td>
<td>3.404</td>
<td>3.456</td>
<td>3.508</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VDD=1.8 ~ 3.6V</td>
<td>3.567</td>
<td>3.640</td>
<td>3.713</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Temp. = -20 ~ 70 ℃</td>
<td>3.763</td>
<td>3.840</td>
<td>3.917</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-2.0%)</td>
<td>3.387</td>
<td>3.456</td>
<td>3.525</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VDD=1.8 ~ 3.6V</td>
<td>3.567</td>
<td>3.640</td>
<td>3.713</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Temp. = -20 ~ 70 ℃</td>
<td>3.763</td>
<td>3.840</td>
<td>3.917</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-2.0%)</td>
<td>3.387</td>
<td>3.456</td>
<td>3.525</td>
<td></td>
</tr>
<tr>
<td>Operating temperature</td>
<td>Topr</td>
<td>-</td>
<td>-20</td>
<td>-</td>
<td>70</td>
<td>℃</td>
</tr>
</tbody>
</table>

1.7.3. DC Characteristics (Ta = 25 ℃, VDD=3V)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input H current</td>
<td>Ih</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>μA</td>
<td>V_i=VDD</td>
</tr>
<tr>
<td>Input Pull-up Resistance</td>
<td>RpU</td>
<td>90</td>
<td>150</td>
<td>210</td>
<td>kΩ</td>
<td>V_i=GND</td>
</tr>
<tr>
<td>Input H voltage</td>
<td>Vih</td>
<td>2.1</td>
<td>-</td>
<td>-</td>
<td>V</td>
<td>-</td>
</tr>
<tr>
<td>Input L voltage</td>
<td>Vil</td>
<td>-</td>
<td>-</td>
<td>0.9</td>
<td>V</td>
<td>-</td>
</tr>
<tr>
<td>Output L Current</td>
<td>IOL2</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>mA</td>
<td>V_o=0.6V</td>
</tr>
<tr>
<td>ROUT output L current</td>
<td>IOL1</td>
<td>-</td>
<td>250</td>
<td>-</td>
<td>mA</td>
<td>V_o=0.3V</td>
</tr>
<tr>
<td>ROUT leakage current</td>
<td>IOLK1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>μA</td>
<td>V_o=VDD, Output off</td>
</tr>
<tr>
<td>Output leakage current</td>
<td>IOLK2</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>μA</td>
<td>V_o=VDD, Output off</td>
</tr>
<tr>
<td>Current on STOP mode</td>
<td>ISTP</td>
<td>-</td>
<td>-</td>
<td>1.0</td>
<td>μA</td>
<td>At STOP mode</td>
</tr>
<tr>
<td>Operating supply current</td>
<td>IDD</td>
<td>-</td>
<td>0.5</td>
<td>1.0</td>
<td>mA</td>
<td>fosc = 3.64MHz</td>
</tr>
</tbody>
</table>
1. Overview

※ Internal RC Oscillator Characteristics Graphs (for reference only)

1) On Writer, Select Device as 27P16 (fosc = 3.64MHz)

- Operating Voltage vs. Frequency (Temp = 25°C)
- Operating Temperature vs. Frequency (VDD = 3.0V)
2) **On Writer, Select Device as 27P16 40kHz (fosc = 3.84MHz)**

**Operating Voltage vs. Frequency (Temp = 25°C)**

**Operating Temperature vs. Frequency (VDD = 3.0V)**
3) **On Writer, Select Device as 27P16_36kHz (fosc = 3.456MHz)**

![Graph 1: Operating Voltage vs. Frequency (Temp = 25°C)]

![Graph 2: Operating Temperature vs. Frequency (VDD = 3.0V)]
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.

![Graph showing IOL vs. VOL for ROUT Port](image_url)

The table is made by measuring 100 samples on a same test board with oscilloscope and current probe. The measured value is the peak value of the current. Min is the smallest current value and Max is the largest current value in 100 samples. Avg. is the average current of 100 samples. Refer to appendix for more detail information.

<table>
<thead>
<tr>
<th>ROUT current probing Circuit</th>
<th>IR LED : SI-153T(AUK)</th>
</tr>
</thead>
</table>
| Measuring tool : Oscilloscope(Lecroy Wavesufer 454) and current probe(Lecroy AP015) | }

### Table of real measurement ROUT Current with IR LED

<table>
<thead>
<tr>
<th>Voltage</th>
<th>-10℃ Min</th>
<th>-10℃ Max</th>
<th>-10℃ Avg</th>
<th>0℃ Min</th>
<th>0℃ Max</th>
<th>0℃ Avg</th>
<th>20℃ Min</th>
<th>20℃ Max</th>
<th>20℃ Avg</th>
<th>50℃ Min</th>
<th>50℃ Max</th>
<th>50℃ Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8v</td>
<td>167</td>
<td>197</td>
<td>184</td>
<td>167</td>
<td>196</td>
<td>182</td>
<td>157</td>
<td>183</td>
<td>173</td>
<td>145</td>
<td>186</td>
<td>168</td>
</tr>
<tr>
<td>3.0v</td>
<td>486</td>
<td>545</td>
<td>524</td>
<td>472</td>
<td>549</td>
<td>522</td>
<td>479</td>
<td>573</td>
<td>521</td>
<td>401</td>
<td>497</td>
<td>455</td>
</tr>
<tr>
<td>3.6v</td>
<td>664</td>
<td>739</td>
<td>706</td>
<td>611</td>
<td>694</td>
<td>658</td>
<td>558</td>
<td>683</td>
<td>615</td>
<td>561</td>
<td>697</td>
<td>625</td>
</tr>
</tbody>
</table>

Unit : mA

Note: The table is made by measuring 100 samples on a same test board with oscilloscope and current probe. The measured value is the peak value of the current. Min is the smallest current value and Max is the largest current value in 100 samples. Avg. is the average current of 100 samples. Refer to appendix for more detail information.
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 cannot 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 (a₀ to a₅), 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 × 2 pages × 4 bits) 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.

![Diagram of Data Memory Configuration](image)

---

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.

<table>
<thead>
<tr>
<th>Input Data</th>
<th>X1 = 0</th>
<th>X1 = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAK (Instruction)</td>
<td>A ← K0-K3</td>
<td>A ← P0-P3</td>
</tr>
<tr>
<td>LAR (Instruction)</td>
<td>A ← R0-R2</td>
<td>A ← CS0-CS1</td>
</tr>
<tr>
<td>Output Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y=0h~3h</td>
<td>K0-K3</td>
<td>P0-P3</td>
</tr>
<tr>
<td>Y=4h~7h</td>
<td>R0-R2</td>
<td>CS0-CS1</td>
</tr>
</tbody>
</table>

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, 4 bits of adder/comparator are connected in parallel as its 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.

![Pulse Generator Diagram]

<table>
<thead>
<tr>
<th>PMR</th>
<th>ROUT Signal</th>
<th>Carrier Frequency ( f_{osc} = 3.64\text{MHz} )</th>
<th>Carrier Frequency ( f_{osc} = 3.84\text{MHz} )</th>
<th>Carrier Frequency ( f_{osc} = 3.456\text{MHz} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( T = 1/\text{PUL} = [96/\text{fosc}] ), T1/T = 1/2</td>
<td>37.92 kHz</td>
<td>40 kHz</td>
<td>36 kHz</td>
</tr>
<tr>
<td>1</td>
<td>( T = 1/\text{PUL} = [96/\text{fosc}] ), T1/T = 1/3</td>
<td>37.92 kHz</td>
<td>40 kHz</td>
<td>36 kHz</td>
</tr>
<tr>
<td>2</td>
<td>( T = 1/\text{PUL} = [64/\text{fosc}] ), T1/T = 1/2</td>
<td>56.88 kHz</td>
<td>60 kHz</td>
<td>54 kHz</td>
</tr>
<tr>
<td>3</td>
<td>( T = 1/\text{PUL} = [64/\text{fosc}] ), T1/T = 1/4</td>
<td>56.88 kHz</td>
<td>60 kHz</td>
<td>54 kHz</td>
</tr>
<tr>
<td>4</td>
<td>( T = 1/\text{PUL} = [88/\text{fosc}] ), T1/T = 4/11</td>
<td>41.36 kHz</td>
<td>43.63 kHz</td>
<td>39.27 kHz</td>
</tr>
<tr>
<td>5</td>
<td>No Pulse (same to P0-P3)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>( T = 1/\text{PUL} = [101/\text{fosc}] ), T1/T = 34/101</td>
<td>36.04 kHz</td>
<td>38.02 kHz</td>
<td>34.22 kHz</td>
</tr>
<tr>
<td>7</td>
<td>( T = 1/\text{PUL} = [91/\text{fosc}] ), T1/T = 46/91</td>
<td>40.00 kHz</td>
<td>42.2 kHz</td>
<td>37.98 kHz</td>
</tr>
</tbody>
</table>

Default value is ‘0’

Table 2-2 PMR selection table

<table>
<thead>
<tr>
<th>PMR</th>
<th>ROUT Signal</th>
<th>Carrier Frequency ( f_{osc} = 3.64\text{MHz} )</th>
<th>Carrier Frequency ( f_{osc} = 3.84\text{MHz} )</th>
<th>Carrier Frequency ( f_{osc} = 3.456\text{MHz} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( T = 1/\text{PUL} = [96/\text{fosc}] ), T1/T = 1/2</td>
<td>37.92 kHz</td>
<td>40 kHz</td>
<td>36 kHz</td>
</tr>
<tr>
<td>1</td>
<td>( T = 1/\text{PUL} = [96/\text{fosc}] ), T1/T = 1/3</td>
<td>37.92 kHz</td>
<td>40 kHz</td>
<td>36 kHz</td>
</tr>
<tr>
<td>2</td>
<td>( T = 1/\text{PUL} = [64/\text{fosc}] ), T1/T = 1/2</td>
<td>56.88 kHz</td>
<td>60 kHz</td>
<td>54 kHz</td>
</tr>
<tr>
<td>3</td>
<td>( T = 1/\text{PUL} = [64/\text{fosc}] ), T1/T = 1/4</td>
<td>56.88 kHz</td>
<td>60 kHz</td>
<td>54 kHz</td>
</tr>
<tr>
<td>4</td>
<td>( T = 1/\text{PUL} = [88/\text{fosc}] ), T1/T = 4/11</td>
<td>41.36 kHz</td>
<td>43.63 kHz</td>
<td>39.27 kHz</td>
</tr>
<tr>
<td>5</td>
<td>No Pulse (same to P0-P3)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>( T = 1/\text{PUL} = [101/\text{fosc}] ), T1/T = 34/101</td>
<td>36.04 kHz</td>
<td>38.02 kHz</td>
<td>34.22 kHz</td>
</tr>
<tr>
<td>7</td>
<td>( T = 1/\text{PUL} = [91/\text{fosc}] ), T1/T = 46/91</td>
<td>40.00 kHz</td>
<td>42.2 kHz</td>
<td>37.98 kHz</td>
</tr>
</tbody>
</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 an active status, system reset occurs, 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.](image)

2.12. Watch Dog Timer (WDT)

Watch dog timer is organized binary of 14 steps. The signal of f\text{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 \(8 \times 6 \times 2^{13}/f_{\text{OSC}}\) (108.026ms at \(f_{\text{OSC}} = 3.64\text{MHz}\)). 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](image)
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 $8 \times 6 \times 2^{10}/\text{fosc}$ 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

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Mnemonic</th>
<th>Function</th>
<th>ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register to Register</td>
<td>LAY</td>
<td>A ← Y</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>LYA</td>
<td>Y ← A</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>LAZ</td>
<td>A ← 0</td>
<td>S</td>
</tr>
<tr>
<td>RAM to Register</td>
<td>LMA</td>
<td>M(X,Y) ← A</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>LMAIY</td>
<td>M(X,Y) ← A, Y ← Y+1</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>LYM</td>
<td>Y ← M(X,Y)</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>LAM</td>
<td>A ← M(X,Y)</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>XMA</td>
<td>A ↔ M(X,Y)</td>
<td>S</td>
</tr>
<tr>
<td>Immediate</td>
<td>LYI i</td>
<td>Y ← i</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>LMIiY i</td>
<td>M(X,Y) ← i, Y ← Y+1</td>
<td>S</td>
</tr>
<tr>
<td>RAM Bit Manipulation</td>
<td>SEM n</td>
<td>M(n) ← 1</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>REM n</td>
<td>M(n) ← 0</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>TM n</td>
<td>TEST M(n) = 1</td>
<td>E</td>
</tr>
<tr>
<td>ROM Address</td>
<td>BR a</td>
<td>if ST = 1 then Branch</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>CAL a</td>
<td>if ST = 1 then Subroutine call</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>RTN</td>
<td>Return from Subroutine</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>LPBi i</td>
<td>PB ← i</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>LBBY</td>
<td>BB ← Y</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>LDWAY</td>
<td>AY ← [@XAY]</td>
<td>S</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>AM</td>
<td>A ← M(X,Y) + A</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>SM</td>
<td>A ← M(X,Y) - A</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>IM</td>
<td>A ← M(X,Y) + 1</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>DM</td>
<td>A ← M(X,Y) - 1</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>IA</td>
<td>A ← A + 1</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>IY</td>
<td>Y ← Y + 1</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>DA</td>
<td>A ← A - 1</td>
<td>B</td>
</tr>
</tbody>
</table>

*ST*: S = Single, E = Extended
<table>
<thead>
<tr>
<th>Category</th>
<th>Mnemonic</th>
<th>Function</th>
<th>ST*1</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>DY</td>
<td>Y ← Y - 1</td>
<td>B</td>
</tr>
<tr>
<td>29</td>
<td>EORM</td>
<td>A ← A ⊕ M(X,Y)</td>
<td>S</td>
</tr>
<tr>
<td>30</td>
<td>NEGA</td>
<td>A ← ̅A + 1</td>
<td>Z</td>
</tr>
<tr>
<td>31</td>
<td>ALEM</td>
<td>TEST A ≤ M(X,Y)</td>
<td>E</td>
</tr>
<tr>
<td>32</td>
<td>ALEi i</td>
<td>TEST A ≤ i</td>
<td>E</td>
</tr>
<tr>
<td>33</td>
<td>MNEZ</td>
<td>TEST M(X,Y) ≠ 0</td>
<td>N</td>
</tr>
<tr>
<td>34</td>
<td>YNEA</td>
<td>TEST Y ≠ A</td>
<td>N</td>
</tr>
<tr>
<td>35</td>
<td>YNEi i</td>
<td>TEST Y ≠ i</td>
<td>N</td>
</tr>
<tr>
<td>36</td>
<td>LAK</td>
<td>A ← K (if X1=0), A ← P (if X1=1)</td>
<td>S</td>
</tr>
<tr>
<td>37</td>
<td>LAR</td>
<td>A ← R (if X1=0), A ← CS (if X1=1)</td>
<td>S</td>
</tr>
<tr>
<td>38</td>
<td>SO</td>
<td>Output(Y) ← 1^2</td>
<td>S</td>
</tr>
<tr>
<td>39</td>
<td>RO</td>
<td>Output(Y) ← 0^2</td>
<td>S</td>
</tr>
<tr>
<td>40</td>
<td>WDTR</td>
<td>Watch Dog Timer Reset</td>
<td>S</td>
</tr>
<tr>
<td>41</td>
<td>STOP</td>
<td>Stop operation</td>
<td>S</td>
</tr>
<tr>
<td>42</td>
<td>LPY</td>
<td>PMR ← Y</td>
<td>S</td>
</tr>
<tr>
<td>43</td>
<td>NOP</td>
<td>No operation</td>
<td>S</td>
</tr>
</tbody>
</table>

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
3. Instruction

(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 from Memory
Status : Set
Format : I
Function : \(Y \leftarrow M(X,Y)\)
<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 : \(A \leftarrow M(X,Y)\)
<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 : \(M(X,Y) \leftrightarrow A\)
<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 \(0 \leq i \leq 15\)
Function : \(Y \leftarrow i\)
<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 \(0 \leq n \leq 3\)
Function :
\[ M(X,Y,n) \leftarrow 1? \]
\[ ST \leftarrow 1 \text{ when } M(X,Y,n)=1, \ ST \leftarrow 0 \text{ when } M(X,Y,n)=0 \]
<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 \(ST = 1\) : \(BA \leftarrow BB, \ PA \leftarrow PB, \ PC \leftarrow a \) (Addr)
When \(ST = 0\) : \(PC \leftarrow PC + 1, \ ST \leftarrow 1\)
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.
3. 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 :

<table>
<thead>
<tr>
<th>PC</th>
<th>PA</th>
<th>BA</th>
</tr>
</thead>
<tbody>
<tr>
<td>a (Addr)</td>
<td>PB</td>
<td>BB</td>
</tr>
<tr>
<td>SR1</td>
<td>PC + 1</td>
<td>PSR1</td>
</tr>
<tr>
<td>SR2</td>
<td>SR1</td>
<td>PSR2</td>
</tr>
<tr>
<td>SR3</td>
<td>SR2</td>
<td>PSR3</td>
</tr>
</tbody>
</table>

When ST = 0 :

<table>
<thead>
<tr>
<th>PC</th>
<th>PA</th>
<th>BA</th>
<th>ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC + 1</td>
<td>PA</td>
<td>BA</td>
<td>1</td>
</tr>
</tbody>
</table>

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 :

<table>
<thead>
<tr>
<th>PC</th>
<th>PA, PB</th>
<th>BA, BB</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR1</td>
<td>SR2</td>
<td>SR3</td>
</tr>
<tr>
<td>PSR1</td>
<td>PSR2</td>
<td>PSR3</td>
</tr>
<tr>
<td>BSR1</td>
<td>BSR2</td>
<td>BSR3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<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 $0 \leq i \leq 15$
Function : $PB \leftarrow i$

**<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 : $BB \leftarrow Y$

**<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 :**

$$
\begin{align*}
SR1 & \leftarrow PC + 1 & \text{PSR1} & \leftarrow PA & \text{BSR1} & \leftarrow BA \\
SR2 & \leftarrow SR1 & \text{PSR2} & \leftarrow PSR1 & \text{BSR2} & \leftarrow BSR1 \\
SR3 & \leftarrow SR2 & \text{PSR3} & \leftarrow PSR2 & \text{BSR3} & \leftarrow BSR2 \\
\text{PA,PC} & \leftarrow \text{XAY(Addr)} \\
\text{AY} & \leftarrow [@XAY] \\
\text{A} & \leftarrow \text{MSB 4-Bit of } [@XAY] \\
\text{Y} & \leftarrow \text{LSB 4-Bit of } [@XAY] \\
\text{PC} & \leftarrow \text{SR1} & \text{PA, PB} & \leftarrow \text{PSR1} & \text{BA} & \leftarrow \text{BSR1} \\
\text{SR1} & \leftarrow \text{SR2} & \text{PSR1} & \leftarrow \text{PSR2} & \text{BSR1} & \leftarrow \text{BSR2} \\
\text{SR2} & \leftarrow \text{SR3} & \text{PSR2} & \leftarrow \text{PSR3} & \text{BSR2} & \leftarrow \text{BSR3} \\
\text{SR3} & \leftarrow \text{SR3} & \text{PSR3} & \leftarrow \text{PSR3} & \text{BSR3} & \leftarrow \text{BSR3}
\end{align*}
$$

**<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 :
Function : \[ A \leftarrow M(X,Y) + A \quad \text{ST} \leftarrow 1(\text{when total}>15), \]
\[ \text{ST} \leftarrow 0(\text{when total} \leq 15) \]
<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 :
Function : \[ A \leftarrow M(X,Y) - A \quad \text{ST} \leftarrow 1(\text{when } A \leq M(X,Y)) \]
\[ \text{ST} \leftarrow 0(\text{when } A > M(X,Y)) \]
<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 :
Function : \[ A \leftarrow M(X,Y) + 1 \quad \text{ST} \leftarrow 1(\text{when } M(X,Y) \geq 15) \]
\[ \text{ST} \leftarrow 0(\text{when } M(X,Y) < 15) \]
<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 :
Function : \[ A \leftarrow M(X,Y) - 1 \quad \text{ST} \leftarrow 1(\text{when } M(X,Y) \geq 1) \]
\[ \text{ST} \leftarrow 0(\text{when } M(X,Y) = 0) \]
<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 : |
Function : \( Y \leftarrow Y - 1 \)  
\( ST \leftarrow 1 \) (when \( Y \geq 1 \))  
\( ST \leftarrow 0 \) (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 : |
Function : \( A \leftarrow M(X,Y) \oplus A \)

<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 : |
Function : \( A \leftarrow \overline{A} + 1 \)  
\( ST \leftarrow 1 \) (when \( A = 0 \))  
\( ST \leftarrow 0 \) (when \( A \neq 0 \))

<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 : $A \leq M(X,Y)$
          ST ← 1 (when $A \leq M(X,Y)$)
          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 : $A \leq i$
          ST ← 1 (when $A \leq i$)
          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 : $M(X,Y) \neq 0$
          ST ← 1 (when $M(X,Y) \neq 0$)
          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: $Y \neq A$
<br>ST ← 1 (when $Y \neq A$)
<br>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 $0 \leq i \leq 15$
Function: $Y \neq i$
<br>ST ← 1 (when $Y \neq i$)
<br>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: $A \leftarrow K$ (when X-reg = 0 or 1)
$A \leftarrow P$ (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: $A \leftarrow R$ (when X-reg = 0 or 1)
$A \leftarrow CS$ (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:**  
**Function:**  
- **K(Y)** ← 1 (Pull-up) if \(0 \leq Y \leq 3\), \(X=0\) or 1  
- **P(Y)** ← 1 (Pull-up) if \(0 \leq Y \leq 3\), \(X=2\) or 3  
- **R(Y-4)** ← 1 (Pull-up) if \(4 \leq Y \leq 7\), \(X=0\) or 1  
- **CS(Y-4)** ← 1 (Pull-up or Hi-Z) if \(4 \leq Y \leq 7\), \(X=2\) 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 \(Ah \leq Y \leq Bh\)  
- 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:**  
**Function:**  
- **K(Y)** ← 0 if \(0 \leq Y \leq 3\), \(X=0\) or 1  
- **P(Y)** ← 0 if \(0 \leq Y \leq 3\), \(X=2\) or 3  
- **R(Y-4)** ← 0 if \(4 \leq Y \leq 7\), \(X=0\) or 1  
- **CS(Y-4)** ← 0 if \(4 \leq Y \leq 7\), \(X=2\) 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 \(Ah \leq Y \leq Bh\)  
- 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
**OTP Programming**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>User Mode</th>
<th>OTP Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>Power</td>
<td>VDD Power (typ. 5V)</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>VPP</td>
<td>General port</td>
<td>Program/Verify Power (typ. 11.5V)</td>
</tr>
<tr>
<td>SCK</td>
<td>General port</td>
<td>Serial Clock Input (open drain)</td>
</tr>
<tr>
<td>SDA</td>
<td>General port</td>
<td>Serial Data input/output (Open drain output)</td>
</tr>
</tbody>
</table>

◆ **ADAM27P16 (16SOP) Pin Assignments**

![ADAM27P16 (16SOP) Pin Assignments Diagram]

◆ **ADAM27P08 (8SOP) Pin Assignments**

![ADAM27P08 (8SOP) Pin Assignments Diagram]

◆ **Pin count is 5pin : 3pin + power(2pin)**

- DATA : SDA (1bit I/O)
- CLOCK : SCK (1bit I/O)
- VPP : VPP (1bit I/O)
- Power : VDD, GND
- N.C pin : don’t care
Pre-notice to programming the device

This device uses command based programming algorithm.

You can read configuration data when you want. You can read code memory when you want, except device protection was enabled.

Blank data is 0xff.

Configuration : OPTION0 address 4000h Read / Write Area

<table>
<thead>
<tr>
<th>bit7</th>
<th>bit6</th>
<th>bit5</th>
<th>bit4</th>
<th>bit3</th>
<th>bit2</th>
<th>bit1</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIZE[1:0]</td>
<td>LOCK[1:0]</td>
<td>RCAL[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Option Description</th>
<th>Option Value</th>
<th>Option Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIZE</td>
<td>MTP Size Definition</td>
<td>11</td>
<td>2kbytes x 1time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01</td>
<td>1’st 1kbyte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00</td>
<td>2’nd 1kbyte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>prohibited</td>
</tr>
<tr>
<td>LOCK</td>
<td>MTP Lock Definition</td>
<td>11</td>
<td>un-lock</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01</td>
<td>lock in 1’st 1kbyte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00</td>
<td>lock in 2’nd 1kbyte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>prohibited</td>
</tr>
<tr>
<td>RCAL</td>
<td>IRC Re-calibration</td>
<td>1111</td>
<td>using CAL0[7:0] in OPTION1(@0x4030)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1110</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1101</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1011</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1010</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0111</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0110</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0101</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0011</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0010</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1001</td>
<td>Using CAL2[6:0] in OPTION2(@0x4020)</td>
</tr>
</tbody>
</table>