1 PRODUCT OVERVIEW

SAM8 PRODUCT FAMILY

Samsung's SAM8 family of 8-bit single-chip CMOS microcontrollers offers a fast and efficient CPU with a wide range of integrated peripherals, in various mask-programmable ROM sizes. Analog its major CPU features are:

— Efficient register-oriented architecture
— Selectable CPU clock sources
— Idle and Stop power-down mode release by interrupt
— Built-in basic timer with watchdog function

The sophisticated interrupt structure recognizes up to eight interrupt levels. Each level can have one or more interrupt sources and vectors. Fast interrupt processing (within a minimum of six CPU clocks) can be assigned to specific interrupt levels.

KS88C6216/C6224/C6232/P6232 MICROCONTROLLERS

KS88C6216/C6224/C6232/P6232 single-chip 8-bit microcontrollers are based on the powerful SAM8 CPU architecture. The internal register file is logically expanded to increase the on-chip register space. KS88C6216/C6224/C6232/P6232 contain 16/32 K bytes of on-chip program ROM.

In line with Samsung's modular design approach, the following peripherals are integrated with the SAM8 core:

— Four programmable I/O ports (total 27 pins)
— One 8-bit basic timer for oscillation stabilization and watchdog functions
— One 8-bit general-purpose timer/counter with selectable clock sources
— One 12-bit counter with selectable clock sources, including Hsync or Csync input
— One interval timer
— PWM block with seven 8-bit PWM circuits
— Sync processor block (for Vsync and Hsync I/O, Csync input, and Clamp signal output)
— DDC and normal Multi-master IIC-bus
— 4-channel A/D converter (8-bit resolution)

KS88C6216/C6224/C6232/P6232 are a versatile microcontrollers which are ideal for use in multi-sync monitors or in general-purpose applications that require sophisticated timer/counter, PWM, sync signal processing, A/D converter, and multi-master IIC-bus support with DDC. They are available in a 42-pin SDIP or a 44-pin QFP package.

OTP

KS88C6216/C6224/C6232 microcontrollers are also available in OTP (One Time Programmable) version named, KS88P6232. KS88P6232 microcontroller has an on-chip 32-Kbyte one-time-programmable EPROM instead of masked ROM. KS88P6232 is comparable to KS88C6216/C6224/C6232, both in function and pin configuration except its ROM size.

FEATURES

CPU

- SAM8 CPU core

Memory

- 16/24/32-Kbyte internal program memory (ROM)
- 464-byte general-purpose register area

Instruction Set

- 78 instructions
- IDLE and STOP instructions added for power-down modes

Instruction Execution Time

- Minimum 500 ns (with 12 MHz CPU clock)

Interrupts

Ten interrupt sources

- Ten interrupt vectors
- Seven interrupt level

Fast interrupt feature

General I/O

Four I/O Ports (total 27pins)

8-Bit Basic Timer

Programmable timer for oscillation stabilization interval control or watchdog timer function

Three selective internal clock frequencies

Timer/Counters

One 8-bit Timer/Counter with several clock sources (Capture mode)

One 12-bit Counter with H-sync and several clock sources

One Interval Timer

Pulse Width Modulator (PWM)

8-bit PWM: 7-CH

Sync-Processor Block

Vsync-I, Hsync-I, Csync-I input and Vsync-O, Hsync-O, Clamp-O output pins

Pseudo sync signal output

Auto SOG detection

Auto Hsync polarity detection

DDC Multi-Master IIC-Bus 1-Ch

Serial Peripheral Interface

Support for Display Data Channel (DDC1/DDC2B/DDC2Bi/DDC2B+)

Normal Multi-Master IIC-Bus 1-Ch

Serial Peripheral Interface

A/D Converter

4-channel; 8-bit resolution

Oscillator Frequency

8 MHz to 12 MHz crystal operation

Internal Max. 12 MHz CPU clock

Operating Temperature Range

– 40 °C to + 85 °C

Operating Voltage Range

4.0 V to 5.5 V

Package Types

42-pin SDIP, 44-pin QFP

BLOCK DIAGRAM

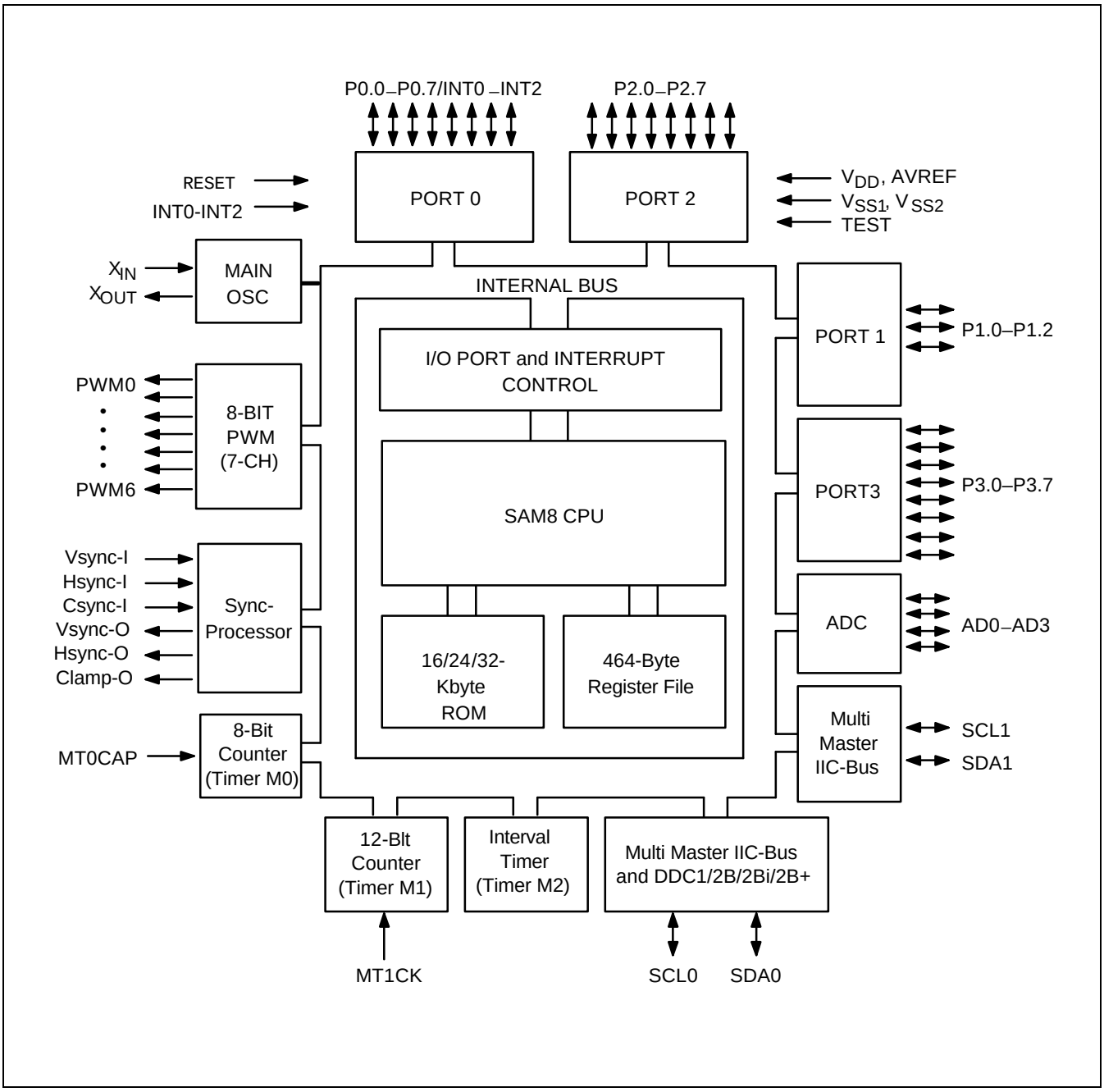


Figure 1-1. Block Diagram

PIN ASSIGNMENTS

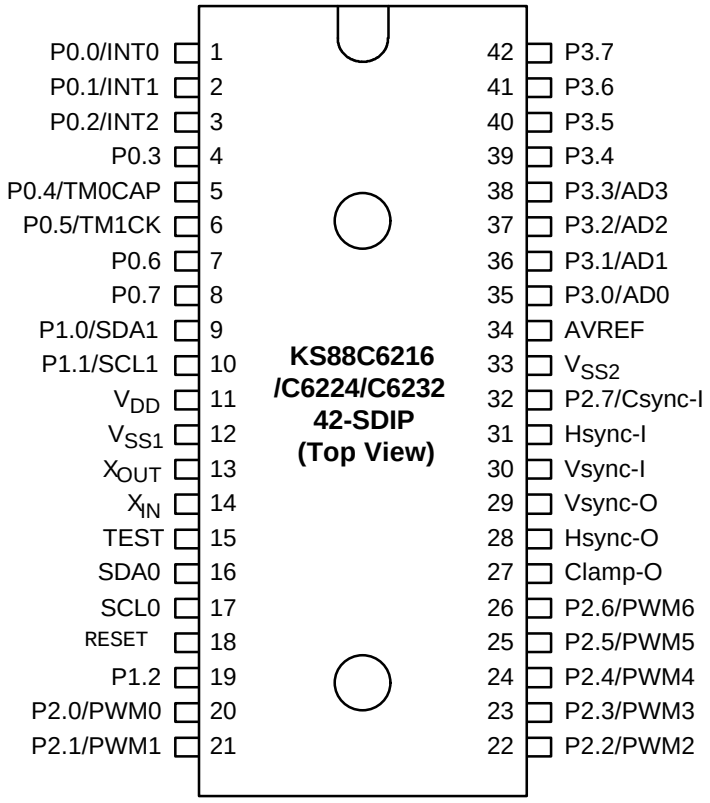


Figure 1-2. KS88C6216/C6224/C6232 42-SDIP Pin Assignment

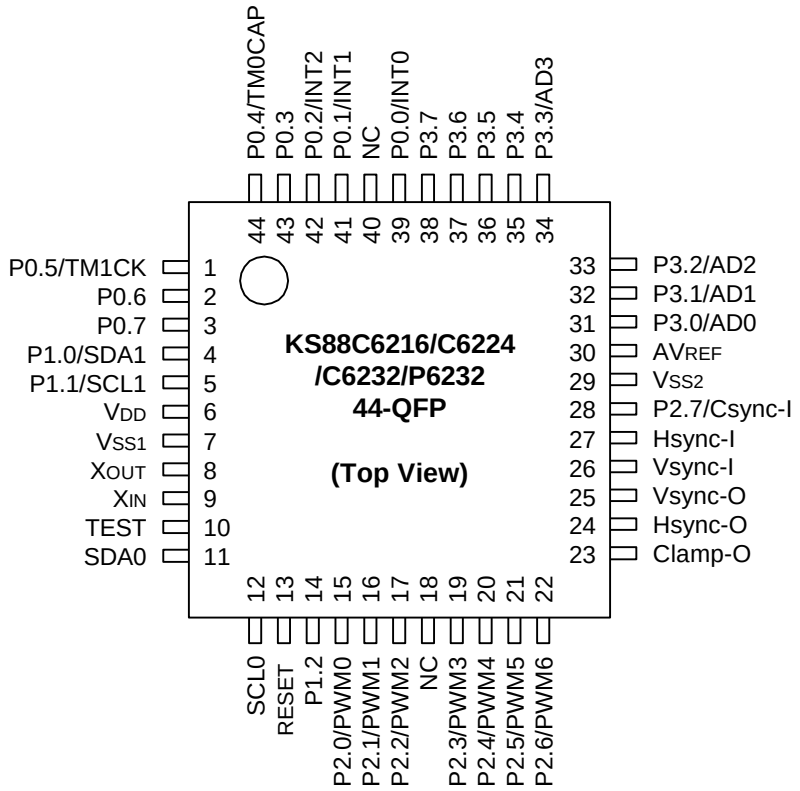


Figure 1-3. KS88C6224/C6232 44-QFP Pin Assignment

PIN DESCRIPTIONS

Table 1-1. KS88C6216/C6224/C6232/P6232 Pin Descriptions

Pin Names	Pin Type	Pin Description	Circuit Type	SDIP Pin Numbers	Shared Functions
P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7	I/O	General-purpose, 8-bit I/O port. Shared functions include three external interrupt inputs and I/O for timer M0 and M1. Selective configuration of port 0 pins to input or output mode is supported.	D-1 D-1 D-1 D-1 D-1 D-1 D-1 D-1	1 2 3 4 5 6 7 8	INT0 INT1 INT2 TM0CAP TM1CK
P1.0 P1.1 P1.2	I/O	General-purpose, 3-bit I/O port. Selective configuration is available for port 1 pins to input, push-pull output, n-channel open-drain mode, or IIC-bus clock and data I/O.	E-1 E-1 E-1	9 10 19	SDA1 SCL1
P2.0 P2.1 P2.2 P2.3 P2.4 P2.5 P2.6 P2.7	I/O	General-purpose, 8-bit I/O port Selective configuration of port 2 pins to input or output mode is supported. The port 2 pin circuits are designed to push-pull PWM output and Csync signal input.	D-1 D-1 D-1 D-1 E-1 E-1 E-1 D-1	20 21 22 23 24 25 26 32	PWM0 PWM1 PWM2 PWM3 PWM4 PWM5 PWM6 Csync-I
P3.0–P3.3 P3.4–P3.7	I/O	General-purpose, 8-bit I/O port Selective configuration port 3 pins to input or output mode is supported. Multiplexed for alternative use as A/D converter inputs AD0–AD3.	D-1 E	35–38, 39–42	AD0–AD3
Hsync-I Vsync-I Clamp-O Hsync-O Vsync-O SDA0 SCL0	I I O O O I/O I/O	The pins are sync processor signal I/O, IIC-bus clock, and data I/O.	A A A A A G-3 G-3	31 30 27 28 29 16 17	–
V_{DD}, V_{SS1}, AVREF, V_{SS2}	–	Power pins ADC power pins	– –	11, 12 34, 33	–
X_{IN}, X_{OUT}	–	System clock I/O pins	–	14, 13	–
RESET	I	System reset pin	B	18	–
TEST	I	Factory test pin input 0V:Normal operation,5V:Factory test mode	–	15	–

PIN CIRCUITS

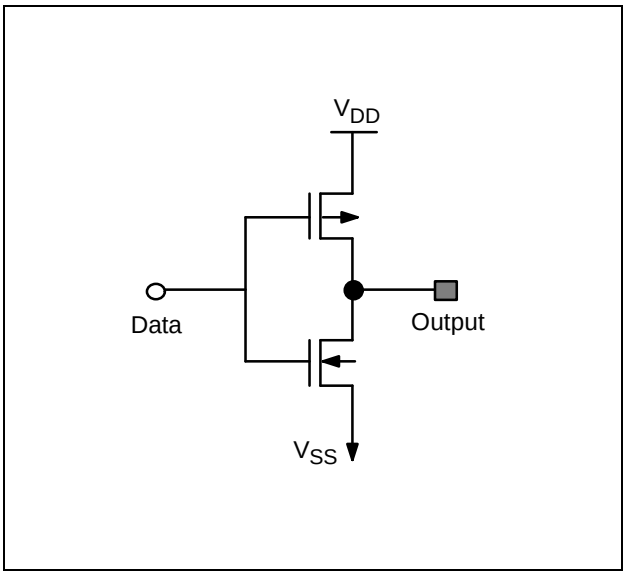


Figure 1-4. Pin Circuit Type A

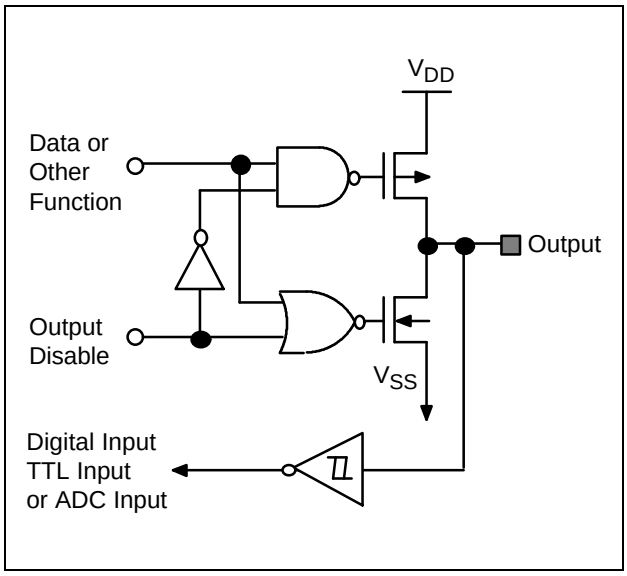


Figure 1-6. Pin Circuit Type D-1

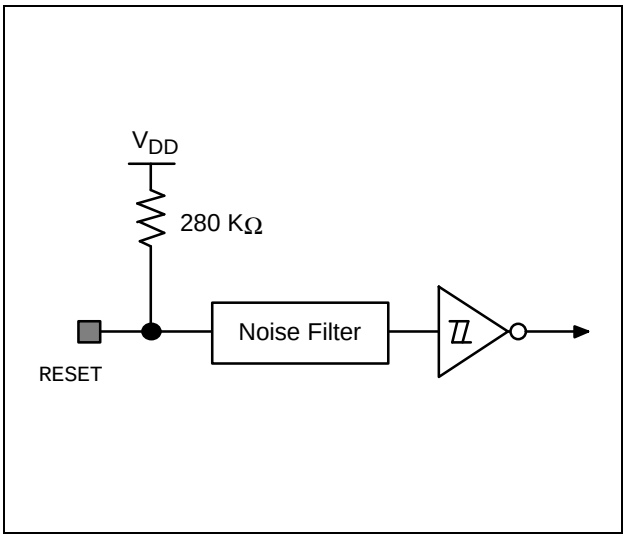


Figure 1-5. Pin Circuit Type B (RESET)

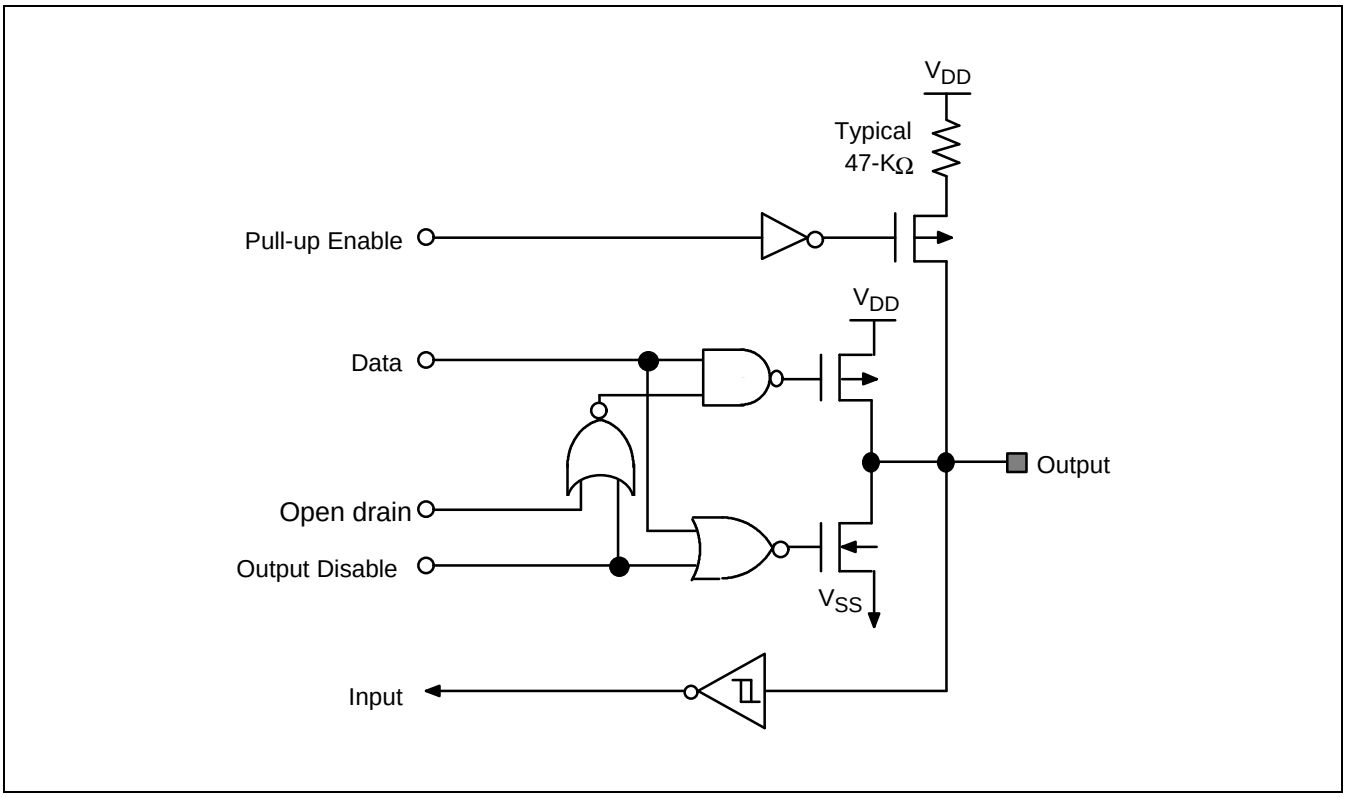


Figure 1-7. Pin Circuit Type E

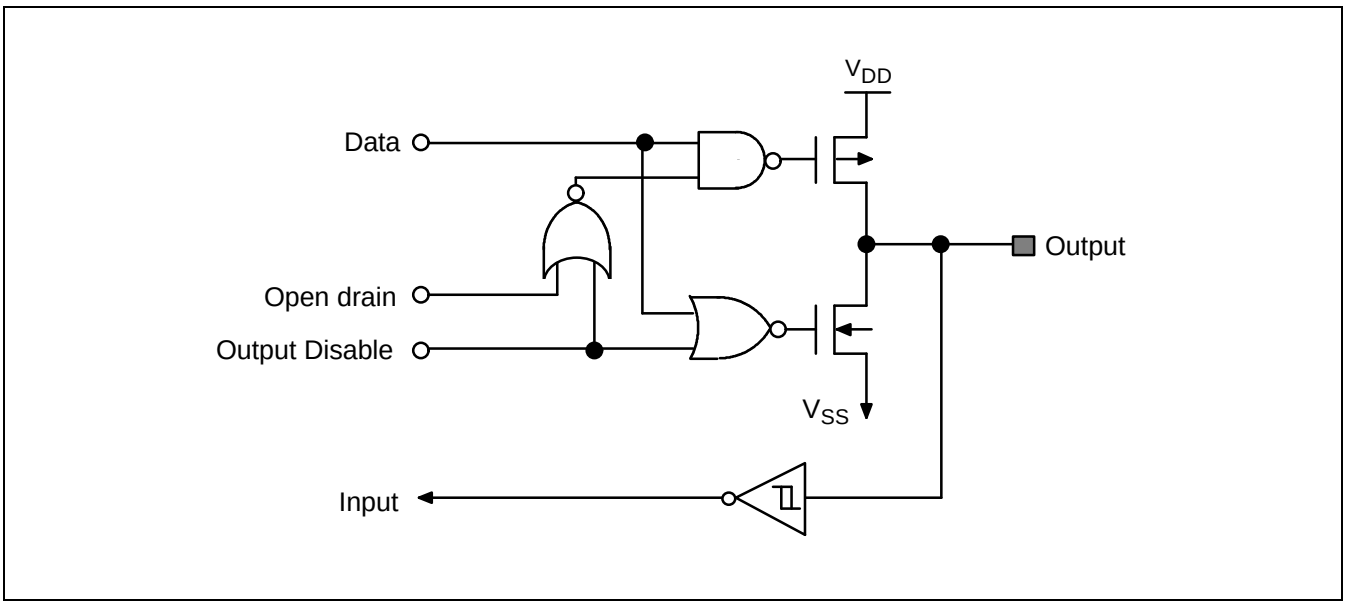


Figure 1-8. Pin Circuit Type E-1

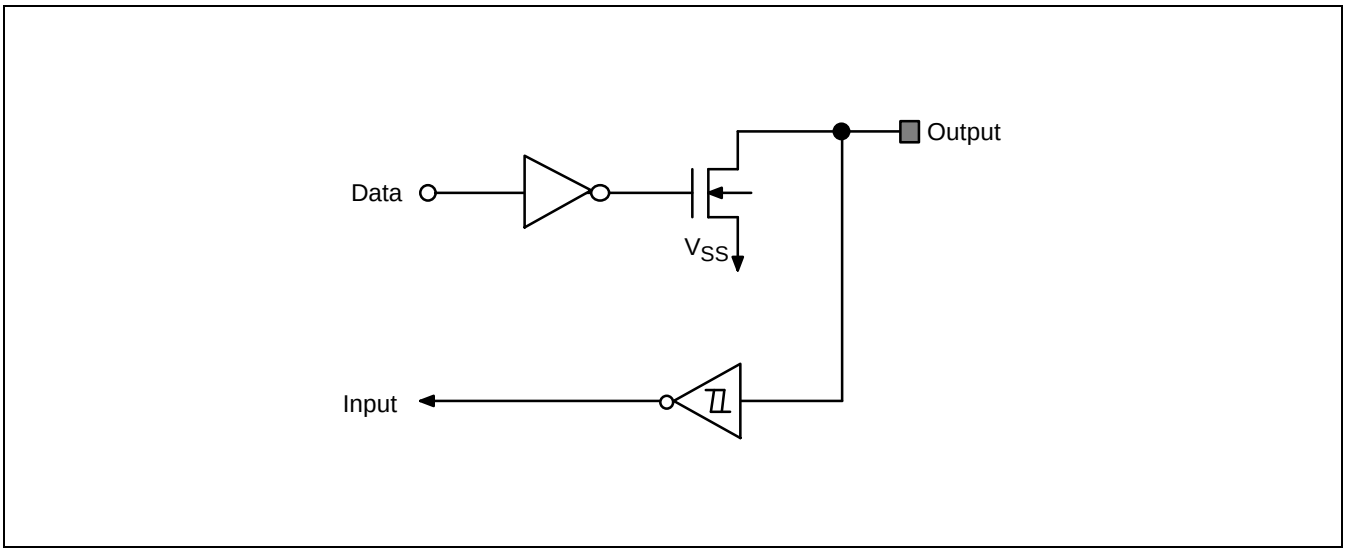


Figure 1-9. Pin Circuit Type G-3

2 ADDRESS SPACES

OVERVIEW

KS88C6216/C6224/C6232 microcontrollers have two types of address space:

— Internal program memory (ROM)

— Internal register file

The 16-bit address and data bus support program memory operations. The separate 8-bit register bus carries addresses and data between the CPU and the internal register file. KS88C6216/C6224/C6232 employ an internal 16/24/32-Kbyte mask-programmable ROM. External memory interface is not implemented.

There are 529 8-bit registers in the internal register file. In this space, there are 464 registers for general use, 19 for CPU and system control, and 46 for peripheral control and data. An area of 16-byte common working register (scratch) is part of the general-purpose register space. Most of these registers serve as either a source or destination address, or as accumulators for data memory operations.

PROGRAM MEMORY (ROM)

Program memory (ROM) stores program code or table data. KS88C6216/C6224/C6232 employ 16/24/32 K bytes of mask-programmable program memory. The memory address range is 0H–7FFFH (see Figure 2-1).

The first 256 bytes of the ROM (0H–0FFH) are reserved for interrupt vector addresses. Unoccupied locations in the address range can be used as normal program memory. When you use the vector address area to store program code, be careful not to overwrite vector addresses stored in these locations.

The ROM address at which program execution starts after a reset is 0100H.

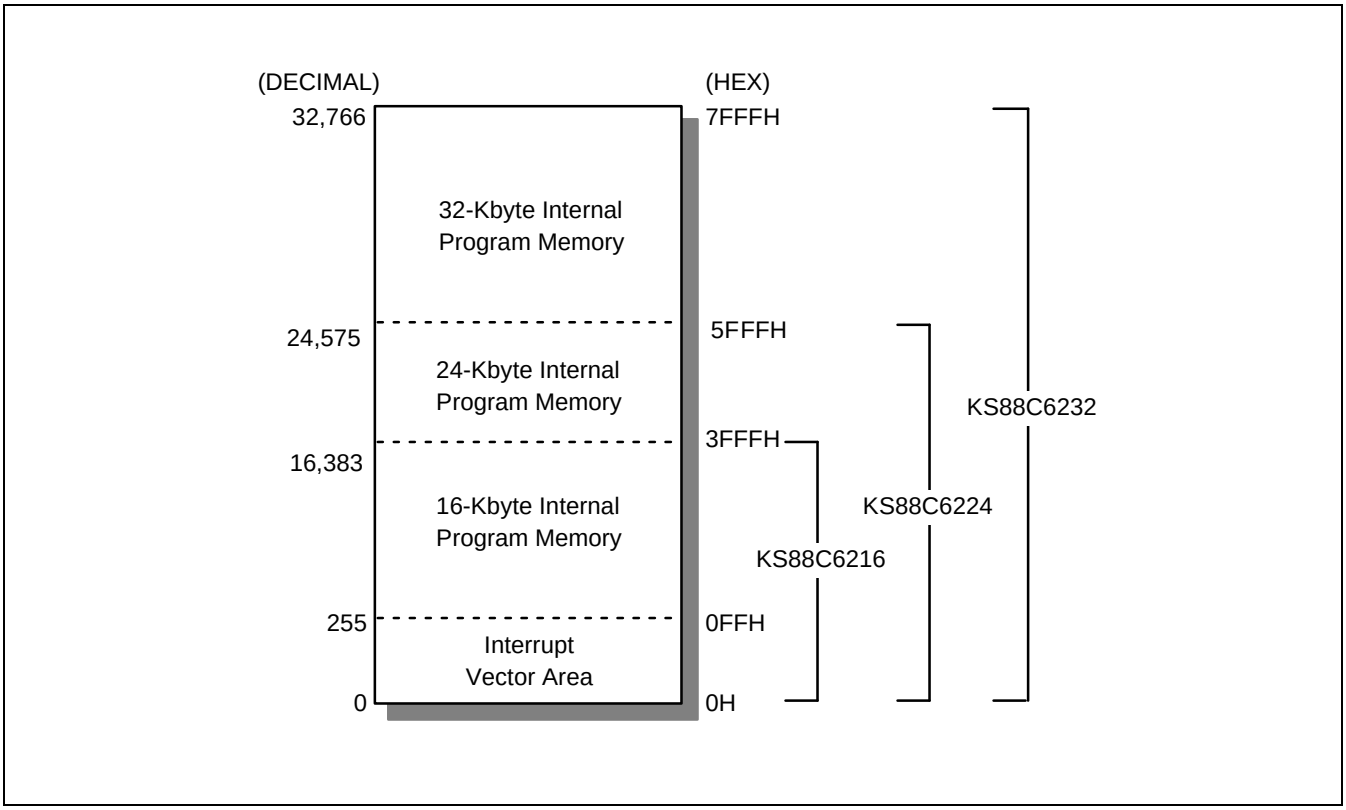


Figure 2-1. Program Memory Address Space

REGISTER ARCHITECTURE

The upper 64-byte area of the KS88C6216/C6224/C6232 internal register files is logically expanded to two 64-byte areas, called *set 1* and *set 2*. The upper 32-byte area of set 1 is divided into two register banks, *bank 0* and *bank 1*. The total physical register space is thereby expanded to 544 bytes. Within this physical space, there are 544-byte registers, of which 529 are addressable.

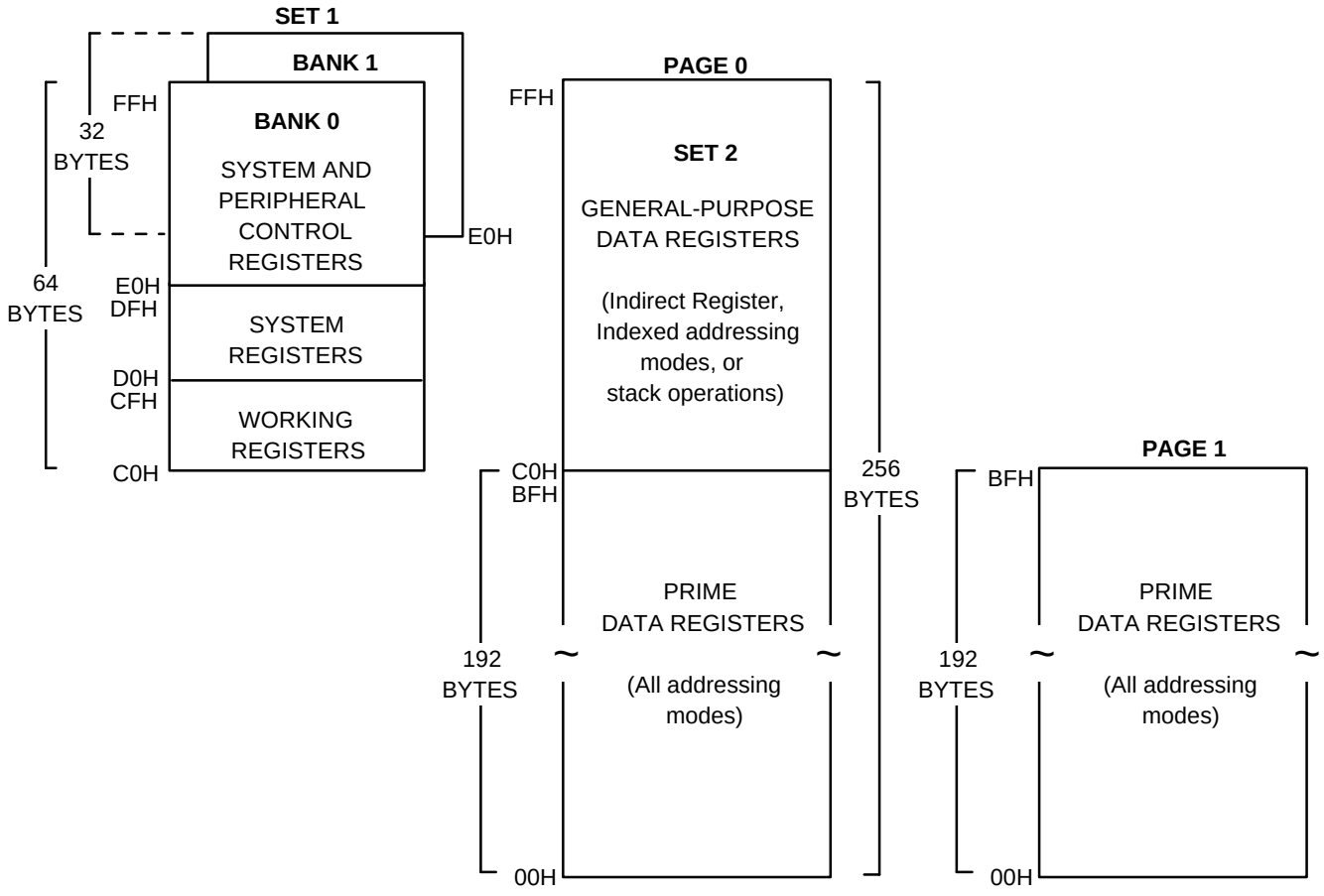
Given the microcontroller's 8-bit register bus architecture, up to 256 bytes of physical register space can be addressed as a single page. The KS88C6216/C6224/C6232 register files have two pages, page 0 and page 1. Page 0 and page 1 contain 256 bytes and 192 bytes respectively.

The extension of physical register space into separately addressable areas (sets, banks, and pages) is enabled by addressing mode restrictions, the select bank instructions SB0 and SB1, and the register page pointer, PP.

Specific register types and areas (in bytes) they occupy in the KS88C6216/C6224/C6232 internal register files are summarized in Table 2-1.

Table 2-1. Register Type Summary

Register Type	Number of Bytes
General-purpose registers (including the 16-byte common working register area)	464
CPU and system control registers	19
Clock, peripheral, I/O control, and data registers	46
Total Addressable Bytes	529



NOTE: To address registers in bank 0, bank 1, and the system register area, you must use the register addressing mode. To address working registers, you must use working register addressing mode.

Figure 2-2. Internal Register File Organization

REGISTER PAGE POINTER (PP)

The SAM8 architecture supports the logical expansion of the physical 256-byte internal register file (which use an 8-bit data bus) to as many as 16 separately addressable register pages. Page addressing is controlled by the register page pointer (PP, DFH). Two logical pages are implemented in KS88C6216/C6224/C6232. These pages are used as general purpose register space.

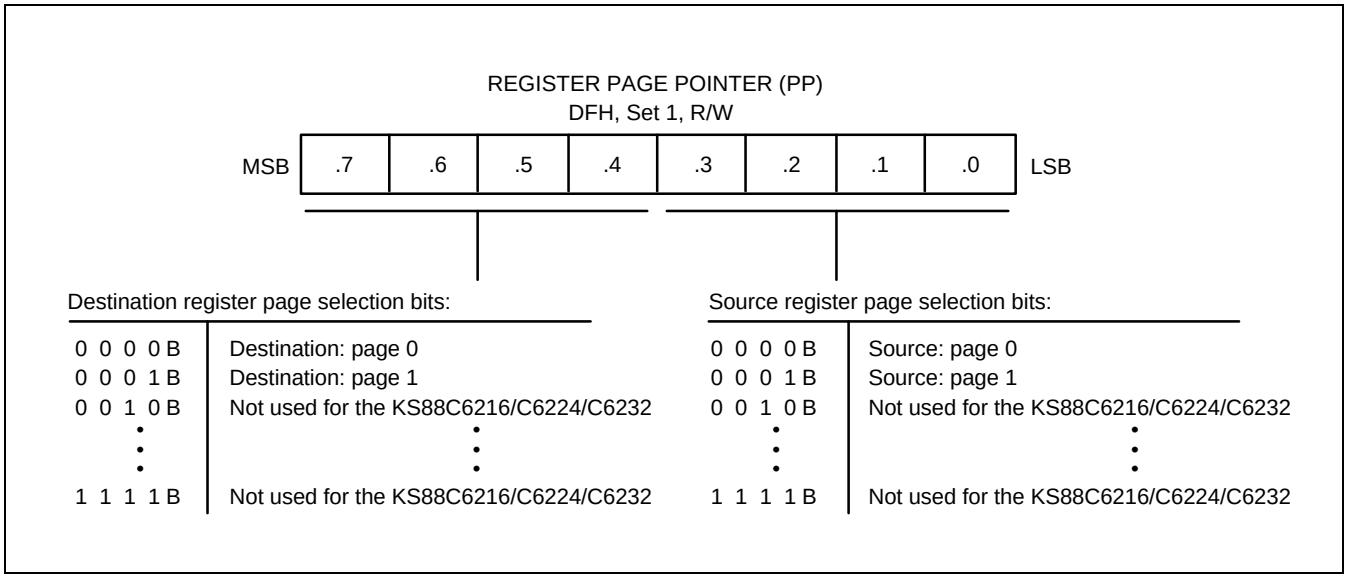


Figure 2-3. Register Page Pointer (PP)

REGISTER SET 1

The term *set 1* refers to the upper 64 bytes of the register file, locations C0H–FFH. The upper 32-byte area of this 64-byte space (E0H–FFH) is divided into two 32-byte register banks, *bank 0* and *bank 1*. You execute the set register bank instructions SB0 or SB1 to address one bank or the other. Bank 0 is automatically selected by a reset operation.

In KS88C6216/C6224/C6232, register locations of only E0H–F5H are addressable in the bank 1 area; the remaining locations (F6H–FFH) are not mapped. The lower 32-byte area of set 1 is not banked and can be addressed at any time. It contains 16 mapped system registers (D0H–DFH) and a 16-byte “scratch” area (C0H–CFH) for working register addressing.

Registers in set 1 are directly accessible at all times using Register addressing mode. The 16-byte working register area can only be accessed using working register addressing. (For more information about working register addressing, please refer to Chapter 3, “Addressing Modes.”)

REGISTER SET 2

The same 64-byte physical space that is used for set 1 register locations C0H–FFH is logically duplicated to add another 64 bytes of space. This expanded area of the register file is called *set 2*. All set 2 locations (C0H–FFH) can be addressed in the page 0 of the KS88C6216/C6224/C6232 register space.

The logical division of set 1 and set 2 is maintained by means of addressing mode. In order to access set 1, you should use resister addressing mode. When you want to access register locations in set 2, you have to select Register Indirect addressing mode or Indexed addressing mode access register locations in set 2.

PRIME REGISTER SPACE

The lower 192 bytes of the 256-byte physical internal register file (00H–BFH) is called the *prime register space,* or more simply, the *prime area*. You can access registers in this address range at page 0 using any of the seven explicit addressing modes (see Chapter 3, “Addressing Modes”). All registers in the prime area can be addressed immediately after a reset.

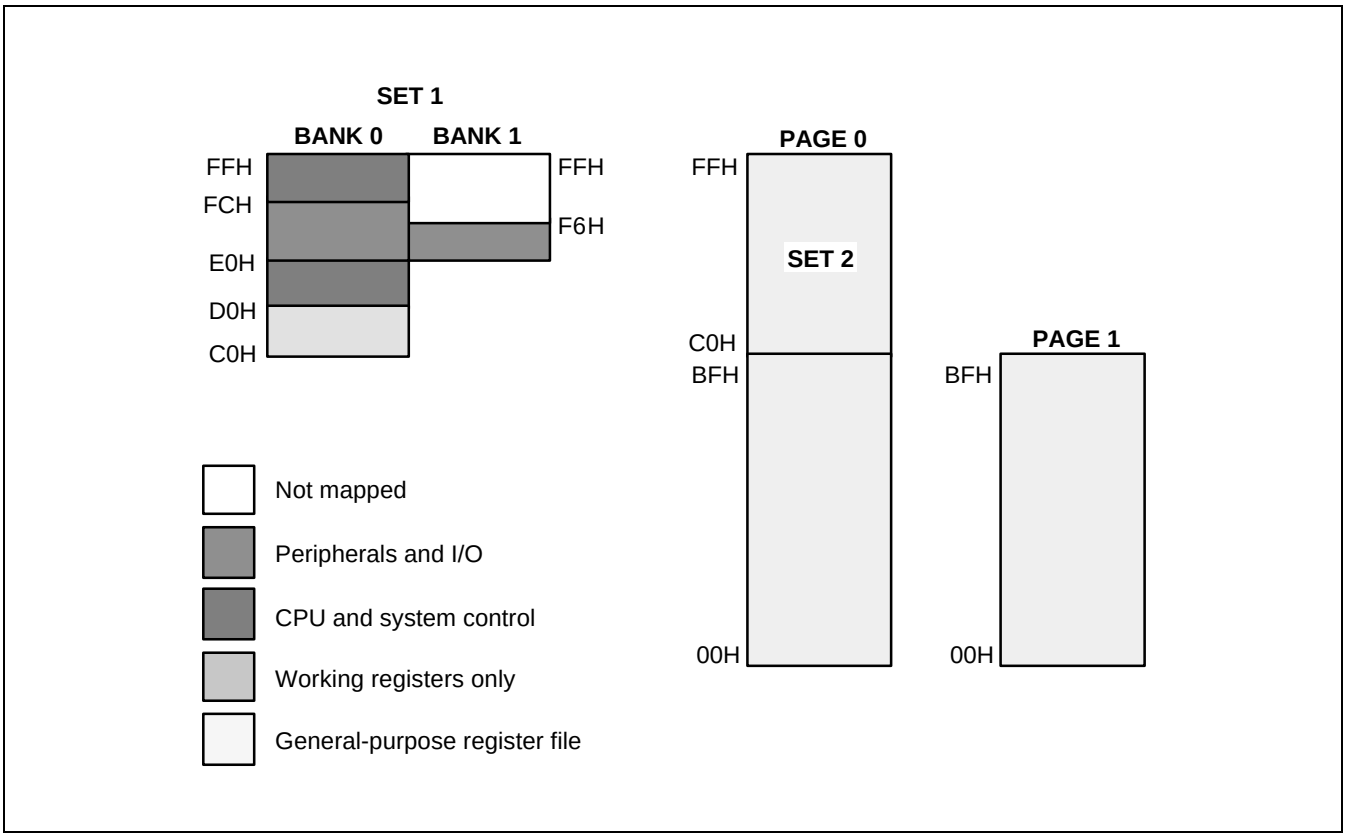


Figure 2-4. Set 1, Set 2, and Prime Area Register Map

WORKING REGISTERS

Instructions can access specific 8-bit registers or 16-bit register pairs using either 4-bit or 8-bit address fields. When 4-bit working register addressing is used, the 256-byte register file can be seen by the programmer as one that consists of 32 8-byte register groups or "slices." Each slice comprises of eight 8-bit registers.

With the two 8-bit register pointers, RP1 and RP0 employed, two working register slices can be selected at any time to form a 16-byte working register block. The register pointers help, you move this 16-byte register block to anywhere in the addressable register file, except for the set 2 area.

The terms *slice* and *block* are used in this manual to help you visualize the size and relative locations of selected working register spaces:

— One working register *slice* is 8 bytes (eight 8-bit working registers; R0–R7 or R8–R15)

— One working register *block* is 16 bytes (sixteen 8-bit working registers; R0–R15)

All the registers in an 8-byte working register slice have the same binary value for their five most significant address bits. This makes it possible for each register pointer to point to one of the 24 slices in the register file. The base addresses for the two 8-byte register slices selected are contained in register pointers RP0 and RP1.

After a reset, RP0 and RP1 always point to the 16-byte common area in set 1 (C0H–CFH).

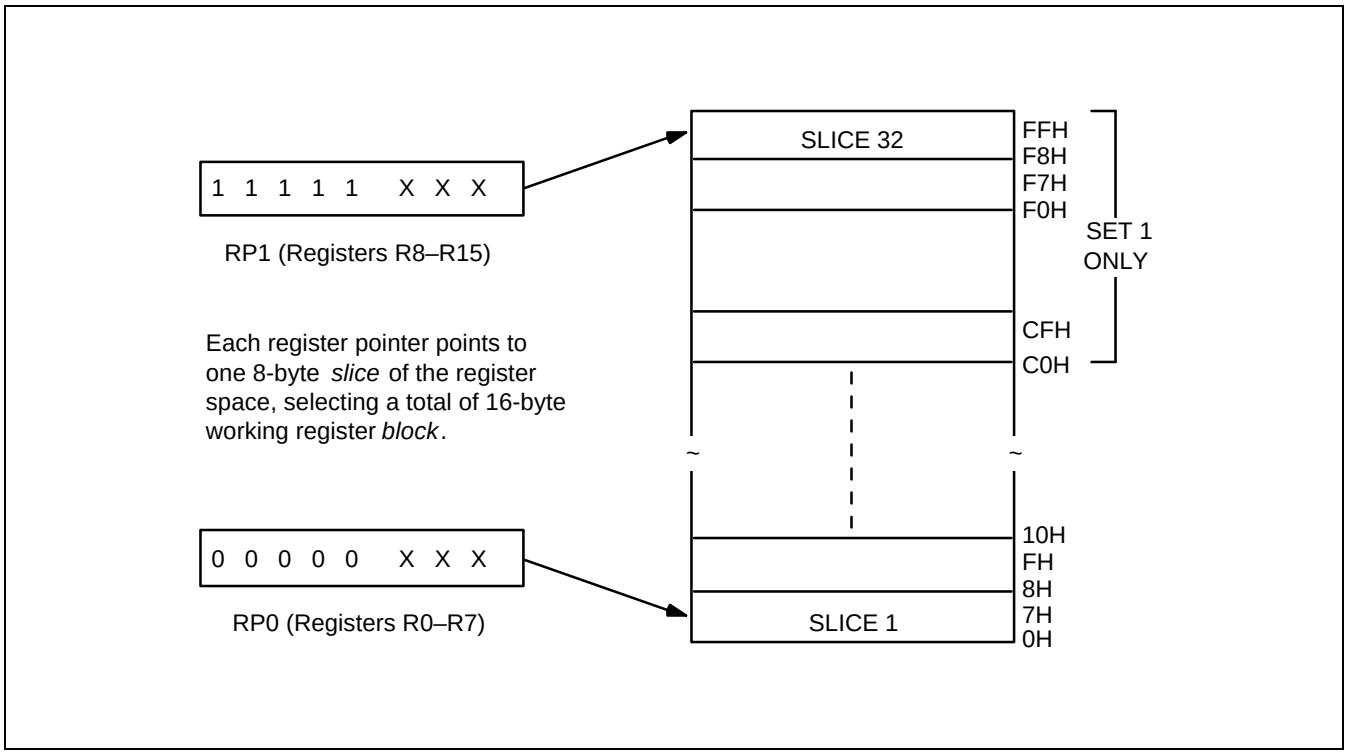


Figure 2-5. 8-Byte Working Register Areas (Slices)

USING THE REGISTER POINTERS

Register pointers of RP0 and RP1 which are mapped to the addresses D6H and D7H in set 1, are used to select two movable 8-byte working register slices in the register file. After a reset, they point to the working register common area: RP0 points to the addresses C0H–C7H, and RP1 points to the addresses C8H–CFH.

You can change a register pointer value, by loading a new value to RP0 and/or RP1 using an SRP or LD instruction (see Figures 2-6 and 2-7).

In working register addressing, you can only access those two 8-bit slices of the register file that are currently pointed to by RP0 and RP1. You cannot use the register pointers to select a working register area in set 2, C0H–FFH, because these locations can be accessed only with Indirect Register or Indexed addressing modes.

The 16-byte working register block selected usually consists of two contiguous 8-byte slices. As a general programming guideline, we recommend that RP0 point to the "lower" slice and RP1 point to the "upper" slice (see Figure 2-6). In some cases, it may be necessary to define working register areas in different (non-contiguous) areas of the register file. In Figure 2-7, RP0 points to the "upper" slice and RP1 to the "lower" slice.

As a register pointer can point to either of the two 8-byte slices in the working register block, you can flexibly define the working register area to support a variety of program requirements.

☞ PROGRAMMING TIP — Setting the Register Pointers

```
SRP     #70H        ; RP0 ← 70H, RP1 ← 78H
SRP1    #48H        ; RP0 ← no change, RP1 ← 48H,
SRP0    #0A0H       ; RP0 ← A0H, RP1 ← no change
CLR     RP0         ; RP0 ← 00H, RP1 ← no change
LD      RP1,#0F8H   ; RP0 ← no change, RP1 ← 0F8H
```

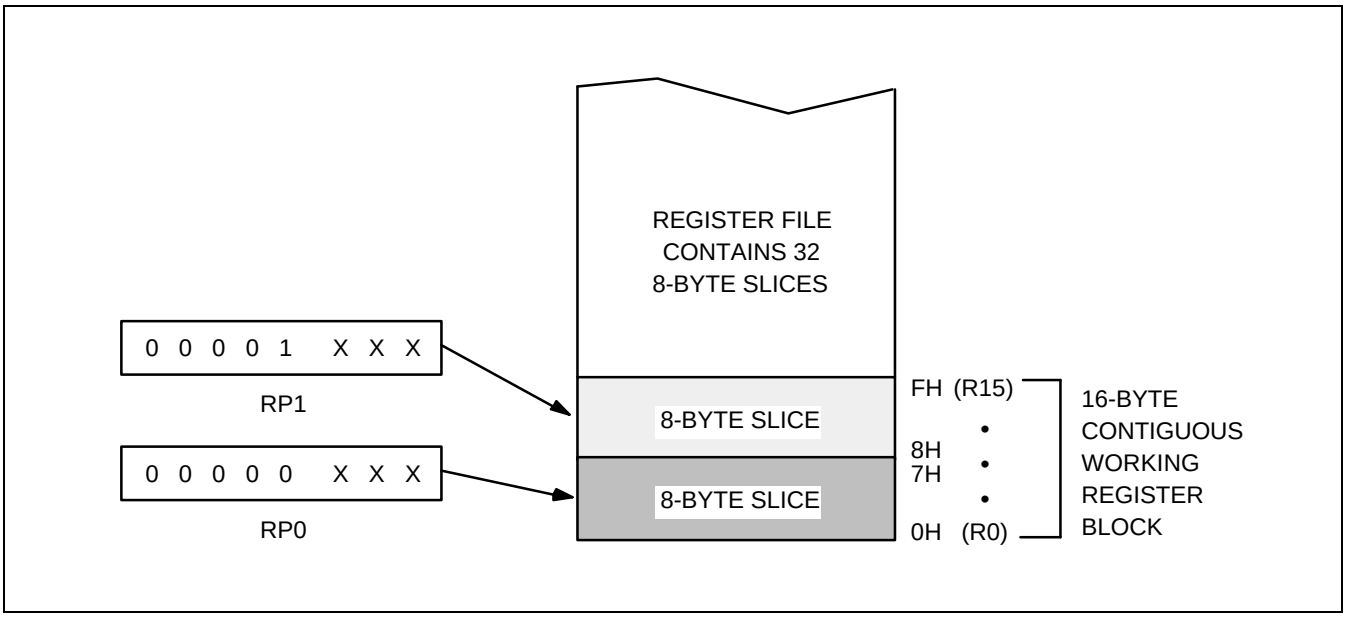


Figure 2-6. Contiguous 16-Byte Working Register Block

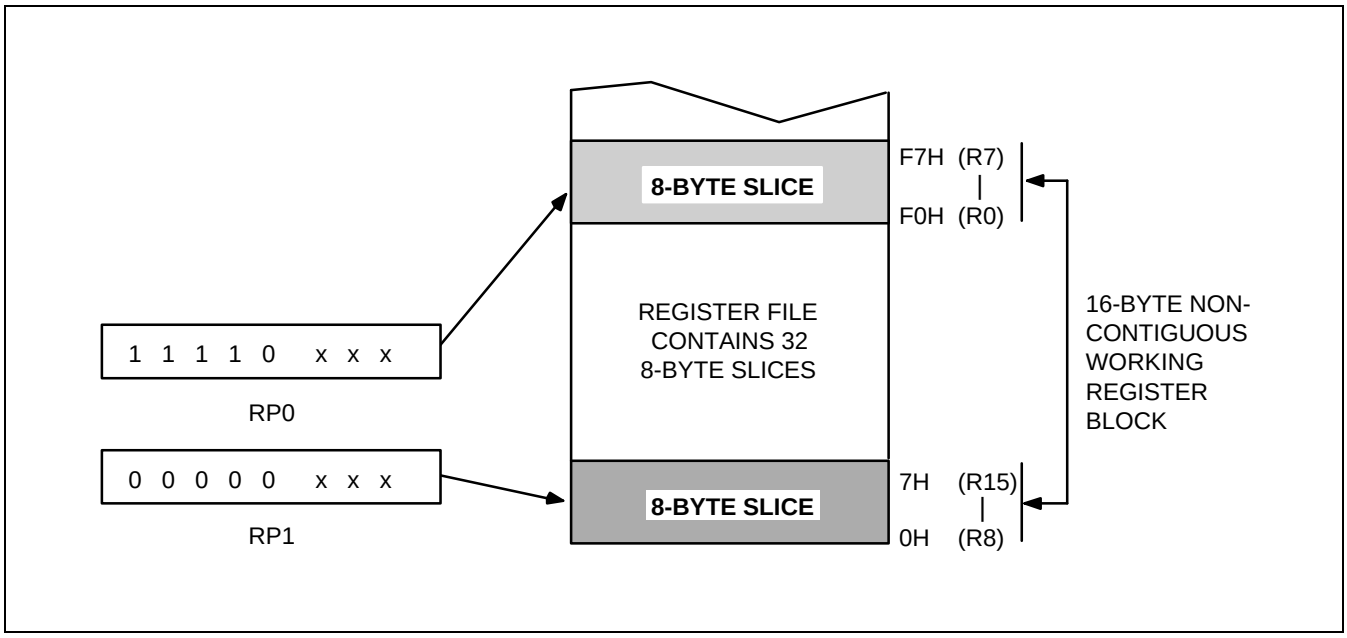


Figure 2-7. Non-Contiguous 16-Byte Working Register Block

☞ PROGRAMMING TIP — Calculate the Sum of a Series of Registers Using the RPs

Calculate the sum of registers 80H–85H using the register pointer and working register addressing. The register addresses from 80H through 85H contain the values 10H, 11H, 12H, 13H, 14H, and 15 H, respectively:

```
SRP0    #80H      ;  RP0 ← 80H
ADD     R0,R1     ;  R0 ← R0 + R1
ADC     R0,R2     ;  R0 ← R0 + R2 + C
ADC     R0,R3     ;  R0 ← R0 + R3 + C
ADC     R0,R4     ;  R0 ← R0 + R4 + C
ADC     R0,R5     ;  R0 ← R0 + R5 + C
```

The sum of these six registers, 6FH, is located in the register R0 (80H). The instruction string used in this example takes 12 bytes of instruction code and its execution time is 36 cycles. If the register pointer is not used to calculate the sum of these registers, the following instruction sequence would have to be used:

```
ADD     80H,81H   ;  80H ← (80H) + (81H)
ADC     80H,82H   ;  80H ← (80H) + (82H) + C
ADC     80H,83H   ;  80H ← (80H) + (83H) + C
ADC     80H,84H   ;  80H ← (80H) + (84H) + C
ADC     80H,85H   ;  80H ← (80H) + (85H) + C
```

The sum of the six registers, here, is also located in the register 80H. This instruction string, however, takes 15 bytes of instruction code instead of 12 bytes, and its execution time is 50 cycles instead of 36 cycles.

REGISTER ADDRESSING

The SAM8 register architecture provides an efficient method of working register addressing that takes full advantage of shorter instruction formats to reduce execution time.

With Register (R) addressing mode, in which the operand value is the content of a specific register or register pair, you can access all locations in the register file except for set 2. With working register addressing, you use a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space.

Registers are addressed either as a single 8-bit register or as a paired 16-bit register space. In a 16-bit register pair, the address of the first 8-bit register is always an even number and the address of the next register is always an odd number. The most significant byte of the 16-bit data is always stored in the even-numbered register; the least significant byte is always stored in the next (+ 1) odd-numbered register.

Working register addressing differs from Register addressing in a way that it uses a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space (see Figure 3-2).

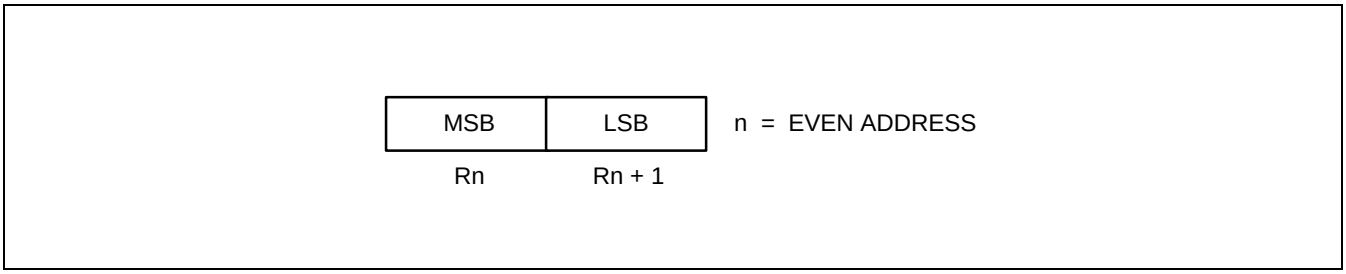


Figure 2-8. 16-Bit Register Pair

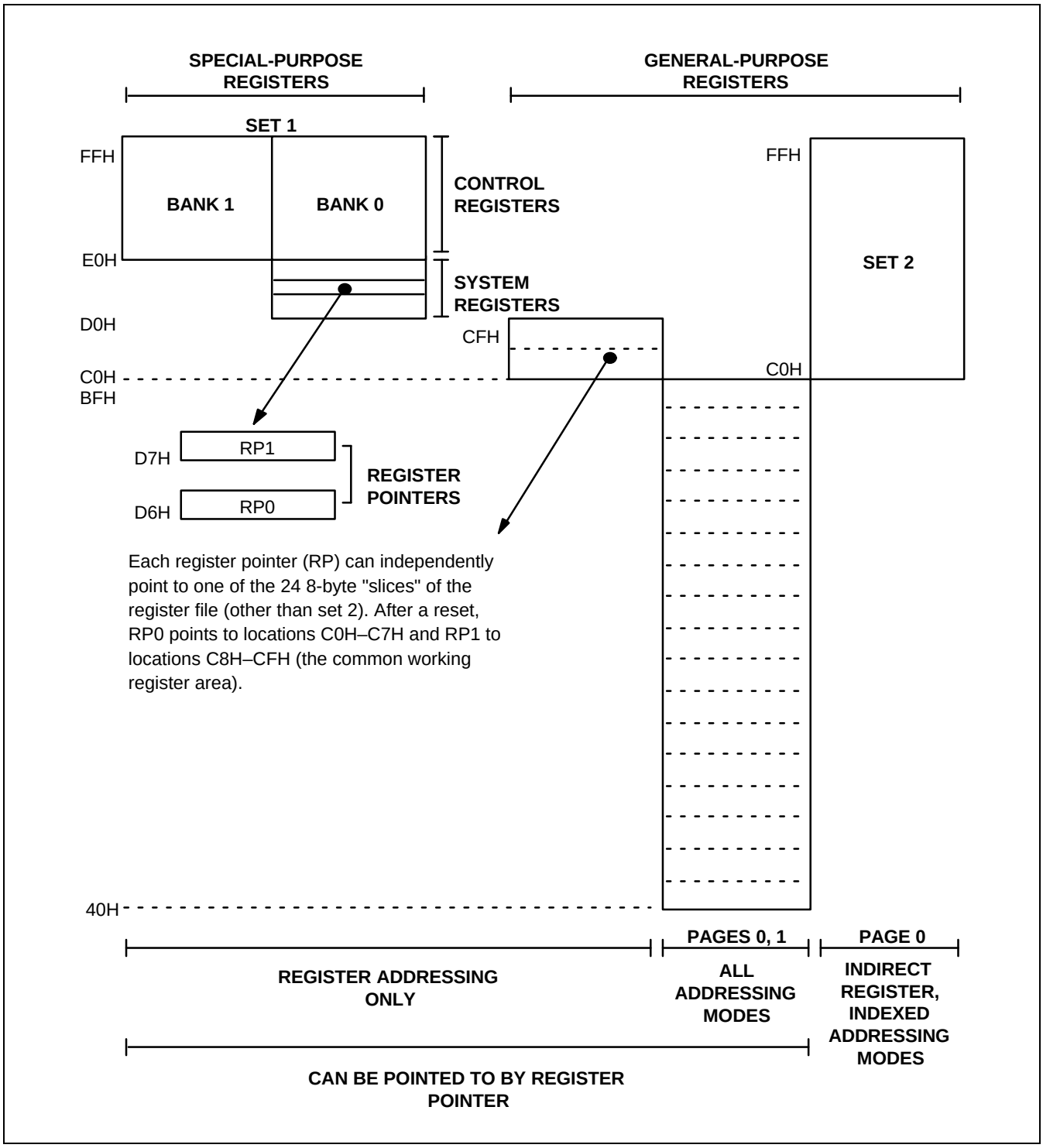


Figure 2-9. Register File Addressing

COMMON WORKING REGISTER AREA (C0H–CFH)

After a reset, register pointers RP0 and RP1 automatically select two 8-byte register slices in set 1, locations C0H–CFH, as the active 16-byte working register block:

RP0 → C0H–C7H
RP1 → C8H–CFH

This16-byte address range is called *common working register area*. That is, locations in this area can be used as working registers by operations that address any location on any page in the register file. Typically, these working registers serve as temporary buffers for data operations between different pages.

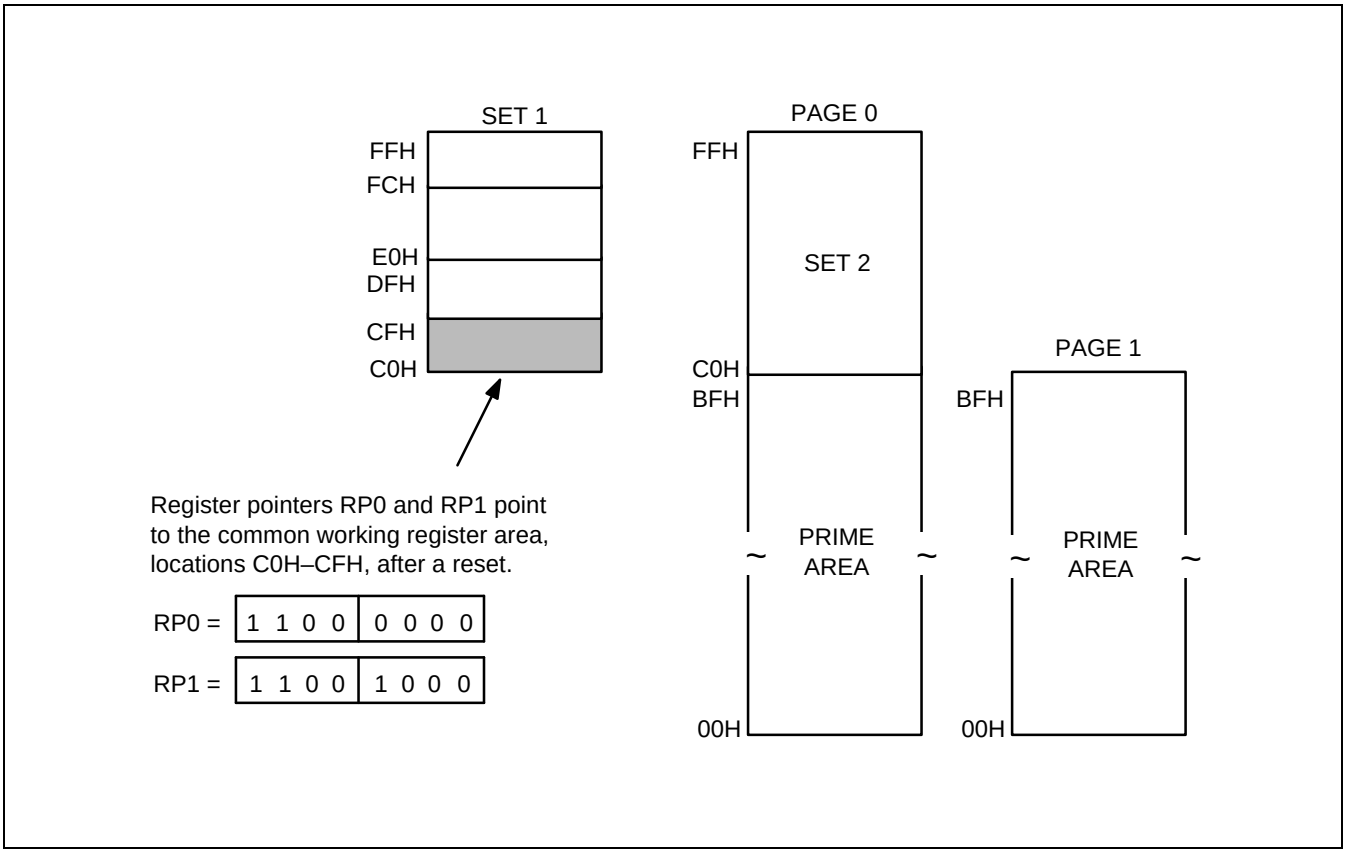


Figure 2-10. Common Working Register Area

☞ PROGRAMMING TIP — Addressing the Common Working Register Area

As the following examples show, you should access working registers in the common area, locations C0H–CFH, using working register addressing mode only.

Examples

```
1. LD      0C2H,40H        ;  Invalid addressing mode!'

   Use working register addressing instead:

   SRP     #0C0H
   LD      R2,40H          ;  R2 (C2H) ← the value in location 40H

2. ADD     0C3H,#45H       ;  Invalid addressing mode!

   Use working register addressing instead:

   SRP     #0C0H
   ADD     R3,#45H         ;  R3 (C3H) ← R3 + 45H
```

4-BIT WORKING REGISTER ADDRESSING

Each register pointer defines a movable 8-byte slice of working register space. The address information stored in a register pointer serves as an addressing "window" that makes it possible for instructions to access working registers very efficiently using short 4-bit addresses. When an instruction addresses a location in the selected working register area, the address bits are concatenated in the following way to form a complete 8-bit address:

— The high-order bit of the 4-bit address selects one of the register pointers ("0" selects RP0; "1" selects RP1);

— The five high-order bits in the register pointer select an 8-byte slice of the register space;

— The three low-order bits of the 4-bit address select one of the eight registers in the slice.

As shown in Figure 2-11, the result of this operation is that the five high-order bits from the register pointer are concatenated with the three low-order bits from the instruction address to form the complete address. As long as the address stored in the register pointer remains unchanged, the three bits from the address will always point to an address in the same 8-byte register slice.

Figure 2-12 shows a typical example of 4-bit working register addressing: the high-order bit of the instruction "INC R6" is "0", which selects RP0. The five high-order bits stored in RP0 (01110B) are concatenated with the three low-order bits of the instruction's 4-bit address (110B) to produce the register address 76H (01110110B).

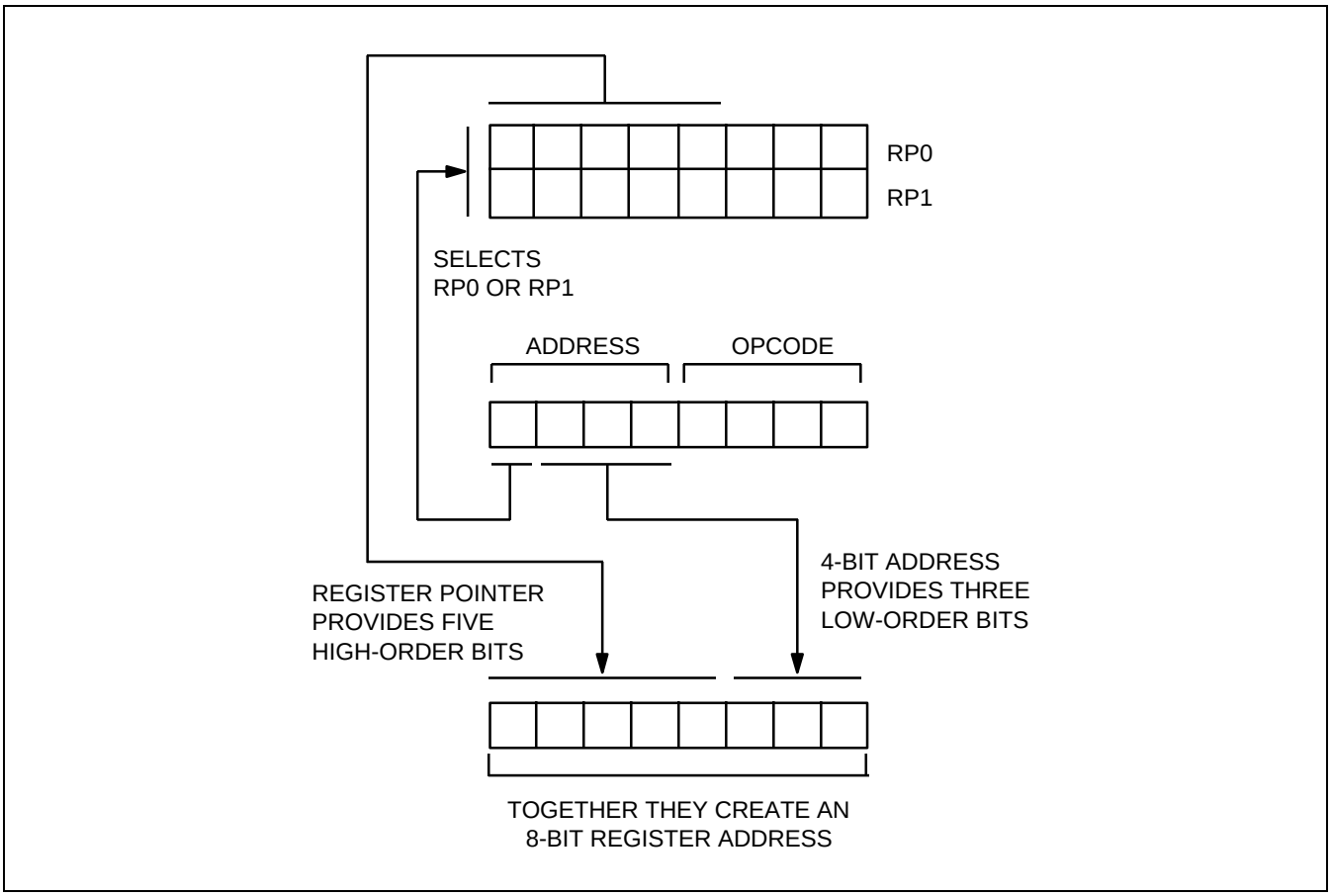


Figure 2-11. 4-Bit Working Register Addressing

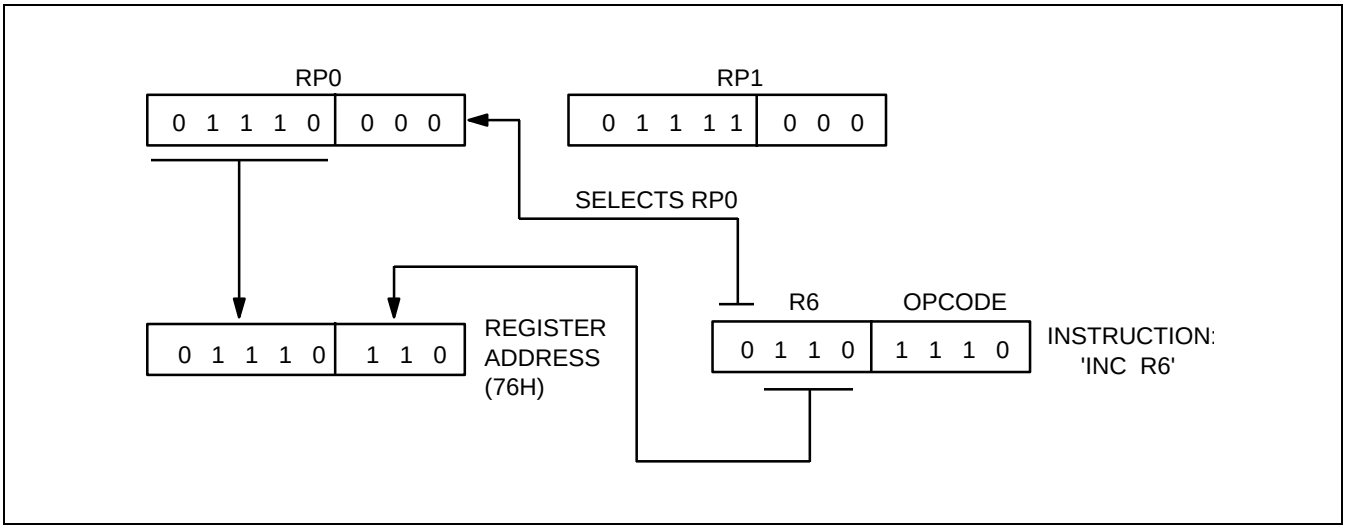


Figure 2-12. 4-Bit Working Register Addressing Example

8-BIT WORKING REGISTER ADDRESSING

You can also use 8-bit working register addressing to access registers in a selected working register area. To initiate 8-bit working register addressing, the upper four bits of the instruction address must contain the value of 1100B. This 4-bit value (1100B) indicates that the remaining four bits have the same effect as 4-bit working register addressing.

As shown in Figure 2-13, the lower nibble of the 8-bit address is concatenated in much the same way as for 4-bit addressing: Bit 3 selects either RP0 or RP1, which then supplies the five high-order bits of the final address; the three low-order bits of the complete address are provided by the original instruction.

Figure 2-14 shows an example of 8-bit working register addressing: the four high-order bits of the instruction address (1100B) specify 8-bit working register addressing. Bit 4 ("1") selects RP1 and the five high-order bits in RP1 (10101B) become the five high-order bits of the register address. The three low-order bits of the register address (011) are provided by the three low-order bits of the 8-bit instruction address. The five address bits from RP1 and the three address bits from the instruction are concatenated to form the complete register address, 0ABH (10101011B).

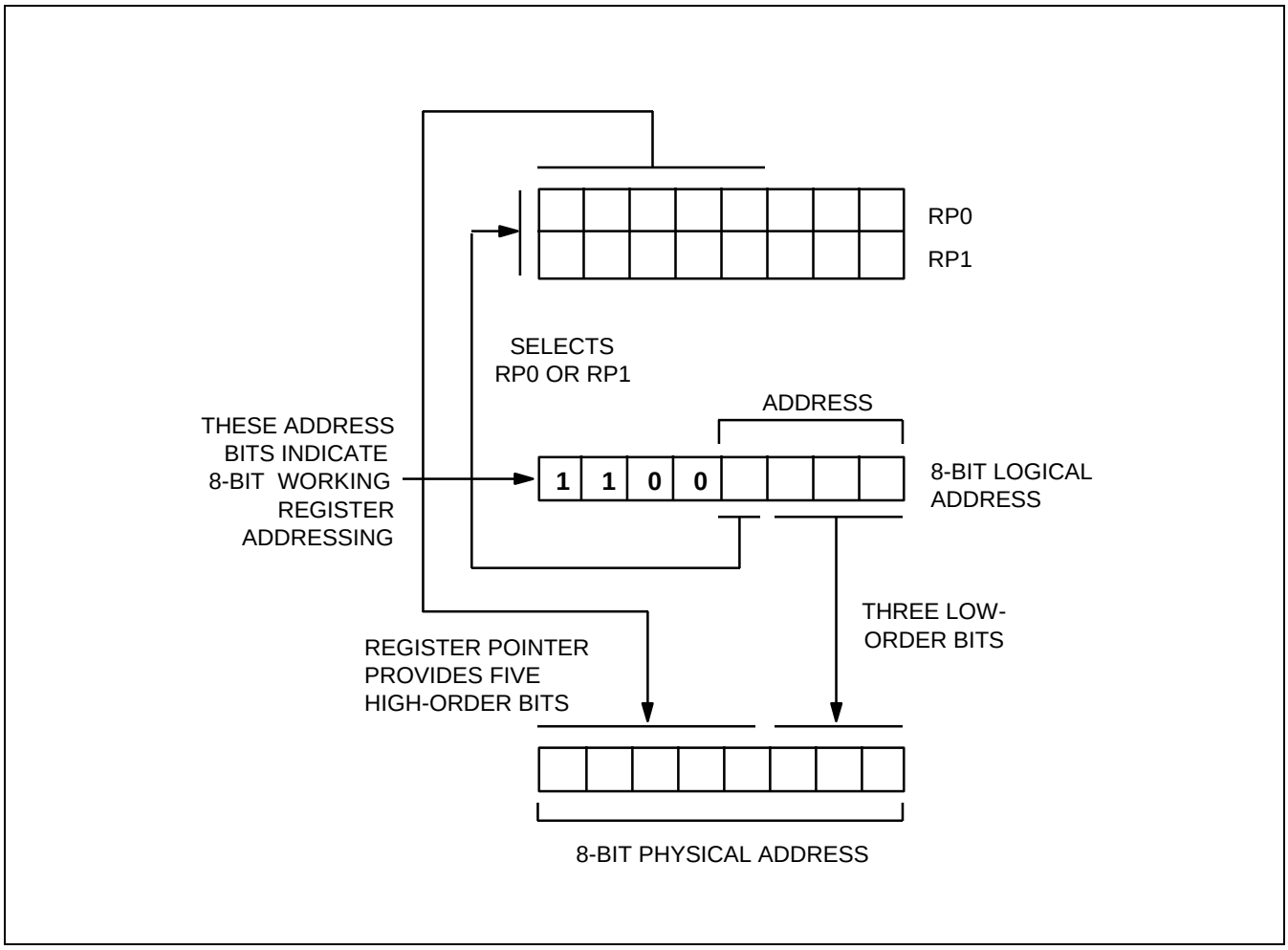


Figure 2-13. 8-Bit Working Register Addressing

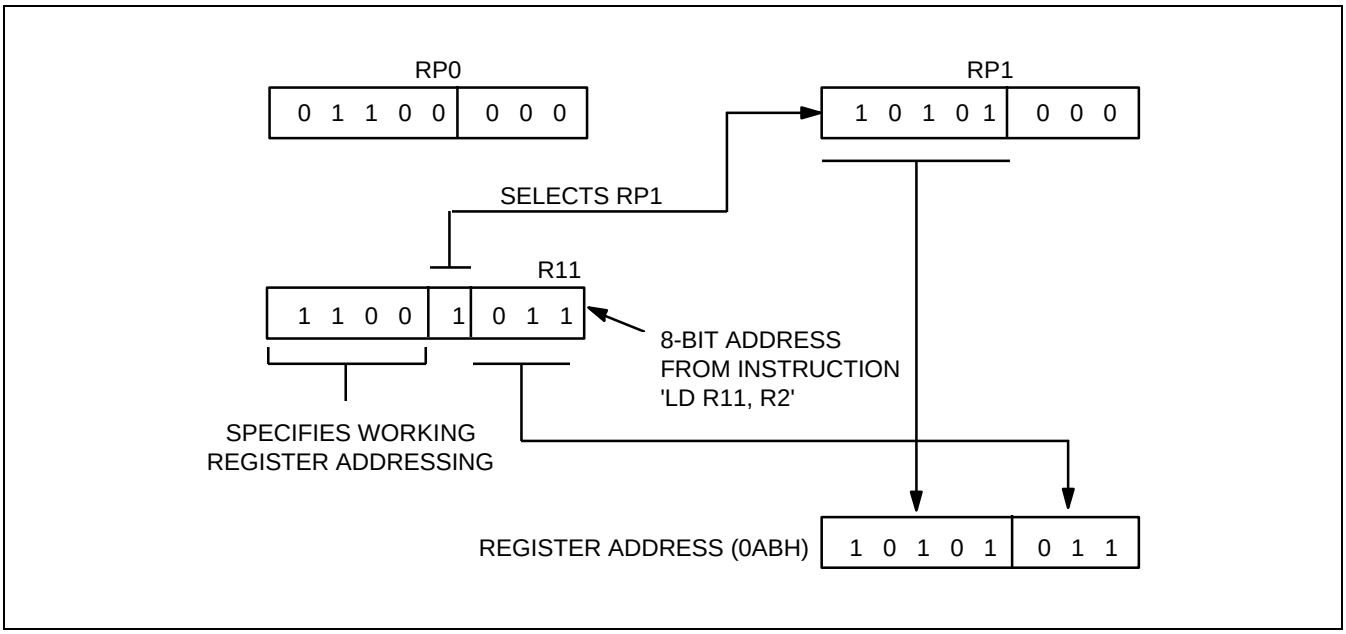


Figure 2-14. 8-Bit Working Register Addressing Example

SYSTEM AND USER STACKS

KS88-series microcontrollers can be programmed to use the system stack for subroutine calls, returns and interrupts and to store data. The PUSH and POP instructions are used to control system stack operations. The KS88C6216/C6224/C6232 architecture supports stack operations in the internal register file.

Stack Operations

Return addresses for procedure calls and interrupts and data are stored on the stack. The contents of the PC are saved to stack by a CALL instruction and restored by the RET instruction. When an interrupt occurs, the contents of the PC and the FLAGS register are pushed to the stack. The IRET instruction then pops these values back to their original locations. The stack address is always decremented *before* a push operation and incremented *after* a pop operation. The stack pointer (SP) always points to the stack frame stored on the top of the stack, as shown in Figure 2-15.

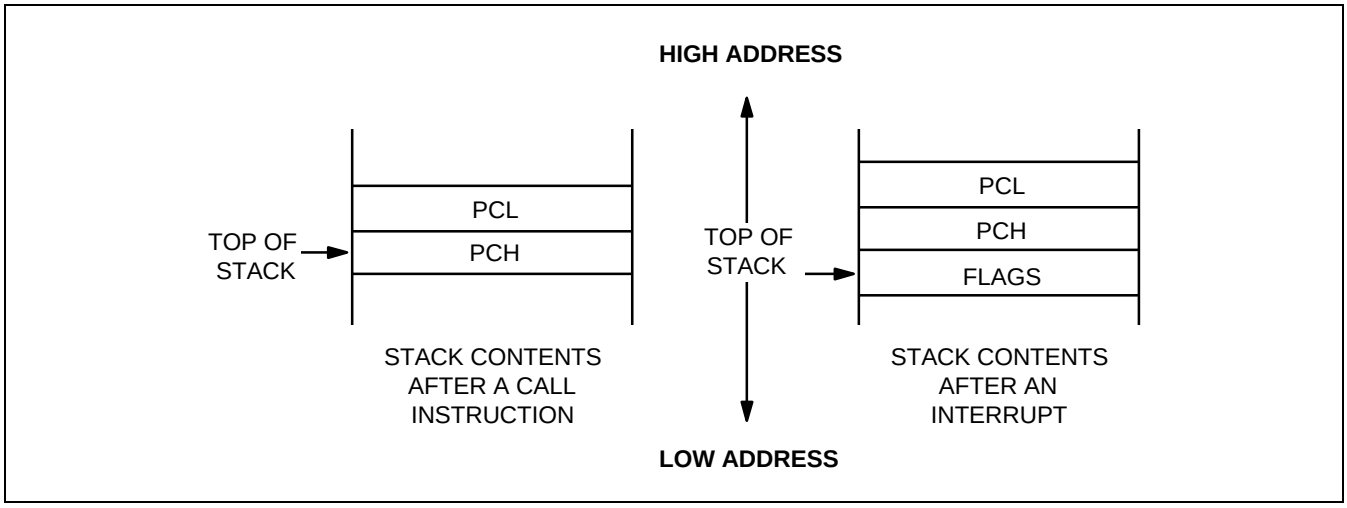


Figure 2-15. Stack Operations

User-Defined Stacks

You can freely define stacks in the internal register file as data storage locations. The instructions PUSHUI, PUSHUD, POPUI, and POPUD support user-defined stack operations.

Stack Pointers (SPL, SPH)

Register locations D8H and D9H contain the 16-bit stack pointer (SP) that is used for system stack operations. The most significant byte of the SP address, SP15–SP8, is stored in the SPH register (D8H) and the least significant byte, SP7–SP0, is stored in the SPL register (D9H). After a reset, the SP value is undetermined.

Because only internal memory space is implemented in KS88C6216/C6224/C6232, the SPL must be initialized to an 8-bit value in the range 00H–FFH. The SPH register is not needed here and can be used as a general-purpose register, if necessary.

When the SPL register contains the only stack pointer value (that is, when it points to a system stack in the register file), you can use the SPH register as a general-purpose data register. However, if an overflow or underflow condition occurs as the result of incrementing or decrementing the stack address in the SPL register during normal stack operations, the value in the SPL register will overflow (or underflow) to the SPH register, overwriting any data that is currently stored there. To avoid overwriting data in the SPH register, you can initialize the SPL value to "FFH" rather than "00H".

☞ PROGRAMMING TIP — Standard Stack Operations Using PUSH and POP

The following example shows you how to perform stack operations in the internal register file using PUSH and POP instructions:

```
LD      SPL,#0FFH       ; SPL ← FFH
                        ; (Normally, the SPL is set to 0FFH by the initialization
                        ; routine)
•
•
•
PUSH    PP              ; Stack address 0FEH ← PP
PUSH    RP0             ; Stack address 0FDH ← RP0
PUSH    RP1             ; Stack address 0FCH ← RP1
PUSH    R3              ; Stack address 0FBH ← R3
•
•
•
POP     R3              ; R3 ← Stack address 0FBH
POP     RP1             ; RP1 ← Stack address 0FCH
POP     RP0             ; RP0 ← Stack address 0FDH
POP     PP              ; PP ← Stack address 0FEH
```

3 ADDRESSING MODES

OVERVIEW

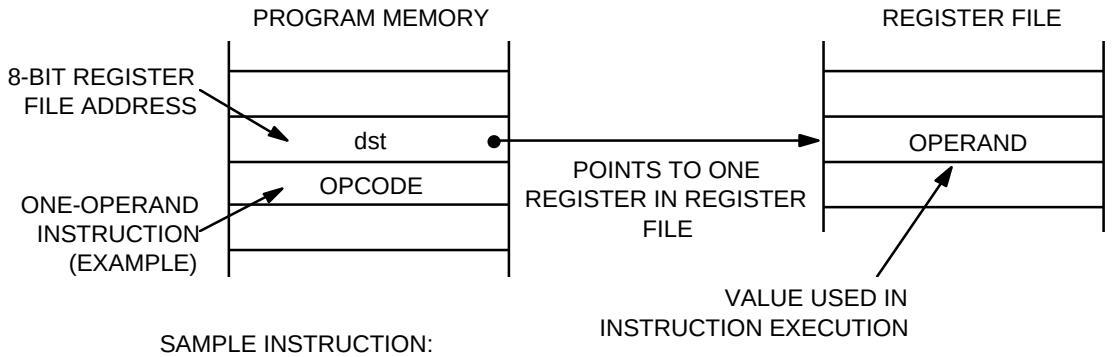
The program counter is used to fetch instructions that are stored in program memory for execution. Instructions indicate the operation to be performed and the data to be operated on. *Addressing mode* is used to determine the location of the data operand. The operands specified in SAM8 instructions may be condition codes, immediate data, or a location in the register file, program memory, or data memory.

The SAM8 instruction set supports seven explicit addressing modes. Not all of these addressing modes are available for each instruction:

— Register (R)
— Indirect Register (IR)
— Indexed (X)
— Direct Address (DA)
— Indirect Address (IA)
— Relative Address (RA)
— Immediate (IM)

REGISTER ADDRESSING MODE (R)

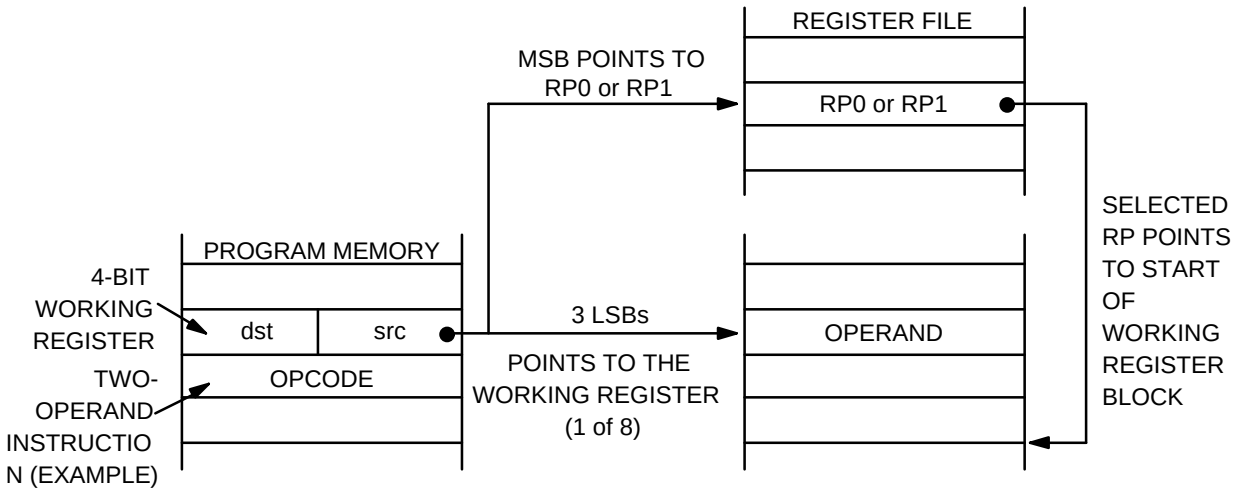
In Register addressing mode, the operand is the content of a specified register or register pair (see Figure 3-1). Working register addressing differs from Register addressing as it uses a register pointer to specify an 8-byte working register space in the register file and an 8-bit register within that space (see Figure 3-2).



SAMPLE INSTRUCTION:

DEC CNTR ; Where CNTR is the label of an 8-bit register address

Figure 3-1. Register Addressing



SAMPLE INSTRUCTION:

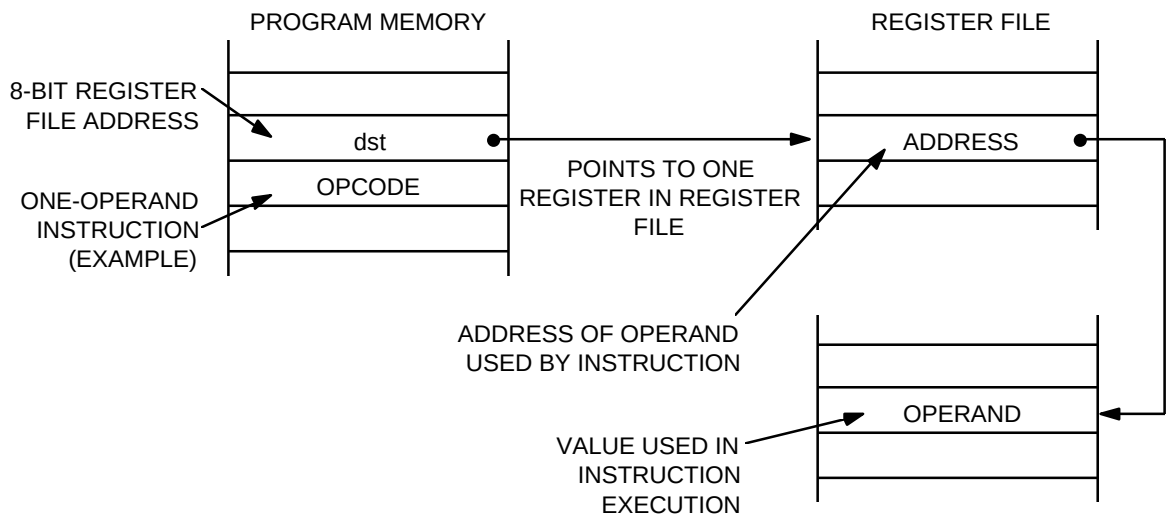
ADD R1,R2 ; Where R1 and R2 are registers in the working register area currently selected

Figure 3-2. Working Register Addressing

INDIRECT REGISTER ADDRESSING MODE (IR)

In Indirect Register (IR) addressing mode, the content of the specified register or register pair is the address of the operand. Depending on the instruction used, the actual address may point to a register in the register file, to program memory (ROM), or to an external memory space, if implemented (see Figures 3-3 through 3-6).

You can use any 8-bit register to indirectly address another register. Any 16-bit register pair can be used to indirectly address another memory location. Remember, however, that locations C0H–FFH in set 1 cannot be accessed using Indirect Register addressing mode.



SAMPLE INSTRUCTION:

RL @SHIFT ; Where SHIFT is the label of an 8-bit register address

Figure 3-3. Indirect Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Continued)

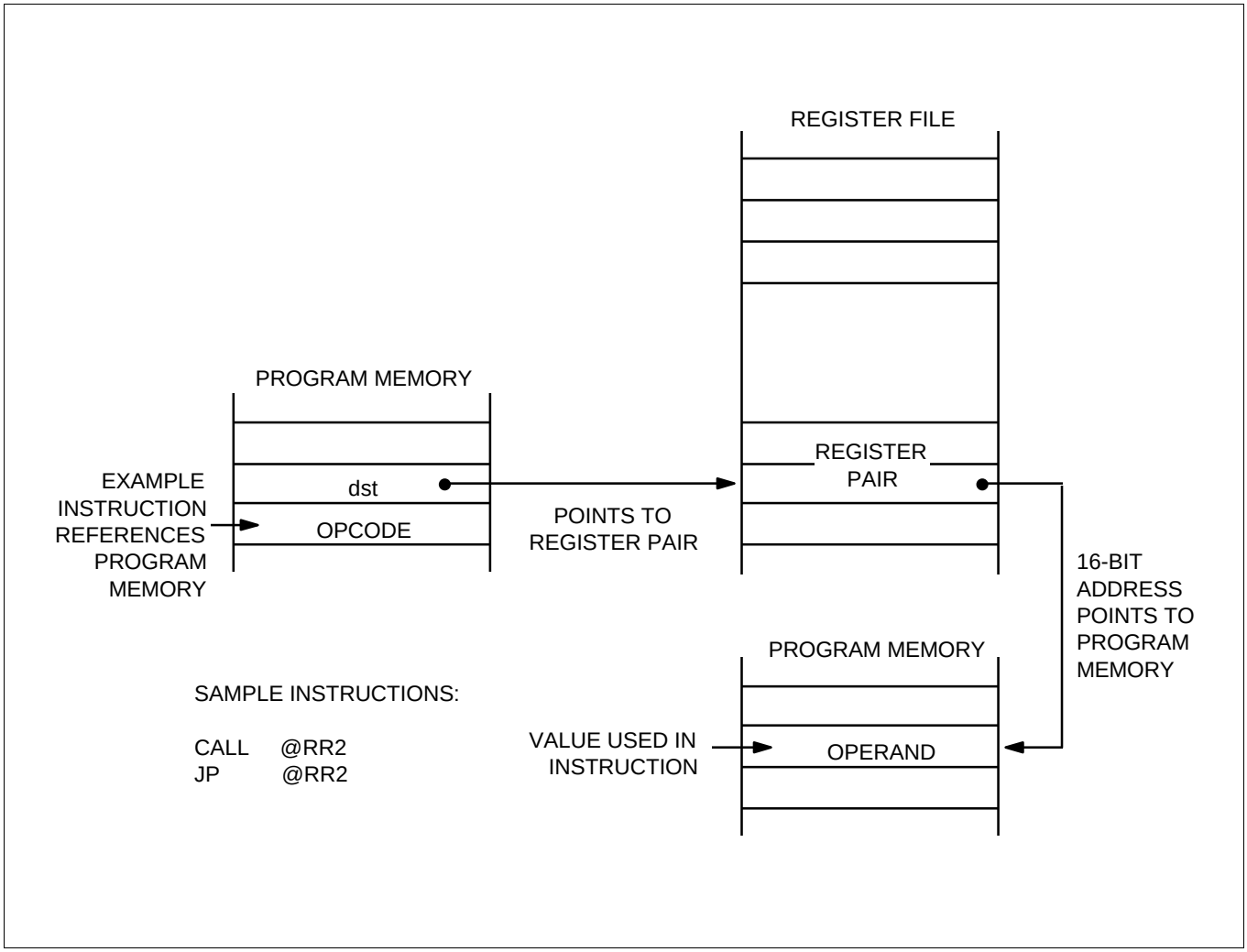


Figure 3-4. Indirect Register Addressing to Program Memory

INDIRECT REGISTER ADDRESSING MODE (Continued)

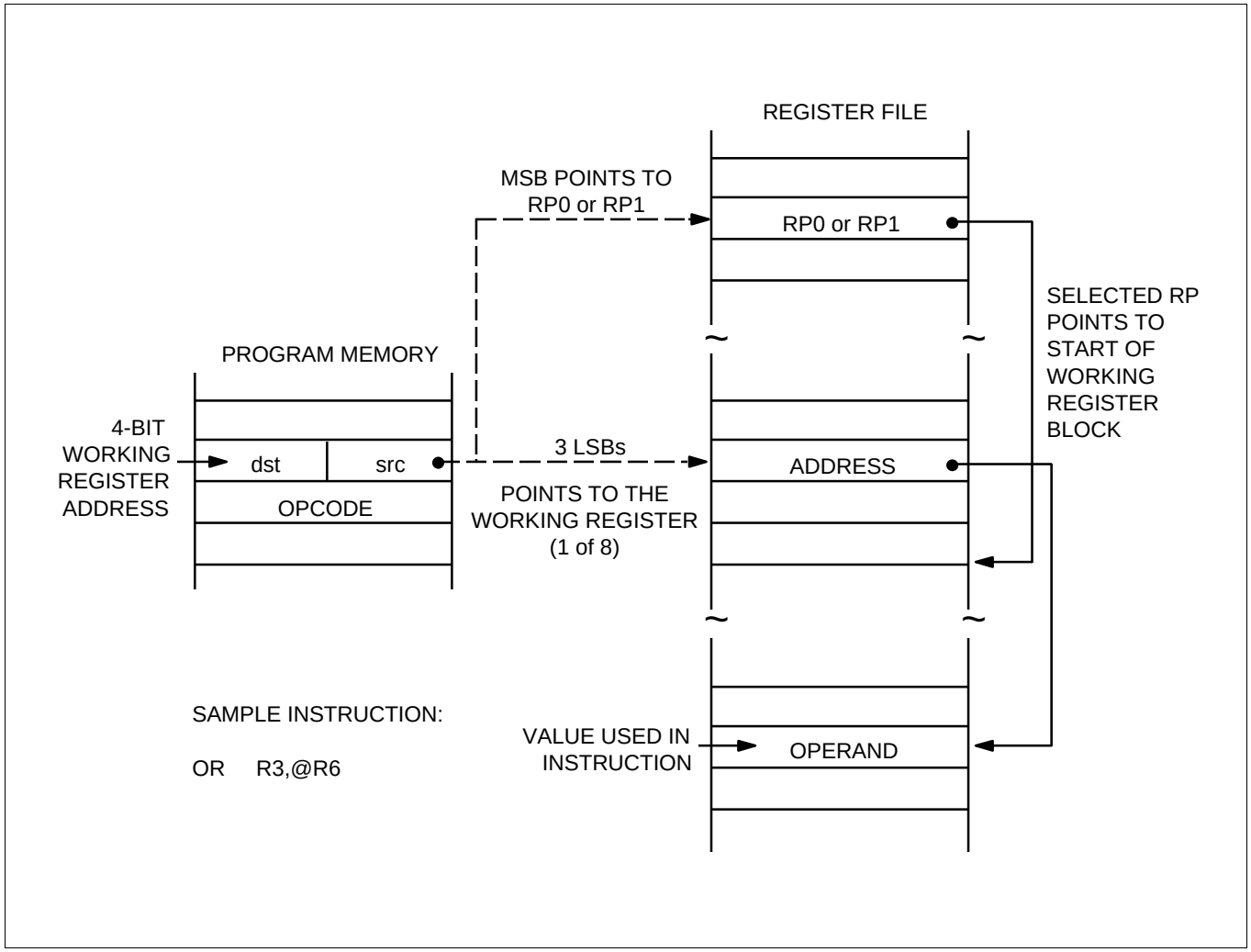
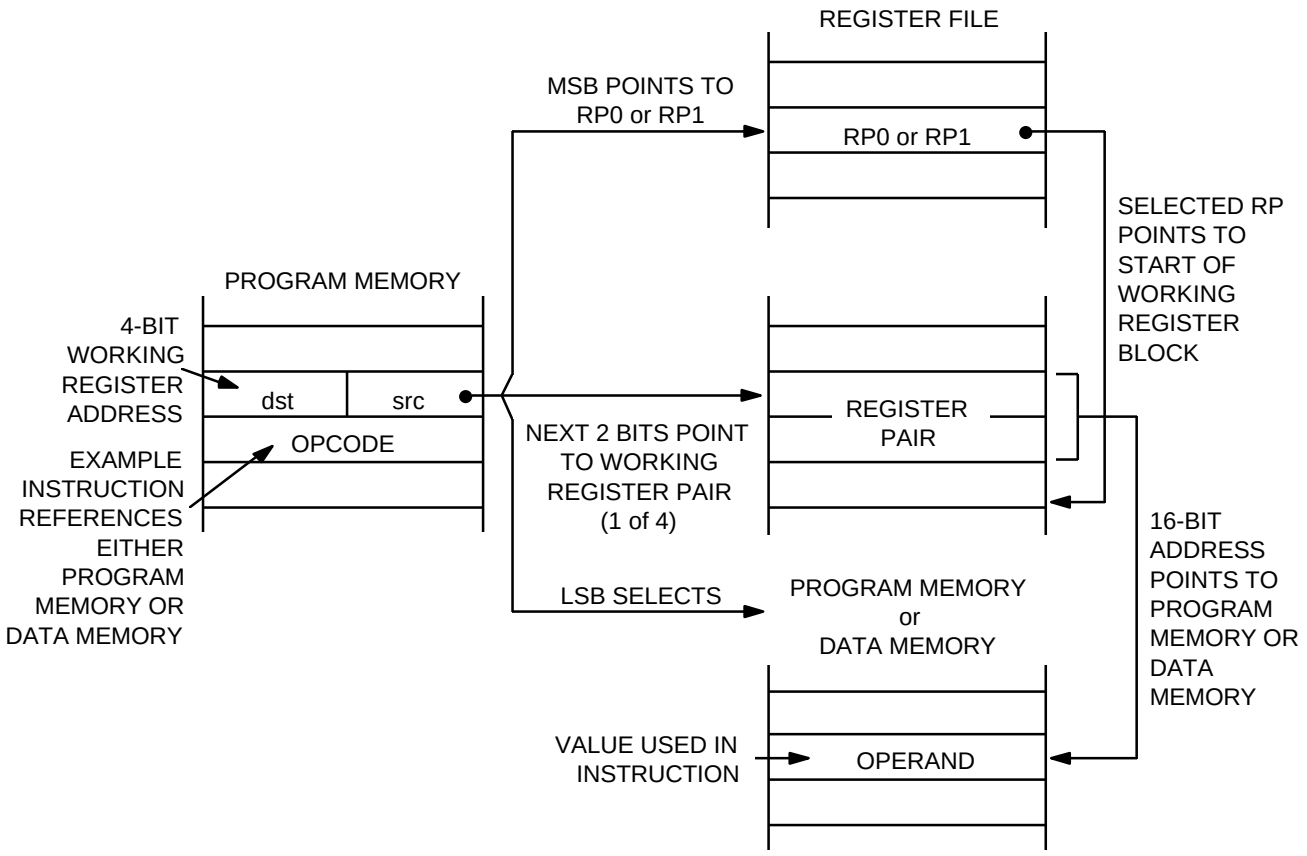


Figure 3-5. Indirect Working Register Addressing to Register File

INDIRECT REGISTER ADDRESSING MODE (Concluded)



SAMPLE INSTRUCTIONS:

```
LDC   R5,@RR6     ;  Program memory access
LDE   R3,@RR14    ;  External data memory access
LDE   @RR4,R8     ;  External data memory access
```

Figure 3-6. Indirect Working Register Addressing to Program or Data Memory

INDEXED ADDRESSING MODE (X)

Indexed (X) addressing mode adds an offset value to a base address during instruction execution in order to calculate the effective operand address (see Figure 3-7). You can use Indexed addressing mode to access locations in the internal register file or in external memory (if implemented). You cannot, however, access locations C0H–FFH in set 1 using Indexed addressing.

In short offset Indexed addressing mode, the 8-bit displacement is treated as a signed integer in the range from –128 to +127. This applies to external memory accesses only (see Figure 3-8).

For register file addressing, an 8-bit base address provided by the instruction is added to an 8-bit offset contained in a working register. For external memory access, the base address is stored in the working register pair designated in the instruction. The 8-bit or 16-bit offset given in the instruction is then added to the base address (see Figure 3-9).

The only instruction that supports Indexed addressing mode for the internal register file is the Load instruction (LD). The LDC and LDE instructions support Indexed addressing mode for internal program memory and for external data memory (if implemented).

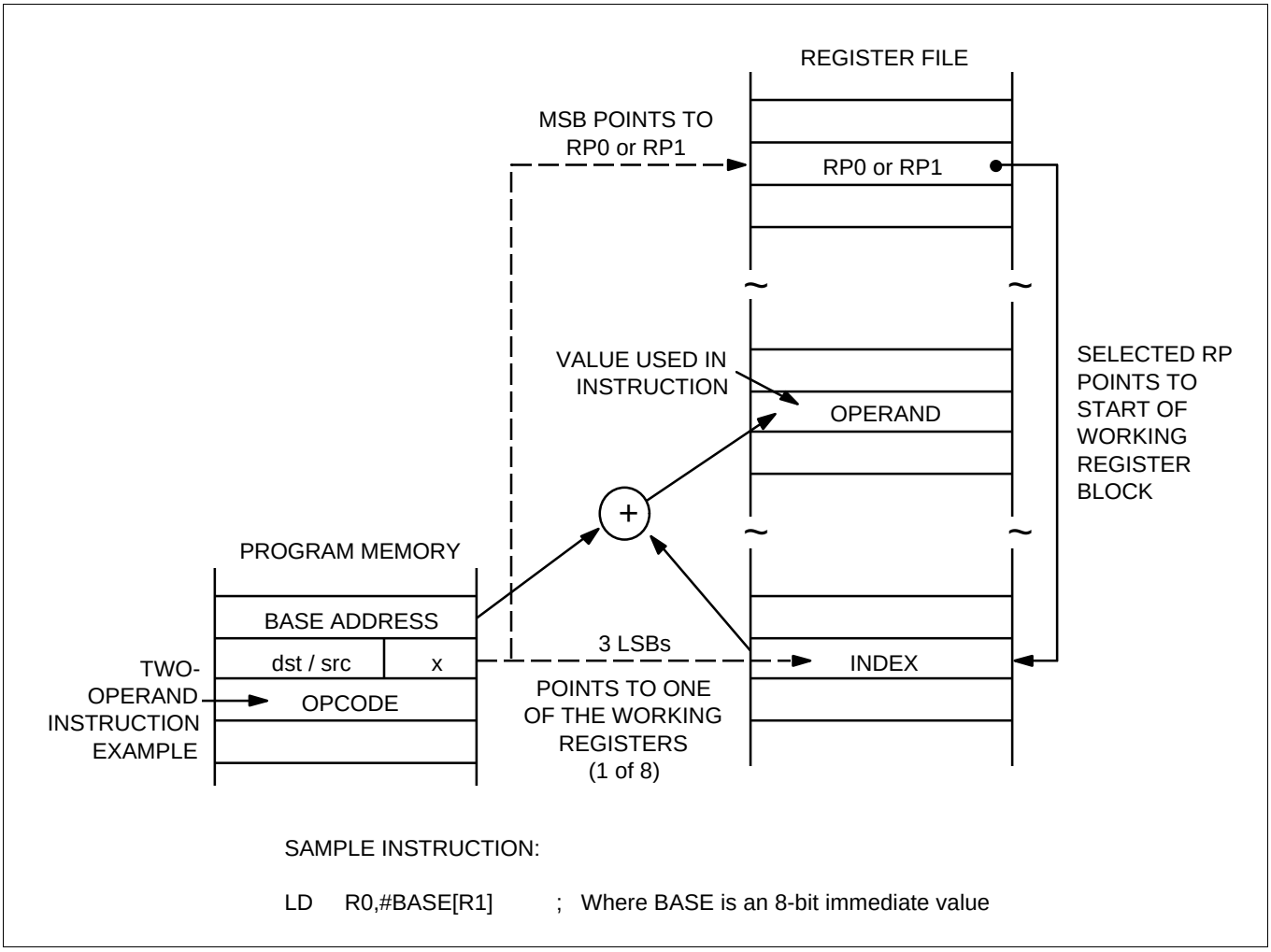
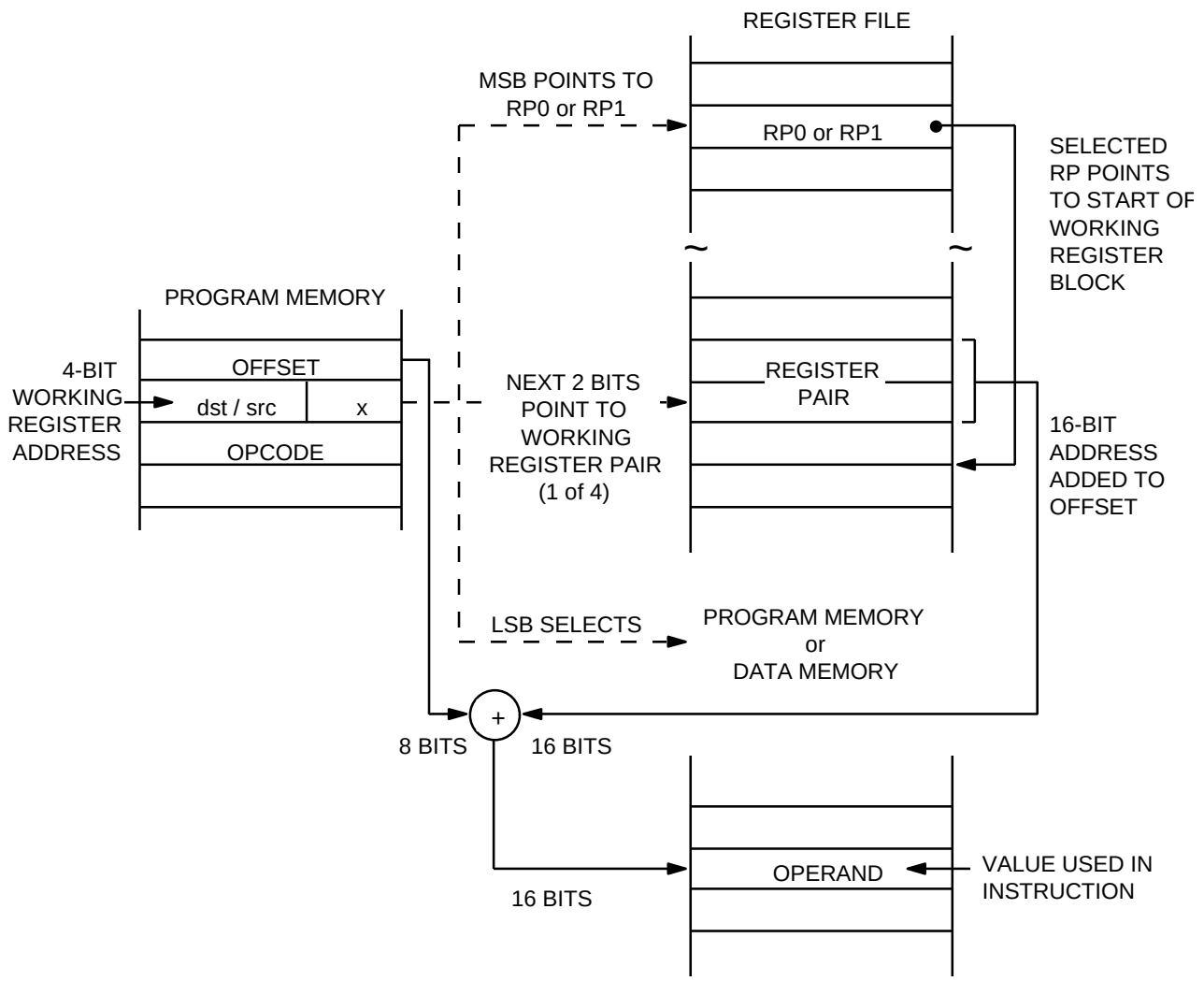


Figure 3-7. Indexed Addressing to Register File

INDEXED ADDRESSING MODE (Continued)



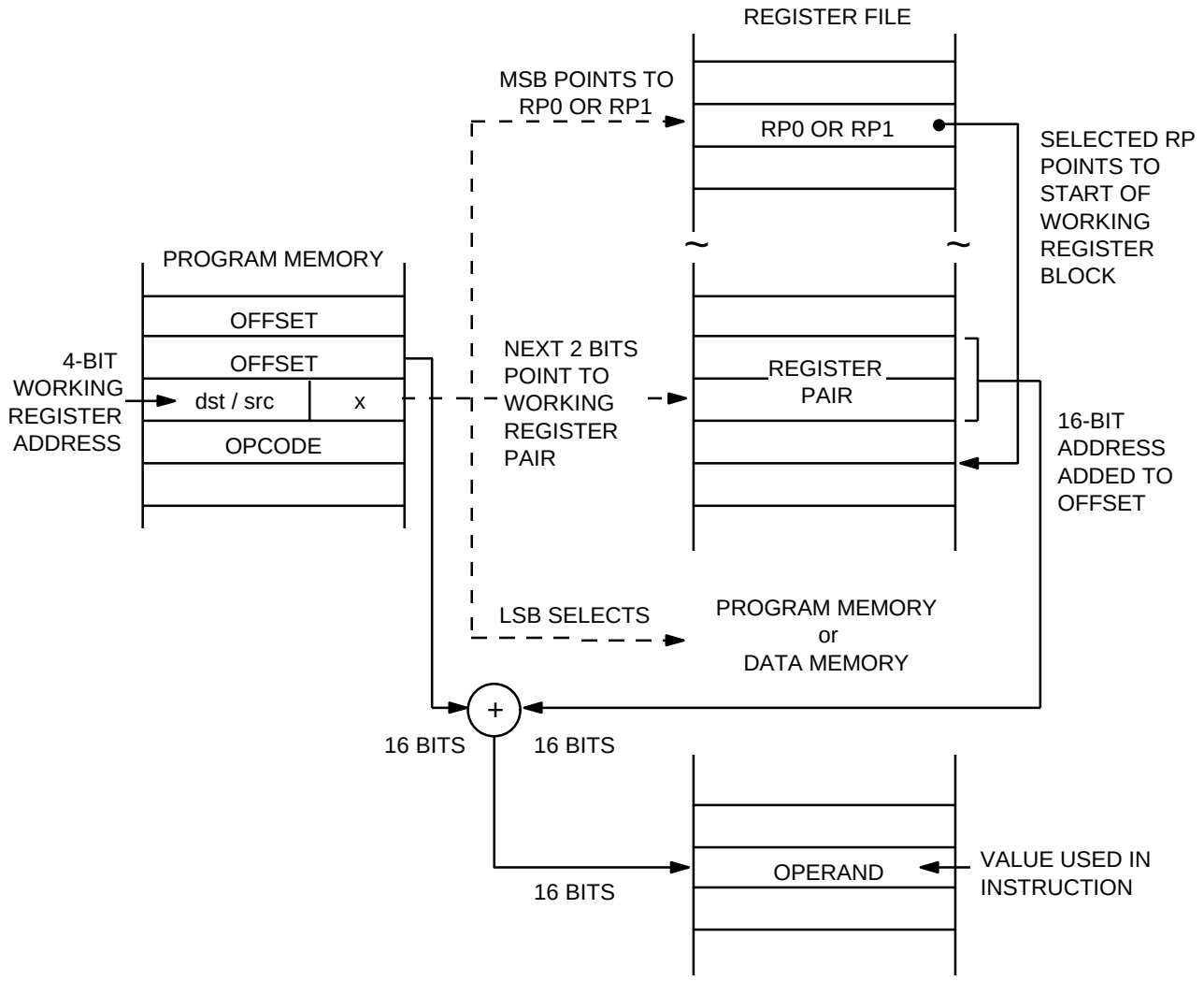
SAMPLE INSTRUCTIONS:

LDC R4,#04H[RR2] ; The values in the program address (RR2 + 04H) are loaded into register R4.

LDE R4,#04H[RR2] ; Identical operation to LDC example, except that external program memory is accessed.

Figure 3-8. Indexed Addressing to Program or Data Memory with Short Offset

INDEXED ADDRESSING MODE (Concluded)



SAMPLE INSTRUCTIONS:

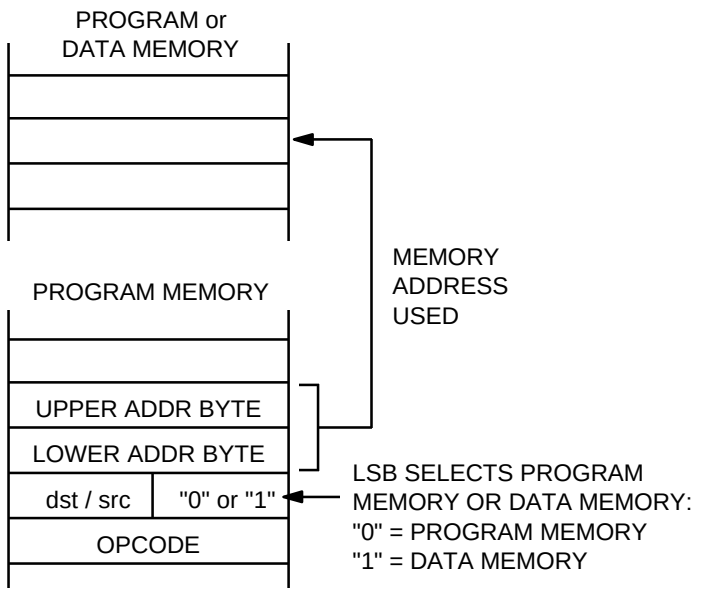
```
LDC   R4,#1000H[RR2]   ; The values in the program address (RR2 + 1000H)
                         are loaded into register R4.

LDE   R4,#1000H[RR2]   ; Identical operation to LDC example, except that
                         external program memory is accessed.
```

Figure 3-9. Indexed Addressing to Program or Data Memory

DIRECT ADDRESS MODE (DA)

In Direct Address (DA) mode, the instruction provides the operand's 16-bit memory address. Jump (JP) and Call (CALL) instructions use this addressing mode to specify the 16-bit destination address that is loaded into the PC whenever a JP or CALL instruction is executed.

The LDC and LDE instructions can use Direct Address mode to specify the source or destination address for Load operations to program memory (LDC) or to external data memory (LDE), if implemented.

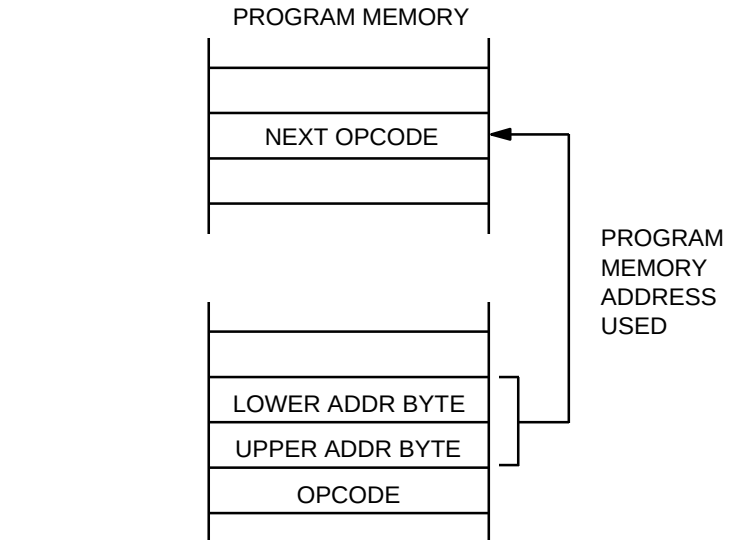


SAMPLE INSTRUCTIONS:

```
LDC   R5,1234H    ;  The values in the program address (1234H)
                     are loaded into register R5.

LDE   R5,1234H    ;  Identical operation to LDC example, except th
                     external program memory is accessed.
```

Figure 3-10. Direct Addressing for Load Instructions

DIRECT ADDRESS MODE (Continued)



SAMPLE INSTRUCTIONS:

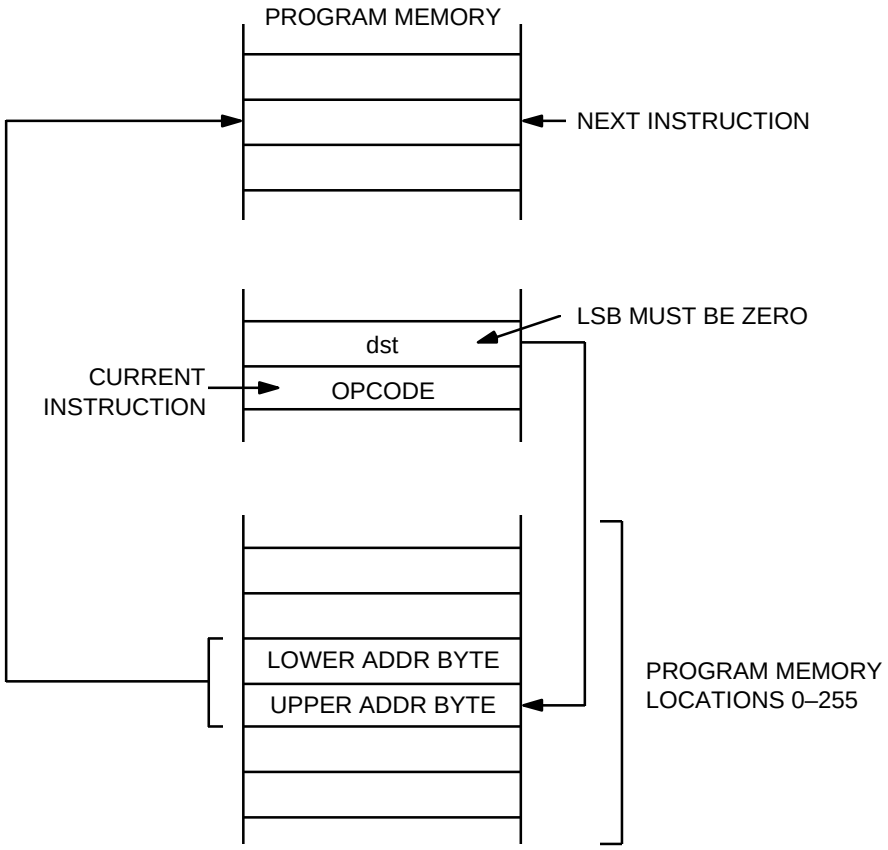
```
JP      C,JOB1      ;  Where JOB1 is a 16-bit immediate address
CALL    DISPLAY     ;  Where DISPLAY is a 16-bit immediate addre
```

Figure 3-11. Direct Addressing for Call and Jump Instructions

INDIRECT ADDRESS MODE (IA)

In Indirect Address (IA) mode, the instruction specifies an address located in the lowest 256 bytes of the program memory. The selected pair of memory locations contains the actual address of the next instruction to be executed. Only the CALL instruction can use Indirect Address mode.

Because Indirect Address mode assumes that the operand is located in the lowest 256 bytes of program memory, only an 8-bit address is supplied in the instruction. The upper bytes of the destination address are assumed to be all zeros.



SAMPLE INSTRUCTION:

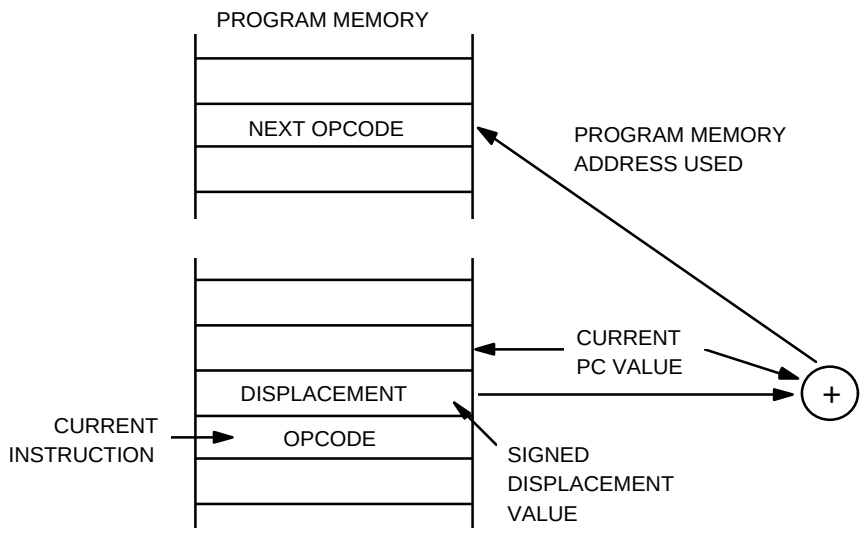
```
CALL    #40H      ;  The 16-bit value in program memory addresses 40H
                     and 41H is the subroutine start address.
```

Figure 3-12. Indirect Addressing

RELATIVE ADDRESS MODE (RA)

In Relative Address (RA) mode, a two's-complement signed displacement between – 128 and + 127 is specified in the instruction. The displacement value is then added to the current PC value. The result is the address of the next instruction to be executed. Before this addition occurs, the PC contains the address of the next instruction immediately following the current instruction.

Several program control instructions use the Relative Address mode to perform conditional jumps. The instructions that support RA addressing are BTJRF, BTJRT, DJNZ, CPIJE, CPIJNE, and JR.



SAMPLE INSTRUCTION:

```
JR    ULT,$+OFFSET        ;  Where OFFSET is a value in the range
                              +127  to  –128
```

Figure 3-13. Relative Addressing

IMMEDIATE MODE (IM)

In Immediate (IM) mode, the operand value used in the instruction is the value supplied in the operand field itself. The operand may be one byte or one word in length, depending on the instruction used. Immediate addressing mode is useful for loading constant values into registers.

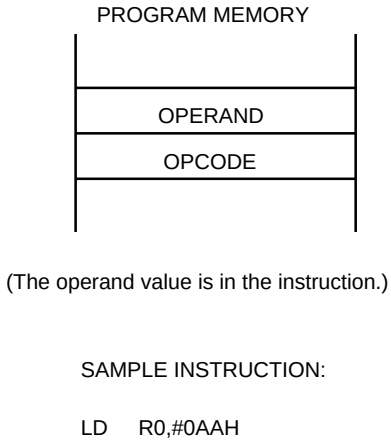


Figure 3-14. Immediate Addressing

4 CONTROL REGISTERS

OVERVIEW

In this chapter, detailed descriptions of the KS88C6216/C6224/C6232 control registers are presented in an easy-to-read format. You can use this chapter as a quick-reference source when writing application programs.

The locations and read/write characteristics of all mapped registers in the KS88C6216/C6224/C6232 register files are presented in Tables 4-1, 4-2, and 4-3. The hardware reset values for these registers are described in Chapter 8, "RESET and Power-Down."

Figure 4-1 illustrates the important features of the standard register description format.

Control register descriptions are arranged in alphabetical order according to register mnemonic. More detailed information about control registers is presented in the context of the specific peripheral hardware descriptions in Part II of this manual.

Table 4-1. Set 1 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Timer M0 counter register	TM0CNT	208	D0H	R (note)
Timer M0 data register	TM0DATA	209	D1H	R (note)
Timer M0 control register	TM0CON	210	D2H	R/W
Basic timer control register	BTCON	211	D3H	R/W
Clock control register	CLKCON	212	D4H	R/W
System flags register	FLAGS	213	D5H	R/W
Register pointer 0	RP0	214	D6H	R/W
Register pointer 1	RP1	215	D7H	R/W
Stack pointer (high byte)	SPH	216	D8H	R/W
Stack pointer (low byte)	SPL	217	D9H	R/W
Instruction pointer (high byte)	IPH	218	DAH	R/W
Instruction pointer (low byte)	IPL	219	DBH	R/W
Interrupt request register	IRQ	220	DCH	R (note)
Interrupt mask register	IMR	221	DDH	R/W
System mode register	SYM	222	DEH	R/W
Page pointer register	PP	223	DFH	R/W

NOTE: You cannot use a read-only register (TM0CNT, TM0DATA, IRQ) as a destination field for the instructions OR, AND, LD, or LDB.

Table 4-2. Set 1, Bank 0 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
Port 0 data register	P0	224	E0H	R/W
Port 1 data register	P1	225	E1H	R/W
Port 2 data register	P2	226	E2H	R/W
Port 3 data register	P3	227	E3H	R/W
Port 0 control register (high byte)	P0CONH	228	E4H	R/W
Port 0 control register (low byte)	P0CONL	229	E5H	R/W
Port 1 control register	P1CON	230	E6H	R/W
Port 2 control register (high byte)	P2CONH	231	E7H	R/W
Port 2 control register (low byte)	P2CONL	232	E8H	R/W
Port 3 control register (high byte)	P3CONH	233	E9H	R/W
Port 3 control register (low byte)	P3CONL	234	EAH	R/W
Port 0 external interrupt control register	P0INT	235	EBH	R/W
Location ECH is not mapped				
Sync control register 0	SYNCON0	237	EDH	R/W
Sync control register 1	SYNCON1	238	EEH	R/W
Sync control register 2	SYNCON2	239	EFH	R/W
Sync port read data register	SYNCRD	240	F0H	R (note)
Timer M1 counter register (high byte)	TM1CNTH	241	F1H	R (note)
Timer M1 counter register (low byte)	TM1CNTL	242	F2H	R (note)
Timer M1 data register (high byte)	TM1DATAH	243	F3H	R (note)
Timer M1 data register (low byte)	TM1DATAL	244	F4H	R (note)
Timer M1 control register	TM1CON	245	F5H	R/W
Timer M2 control register	TM2CON	246	F6H	R/W
A/D converter control register	ADCON	247	F7H	R/W
A/D converter data register	ADDATA	248	F8H	R (note)
Locations F9H–FCH are not mapped				
Basic timer counter register	BTCNT	253	FDH	R (note)
External memory timing register	EMT	254	FEH	R/W
Interrupt priority register	IPR	255	FFH	R/W

NOTE: You cannot use a read-only register (SYNCRD, TM1CNTH, TM1TNCL, TM1DATAH, TM1DATAL, ADDATA, BTCNT) as a destination field for the instructions OR, AND, LD, or LDB.

Table 4-3. Set 1, Bank 1 Registers

Register Name	Mnemonic	Decimal	Hex	R/W
PWM 0 data register	PWM0	224	E0H	R/W
PWM 1 data register	PWM1	225	E1H	R/W
PWM 2 data register	PWM2	226	E2H	R/W
PWM 3 data register	PWM3	227	E3H	R/W
PWM 4 data register	PWM4	228	E4H	R/W
PWM 5 data register	PWM5	229	E5H	R/W
PWM 6 data register	PWM6	230	E6H	R/W
PWM control register	PWMCON	231	E7H	R/W
PWM counter register	PWMCNT	232	E8H	R (note)
DDC control register	DCON	233	E9H	R/W
DDC address register 0	DAR0	234	EAH	R/W
DDC clock control register	DCCR	235	EBH	R/W
DDC control/status register 0	DCSR0	236	ECH	R/W
DDC control/status register 1	DCSR1	237	EDH	R/W
DDC address register 1	DAR1	238	EEH	R/W
Transmit prebuffer data register	TBDR	239	EFH	R/W
Receive prebuffer data register	RBDR	240	F0H	R (note)
DDC data shift register	DDSR	241	F1H	R/W
IIC-bus clock control register	ICCR	242	F2H	R/W
IIC-bus control/status register	ICSR	243	F3H	R/W
IIC-bus address register	IAR	244	F4H	R/W
IIC-bus data shift register	IDSR	245	F5H	R/W
Locations F6H–FFH are not mapped				

NOTE: You cannot use a read-only register (PWMCNT, RBDR) as a destination field for the instructions OR, AND, LD, or LDB.

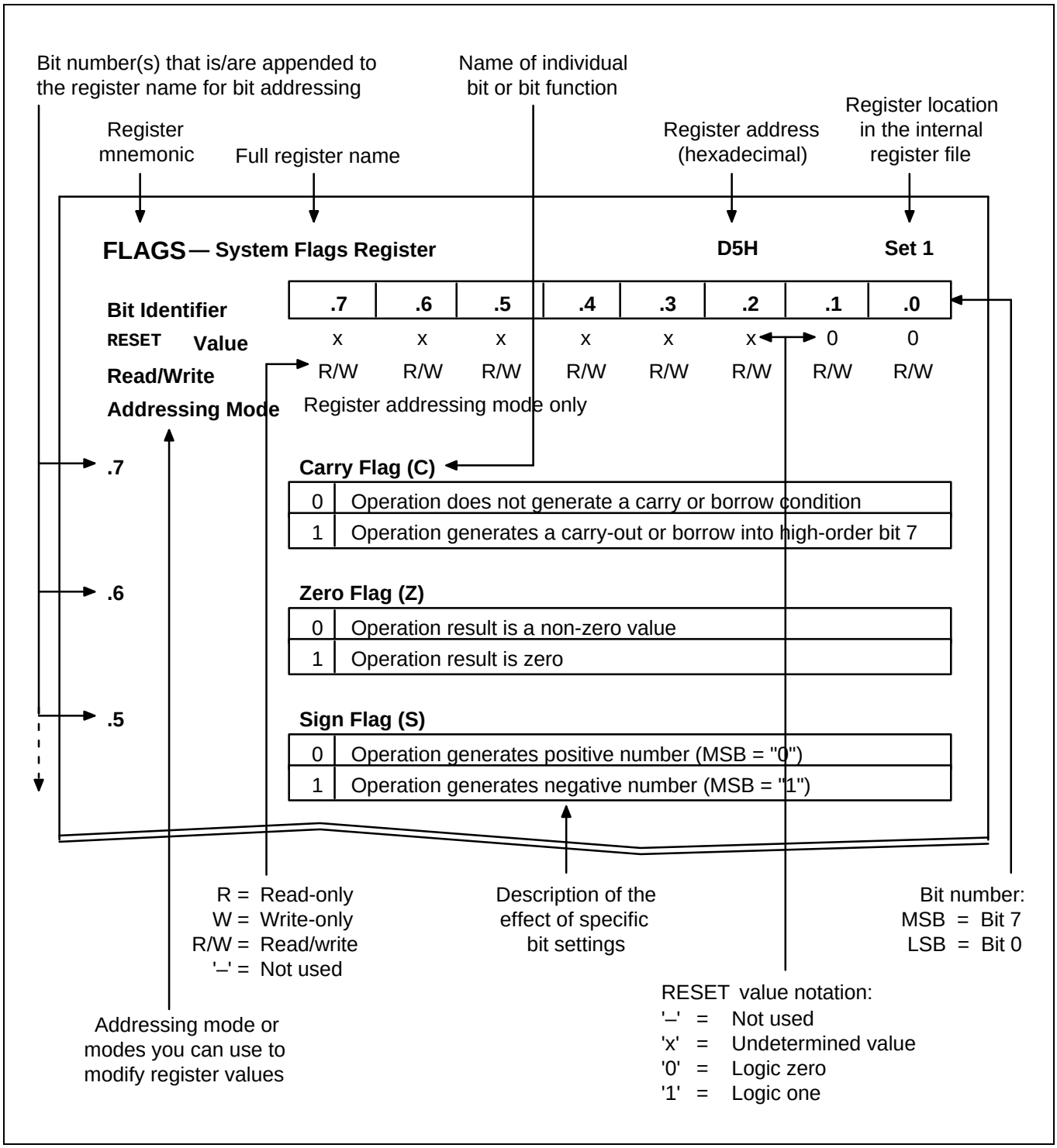


Figure 4-1. Register Description Format

ADCON — A/D Converter Control Register F7H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	0	0	0	1	0	0	0
Read/Write	–	R/W	R/W	R/W	R	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7

Not used for the KS88C6216/C6224/C6232

.6–.4 **Analog Input Pin Selection Bits**

0	0	0	ADC0 (Port 3.0)
0	0	1	ADC1 (Port 3.1)
0	1	0	ADC2 (Port 3.2)
0	1	1	ADC3 (Port 3.3)
1	0	0	Not used for the KC88C6216/C6224/C6232
• • •			"
1	1	1	Not used for the KC88C6216/C6224/C6232

.3 **End-of Conversion (EOC) Flag (read-only)**

0	Conversion not complete
1	Conversion is complete

.2 and .1 **Clock Source Selection Bits**

0	0	$f_{OSC}/16$
0	1	$f_{OSC}/8$
1	0	$f_{OSC}/4$
1	1	f_{OSC}

.0 **Start or Enable Bit**

0	Disable operation
1	Start operation

BTCON — Basic Timer Control Register D3H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.4 **Watchdog Timer Function Disable Bits**

1	0	1	0	Disable watchdog timer function
Others				Enable watchdog timer function

.3 and .2 **Basic Timer Input Clock Selection Bits**

0	0	$f_{OSC}/4096$
0	1	$f_{OSC}/1024$
1	0	$f_{OSC}/128$
1	1	Invalid setting; not used for the KS88C6216/C6224/C6232

.1 **Basic Timer Counter Clear Bit** (1)

0	No effect
1	Clear the basic timer counter value

.0 **Clock Frequency Divider Clear Bit for Basic Timer and Timer M0** (2)

0	No effect
1	Clear basic timer and timer M0 frequency dividers

NOTES:
1. When you write a “1” to BTCON.1, the basic timer counter value is cleared to "00H". Immediately after the write operation, the BTCON.1 value is automatically cleared to “0”.
2. When you write a "1" to BTCON.0, the corresponding frequency divider is cleared to "00H". Immediately after the write operation, the BTCON.0 value is automatically cleared to "0".

CLKCON — System Clock Control Register D4H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 **Oscillator IRQ Wake-up Function Enable Bit**

0	Enable IRQ for main system oscillator wake-up in power-down mode
1	Disable IRQ for main system oscillator wake-up in power-down mode

.6 and .5 **Main Oscillator Stop Control Bits**

0	0	No effect
0	1	No effect
1	0	Stop main oscillator
1	1	No effect

.4 and .3 **CPU Clock (System Clock) Selection Bits** (1)

0	0	Divide by 16 (f_{OSC}/16)
0	1	Divide by 8 (f_{OSC}/8)
1	0	Divide by 2 (f_{OSC}/2)
1	1	Non-divided clock (f_{OSC}) (2)

.2–.0 **Subsystem Clock Selection Bits** (3)

1	0	1	Invalid setting for KS88C6216/C6224/C6232
Others			Select main system clock (MCLK)

NOTES:
1. After a reset, the slowest clock (divided by 16) is selected as the system clock. To select faster clock speeds, load the appropriate values to CLKCON.3 and CLKCON.4.
2. If the oscillator frequency is higher than 12 MHz, this selection is invalid.
3. These selection bits are required only for systems that have a main clock and a subsystem clock. KS88C6216/C6224/C6232 use only the main oscillator clock circuit. For this reason, the setting "101B" is invalid.

DAR0 — DDC Address Register 0 EAH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	1	0	1	0	0	0	0	–
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	–
Addressing Mode	Register addressing mode only							

.7–.1 **Slave Address**

These bits are operate only when receive the slave address. Read enable anytime. Write enable when DCSR0.4 is "0".

.0 Not used for the KS88C6216/C6224/C6232

DAR1 — DDC Address Register 1 EEH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	–
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	–
Addressing Mode	Register addressing mode only							

.7–.1 **Slave Address**

These bits are operate only when receive the slave address. Read enable anytime. Write enable when DCSR0.4 is "0".

.0 Not used for the KS88C6216/C6224/C6232

DCCR — DDC Clock Control Register EBH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7

Transmit acknowledgement enable mode when this bit is "1".

.6 **Tx Clock Selection Bit**

0	$f_{OSC}/16$
1	$f_{OSC}/512$

.5 **DDC Module Interrupt Enable Bit**

0	Disable interrupt
1	Enable interrupt

.4 **DDC Module Interrupt Pending Bit**

0	When write "0" to this bit (write "1" has no effect)
0	When DCSR0.4 is "0"
1	When the misplaced condition error is detected (master mode)
1	When general call or slave address match occurred
1	When arbitration lost (master mode or general call)
1	When an 1-byte transmit or receive operation is terminated
1	When the DCON.0 (SCLF) is set, if DCON.1 (DDC1EN) is "1"
1	As soon as the DDC1 mode is enabled after the prebuffer is used

.3–.0 **Transmit Clock 4-Bit Prescaler Bits (CCR3–CCR0)**

SCL clock = IICLK/(CCR < 3: 0 > +1)
where, IICLK is $f_{OSC}/16$ when DCCR.6 is "0"
IICLK is $f_{OSC}/512$ when DCCR.6 is "1"

DCON — DDC Control Register E9H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	–	–	1	0	0	0
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.4

Not used for the KS88C6216/C6224/C6232.

.3 **Tx/Rx Pre-Buffer Data Registers Enable Bit**

0	Normal IIC-bus mode (Pre-buffer data registers are not used.)
1	Pre-buffer data registers enable mode. This bit is set by writing "1" or by a reset.

.2 **DDC Address Match Bit**

0	When start or stop or reset
1	When DDC received address matchs to DAR0 register

.1 **DDC1 Tx Mode Enable Bit**

0	IIC-bus interface mode (SCL pin is also selected) and SCLF interrupt disable
1	DDC1 Tx mode (VCLK pin is also selected) and SCLF interrupt enable

.0 **SCL Pin Falling Edge Detection Flag**

0	SCL pin level remains high after a reset (when read)
0	*This bit can be cleared by S/W written "0" (when write)*
1	Falling edge can be detected at the SCL pin after a reset or after this flag is cleared by software (when read)
1	*No effect (when write)*

DCSR0 — DDC Control/Status Register 0 ECH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
Addressing Mode	Register addressing mode only							

.7–.6 Master/Slave, Tx/Rx Mode Selection Bits

0	0	Slave receiver mode (Default mode)
0	1	Slave transmitter mode
1	0	Master receiver mode
1	1	Master transmitter mode

.5 Bus Busy Bit

0	IIC-bus is not busy (when read), stop condition generation (when write)
1	IIC-bus is busy (when read), start condition generation (when write)

.4 DDC Module Enable Bit

0	Disable DDC module
1	Enable DDC module

.3 Arbitration Lost Bit

0	Bus arbitration status okay
1	Bus arbitration failed during serial I/O

.2 DDC Address Match Bit

0	When reset or start/stop condition is generated
1	When the received slave address matchs to DAR0, DAR1 register or general call

.1 General Call Bit

0	When reset or start/stop condition is generated
1	When the received slave address is "0000000" (general call)

.0 Received Acknowledgement (ACK) Bit

0	ACK is received
1	ACK is not received

NOTE: Bits 3–0 are read only.

DCSR1 — DDC Control/Status Register 1 EDH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	–	–	–	0	1	0
Read/Write	–	–	–	–	–	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.3

Not used for the KS88C6216/C6224/C6232

.2 **Misplaced Condition Error Bit**

0	When a valid start is detected, or it is reset
0	When it writes "0" to this bit
1	When a misplaced start or stop condition is detected

.1 **Data Buffer Empty Status Bit**

0	When the CPU writes the transmitted data into the TBDR register
1	When the data of the TBDR register is loaded to the DDSR register or STOP condition

.0 **Data Buffer Full Status Bit**

0	When the CPU reads the received data from the RBDR register or STOP condition
1	When the data or matched address is transferred from the DDSR register to the RBDR register

DDSR — DDC Data Shift Register F1H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0	Write enable when DCSR0.4 is "1" and DCON.3 is "0". Read enable anytime

EMT — External Memory Timing Register FEH Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	1	1	1	1	1	0	–
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	–
Addressing Mode	Register addressing mode only							

.7 **External WAIT Input Function Enable Bit**

0	Disable WAIT input function for external device
1	Enable WAIT input function for external device

.6 **Slow Memory Timing Enable Bit**

0	Disable slow memory timing
1	Enable slow memory timing

.5 and .4 **Program Memory Automatic Wait Control Bits**

0	0	No wait (Normal Operation)
0	1	Wait one cycle
1	0	Wait two cycles
1	1	Wait three cycles

.3 and .2 **Data Memory Automatic Wait Control Bits**

0	0	No wait (Normal Operation)
0	1	Wait one cycle
1	0	Wait two cycles
1	1	Wait three cycles

.1 **Stack Area Selection Bit**

0	Select internal register file area
1	Select external data memory area

.0 Not used for the KS88C6216/C6224/C6232

NOTE: As external peripheral interface is not implemented in KS88C6216/C6224/C6232, EMT register is not used. The program initialization routine should clear the EMT register to "00H" after a reset. Modification of EMT values during the normal operation may cause a system malfunction.

FLAGS — System Flags Register D5H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 Carry Flag (C)

0	Operation does not generate a carry or borrow condition
1	Operation generates a carry-out or borrow into high-order bit 7

.6 Zero Flag (Z)

0	Operation result is a non-zero value
1	Operation result is zero

.5 Sign Flag (S)

0	Operation generates a positive number (MSB = "0")
1	Operation generates a negative number (MSB = "1")

.4 Overflow Flag (V)

0	Operation result is $\leq$ +127 or $\geq$ –128
1	Operation result is > +127 or < –128

.3 Decimal Adjust Flag (D)

0	Add operation completed
1	Subtraction operation completed

.2 Half-Carry Flag (H)

0	No carry-out of bit 3 or no borrow into bit 3 by addition or subtraction
1	Addition generated carry-out of bit 3 or subtraction generated borrow into bit 3

.1 Fast Interrupt Status Flag (FIS)

0	Cleared automatically during an interrupt return (IRET)
1	Automatically set to logic one during a fast interrupt service routine

.0 Bank Address Selection Flag (BA)

0	Bank 0 is selected (by executing the instruction SB0)
1	Bank 1 is selected (by executing the instruction SB1)

IAR — Multi-Master IIC-Bus Address Register EBH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	–
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	–
Addressing Mode	Register addressing mode only							

.7–.1 **7-Bit Slave Address Bits**

These bits are operated only when receive the slave address and general call.
Write enable when ICSR, 4 is “0”, but read enable anytime.

.0 Not used for the KS88C6216/C6224/C6232

ICCR — IIC-Bus Clock Control Register F2H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 **Acknowledgment signal Enable/Disable Bit**

0	Disable Acknowledgment signal
1	Enable Acknowledgment signal

.6 **Transmit Clock Selection Bit**

0	$f_{OSC}/16$
1	$f_{OSC}/512$

.5 **Multi-Master IIC-Bus Transmit/Receive Interrupt Enable Bit**

0	Disable interrupt
1	Enable interrupt

.4 **Multi-Master IIC-Bus Transmit/Receive Interrupt Pending Bit**

0	When write "0" to this bit or when ICSR.4 is "0"
1	When 1-byte transmit/receive is terminated, general call or slave address match occurred, or arbitration lost

.3–.0 **ICCR3–ICCR0: Transmit Clock 4-bit Prescaler Bits**

SCL clock = IICLK/ICCR<3:0> + 1
Where, IICLK = $f_{OSC}/16$ when ICCR.6 is “0”, IICLK = $f_{OSC}/512$ when ICCR.6 is “1”

ICSR— IIC-Bus Control/Status Register F3H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R	R	R	R
Addressing Mode	Register addressing mode only							

.7 and .6 **Master/Slave, Transmit/Receive Mode Selection Bits**

0	0	Slave receiver mode (Default mode)
0	1	Slave transmitter mode
1	0	Master receiver mode
1	1	Master transmitter mode

.5 **IIC-Bus Busy Bit**

0	IIC-bus is not busy (when read), IIC-bus interface stop signal generation (when write)
1	IIC-bus is busy (when read), IIC-bus interface start signal generation (when write)

.4 **IIC-Bus Interface Module Enable Bit**

0	Disable IIC-bus data transmit/receive
1	Enable IIC-bus data transmit/receive

.3 **Arbitration Lost Bit**

This bit is set by H/W when the serial I/O interface, in master transmit mode, loses a bus arbitration procedure. In slave mode this flag is set to "1" when ICSR.5 is "1" and ICSR.2 is "0".

.2 **Address Match Bit**

0	When START/STOP/RESET condition was generated
1	When the received slave address matches to IAR register or general call

.1 **General Call Bit**

0	When START/STOP condition was generated
1	When the received slave address is "0000000" (general call)

.0 **Received Acknowledge (ACK) Bit**

0	Acknowledgment signal (ACK) was received
1	Acknowledgment signal (ACK) was not received

IDSR — Multi-Master IIC-Bus Tx/Rx Data Shift Register F3H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0 **Multi Master IIC-bus Transmit/Receive Data Shift Bits**

Write enable when ICSR.4 "1", but Read enable anytime

IMR — Interrupt Mask Register DDH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	x	x	x	x	x	x	x
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7

Not used for the KS88C6216/C6224/C6232

.6 **Interrupt Level 6 (IRQ6) Enable Bit; Multi Master IIC-Bus Interrupt**

0	Disable IRQ6 interrupt
1	Enable IRQ6 interrupt

.5 **Interrupt Level 5 (IRQ5) Enable Bit; Vsync Detection Interrupt**

0	Disable IRQ5 interrupt
1	Enable IRQ5 interrupt

.4 **Interrupt Level 4 (IRQ4) Enable Bit; P0.1 or P0.2 External Interrupt (INT1, INT2)**

0	Disable IRQ4 interrupt
1	Enable IRQ4 interrupt

.3 **Interrupt Level 3 (IRQ3) Enable Bit; P0.0 External Interrupt (INT0)**

0	Disable IRQ3 interrupt
1	Enable IRQ3 interrupt

.2 **Interrupt Level 2 (IRQ2) Enable Bit; DDC (Multi-Master IIC-bus) Interrupt**

0	Disable IRQ2 interrupt
1	Enable IRQ2 interrupt

.1 **Interrupt Level 1 (IRQ1) Enable Bit; Timer 2 Interval/Timer 1 Capture/Overflow Interrupt**

0	Disable IRQ1 interrupt
1	Enable IRQ1 interrupt

.0 **Interrupt Level 0 (IRQ0) Enable Bit; Timer 0 Overflow/Capture Interrupt**

0	Disable IRQ0 interrupt
1	Enable IRQ0 interrupt

NOTE: Only the levels IRQ0–IRQ6 are implemented in the KS88C6216/C6224/C6232 interrupt structure.

IPH — Instruction Pointer (High Byte) DAH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0 **Instruction Pointer Address (High Byte)**

The high-byte instruction pointer value is the upper eight bits of the 16-bit instruction pointer address (IP15–IP8). The lower byte of the IP address is located in the IPL register (DBH).

IPL — Instruction Pointer (Low Byte) DBH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0 **Instruction Pointer Address (Low Byte)**

The low-byte instruction pointer value is the lower eight bits of the 16-bit instruction pointer address (IP7–IP0). The upper byte of the IP address is located in the IPH register (DAH).

IPR — Interrupt Priority Register FFH Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	–	x	x	x	x	x	x
Read/Write	R/W	–	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7, .4, and .1 **Priority Control Bits for Interrupt Groups A, B, and C** (1)

0	0	0	Not used
0	0	1	B > C > A
0	1	0	A > B > C
0	1	1	B > A > C
1	0	0	C > A > B
1	0	1	C > B > A
1	1	0	A > C > B
1	1	1	Not used

.6

Not used for the KS88C6216/C6224/C6232

.5 **Interrupt Group C Priority Control Bit**

0	IRQ5 > IRQ6
1	IRQ6 > IRQ5

.3 **Interrupt Subgroup B Priority Control Bit**

0	IRQ3 > IRQ4
1	IRQ4 > IRQ3

.2 **Interrupt Group B Priority Control Bit**

0	IRQ2 > (IRQ3, IRQ4)
1	(IRQ3, IRQ4) > IRQ2

.0 **Interrupt Group A Priority Control Bit**

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

NOTES:
1. Interrupt group A is IRQ0 and IRQ1. Interrupt group B is IRQ2, IRQ3, and IRQ4. Interrupt group C is IRQ5 and IRQ6.
2. Interrupt levels IRQ6-7 are not used in the KS88C6216/C6224/C6232 interrupt structure. For this reason, IPR.6 (subgroup C) is not used.

IRQ — Interrupt Request Register DCH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	0	0	0	0	0	0	0
Read/Write	–	R	R	R	R	R	R	R
Addressing Mode	Register addressing mode only							

.7

Not used for the KS88C6216/C6224/C6232

.6 **Level 6 (IRQ6) Request Pending Bit; Multi Master IIC-Bus Interrupt**

0	No IRQ6 interrupt pending
1	IRQ6 interrupt is pending

.5 **Level 5 (IRQ5) Request Pending Bit; Vsync Detection Interrupt**

0	No IRQ5 interrupt pending
1	IRQ5 interrupt is pending

.4 **Level 4 (IRQ4) Request Pending Bit; P0.1 or P0.2 External Interrupt**

0	No IRQ4 interrupt pending
1	IRQ4 interrupt is pending

.3 **Level 3 (IRQ3) Request Pending Bit; P0.0 External interrupt**

0	No IRQ3 interrupt pending
1	IRQ3 interrupt is pending

.2 **Level 2 (IRQ2) Request Pending Bit; DDC (Multi-Master IIC-Bus) Interrupt**

0	No IRQ2 interrupt pending
1	IRQ2 interrupt is pending

.1 **Level 1 (IRQ1) Request Pending Bit; Timer 2 Interval/Timer 1 Capture/Overflow Interrupt**

0	No IRQ1 interrupt pending
1	IRQ1 interrupt is pending

.0 **Level 0 (IRQ0) Request Pending Bit; Timer 0 Overflow/Capture Interrupt**

0	No IRQ0 interrupt pending
1	IRQ0 interrupt is pending

P0CONH — Port 0 Control Register (High Byte) E4H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6 **P0.7 Mode Selection Bits**

0	0	Input mode
0	1	Multiplexed input mode
1	x	Push-pull output mode

.5 and .4 **P0.6 Mode Selection Bits**

0	0	Input mode
0	1	Multiplexed input mode
1	x	Push-pull output mode

.3 and .2 **P0.5/TM1CK Mode Selection Bits**

0	0	Input mode
0	1	Multiplexed input mode (P0.5, TM1CK)
1	x	Push-pull output mode

.1 and .0 **P0.4/TM0CAP Mode Selection Bits**

0	0	Input mode
0	1	Multiplexed input mode (P0.4, TM0CAP)
1	x	Push-pull output mode

P0CONL — Port 0 Control Register (Low Byte) E5H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6 **P0.3 Mode Selection Bits**

0	0	Input mode
0	1	Input mode
1	0	Input mode
1	1	Push-pull output mode

.5 and .4 **P0.2/INT2 Mode Selection Bits**

0	0	Input mode (P0.2)
0	1	Input mode, rising edge interrupt detection (INT2)
1	0	Input mode, falling edge interrupt detection (INT2)
1	1	Push-pull output mode

.3 and .2 **P0.1/INT1 Mode Selection Bits**

0	0	Input mode (P0.1)
0	1	Input mode, rising edge interrupt detection (INT1)
1	0	Input mode, falling edge interrupt detection (INT1)
1	1	Push-pull output mode

.1 and .0 **P0.0/INT0 Mode Selection Bits**

0	0	Input mode (P0.0)
0	1	Input mode, rising edge interrupt detection (INT0)
1	0	Input mode, falling edge interrupt detection (INT0)
1	1	Push-pull output mode

P0INT — Port 0 External Interrupt Control Register EBH Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	0	0	0	–	0	0	0
Read/Write	–	R/W	R/W	R/W	–	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .3

Not used for the KS88C6216/C6224/C6232

.6 **P0.2 External Interrupt (IRQ4) Pending Flag** (note)

0	No P0.2 external interrupt pending (when read)
0	*Clear P0.2 interrupt pending condition (when write)*
1	P0.2 external interrupt is pending (when read)

.5 **P0.1 External Interrupt (IRQ4) Pending Flag**

0	No P0.1 external interrupt pending (when read)
0	*Clear P0.1 interrupt pending condition (when write)*
1	P0.1 external interrupt is pending (when read)

.4 **P0.0 External Interrupt (IRQ3) Pending Flag**

0	No P0.0 external interrupt pending (when read)
0	*Clear P0.0 interrupt pending condition (when write)*
1	P0.0 external interrupt is pending (when read)

.2 **P0.2 External Interrupt (IRQ4) Enable Bit**

0	Disable P0.2 interrupt
1	Enable P0.2 interrupt

.1 **P0.1 External Interrupt (IRQ4) Enable Bit**

0	Disable P0.1 interrupt
1	Enable P0.1 interrupt

.0 **P0.0 External Interrupt (IRQ3) Enable Bit**

0	Disable P0.0 interrupt
1	Enable P0.0 interrupt

NOTE: Writing a “1” to an interrupt pending flag (P0.2, P0.1, P0.0) has no effect.

P1CON — Port 1 Control Register E6H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	0	0	0	0	0	0
Read/Write	–	–	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6

Not used for the KS88C6216/C6224/C6232

.5 and .4 **P1.2 Mode Selection Bits**

0	x	Input mode
1	0	Push-pull output mode
1	1	N-channel open-drain output mode (5 V load capability)

.3 and .2 **P1.1/SCL1 Mode Selection Bits**

0	x	Input mode
1	0	Push-pull output mode
1	1	N-channel open-drain output mode (SCL1, 5 V load capability)

.1 and .0 **P1.0/SDA1 Mode Selection Bits**

0	x	Input mode
1	0	Push-pull output mode
1	1	N-channel open-drain output mode (SDA1, 5 V load capability)

NOTE: "x" means "don't care".

P2CONH — Port 2 Control Register (High Byte) E7H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6 **P2.7/Csync-I**

0	x	TTL input mode (Csync-I)
1	x	Push-pull output mode

.5 and .4 **P2.6/PWM6 Mode Selection Bits**

0	0	Input mode
0	1	Push-pull output mode
1	0	Push-pull PWM output mode
1	1	N-channel open-drain PWM output mode (5 V load capability)

.3 and .2 **P2.5/PWM5 Mode Selection Bits**

0	0	Input mode
0	1	Push-pull output mode
1	0	Push-pull PWM output mode
1	1	N-channel open-drain PWM output mode (5 V load capability)

.1 and .0 **P2.4/PWM4 Mode Selection Bits**

0	0	Input mode
0	1	Push-pull output mode
1	0	Push-pull PWM output mode
1	1	N-channel open-drain PWM output mode (5 V load capability)

P2CONL — Port 2 Control Register (Low Byte) E8H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6 **P2.3/PWM3 Mode Selection Bits**

0	x	Input mode
1	0	Push-pull output mode
1	1	Push-pull PWM output mode (5 V load capability)

.5 and .4 **P2.2/PWM2 Mode Selection Bits**

0	x	Input mode
1	0	Push-pull output mode
1	1	Push-pull PWM output mode (5 V load capability)

.3 and .2 **P2.1/PWM1 Mode Selection Bits**

0	x	Input mode
1	0	Push-pull output mode
1	1	Push-pull PWM output mode (5 V load capability)

.1 and .0 **P2.0/PWM0 Mode Selection Bits**

0	x	Input mode
1	0	Push-pull output mode
1	1	Push-pull PWM output mode (5 V load capability)

P3CONH — Port 3 Control Register (High Byte) E9H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6 **P3.7 Mode Selection Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode

.5 and .4 **P3.6 Mode Selection Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode

.3 and .2 **P3.5 Mode Selection Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode

.1 and .0 **P3.4 Mode Selection Bits**

0	0	Input mode
0	1	Input mode with pull-up resistor
1	0	Push-pull output mode
1	1	N-channel open-drain output mode

P3CONL — Port 3 Control Register (Low Byte) EAH Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 and .6 **P3.3/ADC3 Mode Selection Bits**

0	0	Input mode
0	1	Analog Input mode (ADC3)
1	x	Push-pull output mode

.5 and .4 **P3.2/ADC2 Mode Selection Bits**

0	0	Input mode
0	1	Analog Input mode (ADC2)
1	x	Push-pull output mode

.3 and .2 **P3.1/ADC1 Mode Selection Bits**

0	0	Input mode
0	1	Analog Input mode (ADC1)
1	x	Push-pull output mode

.1 and .0 **P3.0/ADC0 Mode Selection Bits**

0	0	Input mode
0	1	Analog Input mode (ADC0)
1	x	Push-pull output mode

PP — Page Pointer Register DFH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.4 **Destination Register Page Selection Bits**

0	0	0	0	Destination: page 0
0	0	0	1	Destination: page 1
0	0	1	0	Not used for the KS88C6216/C6224/C6232
• • •				“
1	1	1	1	Not used for the KS88C6216/C6224/C6232

.3–.0 **Source Register Page Selection Bits**

0	0	0	0	Source: page 0
0	0	0	1	Source: page 1
0	0	1	0	Not used for the KS88C6216/C6224/C6232
• • •				“
1	1	1	1	Not used for the KS88C6216/C6224/C6232

PWMCON — PWM Control Register E7H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	–	–	–	–	–
Read/Write	R/W	R/W	R/W	–	–	–	–	–
Addressing Mode	Register addressing mode only							

.7 and .6 **2-Bit Prescaler value for PWM Counter Input Clock**

0	0	Non-divided input clock
0	1	Input clock divided into two
1	0	Input clock divided into three
1	1	Input clock divided into four

.5 **PWM Counter Enable Bit**

0	Stop PWM counter operation (No current leakage)
1	Start (or resume) PWM counter operation

.4–.0

Not used for the KS88C6216/C6224/C6232

RBDR — Receive Pre-Buffer Data Register F0H Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R	R	R	R	R	R	R	R
Addressing Mode	Register addressing mode only							

.7–.0

It is a read-only register. Read enable anytime. This register will be updated after a data byte is received when the DCSR0.2 is "1" and the DCSR1.0 will be "1". The read operation of this register will clear the DCSR1.0. After the DCSR1.0 is cleared, the register can load the received data again and set the DCSR1.0.

RP0 — Register Pointer 0 D6H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	1	1	0	0	0	–	–	–
Read/Write	R/W	R/W	R/W	R/W	R/W	–	–	–
Addressing Mode	Register addressing only							

.7–.3 **Register Pointer 0 Address Value**

Register pointer 0 can independently point to one of the 18 8-byte working register areas in the register file. Using the register pointers, RP0 and RP1, you can select two 8-byte register slices at one time as active working register space. After a reset, RP0 points to address C0H in register set 1, selecting the 8-byte working register slice C0H–C7H.

.2–.0 Not used for the KS88C6216/C6224/C6232

RP1 — Register Pointer 1 D7H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	1	1	0	0	1	–	–	–
Read/Write	R/W	R/W	R/W	R/W	R/W	–	–	–
Addressing Mode	Register addressing only							

.7–.3 **Register Pointer 1 Address Value**

Register pointer 1 can independently point to one of the 18 8-byte working register areas in the register file. Using the register pointers, RP0 and RP1, you can select two 8-byte register slices at one time as active working register space. After a reset, RP1 points to address C8H in register set 1, selecting the 8-byte working register slice C8H–CFH.

.2–.0 Not used for the KS88C6216/C6224/C6232

SPH — Stack Pointer (High Byte) D8H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0 **Stack Pointer Address (High Byte)**

The high-byte stack pointer value is the upper eight bits of the 16-bit stack pointer address (SP15–SP8). The lower byte of the stack pointer value is located in the register SPL (D9H). The SP value is undefined after a reset.

SPL — Stack Pointer (Low Byte) D9H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0 **Stack Pointer Address (Low Byte)**

The low-byte stack pointer value is the lower eight bits of the 16-bit stack pointer address (SP7–SP0). The upper byte of the stack pointer value is located in the register SPH (D8H). The SP value is undefined after a reset.

SYM — System Mode Register DEH Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	–	–	x	x	x	0	0
Read/Write	R/W	–	–	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 **Tri-State External Interface Control Bit** (1)

0	Normal operation (disable tri-state operation)
1	Set external interface lines to high impedance (enable tri-state operation)

.6 and .5

Not used for the KS88C6216/C6224/C6232

.4–.2 **Fast Interrupt Level Selection Bits** (2)

0	0	0	IRQ0
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	Not used for the KS88C6216/C6224/C6232

.1 **Fast Interrupt Enable Bit** (3)

0	Disable fast interrupt processing
1	Enable fast interrupt processing

.0 **Global Interrupt Enable Bit** (4)

0	Disable global interrupt processing
1	Enable global interrupt processing

NOTES:
1. As external interface is not implemented in KS88C6216/C6224/C6232, SYM.7 must always be "0".
2. You can select only one interrupt level at a time for fast interrupt processing.
3. Setting SYM.1 to "1" enables fast interrupt processing for the interrupt level currently selected by SYM.2–SYM.4.
4. After a reset, you must enable global interrupt processing by executing an EI instruction (not by writing a "1" to SYM.0).

SYNCON0 — Sync Processor Control Register 0 EDH Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R	R	R	R	R
Addressing Mode	Register addressing mode only							

.7 **Sync Input Selection (SIS) Bit**

0	Hsync-I input is selected
1	Csync-I input is selected

.6 **Hsync Blanking Enable Bit (When SYNCON.5 = "1")**

0	Disable (Hsync signal by-pass)
1	Enable Hsync blanking automatically (During the Vsync signal extraction period)

.5 **Vsync-O Output Selection (VOS) Bit**

0	Select VsyncI port input
1	Select 5-bit compare output

.4–.0 **5-Bit Counter**

5-bit counter increases when a high level is detected, while an overflow dose not occur (Stop at "11111"). It decreases when a low level is detected, while an underflow does not occur (Stop at "00000")
When SYNCON0.5 is "1": Sync separation and output mode (When counter value increases to "11111", output the high through the MUX and when counter value decreases to "00000", output becomes low. Resume the previous status when "11111" > counter value > "00000")

SYNCON1 — Sync Processor Control Register 1 EEH Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 **Hsync, Csync, Sync-on-Green (SOG) Compare Output Bit**

0	Hsync-I is not the same as Csync-I (no SOG)
1	Hsync-I is the same as Csync-I (SOG)

.6 and .5 **Clamp Signal Generator Selection Bits**

0	0	Inhibit Clamp signal output (Clamp-O)
0	1	($f_{OSC} \times 2$) clock pulse output (250 ns at 8 MHz f_{OSC})
1	0	($f_{OSC} \times 4$) clock pulse output (500 ns at 8 MHz f_{OSC})
1	1	($f_{OSC} \times 8$) clock pulse output (1 µs at 8 MHz f_{OSC})

.4 **"Front Porch"/"Back Porch" Mode Selection Bit**

0	Generate Clamp-O after the falling edge of Hsync ("back porch" mode)
1	Generate Clamp-O after the rising edge of Hsync ("front porch" mode)

.3 **Clamp Signal Output Status Control Bit**

0	Negative polarity
1	Positive polarity

.2 **Vsync-O Status Control Bit**

0	Do not invert (by-pass)
1	Invert output signal

.1 **Hsync-O Status Control Bit**

0	Do not invert (by-pass)
1	Invert output signal

.0 **Hsync Polarity Detection Bit** (note)

0	Negative polarity
1	Positive polarity

NOTE: This bit may not be accurate when the sync signal is not in a transitional condition. To check Hsync polarity, it uses only 24 clocks.

SYNCON2 — Sync Processor Control Register 2 EFH Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 **Unmixed Hsync Detection Bit (When SYNCON.5 is "1")**

0	Mixed Hsync period with Vsync of a composite sync input (It is recommended not to calculate the sync frequency)
1	Stable Hsync only period without Vsync

.6 **Pseudo sync Factory Test Mode Selection Bit**

0	Normal operating mode
1	Factory test mode (pseudo sync generator mode)

.5 **5-Bit Counter Clock Source Selection Bit**

0	$f_{OSC}/2$
1	$f_{OSC}/3$

.4 and .3 **Pseudo Vsync Generator Output Control Bits (Positive polarity only, 1% duty)**

0	0	Normal sync processor operating mode
0	1	66.7 Hz (8 MHz f_{OSC} and SYNCON2.5 = “0”)
1	0	78.4 Hz (8 MHz f_{OSC} and SYNCON2.5 = “0”)
1	1	90.9 Hz (8 MHz f_{OSC} and SYNCON2.5 = “0”)

.2 and .1 **Pseudo Hsync Generator Output Control Bits (Positive polarity only, 10% duty)**

0	0	Normal sync processor operating mode
0	1	40.00 kHz (8 MHz f_{OSC} and SYNCON2.5 = “0”)
1	0	50.00 kHz (8 MHz f_{OSC} and SYNCON2.5 = “0”)
1	1	66.67 kHz (8 MHz f_{OSC} and SYNCON2.5 = “0”)

.0 **Vsync Detection Interrupt Enable Bit**

0	Disable Vsync interrupt
1	Enable Vsync interrupt (interrupt requests triggered by rising edge)

NOTE: Vsync detection interrupt pending conditions are automatically cleared by hardware after the CPU acknowledges the request and the interrupt service routine is initiated.

SYNCRD — Sync Processor Port Read Data Register F0H Set 1, Bank 0

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	–	–	–	–	0	0	0	0
Read/Write	–	–	–	–	R	R	R	R
Addressing Mode	Register addressing mode only							

.7–.4

Not used for the KS88C6216/C6224/C6232

.3 **Vertical Sync Signal Output Data Bit (Vsync-O)**

0	Low data
1	High data

.2 **Horizontal Sync Signal Output Data Bit (Hsync-O)**

0	Low data
1	High data

.1 **Vertical Sync Signal Input Data Bit (Vsync-I)**

0	Low data
1	High data

.0 **Horizontal Sync Signal Input Data Bit (Hsync-I)**

0	Low data
1	High data

TBDR — Transmit Pre-Buffer Data Register EFH Set 1, Bank 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	x	x	x	x	x	x	x	x
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.0

Write enable when DCSR0.4 is "1", read enable anytime.
If DCON.3 (TBDR Enable bit) is "1", the data written into this register will be automatically downloaded to the DDC Data Shift Register (DDSR) and generate the interrupt request when the module detects the calling address is matched and the bit 0 of the received data is "1" and when the data in the DDS register has been transmitted with received acknowledge bit, DCSR0.0 = "0". At this interrupt service routine the CPU must write the next data to the TBDR register to clear DCSR1.1 and for the auto downloading of data to the DDSR register after the data in the DDSR register is transmitted over again with DCSR0.0 = "0". If the master receiver doesn't acknowledge the transmitted data, DCSR0.0 = "1", the module will release the SDA line for master to generate STOP or Repeated START conditions. The data stored in this register will not be downloaded to the DDSR register until next calling from master.
If DCON.3 (TBDR Enable bit) is "0", the module will pull-down the SCL line in the IIC-bus interrupt service routine when the DCSR0.2 is "1". And the module will release the SCL line if the CPU writes a data to the DDSR registers and the interrupt pending bit is cleared.

TM0CON — Timer m0 Control Register D2H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 **Timer M0 Input Clock Selection Bit**

0	$f_{OSC}/128$
1	$f_{OSC}/4$

.6 and .5 **2 Bit Prescaler**

0	0	No division
0	1	Divided by 2
1	0	Divided by 3
1	1	Divided by 4

.4 **Timer M0 Capture Mode Selection Bit**

0	Capture on rising mode
1	Capture on falling mode

.3 **Timer M0 Counter Clear Bit (TM0CLR)**

0	No effect
1	Clear timer M0 counter, TM0CNT (when write)

.2 **Timer M0 Overflow Interrupt Enable Bit (TM0OVINT)** (1)

0	Disable timer M0 overflow interrupt
1	Enable timer M0 overflow interrupt

.1 **Timer M0 Capture Interrupt Enable Bit (TM0INT)**

0	Disable timer M0 interrupt (2)
1	Enable timer M0 interrupt

.0 **Timer M0 Capture Input Selection Bit (TM0CAPSEL)**

0	TM0CAP input pin selection
1	Vsync output path selection from sync-processor

NOTES:

1. When the captured value is #0FFH, the overflow interrupt does not occurred. If the captured value is changed from #0FFH to #00H, the overflow interrupt occurs. When the captured value is #00H, the overflow interrupt occurs first.
2. When the timer M0 interrupt is disabled, the timer M0 overflow interrupt by f_{OSC} can happen.

TM1CON — Timer M1 Control Register F5H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7 **Capture Signal Source Selection Bit**

0	Signal from timer M2 interval time
1	VsyncO from sync-processor

.6 **VsyncO Capture Egde Selection Bit (When TM1CON.7 = "1")**

0	Capture VsyncO (from sync-processor) on rising edge
1	Capture VsyncO (from sync-processor) on falling edge

.5 **Timer M1 Capture Enable Bit**

0	Disable timer M1 capture
1	Enable timer M1 capture

.4 **Timer M1 Counter Clear Bit (TM1CLR; when write)**

0	No effect
1	Clear timer M1 counter

.3 **Timer M1 Capture/Overflow Interrupt Enable Bit (TM1INT)**

0	Disable timer M1 capture/overflow interrupt
1	Enable timer M1 capture/overflow interrupt

.2 **Timer M1 Capture/Overflow Interrupt Pending Bit (PM1PND)**

0	Interrupt is not pending (when read)
0	Clear this pending bit (when write)
1	Interrupt is pending (when read)
1	No effect (when write)

.1–.0 **Timer M1 Clock Input Selection Bit**

0	0	f_{OSC}/4,096
0	1	f_{OSC}/512
1	0	Hsync or CsyncI from sync-processor
1	1	External clock (TM1CK)

TM2CON — Timer M2 Control Register F5H Set 1

Bit Identifier	.7	.6	.5	.4	.3	.2	.1	.0
RESET Value	1	1	1	1	1	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addressing Mode	Register addressing mode only							

.7–.3 **5-bit Prescale Bits (TM2PS4–TM2PS0)**

0	0	0	0	0	No division
0	0	0	0	1	Divide by 2
0	0	0	1	0	Divide by 3
		• • •			• • •
1	1	1	1	1	Divide by 32

.2 **Timer M2 Interrupt Enable Bit (TM2INT)**

0	Disable timer M2 interrupt
1	Enable timer M2 interrupt

.1 and .0 **Timer M2 Capture Interval Time Selection Bits (When TM2CON.5 is "1")**

0	0	Timer M2 interval (bypass)
0	1	Timer M2 interval × 10
1	0	Timer M2 interval × 20
1	1	Timer M2 interval × 30

NOTES:
1. When the timer M1 capture mode is enabled (TM1CON.5 = "1"), the value of 5/2-bit prescaler is changed only in the timer M1 capture interrupt routine.
2. When the timer M1 capture mode is disabled (TM1CON.5 = "0"), the value of 5-bit prescaler is changed only in the timer M2 interval interrupt routine.

5 INTERRUPT STRUCTURE

OVERVIEW

The SAM8 interrupt structure has three basic components: levels, vectors, and sources. The CPU recognizes eight interrupt levels and supports up to 128 interrupt vectors. When a specific interrupt level has more than one vector address, the vector priorities are established in hardware. Each vector can have one or more sources.

Levels

Interrupt levels are the main unit for interrupt priority assignment and recognition. All peripherals and I/O blocks can issue interrupt requests. In other words, peripheral and I/O operations are interrupt-driven. There are eight interrupt levels: IRQ0–IRQ7, also called level 0–level 7. Each interrupt level directly corresponds to an interrupt request number (IRQn). The total number of interrupt levels used in the interrupt structure varies from device to device. The KS88C6216/C6224/C6232 interrupt structure recognizes seven interrupt levels, IRQ0–IRQ6.

The interrupt level numbers 0 through 7 do not necessarily indicate the relative priority of the levels. They are just identifiers for the interrupt levels that are recognized by the CPU. The relative priority of different interrupt levels is determined by settings in the interrupt priority register, IPR. Interrupt group and subgroup logic controlled by IPR settings lets you define more complex priority relationships between different levels.

Vectors

Each interrupt level can have one or more interrupt vectors, or it may have no vector address assigned at all. The maximum number of vectors that can be supported for a given level is 128. (The actual number of vectors used for KS88-series devices will always be much smaller.) If an interrupt level has more than one vector address, the vector priorities are set in hardware. KS88C6216/C6224/C6232 have nine vectors — one corresponding to each of the nine possible interrupt sources.

Sources

A source is any peripheral that generates an interrupt. A source can be an external pin or a counter overflow. Each vector can have several interrupt sources. In the KS88C6216/C6224/C6232 interrupt structure, each source has its own vector address.

When a service routine starts, the respective pending bit should be either cleared automatically by hardware or cleared "manually" by program software. The characteristics of the source's pending mechanism determine which method would be used to clear its respective pending bit.

INTERRUPT TYPES

The three components of the SAM8 interrupt structure described before — levels, vectors, and sources — are combined to determine the interrupt structure of an individual device and to make full use of its available interrupt logic. There are three possible combinations of interrupt structure components, called interrupt types 1, 2, and 3. The types differ in the number of vectors and interrupt sources assigned to each level (see Figure 5-1):

Type 1: One level (IRQn) + one vector (V_1) + one source (S_1)

Type 2: One level (IRQn) + one vector (V_1) + multiple sources ($S_1 – S_n$)

Type 3: One level (IRQn) + multiple vectors ($V_1 – V_n$) + multiple sources ($S_1 – S_n$, $S_{n+1} – S_{n+m}$)

In the KS88C6216/C6224/C6232 microcontrollers, only interrupt types 1 and 3 are implemented.

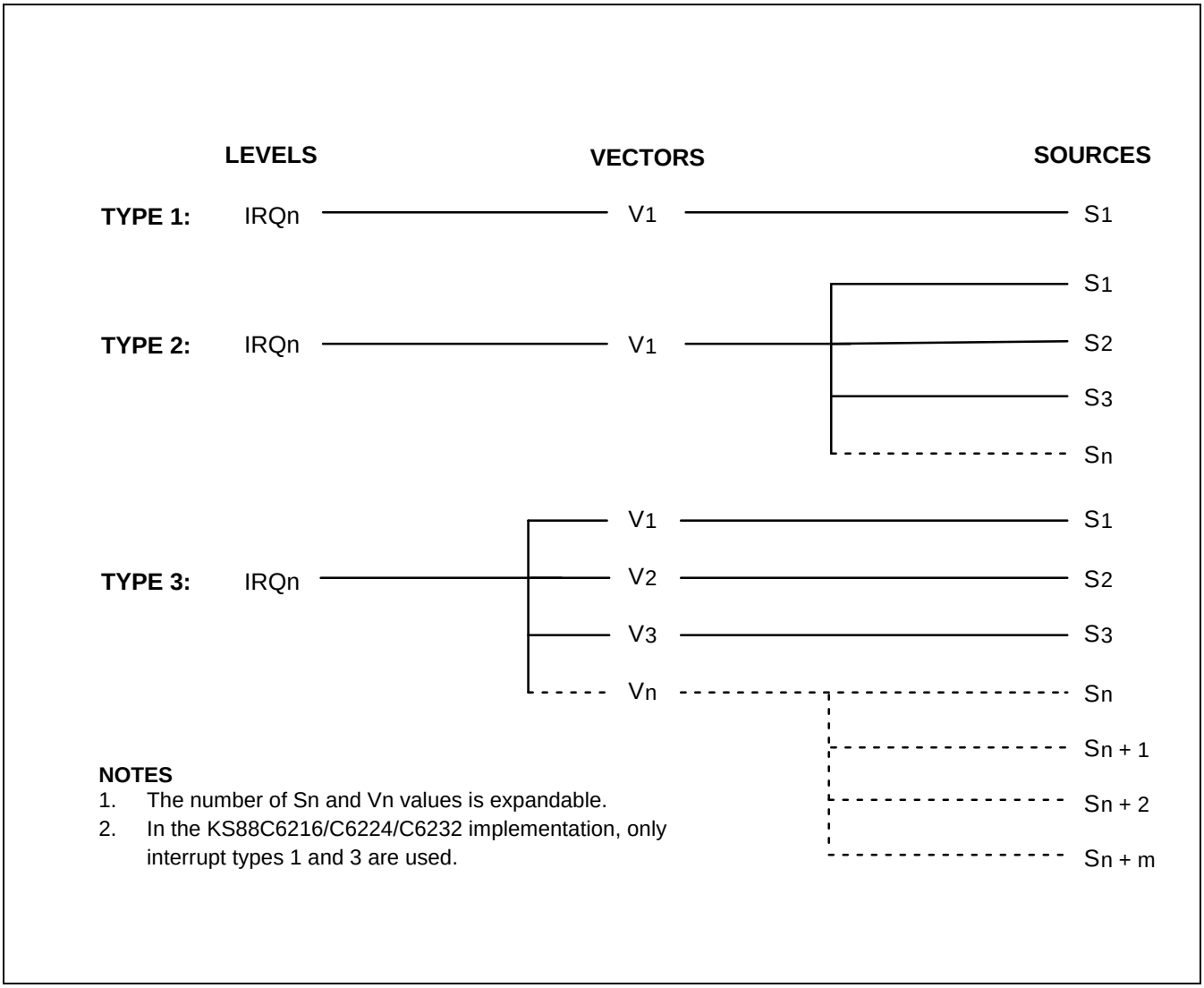


Figure 5-1. KS88-Series Interrupt Types

KS88C6216/C6224/C6232 INTERRUPT STRUCTURE

The KS88C6216/C6224/C6232 microcontrollers support nine interrupt sources. Each interrupt source has a corresponding interrupt vector address. Seven of the eight possible interrupt levels are used in the device-specific interrupt structure, which is shown in Figure 5-2.

When multiple interrupt levels are active, the interrupt priority register (IPR) determines the order in which contending interrupts are to be serviced. If multiple interrupts occur within the same interrupt level, the interrupt with the lowest vector address is usually processed first. (The relative priorities of multiple interrupts within a single level are fixed in hardware.)

When the CPU grants an interrupt request, interrupt processing starts: All other interrupts are disabled and the program counter value and status flags are pushed to stack. The starting address of the service routine is fetched from the appropriate vector address (plus the next 8-bit value to concatenate the full 16-bit address) and the service routine is executed.

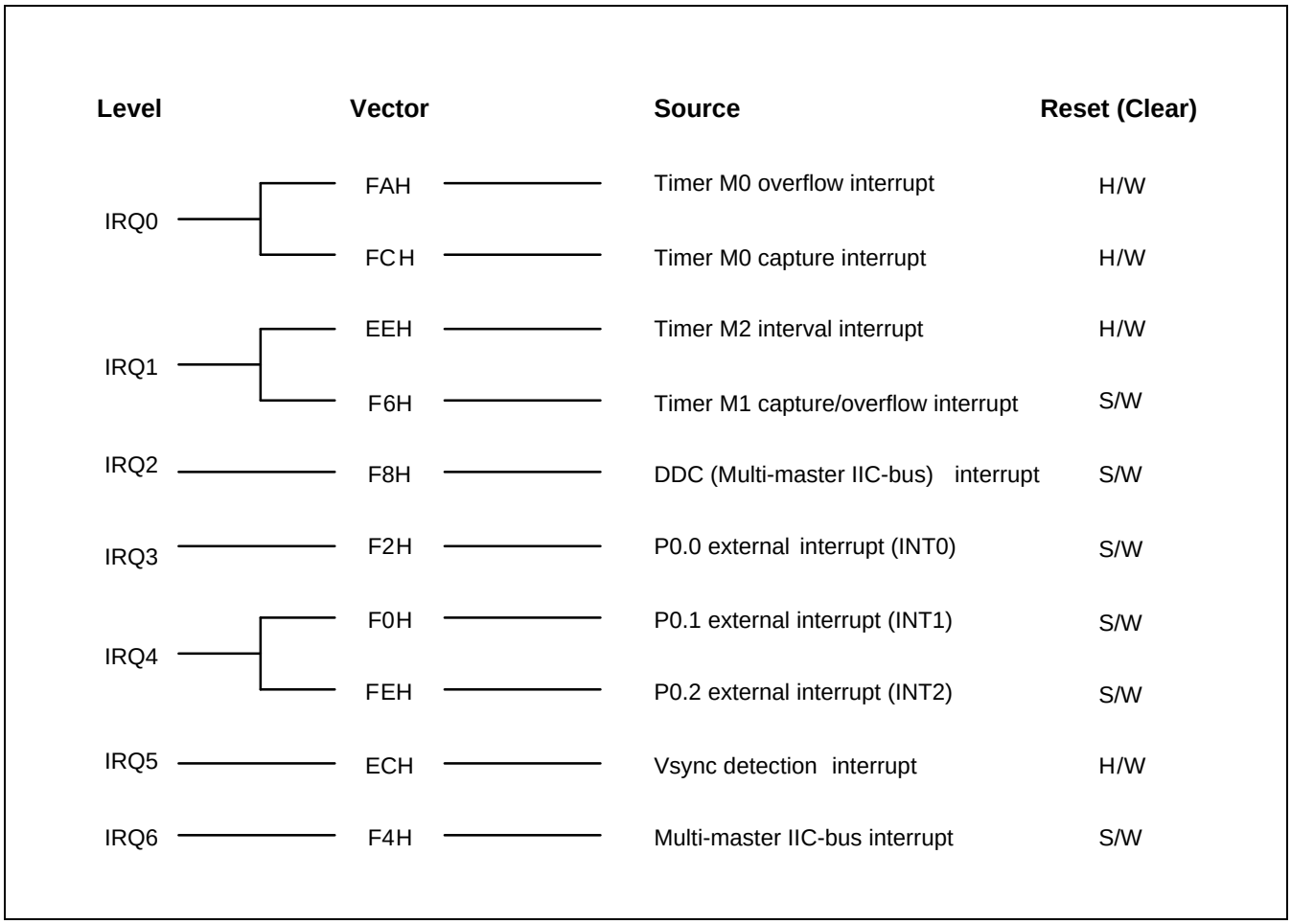


Figure 5-2. KS88C6216/C6224/C6232 Interrupt Structure

INTERRUPT VECTOR ADDRESSES

All interrupt vector addresses for the KS88C6216/C6224/C6232 interrupt structure are stored in the vector address area of the ROM, 00H–FFH (see Figure 5-3). You can allocate unused locations in the vector address area as normal program memory. If you do so, please be careful not to overwrite any of the stored vector addresses. (Table 5-1 lists all vector addresses.)

The program reset address in the ROM is 0100H.

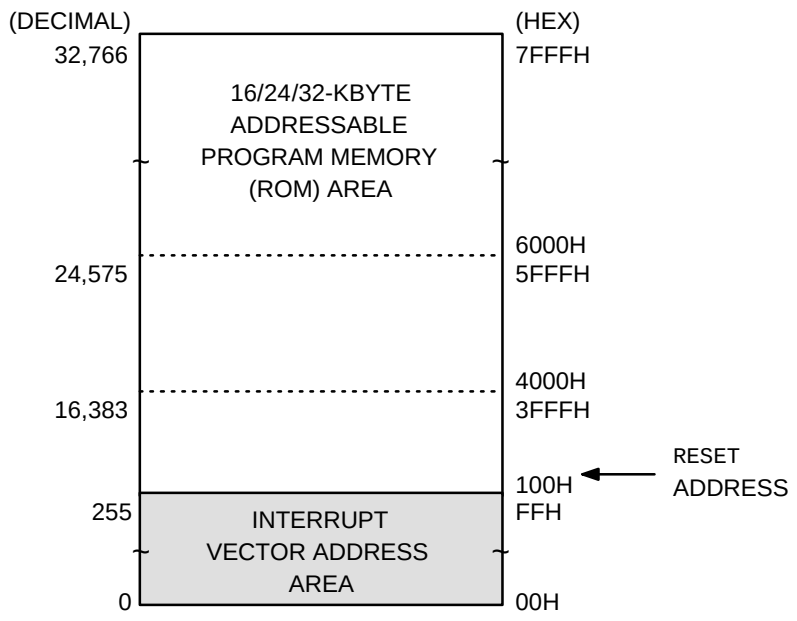


Figure 5-3. ROM Vector Address Area

Table 5-1. KS88C6216/C6224/C6232 Interrupt Vectors

Vector Address		Interrupt Source	Request		Reset/Clear	
Decimal Value	Hex Value		Interrupt Level	Priority in Level	H/W	S/W
250	FAH	Timer M0 overflow interrupt	IRQ0	0	√	
252	FCH	Timer M0 capture interrupt		1	√	
238	EEH	Timer M2 interval interrupt	IRQ1	0	√	
246	F6H	Timer M1 capture/overflow interrupt		1		√
248	F8H	DDC (Multi-master IIC-bus) interrupt	IRQ2	–		√
242	F2H	P0.0 external interrupt (INT0)	IRQ3	–		√
240	F0H	P0.1 external interrupt (INT1)	IRQ4	0		√
254	FEH	P0.2 external interrupt (INT2)		1		√
236	ECH	Vsync detection interrupt	IRQ5	–	√	
244	F4H	Multi-master IIC-bus interrupt	IRQ6	–		√

NOTES:
1. Interrupt priorities are identified in inverse order: "0" is the highest priority, "1" is the next highest, and so on.
2. If two or more interrupts within the same level contend, the interrupt with the lowest vector address usually has priority over one with a higher vector address. The priorities within a given level are fixed in hardware.

ENABLE/DISABLE INTERRUPT INSTRUCTIONS (EI, DI)

Executing the Enable Interrupts (EI) instruction enables the interrupt structure. All interrupts are then serviced as they occur according to the established priorities.

NOTE

The system initialization routine executed after a reset must always contain an EI instruction (assuming one or more interrupts are used in the application).

During the normal operation, you can execute the DI (Disable Interrupt) instruction at any time to globally disable interrupt processing. The EI and DI instructions change the value of bit 0 in the SYM register. Although you can directly manipulate SYM.0 to enable or disable interrupts, it is recommended that you use the EI and DI instructions instead.

SYSTEM-LEVEL INTERRUPT CONTROL REGISTERS

In addition to the control registers for specific interrupt sources, four system-level registers control interrupt processing:

— The interrupt mask register, IMR, enables (un-masks) or disables (masks) interrupt levels.

— The interrupt priority register, IPR, controls the relative priorities of interrupt levels.

— The interrupt request register, IRQ, contains interrupt pending flags for each interrupt level (as opposed to each interrupt source).

— The system mode register, SYM, enables or disables global interrupt processing. (SYM settings also enable fast interrupts and control the activity of external interface, if implemented.)

Table 5-2. Interrupt Control Register Overview

Control Register	ID	R/W	Function Description
Interrupt mask register	IMR	R/W	Bit settings in the IMR register enable or disable interrupt processing for each of the seven interrupt levels, IRQ0–IRQ6.
Interrupt priority register	IPR	R/W	Controls the relative processing priorities of the interrupt levels. The seven levels of KS88C6216/C6224/C6232 are organized into three groups: A, B, and C. Group A is IRQ0 and IRQ1, group B is IRQ2–IRQ4, and group C is IRQ5–IRQ6.
Interrupt request register	IRQ	R	This register contains a request pending bit for each of the seven interrupt levels, IRQ0–IRQ6.
System mode register	SYM	R/W	This register enables and disables dynamic global interrupt processing and fast interrupt processing.

INTERRUPT PROCESSING CONTROL POINTS

Interrupt processing can therefore be controlled in two ways: globally or by specific interrupt level and source. Among the system-level control points in the interrupt structure are:

— Global interrupt enabled and disabled (by EI and DI instructions or by direct manipulation of SYM.0)

— Interrupt level enable/disable settings (IMR register)

— Interrupt level priority settings (IPR register)

— Interrupt source enable/disable settings in the corresponding peripheral control registers

NOTE

When writing an application program that handles interrupt processing, be sure to include the necessary register file address (register pointer) information.

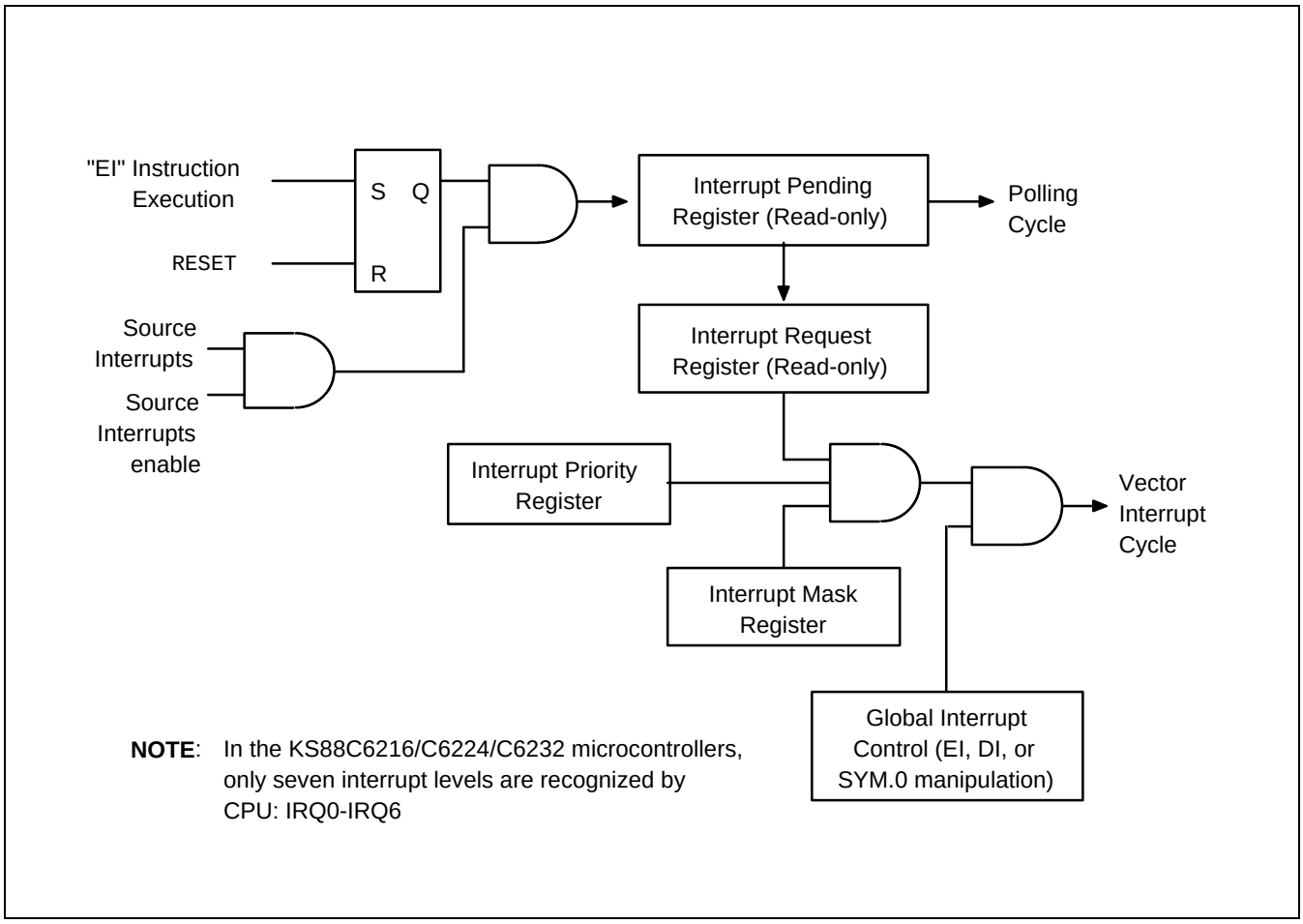


Figure 5-4. Interrupt Function Diagram

PERIPHERAL INTERRUPT CONTROL REGISTERS

For each interrupt source there is a corresponding peripheral control register (or registers) controlling the interrupts generated by the related peripheral. These registers and their locations are listed in Table 5-3.

Table 5-3. Interrupt Source Control Registers

Interrupt Source	Interrupt Level	Control Register(s)	Register Location(s)
Timer M0 (capture) interrupt Timer M0 (overflow) interrupt	IRQ0	TM0CON	Set 1, D2H
Timer M2 interval interrupt Timer M1 capture/overflow interrupt	IRQ1	TM2CON TM1CON	Set 1, bank 0, F6H Set 1, bank 0, F5H
DDC (Multi-master IIC-bus) interrupt	IRQ2	DCCR DCSR0	Set 1, bank 1, EBH Set 1, bank 1, ECH
P0.0 external interrupt	IRQ3	P0CONL, P0INT	Set 1, bank 0, E5H Set 1, bank 0, EBH
P0.1 external interrupt P0.2 external interrupt	IRQ4	P0CONL, P0INT	Set 1, bank 0, E5H Set 1, bank 0, EBH
Vsync detection interrupt	IRQ5	SYNCON2	Set 1, bank 0, EFH
Multi-master IIC-bus interrupt	IRQ6	ICCR ICSR	Set 1, bank 1, F2H Set 1, bank 1, F3H

SYSTEM MODE REGISTER (SYM)

The system mode register, SYM (set 1, DEH), is used to globally enable and disable interrupt processing and to control fast interrupt processing. Figure 5-5 shows the effect of the various control settings.

A reset clears SYM.7, SYM.1, and SYM.0 to "0". Other SYM bit values (for fast interrupt level selection) are undetermined.

The instructions EI and DI enable and disable global interrupt processing, respectively, by modifying the bit 0 value of the SYM register. In order to enable interrupt processing an Enable Interrupt (EI) instruction must be included in the initialization routine, which follows a reset operation. Although you can manipulate SYM.0 directly to enable and disable interrupts during the normal operation, it is recommended to use the EI and DI instructions for this purpose.

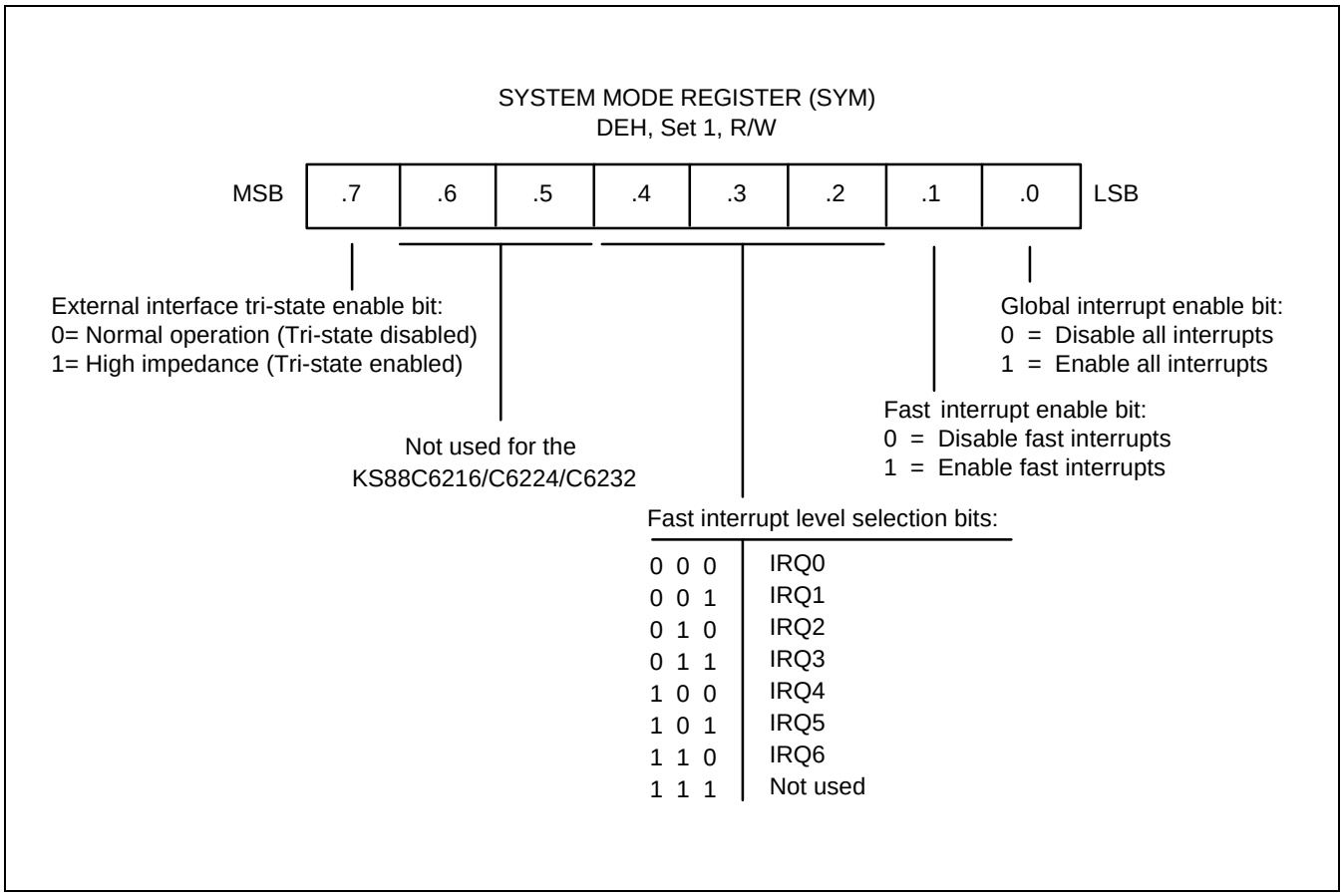


Figure 5-5. System Mode Register (SYM)

INTERRUPT MASK REGISTER (IMR)

The interrupt mask register, IMR (set 1, DDH) is used to enable or disable interrupt processing for individual interrupt levels. After a reset, all IMR bit values are undetermined and must therefore be written to their required settings by the initialization routine.

Each IMR bit corresponds to a specific interrupt level: bit 1 to IRQ1, bit 2 to IRQ2, and so on. When the IMR bit of an interrupt level is cleared to "0", interrupt processing for that level is disabled (masked). When you set a level's IMR bit to "1", interrupt processing for the level is enabled (not masked).

The IMR register is mapped to register location DDH in set 1. Bit values can be read and written by instructions using the Register addressing mode.

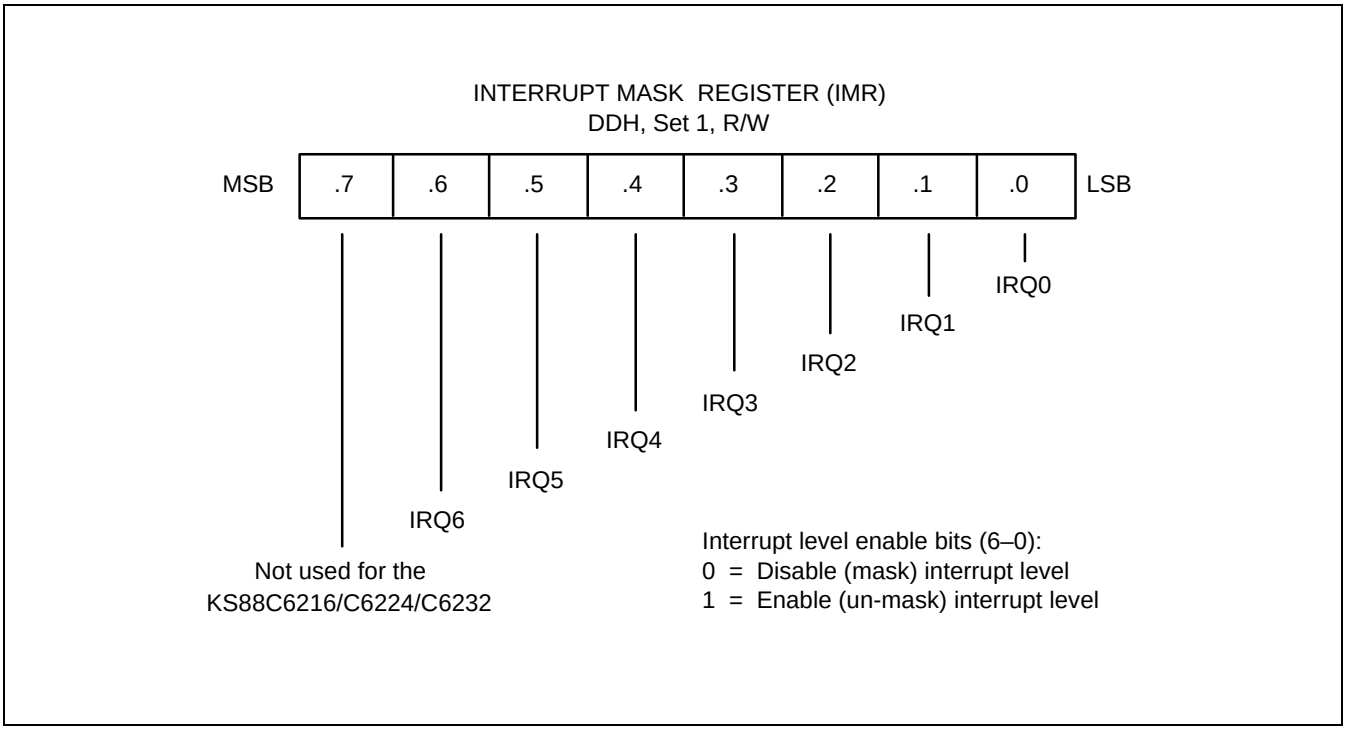


Figure 5-6. Interrupt Mask Register (IMR)

INTERRUPT PRIORITY REGISTER (IPR)

The interrupt priority register, IPR (set 1, bank 0, FFH), is used to set the relative priorities of the interrupt levels in the microcontroller's interrupt structure. After a reset, all IPR bit values are undetermined and must therefore be written to their required settings by the initialization routine.

When more than one interrupt sources are active, the source with the highest priority is serviced first. If two sources belong to the same interrupt level, the source with the lower vector address usually has priority. (This priority is fixed in hardware.)

To support programming of the relative interrupt level priorities, they are organized into groups and subgroups by the interrupt logic. Please note that these groups (and subgroups) are used only by IPR logic for the IPR register priority definitions (see Figure 5-7):

Group A	IRQ0, IRQ1
Group B	IRQ2, IRQ3, IRQ4
Group C	IRQ5, IRQ6

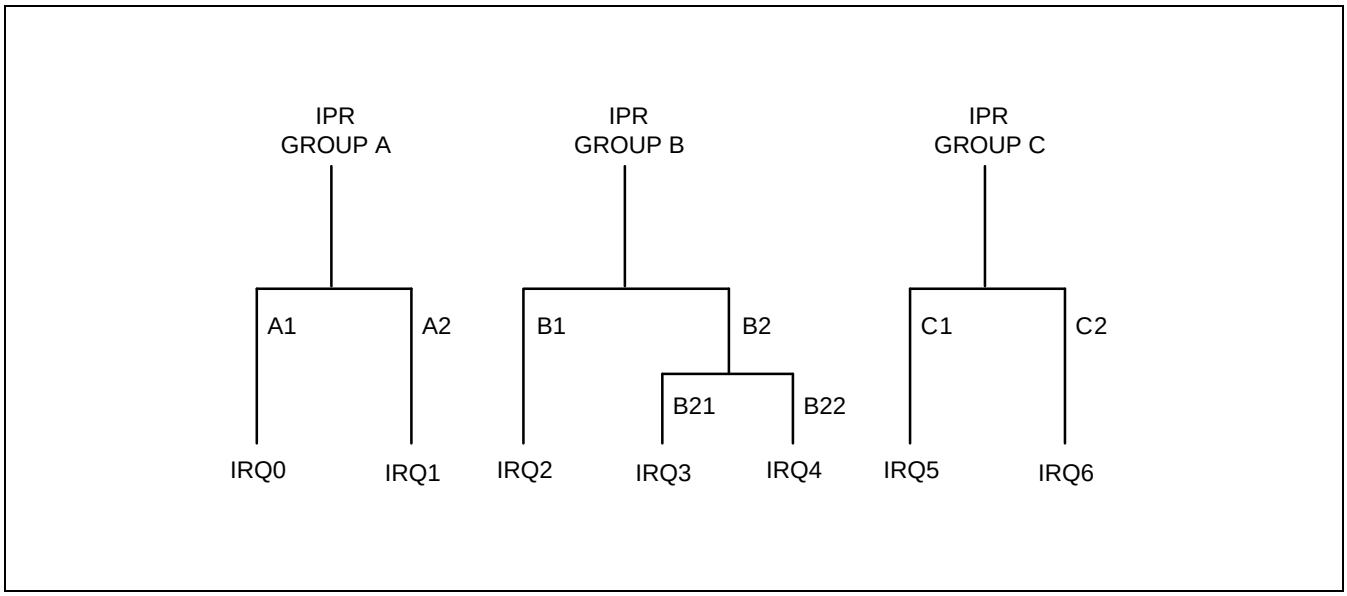


Figure 5-7. Interrupt Request Priority Groups

As you can see in Figure 5-8, IPR.7, IPR.4, and IPR.1 control the relative priority of interrupt groups A, B, and C. For example, the setting "001B" for these bits would select the group relationship B > C > A. The setting "101B" would select the relationship C > B > A.

The functions of the other IPR bit settings are as follows:

— IPR.5 controls the relative priorities of group C interrupts.

— Interrupt group B includes a subgroup that has an additional priority relationship among interrupt levels 2, 3, and 4. IPR.3 defines the subgroup B relationship.

— IPR.2 controls interrupt group B.

— IPR.0 controls the relative priority setting of IRQ0 and IRQ1 interrupts.

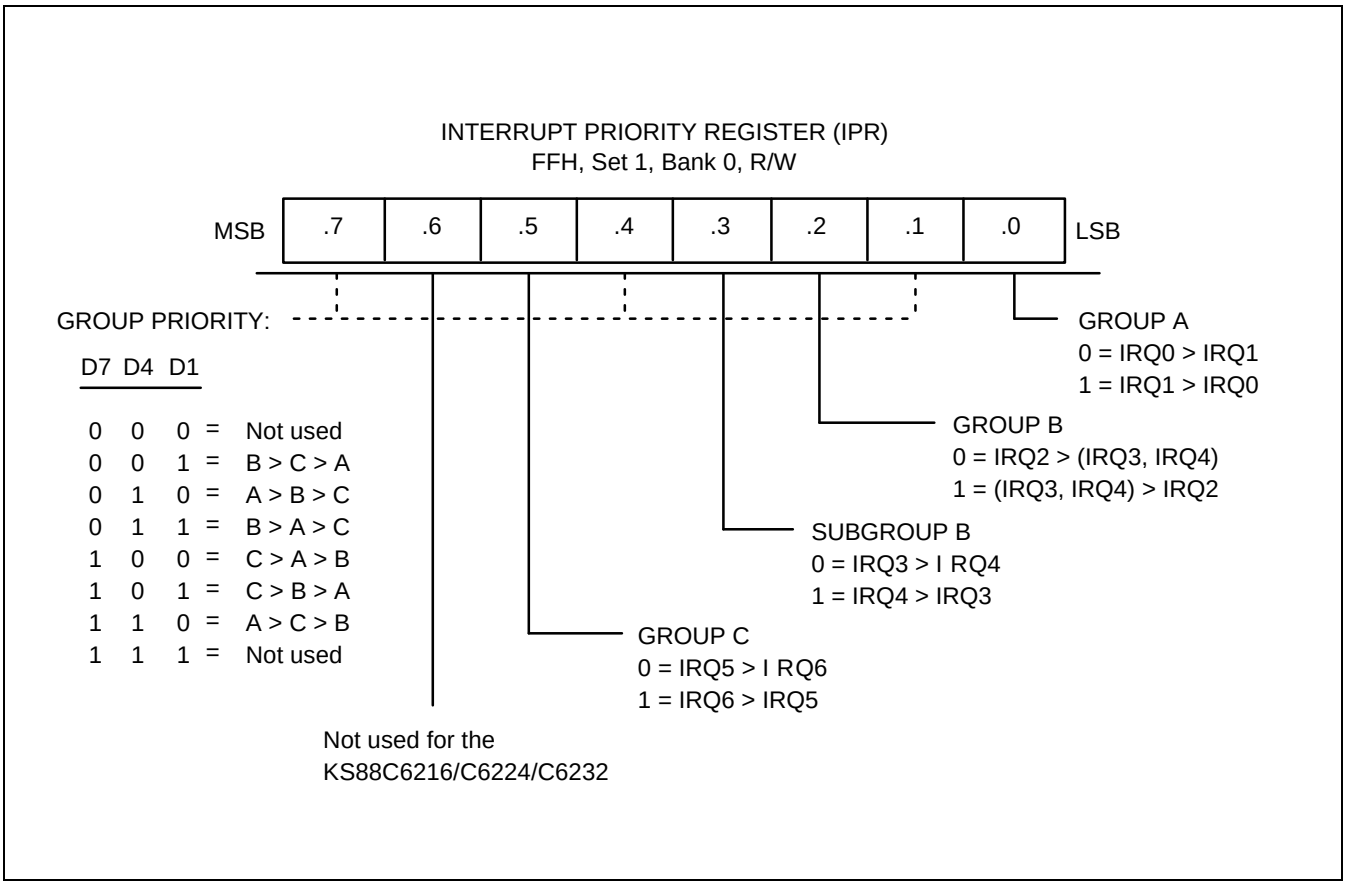


Figure 5-8. Interrupt Priority Register (IPR)

INTERRUPT REQUEST REGISTER (IRQ)

You can poll bit values in the interrupt request register, IRQ (set 1, DCH), to monitor interrupt request status for all levels in the microcontroller's interrupt structure. Each bit corresponds to the interrupt level of the same number: bit 0 to IRQ0, bit 1 to IRQ1, and so on. A "0" indicates that no interrupt request is currently being issued for that level. A "1" indicates that an interrupt request has been generated for that level.

IRQ bit values are addressable in read-only using Register addressing mode. You can read (test) the contents of the IRQ register at any time using bit or byte addressing to determine the current interrupt request status of specific interrupt levels. After a reset, all IRQ status bits are cleared to "0".

You can poll IRQ register values even if a DI instruction has been executed (that is, if global interrupt processing is disabled). If an interrupt occurs while the interrupt structure is disabled, the CPU will not service it. You can, however, still detect the interrupt request by polling the IRQ register. In this way, you can determine which events occurred while the interrupt structure was globally disabled.

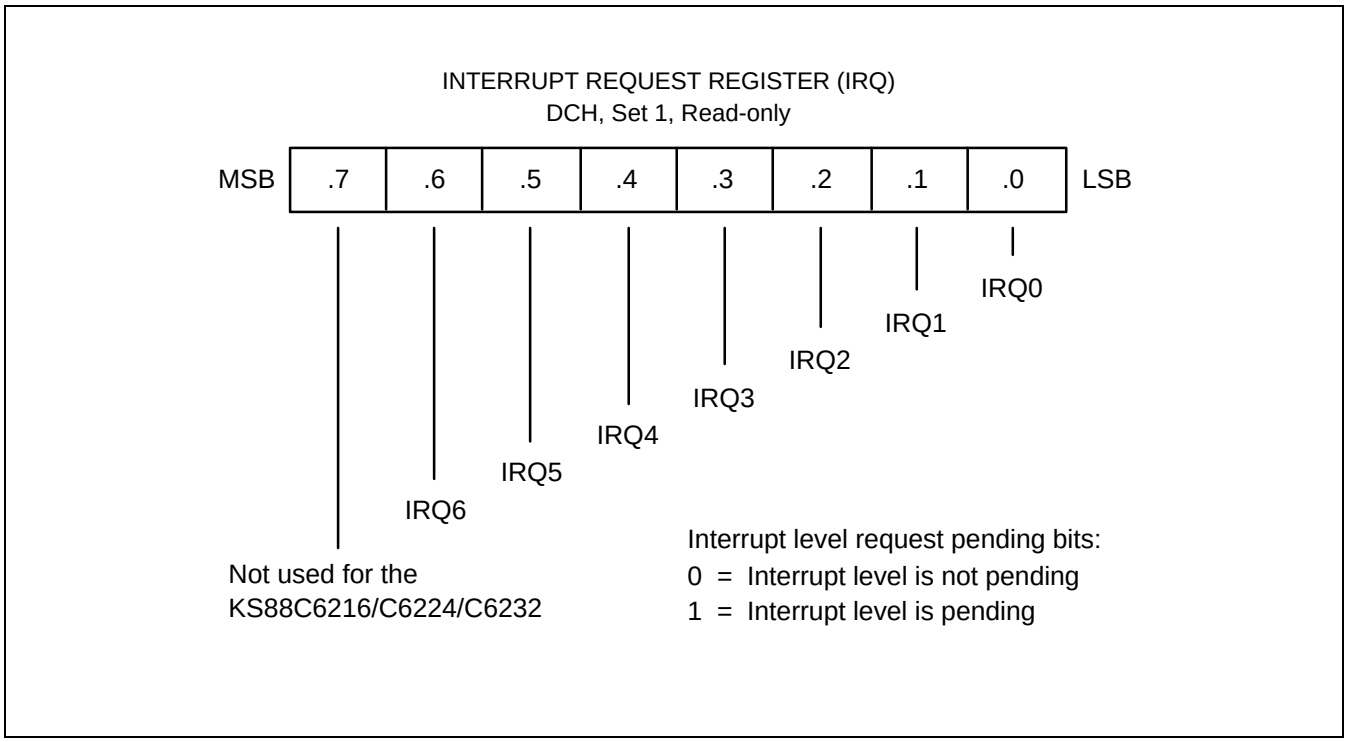


Figure 5-9. Interrupt Request Register (IRQ)

INTERRUPT PENDING FUNCTION TYPES

Overview

There are two types of interrupt pending bits: One type that is automatically cleared by hardware after the interrupt service routine is acknowledged and executed; the other that must be cleared by the application program's interrupt service routine.

Pending Bits Cleared Automatically by Hardware

For interrupt pending bits that are cleared automatically by hardware, interrupt logic sets the corresponding pending bit to "1" when a request occurs. It then issues an IRQ pulse to inform the CPU that an interrupt is waiting to be serviced. The CPU acknowledges the interrupt source, executes the service routine, and clears the pending bit to "0". This type of pending bit is not mapped and cannot, therefore, be read or written by application software.

In the KS88C6216/C6224/C6232 interrupt structure, the timer M0 overflow interrupt (IRQ0, vector FAH), the timer M0 capture interrupt (IRQ0, vector FCH), the timer M2 interval interrupt (IRQ1, vector EEH), and the Vsync detection interrupt (IRQ5, vector ECH) belong to this category of interrupts in which pending conditions are cleared automatically by hardware.

Pending Bits Cleared by the Service Routine

The second type of pending bit is the one that should be cleared by program software. The service routine must clear the appropriate pending bit before a return-from-interrupt subroutine (IRET) occurs. To do this, a "0" must be written to the corresponding pending bit location in the source’s mode or control register.

In the KS88C6216/C6224/C6232 interrupt structure, pending conditions for all interrupt sources, *except* the timer M0 overflow/capture, the timer M2 interval interrupt and the Vsync detection interrupt, must be cleared by the program software's interrupt service routine.

INTERRUPT SOURCE POLLING SEQUENCE

The interrupt request polling and servicing sequence is as follows:

1. A source generates an interrupt request by setting the interrupt request bit to "1".
2. The CPU polling procedure identifies a pending condition for that source.
3. The CPU checks the source's interrupt level.
4. The CPU generates an interrupt acknowledge signal.
5. Interrupt logic determines the interrupt's vector address.
6. The service routine starts and the source's pending bit is cleared to "0" (by hardware or by software).
7. The CPU continues polling for interrupt requests.

INTERRUPT SERVICE ROUTINES

Before an interrupt request is serviced, the following conditions must be met:

— Interrupt processing must be globally enabled (EI, SYM.0 = "1")
— The interrupt level must be enabled (IMR register)
— The interrupt level must have the highest priority if more than one levels are currently requesting service
— The interrupt must be enabled at the interrupt's source (peripheral control register)

When all of the above conditions are met, the interrupt request is acknowledged at the end of the instruction cycle. The CPU then initiates an interrupt machine cycle that completes the following processing sequence:

1. Reset (clear to "0") the interrupt enable bit in the SYM register (SYM.0) to disable all subsequent interrupts.
2. Save the program counter (PC) and status flags to the system stack.
3. Branch to the interrupt vector to fetch the address of the service routine.
4. Pass control to the interrupt service routine.

When the interrupt service routine is completed, the CPU issues an Interrupt Return (IRET). The IRET restores the PC and status flags, setting SYM.0 to "1". It allows the CPU to process the next interrupt request.

GENERATING INTERRUPT VECTOR ADDRESSES

The interrupt vector area in the ROM (00H–FFH) contains the addresses of interrupt service routines that correspond to each level in the interrupt structure. Vectored interrupt processing follows this sequence:

1. Push the program counter's low-byte value to the stack.
2. Push the program counter's high-byte value to the stack.
3. Push the FLAG register values to the stack.
4. Fetch the service routine's high-byte address from the vector location.
5. Fetch the service routine's low-byte address from the vector location.
6. Branch to the service routine specified by the concatenated 16-bit vector address.

NOTE

A 16-bit vector address always begins at an even-numbered ROM address within the range of 00H–FFH.

NESTING OF VECTORED INTERRUPTS

It is possible to nest a higher-priority interrupt request while a lower-priority request is being serviced. To do this, you must follow these steps:

1. Push the current 8-bit interrupt mask register (IMR) value to the stack (PUSH IMR).
2. Load the IMR register with a new mask value that enables only the higher priority interrupt.
3. Execute an EI instruction to enable interrupt processing (a higher priority interrupt will be processed if it occurs).
4. When the lower-priority interrupt service routine ends, restore the IMR to its original value by returning the previous mask value from the stack (POP IMR).
5. Execute an IRET.

Depending on the application, you may be able to simplify the above procedure to some extent.

INSTRUCTION POINTER (IP)

The instruction pointer (IP) is adopted by all the KS88-series microcontrollers to control the optional high-speed interrupt processing feature called *fast interrupts*. The IP consists of register pair DAH and DBH. The names IP of registers are IPH (high byte, IP15–IP8) and IPL (low byte, IP7–IP0).

FAST INTERRUPT PROCESSING

The feature called *fast interrupt processing* allows an interrupt within a given level to be completed in approximately six clock cycles rather than the usual 16 clock cycles. SYM.4–SYM.2 are used to select a specific interrupt level for fast processing and SYM.1 enables or disables fast interrupt processing.

Two other system registers support fast interrupt processing:

— The instruction pointer (IP) contains the starting address of the service routine (and is later used to swap the program counter values), and

— When a fast interrupt occurs, the contents of the FLAGS register is stored in an unmapped, dedicated register called FLAGS' (“FLAGS prime”).

NOTE

1. For the KS88C6216/C6224/C6232 microcontrollers, the service routine for any of the seven interrupt levels (IRQ0–IRQ6) can be selected for fast interrupt processing.
2. When you use a fast interrupt in a multi-source interrupt vector, the fast interrupt may not be processed if you use two sources as interrupt vector in normal mode. But it is possible when you use only one source as interrupt vector.

Procedure for Initiating Fast Interrupts

To initiate fast interrupt processing, follow these steps:

1. Load the start address of the service routine into the instruction pointer (IP).
2. Load the interrupt level number (IRQn) into the fast interrupt selection field (SYM.4–SYM.2)
3. Write a "1" to the fast interrupt enable bit in the SYM register.

Fast Interrupt Service Routine

When an interrupt occurs in the level selected for fast interrupt processing, the following events occur:

1. The contents of the instruction pointer and the PC are swapped.
2. The FLAG register values are written to the FLAGS' (“FLAGS prime”) register.
3. The fast interrupt status bit in the FLAGS register is set.
4. The interrupt is serviced.
5. Assuming that the fast interrupt status bit is set, when the fast interrupt service routine ends, the instruction pointer and PC values are swapped back.
6. The content of FLAGS' (“FLAGS prime”) is copied automatically back to the FLAGS register.
7. The fast interrupt status bit in FLAGS is cleared automatically.

Relationship to Interrupt Pending Bit Types

As described previously, there are two types of interrupt pending bits: One is the type that is automatically cleared by hardware after the interrupt service routine is acknowledged and executed, and the other is the one that must be cleared by the application program's interrupt service routine. You can select fast interrupt processing for interrupts with either type of pending condition clear function — by hardware or by software.

Programming Guidelines

Remember that the only way to enable/disable a fast interrupt is to set/clear the fast interrupt enable bit in the SYM register, SYM.1. Executing an EI or DI instruction globally enables or disables all interrupt processing, including fast interrupts.

NOTE

If you use fast interrupts, remember to load the IP with a new start address when the fast interrupt service routine ends.

☞ PROGRAMMING TIP — Setting Up the Interrupt Control Structure

This example shows you how to enable interrupts for select interrupt sources, disable interrupts for other sources, and set interrupt priorities for the KS88C6216/C6224/C6232 interrupt structure. The following is a sample program:

— Disables the watchdog function.

— Enables the following interrupts: P0.0 external interrupt, timer M0 capture, timer 1 capture/overflow, timer 2 interval interrupt, and DDC interrupt.

— Disables the following interrupts: P0.1 and P0.2 external interrupts, and Vsync detection.

— Sets interrupt priorities as P0.0 > timer M2 > timer M0 > timer M1 > DDC.

```
•
•
•
DI                          ;  Disable interrupts globally
LD      BTCON,#0A0H         ;  Disable watchdog function
LD      P0CONL,#01H         ;  P0.0 ← enable rising edge interrupts
LD      P0INT,#01H          ;  Enable P0.0 external interrupt
                            ;  Disable P0.1 and P0.2 external interrupts
LD      TM0CON,#8FH         ;  Enable timer M0 capture interrupt
                            ;  (capture on rising edges)
                            ;  Enable timer M0 overflow interrupt
LD      TM1CON,#1AH         ;  Enable timer M1 capture/overflow interrupt
LD      TM2CON,#0FFH        ;  Enable timer M2 interval interrupt
LD      TM2DATA,#249        ;  Setting 1ms interval
LD      ICCR,#34H           ;  Enable IIC-bus Tx/Rx interrupt, SCL clock = 100 kHz
LD      SYNCON2,#00H        ;  Disable Vsync detection interrupt
LD      IMR,#0FH            ;  Enable interrupt levels IRQ0, IRQ1, IRQ2, and IRQ3
LD      IPR,#85             ;  IRQ3 > IRQ1 > IRQ0 > IRQ2
                            ;  (P0.0 > timer M0 > timer M2 > timer M1 > DDC)
EI                          ;  Enable interrupts globally
•
•
•
```

☞ PROGRAMMING TIP — Programming Level IRQ0 as a Fast Interrupt

The following example shows you how to program fast interrupt processing for a selected interrupt level — in this case, for the timer M0 capture interrupt:

```
            •
            •
            •
            LD      T0CON,#8FH      ; Enable TM0OVF interrupt
                                    ; Enable TM0CAP interrupt
                                    ; Capture mode (on rising signal edges)
                                    ; Select fOSC/8 as the T0 clock source
            LD      P0CONL,#01H     ; Set P0.4 to capture input mode
            LDW     IPH,#T0_INT     ; IPH ← high byte of interrupt service routine
                                    ; IPL ← low byte of interrupt service routine
            LD      SYM,#02H        ; Enable fast interrupt processing
                                    ; Select IRQ0 for fast interrupt service
            EI                      ; Enable interrupts
            •
            •
            •

FAST_RET:   IRET                    ; IP ← Address of T0_INT (again)
T0_INT:
            •
            •
            •
            (Fast service routine executes)
            •
            •
            •
            LD      TM0CON,#8FH     ; Clear TM0INT interrupt pending bit
            JP      T,FAST_RET
```

7 CLOCK CIRCUIT

OVERVIEW

The clock frequency generated for KS88C6216/C6224/C6232 by an external crystal ranges from 8 MHz to 12 MHz. The maximum CPU clock frequency is 12 MHz. The X_{IN} and X_{OUT} pins connect the external oscillator or clock source to the on-chip clock circuit.

SYSTEM CLOCK CIRCUIT

The system clock circuit has the following components:

— External crystal or ceramic resonator oscillation source (or an external clock source)
— Oscillator stop and wake-up functions
— Programmable frequency divider for the CPU clock (f_{OSC} divided by 1, 2, 8, or 16)
— System clock control register, CLKCON

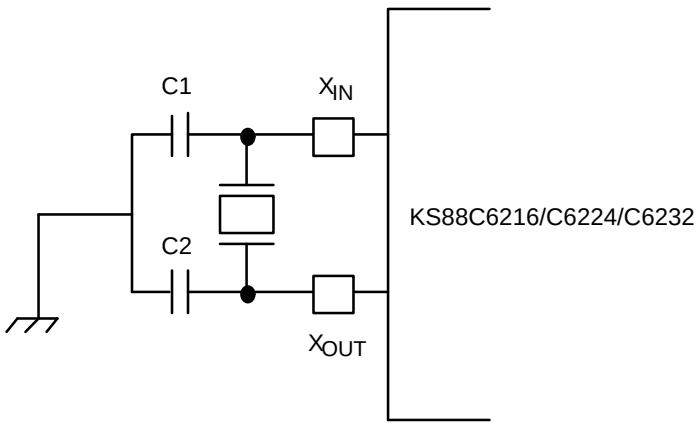


Figure 7-1. Main Oscillator Circuit (External Crystal or Ceramic Resonator)

CLOCK STATUS DURING POWER-DOWN MODES

The two power-down modes, Stop mode and Idle mode, affect the system clock as follows:

— In Stop mode, the main oscillator is halted. Stop mode is released and the oscillator is started by a reset operation or an external interrupt (with RC delay noise filter).

— In Idle mode, the internal clock signal is gated to the CPU, but not to interrupt structure, timers and timer/counters, and the IIC-bus interface functions. Idle mode is released by a reset or by an external or internal interrupt.

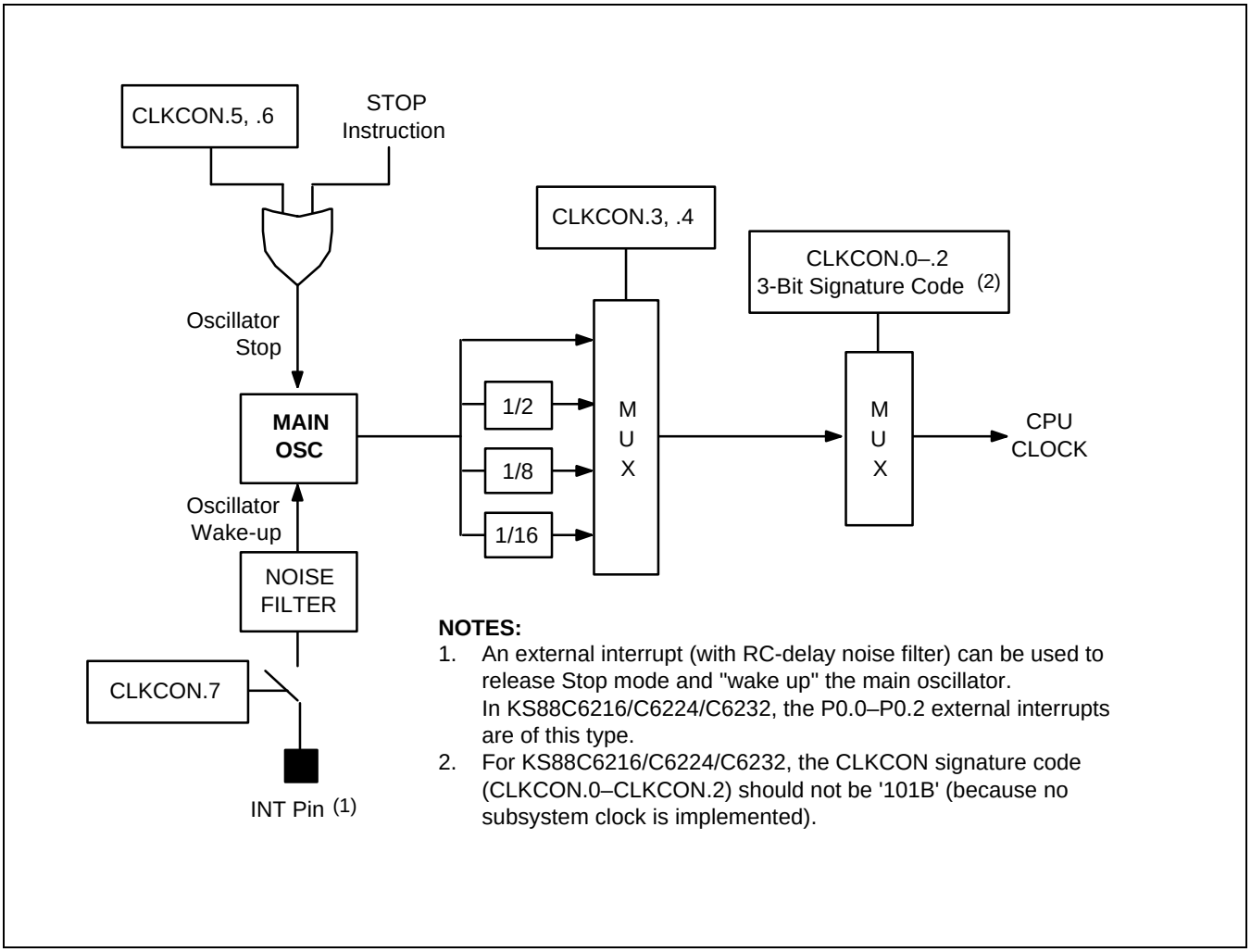


Figure 7-2. System Clock Circuit Diagram

SYSTEM CLOCK CONTROL REGISTER (CLKCON)

The system clock control register, CLKCON, is located in set 1, address D4H. It is read/write addressable and has the following functions:

— Oscillator IRQ wake-up function enable/disable
— Main oscillator stop control
— Oscillator frequency divide-by value
— System clock signal selection

The CLKCON register controls whether or not an external interrupt can be used to trigger a power down mode release. (This is called the "IRQ wake-up" function.) The IRQ wake-up enable bit is CLKCON.7.

After a reset, the external interrupt oscillator wake-up function is enabled, the main oscillator is activated, and the $f_{OSC}/16$ (the slowest clock speed) is selected as the CPU clock. If necessary, you can raise the CPU clock speed to f_{OSC}, $f_{OSC}/2$, or $f_{OSC}/8$.

For the KS88C6216/C6224/C6232 microcontrollers, the CLKCON.2–CLKCON.0 system clock signature code must be any value *other than* "101B". (The "101B" setting is invalid because a subsystem clock is not implemented.) The reset value for the clock signature code is "000B" and should remain so during the normal operation.

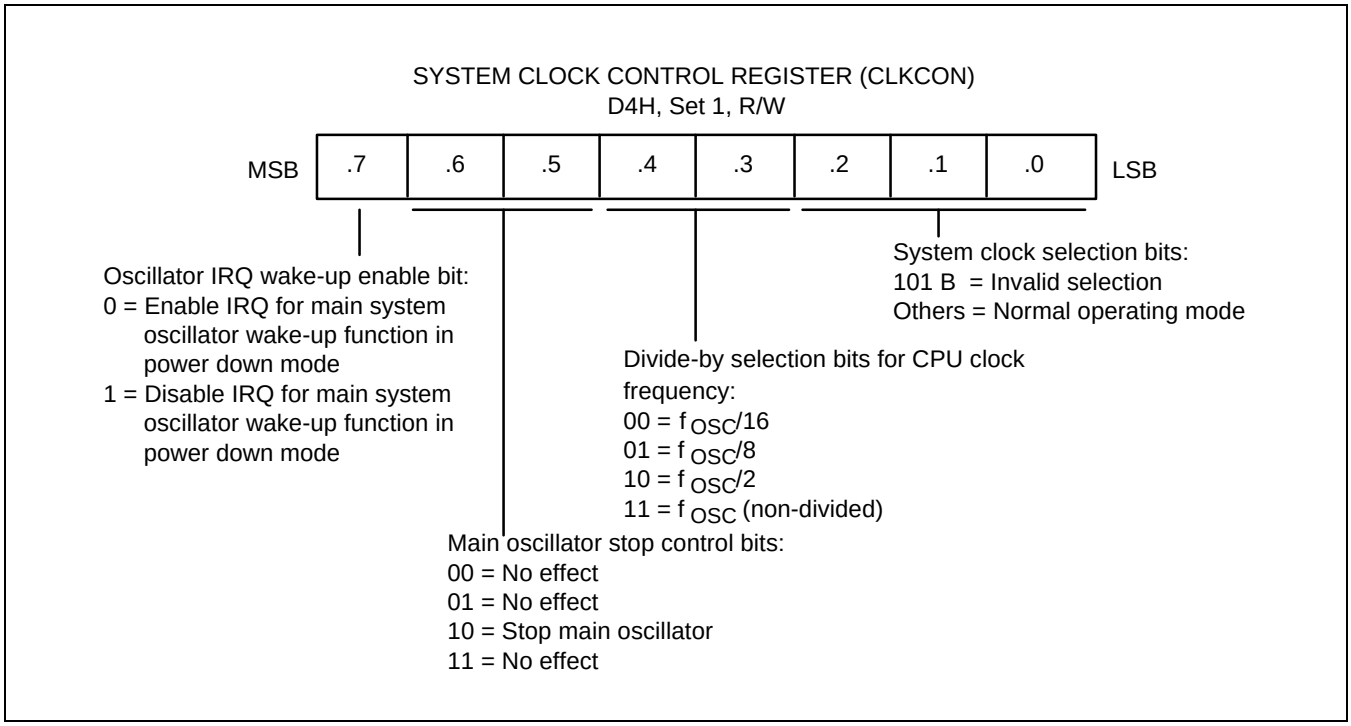


Figure 7-3. System Clock Control Register (CLKCON)

8 RESET and POWER-DOWN

SYSTEM RESET

OVERVIEW

During a power-on reset, the voltage at V_{DD} goes to High level and the RESET pin is forced to Low level. The RESET signal is input through a schmitt trigger circuit where it is then synchronized with the CPU clock. This procedure brings KS88C6216/C6224/C6232 into a known operating status.

To spare time for internal CPU clock oscillation to stabilize, the RESET pin must be held to Low level for a minimum time interval after the power supply comes within tolerance. The minimum required time for oscillation stabilization in a reset operation is 1 millisecond.

Whenever a reset occurs during the normal operation (that is, when both V_{DD} and RESET are at High level), the RESET pin is forced Low and the reset operation starts. All system and peripheral control registers are then reset to their default hardware values (see Tables 8-1, 8-2, and 8-3).

In summary, the following sequence of events occurs during a reset operation:

— All interrupts are disabled.

— The watchdog function (basic timer) is enabled.

— Ports 0–3 are set to input mode.

— Peripheral control and data registers are disabled and reset to their default hardware values.

— The program counter (PC) is loaded with the program reset address in the ROM, 0100H.

— When the programmed oscillation stabilization time interval has elapsed, the instruction stored in the ROM location 0100H (and 0101H) is fetched and executed.

NOTE

To program the duration of the oscillation stabilization interval, you should make the settings appropriate to the basic timer control register, BTCON, *before* entering Stop mode. Also, if you do not want to use the basic timer watchdog function (which causes a system reset if a basic timer counter overflow occurs), you can disable it by writing "1010B" to the upper nibble of BTCON.

HARDWARE RESET VALUES

Tables 8-1, 8-2, and 8-3 list the reset values for CPU and system registers, peripheral control registers, and peripheral data registers after a reset operation. The following notation is used to represent reset values:

— A "1" or a "0" shows the reset bit value as logic one or logic zero, respectively.

— An "x" means that the bit value is undefined after a reset.

— A dash ("–") means that the bit is either not used or not mapped.

Table 8-1. Set 1 Register Values After Reset

Register Name	Mnemonic	Address		Bit Values After Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
Timer M0 counter register	TM0CNT	208	D0H	0	0	0	0	0	0	0	0
Timer M0 data register	TM0DATA	209	D1H	0	0	0	0	0	0	0	0
Timer M0 control register	TM0CON	210	D2H	0	0	0	0	0	0	0	0
Basic timer control register	BTCON	211	D3H	0	0	0	0	0	0	0	0
Clock control register	CLKCON	212	D4H	0	0	0	0	0	0	0	0
System flags register	FLAGS	213	D5H	x	x	x	x	x	x	0	0
Register pointer 0	RP0	214	D6H	1	1	0	0	0	–	–	–
Register pointer 1	RP1	215	D7H	1	1	0	0	1	–	–	–
Stack pointer (high byte)	SPH	216	D8H	x	x	x	x	x	x	x	x
Stack pointer (low byte)	SPL	217	D9H	x	x	x	x	x	x	x	x
Instruction pointer (high byte)	IPH	218	DAH	x	x	x	x	x	x	x	x
Instruction pointer (low byte)	IPL	219	DBH	x	x	x	x	x	x	x	x
Interrupt request register	IRQ	220	DCH	–	0	0	0	0	0	0	0
Interrupt mask register	IMR	221	DDH	–	x	x	x	x	x	x	x
System mode register	SYM	222	DEH	0	–	–	x	x	x	0	0
Page pointer register	PP	223	DFH	0	0	0	0	0	0	0	0

NOTES:
1. As the SYM register is not used for KS88C6216/C6224/C6232, SYM.5 should always be "0". If you accidentally write a “1” to this bit during the normal operation, a system malfunction may occur.
2. Except for TM0CNT, TMODATA, and IRQ, all registers in set 1 are read/write addressable.
3. You cannot use a read-only register as a destination field for the instructions OR, AND, LD, and LDB. The read-only registers in the KS88C6216/C6224/C6232 register file are: TM0CNT, TM0DATA, IRQ, SYNCRD, TM1CNTH, TM1CNTL, TM1DATAH, TM1DATAL, ADDATA, BTCNT, PWMCNT, and RBDR.
4. Interrupt pending flags that must be cleared by software are noted by shaded table cells.

Table 8-2. Set 1, Bank 0 Register Values After Reset

Register Name	Mnemonic	Address		Bit Values After Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
Port 0 data register	P0	224	E0H	0	0	0	0	0	0	0	0
Port 1 data register	P1	225	E1H	–	–	–	–	–	0	0	0
Port 2 data register	P2	226	E2H	0	0	0	0	0	0	0	0
Port 3 data register	P3	227	E3H	0	0	0	0	0	0	0	0
Port 0 control register (high byte)	P0CONH	228	E4H	0	0	0	0	0	0	0	0
Port 0 control register (low byte)	P0CONL	229	E5H	0	0	0	0	0	0	0	0
Port 1 control register	P1CON	230	E6H	–	–	0	0	0	0	0	0
Port 2 control register (high byte)	P2CONH	231	E7H	0	0	0	0	0	0	0	0
Port 2 control register (low byte)	P2CONL	232	E8H	0	0	0	0	0	0	0	0
Port 3 control register (high byte)	P3CONH	233	E9H	0	0	0	0	0	0	0	0
Port 3 control register (low byte)	P3CONL	234	EAH	0	0	0	0	0	0	0	0
Port 0 external interrupt control register	P0INT	235	EBH	–	0	0	0	–	0	0	0
Location ECH is not mapped											
Sync control register 0	SYNCON0	237	EDH	0	0	0	0	0	0	0	0
Sync control register 1	SYNCON1	238	EEH	0	0	0	0	0	0	0	0
Sync control register 2	SYNCON2	239	EFH	0	0	0	0	0	0	0	0
Sync port read data register	SYNCRD	240	F0H	–	–	–	–	0	0	0	0
Timer M1 counter register high	TM1CNTH	241	F1H	–	–	–	–	0	0	0	0
Timer M1 counter register low	TM1CNTL	242	F2H	0	0	0	0	0	0	0	0
Timer M1 data register high	TM1DATAH	243	F3H	–	–	–	–	0	0	0	0
Timer M1 data register low	TM1DATAL	244	F4H	0	0	0	0	0	0	0	0
Timer M1 control register	TM1CON	245	F5H	0	0	0	0	0	0	0	0
Timer M2 control register	TM2CON	246	F6H	1	1	1	1	1	0	0	0
A/D converter control register	ADCON	247	F7H	–	0	0	0	1	0	0	0
A/D converter data register	ADDATA	248	F8H	x	x	x	x	x	x	x	x
Locations F9H–FCH are not mapped											
Basic timer counter register	BTCNT	253	FDH	0	0	0	0	0	0	0	0
External memory timing register	EMT	254	FEH	0	1	1	1	1	1	0	–
Interrupt priority register	IPR	255	FFH	x	x	x	x	x	x	x	x

NOTES:
1. Except for SYNCRD, TM1CNTH, TM1CNTL, TM1DATAH, TM1DATAL, ADDATA, and BTCNT, all registers in set 1, bank 0 are read/write addressable.
2. You cannot use a read-only register as a destination field for the instructions OR, AND, LD, and LDB. The read-only registers in the KS88C6216/C6224/C6232 register file are: TM0CNT, TM0DATA, IRQ, SYNCRD, TM1CNTH, TM1CNTL, TM1DATAH, TM1DATAL, ADDATA, BTCNT, PWMCNT, and RBDR.
3. Interrupt pending flags that must be cleared by software are noted by shaded table cells.

Table 8-3. Set 1, Bank 1 Register Values After Reset

Register Name	Mnemonic	Address		Bit Values After Reset							
		Dec	Hex	7	6	5	4	3	2	1	0
PWM 0 data register	PWM0	224	E0H	0	0	0	0	0	0	0	0
PWM 1 data register	PWM1	225	E1H	0	0	0	0	0	0	0	0
PWM 2 data register	PWM2	226	E2H	0	0	0	0	0	0	0	0
PWM 3 data register	PWM3	227	E3H	0	0	0	0	0	0	0	0
PWM 4 data register	PWM4	228	E4H	0	0	0	0	0	0	0	0
PWM 5 data register	PWM5	229	E5H	0	0	0	0	0	0	0	0
PWM 6 data register	PWM6	230	E6H	0	0	0	0	0	0	0	0
PWM control register	PWMCON	231	E7H	0	0	0	–	–	–	–	–
PWM counter register	PWMCNT	232	E8H	0	0	0	0	0	0	0	0
DDC control register	DCON	233	E9H	–	–	–	–	1	0	0	0
DDC address register 0	DAR0	234	EAH	1	0	1	0	0	0	0	0
DDC clock control register	DCCR	235	EBH	0	0	0	0	1	1	1	1
DDC control/status register 0	DCSR0	236	ECH	0	0	0	0	0	0	0	0
DDC control/status register 1	DCSR1	237	EDH	–	–	–	–	–	0	1	0
DDC address register 1	DAR1	238	EEH	x	x	x	x	x	x	x	–
Transmit prebuffer data register	TBDR	239	EFH	x	x	x	x	x	x	x	x
Receive prebuffer data register	RBDR	240	F0H	x	x	x	x	x	x	x	x
DDC data shift register	DDSR	241	F1H	x	x	x	x	x	x	x	x
IIC-bus clock control register	ICCR	242	F2H	0	0	0	0	1	1	1	1
IIC-bus control/status register	ICSR	243	F3H	0	0	0	0	0	0	0	0
IIC-bus address register	IAR	244	F4H	x	x	x	x	x	x	x	–
IIC-bus data shift register	IDSR	245	F5H	x	x	x	x	x	x	x	x
Locations F6H–FFH are not mapped.											

NOTES:
1. Except for PWMCNT and RBDR, all registers in set 1, bank 1 are read/write addressable.
2. You cannot use a read-only register as a destination field for the instructions OR, AND, LD, and LDB. The read-only registers in the KS88C6216/C6224/C6232 register file are: TM0CNT, TM0DATA, IRQ, SYNCRD, TM1CNTH, TM1CNTL, TM1DATAH, TM1DATAL, ADDATA, BTCNT, PWMCNT, and RBDR.
3. Interrupt pending flags that must be cleared by software are noted by shaded table cells.

POWER-DOWN MODES

STOP MODE

Stop mode is invoked by the instruction STOP (opcode 7FH). In Stop mode, the operation of the CPU and all peripherals is halted. That is, the on-chip main oscillator stops and the supply current is reduced to less than 5 µA. All system functions stop when the clock "freezes," but data stored in the internal register file is retained. Stop mode can be released in one of two ways: by a reset or by an external interrupt (with RC delay).

NOTE

Do not use stop mode if you are using an external clock source as X_{IN} input must be restricted internally to V_{SS} to reduce current leakage.

Using RESET to Release Stop Mode

Stop mode is released when the RESET signal goes active (High level): all system and peripheral control registers are reset to their default hardware values and the contents of all data registers are retained. A reset operation automatically selects a slow clock (1/16) because CLKCON.3 and CLKCON.4 are cleared to "00B". After the programmed oscillation stabilization interval has elapsed, the CPU starts the system initialization routine by fetching the program instruction stored in the ROM location 0100H (and 0101H).

Using an External Interrupt to Release Stop Mode

Only external interrupts with an RC-delay noise filter circuit can be used to release Stop mode. Which interrupt you can use to release Stop mode in a given situation depends on the microcontroller's current internal operating mode. The external interrupts in the KS88C6216/C6224/C6232 interrupt structure that can be used to release Stop mode are:

— External interrupts P0.0 (INT0), P0.1 (INT1), and P0.2 (INT2)

— Timer M0 capture interrupt in capture mode (with rising or falling edge trigger at the TM0CAP pin and Vsync-O from sync-processor.)

Please note the following conditions for Stop mode release:

— If you release Stop mode using an external interrupt, the current values in system and peripheral control registers are unchanged.

— If you use an external interrupt for Stop mode release, you can also program the duration of the oscillation stabilization interval. To do this, you must make the control and clock settings appropriate *before* entering Stop mode.

— If you use an interrupt to release Stop mode, the CLKCON.4 and CLKCON.3 bit-pair setting remains unchanged and the currently selected clock value is used.

— The external interrupt is serviced when the Stop mode release occurs. Following the IRET from the service routine, the instruction right next to the one that initiated Stop mode is executed.

IDLE MODE

Idle mode is invoked by the instruction IDLE (opcode 6FH). In Idle mode, CPU operations are halted while some peripherals remain active. In idle mode, the internal clock signal is gated away from the CPU, but all peripherals timers remain active. Port pins retain the mode (input or output) they had at the time Idle mode was entered.

There are two ways to release idle mode:

1. Execute a reset. All system and peripheral control registers are reset to their default values and the contents of all data registers are retained. The reset automatically selects a *slow clock (1/16)* because CLKCON.4 and CLKCON.3 are cleared to "00B". If interrupts are masked, a reset is the only way to release Idle mode.
2. Activate any enabled interrupt, causing Idle mode to be released. When you use an interrupt to release Idle mode, the CLKCON.4 and CLKCON.3 register values remain unchanged, and the *currently selected clock* value is used. The interrupt is then serviced. When the return-from-interrupt (IRET) occurs, the instruction right next to the one that initiated Idle mode is executed.

NOTE

Only external interrupts can be used to release Stop mode. To release idle mode, you can use either an internally-generated or externally-generated interrupt.

☞ PROGRAMMING TIP — Sample KS88C6216/C6224/C6232 Initialization Routine

The following sample program shows you how to make initial settings for the KS88C6216/C6224/C6232 address space, interrupt vectors, and peripheral functions. Program comments guide you through the steps:

```
;           << Base Number Setting >>

            DECIMAL

;           << Definition >>

TM0_REG     EQU     40H
            ORG     0000H

;           << Interrupt Vector Addresses >>

            ORG     00ECH
            VECTOR  VSYNC_INT           ;  IRQ5
            VECTOR  TM2 _INT            ;  IRQ1
;
            VECTOR  P01_INT             ;  IRQ4
            VECTOR  P00_INT             ;  IRQ3
            ORG     00F6H
            VECTOR  TM1_CAP_OVF_INT     ;  IRQ1
            VECTOR  DDC_INT             ;  IRQ2
            VECTOR  TM0_OVF_INT         ;  IRQ0
            VECTOR  TM0_CAP_INT         ;  IRQ0
            VECTOR  P02_INT             ;  IRQ4

;           << Initialize System and Peripherals >>

            ORG     0100H               ;  Reset address
            LD      BTCON,#0A0H         ;  Disable watchdog timer
            LD      CLKCON,#10H         ;  Select divided-by-two oscillator frequency as CPU clock
                                        ;  Enable IRQ for main system oscillator wake-up

;           < System Register Settings >

            CLR     SYM                 ;  Disable fast interrupts; global interrupt disable
            CLR     EMT                 ;  No access wait time; select internal stack area
            LD      SPH,#00H            ;  Set stack pointer (stack starts from #0FFH)

;           < Interrupt Settings >

            LD      IPR,#8FH            ;  Set interrupt priorities as follows:
                                        ;  IRQ3 > IRQ2 > IRQ1 > IRQ0
            LD      IMR,#0FH            ;  Enable IRQ levels 0, 1, 2, and 3

;           < Timer 0 Settings >

            LD      TM0CON,#8FH         ;  Enable timer M0 overflow and capture interrupts
```

(Continued on next page)

☞ PROGRAMMING TIP — Sample KS88C6216/C6224/C6232 Initialization Routine (Continued)

```
INI_PERI_SET:
            SB0                         ;  Select bank 0
            LD      P0CONH,#0FFH        ;  Set port 0 high byte to push-pull output mode
            LD      P0CONL,#0FDH        ;  Set port 0 low byte to push-pull output mode
            LD      P0INT,#11H          ;  Enable P0.0 external interrupt
                                        ;  Disable P0.1 and P0.2 external interrupts
            LD      P1CON,#00H          ;  Set P1.0–P1.2 to input mode
            LD      P2CONH,#0FFH        ;  Set port 2 high byte to n-channel, open-drain PWM
                                        ;  output mode
            LD      P2CONL,#0FFH        ;  Set port 2 low byte to push-pull PWM output mode
            LD      P3CONH,#55H         ;  Set port 3 high byte to push-pull output mode
            LD      P3CONL,#55H         ;  Set port 3 low byte to push-pull output mode

;           < Timer 1 Settings >

            LD      TM1CON,#2AH         ;  Enable timer M1 capture/overflow interrupt, Timer M1
                                        ;  clock source is HsyncI from sync processor

;           < Sync Processor Settings >

            LD      SYNCON0,#40H        ;  5 bit counter capture mode
            LD      SYNCON1,#50H        ;  Set negative polarity (500 ns at 8 MHz) for clampO
            LD      SYNCON2,#8CH        ;  Pseudo sync output

;           < PWM Settings >

            SB1
            LD      PWMCON,#20H         ;  Start PWM counter, PWM counter clock is fOSC

;           < DDC Tx/Rx Interface Settings >

            LD      DCON,#0AH           ;  Select DDC1 Tx mode
            LD      DCCR,#24H           ;  Enable DDC interrupt, DDC clock is 100 kHz

;           << Initialize Data Registers >>

            SB0                         ;  Select bank 0
            SRP     #0C0H               ;  Set register pointer

;           < Clear all data registers from 00H to FFH >

            LD      R0,#0FFH

RAMCLR:     CLR     @R0
            DJNZ    R0,RAMCLR
```

(Continued on next page)

☞ PROGRAMMING TIP — Sample KS88C6216/C6224/C6232 Initialization Routine (Continued)

```
;           < Initialize Other Registers >

            •
            •
            EI                          ;  You must execute an EI instruction in this position
                                        ;  in the initialization routine to enable servicing of
                                        ;  external interrupts

;           << Main Loop >>

MAIN:       NOP                         ;  Start main loop
            •
            •
            CALL      KEY_SCAN          ;  Sub-program module
            •
            •
            CALL      LED_DISPLAY       ;  Sub-program module
            •
            •
            CALL      JOB               ;  Sub-program module
            •
            •
            JR        t,MAIN            ;  For main loop

;           < Subroutines >

KEY_SCAN:
            NOP
            •
            •
            •
            RET

LED_DISPLAY:
            NOP
            •
            •
            •
            RET

JOB:
            NOP
            •
            •
            •
            RET
```

(Continued on next page)

☞ PROGRAMMING TIP — Sample KS88C6216/C6224/C6232 Initialization Routine (Concluded)

```
;           << Interrupt Service Routines >>

P00_INT:
            PUSH    RP0                 ;  Save old RP0 value
            SRP0    #60H                ;  Set RP0 for P0.0 interrupt service routine
            •
            •
            •
            POP     RP0                 ;  Restore the RP0 value
            IRET                        ;  Return from the interrupt

DDC_INT:
            PUSH    RP0                 ;  Save old RP0 value
            SRP0    #50H                ;  Set RP0 for IIC-bus interrupt service routine
            •
            •
            •
            SB1
            AND     DDCR, #11101111B    ;  Clear DDC interrupt pending bit
            SB0
            POP     RP0                 ;  Restore the RP0 value
            IRET                        ;  Return from the interrupt

TM0_CAP_INT:
            PUSH    RP0                 ;  Save old RP0 value
            SRP     #TM0_REG            ;  TM0_REG value should be defined
            •
            •
            •
            POP     RP0                 ;  Restore the RP0 value
            IRET                        ;  Return from the interrupt

            END
```

9 I/O PORTS

OVERVIEW

The KS88C6216/C6224/C6232 microcontrollers have four I/O ports with a total of 27 pins. Each port can be flexibly configured to meet application design requirements. The CPU accesses ports by directly writing or reading port registers. No special I/O instructions are required. Table 9-1 gives you an overview of port functions:

Table 9-1. KS88C6216/C6224/C6232 Port Configuration Overview

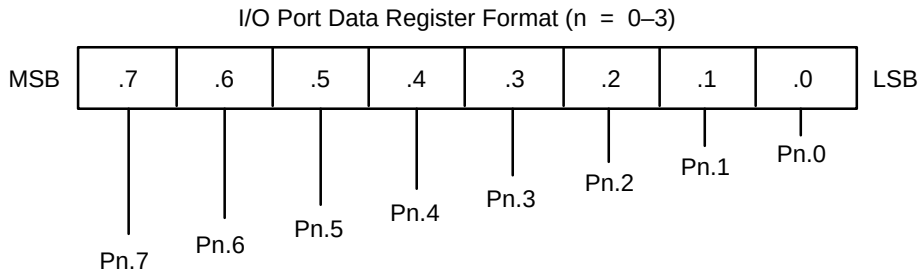
Port	Configuration Options	Programmability
0	8-bit general I/O port. Alternatively used for external interrupt inputs and for timer M0 and M1 input/output functions.	Bit programmable
1	3-bit I/O port for normal I/O or n-channel open drain output. Alternatively used for IIC-bus clock and data I/O.	Bit programmable
2	8-bit I/O port for normal I/O, PWM push-pull outputs, PWM n-channel open-drain outputs with 5-volt load capability, or Csync signal input.	Bit programmable
3	8-bit general I/O port. Alternatively used as n-channel open-drain, push-pull outputs with 5-volt load capability or for normal input with pull-up resistor. Multiplexed for alternative use as A/D converter inputs, AD0–AD3.	Bit programmable

PORT DATA REGISTERS

Data registers for ports 0–3 have the format shown in Figure 9-1. Table 9-2 gives you an overview of the port data register locations:

Table 9-2. Port Data Register Summary

Register Name	Mnemonic	Decimal	Hex	Location	R/W
Port 0 data register	P0	224	E0H	Set 1, bank 0	R/W
Port 1 data register	P1	225	E1H	Set 1, bank 0	R/W
Port 2 data register	P2	226	E2H	Set 1, bank 0	R/W
Port 3 data register	P3	227	E3H	Set 1, bank 0	R/W



NOTE: Port 1 is a 3-bit port. Only bits P1.2–P1.0 of the port 1 data register are mapped. All the other KS88C6216/C6224/C6232 I/O ports are 8-bit.

Figure 9-1. Port Data Register Format

PORT 0

Port 0 is an 8-bit I/O port with individually configurable pins. You can directly access port 0 pins by writing or reading the port 0 data register, P0 (set 1, bank 0, E0H). You can use port 0 for general I/O, or for the following alternative functions:

— Low-byte pins (P0.3–P0.0) can be configured as push-pull outputs, while P0.2–P0.0 as a multiplexed input pins for external interrupts INT2–INT0 with rising or falling edge detection.

— High-byte pins (P0.7–P0.4) can be configured as multiplexed inputs and push-pull outputs. P0.4 can serve as the timer M0 capture input pin (TM0CAP), and P0.5 as the timer M1 clock input pin (TM1CLK).

Two 8-bit control registers are used to configure port 0 pins: P0CONH (set 1, bank 0, E4H) for P0.7–P0.4 and P0CONL (set 1, bank 0, E5H) for P0.3–P0.0. Each byte contains four bit-pairs and each bit-pair configures one pin. The low-byte port 0 control register, P0CONL, is also used to enable and disable the external interrupts, INT2–INT0, at pins P0.2–P0.0, respectively.

Port 0 High-Byte Control Register (P0CONH)

The four bit-pairs in the port 0 high-byte control register, P0CONH, have the following functions:

— To configure individual port 0 pins to multiplexed input mode or push-pull output mode.

— To configure alternative input or output functions for P0.7–P0.4.

Bit-pair 1/0 configures the capture signal input pin for timer M0 at P0.4. Bit-pair 3/2 controls P0.5. When it is set to "01B", the timer M1 clock input is enabled.

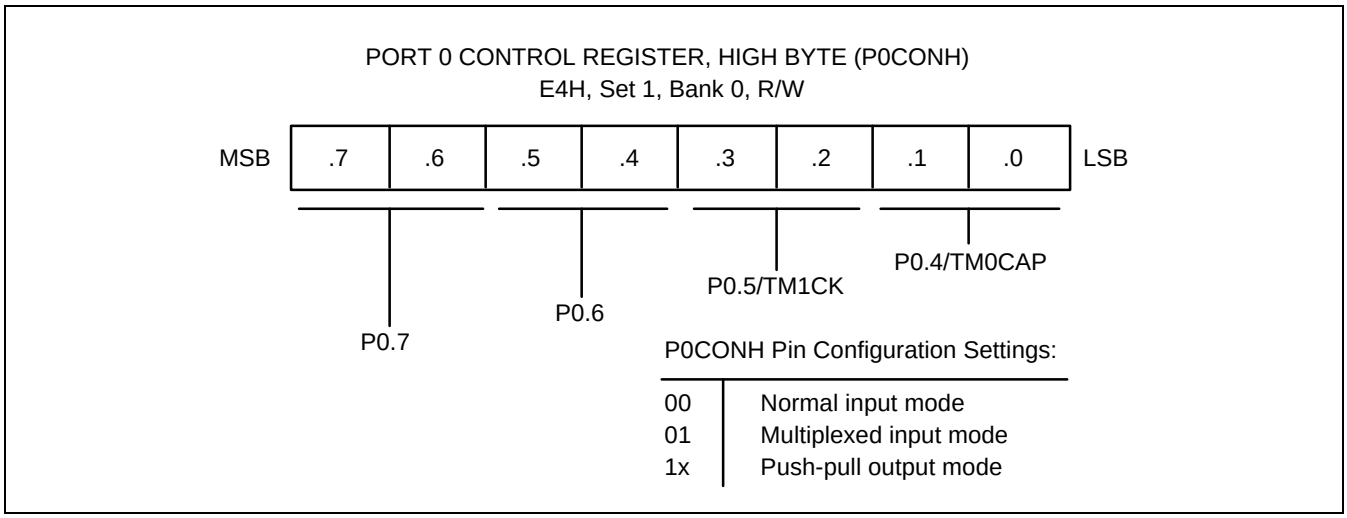


Figure 9-2. Port 0 High-Byte Control Register (P0CONH)

Port 0 Low-Byte Control Register (P0CONL)

The low-byte port 0 pins, P0.3–P0.0 can be configured individually as inputs or as push-pull outputs. You can alternatively configure the pins P0.2–P0.0 as external interrupt inputs with rising or falling edge detection.

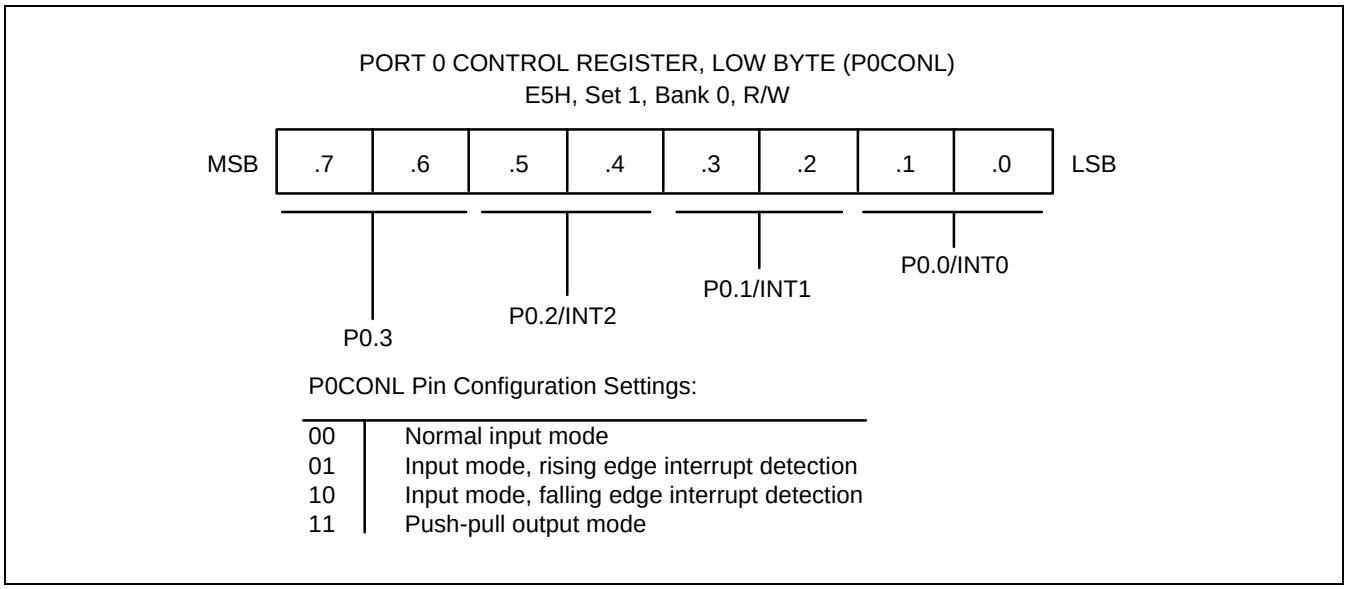


Figure 9-3. Port 0 Low-Byte Control Register (P0CONL)

Port 0 External Interrupt Control Register (P0INT)

The port 0 external interrupt control register, P0INT, is used to enable and disable the external interrupts INT2–INT0 at P0.2–P0.0, respectively, and also to detect and clear external interrupt pending conditions at these pins.

To selectively enable the external interrupts INT0, INT1, and INT2, you set P0INT.0, P0INT.1, and P0INT.2 to “1”, respectively. The application program can poll the corresponding interrupt pending bits — P0INT.4 for INT0, P0INT.5 for INT1, and P0INT.6 for INT2 — to detect external interrupt pending conditions.

After an external interrupt has been serviced, the service routine must clear the pending condition by writing a “0” to the appropriate pending bit. Writing a “1” to the pending bit has no effect.

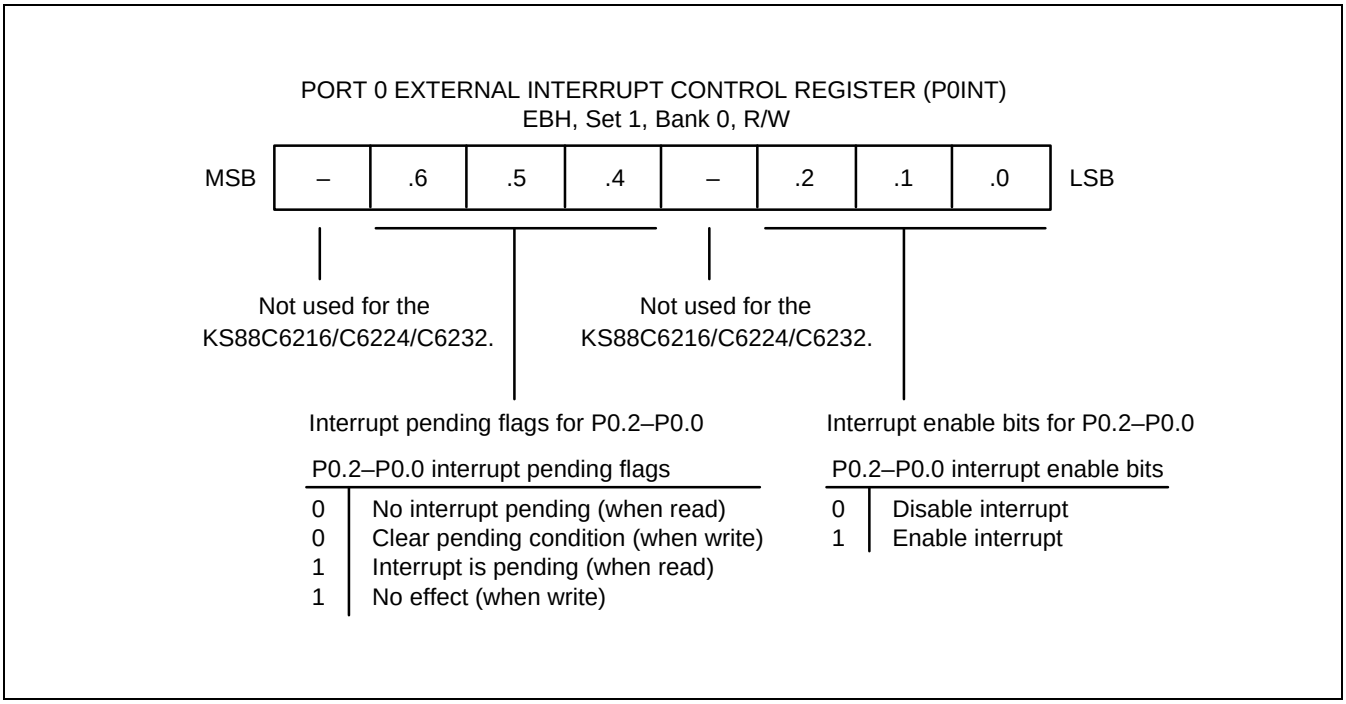


Figure 9-4. Port 0 External Interrupt Control Register (P0INT)

PORT 1

Port 1 is an 3-bit port with individually configurable pins. You can directly access it by writing or reading the port 1 data register, P1 (set 1, bank 0, E1H). You can use port 1 for normal output, input mode, or n-channel open-drain output mode.

The port 1 control register, P1CON (set 1, bank 0, E6H) is used to configure port 1 pins. Each byte contains four bit-pairs and each bit-pair configures one pin.

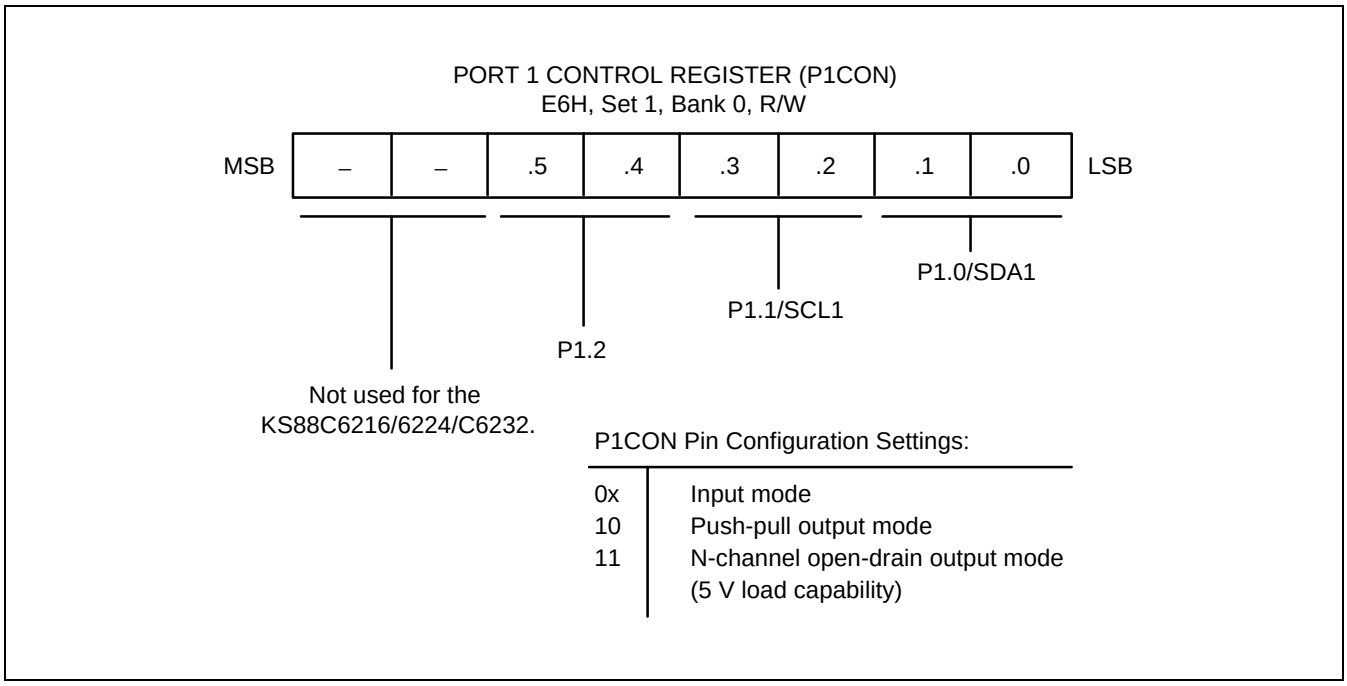


Figure 9-5. Port 1 Control Register (P1CON)

PORT 2

Port 2 is an 8-bit I/O port with individually configurable pins. You can directly access port 2 pins by writing or reading the port 2 data register, P2 (set 1, bank 0, E2H).

Two 8-bit control registers are used to configure port 2 pins: P2CONH (set 1, bank 0, E7H) and P2CONL (set 1, bank 0, E8H) which let you select input mode, normal or PWM push-pull output mode, or PWM n-channel open drain output mode.

PORT 2 CONTROL REGISTER, HIGH BYTE (P2CONH)
E7H, Set 1, Bank 0, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

P2.7/Csync-I
P2.6/PWM6
P2.5/PWM5
P2.4/PWM4

P2CONH Pin Configuration Settings:

bits 7, 6	0x	TTL input mode (Csync-I)
	1x	Push-pull output mode
bits 5–0	00	Input mode
	01	Push-pull output mode
	10	Push-pull PWM output mode (5 V load capability)
	11	N-channel open drain PWM output mode (5 V load capability)

Figure 9-6. Port 2 High-Byte Control Register (P2CONH)

PORT 2 CONTROL REGISTER, LOW BYTE(P2CONL)
E8H, Set 1, Bank 0, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

P2.3/PWM3 P2.2/PWM2 P2.1/PWM1 P2.0/PWM0

P2CONL Pin Configuration Settings:

0x	Input mode
10	Push-pull output mode
11	Push-pull PWM output mode (5 V load capability)

Figure 9-7. Port 2 Low-Byte Control Register (P2CONL)

PORT 3

Port 3 is an 8-bit I/O port with individually configurable pins. You can directly access it by writing or reading the port 3 data register, P3 (set 1, bank 0, E3H). You can selectively configure P3 pins to input or output mode. In input mode, you can also select A/D converter input mode (P3.0–P3.3 only) or normal digital input mode (with or without pull-up resistor). Output mode is push-pull mode or n-channel open-drain mode (P3.4–P3.7 only).

Two 8-bit control registers are used to configure port 3 pins: P3CONH (E9H, set 1, bank 0) for P3.7–P3.4 and P3CONL (set 1, bank 0, EAH) for P3.3–P3.0. Each byte contains four bit-pairs and each bit-pair configures one pin.

PORT 3 CONTROL REGISTER, HIGH BYTE (P3CONH)
E9H, Set 1, Bank 0, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P3.7		P3.6		P3.5		P3.4		

P3CONH Pin Configuration Settings:

00	Input mode
01	Input mode with pull-up resistor
10	Push-pull output mode
11	N-channel open-drain output mode (5 V load capability)

Figure 9-8. Port 3 High-Byte Control Register (P3CONH)

PORT 3 CONTROL REGISTER, LOW BYTE (P3CONL)
EAH, Set 1, Bank 0, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P3.3/ADC3		P3.2/ADC2		P3.1/ADC1		P3.0/ADC0		

P3CONL Pin Configuration Settings:

00	Input mode
01	Analog input mode
1x	Push-pull output mode

Figure 9-9. Port 3 Low-Byte Control Register (P3CONL)

FUNCTION-FIXED PORT

These I/O pins are used only for the input and output of video synchronization signals to the sync processor or IIC-bus interface. The horizontal and vertical sync signals can be monitored directly through the Sync Port Read Data Register (SYNCRD).

Sync signal ports

— Csync-I: Composite (SOG) synchronization input port (TTL level)

— Hsync-I: Horizontal synchronization input (TTL level)

— Vsync-I: Vertical synchronization input and synchro clock for DDC (TTL level)

— Hsync-O: Horizontal synchronization output from the sync processor

— Vsync-O: Vertical synchronization output from the sync processor

— Clamp-O: Clamp signal output with programmable width from the sync processor

DDC and IIC-bus interface ports

— SDA0: DDC and IIC-bus interface serial data

— SCL0: DDC and IIC-bus interface serial clock

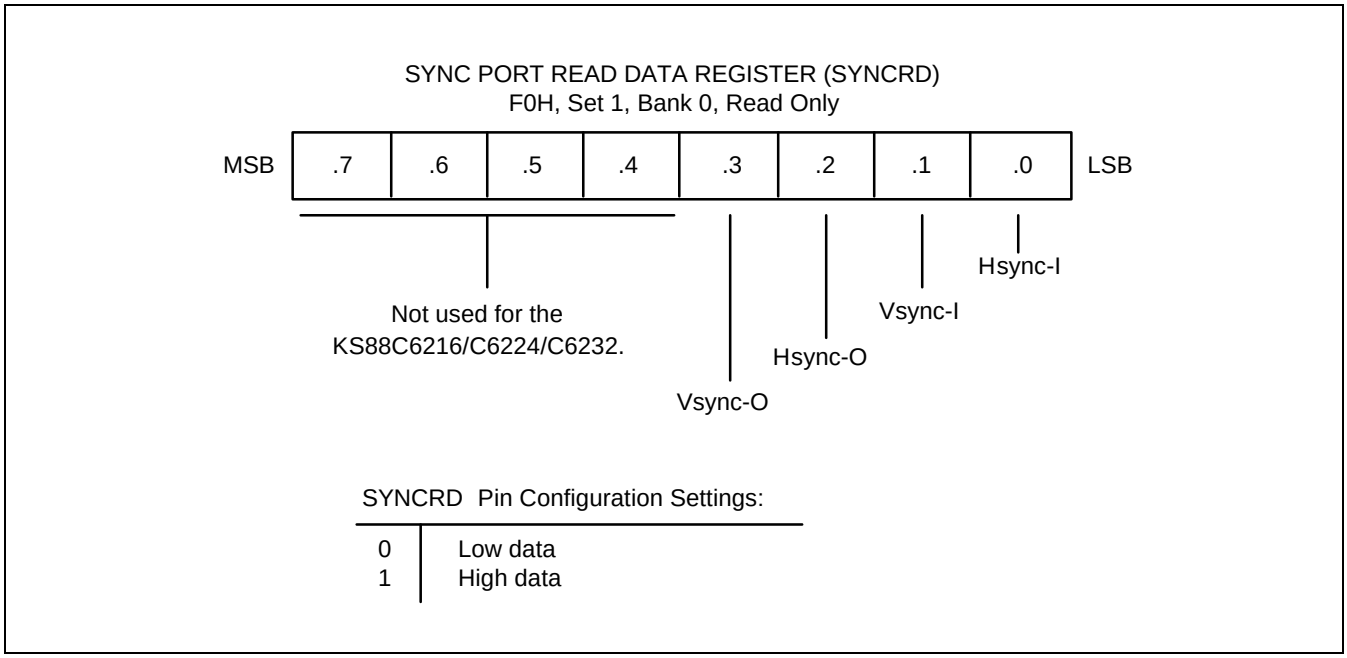


Figure 9-10. Sync Port Read Data Register (SYNCRD)

☞ PROGRAMMING TIP — Configuring I/O Port Pins to Specification

The following sample program shows you how to configure the KS88C6216/C6224/C6232 I/O ports to specification.
The program comments explain the effect of the settings:

```
•
•
•
SB0                          ;  Select bank 0

LD      P0CONH,#0FFH         ;  Set port 0 high byte to push-pull output mode

LD      P0CONL,#0D5H         ;  Set P0.3 to push-pull output mode
                             ;  Set P0.0–P0.2 to rising edge interrupt mode

LD      P0INT,#0FH           ;  Enable port 0 external interrupt

LD      P1CON,#00H           ;  Set port 1 to input mode

LD      P2CONH,#3FH          ;  Set port 2 high byte to PWM n-channel open-drain
                             ;  output mode (5-volt capability) and Csync input mode

LD      P2CONL,#0FFH         ;  Set port 2 low byte to PWM push-pull output mode

LD      P3CONH,#0AAH         ;  Set port 3 high byte to push-pull output mode

LD      P3CONL,#55H          ;  Set port 3 low byte to analog input mode

•
•
•
```

10 BASIC TIMER

OVERVIEW

KS88C6216/C6224/C6232 has a default timer: an 8-bit *basic timer.*

You can use the basic timer (BT) in two different ways:

— As a watchdog timer, it provides an automatic reset mechanism in the event of a system malfunction, or

— Signals the end of the required oscillation stabilization interval after a reset or a Stop mode release.

The functional components of the basic timer block are:

— Clock frequency divider (f_{OSC} divided by 4096, 1024, or 128) with multiplexer

— 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH, read-only)

— Basic timer control register, BTCON (set 1, D3H, read/write)

BASIC TIMER CONTROL REGISTER (BTCON)

The basic timer control register, BTCON, is used to select the input clock frequency, to clear the basic timer counter and frequency dividers, and to enable or disable the watchdog timer function. It is located in set 1, address D3H, and is read/write addressable using register addressing mode.

A reset clears BTCON to "00H". This enables the watchdog function and selects a basic timer clock frequency of f_{OSC}/4096. To disable the watchdog function, you must write the signature code "1010B" to the basic timer register control bits BTCON.7–BTCON.4.

The 8-bit basic timer counter, BTCNT (set 1, bank 0, FDH), can be cleared at any time during the normal operation by writing a "1" to BTCON.1. To clear the frequency dividers for both the basic timer input clock and the timer M0 clock (unless timer M0 uses an external clock source), you should write a "1" to BTCON.0.

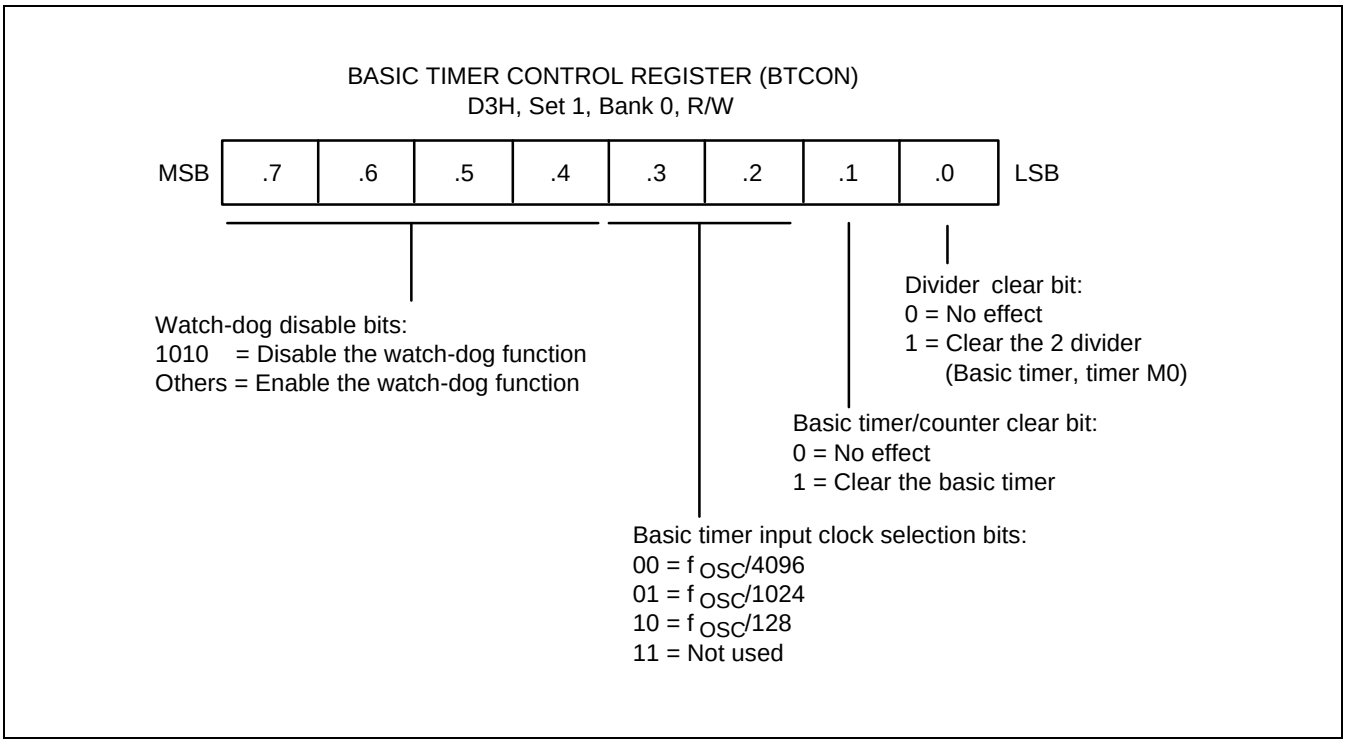


Figure 10-1. Basic Timer Control Register (BTCON)

BASIC TIMER FUNCTION DESCRIPTION

Watchdog Timer Function

You can program the basic timer overflow signal (BTOVF) to generate a reset by setting BTCON.7–BTCON.4 to any value other than "1010B" (The "1010B" value disables the watchdog function). A reset clears BTCON to "00H", automatically enabling the watchdog timer function. A reset also selects the CPU clock (as determined by the current CLKCON register setting),divided by 4096, as the BT clock.

Reset whenever a basic timer counter overflow occurs. During the normal operation, the application program must prevent the overflow and the accompanying reset operation from occurring. To do this, the BTCNT value must be cleared (by writing a "1" to BTCON.1) at regular intervals.

If a system malfunction occurs due to circuit noise or some other error condition, the BT counter clear operation will not be executed and a basic timer overflow will occur, initiating a reset. In other words, during the normal operation, the basic timer overflow loop (a bit 7 overflow of the 8-bit basic timer counter, BTCNT) is always broken by a BTCNT clear instruction. If a malfunction does occur, a reset is triggered automatically.

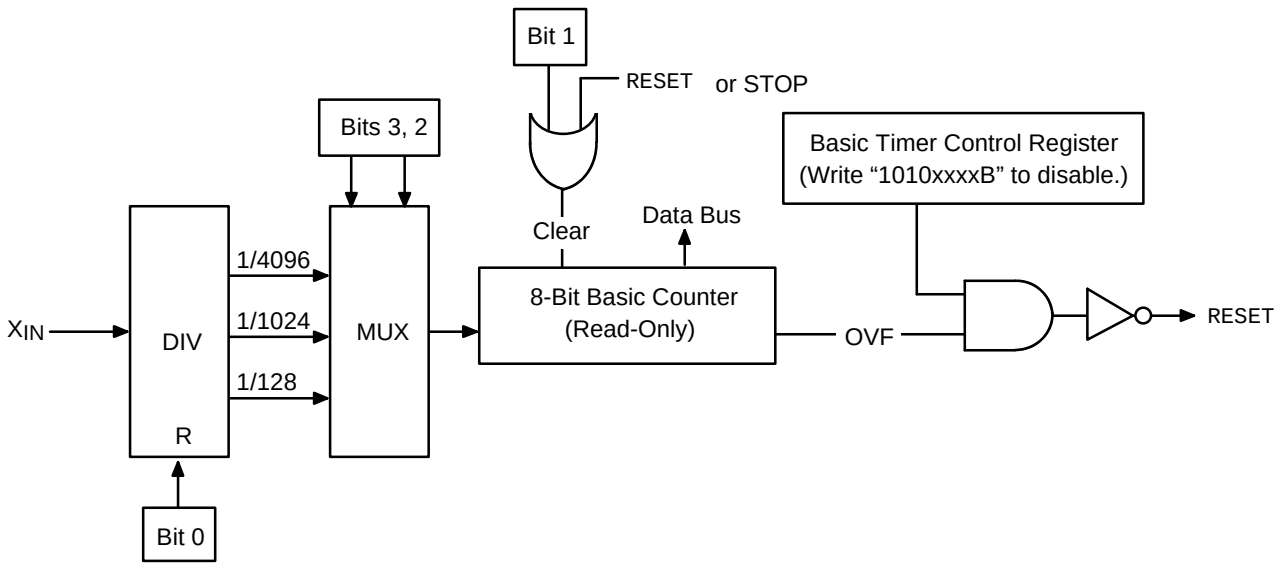
Oscillation Stabilization Interval Timer Function

You can also use the basic timer to program a specific oscillation stabilization interval after a reset or when stop mode has been released by an external interrupt.

In Stop mode, whenever a reset or an external interrupt occurs, the oscillator starts. The BTCNT value then starts increasing at the rate of f_{OSC}/4096 (for reset), or at the rate of the preset clock source (for an external interrupt). When BTCNT.4 overflows, a signal is generated to indicate that the stabilization interval has elapsed and to gate the clock signal off to the CPU so that it can resume the normal operation.

In summary, the following events occur when Stop mode is released:

1. During the stop mode, a power-on reset or an external interrupt occurs to trigger the stop mode release and oscillation starts.
2. If a power-on reset occurred, the basic timer counter would increase at the rate of f_{OSC}/4096. If an external interrupt is used to release Stop mode, the BTCNT value increases at the rate of the preset clock source.
3. Clock oscillation stabilization interval begins and continues until bit 4 of the basic timer counter overflows.
4. When a BTCNT.4 overflow occurs, the normal CPU operation resumes.



NOTE: In a power-on reset operation, the CPU is idle during the required oscillation stabilization interval when BTCNT.4 is set after releasing from RESET or STOP mode, CPU clock starts.

Figure 10-2. Basic Timer Block Diagram

☞ PROGRAMMING TIP — Configuring the Basic Timer

This example shows how to configure the basic timer to sample specifications:

```
        ORG     0100H
RESET   DI                          ; Disable all interrupts
        SB0                         ; Select bank 0
        LD      BTCON,#0AAH         ; Disable the watchdog timer
        LD      CLKCON,#98H         ; Non-divided clock
        CLR     SYM                 ; Disable global and fast interrupts
        CLR     SPL                 ; Stack pointer low byte ← "0"
                                    ; Stack area starts at 0FFH
        •
        •
        •
        SRP     #0C0H               ; Set register pointer ← 0C0H
        EI                          ; Enable interrupts
        •
        •
        •
MAIN    LD      BTCON,#52H          ; Enable the watchdog timer
                                    ; Basic timer clock: fOSC/4096
                                    ; Clear basic timer counter
        NOP
        NOP
        •
        •
        •
        JP      T,MAIN
        •
        •
        •
```

11 TIMER M0

OVERVIEW

The 8-bit *timer M0* is for monitor application. Timer M0 includes capture timer mode using the appropriate TM0CON setting.

Timer M0 has the following functional components:

— Clock frequency divider (f_{OSC} divided by 128 or 4) with multiplexer

— 2-bit prescaler for the timer M0 input clock

— 8-bit counter (TM0CNT; set1, D0H, read-only) and 8-bit reference data register (TM0DATA; set1, D1H, read-only)

— Timer M0 capture or overflow interrupt (IRQ0, vector FAH, FCH) generation

— Timer M0 control register, TM0CON (set 1, D2H, read/write)

FUNCTION DESCRIPTION

Capture Timer Function

The timer M0 module can generate two interrupts: the timer M0 capture interrupt (TM0INT), and the timer M0 overflow interrupt (TM0OVF). TM0INT belongs to interrupt level IRQ0, and is assigned the separate vector address, FCH. TM0OVF is interrupt level IRQ0, vector FAH.

The TM0INT and TM0OVF pending conditions are automatically cleared by hardware after they are serviced.

In capture timer mode, a signal edge that is detected at the TM0CAP pin opens a gate and loads the current counter value into the timer M0 data register (TM0DATA). You can select rising or falling edge to trigger this operation.

Both kinds of timer M0 interrupts can be used in capture mode: the timer M0 overflow interrupt is generated whenever a counter overflow occurs; the timer M0 capture interrupt is generated whenever the counter value is loaded into the timer M0 data register.

By reading captured data value in TM0DATA, and assuming a specific value for the timer M0 clock frequency, you can calculate the internal time of the signal being input to the TM0CAP pin or the vertical sync output signal being output from the sync-processor module.

Timer M0 Control Register (TM0CON)

You use the timer M0 control register, TM0CON, to

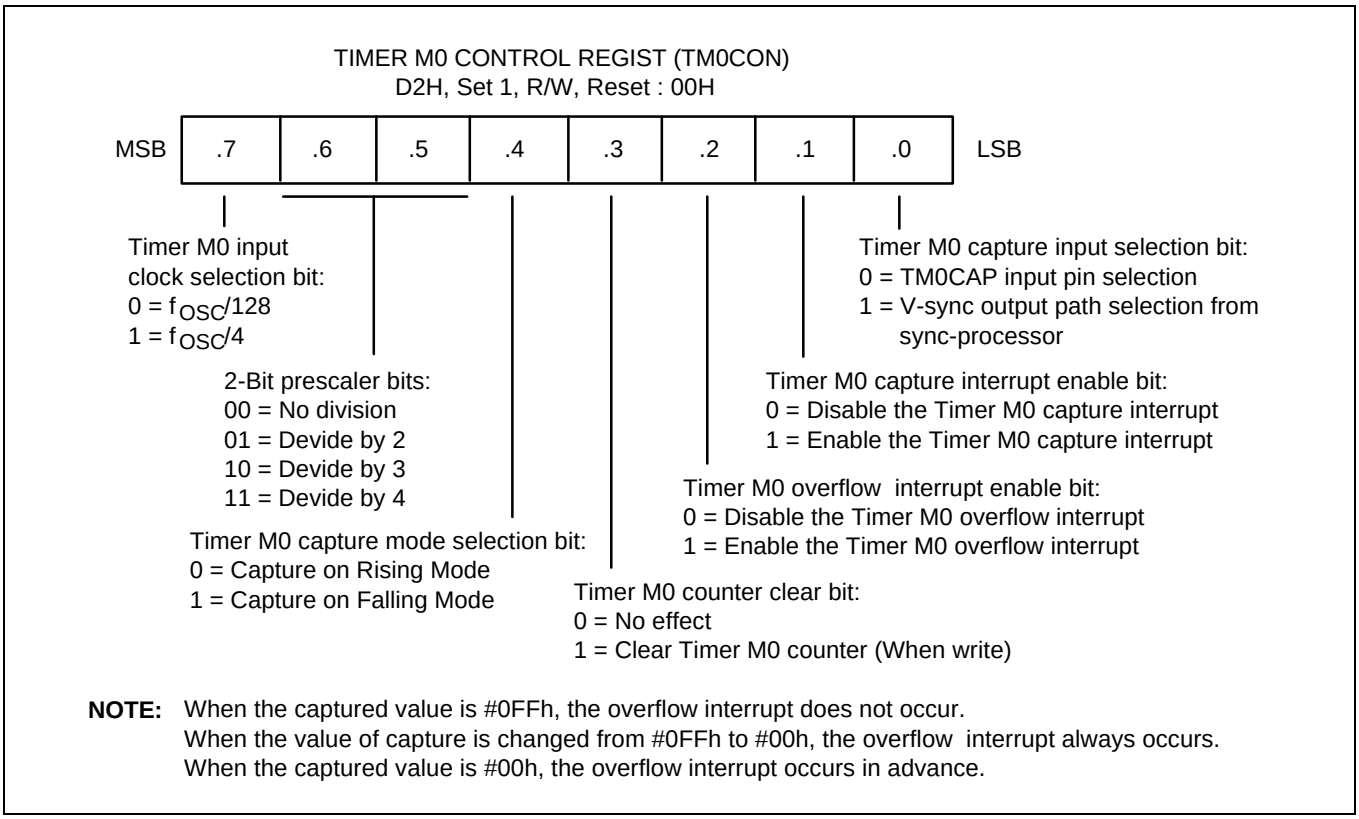
— Select the timer M0 operating mode (capture mode)
— Select the timer M0 input clock frequency
— Clear the timer M0 counter, TM0CNT
— Enable the timer M0 overflow interrupt and timer M0 capture interrupt
— Select a 2-bit prescaler value for the Timer M0 input clock

TM0CON is located in set 1, at address D2H, and is read/write addressable using Register addressing mode.

A reset clears TM0CON to "00H". This sets timer M0 to disable capture timer mode, selects an input clock frequency of $f_{OSC}/128$, and disables timer M0 overflow and capture interrupts. You can clear the timer M0 counter at any time during the normal operation by writing a "1" to TM0CON.2.

The timer M0 overflow interrupt (TM0OVF) is in the interrupt level IRQ0 and has the vector address FAH. When the timer M0 capture interrupt is disabled, the Timer M0 overflow interrupt by clock (f_{OSC}) is possible. When a timer M0 overflow interrupt occurs and is serviced by the CPU, the pending condition is cleared automatically by hardware.

To enable the timer M0 capture interrupt (IRQ0, vector FCH), you must write TM0CON.1 to "1". There is no pending bit cleared by software or static read bit which is H/W pending. After the interrupt request is serviced, the pending condition is automatically cleared by hardware.



NOTE: When the captured value is #0FFh, the overflow interrupt does not occur.
When the value of capture is changed from #0FFh to #00h, the overflow interrupt always occurs.
When the captured value is #00h, the overflow interrupt occurs in advance.

Figure 11-1. Timer M0 Control Register (TM0CON)

BLOCK DIAGRAM

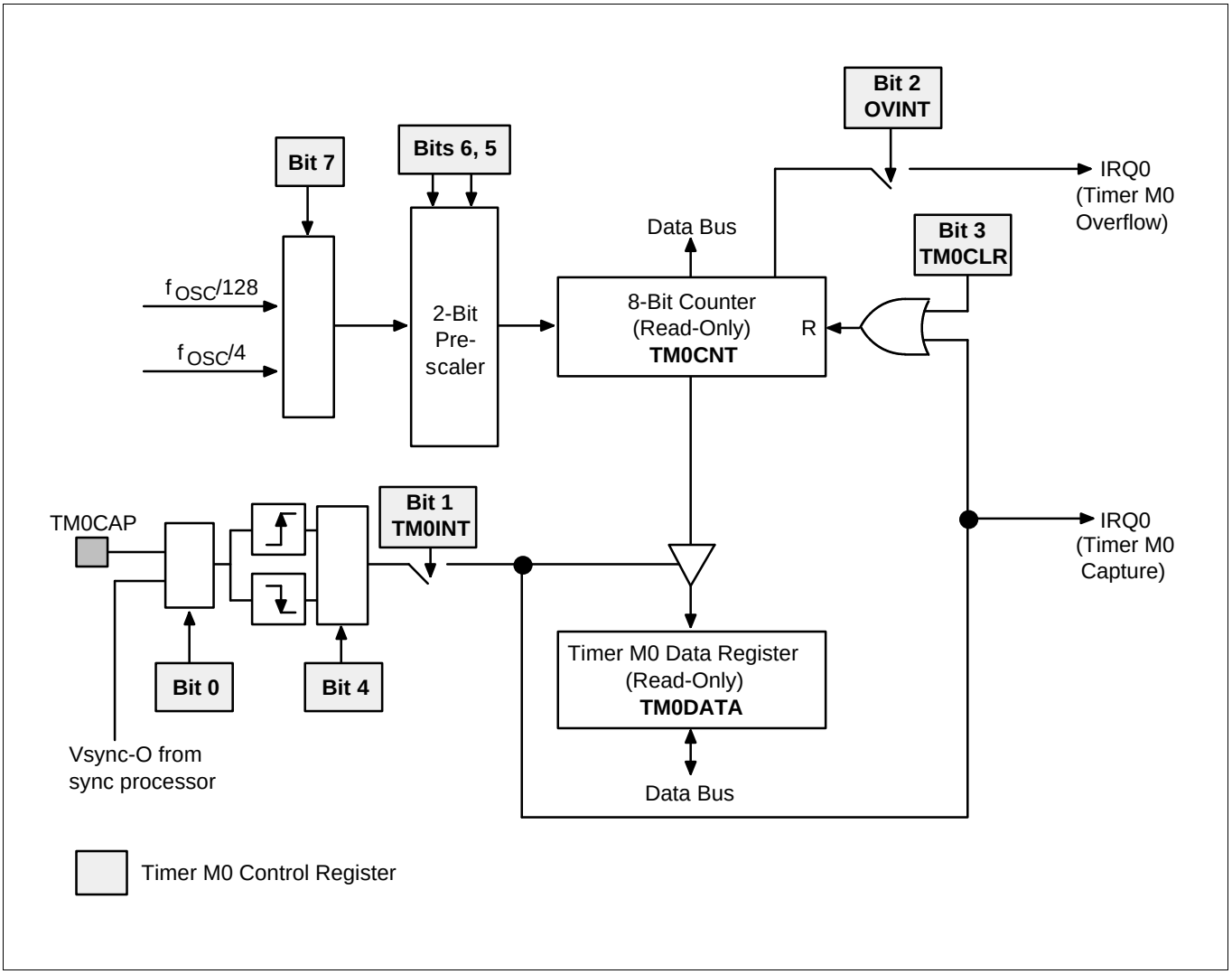


Figure 11-2. Timer M0 Functional Block Diagram

12 TIMER M1

OVERVIEW

The 12-bit *timer M1* is an 12-bit timer/counter for monitor application. Timer M1 offers capture/overflow timer mode using the appropriate TM1CON setting.

Timer M1 has the following functional components:

— Clock frequency selector as the timer M1 clock (f_{OSC} divided by 4096 or 512, H-syncI or C-syncI from sync-processor, TM1CK) with multiplexer

— Capture signal selector from V-syncO (sync-processor) or the timer M2 interval time

— 12-bit counter (TM1CNTH, TM1CNTL; set1, bank0, F1H, F2H, read-only) and 12-bit reference data register (TM1DATAH, TM1DATAL; set1, bank0, F3H, F4H, read-only)

— Timer M1 capture or overflow interrupt (IRQ1, vector F6H) generation

— Timer M1 control register, TM1CON (set 1, bank0, F5H, read/write)

FUNCTION DESCRIPTION

Overflow Timer Function

The timer M1 module generates an overflow signal whenever the timer M1 counter overflow occurs. If you set the timer M1 overflow (or capture) interrupt enable bit, TM1CON.3, to "1", an interrupt is generated whenever an overflow state is detected. After the interrupt request is generated, the counter register value is cleared and counting resumes from "00H".

Program software can poll the pending bit TM1CON.2 to detect when a timer M1 overflow (or capture) interrupt pending condition exists (TM1CON.2 = "1"). When the interrupt request is acknowledged by the CPU and the service routine starts, the interrupt service routine must then clear the interrupt pending condition by writing a "0" to TM1CON.2.

Capture Timer Function

The Timer M1 module can generate, the timer M1 capture (or overflow) interrupt (TM1INT). TM1INT belongs to interrupt level IRQ1, and is assigned the vector address, F6H.

In capture timer mode, a capture signal from V-syncO (sync-processor) or the timer M2 interval timer opens a gate and loads the current counter value into the timer M1 data register (TM1DATA). You can select V-syncO or the timer M2 interval timer as the capture signal source to trigger this operation.

By reading captured data value in TM1DATAH and TM1DATAL, and assuming a specific value for the timer M1 clock frequency, you can calculate the frequency of the signal being input to the H-syncI or C-syncI from sync-processor (TM1CK) by capture signal.

Timer M1 Control Register (TM1CON)

You use the timer M1 control register, TM1CON, to

— Select the capture signal source

— Select the timer M1 clock input

— Clear the timer M1 counter, TM1CNTH and TM1CNTL

— Enable the timer M1 capture or overflow interrupt

— Clear the timer M1 capture or overflow interrupt pending bit

— Enable the timer M1 capture mode

— Select V-syncO capture edge as capture signal source (When TM1CON.7 = "1")

TM1CON is located in set 1, bank0, at address F5H, and is read/write addressable using Register addressing mode.

The setting for bit-pair TM1CON.0 and TM1CON.1 selects the timer M1 counter clock input. The timer M1 capture or overflow interrupt (TM1INT) is in the interrupt level IRQ1 and has the same vector address F6H. TM1CON.2 is the interrupt pending flag for the timer M1 capture or overflow interrupt. To clear a timer M1 interrupt pending condition, the interrupt service routine must write a "0" to TM1CON.2 after the CPU has acknowledged the request. TM1CON.4 is flag to clear the 12-bit Timer M1 counter. To enable the timer M1 capture mode, it writes a "1" to TM1CON.5. TM1CON.7 is flag to select the capture signal source (Timer M2 interval time or V-sync0 from sync-processor) and TM1CON.6 is flag to select the capture edge as the V-sync0 capture signal source.

A reset operation clears TM1CON to "00H", selecting the CPU clock divided by 4,096 as the timer M1 clock and disabling the timer M1 capture or overflow interrupt.

TIMER M1 CONTROL REGISTER (TM1CON)
F5H, Set 1, R/W, Reset : 00H

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

Timer M1 capture signal source selection bit:
0 = signal from the Timer M2 interval time
1 = V-syncO from sync-processor

V-syncO Capture source edge selection bit:
0 = V-syncO rising edge from sync-processor
1 = V-syncO falling edge from sync-processor

Timer M1 capture mode enable bit:
0 = Disable the Timer M1 Capture
1 = Enable the Timer M1 Capture

Timer M1 counter clear bit(When write):
0 = No effect
1 = Clear Timer M1 counter

Timer M1 capture or overflow interrupt enable bit:
0 = Disable Timer M1 capture or overflow interrupt
1 = Enable Timer M1 capture or overflow interrupt

Timer M1 capture or overflow interrupt pending bit:
0 = Interrupt is not pending (When read)
0 = Clear this pending bit (When write)
1 = Interrupt is pending (When read)
1 = No effect (When write)

Timer M1 clock input selection bits:
00 = $f_{OSC}/4096$
01 = $f_{OSC}/512$
10 = H-syncI or C-syncI from sync-processor
11 = External Clock (TM1CK)

Figure 12-1. Timer M1 Control Register (TM1CON)

BLOCK DIAGRAM

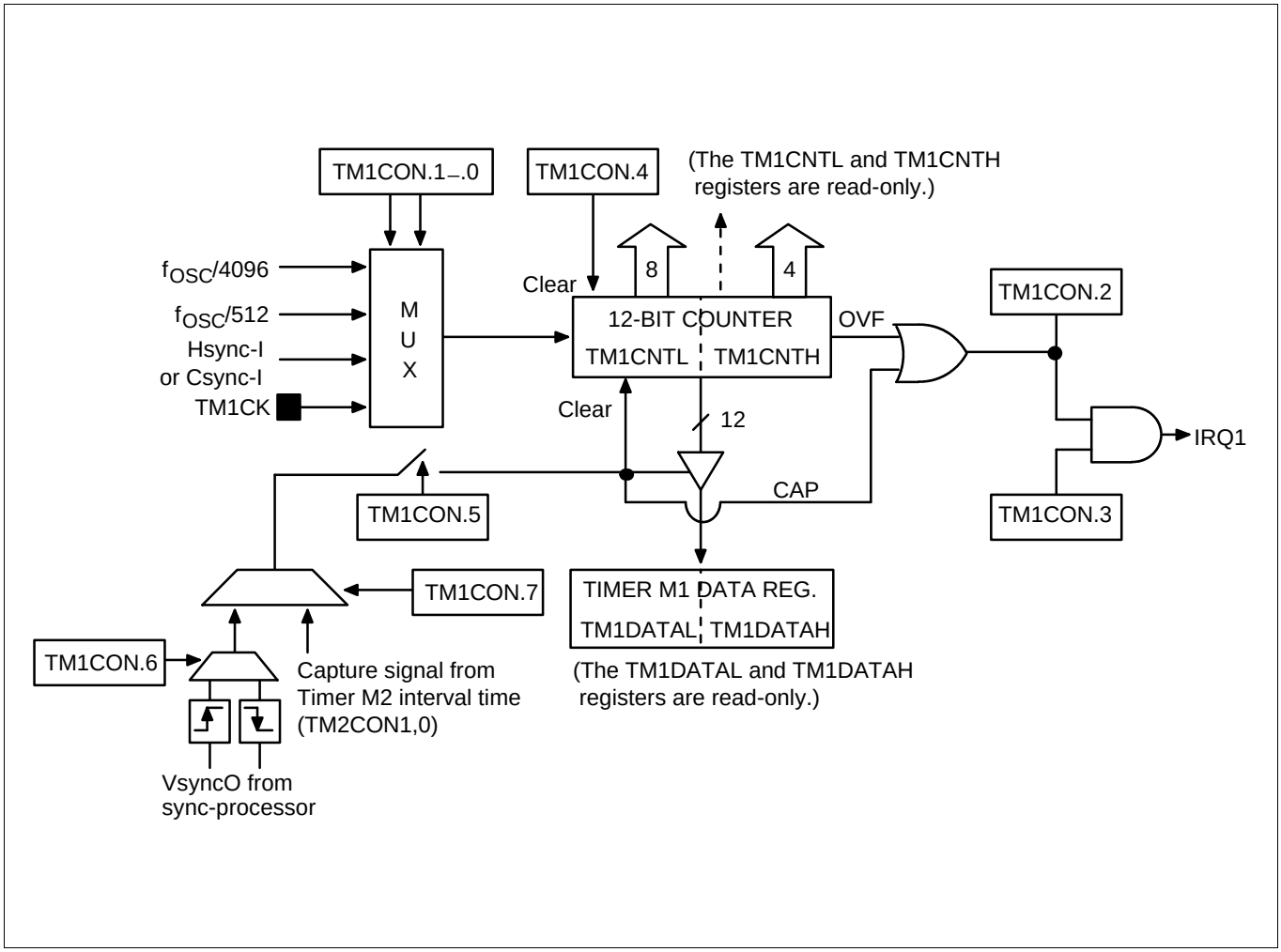


Figure 12-2. Timer M1 Functional Block Diagram

13 TIMER M2

OVERVIEW

The interval timer M2 is no-counter timer for monitor application. Timer M2 offers interval timer mode using the appropriate TM1CON setting.

Timer M2 has the following functional components:

— 5-bit scaler by f_{OSC}/1000 for timer M2 interval source

— Timer M1 capture source selector (When TM1CON.5 is "1") with 2-bit scaler

— Timer M2 interval interrupt (IRQ1, vector EEH) generation

— Timer M2 control register, TM2CON (set 1, F6H, read/write)

FUNCTION DESCRIPTION

Interval Timer Function

The timer M2 module generates an interval interrupt whenever the TM2CON.2 is "1". TM2INT belongs to the interrupt level IRQ1, and is assigned the separate vector address, EEH. The TM2INT pending condition is automatically cleared by hardware when it has been serviced.

Timer M2 Control Register (TM2CON)

You use the timer M2 control register, TM2CON, to

— Select the interval time signal source by 5-bit scaler

— Enable the timer M2 interval interrupt

— Select timer M1 capture interval time by 2-bit scaler (When TM1CON.5 = "1")

TM2CON is located in set 1, bank0, at address F6H, and is read/write addressable using Register addressing mode.

A reset operation clears TM2CON to "F8H" (11111000B), thereby setting the 5-bit scaler value to be divided by 32.

TIMER M2 CONTROL REGISTER (TM2CON)
F6H, Set 1, R/W, Reset : F8H

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

5-bit scaler bits:
00000 = No devision
00001 = Devide by 2
00010 = Devide by 3
. . . .
11111 = Devide by 32

Timer M2 interval interrupt enable bit:
0 = Disable timer M2 interval interrupt
1 = Enable timer M2 interval interrupt

Timer M1 capture interval time selection bits:
00 = Timer M2 interval (bypass)
01 = Timer M2 interval x 10
10 = Timer M2 interval x 20
11 = Timer M2 interval x 30

NOTES:

1. When the timer M1 capture mode is enabled (TM1CON.5 = “1”), the value of 5-bit or 2-bit scaler can be changed only in the timer M1 capture interrupt routine.
2. When the timer M1 capture mode is disabled (TM1CON.5 = “0”), the value of 5-bit scaler can be changed only in the timer M2 interval interrupt routine.

Figure 13-1. Timer M2 Control Register (TM2CON)

BLOCK DIAGRAM

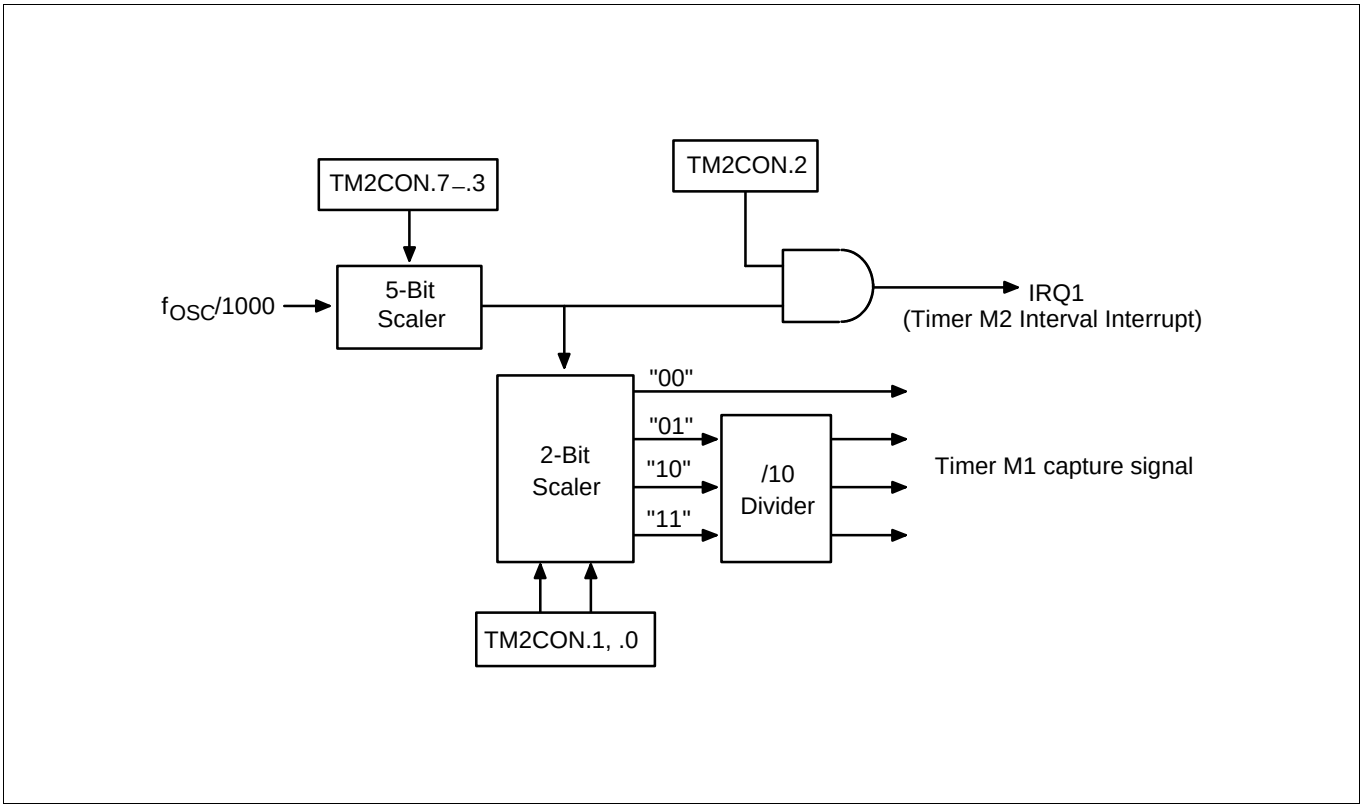


Figure 13-2. Timer M2 Functional Block Diagram

14 ANALOG-TO-DIGITAL CONVERTER

OVERVIEW

The 8-bit A/D converter (ADC) module of KS88C6216/C6224/C6232 employs successive approximation logic to convert analog levels entering one of the four input channels to equivalent 8-bit digital values. The analog input level must lie between the AV_{REF} and AV_{SS} values. The A/D converter has the following components:

— Analog comparator with successive approximation logic

— D/A converter logic (resistor string type)

— 8-bit ADC control register (ADCON)

— Four multiplexed analog data input pins (ADC0–ADC3)

— 8-bit A/D conversion data output register (ADDATA)

— 8-bit digital input port (Alternatively, I/O port)

— AV_{REF} and AV_{SS} pins

FUNCTION DESCRIPTION

To initiate an analog-to-digital conversion procedure, write the channel selection data in the A/D converter control register ADCON to select one of the four analog input pins (ADCn, n = 0–3) and set the conversion start or enable bit, ADCON.0. The read-write ADCON register is located in set 1, bank0, at address F7H.

During the normal conversion, ADC block initially sets the successive approximation register to 80H (approximately the half-way point of an 8-bit register). This register is then updated automatically in each conversion step. The successive approximation block performs 8-bit conversions for one input channel at a time. You can dynamically select different channels by manipulating the channel selection bit value (ADCON.6–.4) in the ADCON register. To start the A/D conversion, you should set, ADCON.0. When a conversion is completed, ADCON.3, the end-of-conversion (EOC) bit, is automatically set to 1 and the result is dumped into the ADDATA register where it can be read. The A/D converter then enters an idle state. Remember to read the contents of ADDATA before another conversion starts. Otherwise, the previous result will be overwritten by the next conversion result.

NOTE

As the A/D converter does not include any sample-and-hold circuitry, it is very important to keep the fluctuation in the analog level at the ADC0–ADC3 input pins to an absolute minimum during the conversion process. Any change in the input level, perhaps due to noise, will invalidate the result. If the chip enters to STOP or IDLE mode in conversion process, there will be a leakage current path in A/D block. You must use STOP or IDLE mode after A/D converting operation is finished.

CONVERSION TIMING

The A/D conversion process requires 4 steps (8 clock edges) to convert each bit. Therefore, a total of 34 clocks are required to complete an 8-bit conversion. With an 8 MHz f_{OSC} clock frequency, one clock cycle is 1 µs (when ADCON.2, .1 are "01"). If each bit conversion requires 4 clocks, the conversion rate is calculated as follows:

start 1 clock + (4 clocks/bit × 8 bits) + EOC 1 clock = 34 clocks, 1 µs x 34 = 34 µs at 8 MHz

A/D CONVERTER CONTROL REGISTER (ADCON)

The A/D converter control register, ADCON, is located at address F7H in set 1, bank0. It has three functions:

— Analog input pin selection (bits 4, 5, and 6)
— End-of-conversion status detection (bit 3)
— Clock source selection (bits 2 and 1)
— A/D operation start or enable (bit 0)

After a reset, the ADC0 pin is automatically selected as the analog data input pin, and the start bit is turned off.

You can select only one analog input channel at a time. Other analog input pins (ADC0–ADC3) can be selected dynamically by manipulating the ADCON.6–.4 bits.

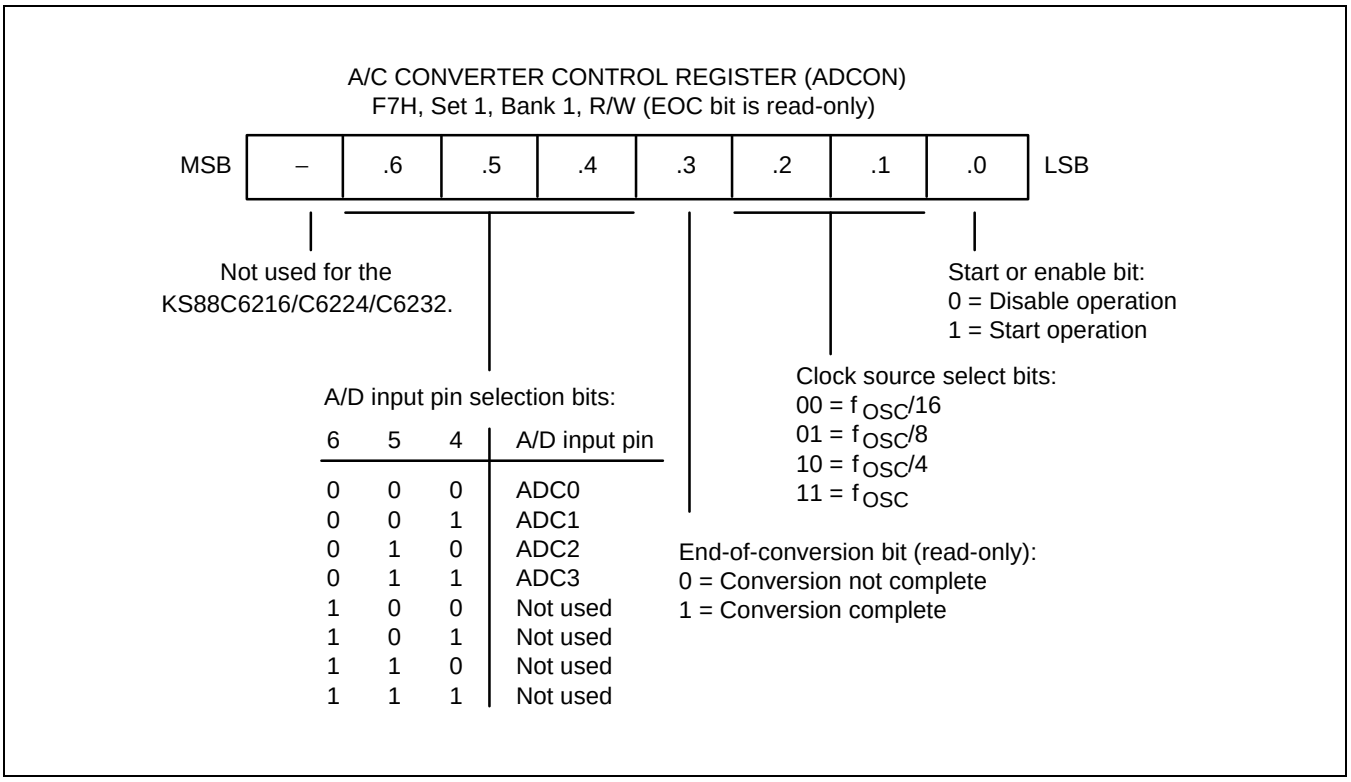


Figure 14-1. A/D Converter Control Register (ADCON)

INTERNAL REFERENCE VOLTAGE LEVELS

In the ADC function block, the analog input voltage level is compared to the reference voltage. The analog input level must remain within the range of AV_{SS} (V_{SS}) to AV_{REF} (usually, $AV_{REF} = V_{DD}$).

Different reference voltage levels are generated internally along the resistor tree during the analog conversion process for each conversion step. The reference voltage level for the first conversion bit is always 1/2 AV_{REF}.

BLOCK DIAGRAM

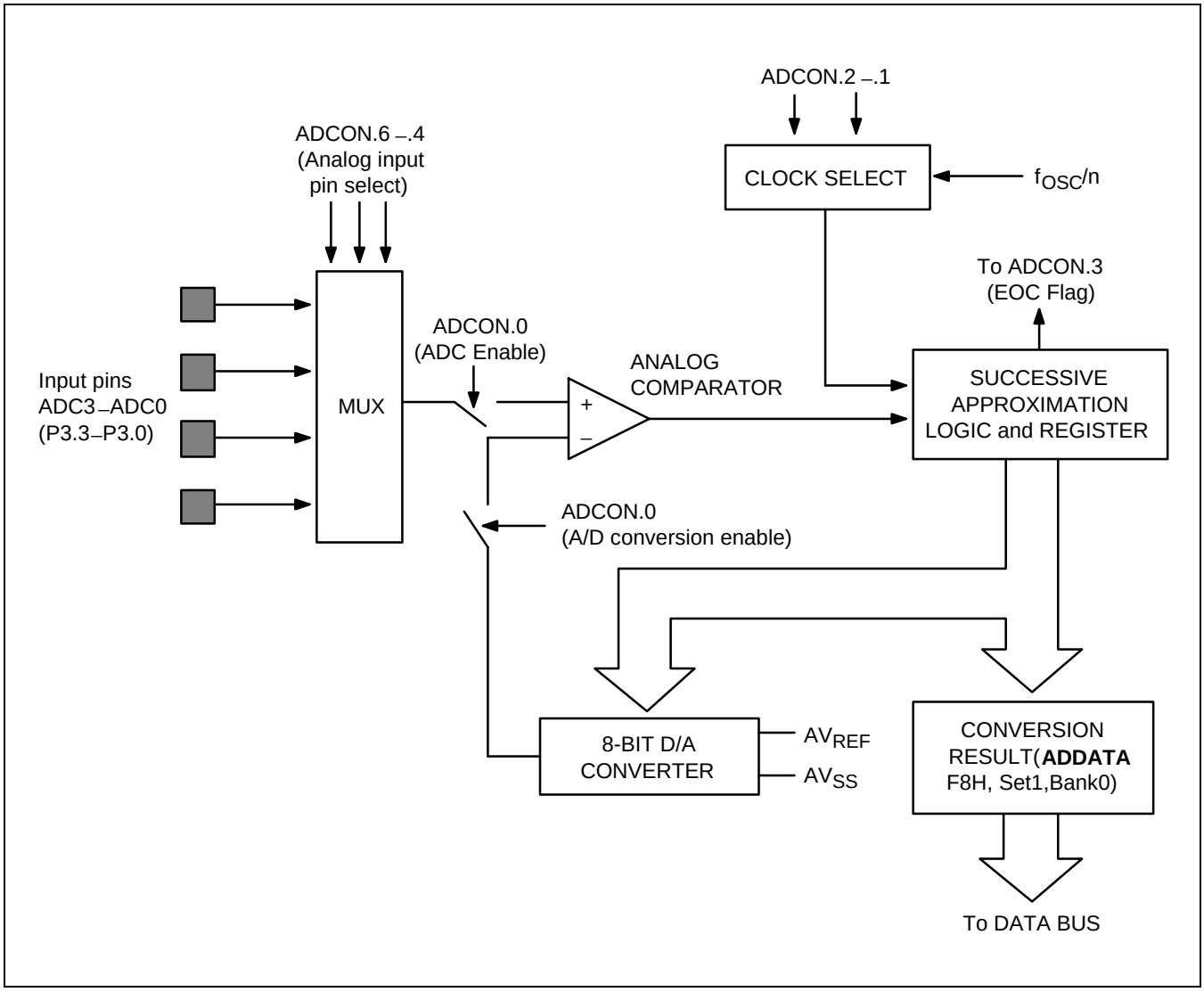


Figure 14-2. A/D Converter Functional Block Diagram

Table 14-1. A/D Converter Electrical Characteristics

(T_A = – 40°C to + 85°C, V_{DD} = 2.7 V to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Resolution			–	8	–	bit
Total accuracy		V_{DD} = 5 V Conversion time = 5µs AV_{REF} = 5 V AV_{SS} = 0 V	–	–	± 2	LSB
Integral linearity error	ILE			–	± 1	
Differential linearity error	DLE			–	± 1	
Offset error of top	EOT			± 1	± 2	
Offset error of bottom	EOB			± 0.5	± 2	
Conversion time (1)	t_{CON}	8 bit conversion 34 x n/f_{OSC} (3), n=1,4,8,16	17	–	170	µs
Analog input voltage	V_{IAN}	–	AV_{SS}	–	AV_{REF}	V
Analog input impedance	R_{AN}	–	2	1000	–	mΩ
Analog reference voltage	AV_{REF}	–	2.5	–	V_{DD}	V
Analog ground	AV_{SS} (4)	–	V_{SS}	–	V_{SS}	V
Analog input current	I_{ADIN}	AV_{REF} = V_{DD} = 5V	–	–	10	µA
Analog block Current (2)	I_{ADC}	AV_{REF} = V_{DD} = 5V	–	1	3	mA
		AV_{REF} = V_{DD} = 3V		0.5	1.5	mA
		AV_{REF} = V_{DD} = 5V When power down mode		100	500	nA

NOTES:

1. "Conversion time" is the time required from the moment a conversion operation starts until it ends.
2. I_{ADC} is an operating current during the A/D conversion.
3. f_{OSC} is the main oscillator clock.
4. V_{SS} shaves with AV_{SS} for KS88C6216/C6224/C6232.

15 PULSE WIDTH MODULATION

PWM MODULE

The KS88C6216/C6224/C6232 microcontrollers include seven 8-bit PWM circuits. They are PWM0–PWM6. The operation of all PWM circuits is controlled by a single control register, PWMCON.

The PWM counter, a 8-bit incrementing counter, is used by the 8-bit PWM circuits. To start the counter and enable the PWM circuits, set PWMCON.5 to "1". If the counter is stopped, it retains its current count value. When restarted, it resumes counting from the retained count value.

By modifying the prescaler value, you can divide the input clock by one (non-divided), two, three, or four. The prescaler output is the clock frequency of the PWM counter.

The PWM counter overflows when it reaches "FFH", and then continues counting from zero.

PWM CONTROL REGISTER (PWMCON)

The control register for the PWM module, PWMCON, is located in set 1, bank 1, at register address E7H. You use PWMCON bit settings to control the following functions in the 8-bit:

— PWM counter operation: stop/start (or resume counting)

A reset clears PWMCON to "00H", disabling all PWM functions.

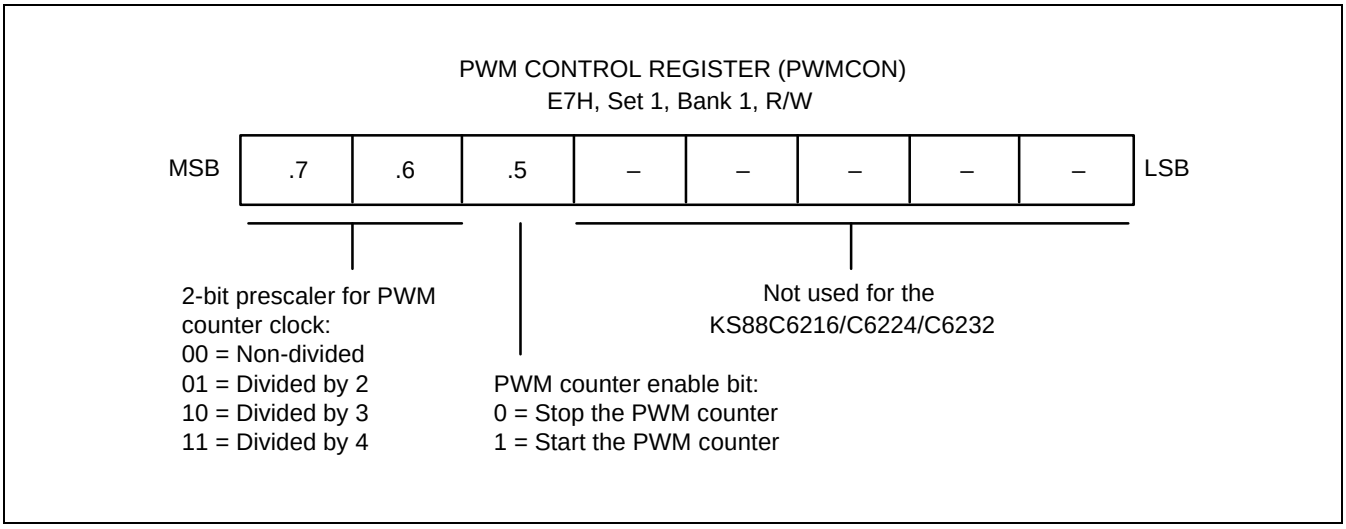


Figure 15-1. PWM Control Register (PWMCON)

PWM0–PWM6

The KS88C6216/C6224/C6232 microcontrollers include seven 8-bit PWM circuits, PWM0–PWM6. Each 8-bit PWM data unit is comprised of an 8-bit basic frame. The 8-bit PWM circuits have the following components:

— 8-bit counter
— 8-bit comparators
— 8-bit PWM data registers (PWM0–PWM6)
— PWM output pins (PWM0–PWM6)

The PWM0–PWM6 circuits are controlled by the PWMCON register (set 1, bank 1, E7H).

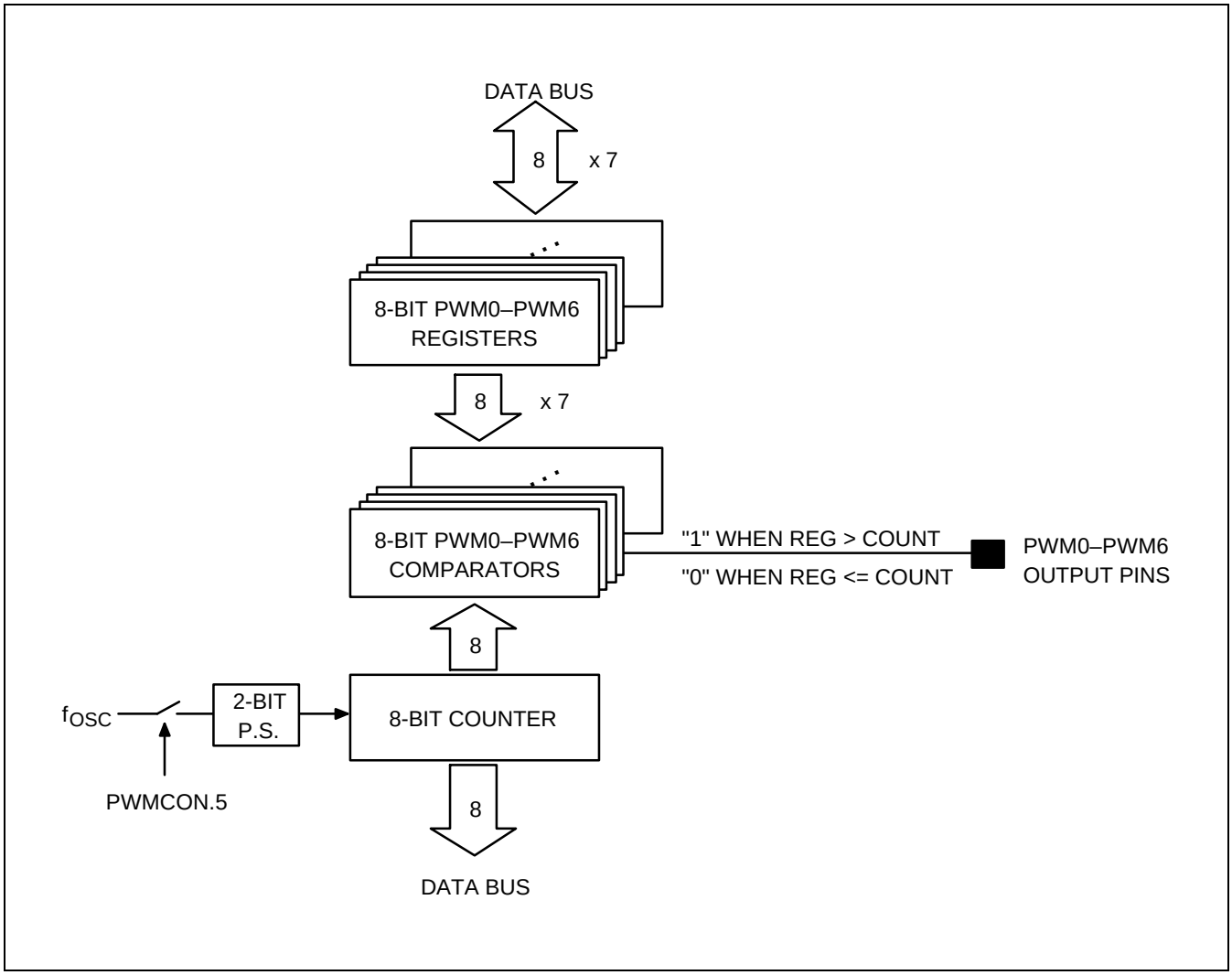


Figure 15-2. Block Diagram for PWM0–PWM6

PWM0–PWM6 FUNCTION DESCRIPTION

All the seven 8-bit PWM circuits have an identical function and each has its own 8-bit data register and 8-bit comparator. Each circuit compares a unique data register value to the 8-bit PWM counter.

The PWM0–PWM6 data registers are located in set 1, bank 1, at locations E0H–E6H, respectively. These data registers are read/write addressable. By loading specific values into the respective data registers, you can modulate the pulse width at the corresponding PWM output pins, PWM0–PWM6. (PWM0–PWM6 correspond to port 2 pins P2.0–P2.6.

The level at the output pins toggles High and Low at a frequency equal to the counter clock, divided by 256 (2^8). The duty cycle of the 8-bit PWM pins ranges from 0% to 99.6% (255/256), based on the corresponding data register values.

To determine the output duty cycle of an 8-bit PWM circuit, its 8-bit comparator sends the output level High when the data register value is greater than the lower 8-bit count value. The output level is Low when the data register value is less than or equal to the lower 8-bit count value. The output level at the PWM0–PWM6 pins remains at Low level for the first 256 counter clocks. Then, each PWM waveform is repeated continuously, at the same frequency and duty cycle, until one of the following three events occurs:

— The counter is stopped

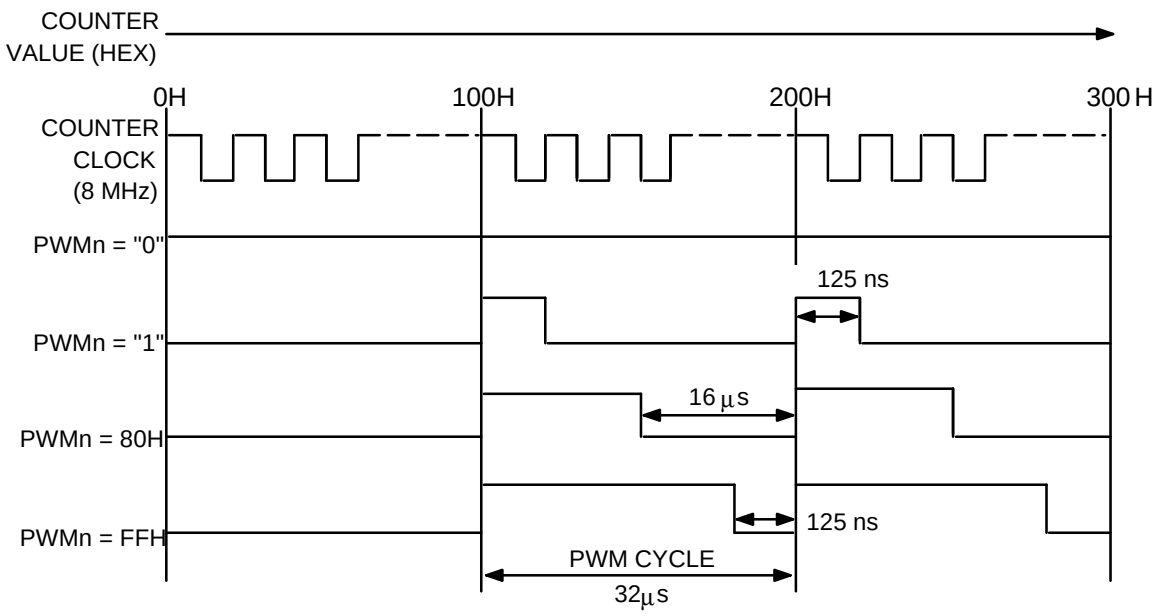
— The counter clock frequency is changed

— A new value is written to the PWM data register

STAGGERED PWM OUTPUTS

The PWM0–PWM6 outputs are staggered to reduce the overall noise level on the pulse width modulation circuits. If you load the same value to the PWM0–PWM6 data registers, a match condition (data register value is equal to the 8-bit count value) will occur on the same clock cycle for all the seven 8-bit PWM circuits.

For example, the PWM0 output is delayed by one-half of a counter clock, PWM1 output by one-half of a counter clock, PWM2 output by one-half of a counter clock, and so on for the subsequent clock cycles (see Figure 15-4).



NOTES

1. A counter clock value of 8 MHz is assumed for all timing values.
2. “n” = 0–6, for PWM0–PWM6.

Figure 15-3. PWM Waveforms for PWM0–PWM6

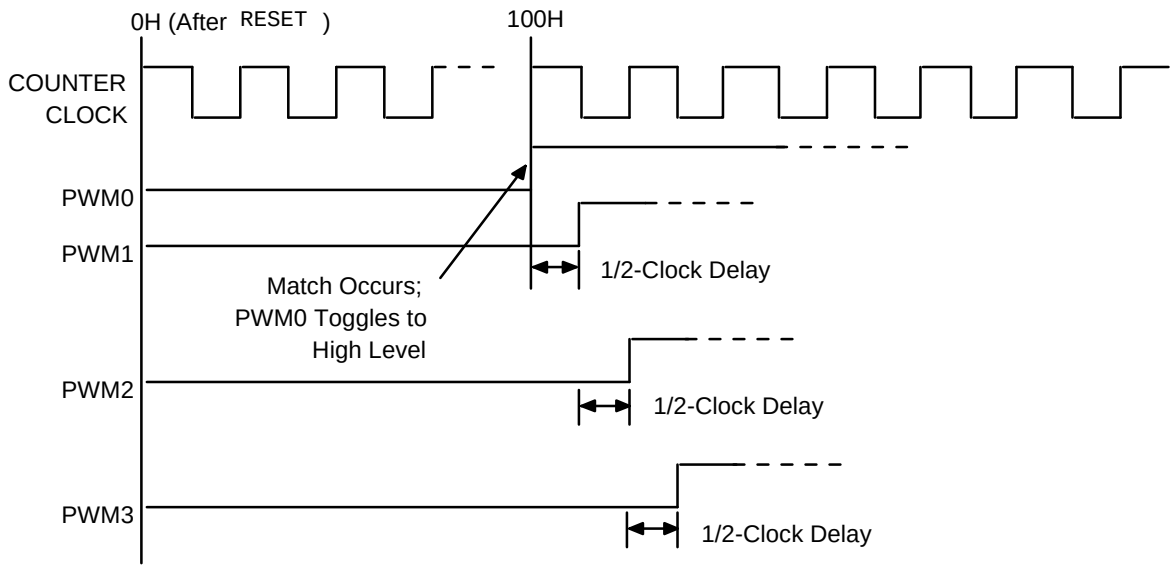


Figure 15-4. PWM Clock to PWM0–PWM6 Output Delays

PWM COUNTER

The PWM counter is an 8-bit incrementing counter. The same 8-bit counter is used by all PWM circuits. To determine the PWM module's base operating frequency, the counter is compared to the PWM data register value.

PWM DATA REGISTERS

A reset operation disables all PWM output. The current counter value is retained when the counter stops. When the counter starts, counting resumes from the retained value.

PWM CLOCK RATE

The timing of the 8-bit output channel is based on the maximum 12 MHz CPU clock frequency. The 2-bit prescaler value in the PWMCON register determines the frequency of the counter clock. You can set PWMCON.6 and PWMCON.7 to divide the CPU clock frequency by one (non-divided), two, three, or four.

As the maximum CPU clock rate for the KS88C6216/C6224/C6232 microcontrollers is 12 MHz, the maximum base PWM frequency is 46.8 kHz (12 MHz divided by 256). This assumes a non-divided CPU clock.

☞ PROGRAMMING TIP — Programming PWM0 to Sample Specifications

This sample program executes a test of the PWM block. The program parameters are as follows:

— The oscillation frequency of the main crystal is 8 MHz

— PWM frequency is 31.2 kHz

```
RESET:  DI
        SB0                          ;  Select bank 0
        •
        •
        •
        LD      P2CONH,#11111111B    ;  Select n-channel, open-drain PWM output
        LD      P2CONL,#11111111B    ;  Select push-pull PWM output
        SB1                          ;  Select bank 1
        OR      PWMCON,#00100000B    ;  PWMCON.5 ← 1; start the counter
                                     ;  PWM counter clock is fOSC
        SB0                          ;  Select bank 0
        EI
        •
        •
        •
PWMstart:
        SB1                          ;  Select bank 1
        LD      PWM0, #80H           ;  Load PWM0 data
        SB0
        RET
```

16 SYNC PROCESSOR

OVERVIEW

The KS88C6216/C6224/C6232 multi-sync signal processor (sync processor) is designed to process horizontal (Hsync) and vertical (Vsync) signals that are input to a multi-sync monitor. The sync processor can perform the following functions:

— Detect sync input signals (Vsync-I, Hsync-I, and Csync-I, also called Screen-On-Green, or SOG)

— Output a pseudo sync signal

— Detect the polarity of sync input signals

— Separate and output sync signals (Hsync-O, Vsync-O, and Clamp-O)

The sync processor circuits are controlled by three control registers: SYNCON0, SYNCON1, and SYNCON2.

Vsync Separation

SYNCON0 register setting controls the output path of the sync processor's 5-bit counter. Using the 5-bit counter, the sync processor can separate the Vsync signal from composite (H+V) sync signal.

The counter value increments when a High level sync signal is detected and decrements when a Low level signal is detected. No overflow or underflow can occur. That is, the 5-bit counter increments until it reaches the maximum value of 11111B and then stops or decrements until it reaches the minimum value of 00000B. You can select $f_{OSC}/2$ or $f_{OSC}/3$ as the counter's clock input source.

When SYNCON0.5 is "1", a High signal level is output to a multiplexer whenever the counter value reaches 11111B and a Low level is output when the counter value reaches 00000B. The signal level remains constant when the counter value is less than 11111B or greater than 00000B.

Clamp Signal Output

SYNCON1 register settings control Clamp signal output and pulse width. Clamp output can be completely inhibited, or it can be generated at two, four, or eight times f_{OSC}. You can specify the signal edge on which the selected Clamp pulse width is to be output ("front porch" or "back porch"). When SYNCON1.6–.5 is set to "00", the clamp signal output is inhibited. In this case the clamp signal level (Clamp-O) can be either "low" (when SYNCON1.3 is set to "1") or "high" (when SYNCON1.3 is "0").

Pseudo Sync Generator

SYNCON2 settings control the pseudo Vsync (66.7 Hz, 78.4 Hz, or 90.9 Hz) and pseudo Hsync (40.00 kHz, 50.0 kHz, or 66.67 kHz) output frequencies when f_{OSC} clock is 8 MHz and SYNCON2.5 is set to "0". The polarity of these frequencies is always positive, with duty cycles of 1% and 10%, respectively. The pseudo sync generator supports factory testing of the sync processor block and also protects a system against the effect of unexpected signals in transition period while mode changing.

Logic for Detecting Sync-On-Green (SOG)

Special logic in the sync processor block can compare Hsync and Csync input signals to detect Sync-On-Green (SOG).

Vsync Detection Interrupt

The sync processor can generate a Vsync detection interrupt (level IRQ5, vector ECH). This interrupt is triggered by a rising signal edge at the Vsync-I pin or Vsync signal from sync-processor. The interrupt pending condition is automatically cleared by hardware when the corresponding service routine is initiated.

Pseudo Sync Generation Formula

- Pseudo Vsync (Positive only, Duty 1%)

— SYNCON2.4–.3 = "01" (SCLKT = 4 MHz → SCLKT/10 → SCLK = 400.00 kHz) → SCLK/60 → /100 = 66.7 Hz

— SYNCON2.4–.3 = "10" (SCLKT = 4 MHz → SCLKT/10 → SCLK = 400.00 kHz) → SCLK/51 → /100 = 78.4 Hz

— SYNCON2.4–.3 = "11" (SCLKT = 4 MHz → SCLKT/10 → SCLK = 400.00 kHz) → SCLK/44 → /100 = 90.9 Hz

- Pseudo Hsync (Positive only, Duty 10%)

— SYNCON2.1–.0 = "01" (SCLKT = 4 MHz) → SCLK/10 → /10 = 40.00 kHz

(Available pseudo Vsync : 66.7Hz, 78.4Hz, 90.9Hz)

— SYNCON2.1–.0 = "10" (SCLKT = 4 MHz) → SCLK/8 → /10 = 50.00 kHz

(Available pseudo Vsync : 66.7Hz, 90.9Hz)

— SYNCON2.1–.0 = "11" (SCLKT = 4 MHz) → SCLK/6 → /10 = 66.67 kHz

(Available pseudo Vsync : 66.7Hz, 78.4Hz)

Hsync Polarity Detection, Unmixed Hsync Detection and Hsync Blanking

The polarity of Hsync signal input to Hsync-I pin is automatically detected. If the Hsync polarity is negative, SYNCON1.0 equals to "0". If the Hsync polarity is positive, SYNCON1.0 equals to "1". This polarity detection bit (SYNCON1.0) not be accurate when the sync level is not in a transitional condition.

In composite sync mode, if SYNCON2.7 is set to "1", the current period of checked Hsync is stable, unmixed with Vsync signal. If SYNCON2.7 is "0", the current period of checked Hsync is mixed with Vsync signal, in which case it is recommended not to calculate the sync frequency. In this mode, the Hsync signal is automatically blanked during the Vsync signal extraction period.

SYNC PROCESSOR CONTROL REGISTER 0 (SYNCON0)

The sync processor control register 0, SYNCON0, is located in set 1, bank 0, at address EDH. It is read/write addressable. SYNCON0 bits 4–0 hold the 5-bit counter value which is used for compare function. Whenever a High signal level is detected, the count value is incremented by one until it reaches the maximum value of "11111B" (No overflow occurs). Whenever a Low signal level is detected, the count value is decremented by one until it reaches the minimum value of "00000B" (No underflow occurs).

NOTE

Compare mode is also called Vsync separation mode. In this mode, output to the multiplexer is enabled. When the counter value is "11111B", the output is High level; when the counter value is "00000B", the output is Low level. Whenever the counter value is less than (<) "11111B", or greater than (>) "00000B", the previous output level is retained.

SYNCON0 settings also control the following sync processor functions:

— Horizontal or composite sync input (Hsync-I or Csync-I) selection

— Automatically enable Hsync blanking or Hsync signal bypass

— Vsync port input or compare mode

— Select the clock source for Vsync-O

See Figure 16-1 for a detailed description of SYNCON0 register settings.

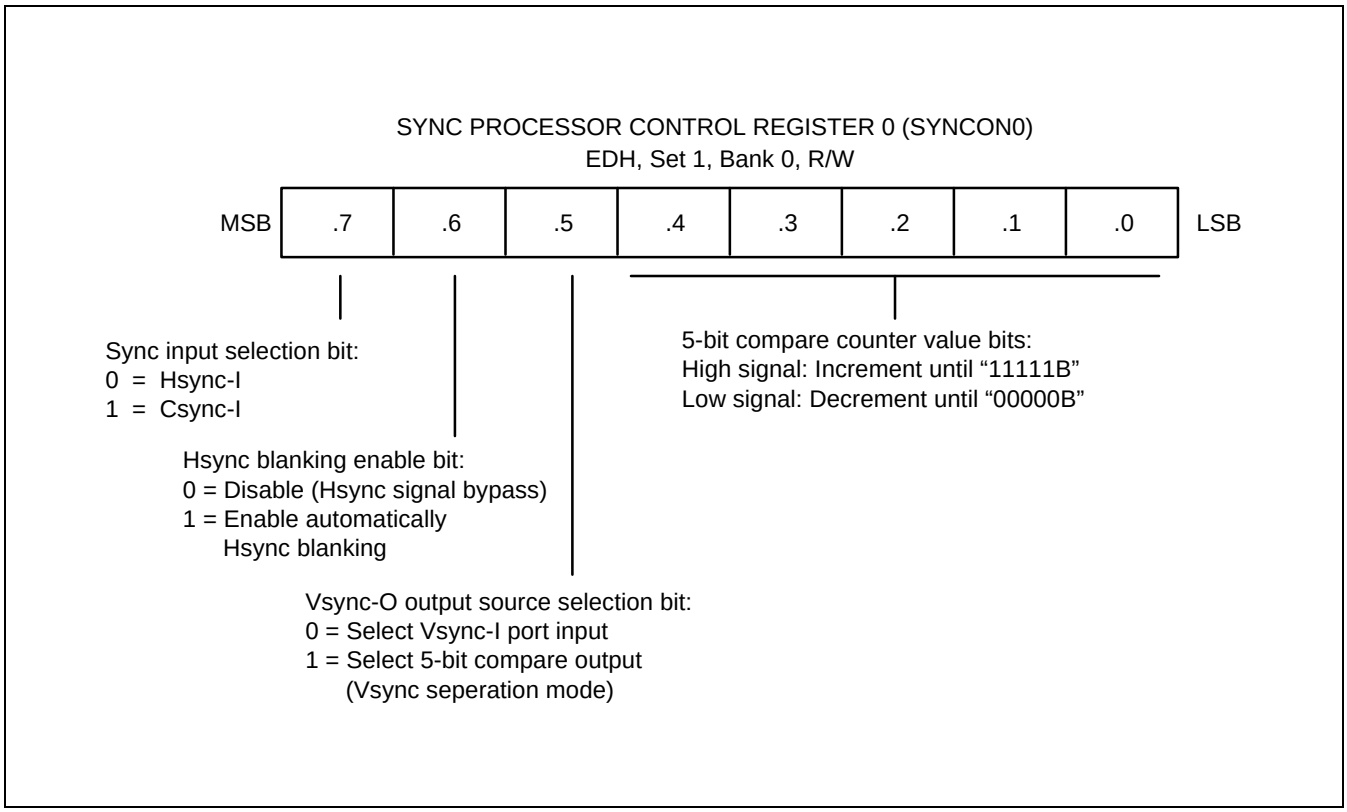


Figure 16-1. Sync Processor Control Register 0 (SYNCON0)

SYNC PROCESSOR CONTROL REGISTER 1 (SYNCON1)

The sync processor control register 1, SYNCON1, is located in set 1, bank 0, at address EEH. It is read/write addressable. Using SYNCON1 settings, you can:

— Compare Hsync and Csync input signals to detect Sync-On-Green (SOG) input

— Generate a clock pulse for Clamp signal output

— Select “front porch” or “back porch” mode for Clamp-O

— Control Clamp-O, Vsync-O, and Hsync-O status

— Detect Hsync polarity

See Figure 16-2 for a detailed description of SYNCON1 register settings.

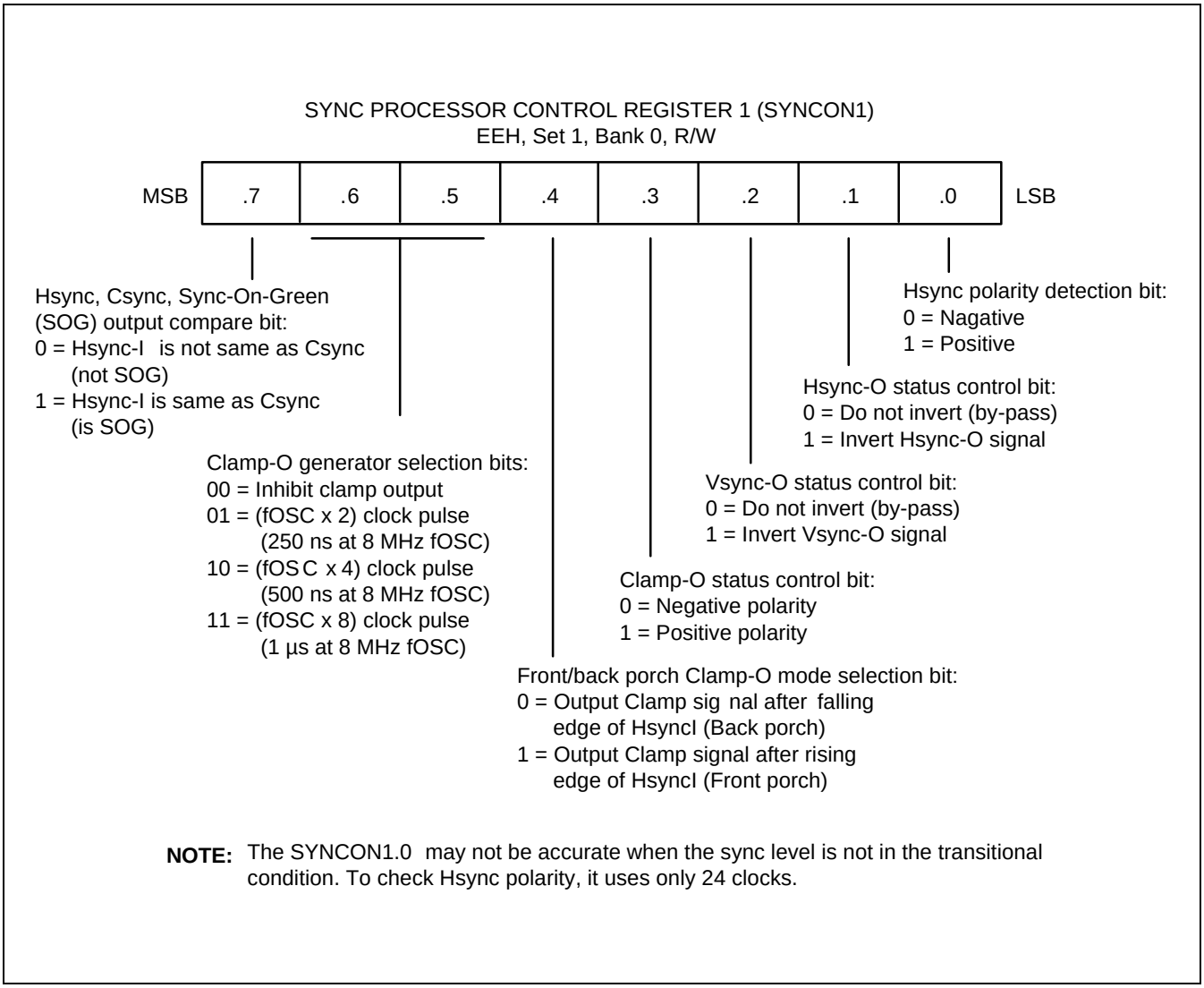


Figure 16-2. Sync Processor Control Register 1 (SYNCON1)

SYNC PROCESSOR CONTROL REGISTER 2 (SYNCON2)

The sync processor control register 2, SYNCON2, is located in set 1, bank 0, at address EFH. It is read/write addressable. Using SYNCON2 settings, you can:

— Detect mixed and unmixed Hsync period in composite sync

— Select the pseudo sync processor's operating mode (normal operation or factory test mode)

— Select the clock source for the 5-bit counter

— Control output of the pseudo Vsync/Hsync generator

— Enable and disable the Vsync detection interrupt

See Figure 16-3 for a detailed description of SYNCON2 register settings.

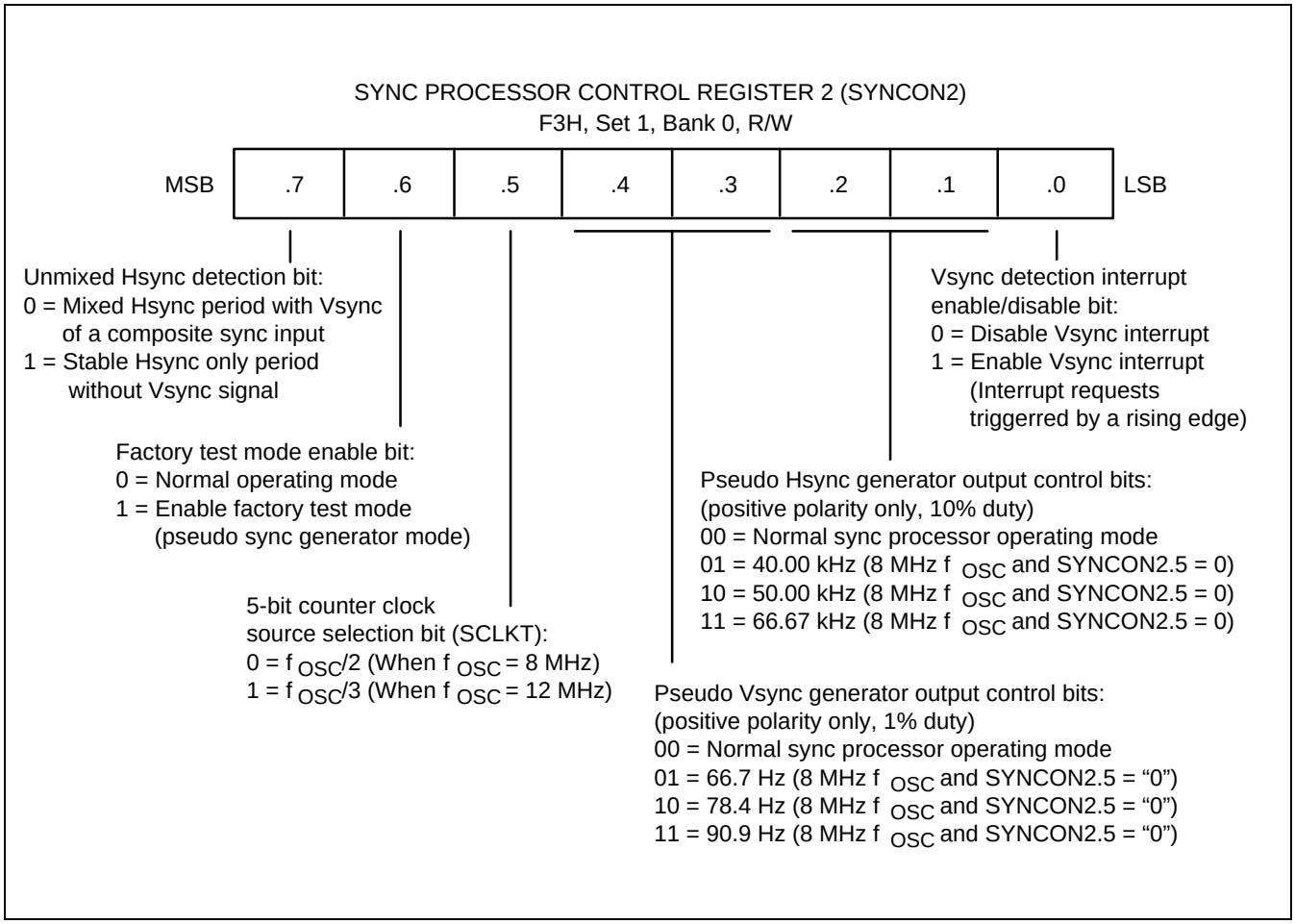


Figure 16-3. Sync Processor Control Register 2 (SYNCON2)

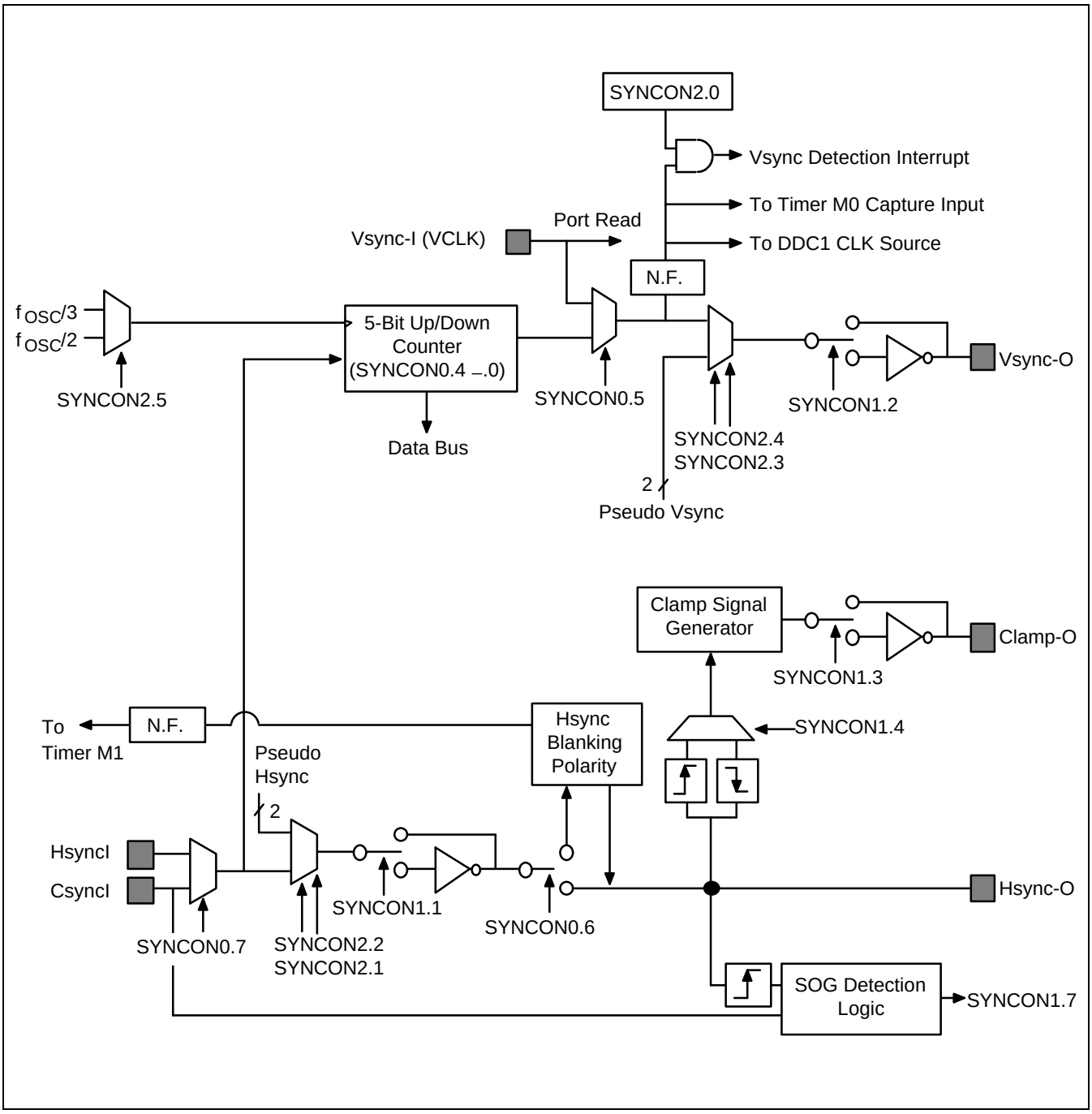


Figure 16-4. Sync Processor Block Diagram

DETECTING SYNC SIGNAL INPUT

You can detect the presence of a sync signal in two ways — directly or indirectly. The direct detection method can be implemented in read port. The indirect detection method is interrupt-driven and uses the KS88C6216/C6224/C6232 sync processor hardware. These methods are explained in detail below.

Direct Detection Method

By reading the input status directly on the sync input pins Vsync-I, Hsync-I, Csync-I, you can detect the presence of the incoming sync for a corresponding output.

To enable direct sync input detection, you set SYNCON0.5 to “0” (for Vsync-I), SYNCON0.7 to “0” (for Hsync-I), and SYNCON0.7 to “1” (for Csync-I).

You then read the state of the input pin(s) over a period of time to detect transitions in the signal level(s). If a transition is detected, it can be assumed that a sync signal is present.

Indirect Detection Method

To indirectly detect vertical sync input at the Vsync-I pin, you use register settings to assign either the timer M0 capture interrupt or the Vsync detection interrupt to this pin.

For indirect detection of horizontal or composite input at the Hsync-I or Csync-I pin, you use the timer 1 input clock source to generate a timer M1 capture/overflow interrupt by capture signal from timer M2 interval or Vsync-O from sync-processor, when a signal level transition occurs.

When the correct settings have been made, the application software polls for the respective interrupts to determine the presence of sync input signals, as follows:

— Indirect Vsync input detection

  Check for the occurrence of a timer M0 capture interrupt (IRQ0) or a Vsync detection interrupt (IRQ5).

— Indirect Hsync input detection

  Check for the occurrence of a timer M1 capture/overflow interrupt (IRQ1).

— Indirect Csync input detection (SOG)

  Check for the occurrence of a timer M1 capture/overflow interrupt (IRQ1).

DETECTING SYNC SIGNAL POLARITY

The KS88C6216/C6224/C6232 sync processor lets you detect the polarity of Vsync or Hsync signals in two ways: 1) using a software-intensive direct detection method, or 2) by an indirect detection method which uses the sync processor hardware.

Polarity detection is not required for Csync because the signal which passes through the Csync-I pin (SOG) is always positive polarity.

Direct Method for Polarity Detection

To use the direct polarity detection method, you read the status of the vertical or horizontal sync signals at the Vsync-I and Hsync-I pins, respectively. You then count the number of signal transitions that occur over a given number of frames, or during a given time interval, and derive an average value. If the average value is “1”, the signal’s polarity is negative. If it is “0”, the polarity is positive.

Another way to determine the polarity is to repeat a counting operation several times over a given time interval. You then compare the totals derived from the series of count operations to determine which one is greater. If the larger count value is for “1s”, the signal polarity is negative. If more “0s” were counted, the polarity is positive.

Indirect Method for Vsync Polarity Detection

To indirectly detect positive or negative Vsync polarity, you use timer M0 to capture the internal time between the rising and falling edges of the Vsync input signal.

To enable indirect Vsync polarity detection, you set TM0CON.0 to “1”. This selects the Vsync output path as input for the timer M0 capture function. Timer M0 must also be set to operate in capture mode.

When the series of capture operations has been completed, you can read the timer M0 counter value to determine the signal’s polarity.

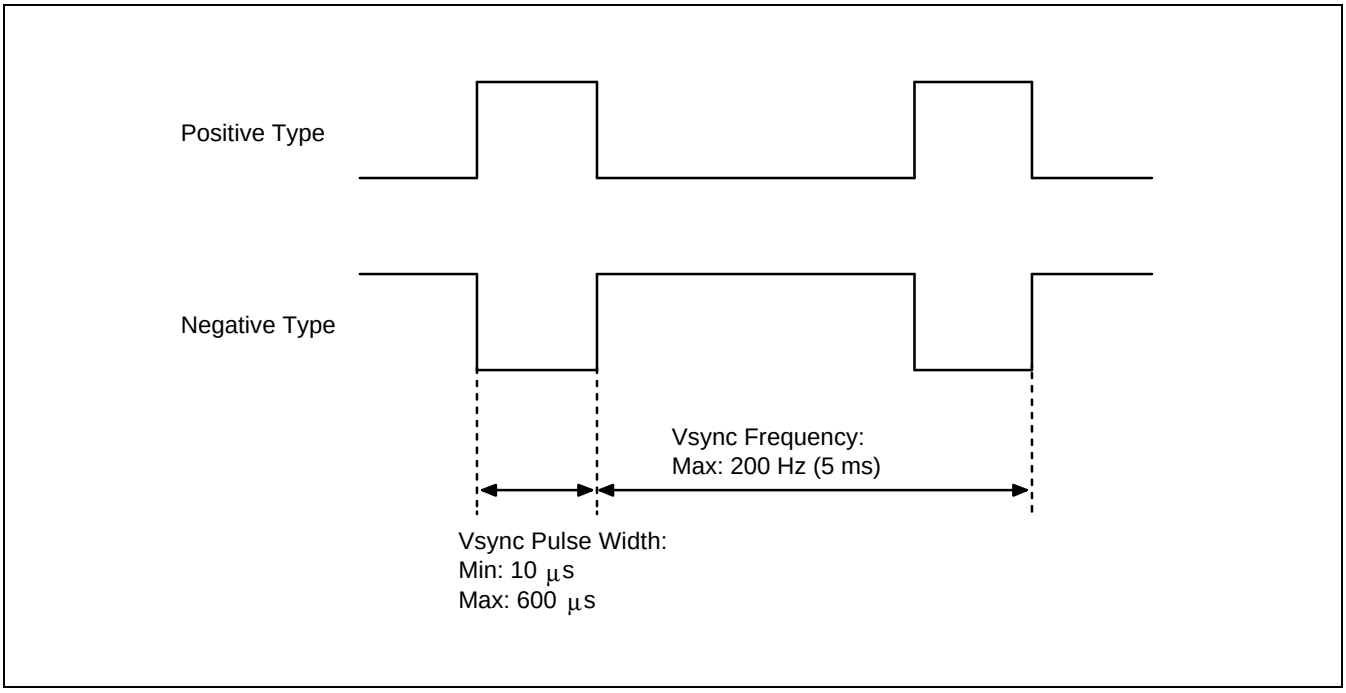


Figure 16-5. Vsync Input Timing Diagram

Indirect Hsync Polarity Detection (Auto Detection)

The polarity of Hsync signal input to Hsync-I is automatically detected by the 5-bit counter of sync-processor. If the negative polarity of Hsync is detected, SYNCON1.0 is "0". If the positive polarity of Hsync is detected, SYNCON1.0 is "1".

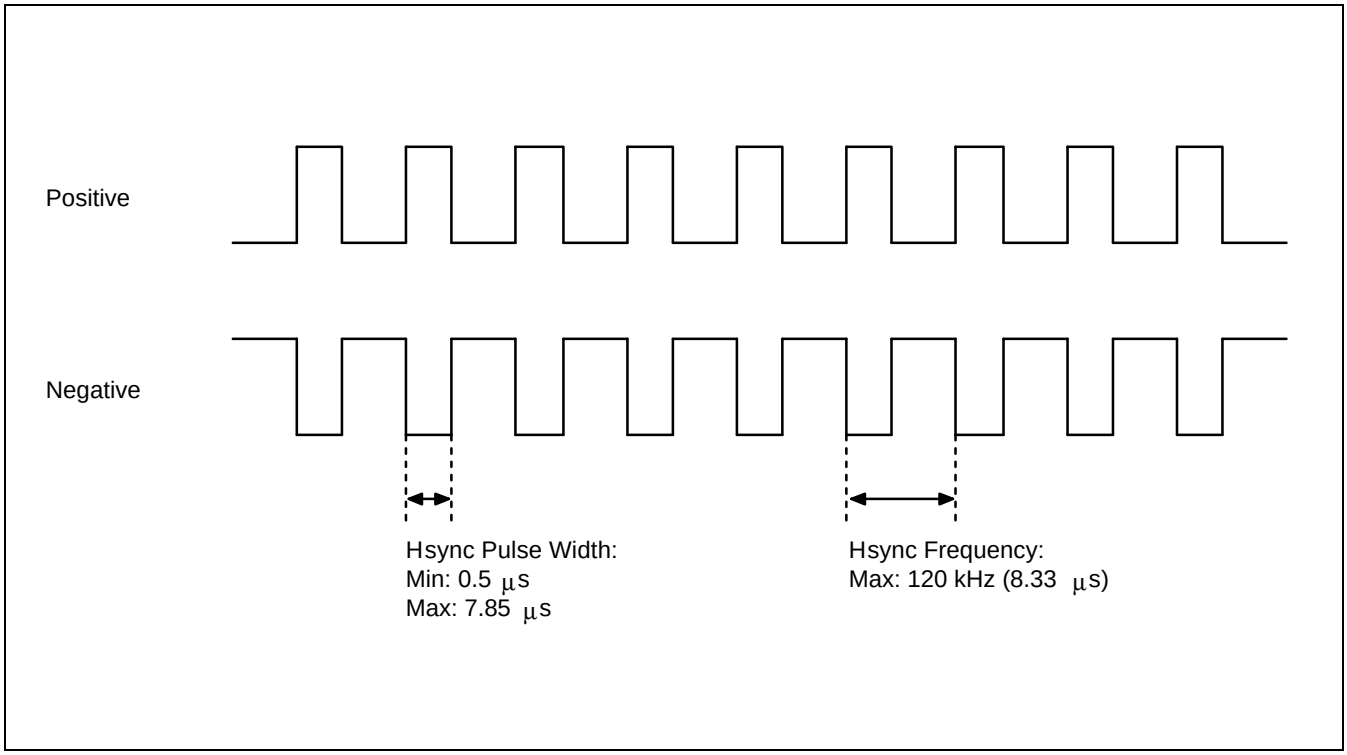


Figure 16-6. Hsync Input Timing Diagram

EXTRACTING VSYNC OUTPUT

When the Vsync input at Hsync-I or Csync-I (P2.7) also contains Hsync signals, you must extract the Vsync component from the Hsync (or Csync) input. To do this, you use the 5-bit up/down counter.

To extract the Vsync component of the input signal, you first set the 5-bit up/down counter to operate in compare mode (SYNCON0.5 = “1”). Vsync output is enabled only when the minimum or maximum threshold value is reached.

During vertical blanking, the counter decreases until it reaches a minimum value while the Hsync-I or Csync-I signal level is negative. Or, the counter value increases until it reaches a maximum value while the Hsync-I or Csync-I signal level is positive (no overflow occurs).

The Vsync detection interrupt (IRQ5) can be enabled to verify that the Vsync signal has been extracted successfully from the mixed input signal.

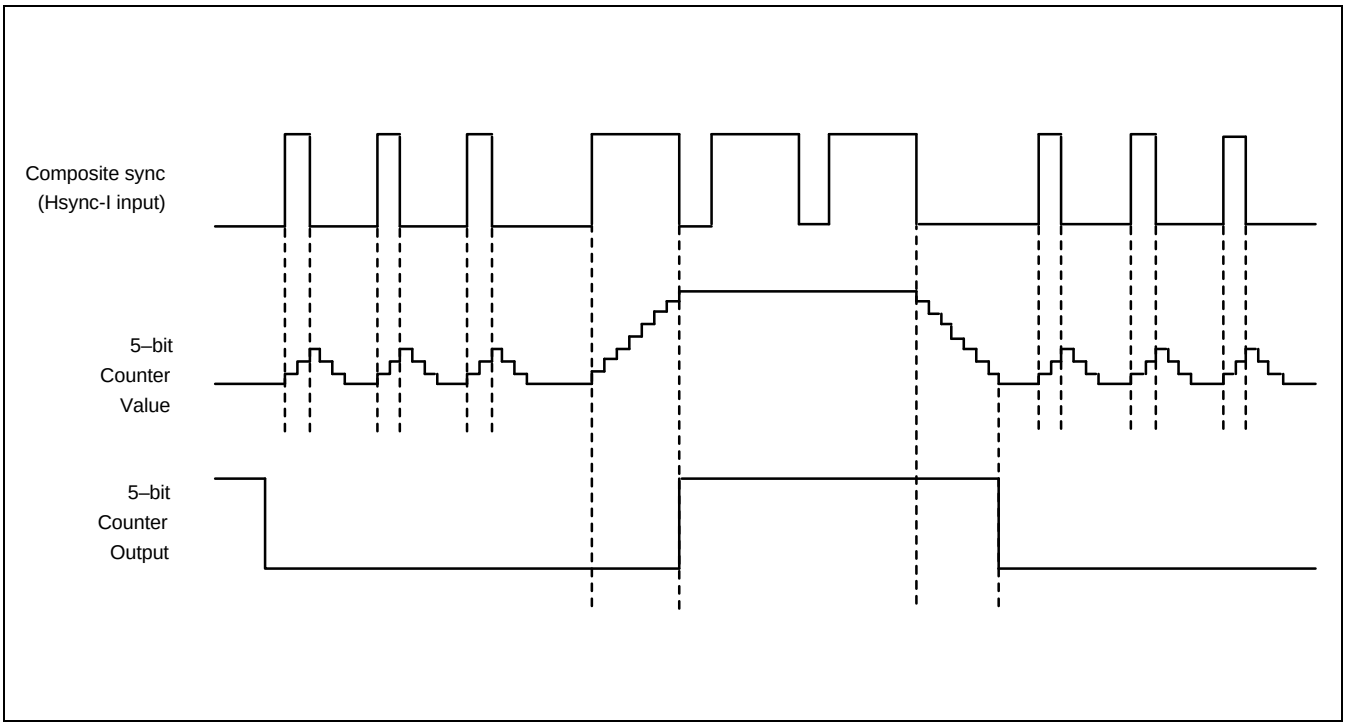


Figure 16-7. Vsync Extraction Using an Up/Down Counter

CLAMP SIGNAL OUTPUT

Clamp signal output (Clamp-O) must be synchronized with Hsync output. The Clamp-O signal can be transmitted to a vertically or horizontally driven integrated circuit to provide a pedestal level for image signals with programmable pulse width.

The Clamp signal is output on the “front porch” of an Hsync signal (NO SOG condition) or on the “back porch” of an Csync signal (SOG condition).

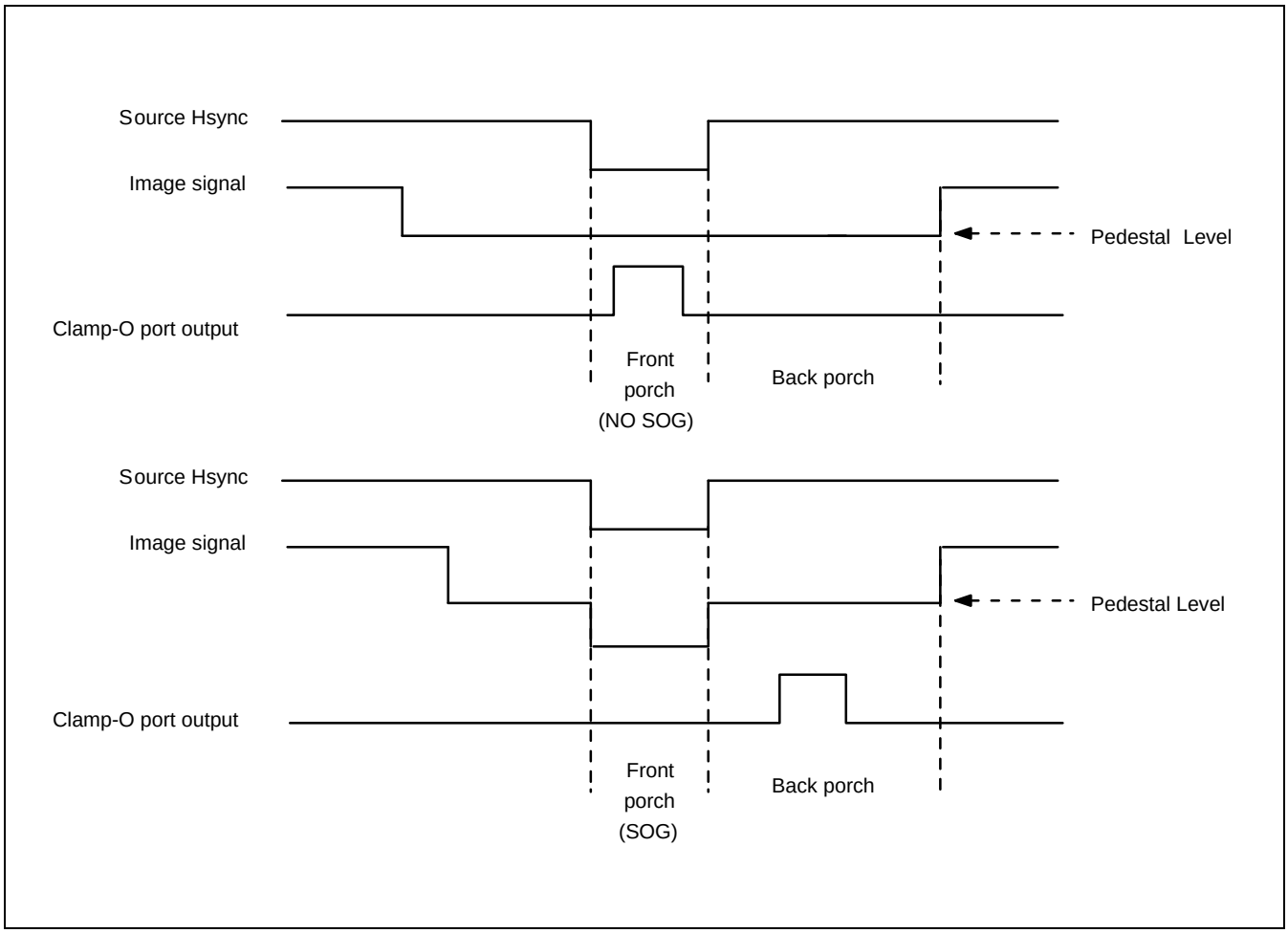


Figure 16-8. Clamp-O Signal (SOG and NO SOG Condition)

DIFFERENTIATING SOG FROM NO SOG

The pulse width at the Csync-I pin is different in SOG and NO SOG conditions. In a SOG condition, the pulse width at Csync-I and Hsync-I is identical. If a NO SOG condition exists, Csync-I has a wider pulse width than Hsync-I because the Csync-I pulse is truncated at the base of the pedestal level (see Figure 16-10).

To differentiate the Csync pulse, you must delay the Csync-I pulse for about 150 ns and then compare its phase with that of the Hsync-I pin signal.

To indicate a SOG condition, comparator logic for Hsync and Csync sets the SYNCON1.7 flag to “1” whenever Csync status differs from Hsync status more than 32 times at the rising edge of Hsync-I. To perform the comparison, first detect the polarity of the Hsync-I signal. Then configure the pin for positive output. (Csync-I is always positive and requires no special settings.) To recognize the SOG condition, you can poll the SYNCON1.7 status flag to detect when it is set to “1”.

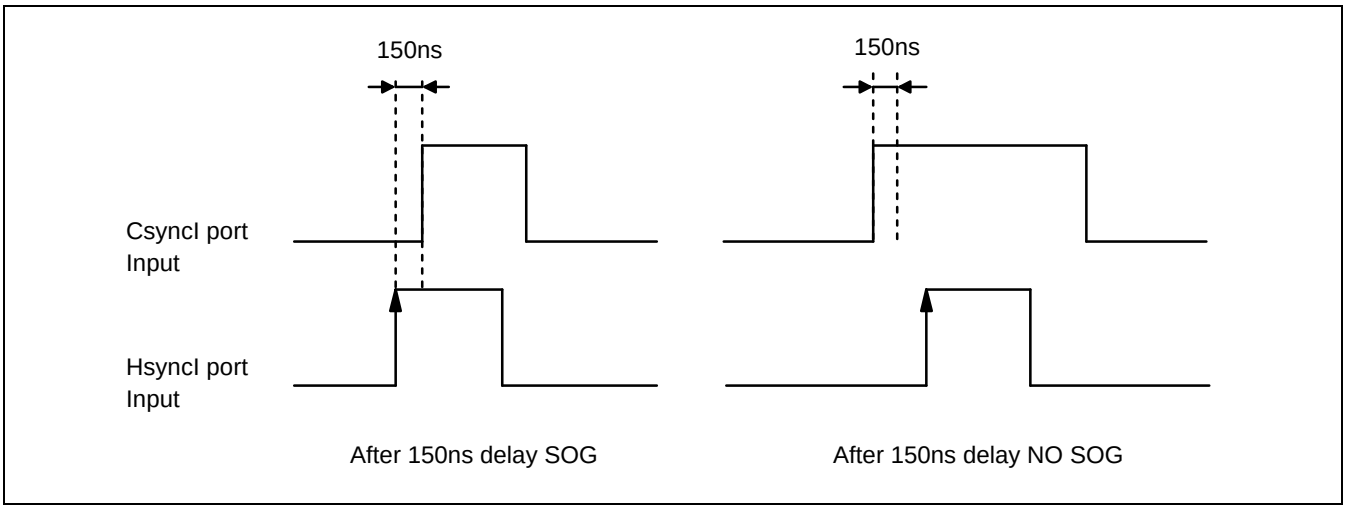


Figure 16-9. Sync Input at the Hsync-I and Csync-I Pins

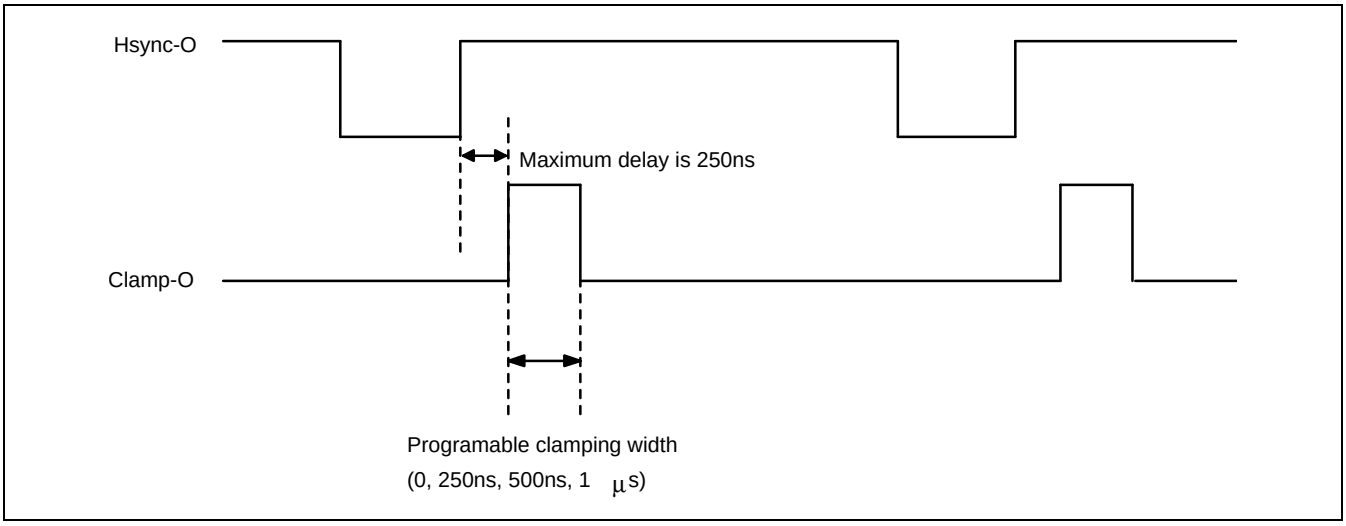


Figure 16-10. Clamp-O Signal Delay Timing

☞ PROGRAMMING TIP — Programming the Sync Processor

This example shows how to program the sync processor to sample specifications. The sample program performs the following actions:

— Confirm the presence of sync signal input

— Detect the polarity of Hsync or Vsync signal input

```
;//*****************************************************************
;//** Title : Definition Flag Ram for user (00h - 0Fh)        *
;//**                                                        *
;//**                                                        *
;//*****************************************************************
;
DeGausport      equ     3       ;P0.3=Degauss control pin(Active High)
Suspndport      equ     4       ;P0.4=Suspend control pin(High)
Offport         equ     5       ;P0.5=Off control pin(Low)
Muteport        equ     7       ;P0.7=Video mute control pin(Low)
;
SelfRastport    equ     0       ;P1.0=Self-Raster input pin
S1              equ     1       ;P1.1=S-correction 1
S2              equ     2       ;P1.2=S-correction 1
;
;X-Ray          equ     0       ;Not used
;Rotation       equ     1       ;P2.1=Pwm1 Out (Rotation)
;H-Size         equ     2       ;P2.2=Pwm2 Out (H-Size)
;Contrast       equ     3       ;Not used
;Brightness     equ     4       ;Not used
;ACL            equ     5       ;P2.5=Pwm5 Out (ACL)
;Hsize_Min      equ     6       ;Not used
ModelSelport    equ     7       ;P2.7=Model Sel.input pin(14":L/15":H)
;
```

```
Pwrkeyport      equ     0       ;P3.0=S/W power key input pin
Ledport         equ     5       ;P3.5=LED control
SCL             equ     6       ;P3.6=SCL(S/W IIC.bus)
SDA             equ     7       ;P3.7=SDA
;
;Fixed port.
;H_Input        equ     1       ;H-Sync. input
;V_Input        equ     2       ;V-Sync. input
;Clamp          equ     3       ;Clamp Output
;H_Out          equ     4       ;H-Sync. Output
;V_Out          equ     5       ;V-Sync. Output
;DDC_Clock      equ     6       ;DDC Clock
;DDC_Data       equ     7       ;DDC Data
;HsyncIport     equ     0
VsyncIport      equ     1       ;SYNCRD.1=VsyncI pin
VsyncOport      equ     3       ;SYNCRD.3=VsyncO pin
;
;//  Define flags
;
SyncP_FGR0      EQU     00h     ;SYNC-PROCESSOR
HSyncFin_FG   equ   7           ;Hsync signal counting every 10ms
VSyncFin_FG     equ     6       ;Vsync signal find flag(Vsync capture interrupt)
VstblFreq_FG    equ     5       ;Stable Vsync frequency input status
HstblFreq_FG    equ     4       ;  "      Hsync
NormSync_FG     equ     3       ;Normal sync output mode(No pseudo sync signal)
SetSepSync_FG   equ     2       ;Indicate separate sync mode
MixedSync_FG    equ     1       ;mixed sync input period(composite sync input mode)
DdcHighSpd_FG   equ     0       ;DDC1 high speed mode(over 400Hz)
;
```

```
SyncP_FGR1          EQU     01h
HPolarity_FG   equ  7       ;Hsync polarity => 1=positive, 0=negative
VPolarity_FG   equ  6       ;Vsync polarity => 1:positive
HNosync_FG          equ     5       ;Hsync freq. < 10 kHz
VNosync_FG          equ     4       ;Vsync freq. < 40 Hz
NoSync_FG           equ     3       ;No Vsync & No Hsync signal
OverHsync_FG        equ     2       ;Hsync over range : over 62 kHz
OverRange_FG        equ     1       ;Vsync over range : over 135 Hz
;
Mute_FGR            EQU     02h
Vmute_FG            equ     7       ;Being video mute
ChkSyncStus_FG      equ     6       ;Video mute time end
MuteWaiting_FG      equ     5       ;Being video mute extension
MuteRelse_FG        equ     4       ;Video mute release
PwrOnWait_FG        equ     3       ;Power-on mute delay (2sec)
PsyncOut_FG         equ     2       ;Pseudo sync output status
NormMwait_FG        equ     1       ;Count mute extension time (350ms)
;
Time_FGR            EQU     03h
KeyDetect_FG        equ     7       ;Key detecting per 10ms
DeGTime_FG          equ     6       ;Degaussing time (3sec)
ChkPwrKey_FGequ     5       ;Checking power-key status per 10ms
;
Status_FGR          EQU     04h
SfRasterIn_FG       equ     7       ;Self-Raster mode
Recall_FG           equ     6       ;Recall function (Continuous key=3sec)
UserDel_FG          equ     5       ;Delete user data in EEPROM(Continuous key=5sec)
FindSyncSrc_FG      equ     4
;
```

```
EepRom_FGR      EQU     05h
UserArea_FG     equ     7       ;Checking EEPROM user data area
ClosHsync_FG    equ     6       ;Searching closest Hsync mode
SavedEep_FG     equ     5       ;Factory data saved EEPROM ?
EepDataRd_FG    equ     4       ;EEPROM data read after mode changing
NoFactSave_FG   equ     3       ;Factory Timming doesn,t exist
;
Tda9109_FGR     EQU     05h
TdaWrite_FG     equ     2       ;Write PWM data to TDA9109
TdaRead_FG      equ     1       ;Read from TDA9109
;
Dpms_FGR        EQU     06h
DpmsStart_FG    equ     7       ;DPMS mode start(3sec after abnormal sync signal)
ChkDpmsCon_FG   equ     6       ;DPMS condition input(H<10KHz,V<40Hz)
PwrOffIn_FG     equ     5       ;Off mode
OffMode_FG      equ     4
Suspend_FG      equ     3       ;Suspend mode
StandBy_FG      equ     2       ;Standby mode
;
IIC_FGR         EQU     07h
DDCCmd_FG       equ     7       ;Get DDC2B+ command
LastByte_FG     equ     6       ;Last Tx Data
CommFail_FG     equ     5       ;Communication fail(Not ACK)
ReWrite_FG      equ     4       ;Rewrite to EEPROM
ReRead_FG       equ     4
Ddc2mode_FG     equ     3
RevA0match_FG   equ     2
Ddc2BTxmode_FG  equ     1
;
```

```
CheckSum        EQU     0Bh     ;DDC2B+
ByteCnt         EQU     0Ch     ;Count number of Tx data
SendType        EQU     0Dh     ;Reply type
NumTxdByte      EQU     0Eh     ;#(Total Tx byte -1) in master Tx mode
RxXCntr         EQU     0Fh     ;Number of Rx data

;//*****************************************************************
;//** Title : Definition general purpose Ram (10h - BFh)    *
;//**                                                       *
;//**                                                       *
;//******************************************************************
;
TB1mSR          EQU     10h     ;1ms interval register(basic time reg.)
TB10mSR         EQU     11h     ;10ms(Count Hsync event signal)
TB100mSR        EQU     12h     ;100ms(Func valid time => 7sec)
DLY1mSR         EQU     13h     ;1ms(Cehcking writecycle time =>10ms)
M10mSR          EQU     14h     ;10ms(Checking mute time => 2sec,350ms)
S100mSR         EQU     15h     ;100ms(Saving start time => 2sec)
DG100mSR        EQU     16h     ;100ms(Degaussing time => 3sec)
DPMS100mSR      EQU     17h     ;Check DPMS start(after No Sync : 3sec)
ChkSRasTime     EQU     18h     ;Check self-raster input(maintain 70ms ?)
;
HCount          EQU     20h     ;Double byte(even address + odd address)
;               EQU     21h
HFreqStCnt      EQU     22h
HFreqSpCnt      EQU     23h
AverageHf       EQU     24h
HfHighNew       EQU     25h     ;Current value of Hsync freq high byte
HfLowNew        EQU     26h
```

```
HfHighData      EQU     27h     ;Saved value of Hsync freq high byte

HfLowData       EQU     28h     ;Real Hsync frequency
                                ; = Low nibble of Hfreq high data+ Hfreq low data
                                ;ex> if, HfHighData=#x2h, HfLowData=#58h
                                ; => Hsync frequency = 258h=60.0Khz

Vcount          EQU     30h     ;total number of timer0 counter within Vsync period

;               EQU     31h     ;Double Byte

NoVTime         EQU     32h     ;Checking the sustaining time of no Vsync signal
                        ;if NoVTime > 30ms(under 33Hz) => Mute

VFreqCHigh      EQU     33h     ;Temp storage of Timer0(Vsync) overflow count

T0OvfCntr       EQU     34h     ;No vsync-int. service if DDC1 high freq. mode

VfCurrNew       EQU     35h     ;Current Vsync freq.

VFreqData       EQU     36h     ;Saved Vsync freq.

VclkCntr        EQU     37h     ;for auto recovery of DDC mode (DDC2B -> DDC1)

;

PolaCntr        EQU     38h     ;Count number of polarity checking

VPolaCntr       EQU     39h     ;Increment when positive polarity

;

UmodeNo         EQU     40h     ;matched number of user mode

FmodeNo         EQU     41h     ;factory mode

ReadData        EQU     42h     ;Readed Data from EEPROM

;

EP_BPlus        EQU     50h     ;EEPROM & RAM data

EP_CONTRAST     EQU     51h     ;KA2504

EP_RGain        EQU     52h

EP_GGain        EQU     53h

EP_BGain        EQU     54h

EP_CoffBRIGHT   EQU     55h     ;KA2504

EP_RCutoff      EQU     56h

EP_GCutoff      EQU     57h

EP_BCutoff      EQU     58h
```

```
EP_ACL          EQU     59h     ;PWM5
EdidAddr        EQU     80h     ;Page1 RAM register
                                ;EDID address(00~7Fh:128-byte)


;//          WORKING REGISTERS -> GENERDAL RAM
;
;R14            EQU     EepSubAddr          ;Sub address of EEPROM/TDA9109
;R15            EQU     EepWrData           ;Data to write in EEPROM/TDA9109
;---> KA2504 Pre-amp control
;R14            EQU     PreAmpSubAddr       ;Sub address
;R15            EQU     PreAmpCtrlData      ;Data address
;
;//          Buffer for DDC2B+ protocol
;
MBusBuff        EQU     0B0h    ;for DDC2B+(00h-0BFh:16-byte)
AbusDstAddr     EQU     0B0h
AbusSrcAddr     EQU     0B1h
AbusPLength     EQU     0B2h
AbusCommand     EQU     0B3h
;   :           :
;               EQU     0BFh
;
;#####################################################
;
;//          DDC EDID AREA : 00h - 7Fh (128-Byte : Page 1)
;
;#####################################################
;//*************************************************************
;//** Title : Define Control register's flags
;//**
```

```
;//**
;//*************************************************************
;// IIC.bus Control Register
;// DCCR(DDC Clock Control Reg)
DTXACKEN        equ     7
DCLKSEL         equ     6
DINTEN          equ     5
DPND            equ     4
CCR3            equ     3
CCR2            equ     2
CCR1            equ     1
CCR0            equ     0
;
;// DCSR0(DDC Control/Status Reg0)
DMTX            equ     7
DSTX            equ     6
DBB             equ     5
DDCEN           equ     4
DAL             equ     3
DADDMAT         equ     2
DADD0           equ     1
DRXACK          equ     0
;
;// DCSR1(DDC Control/Status Reg1)
MISPLS          equ     2       ;Misplaced condition error
DBUFEMT         equ     1       ;Data buffer empty status(0:Write to TBDR, 1:TBDR -> DDSR)
DBUFFUL         equ     0       ;Data buffer full status(0:Read from RBDR, 1:DDSR -> RDBR)
;
;TBDR                            ;Transmit pre-buffer data register
```

```
;RBDR                                   ;Receive pre-buffer data register
;DDSR                                   ;DDC data shift register
;
;// Sync-processor control register
;// SYNCON0
SIS             equ     7               ;Sync input selection(0:Hsync, 1:Csync)
HBLKEN          equ     6               ;Hsync Blanking Enable(0:Bypass, 1:Blanking)
VOSS            equ     5               ;VsyncO source selection(0:VsyncI, 1:5-bit compare out)
;
;// SYNCON1
SOGI            equ     7               ;SOG check
CLMP1           equ     6
CLMP0           equ     5               ;ClampO pulse width
FBPS            equ     4               ;Front/Back porch selection(0:back, 1:front)
CLMPS           equ     3               ;ClampO polarity control(0:nega, 1:posi)
VOS             equ     2               ;VsyncO polarity control(0:by-pass, 1:invert)
HOS             equ     1               ;HsyncO polarity control(0:by-pass, 1:invert)
HPOL            equ     0               ;Hsync polarity detection(0:negative, 1:positive) , read only
;// SYNCON2
UNMIXHSYNC      equ     7               ;Unmixed Hsync detection(0:Mixed sync, 1:Unmixed sync)
CCSS            equ     5               ;5-bit counter clock selection(0:fosc/2, 1:fosc/3)
PVSO1           equ     4               ;Pseude VsyncO selection
PVSO0           equ     3
PHSO1           equ     2               ;Pseude VsyncO selection
PHSO0           equ     1
VINTEN          equ     0               ;Vsync detection interrupt enable(0:disable, 1:enable)
;// DCON
BUFEN           equ     3               ;Tx/Rx pre-buffer data register enable
                                        ;(0:Normal 1:Pre-buffer mode)
DDC1MAT         equ     2               ;DDC address match(0:Not match, 1:match)
```

```
DDC1EN          equ     1       ;DDC1 Tx mode enable
SCLF            equ     0       ;SCL falling edge detection
;
;// Watch-dog(Basic) Timer
BTCLR           equ     1
;
;// Timer0
T0CLR           equ     3       ;Counter clear
T0OVINT         equ     2       ;Overflow interrupt enable
T0INT           equ     1       ;Capture enable
T0CAPSEL        equ     0       ;Capture input selection
                                ; (0:External pin, 1:Vsync from sync-processor)
;// Timer1
T1CAPSEL        equ     7       ;Capture signal source selection(0:Timer2, 1:VsyncO from SP)
VEDGSEL         equ     6       ;VsyncO capture edge selection(0:rising, 1:falling)
T1CAPEN         equ     5       ;Capture enable
T1CLR           equ     4       ;Counter clear
T1INT           equ     3
T1PND           equ     2
;// Timer2
T2INT           equ     2
;
;// Pre-Amp Sub-address Mapping          ;Slave address=DCh

PSubA_Cont      EQU     00h     ;Contrast control
PSubA_SBnBr     EQU     01h     ;bit7=Soft Blanking(1:ON, 0:OFF)
                                ;Bit6-5=Cut-off control offset current switch
                                ;Bit4-0=Brightness control
PSubA_RGain     EQU     02h
PSubA_GGain     EQU     03h
```

```
PSubA_BGain     EQU     04h
PSubA_CoBr      EQU     05h     ;Cut-off brightness control
PSubA_RCo       EQU     06h     ;R Cut-off control
PSubA_GCo       EQU     07h
PSubA_BCo       EQU     08h
PSubA_Sw        EQU     0Ah

;// EEPROM Address Mapping
;
EPA_CoBr        EQU     0F6h    ;KA2504 Cut-off Brightness
EPA_Cont        EQU     0F7h    ;KA2504 Cut-off Contrast
EPA_RGain       EQU     0F8h    ;KA2504 R-Gain
EPA_GGain       EQU     0F9h
EPA_BGain       EQU     0FAh
EPA_RCo         EQU     0FBh    ;KA2504 R-Cut off
EPA_GCo         EQU     0FCh
EPA_BCo         EQU     0FDh
EPA_ACL         EQU     0FEh


;-------------------------------------------------------------
;--- DDC2ab comunication command code  ----
;-------------------------------------------------------------
I_Reset         equ     0F0h
I_IdReq         equ     0F1h
I_AsgnAdr       equ     0F2h
I_CapReq        equ     0F3h
I_ApplRprt      equ     0F5h
;
```

```
I_Attention         equ     0E0h
I_IdReply           equ     0E1h
I_CapReply          equ     0E3h
;I_GetVCP           equ     01h
I_VCPFReply         equ     02h
I_SetVCP            equ     03h
I_GetTiming         equ     07h
I_ResetVCPF         equ     09h
I_DisableVCPF       equ     0Ah
I_EnableVCPF        equ     0Bh
SaveCurrSet         equ     58h
I_TimingReport      equ     4Eh
I_GetEdid           equ     54h
SaveEdid            equ     69h
DelUser             equ     50h
AllModeSve          equ     52h
SaveColorSet        equ     7Dh
BrtContMax          equ     0D2h


;//*************************************************************
;//**          Title : Interrupt Vector Table          **
;//*************************************************************
;
            org     00EEh
            vector  T2Intv_Int          ;Timer2 interval interrupt(IRQ1)
;
            org     00F6h
            vector  T1Cap_Int           ;Timer1 capture interrupt(IRQ1)
```

```
                vector  DDCnFA_Int      ;Multi-master IIC.bus Tx/Rx int(IRQ2)
                vector  T0Ovf_Int       ;Timer0 overflow int.
                vector  VSyncDet_Int    ;Timer0 capture int(IRQ0)
;
                ORG     0100h
;
;//*************************************************************
;//** Title : Main Program start from here
;//**
;//**
;//** Inputs:
;//** Outputs:
;//** Preserves:
;//** Corrupts:
;//*************************************************************
;//***** ********************************************************
;//** Title : << System Reg.Files Initialization >>
;//**
;//**
;//*************************************************************

RESET:          DI                      ;Disable interrupt
                CLR     PP              ;Source,Destination = page0
                CLR     SYM             ;Disable fast interrupt
                LD      SPL,#0FFh       ;Stack pointer
                SRP     #0C0h           ;Working reg.area
                 LD     IMR,#00000111B
                                                ;bit7 -> not used
                                        ;bit6 -> not used
                                ;bit5 -> IRQ5 Vsync Detection Interrupt
                                ;bit4 -> IRQ4 external int P0.1/2
```

```
                                        ;bit3 -> IRQ3 external int P0.0
                                        ;bit2 -> IRQ2 iic.
                        ;bit1 -> IRQ1 Timer2 & Timer1 ovf.
            ;bit0 -> IRQ0 Timer0

            SB0

            CLR     EMT                         ;0 wait, Internal stack area

            LD      IPR,#00010001B              ;Group priority(int) undefined(A>B>C)
                                                ;T2/T1 > T0 > DDC

            LD      CLKCON,#18h                 ;CPU=fx(no division)

            LD      BTCON,#0A2h                 ;WatchDog Timer Disable

;

;Initialize Sync-processor control register

;

            LD      SYNCON2,#10001100B          ;66.7Hz, 50.0KHz(Pseudo sync)
                                                ;Disable Vsync detection int.
                                                ;Bit7=read only

            LD      SYNCON1,#01110000B          ;ClampO=negative polarity
                                                ;VsyncO=5-bit counter compare output

            LD      SYNCON0,#01100000B          ;Automatic Hsync blanking, VsyncO source=VsyncI port

;

;Initialize Timer control register

;

            LD      TM0CON,#10111111B           ;Timer0 clock source=@8MHz/8=1MHz(1us)
                                                ;Capture rising mode
                                                ;Enable capture/overflow interrupt
                                                ;Capture source=Vsync output path from sync-processor

            LD      TM1CON,#00011010B           ;source=HsyncI
                                                ;Capture disable, source=Timer2 interval time*10(10ms)
                                                ;Enable capture interrupt

            LD      TM2CON,#00111101B           ;Timer2 interval=@8MHz/(8*1000)=1ms


;

;*******************************************************************************

MAIN:       CALL    ChkHVPres                   ;Check H/Vsync presence

            CALL    ChkHVPol                    ;Check H/Vsync polarity
```

```
                        ;
                        ;
                JR      MAIN
;*****************************************************************************
;
;
;//***************************************************************************
;//** Title : Check H/V presence, a kind of sync source and mode changing
;//**
;//**
;//** Inputs:
;//** Outputs:
;//** Preserves:
;//** Corrupts: R0,
;//***************************************************************************
ChkHVPres:      TM      SyncP_FGR0,#01<<HSyncFin_FG     ;Checking period of Hsync frequency is 10ms
                                                        ; by Timer1/2 interrupt
                JR      Z,ChkPresnVsync
                AND     SyncP_FGR0,#0FFh-(01<<HSyncFin_FG)
;                                                       ;Every 10ms
                CALL    Chk10msTimer                    ;Time counter(100ms,1sec,2sec,3sec,7sec)
;
CheckHV_Range
                call    ChkHNosyncRange
                JP      C,SyncOffState                  ;Under 10KHz ?
;
ChkHsyncData    CALL    UHSyncChk               ;Check changing rate of Hsync frequency
                JP      C,SyncOffState          ;if, changing rate > 500Hz
                CALL    UPolaChk                ;Check Polarity data
                JP      C,SyncOffState
```

```
;
ChkPresnVsync
            CP      NoVTime,#25                 ;if, NoVTime > 25ms(under 40Hz) => Mute
            JP      UGT,ChkVsyncSrc
            TM      SyncP_FGR0,#01<<VSyncFin_FG     ;Being Vsync signal(Vsync interrupt) ?
            JP      Z,ChkHVPrtn
            AND   SyncP_FGR0,#0FFh-(01<<VSyncFin_FG)
;
            TM    Status_FGR,#01<<FindSyncSrc_FG    ;Already find sync source ?
            JP      NZ,SkipChkVsyncSrc
;
            OR      Status_FGR,#01<<FindSyncSrc_FG  ;First checking source is composite sync
                                                    ;  then seperate sync.
            AND     SyncP_FGR0,#0FFh-(01<<SetSepSync_FG)
            JP      ChkSOGsignal
;
ChkVsyncSrc CLR     NoVTime
            TM      Status_FGR,#01<<FindSyncSrc_FG  ;New sync source ?
            JR      NZ,NoPresVsync
            TM      SyncP_FGR0,#01<<SetSepSync_FG   ;No Vsync input ?
            JR      NZ,NoPresVsync
            OR      SyncP_FGR0,#01<<SetSepSync_FG
            AND     SYNCON0,#0FFh-(01<<VOSS)        ;Changing to separate-sync mode
            OR      TM1CON,#01<<T1CAPEN             ;Enable Timer1 capture mode
            RET
;
NoPresVsync CALL    ClrSyncSrcFlag
            OR      SyncP_FGR1,#01<<VNosync_FG      ;VNosync_FG <- 1
            OR      Dpms_FGR,#01<<ChkDpmsCon_FG     ;ChkDpmsCon_FG <- 1(Start DPMS check)
            TM      SyncP_FGR1,#01<<HNosync_FG      ;No Hsync & No Vsync ?
```

```
                JP      Z,SyncOffState
                OR      SyncP_FGR1,#01<<NoSync_FG
                JP      SyncOffState                    ;Mute
;
SkipChkVsyncSrc
ChkSOGsignal    TM      SYNCON1,#01<<SOGI               ;Check SOG signal input ?
                JR      Z,CountVfreq
                AND     SYNCON1,#0FFh-(01<<FBPS)        ;Back porch
;
CountVfreq      CALL    NormalVfCnt                     ;Calculate Vsync freq.
                CALL    UVSyncChk                       ;Check changing rate of Vsync frequency
                JP      C,SyncOffState
                CALL    ChkHVRange                      ;Check video signal range
                JP      C,SyncOffState                  ;Range Over !
;
StableFreqIn    TM      Mute_FGR,#01<<MuteRelse_FG
                JR      Z,ChkMuteTime
ChkHVPrtn       RET
;
;
ChkMuteTime     TM      Mute_FGR,#01<<ChkSyncStus_FG    ;Checking sync status every Vsync signal event
                JP      Z,VMuteRtn
                TM      Mute_FGR,#01<<MuteWaiting_FG    ;Already previous mute-release step ?
                JP      NZ,MuteRelease
                TM      SyncP_FGR0,#01<<HstblFreq_FG    ;Stable Hsync input ?
                JR      NZ,ChkStblVsync
                OR      Mute_FGR,#01<<ChkSyncStus_FG
                RET
ChkStblVsync    TM      SyncP_FGR0,#01<<VstblFreq_FG    ;Stable Vsync input ?
```

```
            JR      NZ,NormalSyncOut
            OR      Mute_FGR,#01<<ChkSyncStus_FG
            RET
;
NormalSyncOut
            CALL    PosiPolOut                      ;Adjust polarity to positive
            CALL    PolaUpDate                      ;Polarity data update
            LD      SYNCON2,#10000000B              ;Normal sync operation
            AND     Mute_FGR,#0FFh-(01<<PsyncOut_FG)
            OR      SyncP_FGR0,#01<<NormSync_FG     ;Re-start polarity checking
;
LoadDAC     CALL    UpdateHDuty             ;Adjust Hsync duty value in KB2511

            CALL    AdjModeSize             ;Adjust mode size according to Hsync frequency range
            CALL    B_PlusOut               ;Adjust B+ reference value in KB2511
            CALL    S_Correct               ;Adjust S-correction port
;
            OR      EepRom_FGR,#01<<EepDataRd_FG  ;Load PWM data(processing in 'MAIN' routine)
            OR      Mute_FGR,#01<<MuteWaiting_FG  ;Start time checking for mute extension
            AND     Mute_FGR,#0FFh-(01<<ChkSyncStus_FG)
;
            TM      Mute_FGR,#01<<PwrOnWait_FG
            JR      NZ,MuteDelay
            CLR     M10mSR                          ;2sec
            RET

MuteDelay   OR      Mute_FGR,#01<<NormMwait_FG      ;Load  image data -> 350ms delay -> Mute
                                                    ;  release
            CLR     M10mSR
            RET
```

```
;
MuteRelease   AND   Mute_FGR,#0FFh-(01<<ChkSyncStus_FG)
              AND   Mute_FGR,#0FFh-(01<<MuteWaiting_FG)
              AND   Mute_FGR,#0FFh-(01<<Vmute_FG)
              OR    Mute_FGR,#01<<MuteRelse_FG
              OR    P0,#01<<Muteport              ;P0.7 <- 1 : mute port release
;
              LD    R14,#PSubA_SBnBr              ;Off soft blanking(bit7 <- 0, KA2504)
              LD    R15,#00h                      ;R14= device(KA2504) sub-address,
                                                  ;  R15= control data
              CALL  Preamp_RGB_Drv
              RET
;
SyncOffState: TM    Status_FGR,#01<<SfRasterIn_FG ;Self-raster mode ?
              JR    NZ,VMuteRtn
              TM    Mute_FGR,#01<<PsyncOut_FG     ;Already pseudo sync output mode ?
              JR    NZ,VMuteRtn
;
VideoMute     AND   P0,#0FFh-(01<<Muteport)       ;V-Mute(P0.7=Active low) <- 0
              AND   SYNCON1,#11111001B            ;Pseudo Sync=only positive Pol.
              LD    SYNCON2,#10001100B            ;66.7Hz, 50KHz(Pseudo sync)
;
MutePross     LD    R14,#PSubA_SBnBr              ;soft blanking(bit7 <- 1)
              LD    R15,#80h
              CALL  Preamp_RGB_Drv
;
              OR    P1,#01<<S1                    ;P1.2(S1) <- High (Free run=48KHz)
              AND   P1,#0FFh-(01<<S2)             ;P1.3(S2) <- Low
;
              OR    Mute_FGR,#01<<Vmute_FG        ;Set video mute flag
```

```
                OR      Mute_FGR,#01<<PsyncOut_FG          ;Set pseudo sync output flag
                AND     Mute_FGR,#0FFh-(01<<MuteRelse_FG)          ;Clear mute release status flag
                AND     Mute_FGR,#0FFh-(01<<MuteWaiting_FG)        ;Clear mute extension start flag
                AND     Mute_FGR,#0FFh-(01<<NormMwait_FG)
                AND     SyncP_FGR0,#0FFh-(01<<NormSync_FG)         ;Clear sync relation register data
                AND     SyncP_FGR0,#0FFh-(01<<HstblFreq_FG)
                AND     SyncP_FGR0,#0FFh-(01<<VstblFreq_FG)
                OR      Mute_FGR,#01<<ChkSyncStus_FG
                CALL    ClrSyncSrcFlag                     ;Return to default sync source checking mode
VMuteRtn        RET
;
ClrSyncSrcFlag TM       SyncP_FGR0,#01<<DdcHighSpd_FG  ;DDC1 high speed(over 400Hz) mode ?
                JR      NZ,ClrFlagRtn
                OR      SYNCON0,#01<<VOSS                  ;VsyncO=5-bit compare output for
                                                           ;  composite sync
                AND     TM1CON,#0FFh-(01<<T1CAPEN)         ;Disable Timer1 capture mode
                AND     Status_FGR,#0FFh-(01<<FindSyncSrc_FG)
                AND     SyncP_FGR0,#0FFh-(01<<SetSepSync_FG)
ClrFlagRtn      RET                                ;Sync source checking : composite -> separate -> composite ..


;
;//*********************************************************************
;//** Title :      S-Correction
;
;               Hsync freq.   < 35KHz   => S1=L, S2=L
;               (R5=xxKHz)    < 40KHz   => S1=L, S2=H
;                             < 49KHz   => S1=H, S2=L
;                             < 60KHz   => S1=H, S2=H
;//*********************************************************************
```

```
;
H_CountLoad   LD      R4,HfHighData
              AND     R4,#00000011B
              LD      R5,HfLowData
              DIV     RR4,#10
              RET

S_Correct     CALL    H_CountLoad
              CP      R5,#35                    ;Under 35Khz
                      ;
                      ;
              RET


;//*******************************************************************
;//** Title :     Adjust Horizontal Duty Cycle(KB2511)
;
;                 Hsync freq. < 35KHz    => 00h(KB2511 address)=48h
;                             < 41KHz    => 00h=49h
;                             < 46KHz    => 00h=4Ah
;                             < 52KHz    => 00h=4Bh
;                             < 56KHz    => 00h=4Ch
;//*******************************************************************
;
UpdateHDuty   CALL    H_CountLoad
              CP      R5,#35                    ;Under 35KHz
                      ;
                      ;
              RET
```

```
;
;//*******************************************************************
;//** Title :        Adjust Mode Size(PWM6, 14/15")
;15"                 H_sync freq.< 41KHz     => PWM6=#1Bh
;                    (R5=xxKHz)  < 46KHz     => PWM6=#4Ah
;                                < 50KHz     => PWM6=#50h
;                                < 56KHz     => PWM6=#91h
;                                < 62KHz     => PWM6=#CDh
;//*******************************************************************
;
AdjModeSize   CALL    H_CountLoad
              CP      R5,#41                    ;Under 41Khz
                      ;
                      ;
              RET


;//*******************************************************************
;//** Title :        Adjust B_Plus Output
;//*******************************************************************
;
B_PlusOut     LD      R6,EP_BPlus               ;EP_BPlus=KA2511 B+ referance data
              CALL    H_CountLoad
;
              CP      R5,#41                    ;Under 41Khz
                      ;
                      ;
              ADD     R6,#0                     ;51KHz - 55.9KHz
```

```
ModeBplusOut  LD      R14,#0Bh                          ;B+ sub-address
              LD      R15,R6                            ;B+ data
              OR      Tda9109_FGR,#01<<TdaWrite_FG
              CALL    WriteCycle
              RET

;
;//*****************************************************************************
;//** Title : Check mode change and h/v frequency range for mode detection
;//**        Normal Fh <= 500Hz
;//**        Normal Fv <= 1Hz
;//**        if there is in the sync range , Set carry
;//**        HfHighNew-->bit7---> H-polarity
;//**                    bit6---> V-polarity
;//**
;//** Inputs: R0,R1
;//** Outputs:
;//** Preserves:
;//** Corrupts:
;//*****************************************************************************
;//*****************************************************************************
;//** Title : Check New polarity and old polarity
;//**
;//*****************************************************************************
UPolaChk      TM      SyncP_FGR0,#01<<NormSync_FG       ;Pseudo sync output status ?
              JR      Z,PolChkRtn
;
              LD      R0,SyncP_FGR1
              AND     R0,#11000000b
```

```
            LD      R1,HfHighData                   ;bit 7 & 6 are used for POL.
            AND     R1,#11000000b
            CP      R0,R1
            JR      NE,PolaUpDate                   ;compare hv_polarity.
;
PosiPolOut  TM      SyncP_FGR1,#01<<HPolarity_FG
            JR      Z,InvHPola
            AND     SYNCON1,#0FFh-(01<<HOS)         ;HOS=HsyncO status(polarity) control bit
ChkVPola    TM      SyncP_FGR1,#01<<VPolarity_FG
            JR      Z,InvVPola
            AND     SYNCON1,#0FFh-(01<<VOS)         ;VOS=VsyncO status(polarity by-pass)
PolChkRtn   RCF
            RET
;
InvHPola    OR      SYNCON1,#01<<HOS                ;HsyncO=Invert HsyncI signal
            JR      ChkVPola
InvVPola    OR      SYNCON1,#01<<VOS
            JR      PolChkRtn
;
PolaUpDate  LD      R4,HfHighNew
            AND     R4,#00000011b
            LD      R5,SyncP_FGR1
            AND     R5,#11000000b                   ;masking except polarity flag
            OR      R4,R5
            LD      HfHighData,R4
            SCF
            RET

;
```

```
;//*****************************************************************************
;//** Title : Compare Vertical Frequency
;//**
;//**
;//*****************************************************************************
UVSyncChk   LD      R0,VfCurrNew
            CP      R0,VFreqData
            JR      ULT,RevVSub
            SUB     R0,VFreqData

CmpFvRng    CP      R0,#1                           ;compare 1Hz
            JR      UGT,UpdateVfData
            OR      SyncP_FGR0,#01<<VstblFreq_FG    ;Set stable Vsync signal input flag
            RCF
            RET
;
RevVSub     LD      R0,VFreqData                    ;VFreqData= saved stable Vsync frequency
            SUB     R0,VfCurrNew                ;VfCurrNew= inputted new Vsync frequency
            JR      CmpFvRng
;
UpdateVfData AND    SyncP_FGR0,#0FFh-(01<<VstblFreq_FG)
            LD      VFreqData,VfCurrNew             ;refresh v-frequency
            SCF
            RET

;
;//*****************************************************************************
;//** Title : Compare Horizontal Frequency
;//**
```

```
;//**
;//*****************************************************************************
UHSyncChk   LD      R0,HfLowNew
            SUB     R0,HfLowData                    ;R0 = |HfLowNew - HfLowData|
            LD      R1,HfHighNew
            LD      R2,HfHighData
            AND     R2,#00000011b                   ;R1 = |HfHighNew - HfHighData|
            SBC     R1,R2
            JR      C,RevHSub
;
CmpFhRng    CP      R1,#00h                         ;HfHighNew =/ HfHighData      ?
            JR      NE,UpdateHfData
                    ;TM         SYNCON0,#01<<VOSS;Composite sync signal ?
                    ;JR         Z,ChkNormHfRng
                    ;CP         R0,#9                       ;Compare 1KHz
                    ;JR         UGT,UpdateHfData
                    ;JR         StblHsyncIn
ChkNormHfRng CP     R0,#4                           ;Changing rate < 500Hz ?
            JR      UGT,UpdateHfData
StblHsyncIn OR      SyncP_FGR0,#01<<HstblFreq_FG    ;Set stable Hsync signal input flag
;
            LD      R4,HfHighNew
            LD      R5,HfLowNew
            DIV     RR4,#10
            LD      AverageHf,R5
            RCF
            RET
;
RevHSub     LD      R0,HfLowData
```

```
                SUB     R0,HfLowNew
                LD      R1,HfHighData
                LD      R2,HfHighNew
                AND     R1,#00000011b
                SBC     R1,R2
                JR      CmpFhRng
;
UpdateHfData    AND     SyncP_FGR0,#0FFh-(01<<HstblFreq_FG)
                AND     HfHighData,#11000000B
                OR      HfHighData,HfHighNew
                LD      HfLowData,HfLowNew              ;refresh h-frequency
                SCF
                RET


;
;//*****************************************************************************
;//** Title : Normalize vertical Counter
;//**        counter source = @8M/8 =1us(Timer0 capture mode)
;//**        x = Vcount
;//**        Fv=1000000 / x
;//** Inputs:   Vcount,Vcount+1(=net #T0CNT)
;//** Outputs:  VfCurrNew
;//** Preserves:
;//** Corrupts:
;//*****************************************************************************
;Calculation method-1.
NormalVfCnt     CLR     R4                      ;Vsync interval time(Vcount=#0XXXXXus)
                LD      R5,Vcount               ;1/#0XXXXX us(frequency) = 1000000/0XXXXX
```

```
            DIV     RR4,#10             ;               = (1000000/10) / (0XXXXX/10)
            LD      R6,R5               ;               = (100000 /2)  / (00XXXX/2)
            LD      R5,Vcount+1
            DIV     RR4,#10             ;Vcount/10
            LD      R7,R5               ;Low
            RCF
            RRC     R6                  ;
            RRC     R7                  ;RR6(time=XX.XXms)/2
;
            LD      R4,#0C3h            ;#C350=50000
            LD      R5,#50h
            CLR     R0
;
ContiSub    RCF
            SUB     R5,R7
            SBC     R4,R6
            JR      C,FiniVfCal
            ADD     R0,#1                       ;R0=xxHz(Frequency=1/Time)
            JR      C,NoVsyncSignal             ;Overflow(over 256Hz) ?
            JR      ContiSub
FiniVfCal   LD      VfCurrNew,R0
            RET
;
NoVsyncSignal CLR   VfCurrNew                   ;VfCurrNew <- 00Hz
              RET

;
;
;//*****************************************************************************
```

```
;//** Title : Check H/V Sync Range
;//**        27Khz(10Eh) < Fh < 62KHz(15" 26Ch)
;//**          40Hz(28h) < Fv < 135Hz(87h)
;//**
;//**
;//** Inputs:
;//** Outputs:
;//** Preserves:
;//** Corrupts:
;//*****************************************************************************
ChkHVRange:  LD      R2,HfHighNew
             LD      R3,HfLowNew
             RCF
             SUB     R3,#0Eh                    ;#10Eh=27KHz
             SBC     R2,#01
             JR      c,OverHfRange
;
             LD      R2,HfHighNew
             LD      R3,HfLowNew
             RCF
             SUB     R3,#6Ch                    ;#26Ch=62KHz
             SBC     R2,#02
             JR      NC,OverHfRange
             AND     SyncP_FGR1,#0FFh-(01<<NoSync_FG)
             AND     SyncP_FGR1,#0FFh-(01<<OverHsync_FG)
;
ChkVfreqRange
             CP      VfCurrNew,#40              ;40Hz
             JR      ULT,NoVsyncIn
```

```
            CP      VfCurrNew,#135                  ;135Hz
            JR      UGT,OverRange
;
NormHVsyncIn AND    SyncP_FGR1,#0FFh-(01<<VNosync_FG)
            AND     SyncP_FGR1,#0FFh-(01<<NoSync_FG)
            AND     SyncP_FGR1,#0FFh-(01<<OverRange_FG)
            AND     P0,#0FFh-(01<<Suspndport)   ;Stop Suspend
            NOP
            NOP
            NOP
            NOP
            OR      P0,#01<<Offport                 ;Stop Off
;
            TM      Dpms_FGR,#01<<PwrOffIn_FG       ;Power off/suspend mode ?
            JR      Z,ClrDpmsFlags
PreAmpRelese CALL   CtrlPreAmp                      ;Control pre-amp
            CALL    InitialKB2511                   ;TDA9109(Initializing)
            CLR     M10mSR
            CALL    VideoMute                       ;set power on condition
;
ClrDpmsFlags CLR    DPMS100mSR                      ;Clear dpms checking counter
            AND     Dpms_FGR,#00000011B             ;Clear DPMS flags
            TM      Status_FGR,#01<<SfRasterIn_FG   ;Self Raster in ?
            JR      Z,ChkHVRrtn
            CALL    SfRasterEnd                     ;Release self_raster input mode
ChkHVRrtn   RCF
            RET
;
OverHfRange OR      SyncP_FGR1,#01<<OverHsync_FG
```

```
                OR      SyncP_FGR1,#01<<OverRange_FG
                JR      ChkVfreqRange
OverRange       OR      SyncP_FGR1,#01<<OverRange_FG
                JR      OverRngRtn
;
NoVsyncIn       OR      SyncP_FGR1,#01<<VNosync_FG
                OR      Dpms_FGR,#01<<ChkDpmsCon_FG  ;Start checking the maintaining time of
                                                     ;  dpms mode
                ;
                TM      SyncP_FGR1,#01<<HNosync_FG   ;No Hsync & No Vsync ?
                JR      Z,OverRngRtn
                OR      SyncP_FGR1,#01<<NoSync_FG
OverRngRtn      SCF
                RET


;//*****************************************************************************
;//** Title : Check Horizontal No sync  Range
;//**        Fh < 10Khz(64h)
;//**
;//**
;//** Inputs:
;//** Outputs:
;//** Preserves:
;//** Corrupts:
;//*****************************************************************************
ChkHNoSyncRange
                LD      R0,HfHighNew
                LD      R1,HfLowNew
                SUB     R1,#64h                      ;#64h=100=10.0KHz
```

```
                SBC     R0,#00
                JR      C,HNosyncRange
                AND     SyncP_FGR1,#0FFh-(01<<HNosync_FG)
                AND     SyncP_FGR1,#0FFh-(01<<NoSync_FG)
                RET
;
HNosyncRange OR         SyncP_FGR1,#01<<HNosync_FG      ;HNosync_FG <- 1
                OR      Dpms_FGR,#01<<ChkDpmsCon_FG     ;Start DPMS condition counting
                RET


;//*****************************************************************************
;//** Title : Check H/V polarity for every 5ms
;//**
;//**
;//** Inputs: VsyncI port data, SYNCON1.0
;//** Outputs: HPolarity_FG/VPolarity_FG (in SyncP_FGR1)
;//** Preserves:
;//** Corrupts:
;//*****************************************************************************
;
ChkHVPol:       INC     PolaCntr                        ;PolaCntr++
                CP      PolaCntr,#40                    ;About 5ms
                JR      UGT,DetectPol
;
                TM      SYNCRD,#01<<VsyncIport          ;VsyncI signal=High ?
                JR      NZ,DecVPolCntr
                INC     VPolaCntr                       ;VPolaCntr++
                RET
```

```
DecVPolCntr   DEC   VPolaCntr                    ;VPolaCntr--
              RET

;
DetectPol     TM    SYNCON0,#01<<VOSS            ;Composite sync(H+V) ?
              JR    Z,ChkSepSyncPol
              TM    SYNCON1,#01<<HPOL            ;Hsync polarity=positive ?
              JR    NZ,HVposiPola
              AND   SyncP_FGR1,#0FFh-(01<<HPolarity_FG)
              AND   SyncP_FGR1,#0FFh-(01<<VPolarity_FG)
              JR    ChkHVPol_rtn
HVposiPola    OR    SyncP_FGR1,#01<<HPolarity_FG
              OR    SyncP_FGR1,#01<<VPolarity_FG
              JR    ChkHVPol_rtn
;
ChkSepSyncPol
              TM    SYNCON1,#01<<HPOL            ;Seperated sync signal
              JR    NZ,HposiPola
              AND   SyncP_FGR1,#0FFh-(01<<HPolarity_FG)
              JR    ChkVsyncPol
HposiPola     OR    SyncP_FGR1,#01<<HPolarity_FG
ChkVsyncPol   CP    VPolaCntr,#36                ;90%(36EA~40EA/40)
              JR    ULT,ChkHVPol_rtn             ;Error
              OR    SyncP_FGR1,#01<<VPolarity_FG
              CP    VPolaCntr,#216               ;216=256-40EA
              JR    ULT,ChkHVPol_rtn             ;Error -> No update
              AND   SyncP_FGR1,#0FFh-(01<<VPolarity_FG)
ChkHVPol_rtn  CLR   PolaCntr
              CLR   VPolaCntr
```

```
                RET


;
;//*****************************************************************************
;//** Title : Timer 0 overflow interrupt(interval=256us(1us*256))
;//**
;//**
;//** Inputs: fosc(@8MHz)/8=1us(Timer0 clock source)
;//** Outputs: Overflow count for Vsync interval
;//** Preserves:
;//** Corrupts:
;//*****************************************************************************

T0Ovf_Int:      PUSH    PP                          ;256us interval interrupt
                CLR     PP
                INC     VFreqCHigh                  ;Overflow counter for Vsync freq. calcu.
                INC     T0OvfCntr
                POP     PP
                IRET


;//*****************************************************************************
;//** Title : Timer 1 capture interrupt(T1DATA=Number of Hsync signal for 10ms)
;//**
;//**
;//** Inputs: Hsync signal(event counter source)
;//** Outputs: Number of Hsync signal for 10ms(separate sync input mode)
;//** Preserves:
```

```
;//** Corrupts:
;//*****************************************************************************

T1Cap_Int:      SB0
                PUSH    PP
                CLR     PP
                AND     TM1CON,#0FFh-(01<<T1PND)        ;Pending clear
                TM      TM1CON,#01<<T1CAPEN             ;Composite sync signal ?
                JR      Z,T1CapRtn
;
                LD      HfHighNew,TM1DATAH              ;TM1DATA is 12-bit event counter for 10ms
                LD      HfLowNew,TM1DATAL
                CLR     AverageHf
                OR      SyncP_FGR0,#01<<HSyncFin_FG     ;Set Hsync signal find flag
                OR      Time_FGR,#01<<KeyDetect_FG      ;Key scanning(interval time=every 10ms)
                OR      Time_FGR,#01<<ChkPwrKey_FG      ;Power key checking
T1CapRtn        POP     PP
                IRET

;
;//*****************************************************************************
;//** Title : Timer 2 base time interrupt(interval=1ms)
;//**
;//**
;//** Inputs: fosc(@8MHz)/(1000*8)=1ms interrupt
;//** Outputs: Number of Hsync signal for 10ms(composite sync input mode)
;//** Preserves:
;//** Corrupts:
;//*****************************************************************************
```

```
T2Intv_Int:     SB0                                     ;1ms interval timer
                PUSH    PP
                PUSH    R0
                CLR     PP
                TM      SYNCON0,#01<<VOSS               ;Separated sync signal ?
                JP      Z,T2IntvIrtn
;
CompSyncCalcu
                LD      HFreqStCnt,HFreqSpCnt
                LD      HFreqSpCnt,TM1CNTL
                TM      SYNCON2,#01<<UNMIXHSYNC         ;Mixed sync signal(Hsync period with
                                                        ;  Vsync signal) ?
                JR      Z,MixedSyncInput
                TM      SyncP_FGR0,#01<<MixedSync_FG
                JR      Z,NormHcount
;
MixedSyncInput
                ADD     HCount+1,AverageHf              ;if input sync is mixed sync input for this 1ms
                                                        ;  period
                ADC     HCount,#0
                JR      BaseTimer
NormHcount      LD      R0,HFreqSpCnt
                SUB     R0,HFreqStCnt
                ADD     HCount+1,R0                     ;R0=number of Hsync signal for 1ms
                ADC     HCount,#0                       ;
;
BaseTimer       TM      SYNCON2,#01<<UNMIXHSYNC                 ;Not finish mixed-sync period ?
                JR      Z,Check10ms
                AND     SyncP_FGR0,#0FFh-(01<<MixedSync_FG)
```

```
Check10ms   INC   TB1mSR
            CP    TB1mSR,#10
            JR    ULT,T2IntvIrtn
;
            LD    HfHighNew,HCount                ;Number of Hsync signal for 10ms
            LD    HfLowNew,HCount+1
            CLR   TB1mSR
            LDW   HCount,#00h
            OR    SyncP_FGR0,#01<<HSyncFin_FG     ;Set Hsync signal find flag
            OR    Time_FGR,#01<<KeyDetect_FG      ;Key scanning
            OR    Time_FGR,#01<<ChkPwrKey_FG      ;Power key checking
T2IntvIrtn  POP   R0
            POP   PP
            IRET


;
;//*****************************************************************************
;//** Title : 10msec time base
;//**
;//*****************************************************************************
Chk10msTimer: TM  Mute_FGR,#01<<NormMwait_FG
            JR    Z,PwrUpMTime
            INC   M10mSR
            CP    M10mSR,#35                      ;Mute delay=350ms
            JR    UGT,SetMuteChkTime
            JR    TimeB10mS
PwrUpMTime  TM    Mute_FGR,#01<<PwrOnWait_FG
            JR    NZ,TimeB10mS
```

```
                INC     M10mSR
                CP      M10mSR,#200                     ;Power-up mute=2sec
                JR      ULE,TimeB10mS
                OR      Mute_FGR,#01<<PwrOnWait_FG
                AND     Mute_FGR,#0FFh-(01<<Vmute_FG)
SetMuteChkTime
                AND     Mute_FGR,#0FFh-(01<<NormMwait_FG)
                OR      Mute_FGR,#01<<ChkSyncStus_FG    ;ChkSyncStus_FG <- 1
;
TimeB10mS       DEC     TB10mSR
                JR      NE,TBaseRtn
                LD      TB10mSR,#10
;
;//*****************************************************************************
;//** Title : 100msec time base
;//**
;//*****************************************************************************
Time100mS       CALL    ChkDegEndTime
                TM      SyncP_FGR1,#01<<OverRange_FG    ;Over range ?
                JR      NZ,CheckDpmsIn
                TM      Dpms_FGR,#01<<ChkDpmsCon_FG     ;DPMS condition ?
                JR      Z,TBaseRtn
;
CheckDpmsIn     CP      DPMS100mSR,#30                  ;3sec ?
                JR      UGE,SetDpmsChk
                INC     DPMS100mSR
                JR      TBaseRtn
SetDpmsChk      OR      Dpms_FGR,#01<<DpmsStart_FG
TBaseRtn        RET
```

```
;
;
;//*****************************************************************************
;//** Title : Degaussing time check
;//**
;//**
;//*****************************************************************************
;
ChkDegEndTime
            TM      Time_FGR,#01<<DeGTime_FG
            JR      Z,ChkDegEndRet
            DEC     DG100mSR
            JR      NE,ChkDegEndRet
            AND     P0,#0FFh-(01<<DeGausport)  ;Degaussing(3sec) Off
            AND     Time_FGR,#0FFh-(01<<DeGTime_FG)
ChkDegEndRet RET

;
;
;//*****************************************************************************
;//** Title : Timer0 Vsync edge interrupt
;//**
;//**
;//** Inputs : VFreqCHigh(=T0 ovf count), T0DATA(=T0 capture data)
;//** Outputs: Vcount(=Vsync interval time[us])
;//** Preserves:
;//** Corrupts:
;//*****************************************************************************
```

```
;

VSyncDet_Int:  PUSH   PP
               CLR    PP
               CLR    NoVTime
               CP     T0OvfCntr,#10                  ;Over 400Hz(DDC1 mode) ?
               JR     UGT,VsyncDetSrv
               CLR    T0OvfCntr
               OR     SyncP_FGR0,#01<<DdcHighSpd_FG
               POP    PP
               IRET
;
VsyncDetSrv    SB0
               OR     SyncP_FGR0,#01<<VSyncFin_FG    ;bit finsih vsync counter
               LD     Vcount,VFreqCHigh              ;Number of Timer0 overflow counter
               MULT   Vcount,#255
               ADD    Vcount+1,VFreqCHigh        ;VFreqCHigh*256
               ADC    Vcount,#0
               ADD    Vcount+1,TM0DATA               ;Vcount(2-byte)=(#Overflow*256)+T0CNT
               ADC    Vcount,#0
               CLR    VFreqCHigh
               CLR    T0OvfCntr
               AND    SyncP_FGR0,#0FFh-(01<<DdcHighSpd_FG)
;
CheckDdcVclk   TM     IIC_FGR,#01<<Ddc2mode_FG       ;Correct DDC2B mode ?
               JR     NZ,SetMixSyncFlag
               SB1
               TM     DCON,#01<<DDC1EN               ;DDC1 mode ?
               JR     NZ,SetMixSyncFlag
```

```
                INC     VclkCntr                        ;Increment Vsync counter for DDC recovery
SetMixSyncFlag
                SB0
                TM      SYNCON0,#01<<VOSS
                JR      Z,VsyncDetIrtn
                OR      SyncP_FGR0,#01<<MixedSync_FG
VsyncDetIrtn    POP     PP
                IRET



;
;//*****************************************************************************
;//** Title : Interrupt for multi master I2c bus processor
;//**        - Vector address --> 00F8h for Irq1
;//**
;//** Inputs: PC/Control jig -> Monitor (DDC1/2B/2B+)
;//** Outputs: Monitor -> PC/Control jig (DDC1/2B/2B+)
;//** Preserves:
;//** Corrupts:
;//*****************************************************************************

DDCnFA_Int:     SB1
                PUSH    PP
                CLR     PP
;
;-----------------------------------------------------
;----        DDC1 Tx protocol processor      -----
;-----------------------------------------------------
                TM      DCON,#01<<DDC1EN                ;Normal interface mode(No DDC1) ?
```

```
            JR      Z,DDC2Routine
            TM      DCON,#01<<SCLF              ;Is falling edge detected at SCL pin ?
            JP      Z,EdidTx                    ;DDC1 EDID Tx
            LD      PP,#11h
            CLR     TBDR                        ;First EDID(#00h)
            CLR     EdidAddr
            AND     DCON,#0FFh-(01<<DDC1EN)     ;DDC1 -> Normal IIC-bus
            AND     DCCR,#0FFh-(01<<DPND)       ;Clear IIC.bus int. pending bit
            POP     PP
            SB0
            IRET

;
;-----------------------------------------------
;---  DDC2 protocol mode checking    ---
;-----------------------------------------------
DDC2Routine TM      DCSR0,#01<<DMTX             ;Master Tx :DDC2B+ (Monitor -> PC)
            JP      NZ,Master
            TM      DCSR0,#01<<DSTX             ;Slave Tx  :DDC2Bi (Monitor <-> PC)
            JR      NZ,ChkDDC2mode
            TM      DCON,#01<<DDC1MAT           ;DDC2B(#A0h) mode ?
            JP      NZ,DDC2Bmode
            TM      IIC_FGR,#01<<RevA0match_FG  ;Already received slave address #A0h ?
            JP      NZ,DDC2Bmode
;
            OR      IIC_FGR,#01<<Ddc2mode_FG
            OR      IIC_FGR,#01<<DDCCmd_FG      ;if receive mode(DDC2B+/Ci), set DDCCmd_FG
            PUSH    R0
            LD      R0,#MBusBuff            ;DDC2B+ : PC -> Monitor
```

```
                ADD     R0,RxXCntr
                LD      @R0,RBDR                        ;@(#MBusBuff+RxXCntr) <- RBDR(=Rx buffer)
                INC     RxXCntr
                POP     R0
                CLR     VclkCntr                        ;DDC error checking timer
                AND     DCON,#0FFh-(01<<DDC1EN)         ;DDC1 -> Normal IIC-bus interface mode
DdcSrvRtn       AND     DCCR,#0FFh-(01<<DPND)           ;Clear IIC.bus int. pending bit
                POP     PP
                SB0
                IRET

;
;-----------------------------------------------------
;---   DDC2B protocol processor        ---
;-----------------------------------------------------
ChkDDC2mode
                AND     IIC_FGR,#0FFh-(01<<RevA0match_FG)
                TM      DCON,#01<<DDC1MAT               ;Address match as #A0h(DDC2B mode)
                JP      Z,DDC2BiPrss
;
                TM      DCSR0,#01<<DRXACK               ;Is ACK received ?
                JR      NZ,DdcCommFail
;
                TM      IIC_FGR,#01<<Ddc2BTxmode_FG     ;
                JR      NZ,EdidTx
                OR      IIC_FGR,#01<<Ddc2BTxmode_FG     ;Set match flag of slave Tx mode
                                                        ;  address(#A1h)
                CP      DDSR,#00h                       ;In this case : A0h -> 00h -> P & S -> A1h -> ..
                JR      NE,EdidTx
                LD      PP,#11h                         ;In this case : A0h -> 00h -> S -> A1h -> ..
```

```
                INC     EdidAddr                ;EdidAddr : 00h -> 01h (Repeat start case)
;
EdidTx          LD      PP,#11h
                CP      EdidAddr,#7Fh           ;EDID=00h~7Fh(Page1)
                JR      ULE,PrepNextAddr
                CLR     EdidAddr                ;First data
PrepNextAddr    LD      TBDR,@EdidAddr          ;TBDR=Tx buffer
                TM      DCSR1,#01<<DBUFEMT
                JR      NZ,ReloadTxBuff
                INC     EdidAddr
                AND     DCCR,#0FFh-(01<<DPND)   ;Clear IIC.bus int. pending bit
                POP     PP
                SB0
                IRET
ReloadTxBuff    INC     EdidAddr
                LD      TBDR,@EdidAddr          ;For keeping normal pre-buffer mode
                INC     EdidAddr
                AND     DCCR,#0FFh-(01<<DPND)
                POP     PP
                SB0
                IRET
;
DdcCommFail     LD      PP,#11h
                CLR     TBDR                    ;First data of EDID
                CLR     EdidAddr
                AND     DCSR0,#0FFh-(01<<DSTX)  ;Return slave Rx mode
                AND     DCCR,#0FFh-(01<<DPND)   ;Clear IIC.bus int. pending bit
                POP     PP
                SB0
```

```
                IRET

;/ DDC1 -> DDC2B mode
DDC2Bmode       CLR     VclkCntr                        ;For DDC1 recovery mode
                OR      IIC_FGR,#01<<Ddc2mode_FG
                CP      RBDR,#0A0h                      ;Slave address ?
                JR      EQ,RevDdc2bAddr
                LD      PP,#11h
                CP      RBDR,#00h                       ;Sub-address ?
                JR      NE,RandomAddr
                CLR     TBDR                            ;First data of EDID(#00h)
                CLR     EdidAddr
                AND     DCCR,#0FFh-(01<<DPND)           ;Clear IIC.bus int. pending bit
                POP     PP
                SB0
                IRET
RandomAddr      LD      TBDR,@RBDR
                LD      EdidAddr,RBDR
                AND     DCCR,#0FFh-(01<<DPND)           ;Clear IIC.bus int. pending bit
                POP     PP
                SB0
                IRET
;
RevDdc2bAddr    OR      IIC_FGR,#01<<RevA0match_FG      ;Set slave address(#A0h) match flag
                AND     IIC_FGR,#0FFh-(01<<Ddc2BTxmode_FG)      ;Clear slave Tx address match flag
                LD      PP,#11h
                CLR     EdidAddr
                AND     DCCR,#0FFh-(01<<DPND)           ;Clear IIC.bus int. pending bit
                POP     PP
```

```
                SB0
                IRET


;-----------------------------------------------
;---   DDC2Bi protocol processor      ---
;-----------------------------------------------
DDC2BiPrss      NOP
                JP      DdcSrvRtn
;
;//*****************************************************************************
;//** Title : Mater transmitter processor
;//**
;//**
;//*****************************************************************************
Master:         PUSH    IMR
                LD      IMR,#00000011B                  ;Enable T0/T1/T2 int.
                EI
;
                TM      DCSR1,#01<<MISPLS
                JR      NZ,CommFail             ;Mispalced condition error
                TM      DCSR0,#01<<DAL
                JR      NZ,CommFail             ;Bus arbitration failed during communication
                TM      DCSR0,#01<<DRXACK
                JR      NZ,CommFail             ;Not received ACK
;
                PUSH    R0
                PUSH    R2
                PUSH    R3
```

```
            PUSH    R4
            PUSH    R5
            PUSH    R6
            CALL    TxdComPart              ;DDC2B+ communication (Monitor -> PC(Control jig))
            POP     R6
            POP     R5
            POP     R4
            POP     R3
            POP     R2
            POP     R0
            DI
            POP     IMR
            POP     PP
            AND     DCCR,#0FFh-(01<<DPND)           ;Clear IIC.bus int. pending bit
            SB0
            IRET
;
CommFail    CLR     SendType                        ;SendType <- #00h
            CLR     ByteCnt                         ;ByteCnt <- #00h
            DI
            AND     DCSR0,#0FFh-(01<<DBB)           ;Stop IIC.bus interface
            AND     DCCR,#0FFh-(01<<DPND)           ;Clear IIC.bus int. pending bit
            POP     IMR
            POP     PP
            SB0
            IRET

;
;*********************************************************************
```

```
;*****          SendType                                  *****
;*****          1 : Attention                             *****
;*****          2 : Reply Identify                        *****
;*****          3 : Reply Capability                      *****
;*****          4 : Reply Vcp                             *****
;*****          5 : Reply Timing                          *****
;*****          6 : Reply All mode save(EDID data dump)   *****
;*****          7 : Reply Factory Save                    *****
;*********************************************************************

;
;*************************************************************
;-----      Common parts transmit data-----
;-----      Master Transmit in IIC Interrupt      -----
;*************************************************************
TxdComPart:    NOP
               NOP
               RET


;****************************************
VcpTbl         ;DB    V_HPosition     ;$20     ;0  !HPosition Vcp
               ;DB    V_VPosition     ;$30     ;1  !VPosition Vcp
               ;DB    V_HSize         ;$22     ;2  !HSize Vcp
               ;DB    V_VSize         ;$32     ;3  !VSize Vcp
               ;DB    V_Pincushion    ;$24     ;4  !Pincushion Vcp
               ;DB    V_Trapezoid     ;$42     ;5  !Trapeziod Vcp
               ;DB    V_Parallel      ;$40     ;6  !Parallel Vcp
               ;DB    V_Pinbalance    ;$26     ;7  !Pinbalance Vcp
```

```
                ;DB     V_VLinearity        ;$3A            ;8  !VLinearity Vcp
                ;DB     V_Tilt              ;$44            ;9  !Tilt Vcp
                ;DB     V_HSizeMin          ;$E4            ;10 !HSizeMin Vcp
                ;DB     V_SSelect           ;3Ch            ;11
                ;DB     V_VMoire            ;58h            ;12
                                            ;KA2504 Pre-amp
                ;DB     V_Contrast          ;$12            ;13
                ;DB     V_RGain             ;16h            ;14
                ;DB     V_GGain             ;18h            ;15
                ;DB     V_BGain             ;1Ah            ;16
                ;DB     V_CoffBright        ;10h            ;17
                ;DB     V_RCOff             ;6Ch            ;18
                ;DB     V_GCOff             ;6Eh            ;19
                ;DB     V_BCOff             ;70h            ;20
                ;DB     V_ACL               ;F6h            ;21
                ;DB     V_Degauss           ;01h            ;22
;;
;-----------------------------------------------------
;--   Pre_Amp Data Transfer Format   ---
;-----------------------------------------------------
;------          IIC_P_Amp_Start                 -------
;------          Slave Address   :#0DCh          -------
;------          Sub Address     :Rgb_Drv_Tbl    -------
;------          Data            :@DataAddr      -------
;------          IIC_P_Amp_Stop                  -------
;-------------------------------------------------------------------
;
Preamp_RGB_Drv:
                PUSH    R0
```

```
            PUSH    R1
            PUSH    R2

            SB0
            CLR     R2
Preamp_One_Drv
            CALL    IICbus_Start
            LD      R0,#0DCh                    ;KA5204 slave address(#0DCh)
            CALL    P_Amp_Drv_Byte
            TM      IIC_FGR,#01<<CommFail_FG
            JR      NZ,KA2504Stop

            LD      R0,R14                      ;KA2504 sub-address
            CALL    P_Amp_Drv_Byte
            TM      IIC_FGR,#01<<CommFail_FG
            JR      NZ,KA2504Stop
            LD      R0,R15                      ;KA2504 control data
            CALL    P_Amp_Drv_Byte
KA2504Stop  CALL    IICbus_Stop
            TM      IIC_FGR,#01<<CommFail_FG
            JR      Z,PreAmpDrvRtn
            AND     IIC_FGR,#0FFh-(01<<CommFail_FG)
            INC     R2
            CP      R2,#2                       ;Error ?
            JR      ULE,Preamp_One_Drv
PreAmpDrvRtn POP    R2
            POP     R1
            POP     R0
            RET
```

```
;
;.......................................................
;
Rgb_Drv_Tbl   DB    00h                     ;pre_amp (Contrast)
              DB    01h                     ;pre_amp (Brightness)
              DB    02h                     ;pre_amp (R_Gain)
              DB    03h                     ;pre_amp (G_Gain)
              DB    04h                     ;pre_amp (B_Gain)
              DB    05h                     ;pre_amp (OSD Contrast)
              DB    06h                     ;pre_amp (Cut-Off Brightness)
              DB    07h                     ;pre_amp (R_Cutoff)
              DB    08h                     ;pre_amp (G_Cutoff)
              DB    09h                     ;pre_amp (B_Cutoff)
              DB    0Ah                     ;Switch
;
;
;*****************************************************
;******   PreAmp DISPLAY INITIDAL      *******
;******   Mfr. : samsung electronc     *******
;******   Type : KA2504                *******
;*******************************************************
;-------      PreAmp Start Condition      -------
;-----------------------------------------------------------
IICbus_Start  OR    BTCON,#01<<BTCLR        ;Clear Watch-dog timer
              OR    P3,#11000000B           ;P3.7/6 <- High(SDA,SCL)
              CALL  DelayNop                ;IIC-Start
              AND   P3,#0FFh-(01<<SDA)
              CALL  DelayNop
              AND   P3,#0FFh-(01<<SCL)
```

```
                RET
;
;-------------------------------------------------------
;------    IIC_Bus Clock Generation   -------
;-------------------------------------------------------
IIC_Clock_1Bit  OR      P3,#01<<SCL                     ;Clock Generation.
                CALL    DelayNop
                AND     P3,#0FFh-(01<<SCL)
                CALL    DelayNop
                RET
;
DelayNop        NOP
                NOP
                RET
;-----------------------------------------------
;-----      IIC Stop Condition     -------
;-----------------------------------------------
IICbus_Stop     AND     P3,#0FFh-(01<<SDA)
                NOP
                NOP
                NOP
                OR      P3,#01<<SCL
                Call    DelayNop
                OR      P3,#01<<SDA                     ;SDA <- High(Stop condition)
                RET
;-------------------------------------------------------------
;------       SHIFT LEFT ONE BYTE        -------
;------      Parameter : R10  : Shift Data     -------
;------         R11  : Bit Counter              -------
```

```
;-------------------------------------------------------------
P_Amp_Drv_Byte
                LD      R1,#8
ShiftLeft       RLC     R0
                JR      NC,Data_Low
                OR      P3,#01<<SDA

Gen_Clock       OR      P3,#01<<SCL                     ;Clock Generation.
                NOP
                NOP
                AND     P3,#0FFh-(01<<SCL)
                DJNZ    R1,ShiftLeft                    ;R1=DataCntr

ACK_Check       AND     P3CONH,#00111111B               ;SDA(P3.7)=Input
                Call    DelayNop
                OR      P3,#01<<SCL                     ;Acknowledge clock
                NOP
                NOP
                TM      P3,#01<<SDA                     ;Ack in ?
                JR      NZ,ACK_Fail
;
ACK_OK          OR      P3CONH,#11000000B               ;SDA(P3.7)=Output
                AND     P3,#0FFh-(01<<SCL)
                AND     IIC_FGR,#0FFh-(01<<CommFail_FG)
                RET
;
;
ACK_Fail        OR      P3CONH,#11000000B               ;SDA(P3.7)=Output
                AND     P3,#0FFh-(01<<SCL)
```

```
                OR      IIC_FGR,#01<<CommFail_FG
                RET
;
Data_Low        AND     P3,#0FFh-(01<<SDA)
                JR      GEN_Clock


CtrlPreAmp      CALL    TimeDelay
                CALL    TimeDelay
                CALL    CtrlPreDrv_Sub
;
                CALL    TimeDelay
                CALL    TimeDelay
                CALL    CtrlPreDrv_Sub
                RET
;
CtrlPreDrv_Sub  TM      EepRom_FGR,#01<<SavedEep_FG
                JR      NZ,LdEepRGB
;
                LD      EP_CoffBRIGHT,#0C0h
                LD      EP_CONTRAST,#0FFh
                LD      EP_RGain,#3Fh
                LD      EP_GGain,#39h
                LD      EP_BGain,#32h
                LD      EP_RCutoff,#8Ah
                LD      EP_GCutoff,#80h
                LD      EP_BCutoff,#0A6h
                LD      EP_ACL,#76h
                JR      CtrlKA2504
```

```
LdEepRGB    LD      R14,#EPA_CoBr           ;Brightness data
            CALL    ReadEepData
            LD      EP_CoffBRIGHT,ReadData
            INC     R14                     ;Contrast data
            CALL    ReadEepData
            LD      EP_CONTRAST,ReadData
            INC     R14                     ;R-Gain
            CALL    ReadEepData
            LD      EP_RGain,ReadData
            INC     R14                     ;G-Gain
            CALL    ReadEepData
            LD      EP_GGain,ReadData
            INC     R14                     ;B-Gain
            CALL    ReadEepData
            LD      EP_BGain,ReadData
            INC     R14                     ;R-CutOff
            CALL    ReadEepData
            LD      EP_RCutoff,ReadData
            INC     R14                     ;G-CutOff
            CALL    ReadEepData
            LD      EP_GCutoff,ReadData
            INC     R14                     ;B-CutOff
            CALL    ReadEepData
            LD      EP_BCutoff,ReadData
;
            LD      R14,#EPA_ACL            ;ACL data
            CALL    ReadEepData
            LD      EP_ACL,ReadData
```

```
CtrlKA2504      LD      R14,#PSubA_SBnBr        ;Brightness & Soft blanking
                LD      R15,#80h
                CALL    Preamp_RGB_Drv
                SB1
                LD      PWM5,EP_ACL             ;ACL
                SB0
;RepeatPreamp   LD      R14,#PSubA_Cont
                LD      R15,EP_CONTRAST
                CALL    Preamp_RGB_Drv
                LD      R14,#PSubA_CoBr
                LD      R15,EP_CoffBRIGHT
                CALL    Preamp_RGB_Drv
;
                LD      R14,#PSubA_RGain        ;KA2504 sub address
                LD      R15,EP_RGain            ;DataAddr
                CALL    Preamp_RGB_Drv
                LD      R14,#PSubA_GGain
                LD      R15,EP_GGain
                CALL    Preamp_RGB_Drv
                LD      R14,#PSubA_BGain
                LD      R15,EP_BGain
                CALL    Preamp_RGB_Drv
                LD      R14,#PSubA_RCo
                LD      R15,EP_RCutoff
                CALL    Preamp_RGB_Drv
                LD      R14,#PSubA_GCo
                LD      R15,EP_GCutoff
                CALL    Preamp_RGB_Drv
```

```
                LD      R14,#PSubA_BCo
                LD      R15,EP_BCutoff
                CALL    Preamp_RGB_Drv
                RET

;
;-----------------------------------------------------
;-----> Initialize IIC.bus control register
;-----------------------------------------------------
IniDDCmodule    SB1
                LD      DCCR,#10100100b         ;Enable Tx ACK signal
                                                ;Enable IIC.bus Tx/Rx int.
                                                ;Include DDC1 Tx int.
                                                ;100KHz clock speed
                CLR     DCSR0
                LD      DAR0,#0A0h              ;#0A0h=Monitor address(DDC2B)
                LD      DAR1,#6Eh               ;#6Eh=Monitor address(DDC2B+/2Bi)
                LD      DCSR0,#00010000B        ;DCSR.7/6=Master/Slave mode
                                                ;DCSR.5=Start/Stop(When write),
                                                          Busy signal status(Read)
                                                ;DCSR.4=Enable DDC module
                                                ;DCSR.3=Arbitration procedure status
                                                ;DCSR.2=Address-as-slave status
                                                ;DCSR.1=General call
                                                        ;DCSR.0=ACK bit status
                CLR     TBDR                            ;First EDID data
                LD      PP,#11h
                INC     EdidAddr
                CLR     PP
                SB0
                RET
```

```
;
;*****************************************************
;******    Read 1Byte in EEPROM        ******
;******    by S/W IIC.bus interface
;*****************************************************
;
Read1Byte:  PUSH    R0
            PUSH    R1
            PUSH    R2
            CALL    IICbus_Start                    ;IIC.bus protocol start
;
            TM      Tda9109_FGR,#01<<TdaRead_FG
            JR      NZ,LdTdaSlave
            LD      R0,#0A0h                ;Random read= #A0h -> Slave -> S -> #A1h -> READ
            CLR     R2
            JR      ShiftStart

LdTdaSlave  LD      R0,#8Dh                         ;#8C/8Dh=TDA9109 salve address
            LD      R2,#2
            AND     Tda9109_FGR,#0FFh-(01<<TdaRead_FG)
;
ShiftStart  LD      R1,#8                           ;1byte
DataShift   RLC     R0                              ;Rotate left SDADATA(=R0)
            JP      C,Data1
            AND     P3,#0FFh-(01<<SDA)              ;Data 0
            OR      P3,#01<<SCL                     ;Clock Generation.
            NOP
            NOP
            AND     P3,#0FFh-(01<<SCL)
```

```
SDA8bit     DJNZ    R1,DataShift
            AND     P3CONH,#00111111B       ;SDA(P3.7)=Input
            OR      P3,#01<<SCL             ;Acknowledge clock
            NOP
            NOP
            NOP
            TM      P3,#01<<SDA             ;Ack in ?
            JP      NZ,CommuniFail
            OR      P3CONH,#11000000B       ;SDA(P3.7)=Output
            AND     P3,#0FFh-(01<<SCL)
;
            CP      R2,#02                  ;SDACNTR=R2
            JR      UGE,DataRxStart
            CP      R2,#01
            JR      UGE,ReStartSignal
;
            LD      R0,R14                  ;SDADATA <- R14
            INC     R2                      ;SDACNTR++
            JR      ShiftStart
;
ReStartSignal CALL  IICbus_Start
            LD      R0,#0A1h                ;SDADATA <- #0A1h
            INC     R2                      ;SDACNTR++
            JR      ShiftStart
            ;
DataRxStart AND     P3CONH,#00111111B       ;SDA(P3.7)=Input
            NOP
            LD      R1,#8
RotateConti OR      P3,#01<<SCL             ;SCL <- High
```

```
            TM      P3,#01<<SDA                 ;Data value check
            JR      NZ,SetCF
            RCF
            JR      DataRotate
SetCF       SCF
DataRotate  RLC     R0
            AND     P3,#0FFh-(01<<SCL)          ;SCL <- Low
            DJNZ    R1,RotateConti              ;End of 1byte ?
            LD      ReadData,R0
            ;
No_ACK      OR      P3CONH,#11000000B           ;SDA(P3.7)=Output
            OR      P3,#01<<SDA                 ;SDA <- High(ACK=High):communication end
            OR      P3,#01<<SCL                 ;SCL <- High(9th clock)
            NOP
            NOP
            AND     P3,#0FFh-(01<<SCL)          ;SCL <- Low
GenIicStop  CALL    IICbus_Stop
            POP     R2
            POP     R1
            POP     R0
            RET
;
Data1       OR      P3,#01<<SDA
            OR      P3,#01<<SCL                 ;Clock Generation.
            NOP
            NOP
            AND     P3,#0FFh-(01<<SCL)
            JP      SDA8bit
```

```
;
ReadEepData:  PUSH   R3
              PUSH   R4
              PUSH   R5
              PUSH   R6
              CLR    R3
ContiDataRead CALL   Read1Byte
              CP     R3,#2
              JR     UGT,CmpReadData
              CP     R3,#1
              JR     UGT,Read3rd
              CP     R3,#0
              JR     UGT,Read2nd
              LD     R4,ReadData          ;R3=0
              XOR    R5,R4
              XOR    R6,R5
              INC    R3
              JR     ContiDataRead
Read2nd       LD     R5,ReadData          ;R3=1
              INC    R3
              JR     ContiDataRead
Read3rd       LD     R6,ReadData          ;R3=2
              INC    R3
              JR     ContiDataRead
;
CmpReadData   CP     R4,R5
              JR     EQ,RdDataRtn
              CP     R4,R6
              JR     EQ,RdDataRtn
```

```
                CP      R5,R6
                JR      EQ,RdDataRtn
                TM      IIC_FGR,#01<<ReRead_FG
                JR      NZ,RdDataRtn
                OR      IIC_FGR,#01<<ReRead_FG
                CLR     R3
                JR      ContiDataRead
;
RdDataRtn       AND     IIC_FGR,#0FFh-(01<<ReRead_FG)
                POP     R6
                POP     R5
                POP     R4
                POP     R3
                RET
;
;*****************************************************
;******    Write 1Byte in EEPROM          ******
;******      by S/W IIC.bus interface
;*****************************************************
;
Write1Byte:     PUSH    R0
                PUSH    R1
                PUSH    R2
                CALL    IICbus_Start                    ;IIC.bus protocol start
;
                TM      Tda9109_FGR,#01<<TdaWrite_FG
                JR      NZ,WriteTDA
                LD      R0,#0A0h                        ;Write : #A0h -> Sub -> Data
                CLR     R2
```

```
                JR      WriteStart

WriteTDA        LD      R0,#8Ch                 ;#8Ch=TDA9109 salve address
                CLR     R2
WriteStart      LD      R1,#8
TxDataShift     RLC     R0
                JR      C,TxData1
                AND     P3,#0FFh-(01<<SDA)      ;Data 0
                OR      P3,#01<<SCL             ;Clock Generation.
                NOP
                NOP
                AND     P3,#0FFh-(01<<SCL)
Tx1Byte         DJNZ    R1,TxDataShift          ;Acknowledge check
;
                AND     P3CONH,#00111111B       ;SDA(P3.7)=Input
                OR      P3,#01<<SCL             ;Acknowledge clock
                NOP
                NOP
                NOP
                TM      P3,#01<<SDA             ;Ack in ?
                JR      NZ,CommuniFail
;
NextBWrite      OR      P3CONH,#11000000B       ;SDA(P3.7)=Output
                AND     P3,#0FFh-(01<<SCL)
;
                CP      R2,#2
                JR      UGE,EepWriteEnd
                CP      R2,#1
                JR      UGE,DataTxStart
```

```
                LD      R0,R14                          ;SDADATA <- R14(Address)
                INC     R2                              ;SDACNTR++
                JR      WriteStart
                ;
DataTxStart     LD      R0,R15                          ;SDADATA <- R15(Data)
                INC     R2                              ;SDACNTR++
                JR      WriteStart
                ;
TxData1         OR      P3,#01<<SDA
                OR      P3,#01<<SCL                     ;Clock Generation.
                NOP
                NOP
                AND     P3,#0FFh-(01<<SCL)
                AND     P3,#0FFh-(01<<SDA)
                JR      Tx1Byte
;
CommuniFail     OR      P3CONH,#11000000B               ;SDA(P3.7)=Output
                OR      IIC_FGR,#01<<CommFail_FG
                JP      GenIicStop
EepWriteEnd     AND     IIC_FGR,#0FFh-(01<<CommFail_FG)
                JP      GenIicStop


;
;*****************************************************
;******      Write 4Byte in EEPROM       ******
;******        by S/W IIC.bus interface
;*****************************************************
;
WriteNByte:     PUSH    R0
```

```
                PUSH    R1
                PUSH    R2
                CALL    IICbus_Start                    ;IIC.bus protocol start
;
                LD      R0,#0A0h        ;Page write= #A0h -> Word -> D1-> D2 -> D3-> D4-> STOP
                LD      R2,#6                           ;Word addr -> D1 -> D2 -> D3 -> D4
NextTx1Byte     LD      R1,#8
TxNDataShift    RLC     R0
                JR      C,TxNData1
                AND     P3,#0FFh-(01<<SDA) ;Data 0
                OR      P3,#01<<SCL                     ;Clock Generation.
                NOP
                NOP
                AND     P3,#0FFh-(01<<SCL)
TxN1Byte        DJNZ    R1,TxNDataShift
;
                OR      P3,#01<<SCL                     ;Acknowledge clock
                AND     P3CONH,#00111111B               ;SDA(P3.7)=Input
                TM      P3,#01<<SDA                     ;Ack in ?
                JR      NZ,CommuniFail                  ;Fail
                OR      P3CONH,#11000000B               ;SDA(P3.7)=Output
                AND     P3,#0FFh-(01<<SCL)
;
                LD      R0,R14                          ;SDADATA <- R14(Address)
                CP      R2,#5
                JR      ULE,DataTxNStart
                DEC     R2
                JR      NextTx1Byte
;
```

```
DataTxNStart  LD      R0,R15                        ;SDADATA <- R15(Data=#0FFh)
              DJNZ    R2,NextTx1Byte
              JR      EepWriteEnd                   ;Stop condition
;
TxNData1      OR      P3,#01<<SDA
              OR      P3,#01<<SCL                   ;Clock Generation.
              NOP
              NOP
              AND     P3,#0FFh-(01<<SCL)
              AND     P3,#0FFh-(01<<SDA)
              JR      TxN1Byte
;
WriteCycle:   SB0
              CALL    Write1Byte
              TM      Tda9109_FGR,#01<<TdaWrite_FG
              JR      NZ,ChkReWrite
              CLR     DLY1mSR
              CALL    WriteWait                     ;Check write cycle(Typ=6ms, Max=10ms)
;
ChkReWrite    TM      IIC_FGR,#01<<ReWrite_FG       ;ReWrite ?
              JR      NZ,ClrReWrite
              TM      IIC_FGR,#01<<CommFail_FG
              JR      Z,ClrReWrite
              OR      IIC_FGR,#01<<ReWrite_FG
              JR      WriteCycle
;
ClrReWrite    AND     IIC_FGR,#0FFh-(01<<ReWrite_FG)
              AND     IIC_FGR,#0FFh-(01<<CommFail_FG)
WrCycleRtn    AND     Tda9109_FGR,#0FFh-(01<<TdaWrite_FG)
```

```
                RET
;
;
;//*****************************************************************************
;//**    Title : Waiting for write time
;//*****************************************************************************

WriteWait:      PUSH    R0                          ;Check write cycle
                PUSH    R1
IICbusRestart   CALL    IICbus_Start                ;IIC.bus protocol start
;
                LD      R0,#0A0h
                LD      R1,#8                       ;1byte
SlaveA0h        RLC     R0                          ;Rotate left SDADATA(=R0)
                JP      C,ACKData1
                AND     P3,#0FFh-(01<<SDA)          ;Data 0
                OR      P3,#01<<SCL                 ;Clock Generation.
                NOP
                NOP
                AND     P3,#0FFh-(01<<SCL)
TxA0hData       DJNZ    R1,SlaveA0h
                AND     P3CONH,#00111111B           ;SDA(P3.7)=Input
                OR      P3,#01<<SCL                 ;Acknowledge clock
                NOP
                NOP
                NOP
                TM      P3,#01<<SDA                 ;Ack in ?
                JR      NZ,RechkWrCycle
EndWrCycle      AND     P3,#0FFh-(01<<SCL)
```

```
                OR      P3CONH,#11000000B       ;SDA(P3.7)=Output
                CALL    IICbus_Stop
                POP     R1
                POP     R0
                RET
;
RechkWrCycle    CP      DLY1mSR,#10             ;Over 10ms ?
                JR      UGE,EndWrCycle
                AND     P3,#0FFh-(01<<SCL)
                OR      P3CONH,#11000000B       ;SDA(P3.7)=Output
                CALL    IICbus_Stop
                JR      IICbusRestart
;
ACKData1        OR      P3,#01<<SDA
                OR      P3,#01<<SCL             ;Clock Generation.
                NOP
                NOP
                AND     P3,#0FFh-(01<<SCL)
                JR      TxA0hData


;
;//              10ms delay routine
;
TimeDelay       PUSH    R4
                PUSH    R5
                LD      R4,#9
WaitLoop0       LD      R5,#250                 ;Write-waiting time=10ms
WaitLoop1       NOP
                NOP
```

```
            DJNZ    R5,WaitLoop1
ContiDec    DJNZ    R4,WaitLoop0
            POP     R5
            POP     R4
            RET

;-----------------------------------------------------

; < EDID Data >
DDCDUMP:    LD      PP,#11h
            LDW     RR2,#DDCData
            LD      R4,#00h                     ;R4=Address(00h-7Fh:128-Byte)
WriteEDID   LDCI    R5,@RR2                     ;R5=Data
            LD      @R4,R5
            INC     R4
            CP      R4,#80h
            JR      ULT,WriteEDID
            CLR     PP
            RET
            ;------------------------
DDCData     DB      00h,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,00h
            DB      4CH,2DH,70H,4DH,00H,00H,00H,00H
            DB      0AH,05H,01H,00H,2EH,1FH,17H,71H,0E8H,07H,65H,0A0H,57H,46H,9AH,26H
            DB      10H,48H,4CH,0FFH,0FEH,00H,01H,01H,01H,01H,01H,01,01H,01H,01H,01H
            DB      01H,01H,01H,01H,01H,01H,68H,29H,00H,80H,51H,00,24H,40H,30H,90H
            DB      33H,00H,32H,0E6H,10H,00H,00H,18H,01H,01H,01H,01H,01H,01H,01H,01H
            DB      01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H
            DB      01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H
            DB      01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,00H,0BAH
```

```
; < EDID Data >
EDIDtoRAM:  PUSH   R0                          ;Write EDID(128-btye) to EEPROM page0
            PUSH   R1
            PUSH   R2
            PUSH   PP
            LD     PP,#11h
            LD     R4,#00h                     ;R4=RAM address(50h-CFh:128-Byte)
            LD     R14,#00h                    ;R14=Start address of EDID
;
            CALL   IICbus_Start                ;IIC.bus protocol start
;
            LD     R0,#0A0h                    ;Sequential read operation
            CLR    R2                      ;  <= #A0h(Page0) -> Word -> S -> #A1h -> READ....
Shift1Byte  LD     R1,#8                       ;1byte
RotateData  RLC    R0                          ;Rotate left SDADATA(=R0)
            JP     C,SeqData1
            AND    P3,#0FFh-(01<<SDA)          ;Data 0
            OR     P3,#01<<SCL                 ;Clock Generation.
            NOP
            NOP
            AND    P3,#0FFh-(01<<SCL)
Chk1ByteEnd DJNZ   R1,RotateData
            AND    P3CONH,#11110011B           ;SDA(P3.5)=Input
            OR     P3,#01<<SCL                 ;Acknowledge clock
            NOP
            NOP
            NOP
```

```
            TM      P3,#01<<SDA                 ;Ack in ?
            JP      NZ,CommuniFail
            OR      P3CONH,#00001100B           ;SDA(P3.5)=Output
            AND     P3,#0FFh-(01<<SCL)
;
            CP      R2,#02                      ;SDACNTR=R2
            JR      UGE,DataRx
            CP      R2,#01
            JR      UGE,ReStart
;
            LD      R0,R14                      ;SDADATA <- R14
            INC     R2                          ;SDACNTR++
            JR      Shift1Byte
;
ReStart     CALL    IICbus_Start
            LD      R0,#0A1h                    ;SDADATA <- #0A1h
            INC     R2                          ;SDACNTR++
            JR      Shift1Byte
            ;
DataRx      AND     P3CONH,#11110011B           ;SDA(P3.5)=Input
            NOP
            LD      R1,#8
DataRead    OR      P3,#01<<SCL                 ;SCL <- High
            TM      P3,#01<<SDA                 ;Data value check
            JR      NZ,SetCFlag
            RCF
            JR      RxRotate
SetCFlag    SCF
RxRotate    RLC     R0
```

```
                AND     P3,#0FFh-(01<<SCL)          ;SCL <- Low
                DJNZ    R1,DataRead                 ;End of 1byte ?
;
                OR      P3CONH,#00001100B           ;SDA(P3.5)=Output
                AND     P3,#0FFh-(01<<SDA)          ;ACK generation
                OR      P3,#01<<SCL                 ;SCL <- High(9th clock)
                NOP
                NOP
                AND     P3,#0FFh-(01<<SCL)          ;SCL <- Low
                LD      @R4,R0                      ;R4=50h~CFh, R0=Read data
                INC     R4
                CP      R4,#80h                     ;00~7Fh : EDID
                JR      ULE,DataRx
                POP     PP
                JP      No_ACK                      ;Communication stop
;
SeqData1        OR      P3,#01<<SDA
                OR      P3,#01<<SCL                 ;Clock Generation.
                NOP
                NOP
                AND     P3,#0FFh-(01<<SCL)
                JP      Chk1ByteEnd
;
;*****************************************************
;******          KB2511  Initializing        *******
;*****************************************************
InitialKB2511:  CLR     R14                         ;R14=Sub address
Conti2511Ini    LDW     RR2,#TdaFRunTbl
                ADD     R3,R14
```

```
            LDC     R15,@RR2                        ;R15=Data
            OR      Tda9109_FGR,#01<<TdaWrite_FG
            CALL    WriteCycle
            INC     R14
            CP      R14,#0Fh                        ;00-0Fh ?
            JR      ULE,Conti2511Ini
            RET

;// KB2511 Default Data
;H_Duty(40) / H Posi(40) / Free Run(00) / HFocus(90)
;HFocusKey(10) / Vramp(C0) / VPosi(40) / S Correct(20)
;C Correct(20) / Keystone(A0) / EW Size(C0) / B Plus(40)
;V Moire(00) / Side Pin(A0) / Parallel(A0) / VFocus(20)
;;
TdaFRunTbl  DB      4Bh,40h,15h,9Fh,14h,0C0h,40h,16h        ;KB2511(0-7)
            DB      32h,0A0h,0C0h,30h,00h,0A0h,0A0h,20h     ;48KHz free running
```

17 DDC MODULE

OVERVIEW

The KS88C6216/C6224/C6232 microcontroller supports the DDC (Display Data Channel) interface. A pair of serial data (SDA0) and serial clock (SCL0) line (except DDC1 mode) is provided to carry information between the master and peripheral that are connected to the bus. The SDA0 and SCL0 lines are bi-directional. The DDC1 mode uses vertical sync input at the Vsync-I or VCLK (VCLK is input-only). DDC1 is implemented physically using VCLK input and SDA0 output.

Protocols for the DDC2B, DDC2Bi, and DDC2B+ are supported in hardware by multi-master IIC-bus logic and in software by the EDID (Extended Display Identification) and VDIF (Video Display Interface) formats.

To control DDC interface, you write values to the following registers:

— DDC Control Register, DCON
— DDC Clock Control Register, DCCR
— DDC Control/Status Registers 0,1, DCSR0,1
— DDC Data Shift Register, DDSR
— DDC Address Registers 0,1, DAR0,1
— Transmit Pre-buffer Data Register, TBDR
— Receive Pre-buffer Data Register, RBDR

DDC CONTROL REGISTER (DCON)

The programmable DCON register to control the DDC is located at E9H in set 1, bank 1. It is read/write addressable. Only four bits are mapped in this register.

The DCON.0 setting lets you detect falling edges at the serial clock, SCL0. If the DCON.0 is set to "0", the SCL0 (serial clock) is still high after reset (when read), or the bit can be cleared by S/W written "0" (when write). If the DCON.1 is set to "1", falling edge is detected at SCL0 pin after RESET or after this bit is cleared by S/W.

NOTE

When the DDC interrupt is occurred, SCL0 line is not pull-down at the following cases:

— DDC1 mode

— Data buffer empty status bit, DCSR1.1 is "0" when DCON.3 is "1" (only slave mode).

The DCON.1 setting lets you select normal IIC-bus interface mode or DDC1 transmit mode. If you select normal IIC-bus interface mode (DCON.1 = "0"), SCL0 pin is selected for clock line and the SCL0 falling edge (SCLF) interrupt is disabled. Or if you select DDC1 transmit mode (DCON.1 = "1"), VCLK pin is selected for clock line and the SCLF interrupt is enable.

The DCON.2 is a DDC address match bit and read-only. When the received DDC address matches to DAR0 register, DCON.2 is "1". And when it is start, stop or reset condition, DCON.2 is "0". To enable transmit or receive pre-buffer data register, DCON.3 is used. When the transmit or receive pre-buffer data register is not used, DCON.3 is "0" (normal IIC-bus mode). DCON.3 is set by writing One to it or by reset. If DCON.3 is "1", the transmit or receive pre-buffer data register is enable.

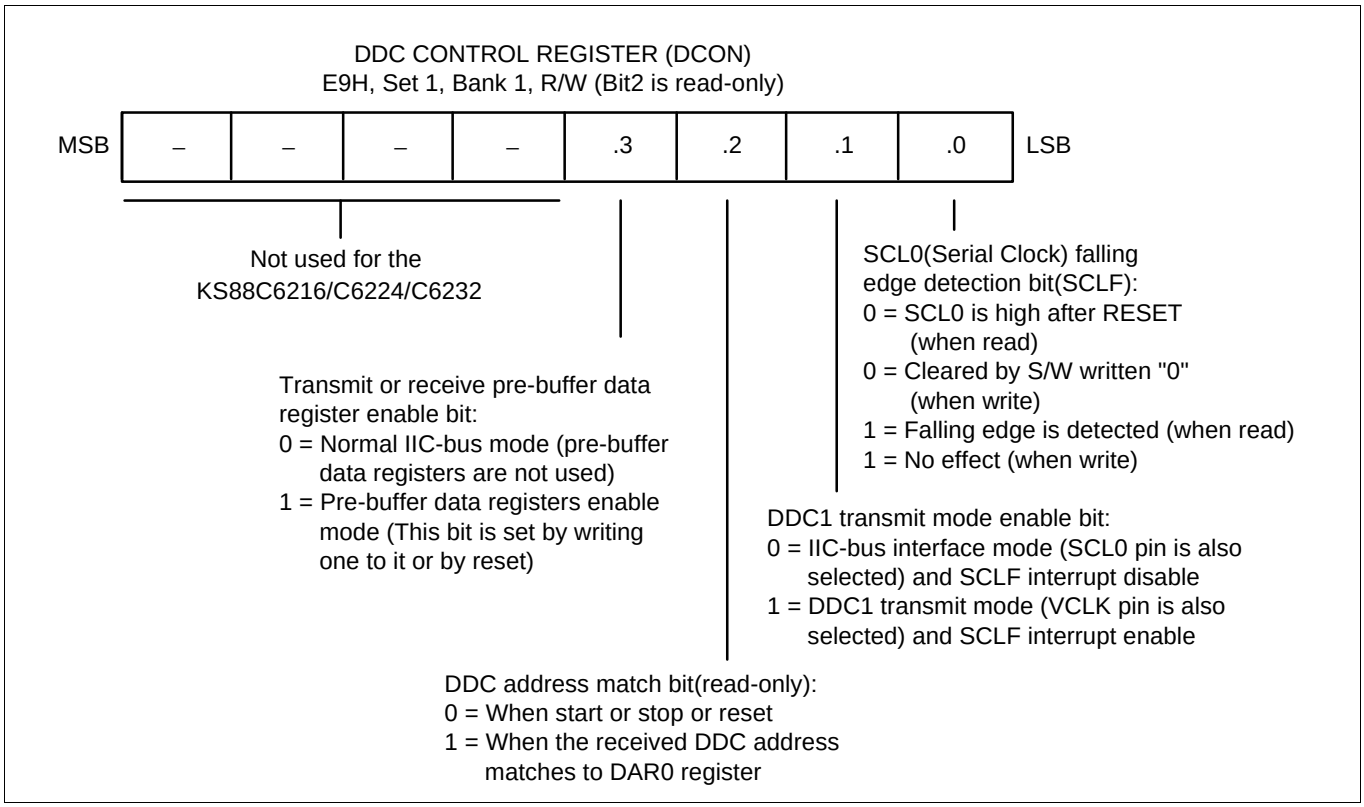


Figure 17-1. DDC Control Register (DCON)

DDC Clock Control Register (DCCR)

The DDC clock control register, DCCR, is located at EBH in set 1, bank 1. It is read/write addressable. DCCR settings control the following functions:

— CPU acknowledge signal (ACK) enable or suppress

— DDC clock source selection (f_{OSC}/16 or f_{OSC}/512)

— DDC interrupt enable or disable

— DDC interrupt pending control

— 4-bit prescaler for the serial clock (SCL0)

When DCCR.7 bit is set to "1", it is enable to acknowledgment signal. DCCR.6 is bit for transmit clock source selection by f_{OSC}/16 or f_{OSC}/512. DCCR.3–DCCR.0 bits (CCR3–CCR0) are 4-bit prescaler for the transmit clock (SCL0). The SCL0 clock may be "Stretched" if a slow slave device holds the clock for clock synchronization.

In the KS88C6216/C6224/C6232 interrupt structure, the DDC interrupt is assigned level IRQ2, vector F8H. To enable this interrupt, you set DCCR.5 to "1". Program software can then poll the DDC interrupt pending bit(DCCR.4) to detect DDC interrupt request. When the CPU acknowledges the interrupt request from the DDC, the interrupt service routine must clear the interrupt pending condition by writing a "0" to DCCR.4.

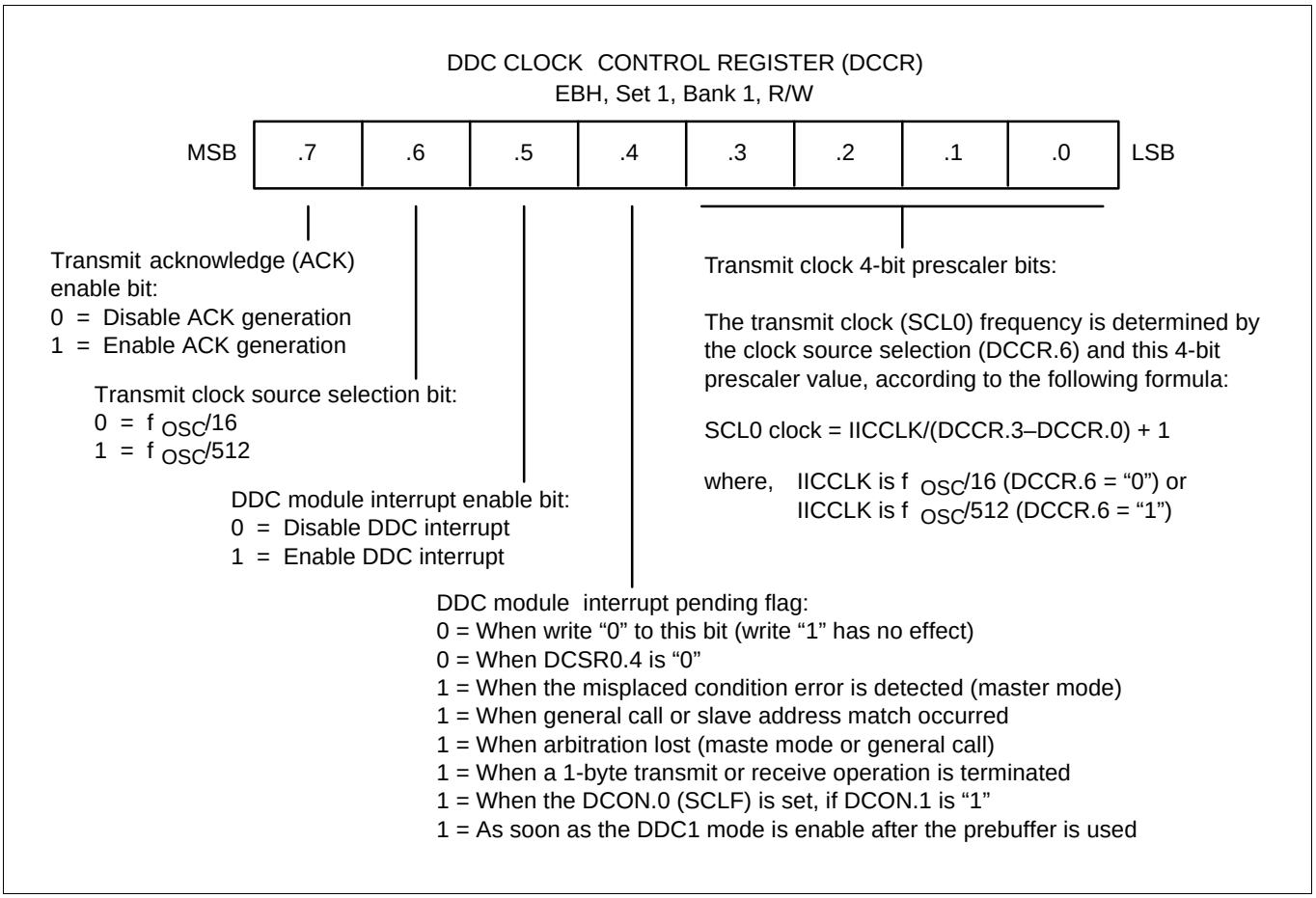


Figure 17-2. DDC Clock Control Register (DCCR)

Table 17-1. Sample Timing Calculations for the DDC Transmit Clock (SCL0)

DCCR.3–DCCR.0 Value	IICLK (DCCR.3–DCCR.0 Settings + 1)	(f_{OSC} = 8 MHz) DCCR.6 = 0 (f_{OSC}/16) IICLK = 500 kHz	(f_{OSC} = 8 MHz) DCCR.6 = 0 (f_{OSC}/512) IICLK = 15.625 kHz
0000	IICLK/1	500 kHz	15.625 kHz
0001	IICLK/2	250 kHz	7.1825 kHz
0010	IICLK/3	116.7 kHz	5.2038 kHz
0011	IICLK/4	125 kHz	3.9063 kHz
0100	IICLK/5	100 kHz	3.1250 kHz
0101	IICLK/6	83.3 kHz	2.6042 kHz
0110	IICLK/7	71.4 kHz	2.2321 kHz
0111	IICLK/8	62.5 kHz	1.9531 kHz
1000	IICLK/9	55.6 kHz	1.7361 kHz
1001	IICLK/10	50 kHz	1.5625 kHz
1010	IICLK/11	45.5 kHz	1.4205 kHz
1011	IICLK/12	41.7 kHz	1.3021 kHz
1100	IICLK/13	38.5 kHz	1.2019 kHz
1101	IICLK/14	35.7 kHz	1.1160 kHz
1110	IICLK/15	33.3 kHz	1.0417 kHz
1111	IICLK/16	32.25 kHz	0.9766 kHz

DDC CONTROL/STATUS REGISTER 0 (DCSR0)

The DDC control/status register 0, DCSR0, is located at ECH in set 1, bank 1. It is read/write addressable. Although the DCSR0 register is read/write addressable, four bits are read only: DCSR0.3–DCSR0.0.

DCSR0 register settings are used to control or monitor the following functions:

— Master/slave transmit or receive mode selection
— Bus busy status flag
— DDC module enable or disable
— Failed bus arbitration procedure status flag
— Received address register match status flag
— General call (00000000B) received status flag
— Last received bit status flag (No ACK = "1", ACK = "0")

DCSR0.3 is automatically set to "1" when a bus arbitration procedure fails over serial I/O interface, while the IIC-bus is set to master mode. If slave mode is selected, DCSR0.3 is automatically set to "1" if the value of DCSR0.7–.4 are changed by program when the busy signal bit, DCSR0.5 is "1", and the address match bit, DCSR0.2 is "0". When the DDC module is transmitting a One to SDA0 line but detected a Zero from SDA0 line in master mode or general call at the slave mode, DCSR0.3 is set.

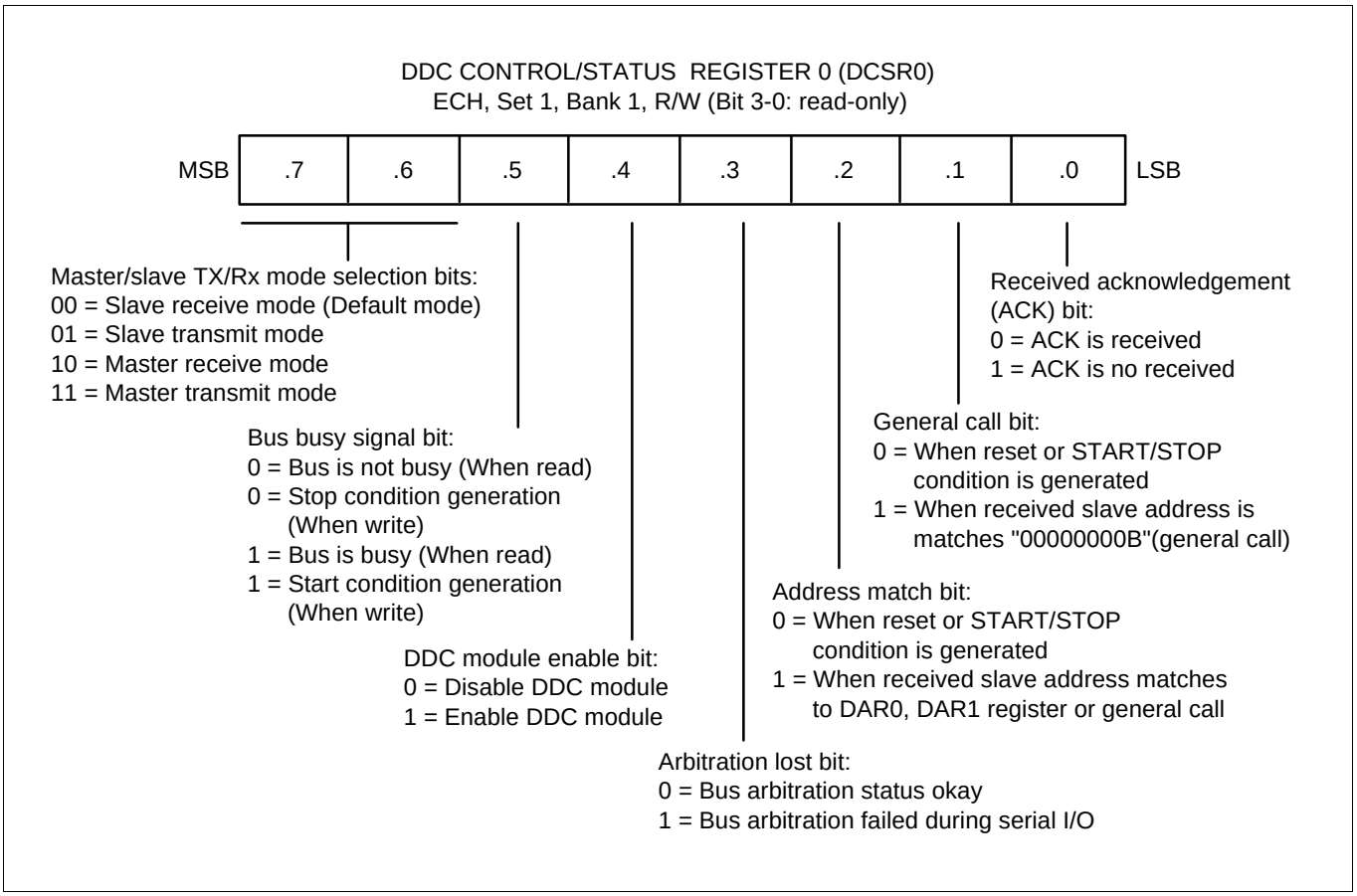


Figure 17-3. DDC Control/Status Register 0 (DCSR0)

DDC CONTROL/STATUS REGISTER 1 (DCSR1)

The DDC control/status register 1, called DCSR1, is located at EDH in set 1, bank 1. It is read/write addressable. Only three bits are mapped in this register. Two bits are read-only: DCSR1.1 and DCSR1.0.

DCSR1 register settings are used to control or monitor the following functions:

— Misplaced condition error flag
— Data buffer empty status flag
— Data buffer full status flag

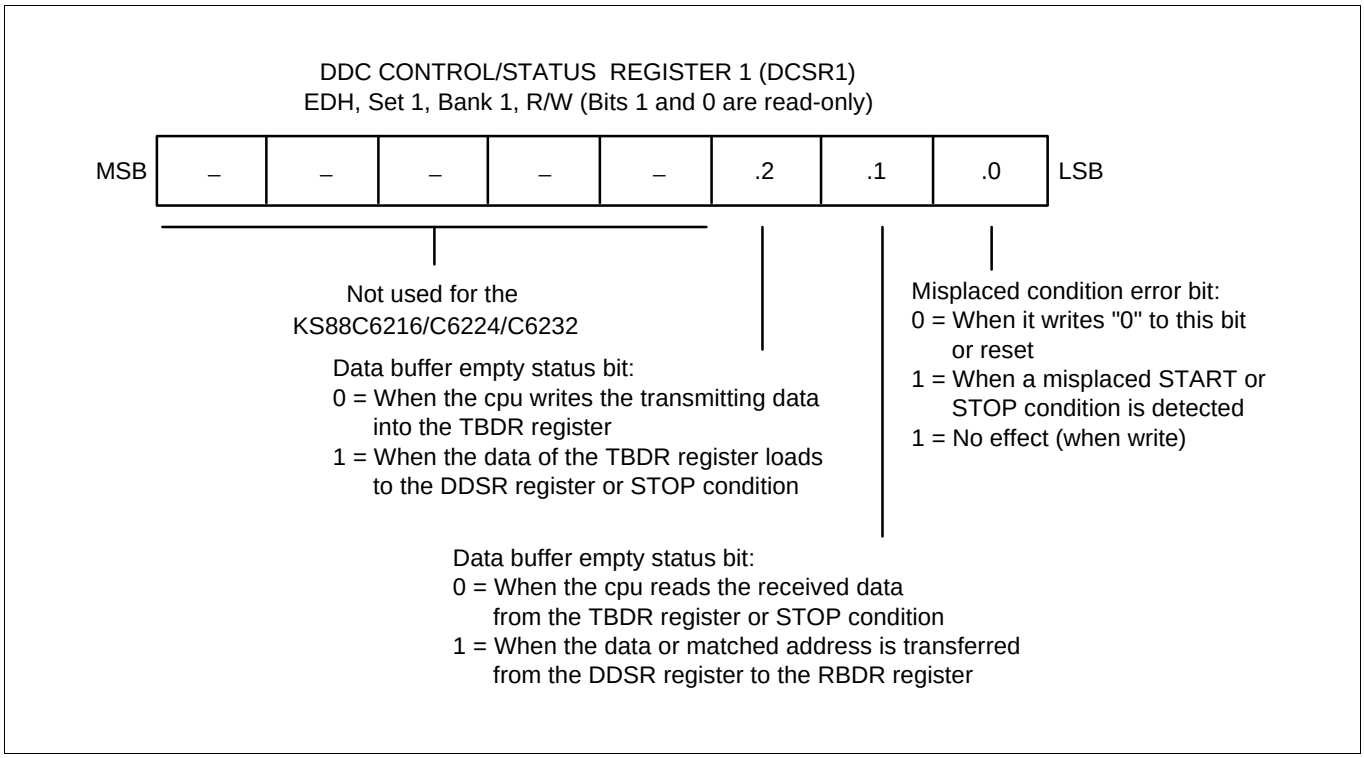


Figure 17-4. DDC Control/Status Register 1 (DCSR1)

DDC DATA SHIFT REGISTER (DDSR)

The DDC data shift register for DDC interface, called DDSR, is located at F1H in set 1, bank 1. It is read/write addressable. The transmitted data output serially from most significant bit (MSB) after writing a data to DDSR. In addition, the received data from the IIC-bus input to DDSR serially from least significant bit (LSB). DDSR register capable to write while DCSR0.4 is set to "1" and DCON.3 is set to "0", and to read anytime regardless of ICSR0.4.

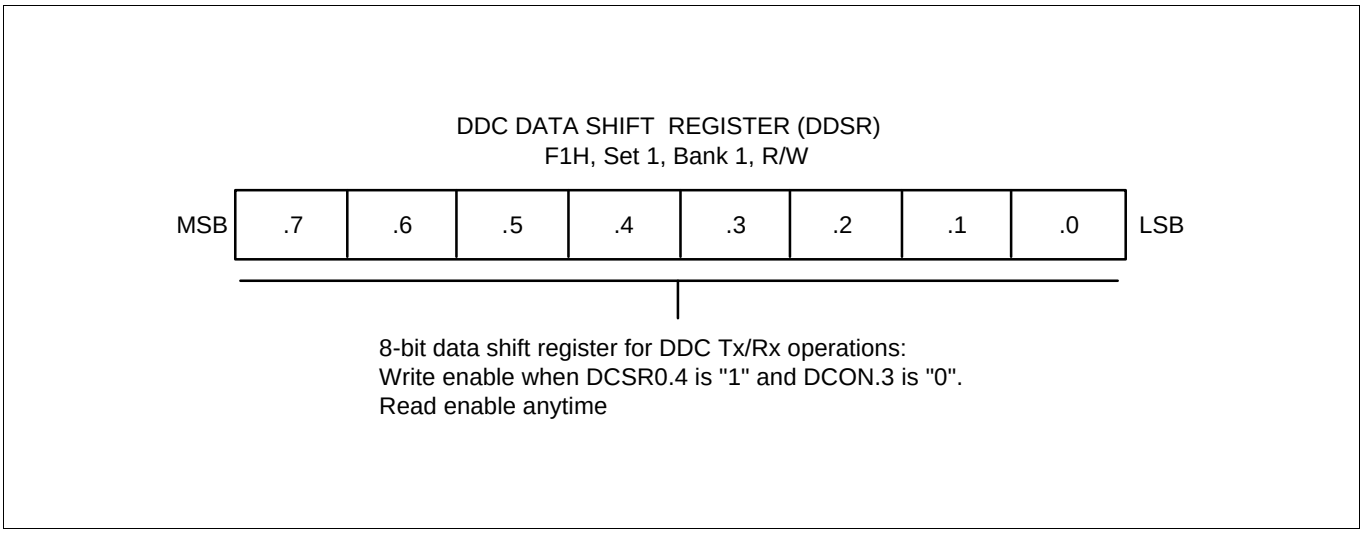


Figure 17-5. DDC Data Shift Register (DDSR)

DDC ADDRESS REGISTER 0 (DAR0)

The DDC address register 0 for DDC interface, called DAR0, is located at F2H in set 1, bank 1. It is read/write addressable. This register is consisted of 7-bit slave address latch (DAR0.0 is not mapped at the KS88C6216/C6224/C6232). DAR0 register is capable to write when DCSR0.4 is "0", and to read anytime regardless of DCSR0.4. 7-bits of the DAR0 register are operate only when receive the slave address.

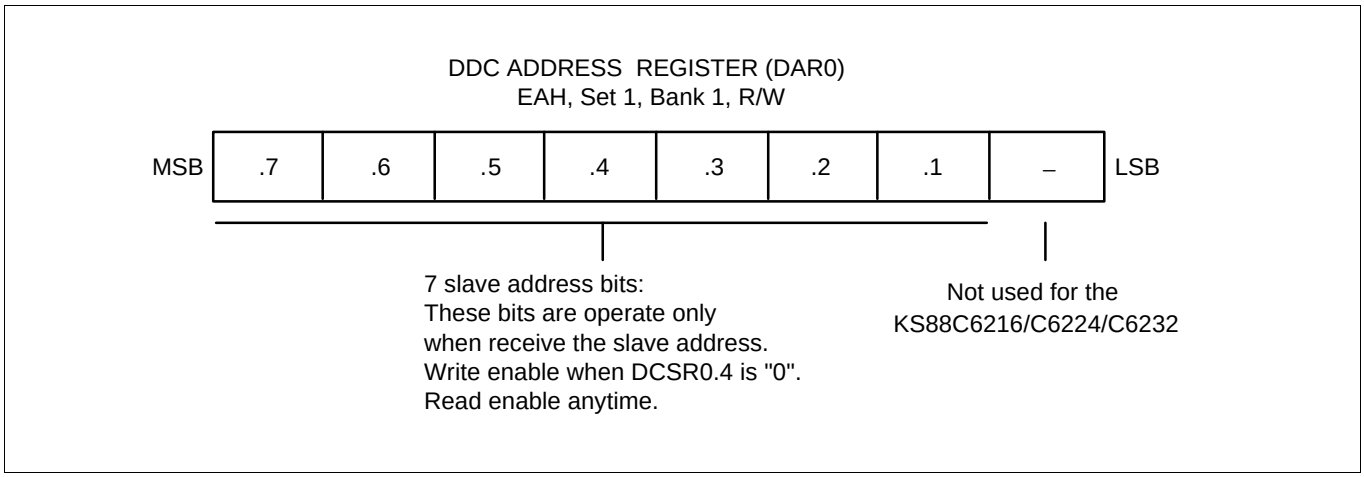


Figure 17-6. DDC Address Register 0 (DAR0)

DDC ADDRESS REGISTER 1 (DAR1)

The DDC address register 1 for DDC interface, called DAR1, is located at EEH in set 1, bank 1. It is read/write addressable. This register is consisted of 7-bit slave address latch (DAR1.0 is not mapped at the KS88C6216/C6224/C6232). DAR1 register is capable to write when DCSR0.4 is "0", and to read anytime regardless of DCSR0.4. 7-bits of the DAR1 register are operate only when receive the slave address.

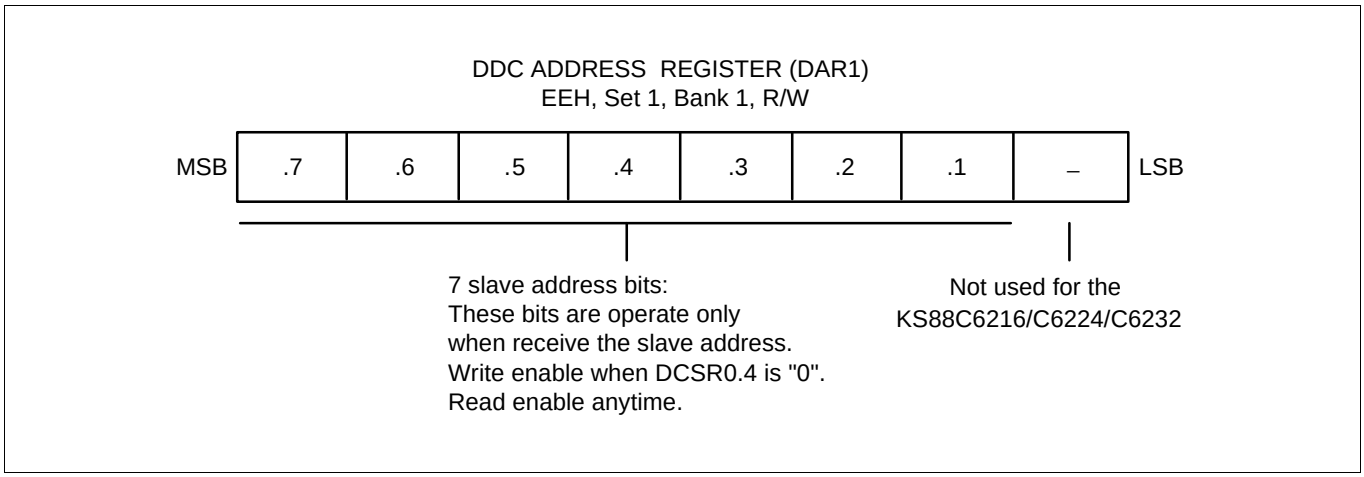


Figure 17-7. DDC Address Register 1 (DAR1)

TRANSMIT PRE-BUFFER DATA REGISTER (TBDR)

The transmit pre-buffer data register, called TBDR, is located at EFH in set 1, bank 1. It is read/write addressable. TBDR register is capable to write when DCSR0.4 is "1", and to read anytime regardless of DCSR0.4.

If DCON.3 is "1", the data written into this register will be automatically downloaded to the DDC data shift register (DDSR) and generate the DDC interrupt request when the module detects the calling address is matched and the bit 0 of the received data is "1" and when the data in the DDSR register has been transmitted with received acknowledge bit, DCSR0.0 = "0". At this interrupt service routine, the cpu must write the next data to the TBDR register to clear DCSR1.1 and for the auto downloading of data to the DDSR register after the data in the DDSR register is transmitted over again with DCSR0.0 = "0". If the master receiver dosen′t acknowledge the transmitted data, DCSR0.0 = "1", the module will release the SDA line for master to generate STOP or Repeated START conditions. The data stored in this register will not be downloaded to the DDSR register until next calling from master.

If DCON.3 is "0", the module will pull-down the SCL line in the IIC-bus interrupt service routine when the DCSR0.2 is "1". And the module will release the SCL line if the cpu writes a data to the DDSR registers and the interrupt pending bit is cleared.

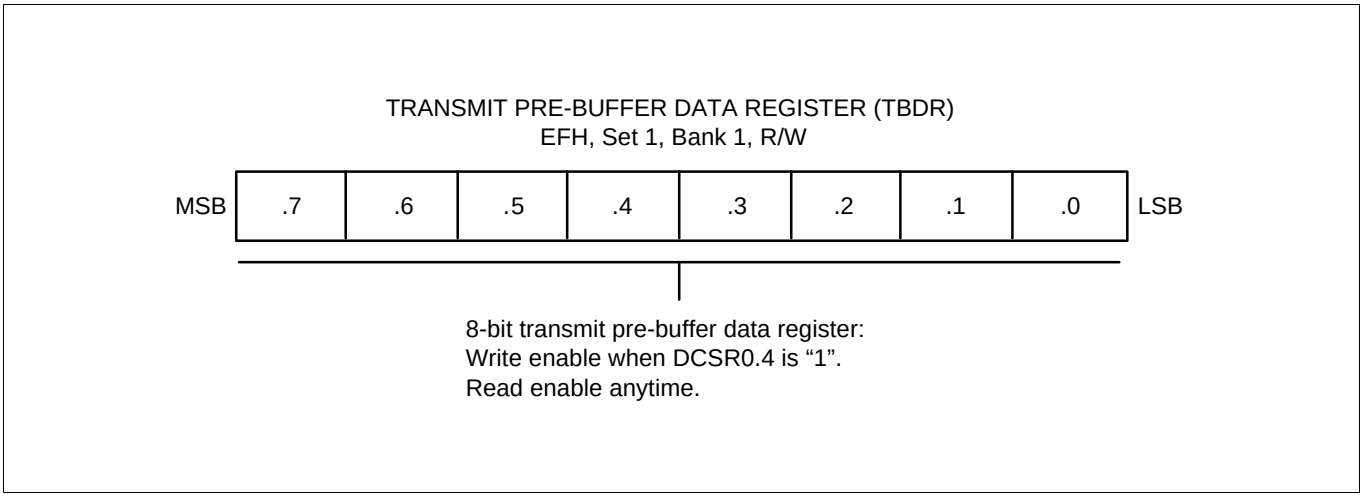


Figure 17-8. Transmit Pre-buffer Data Register (TBDR)

RECEIVE PRE-BUFFER DATA REGISTER (RBDR)

The receive pre-buffer data register, called RBDR, is located at F0H in set 1, bank 1. It is read-only addressable. RBDR register is capable to read anytime.

RBDR register will be updated after a data byte is received when the DCSR0.2 is "1"and the DCSR1.0 will be "1". The read operation of RBDR register will clear the DCSR1.0. Regardless of reading this register, the register can load the received data again and set the DCSR1.0.

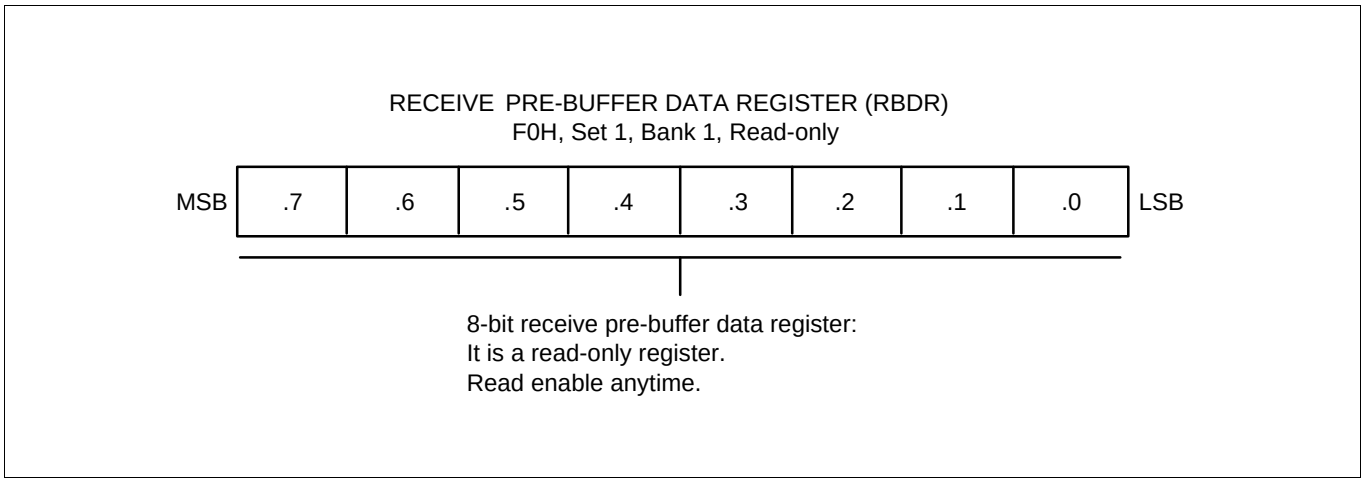


Figure 17-9. Receive Pre-buffer Data Register (RBDR)

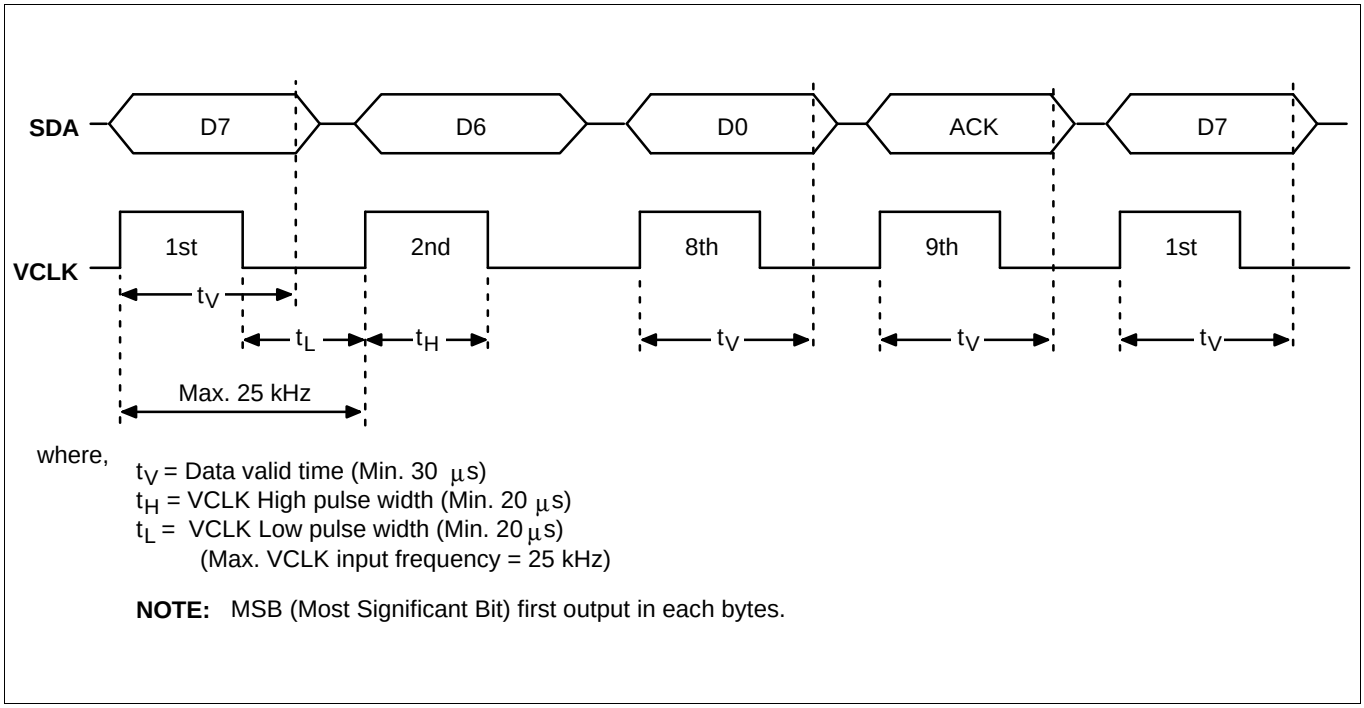


Figure 17-10. DDC1 Mode Timing Diagram (One-Byte Transfer)

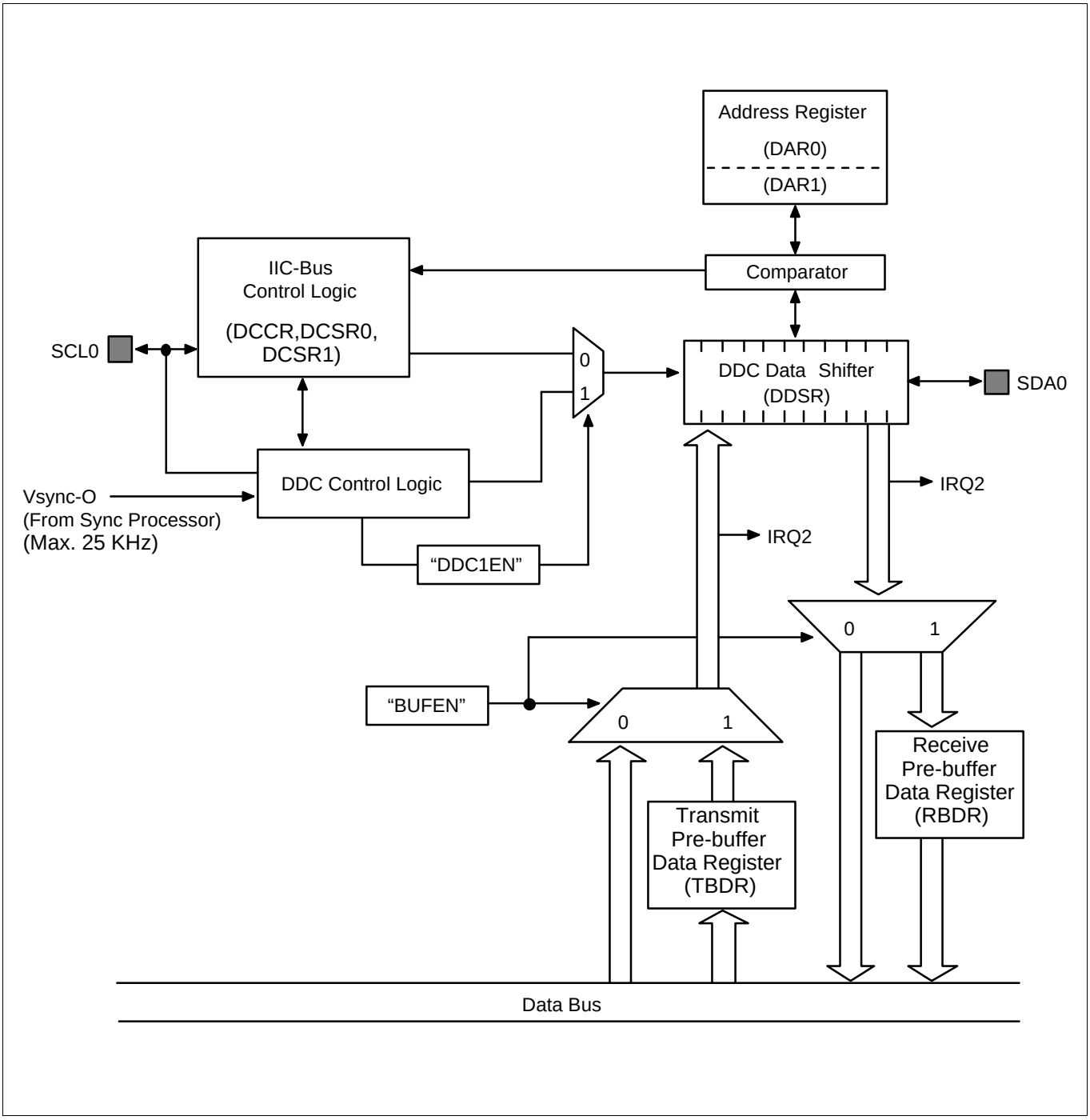


Figure 17-11. DDC Module Block Diagram

THE DDC INTERFACE

DDC2BI MODE

Overview

DDC2B capable graphic hosts have limited and mono-directional communications with the display devices. At the contrary, DDC2Bi mode s an extension of the DDC2B level in order to offer a bi-direction communication between the computer graphic host and the display device. DDC2Bi brings DDC2B+ functionality to DDC2B graphic hosts using a simple S/W driver. So DDC2Bi display device is made by simple S/W upgrade to DDC2B+ capable displays. DDC2Bi protocol relies on the DDC2B H/W definition and the Access bus messages protocol. The Graphic host behaves as an IIC single master host, and the display device behaves as an IIC slave device. The DDC2Bi is a modification of the Access bus multi-master protocol to fit single master communication.

DDC2Bi Host and Display Device

DDC2Bi host is considered as an IIC single master capable device. The virtual IIC slave address of the host is 50/51H. But DDC2Bi display device is considered as a fixed address display device (6E/6F), and uses only IIC slave mode to communicate with the host.

A display dependent devices are geographically located around the display and follow the same DDC2Bi data protocol than the display device. And fixed address IIC slave devices group all the existing stand-alone and brain-less IIC slave device. These devices can coexist and be connected to the DDC/IIC-bus.

DDC2Bi S/W Implementation

In order to describe the display that the received message is of DDC2Bi type, the source address byte bit 0 is set. And when the host expects an answer from the display, the host reads the answer message at the display device slave address 6FH. The checksum is still computed by using the 50H, virtual host address.

A null message can be defined as an Access bus message without any data byte. The null message is used in the following cases:

— To detect that the display is DDC2Bi capable by reading it at 6FH, IIC slave address.

— To describe the host that the display does not have any answer to give to the host

— The enable application report has not been sent prior application messages exchange with the host

DDC2Bi Communication

In the DDC2Bi communication, it is capable to retrials when a communication fails (bus error or bad checksum). So the host is responsible for resending its message and trying to get an answer from the display again. When the communication fail is occurred, the DDC2Bi devices must answer by the retry of host.

The DDC2Bi capable device must properly send and receive all its supported messages. This determines the maximum internal data communication buffer required size for proper display operation. If the device receive a message which size is lager than the maximum supported by the device, the message be accepted entirely by the device, but does not need to be supported internally, and then be discarded. Therefore the DDC2Bi capable device must acknowledge all received data bytes from the host.

18 IIC-BUS INTERFACE

OVERVIEW

The KS88C6216/C6224/C6232 microcontrollers support a multi-master IIC-bus serial interface. A dedicated serial data line (SDA) and a serial clock line (SCL) carry information between bus masters and peripheral devices which are connected to the IIC-bus. The SDA1 and SCL1 lines are bi-directional.

In multi-master IIC-bus mode, multiple KS88C6216/C6224/C6232 microcontrollers can receive or transmit serial data to or from slave devices. The master KS88C6216/C6224/C6232 which initiates a data transfer over the IIC-bus is responsible for terminating the transfer. Standard bus arbitration functions are supported.

To control multi-master IIC-bus operations, you write values to the following registers:

— IIC-bus control register, ICCR

— IIC-bus control/status register, ICSR

— IIC-bus Tx/Rx data shift register, IDSR

— IIC-bus address register, IAR

When the IIC-bus is free, the SDA1 and SCL1 lines are both at High level. A High-to-Low transition of SDA1 initiates a Start condition. A Low-to-High transition of SDA1 while SCL1 remains steady at High level initiates a Stop condition.

Start and Stop conditions are always generated by the bus master. A 7-bit address value in the first data byte that is put onto the bus after the Start condition is initiated determines which slave device the bus master selects. The 8th bit determines the direction of the transfer (read or write).

Every data byte that is put onto the SDA1 line must total eight bits. The number of bytes which can be sent or received per bus transfer operation is unlimited. Data is always sent most-significant bit (MSB) first and every byte must be immediately followed by an acknowledge (ACK) bit.

MULTI-MASTER IIC-BUS CONTROL REGISTER (ICCR)

The multi-master IIC-bus control register, ICCR, is located in set 1, bank 1, at address F2H. It is read/write addressable. ICCR settings control the following IIC-bus functions:

— CPU acknowledge signal (ACK) enable or suppress

— IIC-bus clock source selection (f_{OSC}/16 or f_{OSC}/512)

— Transmit/receive interrupt enable or disable

— Transmit/receive interrupt pending control

— 4-bit prescaler for the serial transmit clock (SCL)

In the KS88C6216/C6224/C6232 interrupt structure, the IIC-bus Tx/Rx interrupt is assigned level IRQ2, vector F8H. To enable this interrupt, you set ICCR.5 to “1”. Program software can then poll the IIC-bus Tx/Rx interrupt pending bit (ICCR.4) to detect IIC-bus receive or transmit requests. When the CPU acknowledges the interrupt request from the IIC-bus, the interrupt service routine must clear the interrupt pending condition by writing a “0” to ICCR.4.

The SCL frequency is determined by the IIC-bus clock source selection (f_{OSC}/16 or f_{OSC}/512) and the 4-bit prescaler value in the ICCR register (see Figure 18-1).

MULTI-MASTER IIC-BUS CONTROL REGISTER (ICCR)
E9H, Set 1, Bank 1, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

IIC-bus acknowledge (ACK) enable bit:
0 = Disable ACK generation
1 = Enable ACK generation

IIC-bus clock source selection bit:
0 = f_{OSC}/16
1 = f_{OSC}/512

IIC-bus Tx/Rx interrupt enable/disable bit:
0 = Disable interrupt
1 = Enable interrupt

IIC-bus Tx/Rx interrupt pending flag:
0 = No interrupt pending (when read)
0 = Clear pending condition (when write)
1 = Interrupt is pending (when read)
1 = No effect (when write)

IIC-bus transmit (Tx) clock prescaler:

The IIC-bus transmit clock (SCL) frequency is determined by the clock source selection (ICCR.6) and this 4-bit prescaler value, according to the following formula:

Tx clock (SCL) = IICCLK/(ICCR.3–ICCR.0) + 1

where IICCLK = f_{OSC}/16 (ICCR.6 = “0”) or
IICCLK = f_{OSC}/512 (ICCR.6 = “1”)

NOTE: A IIC-bus interrupt occurs 1) when a 1-byte transmit or receive operation is terminated, 2) when a general call or a slave address match occurs, or 3) if bus arbitration fails.

Figure 18-1. Multi-Master IIC-Bus Control Register (ICCR)

Table 18-1. Sample Timing Calculations for the IIC-Bus Transmit Clock (SCL)

ICCR.3–ICCR.0 Value	IICLK (ICCR.3–ICCR.0 Settings + 1)	(f_{OSC} = 8 MHz) ICCR.6 = 0 (f_{OSC}/16) IICLK = 500 kHz	(f_{OSC} = 8 MHz) ICCR.6 = 1 (f_{OSC}/512) IICLK = 15.625 kHz
0000	IICLK/1	400 kHz (note)	15.625 kHz
0001	IICLK/2	250 kHz	7.1825 kHz
0010	IICLK/3	116.7 kHz	5.2038 kHz
0011	IICLK/4	125 kHz	3.9063 kHz
0100	IICLK/5	100 kHz	3.1250 kHz
0101	IICLK/6	83.3 kHz	2.6042 kHz
0110	IICLK/7	71.4 kHz	2.2321 kHz
0111	IICLK/8	62.5 kHz	1.9531 kHz
1000	IICLK/9	55.6 kHz	1.7361 kHz
1001	IICLK/10	50 kHz	1.5625 kHz
1010	IICLK/11	45.5 kHz	1.4205 kHz
1011	IICLK/12	41.7 kHz	1.3021 kHz
1100	IICLK/13	38.5 kHz	1.2019 kHz
1101	IICLK/14	35.7 kHz	1.1160 kHz
1110	IICLK/15	33.3 kHz	1.0417 kHz
1111	IICLK/16	31.25 kHz	0.9766 kHz

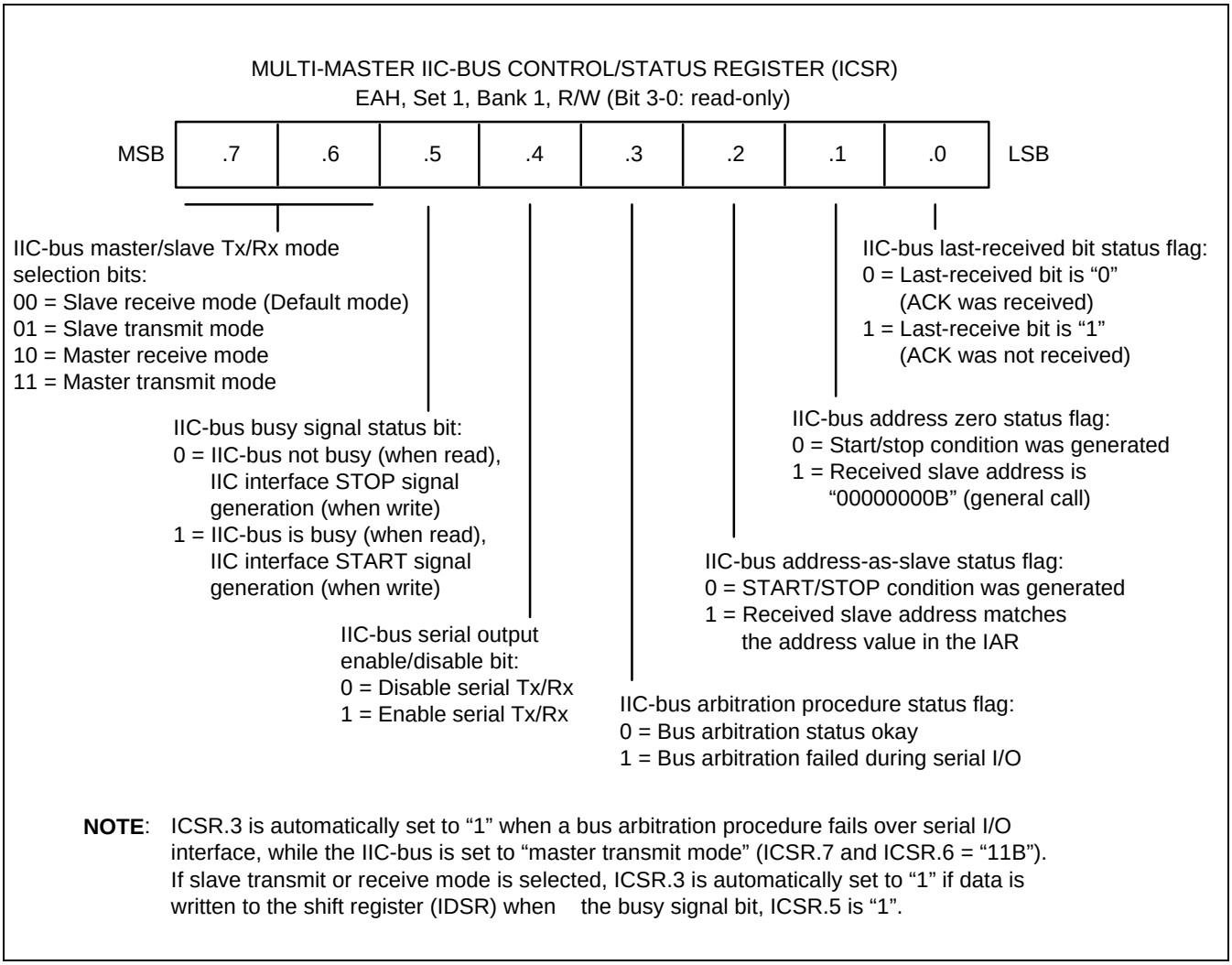
NOTE: Max. IICLK = 400 kHz.

MULTI-MASTER IIC-BUS CONTROL/STATUS REGISTER (ICSR)

The multi-master IIC-bus control/status register, ICSR, is located in set 1, bank 1, at address EAH. Four bits in this register, ICSR.3–ICSR.0, are read-only status flags.

ICSR register settings are used to control or monitor the following IIC-bus functions (see Figure 18-4):

— Master/slave transmit or receive mode selection
— IIC-bus busy status flag
— Serial output enable/disable
— Failed bus arbitration procedure status flag
— Slave address/address register match or general call received status flag
— Slave address 00000000B (general call) received status flag
— Last received bit status flag (not ACK = “1”, ACK = “0”)



NOTE: ICSR.3 is automatically set to “1” when a bus arbitration procedure fails over serial I/O interface, while the IIC-bus is set to “master transmit mode” (ICSR.7 and ICSR.6 = “11B”). If slave transmit or receive mode is selected, ICSR.3 is automatically set to “1” if data is written to the shift register (IDSR) when the busy signal bit, ICSR.5 is “1”.

Figure 18-2. Multi-Master IIC-Bus Control/Status Register (ICSR)

MULTI-MASTER IIC-BUS TRANSMIT/RECEIVE DATA SHIFT REGISTER (IDSR)

The IIC-bus data shift register, IDSR, is located in set 1, bank 1, at address ECH. In a transmit operation, data that is written to the IDSR is transmitted serially, MSB first. (For receive operations, the input data is written into the IDSR register LSB first.)

The ICSR.4 setting enables or disables serial transmit/receive operations. When ICSR.4 = “1”, data can be written to the shift register. The IIC-bus shift register can, however, be read at any time, regardless of the current ICSR.4 setting.

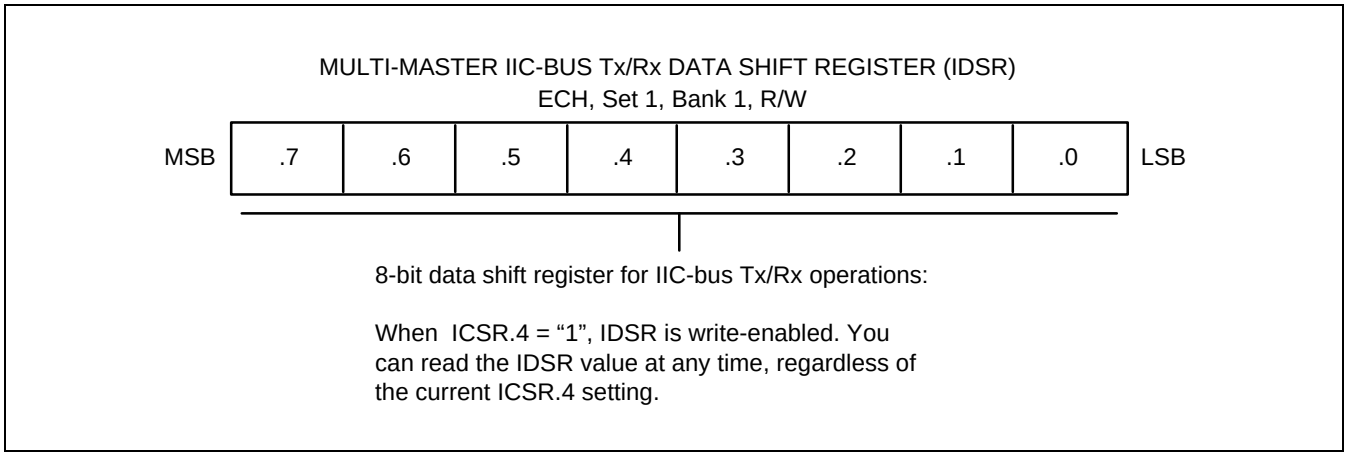


Figure 18-3. Multi-Master IIC-Bus Tx/Rx Data Shift Register (IDSR)

MULTI-MASTER IIC-BUS ADDRESS REGISTER (IAR)

The address register for the IIC-bus interface, IAR, is located in set 1, bank 1, at address EBH. It is used to store a latched 7-bit slave address. This address is mapped to IAR.7–IAR.1; bit 0 is not used (see Figure 18-4).

The latched slave address is compared to the next received slave address. If a match condition is detected, and if the latched value is 00000000B, a general call status is detected.

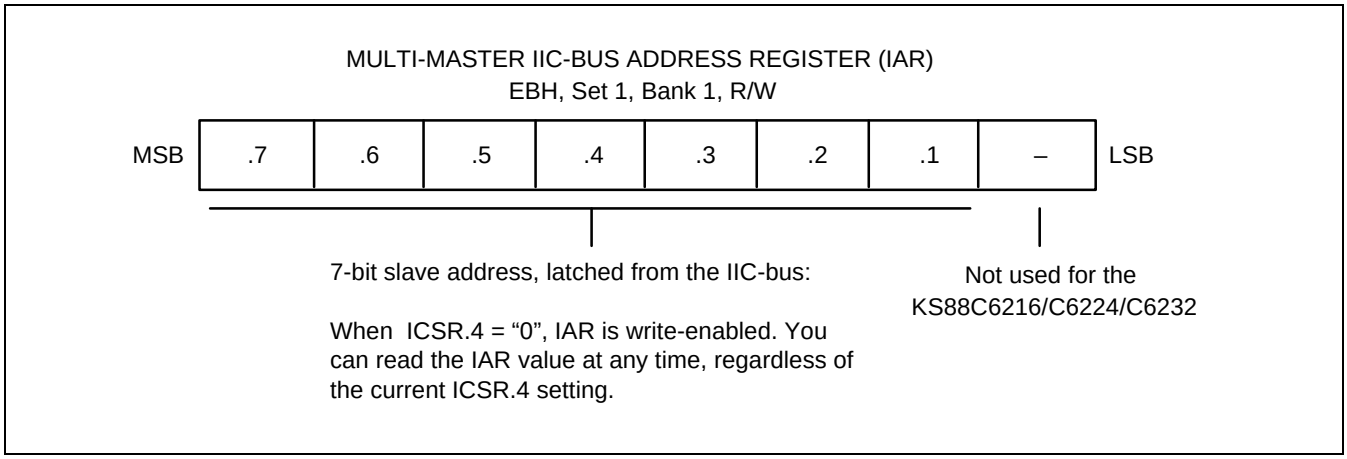
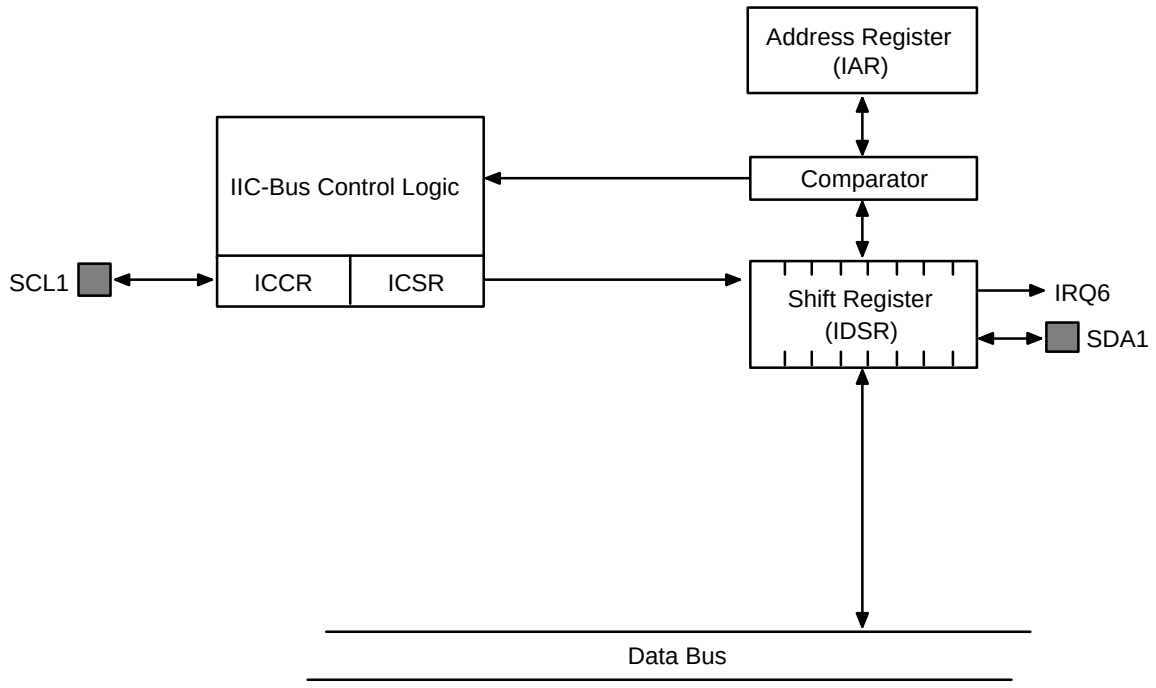


Figure 18-4. Multi-Master IIC-Bus Address Register (IAR)



NOTE: The IIC-bus interrupt (IRQ6) is generated when a 1-byte receive or transmit operation is terminated before the shift operation has been completed.

Figure 18-5. IIC-Bus Block Diagram

THE IIC-BUS INTERFACE

The KS88C6216/C6224/C6232 IIC-bus interface has four operating modes:

— Master transmitter mode

— Master receive mode

— Slave transmitter mode

— Slave receive mode

Functional relationships between these operating modes are described below.

START AND STOP CONDITIONS

When the IIC-bus interface is inactive, it is in slave mode. The interface is therefore always in slave mode when a start condition is detected on the SDA line. (A start condition is a High-to-Low transition of the SDA line while the clock signal, SCL, is High level.) When the interface enters master mode, it initiates a data transfer and generates the SCL signal.

A start condition initiates a one-byte serial data transfer over the SDA line and a stop condition ends the transfer. (A stop condition is a Low-to-High transition of the SDA line while SCL is High level.) Start and stop conditions are always generated by the master. The IIC-bus is “busy” when a start condition is generated. A few clocks after a stop condition is generated, the IIC-bus is again “free”.

When a master initiates a start condition, it sends its slave address onto the bus. The address byte consists of a 7-bit address and a 1-bit transfer direction indicator (that is, write or read). If bit 8 is “0”, a transmit operation (write) is indicated; if bit 8 is “1”, a request for data (read) is indicated.

The master ends the indicated transfer operation by transmitting a stop condition. If the master wants to continue sending data over the bus, it can the generate another start condition and another slave address. In this way, read-write operations can be performed in various formats.

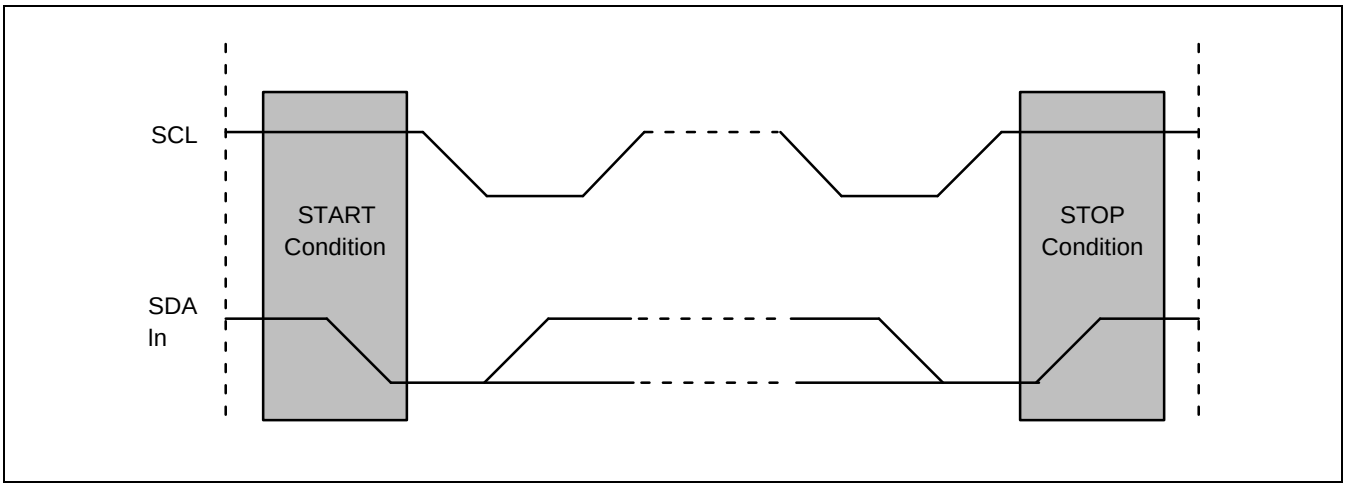


Figure 18-6. Start and Stop Conditions

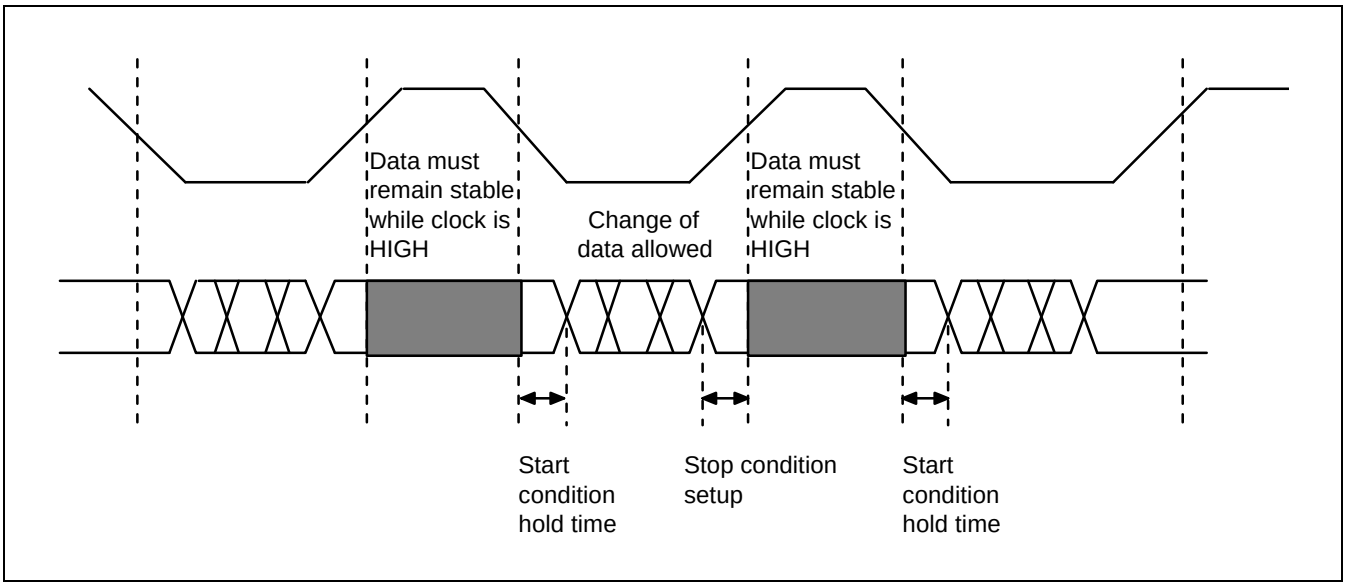
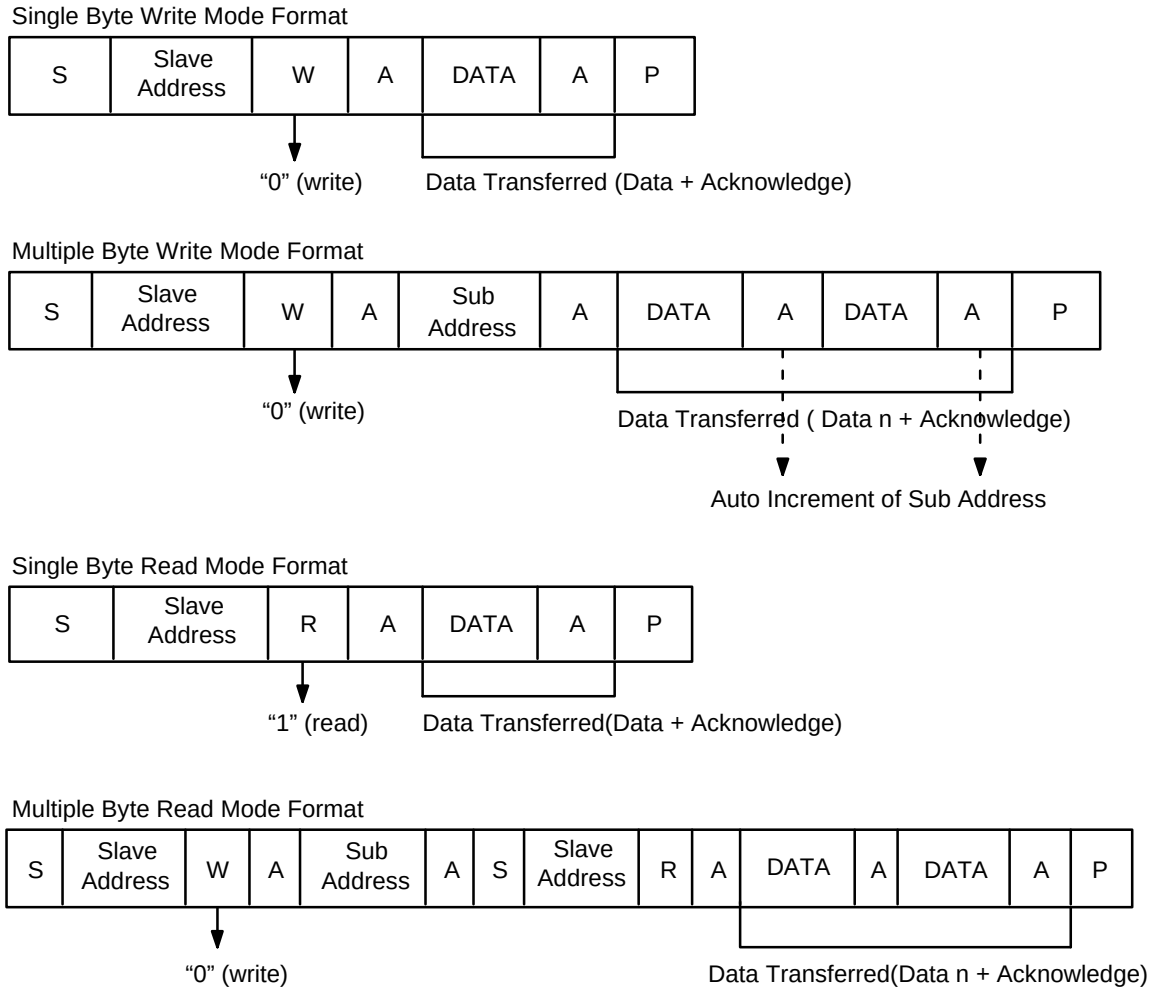


Figure 18-7. Input Data Protocol

DATA TRANSFER FORMATS

Every byte put on the SDA line must be eight bits in length. The number of bytes which can be transmitted per transfer is unlimited. The first byte following a start condition is the address byte. This address byte is transmitted by the master when the IIC-bus is operating in master mode. Each byte must be followed by an acknowledge (ACK) bit. Serial data and addresses are always sent MSB first.



NOTES:
1. S: start, A: acknowledge, P: stop
2. The "Sub Address" indicates the internal address of the slave device.

Figure 18-8. IIC-Bus Interface Data Formats

ACK SIGNAL TRANSMISSION

To complete a one-byte transfer operation, the receiver must send an ACK bit to the transmitter. The ACK pulse occurs at the ninth clock of the SCL line (eight clocks are required to complete the one-byte transfer). The clock pulse required for the transmission of the ACK bit is always generated by the master.

The transmitter releases the SDA line (that is, it sends the SDA line High) when the ACK clock pulse is received. The receiver must drive the SDA line Low during the ACK clock pulse so that SDA is Low during the High period of the ninth SCL pulse.

The ACK bit transmit function can be enabled and disabled by software (ICCR.7). However, the ACK pulse on the ninth clock of SCL is required to complete a one-byte data transfer operation.

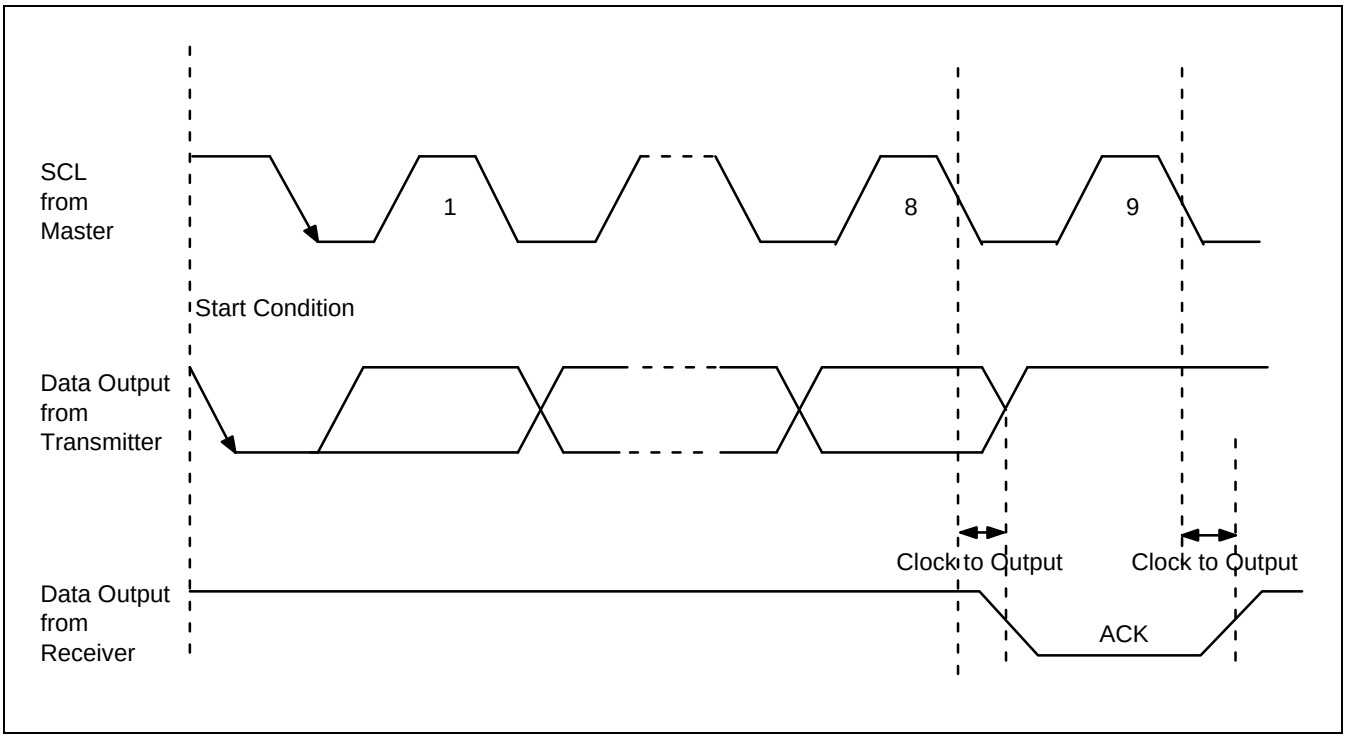


Figure 18-9. Acknowledge Response from Receiver

READ-WRITE OPERATIONS

When operating in transmitter mode, the IIC-bus interface interrupt routine waits for the master (the KS88C6216/C6224/C6232) to write a data byte into the IIC-bus data shift register (IDSR). To do this, it holds the SCL line Low prior to transmission.

In receive mode, the IIC-bus interface waits for the master to read the byte from the IIC-bus data shift register (IDSR). It does this by holding the SCL line Low following the complete reception of a data byte.

BUS ARBITRATION PROCEDURES

Arbitration takes place on the SDA line to prevent contention on the bus between two masters. If a master with a SDA High level detects another master with an SDA active Low level, it will not initiate a data transfer because the current level on the bus does not correspond to its own. The master which loses the arbitration can generate SCL pulses only until the end of the last-transmitted data byte. The arbitration procedure can continue while data continues to be transferred over the bus.

The first stage of arbitration is the comparison of address bits. If a master loses the arbitration during the addressing stage of a data transfer, it is possible that the master which won the arbitration is attempting to address the master which lost. In this case, the losing master must immediately switch to slave receiver mode.

ABORT CONDITIONS

If a slave receiver does not acknowledge the slave address, it must hold the level of the SDA line High. This signals the master to generate a stop condition and to abort the transfer.

If a master receiver is involved in the aborted transfer, it must also signal the end of the slave transmit operation. It does this by not generating an ACK after the last data byte received from the slave. The slave transmitter must then release the SDA to allow a master to generate a stop condition.

CONFIGURING THE IIC-BUS

To control the frequency of the serial clock (SCL), you program the 4-bit prescaler value in the ICCR register. The IIC-bus interface address is stored in IIC-bus address register, IAR. (By default, the IIC-bus interface address is an unknown value.)

☞ PROGRAMMING TIP — Programming the IIC-Bus Interface

This example shows one way to program the IIC-bus module to sample specifications. The program parameters are as follows:

— Main crystal oscillation frequency is 8 MHz

— IIC-bus clock frequency (SCL) is 100 kHz

— The following registers are assigned for the program values:

```
Register 70H      Tx_DATA                    ;  Transmitted data
Register 71H      Rx_Data                    ;  Received data
Register 72H      SubAddr                    ;  Sub-address
Register 73       eepromf_ref                ;  Register for external EEPROM
0                 restartf
1                 eepdataRxf
2                 eepdataTxf
3                 eepreadRxf
4                 eepreadf
5                 eepsubAddrf
```

Additional address information:

— Slave address: #0A0H (external EEPROM-type)

— Sub-address: #10H (data address pointer)

— IIC-bus data: #20H

```
                •
                •
                •
IICunitTest:
                SB1                                  ;  Select bank 1
                TM      eepromf_ref,#01<<eepdataRxf  ;  eepreadRxf - "1"?
                JR      NZ,DataCompare
ChkACKbit:
                TM      ICSR,#01<<bit0               ;  LRB (ACK bit ) = "1"?
                JR      Z,EEPsubAddr
                LD      eepromf_reg,#00H
                OR      eepromf_reg,#01<<restartf    ;  restartf ← "1"
                AND     ICSR,#11011111B              ;  Stop
                JP      IICrtn
EEPsubAddr:
                TM      eepromf_reg,#01<<eepsubAddrf ;  IIC1 → EEPROM (write)
                JR      NZ,EEPsubAddr
                LD      IDSR,#SubADDR                ;  Call EEPROM sub-address
                OR      eepromf_reg,#01<<eepsubAddrf ;  eepsubAddrf ← "1"
                JP      IICrtn
```

☞ PROGRAMMING TIP — Programming the IIC-Bus Interface (Continued)

```
EEPdataTx:
        TM      eeprom_reg,#01<<eepreadf
        JR      NZ,EEPdataRx
        TM      eepromf_reg,#01<<eepdataTxf
        JR      NZ,Tx1Byte
        LD      IDSR,TxDATA                     ;  Byte Tx (IIC1 → EEPROM)
        OR      eepromf_reg,#01<<eepdataTxf
        JP      IICrtn

Tx1Byte:
        AND     IDSR,#11011111B                 ;  Stop
        AND     eepromf_reg,#0FFH-(01<<eepdataTxf)
        OR      eepromf_reg,#01<<restartf       ;  restartf ← "1"
        OR      eepromf_reg,#01<<eepreadf       ;  eepreadf ← "1"
        AND     eepromf_reg,#0FFH-(01<<eepsubAddrf
        JP      IICrtn
EEPreadRx:
        TM      eepromf_reg,#01<<eepreadRxf;    ;  eepreadRxf = "1"?
        JR      NZ,EEPdataRx
        LD      IDSR,#0A9H                      ;  Read mode
        LD      ICSR,#9FH                       ;  IIC1 = Master Rx
        OR      eeprom_reg,#01<<eepreadRxf      ;  eepreadRxf ← "1"
        OR      ICSR,#00100000B                 ;  Repeat start
        JP      IICrtn
EEPdataRx:
        TM      eepromf_reg,#01<<eepdataRxf     ;  eepdataRxf = "1"?
        JR      NZ,DataCompare
        OR      eepromf_reg,#01<<eepdataRxf     ;  eepdataRxf ← "1"
        AND     ICCR,#01111111B                 ;  ACK = High
        JP      IICrtn

DataCompare:
        LD      RxDATA,IDSR
        CP      TxDATA,RxDATA                   ;  TxDATA = RxDATA?
        JP      NE,IICrtn
EEPdataDisp:
        INC     SubADDR                         ;  00H–FFH
        DEC     TxDATA                          ;  FFH–00H
        LD      eepromf_reg,#00H                ;  eepromf_reg clear
        OR      eepromf_reg,#01<<restartf       ;  restartf ← "1"
        AND     ICSR,#11011111B                 ;  Stop
        JP      IICrtn
```

☞ PROGRAMMING TIP — Programming the IIC-Bus Interface (Concluded)

```
WaitIIC:
        LD      R4,#0FFH
IICwait0:
        LD      R5,#0FH
IICwait1:
        NOP
        NOP
        DJNZ    R5,IICwait1
        DJNZ    R4,IICwait0
IICrtn:
        RET
        •
        •
        •
```

19 ELECTRICAL DATA

OVERVIEW

In this section, KS88C6216/C6224/C6232 electrical characteristics are presented in tables and graphs. The information is arranged in the following order:

— Absolute maximum ratings
— D.C. electrical characteristics
— Data retention supply voltage in stop mode
— Stop mode release timing when initiated by a reset
— I/O capacitance
— A/D Converter electrical characteristics
— A.C. electrical characteristics
— Input timing measurement points for P0.0–P0.2, TM0CAP, and TM1CK
— Oscillation characteristics
— Oscillation stabilization time
— Clock timing measurement points for X_{IN}
— Schmitt trigger characteristics

Table 19-1. Absolute Maximum Ratings

(T_A = 25°C)

Parameter	Symbol	Conditions	Rating	Unit
Supply voltage	V_{DD}	–	– 0.3 to + 6.5	V
Input voltage	V_{I1}	Type C (n-channel, open-drain)	– 0.3 to + 7.0	V
	V_{I2}	All port pins except V_{I1}	– 0.3 to V_{DD} + 0.3	
Output voltage	V_O	All output pins	– 0.3 to V_{DD} + 0.3	V
Output current High	I_{OH}	One I/O pin active	– 10	mA
		All I/O pins active	– 60	
Output current Low	I_{OL}	One I/O pin active	+ 30	mA
		Total pin current except port 3	+ 100	
		Sync-processor I/O pins and IIC-bus clock and data pins	+ 150	
Operating temperature	T_A	–	– 40 to + 85	°C
Storage temperature	T_{STG}	–	– 65 to + 150	°C

Table 19-2. D.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 4.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input High voltage	V_{IH1}	All input pins except V_{IH2} and V_{IH3}	0.8 V_{DD}	–	V_{DD}	V
	V_{IH2}	X_{IN}	2.7		V_{DD}	
	V_{IH3}	TTL input (HsyncI, VsyncI, and CsyncI)	2.0		V_{DD}	
Input Low voltage	V_{IL1}	All input pins except V_{IL2} and V_{IL3}	–	–	0.2 V_{DD}	V
	V_{IL2}	X_{IN}			1.0	
	V_{IL3}	TTL input (HsyncI, VsyncI, and CsyncI)			0.8	
Output High voltage	V_{OH1}	I_{OH} = – 8 mA; Port 3 only	V_{DD} – 1.0	–	–	V
	V_{OH2}	I_{OH} = – 2 mA Ports 0, 2, ClampO, H, and VsyncO				
	V_{OH3}	I_{OH} = – 6 mA; Port 1				

Table 19-2. D.C. Electrical Characteristics (Continued)

(T_A = – 40 °C to + 85 °C, V_{DD} = 4.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Output Low voltage	V_{OL1}	I_{OL} = 8 mA; port 3 only	–	–	0.4	V
	V_{OL2}	I_{OL} = 2 mA Port 0, 2, ClampO, H, and VsyncO			0.4	
	V_{OL3}	I_{OL} = 6 mA Port 1; SCL and SDA			0.6	
Input High leakage current	I_{LIH1}	V_{IN} = V_{DD} All input pins except X_{IN}, X_{OUT}	–	–	3	µA
	I_{LIH2}	V_{IN} = V_{DD}; X_{OUT} only	–	–	20	
	I_{LIH3}	V_{IN} = V_{DD}; X_{IN} only	2.5	6	20	
Input Low leakage current	I_{LIL1}	V_{IN} = 0 V; All input pins except X_{IN}, X_{OUT}, and RESET	–	–	– 3	µA
	I_{LIL2}	V_{IN} = 0 V; X_{OUT} only	–	–	– 20	
	I_{LIL3}	V_{IN} = 0 V; X_{IN} only	– 2.5	– 6	– 20	
Output High leakage current	I_{LOH1}	V_{OUT} = V_{DD}	–	–	3	µA
Output Low leakage current	I_{LOL1}	V_{OUT} = 0 V	–	–	– 3	µA
Pull-up resistor	R_{L1}	V_{IN} = 0 V Ports 3.7–3.4	20	47	80	kΩ
	R_{L2}	V_{IN} = 0 V RESET only	150	280	480	
Supply current (note)	I_{DD1}	Operation mode; 12 MHz crystal C1 = C2 = 22pF	–	15	30	mA
	I_{DD2}	Idle mode; 12 MHz crystal C1 = C2 = 22pF		5	10	
	I_{DD3}	Stop mode		1	10	µA

NOTE: Supply current does not include drawn internal pull–up resistors and external loads of output.

Table 19-3. Data Retention Supply Voltage in Stop Mode

(T_A = – 40 °C to + 85 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Data retention supply voltage	V_{DDDR}	Stop mode	2	–	5.5	V
Data retention supply current	I_{DDDR}	Stop mode, V_{DDDR} = 2.0 V	–	–	5	µA

NOTES:
1. During the oscillator stabilization wait time (t_{WAIT}), all CPU operations must be stopped.
2. Supply current does not include drawn through internal pull–up resistors and external output current loads.

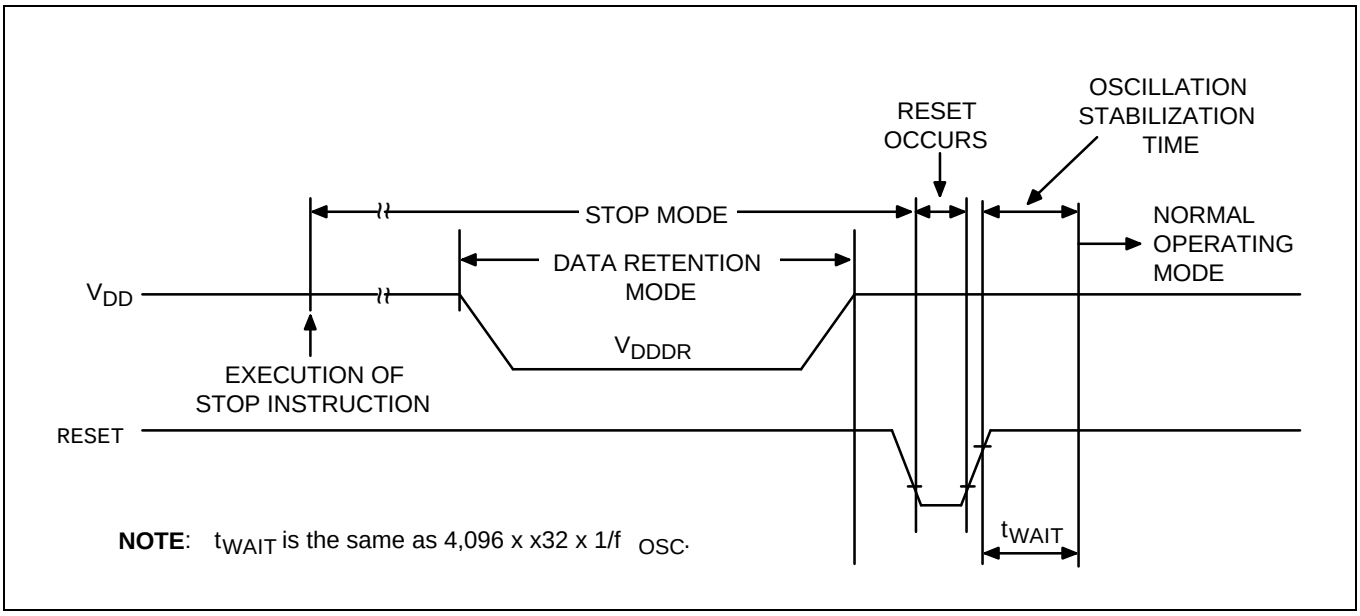


Figure 19-1. Stop Mode Release Timing When Initiated by a Reset

Table 19-4. Input/Output Capacitance

(T_A = –40 °C to + 85 °C, V_{DD} = 0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input capacitance	C_{IN}	f = 1 MHz; unmeasured pins are connected to V_{SS}	–	–	10	pF
Output capacitance	C_{OUT}					
I/O capacitance	C_{IO}					

Table 19-5. A/D Converter Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 4.0 V to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Resolution			–	8	–	bit
Total accuracy		V_{DD} = 5 V Conversion time = 5 µs	–	–	± 2	LSB
Integral linearity error	ILE	AV_{REF} = 5 V		–	± 1	
Differential linearity error	DLE	AV_{SS} = 0 V		–	± 1	
Offset error of top	EOT			± 1	± 2	
Offset error of bottom	EOB			± 0.5	± 2	
Conversion time (1)	t_{CON}	8 bit conversion 34 x n/f_{OSC} (3), n=1,4,8,16	17	–	170	µs
Analog input voltage	V_{IAN}	–	AV_{SS}	–	AV_{REF}	V
Analog input impedance	R_{AN}	–	2	1000	–	mΩ
Analog reference voltage	AV_{REF}	–	2.5	–	V_{DD}	V
Analog ground	AV_{SS} (4)	–	V_{SS}	–	V_{SS}	V
Analog input current	I_{ADIN}	AV_{REF} = V_{DD} = 5V	–	–	10	µA
Analog block Current (2)	I_{ADC}	AV_{REF} = V_{DD} = 5V	–	1	3	mA
		AV_{REF} = V_{DD} = 3V		0.5	1.5	mA
		AV_{REF} = V_{DD} = 5V When power down mode		100	500	nA

NOTES:
1. "Conversion time" is the time required from the moment a conversion operation starts until it ends.
2. I_{ADC} is an operating current during the A/D conversion.
3. f_{OSC} is the main oscillator clock.
4. V_{SS} port shaves with the AV_{SS} for KS88C6216/C6224/C6232.

Table 19-6. A.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 4.0 V to 5.5V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Noise Filter	t_{NF1H} t_{NF1L}	INT0–2, TM0CAP and TM1CK (RC delay)	300	–	–	ns
	t_{NF2}	RESET only (RC delay)	800	–	–	

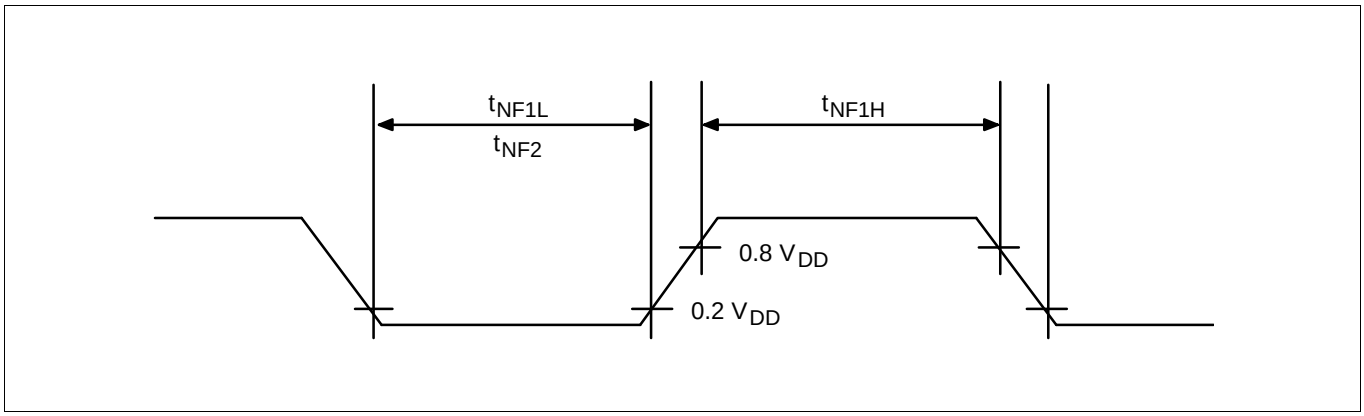
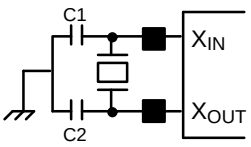
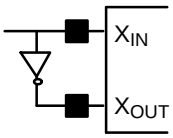


Figure 19-2. Input Timing Measurement Points for P0.0–P0.2, TM0CAP, and TM1CK

Table 19-7. Oscillation Characteristics

(T_A = – 40 °C + 85 °C)

Oscillator	Clock Circuit	Conditions	Min	Typ	Max	Unit
Main crystal or ceramic		V_{DD} = 4.0 V to 5.5 V	8	–	12	MHz
External clock (main)		V_{DD} = 4.0 V to 5.5 V	8	–	12	MHz

NOTE: The maximum oscillator frequency is 12 MHz. If you use an oscillator frequency higher than 12 MHz, you cannot select a non-divided CPU clock using CLKCON settings. That is, you must select one of the divide-by values.

Table 19-8. Oscillation Stabilization Time

(T_A = – 40 °C + 85 °C, V_{DD} = 4.0 V to 5.5 V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Crystal	V_{DD} = 4.0 V to 5.5 V	–	–	20	ms
Ceramic	V_{DD} = 4.0 V to 5.5V	–	–	10	
External clock	X_{IN} input high and low level width (t_{XH}, t_{XL})	25	–	500	ns

NOTE: Oscillation stabilization time is the time required for the CPU clock to return to its normal oscillation frequency after a power-on occurs, or when Stop mode is released.

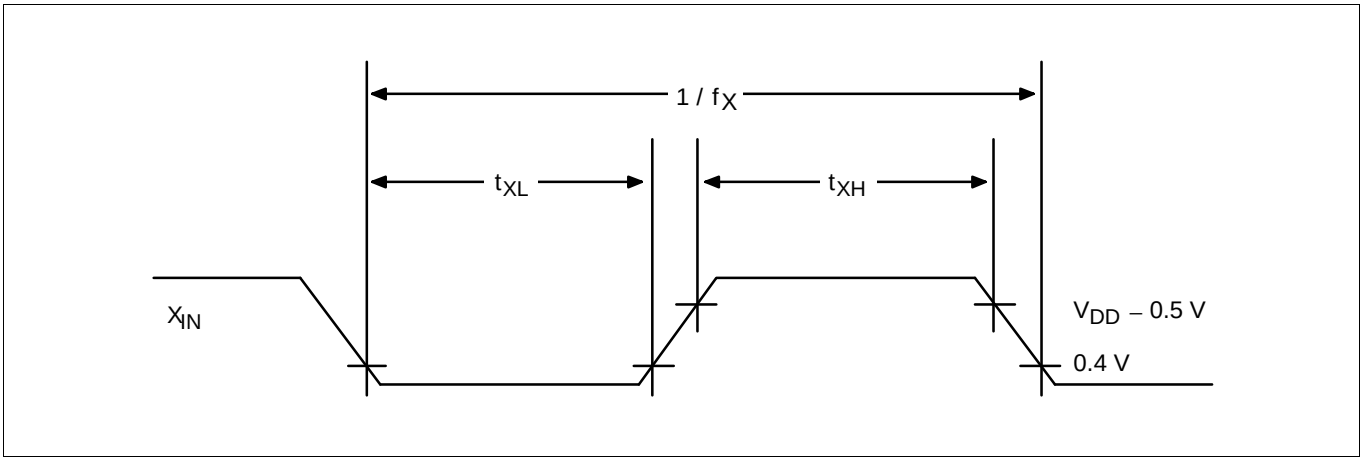


Figure 19-3. Clock Timing Measurement Points for X_{IN}

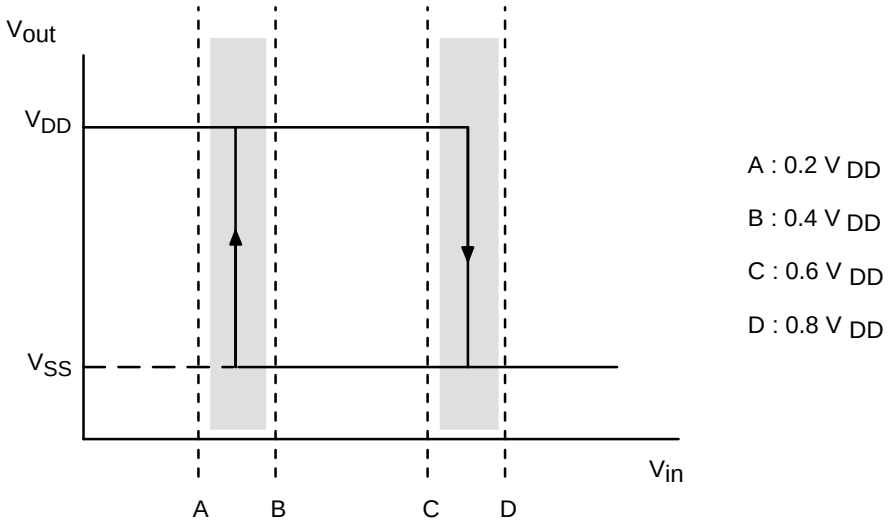
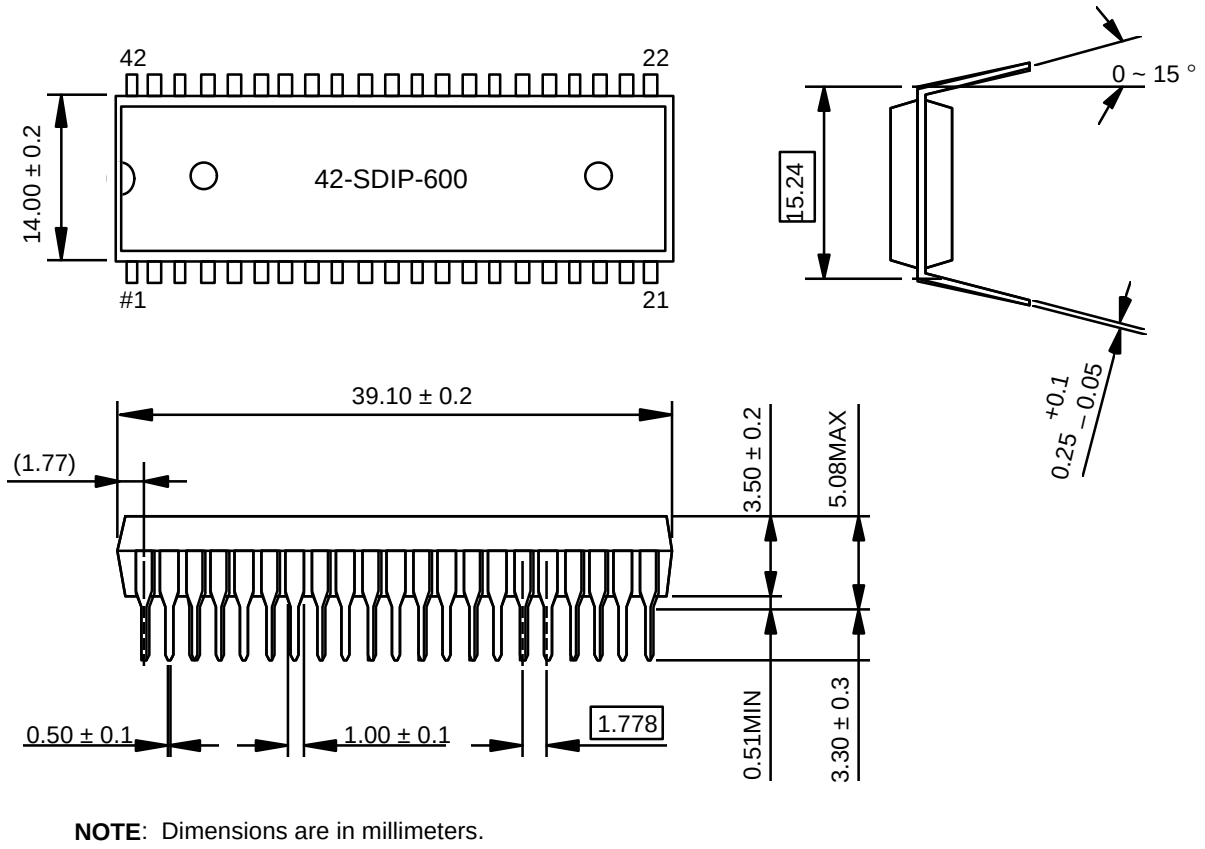


Figure 19-4. Schmitt Trigger Characteristics (Normal Port; except TTL Input)

20 MECHANICAL DATA

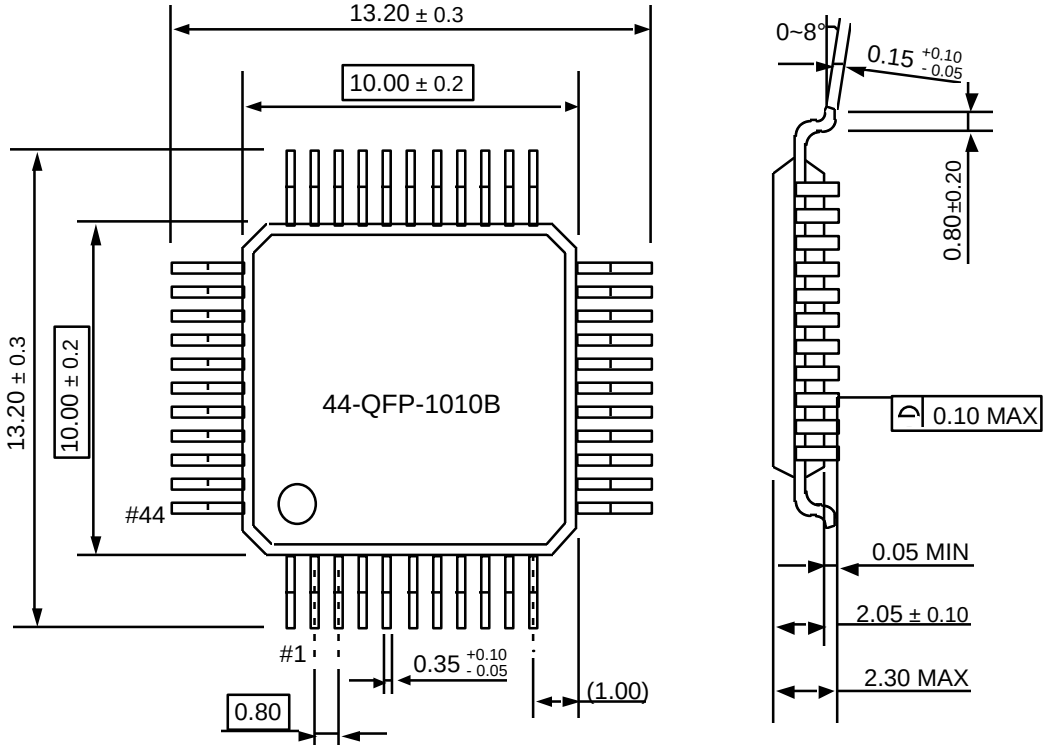
OVERVIEW

The KS88C6216/C6224/C6232 microcontroller is available in a 42-pin SDIP package (Samsung part number 42-SDIP-600) and a 44-QFP package (Samsung part number 44-QFP-1010B).



NOTE: Dimensions are in millimeters.

Figure 20-1. 42-Pin SDIP Package Mechanical Data (42-SDIP-600)



NOTE: Dimensions are in millimeters.

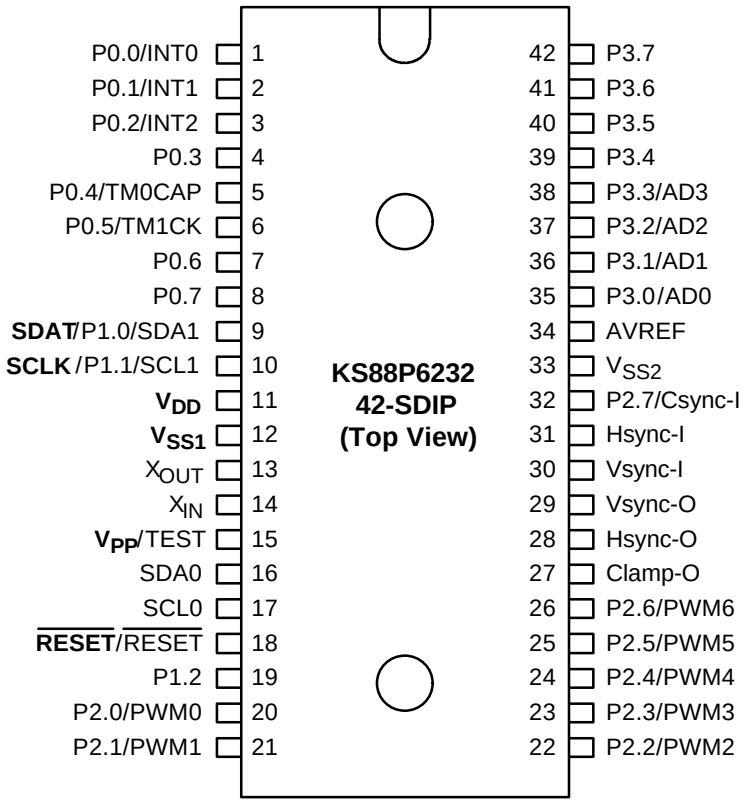
Figure 20-2. 44-Pin QFP Package Mechanical Data (44-QFP-1010B)

21 KS88P6232 OTP

OVERVIEW

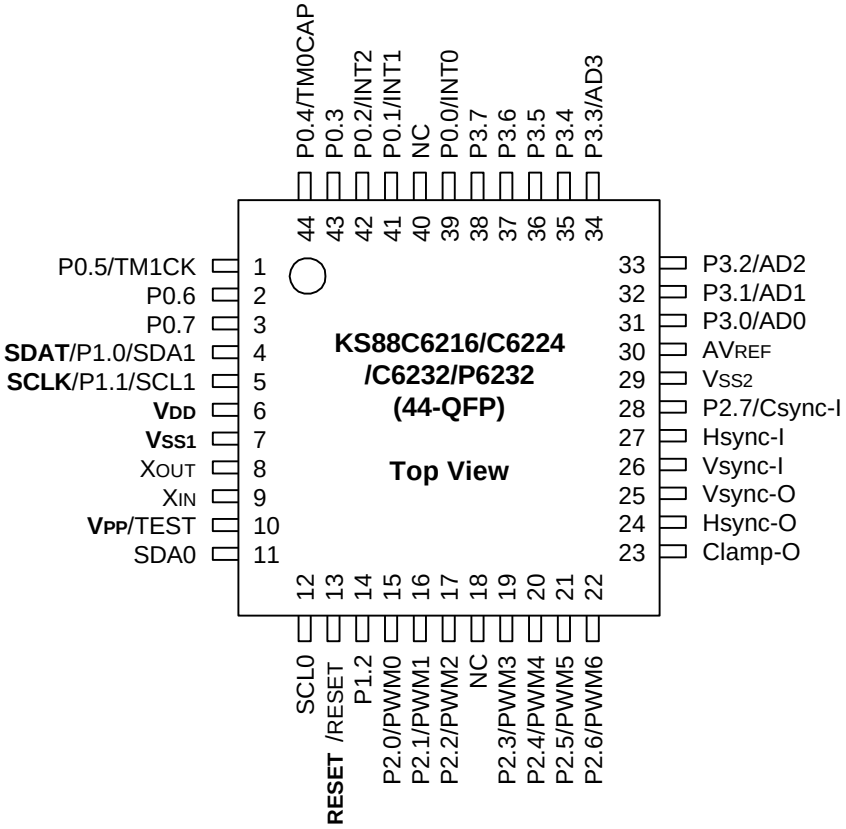
The KS88P6232 single-chip CMOS microcontroller is the OTP (One Time Programmable) version of the KS88C6216/C6224/C6232 microcontrollers. It has an on-chip EPROM instead of masked ROM. The EPROM is accessed by serial data format.

The KS88P6232 is fully compatible with the KS88C6216/C6224/C6232, both in function and in pin configuration. Because of its simple programming requirements, the KS88P6232 is ideal for use as an evaluation chip for the KS88C6216/C6224/C6232.



NOTE: The bolds indicate an OTP pin name.

Figure 21-1. KS88P6232 Pin Assignments (42-SDIP Package)



NOTE: The bolds indicate an OTP pin name.

Figure 21-2. KS88P6232 Pin Assignments (44-QFP Package)

Table 21-1. Descriptions of Pins Used to Read/Write the EPROM

Main Chip Pin Name	During Programming			
	Pin Name	Pin No.	I/O	Function
P1.0	SDAT	9 (4)	I/O	Serial data pin. Output port when reading and input port when writing. Can be assigned as a Input/push-pull output port.
P1.1	SCLK	10 (5)	I	Serial clock pin. Input only pin.
TEST	V_{PP} (TEST)	15 (10)	I	Power supply pin for EPROM cell writing (indicates that OTP enters into the writing mode). When 12.5 V is applied, OTP is in writing mode and when 5 V is applied, OTP is in reading mode. (Option)
RESET	RESET	18 (13)	I	Chip Initialization
V_{DD}/V_{SS1}	V_{DD}/V_{SS1}	11/12 (6/7)	I	Logic power supply pin. V_{DD} should be tied to +5 V during programming.

NOTE: Parentheses indicate 44-QFP OTP pin number.

Table 21-2. Comparison of KS88P6232 and KS88C6216/C6224/C6232 Features

Characteristic	KS88P6232	KS88C6216/C6224/C6232
Program Memory	32-Kbyte EPROM	16/24/32-Kbyte mask ROM
Operating Voltage (V_{DD})	4.0 V to 5.5 V	4.0 V to 5.5V
OTP Programming Mode	V_{DD} = 5 V, V_{PP} (TEST)=12.5V	
Pin Configuration	42SDIP, 44QFP	42SDIP, 44QFP
EPROM Programmability	User Program 1 time	Programmed at the factory

OPERATING MODE CHARACTERISTICS

When 12.5 V is supplied to the V_{PP}(TEST) pin of the KS88P6232, the EPROM programming mode is entered. The operating mode (read, write, or read protection) is selected according to the input signals to the pins listed in Table 16-3 below.

Table 21-3. Operating Mode Selection Criteria

V_{DD}	V_{PP} (TEST)	REG/ MEM	ADDRESS (A15–A0)	R/W	MODE
5 V	5 V	0	0000H	1	EPROM read
	12.5 V	0	0000H	0	EPROM program
	12.5 V	0	0000H	1	EPROM verify
	12.5 V	1	0E3FH	0	EPROM read protection

NOTE: "0" means Low level; "1" means High level.

D.C. ELECTRICAL CHARACTERISTICS

Table 21-4. D.C. Electrical Characteristics

(T_A = – 40 °C to + 85 °C, V_{DD} = 4.0 V to 5.5 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input High leakage current	I_{LIH1}	$V_{IN} = V_{DD}$ All input pins except X_{IN}, X_{OUT}	–	–	3	µA
	I_{LIH2}	$V_{IN} = V_{DD}$; X_{OUT} only	–	–	20	
	I_{LIH3}	$V_{IN} = V_{DD}$; X_{IN} only	2.5	6	20	
Input Low leakage current	I_{LIL1}	V_{IN} = 0 V; All input pins except X_{IN}, X_{OUT}, and RESET	–	–	– 3	µA
	I_{LIL2}	V_{IN} = 0 V; X_{OUT} only	–	–	– 20	
	I_{LIL3}	V_{IN} = 0 V; X_{IN} only	– 2.5	– 6	– 20	
Output High leakage current	I_{LOH1}	$V_{OUT} = V_{DD}$	–	–	3	µA
Output Low leakage current	I_{LOL1}	V_{OUT} = 0 V	–	–	– 3	µA
Pull-up resistor	R_{L1}	V_{IN} = 0 V Ports 3.7–3.4	20	47	80	kΩ
	R_{L2}	V_{IN} = 0 V RESET only	150	280	480	
Supply current (note)	I_{DD1}	Operation mode; 12 MHz crystal C1 = C2 = 22pF	–	15	30	mA
	I_{DD2}	Idle mode; 12 MHz crystal C1 = C2 = 22pF		5	10	
	I_{DD3}	Stop mode		1	10	µA

NOTE: Supply current does not include drawn internal pull–up resistors and external loads of output.

22 DEVELOPMENT TOOLS

OVERVIEW

Samsung provides a powerful and easy-to-use development support system in turnkey form. The development support system is configured with a host system, debugging tools, and support software. For the host system, any standard computer that operates with MS-DOS as its operating system can be used. One type of debugging tool including hardware and software is provided: the sophisticated and powerful in-circuit emulator, SMDS2+, for KS57, KS86, KS88 families of microcontrollers. The SMDS2+ is a new and improved version of SMDS2. Samsung also offers support software that includes debugger, assembler, and a program for setting options.

SHINE

Samsung Host Interface for In-Circuit Emulator, SHINE, is a multi-window based debugger for SMDS2+. SHINE provides pull-down and pop-up menus, mouse support, function/hot keys, and context-sensitive hyper-linked help. It has an advanced, multiple-windowed user interface that emphasizes ease of use. Each window can be sized, moved, scrolled, highlighted, added, or removed completely.

SAMA ASSEMBLER

The Samsung Arrangeable Microcontroller (SAM) Assembler, SAMA, is a universal assembler, and generates object code in standard hexadecimal format. Assembled program code includes the object code that is used for ROM data and required SMDS program control data. To assemble programs, SAMA requires a source file and an auxiliary definition (DEF) file with device specific information.

SASM88

The SASM88 is an relocatable assembler for Samsung's KS88-series microcontrollers. The SASM88 takes a source file containing assembly language statements and translates into a corresponding source code, object code and comments. The SASM88 supports macros and conditional assembly. It runs on the MS-DOS operating system. It produces the relocatable object code only, so the user should link object file. Object files can be linked with other object files and loaded into memory.

HEX2ROM

HEX2ROM file generates ROM code from HEX file which has been produced by assembler. ROM code must be needed to fabricate a microcontroller which has a mask ROM. When generating the ROM code (.OBJ file) by HEX2ROM, the value "FF" is filled into the unused ROM area upto the maximum ROM size of the target device automatically.

TARGET BOARDS

Target boards are available for all KS88-series microcontrollers. All required target system cables and adapters are included with the device-specific target board.

OTPs

One time programmable microcontroller (OTP) for the KS88C6216/C6224/C6232 microcontroller and OTP programmer (Gang) are now available.

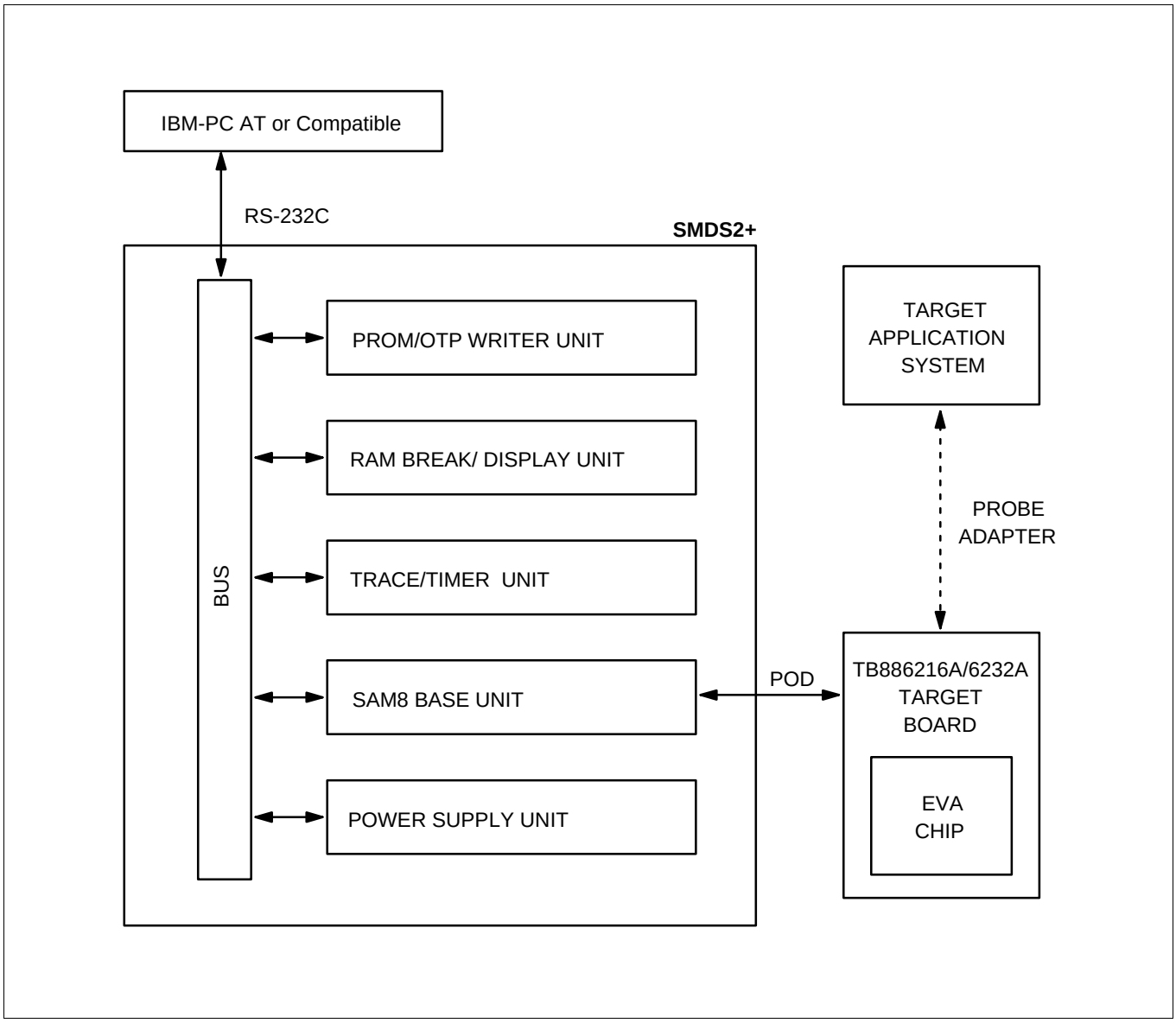


Figure 22-1. SMDS Product Configuration (SMDS2+)

TB886224A/6232A TARGET BOARD

The TB886224A/6232A target board is used for the KS88C6216/C6224/C6232/P6232 microcontroller. It is supported by the SMDS2+ development system.

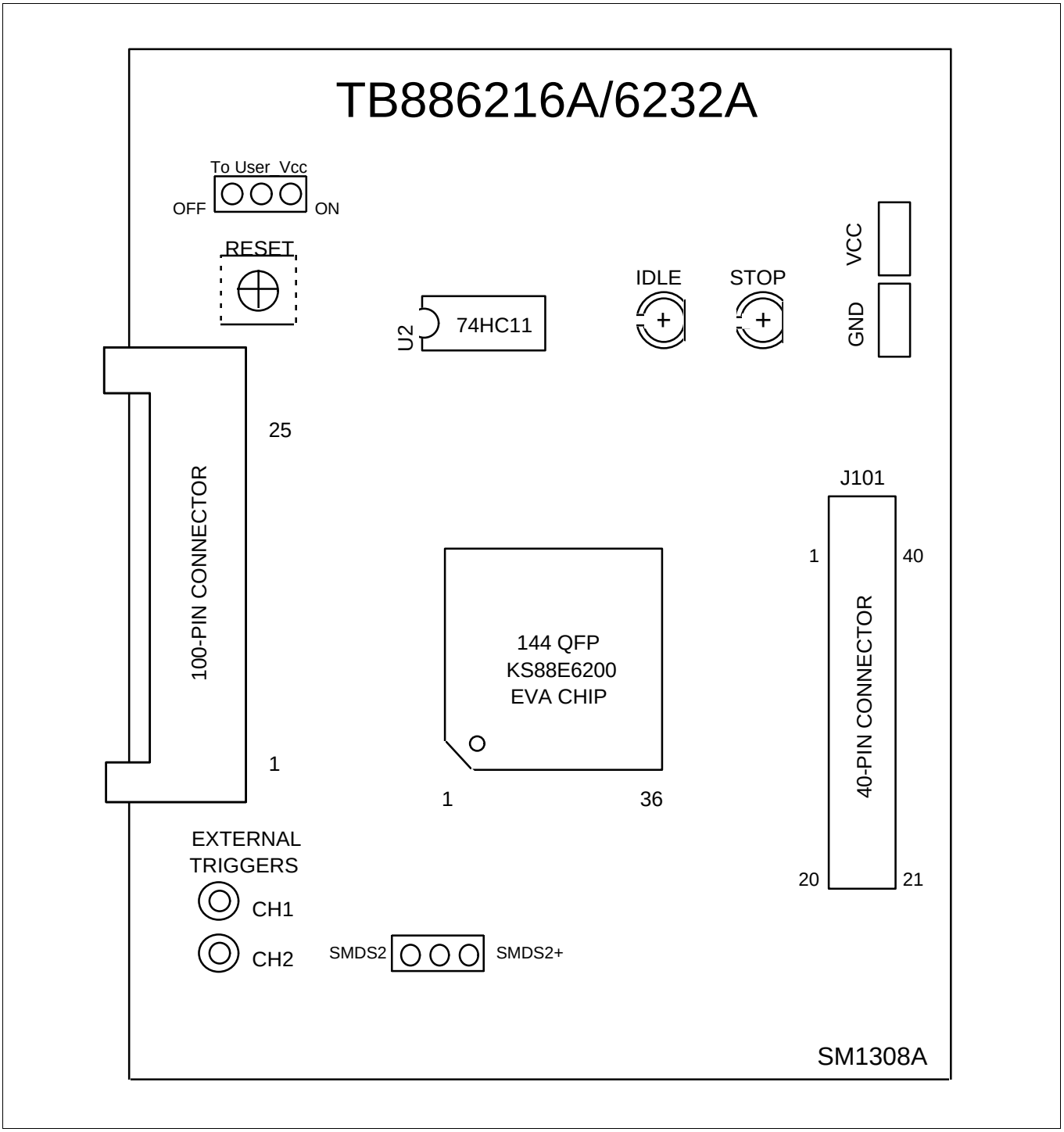


Figure 22-2. TB886224A/6232A Target Board Configuration

Table 22-1. Power Selection Settings for TB886224A/TB886232A

"To User_Vcc" Settings	Operating Mode	Comments
To User_Vcc OFF ON	TB886216A TB886232A V_{CC} V_{SS} TARGET SYSTEM V_{CC} SMDS2/SMDS2+	The SMDS2/SMDS2+ supplies V_{CC} to the target board (evaluation chip) and the target system.
To User_Vcc OFF ON	TB886216A TB886232A External V_{CC} V_{SS} TARGET SYSTEM V_{CC} SMDS2/SMDS2+	The SMDS2/SMDS2+ supplies V_{CC} only to the target board (evaluation chip). The target system must have its own power supply.

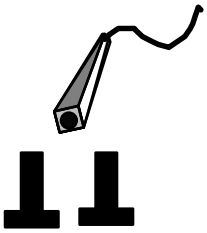
SMDS2+ Selection (SAM8)

In order to write data into program memory that is available in SMDS2+, the target board should be selected to be for SMDS2+ through a switch as follows. Otherwise, the program memory writing function is not available.

Table 22-2. The SMDS2+ Tool Selection Setting

"SW1" Setting	Operating Mode
SMDS2 SMDS2+	R/W* R/W* TARGET BOARD SMDS2+

Table 22-3. Using Single Header Pins as the Input Path for External Trigger Sources

Target Board Part	Comments
EXTERNAL TRIGGERS CH1 CH2	 Connector from external trigger sources of the application system You can connect an external trigger source to one of the two external trigger channels (CH1 or CH2) for the SMDS2+ breakpoint and trace functions.

IDLE LED

This LED is ON when the evaluation chip (KS88E6200) is in idle mode.

STOP LED

This LED is ON when the evaluation chip (KS88E6200) is in stop mode.

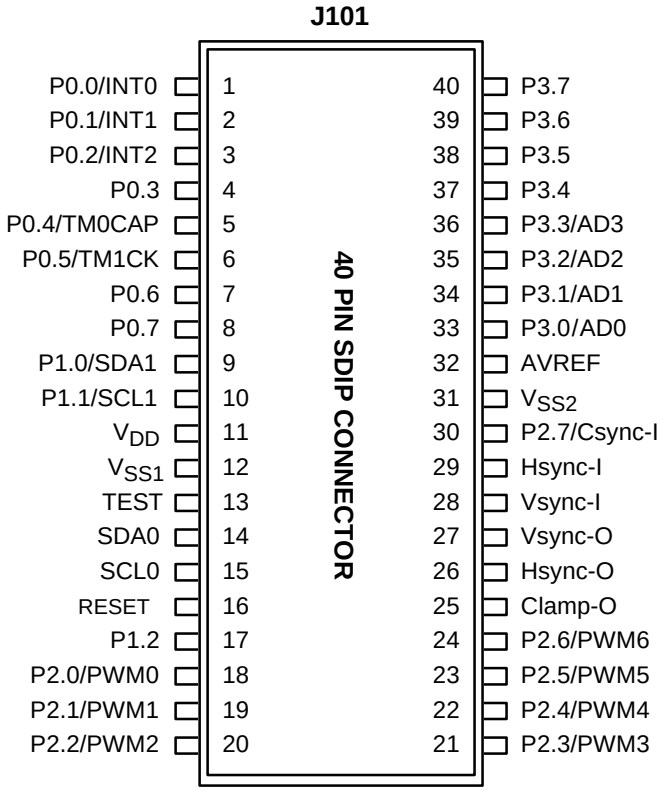


Figure 22-3. 40-Pin Connector for TB886224A/6232A

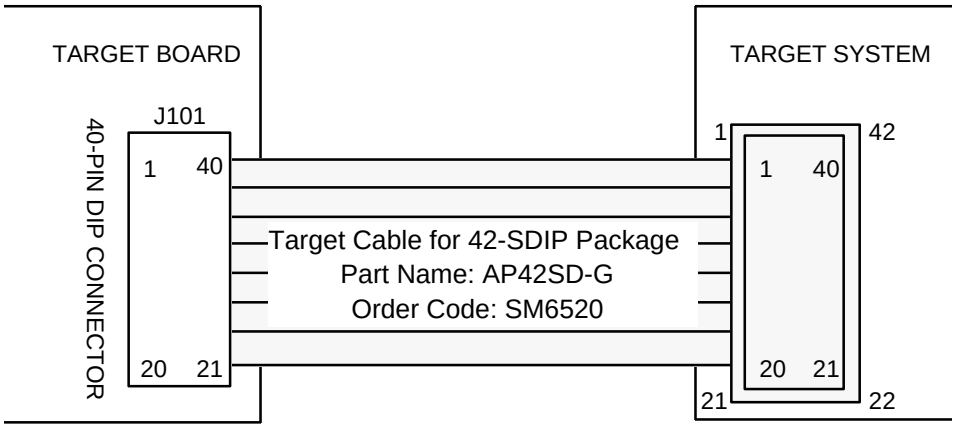


Figure 22-4. TB886224A/6232A Adapter Cable for 42-SDIP Package (KS88C6216/C6224/C6232/P6232)

KS88 SERIES MASK ROM ORDER FORM

Product description:

Device Number: KS88C_____- _____ (write down the ROM code number)

Product Order Form: ☐ Package ☐ Pellet ☐ Wafer Package Type: _____

Package Marking (Check One):

☐ Standard

SEC @ YWW
Device Name

☐ Custom A
(Max 10 chars)

@ YWW
Device Name

☐ Custom B
(Max 10 chars each line)

@ YWW

@ : Assembly site code, Y : Last number of assembly year, WW : Week of assembly

Delivery Dates and Quantities:

Deliverable	Required Delivery Date	Quantity	Comments
ROM code	–	Not applicable	See ROM Selection Form
Customer sample			
Risk order			See Risk Order Sheet

Please answer the following questions:

☞ **For what kind of product will you be using this order?**

☐ New product ☐ Upgrade of an existing product

☐ Replacement of an existing product ☐ Other

If you are replacing an existing product, please indicate the former product name

( )

☞ **What are the main reasons you decided to use a Samsung microcontroller in your product?**

Please check all that apply.

☐ Price ☐ Product quality ☐ Features and functions

☐ Development system ☐ Technical support ☐ Delivery on time

☐ Used same micom before ☐ Quality of documentation ☐ Samsung reputation

Mask Charge (US$ / Won): _____

Customer Information:

Company Name: _____ Telephone number _____

Signatures: _____ _____

(Person placing the order) (Technical Manager)

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)

KS88 SERIES
REQUEST FOR PRODUCTION AT CUSTOMER RISK

Customer Information:

Company Name: _____

Department: _____

Telephone Number: _____ Fax: _____

Date: _____

Risk Order Information:

Device Number: KS88C_____- _____ (write down the ROM code number)

Package: Number of Pins: _____ Package Type: _____

Intended Application: _____

Product Model Number: _____

Customer Risk Order Agreement:

We hereby request SEC to produce the above named product in the quantity stated below. We believe our risk order product to be in full compliance with all SEC production specifications and, to this extent, agree to assume responsibility for any and all production risks involved.

Order Quantity and Delivery Schedule:

Risk Order Quantity: _____ PCS

Delivery Schedule:

Delivery Date (s)	Quantity	Comments

Signatures: _____ _____

(Person Placing the Risk Order) (SEC Sales Representative)

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)

KS88C6216 MASK OPTION SELECTION FORM

Device Number: KS88C6216-_____ (write down the ROM code number)

Attachment (Check one): ☐ Diskette ☐ PROM

Customer Checksum: _____

Company Name: _____

Signature (Engineer): _____

Please answer the following questions:

☞ **Application** (Product Model ID: _____)

☐ Audio	☐ Video	☐ Telecom
☐ LCD Databank	☐ Caller ID	☐ LCD Game
☐ Industrials	☐ Home Appliance	☐ Office Automation
☐ Remocon	☐ Other	

Please describe in detail its application

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)

KS88C6224 MASK OPTION SELECTION FORM

Device Number: KS88C6224-_____ (write down the ROM code number)

Attachment (Check one): ☐ Diskette ☐ PROM

Customer Checksum: _____

Company Name: _____

Signature (Engineer): _____

Please answer the following questions:

☞ **Application** (Product Model ID: _____)

☐ Audio	☐ Video	☐ Telecom
☐ LCD Databank	☐ Caller ID	☐ LCD Game
☐ Industrials	☐ Home Appliance	☐ Office Automation
☐ Remocon	☐ Other	

Please describe in detail its application

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)

KS88C6232 MASK OPTION SELECTION FORM

Device Number: KS88C6232-_____ (write down the ROM code number)

Attachment (Check one): ☐ Diskette ☐ PROM

Customer Checksum: _____

Company Name: _____

Signature (Engineer): _____

Please answer the following questions:

☞ **Application** (Product Model ID: _____)

☐ Audio	☐ Video	☐ Telecom
☐ LCD Databank	☐ Caller ID	☐ LCD Game
☐ Industrials	☐ Home Appliance	☐ Office Automation
☐ Remocon	☐ Other	

Please describe in detail its application

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)

KS88 SERIES OTP FACTORY WRITING ORDER FORM (1/2)

Product Description:

Device Number: KS88P_____-_____ (write down the ROM code number)

Product Order Form: ☐ Package ☐ Pellet ☐ Wafer

If the product order form is package: Package Type: _____

Package Marking (Check One):

☐ Standard ☐ Custom A (Max 10 chars) ☐ Custom B (Max 10 chars each line)

SEC @ YWW Device Name	@ YWW Device Name	@ YWW

@ : Assembly site code, Y : Last number of assembly year, WW : Week of assembly

Delivery Dates and Quantity:

ROM Code Release Date	Required Delivery Date of Device	Quantity

Please answer the following questions:

☞ **What is the purpose of this order?**

☐ New product development ☐ Upgrade of an existing product

☐ Replacement of an existing microcontroller ☐ Other

If you are replacing an existing microcontroller, please indicate the former microcontroller name

( )

☞ **What are the main reasons you decided to use a Samsung microcontroller in your product?**

Please check all that apply.

☐ Price ☐ Product quality ☐ Features and functions

☐ Development system ☐ Technical support ☐ Delivery on time

☐ Used same micom before ☐ Quality of documentation ☐ Samsung reputation

Customer Information:

Company Name: _____ Telephone number _____

Signatures: _____ _____

(Person placing the order) (Technical Manager)

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)

KS88P6232 OTP FACTORY WRITING ORDER FORM (2/2)

Device Number: KS88P6232-_____ (write down the ROM code number)

Customer Checksums: _____

Company Name: _____

Signature (Engineer): _____

Read Protection[1]: ☐ **Yes** ☐ **No**

Please answer the following questions:

☞ **Are you going to continue ordering this device?**

☐ Yes ☐ No

If so, how much will you be ordering? _____pcs

☞ **Application** (Product Model ID: _____)

☐ Audio	☐ Video	☐ Telecom
☐ LCD Databank	☐ Caller ID	☐ LCD Game
☐ Industrials	☐ Home Appliance	☐ Office Automation
☐ Remocon	☐ Other	

Please describe in detail its application

_____

NOTES

1. Once you choose a read protection, you cannot read again the programming code from the EPROM.
2. OTP Writing will be executed in our manufacturing site.
3. The writing program is completely verified by a customer. Samsung does not take on any responsibility for errors occurred from the writing program.

(For duplicate copies of this form, and for additional ordering information, please contact your local Samsung sales representative. Samsung sales offices are listed on the back cover of this book.)