H8/330 Application Notes

(On-Chip Supporting Modules)

Preface

The H8/330 is an original Hitachi high-performance single-chip microcontroller with a high-speed H8/300 CPU core and a set of on-chip peripheral functions optimized for industrial embedded control.

With CPU, ROM, RAM, timers, and dual-port RAM integrated onto a single chip, the H8/330 has a wide range of applications, from small systems to large.

This H8/330 application sheet is divided into an on-chip supporting module part that presents sample tasks using the H8/330's on-chip peripheral functions, and an interface part that gives examples of interfaces between the H8/330 and peripheral chips. These examples are intended for reference by users performing software and hardware design.

The sample tasks in this application sheet have been tested, but users intending to use them should first confirm that they operate as desired.

Contents

Guide to Use of the H8/330 Application Sheet

On-Chip Supporting Modules

Interfaces

Guide to Use of the H8/330 Application Sheet

Organization

This application sheet is divided into two parts as shown in figure 1.

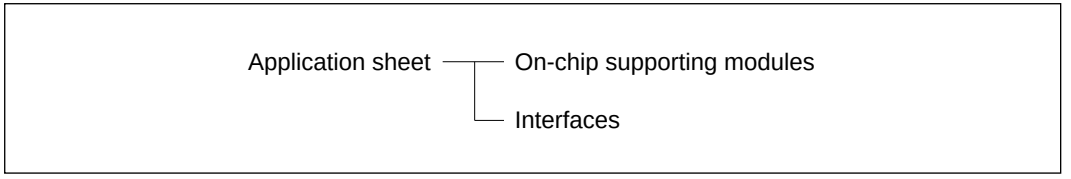


Figure 1 Organization of the Application Sheet

1. On-chip supporting modules

   Usage of the H8/330's on-chip supporting modules (timers, serial communication interface, etc.) is explained through simple sample tasks (pulse cycle measurement etc.).

2. Interfaces

   Examples of circuits that use the H8/330's expanded modes to interface to peripheral chips are described.

1.1 On-Chip Supporting Modules

Usage of on-chip supporting modules is described in the format shown in figure 2.

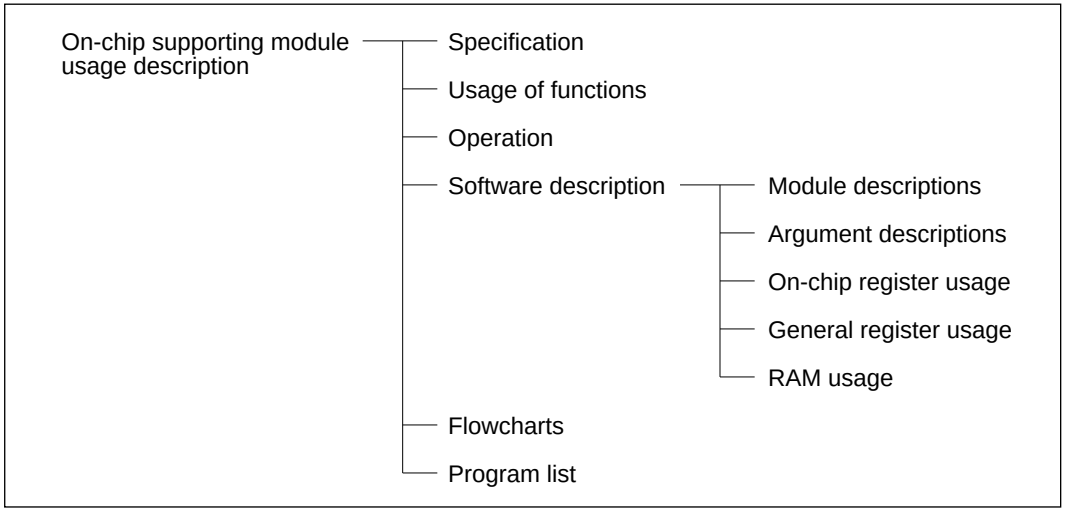


Figure 2 Organization of On-Chip Supporting Module Part

1. Specification

   Gives system specifications for the sample task.

2. Usage of functions

   Describes the features of the on-chip supporting module used by the sample task, and the allocation of on-chip supporting module facilities.

3. Operation

   Uses timing diagrams to explain how the sample task operates.

4. Software description

   a. Module descriptions

      Describes the software modules that operate the task.

   b. Argument descriptions

      Describes the input arguments needed to execute the task, and the output arguments resulting from task execution.

   c. On-chip register usage

      Describes the registers (timer control registers, serial mode register, etc.) in the on-chip register field that are set by the software modules.

   d. General register usage

      Gives the names and functions of the general registers (R0 to R7) used by the software modules.

   e. RAM usage

      Gives the label names and functions of RAM locations used by the software modules.

5. Flowcharts

   Uses general flowcharts to explain the software that executes the sample task.

6. Program list

   Shows program listings of the software that executes the sample task.

1.2 Interfaces

Descriptions of interfaces to peripheral chips (ROM, RAM, timer chip) are organized as shown in figure 3.

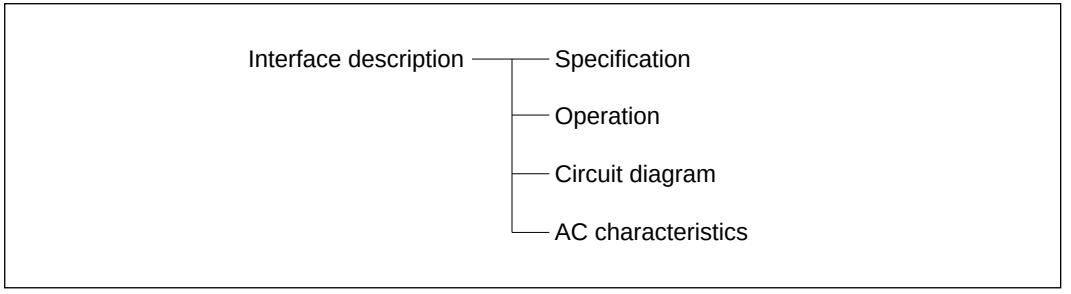


Figure 3 Organization of Interface Part

1. Specification

   Gives circuit specifications: names of connected peripheral chips, memory map, etc.

2. Operation

   Uses timing diagrams to explain how the circuit operates.

3. Circuit diagram

   Shows a diagram of the peripheral chip interface circuit.

4. AC characteristics

   Indicates the AC characteristics of the H8/330 and peripheral chip.

On-Chip Supporting Modules

Specification

1. Control signals from the H8/330's I/O ports control an LCD-II (HD44780) as shown in figure 1 to display a letter (A) on an LCD.

2. The LCD is in a liquid crystal module that includes the LCD-II.

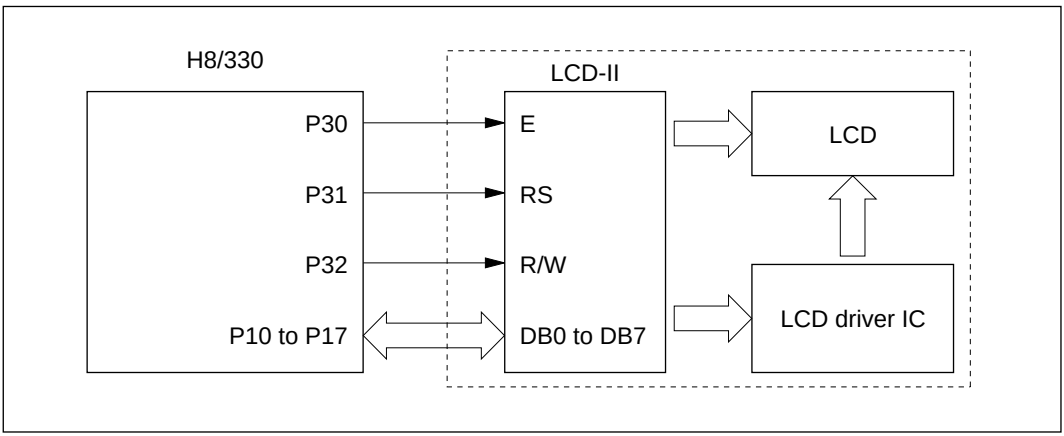


Figure 1 Block Diagram Showing Control of LCD-II by H8/330

Usage of Functions

Table 1 lists the function assignments for this sample task. The H8/330 uses two of its I/O ports (ports 1 and 3) for read/write interfacing to the LCD-II.

Table 1 Assignment of Functions to I/O Ports

I/O port	LCD-II Control Line	Function
P30	E	Data read/write enable signal Data read: LCD-II outputs data on DB0 to DB7 while E signal is high. Data write: LCD-II latches data on fall of E signal.
P31	RS	LCD-II internal register select signal High: Data register Low: Instruction register
P32	R/W	Read/write select signal High: Read (H8/330 ← LCD-II) Low: Write (H8/330 → LCD-II)
P10 to P17	DB0 to DB7	Data lines

Operation

Figure 2 explains how the LCD-II is interfaced. The timing diagrams illustrate the control of the I/O ports in read and write operations.

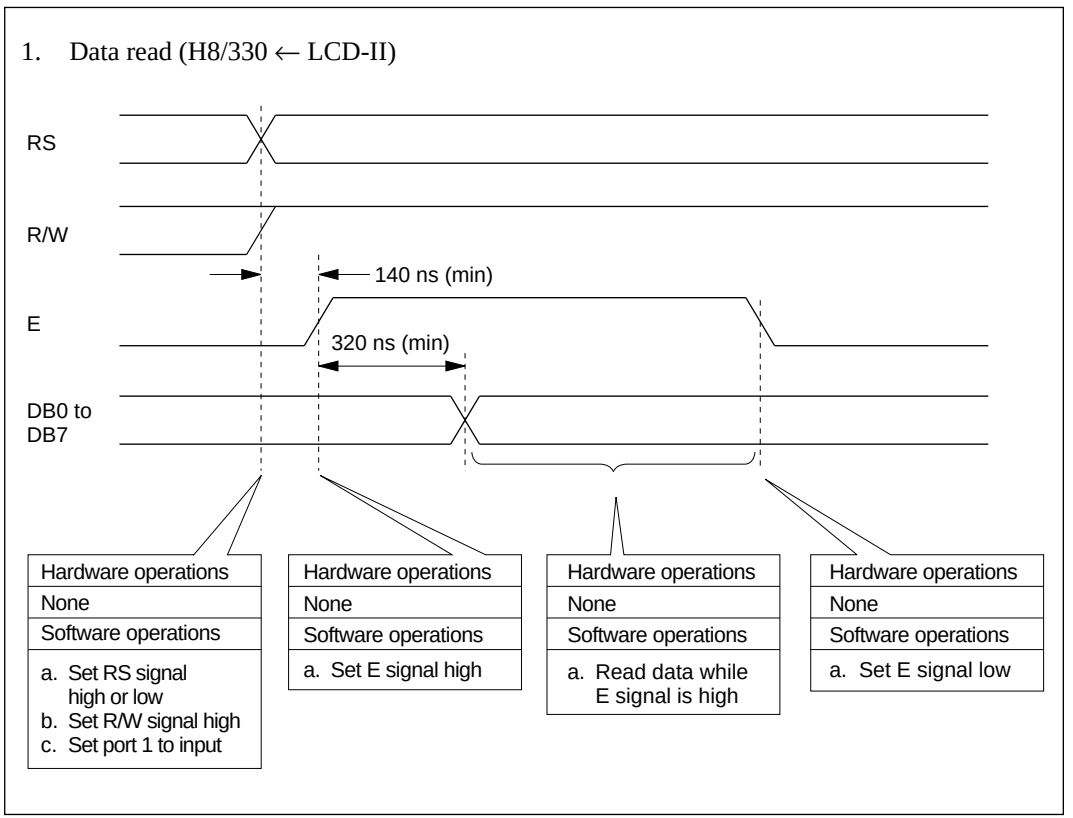
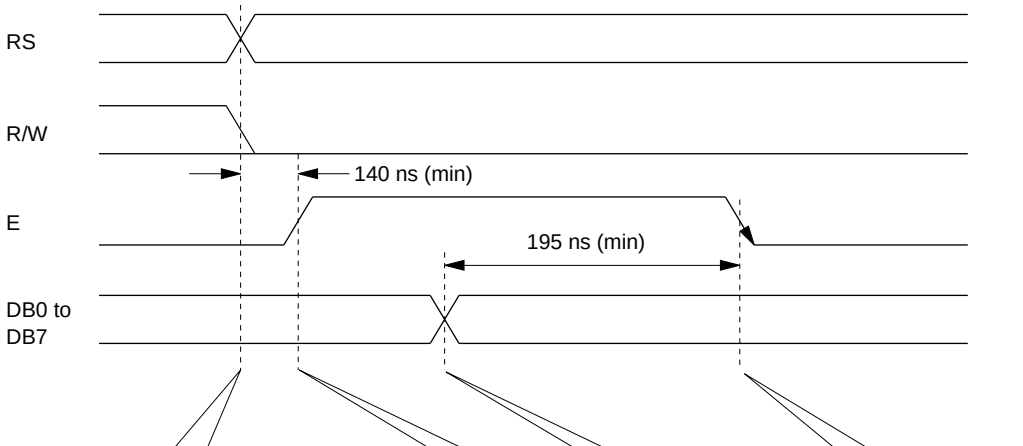


Figure 2 LCD-II Interface Timing (1)

2. Data write (H8/330 → LCD-II)



Hardware operations	Hardware operations	Hardware operations	Hardware operations
None	None	None	None
Software operations	Software operations	Software operations	Software operations
a. Set RS signal high or low b. Set R/W signal low c. Set port 1 to output	a. Set E signal high	a. Output write data from H8/330	a. Set E signal low (LCD-II latches data at fall of E signal)

Figure 2 LCD-II Interface Timing (2)

Software Description

1. Module descriptions

Module	Label	Function
Main program	LCDMN	Initializes I/O ports and loads display data into R1L.
Reset LCD-II	LCD2R	Transfers function set data to LCD-II and resets LCD-II.
Initialize LCD-II	LCD2I	Transfers instructions to LCD-II and initializes LCD-II.
Write ASCII data to LCD-II	ASCWR	Writes ASCII code from input argument to LCD-II.
Check busy flag	BFCHEK	Checks busy flag before interfacing with LCD-II.
Read data from LCD-II	DATRD	Reads data from LCD-II instruction register or data register.
Write data to LCD-II	DATWR	Writes data to LCD-II instruction register or data register.
Wait 15 ms	WAIT	Executes Wait 15 ms, using software timer.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
R1L	Stores ASCII code to be displayed on LCD.	1 byte	Main program	Output
			Write ASCII data to LCD-II	Input
R0L	Selects instruction register or data register. 00: instruction register 01: data register	1 byte	Reset LCD-II Initialize LCD-II Write ASCII data to LCD-II Check busy flag	Output
			Read data from LCD-II Write data to LCD-II	Input
R0H	Stores data to be written to LCD-II.	1 byte	Reset LCD-II Initialize LCD-II Write ASCII data to LCD-II	Output
			Write data to LCD-II	Input
R0H	Stores data read from LCD-II.	1 byte	Read data from LCD-II	Output
			Check busy flag	Input

3. On-chip register usage

Register	Function	Modules Where Used
P3DDR	Sets port 3 to output.	Main program
P1DDR	Switches port 1 between input and output.	Read data from LCD-II, data write
P3DR	Sets control signals output from port 3.	Read data from LCD-II, data write
P1DR	Write: Stores data written from H8/330 Read: Stores data read from LCD-II	Read data from LCD-II, data write

4. General register usage

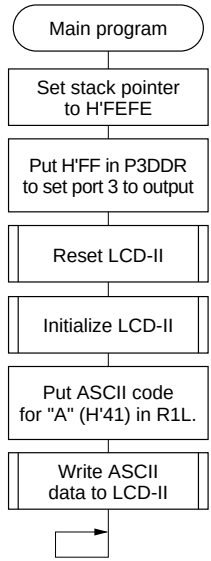
This sample task does not use general registers except for arguments.

5. RAM usage

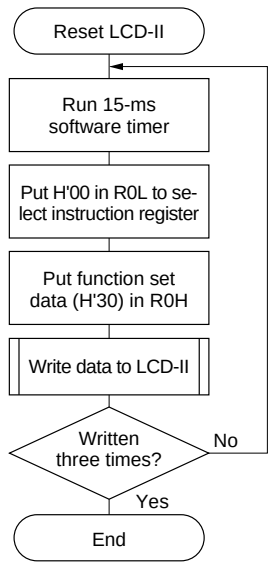
This sample task does not use RAM.

Flowcharts

1. Main program

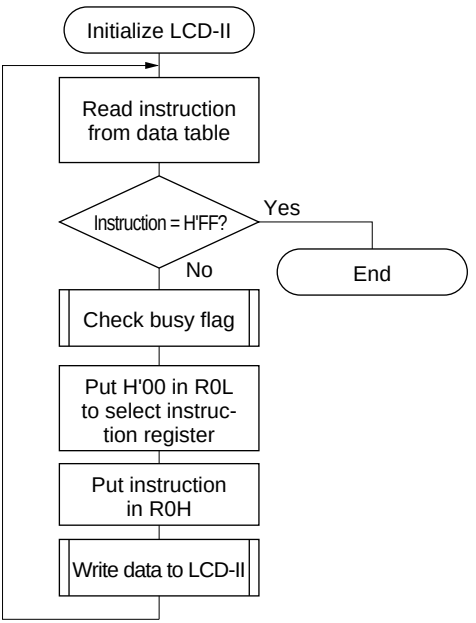


2. Reset LCD-II

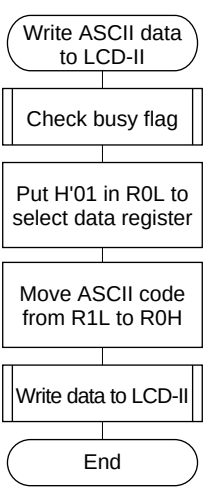


Flowcharts

3. Initialize LCD-II

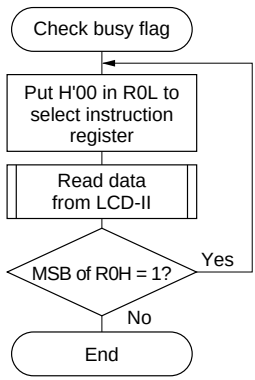


4. Write ASCII data to LCD-II

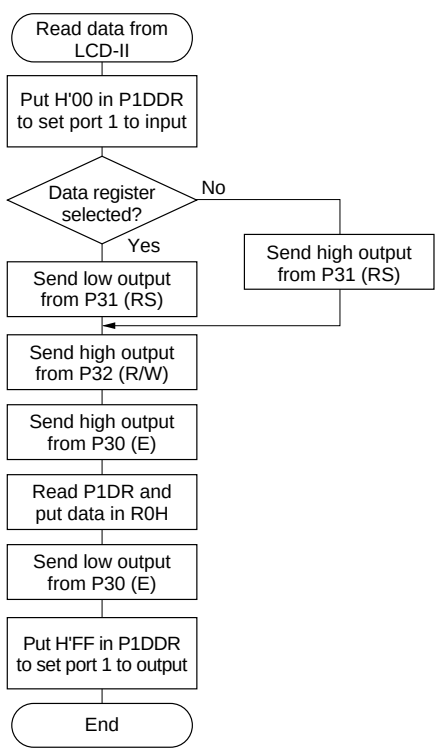


Flowcharts

5. Check busy flag

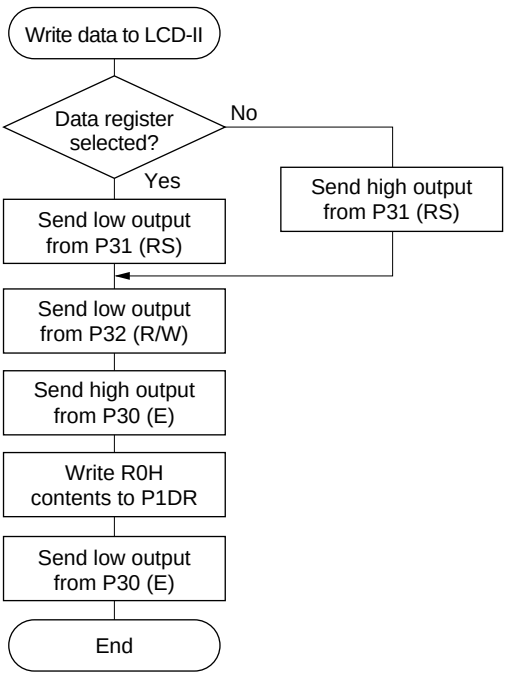


6. Read data from LCD-II



Flowcharts

7. Write data to LCD-II



Program List

Program List

Program List

Program List

Specification

1. This sample task measures the time from the rise of a pulse signal to the next rise (the pulse cycle) as shown in figure 1, and stores the result in RAM.

2. With a 10-MHz clock, pulse cycles from 8 µs to 13.1 ms can be measured in 200-ns steps.

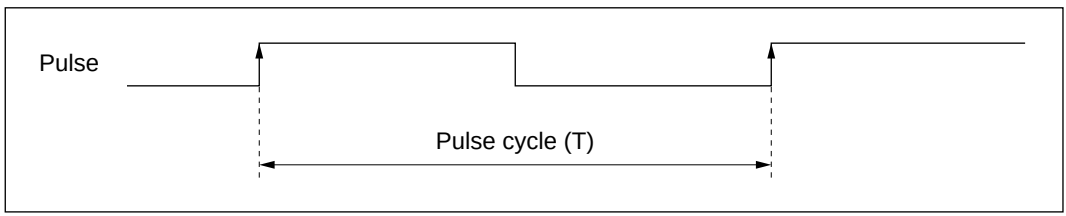


Figure 1 Pulse Cycle Measurement

Usage of Functions

1. Figure 2 shows a block diagram of the 16-bit free-running timer used by this sample task. The 16-bit free-running timer has an input-capture function that detects pulse edges and stores the corresponding timer value in an on-chip register.

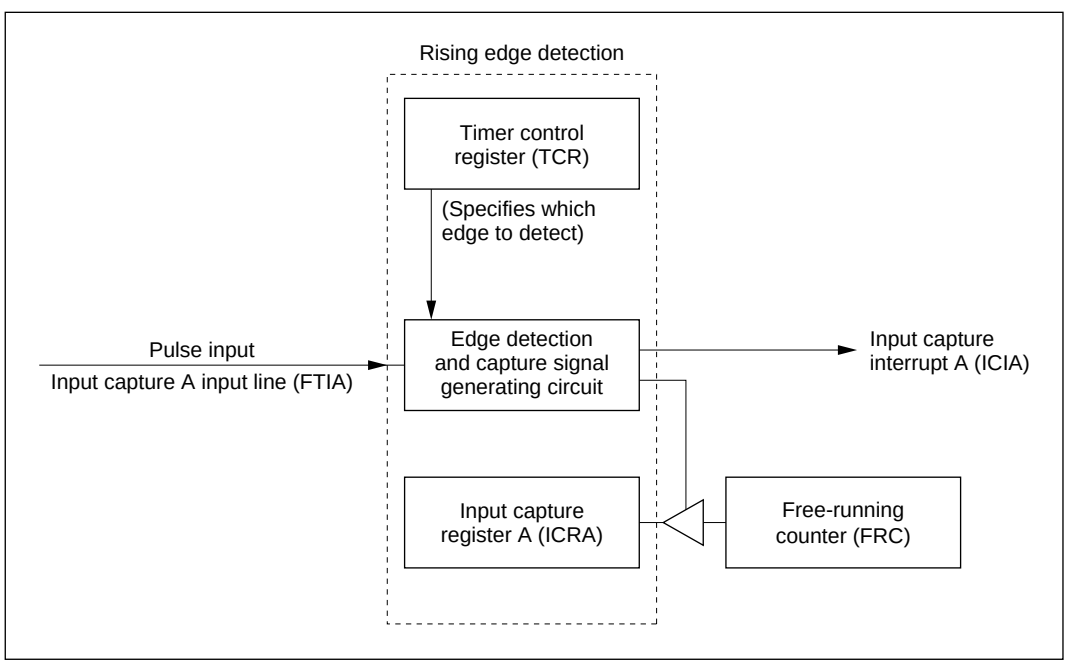


Figure 2 16-Bit Free-Running Timer Block Diagram

Usage of Functions

2. Table 1 lists the function assignments for this sample task. Functions of the 16-bit free-running timer are used to measure pulse cycle length.

Table 1 16-Bit Free-Running Timer Function Assignments

16-Bit Free-Running Timer Function	Description
FTIA	Input of pulse to be measured.
ICIA	Starts pulse cycle measurement at rise of pulse.
ICRA	Detects counter value at rise of pulse.

Operation

Figure 3 explains how the measurement is carried out. The H8/330 uses both hardware and software operations to measure pulse cycle length.

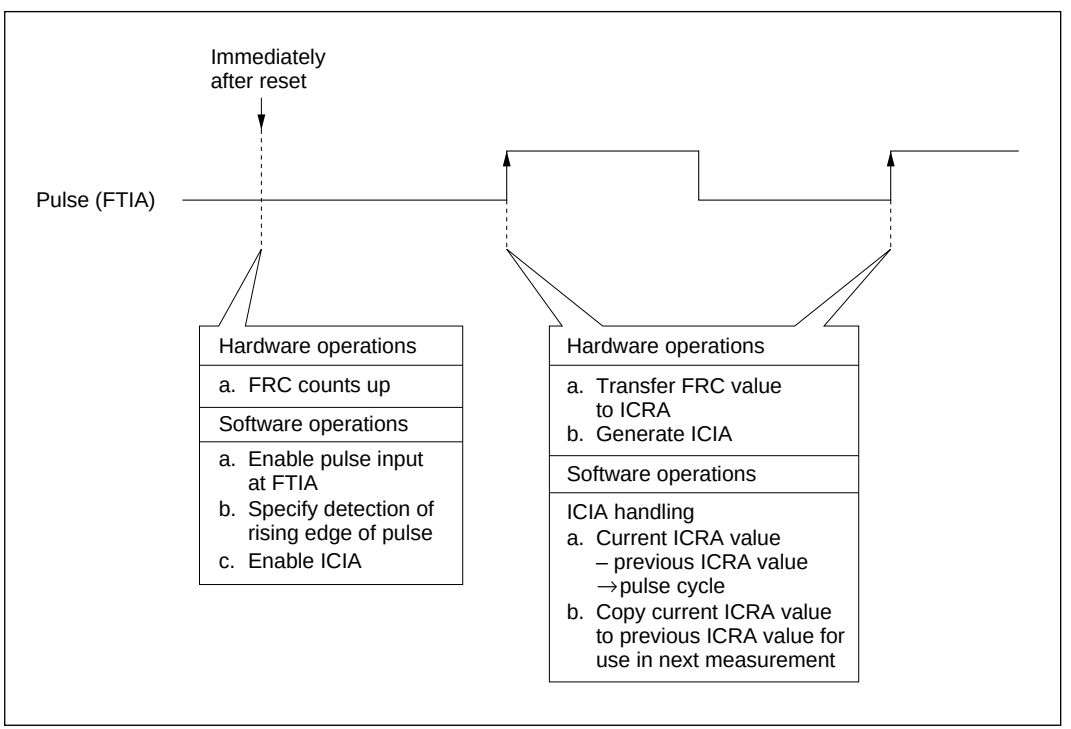


Figure 3 Pulse Cycle Measurement

Software Description

1. Module descriptions

Module	Label	Function
Main program	PWMN	Initializes 16-bit free-running timer and RAM.
Measure pulse cycle	PWINT	ICIA interrupt handler: uses ICRA value to measure pulse cycle.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
PWI_CYCDATA	Stores timer value representing pulse cycle Pulse cycle (ns) = timer value × system clock cycle (100 ns for 10-MHz clock) × 2 (frequency division factor of clock input to FRC)	1 word	Measure pulse cycle	Output

3. On-chip register usage

Register	Function	Modules Where Used
TCR	Designates transfer of FRC value to ICRA when rising edge of pulse is detected.	Main program
TIER	Enables ICIA.	Main program, measure pulse cycle
ICRA	Stores FRC value at rise of pulse; pulse cycle is found from this value.	Measure pulse cycle

4. General register usage

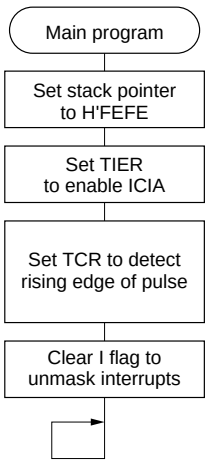
Module	Register	Function
Main program	R0	Used as work register for setting data.
Measure pulse cycle	R0	Used as work register for setting data.

5. RAM usage

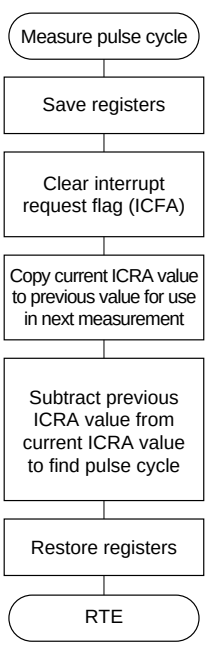
Label	Function	Word Length	Modules Where Used
PWI_OLDDATA	Stores previous ICRA value.	1 word	Measure pulse cycle

Flowcharts

1. Main program



2. Measure pulse cycle



Program List

Program List

Specification

1. This sample task measures the time from the rise of a pulse signal to the next rise (the pulse cycle) as shown in figure 1, and stores the result in RAM.

2. With a 10-MHz system clock, pulse cycles from 5 µs to 13.1 ms can be measured in 200-ns steps.

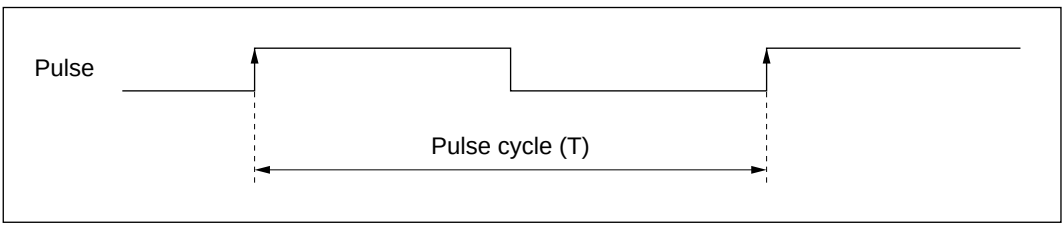


Figure 1 Pulse Cycle Measurement

Usage of Functions

1. Figure 2 shows a block diagram of the 16-bit free-running timer used by this sample task. The 16-bit free-running timer has the following functions:

   a. An input-capture function that detects pulse edges and stores the corresponding timer value in an on-chip register.

   b. A buffer mode that retains the previous timer value when a pulse edge is detected.

   This sample task uses these functions as shown in figure 2 to measure pulse cycle length.

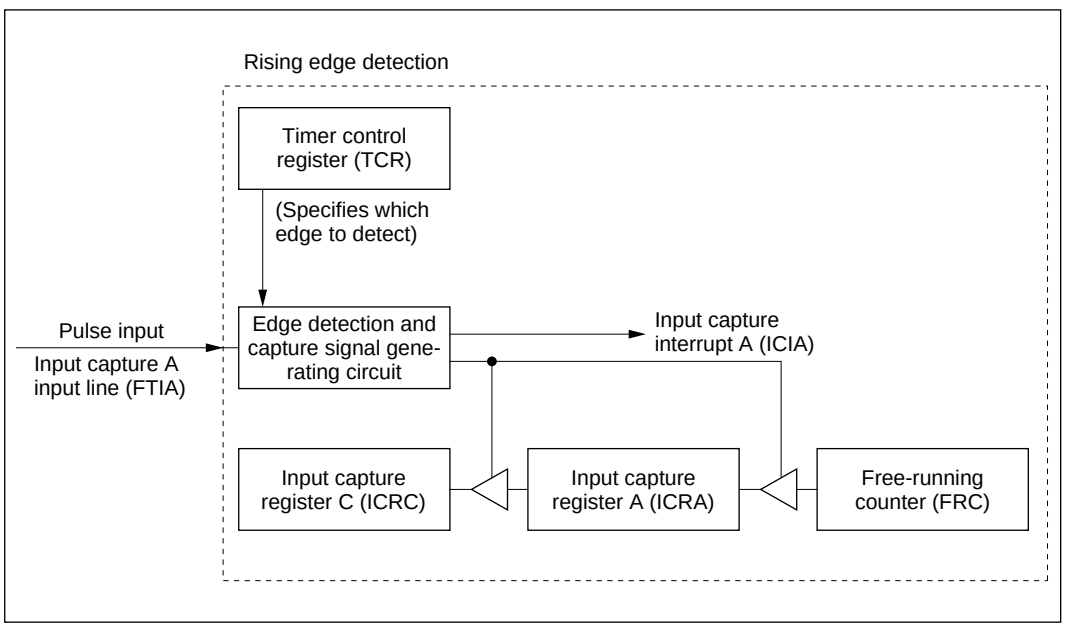


Figure 2 16-Bit Free-Running Timer Block Diagram

Usage of Functions

2. Table 1 lists the function assignments for this sample task. Functions of the 16-bit free-running timer are assigned to measure pulse cycle length.

Table 1 16-Bit Free-Running Timer Function Assignments

16-Bit Free-Running Timer Function	Description
FTIA	Input of pulse to be measured.
ICIA	Starts pulse cycle measurement at rise of pulse.
ICRA	Detects counter value at rise of pulse.
ICRC	Holds counter value at previous rise of pulse.

Operation

Figure 3 explains how the measurement is carried out. The H8/330 uses both hardware and software operations to measure pulse cycle length.

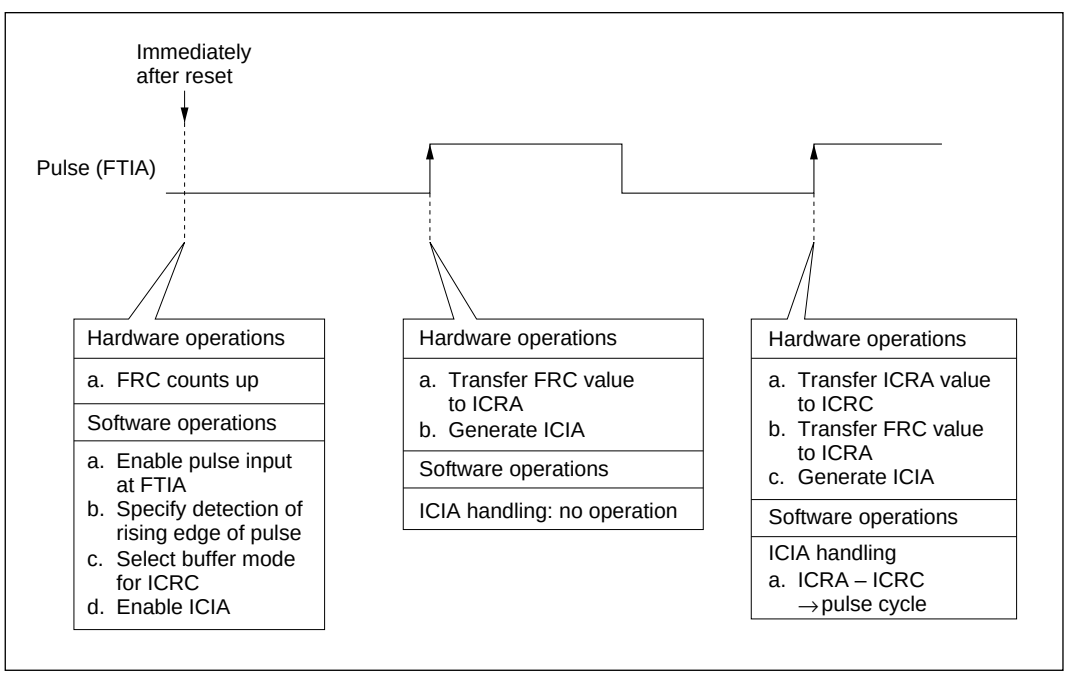


Figure 3 Pulse Cycle Measurement

Software Description

1. Module descriptions

Module	Label	Function
Main program	PWBMN	Initializes 16-bit free-running timer and RAM.
Measure pulse cycle in buffer mode	PWBI	ICIA interrupt handler: uses ICRA value to measure pulse cycle.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
PWB_CYCDATA	Stores timer value representing pulse cycle. Pulse cycle is calculated as follows: Pulse cycle (ns) = timer value × system clock cycle (100 ns for 10-MHz clock) × 2 (frequency division factor of clock input to FRC)	1 word	Measure pulse cycle in buffer mode	Output

3. On-chip register usage

Register	Function	Modules Where Used
TCR	Designates buffer mode: when rising edge of pulse is detected, ICRA value is transferred to ICRC and FRC timer value is transferred to ICRA.	Main program
TIER	Enables ICIA.	Main program, measure pulse cycle in buffer mode
ICRA ICRC	Stores FRC value at rise of pulse; pulse cycle is found from this value.	Measure pulse cycle in buffer mode

4. General register usage

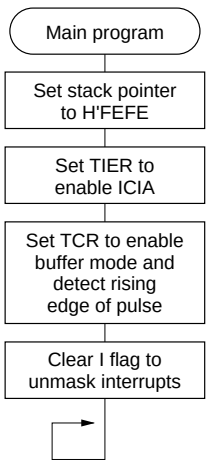
Module	Register	Function
Main program	R0	Used as work register for setting data.
Measure pulse cycle in buffer mode	R0	Used as work register for setting data.

5. RAM usage

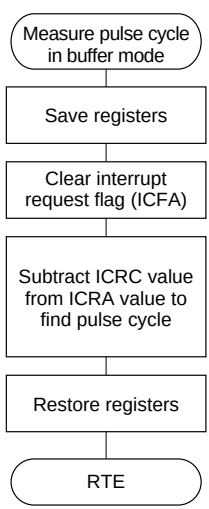
This sample task does not use RAM except for arguments.

Flowcharts

1. Main program



2. Measure pulse cycle in buffer mode



Program List

Program List

Specification

1. This sample task measures high and low pulse widths as shown in figure 1, and stores the results in RAM.

2. With a 10-MHz system clock, high and low pulse widths from 13 µs to 13.1 ms can be measured in 200-ns steps.

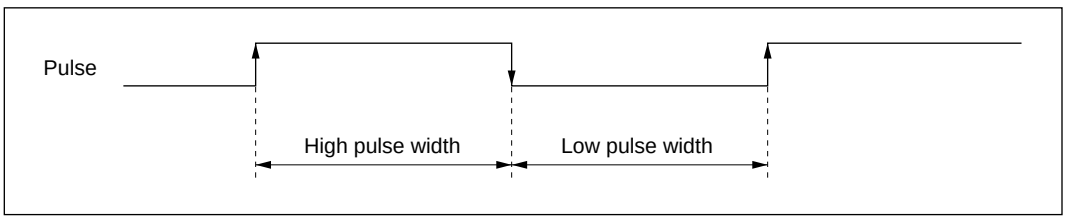


Figure 1 Pulse Width Measurement

Usage of Functions

1. Figure 2 shows a block diagram of the 16-bit free-running timer used by this sample task. The 16-bit free-running timer has the following functions:

   a. An input-capture function that detects rising and falling pulse edges and stores the corresponding timer value in an on-chip register.

   b. An interrupt function that starts interrupt handling when a rising or falling pulse edge is detected.

   This sample task uses these functions as shown in figure 2 to measure high and low pulse width.

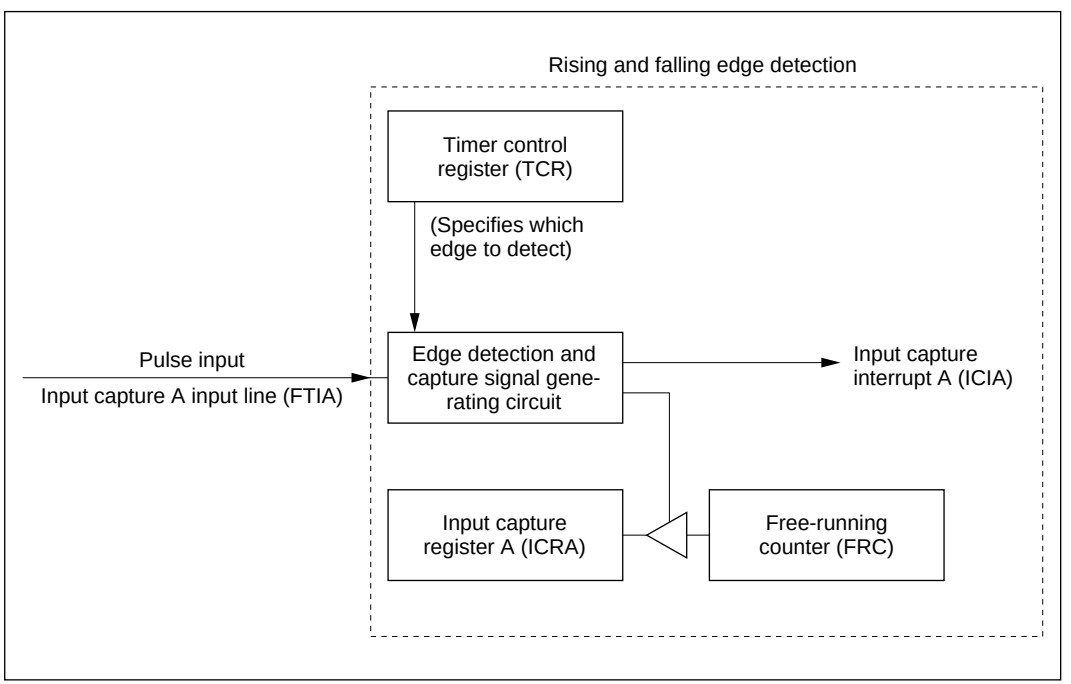


Figure 2 16-Bit Free-Running Timer Block Diagram

Usage of Functions

2. Table 1 lists the function assignments for this sample task. Functions of the 16-bit free-running timer are assigned to measure high and low pulse width.

Table 1 16-Bit Free-Running Timer Function Assignments

16-Bit Free-Running Timer Function	Description
FTIA	Input of pulse to be measured.
ICIA	Starts high and low pulse width measurement at rise and fall of pulse.
ICRA	Detects counter value at rise and fall of pulse.

Operation

Figure 3 explains how the measurement is carried out. The H8/330 uses both hardware and software operations to measure high and low pulse width.

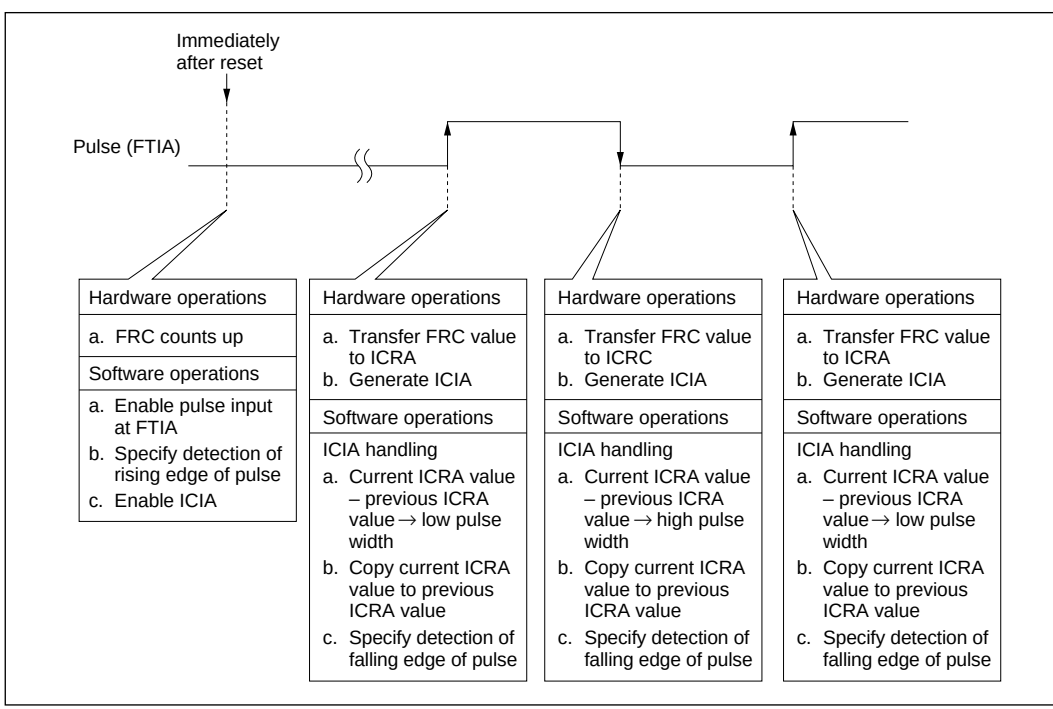


Figure 3 High and Low Pulse Width Measurement

Software Description

1. Module descriptions

Module	Label	Function
Main program	PWHLMN	Initializes 16-bit free-running timer and RAM.
Measure high and low pulse width	PWHLI	ICIA interrupt handler: uses ICRA value to measure high and low pulse width.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
PWH_HDATA	Stores timer value representing high pulse width. High pulse width is calculated as follows: High pulse width (ns) = timer value × system clock cycle (100 ns for 10-MHz clock) × 2 (frequency division factor of clock input to FRC)	1 word	Measure high and low pulse width	Output
PWH_LDATA	Stores timer value representing low pulse width. Low pulse width is calculated as follows: Low pulse width (ns) = timer value × system clock cycle (100 ns for 10-MHz clock) × 2 (frequency division factor of clock input to FRC)	1 word		

3. On-chip register usage

Register	Function	Modules Where Used
TCR	Designates transfer of FRC value to ICRA when rising and falling edges of pulse are detected.	Main program, measure high and low pulse width
TIER	Enables ICIA.	Main program
ICRA	Stores FRC value at rise and fall of pulse; high and low pulse widths are found from this value.	Measure high and low pulse width

Software Description

4. General register usage

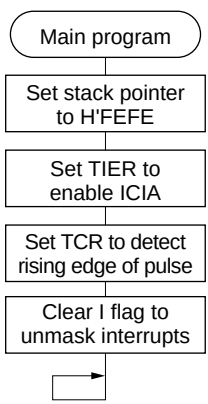
Module	Register	Function
Main program	R0	Used as work register for setting data.
Measure high and low pulse width	R0, R1	Used as work registers for setting data.

5. RAM usage

Label	Function	Word Length	Modules Where Used
PWH_OLDDATA	Stores previous ICRA value.	1 word	Measure high and low pulse width

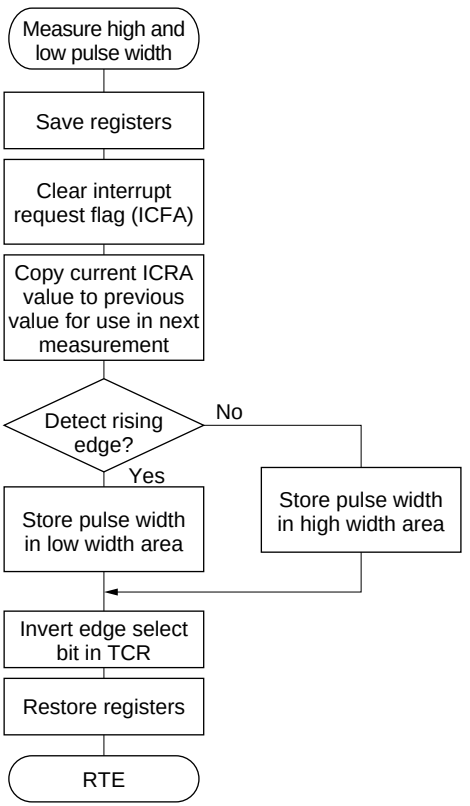
Flowcharts

1. Main program



Flowcharts

2. Measure high and low pulse width



Program List

Program List

Specification

This sample task outputs a pulse with a 50% duty cycle as shown in figure 1. With a 10-MHz system clock, the pulse cycle can be selected in the range from 18 µs to 13.1 ms in 200-ns steps. The number of pulses output can be selected in the range from 1 to 256.

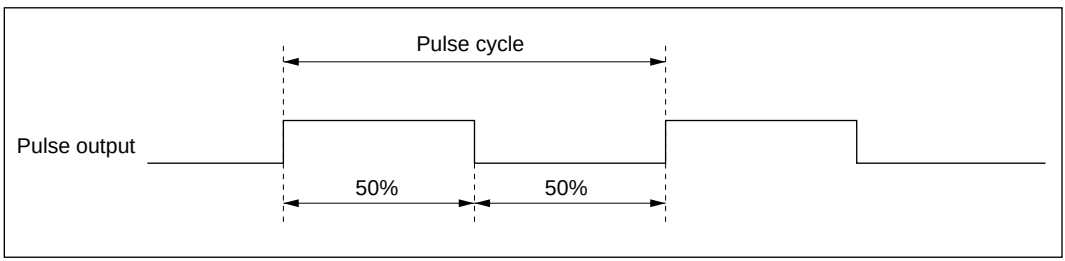


Figure 1 Pulse Output Timing

Usage of Functions

1. Figure 2 shows a block diagram of the 16-bit free-running timer used by this sample task. The 16-bit free-running timer has the following function:

   a. An output compare function by which hardware outputs pulses automatically without software intervention.

   This task uses the preceding function for pulse output as shown in figure 2.

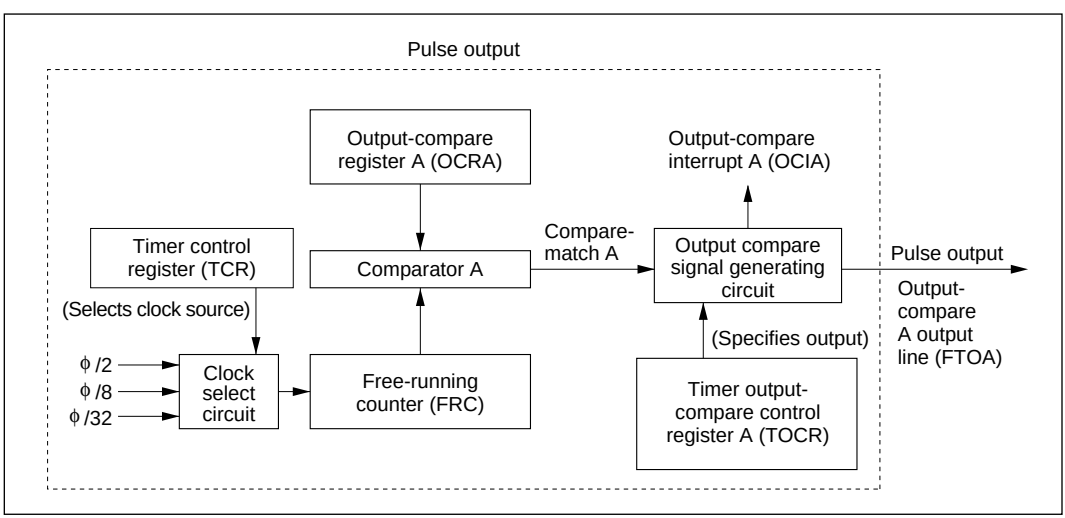


Figure 2 16-Bit Free-Running Timer Block Diagram

Usage of Functions

2. Table 1 lists the function assignments for this sample task. Functions of the 16-bit free-running timer are assigned for pulse output.

Table 1 16-Bit Free-Running Timer Function Assignments

16-Bit Free-Running Timer Function	Description
FTOA	Pulse output
OCRA	Specifies pulse cycle length
OCIA	Switches pulse output level

Operation

Figure 3 explains how pulses are output. The H8/330 uses both hardware and software operations for pulse output.

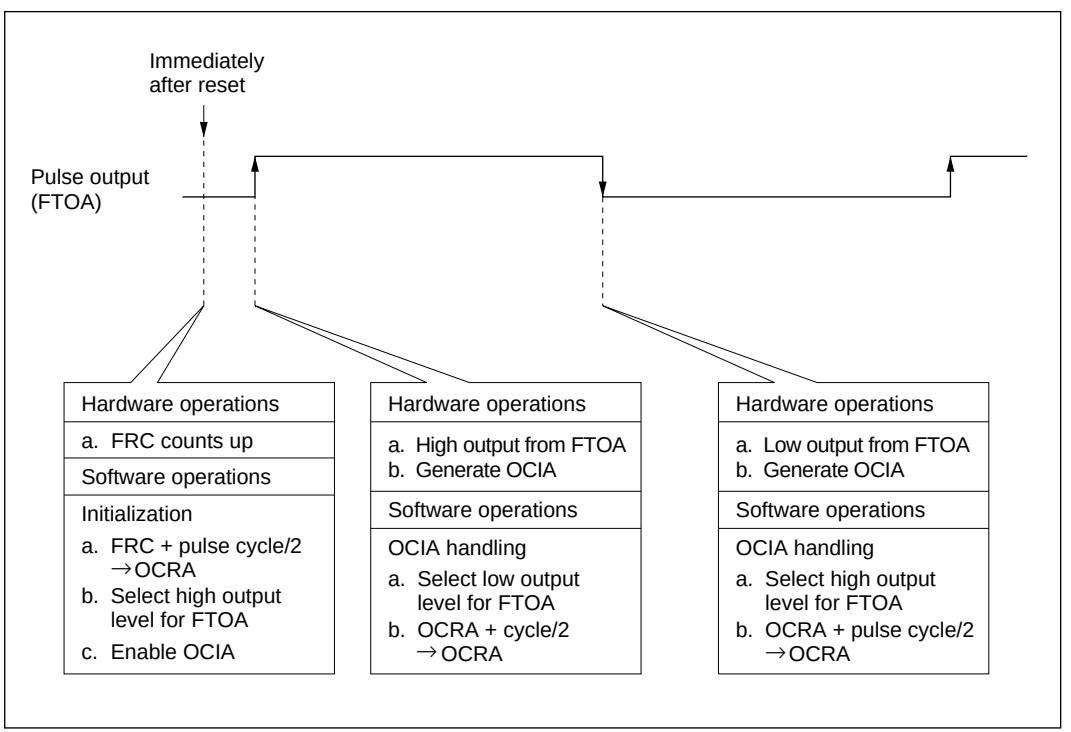


Figure 3 Pulse Output

Software Description

1. Module descriptions

Module	Label	Function
Main program	POUTMN	Initializes 16-bit free-running timer and RAM.
Output pulse	POUTI	Inverts pulse output level and counts number of pulses.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
PUT_CYC	Stores timer value representing pulse cycle. Pulse cycle is calculated as follows: Pulse cycle (ns) = timer value × system clock cycle (100 ns for 10-MHz clock) × 2 (frequency division factor of clock input to FRC)	1 word	Main program	Input
PUT_CNT	Stores output pulse count.	1 byte	Main program	Output
	Actual number of pulses output is one less than designated count.		Output pulse	Input

3. On-chip register usage

Register	Function	Modules Where Used
TCR	Selects clock input to FRC.	Main program
TIER	Enables OCIA.	Main program
TOCR	Selects pulse output level and enables output.	Main program, output pulse
OCRA	Holds half-cycle time of pulse.	Output pulse

4. General register usage

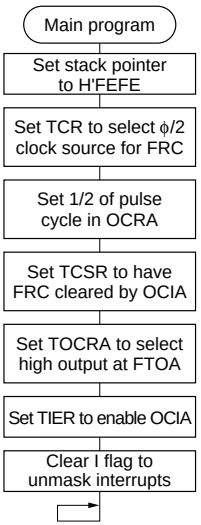
Module	Register	Function
Main program	R0	Used as work register for setting data.
Output pulse	R0	Used as work register for setting data.

5. RAM usage

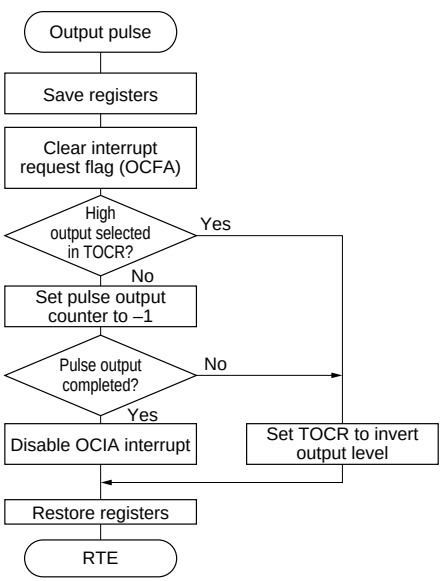
This sample task does not use RAM except for arguments.

Flowcharts

1. Main program



2. Output pulse



Program List

Program List

Specification

1. This sample task outputs a one-shot pulse delayed with respect to a triggering pulse as shown in figure 1.

2. Triggering pulses can be input with a frequency of 80 Hz to 45 kHz.

3. The delay time and pulse width can be varied within the following ranges:

   10 µs ≤ delay < triggering pulse cycle – pulse width
   11 µs ≤ pulse width < triggering pulse cycle – delay

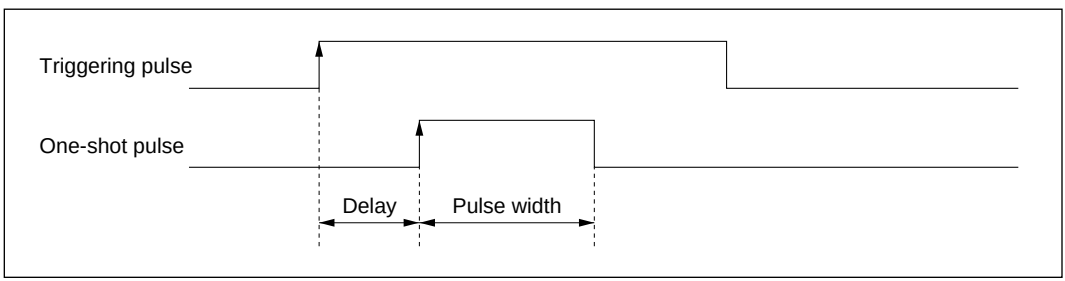


Figure 1 One-Shot Pulse Output Timing

Usage of Functions

1. Figure 2 shows a block diagram of the 16-bit free-running timer used by this sample task. The 16-bit free-running timer has the following functions:

   a. An input-capture function that detects pulse edges and stores the corresponding timer value in an on-chip register.

   b. An output-compare function by which hardware outputs pulses automatically without software intervention.

   This task uses the preceding functions for one-shot pulse output as shown in figure 2.

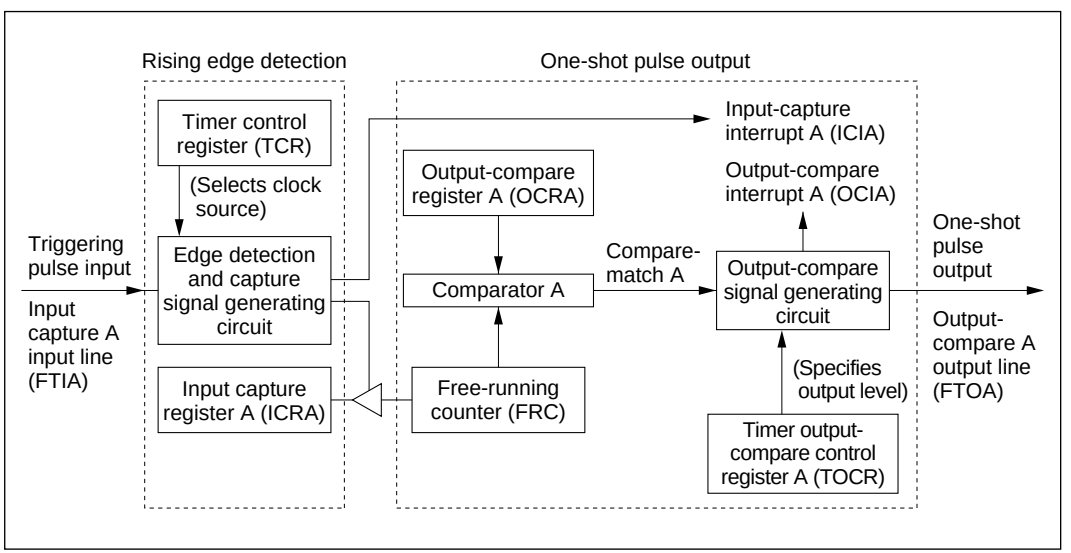


Figure 2 16-Bit Free-Running Timer Block Diagram

Usage of Functions

2. Table 1 lists the function assignments for this sample task. Functions of the 16-bit free-running timer are assigned for one-shot pulse output.

Table 1 16-Bit Free-Running Timer Function Assignments

16-Bit Free-Running Timer Function	Description
FTIA	Input of triggering pulse for one-shot pulse output.
ICIA	Rise of input pulse triggers output of one-shot pulse.
ICRA	Detects time of rise of input pulse.
FTOA	One-shot pulse output.
OCRA	Delay time and pulse width of one-shot pulse output.
OCIA	Switches one-shot pulse output level.

Operation

Figure 3 explains how one-shot pulses are output. The H8/330 uses both hardware and software operations for one-shot pulse output.

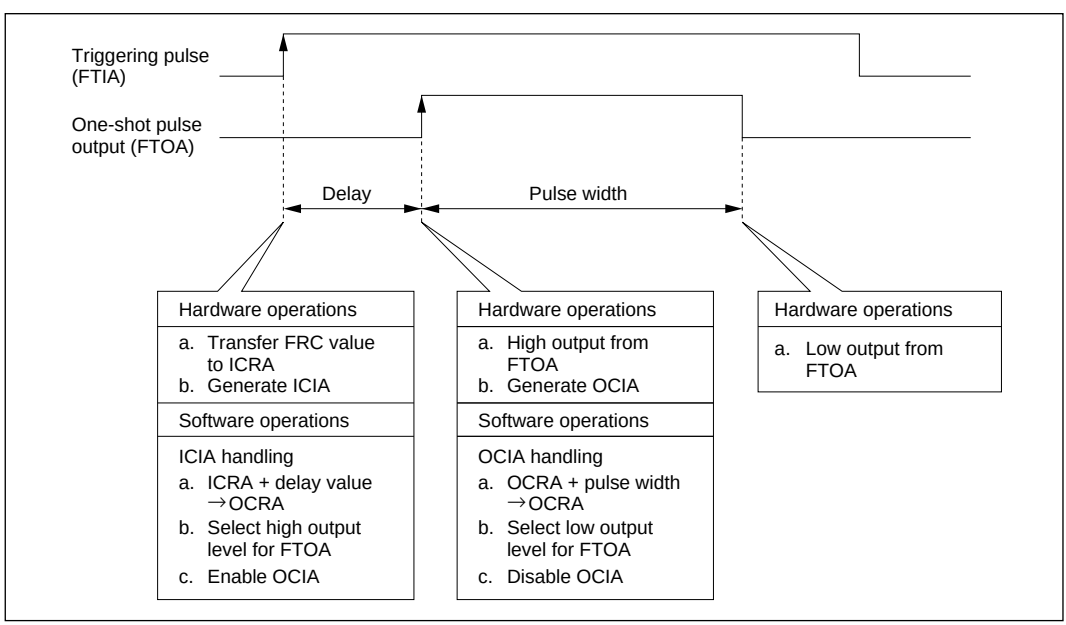


Figure 3 One-Shot Pulse Output

Software Description

1. Module descriptions

Module	Label	Function
Main program	ONEMN	Initializes 16-bit free-running timer and RAM.
Set delay time	SETDLY	Sets delay and rise of one-shot pulse.
Output one-shot pulse	ONEPOUT	Sets pulse width and fall of one-shot pulse.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
SET_DLY	Stores timer value representing one-shot pulse delay. Delay is calculated as follows: Delay (ns) = timer value × system clock cycle (100 ns for 10-MHz clock) × 2 (frequency division factor of clock input to FRC)	1 word	Main program Set delay time	Output Input
ONE_WID	Stores timer value representing pulse width of one-shot pulse. Pulse width is calculated as follows: Pulse width (ns) = timer value × system clock cycle (100 ns for 10-MHz clock) × 2 (frequency division factor of clock input to FRC)	1 word	Main program Output one-shot pulse	Output Input

3. On-chip register usage

Register	Function	Modules Where Used
TCR	Set to transfer FRC value to ICRA when rise of triggering pulse is detected.	Main program
TIER	Enables and disables ICIA and OCIA.	Main program, set delay time
TOCR	Selects one-shot pulse output level and enables output.	Main program, set delay time, output one-shot pulse
ICRA	Stores FRC value at rise of triggering pulse; this value is used to get delay of one-shot pulse.	Set delay time
OCRA	Designates times of rise and fall of one-shot pulse.	Set delay time, output one-shot pulse

4. General register usage

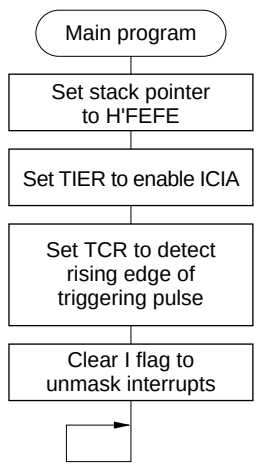
Module	Register	Function
Main program	R0	Used as work register for setting data.
Set delay time	R0, R1	Used as work register for setting data.
Output one-shot pulse	R0, R1	Used as work register for setting data.

5. RAM usage

   This sample task does not use RAM except for arguments.

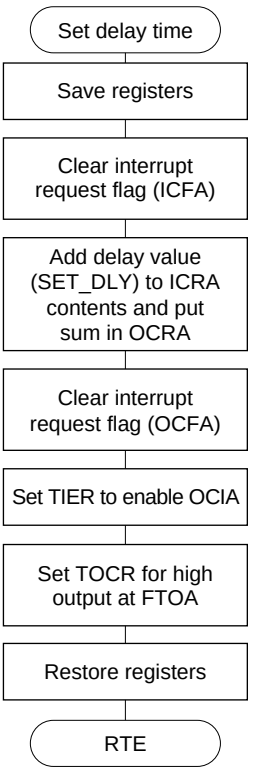
Flowcharts

1. Main program



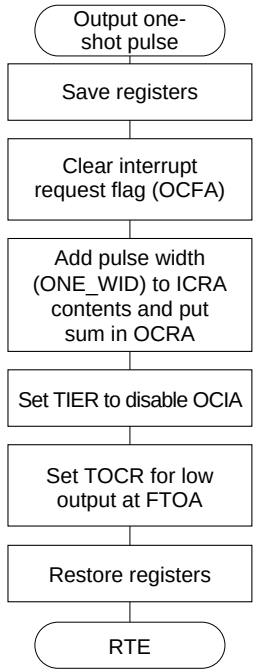
Flowcharts

2. Set delay time



Flowcharts

3. Output one-shot pulse



Program List

Program List

Specification

1. This sample task outputs a variable-duty pulse by varying the high pulse width as shown in figure 1.

2. With a 10-MHz system clock, the cycle length of the output pulse can be set from 3 µs to 200 µs.

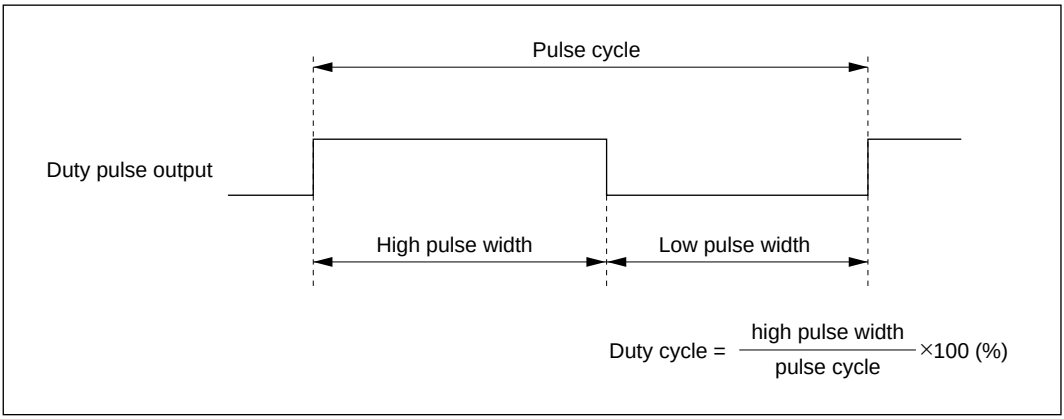


Figure 1 Duty Pulse Output Timing

Usage of Functions

1. Figure 2 shows a block diagram of the eight-bit timer used by this sample task. The eight-bit timer has the following functions:

   a. The timer counter can be cleared by a compare-match event.

   b. Timer output is controlled by a combination of two compare-match events.

   c. Hardware outputs pulses automatically without software intervention.

   This task uses the preceding functions for duty pulse output as shown in figure 2.

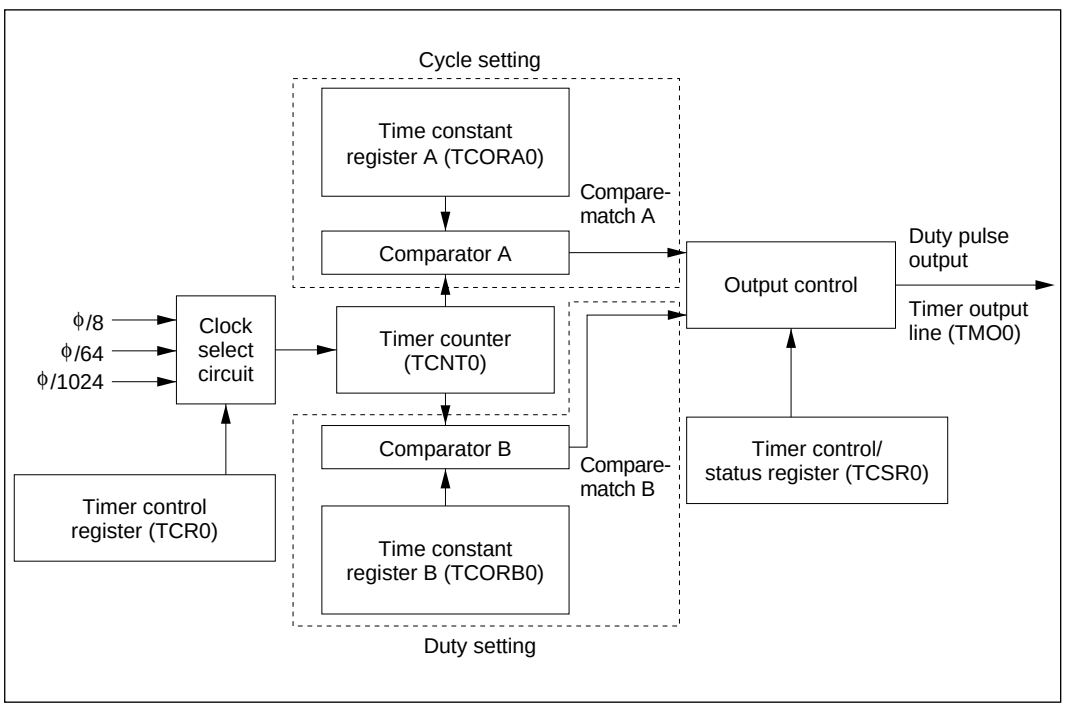


Figure 2 Eight-Bit Timer Block Diagram

Usage of Functions

2. Table 1 lists the function assignments for this sample task. Functions of the eight-bit timer are assigned for duty pulse output.

Table 1 Eight-Bit Timer Function Assignments

Eight-Bit Timer Function	Description
TMO0	Duty pulse output
TCORA0	Specifies pulse cycle length
TCORB0	Specifies duty cycle

Operation

Figure 3 explains how pulses are output. The H8/330 uses both hardware and software operations for duty pulse output.

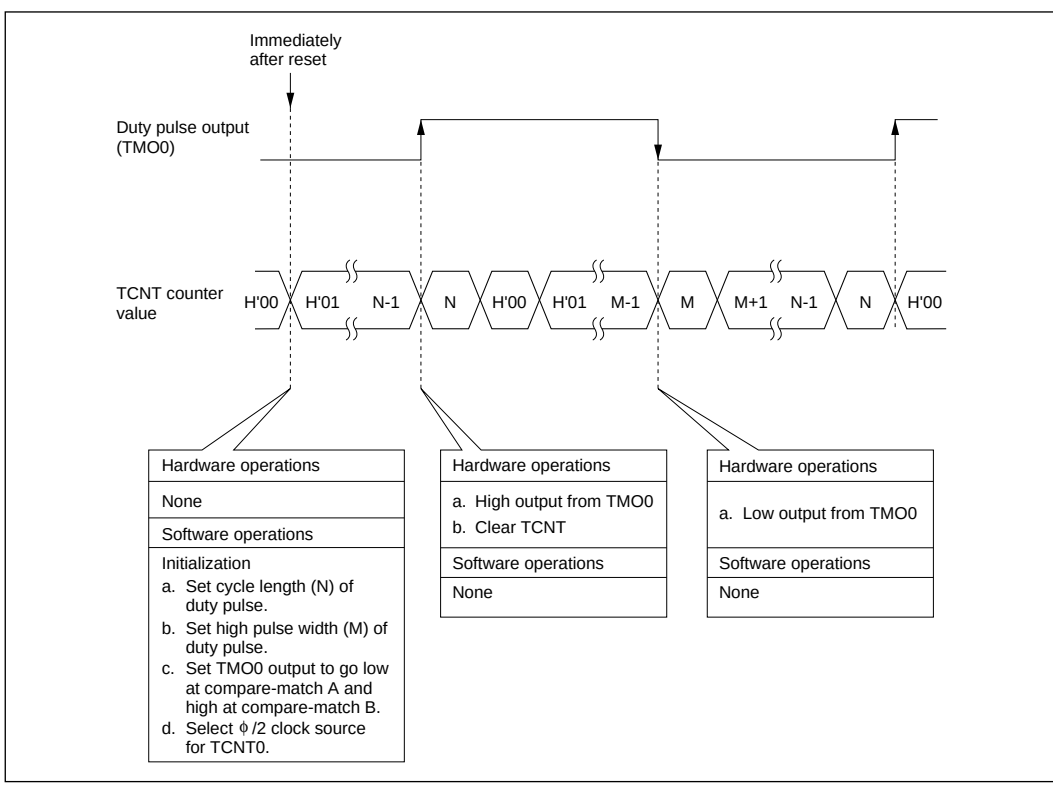


Figure 3 Duty Pulse Output

Software Description

1. Module descriptions

Module	Label	Function
Main program	DUTMN	Sets pulse cycle length and duty cycle in TCORA0 and TCORB0 and enables duty pulse output.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
DUT_DAT	Stores timer value (N) representing high pulse width. High pulse width is calculated as follows: High pulse width (ns) = timer value × system clock cycle (100 ns for 10-MHz clock) × 8 (frequency division factor of clock input to TCNT0)	1 byte	Main program	Input
DUT_TIM	Stores timer value representing pulse cycle length. Pulse cycle length is calculated as follows: Pulse cycle length (ns) = timer value × system clock cycle (100 ns for 10-MHz clock) × 8 (frequency division factor of clock input to TCNT0)	1 byte	Main program	Input

3. On-chip register usage

Register	Function	Modules Where Used
TCR0	Selects clock source for TCNT0 and sets counter clearing conditions.	Main program
TCSR0	Selects TMO0 output levels at compare-match A and B.	Main program
TCORA0	Specifies pulse cycle length.	Main program
TCORB0	Specifies high pulse width.	Main program

4. General register usage

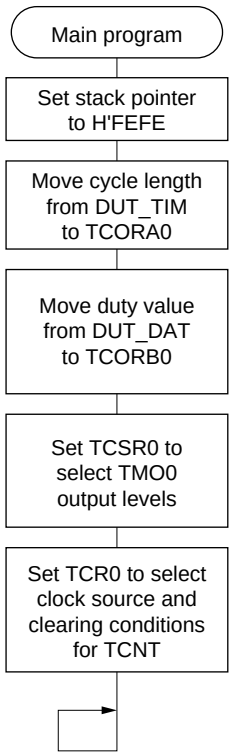
Module	Register	Function
Main program	R0L	Used as work register for setting data.

5. RAM usage

This sample task does not use RAM except for arguments.

Flowcharts

1. Main program



Program List

Program List

Specification

1. This sample task counts rising edges of a pulse signal as shown in figure 1, and stores the result in RAM.

2. With a 10-MHz system clock, the measurement time can be set from 8 µs to 13.1 ms in steps of 200 ns.

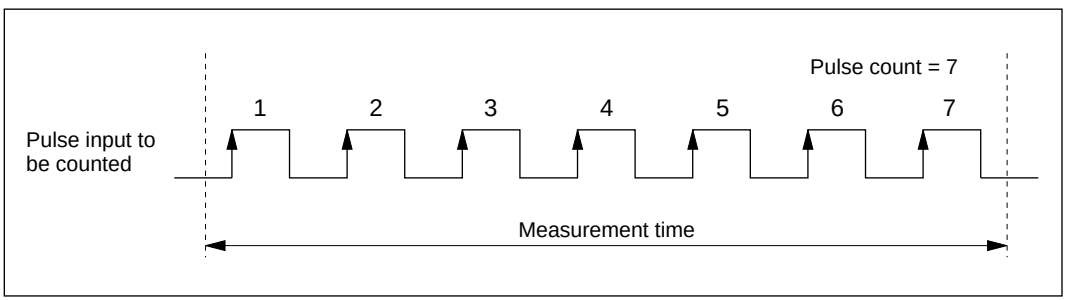


Figure 1 Pulse Counting

Usage of Functions

1. Figure 2 shows block diagrams of the eight-bit and 16-bit timers used by this sample task. These timers have the following functions:

   a. Eight-bit timer: the timer counter counts pulses input on the timer clock input line.

   b. 16-bit timer: generates an interrupt at a designated time.

   This task uses the preceding functions to count pulse rises as shown in figure 2.

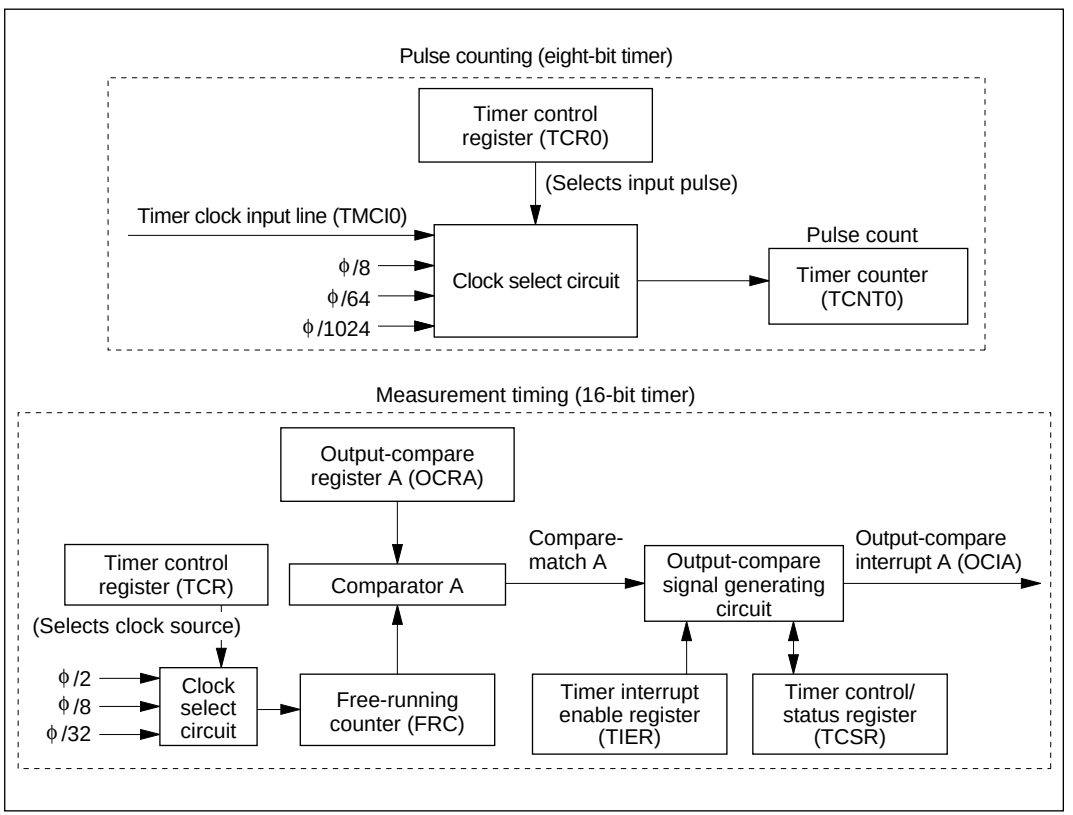


Figure 2 Eight-Bit and 16-Bit Timer Block Diagrams

Usage of Functions

2. Table 1 lists the function assignments for this sample task. Functions of the eight-bit and 16-bit timers are assigned for pulse counting.

Table 1 Eight-Bit and 16-Bit Timer Function Assignments

Eight-Bit and 16-Bit Timer Functions	Description
TMCI0	Input of pulses to be counted.
TCR0	Selects pulse input to TCNT.
TCNT	Counts rising pulse edges.
TCR	Selects clock source for FRC.
TIER	Enables OCIA.
TCSR	Clears interrupt request flag (OCFA).

Operation

Figure 3 explains how pulses are counted. The H8/330 uses both hardware and software operations for pulse counting.

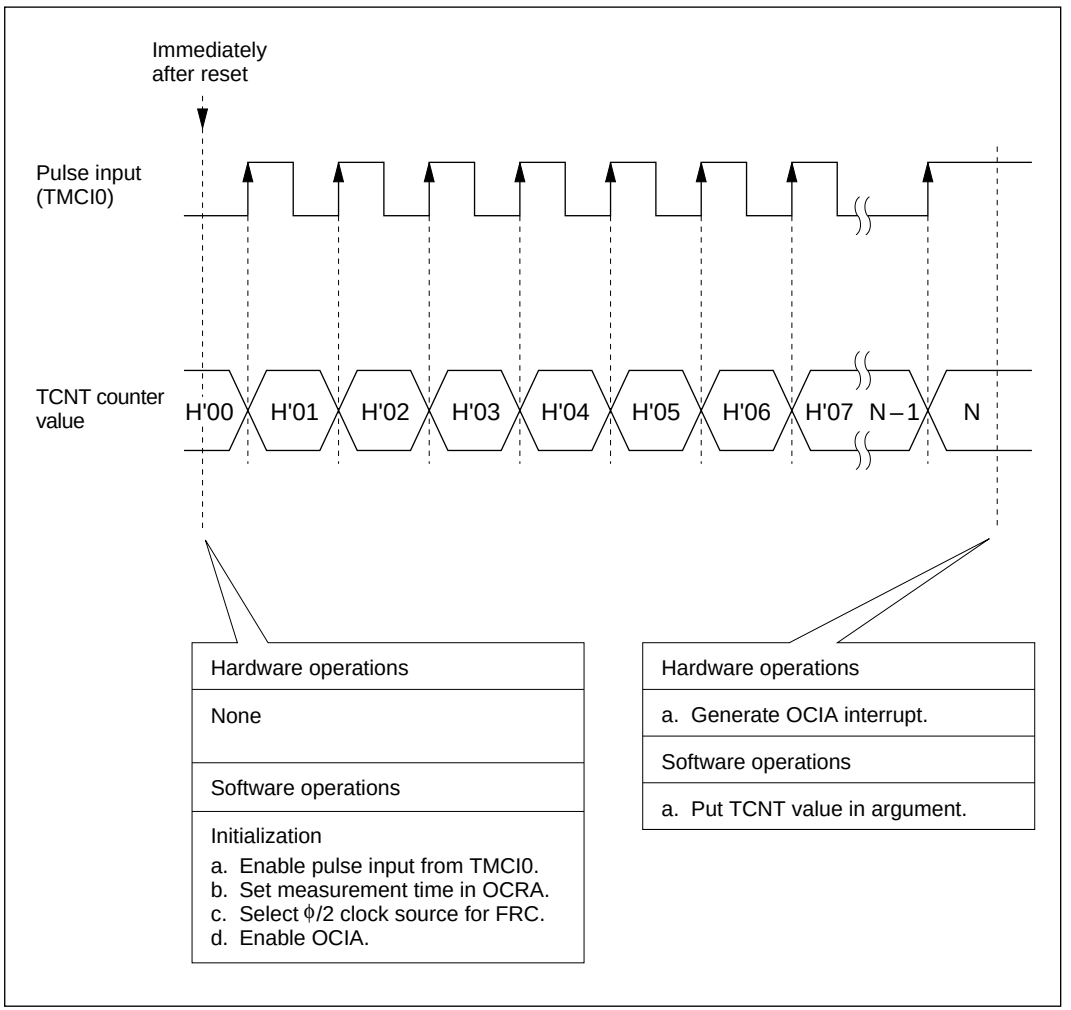


Figure 3 Pulse Counting

Software Description

1. Module descriptions

Module	Label	Function
Main program	PCTMN	Initializes eight-bit and 16-bit timers.
Detect measurement end	PCTEND	OCIA interrupt handler: moves pulse count from TCNT to argument.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
PCT_MEASTIM	Stores timer value representing pulse measurement time. Measurement time is calculated as follows: Measurement time (ns) = timer value × system clock cycle (100 ns for 10-MHz clock) × 2 (frequency division factor of clock input to FRC)	1 word	Main program	Input
PCT_CNT	Stores measured pulse count.	1 byte	Detect measurement end	Output
			Main program	Input

3. On-chip register usage

Register	Function	Modules Where Used
TCR0	Selects pulse input to TCNT.	Main program
TCNT	Counts rising pulse edges.	Main program, detect measurement end
TCR	Selects clock source for FRC.	Main program
TIER	Enables OCIA.	Main program
TCSR	Clears interrupt request flag (OCFA).	Main program

4. General register usage

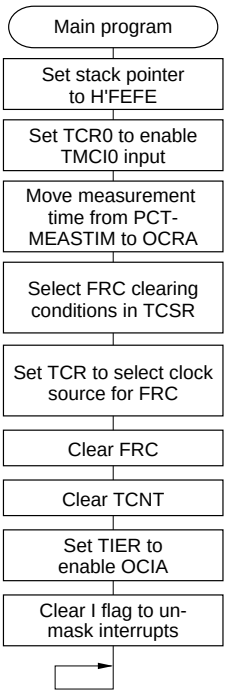
Module	Register	Function
Main program	R0	Used as work register for setting data.
Detect measurement end	R0L	Used as work register for setting data.

5. RAM usage

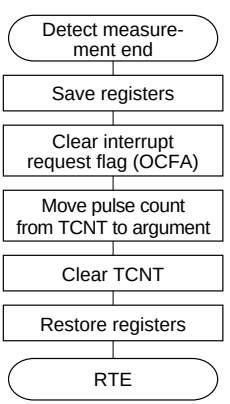
This sample task does not use RAM except for arguments.

Flowcharts

1. Main program



2. Detect measurement end



Program List

Program List

Specification

1. This sample task uses the H8/330's I/O ports to scan a 4×4 key matrix as shown in figure 1.
2. Simultaneous pressing of two or more keys is invalid.
3. Keys are debounced by software.

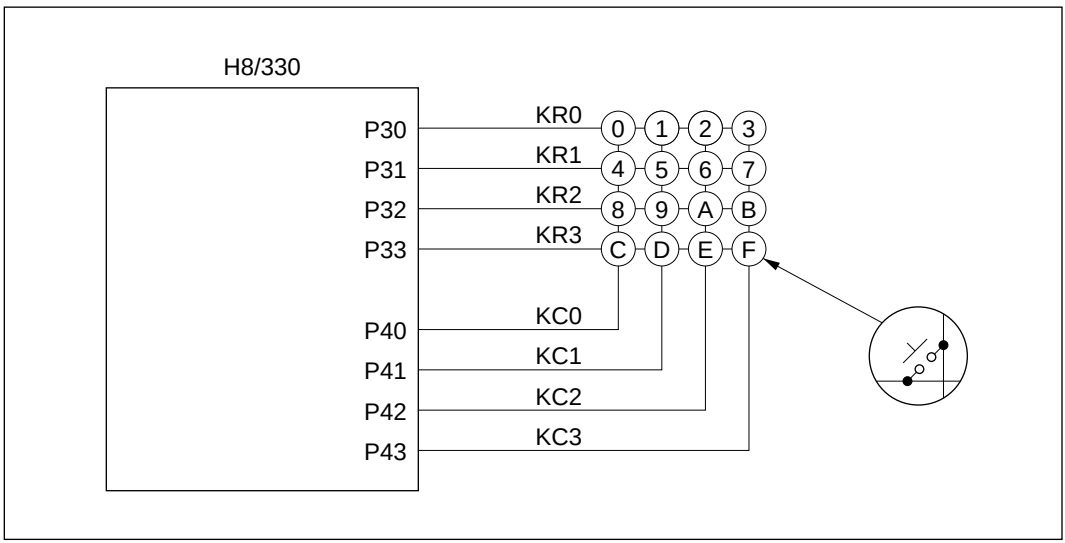


Figure 1 Key Scan of 4×4 Key Matrix

Usage of Functions

1. Figure 2 shows a block diagram of the eight-bit timer used by this sample task. The eight-bit timer has the following functions:

   a. A compare-match function that indicates when the value of a clock-driven timer up-counter (TCNT0) matches a value preset in a time constant register (TCORA0).

   b. A timer output function by which hardware automatically outputs a signal on the timer output line when a compare-match occurs.

   This task uses the compare-match function to implement an 8-ms interval timer.

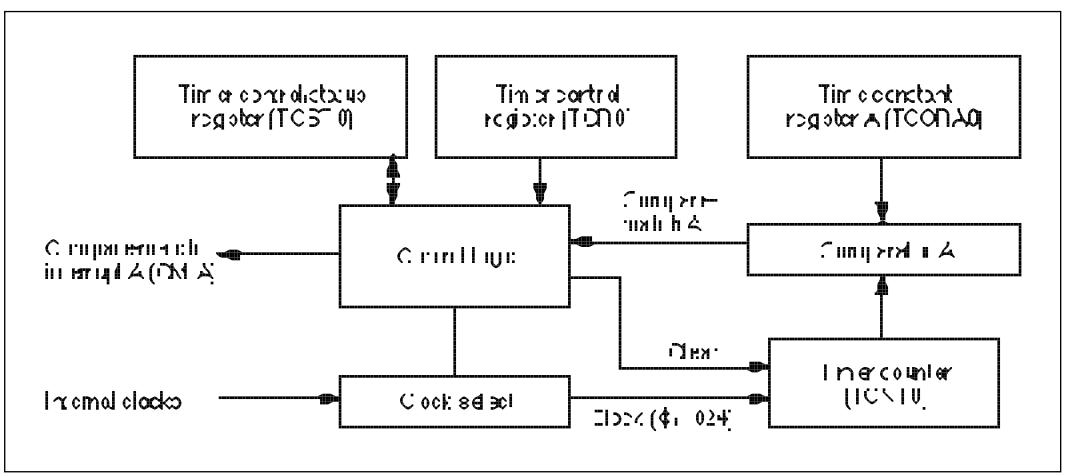


Figure 2 Eight-Bit Timer Block Diagram

Table 1 lists the function assignments for this sample task.

1. Port 3 is used as an output port to strobe the key matrix.

2. Port 4 is used as an input port to read key data.

3. The input pull-up transistors of port 4 are turned on.

4. The eight-bit timer is used as an interval timer to generate compare-match interrupts at 8-ms intervals.

Usage of Functions

Table 1 Eight-Bit Timer and I/O Port Function Assignments

Eight-Bit Timer and I/O Port Functions	Description
CMIA	Generates compare-match A interrupts at 8-ms intervals.
Port 3	Outputs strobe signals to key matrix.
Port 4	Receives key data from key matrix.

Operation

1. Key-scan operation

   Figure 3 shows the timing of the key-scan operation.

   a. P30 is driven low to strobe the KR0 row of the key matrix.

   b. When the strobe signal is output, the key data from row KR0 of the key matrix are input at P40 to P43.

   c. Key presses are detected from the input data.

   d. The strobe signal is shifted and steps a to c are repeated. The scan repeats at 8-ms intervals.

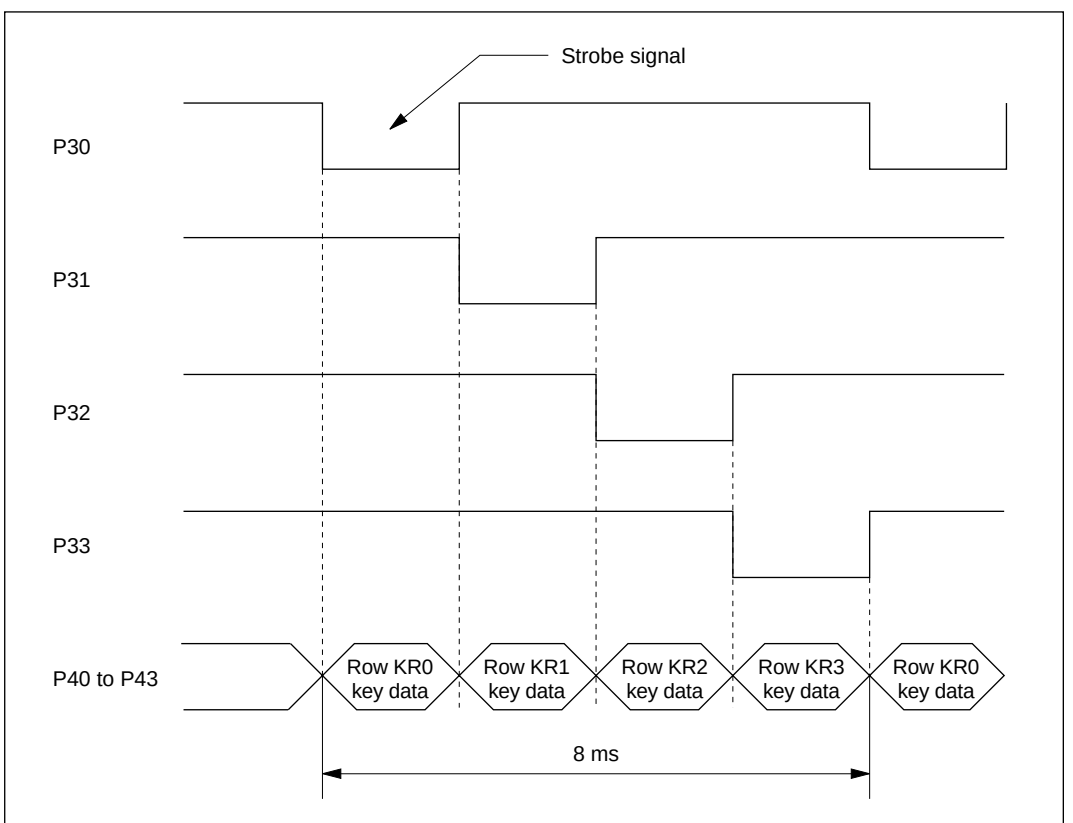


Figure 3 Key Scan Timing

2. Debouncing

Keys are debounced according to the timing diagram in figure 4.

a. Key data are sampled at 8-ms intervals.

b. A check is made for identical data three times in succession.

c. If identical data are not received three times, the key press is ignored.

d. If identical data are received three times, the key press is recognized and flagged as valid.

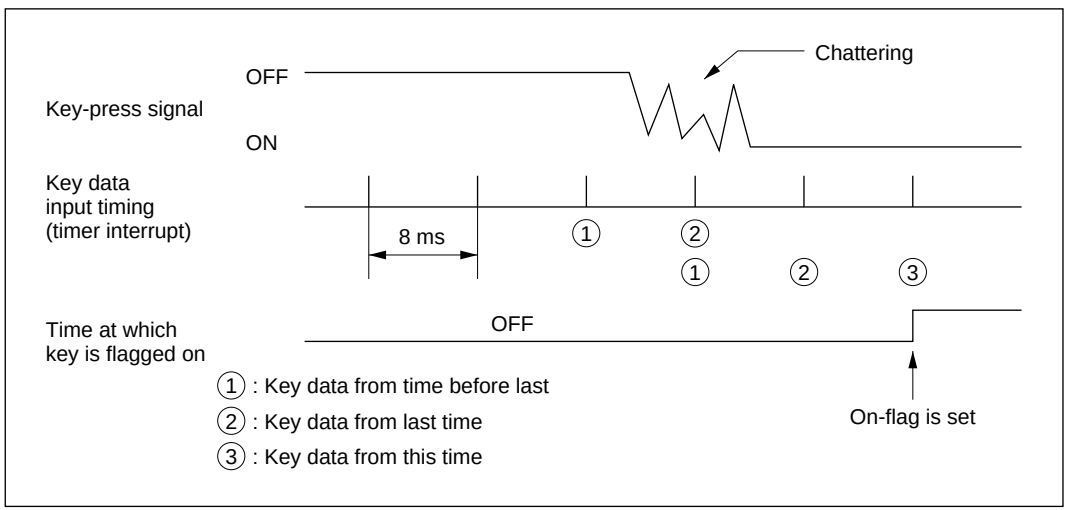


Figure 4 Debouncing Timing Diagram

Software Description

1. Module descriptions

Module	Label	Function
Main program	KEYMN	Initializes stack pointer, eight-bit timer, and I/O ports and enables interrupts.
Key scan	KEYSCN	Outputs strobe signals from port 3 at 8-ms intervals, and captures key data at port 4. Debounces keys by testing for identical key data three times in succession.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
KEY_DAT	Key data obtained from key scan	1 byte	Key scan	Output
			Main program	Input
KEY_SETF	Flag indicating valid key data	1 bit	Key scan	Output
			Main program	Input

3. On-chip register usage

Register	Function	Modules Where Used
P4DR	Turns on pull-up transistors for port 4.	Main program, key scan
P4DDR	Sets port 4 to input.	Main program
P3DR	Outputs low strobe signals from port 3.	Main program
P3DDR	Controls strobe signal output lines in port 3.	Key scan
TCORA	Sets compare-match time for eight-bit timer.	Main program
TCR	Enables eight-bit timer interrupt, and selects clock source and counter clearing conditions.	Main program
TCSR	Clears compare-match flag.	Key scan

Software Description

4. General register usage

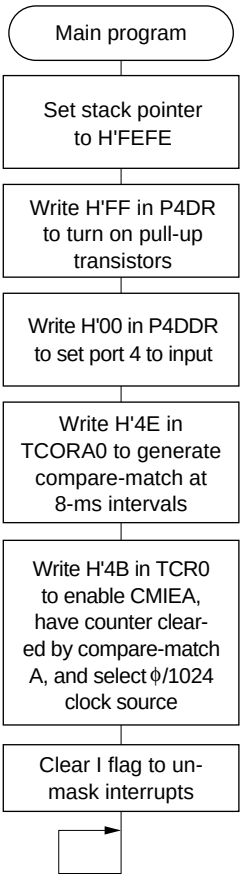
Module	Register	Function
Main program	R0L	Used as work register for setting data.
Key scan	R0L, R0H, R1H	Used as work register for setting data.
Key scan	R1L	Stores H'00 for clearing data.
Key scan	R5L	Used as counter to indicate key position.
Key scan	R5H	Used as P4DR bit shift counter.
Key scan	R6L	Used as P3DDR strobe signal shifter.
Key scan	R6H	Used for detecting key presses in key data received at P4DR.

5. RAM usage

Label	Function	Data Length	Modules Where Used
KEY_NEW	Stores new key data.	1 byte	Key scan
KEY_OLD	Stores old key data.	1 byte	Key scan
KEY_CHAC	Stores key scan count.	1 byte	Key scan
KEY_FLA	Stores KEY_SETF, KEY_CHAF, and KEY_PUSF flags.	1 byte	Key scan

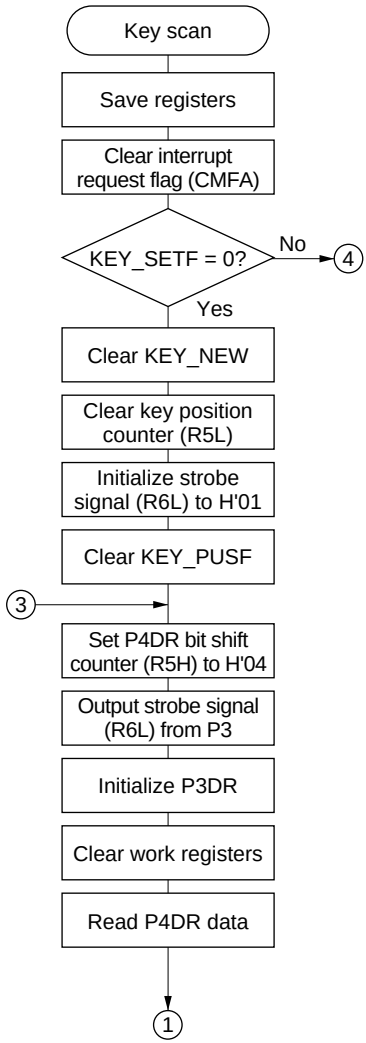
Flowcharts

1. Main program

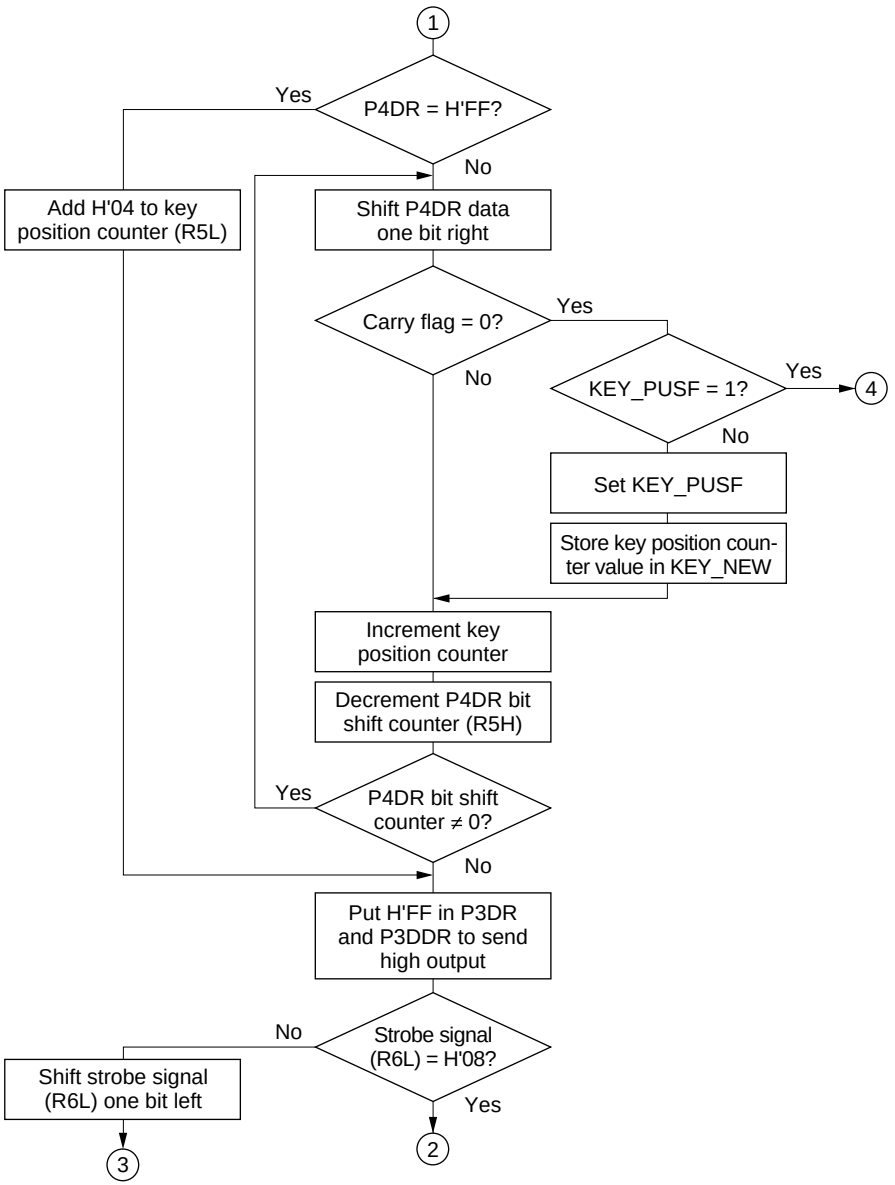


Flowcharts

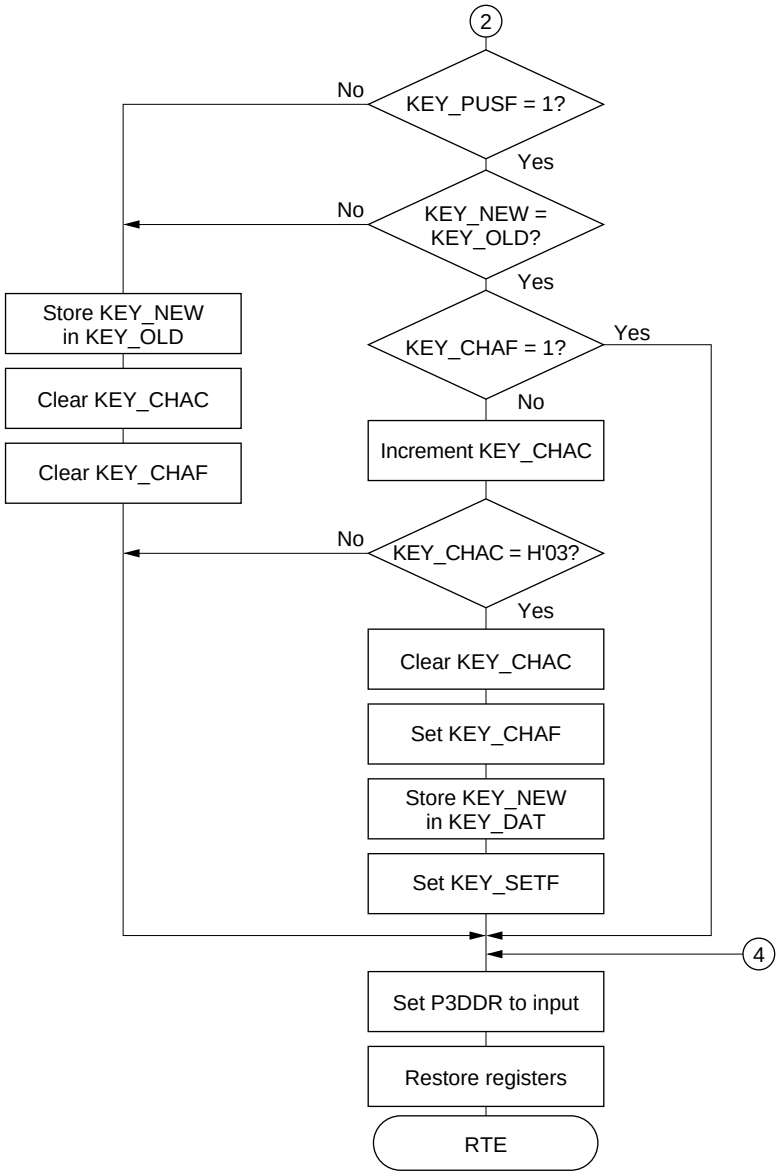
2. Key scan



Flowcharts


1
P4DR = H'FF?
Yes
No
Add H'04 to key position counter (R5L)
Shift P4DR data one bit right
Carry flag = 0?
Yes
No
KEY_PUSF = 1?
Yes
4
No
Set KEY_PUSF
Store key position counter value in KEY_NEW
Increment key position counter
Decrement P4DR bit shift counter (R5H)
P4DR bit shift counter ≠ 0?
Yes
No
Put H'FF in P3DR and P3DDR to send high output
Strobe signal (R6L) = H'08?
No
Yes
Shift strobe signal (R6L) one bit left
3
2

Flowcharts


2
KEY_PUSF = 1?
No
Yes
KEY_NEW = KEY_OLD?
No
Yes
Store KEY_NEW in KEY_OLD
KEY_CHAF = 1?
Yes
Clear KEY_CHAC
No
Clear KEY_CHAF
Increment KEY_CHAC
KEY_CHAC = H'03?
No
Yes
Clear KEY_CHAC
Set KEY_CHAF
Store KEY_NEW in KEY_DAT
Set KEY_SETF
4
Set P3DDR to input
Restore registers
RTE

Program List

Program List

Program List

Specification

1. This sample task outputs a pulse with variable cycle length and variable duty cycle as shown in figure 1.

2. The duty cycle can be can be varied from 0% to 100% with a resolution of 1/250.

3. The pulse cycle length can be set to one of eight values: 50 µs, 200 µs, 800 µs, 3.2 ms, 6.4 ms, 25.6 ms, 51.2 ms, and 102.4 ms.

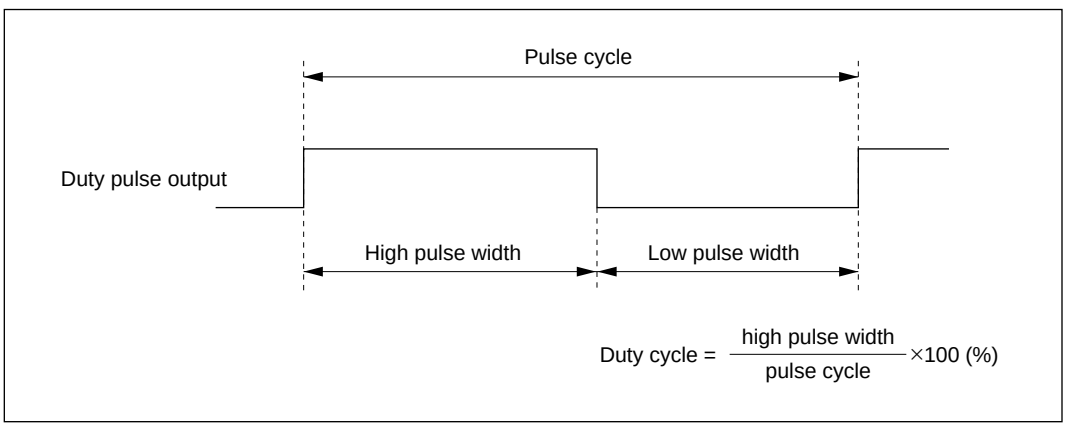


Figure 1 Duty Pulse Output Timing

Usage of Functions

1. Figure 2 shows a block diagram of the PWM timer used by this sample task. The PWM timer has the following functions:

   a. The duty cycle can be can be set from 0% to 100% with a resolution of 1/250.

   b. Eight cycle lengths can be selected, based on the system clock (ϕ).

   c. Hardware outputs duty pulses automatically without software intervention.

   This task uses the preceding functions for duty pulse output as shown in figure 2.

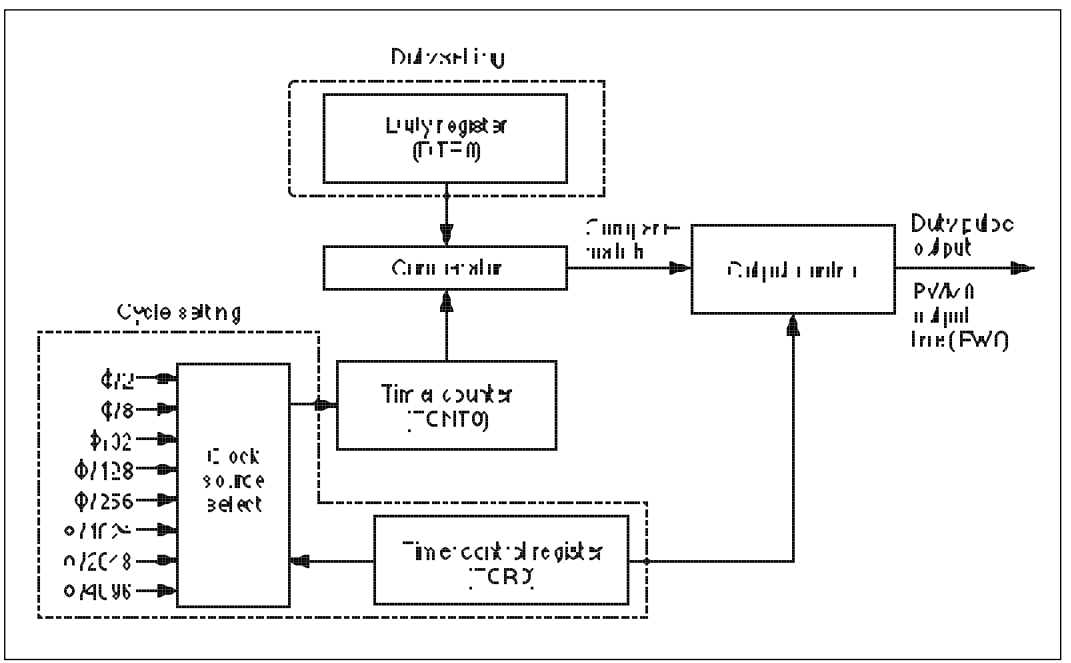


Figure 2 PWM Timer Block Diagram

Usage of Functions

2. Table 1 lists the function assignments for this sample task. Functions of the PWM timer are assigned for duty pulse output.

Table 1 PWM Timer Function Assignments

PWM Timer Function	Description
PW0	Duty pulse output
TCR0	Specifies pulse cycle length
DTR0	Specifies duty cycle

Operation

Figure 3 explains how pulses are output. The H8/330 uses both hardware and software operations for duty pulse output.

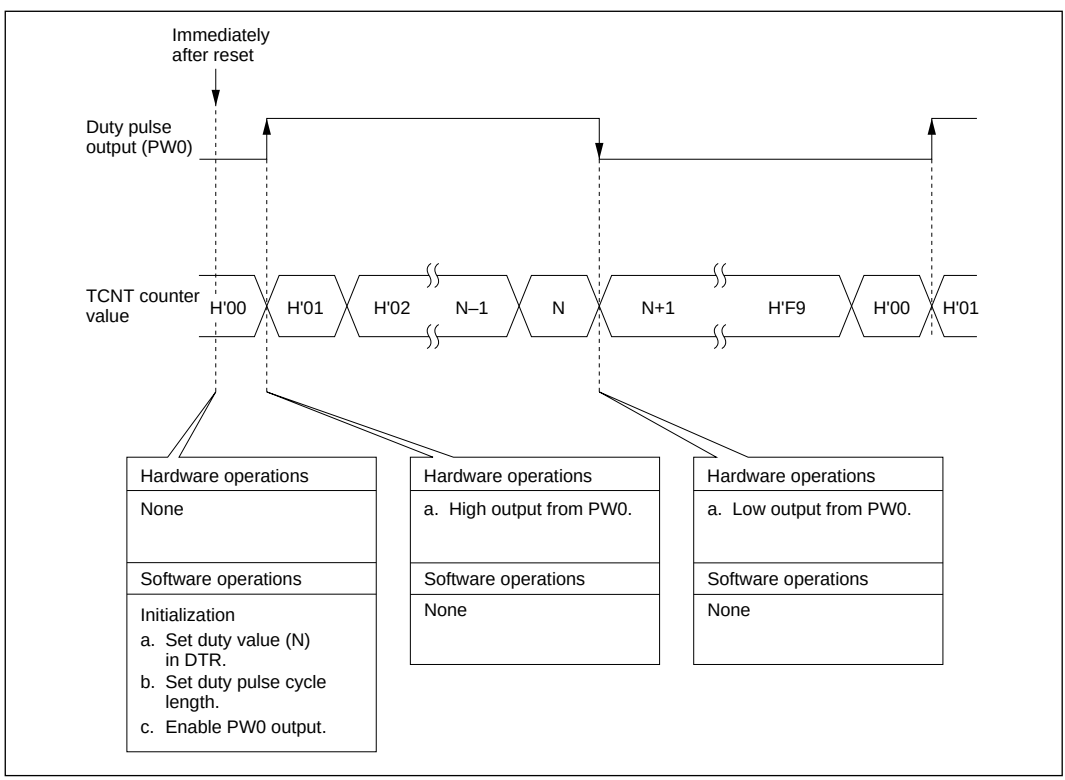


Figure 3 Duty Pulse Output by PWM Timer

Software Description

1. Module descriptions

Module	Label	Function
Main program	PWMMN	Sets duty cycle and pulse cycle length in DTR and TCR and enables duty pulse output.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
PWM_DUT	Stores timer value (N) representing duty cycle, calculated as follows and converted to hexadecimal: Timer value = desired duty cycle (%)/100 × 250	1 byte	Main program	Input
PWM_CYC	Selects one of eight pulse cycle lengths as follows: H'00: ϕ/2 (50 µs at 10 MHz) H'01: ϕ/8 (200 µs at 10 MHz) H'02: ϕ/32 (800 µs at 10 MHz) H'03: ϕ/128 (3.2 ms at 10 MHz) H'04: ϕ/256 (6.4 ms at 10 MHz) H'05: ϕ/1024 (25.6 ms at 10 MHz) H'06: ϕ/2048 (51.2 ms at 10 MHz) H'07: ϕ/4096 (102.4 ms at 10 MHz)	1 byte	Main program	Input

3. On-chip register usage

Register	Function	Modules Where Used
TCR0	Enables PW0 output and selects cycle length.	Main program
DTR0	Sets timer value representing duty cycle.	Main program

4. General register usage

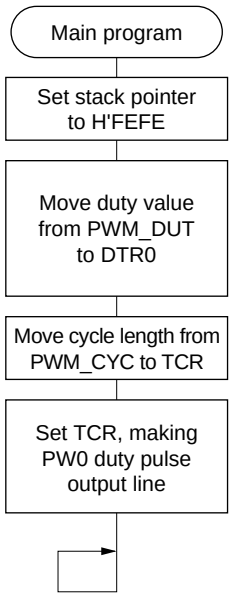
Module	Register	Function
Main program	R0	Used as work register for setting data, and as argument.

5. RAM usage

This sample task does not use RAM except for arguments.

Flowcharts

1. Main program



Program List

Program List

Specification

1. This sample task uses the H8/330's serial communication interface (SCI) in asynchronous mode to exchange one-byte data with a console as shown in figure 1.

2. The communication parameters are 9600 bps, eight data bits, one stop bit, and no parity.

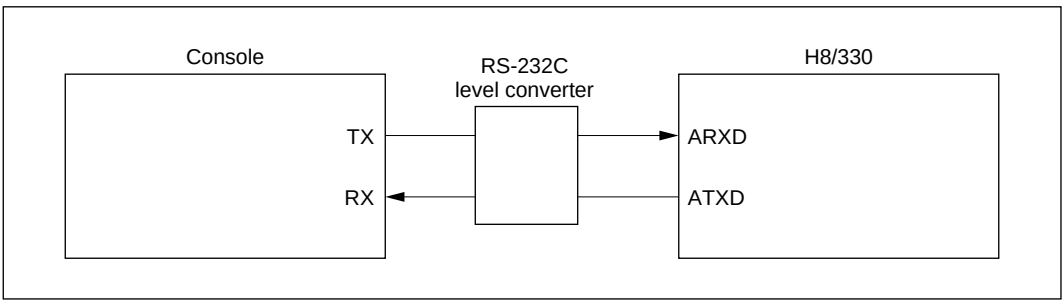


Figure 1 Asynchronous Interface using H8/330's SCI

Usage of Functions

1. Figure 2 shows a block diagram of the SCI used by this sample task. The SCI has the following features:

   a. Selection of asynchronous or clocked synchronous communication.

   b. Selection of communication speed and format.

   c. Receive-end interrupt.

   This task uses these features for console interfacing as shown in figure 2.

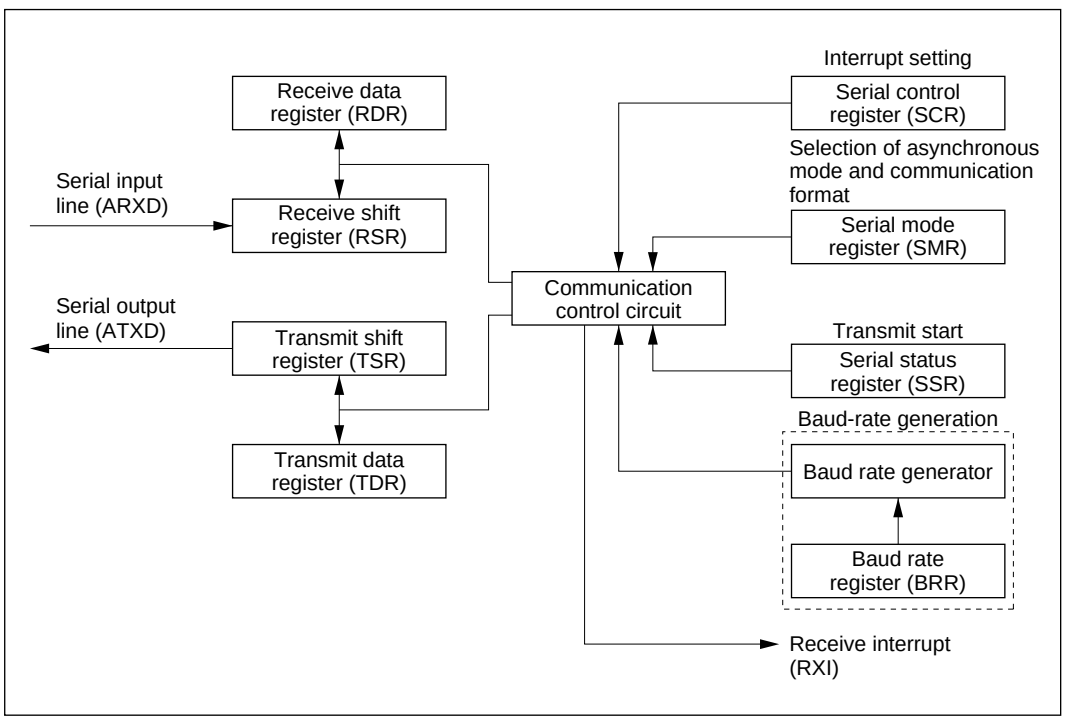


Figure 2 SCI Block Diagram

Usage of Functions

2. Table 1 lists the function assignments for this sample task. SCI functions are assigned for console interfacing.

Table 1 SCI Function Assignments

SCI Function	Description
ARXD	Receives data from console.
ATXD	Sends data to console.
SMR	Selects asynchronous mode of SCI.
SCR	Enables receive interrupt and sets SCI to transmit/receive mode.
SSR	Starts transmitting.
RDR	Stores data received from console.
TDR	Stores data to be sent to console.
BRR	Selects communication speed.

Operation

Figure 3 explains the interfacing procedure. The SCI is controlled in asynchronous mode to interface with a console with the timing shown in figure 3.

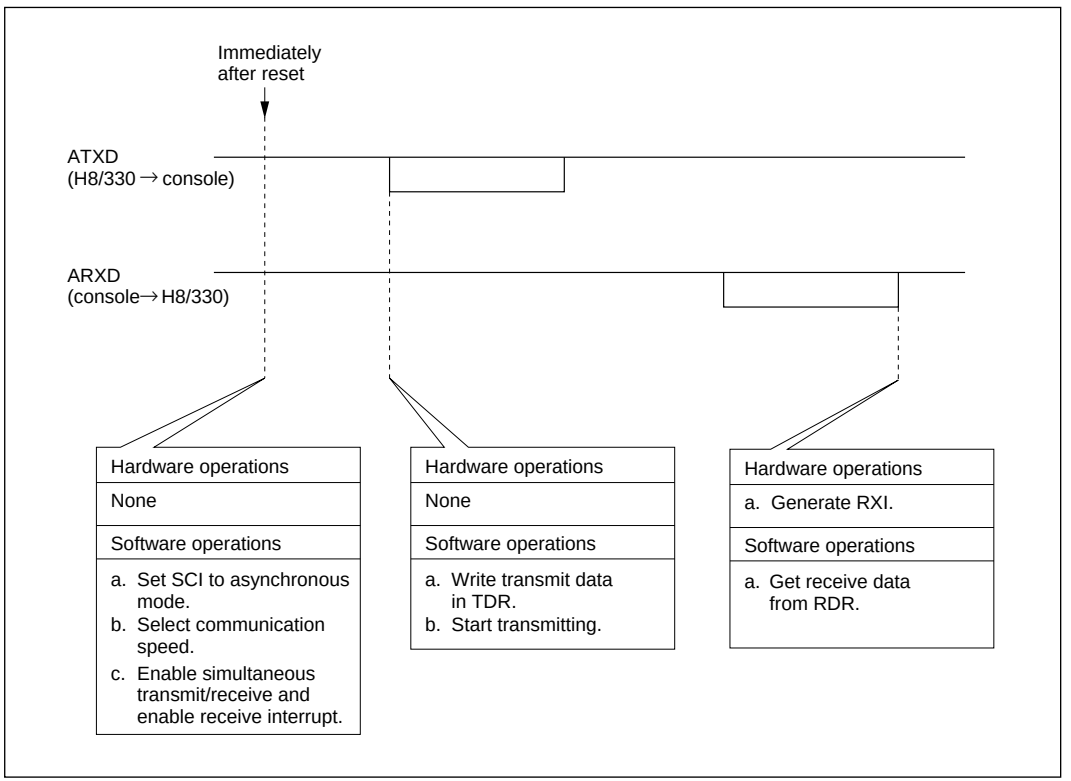


Figure 3 Asynchronous Serial Communication

Software Description

1. Module descriptions

Module	Label	Function
Main program	ASCIMN	Initializes SCI.
Transmit data	ASCITX	Transmits data placed in R0L.
Receive data	ASCIRX	RXI handler: moves data from RDR to output argument.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
R0L	Stores data to be sent to console.	1 byte	Main program	Output
			Transmit data	Input
ASC_ENDF	Flag indicating reception of data from console.	1 bit	Receive data	Output
	1: Data received 2: No data received		Main program	Input
ASC_RXDATA	Stores data received from console.	1 byte	Receive data	Output
			Main program	Input

3. On-chip register usage

Register	Function	Modules Where Used
SMR	Selects SCI mode (asynchronous), communication format, and clock source for baud rate generator (system clock input).	Main program
SCR	Enables receive interrupt and sets SCI to transmit/receive mode.	Main program
SSR	Clearing TDRE (bit 7) to 0 starts transmitter.	Transmit data
RDR	Stores data received from console.	Receive data
TDR	Stores data to be sent to console.	Transmit data
BRR	Selects communication speed.	Main program

Software Description

4. General register usage

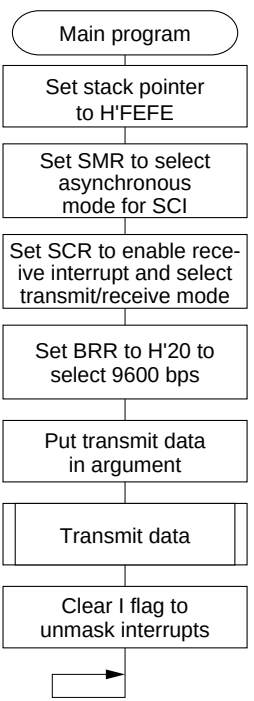
Module	Register	Function
Main program	R0L	Used as work register for setting data.
Transmit data	R0	Used as work register for setting data.
Receive data	R0L	Used as work register for setting data.

5. RAM usage

This sample task does not use RAM except for arguments.

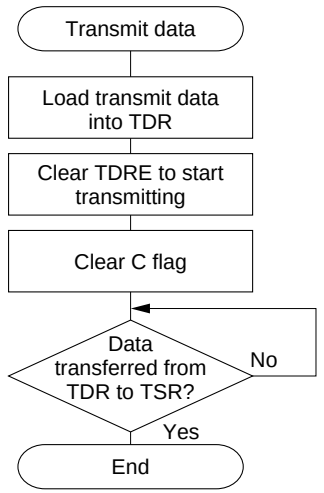
Flowcharts

1. Main program

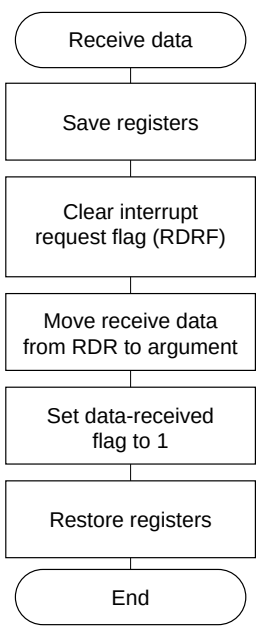


Flowcharts

2. Transmit data



3. Receive data



Program List

Program List

Specification

1. This sample task uses the H8/330's serial communication interface (SCI) in clocked synchronous mode to interface with a four-bit microcontroller (H4019) as shown in figure 1.

2. Transmitting and receiving are performed simultaneously. An I/O port is used to implement a simple protocol.

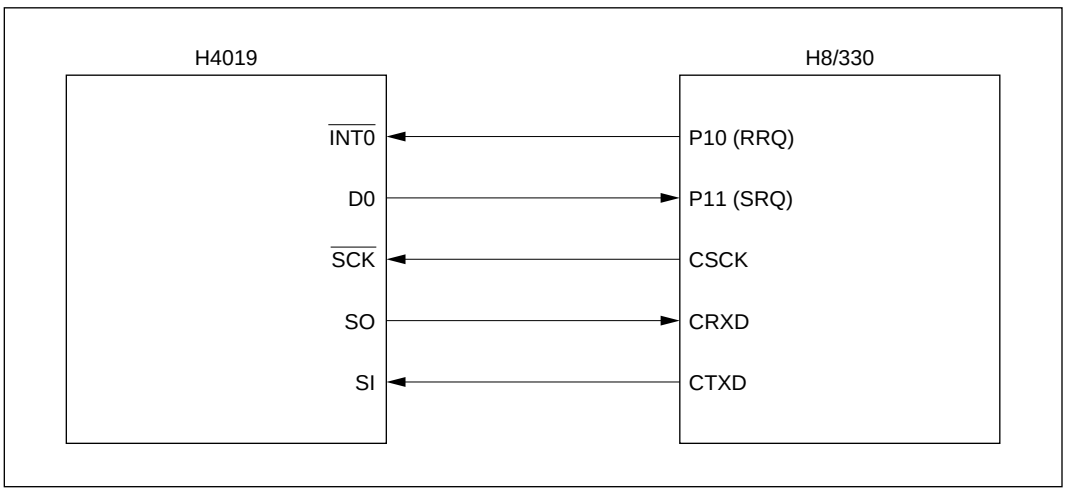


Figure 1 Clocked Synchronous Interface Using H8/330's SCI

Usage of Functions

1. Figure 2 shows a block diagram of the SCI used by this sample task. The SCI has the following features:

   a. Selection of clocked synchronous or asynchronous communication.

   b. Data can be transmitted and received simultaneously (full duplex communication).

   c. Receive-end interrupt.

   This task uses these features to interface with an H4019 as shown in figure 2.

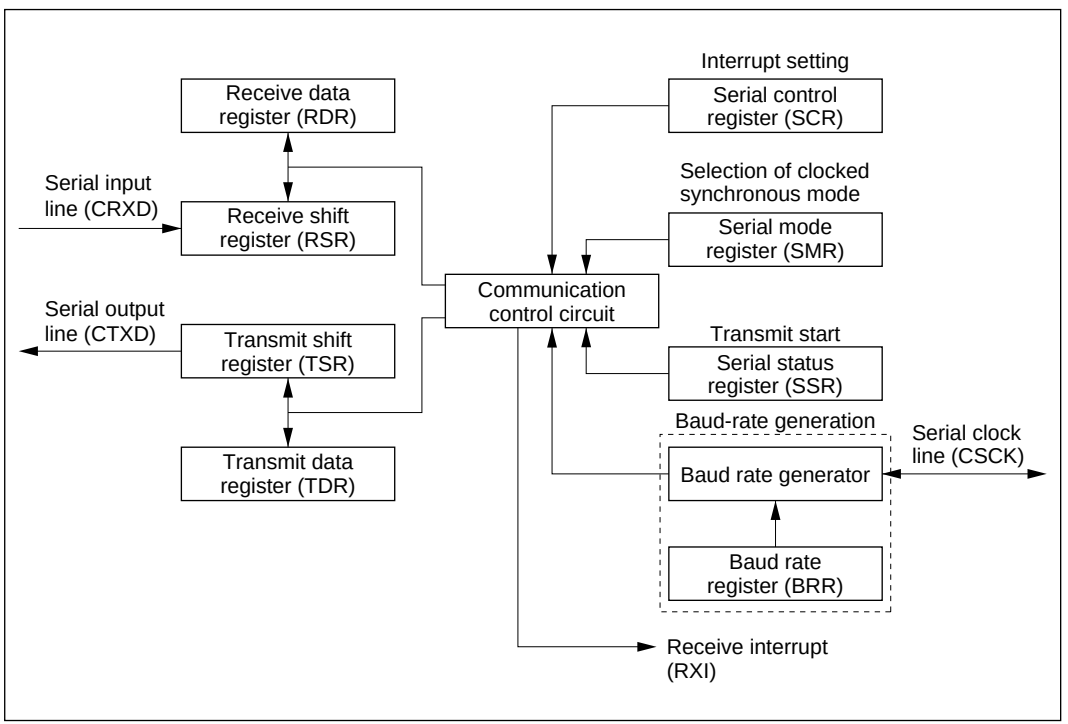


Figure 2 SCI Block Diagram

Usage of Functions

2. Table 1 lists the function assignments for this sample task. SCI functions are assigned for interfacing with an H4019.

Table 1 SCI Function Assignments

SCI Function	Description
CSCK	Transmits serial clock.
CRXD	Receives data from H4019.
CTXD	Sends data to H4019.
SMR	Selects clocked synchronous mode of SCI.
SCR	Enables receive interrupt and sets SCI to transmit/receive mode.
SSR	Starts transmitting.
RDR	Stores data received from H4019.
TDR	Stores data to be sent to H4019.
BRR	Selects communication speed.
P1DDR	Sets port-1 lines to input and output.
P1DR	Sends RRQ and receives SRQ.

Operation

Figure 3 explains the interfacing procedure. The I/O port and the clocked synchronous SCI are controlled to interface with an H4019 with the timing shown in figure 3.

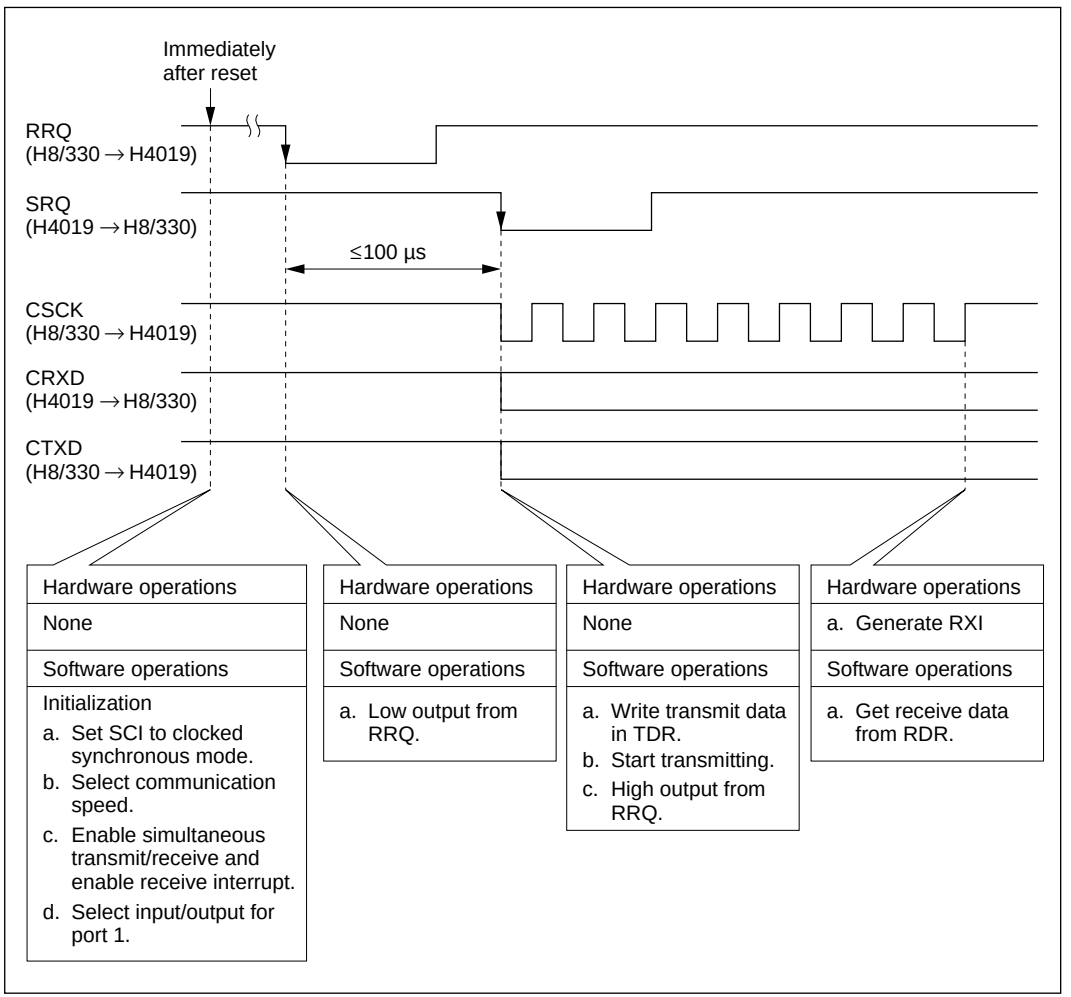


Figure 3 Clocked Synchronous Serial Communication

Software Description

1. Module descriptions

Module	Label	Function
Main program	CSCIMN	Initializes I/O port and SCI.
Transmit data	CSCITX	Transmits data in placed R0L.
Receive data	CSCIRX	RXI handler: moves data from RDR to output argument.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
R0L	Stores data to be sent to H4019.	1 byte	Main program	Output
			Transmit data	Input
C flag	Flag indicating whether transmitting ended successfully. 0: Transmit successful 1: Transmit unsuccessful	1 bit	Transmit data	Output
			Main program	Input
CSC_ENDF	Flag indicating reception of data from H4019. 1: Data received 0: No data received	1 bit	Receive data	Output
			Main program	Input
CSC_RXDATA	Stores data received from H4019.	1 byte	Receive data	Output
			Main program	Input

3. On-chip register usage

Register	Function	Modules Where Used
SMR	Selects SCI mode (clocked synchronous), communication format, and clock source for baud rate generator (system clock input).	Main program
SCR	Enables receive interrupt and sets SCI to transmit/receive mode.	Main program
SSR	Clearing TDRE (bit 7) to 0 starts transmitter.	Transmit data
RDR	Stores data received from H4019.	Receive data
TDR	Stores data to be sent to H4019.	Transmit data
BRR	Selects communication speed.	Main program
P1DDR	Sets port-1 lines to input and output.	Main program
P1DR	Transmits RRQ and receives SRQ.	Main program, transmit data

4. General register usage

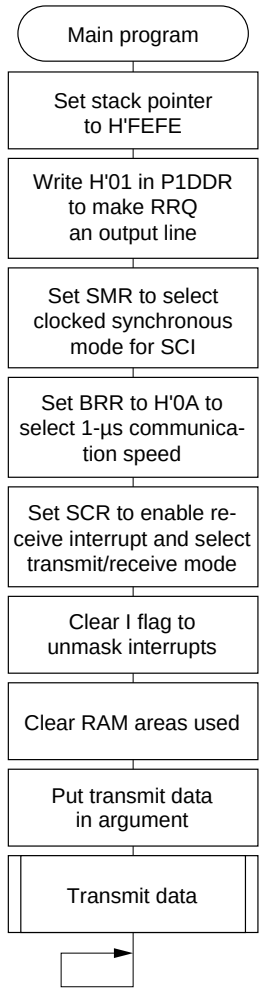
Module	Register	Function
Main program	R0L	Used as work register for setting data.
Transmit data	R0	Used as work register for setting data.
Receive data	R0L	Used as work register for setting data.

5. RAM usage

This sample task does not use RAM except for arguments.

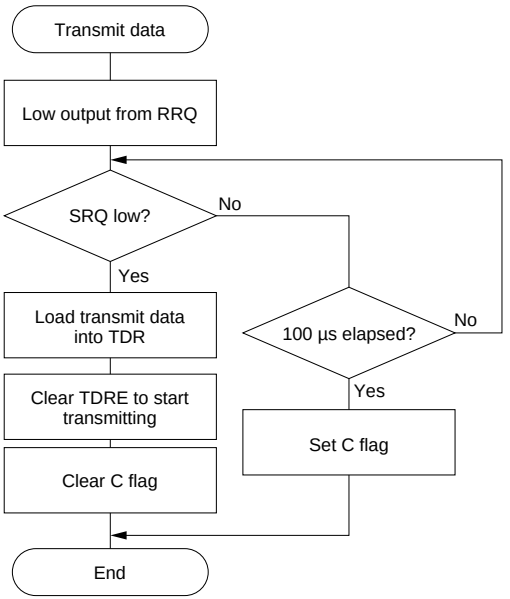
Flowcharts

1. Main program

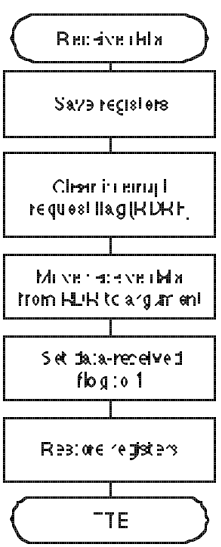


Flowcharts

2. Transmit data



3. Receive data



Program List

Program List

Program List

Specification

1. This sample task inputs a voltage on one channel to the H8/330 as shown in figure 1, measures the voltage by A/D conversion, and stores the measurement result in a one-byte RAM area.

2. The input voltage range is 0 to 5 V.

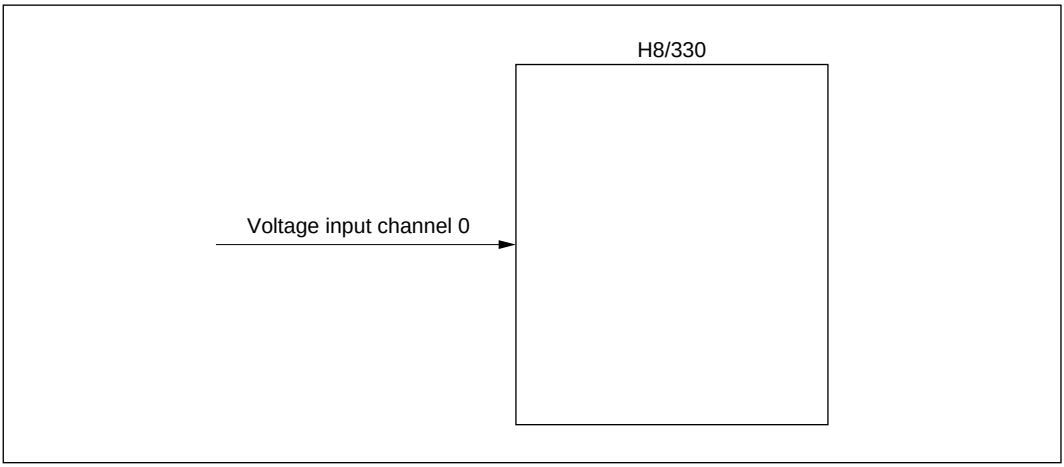


Figure 1 Voltage Measurement by H8/330

Usage of Functions

1. Figure 2 shows a block diagram of the A/D converter used by this sample task. The H8/330 has a successive-approximations A/D converter on-chip. The A/D converter has the following functions:

   a. A sample-and-hold function holds the measured voltage from the start of the measurement until the end.

   b. An interrupt function generates an interrupt at the end of A/D conversion.

   This sample task uses these functions to carry out A/D conversion.

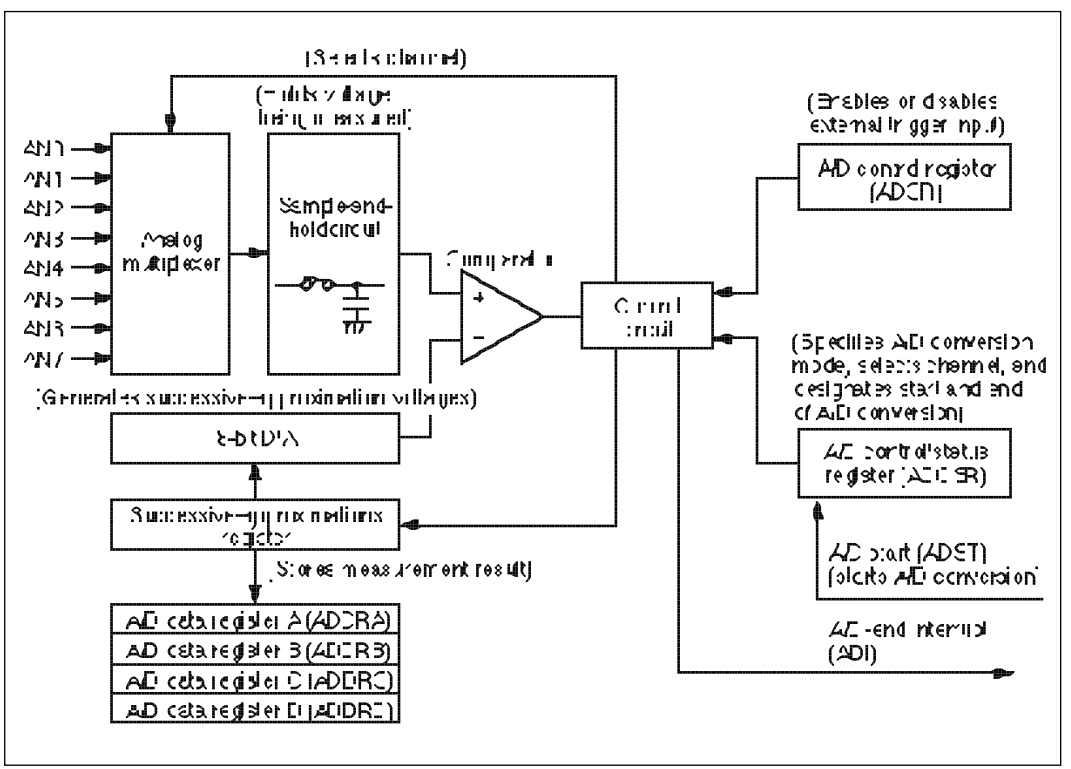


Figure 2 A/D Converter Block Diagram

Usage of Functions

2. Table 1 lists the function assignments for this sample task. Functions of the H8/330's on-chip A/D converter are assigned to perform A/D conversion.

Table 1 A/D Converter Function Assignments

A/D Converter Function	Description
ADCSR	Selects A/D conversion mode (single or scan), channel(s) to be measured, and conversion time, and designates start and end of measurement.
ADDRA	Stores A/D conversion result.

Operation

Figure 3 explains how A/D conversion is carried out. The H8/330 uses both hardware and software operations to perform A/D conversion on one channel.

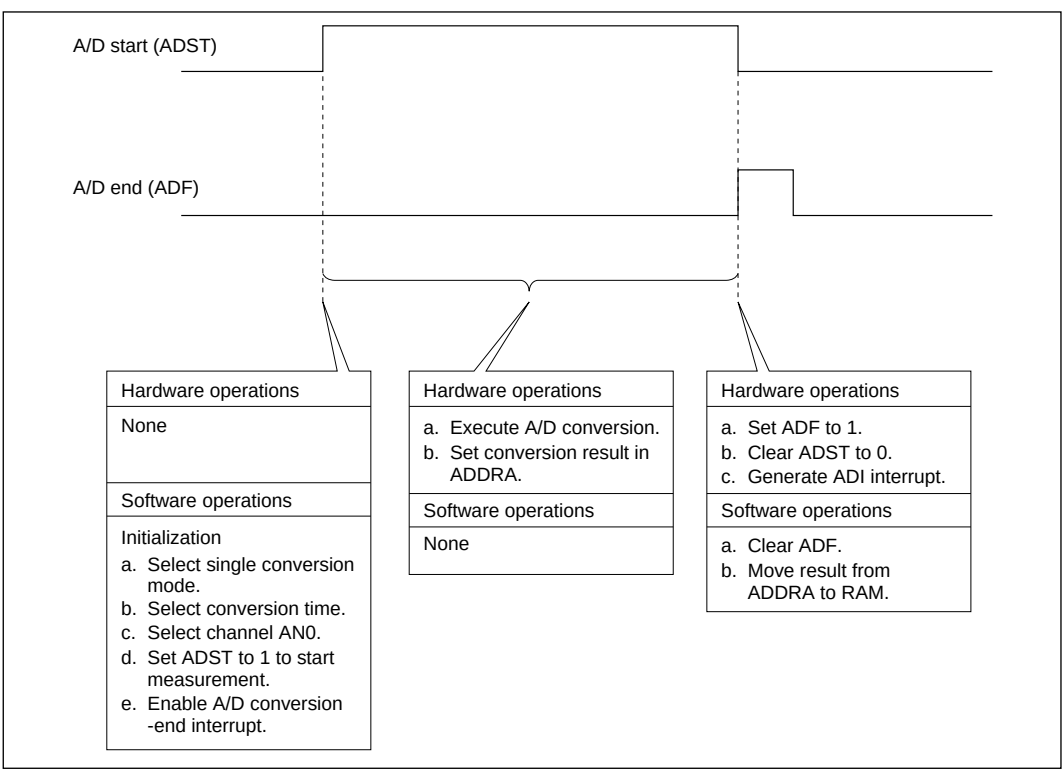


Figure 3 A/D Conversion

Software Description

1. Module descriptions

Module	Label	Function
Main program	ADSIMN	Initializes A/D converter, clears flag, and starts A/D conversion.
Detect A/D end	SINEND	ADI handler: Flags end of A/D conversion and stores result in RAM.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
SIN_DATA	Stores result of single-channel A/D conversion.	1 byte	Detect A/D end	Output
			Main program	Input
SIN_ENDF	Flag indicating end of single-channel A/D conversion.	1 bit	Detect A/D end	Output
	1: A/D conversion completed 0: A/D conversion in progress		Main program	Input

3. On-chip register usage

Register	Function	Modules Where Used
ADCSR	Selects A/D conversion mode (single or scan), channel(s) to be measured, and conversion time, and designates start and end of measurement.	Main program, detect A/D end
ADDRA	Stores result of A/D conversion.	Detect A/D end

4. General register usage

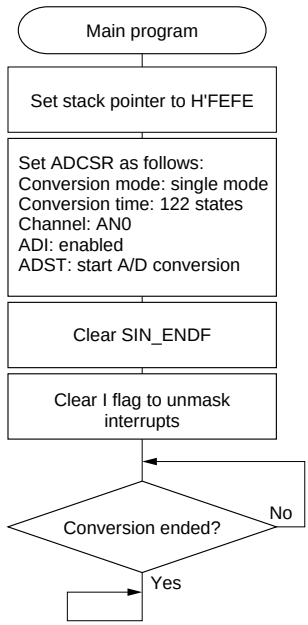
Module	Register	Function
Main program	R0L	Used as work register for setting data.
Detect A/D end	R0L	Used as work register for setting data.

5. RAM usage

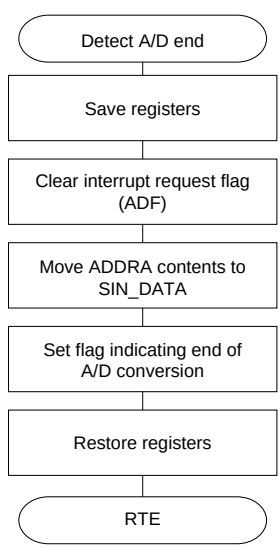
This sample task does not use RAM except for arguments.

Flowcharts

1. Main program



2. Detect A/D end



Program List

Program List

Specification

1. This sample task inputs voltages on eight channel to the H8/330 as shown in figure 1, measures the voltages by A/D conversion, and stores the measurement results in an eight-byte RAM area.

2. The input voltage range is 0 to 5 V.

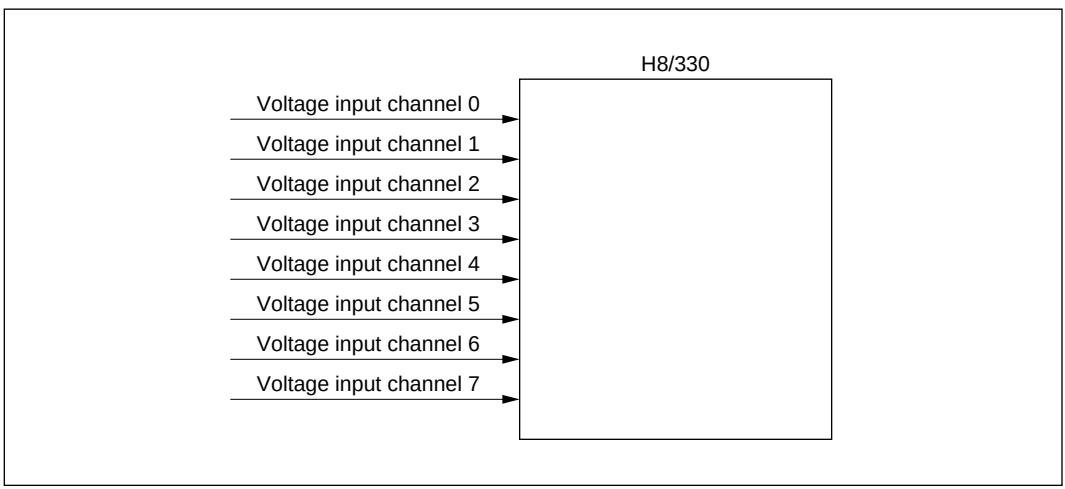


Figure 1 Voltage Measurement by H8/330

Usage of Functions

1. Figure 2 shows a block diagram of the A/D converter used by this sample task. The H8/330 has a successive-approximations A/D converter on-chip. The A/D converter has the following functions:

   a. A scan mode automatically performs A/D conversion on one to four channels (AN0 to AN3 or AN4 to AN7) without software intervention.

   b. A sample-and-hold function holds the measured voltage from the start of the measurement until the end.

   c. An interrupt function generates an interrupt at the end of A/D conversion.

   This sample task uses these functions to carry out A/D conversion.

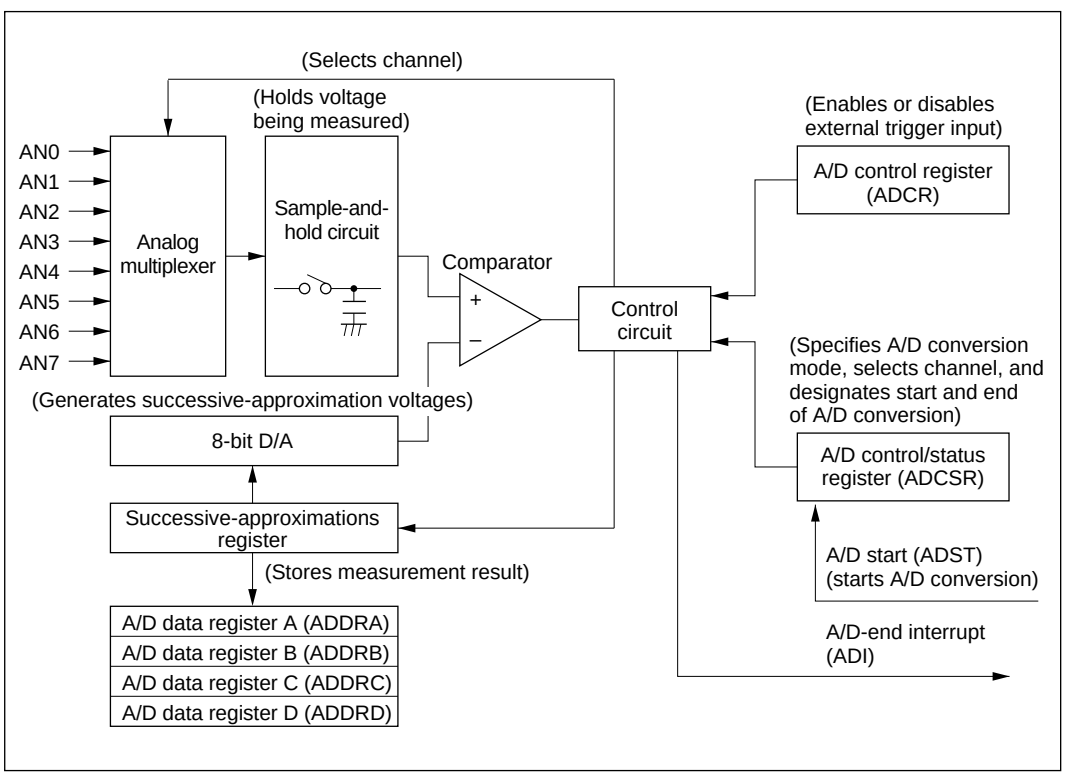


Figure 2 A/D Converter Block Diagram

Usage of Functions

2. Table 1 lists the function assignments for this sample task. Functions of the H8/330's on-chip A/D converter are assigned to perform A/D conversion.

Table 1 A/D Converter Function Assignments

A/D Converter Function	Description
ADCSR	Selects A/D conversion mode (single or scan), channel(s) to be measured, and conversion time, and designates start and end of measurement.
ADDRA	Stores A/D conversion result.
ADDRB	
ADDRC	
ADDRD	

Operation

Figure 3 explains how A/D conversion is carried out. Channels AN0 to AN7 are divided into two groups of four channels each (AN0 to AN3 and AN4 to AN7). Each group is converted in scan mode to generate results that are stored in ADDRA to ADDRD.

The sample task moves the conversion results into RAM.

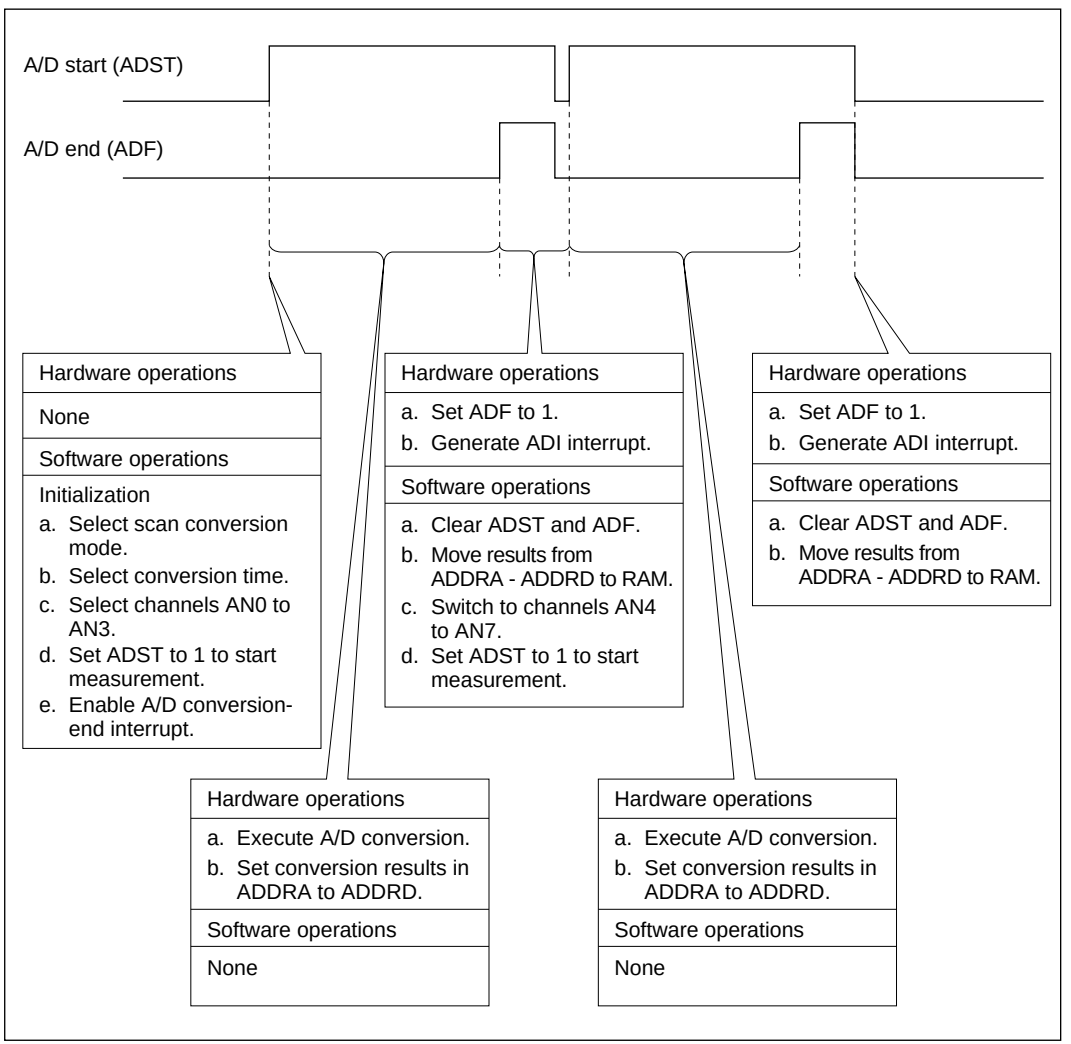


Figure 3 A/D Conversion

Software Description

1. Module descriptions

Module	Label	Function
Main program	ADSCNMN	Initializes A/D converter, clears flag, and starts A/D conversion.
Detect A/D end	SCNEND	ADI handler: Flags end of A/D conversion, switches channels, and stores results in RAM.

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
SCN_DAT0	Stores result of eight-channel A/D conversion.	1 byte	Detect A/D end	Output
SCN_DAT1		1 byte		
SCN_DAT2		1 byte		
SCN_DAT3		1 byte		
SCN_DAT4		1 byte	Main program	Input
SCN_DAT5		1 byte		
SCN_DAT6		1 byte		
SCN_DAT7		1 byte		
SCN_ENDF	Flag indicating final end of eight-channel A/D conversion. 1: A/D conversion completed 0: A/D conversion in progress	1 bit	Detect A/D end	Output
			Main program	Input

3. On-chip register usage

Register	Function	Modules Where Used
ADCSR	Selects A/D conversion mode (single or scan), channel(s) to be measured, and conversion time, and designates start and end of measurement.	Main program, detect A/D end
ADDRA to ADDRD	Stores result of A/D conversion.	Detect A/D end

Software Description

4. General register usage

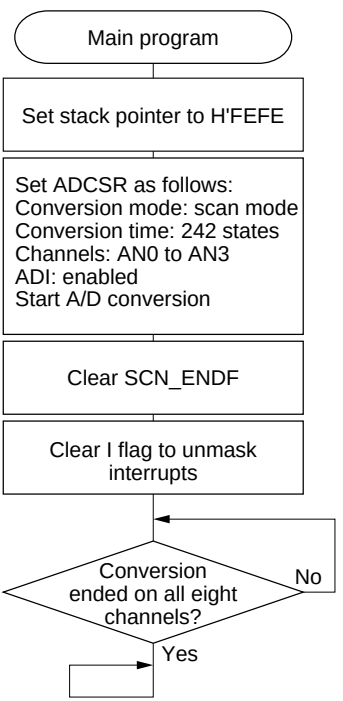
Module	Register	Function
Main program	R0	Used as work register for setting data.
Detect A/D end	R0 to R2	Used as work register for setting data.

5. RAM usage

This sample task does not use RAM except for arguments.

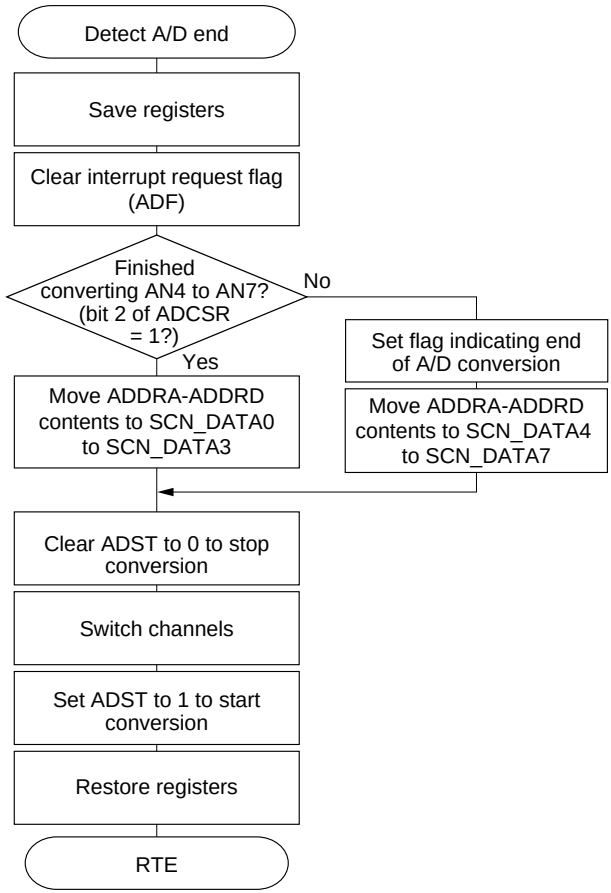
Flowcharts

1. Main program



Flowcharts

2. Detect A/D end



Program List

Program List

Specification

1. This sample task runs the H8/330 in power-down mode (software standby mode) using the test board shown in figure 1.

2. Software standby mode is entered and exited by detecting the state of a mode switch connected to the $\overline{\text{NMI}}$ line.

   Switch on: software standby mode Switch off: program execution

3. A LED indicator displays the H8/330's current operating mode.

   LED on: program execution LED off: software standby

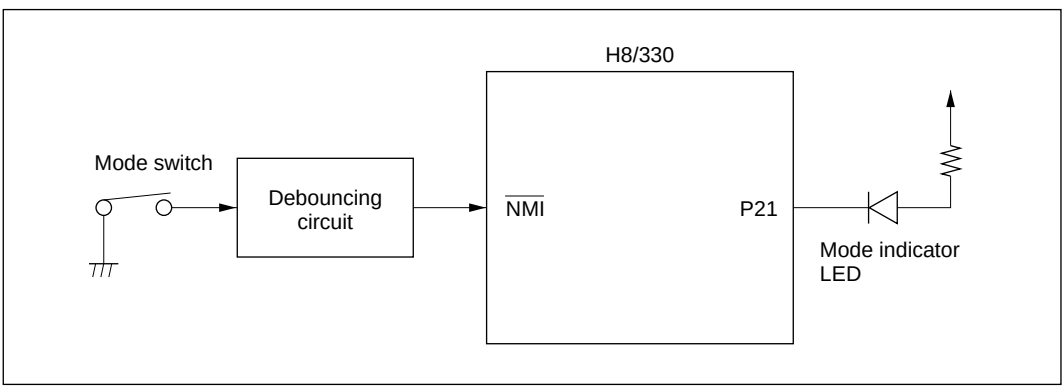


Figure 1 Power-Down Mode Test Board

Usage of Functions

1. Table 1 describes the software standby mode used by this sample task. Software standby mode has the following features:

   a. The clock and on-chip supporting modules halt, so power consumption is lower than in sleep mode.

   b. The waiting time for clock oscillation to stabilize on exit from software standby is selectable.

   Software standby mode is entered by setting the software standby bit (SSBY) in the system control register (SYSCR) to 1, then executing the SLEEP instruction. Software standby mode is exited by an external interrupt.

Table 1 Software Standby Mode

Entry Method	Clock	CPU	On-Chip Supporting Modules	General Registers	RAM	I/O	Exit Methods
Set SSBY to 1, then execute SLEEP instruction	Halt	Halt	Halt	Held	Held	Held	• External interrupt • Reset • STBY input

Operation

Figure 2 explains how software standby mode is entered and exited and shows the timing relationships.

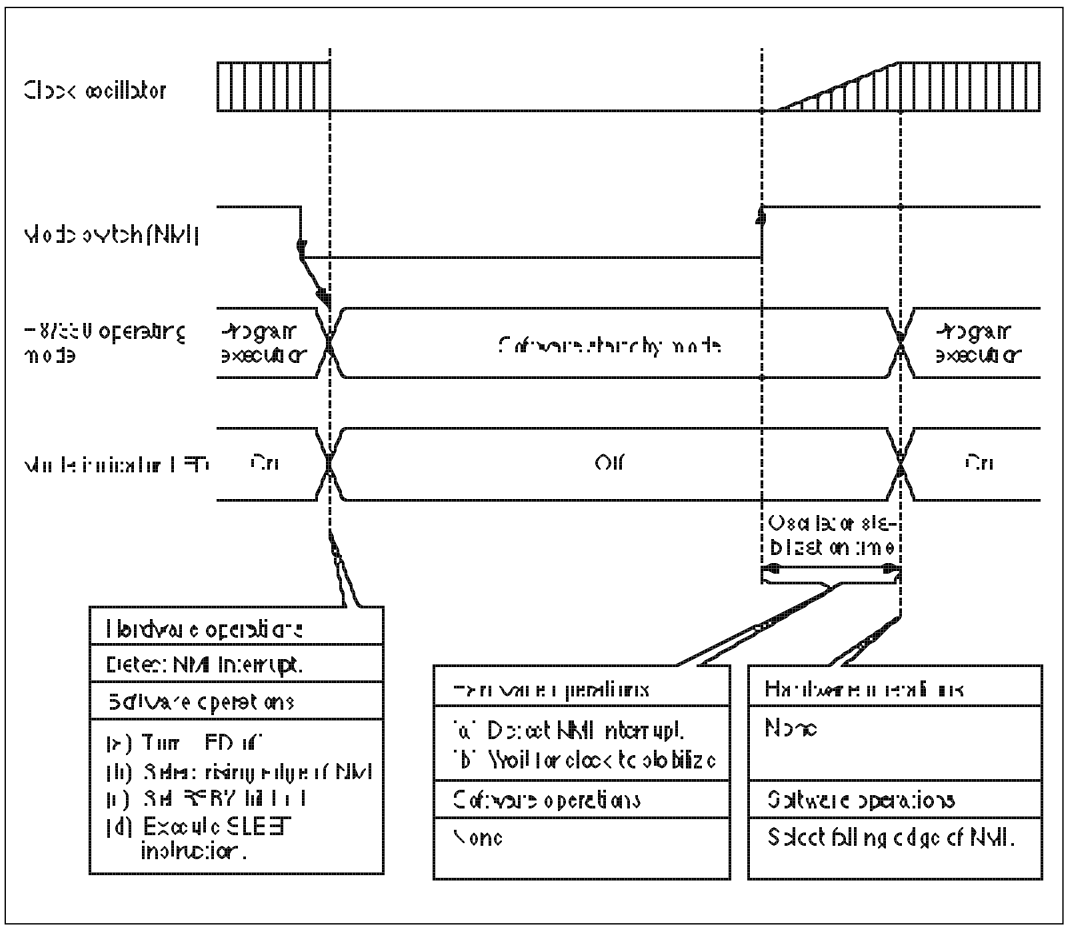


Figure 2 Software Standby Mode

Software Description

1. Module descriptions

Module	Label	Function
Main program	STBMN	Initializes I/O port and NMI interrupt settings.
Return and escape active	STBONOFF	NMI handler (rising and falling edges): enters and exits software standby

2. Argument descriptions

   This sample task does not use arguments.

3. On-chip register usage

Register	Function	Modules Where Used
SYSCR	Selects oscillator stabilization time, NMI edge, and software standby mode transition.	Main program, return and escape active
P2DDR	Controls input/output direction of port 2.	Main program
P2DR	Controls LED and detects switch input state.	Main program

4. General register usage

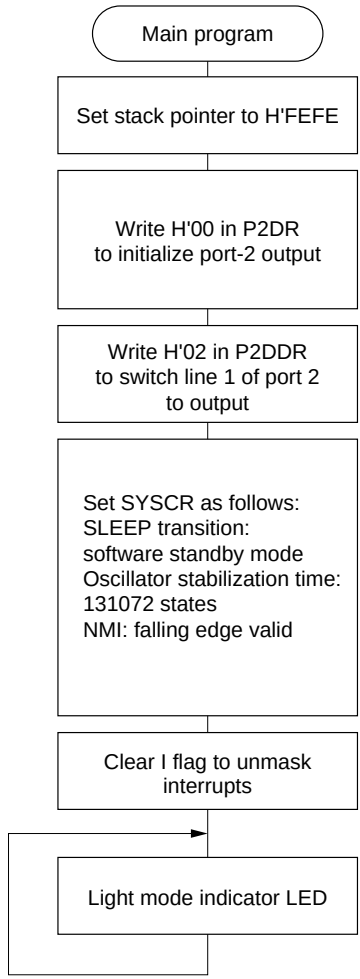
Module	Register	Function
Main program	R0	Used as work register for setting data.

5. RAM usage

   This sample task does not use RAM.

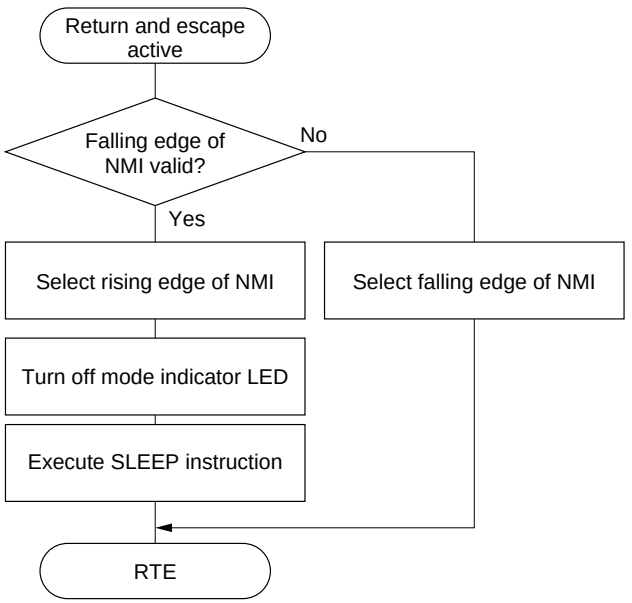
Flowcharts

1. Main program



Flowcharts

2. Return and escape active



Program List

Program List

Specification

1. This sample task runs the H8/330 in power-down mode (sleep mode) using the test board shown in figure 1.

2. Sleep mode is entered by detecting the state of a mode switch.

   Switch on: sleep mode Switch off: program execution

3. A LED indicator displays the H8/330's current operating mode.

   LED on: program execution LED off: sleep mode

4. Sleep mode is exited by a 0.4-second timer interrupt.

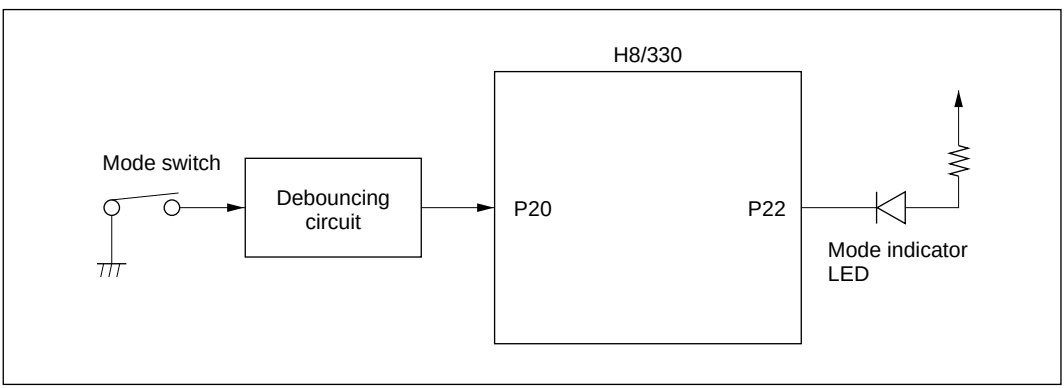


Figure 1 Power-Down Mode Test Board

Usage of Functions

1. Table 1 describes the sleep mode used by this sample task. Sleep mode is entered by executing the SLEEP instruction, and can be exited by an interrupt (external interrupt, timer interrupt, or other interrupt).

Table 1 Sleep Mode

Entry Method	Clock	CPU	On-Chip Supporting Modules	General Registers	RAM	I/O	Exit Methods
Execute SLEEP instruction	Run	Halt	Run	Held	Held	Held	• Interrupt • Reset • STBY input

Operation

Figure 2 explains how sleep mode is entered and exited and shows the timing relationships.

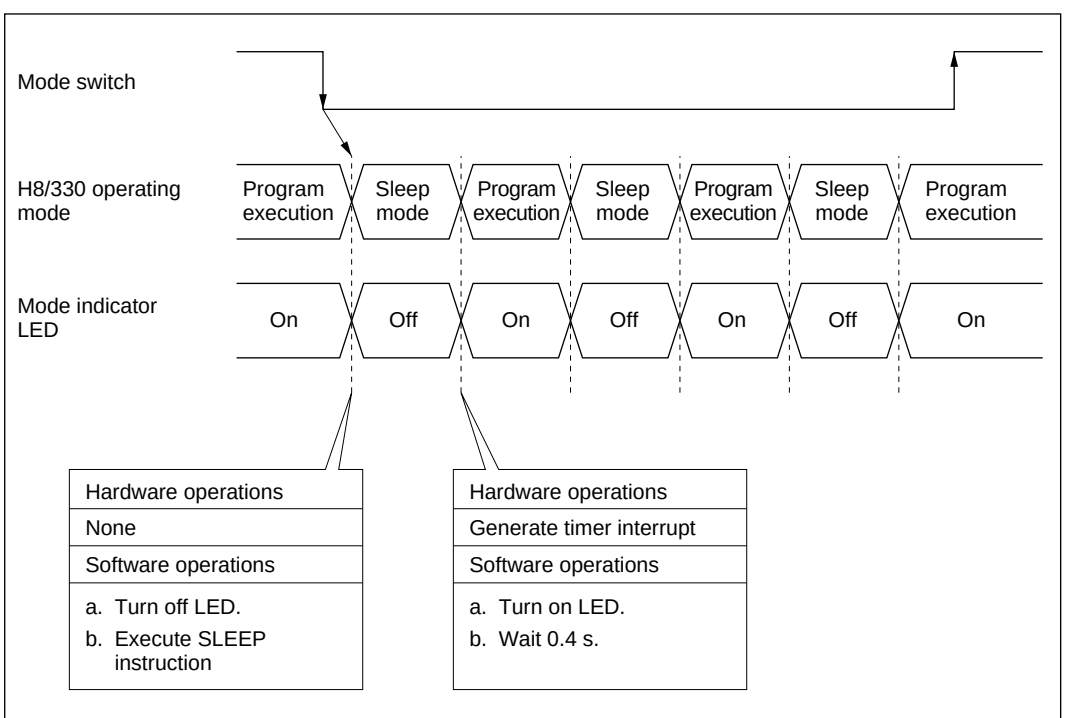


Figure 2 Sleep Mode

Software Description

1. Module descriptions

Module	Label	Function
Main program	SLPMN	Initializes I/O port and 16-bit timer, and executes SLEEP instruction according to switch state.
Return active mode	ACTON	16-bit timer overflow interrupt handler: executed at 0.4-s intervals

2. Argument descriptions

Label or Register	Function	Data Length	Modules Where Used	Input/ Output
ACT_TONF	Flag indicating whether return-active-mode routine has executed. 1: Executed 0: Not executed	1 bit	Return active mode	Output
			Main program	Input

3. On-chip register usage

Register	Function	Modules Where Used
TCR	Selects clock source for FRC.	Main program
TIER	Enables FOVI.	Main program
P2DDR	Controls input/output direction of port 2.	Main program
P2DR	Controls LED and detects switch input state.	Main program

4. General register usage

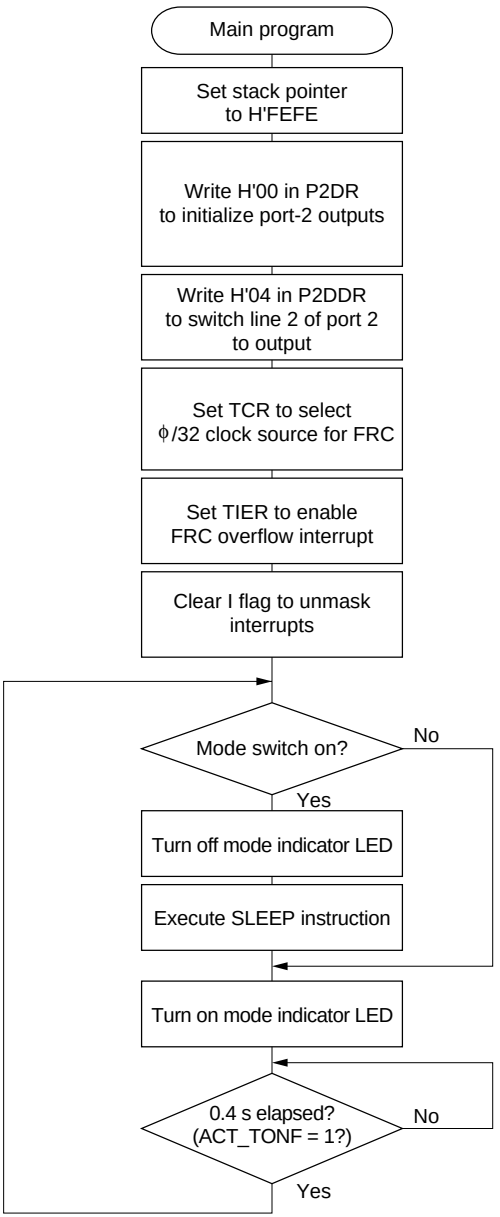
Module	Register	Function
Main program	R0	Used as work register for setting data.

5. RAM usage

This sample task does not use RAM except for arguments.

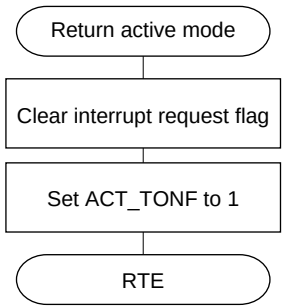
Flowcharts

1. Main program



Flowcharts

2. Return active mode



Program List

Program List

Interfaces

Specification

1. The H8/330's expanded mode (mode 2) is used to interface the H8/330 to EPROM (HN27C256) and SRAM (HM62256-8) as shown in figure 1.

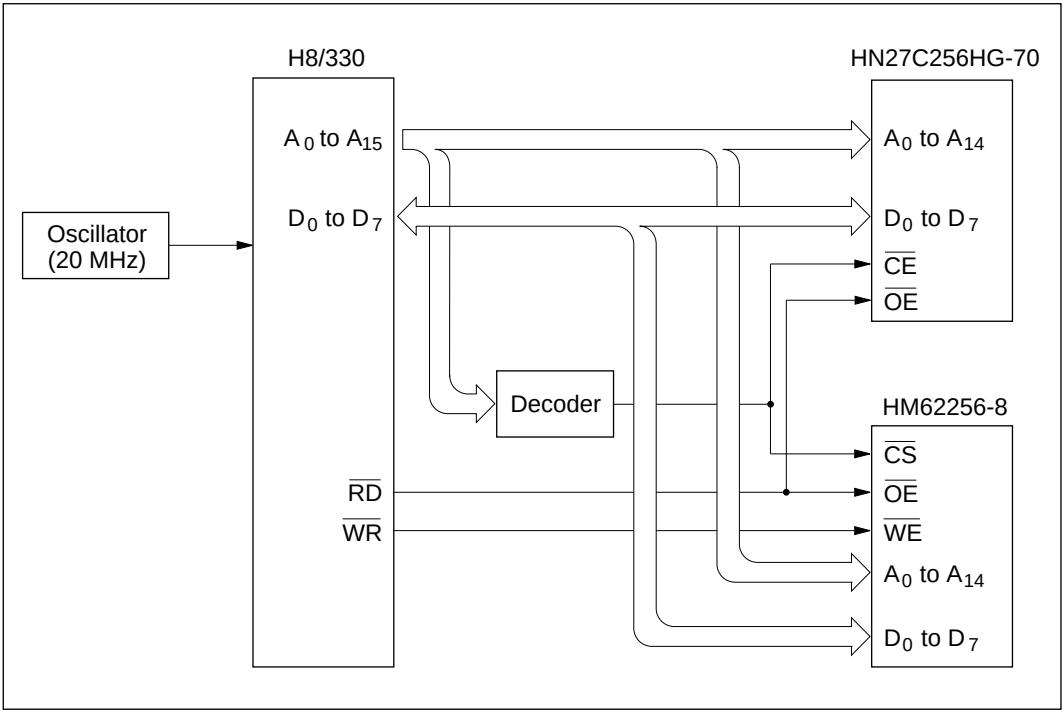


Figure 1 H8/330-Memory Interface Block Diagram

Specification

2. The HN27C256 and HM62256 are allocated as shown in figure 2.

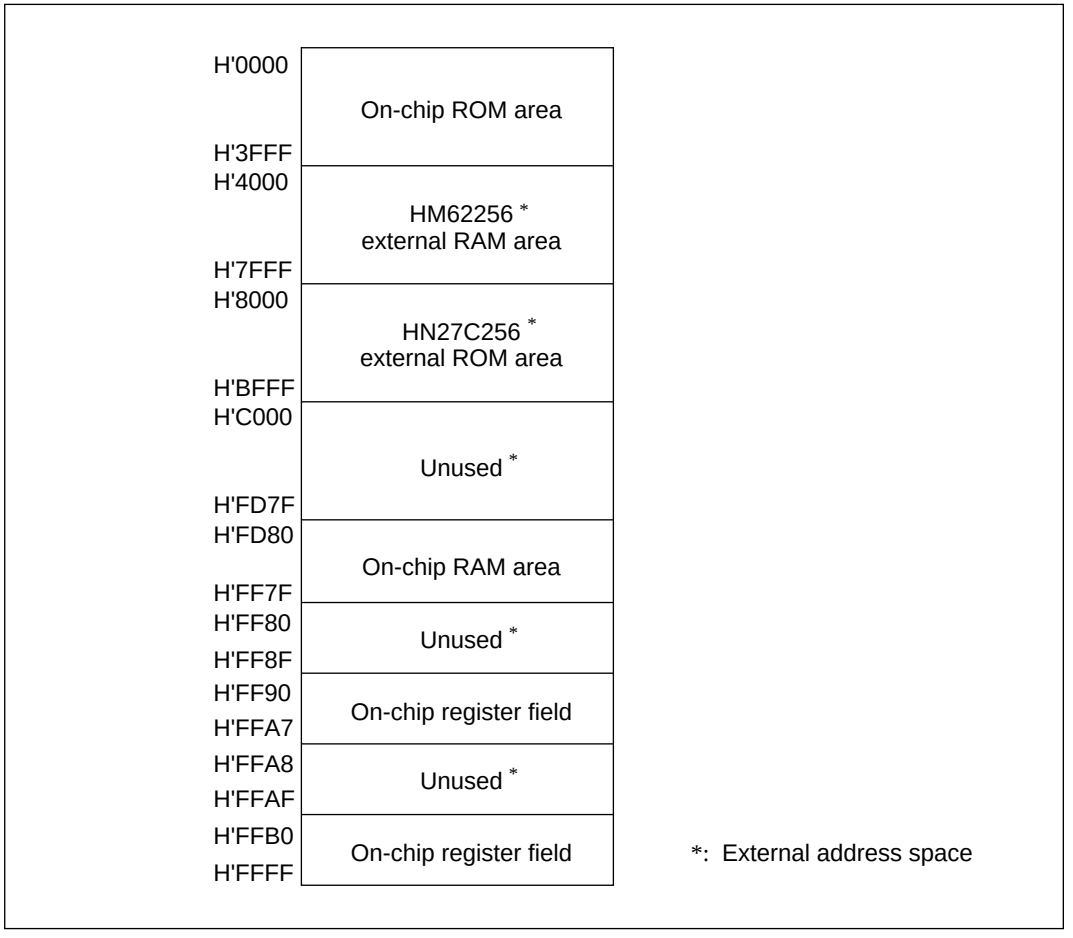


Figure 2 Memory Map

Operation

1. Timing

   a. H8/330 bus cycle (without wait states)

   Figure 3 shows the basic bus cycle of the H8/330 when no wait states are inserted. In the expanded modes external devices are interfaced using three states, as indicated in figure 3.

   In a read cycle, the H8/330 latches data (D_7 to D_0) on the falling edge of T_3. In a write cycle, the H8/330 outputs data with a delay (T_{WDD}) of 60 to 70 ns from the falling edge of T_1.

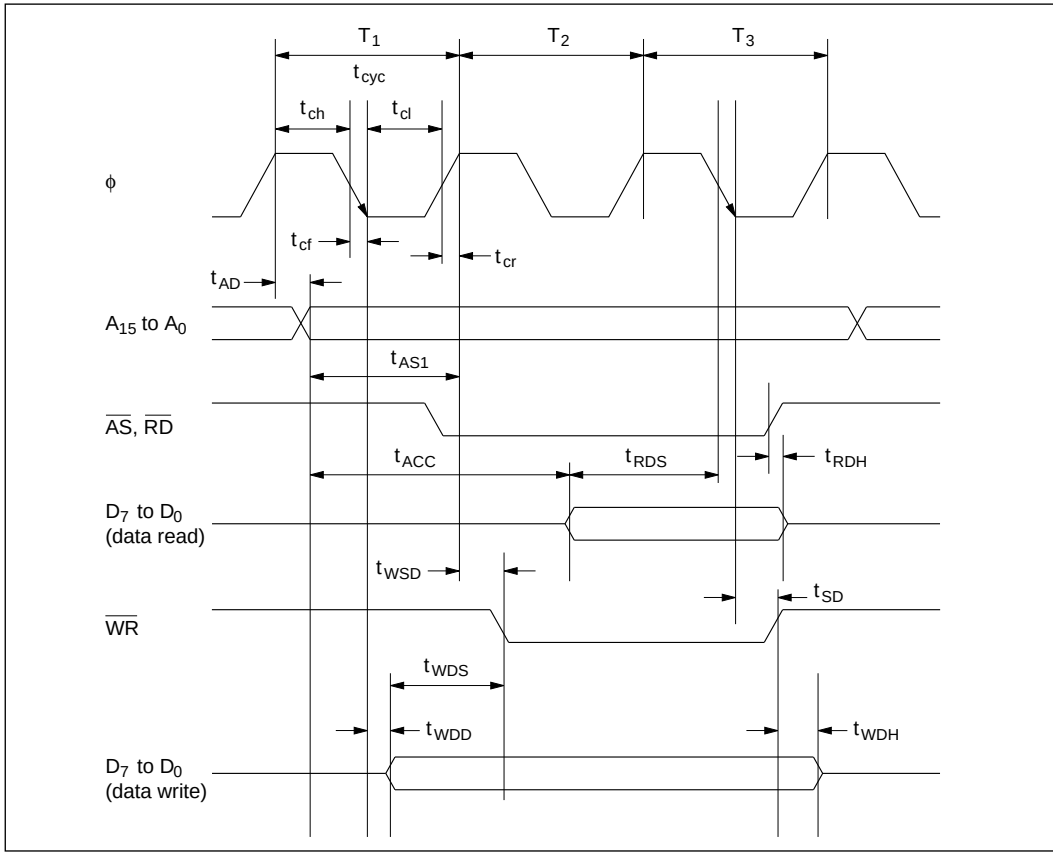


Figure 3 Basic H8/330 Bus Cycle

Operation

b. Memory read timing

Figure 4 shows the timing of a memory read cycle. The timing requirement to be satisfied in a memory read cycle is t_{RDS} (read data setup time: minimum 50 ns), which can be calculated as follows:

HN27C256HG-70: t_{RDS} = 250 ns – (t_{CH} + t_{ASD} + t_{OE} [OE output delay])
= 250 ns – (65 ns + 50 ns + 40 ns)
= 95 ns

HM62256-8: t_{RDS} = 250 ns – (t_{CH} + t_{ASD} + t_{OE} [OE output delay])
= 250 ns – (50 ns + 50 ns + 45 ns)
= 105 ns

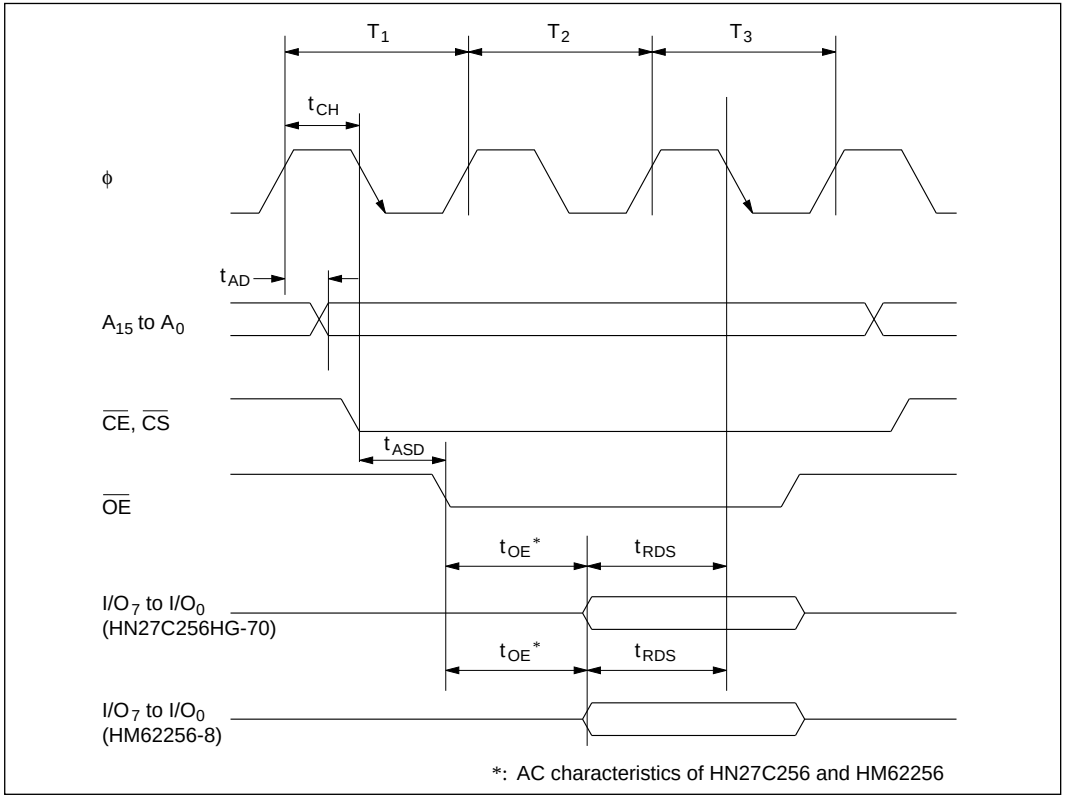


Figure 4 Memory Read Timing Diagram

Operation

c. Memory write timing

Figure 5 shows the timing of a memory write cycle. The timing requirement to be satisfied in this write cycle is the HM62256-8's t_{DW} (input data setup time: minimum 40 ns). Since the H8/330 outputs write data with a delay of t_{WDD} (maximum 60 ns) from the falling edge of T_1, the t_{DW} requirement is easily met, as shown in figure 5.

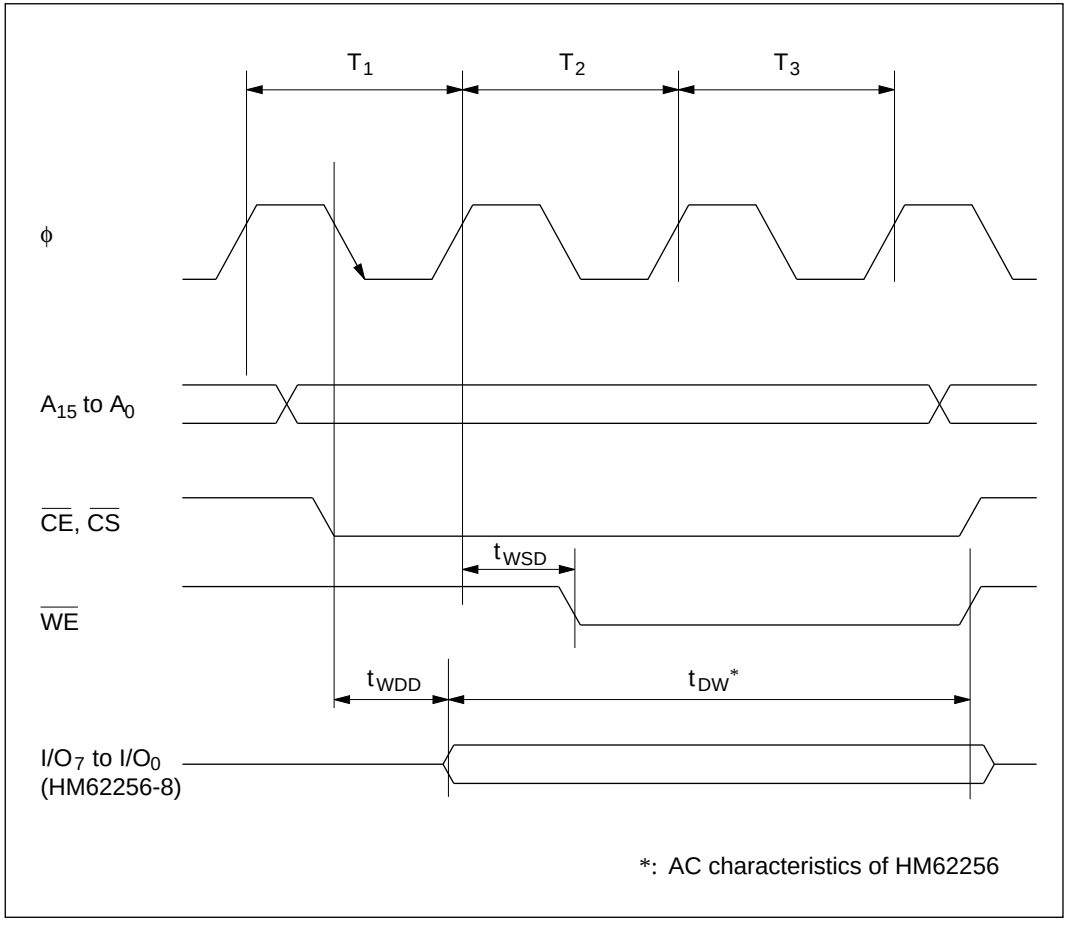
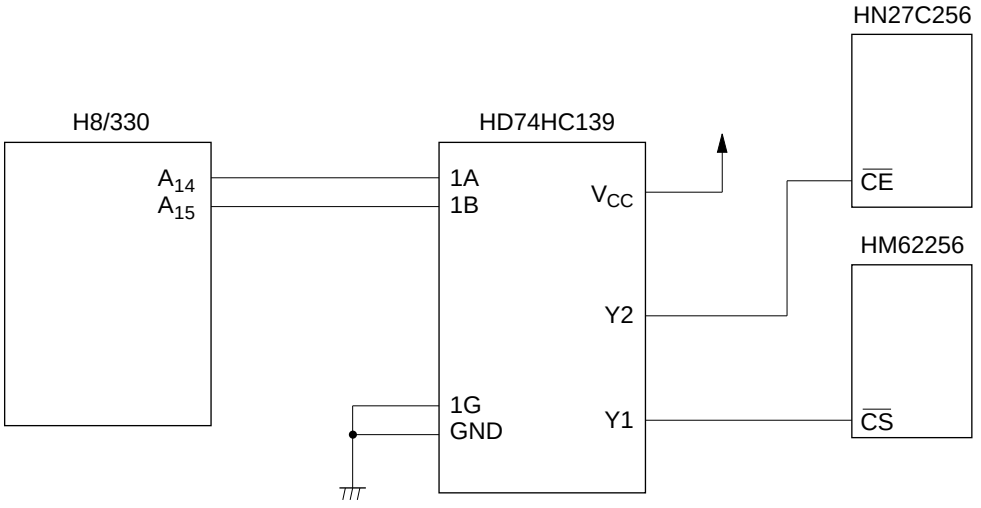


Figure 5 Memory Write Timing Diagram

Operation

1. Address decoding

An HD74HC139 is used for address decoding as indicated in figure 6.



Address	A_{14}	A_{15}	RAM	ROM
H'4000 to H'7FFF	1	0	Enable	Disable
H'8000 to H'BFFF	0	1	Disable	Enable

Figure 6 Address Decoder

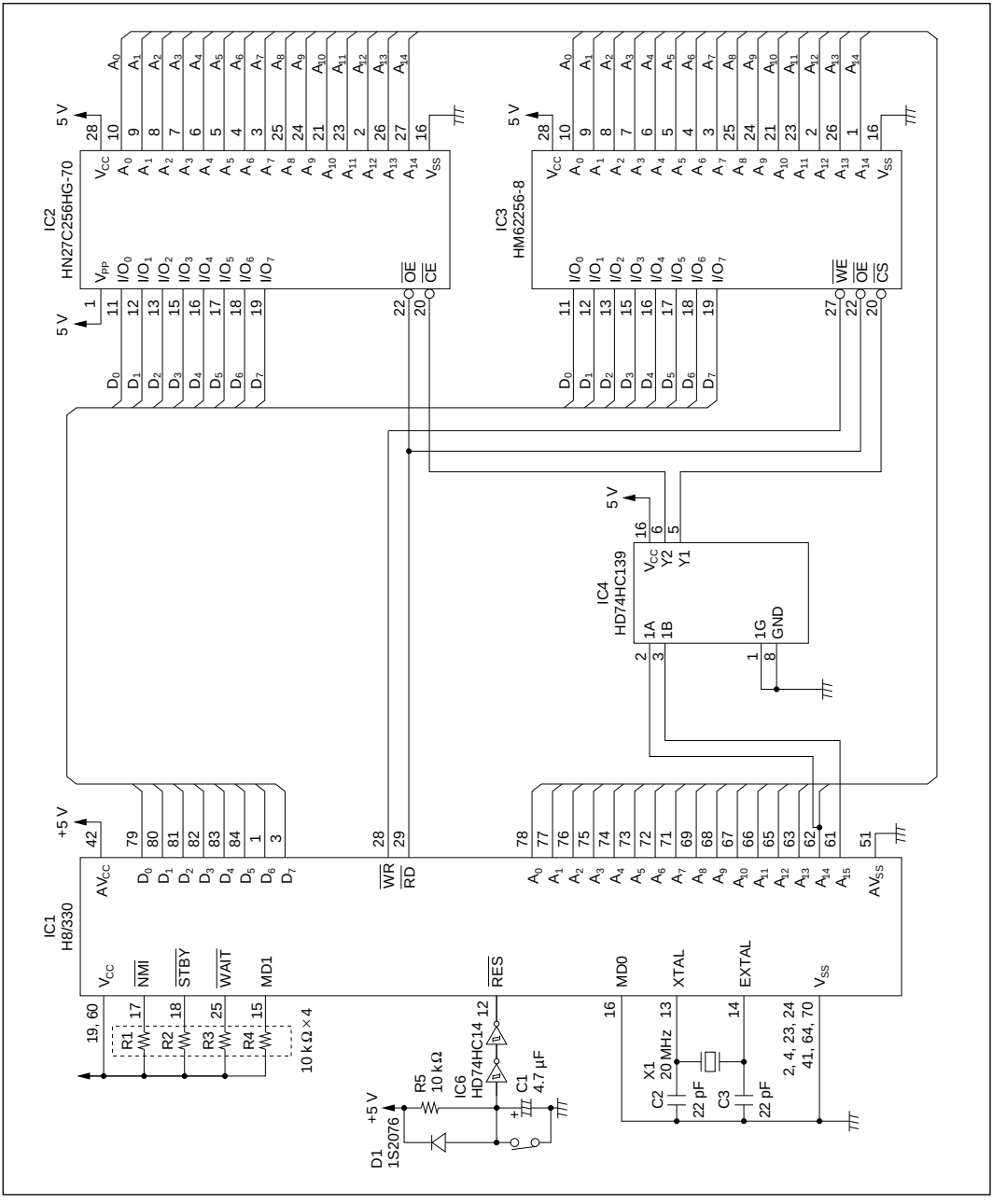


Figure 7 Memory Interface Circuit Diagram

AC Characteristics

1. H8/330

Item	Symbol	6 MHz Min	6 MHz Max	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Unit
Clock cycle time	t_{cyc}	166.7	2000	125	2000	100	2000	ns
Clock low pulse width	t_{CL}	65	—	45	—	35	—	ns
Clock high pulse width	t_{CH}	65	—	45	—	35	—	ns
Clock rise time	t_{cr}	—	15	—	15	—	15	ns
Clock fall time	t_{cf}	—	15	—	15	—	15	ns
Address delay time	t_{AD}	—	70	—	60	—	55	ns
Address hold time	t_{AH}	30	—	25	—	20	—	ns
Address strobe delay time	t_{ASD}	—	70	—	60	—	50	ns
Write strobe delay time	t_{WSD}	—	70	—	60	—	50	ns
Strobe delay time	t_{SD}	—	70	—	60	—	50	ns
Write strobe pulse width	t_{WSW}	200	—	150	—	120	—	ns
Address setup time 1	t_{AS1}	25	—	20	—	15	—	ns
Address setup time 2	t_{AS2}	105	—	80	—	65	—	ns
Read data setup time	t_{RDS}	70	—	55	—	50	—	ns
Read data hold time	t_{RDH}	0	—	0	—	0	—	ns
Write data delay time	t_{WDD}	—	70	—	60	—	60	ns
Read data access time	t_{ACC}	—	270	—	190	—	160	ns
Write data setup time	t_{WDS}	30	—	15	—	10	—	ns
Write data hold time	t_{WDH}	30	—	25	—	20	—	ns
Wait setup time	t_{WTS}	40	—	40	—	40	—	ns
Wait hold time	t_{WTH}	10	—	10	—	10	—	ns
E clock delay time	t_{ED}	—	20	—	20	—	20	ns
E clock rise time	t_{Er}	—	15	—	15	—	15	ns
E clock fall time	t_{Ef}	—	15	—	15	—	15	ns
Read data hold time (synchronized with E clock)	t_{RDHE}	0	—	0	—	0	—	ns
Write data hold time (synchronized with E clock)	t_{WDHE}	50	—	40	—	30	—	ns

AC Characteristics

2. HN27C256HG-70

		HN27C256HG-70			
Item	**Symbol**	**Min**	**Max**	**Unit**	**Test Conditions**
Access time	t_{ACC}	—	70	ns	$\overline{CE} = \overline{OE} = V_{IL}$
$\overline{CE}$-to-output delay time	t_{CE}	—	70	ns	$\overline{OE} = V_{IL}$
$\overline{OE}$-to-output delay time	t_{OE}	—	40	ns	$\overline{CE} = V_{IL}$
Output disable delay time	t_{DF}*	0	30	ns	$\overline{CE} = V_{IL}$
Data output hold time	t_{OH}	5	—	ns	$\overline{CE} = \overline{OE} = V_{IL}$

*: Defined at the point at which the output reaches the open state and the output leve cannot be read.

3. HM62256-8

		HM62256-8		
Item	**Symbol**	**Min**	**Max**	**Unit**
Read cycle time	t_{RC}	85	—	ns
Address access time	t_{AA}	—	85	ns
Chip select access time	t_{ACS}	—	85	ns
Output enable access time	t_{OE}	—	45	ns
Output hold time	t_{OH}	5	—	ns
Chip select to output set	t_{CLZ}	10	—	ns
Output enable to output set	t_{OLZ}	5	—	ns
Chip deselect to output float	t_{CHZ}	0	30	ns
Output disable to output float	t_{OHZ}	0	30	ns

AC Characteristics

3. HM62256-8 (cont)

		HM62256-8		
Item	**Symbol**	**Min**	**Max**	**Unit**
Write cycle time	t_{WC}	85	—	ns
Chip select time	t_{CW}	75	—	ns
Address valid time	t_{AW}	75	—	ns
Address setup time	t_{AS}	0	—	ns
Write pulse width	t_{WP}	60	—	ns
Address hold time	t_{WR}	10	—	ns
$\overline{WE}$ to output float	t_{WHZ}	0	30	ns
Input data set time	t_{DW}	40	—	ns
Input data hold time	t_{DH}	0	—	ns
Output disable to output float	t_{OHZ}	0	30	ns
$\overline{WE}$ to output set	t_{OW}	5	—	ns

Specification

1. The H8/330's expanded mode (mode 2) is used to interface the H8/330 to a timer chip (HD63140) as shown in figure 1.

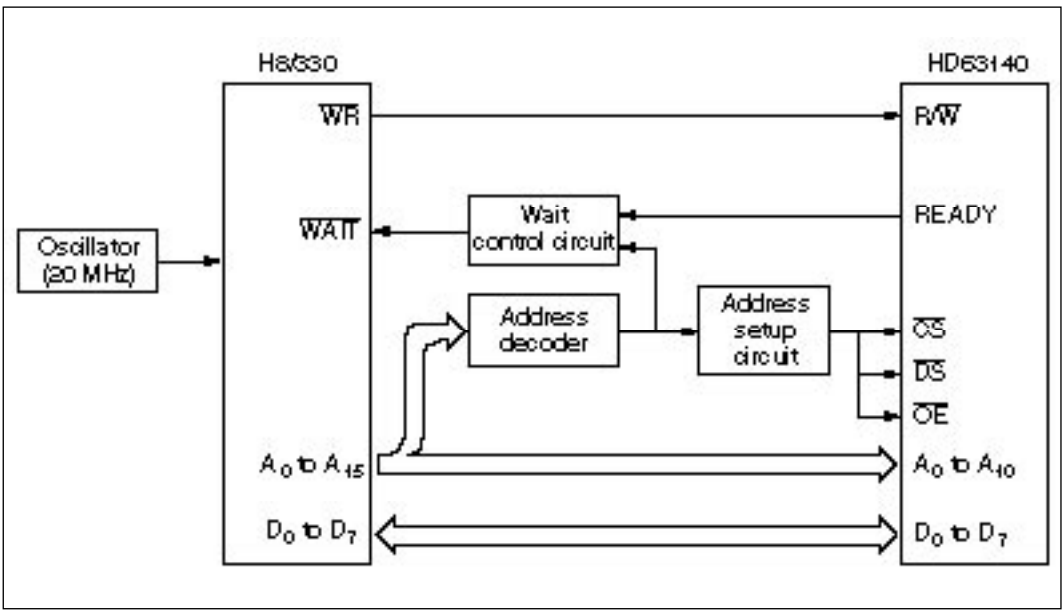


Figure 1 H8/330-HD63140 Interface Block Diagram

Specification

2. The HD63140 is allocated as shown in figure 2.

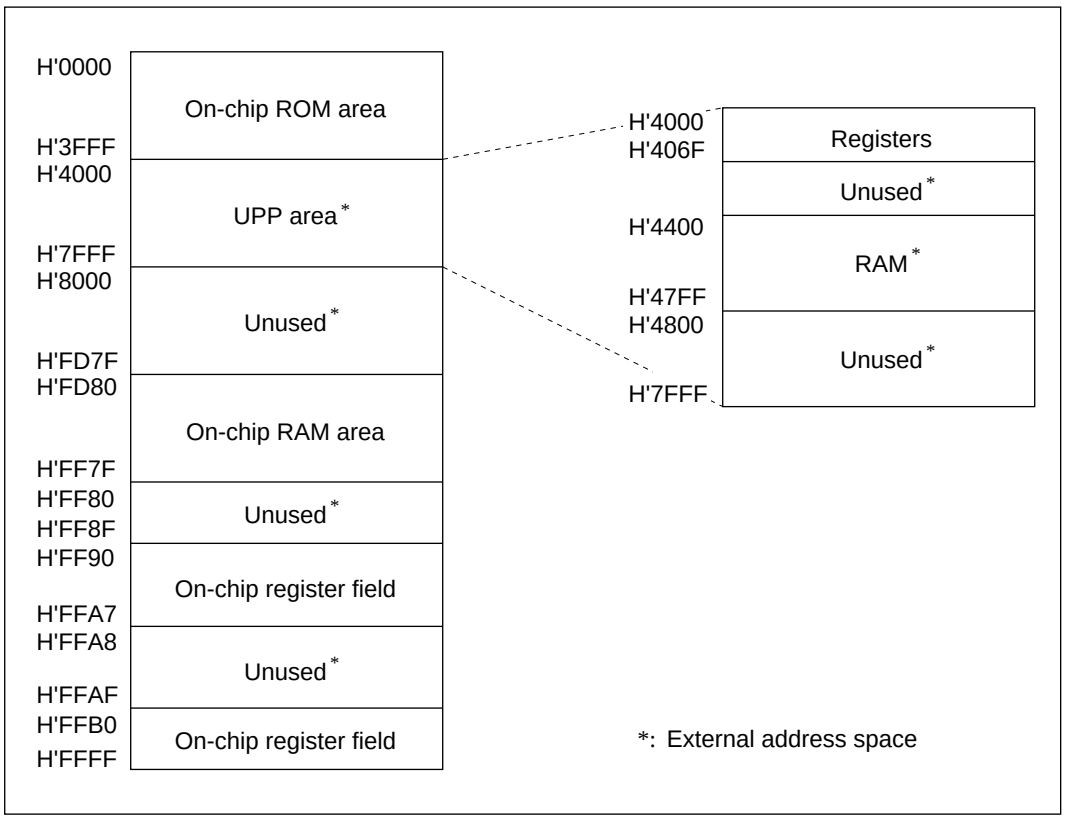


Figure 2 Memory Map

3. When the HD63140's registers and RAM are written or read, the wait control circuit inserts wait states as follows:

   HD63140 registers: 1 state + interval while READY is low
   HD63140 RAM: 1 state

Operation

1. Timing

   a. H8/330 bus cycle (with wait states)

   Figure 3 shows the basic bus cycle of the H8/330 when wait states are inserted. If the WAIT line is low at the falling edge of T_2, a wait state (T_W) is inserted into the basic bus cycle as indicated in figure 3. If the WAIT line is still low at the fall of T_W, another T_W is inserted.

   The H8/330 latches data (D_7 to D_0) on the falling edge of T_3 in a read cycle, and outputs data with a delay of 60 to 70 ns (T_{WDD}) from the falling edge of T_1 in a write cycle. This is the same as when no wait states are inserted.

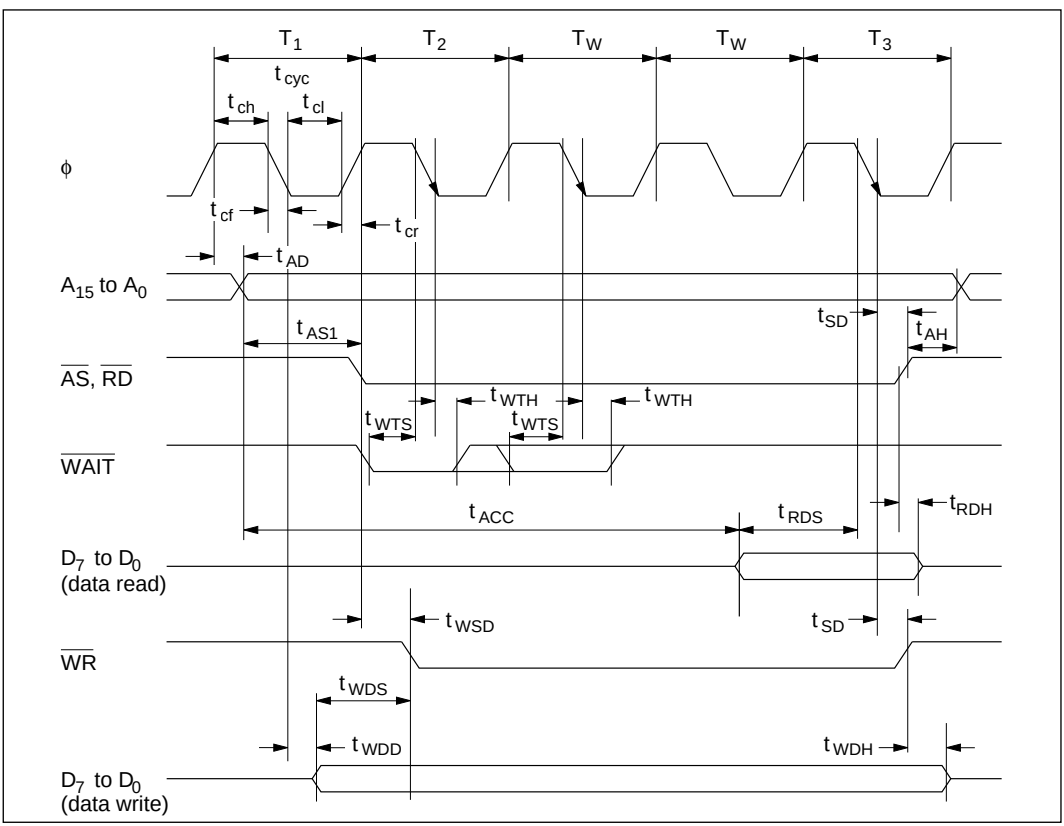


Figure 3 Basic H8/330 Bus Cycle

Operation

b. HD63140 RAM read/write timing

Figure 4 shows the read/write timing for the HD63140's RAM. The timing requirements to be checked are the H8/330's t_{RDS} (read data setup time: minimum 50 ns) and the HD63140's t_{WDS} (write data setup time: minimum 100 ns).

If no wait states are inserted, t_{RDS} cannot be satisfied, so it is necessary to insert one wait state.

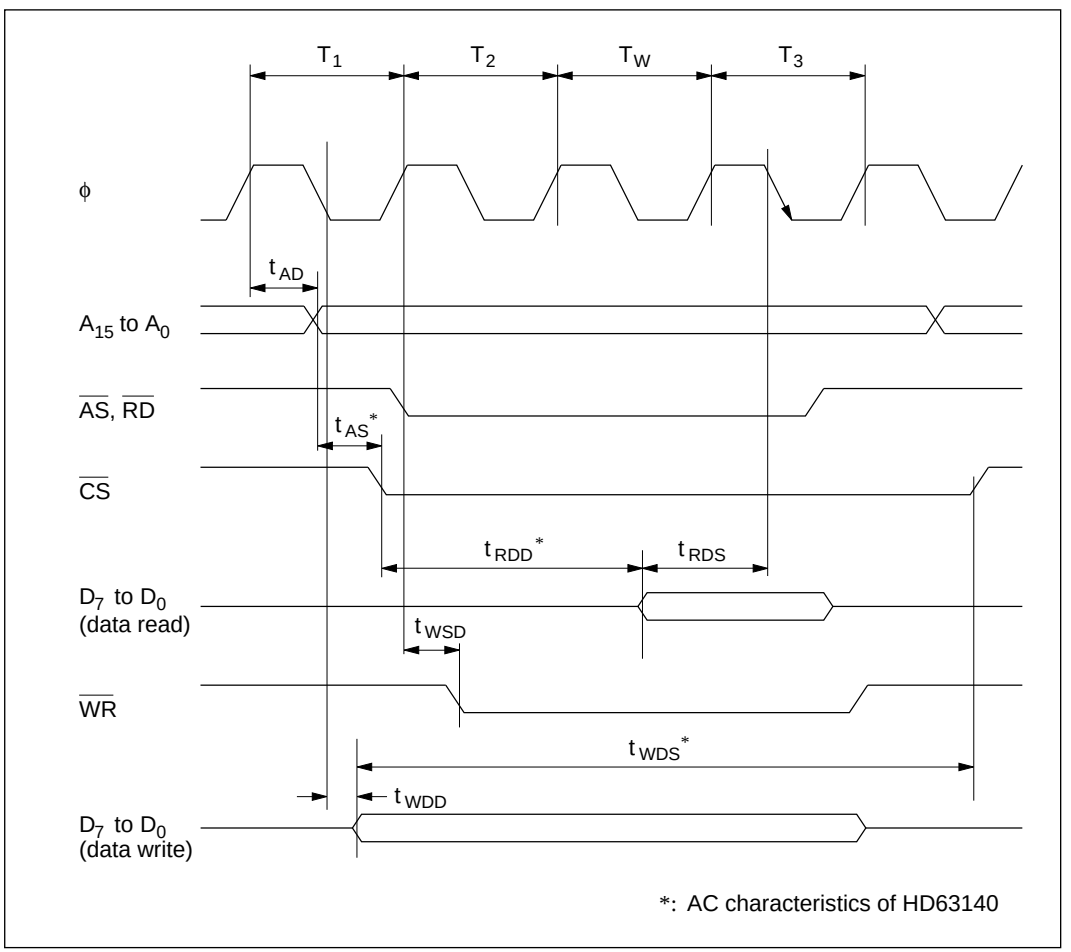


Figure 4 HD63140 RAM Read/Write Timing Diagram

Operation

c. HD63140 register read/write timing

Figure 5 shows the HD63140's register read/write timing. The HD63140's registers cannot be written or read while the READY signal is low (t_{WAIT}: 750 ns to 3 µs), so wait states must be inserted during this interval.

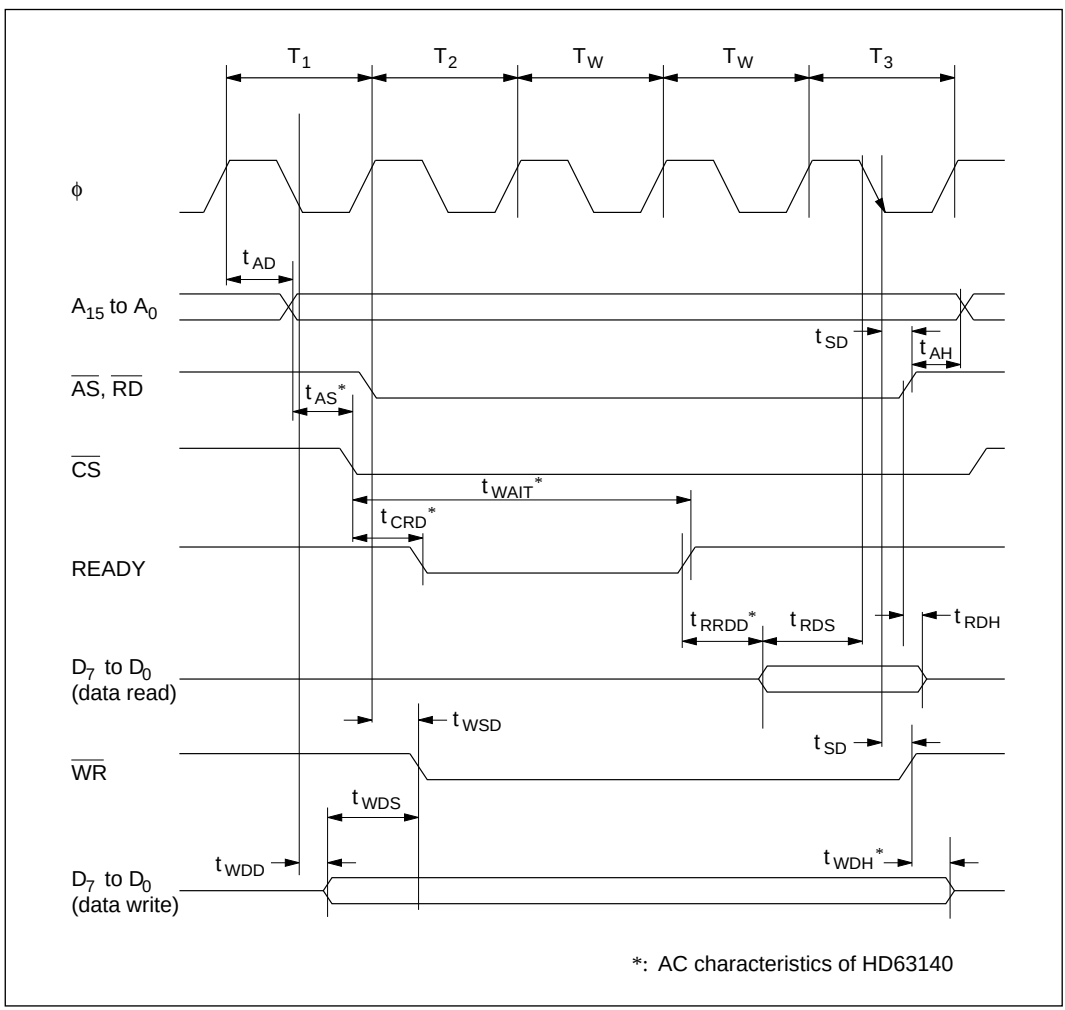


Figure 5 HD63140 Register Read/Write Timing Diagram

Operation

2. Wait control circuit

Figure 6 shows a diagram of the wait control circuit. The Q1 and Q2 outputs insert one wait state when the HD63140's RAM is written or read.

When the HD63140's registers are written or read, Q1 and the HD63140's READY signal are used to insert wait states while the READY signal is low.

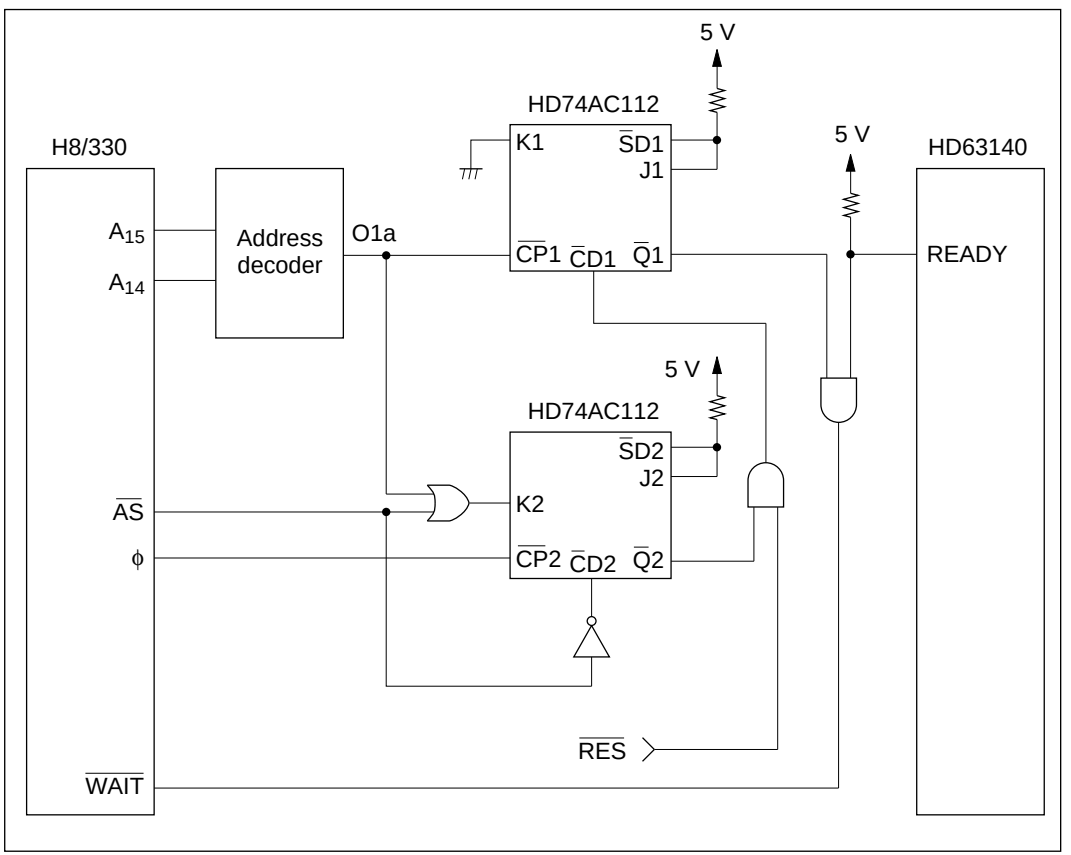


Figure 6 Wait Control Circuit

Operation

Figure 7 shows timing diagrams for the wait-state control circuit.

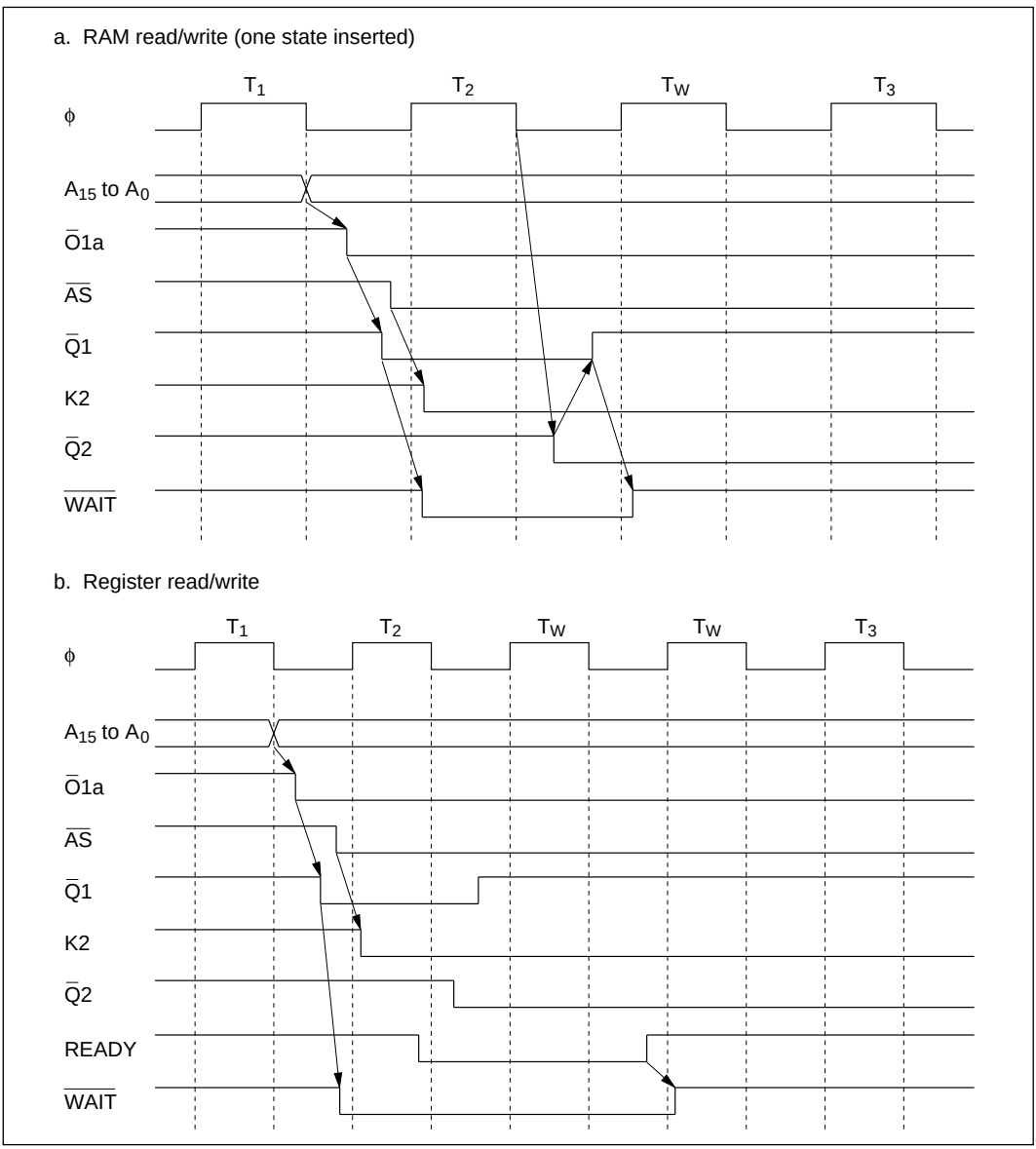


Figure 7 Wait Control Circuit Timing Diagrams

Operation

3. Address setup circuit

Figure 8 shows a diagram of the address setup circuit. The Q1 output generates the HD63140's t_{AS} (address setup time: minimum 30 ns).

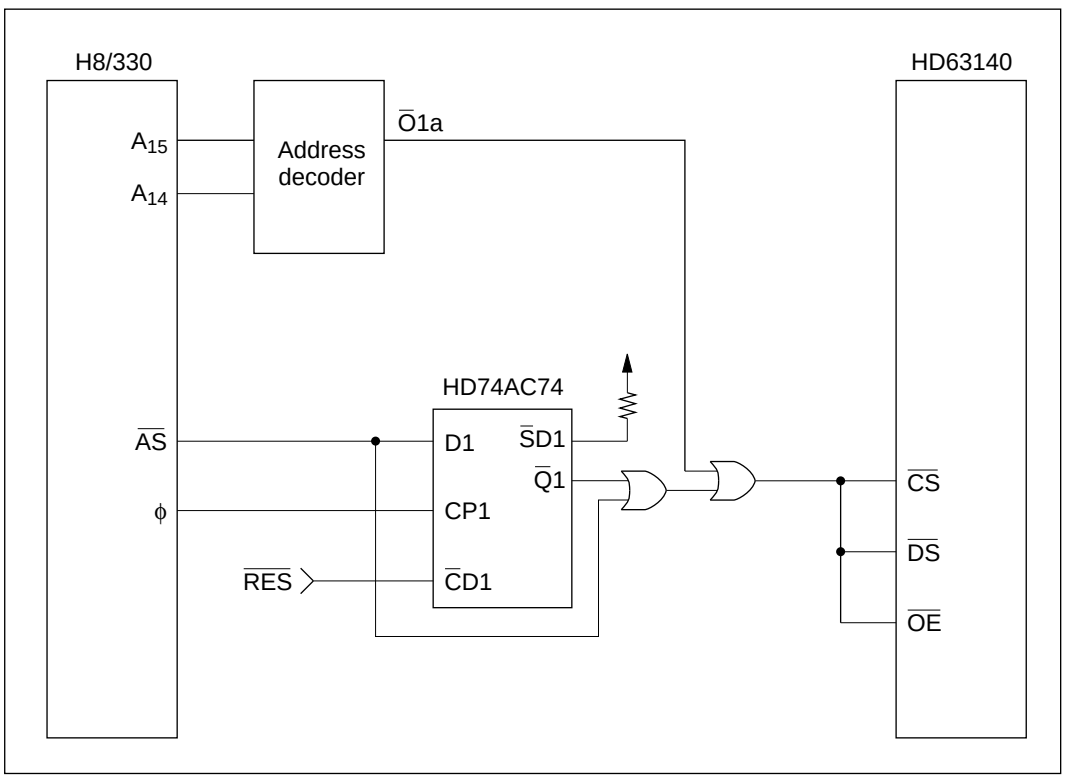


Figure 8 Address Setup Circuit

Operation

Figure 9 shows a timing diagram for the address setup circuit.

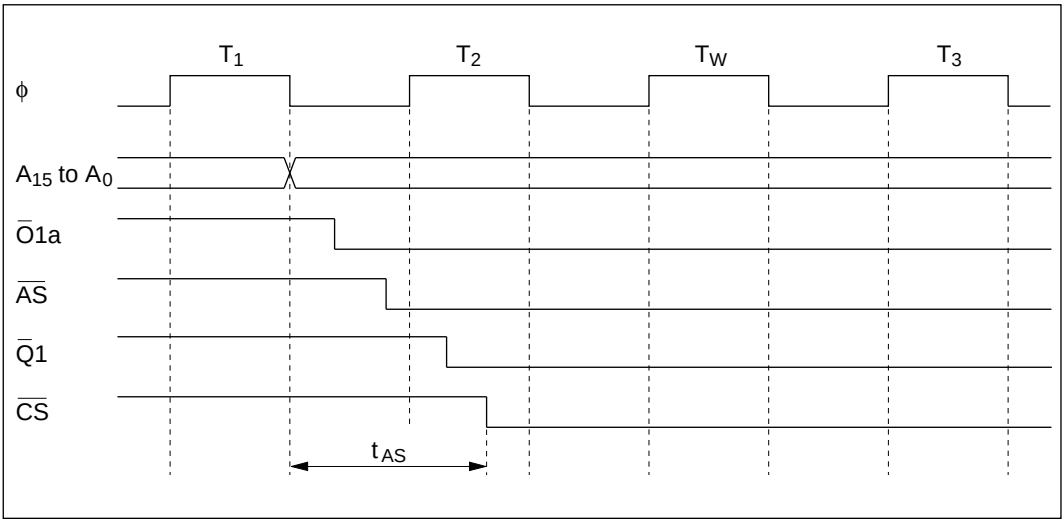
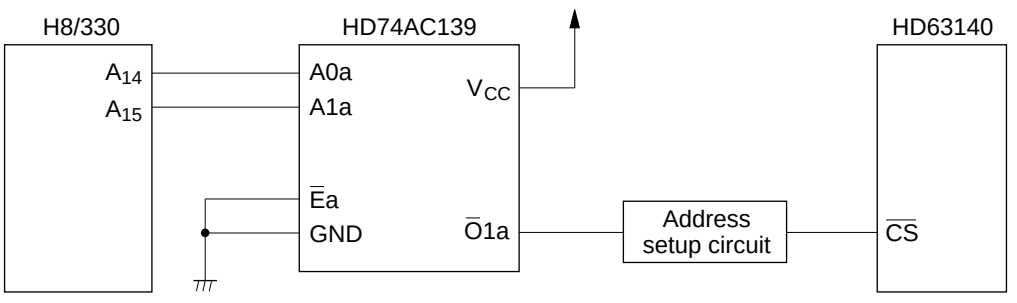


Figure 9 Address Setup Circuit Timing Diagram

Operation

4. Address decoding

An HD74AC139 is used for address decoding as indicated in figure 10.



Address	A_{14}	A_{15}	HD63140
H'4000 to H'7FFF	1	0	Enable

Figure 10 Address Decoder

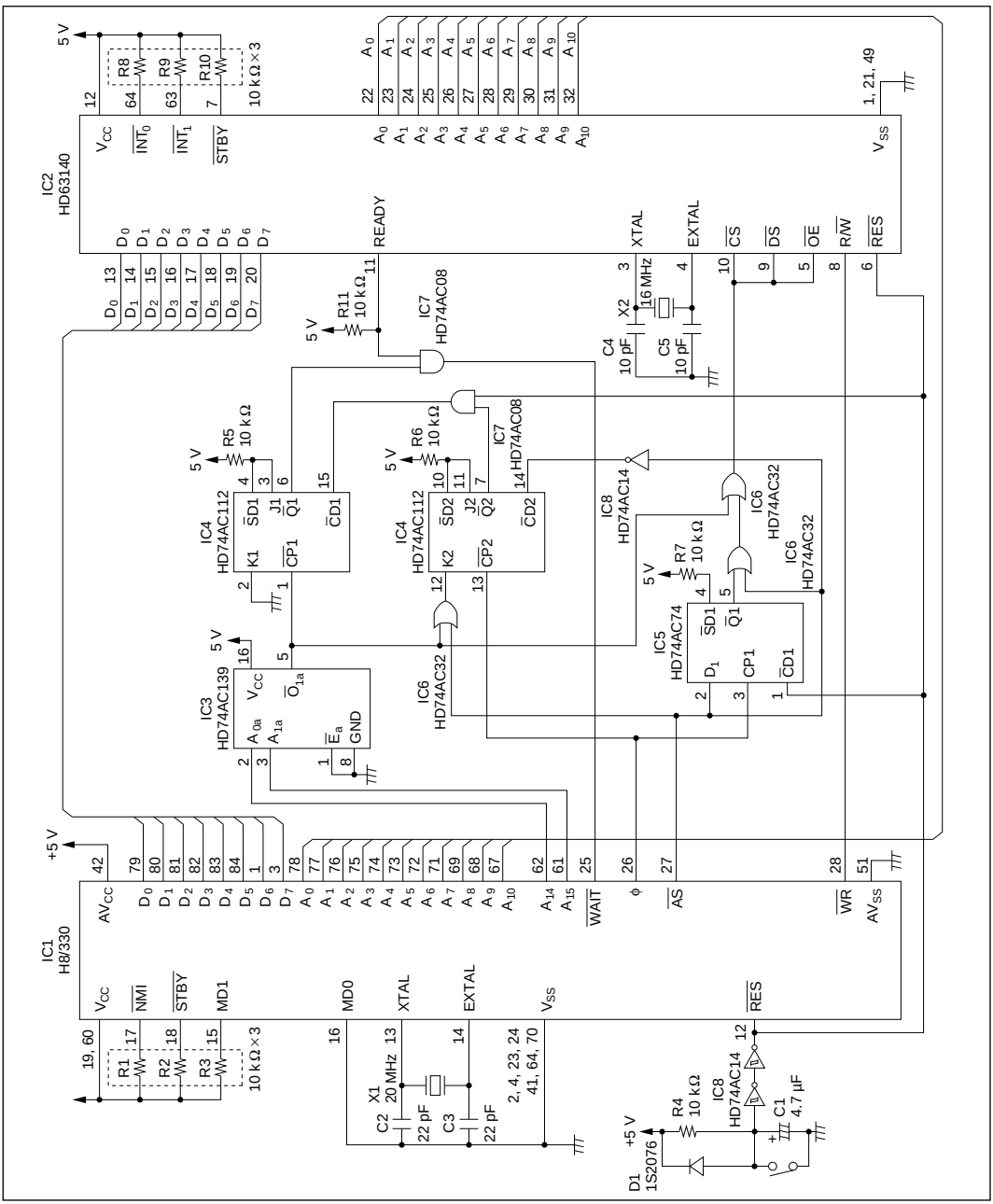


Figure 11 HD63140 Interface Circuit Diagram

AC Characteristics

1. H8/330

Item	Symbol	6 MHz Min	6 MHz Max	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Unit
Clock cycle time	t_{cyc}	166.7	2000	125	2000	100	2000	ns
Clock low pulse width	t_{CL}	65	—	45	—	35	—	ns
Clock high pulse width	t_{CH}	65	—	45	—	35	—	ns
Clock rise time	t_{Cr}	—	15	—	15	—	15	ns
Clock fall time	t_{Cf}	—	15	—	15	—	15	ns
Address delay time	t_{AD}	—	70	—	60	—	55	ns
Address hold time	t_{AH}	30	—	25	—	20	—	ns
Address strobe delay time	t_{ASD}	—	70	—	60	—	50	ns
Write strobe delay time	t_{WSD}	—	70	—	60	—	50	ns
Strobe delay time	t_{SD}	—	70	—	60	—	50	ns
Write strobe pulse width	t_{WSW}	200	—	150	—	120	—	ns
Address setup time 1	t_{ASI}	25	—	20	—	15	—	ns
Address setup time 2	t_{AS2}	105	—	80	—	65	—	ns
Read data setup time	t_{RDS}	70	—	55	—	50	—	ns
Read data hold time	t_{RDH}	0	—	0	—	0	—	ns
Write data delay time	t_{WDD}	—	70	—	60	—	60	ns
Read data access time	t_{ACC}	—	270	—	190	—	160	ns
Write data setup time	t_{WDS}	30	—	15	—	10	—	ns
Write data hold time	t_{WDH}	30	—	25	—	20	—	ns
Wait setup time	t_{WTS}	40	—	40	—	40	—	ns
Wait hold time	t_{WTH}	10	—	10	—	10	—	ns
E clock delay time	t_{ED}	—	20	—	20	—	20	ns
E clock rise time	t_{Er}	—	15	—	15	—	15	ns
E clock fall time	t_{Ef}	—	15	—	15	—	15	ns
Read data hold time (synchronized with E clock)	t_{RDHE}	0	—	0	—	0	—	ns
Write data hold time (synchronized with E clock)	t_{WDHE}	50	—	40	—	30	—	ns

AC Characteristics

2. HD63140

Item		Symbol	Min	Typ	Max	Unit
Oscillator stabilization time		t_{RC}	—	—	20	ms
Operating frequency		f_{opr}	1.0	—	4.0	MHz
Output clock frequency		f_{CLK}	—	f_{opr} ¥ 2	—	MHz
Output clock high pulse width		t_{CWH}	55	—	—	ns
Output clock low pulse width		t_{CWL}	55	—	—	ns
Output clock rise time		t_{Cr}	—	—	10	ns
Output clock fall time		t_{Cf}	—	—	10	ns
Address setup time		t_{AS}	30	—	—	ns
Address hold time		t_{AH}	5	—	—	ns
Delay time from $\overline{CS}$ low to READY low (except RAM)		t_{CRD1}	—	—	60	ns
Delay time from $\overline{DS}$ low to READY high	UDR (UPP)*	t_{WAIT}	—	—	3	µs
	Other*		—	—	750	ns
Delay time from $\overline{DS}$ high to READY low		t_{CRD2}	—	—	80	ns
$\overline{DS}$ high pulse width		t_{DWH}	80	—	—	ns
R/$\overline{W}$ setup time		t_{RS}	10	—	—	ns
R/$\overline{W}$ hold time		t_{RH}	5	—	—	ns
Read data delay time (RAM)		t_{RDD}	—	—	140	ns
Read data delay time from READY high		t_{RRDD}	—	—	60	ns
Read data delay time from $\overline{OE}$		t_{ORDD}	—	—	80	ns
Read data hold time		t_{RDH}	10	—	—	ns
Read data hold time from $\overline{OE}$ high		t_{ORDH}	10	—	—	ns
Write data delay time*		t_{WDD}	—	—	120	ns
Write data setup time		t_{WDS}	100	—	—	ns
Write data hold time from READY high*		t_{WH}	120	—	—	ns
Low write pulse width (RAM)		t_{WWL}	100	—	—	ns
Write data hold time		t_{WDH}	5	—	—	ns
READY turn-off time from $\overline{CS}$ high		t_{RTO}	—	—	50	ns

*: f_{opr} = 4 MHz. Value is inversely proportional to operating frequency f_{opr}.

Specification

1. The H8/330 is interfaced to a microcomputer system power controller IC (HA16103) as shown in figure 1.

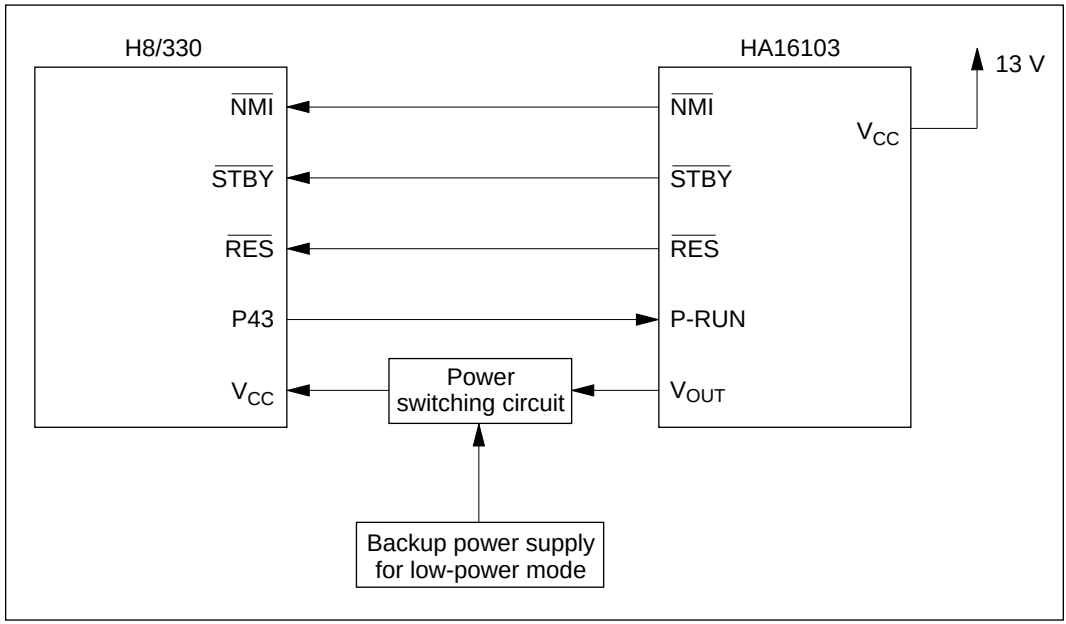


Figure 1 H8/330-HA16103 Interface Block Diagram

2. The HA16103 carries out the following supervisory functions in a microcomputer system.

   a. A watchdog timer function detects microcomputer program crashes.

   b. A low-voltage supervision function monitors the output voltage, and controls the microcomputer's low-power (standby) mode if a voltage drop occurs.

   c. A power-on reset function resets the microcomputer when it is powered on.

   d. A power regulation function supplies the power used by the microcomputer system.

Operation

1. Timing

   a. Watchdog timer function

   The timing diagram in figure 2 illustrates the watchdog timer operation of the HA16103. The HA16103 has an internal band-pulse filter that detects pulse width. If the P-RUN signal deviates from the set frequency, reset pulses are output as shown in figure 2.

   Figure 3 indicates the minimum pulse width tmin of the P-RUN signal in relation to external circuit constants (R_f, C_f). R_f and C_f should be adjusted to give t_{min} as shown in figure 3.

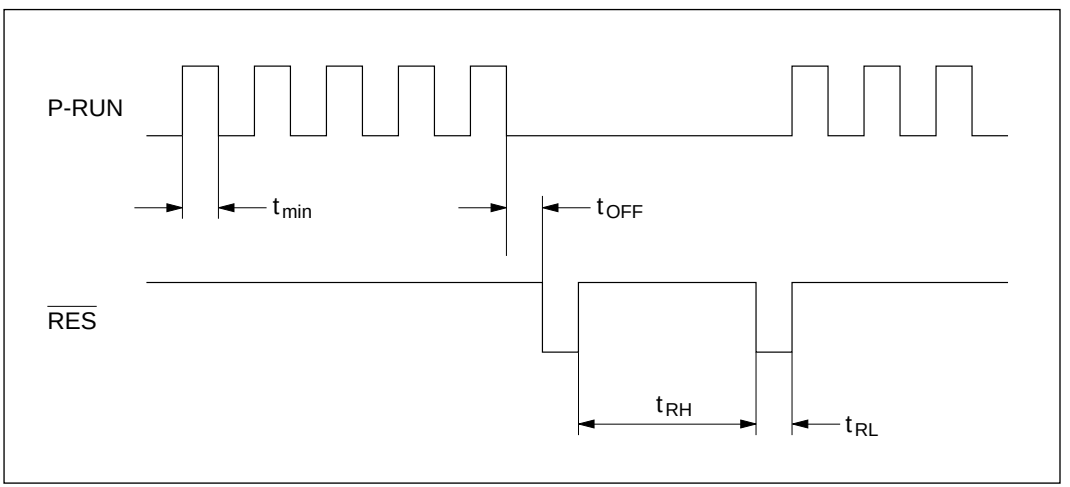


Figure 2 HA16103 Watchdog Timer Timing Diagram

Operation

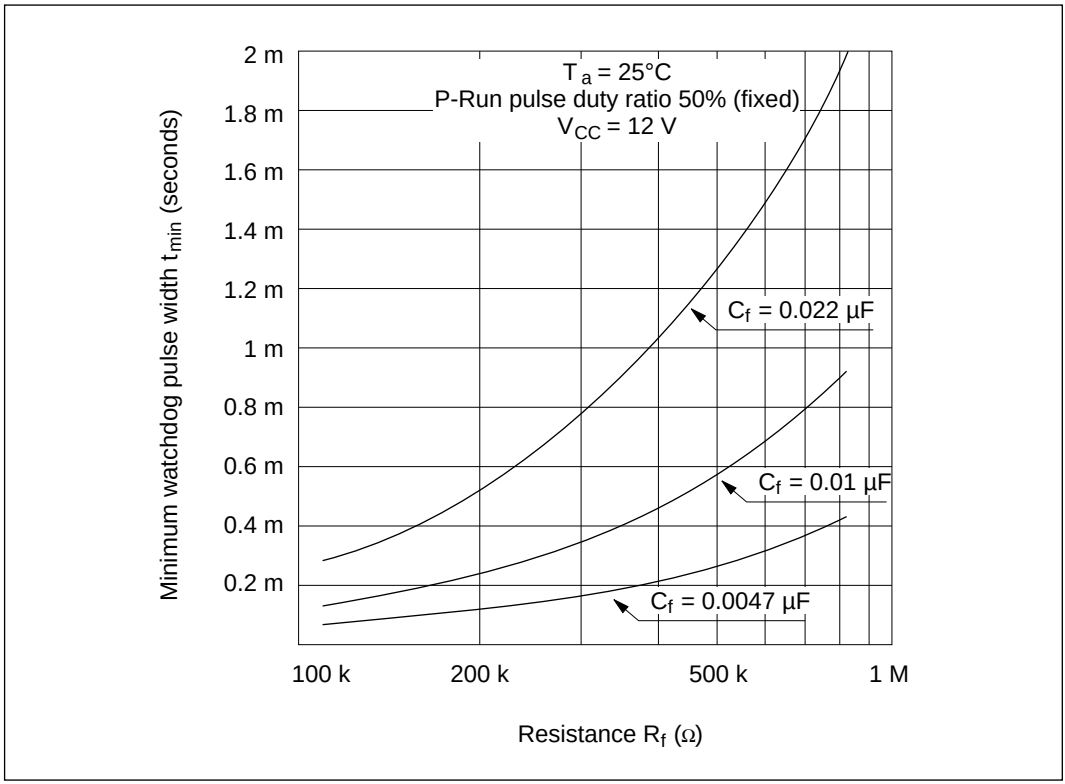


Figure 3 Minimum Pulse Width t_{min} and External Constants (R_f, C_f)

Operation

b. Low-voltage supervision function

The timing diagram in figure 4 illustrates the low-voltage supervision operation of the HA16103. The microcomputer's NMI, RES, and STBY signals are controlled as shown in figure 4 when the output voltage drops below V_{th1} and V_{th2}.

When NMI goes low, microcomputer software can take backup action.

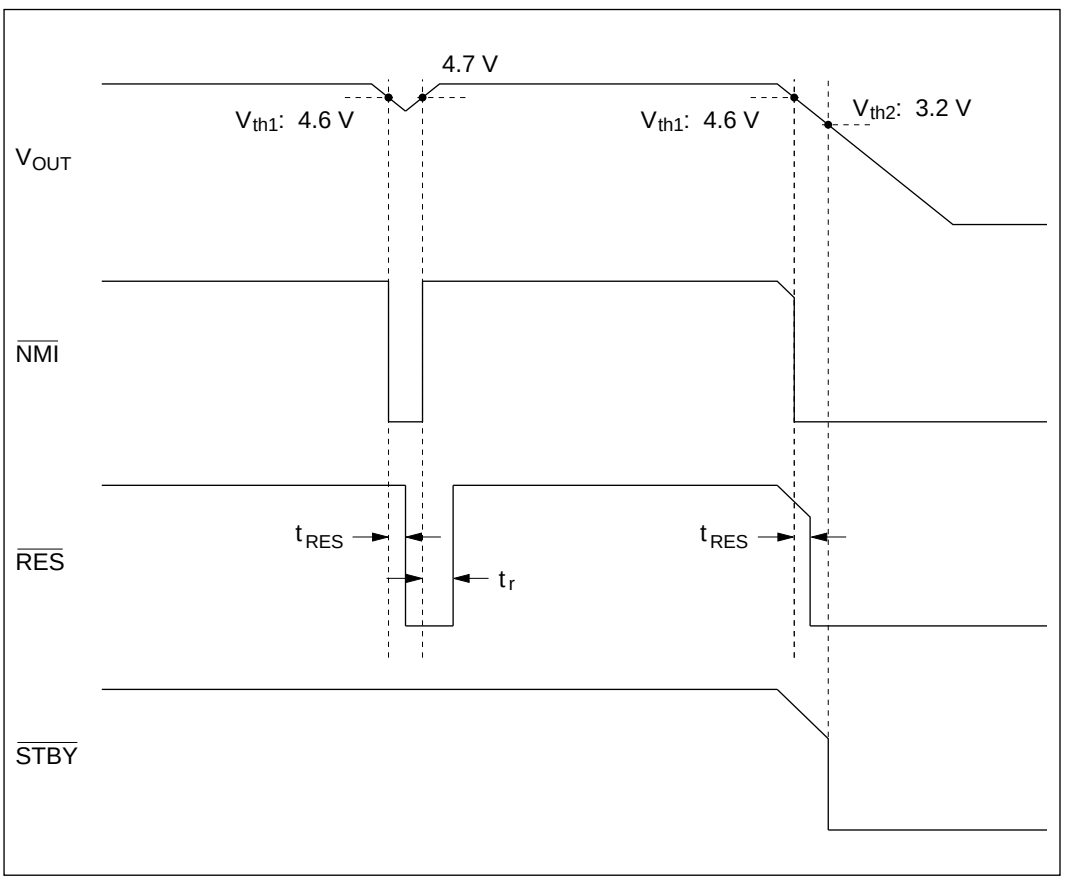


Figure 4 Timing Diagram of Low-Voltage Supervision Function

Operation

c. Power-on reset function

The timing diagram in figure 5 illustrates the power-on reset operation of the HA16103. The microcomputer is reset during the interval t_{ON} generated by the HA16103.

Figure 6 indicates the value of t_{ON} in relation to external circuit constants (R_R, C_R). R_R and C_R should be adjusted to give t_{ON} as shown in figure 3.

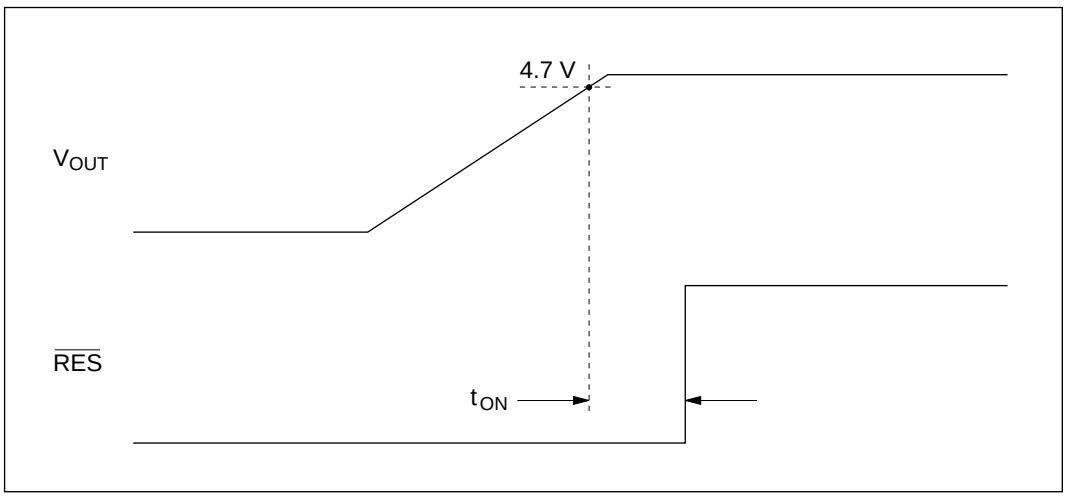


Figure 5 Power-on Reset Timing Diagram

Operation

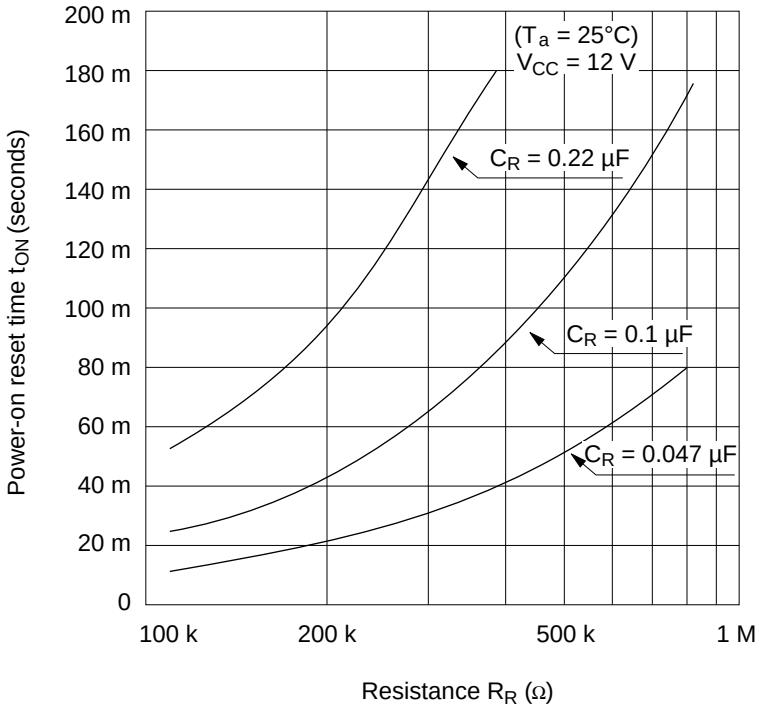


Figure 6 t_{ON} and External Constants (R_R, C_R)

Operation

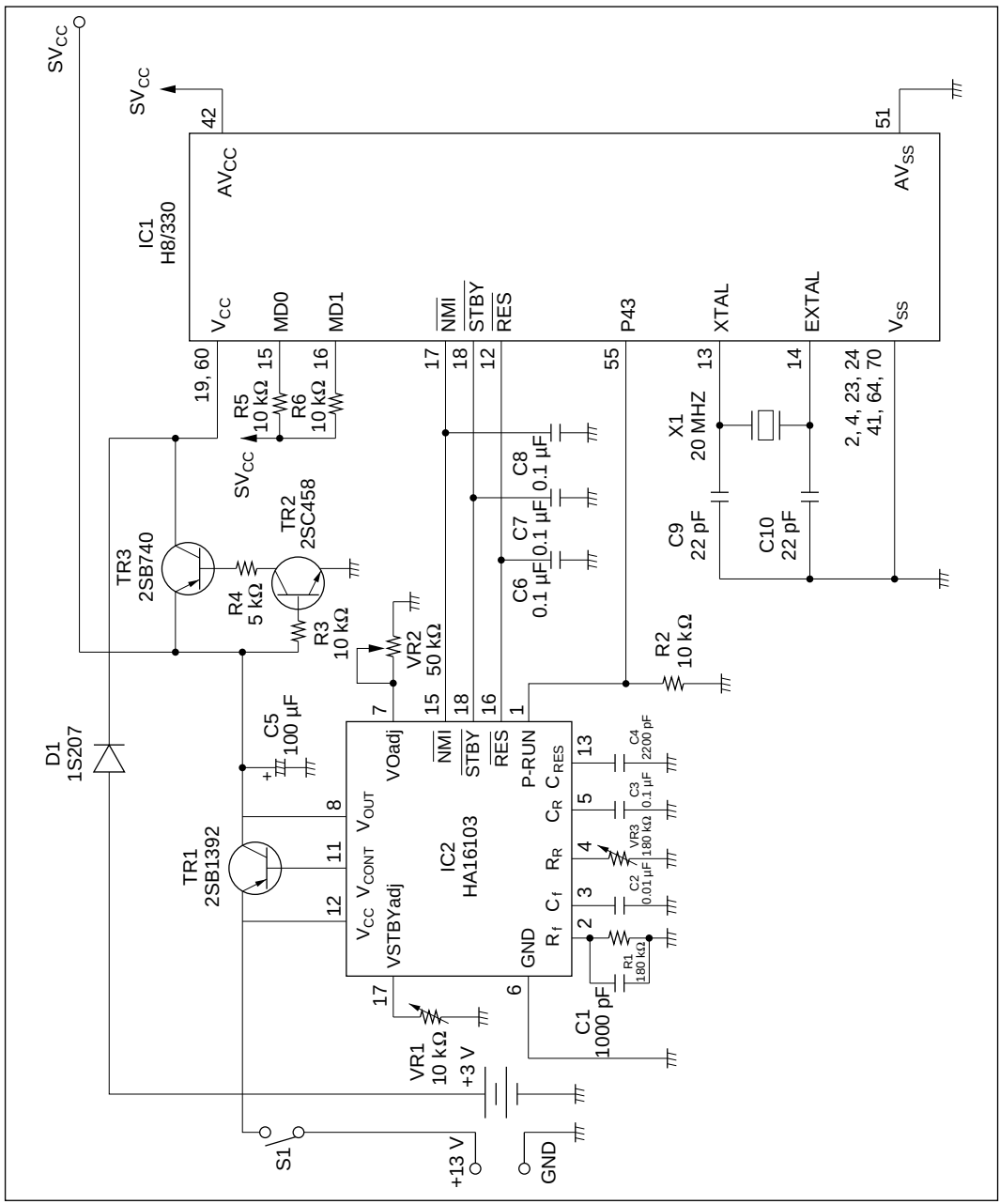


Figure 7 HA16103 Interface Circuit Diagram

AC Characteristics

1. H8/330

Item	Symbol	6 MHz		8 MHz		10 MHz		Unit
		Min	Max	Min	Max	Min	Max	
$\overline{RES}$ setup time	t_{RESS}	200	—	200	—	200	—	ns
$\overline{RES}$ pulse width	t_{RESW}	10	—	10	—	10	—	t_{cyc}
Mode programming setup time	t_{MDS}	4	—	4	—	4	—	t_{cyc}
$\overline{NMI}$ setup time ($\overline{NMI}$, $\overline{IRQ_0}$ to $\overline{IRQ_7}$)	t_{NMIS}	150	—	150	—	150	—	ns
$\overline{NMI}$ hold time ($\overline{NMI}$, $\overline{IRQ_0}$ to $\overline{IRQ_7}$)	t_{NMIH}	10	—	10	—	10	—	ns
Interrupt pulse width ($\overline{NMI}$, $\overline{IRQ_0}$ to $\overline{IRQ_2}$) (for recovery from software standby mode)	t_{NMIW}	200	—	200	—	200	—	ns
Crystal oscillator settling time (reset)	t_{OSC1}	20	—	20	—	20	—	ms
Crystal oscillator settling time (software standby)	t_{OSC2}	10	—	10	—	10	—	ms

AC Characteristics

2. HA16103

Item		Symbol	Min	Typ	Max	Unit	Test Conditions
Entire chip	Supply current	I_{CCL}	—	8	12	mA	V_{CC} = 12 V
Regulator unit	Output voltage	V_{01}	4.80	5.00	5.20	V	V_{CC} = 6 to 17.5 V I_{OUT} = 0.5 A, R1 = 30 kΩ
		V_{02}	4.70	5.00	5.30	V	V_{CC} = 6 to 17.5 V I_{OUT} = 1 A, R1 = 30 kΩ
	Input voltage stability	Voline	–50	—	50	mV	V_{CC} = 6 to 17.5 V I_{OUT} = 1 A, R1 = 30 kΩ
	Load current stability	Voload	–100	—	100	mV	I_{OUT} = 10 mA to 0.5 A, R1 = 30 kΩ
	Ripple rejection	RREJ	45	75	—	dB	Vi = 0.5 Vrms, fi = 1 kHz, R1 = 30 kΩ
	Output voltage temperature coefficient	$\frac{\delta Vo}{\delta T}$	—	0.6	—	mV/°C	V_{CC} = 12 V, R1 = 30 kΩ
Clock input unit	Input low voltage	V_{IL}	—	—	0.8	V	
	Input high voltage	V_{IH}	2.0	—	—	V	
	Input low current	I_{IL}	–120	–60	—	µA	V_{IL} = 0 V
	Input high current	I_{IH}	—	0.3	0.5	mA	V_{IH} = 5 V
$\overline{NMI}$ output unit	$\overline{NMI}$ low voltage	V_{OL1}	—	—	0.4	V	I_{OL1} = 2 mA
	$\overline{NMI}$ high voltage	V_{OH1}	—	V_{out1} (V_{out2})	—	V	
	V_{OUT} at start of $\overline{NMI}$ function	V_{NMI}	—	0.7	1.4	V	
$\overline{STBY}$ output unit	$\overline{STBY}$ low voltage	V_{OL2}	—	—	0.4	V	I_{OL2} = 2 mA
	$\overline{STBY}$ high voltage	V_{OH2}	—	V_{out1} (V_{out2})	—	V	
	V_{OUT} at start of $\overline{STBY}$ function	V_{STBY}	—	0.7	1.4	V	
$\overline{RES}$ output unit	$\overline{RES}$ low voltage	V_{OL3}	—	—	0.4	V	I_{OL3} = 2 mA
	$\overline{RES}$ high voltage	V_{OH3}	—	V_{out1} (V_{out2})	—	V	
	V_{OUT} at start of $\overline{RES}$ function	V_{RES}	—	0.7	1.4	V	
	Power-on time	t_{ON}	25	40	60	ms	R_f = 180 kΩ, R_R = 180 kΩ, C_f = 0.01 µF, C_R = 0.1 µF

AC Characteristics

2. HA16103 (cont)

Item			Symbol	Min	Typ	Max	Unit	Test Conditions
$\overline{RES}$ output unit	Clock-off reset time		t_{OFF}	80	130	190	ms	R_f = 180 kΩ, R_R = 180 kΩ, C_f = 0.01 µF, C_R = 0.1 µF
	Reset pulse low time		t_{RL}	15	20	30	ms	R_f = 180 kΩ, R_R = 180 kΩ, C_f = 0.01 µF, C_R = 0.1 µF
	Reset pulse high time		t_{RH}	37	60	90	ms	R_f = 180 kΩ, R_R = 180 kΩ, C_f = 0.01 µF, C_R = 0.1 µF
Low-input disabling unit (LVI)	Detection point (1)		V_{H1}	4.40	4.60	4.80	V	
	Detection point (1) hysteresis width		V_{HYS1}	50	100	150	mV	
	Detection point (2)		V_{H2}	2.9	3.2	3.5	V	
	Detection point (2) hysteresis width		V_{HYS2}	1.35	1.5	1.65	V	14 pin open
	Reset pulse delay time	When disabled	t_{RES}	—	200	—	µs	C_{RES} = 2200 pF
		At recovery	t_r	—	200	—	µs	C_{RES} = 2200 pF