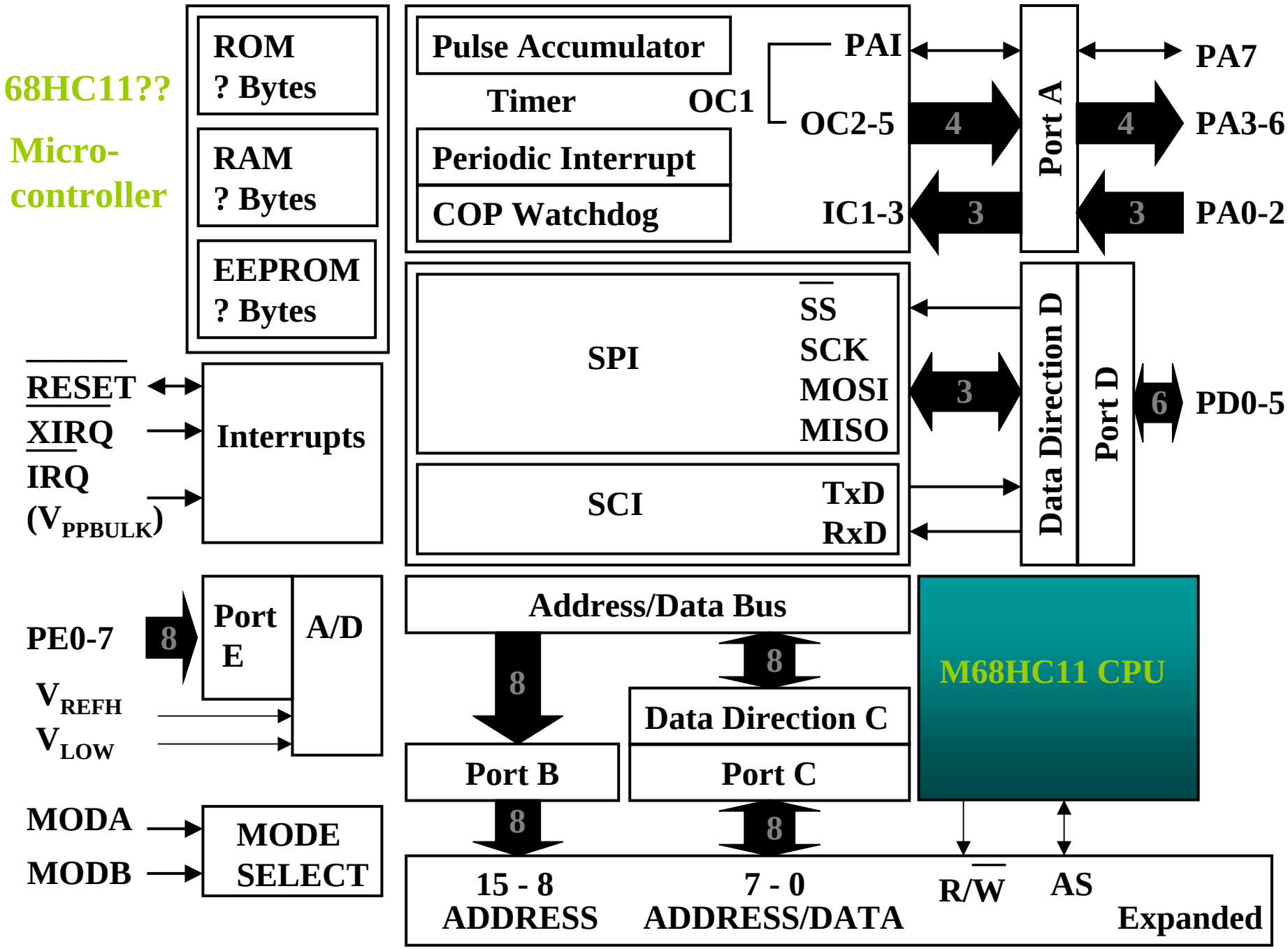
APPLIED MECHATRONICS

M68HC11 MICROCONTROLLER

Prof. P.M. Mujumdar

DEPT. OF AEROSPACE ENGINEERING

IIT BOMBAY


68HC11??
Micro-controller
ROM
? Bytes
RAM
? Bytes
EEPROM
? Bytes
Pulse Accumulator
Timer
OC1
Periodic Interrupt
COP Watchdog
PAI
OC2-5
IC1-3
4
3
Port A
PA7
4
PA3-6
3
PA0-2
SPI
SS
SCK
MOSI
MISO
3
SCI
TxD
RxD
Data Direction D
Port D
6
PD0-5
RESET
XIRQ
IRQ
(VPPBULK)
Interrupts
PE0-7
8
Port E
A/D
VREFH
VLOW
Address/Data Bus
8
8
Data Direction C
Port B
Port C
8
8
M68HC11 CPU
MODA
MODB
MODE SELECT
15 - 8 ADDRESS
7 - 0 ADDRESS/DATA
R/W
AS
Expanded

M68HC11E Series

Device	RAM	ROM	EPROM	EEPROM
MC68HC11E0	512	—	—	—
MC68HC11E1	512	—	—	512
MC68HC11E9	512	12K	—	512
MC68HC711E9	512	—	12K	512
MC68HC11E20	768	20K	—	512
MC68HC711E20	768	—	10K	512
MC68HC811E2	256	—	—	2048

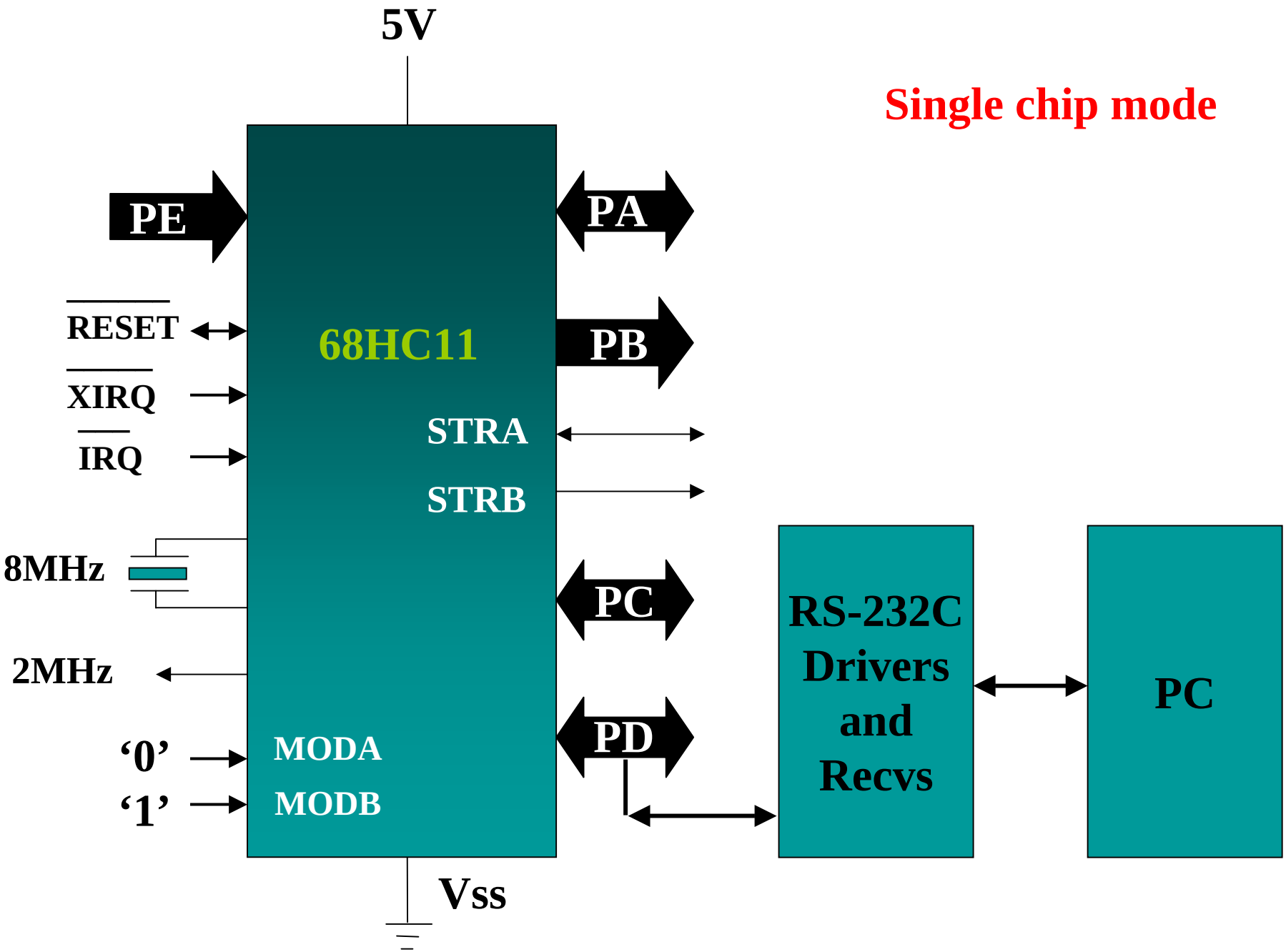
The 68HC11 Operation Modes

- **Single chip mode :**
  a mode in which the 68HC11 functions without external address and data buses. The 68HC11 has 5 I/O ports (A, B, C, D, and E) to use in this mode.

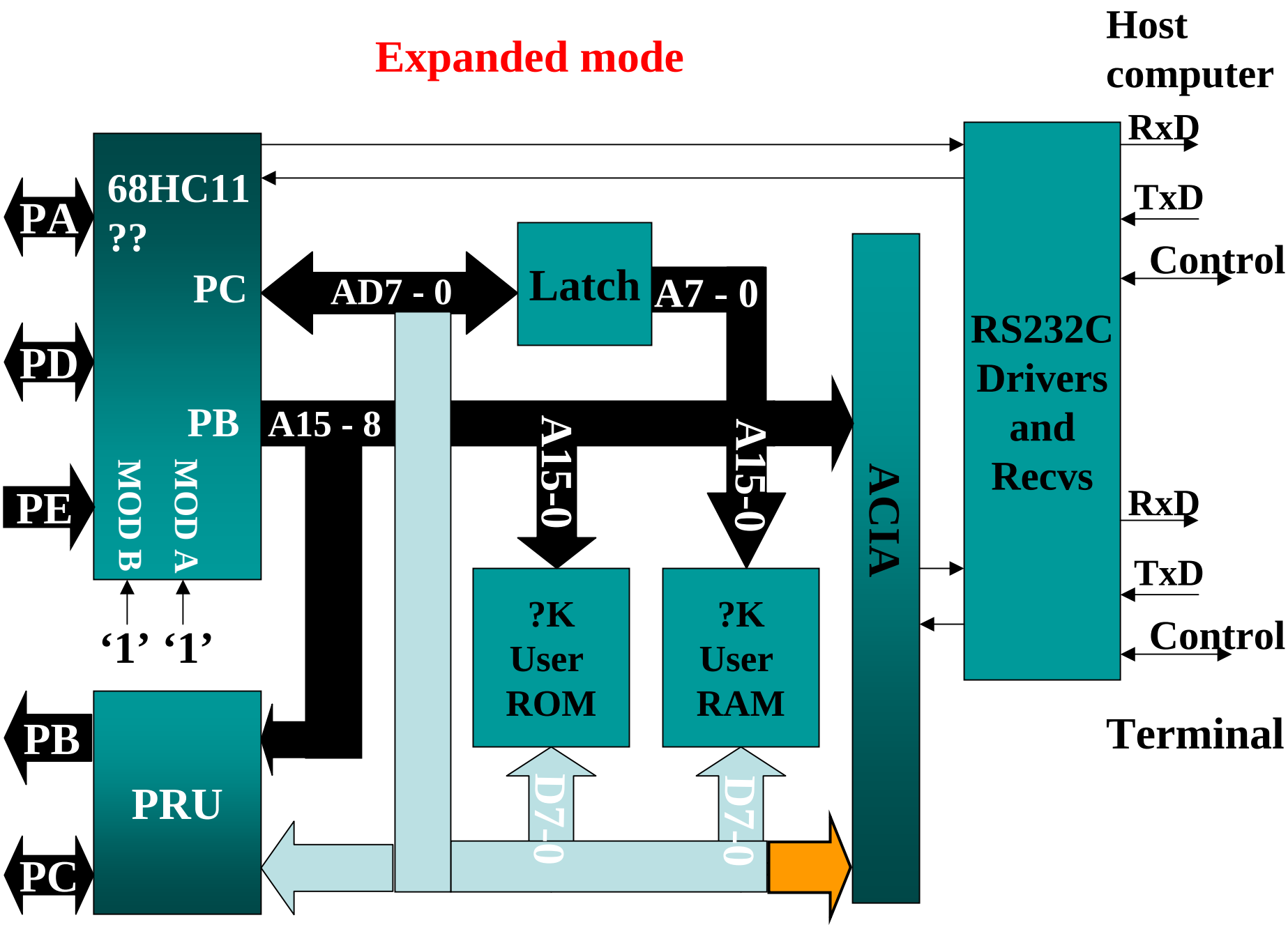
- **Expanded mode:**
  a mode in which the 68HC11 has the capability to access a 64KB address space. In this mode, port B is used as the upper address signals (A15-A8) and port C is used as time-multiplexed address/data bus (A7/D7-A0/D0). Only three I/O ports are available for direct use.

- **Special test mode: mainly used by Motorola in fabrication testing.**

- **Special Bootstrap mode:**
  a mode in which a bootstrap ROM is enabled. The bootstrap ROM contains a loader program that will be executed after the RESET signal is going high and this program will load in a 256-byte program from the SCI subsystem to the on-chip SRAM and then transfer the CPU control to that loaded program.


5V
Single chip mode
PE
PA
RESET
68HC11
PB
XIRQ
STRA
IRQ
STRB
8MHz
PC
2MHz
RS-232C Drivers and Recvs
PC
PD
‘0’
MODA
‘1’
MODB
Vss

Expanded mode

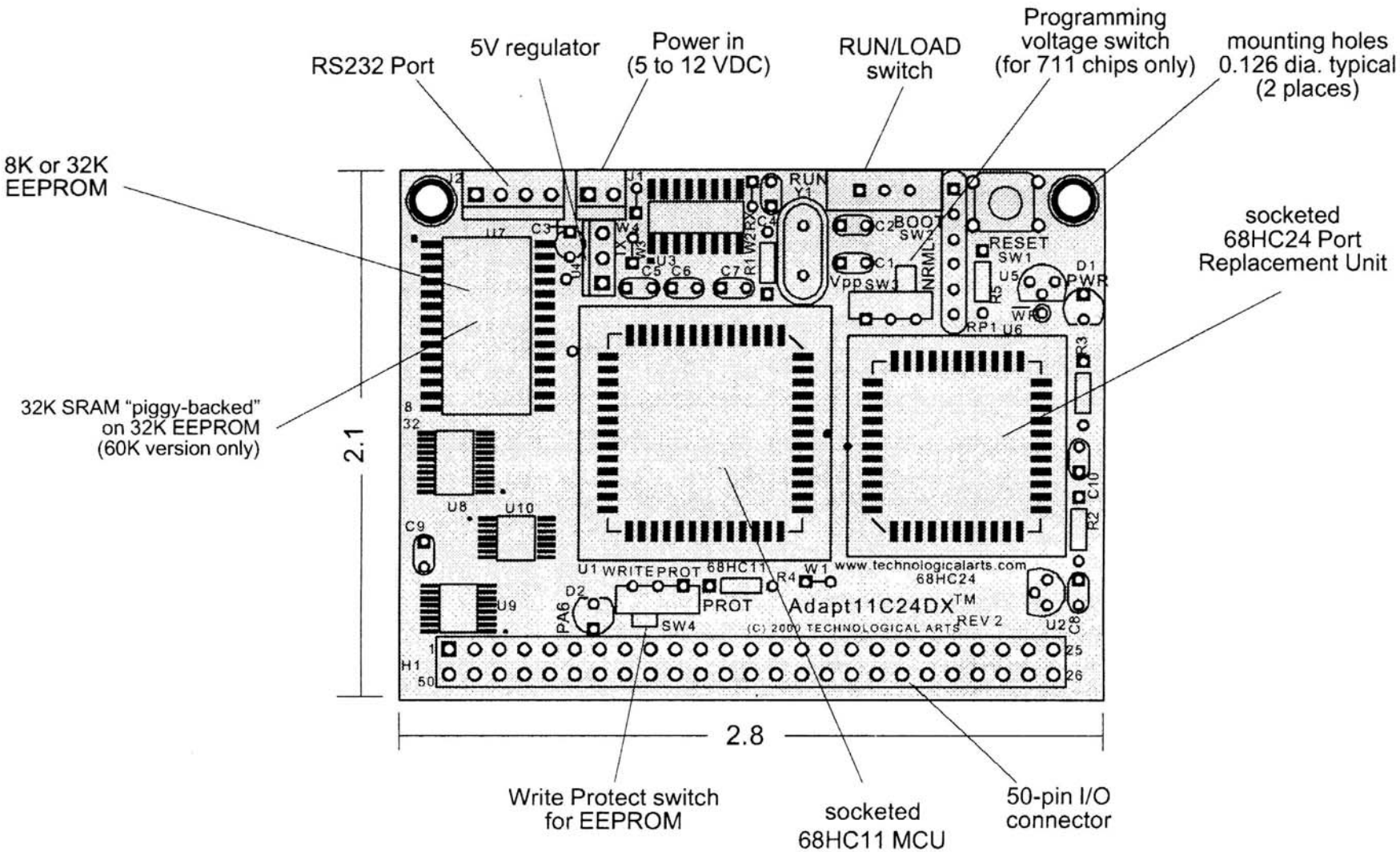


The 68HC24 Port Replacement Unit

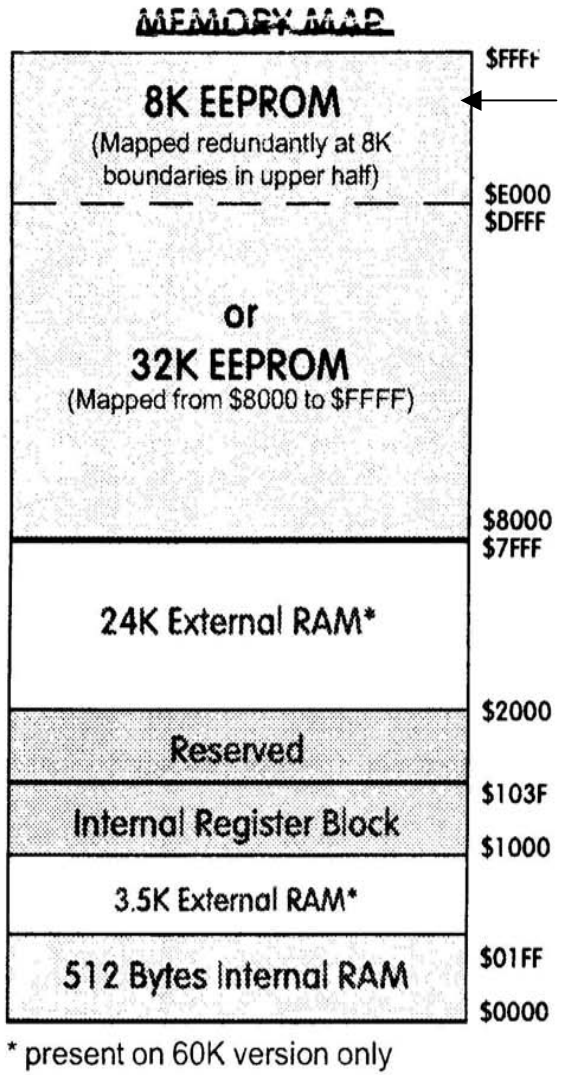
- In the early years of the 68HC11, the amount of on-chip EPROM or EEPROM that can be implemented onto the microcontroller chip is very limited.
- The software of the target embedded product was to be placed in the on-chip ROM.
- During the product development stage, software need to be modified many times. The expanded mode must be chosen and external EPROM must be used so that software can be modified and tested.
- Ports B and C, which were needed in the final product, were lost in the expanded mode during the product development phase.
- The port replacement unit 68HC24 was designed to regain ports B and C so that the products can be evaluated and tested.
- After the design had been tested and evaluated to be satisfactory, the software in the external EPROM could be moved into the internal ROM without modification as long as the external EPROM and internal ROM occupied the same memory space.

Adapt11C24DX Module
Mechanical Data and Pin Configuration

Dimensions in inches



ADAPT11C24DX Boards

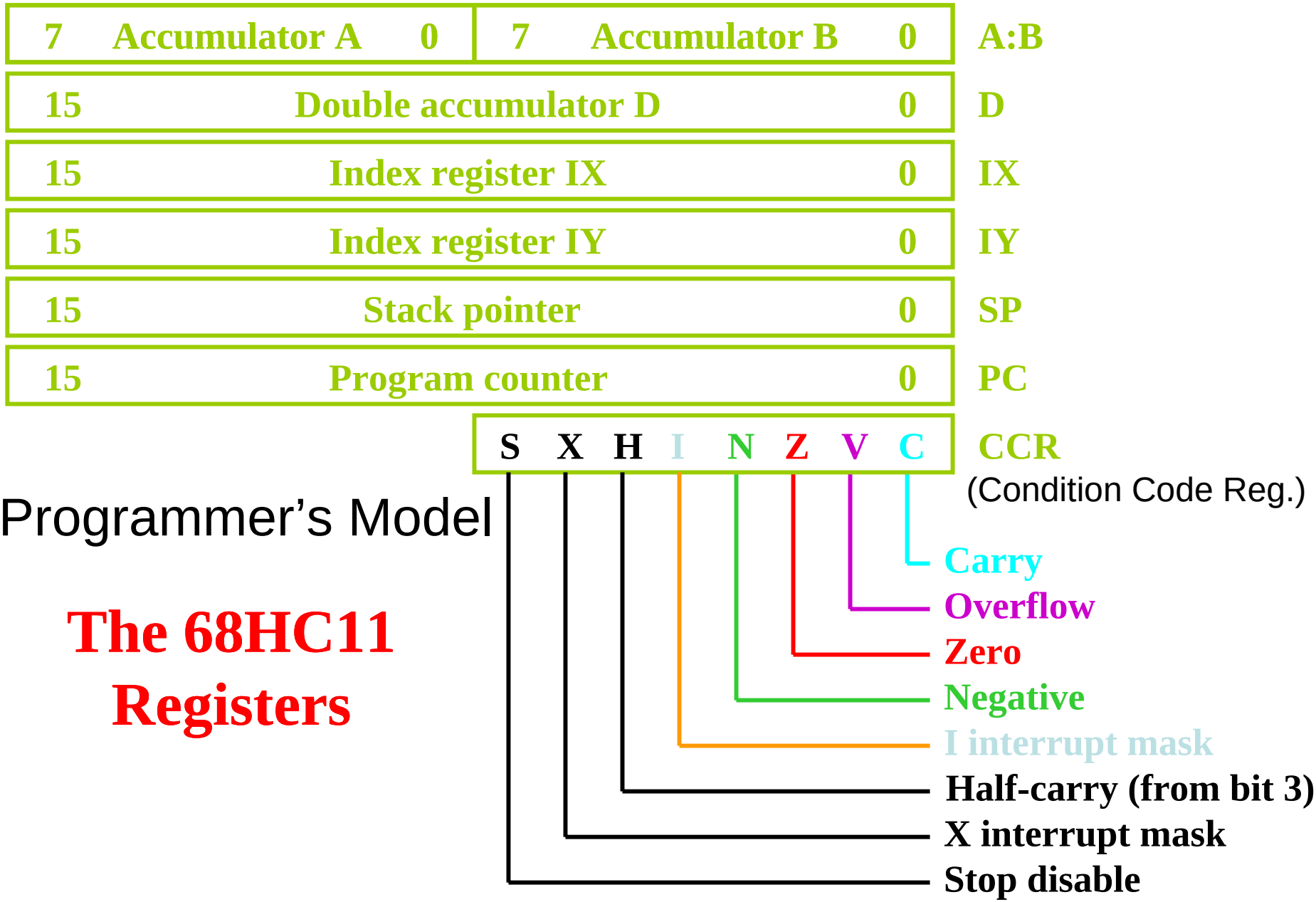


* present on 60K version only

Normal mode
Int. Vectors
$FFC0-$FFFF

Pin Assignments

1	PD2/MISO	50	GROUND
2	PD3/MOSI	49	GROUND
3	PD4/SCK	48	PD0/RXD
4	PD5/SS*	47	+5V
5	PD1/TXD	46	IRQ*
6	PA7/PAI/OC1	45	XIRQ*
7	PA6/OC2/OC1	44	RESET*
8	PA5/OC3/OC1	43	STRB
9	PA4/OC4/OC1	42	PC7
10	PA3/IC4/OC5/OC1	41	PC6
11	PA2/IC1	40	PC5
12	PA1/IC2	39	PC4
13	PA0/IC3	38	PC3
14	PB7	37	PC2
15	PB6	36	PC1
16	PB5	35	PC0
17	PB4	34	STRB
18	PB3	33	E
19	PB2	32	STRA
20	PB1	31	VRL
21	PB0	30	VRH
22	PE0/AN0	29	PE4/AN4
23	PE1/AN1	28	PE5/AN5
24	PE2/AN2	27	PE6/AN6
25	PE3/AN3	26	PE7/AN7



Programmer's Model

The 68HC11 Registers

A flag is a flip flop that keeps track of a changing condition during the execution of some instructions. (See pages 1-1 - 1-4, M68HC11 Reference Manual).

M68HC11 I/O PORTS

	Features	Applications
Port A	Digital, bidirectional port with timer counter circuitry	Measure pulse width 4 PWM Outputs Detecting rising and falling edges Digital I/O
Port B	Digital output only	Can be connected to a DAC Digital output lines
Port C	Digital Bidirectional	I/O operations
Port D	Bidirectional serial communication	Synchronous asynchronous serial communication Other pins for I/O
Port E	Analog input port	Sensor outputs

Vector Interrupt Table

Vector Address	Interrupt Source	CCR Mask Bit	Local Mask
FFC0, C1 – FFD4, D5	Reserved	—	—
FFD6, D7	SCI serial system[1] • SCI receive data register full • SCI receiver overrun • SCI transmit data register empty • SCI transmit complete • SCI idle line detect	I	 RIE RIE TIE TCIE ILIE
FFD8, D9	SPI serial transfer complete	I	SPIE
FFDA, DB	Pulse accumulator input edge	I	PAII
FFDC, DD	Pulse accumulator overflow	I	PAOVI
FFDE, DF	Timer overflow	I	TOI
FFE0, E1	Timer input capture 4/output compare 5	I	I4/O5I
FFE2, E3	Timer output compare 4	I	OC4I
FFE4, E5	Timer output compare 3	I	OC3I
FFE6, E7	Timer output compare 2	I	OC2I
FFE8, E9	Timer output compare 1	I	OC1I
FFEA, EB	Timer input capture 3	I	IC3I
FFEC, ED	Timer input capture 2	I	IC2I
FFEE, EF	Timer input capture 1	I	IC1I
FFF0, F1	Real-time interrupt	I	RTII
FFF2, F3	$\overline{\text{IRQ}}$ (external pin)	I	None
FFF4, F5	$\overline{\text{XIRQ}}$ pin	X	None
FFF6, F7	Software interrupt	None	None
FFF8, F9	Illegal opcode trap	None	None
FFFA, FB	COP failure	None	NOCOP
FFFC, FD	Clock monitor fail	None	CME
FFFE, FF	$\overline{\text{RESET}}$	None	None

1. Interrupts generated by SCI; read SCSR to determine source. Refer to HPRIO register to determine priority of interrupt.

Summary of Features of the Adapt11C24DXSP32K Microcontroller Board

- 8-bit CPU M68HC11 + M68HC24 PRU
- 512 bytes SRAM
- 32K bytes external EEPROM
- 0 KB ROM
- 3/4 input capture channels
- 5/4 output compare functions
- one 8-bit pulse accumulator
- one serial communication interface (SCI)
- one serial peripheral interface (SPI)
- real-time interrupt (RTI) circuit
- 4-channel 8-bit A/D converter
- 8 bit parallel I/O
- computer operate properly (COP) watchdog timer

M68HC11 Timer Functions

Applications that Requires a Dedicated Timer System

- **delay creation and measurement**
- **period measurement**
- **event counting**
- **time-of-day tracking**
- **period interrupt generation to remind the processor to perform routine tasks**
- **waveform generation**
- **etc.**

A Summary of the 68HC11 Timer Functions

1. Main timer

- 16-bit non-stop timer
- read-only after reset

2. Input capture function

- three channels -- 1 to 3
- each channel has a 16-bit latch, edge-detection logic, flag bit and interrupt logic
- will load the current main timer value into the input capture register when the selected signal edge is detected
- can be used to measure the signal frequency, period and pulse width and as time reference

3. Output compare functions

- five channels (OC1…OC5)
- each channel has a 16-bit comparator, 16-bit register, action pin, interrupt request circuit, forced-compare function
- continuously compare the value of the 16-bit compare register with that of the main timer and may optionally trigger an action on a pin, generate an interrupt
- is often used to create a delay and generate a waveform

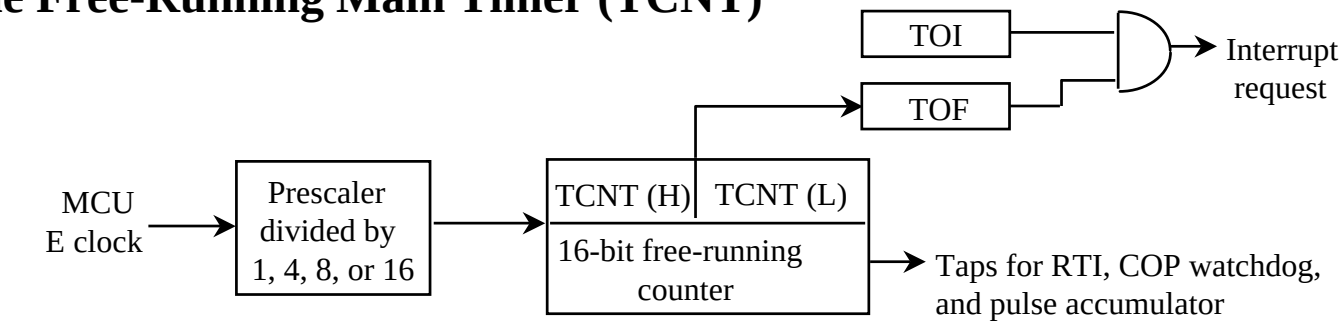
4. Real-time interrupt

- generates periodic interrupts when enabled
- interrupt period is programmable

5. Pulse accumulator

- has an 8-bit counter
- has two operation modes
- used to measure events, frequency, or duration of a pulse width

The Free-Running Main Timer (TCNT)



68HC11 Main Timer System Circuit

	7	6	5	4	3	2	1	0	
	TOI	RTII	PAOII	PAII	0	0	**PR1**	**PR0**	TMSK2
value after reset	0	0	0	0	0	0	0	0	
	TOF	RTIF	PAOVF	PAIF					TFLG2
value after reset	0	0	0	0	0	0	0	0	

- The main timer is cleared to 0 on reset and is read-only except in test mode.
- The timer overflow flag (TOF) in *timer flag register 2* (TFLG2) will be set to 1 when the count changes from $FFFF to $0000.
- The TOF flag can be cleared by writing a 1 to it.
- An interrupt may be generated if the *timer overflow interrupt enable* bit (TOI) in the timer mask register 2 (TMSK2) is set to 1.
- The timer counter register is meant to be read by a 16-bit read instruction such as LDD or LDX.

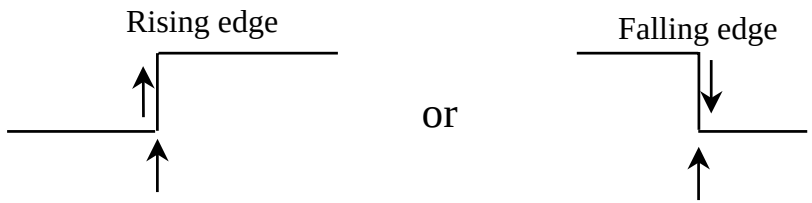
The prescale factor for the main timer is selected by bits 1 and 0 of the timer mask register 2 as shown in the table.

Main Timer Clock Frequency vs. PR1 and PR0 Values

			overflow period	
PR1	PR0	prescale factor	2 MHz E clock	1 MHz E clock
0	0	1	32.77 ms	65.54 ms
0	1	4	131.1 ms	262.1 ms
1	0	8	262.1 ms	524.3 ms
1	1	16	524.2 ms	1.049 s

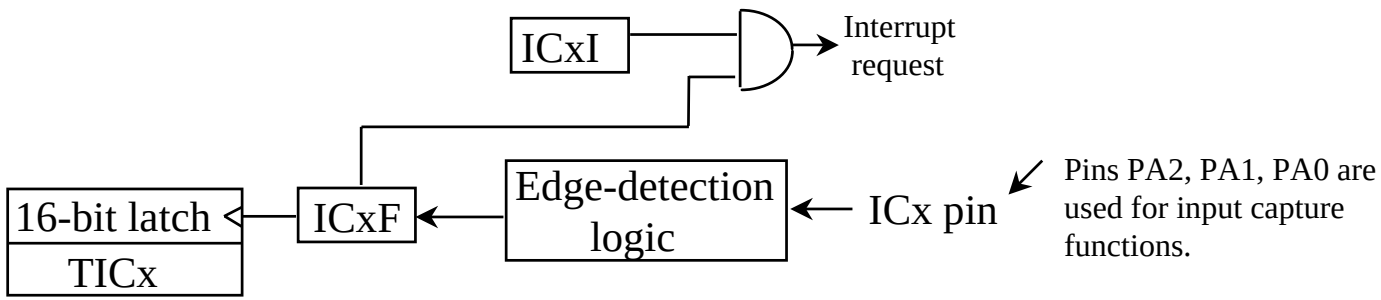
Input Capture Functions

- *Physical time* is often represented by the contents of the main timer.
- The occurrence of an *event* is represented by a signal edge (rising or falling edge).
- The time when an event occurs can be recorded by latching the count of the main timer when a signal arrives.



Events Represented By Signal Edges

- The 68HC11 has three input capture channels (IC1, IC2, & IC3) to implement this operation.
- Each input capture channel has a 16-bit input capture register, a flag, edge-detection logic, and interrupt request circuit.



Input-Capture Function Block Diagram

- The edge to be captured is selected by programming the register TCTL2.

	EDG1A	EDG1B	EDG2A	EDG2B	EDG3A	EDG3B
value after reset	0	0	0	0	0	0

TCTL2

(Timer Control Reg. 2)

EDGxB	EDGxA	
0	0	capture disabled
0	1	capture on rising edge
1	0	capture on falling edge
1	1	capture on both edges

x = 1,2,3

- **Registers related to input capture**

1. Timer Mask Register 1 (TMSK1):

7	6	5	4	3	2	1	0
OC1I	OC2I	OC3I	OC4I	OC5I	**IC1I**	**IC2I**	**IC3I**
0	0	0	0	0	0	0	0

- the lowest three bits (bits 2 to 0) of this register enable/disable the interrupt from the proper input capture channel
- the upper five bits (bits 7 to 3) of this register enable/disable the interrupt from the corresponding output compare channels

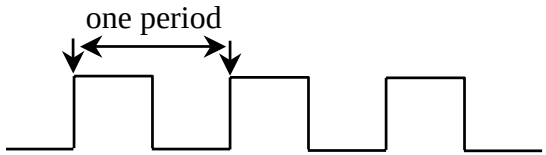
2. Timer Flag Register 1 (TFLG1):

7	6	5	4	3	2	1	0
OC1F	OC2F	OC3F	OC4F	OC5F	**IC1F**	**IC2F**	**IC3F**
0	0	0	0	0	0	0	0

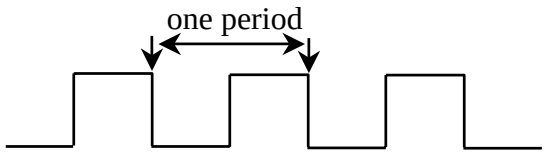
- the lowest three bits (bits 2 to 0) of this register are input capture flags
- the arrival of a signal edge will set one of the input capture flags
- the upper five bits (bits 7 to 3) of this register are output compare flags

Applications of Input Capture function

- Event arrival time recording
- Period measurement: the input capture function captures the main timer values corresponding to two consecutive rising or falling edges



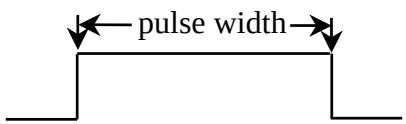
(a) capture two rising edges



(b) capture two falling edges

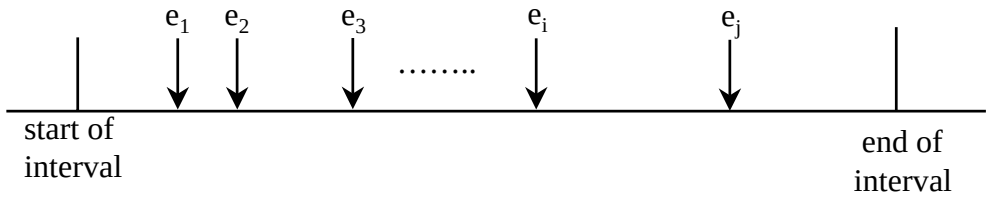
Period Measurement By Capturing Two Consecutive Falling or Rising Edges

- Pulse width measurement: capture the rising and falling edges



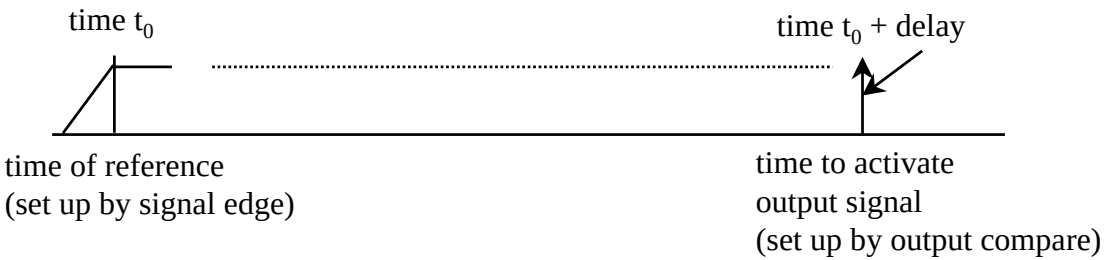
Pulse-Width Measurement Using Input Capture Function

- Interrupt generation: three input capture functions can be used as three edge-sensitive interrupt sources.
- Event counting: by counting the number of signal edges arrived during a period



Using An Input Capture Function For Event Counting

- Time reference: often used in combination with an output compare function

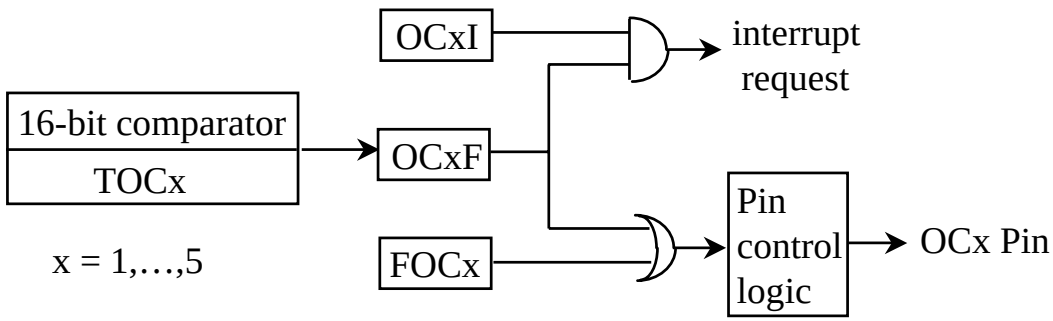


A Time Reference Application

Output Compare Functions

- five output compare channels: OC1-OC5
- port A pins PA7-PA3 are associated with output compare channels OC1-OC5 respectively
- Each output compare channel consists of

  1. a 16-bit comparator
  2. a 16-bit compare register (TOCx, x = 1,…,5)
  3. an output action pin
  4. an interrupt request circuit
  5. a forced-compare function (FOCx, x = 1,…,5)
  6. control logic



Output-Compare Function Block Diagram

- To use an output compare function,

  1. make a copy of the main timer
  2. add to this copy a value equal to the desired delay
  3. store the sum onto an output-compare register

- The actions that can be activated on an output compare pin include

  1. pull up to high
  2. pull down to low
  3. toggle

  The action is determined by the timer control register 1 (TCTL1):

	7	6	5	4	3	2	1	0
	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5
value after reset	0	0	0	0	0	0	0	0

OMx	OLx	
0	0	OCx does not affect pin
0	1	Toggle OCx pin on successful compare
1	0	Clear OCx pin on successful compare
1	1	Set OCx pin on successful compare

x = 2,…,5

The Pulse Accumulator

- 8-bit pulse accumulator (PACNT)
- two operation modes: *event counting* and *gated accumulation* modes
- PACNT is clocked by the PAI input in event counting mode
- PACNT is clocked by the E-divided-by-64 clock in gated accumulation mode
- The PAI pin (PA7 pin) must be configured for input to enable pulse accumulator
- There are two interrupt sources: *PAI pin edge* and the *rollover of PACNT from $FF to $00*
- Four registers are related to the operation of the PACNT: TMSK2, TFLG2, PACTL, PACNT

	7	6	5	4	3	2	1	0	
	TOI	RTII	**PAOVI**	**PAII**	0	0	PR1	PR0	TMSK2
value after reset	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
	TOF	RTIF	**PAOVF**	**PAIF**	0	0	0	0	TFLG2
value after reset	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
	DDRA7	**PAEN**	**PAMOD**	**PEDGE**	0	0	RTR1	RTR0	PACTL
value after reset	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	PACNT
value after reset	-	-	-	-	-	-	-	-	

- The bits 5 and 4 of TMSK2 enables/disables PACNT overflow and PAI edge interrupt respectively.
- The bits 5 and 4 of TFLG2 are pulse accumulator overflow and PAI edge flag respectively.

The Pulse Accumulator Control Register (PACTL)

- bit 7 (DDRA7): 0 -- configure PA7 pin for input; 1 -- configure PA7 for output
- bit 6 (PAEN): 0 -- disable PA function; 1 -- enable PA function
- bit 5 (PAMOD): 0 -- select event-counting mode; 1 -- select gated accumulation mode
- bit 4 (PEDGE): its meaning depends on bit 5

PAMOD	PEDGE	Action on clock
0	0	PAI falling edge increments the PACNT counter
0	1	PAI rising edge increments the PACNT counter
1	0	A low on PAI pin inhibits counting
1	1	A high on PAI pin inhibits counting

Use the PA function to measure frequency

- Set up pulse accumulator to operate in event-counting mode
- Connect the unknown signal to the PAI pin
- Select the active edge (rising or falling)
- Use one of the output compare function to create a delay of one second
- Use a memory location to keep track of the number of active edges arrived in one second.
- Enable pulse accumulator interrupt on active edge. The PA interrupt service routine increments the signal count by 1.
- Disable the pulse accumulator interrupt at the end of one second.

A/D Conversion with M68HC11

A Summary of the 68HC11 A/D Converter

- Eight-channel, 8-bit, multiplexed input, successive-approximation conversion method.
- A weighted array of capacitors are used to implement the successive-a pproximationmethod.
- A clock signal is required to control the A/D conversion which must have a frequency no lower than 750 KHz.
- Reference voltages are required for the conversion: one is high reference (V_{RH}) voltage, the other is low reference (V_{RL}) voltage. The difference between V_{RH} and V_{RL} cannot be lower than 2.5 V.
- The conversion is ratiometric. The input voltage V_{RL} converts to $00 and the input voltage V_{RH} converts to $FF.

The A/D Control/Status Register (ADCTL)

	7	6	5	4	3	2	1	0
	CCF	0	SCAN	MULT	CD	CC	CB	CA
value after reset	0	0	u	u	u	u	u	u

CCF: conversion complete flag. Set to 1 when all four A/D registers contains valid conversion results. This flag is set 129 clock cycles after this register is written into.
SCAN: continuous scan mode.
MULT: multiple-channel and single-channel control
CD - CA: channel select D to A

CD	CC	CB	CA	channel signal	result in ADRx if MULT = 1
0	0	0	0	AN0	ADR1
0	0	0	1	AN1	ADR2
0	0	1	0	AN2	ADR3
0	0	1	1	AN3	ADR4
0	1	0	0	AN4*	ADR1
0	1	0	1	AN5*	ADR2
0	1	1	0	AN6*	ADR3
0	1	1	1	AN7*	ADR4
1	0	0	0	reserved	ADR1
1	0	0	1	reserved	ADR2
1	0	1	0	reserved	ADR3
1	0	1	1	reserved	ADR4
1	1	0	0	V_{RH} pin**	ADR1
1	1	0	1	V_{RL} pin**	ADR2
1	1	1	0	$V_{RH}/2$**	ADR3
1	1	1	1	reserved	ADR4

* Not available in 48-pin package
** These channels are intended for factory testing

The OPTION Register

	7	6	5	4	3	2	1	0
	ADPU	**CSEL**	IREQ	DLY	CME	0	CR1	CR0
value after reset	0	0	0	1	0	0	0	0

ADPU: A/D power up. When set to 1, it enables the A/D converter. After setting this bit, the user must wait at least 100 μs before using the A/D converter.

CSEL: clock select. When set to 1, the RC clock signal is selected. Otherwise, the E clock is selected.

The Interpretation of Conversion Results

Let

x be the A/D conversion result
range = (V_{RH} - V_{RL})

A conversion result 255 corresponds to the voltage V_{RH}.
A conversion result 0 corresponds to the result V_{RL}.

Then the voltage corresponding to result x is

$$Vx = V_{RL} + (range \times x) \div 255$$

For example, if V_{RL} = 4 V and V_{RH} = 1 V, the A/D conversion results 10, 40, 60, 127, and 210 corresponds to the following voltages:

10	corresponds to	1 + (10 × 3) ÷ 255 = 1.12 V
40	corresponds to	1 + (40 × 3) ÷ 255 = 1.47 V
60	corresponds to	1 + (60 × 3) ÷ 255 = 1.71 V
127	corresponds to	1 + (127 × 3) ÷ 255 = 2.49 V
210	corresponds to	1 + (210 × 3) ÷ 255 = 3.47 V

The Procedure for Using the A/D Converter

Step 1. Convert the hardware properly. Scale and shift the analog inputs, when necessary, so that they fall between V_{RH} and V_{RL}.

Step 2. Set the ADPU bit of OPTION register to enable the A/D converter.

Step 3. Select the appropriate clock signal by setting or clearing the CSEL bit of the OPTION register.

Step 4. Wait for the A/D converter to stabilize.

Step 5. Select the appropriate channel(s) and operation modes by programming the ADCTL register.

Step 6. Wait until the CCF flag of the ADCTL register becomes 1 and collect the conversion results.

Data types

- Bit data
- 8-bit data signed unsigned
- 16bit unsigned fractions
- 16bit addresses
- A byte 8 bit wide
- A word 16 bit wide on two consecutive location with MSB at lower value address

Pulse Width Modulation Code

```
#include <hc11.h>
#pragma interrupt_handler int_handler
 //  directive required to define the interrupt service routine.
long int pulsewidth = 20000;
float dutycycle = 0.03;
long int onduty, offduty;
int on = 0;
 /* pulsewidth is the variable used for the total time period. Onduty is the
variable for on time and Offduty for off time. Onduty +Offduty = Pulsewidth. */

void int_handler(void)
{
   on = 1 - on;
  /* Next Interrupt */
  if(on == 1) {
    TOC2 += onduty;
  } else {
    TOC2 += offduty;
   }
/ * TOC2 is the Timer Output Compare 16 bit  register which is loaded with the
specified value. When the value of free running counter is equal to the value
stored in TOC2 register , the I/O pin OC2 (PA6) toggles.Also the interrupt
occurs setting the flag in TFLG1 register. * /
```

```
    /* Clear the flag OC2F */
      TFLG1 |= 0x40;

      /* Set PA4 bit of PORTA so we know we got an interrupt */
      PORTA |= 0x10;
    }
```

/ * The int_handler function is the Interrupt Service Routine which is called when the Timer output compare interrupt occurs * /

/ * When an interrupt occurs the interrupt flag in TFLG1 is automatically set which needs to be cleared to access another interrupt. * /

```
void main(void)
{
long int start;

onduty = (int)((double)pulsewidth * dutycycle);
offduty = pulsewidth - onduty;
```

/ * ONDUTY and OFFDUTY FORMULA * /

```
PORTA = 0;
/* Set OC2 to toggle mode after successful compare by setting the bits in TCTL1
register. */
TCTL1 = 0x40;
/* Read the counter value from TCNT 16 bit register.  */
start = TCNT;
/* We start after 1s */
TOC2 = start + 0x1000;
/* Clear the flag OC2F by setting the bits in TFLG1 register  */
TFLG1 |= 0x40;
/* Enable interrupt on OC2I  by setting the bits in TMSK1 register */
TMSK1 |= 0x40 ;
INTR_ON() ;
/ * This is to enable interrupt * /
while(1);
/ * infinite loop * /

}
#include "vectors.c"
/ *  interrupt vector addresses are defined in this file * /
```

TMSK 1 and TFLG1 registers

Address: $1022 - Timer Interrupt Mask 1 Register (TMSK1)

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OC1I	OC2I	OC3I	OC4I	OC5I	IC1I	IC2I	IC3I
Reset:	0	0	0	0	0	0	0	0

Figure 10-22. Output Capture Interrupt Enable Bits (OCxI)

Address: $1023 - Timer Interrupt Flag 1 Register (TFLG1)

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	OC1F	OC2F	OC3F	OC4F	OC5F	IC1F	IC2F	IC3F
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 10-23. Output Capture Flags (OCxF)

OCxI — Output Compare Interrupt Enable Bits (x = 1, 2, 3, 4, or 5)

OCxF — Output Compare Flags (x = 1, 2, 3, 4, or 5)

TCTL1 register

Address: $1020

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5
Reset:	0	0	0	0	0	0	0	0

Figure 10-25. Timer Control Register 1 (TCTL1)

OMx	OLx	Configuration
0	0	OCx does not affect pin (OC1 still may)
0	1	Toggle OCx pin on successful compare
1	0	Clear OCx pin on successful compare
1	1	Set OCx pin on successful compare

VECTORS.C FILE

The Vectors.C lists all the Interrupt Service Routines used by 68HC11

```
/ * As is, all interrupts except reset jumps to 0xffff, which is most
     * likely not going to useful. To replace an entry, declare your function,
     * and then change the corresponding entry in the table. For example,
     * if you have a SCI handler (which must be defined with
     * #pragma interrupt_handler ...) then in this file:
     * add
     *      extern void SCIHandler();
     * before the  table.
     * In the SCI entry, change:
     *      DUMMY_ENTRY,
     * to
     *  SCIHandler,
     */

    extern void _start (void);          /* entry point in crt??.s */
    #define DUMMY_ENTRY                 (void (*)(void))0xFFFF

    #pragma abs_address:0xffd6

    /* change the above address if your vector starts elsewhere
     */
```

```
void (*interrupt_vectors[])(void) =

        /*to cast a constant, say 0xb600, use   (void (*)())0xb600  */

    {   DUMMY_ENTRY,        /* SCI */
        DUMMY_ENTRY,        /* SPI */
        DUMMY_ENTRY,        /* PAIE */
        DUMMY_ENTRY,        /* PAO */
        DUMMY_ENTRY,        /* TOF */
        DUMMY_ENTRY,        /* TOC5 */
        DUMMY_ENTRY,        /* TOC4 */
        DUMMY_ENTRY,        /* TOC3 */
        int_handler,        /* TOC2 */
        DUMMY_ENTRY,        /* TOC1 */
        DUMMY_ENTRY,        /* TIC3 */
        DUMMY_ENTRY,        /* TIC2 */
        DUMMY_ENTRY,        /* TIC1 */
        DUMMY_ENTRY,        /* RTI */
        DUMMY_ENTRY,        /* IRQ */
        DUMMY_ENTRY,        /* XIRQ */
        DUMMY_ENTRY,        /* SWI */
        DUMMY_ENTRY,        /* ILLOP */
        DUMMY_ENTRY,        /* COP */
        DUMMY_ENTRY,        /* CLM */
        _start              /* RESET */

    } ;

#pragma end_abs_address
```

Analog to Digital Conversion

```
#include<stdio.h>
  /*  ICC11  header file for standard input and output , similar to one used in  C */
#include<hc11.h>
 /* ICC11 header file for using the registers of  68HC11. This file contains all the
      registers mapped with  their respective addresses. */
#include"print.c"
 /* User created C file, for outputting the data on serial port to be viewed on
      HyperTerminal, a serial port utility of Windows. This file consists of myprint
      function which is used to output the data.  */
      void main()
         {
             char ch;
             int res_1,res_2,res_3,res_4;
             unsigned int i; /* variables defined */

         setbaud(BAUD 9600) ;   // This function is used to set the baud rate of serial
                                                          communication.
             OPTION = 0X90;
```

What is the OPTION Register ?

It is a 8-bit special purpose register for setting internal system configuration during initialization. Bit 7 of the OPTION register is used to enable the A to D conversion.

```
ADCTL = 0X23;
```

The *ADCTL* register : This register is used to configure the Analog to Digital Converter.

Writing action to this register initiates the analog to digital conversion (Start of Conversion). End of analog to digital conversion is indicated by the CCF bit which gets set after an A/D conversion cycle.

```
while(ADCTL==0X23)
                    {}  // Wait for the CCF flag to set
                    res_1 = ADR1; // store the result from ADR1 result register to user variable
                    PORTB=res_1; // output the digital data on  I/O Port for D/A Conversion
                    myprint_int(res_1); // print the result on Hyper Terminal using the myprint
                                          function in print.c file

} // End of the code
```

ADCTL — A/D Control/Status $x**030**

	Bit 7	6	5	4	3	2	1	Bit 0
	CCF	0	SCAN	MULT	CD	CC	CB	CA
RESET:	I	0	I	I	I	I	I	I

I = Indeterminate value

CCF — Conversions Complete Flag
A read-only status indicator, this bit is set when all four A/D result registers contain valid conversion results. Each time the ADCTL register is overwritten, this bit is automatically cleared to zero and a conversion sequence is started. In the continuous mode, CCF is set at the end of the first conversion sequence.

Bit 6 — Not implemented. Reads always return zero and writes have no effect.

SCAN — Continuous Scan Control
0 = Do four conversions and stop
1 = Convert four channels in selected group continuously

MULT — Multiple Channel/Single Channel Control
0 = Convert single channel selected
1 = Convert four channels in selected group

CD–CA — Channel Select D through A
Refer to **Table 24**. When a multiple channel mode is selected (MULT = 1), the two least significant channel select bits (CB and CA) have no meaning and the CD and CC bits specify which group of four channels is to be converted.

Table 24 A/D Converter Channel Assignments

Channel Select Control Bits CD:CC:CB:CA	Channel Signal	Result in ADRx if MULT = 1
0000	AN0	ADR1
0001	AN1	ADR2
0010	AN2	ADR3
0011	AN3	ADR4
0100	AN4	ADR1
0101	AN5	ADR2
0110	AN6	ADR3
0111	AN7	ADR4
10XX	Reserved	ADR1–ADR4
1100	V_{RH}[1]	ADR1
1101	V_{RL}[1]	ADR2
1110	$(V_{RH})/2$[1]	ADR3
1111	Reserved[1]	ADR4

OPTION — System Configuration Options **$x039**

	Bit 7	6	5	4	3	2	1	Bit 0
	ADPU	CSEL	IRQE*	DLY*	CME	FCME*	CR1*	CR0*
RESET:	0	0	0	1	0	0	0	0

*Can be written only once in first 64 cycles out of reset in normal modes, or at any time in special modes.

ADPU — A/D Power-Up

This bit is implemented on the MC68HC11F1 only. On the MC68HC11FC0, reads always return zero and writes have no effect.

0 = A/D system disabled
1 = A/D system enabled

CSEL — Clock Select

This bit is implemented on the MC68HC11F1 only. On the MC68HC11FC0, reads always return zero and writes have no effect.

0 = A/D and EEPROM use system E clock
1 = A/D and EEPROM use internal RC clock

IRQE — $\overline{\text{IRQ}}$ Select Edge Sensitive Only

0 = Low level recognition
1 = Falling edge recognition

DLY — Enable Oscillator Start-Up Delay on Exit from STOP

0 = No stabilization delay on exit from STOP
1 = Stabilization delay of 4064 E-clock cycles is enabled on exit from STOP

CME — Clock Monitor Enable

0 = Clock monitor disabled; slow clocks can be used
1 = Slow or stopped clocks cause clock failure reset

FCME — Force Clock Monitor Enable

0 = Clock monitor circuit follows the state of the CME bit
1 = Clock monitor circuit is enabled until the next reset

In order to use both STOP and the clock monitor, the CME bit should be written to zero prior to executing a STOP instruction and rewritten to one after recovery from STOP. FCME should be kept cleared if the user intends to use the STOP instruction.

CR[1:0] — COP Timer Rate Select