# RC 4000 GENERAL TECHNICAL DESCRIPTION

## CENTRAL PROCESSOR

Chapter 5. INTERRUPT UNIT.



RCSL51 - VB237 June 1968/MSP
pp. 5-1:5-7

Chapter 5

# I N T E R R U P T  U N I T

## 5.1. INTRODUCTION.

The program interrupt system permits an automatic switching from the current sequence of instructions to another sequence in immediate response to specific internal or external events.

The interrupt system is realized by the Interrupt Unit and a microprogram routine. The following program instructions have special relation to the interrupt system.

| | |
|---|---|
| Clear Interrupt Register | (IC) |
| Store Interrupt Register | (IS) |
| Load Mask Register | (ML) |
| Store Mask Register | (MS) |
| Jump with Interrupt Enabled | (JE) |
| Jump with Interrupt Disabled | (JD) |

The effect of these instructions is described in the RC 4000 Reference Manual.

The interrupt system performs the following functions: (1) Collection of interrupt signals, (2) Interrogation of interrupt signals, (3) Selection among competitive interrupt requests, (4) Saving of return information, and (5) Branching to the interrupt program.

The RC 4000 can collect up to 24 interrupt signals in the interrupt register, IR(0:23). The monitor has selective control over these interrupt lines by means of the mask register, IM(0:23). For each of the 24 interrupt lines a mask register bit defines whether an interrupt request will be honoured (mask bit = 1) or ignored (mask bit = 0).

The signal ITR Request is generated if IR(n) $\wedge$ IM(n) = 1 for any n. The ITR Request signal is interrogated once in every instruction cycle. In case of interrupt, the contents of the instruction counter (IC) will

be stored in storage word 10, before branching to an address kept in storage word 12.

The problem of simultaneous interrupt signals is solved by means of a priority logic, which selects the left-most signal first. The interrupt signals are numbered from 0 to 23 after their position in the register. The interrupt number (ITRno) is stored in storage word 8 as a word address, i.e. with the unit position in bit 22 and with bit 23 equal to zero. The interruption program uses this interrupt number to branch to a specific service routine. When the interrupt number has been transferred to SB, the interrupt bit in question is reset.

Only the instruction counter is stored as return information about the interrupted program. The interruption program is responsible for saving and restoring the contents of the W registers and the EX register.

The entire interruption system can be disabled for short intervals when an interruption would be awkward (e.g. while a previous interrupt is being processed). The disabling and enabling is performed by the privileged instructions Jump with Interrupt Disabled (JD) and Jump with Interrupt Enabled (JE). When the system is disabled, interrupt signals are still collected in the IR register but not interrogated. The system is automatically disabled when the interruption routine in the microprogram is entered. It is enabled again when the first JE instruction is executed, that is when the necessary interrupt administration has been finished.

It is possible to cancel an interrupt signal before it will give rise to program interruption. This can be done by resetting the interrupt bit in question during execution of the instruction Clear Interrupt Register (IC).

The interrupt signals can be classified according to priority as follows:

| | |
|---|---|
| IR(0) | Instruction Exception |
| IR(1) | Integer Overflow |
| IR(2) | Floating-Point Overflow |
| IR(3:23) | External Interruption |

A description of the above-mentioned interrupt situations can be found in the RC 4000 Reference Manual. The interrupt system can be de-

livered in a limited version. The actual installation is equipped with 8 interrupt channels (IR(0:7)). The 8 channels are used as follows:

| | |
|---|---|
| IR(0) | Instruction Exception |
| IR(1) | Integer Overflow |
| IR(2) | = 0 |
| IR(3) | Clock pulse each 1 second |
| IR(4) | Clock pulse each 20 milliseconds |
| IR(5:7) | = 0 |

## 5.2. INTERRUPT BUFFER, IB.

[ITR - 07,08]

The interrupt input signals are generated asynchronously in proportion to the Interrupt Unit. To obtain the necessary synchronization the interrupt signals are buffered in the Interrupt Buffer Register (IB).

Between the interrupt input terminal and IB is placed a trigger circuit so that IB is set on the leading edge of the interrupt signal. The trigger circuit utilizes the signal delay through the chain of 2-input gates.

IB is reset when the interrupt signal is transferred to IR and gated by the clock pulse Time430t030.

## 5.3. INTERRUPT REGISTER, IR.

[ITR - 01,02]

IR stores the interrupt signal until it is processed or until it is reset by an IC instruction.

IR(n) is set during the clock pulse Time280t380 if IB(n) = 1. IR(n) is reset in the microprogram interrupt response routine if the interrupt in question is selected. This resetting is controlled by the microinstruction GiBUSfITRno which is the BUS gate signal for ITRno (interrupt number).

## 5.4. INTERRUPT MASK, IM.

[ITR - 13,14]

IM is loaded during execution of the program instruction ML (Load Mask), and gated by the microinstruction GiIMfSB.

If IM(n) = 0, the contents of IR(n) will be ignored, i.e. IR(n) = 1 cannot give rise to a program interruption. IR(n) can, however, still be read and processed by means of the program instruction IS (Store Interrupt).

## 5.5. PROGRAM DESCRIPTION.

This section describes the complete 24-bit version of the interrupt system.

```
begin
  register IB(0:23), IR(0:23), IM(0:23);
  comment The priority logic is realized by a combinational network,
    based upon a division of the 24-bits into 6 groups each of 4 bits,
    identified as Group(0), Group(2), ..., Group(5). The selected group
    is called GrSl;
  comb net PR(0:23) = IR(0:23) ∧ 1conIM(1:23);
  comb net Group(0:0) = or PR(0:3),
           Group(1:1) = or PR(4:7),
           Group(2:2) = or PR(8:11),
           Group(3:3) = or PR(12:15),
           Group(4:4) = or PR(16:19),
           Group(5:5) = or PR(20:23),

           GrSl(0:0) = Group(0),
           GrSl(1:1) = -,Group(0) ∧ Group(1),
           GrSl(2:2) = -,Group(0) ∧ Group(1) ∧ Group(2),
           GrSl(3:3) = -,Group(0) ∧ -,Group(1) ∧ -,Group(2) ∧ Group(3),
           GrSl(4:4) = -,Group(0) ∧ -,Group(1) ∧ -,Group(2) ∧ -,Group(3)
                                                          ∧ Group(4),
           GrSl(5:5) = -,Group(0) ∧ -,Group(1) ∧ -,Group(2) ∧ -,Group(3)
                                                 ∧ -,Group(4) ∧ Group(5);
```

comment The interrupt number ITRno is then decoded in accordance with the following tables:

| Group(0) | Group(1) | Group(2) | Group(3) | Group(4) | Group(5) | Selected Group | ITRno(18:20) |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | | ϕ ϕ ϕ |
| 0 | 0 | 0 | 0 | 0 | 1 | GrSl(5) | 1 0 1 |
| 0 | 0 | 0 | 0 | 1 | ϕ | GrSl(4) | 1 0 0 |
| 0 | 0 | 0 | 1 | ϕ | ϕ | GrSl(3) | 0 1 1 |
| 0 | 0 | 1 | ϕ | ϕ | ϕ | GrSl(2) | 0 1 0 |
| 0 | 1 | ϕ | ϕ | ϕ | ϕ | GrSl(1) | 0 0 1 |
| 1 | ϕ | ϕ | ϕ | ϕ | ϕ | GrSl(0) | 0 0 0 |

Bit number in group:

| (0) | (1) | (2) | (3) | ITRno(21,22) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | ϕ ϕ |
| 0 | 0 | 0 | 1 | 1 1 |
| 0 | 0 | 1 | ϕ | 1 0 |
| 0 | 1 | ϕ | ϕ | 0 1 |
| 1 | ϕ | ϕ | ϕ | 0 0 |

```
comb net
  ITRno(18:18) = or GrSl(4,5),
  ITRno(19:19) = or Grl(2,3),
  ITRno(20:20) = or GrSl(1,3,5),
  ITRno(21:21) = GrSl(0) ∧ PR(0,1) = 0 ∨ GrSl(1) ∧ PR(4,5) = 0
               ∨ GrSl(2) ∧ PR(8,9) = 0 ∨ GrSl(3) ∧ PR(12,13) = 0
               ∨ GrSl(4) ∧ PR(16,17) = 0 ∨ GrSl(5) ∧ PR(20,21) =0,
  ITRno(22:22) = GrSl(0) ∧ -,PR(0) ∧ (PR(1) ∨ -,PR(2))
               ∨ GrSl(1) ∧ -,PR(4) ∧ (PR(5) ∨ -,PR(6))
               ∨ GrSl(2) ∧ -,PR(8) ∧ (PR(9) ∨ -,PR(10))
               ∨ GrSl(3) ∧ -,PR(12) ∧ (PR(13) ∨ -,PR(14))
               ∨ GrSl(4) ∧ -,PR(16) ∧ (PR(17) ∨ -,PR(18))
               ∨ GrSl(5) ∧ -,PR(20) ∧ (PR(21) ∨ -,PR(22));
comment The signal NOinGr (Number in group) denotes the bit number in the
  selected group. NOinGr and GrSl determines which IR bit is to be reset;
comb net
  NOinGr(0:3) = ITRno(21,22) = b00 con ITRno(21,22) = b01
              con ITRno(21,22) = b10 con ITRno(21,22) = b11;
comment ITR Request is used (together with ITR Enable) as a jump condi-
  tion for the microprogram;
comb net ITR Request(0:0) = or Group(0:5);
end;
```

## 5.6. TIMER.

[ITR - 30]

The interrupt channels 3 and 4 are connected to a Timer, which generates two pulses, one for every 1 second and one for every 20 milliseconds.

The Timer uses the mains frequency (50 cycles) as basic frequency. A 220 volts, 50 cycles voltage from the motor generator is transformed to a 6 volts signal. This signal is then converted in the pulseshaper PS301 and used as 20 milliseconds interrupt signal on channel number 4.

The 20 milliseconds pulses are used as counting pulses to a binary counter, which counts modulo 50. The counter generates an output signal after 50 counts, i.e. every 1 second. The signal is used as interrupt signal on channel number 3.