RCSL: 51-VB527
Author: M. Strange
Edited: September 1969

INTERRUPT UNIT FOR

THE RC 4000 COMPUTER

A/S REGNECENTRALEN
Falkoneralle 1
2000 Copenhagen F

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.

2. 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, identifi-
    ed as Group(0), Group(2), ..., Group(5). The selected group is called
    GrSl;
  comb net RM(0:23) = IR(0:23) ∧ 1conIM(1:23);
  comb net Group(0:0) = or RM(0:3),
           Group(1:1) = or RM(4:7),
           Group(2:2) = or RM(8:11),
           Group(3:3) = or RM(12:15),
           Group(4:4) = or RM(16:19),
           Group(5:5) = or RM(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 GrSl(2,3),
    ITRno(20:20) = or GrSl(1,3,5),
    ITRno(21:21) = GrSl(0) ∧ RM(0,1) = 0 ∨ GrSl(1) ∧ RM(4,5) = 0
                 ∨ GrSl(2) ∧ RM(8,9) = 0 ∨ GrSl(3) ∧ RM(12,13) = 0
                 ∨ GrSl(4) ∧ RM(16,17) = 0 ∨ GrSl(5) ∧ RM(20,21) = 0,
    ITRno(22:22) = GrSl(0) ∧ -,RM(0) ∧ (RM(1) ∨ -,RM(2))
                 ∨ GrSl(1) ∧ -,RM(4) ∧ (RM(5) ∨ -,RM(6))
                 ∨ GrSl(2) ∧ -,RM(8) ∧ (RM(9) ∨ -,RM(10))
                 ∨ GrSl(3) ∧ -,RM(12) ∧ (RM(13) ∨ -,RM(14))
                 ∨ GrSl(4) ∧ -,RM(16) ∧ (RM(17) ∨ -,RM(18))
                 ∨ GrSl(5) ∧ -,RM(20) ∧ (RM(21) ∨ -,RM(22));

  comment The signal NOinGr (Number in group) denotes the bit number in the
    selected group. NOinGr and GrSl determine 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 condition
    for the microprogram;
  comb net ITR Request(0:0) = or Group(0:5);
end;
```