D950-CORE

Preliminary Specification

SGS-THOMSON MICROELECTRONICS

January 1995

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

SGS-THOMSON MICROELECTRONICS

D950-CORE

16-Bit Fixed Point Digital Signal Processor (DSP) Core

PRELIMINARY DATA

The D950-CORE is a general purpose programmable 16-bit fixed point Digital Signal Processor (DSP) Core.

The D950-CORE can be embedded as a megafunction in a gate array or cell-based device through the combination of the DSP core, the digital cell 0.5µ library and the module generators for RAM and ROM

Simply by adding peripherals and memories for data and instruction, a powerful Application Specific Digital Signal Processor (AS-DSP) can be easily built around the D950-CORE to your application's needs.

Another key to the high performance of the D950-CORE is the high degree of parallelism which allows the D950-CORE to perform simultaneously numerous functions that require several machine cycles with other processors.

- **40 Mips - 25nsec instruction cycle time**
- **HARVARD architecture**
- **Two 64k x 16-bit data memory spaces**
- **One 64k x 16-bit program memory space**
- **16 x 16-bit parallel multiplier**
- **40-bit barrel shifter unit**
- **40-bit ALU (40-bit for arithmetical operations)**
- **Two 40-bit extended precision accumulators**
- **Fractional and integer arithmetic with support for floating point and multi-precision**
- **Bit manipulation for MCU functions**
- **Immediate and computed branchs and subroutine calls**
- **No overhead nested hardware loops**
- **One interrupt pin and interface to an interrupt controller peripheral**
- **Bootstrap loading capability**
- **Stack in data memory space**
- **Two address calculation units with modulo and bit-reverse capability**
- **General purpose 8-bit I/O port**
- **Dedicated hardware for Emulation and Test, IEEE 1149.1 (JTAG) compatible**
- **Co-processor interface and co-processor dedicated instructions**
- **Operating frequency down to DC**
- **Single 3.3V power supply**
- **Low-power standby mode**
- **HCMOS5 0.5 µ (triple-level-metal) technology**

Built in ”state-of-the-art” 0.5µm triple-level-metal CMOS technology, the D950-CORE includes an arithmetic Data Calculation Unit (DCU), a Program Control Unit (PCU) and an Address Calculation Unit (ACU) able to manage up to 64k (program) and 128k (data) x 16-bit memory spaces. An Emulation and Test Unit (ETU) allows easy implementation and test for AS-DSP devices.

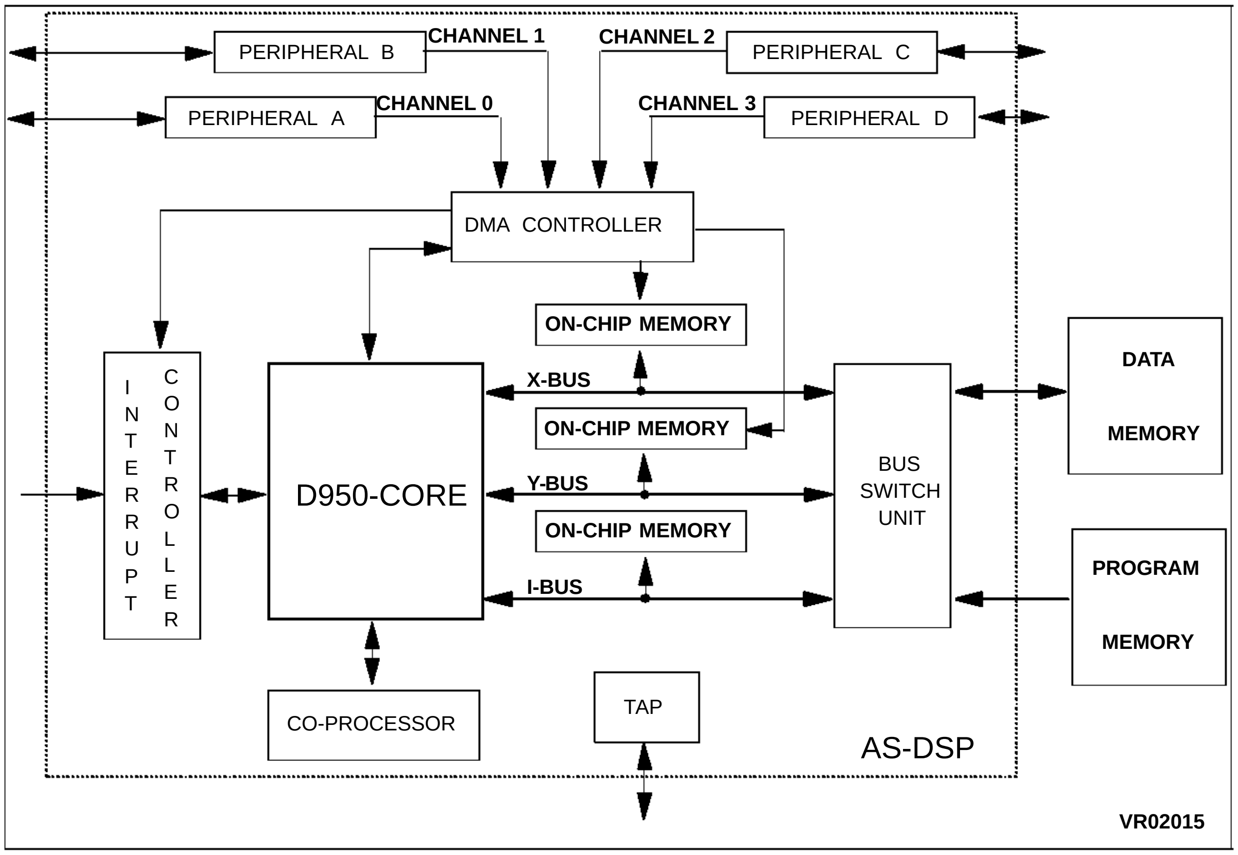
In addition to the preceeding features, a set of high level development tools (hardware and software) and a complete design package give substantial advantages to users in the form of a performant design environment, rapid prototyping, first time silicon success and built-in test strategies for a global solution in AS-DSP development :

- **JTAG PC board with graphic windowed high level source debugger for AS-DSP emulation,**
- **Complete crash-barrier chain (assembler / simulator / linker) running on PC and SUN,**
- **Complete GNU chain (assembler / simulator / linker / C compiler / C debugger) running on SUN,**
- **VHDL model (SYNOPSYS),**
- **Macrocells dedicated to the DSP Core to build user’s applications,**
- **Standard cells library, I/O library and Data-Path generator,**
- **Memory generators (RAM, ROM).**

1 INTRODUCTION

This data sheet is related to the D950-CORE. Some typical dedicated peripherals developped in the friendly SGS-THOMSON 0.5µ library are described in the annex. Simply by adding these peripherals (and/or others) and memory for data and instruction, a powerful AS-DSP can be easily built around the D950-CORE to your application's needs (see figure).

Figure 1. AS-DSP architecture example.



2 PIN DESCRIPTION

2.1 DATA BUSES (70 PINS)

Pin Name	Type	Description
XD0-XD15	**I/O**	**X Data Bus. Hi-Z during cycles with no X-bus exchange.**
XA0-XA15	**O**	**X Address bus. Hi-Z when in Hold.**
$\overline{\textbf{XRD}}$	**O**	**X-bus read strobe. Active low. Hi-Z when in Hold.**
$\overline{\textbf{XWR}}$	**O**	**X-bus write strobe. Active low. Hi-Z when in Hold.**
$\overline{\textbf{XBS}}$	**O**	**X-bus strobe. Active low. Hi-Z when in Hold. Asserted low at the beginning of a valid X-bus cycle.**
YD0-YD15	**I/O**	**Y Data Bus. Hi-Z during cycles with no Y-bus exchange.**
YA0-YA15	**O**	**X Address bus. Hi-Z when in Hold.**
$\overline{\textbf{YRD}}$	**O**	**Y-bus read strobe. Active low. Hi-Z when in Hold.**
$\overline{\textbf{YWR}}$	**O**	**Y-bus write strobe. Active low. Hi-Z when in Hold.**
$\overline{\textbf{YBS}}$	**O**	**Y-bus strobe. Active low. Hi-Z when in Hold. Asserted low at the beginning of a valid Y-bus cycle.**

2.2 PROGRAM BUS (35 PINS)

Pin Name	Type	Description
ID0-ID15	**I/O**	**Instruction data bus.** **Hi-Z during cycles with no I-bus exchange.**
IA0-IA15	**O**	**Instruction address bus.** **Hi-Z when in Hold.**
$\overline{\text{IRD}}$	**O**	**I-bus read strobe. Active low.** **Hi-Z when in Hold.**
$\overline{\text{IWR}}$	**O**	**I-bus write strobe. Active low.** **Hi-Z when in Hold.**
$\overline{\text{IBS}}$	**O**	**I-bus strobe. Active low.** **High level in HoldAsserted low at the beginning of a valid I-bus cycle.**

2.3 BUS CONTROL (3 PINS)

Pin Name	Type	Description
$\overline{\text{DTACK}}$	**I**	**Data transfer acknowledge. Active low.** **Sampled on CLKOUTrising edge.** **Controls bus cycle extension by insertion of wait-states.**
$\overline{\text{HOLD}}$	**I**	**Hold bus request signal. Active low.** **Asserted by a peripheral (DMA controller) requiring bus mastership.** **Halts program execution and tri-states buses.**
$\overline{\text{HOLD ACK}}$	**O**	**Hold Acknowledge output. Active low.** **Indicates that all buses are in Hi-Z.**

2.4 GENERAL PURPOSE P-PORT (8 PINS)

Pin Name	Type	Description
P0-P7	**I/O**	**8-bit bidirectional parallel port.** **Each pin can be individually programmed as input or output and as level or falling edge sensitive input conditions for test by branch and conditional instructions.**

2.5 CLOCK (2 PINS)

Pin Name	Type	Description
CLKIN	**I**	**Clock input.**
CLKOUT	**O**	**Clock output. Period is CLKIN (half instruction cycle).**

2.6 CONTROL (10 PINS)

Pin Name	Type	Description
$\overline{\text{IT}}$	**I**	**Maskable Interrupt Request Input. Falling edge sensitive.**
$\overline{\text{ITACK}}$	**O**	**Maskable Interrupt Request Acknowledge. Active Low. Asserted low at the beginning of Interrupt servicing.**
$\overline{\text{EOI}}$	**O**	**End of maskable Interrupt routine output. Active low. Asserted low at the end of current interrupt request processing.**
$\overline{\text{LP}}$	**I**	**Low power. Falling edge sensitive. Stops the processor after execution of the currently decoded instruction and enters low-power standby state (in this state, the clock generator is stopped).**
$\overline{\text{LPACK}}$	**O**	**Low power Acknowledge. Active low. Asserted low at the end of execution of the last instruction following detection of $\overline{\text{LP}}$ falling edge or at the end of a LPSTOP instruction.**
$\overline{\text{RESET}}$	**I**	**Reset input. Active low. Initializes the processor to the RESET state and the clock generator. Forces Program Counter value to reset address and execution of NOP instruction.**
MODE	**I**	**Mode input select. Forces reset address to 0x0000 (resp. 0xFC00) when low (resp. high).**
VCI	**O**	**Valid co-processor instruction decoded. Asserted high while decoding a co-processor dedicated instruction. Indicates that the co-processor instruction will be executed at the following instruction cycle.**
IRD_WR	**O**	**Indicates program memory RD/WR cycle during execution of Read or Write Program memory instruction.**
INCYCLE	**O**	**Instruction cycle. Asserted high at the beginning of cycle.**

2.7 EMULATION (7 PINS)

Pin Name	Type	Description
$\overline{\text{ERQ}}$	**I**	**Emulator Halt Request. Active low.** **Halts program execution and enters emulation mode.**
IDLE	**O**	**Output flag indicating if the processor is halted or executing an instruction in Emulation mode.**
HALTACK	**O**	**Halt Acknowledge. Active high.** **Asserted high when the processor is halted and under control of the emulator.**
SNAP	**O**	**Snapshot output. Active high** **Asserted high when executing an instruction in Snapshot mode.**
AXEBP	**I**	**When high, enables breakpoint on X address bus.**
AYEBP	**I**	**When high, enables breakpoint on Y address bus.**
AIEBP	**I**	**When high, enables breakpoint on I address bus.**

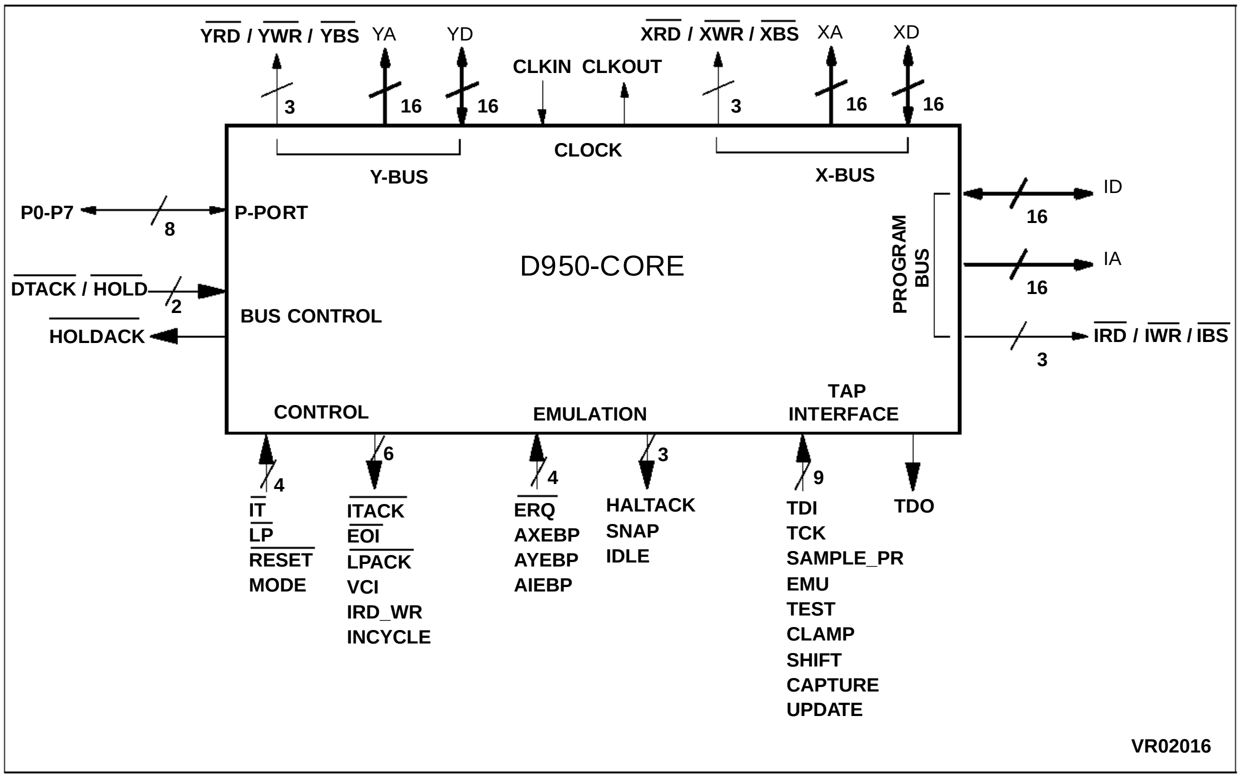
2.8 TAP CONTROLLER INTERFACE (10 PINS)

Pin Name	Type	Description
TDI	**I**	**JTAG input TDI.**
TCK	**I**	**JTAG input TCK.**
TDO	**O**	**Processor Scan output to JTAG controller.**
SAMPLE_PR	**I**	**Dedicated TAP instruction. Sample Preload D950-CORE Scan register.**
EMU	**I**	**Dedicated TAP instruction. Emulation D950-CORE.**
TEST	**I**	**Dedicated TAP instruction. Test D950-CORE.**
CLAMP	**I**	**TAP controller state decode. Clamp.**
SHIFT	**I**	**TAP controller state decode. Shift Data register.**
CAPTURE	**I**	**TAP controller state decode. Capture Data register.**
UPDATE	**I**	**TAP controller state decode. Update Data register.**

2.9 SUPPLY (2 PINS)

Pin Name	Type	Description
VDD	**I**	**Positive Supply.**
GND	**I**	**Ground pin.**

Figure 2. D950-CORE I/O Interface.



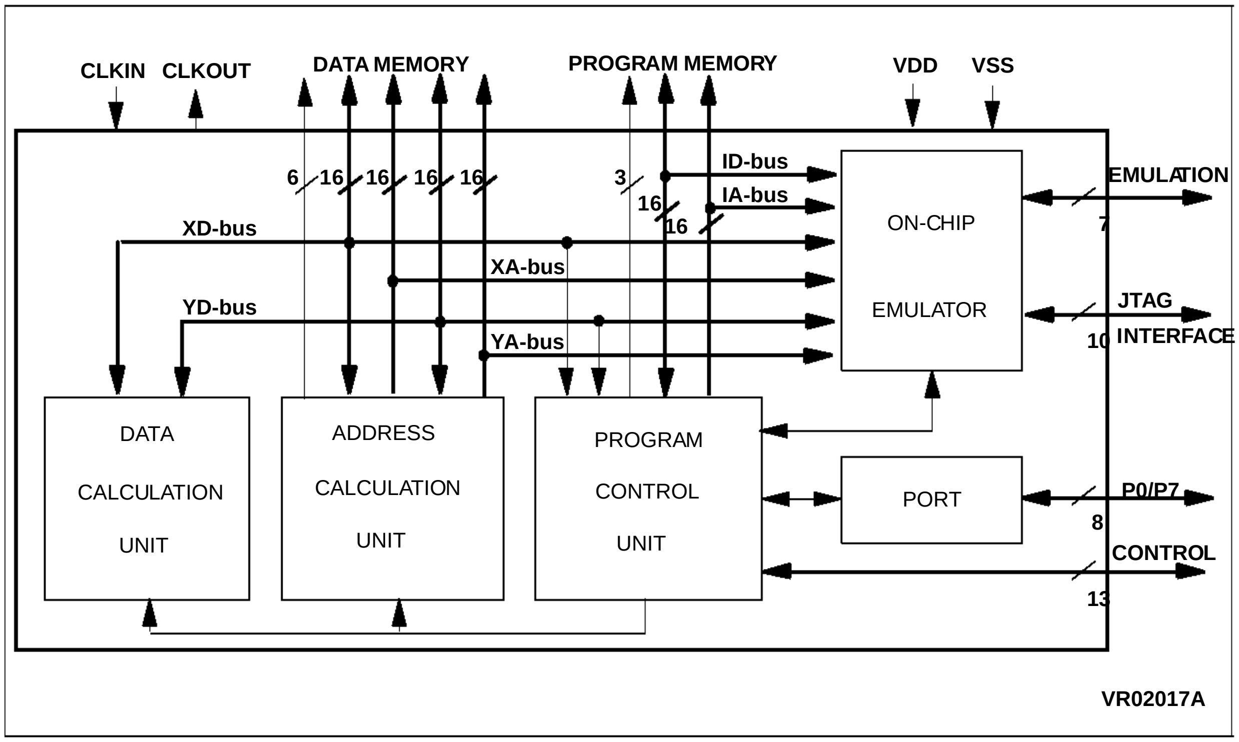
3 HARDWARE ARCHITECTURE

3.1 FUNCTIONAL OVERVIEW

The D950-CORE is composed of four main units (see figure) :

. Data Calculation Unit (DCU),
. Address Calculation Unit (ACU),
. Program Control Unit (PCU),
. Emulation and Test Unit (ETU).

Figure 3. D950-CORE Block Diagram



These units are organized in an HARVARD architecture around three bidirectional 16-bit buses

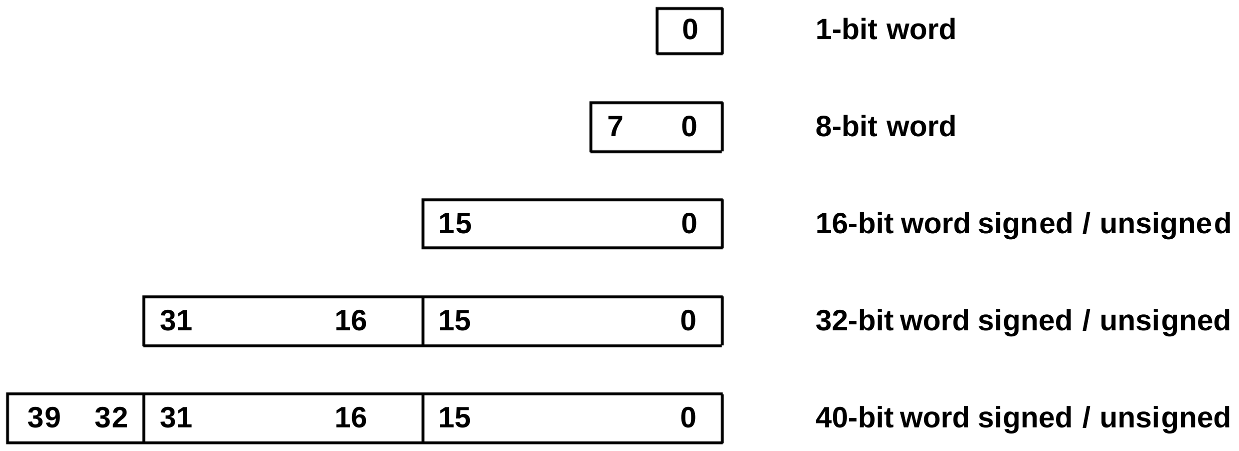
. two for data (XD / YD),
. one for instruction (ID),

each of these buses being dedicated to an unidirectional 16-bit address bus (XA / YA / IA).

Data buses (XD / YD and XA / YA) are provided externally. Data memories (RAM, ROM) and peripherals registers are to be mapped in these address spaces.
Instruction bus (ID / IA) gives access to Program memory (RAM, ROM).
Data and Instruction buses share the same bus control interface ($\overline{\text{RD}}$ /$\overline{\text{WR}}$ /$\overline{\text{BS}}$).

Data / Instruction Buses			Corresponding Address Bus		
XD	**Bidirectional**	**16-bit**	**XA**	**Unidirectional**	**16-bit**
YD	**Bidirectional**	**16-bit**	**YA**	**Unidirectional**	**16-bit**
ID	**Bidirectional**	**16-bit**	**IA**	**Unidirectional**	**16-bit**

Depending on the calculation mode, the D950-CORE DCU computes operands which can be considered as 16 or 32-bit, signed or unsigned. It includes a 16 x 16-bit parallel multiplier able to implement MAC-based functions in one cycle per MAC. A 40-bit Arithmetic and Logic Unit, including a 8-bit extension for arithmetic operations, implements a wide range of arithmetic and logic functions. A 40-bit barrel shifter unit is also included as well as a bit manipulation unit able to handle MCU processing through bit operations.
Different types of word lengths are available for manipulation.


0
1-bit word
7 0
8-bit word
15 0
16-bit word signed / unsigned
31 16 15 0
32-bit word signed / unsigned
39 32 31 16 15 0
40-bit word signed / unsigned

Different formats of word are available for manipulation.

Format		Minimum	Maximum
fractional	**signed**	**- 1**	**+ 0.999969481**
	unsigned	**0**	**+ 1.99996948**
integer	**signed**	**- 32768**	**+ 32767**
	unsigned	**0**	**+ 65535**

The D950-CORE ACU generates an address for each of the two identical Data memory spaces and updates them at each instruction, allowing execution of instructions performing up to two register-to-memory moves in one cycle. Various addressing modes including indirect linear and modulo addressing, immediate and stack pointer relative are implemented.

The D950-CORE PCU updates the Program Counter (PC) according to the current instruction and/or internal and external events. It performs program address generation, instruction fetch and decoding, exception processing and hardware loop control. By default, the PC is incremented by 1.

The D950-CORE ETU contains three independent parts sharing the same external interface : Emulation part, Core Scan Registers (CSR), and Test part (for production test purpose only). Access to these units is performed through dedicated I/O pins allowing the ETU to interface with an outside JTAG TAP controller or as primary accesses of the final AS-DSP chip, according to the IEEE 1149.1 JTAG standard.

An 8-bit general purpose parallel port (P0-P7) can be also configured (input or output). A test condition is attached to each bit to test external events.

Control of the chip is performed through interface pins related to interrupt, low-power mode, reset and miscellaneous functions.

Clock input is provided on the CLKIN pin which is buffered to the CLKOUT pin.

3.2 BLOCK DESCRIPTION

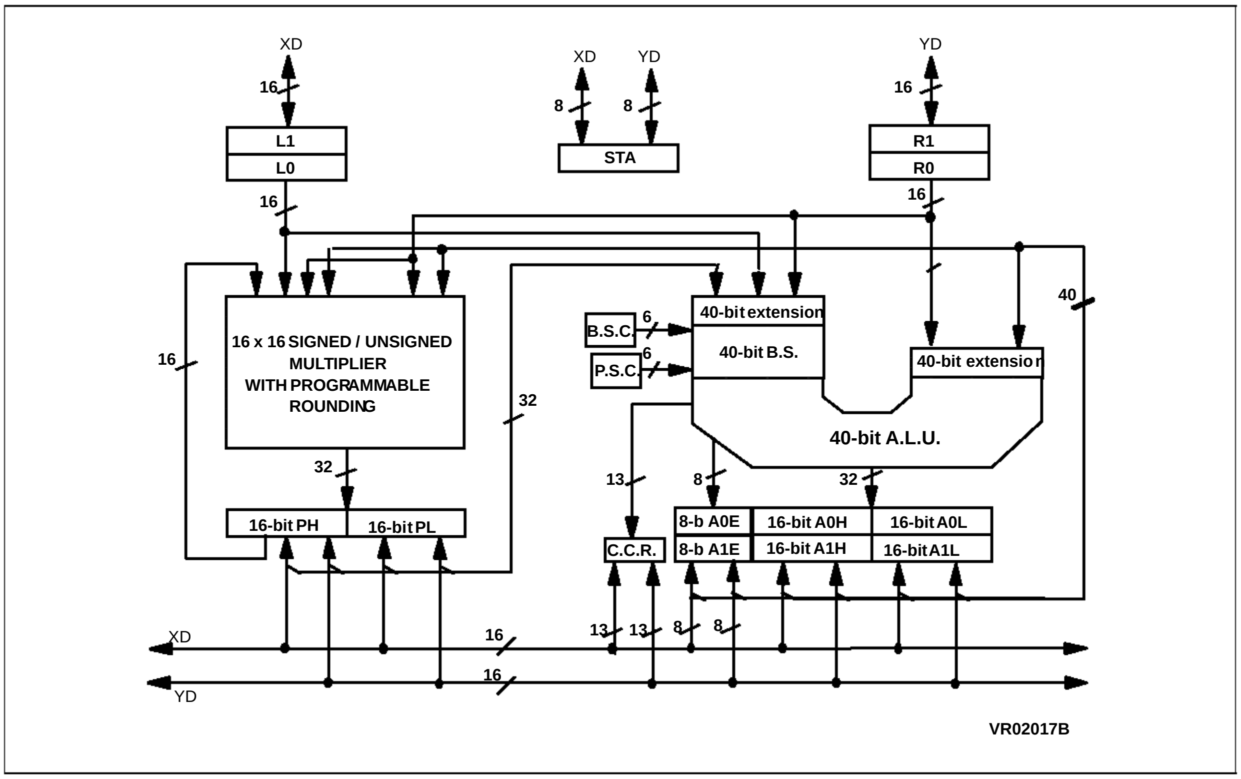
3.2.1 Data Calculation Unit (DCU)

3.2.1.1 Description

The D950-CORE DCU includes (see figure) the following main components :

- **a 16 x 16-bit signed/unsigned fractional/integer parallel multiplier,**
- **a 40-bit Barrel Shifter Unit (BSU) with a maximum right or left shift value of 32,**
- **a 40-bit Arithmetic and Logic Unit (ALU) implementing a wide range of arithmetic and logic functions with an 8-bit extension for arithmetic operations,**
- **a 16-bit Bit Manipulation Unit (BMU) implementing bit operations on internal registers and/or on 16-bit data RAM with an 8/16-bit mask,**
- **a set of working registers composing the register file,**
- **8 bits of STA register for operating modes,**
- **13 bits of CCR for ALU flags,**
- **a 6-bit Barrel Shifter Control register (BSC), a 6-bit Product Shift Control register (PSC) dedicated to MAC operations.**

Figure 4. D950-CORE Data Calculation Unit

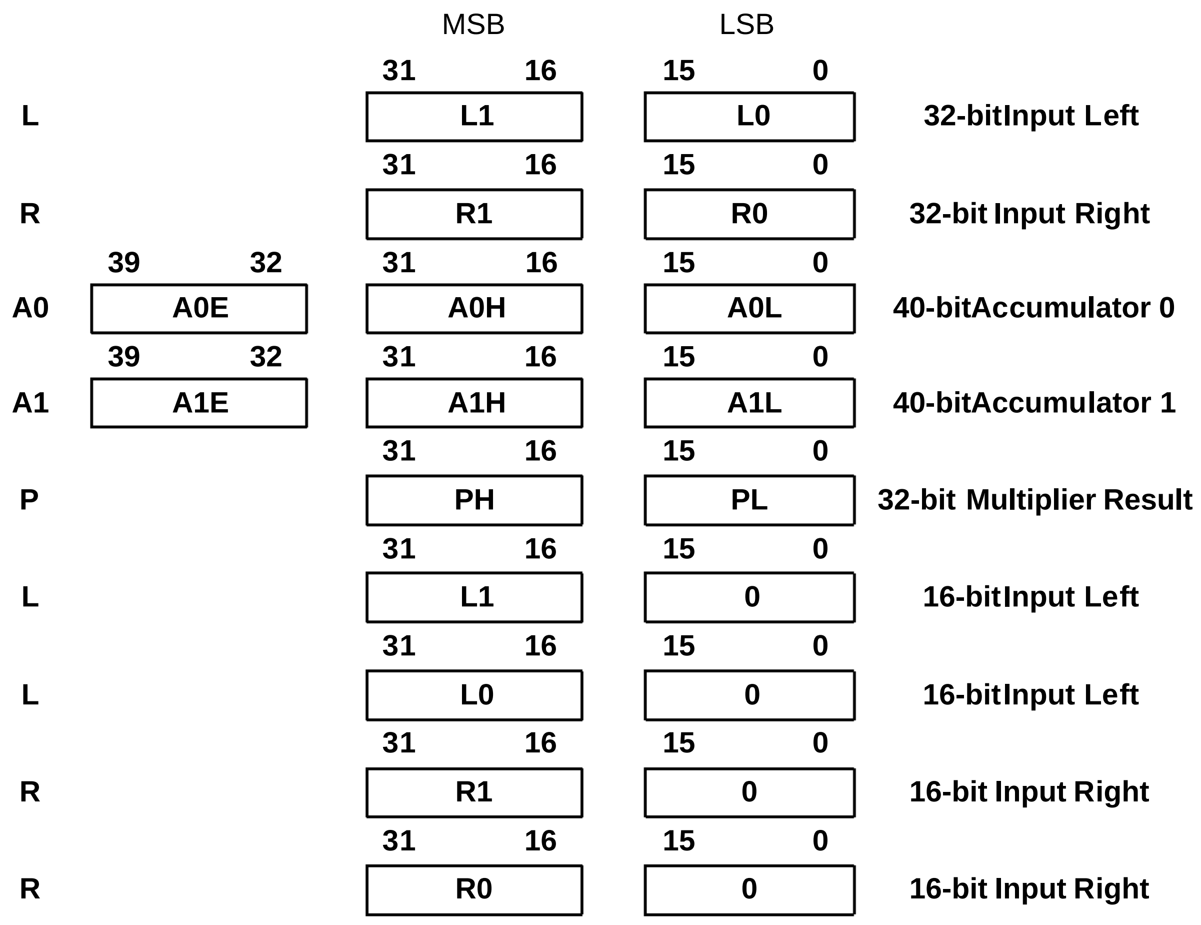


3.2.1.2 Registers

The D950-CORE DCU includes different registers which can be read / written through the X and Y buses.
All the DCU parts (Multiplier, BSU, ALU, BMU) operate on these registers.
These registers are of two types and are all direct addressed :

. Data registers :

- **L0 / L1 : 2 x 16-bit input Left registers,**
- **R0 / R1 : 2 x 16-bit input Right registers,**
- **A0 / A1 : 2 x 40-bit Accumulators, each made of the concatenation of an 8-bit extension A0E (resp. A1E), a 16-bit MSB A0H (resp. A1H) and a 16-bit LSB A0L (resp. A1L) registers dedicated to extended precision calculations in order to provide up to 240 dB of dynamic range,**
- **P : 32-bit multiplier result register made of the concatenation of PH (MSB) and PL (LSB) 16-bit registers.**


MSB
LSB
31 16 15 0
L L1 L0 32-bit Input Left
31 16 15 0
R R1 R0 32-bit Input Right
39 32 31 16 15 0
A0 A0E A0H A0L 40-bit Accumulator 0
39 32 31 16 15 0
A1 A1E A1H A1L 40-bit Accumulator 1
31 16 15 0
P PH PL 32-bit Multiplier Result
31 16 15 0
L L1 0 16-bit Input Left
31 16 15 0
L L0 0 16-bit Input Left
31 16 15 0
R R1 0 16-bit Input Right
31 16 15 0
R R0 0 16-bit Input Right

. Control registers :

- STA : Bits 0 to 7 are dedicated to DCU (see 3.2.6.1).
- CCR : Bits 0 to 12 are dedicated to DCU (see 3.2.6.2).
- BSC : 6-bit Barrel Shifter Control register.
BSC contains a 6-bit signed shift value (2's complement).
If the value is positive (resp. negative), all shift using BSC content will provide a left (resp. right) shift.
After reset, BSC value is 0.
- PSC : 6-bit Product Shift Control register.
PSC contains a 6-bit signed shift value.
If the value is positive (resp. negative), there will be a left (resp. right) shift on P-register.
After reset, PSC value is 0.

3.2.1.3 Multiplier

The D950-CORE multiplier performs 16 x 16-bit multiplications with the following implementations (see SL and SR bits of STA register) :

SL	LL	SR	LR	Multiplication
0	**X**	**0**	**X**	**Unsigned L-source x Unsigned R-source**
1	**0**	**0**	**X**	**Signed L-source x Unsigned R-source**
0	**X**	**1**	**0**	**Unsigned L-source x Signed R-source**
1	**0**	**1**	**0**	**Signed L-source x Signed R-source**
SL	**1**	**SR**	**X**	**Unsigned L0 x Signed/Unsigned R-source (depending on SR-bit) or Signed/Unsigned L-source x Signed/Unsigned R-source**
SL	**X**	**SR**	**1**	**Signed/Unsigned L-source x Unsigned R0 (depending on SL-bit) or Signed/Unsigned L-source x Signed/Unsigned R-source**

The operands, which can be considered as 16 or 32-bit, are provided by a subset of the register file and stored in L1 / L0 and R1 / R0 through X and Y buses.

The multiplication is performed in one single instruction cycle and the result is loaded in the 32-bit P register. The product can be either integer or fractional (see I-bit of STA register).

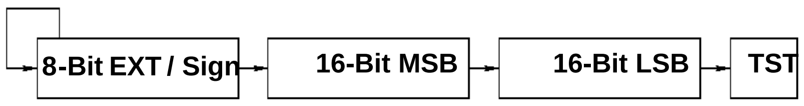
I	Product
0	**Fractional L-source x Fractional R-source**
1	**Integer L-source x Integer R-source**

Rounding of the product is explicitly defined in the instructions involving the multiplier (see 4.1.2.2).

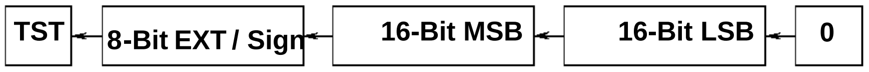
3.2.1.4 Barrel Shifter Unit (BSU)

The D950-CORE BSU provides a complete set of shifting functions :

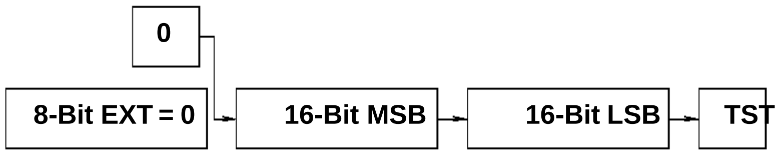
- **Arithmetic Shift with 40-bit inputs (either a 32-bit operand sign extended to 40-bit or a 40-bit accumulator) and providing a valid 40-bit result.**
  - **Right: shifts the 40-bit input data to the right, the upper part being sign extended,**



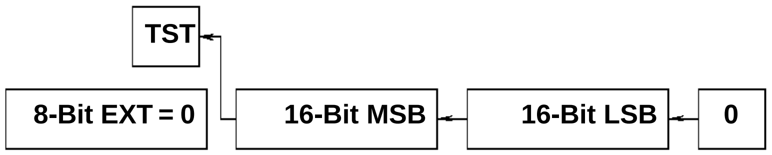
  - **Left: shifts the 40-bit input data to the left, the lower part being fed by 0,**



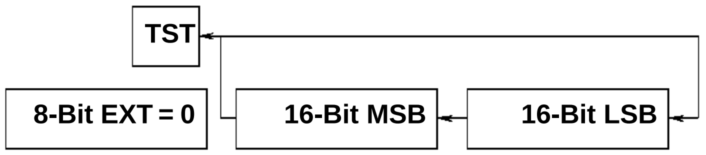
- **Logical Shift providing a 32-bit result which is loaded into a 40-bit accumulator, the 8-bit extension of which is reset.**
  - **Right: shifts the 32-bit input data to the right, the upper part being fed by 0,**



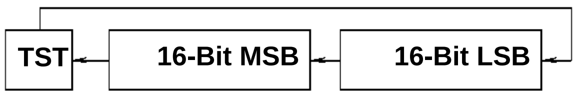
  - **Left: shifts the 32-bit input data to the left, the lower part being fed by 0,**



- **Rotation**
  - **Right: rotates the input data to the right (only through BSC register),**
  - **Left: rotates the 32-bit input data to the left,**



  - **Left with TST: rotates the 33-bit data made of the concatenation of TST-bit of CCR with the 32-bit input data to the left (the LSB of the 32-bit inputs is fed by TST-bit, the MSB of the 32-bit input feds the TST-bit of CCR).**



When using a pure shift instruction, the TST-bit of the CCR is fed by the last bit shifted out.

VR02017O

The shift value to be considered by the BSU is a signed value which may be provided in three different ways:

- by the instruction (shift defined in the instruction : see 4.1.2.2.),
- by the BSC register (shift defined in the ALU code : if BSC contains a positive (resp. negative) value, all shifts using BSC content will provide a left (resp. right) shift),
- by the PSC register (shift defined in the MAC instruction : if PSC contains a positive (resp. negative) value, all shifts using BSC content will provide a left (resp. right) shift).

3.2.1.5 Arithmetic and Logic Unit (ALU)

The D950-CORE ALU is 40-bit wide and implements about sixty ALU functions. It includes an 8-bit extension for arithmetical operations.

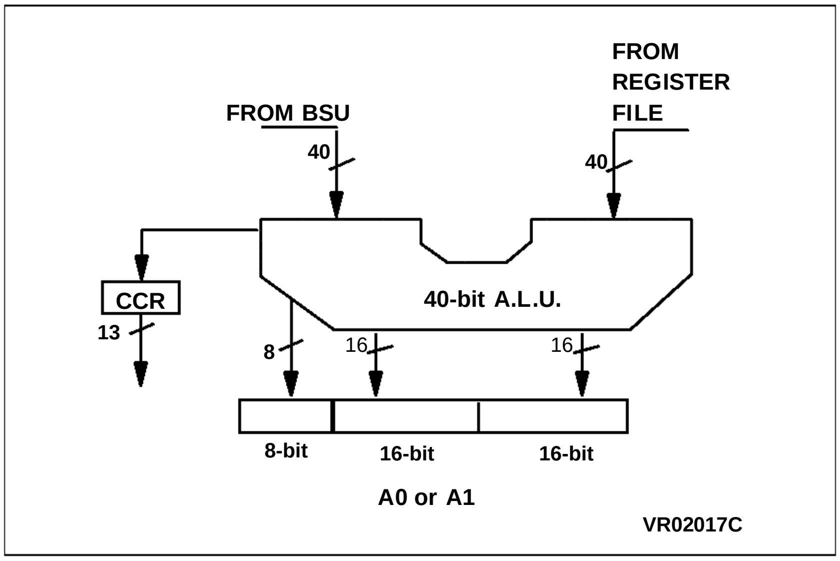
The calculation mode is controlled by both the instruction and the corresponding bits of the STA register (see 3.2.6.1). The ALU has two inputs (see figure) : the left (always the output of the BSU) and the right one (fed by the registers constituting the register file).

In case of logical operations, the ALU is fed with 32-bit wide operands, 0-extended to 40-bit. Then, the ALU generates a 40-bit result which is always stored in A0 or A1 (A0E and A1E extension registers being reset).

In case of arithmetical operations, the ALU is fed with 40-bit wide operands.

- If the operand is an accumulator, the entire 40-bit register (A0E/A0H/A0L or A1E/A1H/A1L) feds the 40-bit ALU.
- If not, the 32-bit register is considered as being sign extended to a 40-bit format. The extended ALU then generates a 40-bit wide result which is always stored in A0E/A0H/A0L or A1E/A1H/A1L.

Figure 5. D950-CORE Alu Operations



The ALU output is always one of the two accumulators and the CR (with the exception of particular ALU codes which affect only CR or an accumulator). The ALU operations can be partitioned into three different groups (see 4.1.2.2), depending on the number of operands the operation implies :

ALU Code	Number of Sources	Number of Destinations
3 operands	**2**	**1**
2 operands	**1**	**1**
1 operand	**1 (*)**	**1 (*)**

(*) Source = Destination

Specific ALU codes (see 4.1.2.2) are used to implement a non-restoring conditional add/subtract division algorithm.
The division can be signed or unsigned. The dividend must be a 32-bit operand and sign extended to 40-bit and located in the 40-bit accumulator. The divisor must be a 16-bit operand located in R0 or R1 (LR-bit of STA register must be low).
In order to obtain a valid result, the absolute value of the dividend must be strictly smaller than the absolute value of the divisor (considering operand is in a fractional format).

Special features are implemented in the D950-CORE in order to process multi-precision data (see DMULT instruction for double-precision MAC operations).

Different overflow preventions exist in the D950-CORE (see SAT and ES bits of STA register) :

- **the first concerns the multiplier when multiplying 0x8000 by 0x8000 in signed/signed fractional mode.**
  In this case, the saturation block forces the multiplier result to 0x7FFFFFFF,
- **the other concerns the ALU when result overflows.**
  In this case, provided one of the two optional saturation modes (32-bit saturation or 40-bit saturation) has been selected, the accumulator destination is set to plus or minus the maximum value.

Two rounding operations are enabled in the D950-CORE (see RND-bit of STA register) :

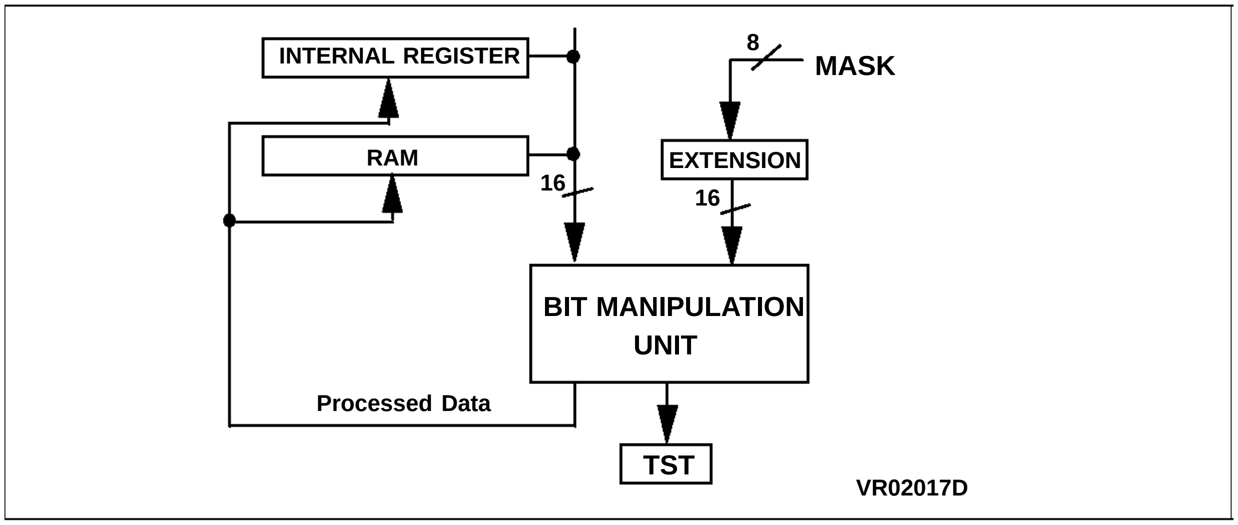
- **the first concerns the multiplier result stored in P register explicitly defined by the instruction. A two's complement rounding is performed on the result which is stored in the 16-bit PH register (see 4.1.2.2),**
- **the other concerns the 40-bit accumulator (either two's complement or convergent rounding) provided by ALU operation (see RND-bit of STA register).**

3.2.1.6 Bit Manipulation Unit (BMU)

The D950-CORE BMU (see figure) allows bit manipulation operations on 16-bit data sources accessed in 3 different modes: direct, indirect and register addressing, through dedicated instructions (see 4.1.2.3).
An 8-bit mask is applied to enable the following operations on a bit-per-bit basis :

. TSTL : bit test low,
. TSTH : bit test high,
. TSTHSET : bit test high and set,
. TSTLCLR : bit test low and reset.

Figure 6. D950-CORE Bit Manipulation Unit



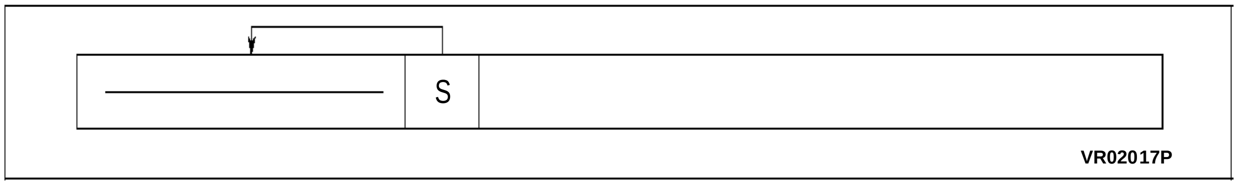
This 8-bit mask is extended to a 16-bit mask in three different ways :

. 8-bit value on MSBs, 0x00 on LSBs,
. 0x00 on MSBs, 8-bit value on LSBs,
. 8-bit value on MSBs, 8-bit value on LSBs (*).

(*) In this case, the mask value is the same on MSB and LSB.

15 8	7 0
MASK	0
0	MASK
MASK	MASK

Note : **For registers with a length less than 16-bit (AIE, BSC, PSC), the signed value data is sign-extended to a 16-bit signed value data before being tested.**



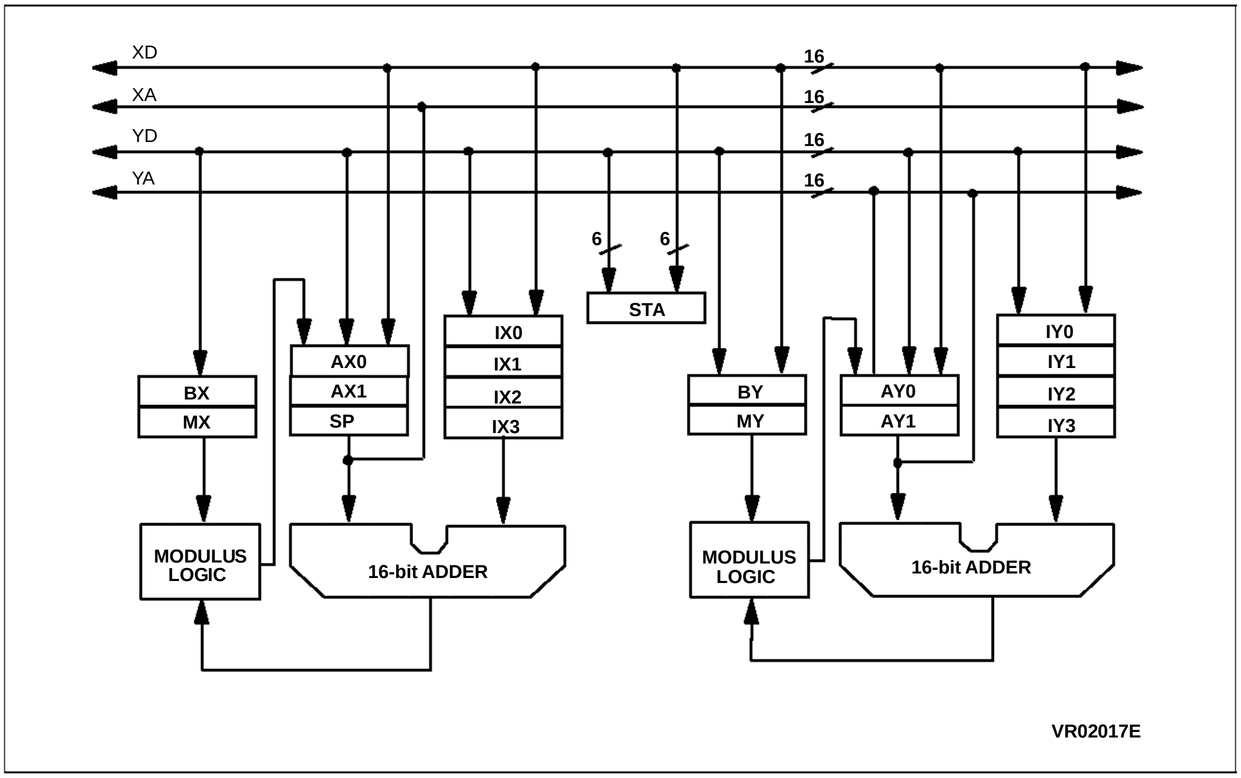
3.2.2 Address Calculation Unit (ACU)

3.2.2.1 Description

The D950-CORE ACU includes (see figure) two identical address generators (one for each Data memory space), each containing :

- **2 x 16-bit address registers,**
- **4 x 16-bit index registers,**
- **an adder performing address register update,**
- **2 x 16-bit Base and Maximum address registers for modulo addressing with dedicated logic for address comparison and calculation.**

Figure 7. D950-CORE Address Calculation Unit



In addition to these two address generators, the D950-CORE ACU includes :

. a 16-bit Stack Pointer (SP) register for the X-memory space mapped stack,
. 6 bits of STA register for addressing modes.

3.2.2.2 Registers

The D950-CORE ACU includes two types of registers :

. Data registers :

- **2 x 16-bit pointer registers and 4 x 16-bit index registers are dedicated for each Data memory space:**
  . AX0/AX1 (pointer), IX0/IX1/IX2/IX3 (index) for X-memory space,
  . AY0/AY1 (pointer), IY0/IY1/IY2/IY3 (index) for Y-memory space.
  In addition to these registers, a 16-bit SP register addresses the stack located in the X-memory space.
  All these registers are directly addressed by instructions.
- **2 x 16-bit base and maximum address registers are dedicated for each Data memory space :**
  . BX (Base), MX (Maximum) for X-memory space,
  . BY (Base), MY (Maximum) for Y-memory space.
  These four registers are mapped in Y-memory space.

. Control register :

- STA : Bits 8 to 13 are dedicated to ACU (see 3.2.6.1).

Note **: Index register values are 16-bit signed.**

3.2.2.3 Addressing modes

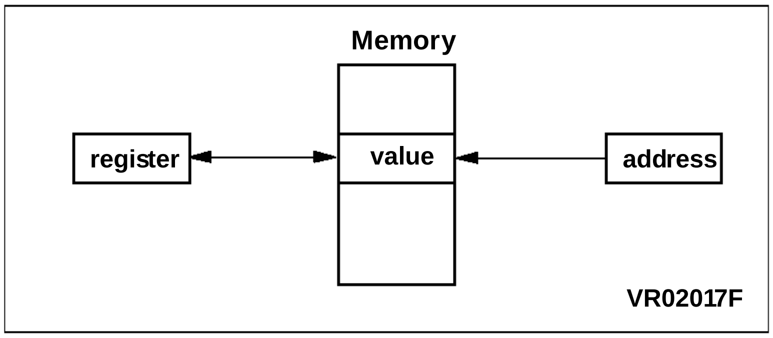
The D950-CORE provides a large range of addressing modes

Addressing Modes	Type
DIRECT	
INDIRECT	**LINEAR POST- INCREMENT**
	MODULO POST-INCREMENT
	BIT-REVERSE POST-INCREMENT
	INDEXED
IMMEDIATE	
STACK	

. Direct addressing

Memory direct addressing instructions require one extension word to provide the memory address and are executed in two cycles. They are used for data moves between memory and direct addressable registers.
Registers addressable by the instruction code include :
- 16 for DCU (L1/L0/R1/R0/A1E/A1H/A1L/A0E/A0H/A0L/PH/PL/BSC/PSC/STA/CCR),
- 13 for ACU (AX0/AX1/IX0/IX1/IX2/IX3/AY0/AY1/IY0/IY1/IY2/IY3/SP),
- 3 for PCU (LS/LC/LE).

Figure 8a. Direct Addressing



. Indirect addressing (see RX1, RX0, MY1, MY0, MX1 and MX0 bits of STA register)

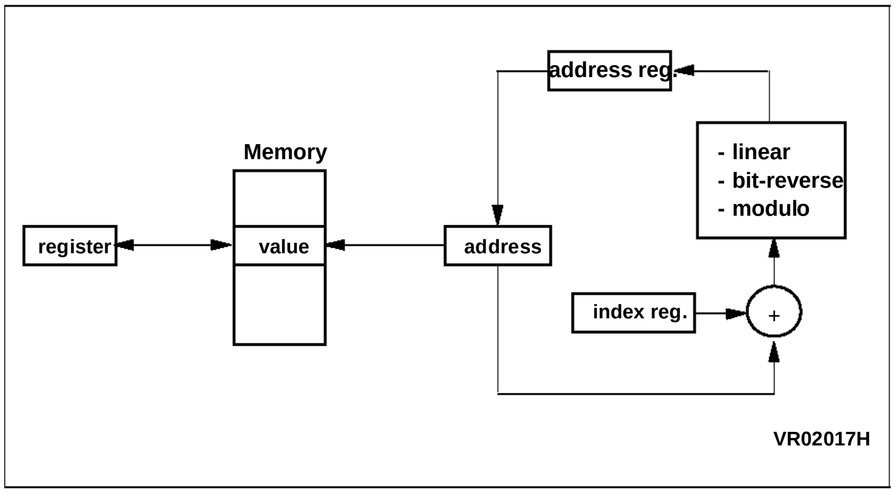
The instruction specifies the address register (AX0, AX1, AY0, AY1) of the operand to process and the address calculation to be performed, according to STA register content.
At the end of the instruction, the new address register (AXi / AYi) contains the previous selected address (AXi / AYi) post-incremented by the corresponding index registers (IXi / IYi).
Four types of indirect addressing modes are implemented :

- Linear addressing with post-increment
Address modification is done using the normal 16-bit 2's complement linear arithmetic.

- Modulo addressing with post-increment
This mode can be selected individually for AX0, AX1, AY0, AY1 register (see MX0, MX1, MY0, MY1 bits of STA register).
BX / MX : 16-bit register Base / Maximum address for AX0 / AX1,
BY / MY : 16-bit register Base / Maximum address for AY0 / AY1.
Base and Maximum addresses can be defined to any value, provided that :
. maximum address is greater than base address,
. the starting address is initialized within the Base/Maximum address range,
. the index absolute value is less than or equal to Maximum address - Base address.

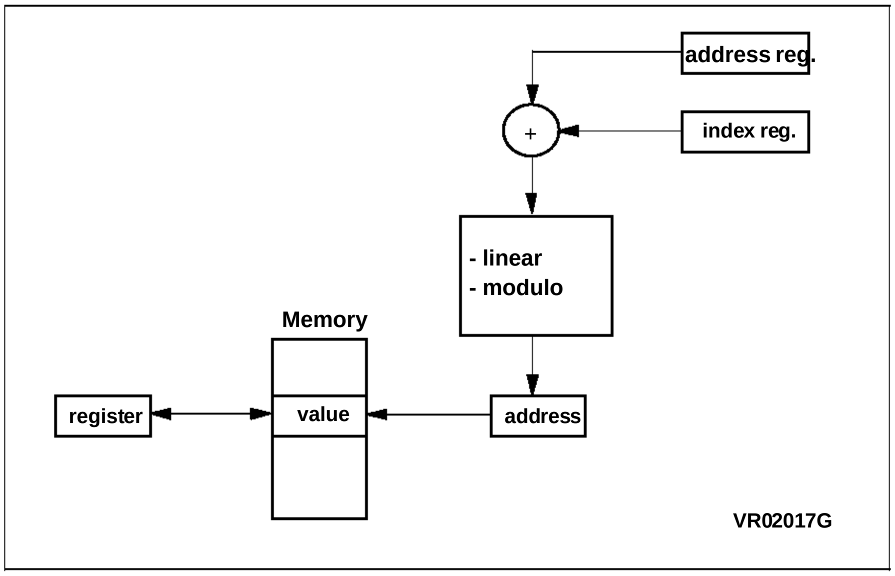
- **Bit reverse addressing (on X-memory space only) with post-increment**
  This mode can be selected for AX0, AX1 (see RX0, RX1 bits of STA register).
  It generates the bit-reversed address for 2^k point FFT implementation (Index value = 2k-1).

Figure 8b1. Indirect Addressing with Post-Modification.



- **Indirect indexed addressing**
  In this mode, the address of the operand is the sum of the contents of the address register (AXi, AYi or STA) and the contents of the selected index register (IXi or IYi).
  This addition occurs before the operand is accessed and therefore requires an extra instruction cycle. The contents of the selected address and index registers are unchanged.

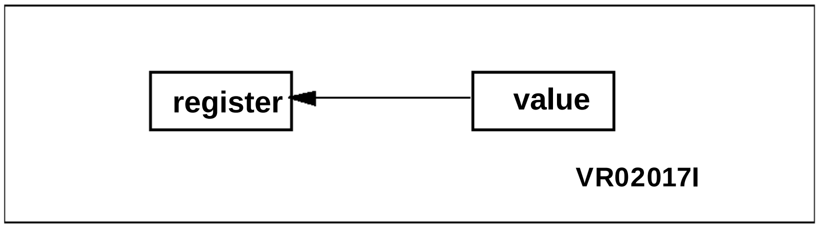
Figure 8b2. Indirect Addressing without Post-Modification.



. Immediate data addressing

This mode allows direct register loading.
If the data is 16-bit long (see LR and LL bits of STA register), this mode requires one word of instruction extension to store the data.
Immediate short data addressing is possible on 6-bit data, without instruction extension.
If AXi, AYi or STA are concerned, the 6 LSB's are loaded from the instruction and the MSB's are unchanged. For all other registers, the MSB's are fed with 0.

Figure 8c. Immediate Addressing.



. Addressing modes with stack operation

A 16-bit Stack Pointer (SP) register is available for X-memory space. It can be initialized to any value, provided it points to a stack dedicated memory area. The stack size is limited to the available memory. No provision is taken to detect stack overflows or underflows. After reset, SP register is not initialized.
The following addressing modes are possible with the 16-bit SP register :

- **PUSH (SP pre-decrement) or POP (SP post-increment) for register to-stack move, memory-to-stack move and for immediate value-to-stack move,**
- **indirect indexed addressing for register-to-stack move.**

3.2.3 Program Control Unit (PCU)

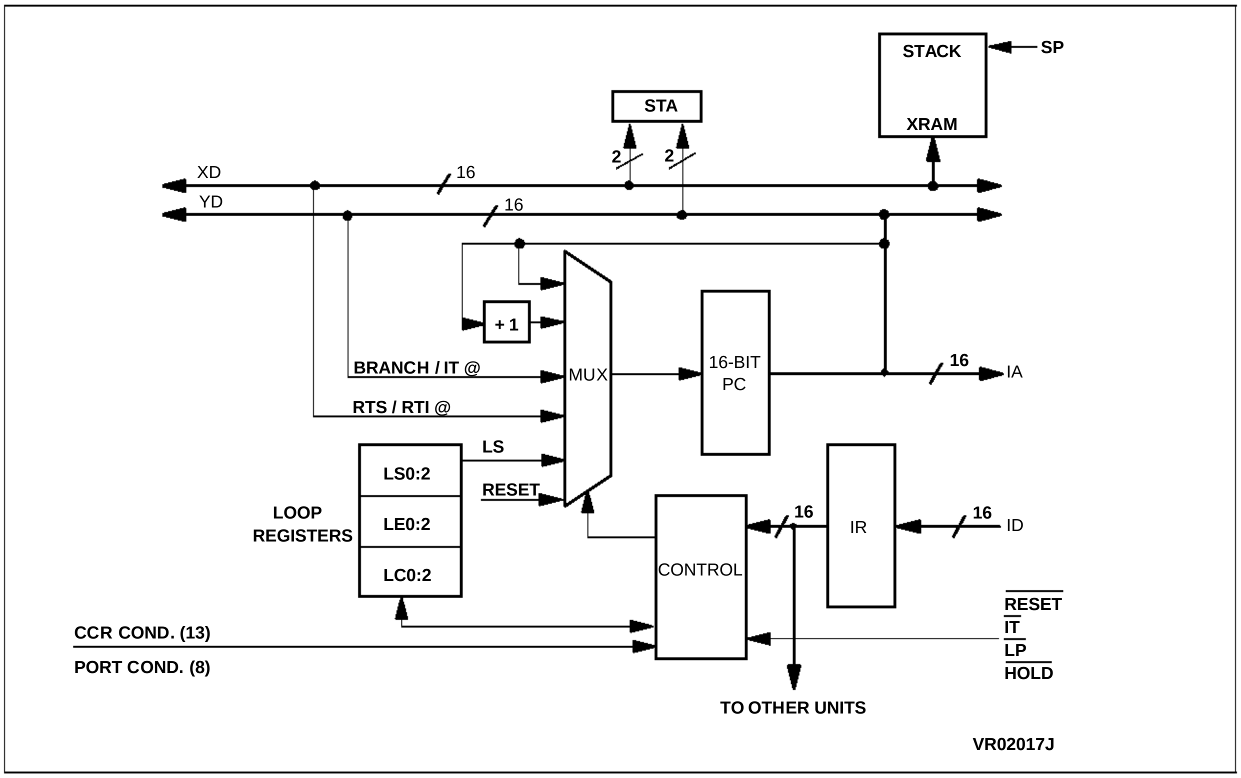
3.2.3.1 Description

The D950-CORE PCU includes (see figure) the following main components :

. a 16-bit Program Counter (PC),
. 9 x 16-bit Loop registers (3 x LS, 3 x LE, 3 x LC),
. Branch and Hardware Loops control logic including CCR and PORT condition decoding,
. 2 bits of STA register for interrupt control,
. 2 bits of CCR for loop management,
. Reset, Hold and Low-Power operation control logic,
. Stack control logic for automatic PC save and restore in Subroutine Calls and Interrupts.

Note **: The Stack is implemented in a user-defined dedicated X-RAM area. The Stack pointer and its control logic are included in the ACU (see 3.2.2.1).**

Figure 9. D950-CORE Program Control Unit



3.2.3.2 Registers

The D950-CORE PCU includes two types of registers :

. Data registers :

- LS0 / LS1 / LS2 : 3 x 16-bit Loop Start address registers,
- LE0 / LE1 / LE2 : 3 x 16-bit Loop End address registers,
- LC0 / LC1 / LC2 : 3 x 16-bit Loop Count registers.

All these registers are direct addressing through LSP instruction (see § 3.2.3.5)
After reset, LSP = 0. (No hardware loop is selected).

. Control registers :

- STA : Bits 14 and 15 are dedicated to PCU (see 3.2.6.1).
- CCR : Bits 14 and 15 are dedicated to PCU (see 3.2.6.2).

3.2.3.3 Instruction pipeline

Instruction execution is performed in a 3-stage pipeline : fetch/decode/execute.
While instruction n is executed, instruction n+1 is decoded and instruction n+2 is fetched.
The instruction cycle period is twice the CLKIN period.

According to the number of words used, D950-CORE instructions can be of two types (see 4.1.1) :

. one word instruction
Inside this group, most D950-CORE instructions are one cycle instructions (all arithmetic and logic instructions except instructions performing double precision multiplication and bit manipulations).
Some instructions are multiple cycle instructions.
Instructions causing a program flow change (JUMP, CALL, RTS, RTI, SWI, RESET, BREAK, CONTINUE) are executed in three cycles.

. instructions with extension words
As one Program memory word is fetched at each cycle, if an instruction needs extension words, they are fetched during the cycles following the first fetch.

3.2.3.4 Interrupt sources

The D950-CORE includes three interrupt sources which are, from highest to lowest priority :

Sources		Priority
RESET	**Non-Maskable**	**Highest**
SWI	**Non-Maskable**	↑
INT	**Maskable**	**Lowest**

. RESET : non maskable (internally vectorized), either hardware or software (see § 5.3.3)

In hardware, when a low level is applied to the $\overline{\text{RESET}}$ input, the CLOCK generator is resynchronized,the PC is reset, execution of NOP instructions is forced and control registers are initialized.
In order to get a valid reset, the low level must be applied for a minimum of ten CLKIN cycles(i.e five D950-CORE cycles).
In software, the RESET instruction is a 3-cycle instruction having the same effects as a hardwarereset, except the CLOCK generator is not resynchronized.
The reset address is 0x0000. By setting the MODE pin to 1, the alternate reset address 0x6C00 is selected

. SWI : non maskable (internally vectorized) software interrupt.
SWI is a 3-cycle instruction whose interrupt routine address is 0x0002. During these three cycles STA register, CCR and the return address are saved onto the stack and the first instruction of the SWI routine is fetched.
Return from the SWI routine is performed through RTI, a 3-cycle instruction during which the return address, STA register and CCR are retrieved from the stack.
SWI routine is non-interruptible by an external interrupt request.

. IT : maskable external interrupt (see EI and IPE bits of STA register) (see § 5.3.5)

- Start of Interrupt :
External interrupt is disabled on reset and is enabled by setting EI bit to 1.
As soon as an $\overline{\text{IT}}$ falling edge is memorized and recognized by the PCU at the beginning of instruction cycle, IPE-bit is set and, provided $\overline{\text{IT}}$ has been previously enabled, $\overline{\text{ITACK}}$ signal is asserted low to acknowledge the interrupt. $\overline{\text{ITACK}}$ stays at the low state during one cycle allowing the interrupt vector to be provided by the controller on Y-bus. Then IPE-bit is reset. Interrupt start processing requires three cycles to read the interrupt vector and to fetch the corresponding instruction. Meanwhile, STA register, CCR and the return address are automatically saved onto the stack, located in X-memory space.

- Return from Interrupt :
Return from the interrupt is performed by RTI instruction. $\overline{\text{EOI}}$ signal is then asserted low, allowing the controller to arbitrate pending interrupt requests and to issue, if required, the next interrupt request to the D950-CORE.
An interrupt request recognized while decoding or executing a delayed branch instruction is not acknowledged until all operations related to the branch have been completed.
In addition to this external interrupt source, a powerful interrupt controller is available as peripheral of the D950-CORE (see 6.3).

Notes : IT should not be asserted low if a previous request has not been acknowledged. In this case, the previous request will not be processed.
EI and IPE bits are not affected when STA register is restored.

3.2.3.5 Loop controller

Loop Instruction	Body of loop	Loop value
REPEAT	**Single instruction**	**Immediate**
	Block of instructions	**Immediate**
	Block of instructions	**Computed**

. Hardware loop resources

The program sequencer includes a powerful hardware loop mechanism allowing the nesting of up to three levels of loops by using nine 16-bit registers organized in three banks of three registers.
Each bank includes one loop start address register (LS), one loop end address register (LE) and one loopcount register (LC). These registers can be read and written by register move instructions, allowing extension of the number of nested loops by software.
The currently selected loop register bank is pointed to by LSP1 and LSP0 bits of the CCR. When the current level changes, this 2-bit register is incremented or decremented through dedicated instructions (see 4.1.2.5) to modify the selected bank.

. Loop operation : REPEAT instruction

The REPEAT instruction performs automatic management of the different loop registers (LS, LC, LE and LSP) and defines the number of iterations and address of the last instruction of the loop.
The loop begins at the instruction following REPEAT. Conditional instructions CONTINUE and BREAK can be put within a loop. Their effect is to restart (resp. exit from) the loop when the condition is verified.

Notes : **The maximum repeat count value of 2^{16} is obtained by setting LC to 0xFFFF.**
An endless loop can be set up by initialiazing LC to:
. 0x0000 for REPEAT block,
. 0x0001 for REPEAT single.

3.2.3.6 Sequence control

The PC is incremented every cycle when the program flow is linear.
Non linear sequencing occurs in the following cases :

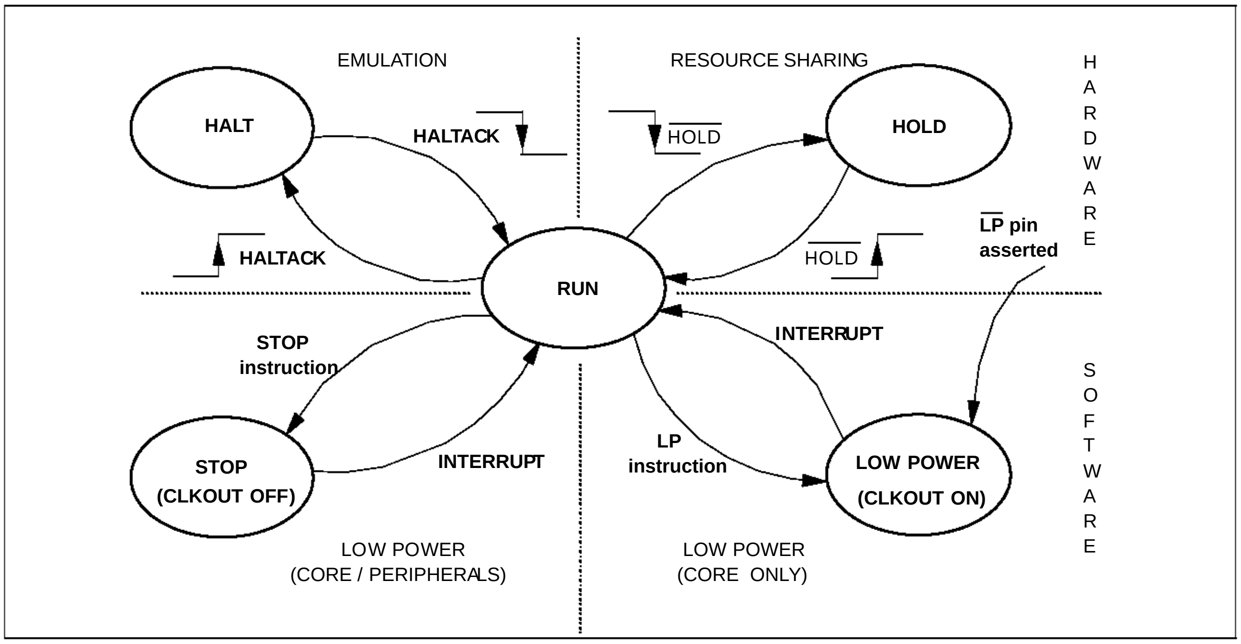
- **JUMP instructions (JUMP immediate or computed / delayed or not),**
- **CALL and RTS instructions (CALL can be immediate or computed / delayed or not / conditional or not),**
- **Interrupts and RTI instruction,**
- **Processing of automatic loops.**

Extension of program memory space to more than 64k x 16-bit can be achieved by including a memory-mapped instruction page register (IPR) into the D950-CORE glue logic.
This register will be read or written by move instructions.
Due to the pipe-line of instruction execution, changing page by loading a value into IPR will be effective at the time of execution of the following instruction, which is read in the current page.
This operation will work properly if no interrupt occurs between the load IPR and the JMP.
To avoid the need to disable interrupts by software before page change, a special memory mapped register address has been defined for IPR at address 0x0062.Y. Whenever a write with direct address or a POP with direct address is attempted at this address, execution of the following instruction can not be interrupted.

3.2.3.7 Halting program execution

The different ways to halt the program execution are (see figure) :

Figure 10. Different Ways to halt Program Execution.



. Low Power mode
- Entering Low Power mode :
There are two ways to enter the Low Power mode :

. execution of LP instruction.
The LP instruction being a 3-cycle conditional instruction, the Low Power mode is entered after the last execute cycle of LP instruction.
. driving $\overline{\text{LP}}$ to low state
$\overline{\text{LP}}$ is falling edge sensitive. Low Power mode can be entered only if the processor is not in HOLD state or in Emulation mode.
The instruction decoded at the time a LP request is recognized is executed.
Entering Low Power mode is acknowledged by driving $\overline{\text{LPACK}}$ low.
When operated in Low Power mode, the D950-CORE enters an idle state in which :
. the clock generator is stopped (internal cycle clock) and CLKOUT remains active,
. the internal state of the processor is frozen,
. X and Y data buses stay driven to Hi-Z,
. the bus address lines remain at their current value and control lines remain at inactive high state.

- Exit of Low Power mode :
Exit of the Low Power mode is initiated by detecting a falling edge on $\overline{\text{IT}}$.
The processor clock generator is restarted and $\overline{\text{LPACK}}$ is driven to high state.
If interrupts were disabled, program execution restarts from the current PC and interrupt handshake signals $\overline{\text{ITACK}}$ and EOI are not activated.
If interrupts were enabled, a normal interrupt process starts.

. STOP mode

- Entering STOP mode :
STOP mode is entered by STOP instruction. STOP instruction is processed as the LP instruction, excepted that CLKOUT is stopped at the same time as the internal clock is stopped. $\overline{\text{LPACK}}$ signal is activated in the same way as for LP instruction.

- Exit of STOP mode :
Exit of the STOP mode is performed by detection of an interrupt request with the same conditions as for exit of LP. $\overline{\text{LPACK}}$ signal is activated in the same way as for $\overline{\text{LP}}$.

. HOLD state
This function allows the release of the buses for another device such as a DMA controller (see § 5.3.6).

- Entering HOLD state :
HOLD signal is sampled at the beginning of every cycle. When $\overline{\text{HOLD}}$ is recognized low, the processor releases the I-bus immediately and X and Y buses after execution of the currently decoded instruction. Bus address, data and control lines are then tri-stated.
Program execution is stopped and $\overline{\text{HOLDACK}}$ is asserted low during HOLD state.

Note **: HOLD state can not be entered when the processor is in Emulation mode.**

- **Exit of HOLD state :**
  The processor recovers bus mastership as soon as $\overline{\text{HOLD}}$ is sampled high and next instruction is fetched.

. HALT state
This function is used in emulation mode only and allows to stop program execution by the ETU unit (see 3.2.4).

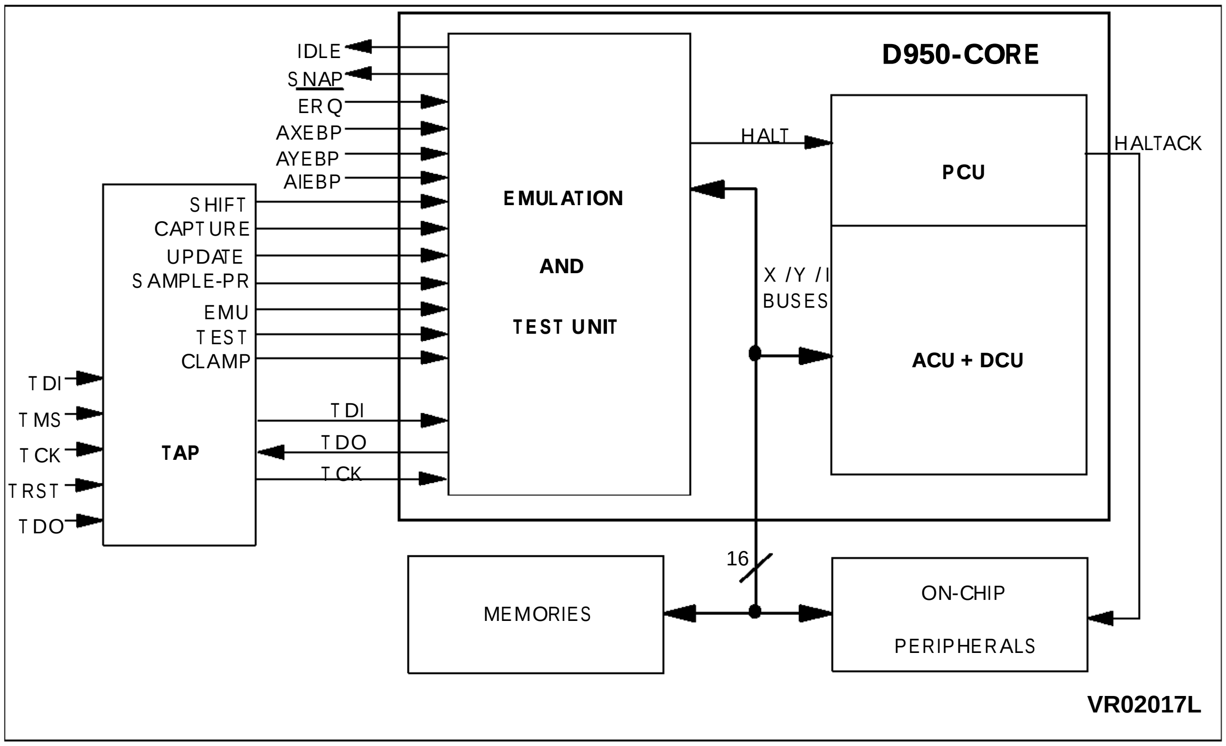
. Memory moves with wait states (see § 5.3.4)
$\overline{\text{DTACK}}$ input allows to strech instruction cycles to access slower memory and/or peripherals.
$\overline{\text{DTACK}}$ is sampled on the rising edge of CLKOUT. If $\overline{\text{DTACK}}$ is recognized high on the third rising edge of the cycle, the cycle is extended by one CLKOUT cycle.
CLKOUT extension cycles are added by the clock generator until $\overline{\text{DTACK}}$ is recognized low.
Note **: $\overline{\text{DTACK}}$ generation can be controlled by the Bus Switch Interface peripheral (see 6.2).**

3.2.4 Emulation and Test Unit (ETU)

3.2.4.1 Description

The D950-CORE ETU (see figure) performs functions dedicated to emulation and test through the external IEEE 1149.1 JTAG interface.
The ETU operations are controlled by an on-chip Test Access Port (TAP) and an off-chip Emulator by means of dedicated control I/Os.

Figure 11. D950-CORE Emulation and Test Unit.



Entering the Emulation mode can be done through two different ways :

. by asserting low $\overline{\text{ERQ}}$ input pin,
. by meeting a valid breakpoint condition or executing an instruction in single step mode,

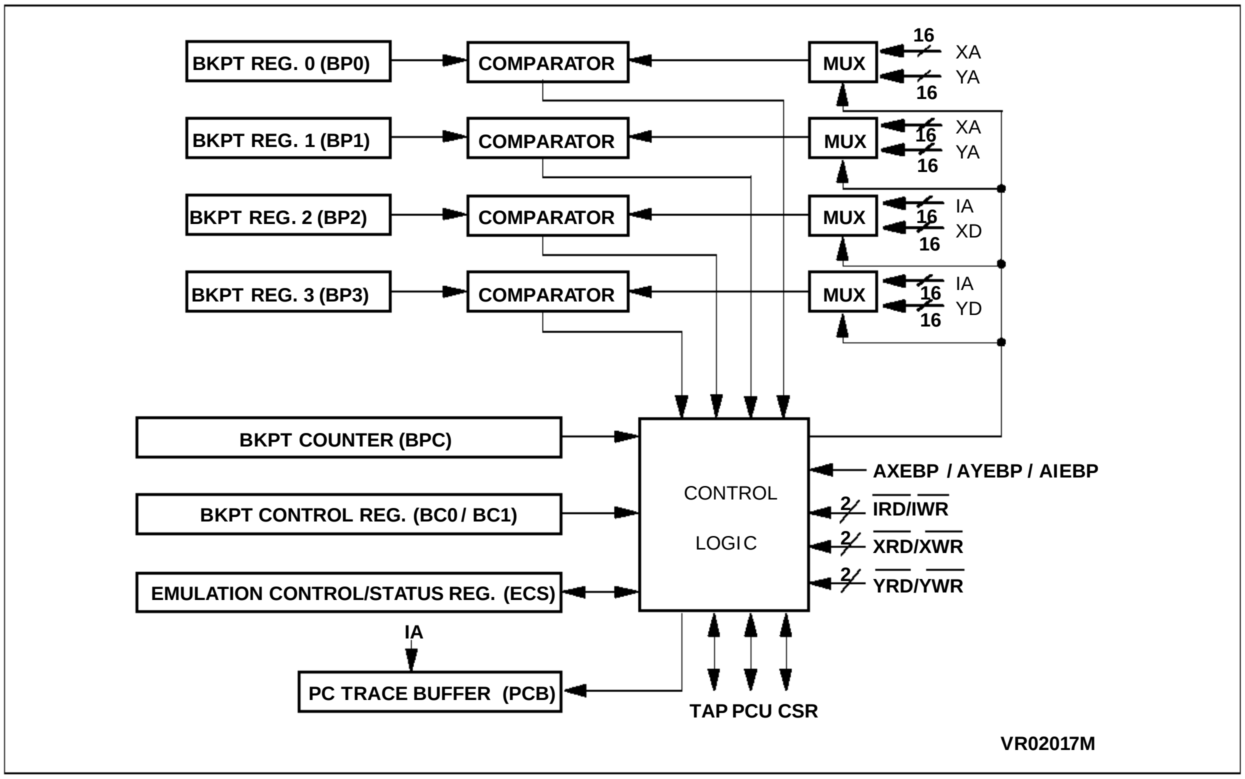
In emulation mode, most of D950-CORE instructions can be executed, including arithmetic/logic, JUMP/CALL and move instructions to display the processor status (memories and registers) and restore the context.

Exiting the emulation mode is controlled by the PC-board emulator through the JTAG interface.

The Emulation resources (see figure) includes :

- **a set of four Breakpoint registers (BP0, BP1, BP2, BP3) which can be affected by Program or Data memory,**
- **a Breakpoint counter (BPC),**
- **a Program Counter Trace Buffer (PCB) able to store the address of the 6 last executed instructions,**
- **three control registers for Breakpoint condition programming,**
- **a control logic for instruction execution through the PC-board emulator control.**

Figure 12. D950-CORE Emulation Block Diagram


BKPT REG. 0 (BP0)
BKPT REG. 1 (BP1)
BKPT REG. 2 (BP2)
BKPT REG. 3 (BP3)
COMPARATOR
MUX
16
XA
YA
IA
XD
YD
BKPT COUNTER (BPC)
BKPT CONTROL REG. (BC0 / BC1)
EMULATION CONTROL/STATUS REG. (ECS)
IA
PC TRACE BUFFER (PCB)
CONTROL LOGIC
AXEBP / AYEBP / AIEBP
2
$\overline{\text{IRD}}$/$\overline{\text{IWR}}$
$\overline{\text{XRD}}$/$\overline{\text{XWR}}$
$\overline{\text{YRD}}$/$\overline{\text{YWR}}$
TAP PCU CSR
VR02017M

The emulation and TAP controller interfaces (see 2.7 and 2.8) include pins of different types :

- **. Scan control (TDI, TDO, TCK),**
- **. TAP instructions (SAMPLE_PRE, EMU, TEST, CLAMP),**
- **. TAP controller states (UPDATE, SHIFT, CAPTURE),**
- **. Emulation control (ERQ, IDLE, SNAP, HALTACK, AIEBP, AXEBP, AYEBP).**

3.2.4.2 Registers

The registers are of two types, all mapped in the Y-memory space :

. Data registers :

- BP0, BP1, BP2, BP3 : Breakpoint 0, 1, 2, 3 values
16-bit breakpoint registers allowing breakpoints :

Breakpoint Register	Breakpoint Location	Bus
BP3	**Program memory address or Data memory data**	**IA or YD**
BP2		**IA or XD**
BP1	**Data memory Address**	**XA or YA**
BP0		

- BPC : Breakpoint counter register
This 16-bit register is initialized by a load instruction and decremented each time a valid condition is met on a count enabled breakpoint.
Breakpoints with and without a count condition can be set simultaneously.
After reset, BPC is not initialized.

- PCB : Program Counter Trace Buffer register
A 6-word circular buffer allows the user to keep trace of the PC value for the six last executed instructions.
PCB stores one address value per instruction, whatever the instruction type (single cycle, single word, multiple cycles, multiple words).

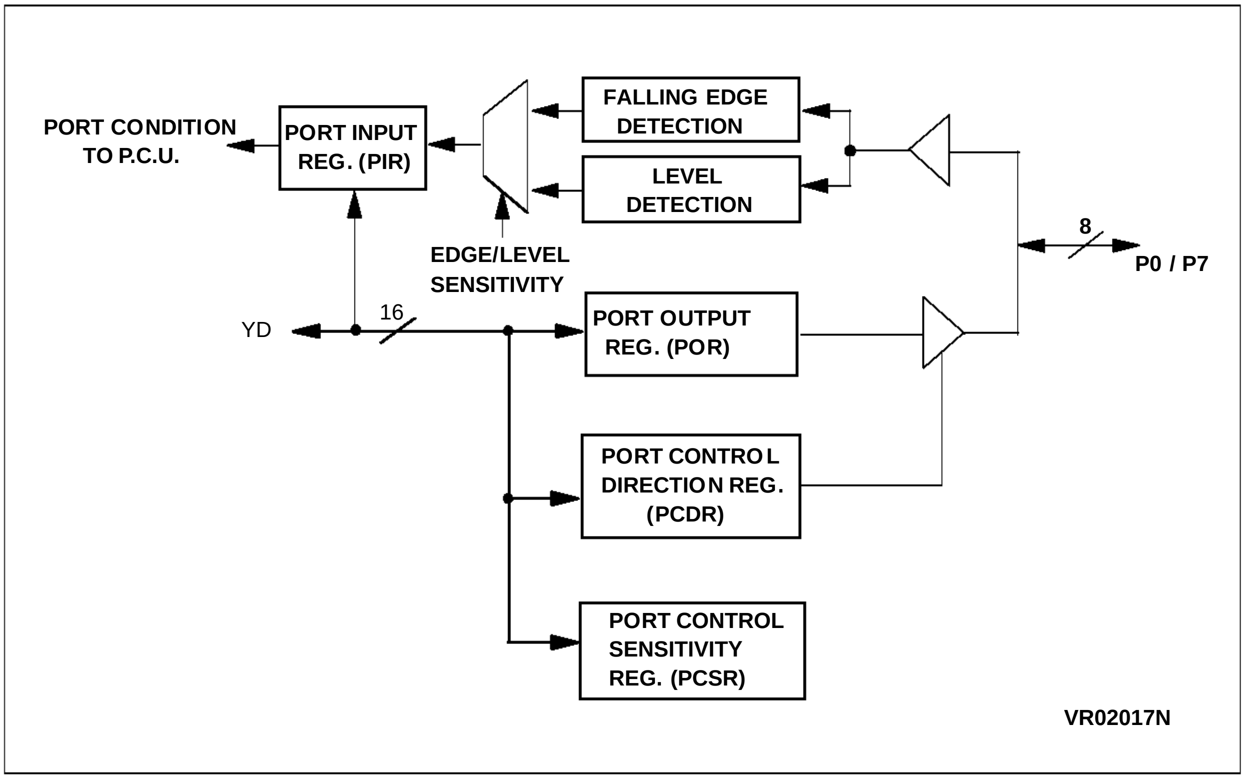
. Control and Status registers :
Three breakpoint control registers allow simple or multiple breakpoints (conditional or not) to be set and counting breakpoint events to be enabled.

3.2.5 General Purpose P-Port

3.2.5.1 Description

The P-Port is an 8-bit (P0/P7) general purpose parallel port in which each port pin can be individually programmed as input (level or falling edge sensitive) or output (see figure).
The data direction and sensitivity for each bit are programmed through PCDR and PCSR 8-bit registers.
Port inputs are sampled on each CLKOUT rising edge. Detection of a level change is performed, provided the input remains at the same level for, at least, one CLKOUT cycle.
The Port input data is stored into the 8-bit Port Input Register (PIR)
The Port output data is stored into the 8-bit Port Output Register (POR)
Note **: The significant bit are 8-LSBs (8-MSBs = 0 when reading).**

Figure 13. D950-CORE Parallel I/O port.



3.2.5.2 Registers

The Port Control Direction Register PCDR defines the data direction of each port pin. After reset, PCDR default value is 0 (Port pins are configured as inputs).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	P7D	P6D	P5D	P4D	P3D	P2D	P1D	P0D

PiD : Port pin direction
0 : Input port pin (def.)
1 : Output port pin

Note **: - means RESERVED (read : 0, write : don't care)**

The Port Control Sensitivity Register PCSR defines sensitivity of each port pin. After reset, PCSR default value is 0 (Port pins are configured as level-sensitive).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	P7S	P6S	P5S	P4S	P3S	P2S	P1S	P0S

PiS : Port pin sensitivity
0 : Level sensitive (def.)
1 : Edge sensitive

Note **: - means RESERVED (read : 0, write : don't care)**

3.2.6 Common control registers

3.2.6.1 STA : Status register

STA is a 16-bitstatus register shared by both the DCU, the ACU and the PCU.
Bits 0 to 7 are dedicated to DCU (defines the calculation mode for certain instructions and gives precision about the type of operands to be used), bits 8 to 13 to ACU (initializes circular and bit-reverse addressing modes), bits 14 and 15 to PCU (controls interrupt).
After reset, STA default value is 0x004C.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EI	IPE	RX1	RX0	MY1	MY0	MX1	MX0	RND	ES	SAT	I	SR	SL	LR	LL
PCU		ACU						DCU							

EI : **Enable Interrupt**
- **0 : Interrupt is disabled (def.)**
- **1 : Interrupt is enabled**

IPE : **Interrupt Pending**
- **0 : Reset by hardware when the interrupt is acknowledged (def.)**
- **1 : Set by hardware when the trigger event occurs or by the programmer to generate an interrupt.**

RX1 : **Bit reverse addressing mode for AX1**
- **0 : No bitreverse addressing mode for AX1 (def.)**
- **1 : Bit reverse addressing mode is selected for AX1**

RX0 : **Bit reverse addressing mode for AX0**
- **0 : No bitreverse addressing mode for AX0 (def.)**
- **1 : Bit reverse addressing mode is selected for AX0**

MY1 : **Modulo addressing mode for AY1**
- **0 : No modulo addressing mode for AY1 (def.)**
- **1 : Modulo addressing mode is selected for AY1.**
  AY1 is updated through the Y-memory space modulo logic.

MY0 : **Modulo addressing mode for AY0**
- **0 : No modulo addressing mode for AY0 (def.)**
- **1 : Modulo addressing mode is selected for AY0.**
  AY0 is updated through the Y-memory space modulo logic.

MX1 : **Modulo addressing mode for AX1**
- **0 : No modulo addressing mode for AX1 (def.)**
- **1 : Modulo addressing mode is selected for AX1.**
  AX1 is updated through the X-memory space modulo logic.

MX0 : Modulo addressing mode for AX0
- **0 : No modulo addressing mode for AX0 (def.)**
- **1 : Modulo addressing mode is selected for AX0.**
  AX0 is updated through the X-memory space modulo logic.

RND : Rounding type
- **0 : Convergent rounding for ALU operation (def.)**
- **1 : Two’s complement rounding for ALU operation**

ES : Extended Saturation
- **0 : The saturation is active when a 32-bit overflow occurs (if SAT=1)**
- **1 : The saturation is active when a 40-bit overflow occurs (if SAT=1) (def.)**

SAT : Saturation
- **0 : ALU is not in saturated mode (def.)**
- **1 : ALU is in saturated mode**

I : Integer Product
- **0 : Product is in fractional format (if signed * signed, one bit is shifted left before storing the result into P register) (def.)**
- **1 : Product is in integer format (no shift and direct transfer into P register)**

SR : Right side operand type (only used for product calculation and division)
- **0 : Right side operand is unsigned**
- **1 : Right side operand is signed (def.)**

SL : Left side multiplicand type (only used for product calculation)
- **0 : Left side multiplicand is unsigned**
- **1 : Left side multiplicand is signed (def.)**

LR : Right side long data
- **0 : Normal 16-bit data mode (def.)**
- **1 : Data contained in R0 and R1 is long 32-bit data (the 16 MSB’s in R1, the 16 LSB’s in R0)**

LL : Left side long data
- **0 : Normal 16-bit data mode (def.)**
- **1 : Data contained in L0 and L1 is long 32-bit data (the 16 MSB’s in L1, the 16 LSB’s in L0)**

3.2.6.2 CCR:Code Condition Register

CCR is a 16-bit register shared by both the DCU (bits 0 to 12) and the PCU (bits 14 and 15).
It is affected each time an ALU operation occurs and gives information on the last result stored in A0 or A1 accumulator.
After reset, CCR default value is 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSP1	LSP0	-	TST	C31	NQ	CS	PAR	MN	N	EXT	MOVF	OVF	Z	C	S

PCU **DCU**

LSP1 / LSP0 Loop Stack Pointer
00 : No loop / Bank 1 (def.)
01 : Loop level 1 / Bank 1
10 : Loop level 2 / Bank 2
11 : Loop level 3 / Bank 3

TST : Result of the test instructions in bit manipulation or last bit shifted out in pure shift operations

C31 : Carry value generated out of bit 31 during the last ALU operation (always loaded except for DMULT instruction)

NQ : 1's complement of next quotient bit (only affected by DIVS and DIVQ instructions)

CS : Compared sign updated by CMPS instruction as the XOR of the two ALU operand signs (bit 31) (used also by DIVS, RESQ and RESR instructions)

PAR : Parity of the last ALU result.
0 : Bit 16 of the last 40-bit ALU result is 0 (def.)
1 : Bit 16 of the last 40-bit ALU result is 1

MN : Memorized normalized
0 : Reset when tested by a conditional instruction (def.)
1 : Set when ALU result is normalized

N : Normalized
0 : ALU result is not normalized (def.)
1 : ALU result is normalized (bits 31 to 39 are equal, opposed to bit 30)

EXT : Extension
0 : The 8-bit
1 : The last ALU result overflows the 32-bit format

MOVF :Memorized overflow
0 : Reset when tested by a conditional instruction (def.)
1 : Set when the last ALU result overflows the 40-bit format

OVF : Overflow

0 : An arithmetic overflows does not occur for the last 40-bit ALU result (def.)

1 : An arithmetic overflow occurs for the last 40-bit ALU result

Z : Zero

0 : ALU result is different from zero (def.)

1 : ALU result is zero

C : Carry value generated out of bit 39 during the last ALU operation

S : Sign

0 : ALU result is positive (def.)

1 : ALU result is negative

Note **: - means RESERVED (read : 0, write : don't care)**

3.2.6.3 Condition Table

	BIT	TRUE	FALSE
Test of CCR bits		**ALWAYS**	**NEVER**
	Z	**EQ.Z**	**NO EQ.Z**
	S XOR OVF	**LT.Z**	**NO LT.Z**
	(S XOR OVF) OR Z	**LT/E.Z**	**NO LT/E.Z**
	C	**C**	**NO C**
	S	**S**	**NO S**
	EXT	**EXT**	**NO EXT**
	OVF	**OVF**	**NO OVF**
	MOVF	**MOVF**	**NO MOVF**
	N	**N**	**NO N**
	MN	**MN**	**NO MN**
	PAR	**PAR**	**NO PAR**
	C31	**C31**	**NO C31**
	TST	**TST**	**NO TST**
Test of PORT bits	**P0**	**P0**	**NO P0**
	P1	**P1**	**NO P1**
	P2	**P2**	**NO P2**
	P3	**P3**	**NO P3**
	P4	**P4**	**NO P4**
	P5	**P5**	**NO P5**
	P6	**P6**	**NO P6**
	P7	**P7**	**NO P7**

3.2.6.4 Direct Address Register Table

Register Name	Function	Location
AX0	**X address register**	**ACU**
AX1	**X address register**	**ACU**
AY0	**Y address register**	**ACU**
AY1	**Y address register**	**ACU**
IX0	**X index register**	**ACU**
IX1	**X index register**	**ACU**
IX2	**X index register**	**ACU**
IX3	**X index register**	**ACU**
IY0	**Y index register**	**ACU**
IY1	**Y index register**	**ACU**
IY2	**Y index register**	**ACU**
IY3	**Y index register**	**ACU**
SP	**Stack Pointer register**	**ACU**
LS	**Loop Start register**	**PCU**
LC	**Loop Count register**	**PCU**
LE	**Loop End register**	**PCU**
L0	**DCU input left register (LSB)**	**DCU**
L1	**DCU input left register (MSB)**	**DCU**
R0	**DCU input right register (LSB)**	**DCU**
R1	**DCU input right register (MSB)**	**DCU**
PL	**Product register (LSB)**	**DCU**
PH	**Product register (MSB)**	**DCU**
CCR	**Condition Code Register**	**DCU**
STA	**Status register**	**DCU**
A0L	**Accumulator 0 (LSB)**	**DCU**
A0H	**Accumulator 0 (MSB)**	**DCU**
A0E	**Accumulator 0 (Extension)**	**DCU**
BSC	**Barrel Shifter Control register**	**DCU**
A1L	**Accumulator 1 (LSB)**	**DCU**
A1H	**Accumulator 1 (MSB)**	**DCU**
A1E	**Accumulator 1 (Extension)**	**DCU**
PSC	**Product Shift Control register**	**DCU**

Note **: Memory mapping is described in the appendix (see § 7.1)**

4 SOFTWARE ARCHITECTURE

4.1 INSTRUCTION SET

4.1.1 Overview

**The instruction execution is performed in a 3-stage pipeline : fetch/decode/execute.
While instruction n is executed, instruction n+1 is decoded and instruction n+2 is fetched.
The instruction cycle period is twice the CLKIN period.**

According to the number of words used, D950-CORE instructions can be of two types :

. one word instruction

Inside this group, most of D950-CORE instructions are one cycle instructions :

- **all arithmetic and logic instructions with or without parallel data moves, excepted instructions performing double precision multiplication and bit manipulations,**
- **register to register data move,**
- **memory to register indirect data move.**

Some instructions are multiple cycle instructions :

- **double precision MAC (two cycles),**
- **indirect indexed register move (two cycles),**
- **indirect indexed register to stack move (two cycles),**
- **register to Program memory transfer (four cycles).**

Instructions causing a program flow change (RTS, RTI, SWI, RESET, BREAK, CONTINUE) are executed in one to three cycles.

. extension word instructions

As one Program memory word is fetched at each cycle, if an instruction needs extension words, they are fetched during the cycles following the first fetch. Then, execution of the instruction starts two cycles after its first fetch cycle.

- **memory to register data move in direct addressing mode (2-words / 2-cycles) (second word =address value),**
- **immediate register load (2-words / 2-cycles) (second word = register value),**
- **repeat block up to 511 times (2-words / 2-cycles) (second word = LE),**
- **repeat single up to 2^{16}-1 times (2-words / 2-cycles) (second word = LC),**
- **repeat block computed (2-words / 2-cycles) (second word = LC),**
- **bit manipulations (2-words / 2-cycles) (second word = mask),**
- **immediate push (2-words / 3-cycles) (second word = immediate value),**
- **push/pop in direct addressing mode (2-words/3-cycles) (second word = direct address),**
- **repeat block up to 2^{16}-1 times (3-words/3-cycles) (second word = LC, third word = LE).**
- **JUMP and CALL instructions.**

4.1.1.1 Register list

The registers used in the D950-CORE instruction set are :

. AX0, AX1, AY0, AY1 address pointers,
. IX0, IX1, IX2, IX3, IY0, IY1, IY2, IY3 index registers,
. SP stack pointer,
. LS, LC, LE loop registers,
. A0E, A0H, A0L, A1E, A1H, A1Laccumulator registers,
. PH, PL product registers,
. CCR code condition register,
. STA status register,
. BSC barrel shifter control register,
. PSC product shift control register.

4.1.1.2 Code condition list

. ALU related conditions :

The following table indicates how the 13-bits of CCR related to DCU are affected by ALU operations :

Type **(*)**	**TST**	**C31**	**NQ**	**CS**	**PAR**	**MN**	**N**	**EXT**	**MOV**	**OVF**	**Z**	**C**	**S**
arithmetical		**X**			**X**	**X**	**X**		**X**	**X**	**X**	**X**	**X**
logical								**X**					
division			**X**	**X**									
pure shift or bit manipulation	**X**												

(*) and their negative conditions associated :
NOTST, NOC31, NONQ, NOCS, NOPAR, NOMN, NON, NOEXT, NOMOVF, NOOVF, NOZ, NOC, NOS.

. Parallel Port related conditions :

- P0 (NOP0), P1 (NOP1), P2 (NOP2), P3 (NOP3), P4 (NOP4), P5 (NOP5), P6 (NOP6), P7 (NOP7),

4.1.1.3 Branch conditions

. ALU related conditions (see 4.1.1.2)
. Parallel Port related conditions (see 4.1.1.2).

4.1.2 Instruction set

The D950-CORE instruction set is divided into different groups, according to operation type :

. Assignment,
. ALU,
. Bit Manipulation,
. Program control,
. Loop control,
. Co-processor,
. Stack.

Note **: Inside this instruction set, following notations are used :**
reg : D950-CORE internal register
AX (resp. AY) : address pointer for X (resp. Y) memory space
IX (resp. IY) : index pointer for X (resp. Y) memory space
L : input left register of DCU (L1 16-MSBs / L0 16-LSBs)
R : input right register of DCU (R1 16-MSBs / R0 16 LSBs)
A : 40-bit accumulator (A0 or A1)
AiH : 16-MSB of the Ai accumulator
P : Product result of the multiplier
i,j,k,m,n,p,q,x,y : 0 or 1
r,s : 0,1,2 or 3
xx : 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14 or 15

4.1.2.1 Assignment

. Assignment operations

Figure 14a. Assignment Operations

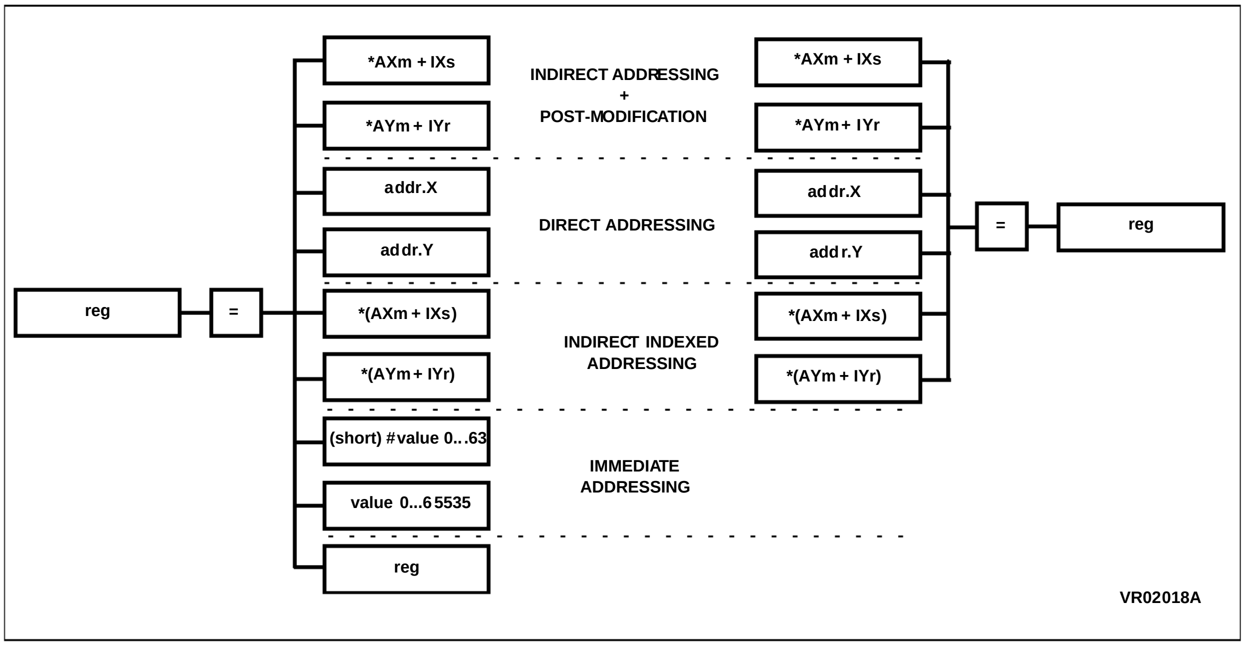
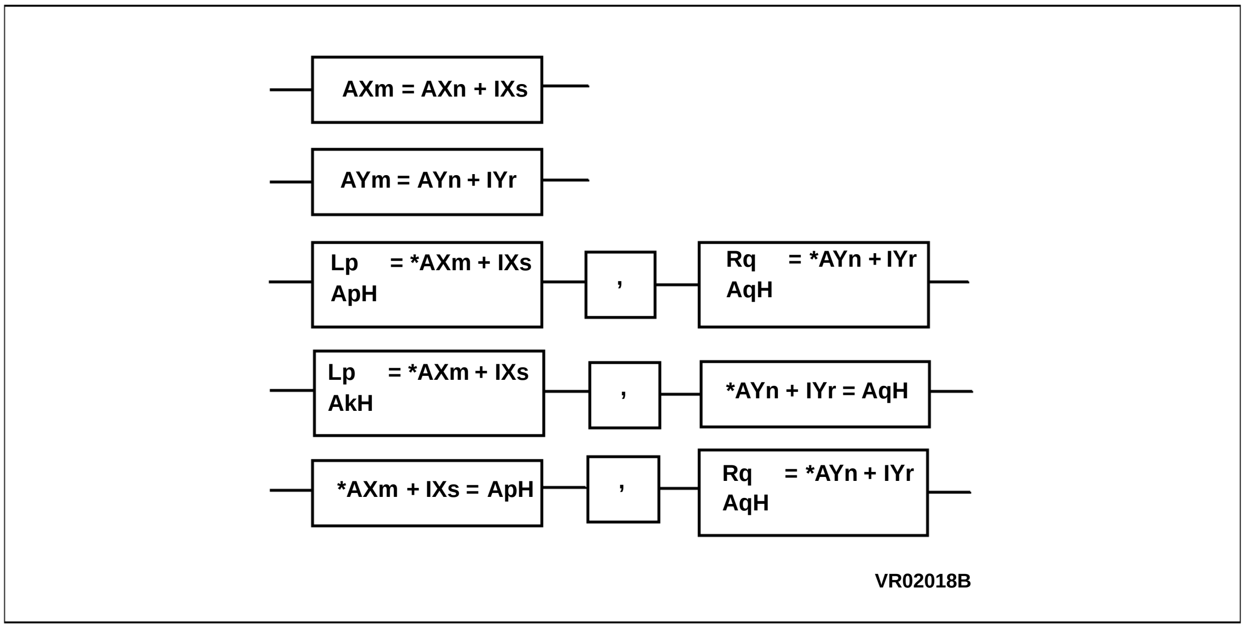


Figure 14b. Assignment Operations.



. Register/Program Memory Assignment operations

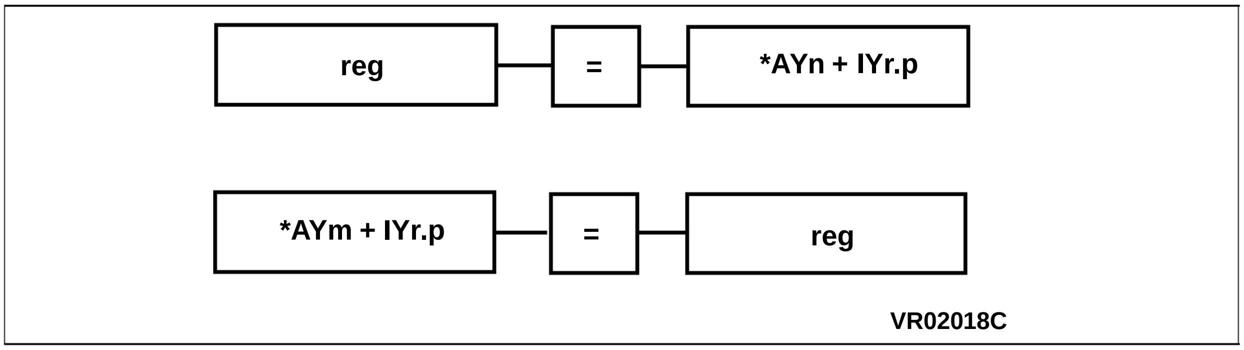


Figure 15. Register / Program Memory Assignment Operations.

4.1.2.2 ALU

. one-word operand (oper_1)

CLR	**Clear Accumulator (AiE:AiH:AiL = 0)**
CLRH	**Clear 24 MSBs of Accumulator (AiE:AiH = 0)**
CLRL	**Clear 16 LSBs of Accumulator (AiL = 0)**
CLRE	**Clear 8 Extension bits of Accumulator (AiE = 0)**
SET	**Set Accumulator (AiE:AiH:AiL = 0xFF FFFF FFFF)**
SETH	**Set 24 MSBs of Accumulator (AiE:AiH = 0xFF FFFF)**
SETL	**Set 16 LSBs of Accumulator (AiL = 0xFFFF)**
SEXT	**Accumulator is sign extended (AiE loaded with 8 times the MSB of AiH:AiL)**
ROUND	**Rounds the 40-bitaccumulator value**

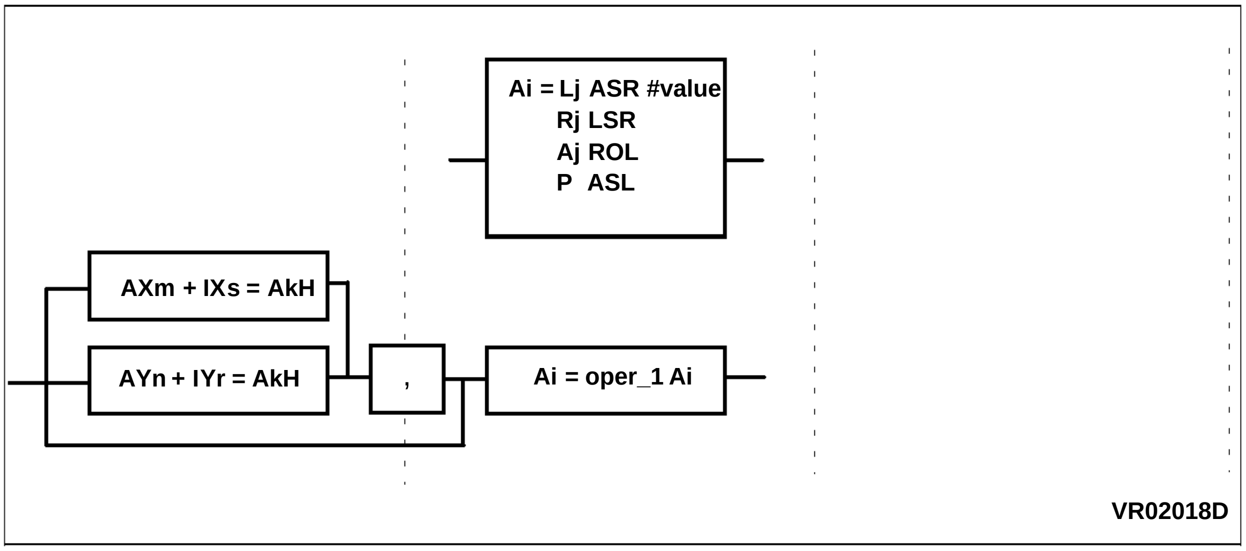


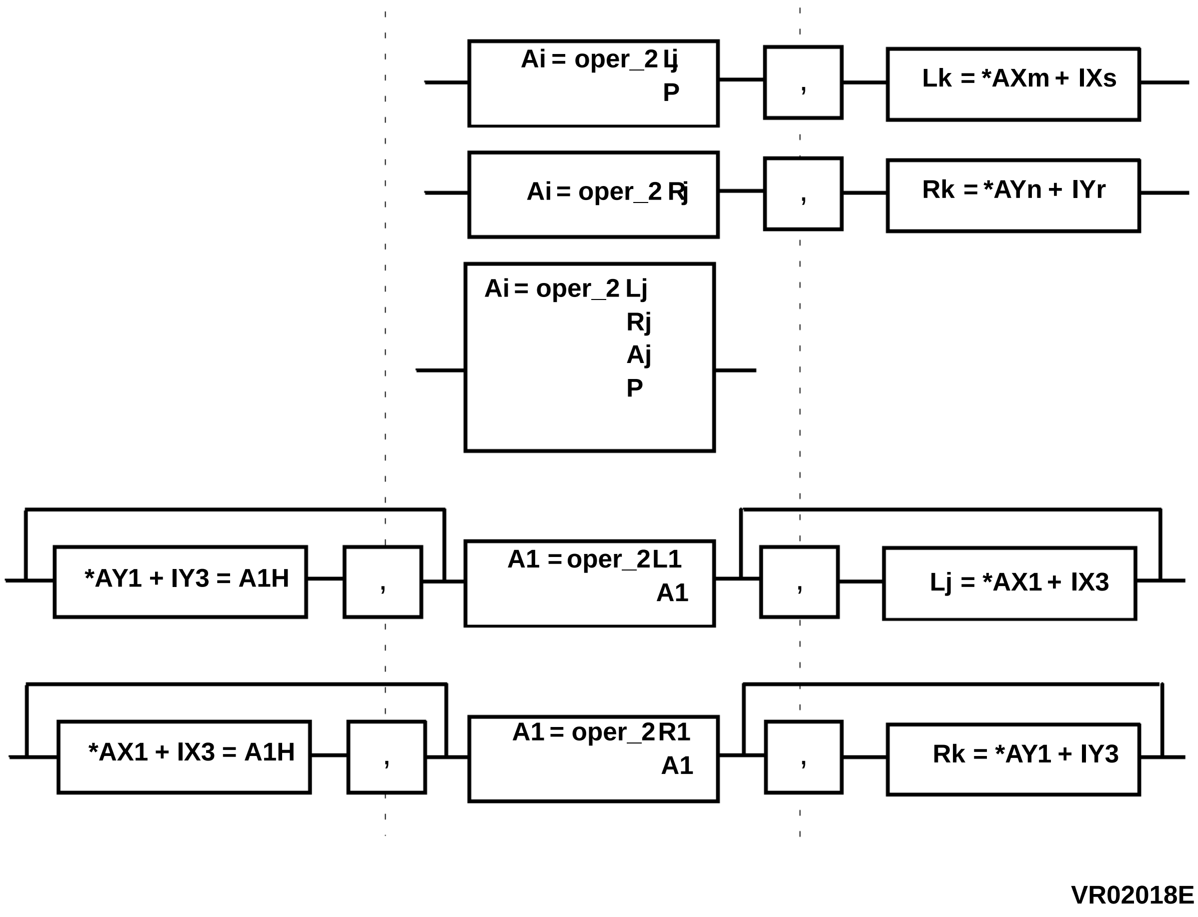
Figure 16. ALU One-Word Operand (oper_1).

Note :	**value**	**=**	**1...32 for ASR**	**value**	**=**	**0...32 for ROL**
	value	**=**	**1...32 for LSR**	**value**	**=**	**0...31 for ASL**

. two-word operand (oper_2)

ABS	**Absolute value**
ASB	**Arithmetical Shift with BSC**
ASL	**1-bit Arithmetic Shift Left**
ASR	**1-bit Arithmetic Shift Right**
ASR16	**16-bit Arithmetic Shift Right**
CHKDIV	**Check Validity of Division**
CMP0	**Compare to 0**
COM	**Logical Complement**
DEC	**Decrement Accumulator**
DECH	**Decrement 24 MSBs of Accumulator**
DIVQ	**One Step of Division**
DIVS	**First Step of Division**
EDGE	**Exponent value of a number**
EQU	**Equal**
INC	**Increment**
INCH	**Increment 24 MSBs of Accumulator**
LSB	**Logical Shift with BSC**
LSL	**1-bit Logical Shift Left**
LSL16	**16-bit Logical Shift Left**
LSR	**1-bit Logical Shift Right**
LSR16	**16-bit Logical Shift Right**
MAX	**Maximum value of determination**
MIN	**Minimum value of determination**
NEG	**Negation**
ROB	**Rotation with BSC (Left or Right)**
ROL	**1-bit Rotation Left**
ROLTST	**1-bit Rotation Left with Test**
ROL16	**16-bit Rotation Left**
RESQ	**Restore Quotient**
RESR	**Restore Remainder**
+=	**Last Step of Positive MAC (PSC used for shift value)**
-=	**Last Step of Negative MAC (PSC used for shift value)**

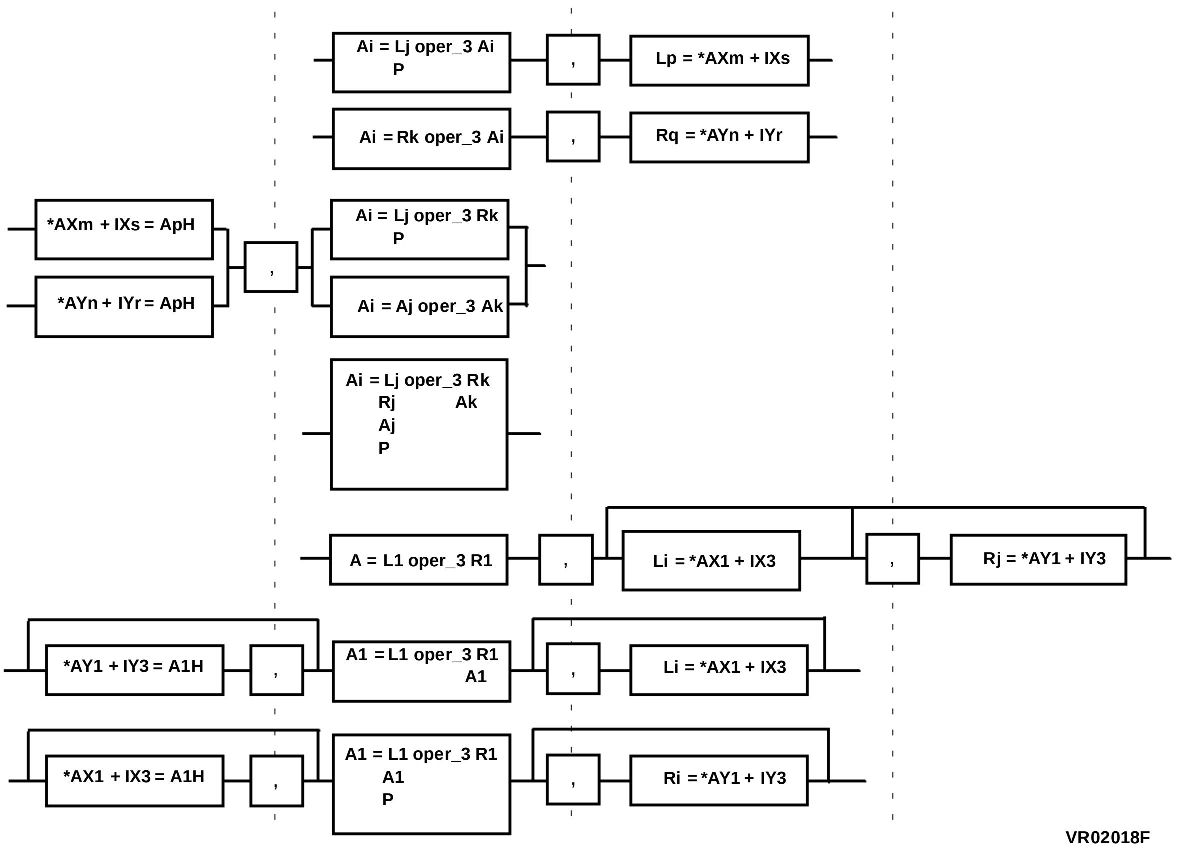
Figure 17. ALU Two-Word Operand (oper_2).



. three-word operand (oper_3)

ADD	**Addition**
ADDC	**Addition with Carry**
ADDS	**Addition with Shift**
AND	**Logical AND**
CMP	**Compare**
CMPS	**Compare Sign**
OR	**Logical OR**
SUB	**Subtraction**
SUBC	**Subtraction with Carry**
SUBR	**Reversed Subtraction**
SUBRC	**Reversed Subtraction with Carry**
SUBRS	**Reversed Subtraction with Shift**
SUBS	**Subtraction with Shift**
XOR	**Exclusive OR**

Figure 18. ALU Three-Word Operand (oper_3).



DMULT Double precision multiplication
MULT Multiplication

Figure 19a. Multiplication.

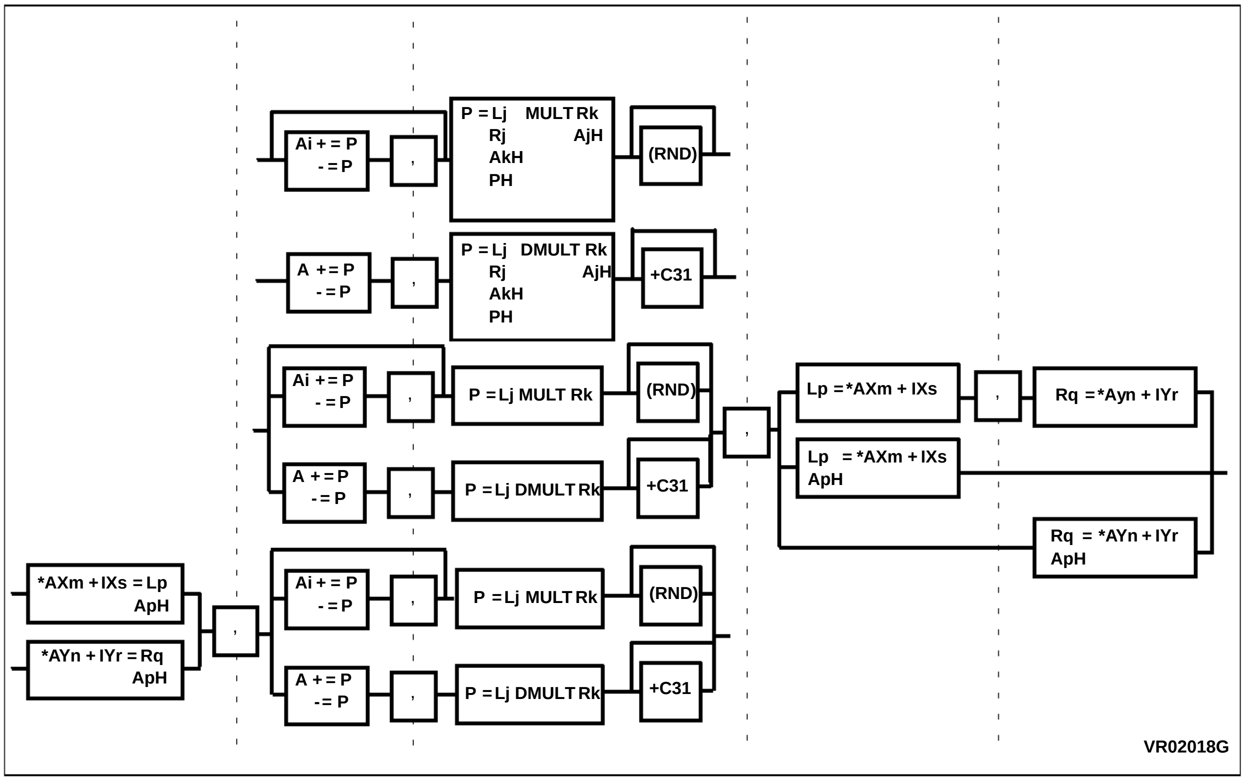
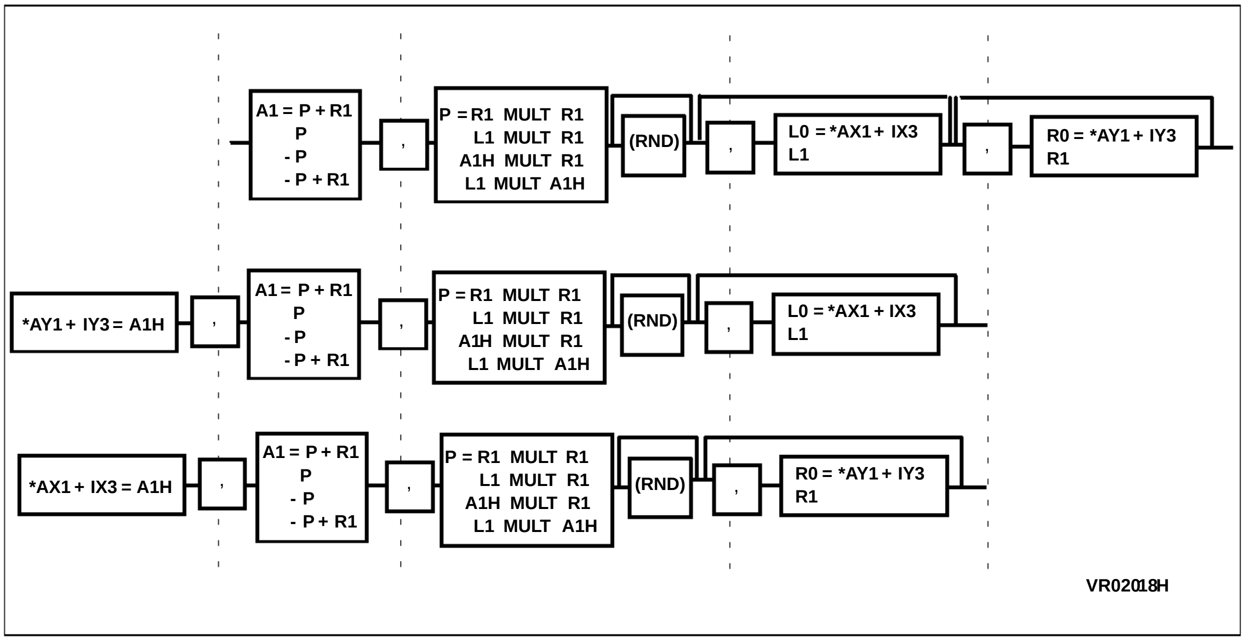
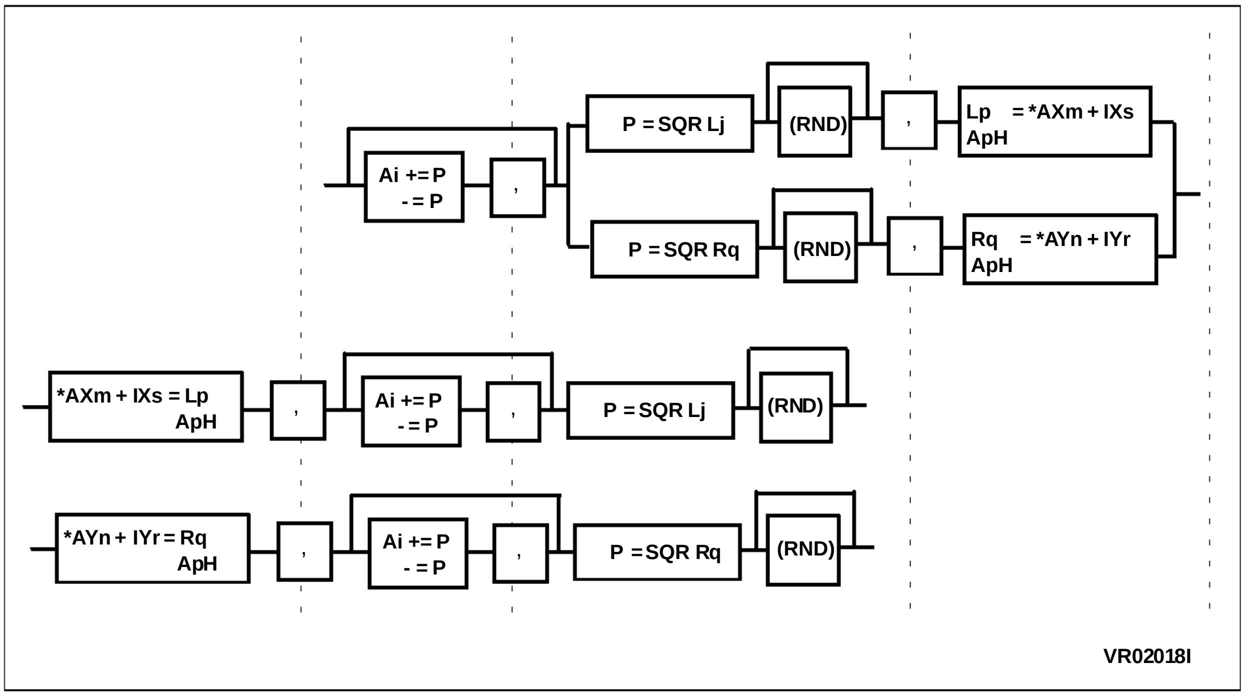


Figure 19b. ALU Multiplication.



SQR Square

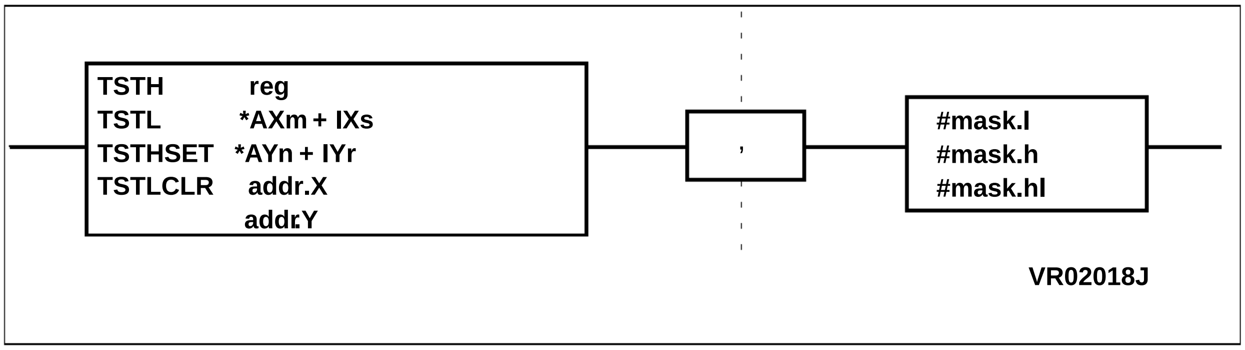
Figure 20. Square.



4.1.2.3 Bit Manipulation

TSTH	**Bit Test High**
TSTL	**Bit Test Low**
TSTHSET	**Bit Test High and Set**
TSTLCLR	**Bit Test Low and Reset**

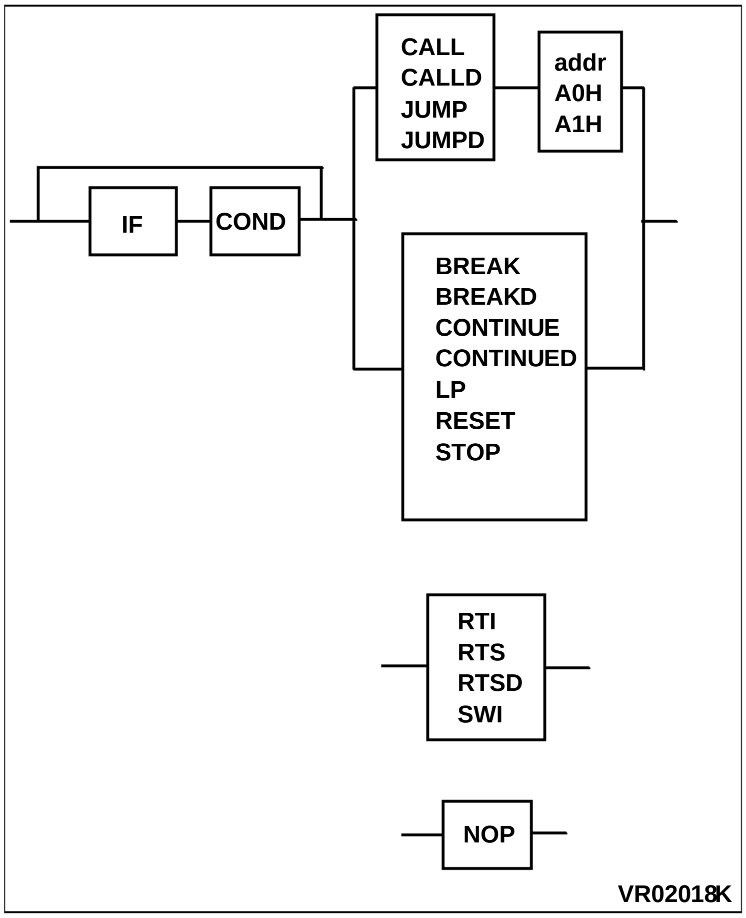
Figure 21. Bit Manipularion.



4.1.2.4 Program control

BREAK	Break
BREAKD	Break Delayed
CALL	Jump to Subroutine
CALLD	Call Delayed
CONTINUE	Continue
CONTINUED	Continue Delayed
JUMP	Jump
JUMPD	Jump Delayed
LP	Low Power
NOP	No operation
RESET	Reset
RTI	Return from interrupt
RTS	Return from subroutine
RTSD	Return from subroutine Delayed
STOP	Stop
SWI	Software interrupt

Figure 22. Program Control.

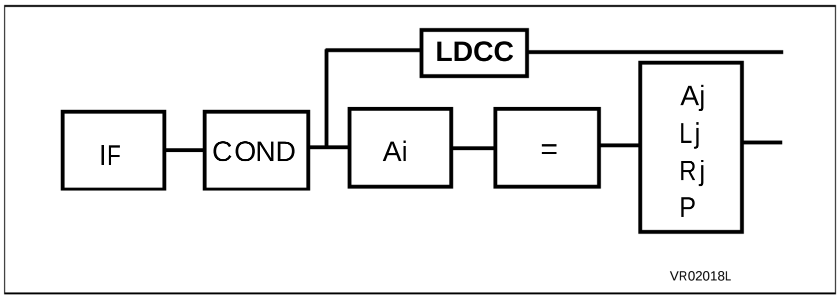


Note **: The condition table is described in § 3.2.6.3**

4.1.2.5 Conditional Assignment instruction

LDCC	**Load conditional**

Figure 23. Conditional Assignment.

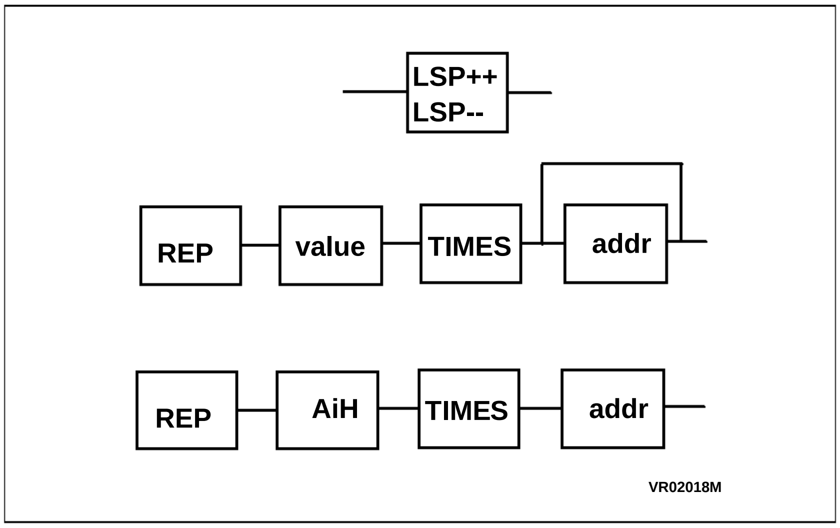


Note **: The condition table is described in § 3.2.6.3**

4.1.2.6 Loop control

REP	**Automatic management of the loop registers (LSC, LE and SP)**
LSP--	**Decrement LSP 2-bit register**
LSP++	**Increment LSP 2-bit register**

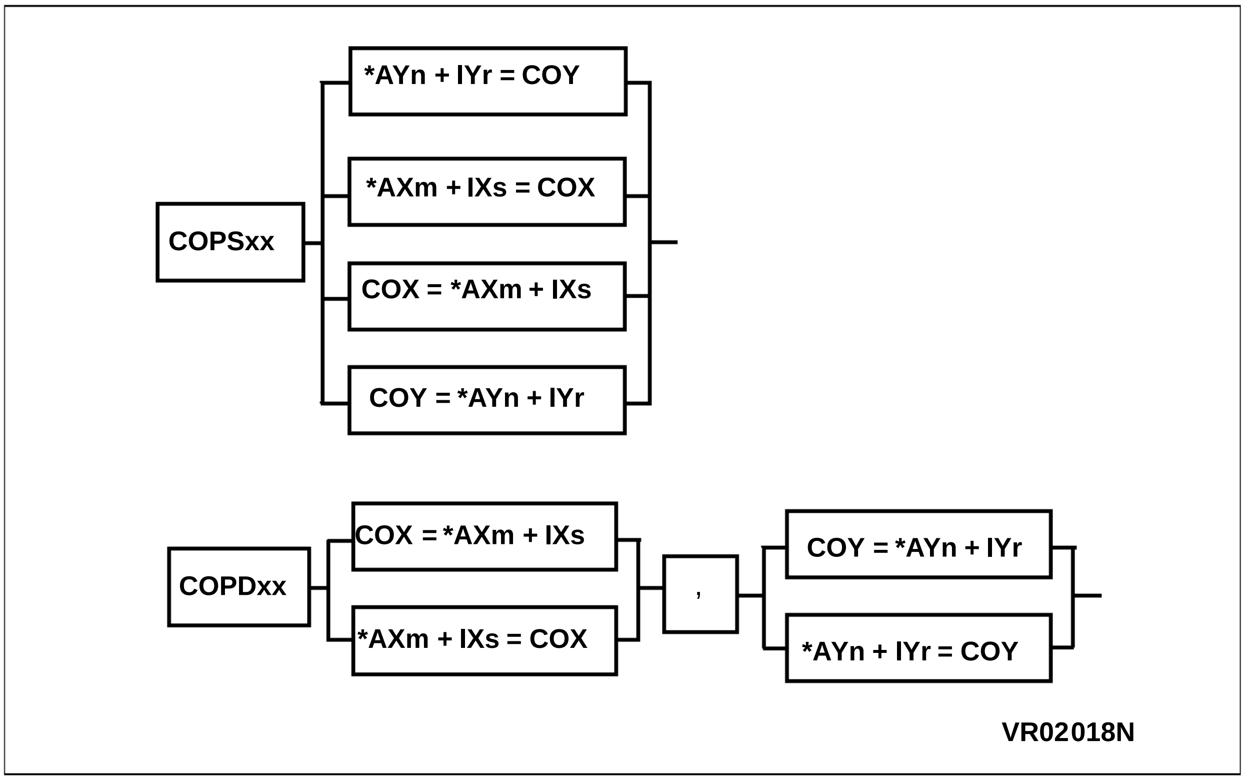
Figure 24. Loop Control.



4.1.2.7 Co-processor

COPD	**Co-processor Double Move**
COPS	**Co-processor Simple Move**

Figure 25. Co-processor.



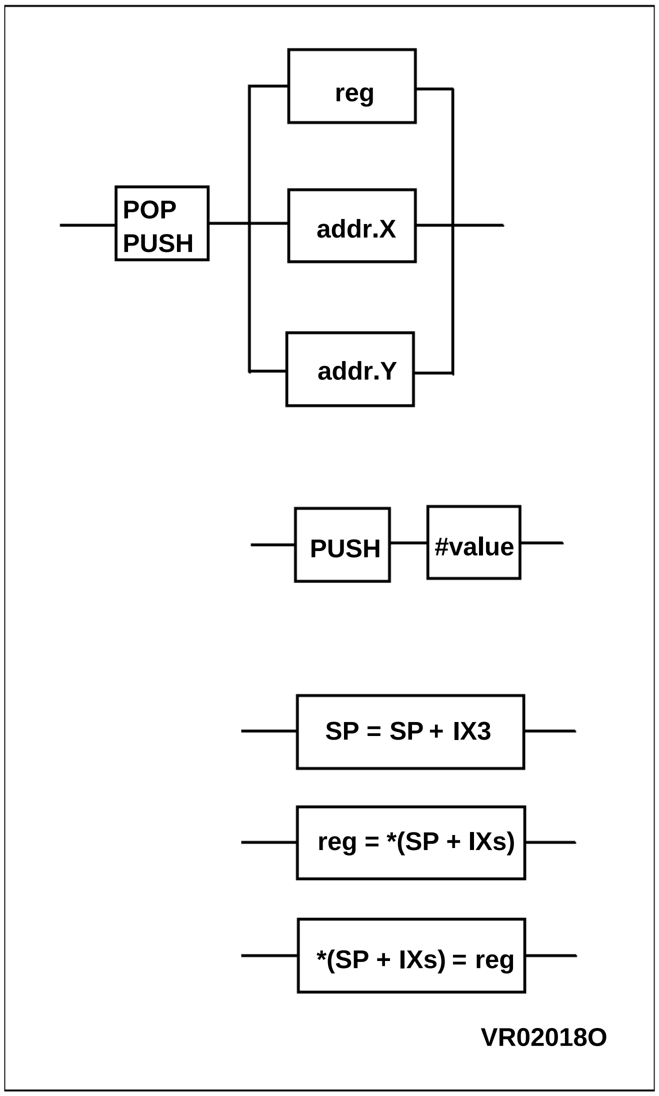
Note **: Increments allowed depend on register used (for COPD instruction only):**

AX0 :	**IX0/IX1**
AX1 :	**IX2/IX3**
AY0 :	**IY0/IY1**
AY1 :	**IY2/IY3**

4.1.2.8 Stack

POP	**Retrieved from Stack**
PUSH	**Saved on the Stack**

Figure 26. Stack.



4.1.3 D950-CORE instruction cycle and instruction word counts

INSTRUCTION WORD & CYCLE NUMBERS

Instruction Group / Subgroup	Words	Cycles
Assignment indirect single	**1**	**1**
Assignment indirect double	**1**	**1**
Assignment direct	**2**	**2**
Assignment indirect indexed	**1**	**2**
Assignment immediate short	**1**	**1**
Assignment immediate long	**2**	**2**
Assignment register to register	**1**	**1**
Assignment register / PRAM	**1**	**4**
ALU 1-word operand	**1**	**1**
ALU 2-word operand	**1**	**1**
ALU 3-word operand	**1**	**1**
MULT	**1**	**1**
DMULT	**1**	**2**
ALU Multiplication	**1**	**1**
SQR	**1**	**1**
Bit Manipulation	**2**	**2**
Program Control Non Delayed	**1**	**3**
Program Control Delayed	**1**	**1 or 2**
Conditional Assignment	**1**	**1**
LDCC	**1**	**2**
REPEAT (single) < 512	**1**	**1**
REPEAT (single) > 512	**2**	**2**
REPEAT (block) < 512	**2**	**2**
REPEAT (block) > 512	**3**	**3**
COPD	**1**	**1**
COPS	**1**	**1**
PUSH / POP register	**1**	**1**
PUSH / POP direct address	**2**	**3**
PUSH immediate value	**2**	**3**
INC SP	**1**	**1**
Register / Stack indirect indexed	**1**	**2**

5 ELECTRICAL SPECIFICATIONS

5.1 DC ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
VDD	**Power Supply Voltage**	**-0.3 / VDD + 0.3**	**V**
VIN	**Input Voltage**	**-0.3 / VDD + 0.3**	**V**
TA	**Operating Temperature Range**	**-40 / +85**	**°C**
TSTG	**Storage Temperature Range**	**-55 / +150**	**°C**

5.2 AC ELECTRICAL CHARACTERISTICS (core level)

Conditions : VDD= 3.3V ± 10 % - Junction temperature : 0°C to 85°C

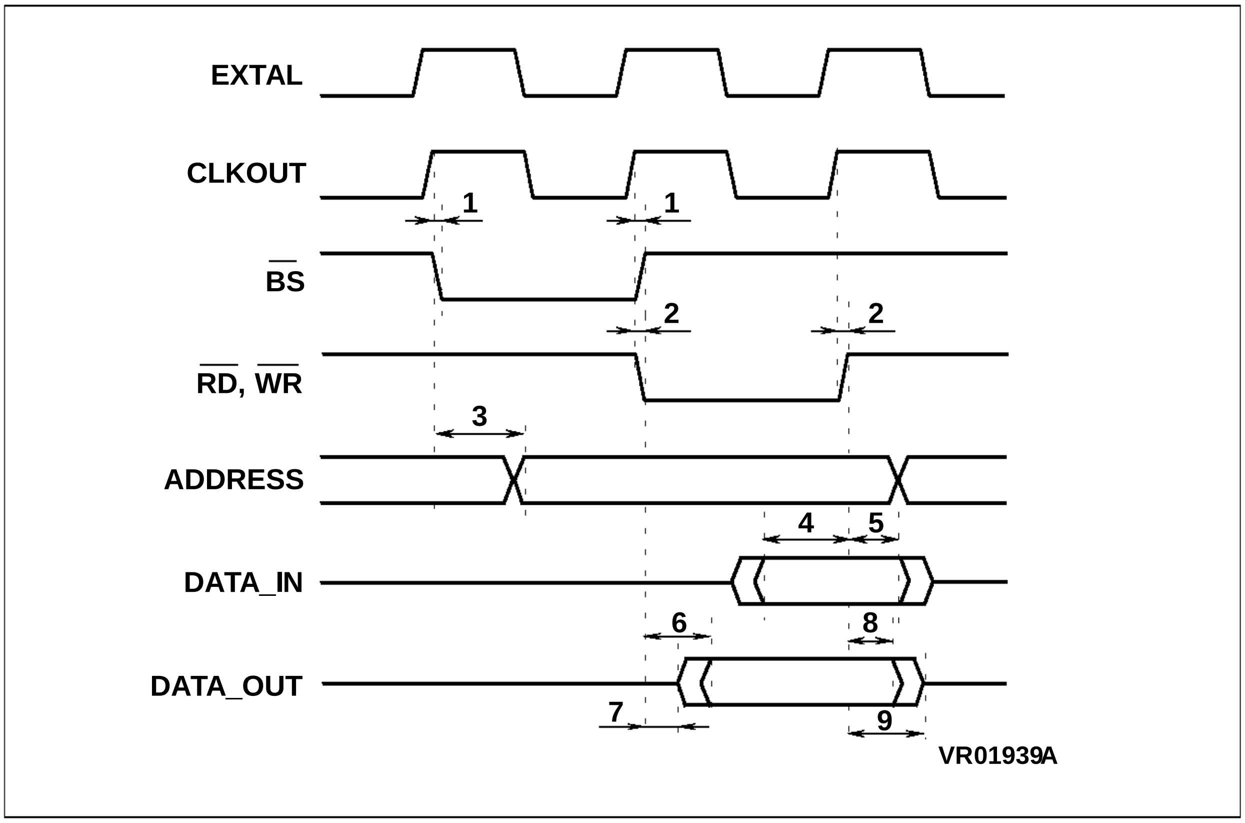
Symbol	Power supply	Min	Typ	Max	Unit
VDD	**Power supply**	**3**	**3**	**4**	**V**
VIL	**Input low level**	**0**			**V**
VIH	**Input high level**			**VDD + 0.3**	**V**
VOL	**Output low level**			**0**	**V**
VOH	**Output high level**	**1**			**V**
IDD	**Operating current**				**mA/MHz**
ILP	**Low Power current**				**mA/MHz**
ISTOP	**Stop current**				**µA/MHz**

5.3 TIMING CHARACTERISTICS

Conditions : VDD = 3.3V ± 10 % - Junction temperature : 0°C to 85°C

5.3.1 Bus AC Electrical characteristics (for X, Y and I buses)

Figure 27. BUS AC Electrical Characteristics (For X, Y and I buses).

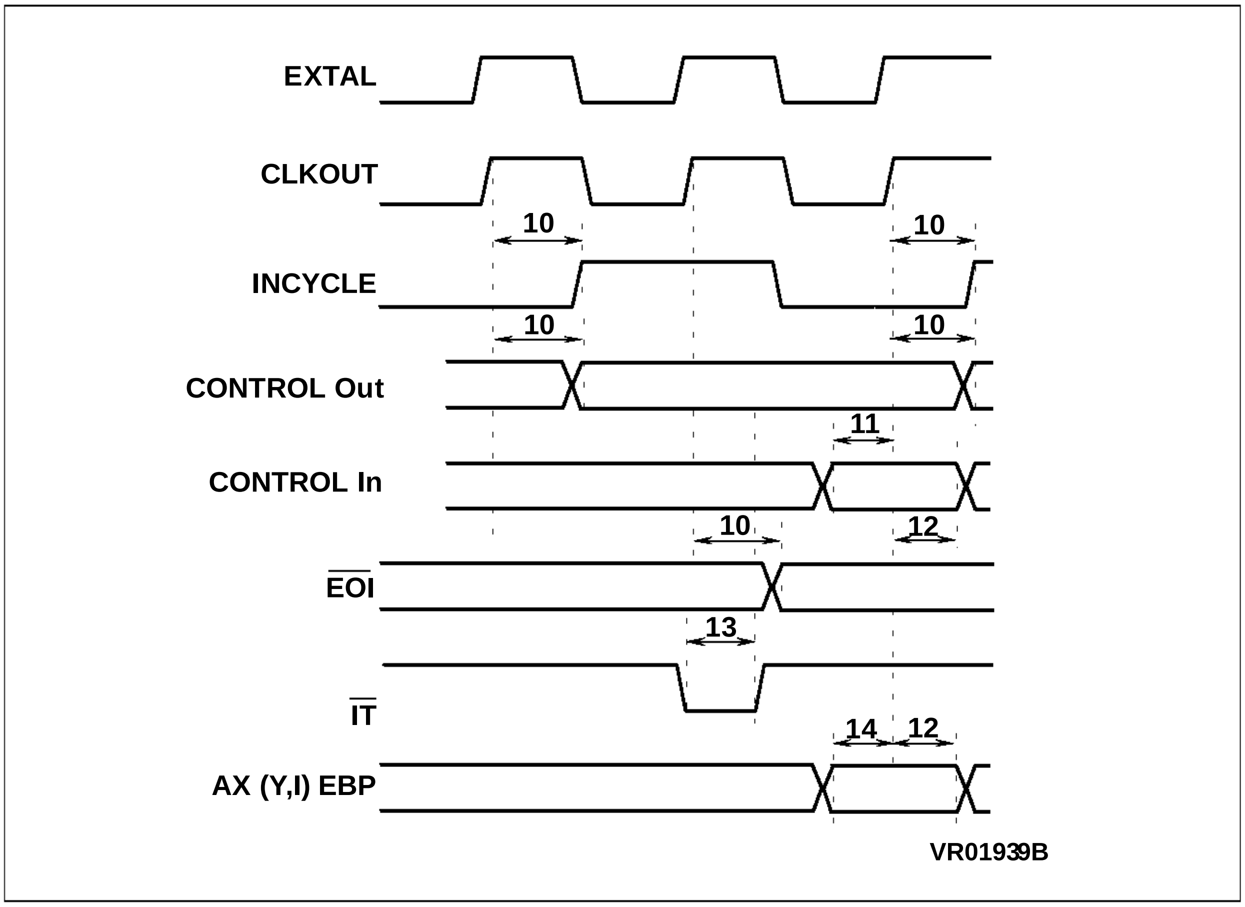


Num	Parameter	Min	Typ	Max.	Unit
T1	**CLKOUT High to BS Low/High**	**0.5**	**1.5**	**3**	**ns**
T2	**CLKOUT Hi. to RD /WR Lo/Hi.**	**0**	**0.5**	**1**	**ns**
T3	**CLKOUT High to Address Valid**	**0.5**	**2**	**4**	**ns**
T4	**DATA_IN Setup to RD High**	**3.5**			**ns**
T5	**DATA_IN Hold from RD High**	**0**			**ns**
T6	**WR Low to DATA_OUT Valid**	**0.5**	**2.5**	**5**	**ns**
T7	**WR Low to DATA_OUT Lo-Z**	**0.5**	**1.5**	**3**	**ns**
T8	**WR High to DATA_OUT invalid**	**0.5**	**1.5**	**3**	**ns**
T9	**WR High to DATA_OUT Hi-Z**	**0.5**	**1.5**	**3**	**ns**

Note **: C_load = 5 pF for all outputs**

5.3.2 Control I/O Electrical characteristics

Figure 28. Control I/O Electrical Characteristics.

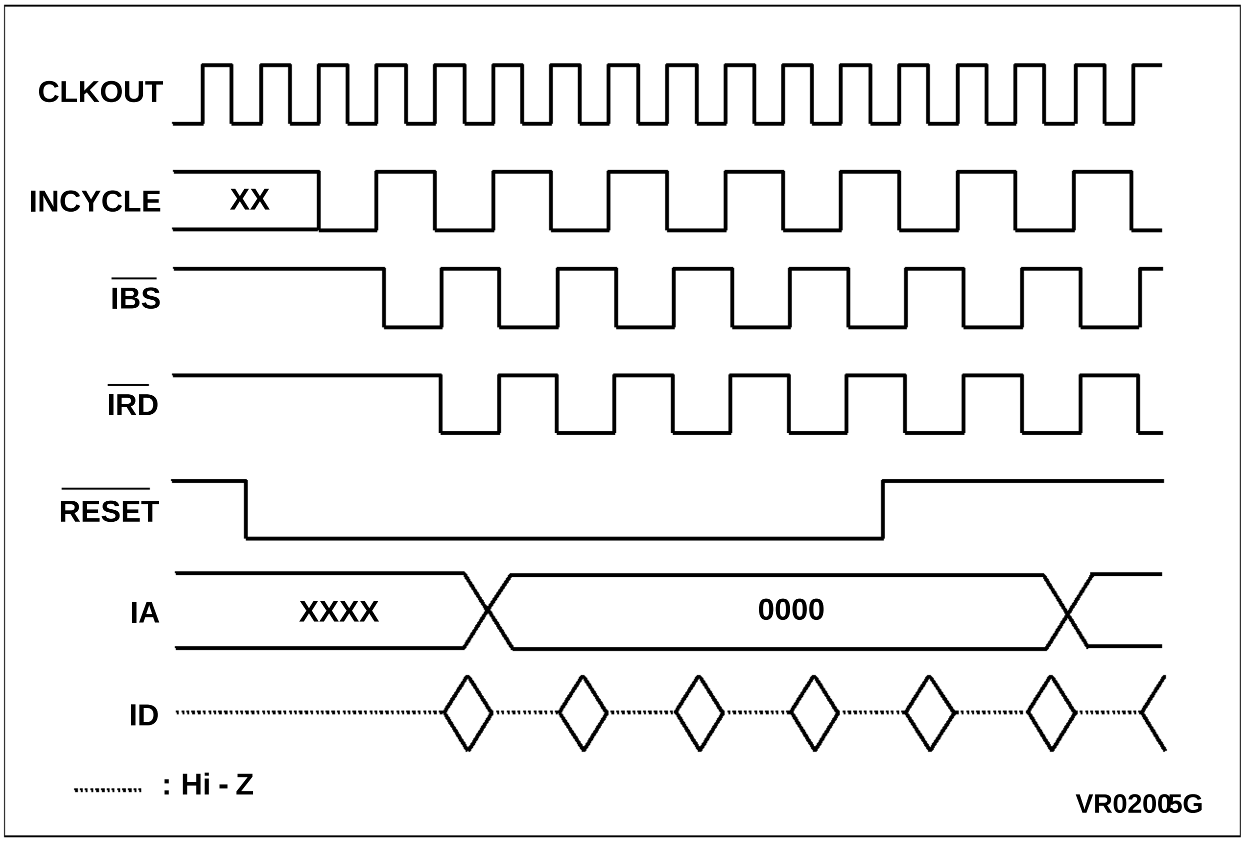


Num	Parameter	Min	Typ	Max.	Unit
T10	**CLKOUT High to CONTROL out valid**	**0.5**	**2.5**	**5**	**ns**
T11	**CONTROL In Setup to CLKOUT High**	**3**			**ns**
T12	**CONTROL In Hold from CLKOUT High**	**1**			**ns**
T13	**$\overline{\text{IT}}$ pulse min. duration**	**5**			**ns**
T14	**AX(Y,I)EBP Setup to CLKOUT High**	**10**			**ns**

Note **: C_load = 5 pF for all outputs**
CONTROL In/Out are those defined in section Pin description (2.6 and 2.7).

5.3.3 Hardware Reset

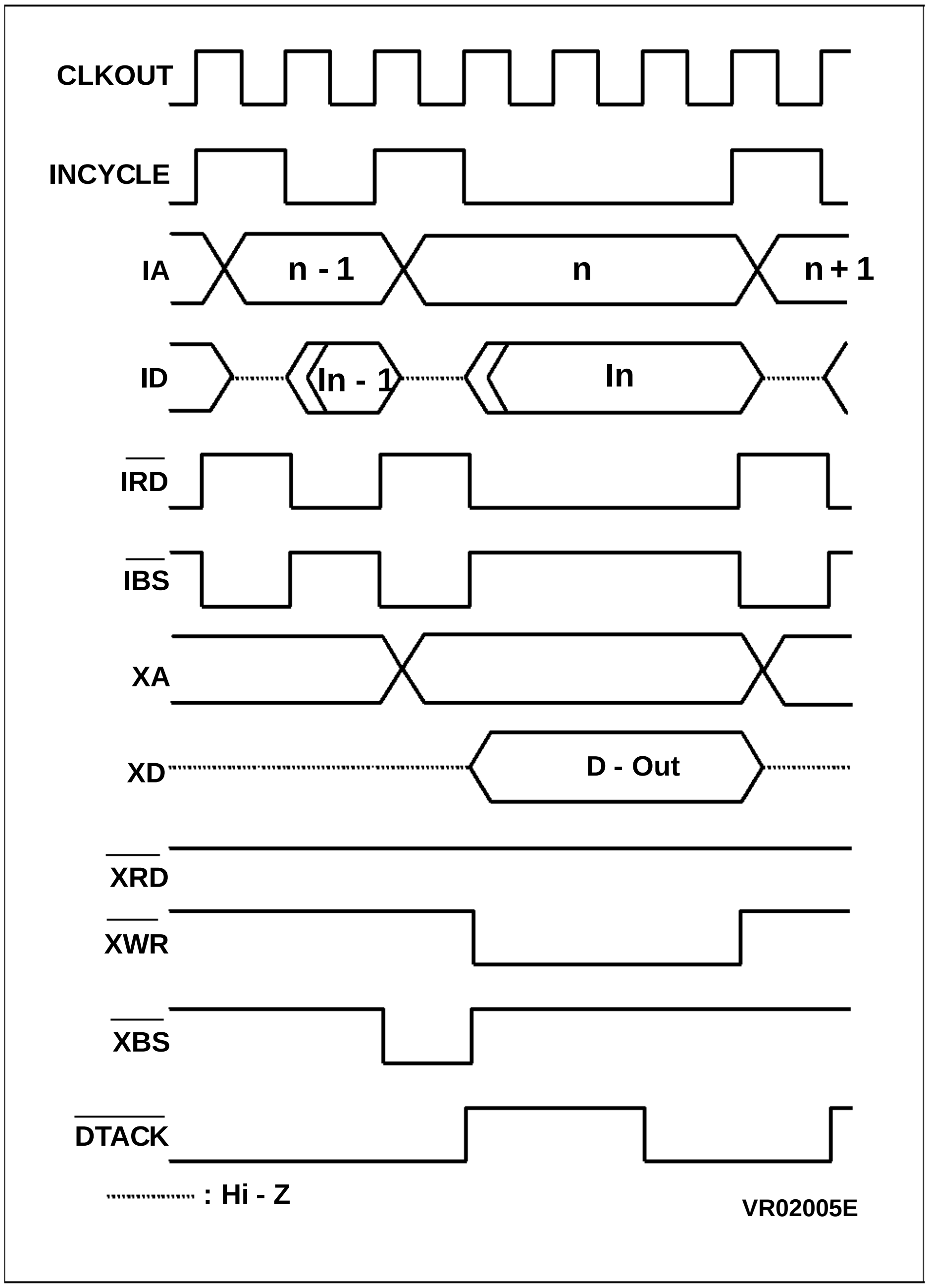
Figure 29. Hardware Reset.



5.3.4 Wait States

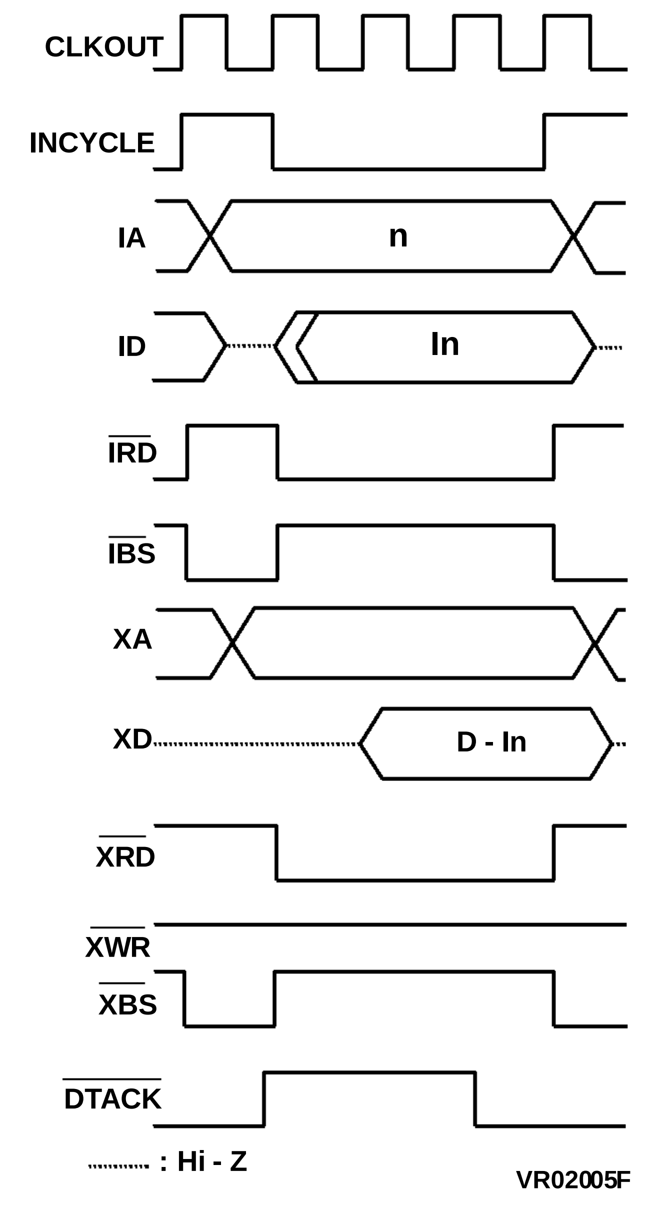
5.3.4.1 Write X-bus with 2 wait-states

Figure 30. Write X-Bus with 2 Wait-States.



5.3.4.2 Read X-bus with 2 wait-states

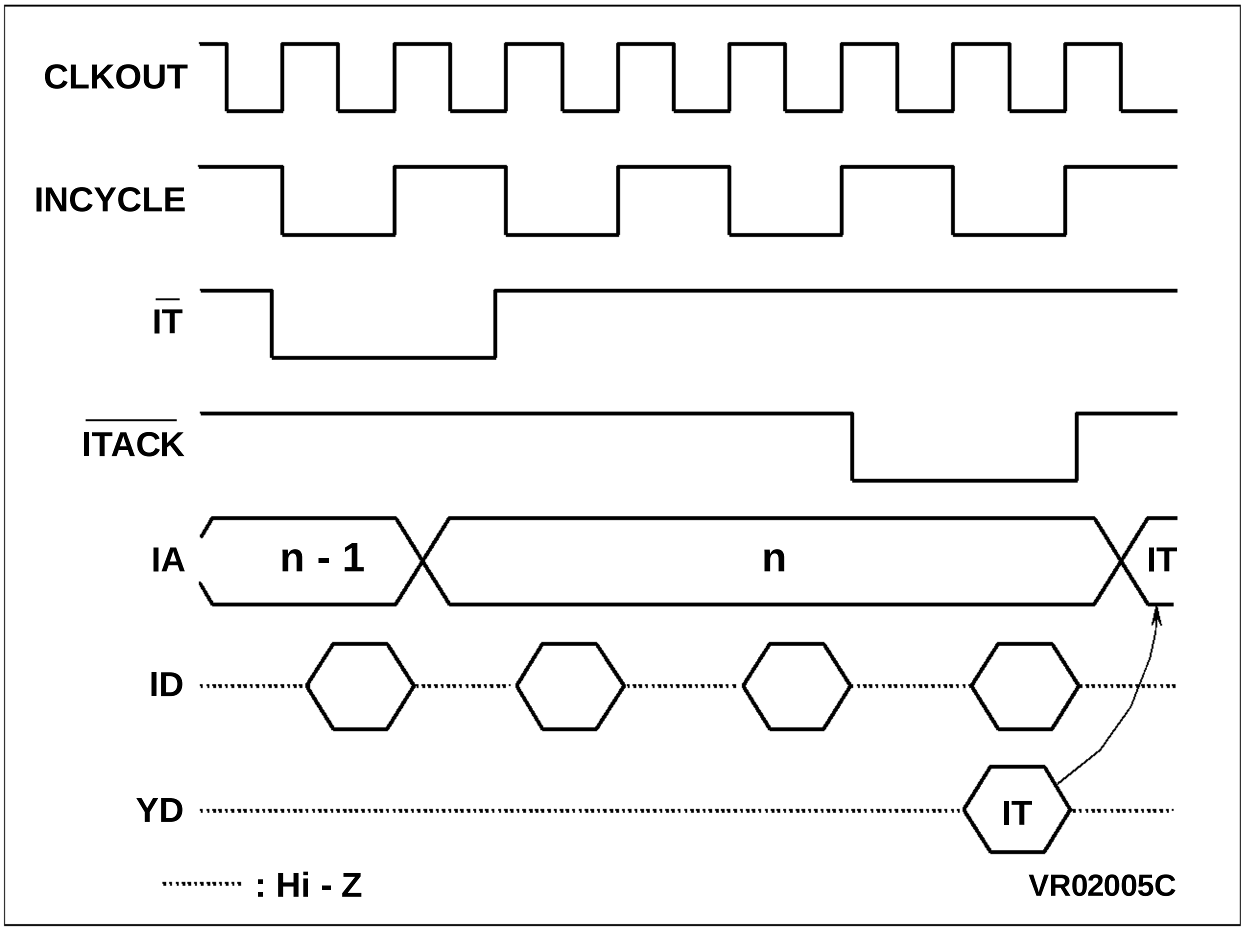
Figure 31. Read X Bus with 2 Wait-States.



5.3.5 Interrupt

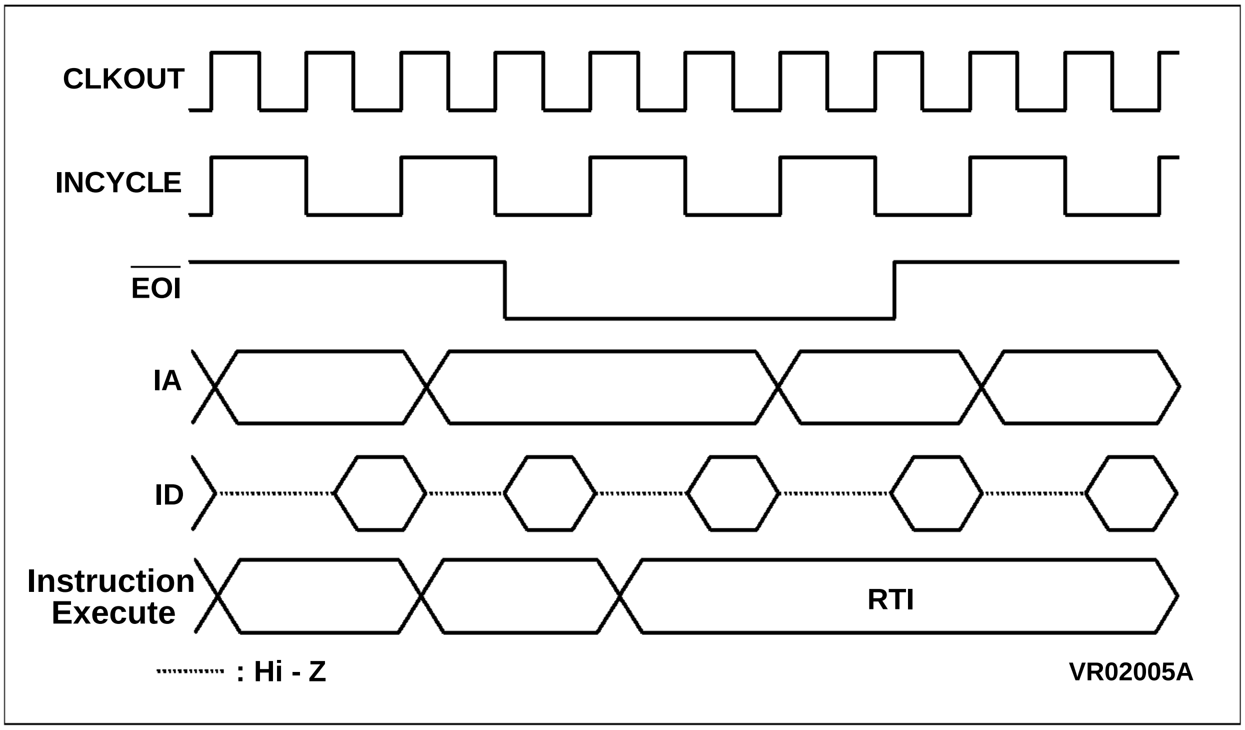
5.3.5.1 Start of Interrupt

Figure 32. Interrupt Start.



5.3.5.2 Return from Interrupt

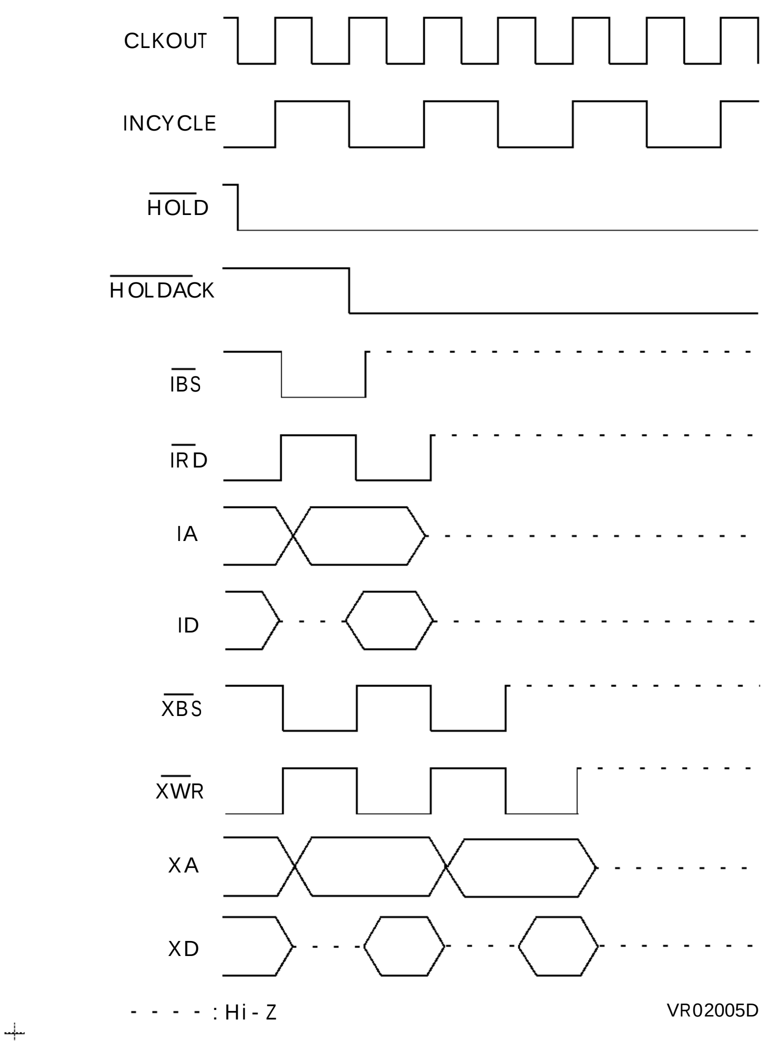
Figure 33. Return from Interrupt.



5.3.6 HOLD

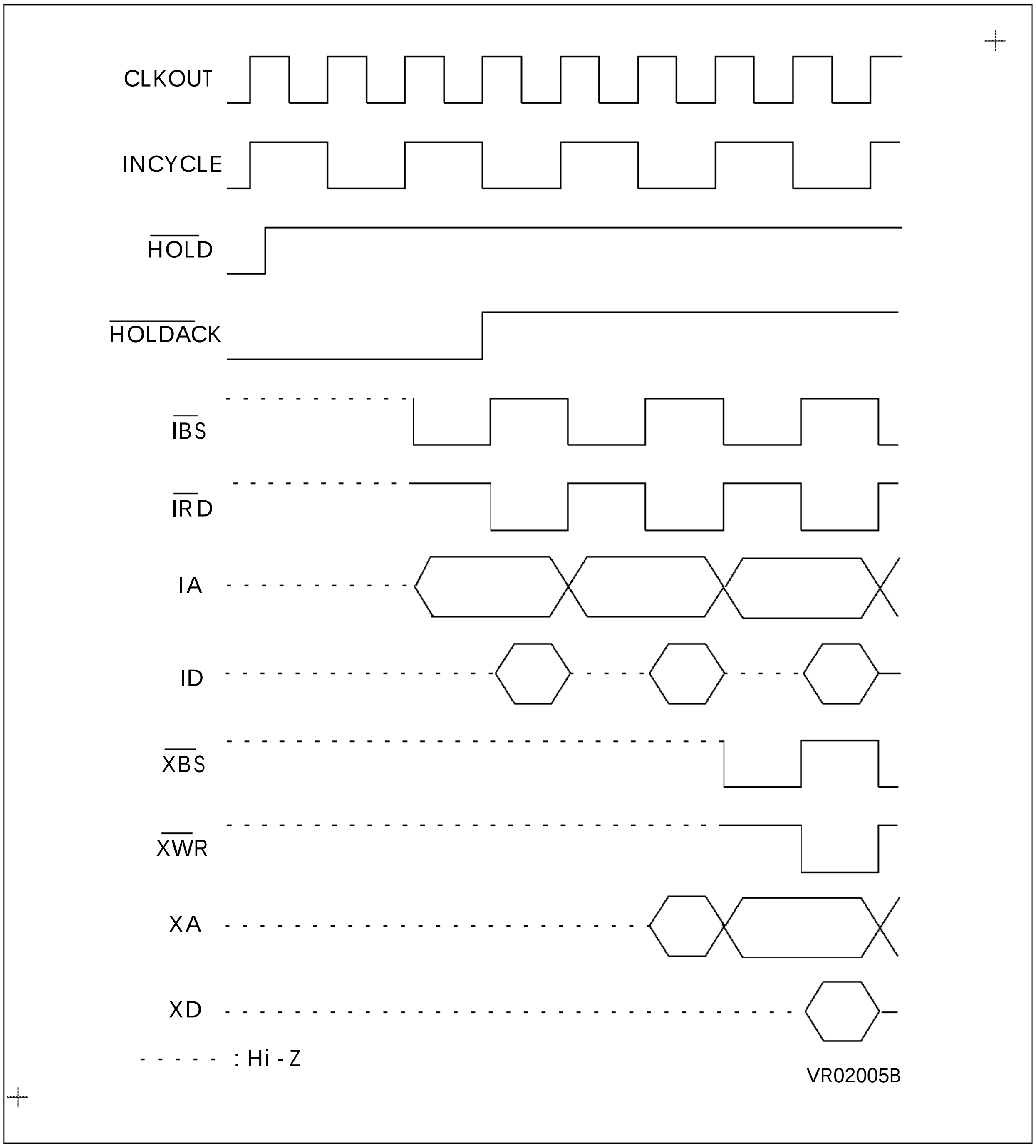
5.3.6.1 HOLD (1)

Figure 34.Hold (1).



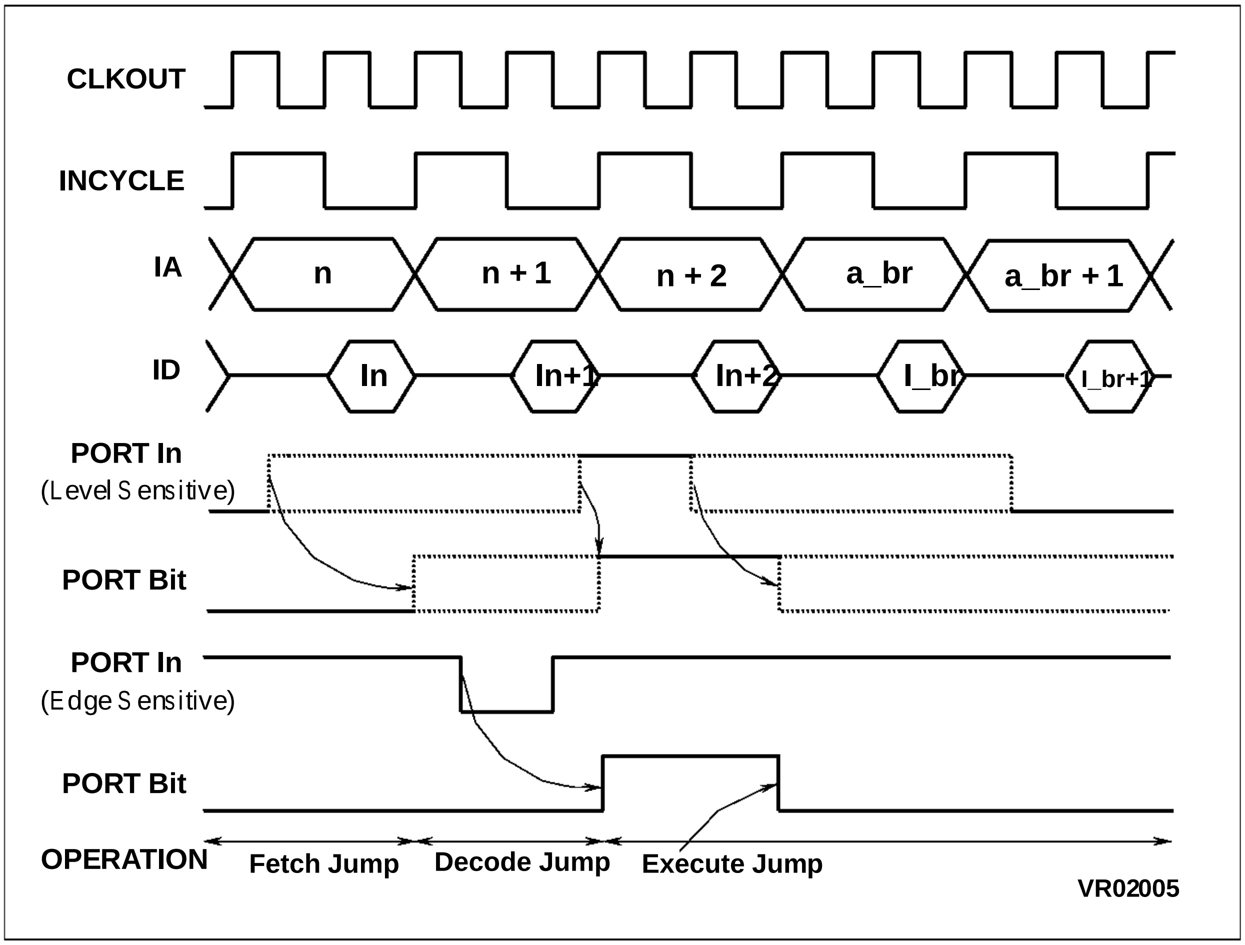
5.3.6.2 HOLD (2)

Figure 35. Hold (2).



5.3.7 JUMP on Port Condition

Figure 36. Jump on Port Condition.



NOTES :

6 ANNEX - HARDWARE PERIPHERAL LIBRARY

Specifications of predined peripheral functions designed for integration with the D950-CORE are given in this chapter. Example of an AS-DSP built around the D950-CORE and these associated peripherals is given at the beginning of this data sheet. Other peripherals are under development. Please, contact your local marketing support for additional information.

These peripherals are :

. Co-processor,
. Bus Switch Unit (BSU),
. Interrupt controller,
. DMA controller.

6.1 CO-PROCESSOR

SGS-THOMSON can design any dedicated co-processor according to customer request.

The D950-CORE instruction set includes two co-processor dedicated one-word instructions (see 4.1.2.6) allowing one (COPS) or two (COPD) parallel data moves between X or Y-memory space and co-processor registers.

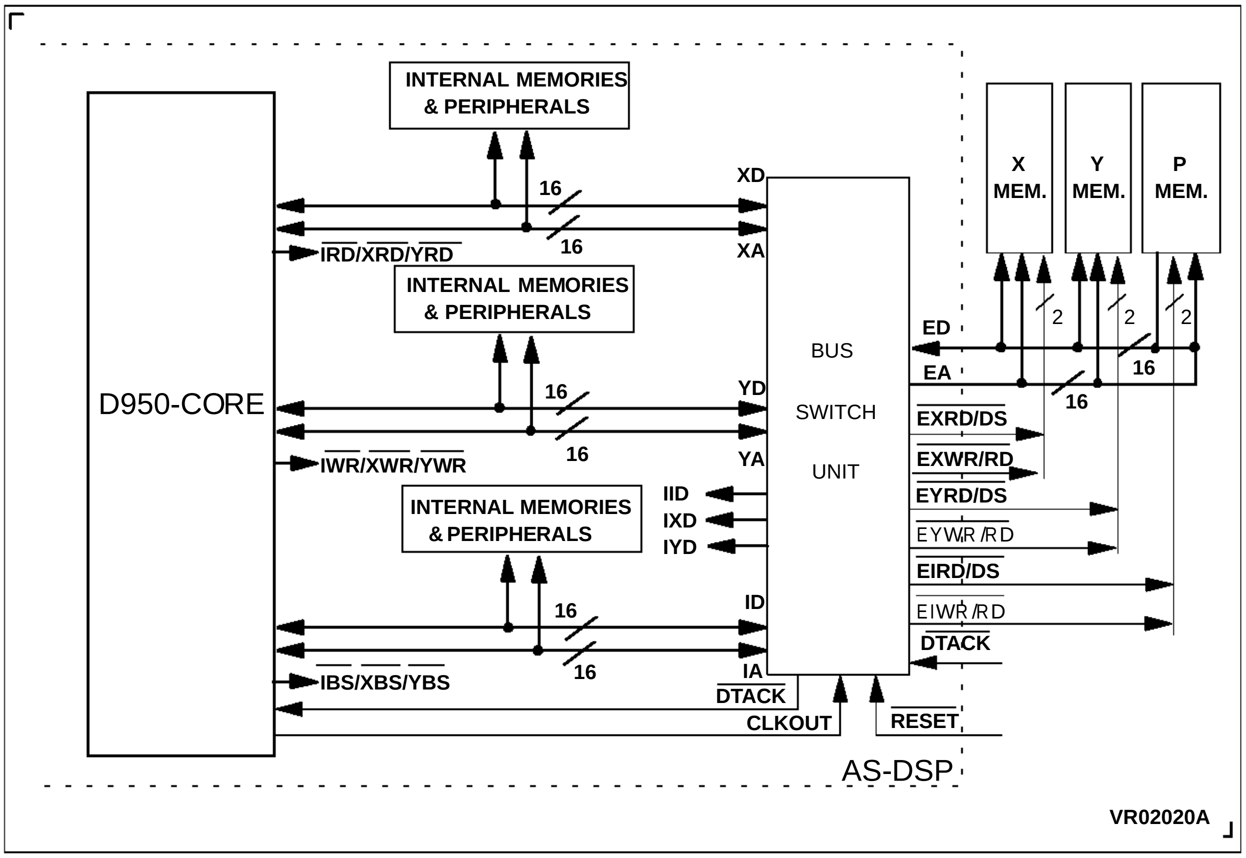
While a co-processor instruction is decoded by the D950-CORE, the VCI output is asserted high, indicating to the co-processor that such an instruction will be executed at the next cycle.

Control and status registers (at least one of each) must be included in the co-processor in order to allow its initialization in various operating modes as well as to return to the D950-CORE information on operations in progress and status when requested.

6.2 BUS SWITCH UNIT (BSU)

The D950-CORE Bus Switch Unit (BSU) enables extension of X, Y and I memories off-chip (see figure) allowing multiple possible configurations for AS-DSP built around the D950-CORE.

Figure 37. D950-CORE Bus Switch Unit.



6.2.1 I/O interface

Refering to the D950-CORE pin description, the BSU I/O interface signals(*) are of two types :

(*) input and output are related to BSU, internal and external are related to AS-DSP.

. on the D950-CORE side
- **IA0/IA15 (I address bus) and ID0/ID15 (I data bus) with their associated control signals :**
  - **. $\overline{\text{IRD}}$ (read / input),**
  - **. $\overline{\text{IWR}}$ (write / input),**
  - **. $\overline{\text{IBS}}$ (bus strobe / input),**
- **XA0/XA15 (X address bus) and XD0/XD15 (X data bus) with their associated control signals :**
  - **. $\overline{\text{XRD}}$ (read / input),**
  - **. $\overline{\text{XWR}}$ (write /input),**
  - **. $\overline{\text{XBS}}$ (bus strobe / input),**
- **YA0/YA15 (Y address bus) and YD0/YD15 (Y data bus) with their associated control signals :**
  - **. $\overline{\text{YRD}}$ (read / input),**
  - **. $\overline{\text{YWR}}$ (write / input),**
  - **. $\overline{\text{YBS}}$ (bus strobe / input),**
- **$\overline{\text{DTACK}}$ (data transfer acknowledge / output),**
- **CLKOUT (clock / input),**

. on the internal memory side
- **IID (internal I-memory space deselect / output),**
- **IXD (internal X-memory space deselect / output),**
- **IYD (internal Y-memory space deselect / output),**

. on the external side
- **EA0/EA15 (external address bus) and ED0/ED15 (external data bus) with their associatedcontrol signals :**
  - **. $\overline{\text{EXRD/DS}}$ (external X-bus read (*) or data strobe (**) / output),**
    $\overline{\text{EXWR/RD}}$ (external X-bus write (*) or read/write (**) / output),
  - **. $\overline{\text{EYRD/DS}}$ (external Y-bus read (*) or data strobe (**) / output),**
    $\overline{\text{EYWR/RD}}$ (external Y-bus write (*) or read/write (**) / output),
  - **. $\overline{\text{EIRD/DS}}$ (external I-bus read (*) or data strobe (**) / output),**
    $\overline{\text{EIWR/RD}}$ (external I-bus write (*) or read/write (**) / output),

(*) INTEL type
(**) MOTOROLA type
- **$\overline{\text{RESET}}$ (reset / input),**
- **$\overline{\text{DTACK}}$ (data transfer acknowledge / input).**

6.2.2 Operation

The BSU recognizes a bus cycle when $\overline{\text{IBS}}$, $\overline{\text{XBS}}$ or $\overline{\text{YBS}}$ is activated. It decodes the address value to determine if an external memory access is requested on the I, X or Y-bus and generates the appropriated signals on the external bus side.

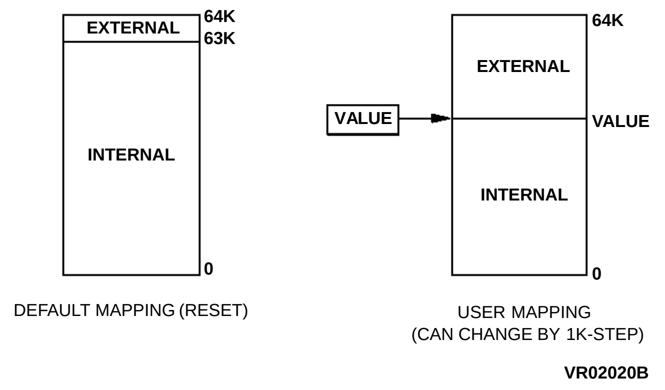
If more than one external memory access are attempted at one instruction cycle, they are serviced sequentially in the following order : I-bus, X-bus, Y-bus.
If one or more external memory access are attempted in read mode, the corresponding internal memory space can be disabled using IID (for I-bus), IXD (for X-bus) or IYD (for Y-bus), assigned low until the end of the instruction cycle.

Each external access requires one basic instruction clock cycle (two CLKOUT cycles), extended by, at least, one wait-state (one CLKOUT cycle). The number of wait-states can be extended either by software with the BSU control registers (see 6.2.3) or by hardware with the two $\overline{\text{DTACK}}$ signals.
During each external memory access and according to the selected interface (INTEL or MOTOROLA) and bus (X, Y or I), the corresponding external control signals are assigned low and synchronized to the rising edge of CLKOUT.

6.2.3 BSU control registers

The BSU is software controlled by three control registers mapped in the Y-memory space defining type of memory used, internal to external boundary address crossing and software wait-states count.

Figure 38. Default and User Mapping Examples.



These control registers include a reference address on bits 4 to 9, where the internal/external memory boundary value is stored (see figure); and software wait-states count on bits 0 to 3, allowing up to 16 wait-states.

External addressing is recognized by comparing these address bits for each valid address from IA, XA and YA to the reference address contained into the corresponding control register.
If the address is greater or equal to the reference value, an external access proceeds.

. XER : X-memory space control register
After reset, XERdefault value is 0x83FF.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM	-	-	-	-	-	XA15	XA14	XA13	XA12	XA11	XA10	W3	W2	W1	W0

IM : Intel / Motorola
0 : Motorola type for memories
1 : Intel type for memories (def.)

XA15 / XA10 : X-memory space Map for boundary on-chip or off-chip

W3 / W0 : Wait state count (1 to 16) for off-chip access (X-memory space)

Note **: - means RESERVED (read : 0, write : don't care)**

. YER : Y-memory space control register
After reset, YERdefault value is 0x83FF.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM	-	-	-	-	-	YA15	YA14	YA13	YA12	YA11	YA10	W3	W2	W1	W0

IM : Intel / Motorola
0 : Motorola type for memories
1 : Intel type for memories (def.)

YA15 / YA10 : Y-memory space Map for boundary on-chip or off-chip

W3 / W0 : Wait state count (1 to 16) for off-chip access (Y-memory space)

Note **: - means RESERVED (read : 0, write : don't care)**

. IER : Instruction memory control register
After reset, IERdefault value is 0x83FF.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM	-	-	-	-	-	IA15	IA14	IA13	IA12	IA11	IA10	W3	W2	W1	W0

IM :Intel / Motorola
0 : Motorola type for memories
1 : Intel type for memories (def.)

IA15 / IA10 : **I-memory space Map for boundary on-chip or off-chip**

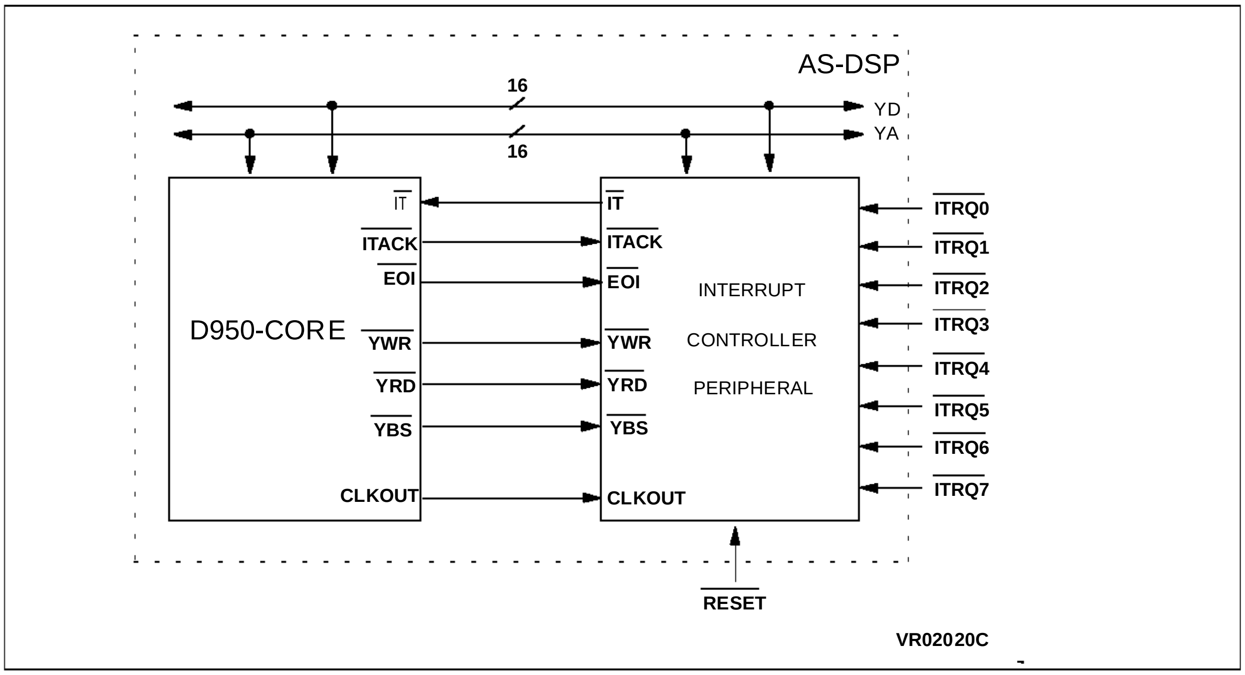
W3 / W0 : **Wait state count (1 to 16) for off-chip access (I-memory space)**

Note **: - means RESERVED (read : 0, write : don't care)**

6.3 INTERRUPT CONTROLLER

The D950-CORE interrupt controller is a peripheral allowing management of up to eight interrupt sources (see figure).

Figure 39. D950-CORE Interrupt Controller Peripheral.



6.3.1 I/O interface

Refering to the D950-CORE pin description, the interrupt controller I/O interface signals$^{(*)}$ are of two types :

(*) input and output are related to interrupt controller, external is related to AS-DSP.

. on the D950-CORE side
- **$\overline{\text{IT}}$ (maskable interrupt request / output),**
- **$\overline{\text{ITACK}}$ maskable interrupt request acknowledge / input),**
- **$\overline{\text{EOI}}$ (end of maskable interrupt routine / output),**
- **YA0/YA15 (Y address bus) and YD0/YD15 (Y data bus) with their associated control signals :**
  - **. $\overline{\text{YRD}}$ (read / input),**
  - **. $\overline{\text{YWR}}$ (write / input),**
  - **. $\overline{\text{YBS}}$ (bus strobe / input),**
- **CLKOUT clock / input),**

. on the external side
- **$\overline{\text{ITRQ}}$ (7:0) (8 interrupt requests / inputs),**
- **$\overline{\text{RESET}}$ (reset / input).**

6.3.2 Interrupt controller peripheral registers

The interrupt controller interface is software controlled by thirteen status/control registers mapped in the Y-memory space. Status registers are not protected against writing :

. IVi : Interrupt vector register (one register associated to each external interrupt)

IVi contains the first address of the interrupt routine associated to each $\overline{ITRQi}$ interrupt input(with 0 i 7). The register content of the interrupt under service is assigned on the YD-bus during the cycle following the $\overline{ITACK}$ falling edge.

After reset, IVi default value is 0.

. IMR : Interrupt Mask register

Each interrupt $\overline{ITRQi}$ can be masked individually when IMi corresponding bit is set. In this case, no activity on $\overline{ITRQi}$ is taken into account.

$\overline{ITRQi}$ can be activated on a low level or on a falling edge, according to associated ISi status.

When associated ISi-bit is reset (def. value), $\overline{ITRQi}$ must stay low until the $\overline{ITACK}$ falling edge to be taken into account.

After reset, IMR default value is 0x5555.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IS7	IM7	IS6	IM6	IS5	IM5	IS4	IM4	IS3	IM3	IS2	IM2	IS1	IM1	IS0	IM0

IMi : Interrupt Mask
0 : Interrupt i is not masked
1 : Interrupt i is masked (def.)

ISi : Sensitivity
0 : $\overline{ITRQi}$ is active on a low level (def.)
1 : $\overline{ITRQi}$ is active on a falling edge

Note **: - means RESERVED (read : 0, write : don't care)**

. IPR : Interrupt Priority register

IPR contains the priority level of each $\overline{ITRQi}$ interrupt input.

Interrupt priority level is a 2-bit value, so can be 0,1,2 or 3 (0 lowest priority, 3 highest priority).

When two $\overline{ITRQi}$ of same priority level are requested during the same cycle, the first acknowledged interrupt is the interrupt corresponding to the lowest number.

After reset, IPR default value is 0.

15 - 14	13 - 12	11 - 10	9 - 8	7 - 6	5 - 4	3 - 2	1 - 0
IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0

IPi : Interrupt Priority level (0, 1, 2 or 3) (def. 0)

Note **: - means RESERVED (read : 0, write : don't care)**

. ICR : Interrupt Control register

ICR displays the current priority level and up to four stacked priority levels.

The current priority level is coded using 3 bits but only five different values are available :

PRIORITY LEVEL	CODING	ACCEPTABLE IT LEVEL PRIORITY
- 1	111	0,1,2,3
0	0	1,2,3
1	1	2,3
2	10	3
3	11	
Reserved	100 - 110	

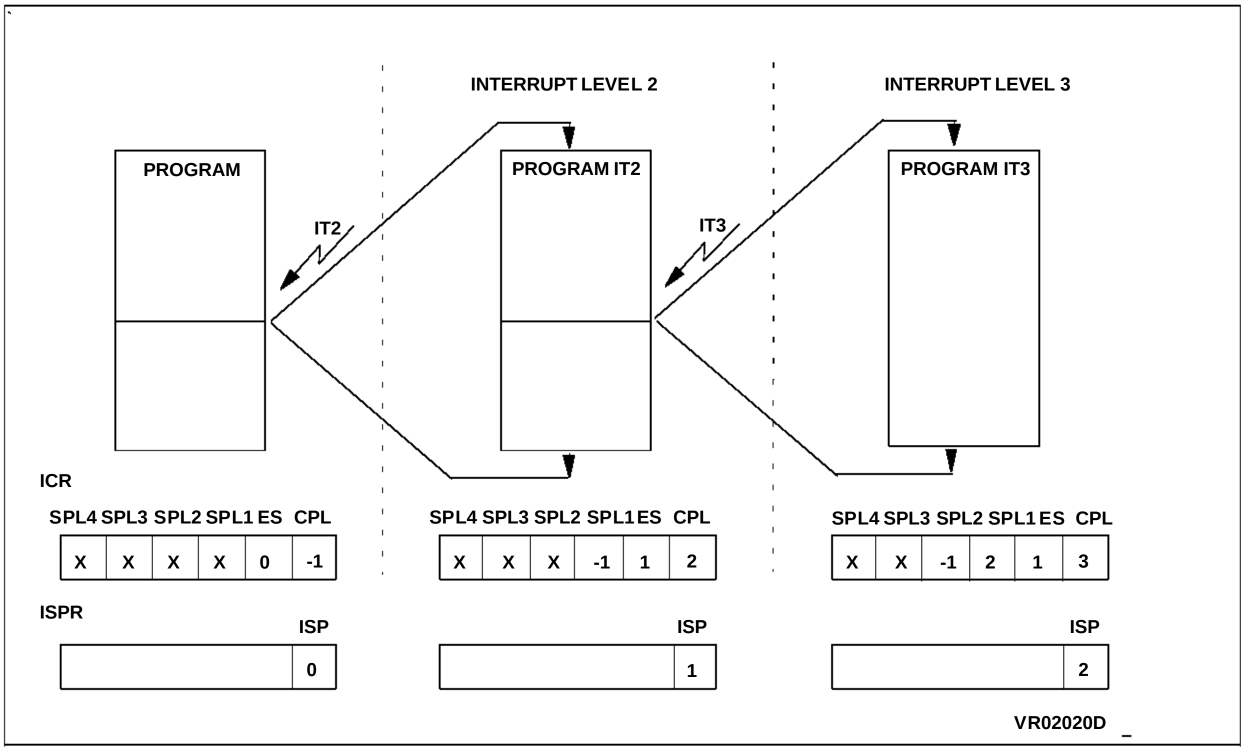
Note **: The D950-CORE interrupts (SWI, RESET) are priority level 4 (highest level).**
An interrupt request is acknowledged when its priority level is strictly higher than the current priority level. In this case, the current priority level becomes the interrupt priority level and the previous current priority level is pushed onto the stack and displayed as SPL1.
The process is repeated over a range of four interrupt requests and the four previous current priority levels are displayed as SPL1, SPL2, SPL3 and SPL4. If less than four interrupts are pushed onto the stack, the unused Stack Priority Level words are reset to ”000”.
At the end of the interrupt routine, the priority levels are popped from the stack.
If the SPLi values are directly written, the register content is not more significant but the interrupt routine procedure is not affected. The only way to affect this is to reset the AS-DSP.
After reset, ICR default value is 0x000B.

15 - 14 - 13	12 - 11 - 10	9 - 8 - 7	6 - 5 - 4	3	2 - 1 - 0
SPL4	SPL3	SPL2	SPL1	ES	CPL

SPL4 :	**3-bit 4th stacked priority level**
SPL3 :	**3-bit 3rd stacked priority level**
SPL2 :	**3-bit 2nd stacked priority level**
SPL1 :	**3-bit 1st stacked priority level**
ES :	**Empty Stack flag** **0 : Stack is used** **1 : Stack is not used (def.)**
CPL :	**Current Priority level (-1, 0, 1, 2 or 3) (def. 011)**

Notes **: After reset, no interrupt request from interrupt controller is acknowledged.**
- means RESERVED (read : 0, write : don’t care)

Figure 40. ICR and ISPR Operation.



. ISPR Interrupt Stack Pointer Register

ISPR contains the number of stacked priority levels.

If the ISPR value is directly written, the SPLi/CPL values are modified. So the ICR register content is no longer significant but the interrupt routine procedure is not affected.

After reset, ISPR default value is 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2 - 1 - 0
-	-	-	-	-	-	-	-	-	-	-	-	-	ISPR

. ISPR : Number of stacked priority levels (0, 1, 2 or 3)

Note **: - means RESERVED (read : 0, write : don't care)**

. ISR Interrupt Status register

ISR contains the eight interrupt pending bits, each being associated to one $\overline{\text{ITRQi}}$ interrupt input.

IPEi-bit is set when the interrupt request is recorded and is reset when the interrupt request is acknowledged ($\overline{\text{ITACK}}$ falling edge).

An interrupt request will not be acknowledged when IPEi-bit is reset by direct register write.

An interrupt request will be generated whatever the state of $\overline{\text{ITRQi}}$ when IPEi-bit is set by a direct register write.

When only some pending interrupt requests need to be acknowledged, the IPEi bits of the other $\overline{\text{ITRQi}}$ interrupt inputs must be reset.

When none of the pending interrupt requests need to be acknowledged, the IPE-bit of CCR registermust be first reset and ISRmust be reset.
When IMi-bit is set, the corresponding IPEi-bit is reset.
After reset, ISRdefault value is 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	IPE7	IPE6	IPE5	IPE4	IPE3	IPE2	IPE1	IPE0

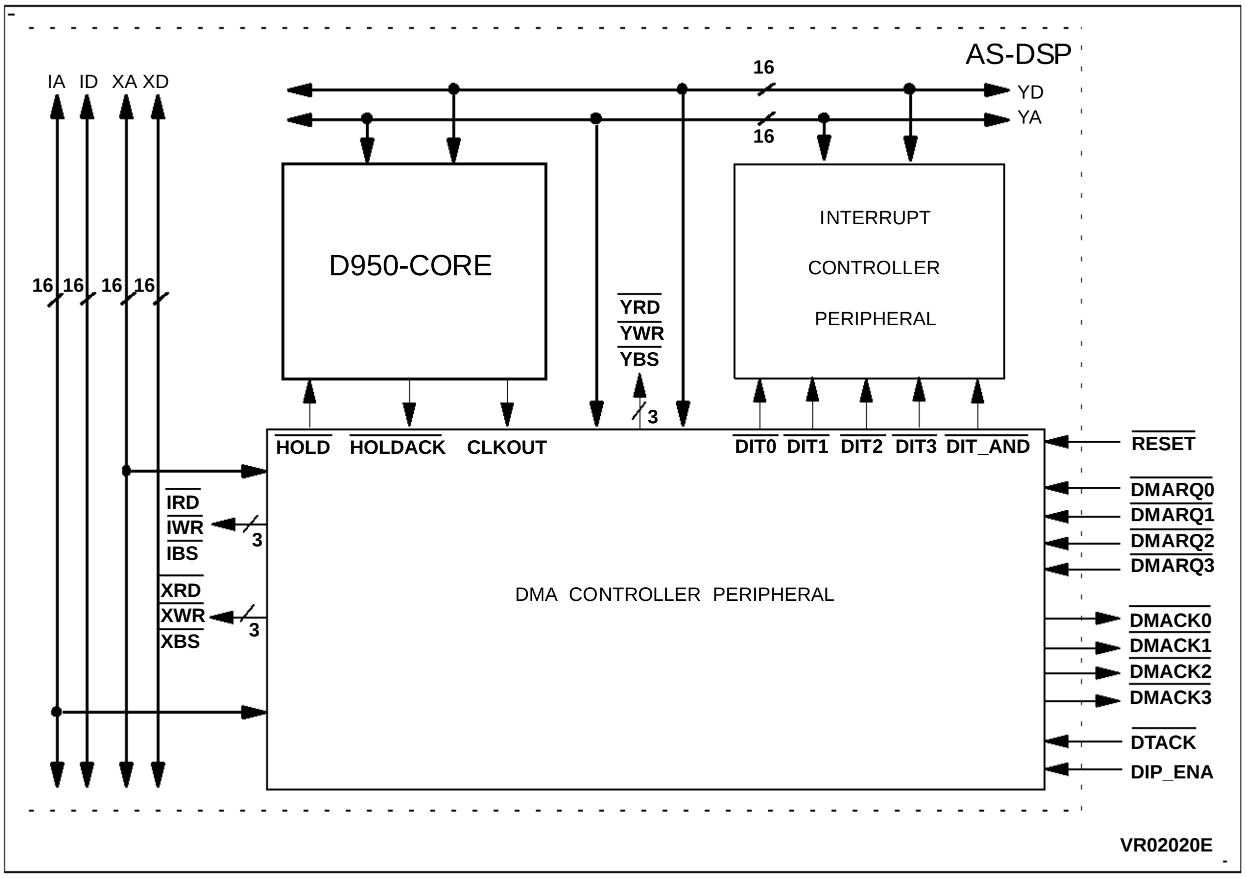
IPEi : Interrupt Pending bit
0 : Reset when interrupt request is acknowledged (def.)
1 : Set when interrupt request is recorded

Note **: - means RESERVED (read : 0, write : don't care)**

6.4 DMA CONTROLLER

The D950-CORE DMA controller allows data transfer management between AS-DSP memories and external peripherals, without using AS-DSP capabilities (see figure).

Figure 41. D950-CORE DMA CONTROLLER Peripheral.



6.4.1 I/O interface

Refering to the D950-CORE pin description, the DMA controller I/O interface signals[*] are of different types:

(*) input and output are related to DMA controller, internal and external are related to AS-DSP.

. on the D950-CORE side
- **$\overline{\text{HOLD}}$ (hold request / output),**
- **$\overline{\text{HOLDACK}}$ (hold acknowledge / input),**
- **CLKOUT clock / input).**

. on the internal memory side
- **XA0/XA15 (Y address bus) with associated control signals :**
  - **. $\overline{\text{XRD}}$ (read / output),**
  - **. $\overline{\text{XWR}}$ (write / output),**
  - **. $\overline{\text{XBS}}$ (bus strobe / output),**
- **YA0/YA15 (Y address bus) and YD0/YD15 (Y data bus) with their associated control signals :**
  - **. $\overline{\text{YRD}}$ (read / input/output),**
  - **. $\overline{\text{YWR}}$ (write / input/output),**
  - **. $\overline{\text{YBS}}$ (bus strobe / output),**
- **IA0/IA15 (I address bus) with associated control signals :**
  - **. $\overline{\text{IRD}}$ (read / output),**
  - **. $\overline{\text{IWR}}$ (write / output),**
  - **. $\overline{\text{IBS}}$ (bus strobe / output),**

. on the interrupt controller side
- **$\overline{\text{DIT(3:0)}}$ (interrupt request / output),**
- **$\overline{\text{DIT\_AND}}$ (common interrupt request / output),**

. on the external side
- **$\overline{\text{DMARQ(3:0)}}$ (request / one input per channel),**
- **$\overline{\text{DMACK(3:0)}}$ (acknowledge / one output per channel),**
- **DIP_ENA (DITi output enable / input),**
- **$\overline{\text{RESET}}$ (reset / input),**
- **$\overline{\text{DTACK}}$ (cycle extension / input),**

6.4.2 Operation

The DMA controller interface contains (see figure) four independant channels allowing data transfer on I-memory space and simultaneous data transfer on X and Y-memory spaces. When requests occur at the same time on different channels to transfer data on the same bus, the requests are concatenated to be acknowledged during the same transfer, according to the following fixed priority (see table) :

Priority	Channel	Level
0	**0**	**Highest**
1	**1**	↑
2	**2**	
3	**3**	**Lowest**

The DMA transfer is based on DSP cycle stealing operation :

. the DMA Controller generates a "hold request" to the AS-DSP,

. the AS-DSP sends back a "hold acknowledge" to the DMA controller and enters the hold state (bus released, clock stopped),

. the DMA controller manages the data transfer and enters its idle state at the end of the transfer until reception of a new DMA request. "Hold request" signal is removed and AS-DSP clock restarts.

The data transfer duration is n+2 cycles, splitted into :

. one cycle inserted at the beginning of the transfer when bus controls are released by the D950-CORE, n cycles for the number of data words to be transferred, another cycle is inserted at the end of the transfer when bus controls are released by the DMA controller.

Data to be transferred can be single or block. The "DMA request" signal is well adapted to such data transfers by being either edge (single) or level (block) sensitive. Nevertheless, data block can be transferred one data at one time using an edge sensitive request signal.

A double buffering mechanism is available to deal with data blocks requiring the allocation of 2N addresses for the transfer of a N data block.

An interrupt can be used to warn AS-DSP that a predefined number of data has been transferred and is ready to be processed. Interrupt requests are sent from the DMA controller to the Interrupt controller. The selected channels must be edge sensitive and the user has to define the proper priority.

There are two ways to connect the DMA and the Interrupt controllers, according to DIP_ENA static pin :

. either connect each DMA channel interrupt request ($\overline{\text{DITi}}$, active on falling edge) to each interrupt input ($\overline{\text{ITRQi}}$) (DIP_ENA = 0),

. or, if the number of available interrupt sources in the Interrupt controller is low, connect the logical AND of the $\overline{\text{DITi}}$ signals ($\overline{\text{DIT\_AND}}$) to a single interrupt input (ITRQi)

(DIP_ENA = 1), the interrupt pending bits (DIPi) of the DAIC register allowing to distinguish the four possible interrupt sources (see 6.4.3).

6.4.3 DMA controller peripheral registers

. Address registers

Two 16-bit registers (unsigned) are dedicated per channel for transfer address :

- DIA : initial address

This register contains the initial address of the selected address bus (see DBC-bit of DGC).

- DCA : current address

This register contains the value to be transferred to the selected address bus (see DBC-bit of DGC)during the next transfer.

The different DCA values are :

RESET	DAI	DLA	DCC = 0	DCA(n+1)
1	**X**	**X**	**X**	**0**
0	**0**	**X**	**X**	**DCA(n)**
0	**1**	**0**	**X**	**DCA(n) + 1**
0	**1**	**1**	**0**	**DCA(n) + 1**
0	**1**	**1**	**1**	**DIA**

Note **: See DAIC register for DAI and DLA definitions**

. Counting registers

Two 16-bit registers (unsigned) per channel are dedicated for transfer count.

For a transfer of a N data block, DIC and DCC registers have to be loaded with N-1.

When DCC content is 0 (valid transfer count), it is loaded with DIC content for the next transfer.

- DIC : initial count

This register contains the total number of transfers of the entire block.

- DCC : current count

This register contains the remaining number of transfers to be done to fill the entire block.

It is decremented after each transfer.

The DCC values are :

RESET	DCC = 0	DCA(n+1)
1	**X**	**0**
0	**0**	**DCA(n) - 1**
0	**1**	**DIC**

. Control registers

Three 16-bit control registers are dedicated to the DMA controller interface :

- DGC : General control register

Four bits are dedicated for each DMA channel (bits 0 to 2 to channel 0, bits 4 to 6 to channel 1,bits 8 to 10 to channel 2, bits 12 to 14 to channel 3).
After reset, DGC default value is 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	DRW3	DBC1	DBC0	-	DRW2	DBC1	DBC0	-	DRW1	DBC1	DBC0	-	DRW0	DBC1	DBC0

DBC1/DBC0 : Bus choice for data transfer
00 : X-bus (def.)
01 : Y-bus
10 : I-bus
11 : reserved

DRWi : Data transfer direction
0 : Write access (def.)
1 : Read access

- DAIC : Address/Interrupt control register

Four bits are dedicated for each DMA channel (bits 0 to 3 to channel 0, bits 4 to 7 to channel 1, bits 8 to 11 to channel 2, bits 12 to 15 to channel 3).
After reset, DAIC default value is 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAI3	DLA3	DIP3	DIE3	DAI2	DLA2	DIP2	DIE2	DAI1	DLA1	DIP1	DIE1	DAI0	DLA0	DIP0	DIE0

DAIi : Address increment
0 : DCAi content unchanged (def.)
1 : DCAi content modified according to DLAi state

DLAi : Load address
0 : DCAi content incremented after each data transfer (def.)
1 : DCAi content loaded with DIA content if DCCi value is 0 or DCAi content incremented if DCCi value not equal to 0

DIPi : Interrupt pending
0 : No pending interrupt on channel i (def.)
1 : Pending interrupt on channel i (enabled if DIP_ENA input is high)

DIEi : Enable Interrupt
0 : Interrupt request output associated to channel i is masked (def.)
1 : Interrupt request output associated to channel i is not masked

- DMS : Mask Sensitiviy control register
Two bits are dedicated for each DMA channel (bits 0 and 1 to channel 0, bits 4 and 5 to channel 1,bits 8 and 9 to channel 2, bits 12 and 13 to channel 3).
After reset, DAICdefault value is 0x3333.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	DSE3	DMK3	-	-	DSE2	DMK2	-	-	DSE1	DMK1	-	-	DSE0	DMK0

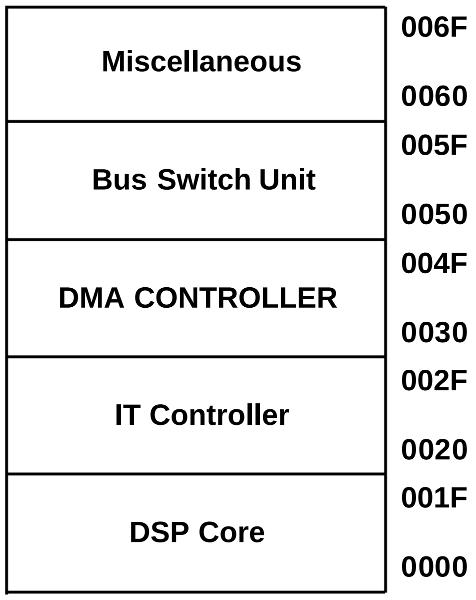
DSEi : DMA Sensitivity
0 : Low level
1 : Falling edge (def.)

DMKi : DMA Mask
0 : DMA channel not masked
1 : DMA channel masked (def.)

7 APPENDIX

7.1 MEMORY MAPPING (Y-memory space)

7.1.1 General mapping



7.1.2 Registers related to the D950-CORE

Register address	Register Name	Function	Location
0x0000	**BX**	**Modulo base address for X-memory space**	**ACU**
0x0001	**MX**	**Modulo maximum address for X-memory space**	**ACU**
0x0002	**BY**	**Modulo base address for Y-memory space**	**ACU**
0x0003	**MY**	**Modulo maximum address for Y-memory space**	**ACU**
0x0004	**POR**	**Port Output Register**	**PORT**
0x0005	**PIR**	**Port Input Register**	**PORT**
0x0006	**PCDR**	**Port Control Direction Register**	**PORT**
0x0007	**PCSR**	**Port Control Sensitivity Register**	**PORT**
0x0008 to 0x001F	**Reserved for test and emulation**		
0x0062	**IPR**	**Instruction Page Register**	**Peripherals**

7.1.3 Registers related to the interrupt controller

Register address	Register Name	Function	Location
0x0020	**IV0**	**Interrupt Vector 0 address**	**IT Controller**
0x0021	**IV1**	**Interrupt Vector 1 address**	**IT Controller**
0x0022	**IV2**	**Interrupt Vector 2 address**	**IT Controller**
0x0023	**IV3**	**Interrupt Vector 3 address**	**IT Controller**
0x0024	**IV4**	**Interrupt Vector 4 address**	**IT Controller**
0x0025	**IV5**	**Interrupt Vector 5 address**	**IT Controller**
0x0026	**IV6**	**Interrupt Vector 6 address**	**IT Controller**
0x0027	**IV7**	**Interrupt Vector 7 address**	**IT Controller**
0x0028	**ICR**	**Interrupt Control Register**	**IT Controller**
0x0029	**IMR**	**Interrupt Mask / Sensitivity Register**	**IT Controller**
0x002A	**IPR**	**Interrupt Priority Register**	**IT Controller**
0x002B	**ISPR**	**Interrupt Stack Pointer Register**	**IT Controller**
0x002C	**ISR**	**Interrupt Status Register**	**IT Controller**

7.1.4 Registers related to the DMA controller

Register address	Register Name	Function	Location
0x0030	**DIA0**	**DMA channel 0 initial address**	**DMA Controller**
0x0031	**DIA1**	**DMA channel 1 initial address**	**DMA Controller**
0x0032	**DIA2**	**DMA channel 2 initial address**	**DMA Controller**
0x0033	**DIA3**	**DMA channel 3 initial address**	**DMA Controller**
0x0034	**DCA0**	**DMA channel 0 current address**	**DMA Controller**
0x0035	**DCA1**	**DMA channel 1 current address**	**DMA Controller**
0x0036	**DCA2**	**DMA channel 2 current address**	**DMA Controller**
0x0037	**DCA3**	**DMA channel 3 current address**	**DMA Controller**
0x0038	**DIC0**	**DMA channel 0 initial count**	**DMA Controller**
0x0039	**DIC1**	**DMA channel 1 initial count**	**DMA Controller**
0x003A	**DIC2**	**DMA channel 2 initial count**	**DMA Controller**
0x003B	**DIC3**	**DMA channel 3 initial count**	**DMA Controller**
0x003C	**DCC0**	**DMA channel 0 current count**	**DMA Controller**
0x003D	**DCC1**	**DMA channel 1 current count**	**DMA Controller**
0x003E	**DCC2**	**DMA channel 2 current count**	**DMA Controller**
0x003F	**DCC3**	**DMA channel 3 current count**	**DMA Controller**
0x0040	**DGC**	**DMA General Control**	**DMA Controller**
0x0041	**DMS**	**DMA Mask Sensitivity**	**DMA Controller**
0x0042	**DAIC**	**DMA Address / Interrupt Control**	**DMA Controller**

7.1.5 Registers related to the Bus Switch Unit

Register address	Register Name	Function	Location
0x0050	**IER**	**External I-bus control register**	**BSU**
0x0051	**XER**	**External X-bus control register**	**BSU**
0x0052	**YER**	**External Y-bus control register**	**BSU**