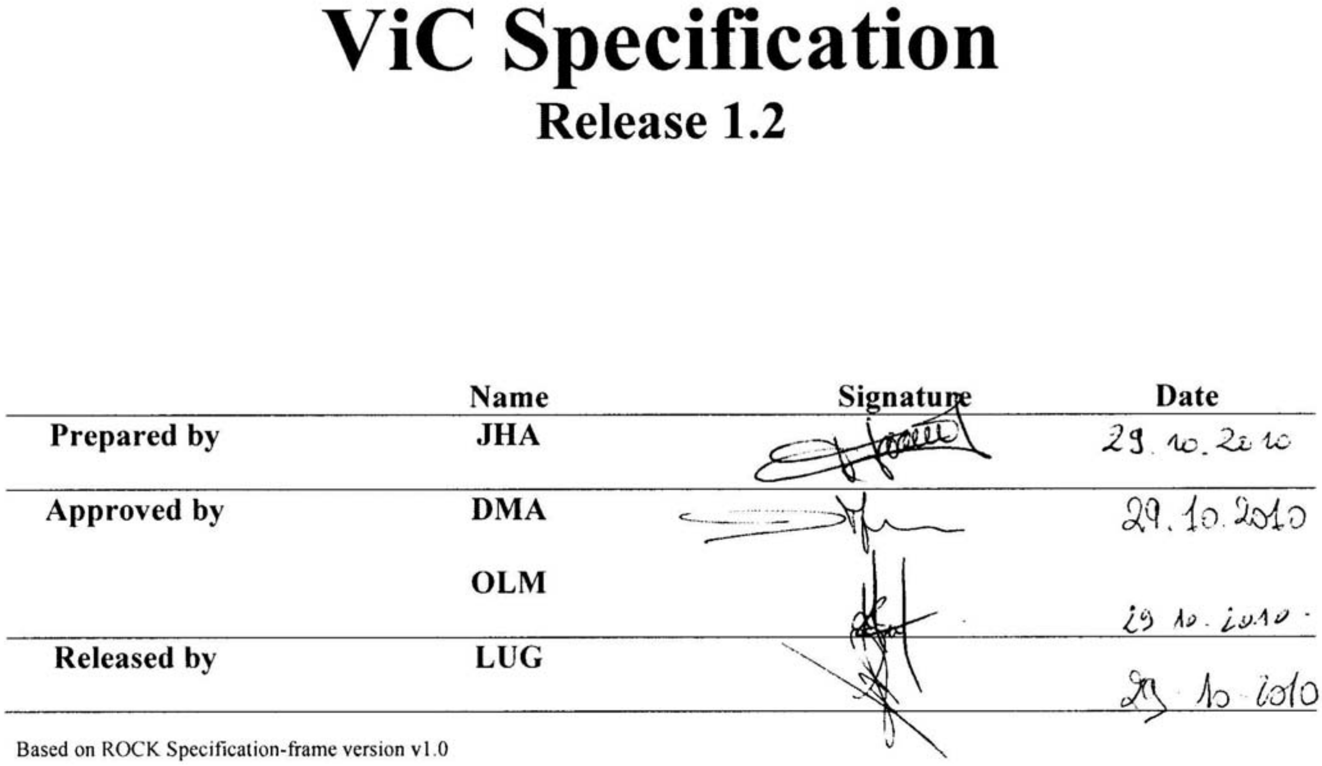
Flip80251-Hurricane revision 1.1



ViC Specification

Release 1.2

	Name	Signature	Date
Prepared by	JHA		29. 10. 2010
Approved by	DMA		29. 10. 2010
	OLM		29. 10. 2010.
Released by	LUG		29. 10. 2010

Based on ROCK Specification-frame version v1.0

Description of the modification: Update of timing diagrams of accesses with wait states insertion

Dolphin Integration
39 Avenue du Granier - BP 65 - Inovallée - 38242-MEYLAN - FRANCE
Tel : + 33 4 76 41 10 96 - Fax : + 33 4 76 90 29 65 - Web : www.dolphin-integration.com

USER WARNING

The Virtual Component (ViC) hereby specified must be used following the rules described in the User Manual and ViC specification documents.
Any rule violation or deviation to a rule may result in a failure or a performance decrease of the ViC, and will invalidate the Trust Commitment clause of the appropriate Purchase Agreement.

As there is no standard specification for Virtual Components, ours are enriched in terms of completeness of data for SoC integration, but optimized in terms of the features necessary for most users. Enhancements with additional features are prepared for those who strive for specific requirements.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Interpretation of any unspecified point is absolutely up to the designer of this circuit
Disclosed under NDA only.

1. GENERAL DESCRIPTION

1.1 Introduction

The virtual component Flip80251 "Hurricane" is an accelerated 80251 microcontroller, which is an enhanced version of the standard 80C51 with 16-bit and 32-bit capability. It is 100 % binary upward compatible with the standard 8xC251 but optimized for the highest Performance. Flip80251-Hurricane targets 8-16 microcontroller applications requiring an increase processing power, a reduced operating frequency or a larger memory space.
Its enhanced pipeline architecture provides a rise of processing speed an average of 3.0 X running at the same clock frequency as a standard 8xC251 real component and up to 40 times faster than a standard 80C51.
The extra processing power can be used, either to increase the performance of any 80C51/8xC251-based application, or to run the same application at a slower clock speed, so as to save power! The architecture can provide a significant code size reduction when compiling C program while fully preserving assembly code written for 80C51 microcontroller.

Flip80251-Hurricane, as any other member of the Flip family of microcontroller, is fully configurable, which means it is delivered in the precise configuration meeting user's requirements together with its Virtual Testbench for ensuring a SoC right-on-first pass per the VSIA standard.

1.2 Features

1.2.1 Key Features

- Full binary code upward compatibility with standard 80C51/8xC251 products.
- Available in two configurations:
  - *Binary mode*: providing full binary code compatibility with the legacy 80C51/52.
  - *Source mode*: providing an enriched instruction set with 16/32-bit capability.
- A pipelined architecture providing single cycle execution for most of the instructions when the pipeline is full.
- Speed is on average 3.0 X faster than the legacy 8xC251 based on the total number of clock cycles for all the instructions and speed is up to 40 times faster than a standard 80C51.
- Conversely, the clock frequency can be divided by 3.0 on average while preserving the same performance than with the standard 8xC251.
- Enriched instruction set
  - 16-bit arithmetic and logic instructions
  - Compare and conditional jump instructions
- 24-bit linear addressing that allows to access to up to 8 Mbytes of program memory and up to 8 Mbytes of data memory
- Hardware control Wait state solution for asynchronous peripherals and low cost memories.
- 64-Kbyte extended stack space.
- Extra dedicated output bus (Bus Monitor) for real time trace disassembly of the code execution, easing hardware/software co-verification.
- Power saving mode (idle and power-down).
- Improvements of C-Code efficiency, which give a code size reduction up to a factor 3 when compared with the C51 C-compiler.
- De-multiplexed Address/Data Bus to allow easy connection to memory.

1.2.2 Configurable Features

- A full set of peripherals: up to 5 PCA modules, up to 3 Timer/Counters, a Watchdog Timer, a serial port, a SPI interface …
- Support connection with Direct Memory Access (DMA) controller
- Real-time emulation solution thanks to the BIRD (Built-In Real-time debugger).
- Up to 7 additional interrupts available.

1.3 Applications

- **Central Processing Unit**
  - High-Speed Modems
  - Printers
  - DVD ROM and players
  - Scanner
  - High-End Joystick

- **Peripherals Controllers**
  - Smart card
  - Portable Telecom Devices

1.4 Block Diagram

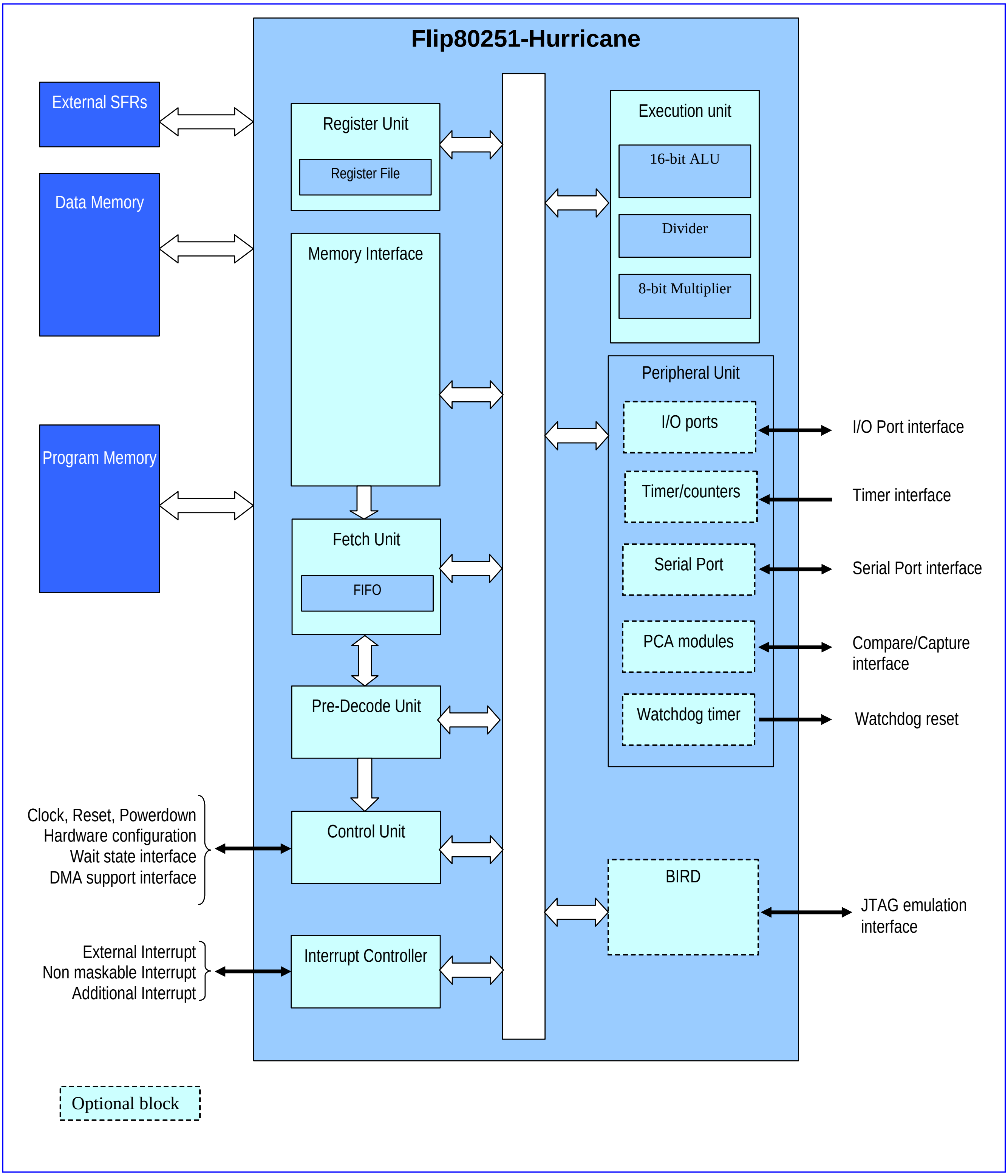


Figure 1: Flip80251-Hurricane Block Diagram

1.5 Difference between the Flip80251-Hurricane and the standard 8xC251

Although the Flip80251-Hurricane is 100 % code compatible with the standard 8xC251, there are few differences described below.

1.5.1 *Configuration bytes*

The Flip80251-Hurricane does not include the two configuration bytes UCONFIG0 and UCONFIG1 that are parts of the standard. However, the main configurable modes can be retrieved as explained below.

- **Binary or Source mode**
  The Flip80251-Hurricane is delivered either in binary or source mode. These two configurations correspond to the SRC configuration bit in UCONFIG0 register of the standard 8xC251
- **0 to 3 wait state generation**
  The Flip80251 provides a real-time wait state solution for dynamic bus control. The hardware controlled wait state feature is much more flexible and can advantageously replace the previous wait state configuration. Please refer to the chapter 9.3 for more details on wait state solution.
- **Extended ALE**
  The Flip80251-Hurricane provides de-multiplexed address and data bus. There is no need of ALE signal.
- **2 or 4 bytes interrupt frames**
  The *intrmode* input corresponds to the INTR configuration bit in UCONFIG1 register of the standard 8xC251. When set, the interrupt push 4 bytes onto the stack (the three bytes of PC register and PSW1). When clear, the interrupt push 2 bytes onto the stack (the two lower bytes of PC register).
- **Extended addressability of 256 Kbytes external memory**
  The Flip80251-Hurricane already provides support for 16 MBytes memory address space.
- **Page mode**
  The Flip80251-Hurricane does not provide support for page mode. An instruction fetch takes one clock cycle, assuming *prgbusy* (program wait state control) signal is not asserted. Both program and data memory interfaces were designed to target 16-bit synchronous single port memory.
- **Mapping the internal code memory for look up table**
  This configuration is not supported. It is not possible with the Flip80251-Hurricane to map the upper 8 K bytes of internal code memory to data memory addresses 00:E0000h-00:FFFFh.

1.5.2 *Clock and reset*

- **State time**
  The basic unit of time in the standard 8xC251 is the *state time* (or state), which is two oscillator periods. There is no definition of state time with the Flip80251. The basic unit of time is an oscillator period, also known as *clock cycle*.
- **Peripheral cycle**
  The 8xC251 peripherals operate on a peripheral cycle, which is six state times, or also 12 clock cycles. In order to keep the compatibility with their originals models, the Flip80251 peripherals

preserve the notion of *peripheral cycle* by using a signal (called *clkdiv12*) which is active once per 12 clock cycles.

- **Reset**
  The reset input of the Flip80251 (*corerst*) is an asynchronous active high reset, which is connected to the reset of internal flip-flops. This signal is synchronized internally with *coreclk* signal in order to provide a synchronous release of the internal reset signal
  A reset is accomplished by holding the *corerst* pin high for at least one clock (*coreclk*) cycle. The Flip80251 responds by generating an internal synchronous reset. When *corerst* is pulled low, it takes 1 or 2 clock cycles to release the internal reset signal.

Note that in case of configuration including the BIRD-JTAG emulation interface, the JTAG logic needs to be initialized even if it is not used.

1.5.3 Speed

Thanks to its enhanced pipelined architecture, the Flip80251 is on average three times faster as the 8xC251 standard. As a result, usage of programs developed for a standard 80C251 could lead to problems if the behavior of the tasks carried out in the program relies on program execution speed.

1.5.4 Connection

The Flip80251-Hurricane is delivered is a fully de-multiplexed address and data bus configuration. Additionally, the I/O port pin and the peripherals alternate function are separated. There are no internal tri-state busses, so the bi-directional functions are provided as input signal, output signal and direction control signal. Connection advices are provided within the user guide.

1.5.5 Memories

The Flip80251-Hurricane is provided with separated bus for accessing to the CODE memory and accessing to the DATA memory. The advantage of such solution (HARVARD architecture) is to speed up the program execution. Thanks to the dedicated bus for the CODE memory access, the CPU can fetch instruction at almost each clock cycle, which enables to keep the pipeline full. The 2 memory interfaces (CODE and DATA) could be multiplexed outside the core in order to access to a common memory (e.g. a Flash memory). In such a case, the arbitration between the two busses has to be performed by using respectively the program wait state interface or the data wait state interface. Note that the performance would decrease. Please refer to the chapter 3.1.1 for more details.

1.5.6 Peripherals and interrupt

The Flip80251- Hurricane has the same peripherals and interrupts system than the Intel 8xC251Sx. There is an exception for the Watchdog timer which includes an additional control register which is not part of the standard.
Regarding the interrupt system, in addition to the interrupt of the Intel 8xC251Sx, the Flip80251-Hurricane implements also a Non Maskable Interrupt input (*intnmi)* as described in the TSC8xC251GxD specification.

1.5.7 Advanced features in the Flip80251

The Flip80251-Hurricane can be delivered with a number of advanced features that are not available in standard parts, such as:

- Additional interrupts
- DMA controller support
- BIRD-JTAG emulation interface
- Bus monitor
- External SFR bus
- Port direction signal.

2. FLIP80251 CONNECTIONS

2.1 Logic symbol

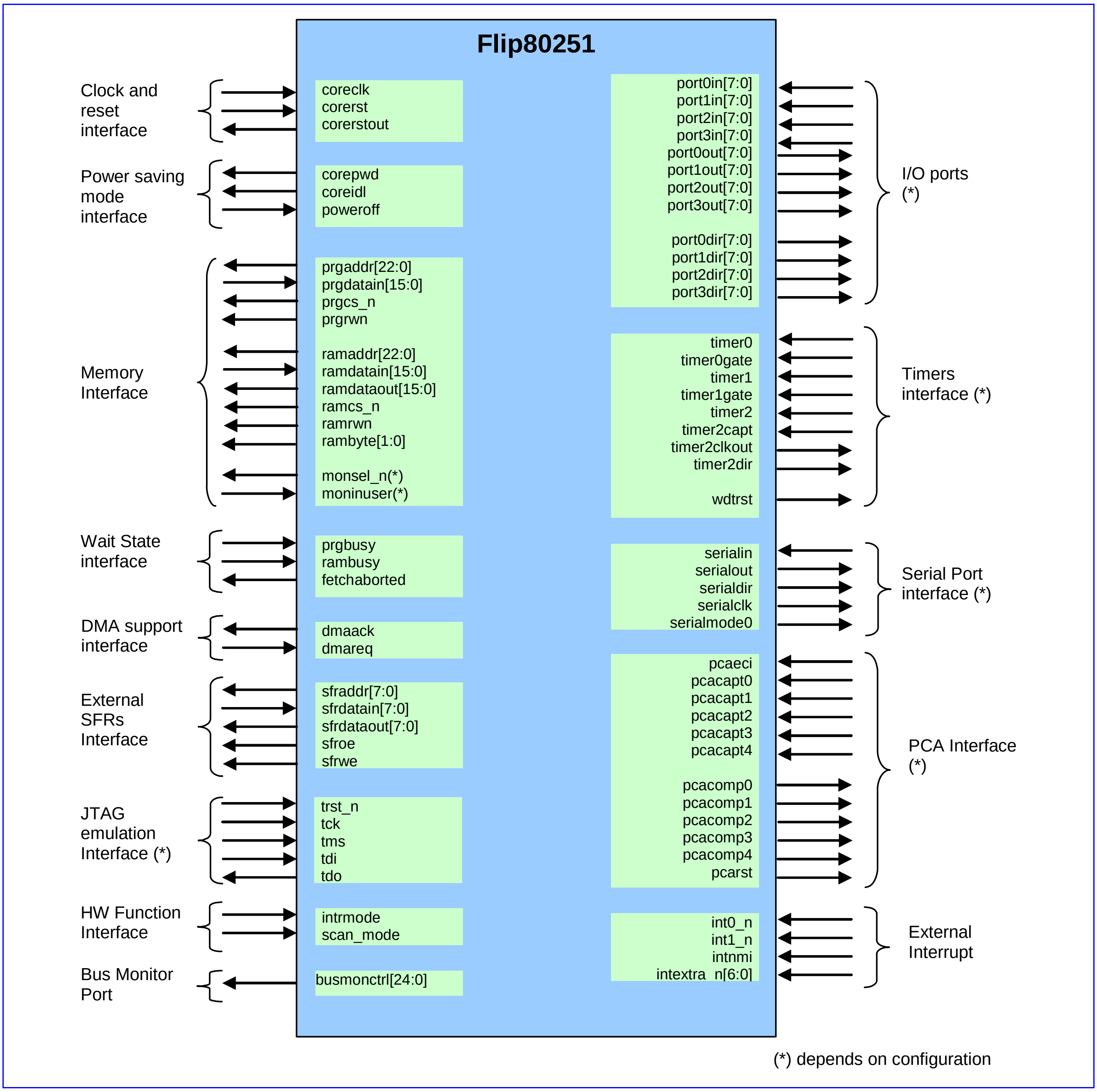


Figure 2: Flip80251-Hurricane logic symbol

2.2 List of I/O pins

I: input / O: output
Some I/O pins does not exist in certain configurations.

Name	I/O	Function
coreclk	I	Core Clock.
corerst	I	Asynchronous reset of the core. Active high.
corerstout	O	Combined reset signal applied to the core. Can be used to reset user's peripherals. It can be activated either by the external reset input (corerst), by the software debugger (BIRD) or by a peripheral (e.g. watchdog timer)

Table 1: Flip80251 general pin description

Name	I/O	Function
corepwd	O	Power-down mode indication. Active high. It can be used to turn off the clock during power-down mode.
coreidl	O	Idle mode indication. Active high. It can be used to stop the CPU clock during idle mode while the peripherals continue to be clocked.
poweroff	I	Power off flag (PCON.4) Use to set by hardware POF bit as VCC rises above a TBD voltage to indicate that power has been off or VCC had fallen below TBD voltage and that on-chip volatile memory is indeterminate.

Table 2: Power saving mode interface pin description

<table>
<tr><th>Name</th><th>I/O</th><th>Function</th></tr>
<tr><td>prgaddr[22:0]</td><td>O</td><td>Program memory address bus.
Allow to access the logical byte address range [80:0000h-FF:FFFFh]</td></tr>
<tr><td>prgdatain[15:0]</td><td>I</td><td>Data input from program memory.</td></tr>
<tr><td>prgcs_n</td><td>O</td><td>Program memory chip select enable. Active low.</td></tr>
<tr><td>prgrwn</td><td>O</td><td>Program memory read/not write.</td></tr>
<tr><td>ramaddr[22:0]</td><td>O</td><td>Data memory address bus.
Allow to access logical byte address range [00:0000h-7F:FFFFh]</td></tr>
<tr><td>ramdatain[15:0]</td><td>I</td><td>Data input from data memory.</td></tr>
<tr><td>ramdataout[15:0]</td><td>O</td><td>Data output for data memory.</td></tr>
<tr><td>ramcs_n</td><td>O</td><td>Data memory chip select. Active low.</td></tr>
<tr><td>ramrwn</td><td>O</td><td>Data memory read/not write.</td></tr>
<tr><td>rambyte[1:0]</td><td>O</td><td>Data memory byte control (for write operation):
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>'00'</td><td>Reserved</td></tr>
<tr><td>'01'</td><td>Write only the least significant byte (ramdataout[7:0])</td></tr>
<tr><td>'10'</td><td>Write only the most significant byte (ramdataout[15:8])</td></tr>
<tr><td>'11'</td><td>Write the complete 16-bit data (ramdataout[15:0])</td></tr>
</table></td></tr>
</table>

monsel_n[1]	O	Selection of monitor memory. Active low.
moninuser[1]	I	Indicates that monitor program is mapped into user application program. Otherwise, the monitor program is located into a separated memory. Active high

Table 3: Memory Interface description

Name	I/O	Function
prgbusy	I	Program memory busy signal. Active high. When high, indicates that the program memory need a longer access for the fetch and that wait states must be inserted.
rambusy	I	Data memory busy signal. Active high. When high, indicates that the data memory need a longer access for the read/write operation and that wait states must be inserted.
fetchaborted	O	Fetch abortion signal. When high, indicates that the current data fetch is aborted and then prgbusy must be de-asserted.

Table 4: Memory wait state interface description

Name	I/O	Function
dmareq	I	Request from a peripheral (e.g. DMA controller) to take the control of memory bus. Active high
dmaack	O	Indication that bus control is granted to the requester. Active high.

Table 5: DMA support interface description

Name	I/O	Function
sfraddr[7:0]	O	Address bus for external User-defined Special Function Registers
sfrdatain[7:0]	I	Data read from external SFRs.
sfrdataout[7:0]	O	Data output for external SFRs write.
sfrwe	O	SFR write enable control. Active high.
sfroe	O	SFR read control. Active high.

Table 6: External SFR interface description

Name	I/O	Function
port0in[7:0]	I	Port 0 Input.
port1in[7:0]	I	Port 1 Input.
port2in[7:0]	I	Port 2 Input.
port3in[7:0]	I	Port 3 Input.
port0out[7:0]	O	Port 0 Output.
port1out[7:0]	O	Port 1 Output.
port2out[7:0]	O	Port 2 Output.
port3out[7:0]	O	Port 3 Output.

[1] Only in case of configuration including the Built-In Real-time debugger (BIRD)

port0dir[7:0]	O	Port direction indication/control signal for port 0.
port1dir[7:0]	O	Port direction indication/control signal for port 1.
port2dir[7:0]	O	Port direction indication/control signal for port 2.
port3dir[7:0]	O	Port direction indication/control signal for port 3.

Table 7: I/O Ports pin description

Name	**I/O**	**Function**
timer0	I	Timer 0 input.
timer0gate	I	Timer 0 Gate control input. Active high
timer1	I	Timer 1 input.
timer1gate	I	Timer 1 Gate control input. Active high
timer2	I	Timer 2 input.
timer2capt	I	Timer 2 Capture control input
timer2clkout	O	Timer 2 clock output in mode3.
timer2dir	O	Timer 2 direction control signal for multiplexing timer2clkout and timer2capt.
wdtrst	O	Watchdog timer overflow. Active high

Table 8: Timers interface description

Name	**I/O**	**Function**
serialin	I	Serial port data input.
serialout	O	Serial port data output.
serialclk	O	Serial Clock output for mode 0.
serialdir	O	Serial Data direction for mode 0.
serialmode0	O	Serial mode 0 indicator. Useful for pin compatible version

Table 9: Serial Port interface description

Name	I/O	Function
pcaeci	I	PCA external input clock
pcacapt0	I	Capture input 0
pcacapt1	I	Capture input 1
pcacapt2	I	Capture input 2
pcacapt3	I	Capture input 3
pcacapt4	I	Capture input 4
pcacomp0	O	Compare output 0
pcacomp1	O	Compare output 1
pcacomp2	O	Compare output 2
pcacomp3	O	Compare output 3
pcacomp4	O	Compare output 4
pcarst	O	PCA software reset signal. Active high.

Table 10: PCA interface description

Name	I/O	Function
int0_n	I	External Interrupt 0. When level-activated, active low. When transition-activated, active when falling edge.
int1_n	I	External Interrupt 1. When level-activated, active low. When transition-activated, active when falling edge.
intnmi	I	Non-maskable interrupt input. Active high.
intextra_n[6-0]	I	Additional external interrupts. Active Low.

Table 11: External interrupts description

Name	I/O	Function
intrmode[1]	I	Interrupt mode. When low, the interrupt push two bytes (PC[15:8] & PC[7:0]) onto the stack. Otherwise, the interrupts push four bytes (PSW1 + PC[23:16] & PC[15:8] & PC[7:0]) onto the stack.
scan_mode	I	Scan mode selection. Must be set to 1 during scan test

Table 12: Special Hardware function pin description

[1] The value of this pin is evaluated only the first clock cycle following the exit of reset state

Name	I/O	Function
busmonctrl[24]	O	When high, indicates that the sequencer is in the first cycle of the execution stage of the current instruction
busmonctrl[23:0]	O	Current value of the program counter (PC)

Table 13: Bus monitor interface description

✍: This bus is provided for debug purpose. It adds no gate to the design.

Name	I/O	Function
trst_n (1)	I	JTAG Reset, active low.
tck	I	JTAG clock.
tms	I	JTAG mode select (sampled on tck' rising edge)
tdi	I	JTAG Data Input (sampled on tck' rising edge). Serial test instruction and data are received by the test logic at tdi.
tdo	O	JTAG Data Output (synchronous with tck' falling edge). It is the serial output for the data registers from the test logic.

Table 14: JTAG emulation interface description

(1) In case of configuration with BIRD, *trst_n* input is optional. In such a case, JTAG logic is initialized thanks to the JTAG protocol: if *tms* is set to 1 during at least 5 *tck* clock cycles, the TAP controller enters in **TEST-LOGIC-RESET** state and JTAG logic is properly initialized.

3. FLIP80251 ADDRESS SPACES

The Flip80251 have three address spaces: a Memory Space, a Special Function Register (SFR) Space and a Register File.

3.1 Flip80251 Architecture

It is convenient to view the un-segmented, 16 Mbytes memory space as consisting of 256 regions of 64 Kbytes, numbered 00: to FF:

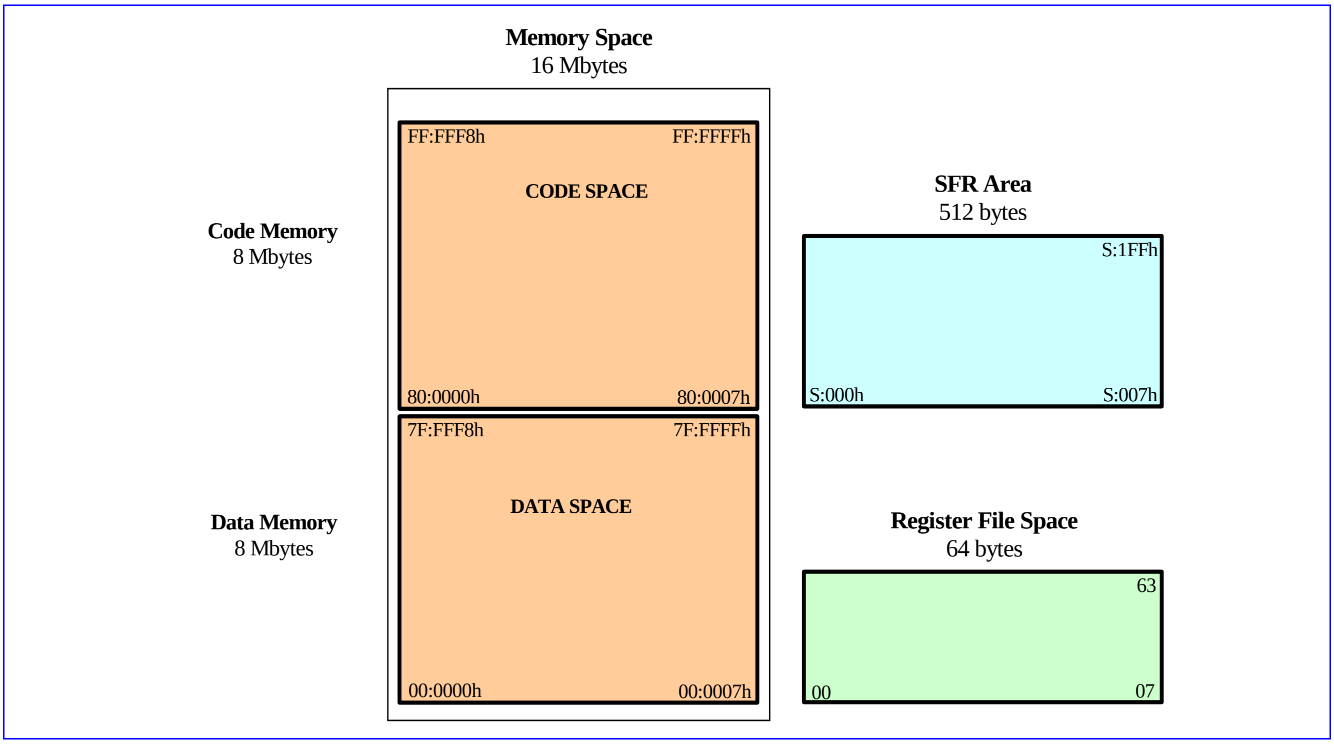


Figure 3: Flip80251 Address space

3.1.1 Memory Space

In the Flip80251, the full 16 Mbytes address range is supported. The Flip80251 can address up to 8 Mbytes of CODE memory and up 8 Mbytes of DATA memory while a standard C251 provides a maximum of 256 Kbytes of memory.

In order to speed up the program execution, the Flip80251 is available in a fully de-multiplexed address/data bus version. Additionally, the program memory and the data memory have two separated interfaces.

It is up to the designer using the Flip80251 to decide how to partition the memory space and the required size of program and data memory. However, there are special cautions to take when mapping memories:

The region [00:0000h-7F:FFFFh] are part of the DATA memory space (lower 8 Mbytes) and any access to this region activates the data memory interface.

The region [80:0000h-FF:FFFFh] are part of the CODE memory space (upper 8 Mbytes) and any access to this region activates the program memory interface.

3.1.1.1 Program Memory (CODE space)

The reset address of the instruction fetch is FF:0000h. The instruction fetch takes only one clock cycle. The Flip80251 only does extra program memory fetches during a jump/branch.
There is no internal (on-chip) program memory definition. So, as opposed to the standard 8xC251, the Flip80251 fetches the instruction always two bytes at a time (16-bit fetch), in one clock cycle.
The program memory is dedicated to the instruction fetch. However, there are a restricted number of instructions that enable to read data into program memory.
Additionally, the program memory interface of the Flip80251-Hurricane provides write capability that can be used, for instance, to download code into code memory (in case of writable program memory). There are also a restricted number of instructions that enable to write into program memory, that are listed below:

Program memory read	Program memory write
Data Transfer instructions	
MOV Rm, @DRk	MOV @DRk, Rm
MOV WRj, @DRk	MOV @DRk, WRj
MOV Rm, @DRk+dis24	MOV @DRk+dis24, Rm
MOV WRj, @DRk+dis24	MOV @DRk+dis24, WRj
MOVC A, @A+DPTR	
MOVC A, @A+PC	
Arithmetic instructions	
ADD Rm, @DRk	
SUB Rm, @DRk	
CMP Rm, @DRk	
Logical instructions	
ANL Rm, @DRk	
ORL Rm, @DRk	
XRL Rm, @DRk	

Table 15: Program memory read/write instructions

3.1.1.2 Data Memory (DATA space)

Except the region 00h and 01h that have a special behavior and can be accessed with several addressing modes, all the others regions (02: to 7F:) are only accessible with 24-bit indirect and 24-bit displacement instructions (or with a MOVX instruction by changing the value of DXPL).

✋ *General-purpose data memory begins at 00:0020h (see Figure 7: Register File Location 0-7).*

3.1.2 SFR Space

The SFR space can accommodate up to 512 8-bit special function registers with addresses S:000h-S:1FFh. Some of these locations may be unimplemented in a particular configuration. In the 8xC251 architecture, the prefix “S:” is used with SFR addresses to distinguish them from the memory space addresses 00:0000h-00:01FFh. [1]

[1] *Locations S:000h-S:07Fh and S:100h-S:1FFh are unimplemented.*

3.1.3 Register File

The register file has its own address space. The 64 locations in the register file are numbered decimally from 0 to 63. Locations 0-7 represent one of four register banks, each having 8 registers. The 32 bytes required for these banks occupy locations 00:0000h - 00:001Fh in the memory space. Register file locations 8-63 do not appear in the memory space.

3.2 Compatibility with 80C51 architecture

The address spaces in the 80C51 architecture are mapped into the address spaces in the 8xC251 architecture. This mapping allows code written for 80C51 micro-controllers to run on 8xC251 micro-controllers.

Internal data memory locations 00h-7Fh can be addressed directly and indirectly. Internal data locations 80h-0FFh can only be addressed indirectly. Direct addressing to these locations accesses the SFRs. The Code memory (64 Kbytes) has a separate memory space. Data in the Code memory can be accessed only with the MOVC instruction. Similarly, the 64 Kbytes external data memory can be accessed only with the MOVX instruction.

The register file (registers R0-R7) contains four switch-able register banks, each having eight registers. The 32 bytes required for the four banks occupy locations 00h-1Fh in the on-chip data memory.

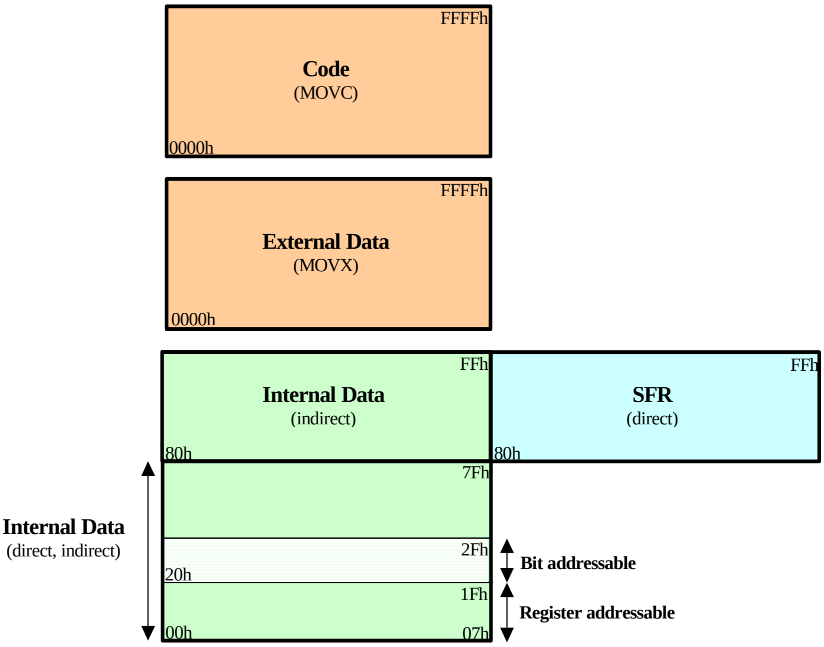


Figure 4: 80C51 Address spaces

✋ *: In binary mode or in source mode, the reset address of the Flip80251-Hurricane is FF:0000h*

The Figure 5 shows how the address spaces in the 80C51 architecture map into the address spaces in the 8xC251 architecture.

The 64 Kbytes code memory for 80C51 micro-controllers maps into region FF: of the memory space for 8xC251 micro-controllers. Assemblers for 8xC251 micro-controllers assemble code for 80C51 micro-controllers into region FF:, and data accesses to code memory (MOVC) are redirected to this region. The assembler also maps the interrupt vectors to region FF:. This mapping is transparent to the user; code executes just as with an 80C51 micro-controller, without modification.

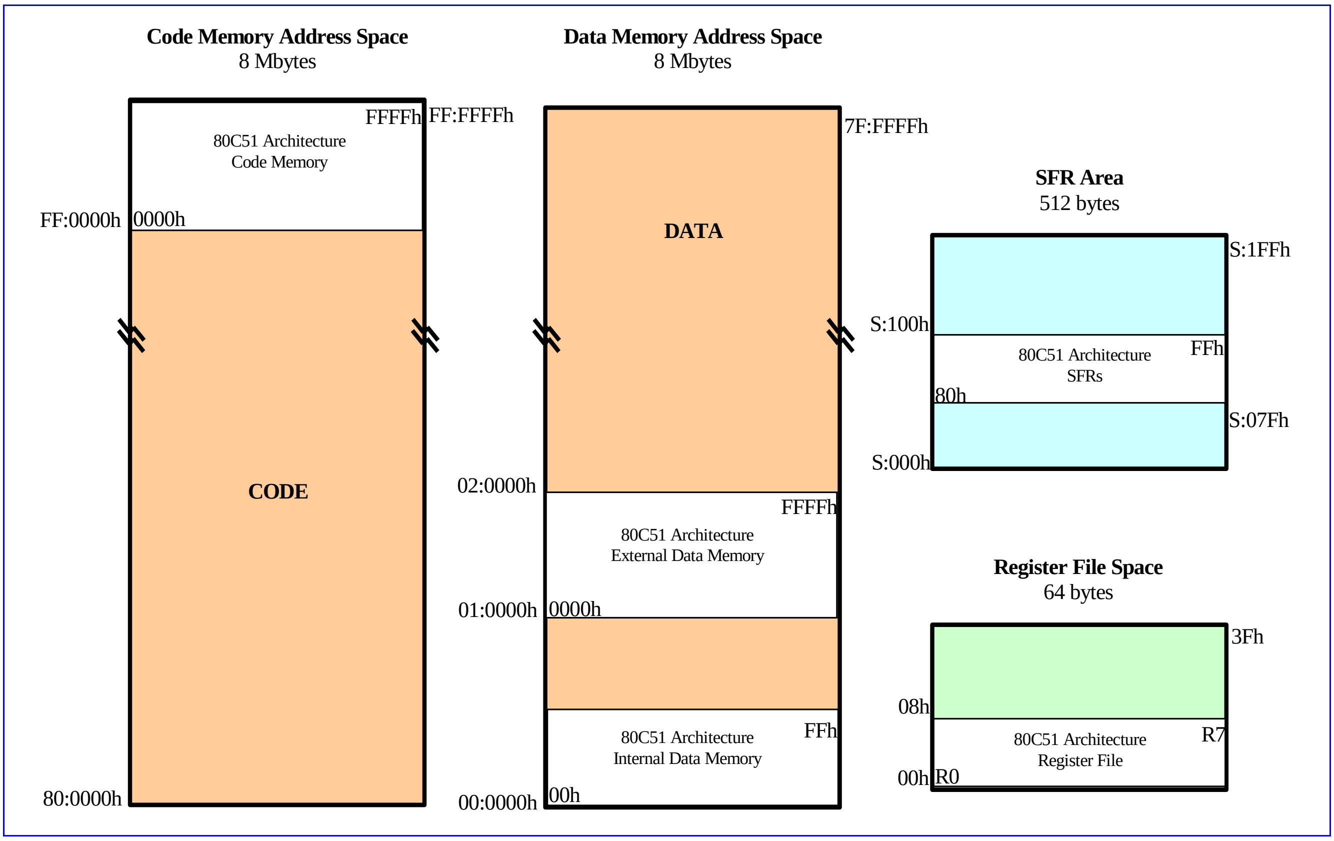


Figure 5: Mapping 80C51 address spaces to Flip80251 architecture

Memory Type	80C51 Architecture			8xC251 Architecture
	Size	Location	Data addressing	Location
Code	64 Kbytes	[0000h-FFFFh]	Indirect using MOVC	[FF:0000h-FFFFFh]
External Data	64 Kbytes	[0000h-FFFFh]	Indirect using MOVX	[01:0000h-01:FFFFh]
Internal Data	128 bytes	[00h-7Fh]	Direct, Indirect	[00:0000h-00:007Fh]
	128 bytes	[80h-FFh]	Indirect	[00:0080h-00:00FFh]
SFRs	128 bytes	[S:80h-S:FFh]	Direct	[S:080h-S:0FFh]
Register	8 bytes	[R0-R7]	Register	[00:0000h-00:001Fh]

Table 16: Address Mappings

The 64-Kbyte external data memory for 80C51 micro-controllers is mapped into the memory region specified by bits 16–23 of the data pointer DPX, i.e., DPXL. DPXL is accessible as register file location 57 and also as the SFR at S:084h. The reset value of DPXL is 01h, which maps the external memory to region 01. You can change this mapping by writing a different value to DPXL. A mapping of the 80C51 micro-controller external data memory into any 64-Kbyte memory region in the 8xC251 architecture provides complete run-time compatibility because the lower 16 address bits are identical in the two address spaces.

The 256 bytes of on-chip data memory for 80C51 micro-controllers (00h-FFh) are mapped to addresses [00:0000h-00:00FFh] to ensure complete run-time compatibility. In the 80C51 architecture, the lower 128 bytes (00h-7Fh) are directly and indirectly addressable; however the upper 128 bytes are accessible by indirect addressing only. In the 8xC251 architecture, all locations in region 00: are accessible by direct, indirect, and displacement addressing.

The 128-byte SFR space for 80C51 micro-controllers is mapped into the 512-byte SFR space of the 8xC251 architecture starting at address S:080h. This provides complete compatibility with direct addressing of 80C51 micro-controller SFRs (including bit addressing). The SFR addresses are unchanged in the new architecture. In the 8xC251 architecture, SFRs A, B, DPL, DPH, and SP (as well as the new SFRs DPXL and SPH) reside in the register file for high performance. However, to maintain compatibility, they are also mapped into the SFR space.

3.3 Register File

The Flip80251-Hurricane register file consists of 40 byte locations: 0-31 and 56-63. These locations are accessible as bit, bytes, words, and dwords. Several locations are dedicated to special registers; the others are general-purpose registers.

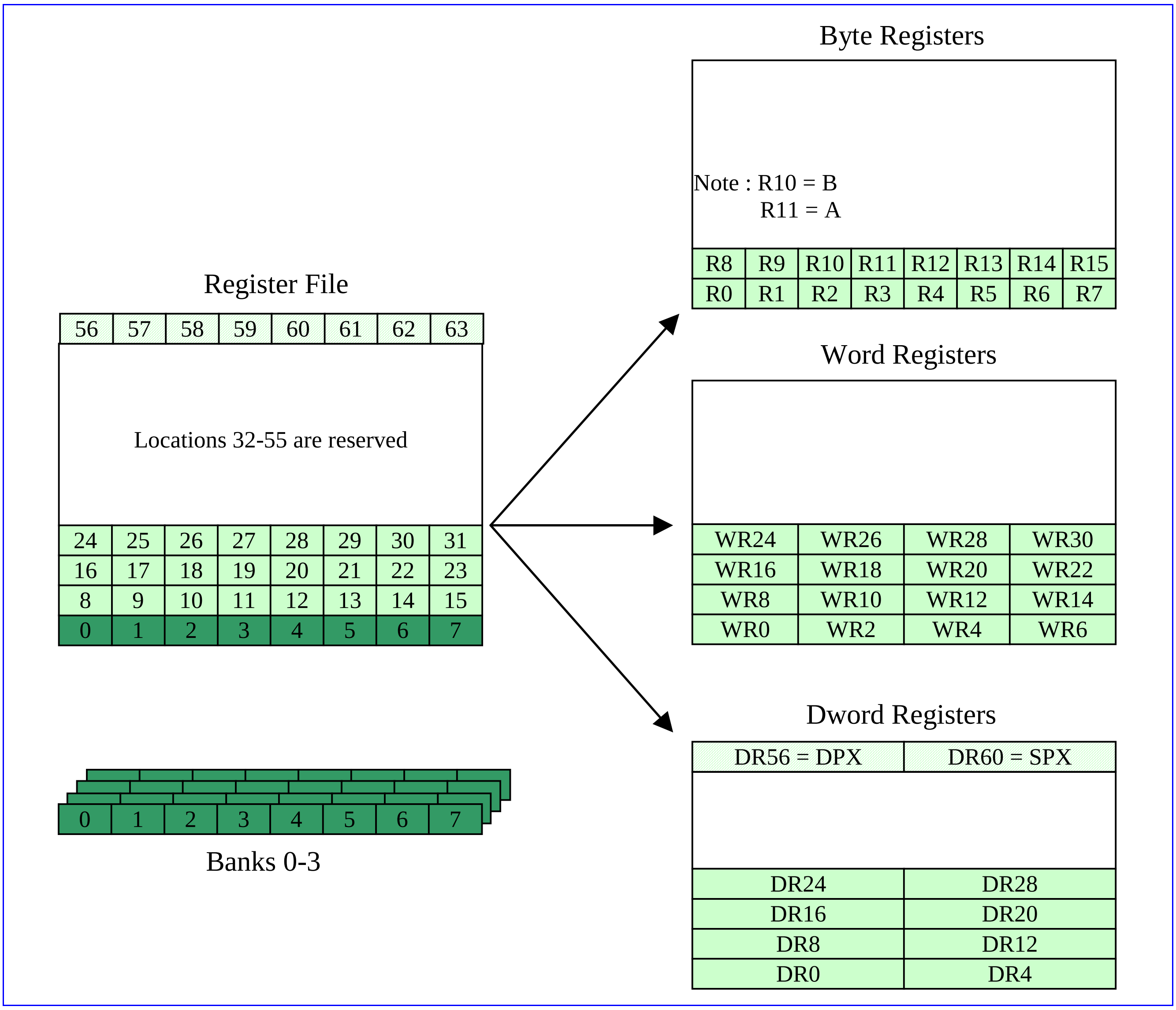


Figure 6: The Register File

- Register file locations 0-7 actually consist of four switchable banks of eight registers each. The four banks are always accessible as locations [00:0000h-00:001Fh] in the memory address space (see Table 17: Register Bank Selection). Only one Register Bank is used at a time when an instruction uses R0 to R7. 2 bits in Processor Status Word (PSW), called RS1 and RS0, control the selection of the Register Bank. This bank selection can be used for fast context switches. Bank 0 is selected upon reset. Indirect addressing mode used R0 and R1 as index registers
- Register file locations 8-31 and 56-63 are always accessible by their register file address. These locations are implemented as registers in the CPU. They are not accessible in the memory address space.
- Register file locations 32-55 are reserved and cannot be accessed.

☞: *In a standard 8xC251, the register banks are implemented as the first 32 bytes of on-chip RAM. In the Flip80251, the register banks are implemented as Flip-Flops within the core in order to speed up execution time of instructions using these registers. Then, memory access in the address range 00:0000h-00:001Fh are redirected to the Flip-Flops and the external data memory is never accessed for the address below 00:001Fh.*

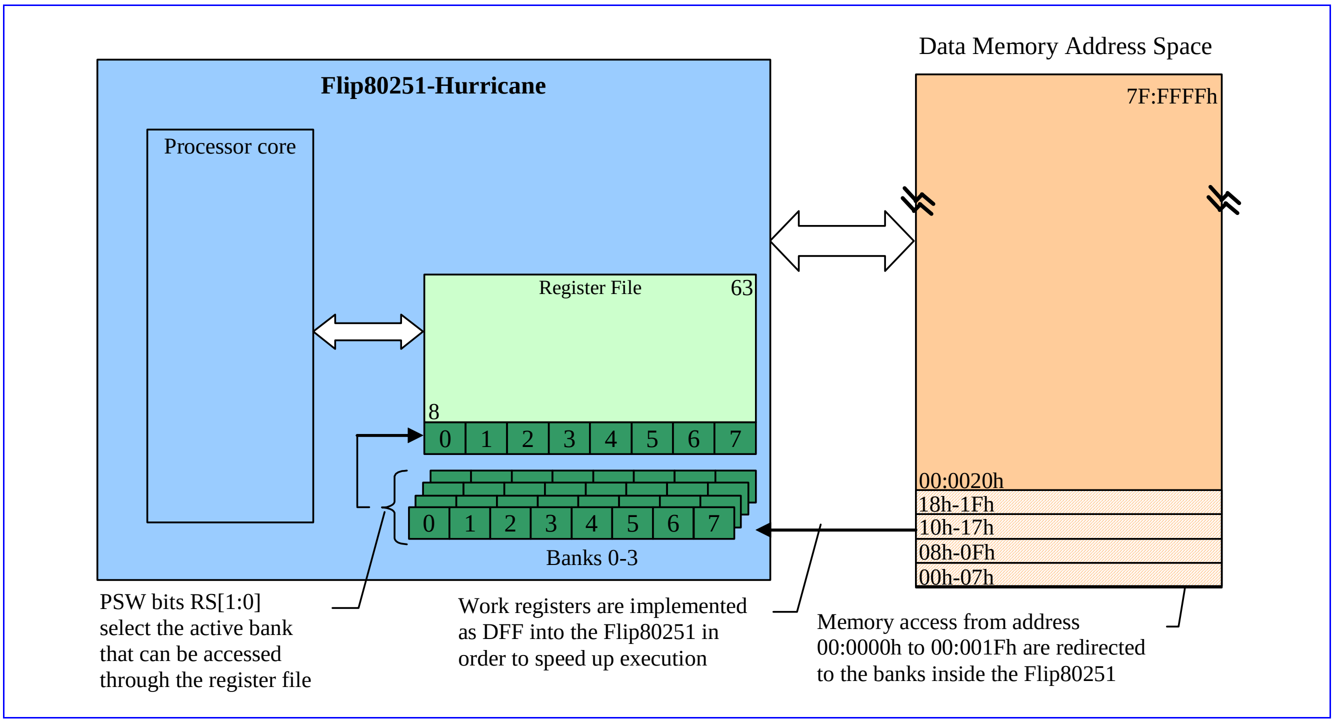

Data Memory Address Space
Flip80251-Hurricane
7F:FFFFh
Processor core
Register File
63
8
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
Banks 0-3
00:0020h
18h-1Fh
10h-17h
08h-0Fh
00h-07h
PSW bits RS[1:0] select the active bank that can be accessed through the register file
Work registers are implemented as DFF into the Flip80251 in order to speed up execution
Memory access from address 00:0000h to 00:001Fh are redirected to the banks inside the Flip80251

Figure 7: Register File Location 0-7

Register file address	Register bank	RS1	RS0	Data memory address range
R0-R7	Bank 0	0	0	[00:0000h - 00:0007h]
R0-R7	Bank 1	0	1	[00:0008h - 00:000Fh]
R0-R7	Bank 2	1	0	[00:0010h - 00:0017h]
R0-R7	Bank 3	1	1	[00:0018h - 00:001Fh]

Table 17: Register Bank Selection

3.3.1 *Byte, Word and Dword Registers*

Depending on its location in the register file, a register is addressable as a byte, a word, and/or a dword, as shown on the right side of Figure 6. A register is named for its lowest numbered byte location. For example:

- R4 is the byte register consisting of location 4.
- WR4 is the word register consisting of registers 4 and 5.
- DR4 is the dword register consisting of registers 4 to 7.

Locations R0-R15 are addressable as bytes, words, or dwords. Locations 16–31 are addressable only as words or dwords. Locations 56-63 are addressable only as dwords. Registers are addressed only by the names shown in Figure 6 - except for the 32 registers that comprise the four banks of registers R0–R7, which can also be accessed as locations [00:0000h–00:001Fh] in the memory space.

3.3.2 *Dedicated Registers*

The register file has four dedicated registers:

- R10 is the B-register
- R11 is the Accumulator
- DR56 is the extended data pointer, DPX
- DR60 is the extended stack pointer SPX

These registers are located in the register file; however, R10, R11, and some bytes of DR56 and DR60 are also accessible as SFRs. The bytes of DPX and SPX can be accessed in the register file only by addressing the dword registers. The dedicated registers in the register file and their corresponding SFRs are illustrated in Figure 8 and listed in Table 18.

Register File						**SFRs**	
Name			Mnemonic	Reg.	Location	Mnemonic	Address
Stack Pointer (SPX)	-		-	DR60	60	-	-
	-		-		61	-	-
	Stack Pointer, High		SPH		62	SPH	S:0BEh
	Stack Pointer, Low		SP		63	SP	S:081h
Data Pointer (DPX)	-		-	DR56	56	-	-
	Data Pointer, Extended Low		DPXL		57	DPXL	S:084h
	DPTR	Data Pointer, High	DPH		58	DPH	S:083h
		Data Pointer, Low	DPL		59	DPL	S:082h
Accumulator (A register)			A	R11	11	ACC	S:0E0h
B Register			B	R10	10	B	S:0F0h

Table 18: Dedicated Registers in the Register file and their corresponding SFRs

3.3.2.1 Accumulator and B register

The 8-bit *accumulator* (ACC) is byte register R11, which is also accessible in the SFR space as ACC at S:0E0h (Figure 8). The *B register*, used in multiplies and divides, is register R10, which is also accessible in the SFR space as B at S:0F0h. Accessing ACC or B as a register is faster than accessing them as SFRs. Instructions in the 80C51 architecture use the accumulator as the primary register for data moves and calculations. However, in the 8xC251 architecture, any of registers R0–R15 can serve for these tasks[1]. As a result, the accumulator does not play the central role that it has in 80C51 micro-controllers.

3.3.2.2 Extended Data Pointer, DPX

Dword register DR56 is the *extended data pointer*, DPX (Figure 8). The lower three bytes of DPX (DPL, DPH, and DPXL) are accessible as SFRs. DPL and DPH comprise the 16-bit *data pointer* DPTR. While instructions in the 80C51 architecture always use DPTR as the data pointer, instructions in the 8xC251 architecture can use any word or dword register as a data pointer. DPXL, the byte in location 57, specifies the region of memory (00:-FF:) that maps into the 64-Kbyte external data memory space in the 80C51 architecture. In other words, the MOVX instruction addresses the region specified by DPXL when it moves data to and from external memory. The reset value of DPXL is 01h.

3.3.2.3 Extended Stack Pointer, SPX

Dword register DR60 is the *stack pointer*, SPX (Figure 8). The byte at location 63 is the 8-bit stack pointer, SP, in the 80C51 architecture. The byte at location 62 is the *stack pointer high*, SPH. The two bytes allow the stack to extend to the top of memory region 00: . SP and SPH can be accessed as SFRs.

Two instructions, PUSH and POP directly address the stack pointer. Subroutine calls (ACALL, ECALL, LCALL) and returns (ERET, RET, RETI) also use the stack pointer. To preserve the stack, do not use DR60 as a general-purpose register.

[1] Bits in the PSW and PSW1 registers reflect the status of the Accumulator. There are no equivalent status indicators for the other registers.

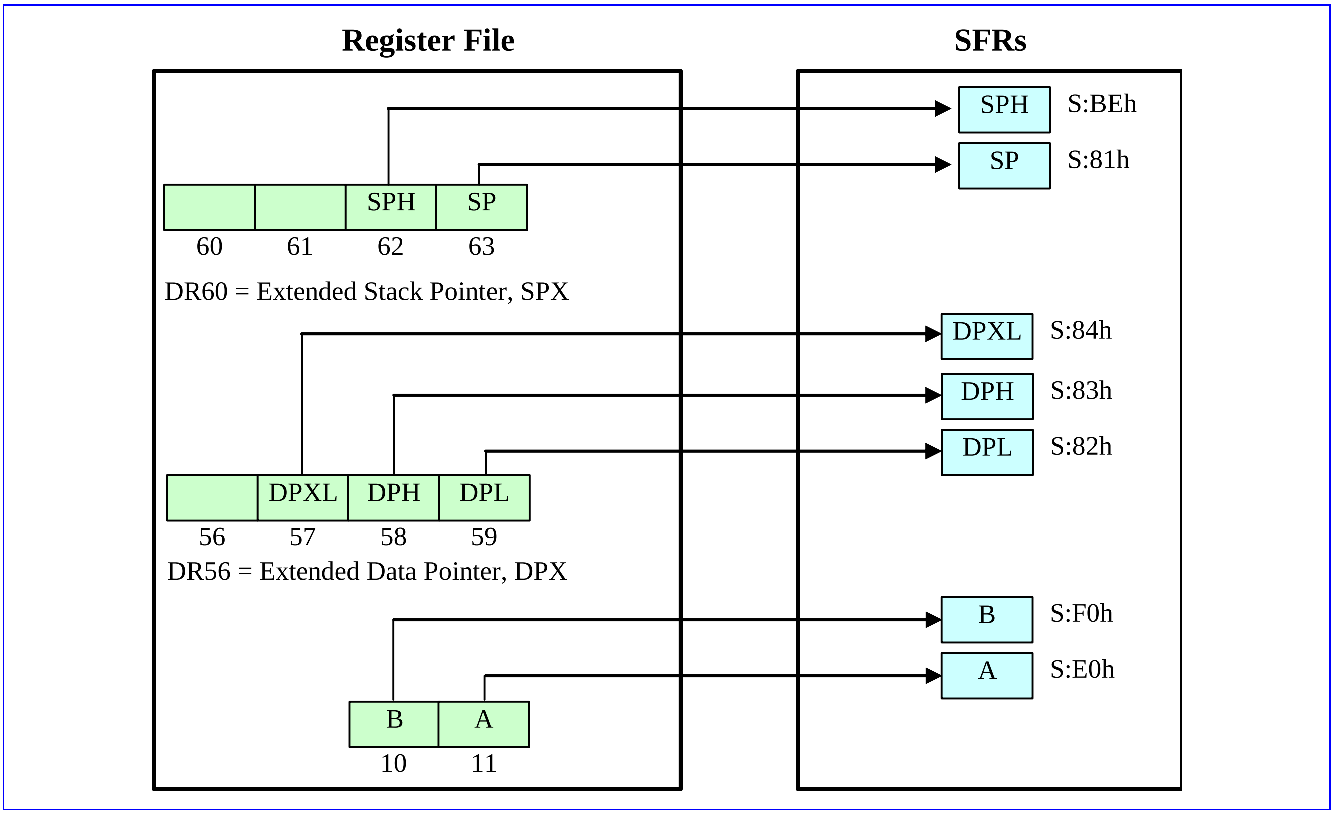


Figure 8: Dedicated Registers in the Register File and their corresponding SFRs

3.4 Special Functions Registers

The special function registers (SFRs) reside in their associated peripherals or in the core. The following tables shows the SFR address space with the SFR mnemonics and reset values. SFR addresses are preceded by “S:” to differentiate them from addresses in the memory space. Unoccupied locations in the SFR space (the blank locations in Table 19) are unimplemented, i.e., no register exists. If an instruction attempts to write to an unimplemented SFR location, the instruction executes, but nothing is actually written. If an unimplemented SFR location is read, it returns an unspecified value.

Despite the fact that 8xC251 architecture defines up 512 SFR locations (S:000h-S:1FFh), the C251 instruction set, as defined by Intel, allows to access only to SFR locations from S:080h to S:0FFh. In others words, there is no instruction that enables to access SFR locations S:000h-S:07Fh and S:100h-S:1FFh.

⬜	Core SFRs	■	I/O ports SFRs	■	Port direction SFRs
■	Timers 0/1 SFRs	■	Timer2 SFRs	■	WDT SFRs
■	UART SFRs	■	I2CM SFRs	■	SPI SFRs
■	PCA SFRs	■	PWM SFRs	■	CPMU SFRs
■	Additional IRQ SFRs	■	BIRD/POC SFRs	■	I2CS SFRs

F8	AIPL	CH	CCAP0H	CCAP1H	CCAP2H	CCAP3H	CCAP4H		FF
F0	B	STCON	SRXBUF	STXBUF		SSTAT0	SSTAT1	AIPH	F7
E8	AIE	CL	CCAP0L	CCAP1L	CCAP2L	CCAP3L	CCAP4L		EF
E0	ACC	MCON	MRXBUF	MTXBUF	MPRESC	MSTAT0	MSTAT1	MIEN0	E7
D8	CCON	CMOD	CCAPM0	CCAPM1	CCAPM2	CCAPM3	CCAPM4	CCAPO	DF
D0	PSW	PSW1	MIEN1		MCADDR	SIEN0	SIEN1	SSADDR	D7
C8	T2CON	T2MOD	RCAP2L	RCAP2H	TL2	TH2			CF
C0	AIF								C7
B8	IPL0	SADEN					SPH		BF
B0	P3	SPCR	SPDR	SPSR				IPH0	B7
A8	IE0	SADDR			P0_DIR	P1_DIR	P2_DIR	P3_DIR	AF
A0	P2	MPAGE	PWMC	PWMDC LSB	PWMDC MSB	WDTCON	WDTRST		A7
98	SCON	SBUF							9F
90	P1						CPUINFO	MMCON	97
88	TCON	TMOD	TL0	TL1	TH0	TH1	CCMCON	CCMVAL	8F
80	P0	SP	DPL	DPH	DPXL	XTALCON	CLKCON	PCON	87

Table 19: Flip80251 SFRs Memory Map

✋ *: Blank areas represent unimplemented SFR locations.*

SFR are listed in alphabetical order

Mnemonic	Address	Description	Reset value
ACC	S:0E0h	Accumulator	00h
B	S:0F0h	B Register	00h
DPH	S:083h	Data Pointer high byte	00h
DPL	S:082h	Data Pointer low byte	00h
DPXL	S:084h	Data Pointer Extended low byte	01h
MPAGE	S:0A1h	Memory page register	00h
PCON	S:087h	Power Control	00h
PSW	S:0D0h	Program Status Word	00h
PSW1	S:0D1h	Program Status Word 1	00h
SP	S:081h	Stack Pointer low - LSB of SPX	07h
SPH	S:0BEh	Stack Pointer high - MSB of SPX	00h

Table 20: Core SFRs

Mnemonic	Address	Description	Reset value
AIE	S:0E8h	Additional interrupt enable register	00h
AIF	S:0C0h	Additional interrupt flag register	00h
AIPH	S:0F7h	Additional interrupt priority high register	00h
AIPL	S:0F8h	Additional interrupt priority low register	00h
IE0	S:0A8h	Interrupt Enable Control 0	00h
IPH0	S:0B7h	Interrupt Priority Control high byte 0	00h
IPL0	S:0B8h	Interrupt Priority Control low byte 0	00h

Table 21: Interrupt SFRs

Mnemonic	Address	Description	Reset value
P0	S:080h	Port0	FFh
P0_DIR	S:0ACh	Port0 direction	FFh
P1	S:090h	Port1	FFh
P1_DIR	S:0ADh	Port1 direction	FFh
P2	S:0A0h	Port2	FFh
P2_DIR	S:0AEh	Port2 direction	FFh
P3	S:0B0h	Port3	FFh
P3_DIR	S:0AFh	Port3 direction	FFh

Table 22: I/O ports SFRs

Mnemonic	Address	Description	Reset value
SADDR	S:0A9h	Slave Address	00h
SADEN	S:0B9h	Slave Address mask	00h
SBUF	S:099h	Serial Buffer	00h
SCON	S:098h	Serial Control	00h

Table 23: Serial Port SFRs

Mnemonic	Address	Description	Reset value
T2CON	S:0C8h	Timer/Counter 2 control	00h
T2MOD	S:0C9h	Timer/Counter 2 mode control	00h
TCON	S:088h	Timer/Counter 0 and 1 control	00h
TH0	S:08Ch	Timer/Counter 0 high byte	00h
TH1	S:08Dh	Timer/Counter 1 high byte	00h
TH2	S:0CDh	Timer/Counter 2 high byte	00h
TL0	S:08Ah	Timer/Counter 0 low byte	00h
TL1	S:08Bh	Timer/Counter 1 low byte	00h
TL2	S:0CCh	Timer/Counter 2 low byte	00h
TMOD	S:089h	Timer/Counter 0 and 1 mode control	00h
RCAP2H	S:0CBh	Timer 2 Reload/Capture high byte	00h
RCAP2L	S:0CAh	Timer 2 Reload/Capture low byte	00h
WDTCON	S:0A5h	Watchdog Timer control	07h
WDTRST	S:0A6h	Watchdog Timer enable	00h

Table 24: Timers SFRs

Mnemonic	Address	Description	Reset value
CCAP0H	S:0FAh	PCA Compare/Capture Module 0 high byte	00h
CCAP0L	S:0EAh	PCA Compare/Capture Module 0 low byte	00h
CCAP1H	S:0FBh	PCA Compare/Capture Module 1 high byte	00h
CCAP1L	S:0EBh	PCA Compare/Capture Module 1 low byte	00h
CCAP2H	S:0FCh	PCA Compare/Capture Module 2 high byte	00h
CCAP2L	S:0ECh	PCA Compare/Capture Module 2 low byte	00h
CCAP3H	S:0FDh	PCA Compare/Capture Module 3 high byte	00h
CCAP3L	S:0EDh	PCA Compare/Capture Module 3 low byte	00h
CCAP4H	S:0FEh	PCA Compare/Capture Module 4 high byte	00h
CCAP4L	S:0EEh	PCA Compare/Capture Module 4 low byte	00h
CCAPM0	S:0DAh	PCA Compare/Capture Mode for Module 0	00h
CCAPM1	S:0DBh	PCA Compare/Capture Mode for Module 1	00h
CCAPM2	S:0DCh	PCA Compare/Capture Mode for Module 2	00h
CCAPM3	S:0DDh	PCA Compare/Capture Mode for Module 3	00h
CCAPM4	S:0DEh	PCA Compare/Capture Mode for Module 4	00h
CCAPO	S:0DFh	PCA Output for PWM and high speed mode	00h
CCON	S:0D8h	PCA Timer/Counter Control	00h
CH	S:0F9h	PCA Timer/Counter high byte	00h
CL	S:0E9h	PCA Timer/Counter low byte	00h
CMOD	S:0D9h	PCA Timer/Counter Mode	00h

Table 25: Programmable Counter Array (PCA) SFRs

Mnemonic	Address	Description	Reset value
CCMCON	S:08Eh	Communication Control Register	00h
CCMVAL	S:08Fh	Communication Value Register	00h
CPUINFO	S:096h	CPU information (read only register)	00h
MMCON	S:097h	Monitor mode Control register	07h

Table 26: Debug SFRs

Mnemonic	Address	Description	Reset value
PWMC	S:0A2h	PWM Control Register	00h
PWMDCLSB	S:0A3h	PWM Duty Cycle LSB Register	00h
PWMDCMSB	S:0A4h	PWM Duty Cycle MSB Register	00h

Table 27: PWM SFRs

Mnemonic	Address	Description	Reset value
MCON	S:0E1h	I2CM Control register	00h
MRXBUF	S:0E2h	I2CM Reception buffer	00h
MTXBUF	S:0E3h	I2CM Transmission Buffer	00h
MPRESC	S:0E4h	I2CM Pre-scalar clock register	00h
MSTAT0	S:0E5h	I2CM Status register 0	00h
MSTAT1	S:0E6h	I2CM Status register 1	00h
MIEN0	S:0E7h	I2CM Interrupt Enable register 0	00h
MIEN1	S:0D2h	I2CM Interrupt Enable register 1	00h
MCADDR	S:0D4h	I2CM Call Address register	00h

Table 28: I2CM SFRs

Mnemonic	Address	Description	Reset value
STCON	S:0F1h	I2CS Transfer Control register	00h
SRXBUF	S:0F2h	I2CS Reception Buffer	00h
STXBUF	S:0F3h	I2CS Transmission Buffer	00h
SSTAT0	S:0F5h	I2CS Status register 0	00h
SSTAT1	S:0F6h	I2CS Status register 1	00h
SIEN0	S:0D5h	I2CS Interrupt Enable register 0	00h
SIEN1	S:0D6h	I2CS Interrupt Enable register 1	00h
SSADDR	S:0D7h	I2CS Self Address register	00h

Table 29: I2CS SFRs

Mnemonic	Address	Description	Reset value
SPCR	S:0B1h	SPI Control Register	04h
SPDR	S:0B2h	SPI Data Register	00h
SPSR	S:0B3h	SPI Status Register	00h

Table 30: SPI SFRs

Mnemonic	Address	Description	Reset value
CLKCON	S:086h	Clock Control Register	00h
XTALCON	S:085h	Crystal control Register	00h

Table 31: CPMU SFRs

✋*: If a feature or a peripheral is not part of the user’s configuration, the dedicated SFR locations can be freely used for implementing user’s peripheral registers*

4. PROGRAMMING WITH THE FLIP80251

The instruction set for the 8xC251 architecture is a superset of the instruction set for the 80C51 architecture. This chapter describes the addressing modes and summarizes the instruction set, which is divided into data instructions, bit instructions, and control instructions.

4.1 Source mode or Binary mode opcodes

Source mode and *Binary mode* refer to the two ways of assigning opcodes to the instruction set of the 8xC251 architecture. Depending on the application, one mode or the other may produce more efficient code. The mode is determined by the configuration chosen for the Flip80251-Hurricane.
The Flip80251 architecture has two types of instructions:

- Instructions that comes from the 80C51 Architecture
- Instructions that are unique to the 8xC251 Architecture

The Figure 9 shows the opcode map for the binary mode. Area I and II make up the opcode map for the instructions that are unique to the 80C51 Architecture. The opcode values for areas II and III are identical (06h-FFh). To distinguish between the two areas in binary mode, the opcodes in area III are given the prefix A5h (the A5h instruction is not implemented in the native 80C51 Architecture). The area III opcodes are thus A506h-A5FFh.

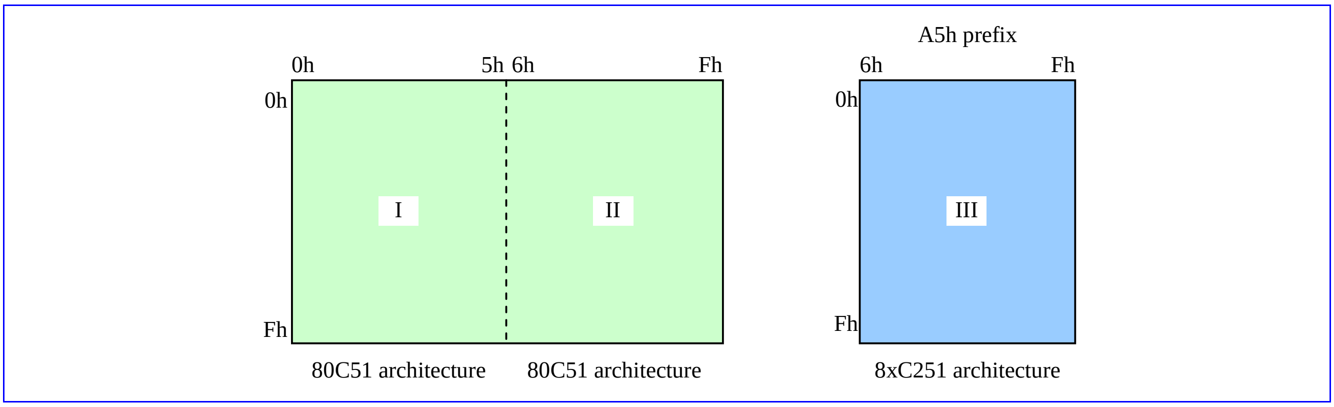


Figure 9: Binary mode opcode map

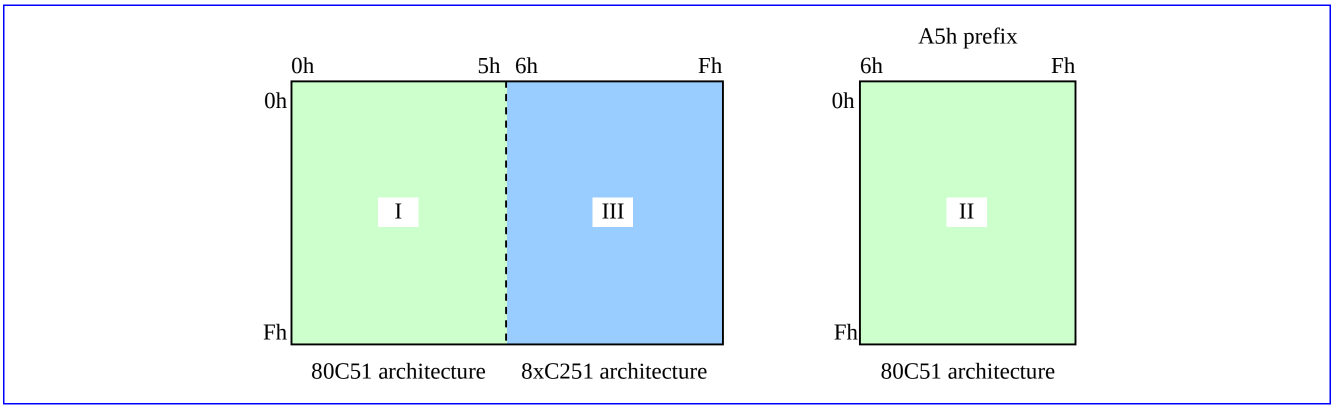


Figure 10: Source mode opcode map

Figure 10 shows the opcode map for source mode. Areas II and III have switched places (compare with the Figure 9)
In source mode, opcodes for instructions in area II require the A5h escape prefix while opcodes for instructions in area III (8xC251 Architecture) do not.
To illustrate the difference between the binary mode and source mode opcodes, the table below shows the opcode assignments for three sample instructions.

Instruction	Opcode	
	Binary mode	**Source mode**
INC A	08h	08h
ADD A, R4	2Ch	A5h 2Ch
ADD R4, R4	A5h 2Ch 44h	2Ch 44h

Table 32: Examples of opcodes in binary and source modes

4.1.1 *Selecting Binary Mode or Source Mode*

If you have code that was written for an 80C51 micro-controller and you want to run it unmodified on the Flip80251 micro-controller, choose *binary mode*. You can use the object code without reassembling the source code.

An instruction with a prefixed opcode requires one more byte for code storage, and if an additional fetch is required for the extra byte, the execution time is increased by one clock cycle. This means that using fewer prefixed opcodes produces more efficient code.
If a program uses only instructions from the 80C51 architecture, the binary-mode code is more efficient because it uses no prefixes. On the other hand, if a program uses many more new instructions than instructions from the 80C51 architecture, source mode is likely to produce more efficient code.

If your program is written in C language, in a vast majority of cases, the **Source mode** enable to generate the more efficient code (code size and speed).

In any case, if the choice is not clear, the better mode can be found by experimenting with an ISS (instruction set simulator).

4.2 Programming features of the C251 architecture

The instruction set for 8xC251 micro-controllers provides the user with new instructions that exploit the features of the architecture while maintaining compatibility with the instruction set for 80C51 micro-controllers. Many of the new instructions operate on 8-bit, 16-bit, or 32-bit operands. (In comparison with 8-bit and 16-bit operands, 32-bit operands are accessed with fewer addressing modes.) This capability increases the ease and efficiency of programming 8xC251 micro-controllers in a high-level language such as C. The instruction set is divided into data, bits, and control instructions. Data instructions process 8-bit, 16-bit, and 32-bit data; bit instructions manipulate bits; and control instructions manage program flow.

4.2.1 Data types

The table below lists the data types that are addressed by the instruction set. Words or dwords (double words) can be stored in memory starting at any byte address; alignment on two–byte or four–byte boundary is not required. Words and Double Words are stored in memory and the register file in big endian form.

Data types	Number of bits
Bit	1
Byte	8
Word	16
Dword (double word)	32

Table 33: Data types

- **Order of byte storage for words and double words**

Flip80251 micro-controllers store words (2 bytes) and double words (4 bytes) in memory and in the register file in big endian form. In memory storage, the most significant byte (MSB) of the word or double word is stored in the memory byte specified in the instruction; the remaining bytes are stored at higher addresses, with the least significant byte (LSB) at the highest address. Words and double words can be stored in memory starting at any byte address. In the register file, the MSB is stored in the lowest byte of the register specified in the instruction. The code in Figure 11 shows the storage of words and double words in big endian form.

4.2.2 Register notations

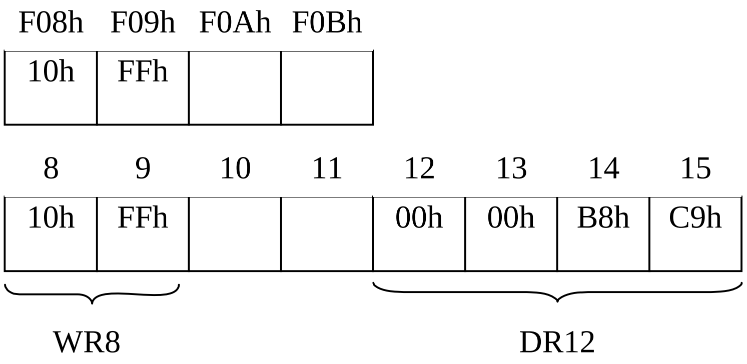
In register–addressing instructions, specific indices denote the registers that can be used in that instruction. For example, the instruction ADD A,Rn uses “Rn” to denote any one of R0, R1, ..., R7; i.e., the range of n is 0-7. The instruction ADD Rm,#data uses “Rm” to denote R0, R1, ..., R15; i.e., the range of m is 0-15. When an instruction contains two registers of the same type (e.g., MOV Rmd, Rms) the first index “d” denotes “destination” and the second index “s” denotes “source”.

Register type	Register ymbol	Destination register	Source register	Register Range
Byte	Ri	-		R0, R1
	Rn	-		R0-R7
	Rm	Rmd	Rms	R0-R15
Word	WRj	WRjd	WRjs	WR0, WR2, WR4,..., WR30
Dword	DRk	DRkd	DRks	DR0, DR4, DR8,…, DR28, DR56, DR60

Table 34: Notation for byte registers, Word registers and Dword registers

4.2.3 Address notations

In the 8xC251 Architecture, memory addresses include a region number (00:, 01:, ..., FF:). SFR addresses have a prefix “S:” (S:000h-S:1FFh). The distinction between memory addresses and SFR addresses is necessary, because memory locations 00:0000h-00:01FFh and SFR locations S:000h-S:1FFh can both be directly addressed in an instruction[1].



Contents of register file and memory after execution: MOV WR8, #10FFh
MOV 00:0F08h, WR8
MOV DR12, #0000B8C9h

Figure 11: Word and Double word storage in Big-endian form

Instructions in the 80C51 Architecture use 80h-FFh as addresses for both memory locations and SFRs, because memory locations are addressed only indirectly and SFR locations are addressed only directly. For compatibility, software tools for Flip80251 controllers recognize this notation for instructions in the 80C51 Architecture. No change is necessary in any code written for 80C51 micro-controllers. For new instructions in the 8xC251 Architecture, the memory region prefixes (00:, 01:, ..., FF:) and the SFR prefix (S:) are required. Also, software tools for the 8xC251 Architecture permit 00: to be used for memory addresses 00h-FFh and permit the prefix S: to be used for SFR addresses in instructions in the 80C51 Architecture.

[1] Locations S:000h-S:07Fh and S:100h-S:1FFh are unimplemented

4.2.4 Addressing modes

The C251 architecture supports the following addressing modes:

Addressing mode	Description
Register addressing	The instruction specifies the register that contains the operand.
Immediate addressing	The instruction contains the operand.
Direct addressing	The instruction contains the operand address.
Indirect addressing	The instruction specifies the register that contains the operand address.
Displacement addressing	The instruction specifies a register and an offset. The operand address is the sum of the register contents (the base address) and the offset.
Relative addressing	The instruction contains the signed offset from the next instruction to the target address (e.g., the jump address).
Bit addressing	The instruction contains the bit address.

Table 35: Flip80251 Addressing mode

4.3 Program Status Words

The Program Status Word (PSW) register and the Program Status Word 1 (PSW) register contain four types of bits.

- CY, AC, OV, N, Z are flags set by hardware to indicate the result of an operation.
- The P bit indicates the parity of the accumulator.
- Bits RS0 and RS1 are programmed by software to select the active register bank for register R0-R7.
- F0 and UD are available to the users as general-purpose flags.

PSW (S:D0h)

Program Status Word Register

BIT	7	6	5	4	3	2	1	0
FIELD	CY	AC	F0	RS1	RS0	OV	UD	P
RESET	0000 0000b							

<table>
<tr><th>Bit Number</th><th>Bit Mnemonic</th><th>Function</th></tr>
<tr><td>7</td><td>CY</td><td><b>Carry Flag:</b>
The carry flag is set by an addition instruction (ADD, ADDC) if there is a carry out of the MSB. It is set by a subtraction (SUB, SUBB) or compare (CMP) if a borrow is needed for the MSB. The carry flag is also affected by some rotate and shift instructions, logical bit instructions, bit move instructions, and the multiply (MUL) and decimal adjust (DA) instructions.</td></tr>
<tr><td>6</td><td>AC</td><td><b>Auxiliary Carry Flag:</b>
The auxiliary carry flag is affected only by instructions that address 8-bit operands. The AC flag is set if an arithmetic instruction with an 8-bit operand produces a carry out of bit 3 (from addition) or a borrow into bit 3 (from subtraction). Otherwise, it is cleared. This flag is useful for BCD arithmetic.</td></tr>
<tr><td>5</td><td>F0</td><td><b>Flag 0:</b>
This general purpose flag is available to the user</td></tr>
<tr><td>4:3</td><td>RS1:0</td><td><b>Register Bank Select bits 1 and 0:</b>
<table>
<tr><th>RS1</th><th>RS0</th><th>Register Bank</th><th>Address</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>00h-07h</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>08h-0Fh</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>10h-17h</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>18h-1Fh</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>OV</td><td><b>Overflow Flag:</b>
This bit is set if an addition or subtraction of signed variables results in an overflow error (i.e., if the magnitude of the sum or difference is too great for the seven LSBs in 2’s-complement representation). The overflow flag is also set if a multiplication product overflows one byte or if a division by zero is attempted.</td></tr>
<tr><td>1</td><td>UD</td><td><b>User-definable flag:</b>
This general purpose flag is available to the user</td></tr>
<tr><td>0</td><td>P</td><td><b>Parity Bit:</b>
This bit indicates the parity of the accumulator. It is set if an odd number of bits in the accumulator is set. Otherwise, it is cleared. Not all instructions update the parity bit. The parity bit is set or cleared by instructions that change the contents of the accumulator (ACC, Register R11).</td></tr>
</table>

Figure 12: Program Status Word Register (PSW)

PSW1 (S:D1h)

Program Status Word 1 Register

BIT	7	6	5	4	3	2	1	0
FIELD	CY	AC	N	RS1	RS0	OV	Z	--
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	CY	**Carry Flag:** Identical to the CY bit in the PSW register
6	AC	**Auxiliary Carry Flag:** Identical to the AC bit in the PSW register
5	N	**Negative Flag:** This bit is set if the result of the last logical or arithmetic operation was negative (i.e. bit 15 = 1). Otherwise it is cleared.
4:3	RS1:0	**Register Bank Select Bits 0 and 1:** Identical to the RS1:0 bits in the PSW register
2	OV	**Overflow Flag:** Identical to the OV bit in the PSW register
1	Z	**Zero Flag:** This flag is set if the result of the last logical or arithmetic operation is zero. Otherwise it is cleared.
0	--	**Reserved**. The value read from this bit is 0.

Figure 13: Program Status Word 1 Register (PSW1)

The PSW and PSW1 registers are read/write registers; however, the parity bit in the PSW is not affected by a write. Individual bits can be addressed with the bit instructions. The PSW and PSW1 bits are used implicitly in the conditional jump instructions.

The PSW register is identical to the PSW register in 80C51 micro-controllers. The PSW1 register exists only in 80251 micro-controllers. Bits CY, AC, RS0, RS1, and OV in PSW1 are identical to the corresponding bits in PSW, i.e., the same bit can be accessed in either register. Table 36 lists the instructions that affect the CY, AC, OV, N and Z bits.

Instruction type	Instruction	Flag affected (1)				
		CY	OV	AC (2)	N	Z
Arithmetic	ADD, ADDC, SUB, CMP	X	X	X	X	X
	INC, DEC				X	X
	MUL, DIV (3)	0	X		X	X
	DA	X			X	X
Logical	ANL, ORL, XRL, CLR A, CPL A, RL, RR, SWAP				X	X
	RLC, RRC, SRL, SLL, SRA (4)	X			X	X
Program Control	CJNE	X			X	X
	DJNZ				X	X

Table 36: The effects of instructions on the PSW and PSW1 flags

1. X = the flag can be affected by the instruction. 0 = the flag is cleared by the instruction.
2. The AC flag is affected only by operations on 8–bit operands.
3. If the divisor is zero, the OV flag is set, and the other bits are meaningless.
4. For SRL, SLL and SRA instructions, the last bit shifted out is stored in the CY bit.

4.4 Data Instructions

Data instructions consist of arithmetic, logical, and data-transfer instructions for 8-bit, 16-bit, and 32-bit data. This section describes the data addressing modes and the set of data instructions.

4.4.1 Data Addressing Mode

References to registers R0–R7, WR0–WR6, DR0, and DR2 always refer to the register bank that is currently selected by the PSW and PSW1 registers (see Program Status Words section). Registers in all banks (active and inactive) can be accessed as memory locations in the range 00h-1Fh. Instructions from the 80C51 architecture access external memory through the region of memory specified by byte DPXL in the extended data pointer register, DPX (DR56). Following reset, DPXL contains 01h, which maps the external memory to region 01:. You can specify a different region by writing to DR56 or the DPXL SFR.

4.4.1.1 Register addressing

- 8xC251 architecture
  In the register addressing mode, the operand(s) in a data instruction are in byte registers (R0-R15), word registers (WR0, WR2, ..., WR30), or dword registers (DR0, DR4, ..., DR28, DR56, DR60).
- 80C51 architecture
  Instructions address registers R0–R7 only.

4.4.1.2 Immediate addressing

- 8xC251 architecture
  In the immediate addressing mode, the instruction contains the data operand itself. Byte operations use 8-bit immediate data (#data); word operations use 16-bit immediate data (#data16). Dword operations use 16-bit immediate data in the lower word, and either zeros in the upper word (denoted by #0data16), or ones in the upper word (denoted by #1data16). Move instructions that place 16-bit immediate data into a dword register (DRk), place the data either into the upper word while leaving the lower word unchanged (MOVH), or into the lower word with a sign extension (MOVS) or a zero extension (MOVZ). The increment and decrement instructions contain immediate data (#short = 1, 2, or 4) that specifies the amount of the increment/decrement.
- 80C51 architecture
  Instructions use only 8-bit immediate data (#data).

4.4.1.3 Direct addressing

- 8xC251 architecture
  In the direct addressing mode, the instruction contains the address of the data operand. The 8-bit direct mode addresses on-chip RAM (dir8 = 00:0000h-00:007Fh) as both bytes and words, and addresses the SFRs (dir8 = S:080h-S:0FFh) as bytes only. The 16-bit direct mode addresses both bytes and words in memory (dir16 = 00:0000h-00:FFFFh).

- 80C51 architecture
  The 8-bit direct mode addresses 128 bytes of on-chip RAM (dir8 = 00h-7Fh) as bytes only and the SFRs (dir8 = 80h-FFh) as bytes only.

Mode	Address range of operand	Assembly language Reference	Comments
Register	00:0000h-00:001Fh	R0-R7 (Bank selected by PSW)	
Immediate	Operand in Instruction	#data = #00h-#FFh	
Direct	00:0000h-00:007Fh	dir8 = 00h-7Fh	Internal RAM
	SFRs	dir8 = 80h-FFh or SFR mnemonic	SFR address
Indirect	00:0000h-00:00FFh	@R0, @R1	Accesses internal RAM
	00:0000h-7F:FFFFh[6]	@DPTR, @R0, @R1	Accesses external data memory (MOVX)
	80:0000h-FF:FFFFh	@A+DPTR, @A+PC	Accesses code memory (MOVC)

Table 37: Addressing Modes for Data instruction in the 80C51 architecture

4.4.1.4 Indirect addressing

In arithmetic and logical instructions that use indirect addressing, the source operand is always a byte, and the destination is either the accumulator or a byte register (R0–R15). The source address is a byte, word, or dword. The two architectures do indirect addressing via different registers:

- 8xC251 architecture
  Memory is indirectly addressed via word and dword registers:
  - Word register (@WRj, j = 0, 2, 4, ..., 30).
    The 16-bit address in WRj can access locations 00:0000h-00:FFFFh.
  - Dword register (@DRk, k = 0, 4, 8, ..., 28, 56, and 60).
    The 24 least significant bits can access the entire 16 Mbytes address space. The upper eight bits of DRk must be 0. If you use DR60 as a general data pointer, be aware that DR60 is the extended stack pointer register SPX.
- 80C51 architecture
  Instructions use indirect addressing to access on-chip RAM, code memory, and external data RAM. Instructions access external memory through the region of memory specified by byte DPXL in the extended data pointer register, DPX.
  - Byte register (@Ri, i = 0,1).
    Registers R0 and R1 indirectly address on-chip memory locations 00h-FFh. When MOVX instructions use this indirect mode, the MSB of the16-bit address is filled with the content of MPAGE register (S: 0A1h). Then, it allows MOVX @Ri instruction to access to 64 Kbytes of external data memory. Usually, in 80C51 application, the Port 2 is used to this address extension. In order to keep software compatibility with existing 80C51 program, the register MPAGE is also updated by any value written at P2 register.
  - 16-bit data pointer (@DPTR).
    The MOVX instructions use these indirect modes to access the page of the external data RAM pointed by the extended data pointer (DPX).

[6] If the programmer try to access to the program memory (region above 80:0000h) with a MOVX instruction, the address wraps around in the data memory space (00:0000h-7F:FFFFh) and a data memory access is performed.

- 16-bit data pointer (@A+DPTR).
  The MOVC instructions use these indirect modes to access the current 64 K page of the code memory.
- 16-bit program counter (@A+PC).
  The MOVC instruction uses this indirect mode to access the current 64 K page of the code memory.

4.4.1.5 Displacement addressing

Several move instructions use displacement addressing to move bytes or words from a source to a destination. Sixteen-bit displacement addressing (@WRj+dis16) accesses indirectly the lowest 64 Kbytes in memory. The base address can be in any word register WRj. The instruction contains a 16-bit signed offset that is added to the base address. Only the lowest 16 bits of the sum are used to compute the operand address. If the sum of the base address and a positive offset exceeds FFFFh, the computed address wraps around within region 00: (e.g. F000h + 2005h becomes 1005h). Similarly, if the sum of the base address and a negative offset is less than zero, the computed address wraps around the top of region 00: (e.g., 2005h + F000h becomes 1005h).
24-bit displacement addressing (@DRk+dis24) accesses indirectly the entire 16 Mbytes address space. The base address must be in DR0, DR4, ..., DR24, DR28, DR56, or DR60. The instruction contains a 16-bit signed offset that is added to the base address.

Mode	Address range of operand	Assembly language Reference	Comments
Register	[00:0000h-00:001Fh]	R0-R15, WR0-WR30, DR0-DR28, DR56, DR60	R0-R7, WR0-WR6, DR0-DR4 are in the register bank currently selected by the PSW and the PSW1
Immediate 2 bits	(N.A) Operand in Instruction	#short = 1, 2 or 4	Used only in increment and decrement instruction
Immediate 8 bits	(N.A) Operand in Instruction	#data8 = #00h-#FFh	
Immediate 16 bits	(N.A) Operand in Instruction	#data16 = #0000h-#FFFFh	
Direct, 8 address bits	[00:0000h-00:007Fh]	dir8 = 00:0000h-00:0007Fh	Internal RAM
	SFRs	dir8 = S:080h-S:0FFh or SFR mnemonic	SFR address
Direct, 16 address bits	[00:0000h-00:FFFFh]	dir16 = 00:0000h-00:FFFFh	
Indirect, 16 address bits	[00:0000h-00:FFFFh]	@WR0-@WR30	
Indirect, 24 address bits (*)	[00:0000h-FF:FFFFh]	@DR0-@DR28, @DR56, @DR60	
Displacement, 16 address bits	[00:0000h-00:FFFFh]	@WRj+dis16 = @WR0+0h through @WR30+FFFFh	Offset is signed, address wraps around in region 00:
Displacement, 24 address bits (*)	[00:0000h-FF:FFFFh]	@DRk+dis24= @DR0+0h through @DR28+FFFFh, @DR56+(0-FFFFh), @DR60+(0-FFFFh)	Offset is signed.

Table 38: Addressing Modes for Data instruction in the Flip80251 architecture

(*) A 24-bit indirect or displacement access at address below 80:0000h is a read or write access to the data memory, while a 24-bit indirect or displacement access at address above 80:0000h is a read or write access to the program memory.

☞ In case of word access at data memory boundary (7F:FFFFh) the second byte access will be performed at the next address (80:0000h) and then activate the program memory control signals. The address **does not** wrap around in region 00:.

4.4.2 Arithmetic instruction

The set of arithmetic instructions is greatly expanded in the 8xC251 architecture. The ADD and SUB instructions operate on byte and word data that is accessed in several ways:

- as the contents of the accumulator, a byte register (Rn), or a word register (WRj)
- in the instruction itself (immediate data)
- in memory via direct or indirect addressing

The ADDC and SUBB instructions are the same as those for 80C51 micro-controllers.
The CMP (compare) instruction calculates the difference of two bytes or words and then writes to flags CY, OV, AC, N, and Z in the PSW and PSW1 registers. The difference is not stored. The operands can be addressed in a variety of modes. The most frequent use of CMP is to compare data or addresses preceding a conditional jump instruction.
The INC (increment) and DEC (decrement) instructions for 80C51 micro-controllers are supplemented by instructions that can address byte, word, and dword registers and increment or decrement them by 1, 2, or 4 (denoted by #short). These instructions are supplied primarily for register-based address pointers and loop counters.

The 8xC251 architecture provides the MUL (multiply) and DIV (divide) instructions for unsigned 8-bit and 16-bit data. Signed multiply and divide are left for the user to manage through a conversion process. The following operations are implemented:

- eight-bit multiplication: 8 bits × 8 bits → 16 bits
- sixteen-bit multiplication: 16 bits × 16 bits → 32 bits
- eight-bit division: 8 bits ÷ 8 bits → 16 bits (8-bit quotient, 8-bit remainder)
- sixteen-bit division: 16 bits ÷ 16 bits → 32 bits (16-bit quotient, 16-bit remainder)

These instructions operate on pairs of byte registers (Rmd,Rms), word registers (WRjd,WRjs), or the accumulator and B register (A,B).
For 8-bit register multiplies, the result is stored in the word register that contains the first operand register. For example, the product from an instruction MUL R2, R9 is stored in WR2. Similarly, for 16-bit multiplies, the result is stored in the dword register that contains the first operand register. For example, the product from the instruction MUL WR7, WR16 is stored in DR4.

For 8-bit divides, the operands are byte registers. The result is stored in the word register that contains the first operand register. The quotient is stored in the lower byte, and the remainder is stored in the higher byte. A 16-bit divide is similar. The first operand is a word register, and the result is stored in the double word register that contains that word register. If the second operand (the divisor) is zero, the overflow flag (OV) is set and the other bits in PSW and PSW1 are meaningless.

4.4.3 Logical instruction

The 8xC251 architecture provides a set of instructions that perform logical operations. The ANL, ORL, and XRL (logical AND, logical OR, and logical exclusive OR) instructions operate on bytes and words that are accessed via several addressing modes. A byte register, word register, or the accumulator can be logically combined with a register, immediate data, or data that is addressed directly or indirectly. These instructions affect the Z and N flags.

In addition to the CLR (clear), CPL (complement), SWAP (swap), and four rotate instructions that operate on the accumulator, 8xC251 micro-controllers have three shift commands for byte and word registers:

- SLL (Shift Left Logical) shifts the register one bit left and replaces the LSB with 0
- SRL (Shift Right Logical) shifts the register one bit right and replaces the MSB with 0
- SRA (Shift Right Arithmetic) shifts the register one bit right; the MSB is unchanged

4.4.4 Data Transfer instruction

Data transfer instructions copy data from one register or memory location to another. These instructions include the move instructions and the exchange, push, and pop instructions. Instructions that move only a single bit are listed with the other bit instructions in the chapter 4.5.

MOV (Move) is the most versatile instruction, and its addressing modes are expanded in the 8xC251 architecture. MOV can transfer a byte, word, or dword between any two registers or between a register and any location in the address space.

The MOVX (Move External) instruction moves a byte from external memory to the accumulator or from the accumulator to memory. The external memory is in the region specified by DPXL, whose reset value is 01h.” The MOVC (Move Code) instruction moves a byte from code memory to the accumulator. MOVS (Move with Sign Extension) and MOVZ (Move with Zero Extension) move the contents of an 8-bit register to the lower byte of a 16-bit register. The upper byte is filled with the sign bit (MOVS) or zeros (MOVZ). The MOVH (Move to High Word) instruction places 16-bit immediate data into the high word of a dword register.

The XCH (Exchange) instruction interchanges the contents of the accumulator with a register or memory location. The XCHD (Exchange Digit) instruction interchanges the lower nibble of the accumulator with the lower nibble of a byte in on-chip RAM. XCHD is useful for BCD (binary coded decimal) operations.

The PUSH and POP instructions facilitate storing information (PUSH) and then retrieving it (POP) in reverse order. Push can push a byte, a word, or a dword onto the stack, using the immediate, direct, or register addressing modes. POP can pop a byte or a word from the stack to a register or to memory.

4.5 Bit Instructions

A bit instruction addresses a specific bit in a memory location or SFR. There are four categories of bit instructions:

- SETB (Set Bit), CLR (Clear Bit), CPL (Complement Bit). These instructions can set, clear or complement any addressable bit.

- ANL (And Logical), ANL/ (And Logical Complement), ORL (OR Logical), ORL/ (Or Logical Complement). These instructions allow ANDing and ORing of any addressable bit or its complement with the CY flag.
- MOV (Move) instructions transfer any addressable bit to the carry (CY) bit or vice versa.
- Bit-conditional jump instructions execute a jump if the bit has a specified state. The bit conditional jump instructions are classified with the control instructions and are described in the following Conditional Jumps section.

4.5.1 Bit addressing

The bits that can be individually addressed are in the on-chip RAM and the SFRs. The bit instructions that are unique to the 8xC251 architecture can address a wider range of bits than the instructions from the 80C51 architecture.

There are some differences in the way the instructions from the two architectures address bits. In the 80C51 architecture, a bit (denoted by bit51) can be specified in terms of its location within a certain register, or it can be specified by a bit address in the range 00h-7Fh. The 8xC251 architecture does not have bit addresses as such. A bit can be addressed by name or by its location within a certain register, but not by a bit address.

Table 39 illustrates bit addressing in the two architectures by using two sample bits:

- RAM_REG_b2 is bit 2 in RAM_REG, which is location 2Fh. “RAM_REG_b2” and “RAM_REG” are assumed to be defined in user code.
- TF0 is bit 5 in TCON, which is an SFR at location 88h.

Architecture	**Bit-addressable locations**	
	On-chip RAM	**SFRs**
8xC251	20h-7Fh	All defined SFRs
80C51	20h-2Fh	SFRs with address ending in 0h or 8h (e.g. 88h, A0h,…)

Table 39: Bit-addressable Locations

☞ *:“Bit” denotes a bit that is addressed by a new instruction in the MCS 251 architecture and “bit51” denotes a bit that is addressed by an instruction in the MCS 51 architecture.*

Location	**Adressing Mode**	**80C51 Architecture**	**8xC251 Architecture**
On-chip RAM	Register Name	RAM_REG.2	RAM_REG.2
	Register Address	2Fh.2	2Fh.2
	Bit Name	RAM_REG_b2	RAM_REG_b2
	Bit Address	7Ah	N.A.
SFR	Register Name	TCON.5	TCON.5
	Register Address	88.5h	88.5h
	Bit Name	TF0	TF0
	Bit Address	8Dh	N.A.

Table 40: Example of Bit addressing

Architecture	Variants	Bit Address	Memory/SFR Address
8xC251 (bit)	Memory	NA	20h.0-7Fh.7
	SFR	NA	All defined SFRs
80C51 (bit)	Memory	00h-7Fh	20h.0-2Fh.7
	SFR	80h-FFh	XXh.0-XXh.7, where XX= 80, 88, 90, 98, A0, A8, B0, B8, C0, C8, D0, D8, E0, E8, F0 or F8

Table 41: Addressing Modes for Bit Instructions

4.6 Control Instructions

Control instructions “instructions that change program flow” include calls, returns, and conditional and unconditional jumps. Instead of executing the next instruction in the queue, the processor executes a target instruction. The control instruction provides the address of a target instruction either implicitly, as in a return from a subroutine, or explicitly, in the form of a relative, direct, or indirect address.

The Flip80251-Hurricane have a 24-bit program counter (PC), which allows a target instruction to be anywhere in the 16 Mbytes address space, but only the upper 8 Mbytes are assumed to be Code memory (80:0000h to FF:FFFFh). As discussed in this section, some control instructions restrict the target address to the current 2 Kbytes or 64 Kbytes address range by allowing only the lowest 11 or lowest 16 bits of the program counter to change.

4.6.1 Addressing mode for Control Instructions

- Relative addressing:
  The control instruction provides the target address as an 8-bit signed offset (rel) from the address of the next instruction.
- Direct addressing:
  The control instruction provides a target address, which can have 11 bits (addr11), 16 bits (addr16), or 24 bits (addr24). The target address is written to the PC.
  - addr11: Only the lower 11 bits of the PC are changed; i.e., the target address must be in the 2-Kbyte block that includes the first byte of the next instruction.
  - addr16: Only the lower 16 bits of the PC are changed; i.e., the target address must be in the 64-Kbyte region that includes the first byte of the next instruction.
  - addr24: The target address can be anywhere in the 16-Mbyte address space.
- Indirect addressing:
  There are two types of indirect addressing for control instructions:
  - For the instructions LCALL @WRj and LJMP @WRj, the target address is in the current 64 Kbytes region. The 16-bit address in WRj is placed in the lower 16 bits of the PC. The upper eight bits of the PC remain unchanged from the address of the next instruction.
  - For the instruction JMP @A+DPTR, the sum of the accumulator and DPTR is placed in the lower 16 bits of the PC. The upper eight bits of the PC remain unchanged from the address of the next instruction.

✋: ***This is a different behavior for the*** **JMP @A+DPTR** ***instruction than the standard C251, where this instruction always point to region FF:.***

Description	Address bits provided	Address range
Relative, 8-bit relative address	8	-128 to 127 from first byte of next instruction
Direct, 11-bit address (addr11)	11	Current 2 Kbytes
Direct, 16-bit address (addr16)	16	Current 64 Kbytes
Direct, 24-bit address (addr24) [7]	24	00:0000h-FF:FFFFh
Indirect (@WRj) [1]	16	Current 64 Kbytes
Indirect (@A+DPTR)	16	Current 64 Kbytes

Table 42: Addressing Modes for Control Instructions

4.6.2 Conditional Jumps

The 8xC251 architecture supports bit-conditional jumps, compare-conditional jumps, and jumps based on the value of the accumulator. A bit-conditional jump is based on the state of a bit. In a compare-conditional jump, the jump is based on a comparison of two operands. All conditional jumps are relative, and the target address (rel) must be in the current 256-byte block of code. The instruction set includes three kinds of bit-conditional jumps:

- JB (Jump on Bit): Jump if the bit is set.
- JNB (Jump on Not Bit): Jump if the bit is clear.
- JBC (Jump on Bit then Clear it): Jump if the bit is set; then clears it.

Compare-conditional jumps test a condition resulting from a compare (CMP) instruction that is assumed to precede the jump instruction. The jump instruction examines the PSW and PSW1 registers and interprets their flags as though they were set or cleared by a compare (CMP) instruction. Actually, the state of each flag is determined by the last instruction that could have affected that flag. The condition flags are used to test one of the following six relations between the operands:

- equal (=), not equal (≠)
- greater than (>), less than (<)
- greater than or equal (≥), less than or equal (≤)

For each relation there are two instructions, one for signed operands and one for unsigned operands (see table below).

Operand Type	Relation					
	=	≠	>	<	≥	≤
Unsigned	JE	JNE	JG	JL	JGE	JLE
Signed			JSG	JSL	JSGE	JSLE

Table 43: Compare-conditional Jump Instructions

[7] These modes are not used by instructions in the C51 architecture

4.6.3 Unconditional Jumps

There are five unconditional jumps. NOP and SJMP jump to addresses relative to the program counter. AJMP, LJMP, and EJMP jump to direct or indirect addresses.

- NOP (No Operation) is an unconditional jump to the next instruction.
- SJMP (Short Jump) jumps to any instruction within -128 to 127 of the next instruction.
- AJMP (Absolute Jump) changes the lowest 11 bits of the PC to jump anywhere within the current 2-Kbyte block of memory. The address can be direct or indirect.
- LJMP (Long Jump) changes the lowest 16 bits of the PC to jump anywhere within the current 64-Kbyte region.
- EJMP (Extended Jump) changes all 24 bits of the PC to jump anywhere in the 16-Mbyte address space. The address can be direct or indirect.

4.6.4 Call and Returns

The 8xC251 architecture provides relative, direct, and indirect calls and returns.

- ACALL (Absolute Call) pushes the lower 16 bits of the next instruction address onto the stack and then changes the lower 11 bits of the PC to the 11-bit address specified by the instruction. The call is to an address that is in the same 2-Kbyte block of memory as the address of the next instruction.
- LCALL (Long Call) pushes the lower 16 bits of the next-instruction address onto the stack and then changes the lower 16 bits of the PC to the 16-bit address specified by the instruction. The call is to an address in the same 64-Kbyte block of memory as the address of the next instruction.
- ECALL (Extended Call) pushes the 24 bits of the next instruction address onto the stack and then changes the 24 bits of the PC to the 24-bit address specified by the instruction. The call is to an address anywhere in the 16 Mbytes memory space.
- RET (Return) pops the top two bytes from the stack to return to the instruction following a subroutine call. The return address must be in the same 64-Kbyte region.
- ERET (Extended Return) pops the top three bytes from the stack to return to the address following a subroutine call. The return address can be anywhere in the 16-Mbyte address space.
- RETI (Return from Interrupt) provides a return from an interrupt service routine. The operation of RETI depends on the *intrmode* input pin level:
    - When *intrmode* is low, an interrupt pushes the two lower bytes of the PC onto the stack in the following order: PC.7:0, PC.15:8. The RETI instruction pops these two bytes and uses them as the 16-bit return address in region FF:. RETI also restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed.
    - When *intrmode* is high, an interrupt pushes the three PC bytes and PSW1 onto the stack in the following order: PSW1, PC.23:16, PC.7:0, PC.15:8. The RETI instruction pops these four bytes and then returns to the specified 24-bit address, which can be anywhere in the 16 Mbytes address space. RETI also clears the interrupt request.

✋: *The TRAP instruction is usually reserved for the development of emulation features.*

5. INSTRUCTION SET REFERENCE

This chapter contains reference for the instructions in the 8xC251 architecture. It includes opcode map and a summary of the instructions with instruction lengths and execution time.

The instruction execution times are given for code executing from a code memory and for data that is read from and written to a data memory that are both assumed to be fast enough to be accessed within one clock cycle. Accessing slower memories that need to insert wait states increases execution times.

5.1 Addressing mode acronyms

Addressing mode	Description	8xC251	80C51
dir8	An 8-bit direct address. This can be a memory address (00:0000h-00:00FFh) or an SFR address (S:00h - S:0FFh).	**X**	**X**
dir16	A 16-bit memory address (00:0000h-00:FFFFh) used in direct addressing.	**X**	

Table 44: Notation for Direct Addressing

Addressing mode	Description	8xC251	80C51
#data	An 8-bit constant that is immediately addressed in an instruction.	**X**	**X**
#data 16	A 16-bit constant that is immediately addressed in an instruction.	**X**	
#0data16 **#1data16**	A 32-bit constant that is immediately addressed in an instruction. The upper word is filled with zeros (#0data16) or ones (#1data16).	**X**	
#short	A constant, equal to 1, 2, or 4, that is immediately addressed in an instruction.	**X**	

Table 45: Notation for Immediate Addressing

Addressing mode	Description	8xC251	80C51
bit51	Direct addressable bit (bit number = 00h-FFh) in internal data RAM or a special function register. Bits 00h-7Fh are the 128 bits at byte address 20h-2Fh in the internal RAM. Bits 80h-FFh are the 128 bits in the 16 SFR’s with addresses that end in 0h or 8h: S:80h, S:88h, S:90h, ..., S:F0h, S:F8h.		**X**
bit	A directly addressable bit in memory locations 00:0020h-00:007Fh or in any defined SFR.	**X**	

Table 46: Notation for Bit Addressing

Addressing mode	Description	8xC251	80C51
rel	A signed (two's complement) 8-bit relative address. The destination is -128 to +127 bytes relative to first byte of the next instruction.	X	X
addr11	An 11-bit destination address. The destination is in the same 2-Kbyte block of memory as the first byte of the next instruction.		X
addr16	A 16-bit destination address. A destination can be anywhere within the same 64-Kbyte region as the first byte of the next instruction.		X
addr24	A 24-bit destination address. A destination can be anywhere within the 16-Mbyte address space.	X	

Table 47: Notation for Control Instructions

Addressing mode	Description	8xC251	80C51
@Ri	Internal data location (00h-FFh) addressed indirectly through byte registers R0 or R1.		X
Rn **n**	Register R0-R7 of the currently selected register bank. Byte register index: n=0-7		X
Rm **Rmd** **Rms** **m, md, ms**	Byte register R0-R15 of the register file Destination register Source register Byte register index: m, md, ms=0-15	X	
WRj **WRjd** **WRjs** **@WRj** **@WRj+dis16** **j, js, jd**	Word Register WR0, WR2,…,WR30 of the register file Destination register Source register A memory location (00:0000h-00:FFFFh) addressed indirectly through word register WR0-WR30 A memory location (00:0000h-00:FFFFh) addressed indirectly through word register (WR0-WR30) + displacement value, where the displacement value is from 0 to 64 Kbytes. Word register index: j, js, jd=0-30	X	
DRk **DRkd** **DRks** **@DRk** **@DRk+dis24** **k, kd, ks**	Dword register DR0, DR4, ..., DR28, DR56, DR60 of the register file Destination Register Source Register A memory location (00:0000h-FF:FFFFh) addressed Indirectly through dword register DR0-DR28, DR56, DR60 A memory location (00:0000h-FF:FFFFh) addressed Indirectly through dword register (DR0-DR28, DR56, DR60) + displacement value, where the displacement value is from 0 to 64 Kbytes Dword register index: k, kd, ks = 0, 4, 8, ..., 28, 56, 60	X	

Table 48: Notation for Register Operands

5.2 Size and execution time summary

5.2.1 Arithmetic Instructions

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
ADD A, Rn ([8])	1	2	2	4	1	1
ADD A, dir8	2	2	2	2	3	3
ADD A, @Ri	1	2	4	6	3	3
ADD A, #data	2	2	2	2	1	1
ADDC A, Rn	1	2	2	4	1	1
ADDC A, dir8	2	2	2	2	3	3
ADDC A, @Ri	1	2	4	6	3	3
ADDC A, #data	2	2	2	2	1	1
SUBB A, Rn	1	2	2	4	1	1
SUBB A, dir8	2	2	2	2	3	3
SUBB A, @Ri	1	2	4	6	3	3
SUBB A, #data	2	2	2	2	1	1
ADD Rmd, Rms	3	2	4	4	1	1
ADD WRjd, WRjs	3	2	6	4	1	1
ADD DRkd, DRks	3	2	10	4	2	2
ADD Rm, #data	4	3	6	6	1	1
ADD WRj, #data16	5	4	8	8	1	1
ADD DRk, #0data16	5	4	12	8	2	2
ADD Rm, dir8	4	3	6	6	3	3
ADD WRj, dir8	4	3	8	6	3	3
ADD Rm, dir16	5	4	6	8	3	3
ADD WRj, dir16	5	4	8	8	3	3
ADD Rm, @WRj	4	3	6	6	3	3
ADD Rm, @DRk	4	3	8	6	3	3
SUB Rmd, Rms	3	2	4	4	1	1
SUB WRjd, WRjs	3	2	6	4	1	1
SUB DRkd, DRks	3	2	10	4	2	2
SUB Rm, #data	4	3	6	6	1	1
SUB WRj, #data16	5	4	8	8	1	1
SUB DRk, #0data16	5	4	12	8	2	2
SUB Rm, dir8	4	3	6	6	3	3
SUB WRj, dir8	4	3	8	6	3	3
SUB Rm, dir16	5	4	6	8	3	3
SUB WRj, dir16	5	4	8	8	3	3
SUB Rm, @WRj	4	3	6	6	3	3
SUB Rm, @DRk	4	3	8	6	3	3

Table 49: Summary of Add and Subtract Instructions

[8] A shaded cell denotes an 80C51 instruction.

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
INC A ([9])	1	1	2	2	1	1
INC Rn	1	2	2	4	1	1
INC dir8	2	2	4	4	4	4
INC @Ri	1	2	6	8	4	4
DEC A	1	1	2	2	1	1
DEC Rn	1	2	2	4	1	1
DEC dir8	2	2	4	4	4	4
DEC @Ri	1	2	6	8	4	4
INC DPTR	1	1	2	2	1	1
INC Rm, #short	3	2	4	2	1	1
INC WRj, #short	3	2	4	2	1	1
INC DRk, #short	3	2	8	6	2	2
DEC Rm, #short	3	2	4	2	1	1
DEC WRj, #short	3	2	4	2	1	1
DEC DRk, #short	3	2	10	8	2	2

Table 50: Summary of Increment and Decrement Instructions

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
MUL AB	1	1	10	10	2	2
DIV AB	1	1	20	20	10	10
DA A	1	1	2	2	1	1
MUL Rmd, Rms	3	2	12	10	2	2
MUL WRjd, WRjs	3	2	24	22	8	8
DIV Rmd, Rms	3	2	22	20	10	10
DIV WRjd, WRjs	3	2	42	40	19	19

Table 51: Summary of Multiply, Divide and Decimal-adjust Instructions

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
CMP Rmd, Rms	3	2	4	2	1	1
CMP WRjd, WRjs	3	2	6	4	1	1
CMP DRkd, DRks	3	2	10	8	2	2
CMP Rm, #data	4	3	6	4	1	1

[9] A shaded cell refers to an 80C51 instruction.

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
CMP WRj, #data16	5	4	8	6	1	1
CMP DRk, #0data16	5	4	12	10	2	2
CMP DRk, #1data16	5	4	12	10	2	2
CMP Rm, dir8	4	3	6	4	3	3
CMP WRj, dir8	4	3	8	6	3	3
CMP Rm, dir16	5	4	6	4	3	3
CMP WRj, dir16	5	4	8	6	3	3
CMP Rm, @WRj	4	3	6	4	3	3
CMP Rm, @DRk	4	3	8	6	3	3

Table 52: Summary of Compare Instructions

5.2.2 Logical Instructions

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
ANL A, Rn ([10])	1	2	2	4	1	1
ANL A, dir8	2	2	2	2	3	3
ANL A, @Ri	1	2	4	6	3	3
ANL A, #data	2	2	2	2	1	1
ANL dir8, A	2	2	4	4	4	4
ANL dir8, #data	3	3	6	6	4	4
ORL A, Rn	1	2	2	4	1	1
ORL A, dir8	2	2	2	2	3	3
ORL A, @Ri	1	2	4	6	3	3
ORL A, #data	2	2	2	2	1	1
ORL dir8, A	2	2	4	4	4	4
ORL dir8, #data	3	3	6	6	4	4
XRL A, Rn	1	2	2	4	1	1
XRL A, dir8	2	2	2	2	3	3
XRL A, @Ri	1	2	4	6	3	3
XRL A, #data	2	2	2	2	1	1
XRL dir8, A	2	2	4	4	4	4
XRL dir8, #data	3	3	6	6	4	4
CLR A	1	1	2	2	1	1
CPL A	1	1	2	2	1	1
RL A	1	1	2	2	1	1
RLC A	1	1	2	2	1	1

[10] A shaded cell refers to an 80C51 instruction.

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
RR A	1	1	2	2	1	1
RRC A	1	1	2	2	1	1
SWAP A	1	1	4	4	1	1
ANL Rmd, Rms	3	2	4	2	1	1
ANL WRjd, WRjs	3	2	6	4	1	1
ANL Rm, #data	4	3	6	4	1	1
ANL WRj, #data16	5	4	8	6	1	1
ANL Rm, dir8	4	3	6	4	3	3
ANL WRj, dir8	4	3	8	6	3	3
ANL Rm, dir16	5	4	6	4	3	3
ANL WRj, dir16	5	4	8	6	3	3
ANL Rm, @WRj	4	3	6	4	3	3
ANL Rm, @DRk	4	3	8	6	3	3
ORL Rmd, Rms	3	2	4	2	1	1
ORL WRjd, WRjs	3	2	6	4	1	1
ORL Rm, #data	4	3	6	4	1	1
ORL WRj, #data16	5	4	8	6	1	1
ORL Rm, dir8	4	3	6	4	3	3
ORL WRj, dir8	4	3	8	6	3	3
ORL Rm, dir16	5	4	6	4	3	3
ORL WRj, dir16	5	4	8	6	3	3
ORL Rm, @WRj	4	3	6	4	3	3
ORL Rm, @DRk	4	3	8	6	3	3
XRL Rmd, Rms	3	2	4	2	1	1
XRL WRjd, WRjs	3	2	6	4	1	1
XRL Rm, #data	4	3	6	4	1	1
XRL WRj, #data16	5	4	8	6	1	1
XRL Rm, dir8	4	3	6	4	3	3
XRL WRj, dir8	4	3	8	6	3	3
XRL Rm, dir16	5	4	6	4	3	3
XRL WRj, dir16	5	4	8	6	3	3
XRL Rm, @WRj	4	3	6	4	3	3
XRL Rm, @DRk	4	3	8	6	3	3
SLL Rm	3	2	4	2	1	1
SLL WRj	3	2	4	2	1	1
SRA Rm	3	2	4	2	1	1
SRA WRj	3	2	4	2	1	1
SRL Rm	3	2	4	2	1	1
SRL WRj	3	2	4	2	1	1

Table 53: Summary of Logical Instructions (continued)

5.2.3 Data transfer

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
MOVC A, @A+DPTR ([11])	1	1	12	12	2	2
MOVC A, @A+PC	1	1	12	12	2	2
MOVX A, @Ri	1	1	8	10	3	3
MOVX A,@DPTR	1	1	6	6	3	3
MOVX @Ri, A	1	1	8	8	2	2
MOVX @DPTR, A	1	1	8	8	2	2
MOVH DRk, #data16	5	4	6	4	1	1
MOVS WRj, Rm	3	2	4	2	1	1
MOVZ WRj, Rm	3	2	4	2	1	1

Table 54: Summary of Move Instructions (1/3)

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
MOV A, Rn	1	2	2	4	1	1
MOV A, dir8	2	2	2	2	3	3
MOV A, @Ri	1	2	4	6	3	3
MOV A, #data	2	2	2	2	1	1
MOV Rn, A	1	2	2	4	1	1
MOV Rn, dir8	2	3	2	4	3	3
MOV Rn, #data	2	3	2	4	1	1
MOV dir8, A	2	2	4	4	2	2
MOV dir8, Rn	2	3	4	6	2	2
MOV dir8, dir8	3	3	6	6	4	4
MOV dir8, @Ri	2	3	6	8	4	4
MOV dir8, #data	3	3	6	6	2	2
MOV @Ri, A	1	2	6	8	2	2
MOV @Ri, dir8	2	3	6	8	4	4
MOV @Ri, #data	2	3	6	8	2	2
MOV DPTR, #data16	3	3	4	4	1	1

Table 55: Summary of Move Instructions (2/3)

[11] A shaded cell refers to an 80C51 instruction.

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
MOV Rmd, Rms	3	2	4	2	1	1
MOV WRjd, WRjs	3	2	4	2	1	1
MOV DRkd, DRks	3	2	6	4	2	2
MOV Rm, #data	4	3	6	4	1	1
MOV WRj, #data16	5	4	6	4	1	1
MOV DRk, #0data16	5	4	10	8	2	2
MOV DRk, #1data16	5	4	10	8	2	2
MOV Rm, dir8	4	3	6	4	3	3
MOV WRj, dir8	4	3	8	6	3	3
MOV DRk, dir8	4	3	12	10	5	5
MOV Rm, dir16	5	4	6	4	3	3
MOV WRj, dir16	5	4	8	6	3	3
MOV DRk, dir16	5	4	12	10	5	5
MOV Rm, @WRj	4	3	6	4	3	3
MOV Rm, @DRk	4	3	8	6	3	3
MOV WRjd,@WRjs	4	3	8	6	3	3
MOV WRj,@DRk	4	3	10	8	3	3
MOV dir8, Rm	4	3	8	6	2	2
MOV dir8, WRj	4	3	10	8	2	2
MOV dir8, DRk	4	3	14	12	4	4
MOV dir16, Rm	5	4	8	6	2	2
MOV dir16, WRj	5	4	10	8	2	2
MOV dir16, DRk	5	4	14	12	4	4
MOV @WRj, Rm	4	3	8	6	2	2
MOV @DRk, Rm	4	3	10	8	2	2
MOV @WRjd, WRjs	4	3	10	8	2	2
MOV @DRk, WRj	4	3	12	10	2	2
MOV Rm, @WRj+dis16	5	4	12	10	3	3
MOV WRj, @WRj+dis16	5	4	14	12	3	3
MOV Rm, @DRk+dis24	5	4	14	12	3	3
MOV WRj, @DRk+dis24	5	4	16	14	3	3
MOV @WRj+dis16, Rm	5	4	12	10	2	2
MOV @WRj+dis16, WRj	5	4	14	12	2	2
MOV @DRk+dis24, Rm	5	4	14	12	2	2
MOV @DRk+dis24, WRj	5	4	16	14	2	2

Table 56: Summary of Move Instructions (3/3)

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
XCH A, Rn ([12])	1	2	6	8	1	1
XCH A, dir8	2	2	6	6	4	4
XCH A, @Ri	1	2	8	10	3	3
XCHD a, @Ri	1	2	8	10	4	4
PUSH dir8	2	2	4	4	4	4
POP dir8	2	2	6	6	4	4
PUSH #data	4	3	8	6	3	3
PUSH #data16	5	4	10	10	3	3
PUSH Rm	3	2	8	6	3	3
PUSH WRj	3	2	10	8	3	3
PUSH DRk	3	2	18	16	5	5
POP Rm	3	2	6	4	3	3
POP WRj	3	2	10	8	3	3
POP DRk	3	2	18	16	4	4

Table 57: Summary of Exchange, Push and Pop Instructions

5.2.4 Program Branching

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode (*)	Source mode (*)	Binary mode (*)	Source mode (*)
JC rel	2	2	2/8	2/8	2/5	2/5
JNC rel	2	2	2/8	2/8	2/5	2/5
JB bit51, rel	3	3	4/10	4/10	4/7	4/7
JNB bit51, rel	3	3	4/10	4/10	4/7	4/7
JBC bit51, rel	3	3	8/14	8/14	4/7	4/7
JZ rel	2	2	4/10	4/10	2/5	2/5
JNZ rel	2	2	4/10	4/10	2/5	2/5
CJNE A, dir8, rel	3	3	4/10	4/10	5/8	5/8
CJNE A, #data, rel	3	3	4/10	4/10	3/6	3/6
CJNE Rn, #data, rel	3	4	4/10	6/12	3/6	3/6
CJNE @Ri, #data, rel	3	4	6/12	8/14	5/8	5/8
DJNZ Rn, rel	2	3	4/10	6/12	3/6	3/6
DJNZ dir8, rel	3	3	6/12	6/12	5/8	5/8
JE rel	3	2	4/10	2/8	2/5	2/5
JNE rel	3	2	4/10	2/8	2/5	2/5
JG rel	3	2	4/10	2/8	2/5	2/5
JLE rel	3	2	4/10	2/8	2/5	2/5

[12] A shaded cell refers to an 80C51 instruction.

[*] Execution time depends on whether the branch is not taken / taken. The last figures correspond to a program jump.

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode (*)	Source mode (*)	Binary mode (*)	Source mode (*)
JSL rel	3	2	4/10	2/8	2/5	2/5
JSLE rel	3	2	4/10	2/8	2/5	2/5
JSG rel	3	2	4/10	2/8	2/5	2/5
JSGE rel	3	2	4/10	2/8	2/5	2/5
JB bit, rel	5	4	8/14	6/12	4/7	4/7
JNB bit, rel	5	4	8/14	6/12	4/7	4/7
JBC bit, rel	5	4	14/20	12/18	4/7	4/7

Table 58: Summary of Conditional Jump Instructions (continued)

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
AJMP addr11 ([13])	2	2	6	6	4	4
LJMP addr16	3	3	10	10	4	4
SJMP rel	2	2	8	8	4	4
JMP @A+DPTR	1	1	10	10	4	4
NOP	1	1	2	2	1	1
EJMP addr24	5	4	12	10	5	5
EJMP @DRk	3	2	14	12	5	5
LJMP @WRj	3	2	12	10	5	5

Table 59: Summary of Unconditional Jump Instructions

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
ACALL addr11	2	2	18	18	4	4
LCALL addr16	3	3	18	18	4	4
RET	1	1	14	14	6	6
RETI	1	1	14	14	5/7([14])	5/7([15])
ECALL @DRk	3	2	28	26	5	5
ECALL addr24	5	4	28	26	5	5
LCALL @WRj	3	2	20	18	4	4
ERET	3	2	18	16	7	7
TRAP	2	1	24	22	6	6

Table 60: Summary of Call and Return Instructions

[13] A shaded cell refers to an 80C51 instruction.

[14] Execution time depends on the interrupt mode configuration

[15] Execution time depends on the interrupt mode configuration

5.2.5 Boolean Variable Manipulation

Instruction Mnemonic	Bytes		Standard 8xC251 Clock cycles		Flip80251 Clock cycles	
	Binary mode	Source mode	Binary mode	Source mode	Binary mode	Source mode
CLR CY ([16])	1	1	2	2	1	1
CLR bit51	2	2	4	4	4	4
SETB CY	1	1	2	2	1	1
SETB bit51	2	2	4	4	4	4
CPL CY	1	1	2	2	1	1
CPL bit51	2	2	4	4	4	4
ANL CY, bit51	2	2	2	2	3	3
ANL CY, /bit51	2	2	2	2	3	3
ORL CY, bit51	2	2	2	2	3	3
ORL CY, /bit51	2	2	2	2	3	3
MOV CY, bit51	2	2	2	2	3	3
MOV bit51, CY	2	2	4	4	4	4
CLR bit	4	3	8	6	4	4
SETB bit	4	3	8	6	4	4
CPL bit	4	3	8	6	4	4
ANL CY, bit	4	3	6	4	3	3
ANL CY, /bit	4	3	6	4	3	3
ORL CY, bit	4	3	6	4	3	3
ORL CY, /bit	4	3	6	4	3	3
MOV CY, bit	4	3	6	4	3	3
MOV bit, CY	4	3	8	6	4	4

Table 61: Summary of Bit Instructions

[16] A shaded cell refers to an 80C51 instruction.

5.3 Opcode Map

Bin.	0	1	2	3	4	5	6-7	8-F
Src.	**0**	**1**	**2**	**3**	**4**	**5**	**A5x6-A5x7**	**A5x8-A5xF**
0	NOP	AJMP addr11	LJMP addr16	RR A	INC A	INC dir8	INC @Ri	INC Rn
1	JBC bit51, rel	ACALL addr11	LCALL addr16	RRC A	DEC A	DEC dir8	DEC @Ri	DEC Rn
2	JB bit51, rel	AJMP addr11	RET	RL A	ADD A, #data	ADD A, dir8	ADD A, @Ri	ADD A, Rn
3	JNB bit51, rel	ACALL addr11	RETI	RLC A	ADDC A, #data	ADDC A, dir8	ADDC A, @Ri	ADDC A, Rn
4	JC rel	AJMP addr11	ORL dir8, A	ORL dir8, #data	ORL A, #data	ORL A, dir8	ORL A, @Ri	ORL A, Rn
5	JNC rel	ACALL addr11	ANL dir8, A	ANL dir8, #data	ANL A, #data	ANL A, dir8	ANL A, @Ri	ANL A, Rn
6	JZ rel	AJMP addr11	XRL dir8, A	XRL dir8, #data	XRL A, #data	XRL A, dir8	XRL A, @Ri	XRL A, Rn
7	JNZ rel	ACALL addr11	ORL CY, bit51	JMP @A+DPTR	MOV A, #data	MOV dir8, #data	MOV @Ri, #data	MOV Rn, #data
8	SJMP rel	AJMP addr11	ANL CY, bit51	MOVC A, @A+PC	DIV AB	MOV dir8, dir8	MOV dir8, @Ri	MOV dir8, Rn
9	MOV DPTR, #data16	ACALL addr11	MOV bit51, CY	MOVC A, @A+DPTR	SUBB A, #data	SUBB A, dir8	SUBB A, @Ri	SUBB A, Rn
A	ORL CY, /bit51	AJMP addr11	MOV CY, bit51	INC DPTR	MUL AB	ESC	MOV @Ri, dir8	MOV Rn, dir8
B	ANL CY, /bit51	ACALL addr11	CPL bit51	CPL CY	CJNE A,#data,rel	CJNE A,dir8,rel	CJNE @Ri,#data, rel	CJNE Rn,#data, rel
C	PUSH dir8	AJMP addr11	CLR bit51	CLR CY	SWAP A	XCH A, dir8	XCH A, @Ri	XCH A, Rn
D	POP dir8	ACALL addr11	SETB bit51	SETB CY	DA A	DJNZ dir8, rel	XCHD A, @Ri	DJNZ Rn, rel
E	MOVX A, @DPTR	AJMP addr11	MOVX A, @Ri		CLR A	MOV A, dir8	MOV A, @Ri	MOV A, Rn
F	MOVX @DPTR, A	ACALL addr11	MOVX @Ri, A		CPL A	MOV dir8, A	MOV @Ri, A	MOV Rn, A

Table 62: Instructions for 80C51 micro-controllers

Bin.	A5x8	A5x9	A5xA	A5xB	A5xC	A5xD	A5xE	A5xF
Src.	x8	x9	xA	xB	xC	xD	xE	xF
0	JSLE rel	MOV Rm, @WRj+dis	MOVZ WRj, Rm	INC R,#short (1) MOV reg,ind			SRA reg	
1	JSG rel	MOV @WRj+dis,Rm	MOVS WRj, Rm	DEC R,#short (1) MOV ind,reg			SRL reg	
2	JLE rel	MOV Rm,@DRk+dis			ADD Rm, Rm	ADD WRj, WRj	ADD reg, op2 (2)	ADD DRk, DRk
3	JG rel	MOV @DRk+dis, Rm					SLL reg	
4	JSL rel	MOV WRj,@WRj+dis			ORL Rm, Rm	ORL WRj, WRj	ORL reg, op2 (2)	
5	JSGE rel	MOV @WRj+dis, WRj			ANL Rm, Rm	ANL WRj, WRj	ANL reg, op2 (2)	
6	JE rel	MOV WRj,@DRk+dis			XRL Rm, Rm	XRL WRj, WRj	XRL reg, op2 (2)	
7	JNE rel	MOV @DRk+dis, WRj	MOV op1,reg (2)		MOV Rm, Rm	MOV WRj, WRj	MOV reg, op2 (2)	MOV DRk, DRk
8		LJMP @WRj EJMP @DRk	EJMP addr24		DIV Rm, Rm	DIV WRj, WRj		
9		LCALL @WRj ECALL @DRk	ECALL addr24		SUB Rm, Rm	SUB WRj, WRj	SUB reg, op2 (2)	SUB DRk, DRk
A		Bit Instructions (3)	ERET		MUL Rm, Rm	MUL WRj, WRj		
B		TRAP			CMP Rm, Rm	CMP WRj, WRj	CMP reg, op2 (2)	CMP DRk, DRk
C			PUSH op1					
D			POP op1 (4)					
E								
F								

Table 63: Instructions to 8xC251 architecture

1. R=Rm/WRj/DRk
2. op1, op2 are described in Table 64
3. Refer to Table 66 and Table 67.
4. Refer to Table 68

5.4 Instructions encoding

Instructions	Byte 0		Byte 1		Byte 2		Byte 3
Oper Rmd, Rms	x	C	md	ms			
Oper WRjd, WRjs	x	D	jd/2	js/2			
Oper DRkd, DRks	x	F	kd/4	ks/4			
Oper Rm, #data	x	E	m	0	#data		
Oper WRj, #data16	x	E	j/2	4	#data (high)		#data (low)
Oper DRk, #0data16	x	E	k/4	8	#data (high)		#data (low)
MOVH DRk, #data16 MOV DRk, #1data16 CMP DRk, #1data16	7 7 B	A E E	k/4	C	#data (high)		#data (low)
Oper Rm, dir8	x	E	m	1	dir8 addr		
Oper WRj, dir8	x	E	j/2	5	dir8 addr		
Oper DRk, dir8	x	E	k/4	D	dir8 addr		
Oper Rm, dir16	x	E	m	3	dir16 addr (high)		dir16 addr (low)
Oper WRj, dir16	x	E	j/2	7	dir16 addr (high)		dir16 addr (low)
Oper DRk, dir16 (1)	x	E	k/4	F	dir16 addr (high)		dir16 addr (low)
Oper Rm, @WRj	x	E	j/2	9	m	0	
Oper Rm, @DRk	x	E	k/4	B	m	0	

Table 64: Encoding for Data instructions

(1) These operands are valid only for MOV instruction.

x	Operation	Notes
2	ADD reg, op2	All data addressing modes are supported (1)
9	SUB reg, op2	
B	CMP reg, op2	
4	ORL reg, op2 (2)	
5	ANL reg, op2 (2)	
6	XRL reg, op2 (2)	
7	MOV reg, op2	
8	DIV reg, op2	Two modes only: reg,op2 =Rmd, Rms or WRjd, WRjs
A	MUL reg, op2	

Table 65: Encoding for High nibble byte 0 of Data instructions

(1) DRk, dir16 is valid for MOV instruction only.
(2) For ORL, ANL and XRL, neither reg nor op2 can be DRk

All of the bit instructions in the 8xC251 architecture have opcode A9, which serve as an escape byte (similar to A5). The high nibble of byte 1 specifies the bit instruction.

	Instructions	Byte 0 (x)		Byte 1			Byte 2	Byte 3 (*)
1	Bit instr (dir8)	A	9	xxxx	0	bit	dir8 addr	rel addr

Table 66: Encoding for Bit instructions

(*) only for jump bit instructions (JBC, JB, JNB)

xxxx	Bit Instruction
1	JBC bit
2	JB bit
3	JNB bit
7	ORL CY, bit
8	ANL CY, bit
9	MOV bit, CY
A	MOV CY, bit
B	CPL bit
C	CLR bit
D	SETB bit
E	ORL CY, /bit
F	ANL CY, /bit

Table 67: Encoding for high nibble byte1 of Bit instructions

Instruction	Byte 0(x)		Byte1		Byte 2	Byte 3
PUSH #data	C	A	0	2	#data	
PUSH #data16	C	A	0	6	#data16 (high)	#data16 (low)
PUSH Rm	C	A	m	8		
PUSH WRj	C	A	j/2	9		
PUSH DRk	C	A	k/4	B		
POP Rm	D	A	m	8		
POP WRj	D	A	j/2	9		
POP DRk	D	A	k/4	B		

Table 68: Encoding for PUSH/POP instructions

Instruction	Byte0		Byte1		Byte2	Byte3
EJMP addr24	8	A	addr[23:16]		addr[15:8]	addr[7:0]
ECALL addr24	9	A	addr[23:16]		addr[15:8]	addr[7:0]
LJMP @WRj	8	9	j/2	4		
LCALL @WRj	9	9	j/2	4		
EJMP @DRk	8	9	k/4	8		
ECALL @DRk	9	9	k/4	8		
ERET	A	A				
JE rel	6	8	rel			
JNE rel	7	8	rel			
JLE rel	2	8	rel			
JG rel	3	8	rel			
JSL rel	4	8	rel			
JSGE rel	5	8	rel			
JSLE rel	0	8	rel			
JSG rel	1	8	rel			
TRAP	B	9				

Table 69: Encoding for Control instructions

Instruction	Byte 0		Byte 1		Byte 2		Byte 3
MOV Rm, @WRj+dis16	0	9	m	j/2	dis[15:8]		dis[7:0]
MOV WRjd, @WRjs+dis16	4	9	jd/2	js/2	dis[15:8]		dis[7:0]
MOV Rm, @DRk+dis24	2	9	m	k/4	dis[15:8]		dis[7:0]
MOV WRj, @DRk+dis24	6	9	j/2	k/4	dis[15:8]		dis[7:0]
MOV @WRj+dis16, Rm	1	9	m	j/2	dis[15:8]		dis[7:0]
MOV @WRjd+dis16, WRjs	5	9	js/2	jd/2	dis[15:8]		dis[7:0]
MOV @DRk+dis24, Rm	3	9	m	k/4	dis[15:8]		dis[7:0]
MOV @DRk+dis24, WRj	7	9	j/2	k/4	dis[15:8]		dis[7:0]
MOVS WRj, Rm	1	A	j/2	m			
MOVZ WRj, Rm	0	A	j/2	m			
MOV WRjd, @WRjs	0	B	js/2	8	jd/2	0	
MOV WRj, @DRk	0	B	k/4	A	j/2	0	
MOV @WRjd, WRjs	1	B	jd/2	8	js/2	0	
MOV @DRk, WRj	1	B	k/4	A	j/2	0	
MOV dir8, Rm	7	A	m	1	dir8 addr		
MOV dir8, WRj	7	A	j/2	5	dir8 addr		
MOV dir8, DRk	7	A	k/4	D	dir8 addr		
MOV dir16, Rm	7	A	m	3	dir16 addr (high)		dir16 addr (low)
MOV dir16, WRj	7	A	j/2	7	dir16 addr (high)		dir16 addr (low)
MOV dir16, DRk	7	A	k/4	F	dir16 addr (high)		dir16 addr (low)
MOV @WRj, Rm	7	A	j/2	9	m	0	
MOV @DRk, Rm	7	A	k/4	B	m	0	

Table 70: Encoding for Displacement/Extended MOVE instructions

	Instructions	Byte 0 (x)		Byte 1		
1	INC Rm, #short	0	B	m	00	vv
2	INC WRj, #short	0	B	j/2	01	vv
3	INC DRk, #short	0	B	k/4	11	vv
4	DEC Rm, #short	1	B	m	00	vv
5	DEC WRj, #short	1	B	j/2	01	vv
6	DEC DRk, #short	1	B	k/4	11	vv

Table 71: Encoding for Increment/Decrement instructions

vv	#short
00	1
01	2
10	4

Table 72: Encoding for byte1 of Increment/Decrement instructions

	Instruction	Byte 0		Byte 1	
1	SRA Rm	0	E	m	0
2	SRA WRj	0	E	j/2	4
3	SRL Rm	1	E	m	0
4	SRL WRj	1	E	j/2	4
5	SLL Rm	3	E	m	0
6	SLL WRj	3	E	j/2	4

Table 73: Encoding for Shift instructions

5.5 Performance comparison

Binary mode		Source mode	
Number of opcodes	**Speed improvement**	**Number of instructions**	**Speed improvement**
10	6.00	2	8.00
1	5.00	1	7.00
9	4.50	8	6.00
5	4.00	2	5.20
5	3.00	5	5.00
4	2.67	1	4.67
2	2.50	1	4.50
2	2.33	28	4.00
126	2.00	1	3.67
17	1.56	2	3.33
22	1.50	1	3.20
2	1.43	6	3.00
3	1.38	1	2.75
14	1.33	1	2.73
2	1.27	7	2.67
1	1.08	1	2.40
10	1.00	1	2.29
20	0.67	1	2.11
		52	2.00
		2	1.64
		4	1.50
		8	1.43
		21	1.33
Total	**Weighted average**	**Total**	**Weighted average**
255	**2.1 X**	**157**	**2.8 X**

Table 74: Instruction Set Speed comparison between standard C251 and Flip80251-Hurricane

The overall speed improvement is approximately **2.8 X** using instructions in source mode. Anyway, actual speed improvement depends on application program.

6. FLIP80251 INNER PERIPHERALS

Flip80251 peripherals are compatible with the original 8xC251. As in the original 8xC251 the number of pin is limited, some signals need to be multiplexed, especially the peripheral signals. In order to provide more flexibility to the designer, the peripherals of the Flip80251 are not constrained by external pin functionality. So, all peripherals can be brought out separately. This makes connection easier, and enables all ports to be used at the same time as the alternate peripheral functions.

Original 8xC251 pin	Flip80251 signal
P1.0	timer2
P1.1	timer2capt
P1.1	timer2clkout
P1.2	pcaeci
P1.3	pcacomp0
P1.3	pcacapt0
P1.4	pcacomp1
P1.4	pcacapt1
P1.5	pcacomp2
P1.5	pcacapt2
P1.6	pcacomp3
P1.6	pcacapt3
P1.7	pcacomp4
P1.7	pcacapt4
P3.0	serialin
P3.1	serialout
P3.1	serialclk
P3.2	int0_n
P3.3	int1_n
P3.4	timer0
P3.2	timer0gate
P3.5	timer1
P3.3	timer1gate

Table 75: Relation between the Flip80251 signals and the original 8xC251 pins.

6.1 I/O Ports

The I/O ports consist in four SFR registers, one for each port, plus four others SFR registers, for each port direction, and the I/O port interface. The SFR register for each port is an 8-bit register, which can be addressed at the SFR location for that port.

I/O Ports	Assembly reference	SFR address
Port0	P0	S:080h
Port1	P1	S:090h
Port2	P2	S:0A0h
Port3	P3	S:0B0h

Table 76: I/O Ports registers

✋*: The port registers should not be confused with the port pins; the data on the registers may be not the same as on the pins!*

The I/O ports interface corresponds to three different kinds of signals:

- Port output signals.
- Port input signals.
- Port direction indication signals.

The port output signals represent the contents of the port registers. Each port has a corresponding SFR register and any write into theses SFR registers is directly forwarded to its port output signal: *port0out*, *port1out*, *port2out* and *port3out*.

✋*: The port input signals port0in, port1in, port2in and port3in are connected directly to the Flip80251. They are not sampled. The signals can be sampled externally, if they are changing whilst being read by the application program. These port input signals are used by all instructions that perform only a SFR read at the SFR location for that port.*

6.1.1 Read-Modify-Write feature

Some instructions that read a port read the SFR and others read the pin.
The instructions that read the SFR rather than the pin are the ones that read a value, possibly change it, and then rewrite it to the register. These instructions are called '**read-modify-write**' instructions.

Instructions	Description	Example with I/O ports
ANL	Logical AND	ANL P0, A
ORL	Logical OR	ORL P0, A
XRL	Logical eXclusive OR	XRL P0, A
JBC	Jump if bit and clear it	JBC P1.0, rel
CPL	Complement bit	CPL P1.1
INC	Increment	INC P2
DEC	Decrement	DEC P2
DJNZ	Decrement and jump if not zero	DJNZ P2, rel
MOV bit, CY	Move carry bit to a bit	MOV P3.7, CY
CLR bit	Clear a bit	CLR P3.6
SETB bit	Set a bit	SET P3.5

Table 77: Read-modify-write instructions

✋*: The last three instructions in the previous table are read-modify-write instructions, because it is not possible to read only one bit! So, these instructions read the byte, modify the addressed bit, and then write the new byte to the register.*

The instructions that read the port pin rather than the SFR are all instructions that perform only a SFR read at the SFR location for that port (e.g. MOV A, P0)

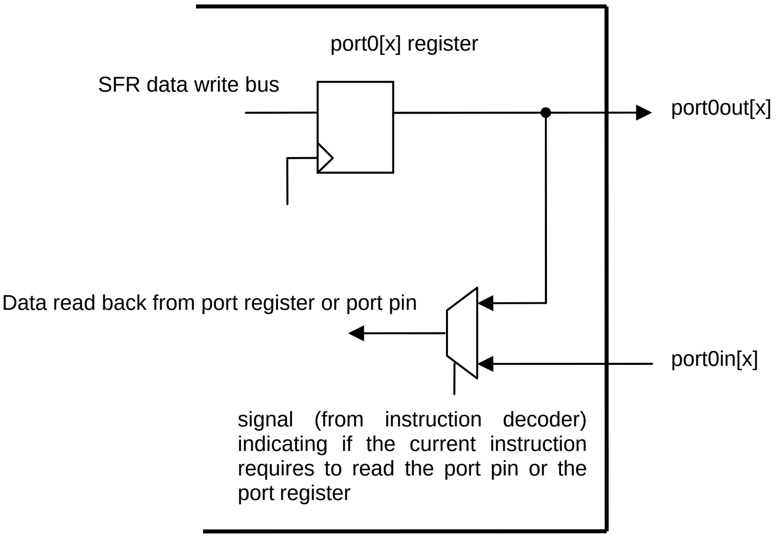

port0[x] register
SFR data write bus
port0out[x]
Data read back from port register or port pin
port0in[x]
signal (from instruction decoder) indicating if the current instruction requires to read the port pin or the port register

Figure 14: I/O port structure

6.1.2 Port direction signals

The port direction signals can be accessed at SFR location. Just like the output port signals, each port direction has a corresponding SFR register and any write into theses SFR registers is directly forwarded to its port direction signal: *port0dir*, *port1dir*, *port2dir* and *port3dir*.

I/O Ports direction	Assembly reference	SFR Address
Port0 direction	P0_DIR	S:0ACh
Port1 direction	P1_DIR	S:0ADh
Port2 direction	P2_DIR	S:0AEh
Port3 direction	P3_DIR	S:0AFh

Table 78: Port direction registers

Inside a chip, it is usual for a port to be used only as an input, or an output, to avoid using internal tri-state buffers. In this case, only the relevant input or output signal needs to be connected. If the port needs to be bi-directional for connecting to an external device, an additional direction control signal maybe needed. In the Flip80251, a feature called Port Direction signals is available. If this feature were used, the direction signal would be connected to the direction control input of the pad. When the direction signal is high the port is an input, when the direction pin is low the port is an output.

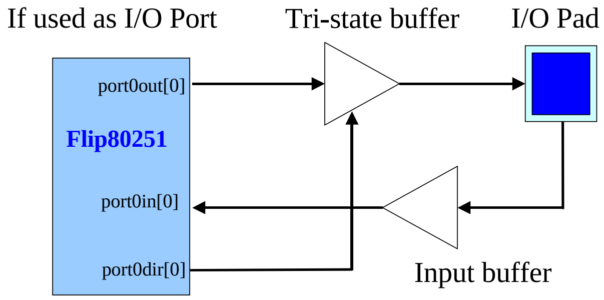


Figure 15: Example of implementation of a bi-directional port

✋: *In general we recommend that a ‘0’ on Port Direction signal indicates the port is configured as an output and a ‘1’ indicates the port is configured as an input. In this way the I/O pins are configured as input after reset and hence avoid an input pin from damaging an external circuit after power up. However, it is up to individual applications to determine whether a ‘1’ represents input or output direction.*

Writing a ‘1’ to the port direction SFR bit, sets that port bit to be an input, writing a ‘0’ causes the port bit to be an output. On reset these SFRs are set to FFh, which sets all ports to be inputs.

6.1.3 I/O ports registers

Mnemonic	Address	Description	Reset value
P0	S:080h	Port0	FFh
P0_DIR	S:0ACh	Port0 direction	FFh
P1	S:090h	Port1	FFh
P1_DIR	S:0ADh	Port1 direction	FFh
P2	S:0A0h	Port2	FFh
P2_DIR	S:0AEh	Port2 direction	FFh
P3	S:0B0h	Port3	FFh
P3_DIR	S:0AFh	Port3 direction	FFh

Table 79: I/O Ports registers

P0_DIR (S:ACh)
P1_DIR (S:ADh)
P2_DIR (S:AEh)
P3_DIR (S:AFh)
Port x Direction Register (x=0, 1, 2, 3)

BIT	7	6	5	4	3	2	1	0
FIELD	PxDIR7	PxDIR6	PxDIR5	PxDIR4	PxDIR3	PxDIR2	PxDIR1	PxDIR0
RESET	1111 1111b							

Bit Number	Bit Mnemonic	Function
7:0	PxDIR7	**Direction of Port x bit 7** Set to configure Portx.7 as an input. Clear to configure Portx.7 as an output
	PxDIR6	**Direction of Port x bit 6** Set to configure Portx.6 as an input. Clear to configure Portx.6 as an output
	PxDIR5	**Direction of Port x bit 5** Set to configure Portx.5 as an input. Clear to configure Portx.5 as an output
	PxDIR4	**Direction of Port x bit 4** Set to configure Portx.4 as an input. Clear to configure Portx.4 as an output
	PxDIR3	**Direction of Port x bit 3** Set to configure Portx.3 as an input. Clear to configure Portx.3 as an output
	PxDIR2	**Direction of Port x bit 2** Set to configure Portx.2 as an input. Clear to configure Portx.2 as an output
	PxDIR1	**Direction of Port x bit 1** Set to configure Portx.1 as an input. Clear to configure Portx.1 as an output
	PxDIR0	**Direction of Port x bit 0** Set to configure Portx.0 as an input. Clear to configure Portx.0 as an output

Figure 16: Port direction registers (Px_DIR)

6.2 Timer/Counters and watchdog timer

Depending on configuration, the Flip80251-Hurricane has either two or three general-purpose 16-bit timer/counters. The timer/counters support all operation modes of the standard 8xC251. Each timer/counter has dedicated input signals. Additionally a hardware watchdog timer is available.

6.2.1 Timer/counter overview

The following is a brief description of the signals and which port pins they would use in the original 80C51.

- Counter/Timer0 uses the signals *timer0* and *timer0gate*. The *timer0* signal is used as a counter input and *timer0gate* as an active-high gate of this counter. They are the same as the alternate functions of pins P3.4 and P3.2 (*int0_n*) respectively in the original 8xC251.
- Similarly, Counter/Timer1 uses the signals *timer1* and *timer1gate*, which are the same as the alternate functions of pins P3.5 and P3.3 (*int1_n*) respectively in the original 8xC251.
- Counter/Timer 2 uses the signals *timer2*, *timer2capt*, *timer2clkout* and *timer2dir*. The *timer2* signal is used as a counter input and *timer2capt* is used to trigger either a timer capture or reload. *timer2clkout* is the clock output in mode 3 (programmable clock out). *timer2dir* can be used to multiplex *timer2capt* and *timer2clkout*. The *timer2* and *timer2capt/timer2clkout* pins are the same as the alternate functions of pins P1.0 and P1.1 respectively in the original 8xC251.

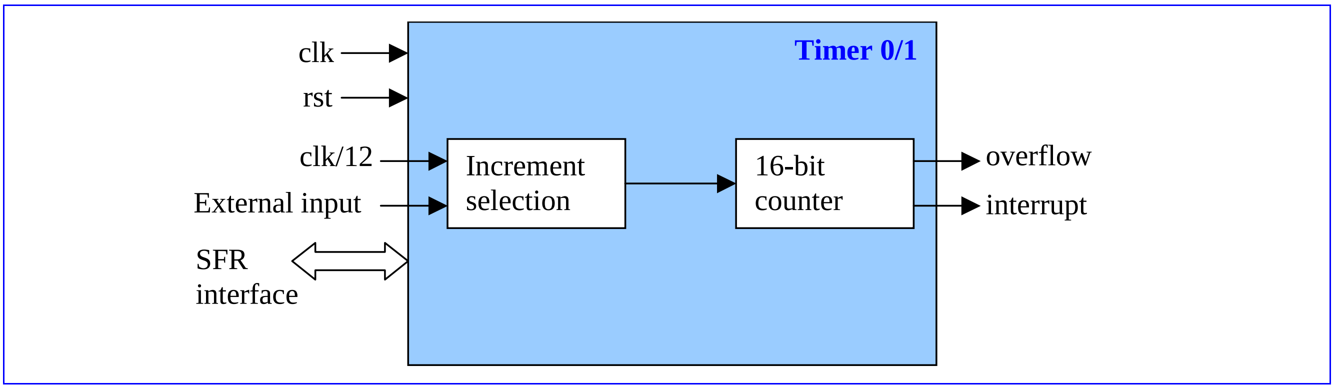


Figure 17: Timer/Counter 0&1 block diagram

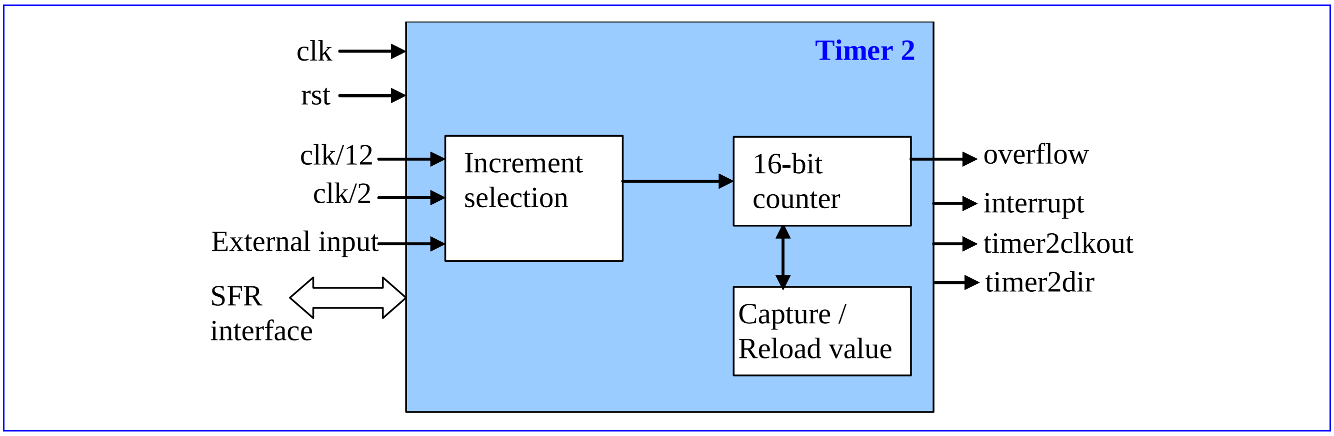


Figure 18: Timer/Counter 2 block diagram

6.2.1.1 *Timer or Counter Mode*

The selection of timer/counter mode is controlled by the C/T bit in the TMOD (Timer/counter MODE control register) for timer0/timer1 and T2MOD for timer2.

- Timer Mode

Although the 8xC251 has a machine cycle which is shorter than 12 clock cycles, in order to obtain compatibility with standard 80C51 and 8xC251 timers, the register [THx:TLx] is increased every 12 peripheral clock cycles when enabled.

- Counter Mode

In counter mode operation, the register [THx:TLx] is incremented in response to falling edge of its corresponding input. In order to obtain compatibility with standard 80C51, the input is sampled every 12 peripheral clock cycle.

6.2.1.2 *T/C 0 and T/C 1 modes overview*

The used of timer0 and timer1 are almost the same (except in mode 3). Both timers have four operation modes:

Mode	Operation
Mode 0	13-bit timer/counter
Mode 1	16-bit timer/counter
Mode 2	Auto reload 8-bit timer/counter
Mode 3	Timer0: two 8-bit timer/counters Timer0: stopped

6.2.1.3 *T/C 2 modes overview*

Mode	Operation
Mode 0	Capture
Mode 1	Auto reload
Mode 2	Baud rate generator
Mode 3	Programmable clock-out

6.2.2 T/C 0 and T/C 1 operation modes

6.2.2.1 Mode 0: 13-bit timer/counter

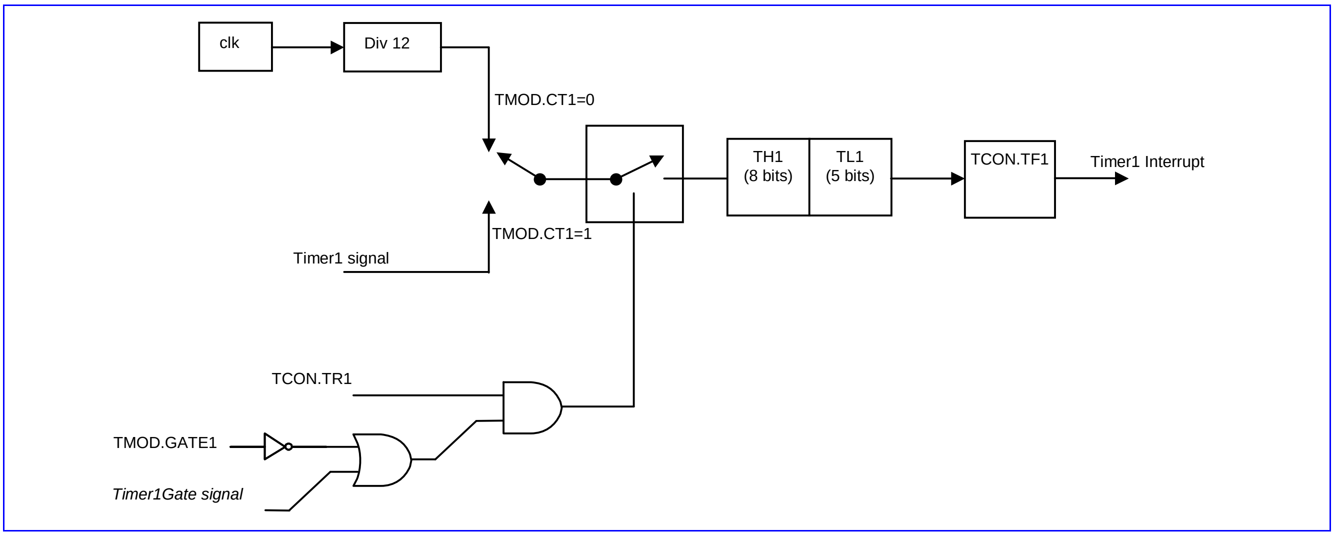


Figure 19: Timer/Counter 1 mode 0: 13-bit counter

6.2.2.2 Mode 1: 16-bit timer/counter

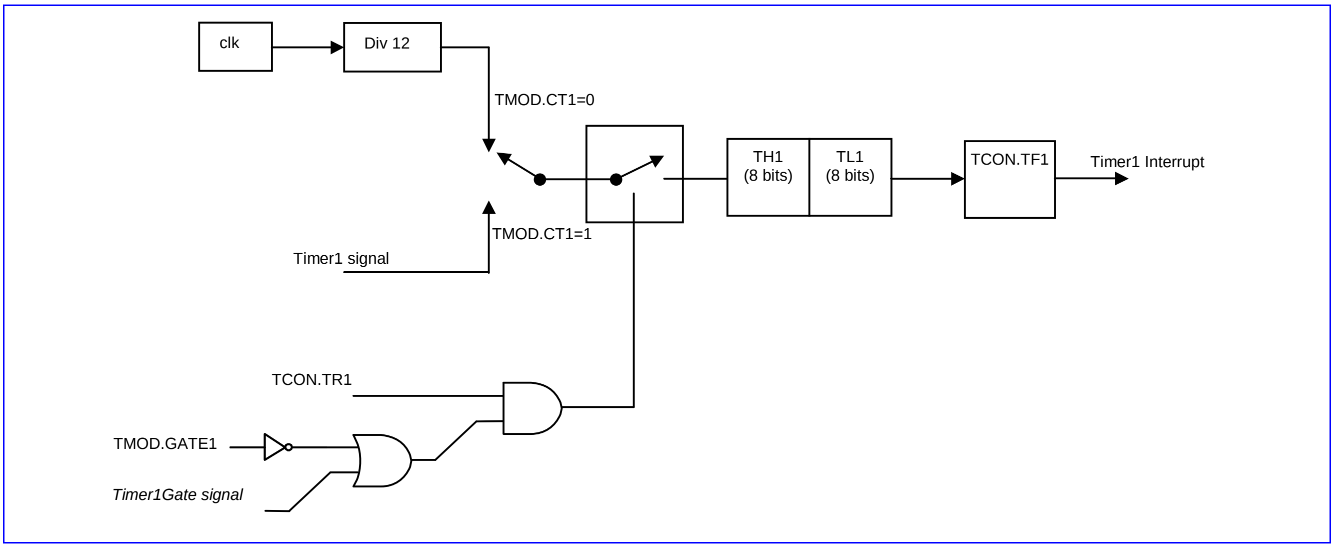


Figure 20: Timer/Counter 1 Mode 1: 16-bit counter

6.2.2.3 Mode 2: Auto reload 8-bit timer/counter

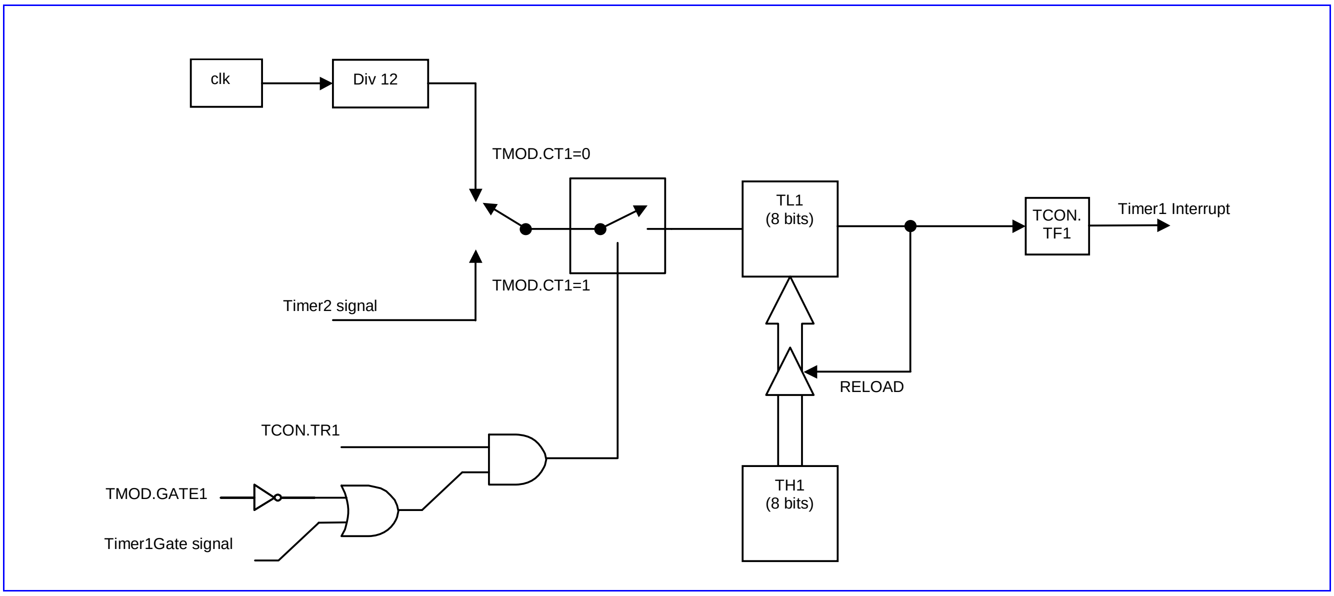


Figure 21: Timer/Counter 1 Mode 2: 8-bit auto-reload counter

6.2.2.4 Mode 3: T/C 0: two 8-bit timer/counters. T/C 1: stopped

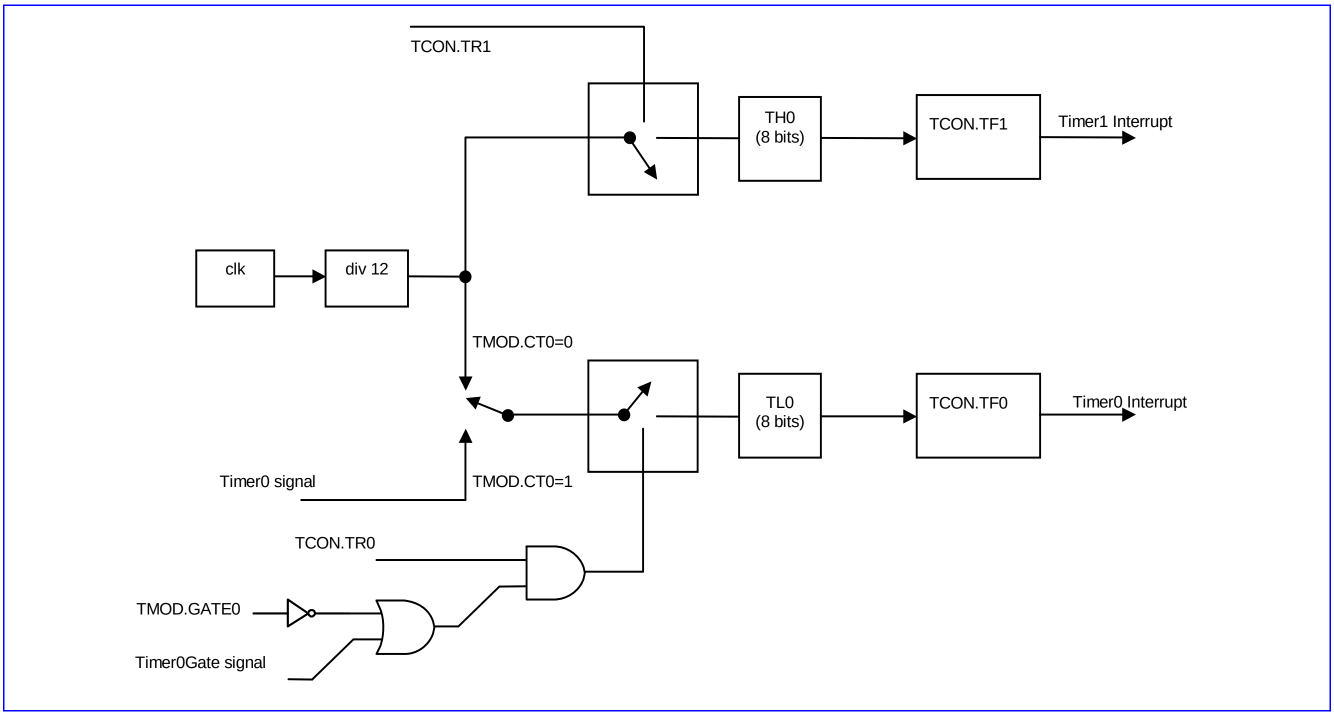


Figure 22: Timer/Counter 0 Mode 3: Two 8-bit counters

6.2.3 T/C 2 operation modes

6.2.3.1 Mode 0: capture

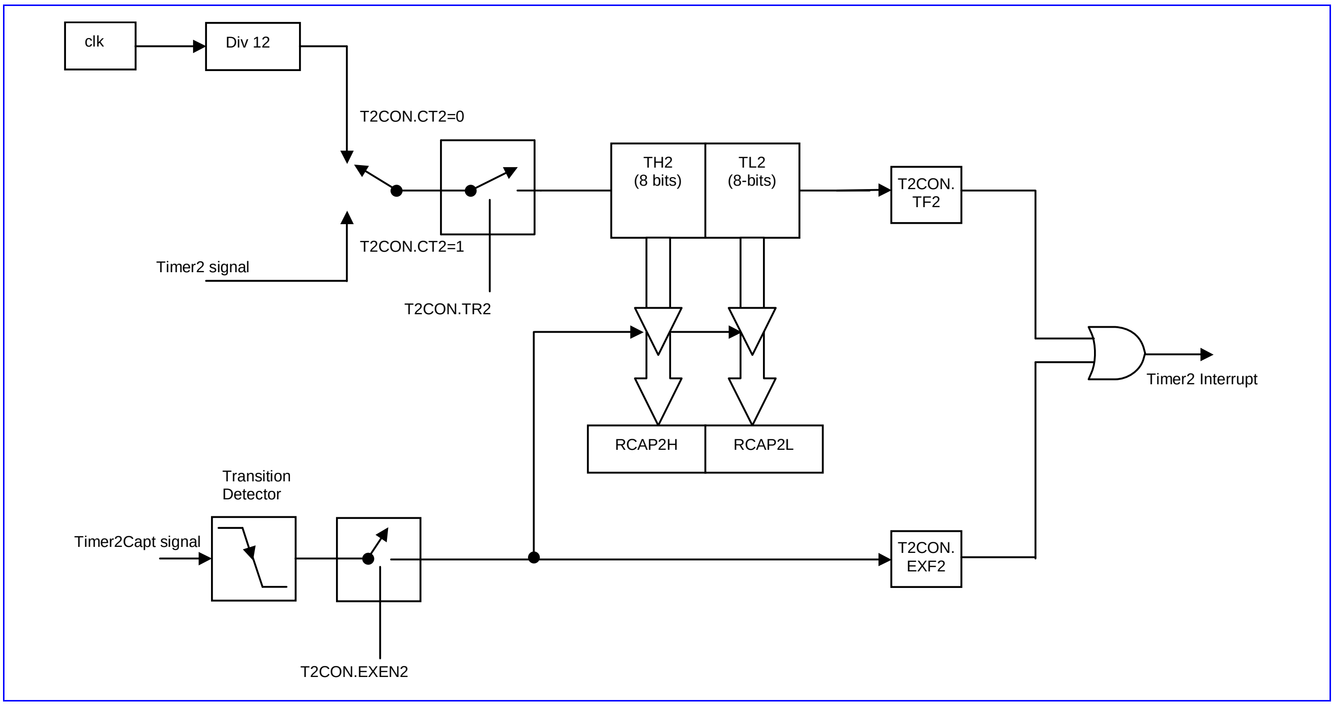


Figure 23: Timer/Counter 2 in Capture Mode

6.2.3.2 Mode 1: auto reload

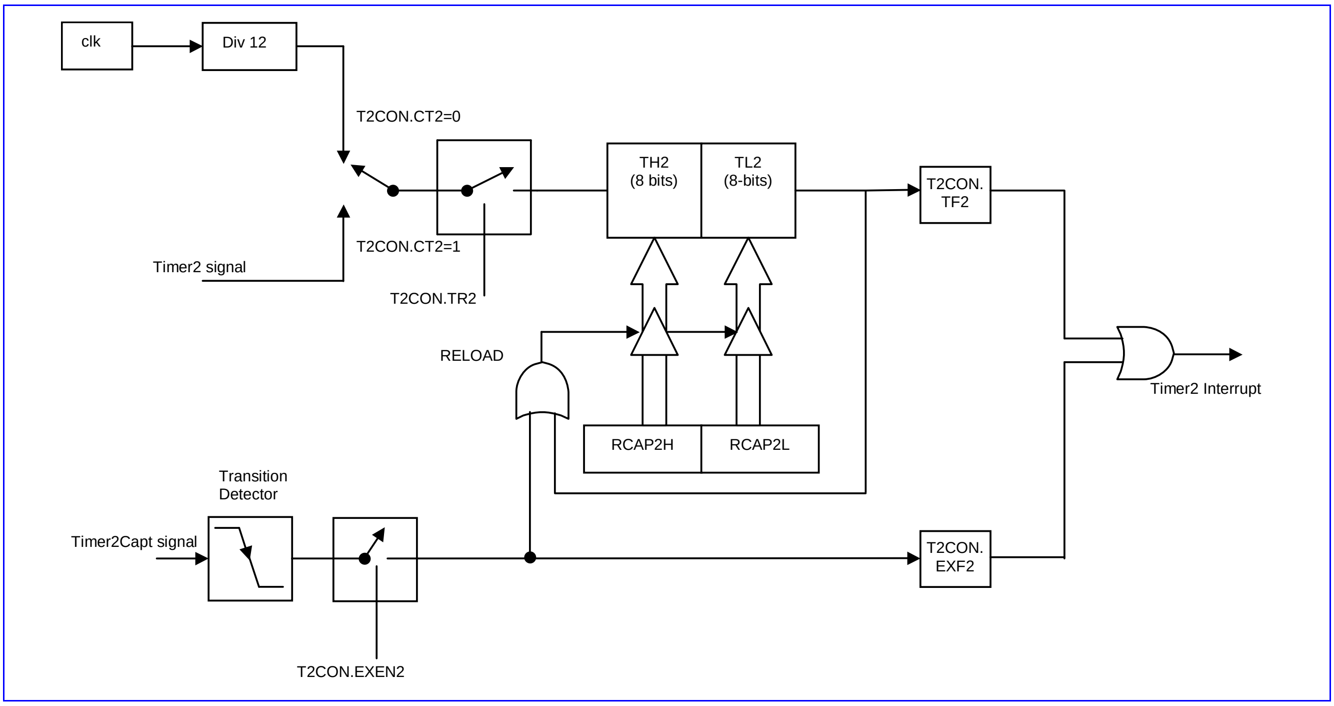


Figure 24: Timer/Counter 2 in Auto-Reload Mode (DCEN = 0)

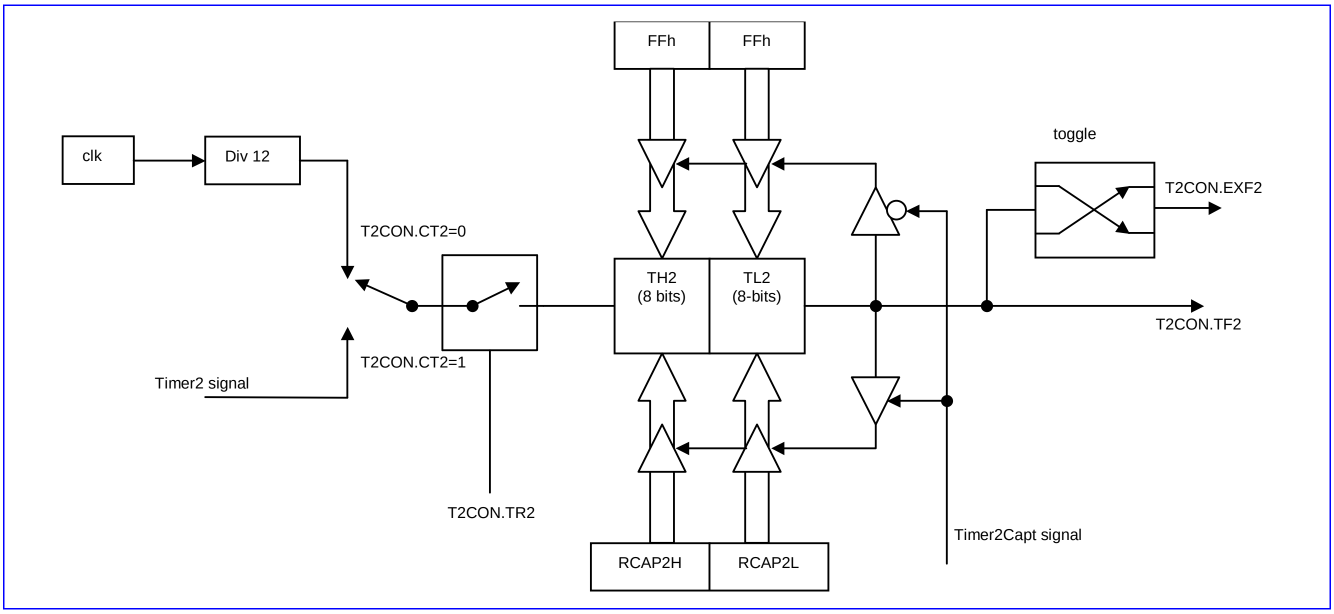


Figure 25: Timer/Counter 2 in Auto-Reload Mode (DCEN = 1)

6.2.3.3 Mode 2: baud rate generator

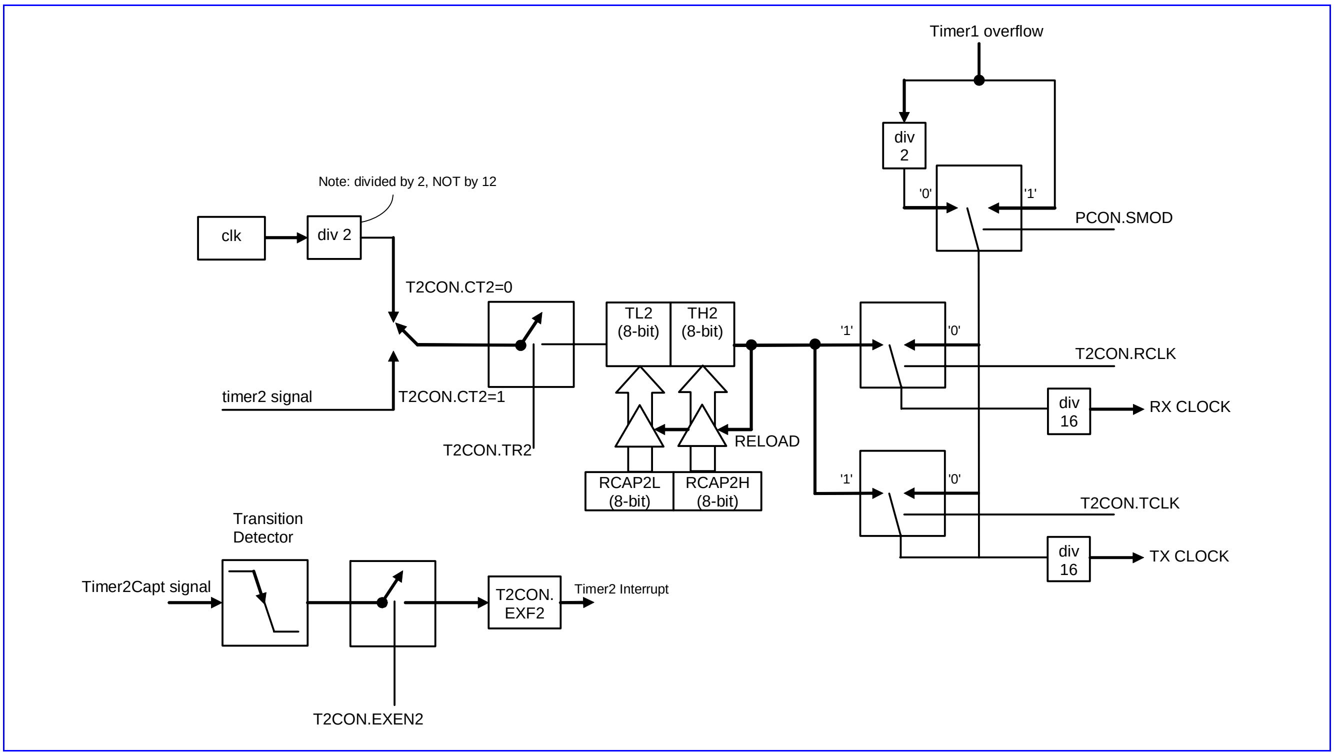


Figure 26: Timer/Counter 2 in Baud Rate Generator Mode

6.2.3.4 Mode 3: programmable clock-out

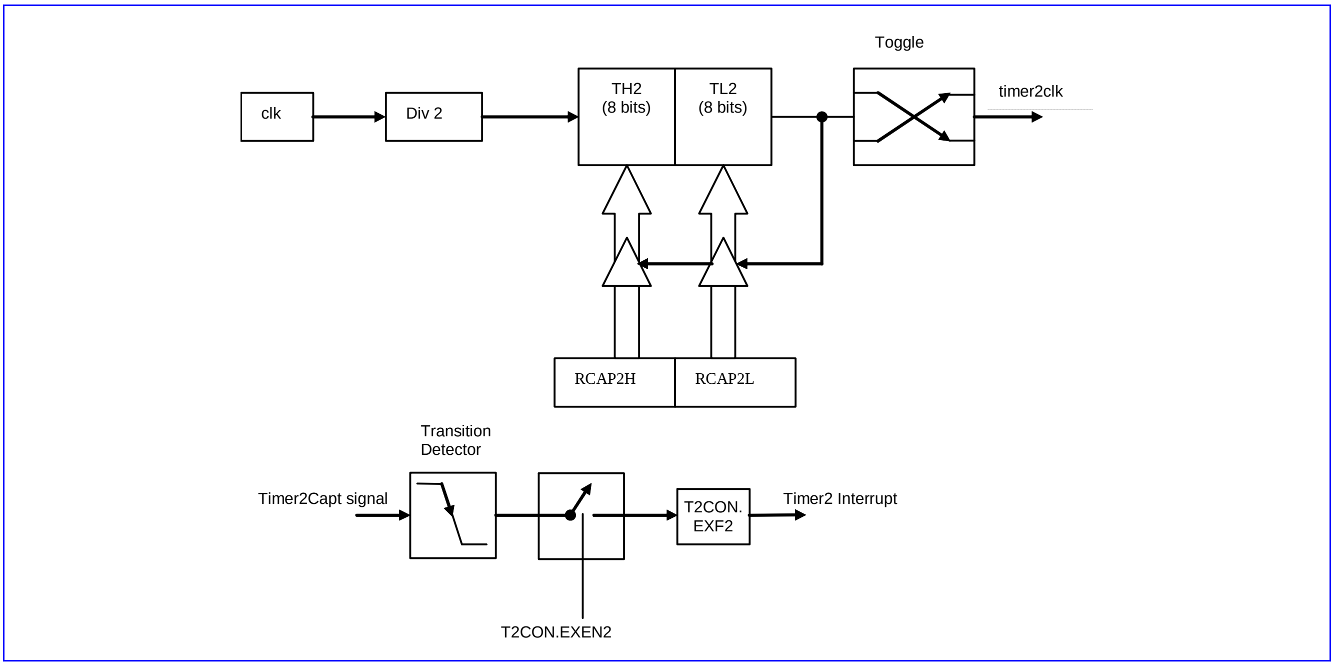


Figure 27: Timer/Counter 2 in Clock-out Mode

6.2.4 Watchdog Timer (WDT)

6.2.4.1 Overview

The hardware watchdog timer (WDT) automatically resets the chip if it is allowed to time out. The WDT provides a means of recovering from routines that do not complete successfully due to software malfunctions.
Note: in order to offer a higher flexibility than the standard 8xC251 Watchdog Timer, the WDT includes a control register (WDTCON) that enables to configure the time-out period. This register was not part of the standard.

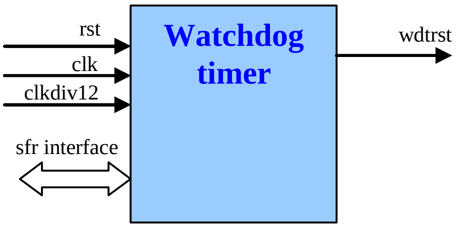


Figure 28: WDT logic symbol

6.2.4.2 Description

The watchdog timer (WDT) is a 20-bit counter that is incremented every *clkdiv12* cycle.
The WDTRST register (S:0A6h) is used to enable/reset the WDT.
The WDTCON register (S:0A5h) enables to select the bit of the WDT counter that will be used as the counter overflow indicator, i.e. it enables to configure the time-out period.
Three operations control the WDT:

- Device reset by the external RESET signal clears and disables the WDT.
- Writing into the WDTCON register to configure the time-out period.
- Writing a specific two-byte sequence to the WDTRST register clears and enables the WDT.

Important Note: Once the WDT is enabled, it is not possible to write into the WDTCON register.

The WDTRST is a write-only register; attempts to read it will return 0FFh. The watchdog timer itself is not readable or writable but the 5 Most Significant bits of the WDTCON register (implemented as read-only bits) provide information about the internal status of the WDT.

WDT overflow configuration bits			WDT counter size	Time-out period [17] (clock cycles)	Time-out period (@ 12 MHz)
WDTOV2	WDTOV1	WDTOV0			
0	0	0	13-bit	8 K	8.2 ms
0	0	1	14-bit	16 K	16.4 ms
0	1	0	15-bit	32 K	32.8 ms
0	1	1	16-bit	64 K	65.5 ms
1	0	0	17-bit	128 K	131.1 ms
1	0	1	18-bit	256 K	262.1 ms
1	1	0	19-bit	512 K	524.3 ms
1	1	1	20-bit	1024 K	1048.6 ms

Table 80: WDT time-out period selection

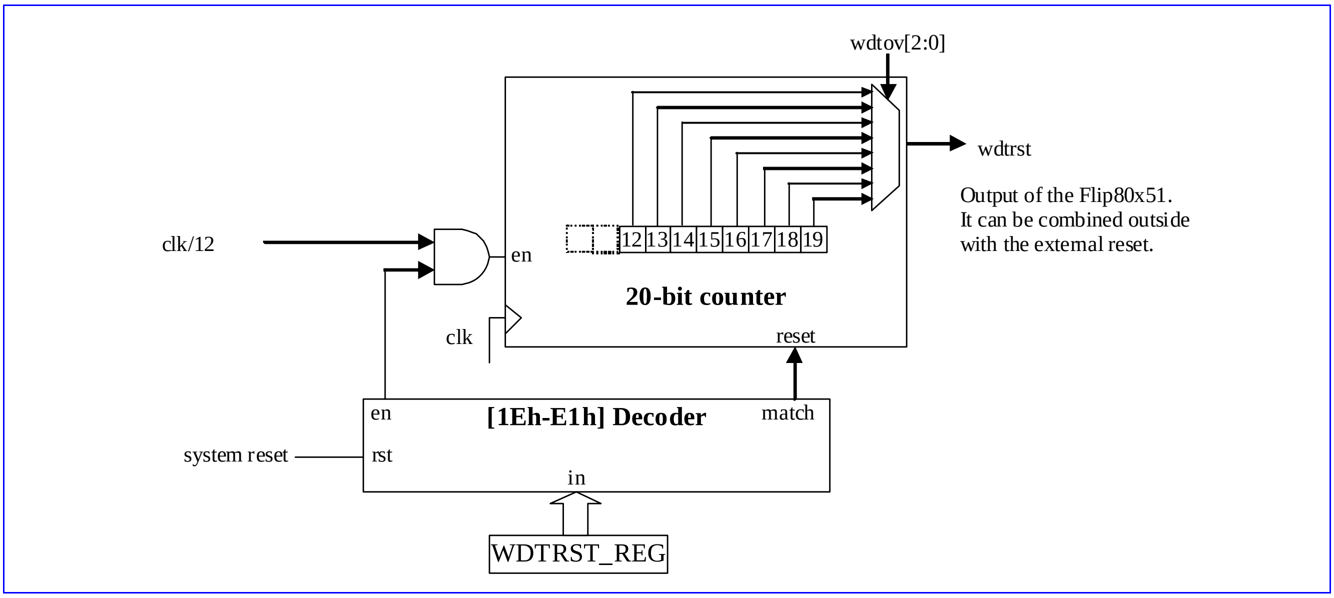


Figure 29: WDT block diagram

6.2.4.3 Using the WDT

To use the WDT to recover from software malfunctions, the user should control the WDT as follow:

- Configure the WDT using the WDTCON (S:0A5h) register, if needed.
- Enable the WDT by writing the two bytes sequence 1Eh-E1h to the WDTRST (S:0A6h). The WDT begins counting from zero.
- Repeatedly for the duration of the program execution, write the two bytes sequence 1Eh-E1h to the WDTRST register to clear and enable the WDT before it overflows. The watchdog timer starts over at zero.

[17] The WDT timer-out period depends on the system clock frequency

If the WDT overflows, it generates a one-clock cycle pulse, on the *wdtrst* output, which may be used to reset the system (the external RESET signal (*rst*) and the reset signal from WDT (*wdtrst*) are not combined internally).

If *wdtrst* is combined with another reset signal to provide the system reset (*rst*), then an overflow of the WDT initiates a system reset that will clear the WDT and disables it.

Note: If it is combined with the external reset, wdtrst needs to be latched in the reset management module; otherwise some glitches may appear on the reset signal.

6.2.4.4 WDT during power-down mode

The power-down mode stops the peripheral clock. This causes the WDT to stop counting and to hold its count. The WDT resumes counting from where it left off if the power-down mode is wake up. To ensure that the WDT does not overflow shortly after exiting the power-down mode, clear the WDT just before entering power-down. If the power-down mode is terminated by a reset, the WDT is cleared and disabled.

6.2.5 Timers registers

Mnemonic	Address	Description	Reset value
TCON	S:088h	Timer 0/1 control register	00h
TMOD	S:089h	Timer 0/1 mode select register	00h
T2CON	S:0C8h	Timer 2 control register	00h
T2MOD	S:0C9h	Timer 2 mode select register	00h
TH0	S:08Ch	Timer 0 high byte register	00h
TL0	S:08Ah	Timer 0 low byte register	00h
TH1	S:08Dh	Timer 1 high byte register	00h
TL1	S:08Bh	Timer 1 low byte register	00h
TH2	S:0CDh	Timer 2 high byte register	00h
TL2	S:0CCh	Timer 2 low byte register	00h
RCAP2H	S:0CBh	Timer 2 reload/capture high byte register	00h
RCAP2L	S:0CAh	Timer 2 reload/capture low byte register	00h
WDTCON	S:0A5h	Watchdog timer control register	07h
WDTRST	S:0A6h	Watchdog timer enable register	00h

Table 81: Timers registers

TH0 (S:8Ch)

Timer 0 High Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**High Byte of timer 0**

Figure 30: Timer 0 High Byte Register (TH0)

TH1 (S:8Dh)

Timer 1 High Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**High Byte of timer 1**

Figure 31: Timer 1 High Byte Register (TH1)

Interpretation of any unspecified point is absolutely up to the designer of this circuit

Disclosed under NDA only.

TH2 (S:CDh)

Timer 2 High Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**High Byte of timer 2**

Figure 32: Timer 2 High Byte Register (TH2)

TL0 (S:8Ah)

Timer 0 Low Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Low Byte of timer 0**

Figure 33: Timer 0 High Byte Register (TL0)

TL1 (S:8Bh)

Timer 1 Low Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Low Byte of timer 1**

Figure 34: Timer 1 Low Byte Register (TL1)

TL2 (S:CCh)

Timer 2 Low Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Low Byte of timer 2**

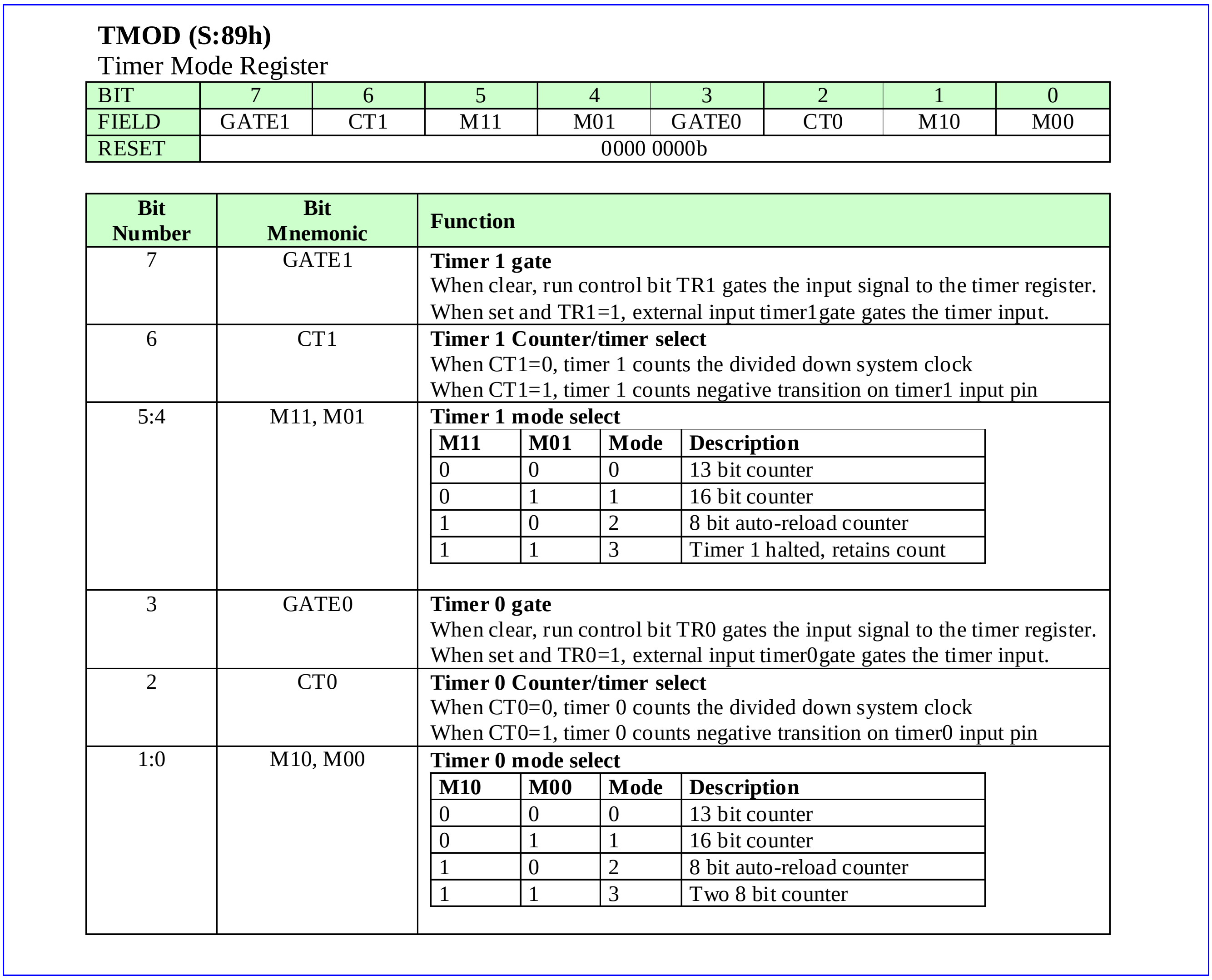
Figure 35: Timer 2 Low Byte Register (TL2)

TCON (S:88h)
Timer Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	TF1	TR1	TF0	TR0	IE1_	IT1	IE0_	IT0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	TF1	**Timer 1 overflow flag** Set by hardware when the timer 1 overflows. Cleared by hardware when the processor vectors to the interrupt routine
6	TR1	**Timer 1 run control bit** Set/cleared by software to turn timer 1 on/off
5	TF0	**Timer 0 overflow flag** Set by hardware when the timer 0 overflows. Cleared by hardware when the processor vectors to the interrupt routine
4	TR0	**Timer 0 run control bit** Set/cleared by software to turn timer 0 on/off
3	IE1_	**External interrupt 1 edge flag. Hardware controlled** Set when external interrupt 1 is detected. Cleared when interrupt is processed.
2	IT1	**External interrupt 1 signal type control bit.** Set to specify External interrupt 1 as falling edge triggered. Cleared to specify External interrupt 1 as low level triggered.
1	IE0_	**External interrupt 0 edge flag. Hardware controlled** Set when external interrupt 0 is detected. Cleared when interrupt is processed
0	IT0	**External interrupt 0 signal type control bit.** Set to specify External interrupt 0 as falling edge triggered. Cleared to specify External interrupt 0 as low level triggered.

Figure 36: Timer/Counter 0&1 control Register (TCON)



TMOD (S:89h)

Timer Mode Register

BIT	7	6	5	4	3	2	1	0
FIELD	GATE1	CT1	M11	M01	GATE0	CT0	M10	M00
RESET	0000 0000b							

<table>
<tr><th>Bit Number</th><th>Bit Mnemonic</th><th>Function</th></tr>
<tr><td>7</td><td>GATE1</td><td><b>Timer 1 gate</b>
When clear, run control bit TR1 gates the input signal to the timer register.
When set and TR1=1, external input timer1gate gates the timer input.</td></tr>
<tr><td>6</td><td>CT1</td><td><b>Timer 1 Counter/timer select</b>
When CT1=0, timer 1 counts the divided down system clock
When CT1=1, timer 1 counts negative transition on timer1 input pin</td></tr>
<tr><td>5:4</td><td>M11, M01</td><td><b>Timer 1 mode select</b>
<table>
<tr><th>M11</th><th>M01</th><th>Mode</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>13 bit counter</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>16 bit counter</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>8 bit auto-reload counter</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>Timer 1 halted, retains count</td></tr>
</table></td></tr>
<tr><td>3</td><td>GATE0</td><td><b>Timer 0 gate</b>
When clear, run control bit TR0 gates the input signal to the timer register.
When set and TR0=1, external input timer0gate gates the timer input.</td></tr>
<tr><td>2</td><td>CT0</td><td><b>Timer 0 Counter/timer select</b>
When CT0=0, timer 0 counts the divided down system clock
When CT0=1, timer 0 counts negative transition on timer0 input pin</td></tr>
<tr><td>1:0</td><td>M10, M00</td><td><b>Timer 0 mode select</b>
<table>
<tr><th>M10</th><th>M00</th><th>Mode</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>13 bit counter</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>16 bit counter</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>8 bit auto-reload counter</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>Two 8 bit counter</td></tr>
</table></td></tr>
</table>

Figure 37: Timer/Counter 0&1 mode select register (TMOD)

T2CON (S:C8h)

Timer 2 Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	TF2	**Timer 2 overflow flag** Set by hardware when the timer 2 overflows. Must be cleared by software *Note: TF2 is not set if RCLK=1 or TCLK=1*
6	EXF2	**Timer 2 external flag** Set by hardware (if EXEN2=1) when a negative transition on timer2capt is detected. Must be cleared by software *Note: EXF2 is not set if DCEN=1*
5	RCLK	**Receive clock bit.** If set, baud rate generator for the serial port 1 and 3 uses timer 2'overflow for its reception clock. If clear, it uses timer 1.
4	TCLK	**Transmit clock bit** If set, baud rate generator for the serial port 1 and 3 uses timer 2 overflow for its transmission clock. If clear, it uses timer 1.
3	EXEN2	**Timer 2 external enable bit** If set, enable a capture or a reload to occur as a result of a negative transition on timer2capt (if timer 2 is not being used to clock the serial port). If clear, timer 2 ignores events on timer2capt.
2	TR2	**Timer 2 run control bit** Set to start timer 2 running. Clear to stop the timer 2.
1	C/T2#	**Timer 2 counter/timer select** Set for counter operation: timer2 counts the negative transition on external pin timer2. Clear for timer operation: timer 2 counts the divided system clock.
0	CP/RL2#	**Capture reload bit** Set to capture on negative transitions on timer2capt if EXEN2=1. Clear to auto-reload on timer 2 overflow or negative transition on timer2capt if EXEN2=1. *Note: CP/RL2# is ignored and timer 2 is forced to auto reload on timer 2 overflow if RCLK=1 or TCLK=1.*

Figure 38: Timer/Counter 2 control register (T2CON)

T2MOD (S:C9h)
Timer2 Mode Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	--	--	--	--	T2OE	DCEN
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:2	--	**Reserved** The value read from these bits is indeterminate
1	T2OE	**Timer 2 output enable** In clock out mode, enables the programmable clock output
0	DCEN	**Down count Enable bit** If clear, configure timer 2 as an up counter. If set, configure timer 2 as an up/down counter.

Figure 39: Timer/Counter 2 mode select register (T2MOD)

WDTRST (S:A6h) write only
Watchdog Timer Reset Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Watchdog timer control data**

Figure 40: WDT reset register (WDTRST)

WDTCON (S:A5h)
Watchdog Timer Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	WDT3	WDT2	WDT1	WDT0	WDTR	WOV2	WOV1	WOV0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:4	WDT[3:0]	**4 less significant bits of the 20-bit watchdog timer** Read only
3	WRUN	**WDT run control bit** Read only
2:0	WOV[2:0]	**WDT Overflow control bits** When all three bits are set to 1, the watchdog timer has a nominal period of 1024 K clock cycles (20-bit counter). When all three bits are cleared to 0, the time-out period is 8 K clock cycles (13-bit counter)

Figure 41: WDT control register (WDTCON)

6.3 Serial Port (UART)

6.3.1 General description

The Flip80251-Hurricane provides a standard serial communication interface that supports all operation modes of the standard 8xC251. However, there is a small difference in pin functions. In the Flip80251-Hurricane separated serial port pins are available so that the serial communication does not affect the I/O port functions.
In the original 8xC251, in mode 0 of serial operation, one pin is used bi-directionally to transmit and receive data. In the Flip80251-Hurricane, the functions are separated out to the *serialin* and *serialout* signals, with the *serialdir* signal indicating whether it is a transmission (low level) or a reception (high level). The serialmode0 and *serialclk* signals are brought out as separate signals for mode 0 use. For others mode, *serialin* operates as RXD and *serialout* as TXD.

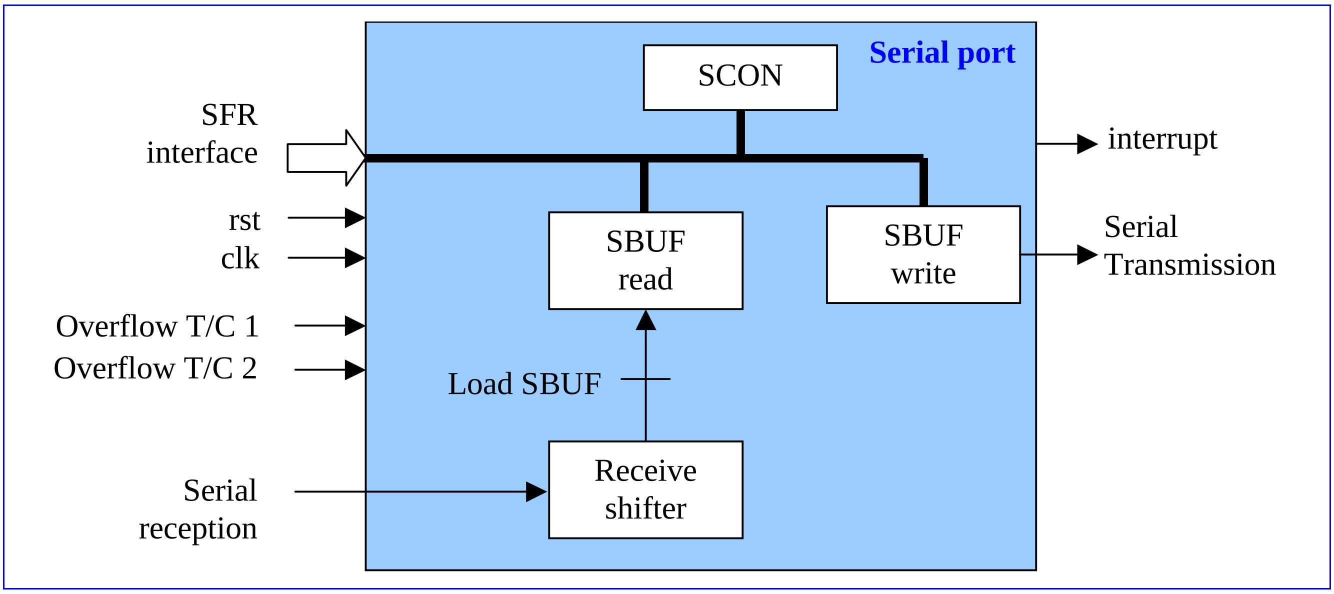


Figure 42: Serial port block diagram

6.3.2 Operation mode

6.3.2.1 Mode 0 (synchronous mode, half duplex)

In Mode 0 operation, the data transmission is carried out by the *serialout* signal and reception of data is handled by *serialin* signal. The transmission clock is output by the *serialclk* signal and the direction of signal transmission is indicated by the *serialdir* signal. The *serialclk* and *serialdir* signals are only activated in Mode 0. They are pulled high in other operational modes. The reason for separating the *serialin* and *serialout* signals is to avoid the use of internal tri-state buffers within the core.
With a simple additional circuit, the serial pins can be configured as a standard 8xC251. The signal *serialmode0,* which is active only when the serial port operates in mode 0, is provided to this aim.

Note: serialmode0 ← not (SCON.SM1 and SCON.SM0)

In Mode 0, the data transmission/receipt is 8 bit (no start bit and no stop bit). The clock frequency of data transmission is fixed at the micro-controller's clock / 12. To select the mode 0, clear SCON.SM0 and SCON.SM1.

6.3.2.1.1 Transmission

To send out data, clear the SCON.REN bit and write the data into the SBUF special function register. The data will then be shifted out (LSB first, MSB last), at the *serialout* pin.

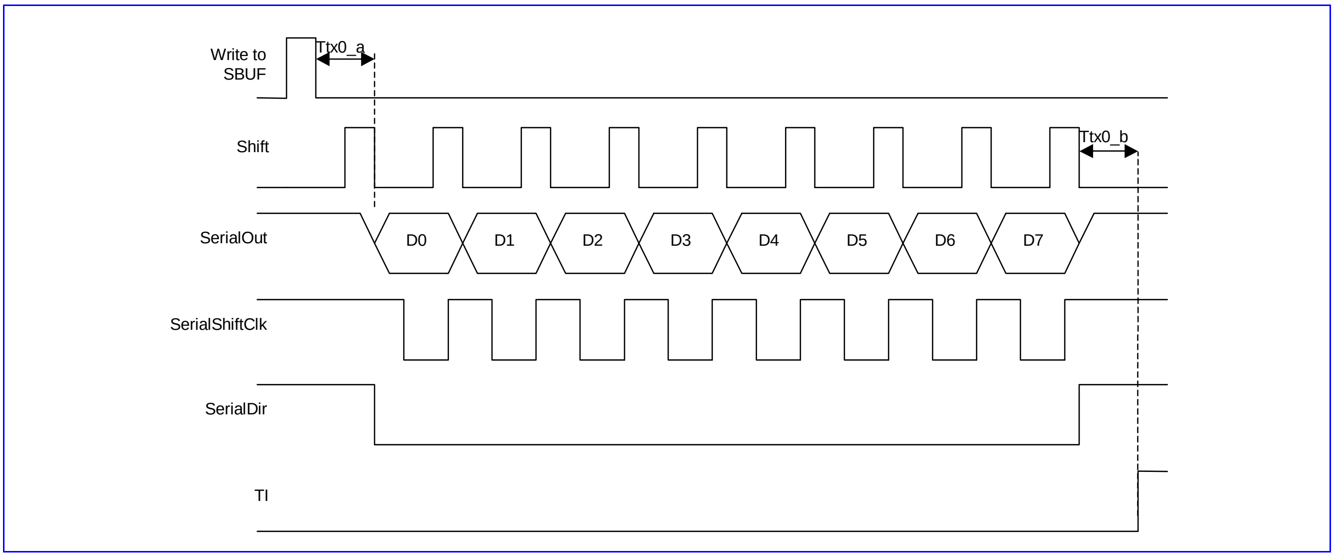


Figure 43: Serial Transmit Mode 0 diagram

6.3.2.1.2 Reception

To receive data, set the SCON.REN bit and clear the SCON.RI, this will enable the receive function. When received the data value can be read from the SBUF special function register.

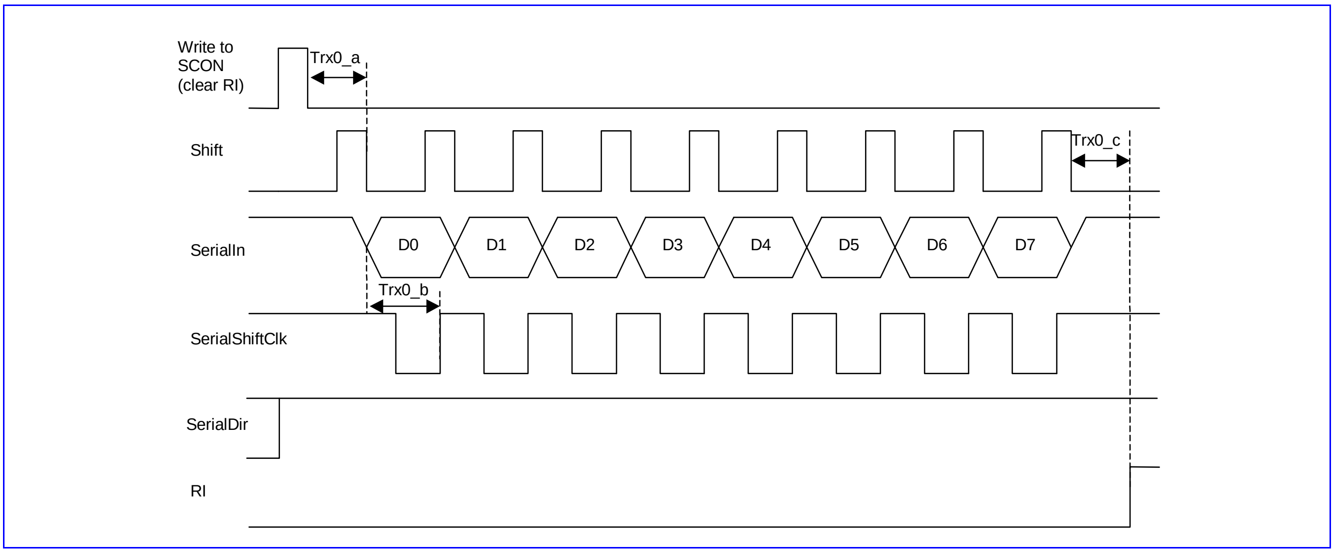


Figure 44: Serial Receive Mode 0 diagram

Time	Clk cycles (max.)	Time	Clk cycles (max.)
Ttx0_a	12	Trx0_a	12
Ttx0_b	18	Trx0_b	2
		Trx0_c	18

Table 82: Transmission/reception delay in serial mode 0

6.3.2.2 Mode 1 (asynchronous mode, full duplex)

In Mode 1, data is transmitted through *serialout* signal and received through *serialin* signal. The data is composed of 10 bits: starting with a start bit “0”, then followed by 8 data bits (LSB first, MSB last), and then the stop bit “1”. The Baud Rate in Mode 1 is controlled by Timer1 or Timer2 and is programmable. Please refer to Programming the Baud Rate, in later part of this chapter for details. To select the mode 1, clear SCON.SM0 and set SCON.SM1.

6.3.2.2.1 Transmission

To send out data, clear the SCON.REN bit and write the data into the SBUF special function register. The data will then be shifted out (LSB first, MSB last), at the *serialout* pin.

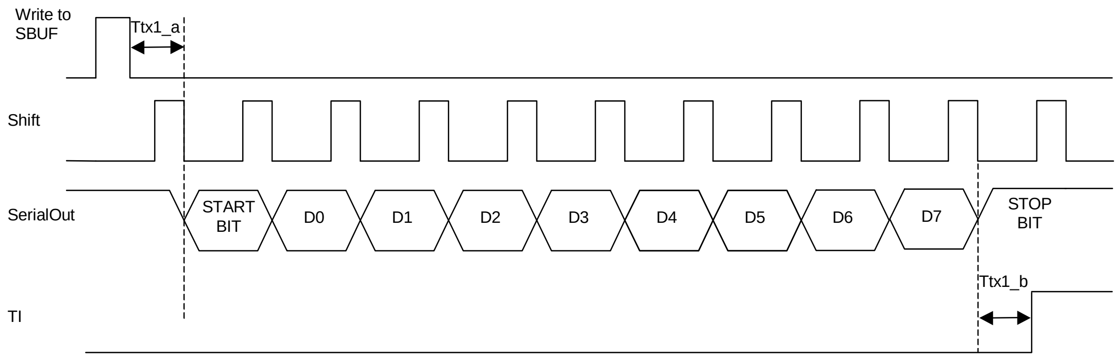


Figure 45: Serial Transmit Mode 1 diagram

6.3.2.2.2 Reception

To receive data, set the SCON.REN bit and clear the SCON.RI, this will enable the receive function. When received the data value can be read from the SBUF special function register.

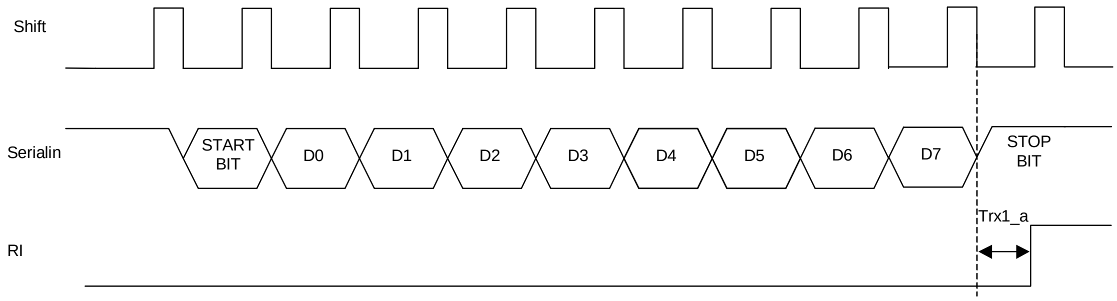


Figure 46: Serial Receive Mode 1 diagram

Time	clk cycles (max.)
Ttxl_a	baudrate
Ttxl_b	18
Trxl_a	18

Table 83: Transmission/reception delay in serial mode 1

6.3.2.3 Mode 2 (asynchronous mode, full duplex)

In Mode 2, data is transmitted through *serialout* signal and received through *serialin* signal. The data is composed of 11 bits: 1 start bit, 8 data bits, 1 TB8 bit (in SCON) and the stop bit. The extra TB8 bit is for use in a multiprocessor communication environment. When multiprocessor communication support is not needed, this bit can also be used as a parity bit. The data transfer rate in Mode 2 is fixed as clk/32 or clk/64. Timer 1 and Timer 2 are independent of the Baud Rate generation and can be used for other purposes. To select the mode 2, set SCON.SM0 and clear SCON.SM1.

6.3.2.3.1 Transmission

To send out data, clear the SCON.REN bit and write the data into the SBUF special function register. The data will then be shifted out (LSB first, MSB last), at the *serialout* pin.

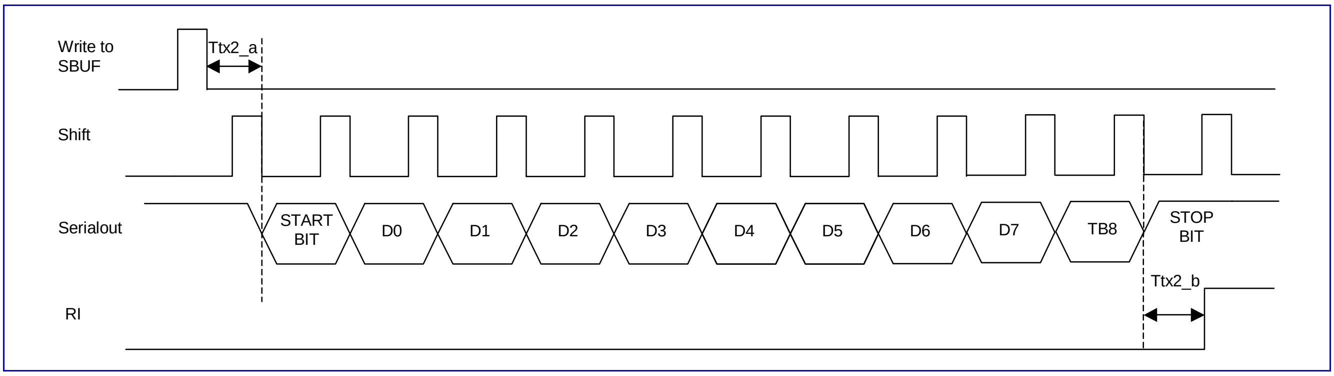


Figure 47: Serial Transmit Mode 2 diagram

6.3.2.3.2 Reception

To receive data, set the SCON.REN bit and clear the SCON.RI, this will enable the receive function. When received the data value can be read from the SBUF special function register.

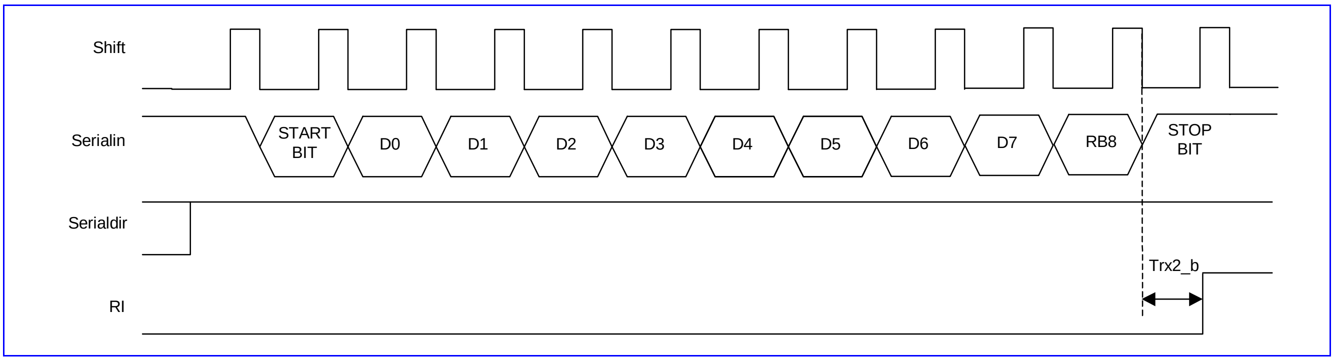


Figure 48: Serial Receive Mode 2 diagram

Time	Clk cycles (max.)
Ttx2_a	baudrate
Ttx2_b	18
Trx2_a	18

Table 84: Transmission/reception delay in serial mode 2

6.3.2.4 Mode 3 (asynchronous mode, full duplex)

The operation of Mode 3 is same as Mode 2. The only difference is that Timer1 (or Timer 2) controls the Baud Rate. Serial Mode 3 has the same timing diagram as Mode 2 (above), but the source of the shift pulse is different. To select the mode 1, set SCON.SM0 and SCON.SM1.

6.3.3 Framing bit error detection

Framing bit error detection is provided for the mode 1, 2 and 3. To enable the framing bit error detection feature, set the PCON.SMOD0 bit. When this feature is enabled, the receiver checks each incoming data frame for a valid stop bit. If a valid stop bit is not found, the serial sets the SCON.FE bit.

Software may examine the SCON.FE bit after each reception to check for data errors. Once set, only software or a reset can clear the SCON.FE bit. Subsequently received frames with valid stop bit cannot clear the FE bit.

6.3.4 Multiprocessor communication

Modes 2 and 3 provide a SCON.TB8 bit to facilitate a multiprocessor communication. To enable this feature, set the SCON.SM2 bit. Then the serial port can differentiate between data frames (SCON.TB8 clear) and address frames (SCON.TB8 set). This allows the micro controller to function as a slave processor in an environment where multiple slave processors share a single serial line.

When the multiprocessor communication feature is enabled, the receiver ignores frames with the ninth bit (SCON.TB8) clear. The receiver examines frames with the ninth bit set for an address match. If the received address matches the slave's address, the serial port set the SCON.RB8 bit and the SCON.RI bit, generating an interrupt.

The addressed slave's software then clears the SCON.SM2 bit and prepares to receive the data bytes. The others slaves are unaffected by these data bytes because they are waiting for address frames (ninth bit set).

Note that the IE.ES bit must be set to allow SCON.RI bit to generate an interrupt.

6.3.5 Automatic address recognition

The automatic address recognition feature is enabled when the multiprocessor communication feature is enabled (SCON.SM2 bit set). Implemented in hardware, this feature enhances the multiprocessor communication feature by allowing the serial port to examine the address of each incoming address frame (ninth bit set). Only when the serial port recognizes its own address does the receiver set the SCON.RI bit to generate an interrupt. This ensures that the CPU is not interrupted by frames addressed to other devices.

If desired, you may enable the automatic address recognition feature in mode 1. In this mode, the stop bit takes the place of the TB8.bit. The SCON.RI bit is set only when the received frame address matches the device's address and is terminated by a valid stop bit.

Note that the multiprocessor communication and automatic address recognition features cannot be enabled in mode 0, i.e. setting the SCON.SM2 bit in mode 0 has no effect.

To support automatic address recognition, a device is identified by a given address and a broadcast address.

6.3.5.1 Given address

Each device has an individual address that is specified in the SADDR special function register. The SADEN special function register is a mask byte that contains don't-care bits (defined by zeros) to form the device's given address. These don't-care bits provide the flexibility to address one or more slaves at a time. The following example illustrates how a given address is formed. Note that to address a device by its individual address, the SADEN mask byte must be FFh.

6.3.5.2 Broadcast address

A broadcast address is formed from the logical OR of the SADDR and SADEN register with zeros defined as don't-care bits. The use of don't-care bits provides flexibility in defining the broadcast address, however in most applications, a broadcast address is FFh

6.3.5.3 Addressing a slave serial port

A slave serial port wills response both to its given address and it broadcast address. With the following configuration,

SADDR	: 01101001
SADEN	: 11111011
Given	: 01101x01
Broadcast	: 11111x11

The master can communicate with this slave with 4 addresses (2 given and 2 broadcast):

01101001 and 01101101 (given address)
11111011 and 11111111 (broadcast address)

6.3.5.4 Reset addresses

On reset, the SADDR and SADEN special function registers are initialized to 00h, i.e. the given and broadcast addresses are xxxxxxxx (all don't-care bits). This ensures that the serial port is backwards compatible with the 80C51 micro controllers that do not support automatic address recognition.

6.3.6 Programming the Baud Rate

6.3.6.1 Mode 0

In Serial Mode 0, the Baud Rate is fixed to clk/12. No Timer/Counters need to be set up (only the SCON register).

6.3.6.2 Modes 1 & 3 - Timer1 generating Baud Rate

Timer 1 generates the Receive Clock when T2CON.RCLK=0 and the Transmit Clock when T2CON.TCLK=0, (or always in the Flip80251-Hurricane without the Timer2). Timer1 should be set up in timer auto-reload mode.

$$\text{Baud Rate} = \frac{\text{(PCON.SMOD+1)*clk}}{\text{32*12*(256-TH1)}}$$

Given a baud rate, the reload value for TH1 is

$$\text{TH1} = 256 - \frac{\text{(PCON.SMOD+1)*clk}}{\text{384*Baud Rate}}$$

If TH1 is not an integer value then either the Baud Rate or clk frequency must be changed.

6.3.6.3 Modes 1 & 3 - Timer2 generating Baud Rate

Timer 2 can generate the Receive Clock in the Flip80251-Hurricane, when T2CON.RCLK=1 and the Transmit Clock when T2CON.TCLK=1. If Timer2 is being clocked internally,

$$\text{Baud Rate} = \frac{\text{clk}}{\text{32*(65536-(RCAP2H,RCAP2L))}}$$

The reload value for RCAP2H, RCAP2L is given by

$$\text{RCAP2H, RCAP2L} = 65536 - \frac{\text{clk}}{\text{32*Baud Rate}}$$

Otherwise if Timer2 is being clocked by the Timer2 signal, Baud Rate = Timer2 Overflow rate/16

6.3.6.4 Mode 2

In serial mode 2 the Baud Rate is fixed to (PCON.SMOD +1)/64.

6.3.7 Serial port registers

Mnemonic	Address	Description	Reset value
SCON	S:098h	Serial port control	00h
SBUF[18]	S:099h	Serial buffer	00h
SADDR	S:0A9h	Serial address	00h
SADEN	S:0B9h	Serial address Enable	00h

Table 85: Serial port registers

SADDR (S:A9h)
Slave Individual Address Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Slave individual address**

Figure 49: UART Slave Individual Address Register (SADDR)

SADEN (S:B9h)
Slave Address Mask Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Mask data for slave individual address**

Figure 50: UART Slave Individual Address Mask Register (SADEN)

SBUF (S:99h)
Serial Buffer

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Data sent/received bu serial I/O port**

Figure 51: UART Serial Buffer Register (SBUF)

[18] *Two separate registers constitute the SBUF SFR. Writing to SBUF load the transmit register; reading from SBUF accesses the receive buffer*

SCON (S:98h)

Serial control Register

BIT	7	6	5	4	3	2	1	0
FIELD	FE SM0	SM1	SM2	REN	TB8	RB8	TI	RI
RESET	0000 0000b							

<table>
<tr><th>Bit Number</th><th>Bit Mnemonic</th><th>Function</th></tr>
<tr><td>7</td><td>FE
SM0</td><td>**Framing Error Bit:**
To select this function, set the PCON.SMOD0 bit.
FE is set by hardware to indicate an invalid stop bit and cleared by software. FE is not cleared by valid frames
Serial port mode bit 0
To select this function, clear the PCON.SMOD0 bit.
Software writes to SM1 and SM0 to select the serial port' operating mode.</td></tr>
<tr><td>6</td><td>SM1</td><td>**Serial port mode bit 1**
<table>
<tr><th>SM0</th><th>SM1</th><th>Mode</th><th>Description</th><th>Baud rate</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Shift register</td><td>Clk/12</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>8 bit UART</td><td>Variable</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>9 bit UART</td><td>Clk/32 or Clk/64</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>9 bit UART</td><td>Variable</td></tr>
</table></td></tr>
<tr><td>5</td><td>SM2</td><td>**Serial port mode bit 2**
Software writes to bit SM2 to enable or disable the multiprocessor communication and automatic address recognition features.</td></tr>
<tr><td>4</td><td>REN</td><td>**Receiver Enable Bit**
Set for reception, clear for transmission</td></tr>
<tr><td>3</td><td>TB8</td><td>**Transmit bit 8**
In mode 2 and 3, software writes the ninth data bit to be transmitted to TB8. Not used in mode 1 and 0</td></tr>
<tr><td>2</td><td>RB8</td><td>**Receiver bit 8. (Not used in mode 0)**
Set or cleared by hardware to reflect the stop bit in mode 1. SM2 must be cleared
Set or cleared by hardware to reflect the ninth bit in mode 2 & 3. SM2 must be set.</td></tr>
<tr><td>1</td><td>TI</td><td>**Transmit interrupt flag**
Set by the transmitter after the last data bit transmitted. Cleared by software</td></tr>
<tr><td>0</td><td>RI</td><td>**Receive interrupt flag**
Set by the receiver after the last data bit of a frame has been received. Cleared by software</td></tr>
</table>

Figure 52: Serial Port control register (SCON)

6.4 Programmable counter array (PCA)

6.4.1 PCA overview

The PCA consists of a 16-bit timer/counter and up to five 16-bit compare/capture modules. The timer/counter is used as a common time base/event counter for the compare/capture modules, distributing the current count to the modules through a 16-bit bus. The operating modes of the five compare/capture modules determine the functions performed by the PCA. Each module can be independently programmed to provide input capture, output compare or pulse width modulation. Only the module 4 has additional watchdog-timer mode.

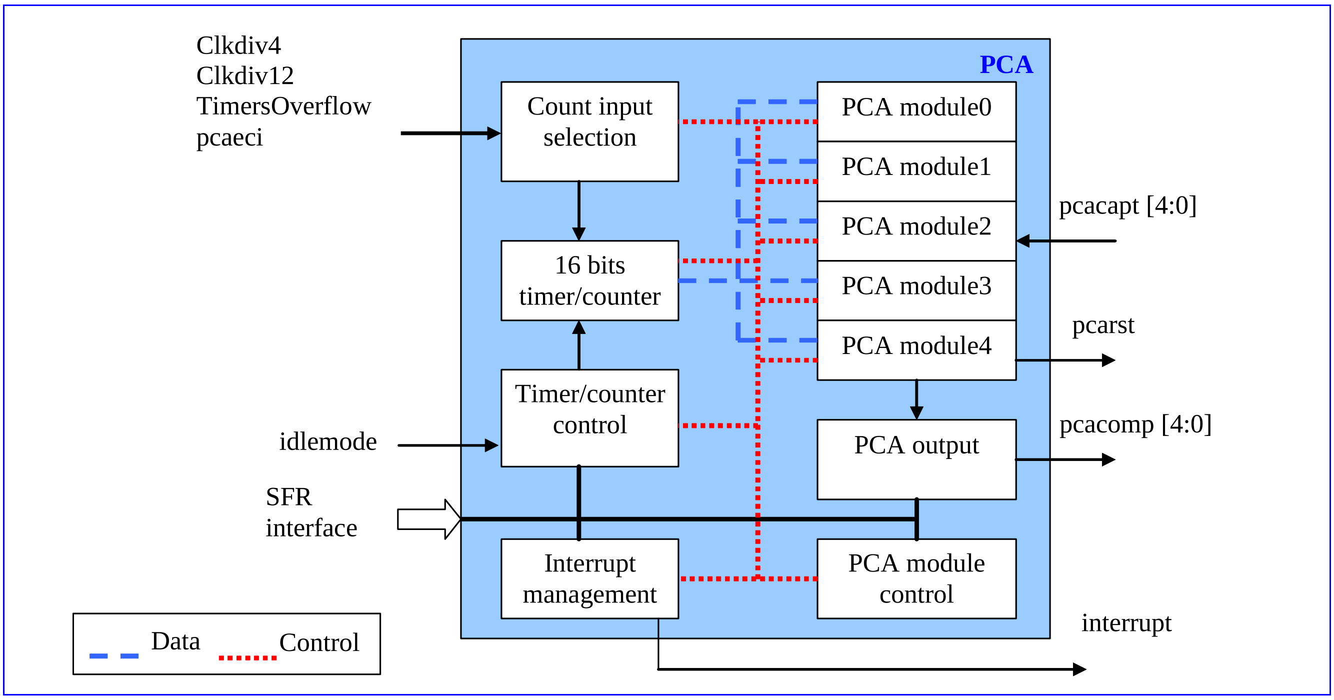


Figure 53: PCA block diagram

6.4.2 PCA timer/counter

The CH/CL special function register pair operates as a 16-bit timer/counter. The selected input increments the CL (low byte) register. When CL overflows, the CH (high byte) register increments immediately (and not two cycles after as the standard 8xC251). When CH overflows it sets the PCA overflow flag (CCON.CF) generating a PCA interrupt request if the CMOD.ECF bit is set.
The CMOD.CPS1 and CMOD.CPS0 bits select one of four signals as input to the timer/counter:

- CLK/12. Provides a clock pulse every peripheral cycle (12 clock cycles)
- CLK/4. Provides a clock pulse every 4-clock cycle.
- Timer0 overflow. The CL is incremented when timer 0 overflows. This selection provides the PCA with a programmable frequency input.
- ECI. The CPU samples the *pcaeci* signal every 4-clock cycle. The first clock pulse that occurs following a high-to-low transition at the *pcaeci* signal increments the CL register. The maximum input frequency for this input selection is CLK/8

Setting the run control bit (CCON.CR) turns the PCA timer/counter on, if the output of the NAND gate (see Figure 54) equals "1". The PCA timer/counter continues to operate during idle mode unless the CMOD.CIDL bit is set. The CPU can read the contents of the CH and CL registers at any time. However, writing to them is inhibited while they are counting (CCON.CR is set)

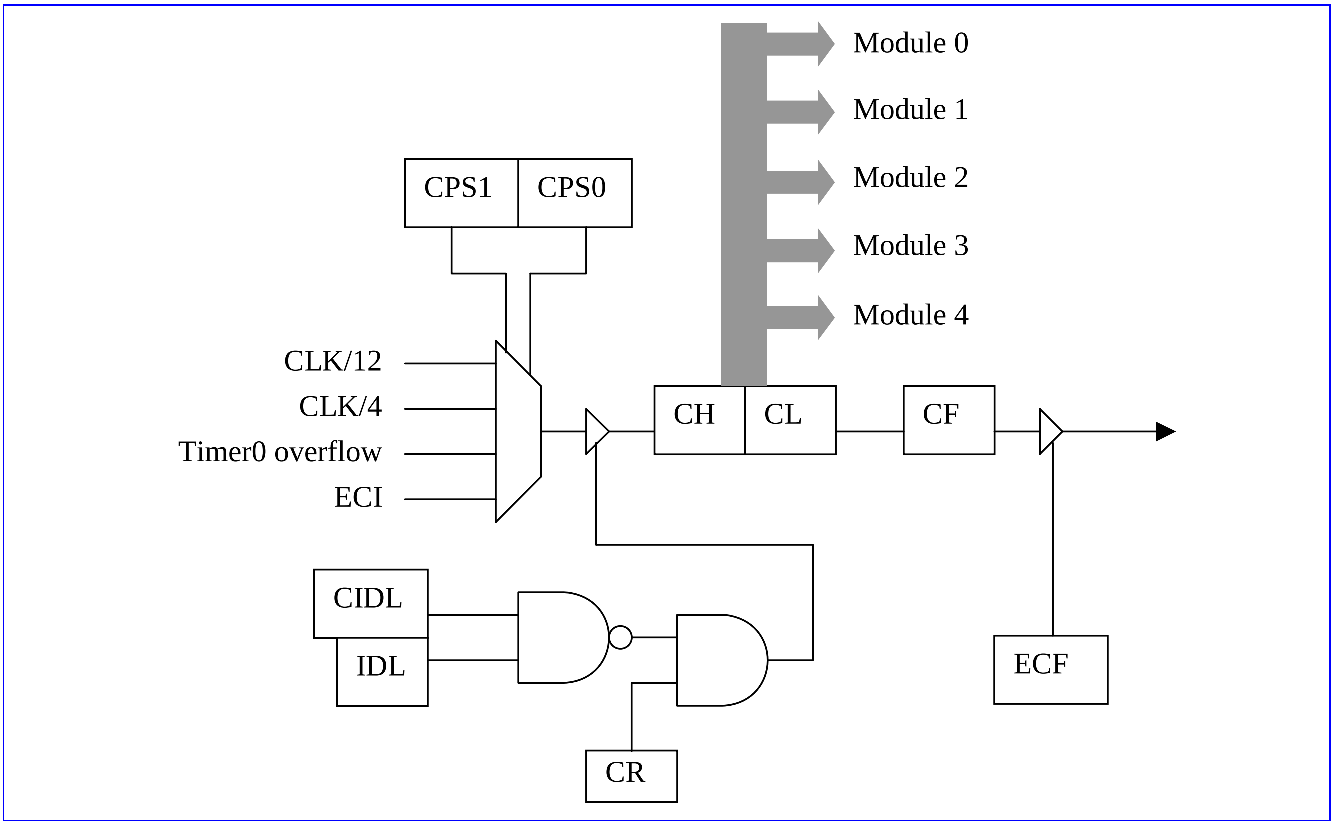


Figure 54: Programmable Counter Array

6.4.3 PCA compare/capture modules

Each compare/capture module is made up of a compare/capture register pair (CCAPxH/CCAPxL), a 16 bits comparator, and various logic gates and signal transition selectors. The registers store the time or count at which an external event occurred (capture) or at which an action should occur (comparison). In the PWM mode, the low byte register controls the duty cycle of the output waveform.

The logical configuration of a compare/capture module depends on its mode of operation. Each module can be independently programmed for operation in any of the following modes:

- 16-bit capture mode with triggering on the positive edge, negative edge or either edge.
- Compare modes: 16-bit software timer, 16-bit high-speed output, 16-bit watchdog timer (module 4 only) or 8-bit pulse width modulation.
- No operation.

Bit combinations programmed into a compare/capture modules mode register (CCAPMx) determine the operating mode.

The compare/capture modules perform their programmed functions when their common time base, the PCA timer/counter, runs. The timer/counter is turned on and off with the CCON.CR bit. To disable any given module, program it for the no operation mode. The occurrence of a capture, software timer, or high-speed output event in a compare/capture modules sets the module's compare/capture flag (CCON.CCFx) and generates a PCA interrupt request if the corresponding enable bit in the CCAPMx register is set.

The CPU can read or write the CCAPxH and CCAPxL registers at any time.

6.4.4 16-bit Capture Mode

The capture mode provides the PCA with the ability to measure periods, pulse widths, duty cycles, and phase differences at up to five separate inputs. Signals *capture0* through *capture4* are sampled for signal transitions (positive and/or negative as specified). When a compare/capture module programmed for the capture mode detects the specified transition, it captures the PCA timer/counter value. It records the time at which an external event is detected, with a resolution equal to the timer/counter clock period.

To program a capture/compare module for the 16-bit capture mode, program the CCAPMX.CAPPx and CCAPMx.CAPNx bits as follows:

- To trigger the capture on a positive transition, set CAPPx and clear CAPNx.
- To trigger the capture on a negative transition, set CAPNx and clear CAPPx.
- To trigger the capture on a positive or a negative transition set both CAPPx and CAPNx.

For modules in the capture mode, detection of a valid signal transition at the *pcacapt[*x*]* signal causes hardware to load the current PCA timer/counter value into the compare/capture registers (CCAPxH/CCAPxL) and to set the module's compare/capture flag (CCON.CCFx). If the corresponding interrupt enable bit (CCAPMx.ECCFx) is set, the PCA sends an interrupt request to the interrupt handler.

Since hardware does not clear the event flag when the interrupt is processed, the user must clear the flag in software. A subsequent capture by the same module overwrites the existing captured value. To preserve a captured value, save it in RAM with the interrupt service routine before the next event occurs.

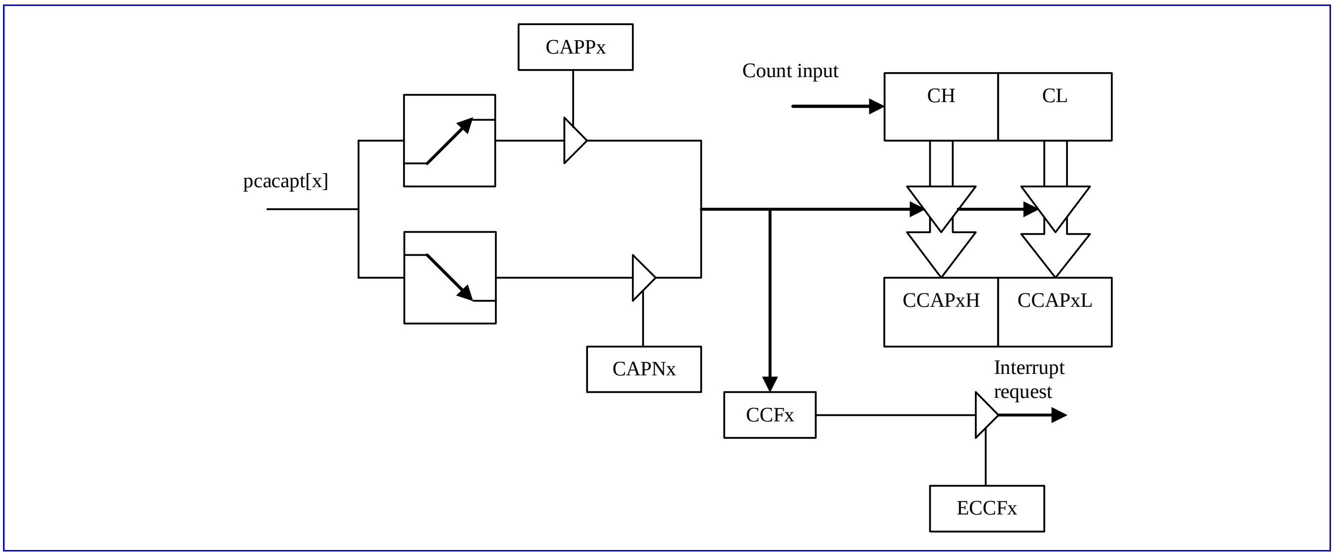


Figure 55: PCA 16-bit Capture Mode

6.4.5 Compare Modes

The compare function provides the capability for operating the five modules as timers, event counters, or pulse width modulators. Four modes employ the compare function: 16-bit software timer mode, high-speed output mode, WDT mode and PWM mode. In the first three of these, the compare/capture module continuously compares the 16-bit PCA timer/counter value with the 16-bit value pre-loaded into the module’s CCAPxH/CCAPxL register pair. In the PWM mode, the module continuously compares the value in the low-byte PCA timer/counter register (CL) with an 8-bit value in the CCAPxL module register. Comparisons are made every 4-clock cycle to match the fastest PCA timer/counter clocking rate.

Setting the CCAPMx.ECOMx bit selects the compare function for that module. To use the modules in the compare modes, observe the following general procedure:

- Select the module’s mode of operation
- Select the input signal for the PCA timer/counter.
- Load the comparison value into the module’s compare/capture register pair.
- Set the PCA timer/counter run control bit.
- After a match causes an interrupt, clear the module’s compare/capture flag.

6.4.5.1 16-bit Software Timer Mode

To program a compare/capture module for the 16-bit software timer mode, set the CCAPMx.ECOMx and CCAPMx.MATx bits.

A match between the PCA timer/counter and the compare/capture registers (CCAPxH/CCAPxL) set the module’s compare/capture flag (CCON.CCFx). This generates an interrupt request if the corresponding interrupt enable bit (CCAPMx.ECCFx) is set. Since hardware does not clear the compare/capture flag when the interrupt is processed, the user must clear the flag in software. During the interrupt routine, a new 16-bit compare value can be written to the compare/capture registers (CCAPxH/CCAPxL).

Note: In order to prevent an invalid match while updating these registers, user software should write CCAPxL first then CCAPxH. A write to CCAPxL clears the ECOMx bit disabling the compare function, while a write to CCAPxH sets the ECOMx bit, re-enabling the compare function.

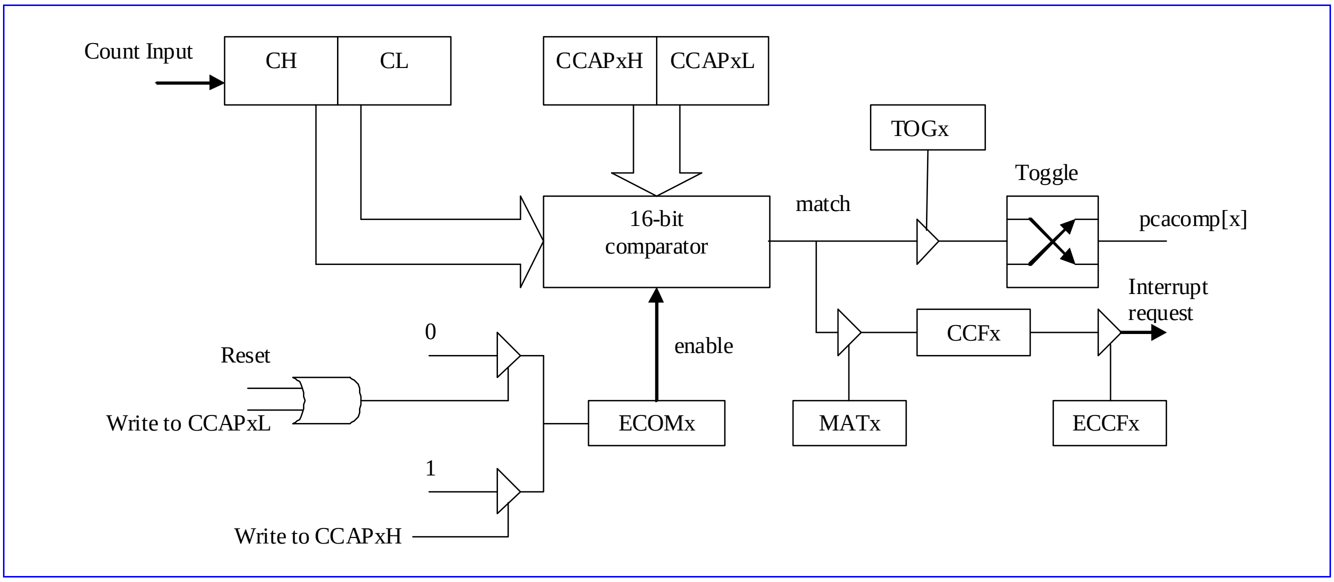


Figure 56: PCA software Timer and High-speed Output modes

6.4.5.2 High-speed Output Mode

The High-speed output mode generates an output signal by toggling the module's *pcacomp[x]* output signal when match occurs. This provides greater accuracy than toggling output signal in software because the toggle occurs before the interrupt request is serviced. Thus, interrupt response time does not affect the accuracy of the output.

To program a compare/capture module for high-speed output mode, set CCAPMx.ECOMx, CCAPMx.MATx and CCAPMx.TOGx bits. A match between the PCA timer/counter and the compare/capture registers (CCAPxH/CCAPxL) toggles the *pcacomp[x]* signal and set the module's compare/capture flag (CCON.CCFx). By setting or clearing the *pcacomp[x]* signal in software, the user selects whether the match toggles the signal from low to high or high to low.

The user also has the option of generating an interrupt request when the match occurs by setting the corresponding interrupt enable bit (CCAPMx.ECCFx). Since hardware does not clear the compare/capture flag when the interrupt is processed, the user must clear the flag in software.

If the user does not change the compare/capture registers in the interrupt routine, the next toggle occurs after the PCA timer/counter rolls over and count again matches the comparison value. During the interrupt routine, a new 16-bit compare value can be written to the compare/capture registers (CCAPxH/CCAPxL).

Note: in order to prevent an invalid match while updating these registers, user software should write CCAPxL first then CCAPxH. A write to CCAPxL clears the ECOMx bit disabling the compare function, while a write to CCAPxH sets the ECOMx bit, re-enabling the compare function.

6.4.5.3 PCA watchdog timer mode

In addition to the Flip80251's hardware WDT, the PCA provides a programmable-frequency 16-bit WDT as a mode option on compare/capture module4. This mode generates a device reset when the count in the PCA timer/counter matches the value stored in the module 4 compare/capture registers (CCAP4H/CCAP4L). The PCA WDT reset signal (pcarst) is available as an independent output. The external reset (rst), the hardware WDT reset (wdtrst) and the PCA WDT reset are not internally combined. The user is free to combine or to use them independently. Module 4 is the only PCA module that has the WDT mode. When not programmed as a WDT, it can be used in the other modes.

To program module 4 for the PCA WDT mode, set the CCAPM4.ECOM4, CCAPM4.MAT4 and CMOD.WDTE. Also select the desired input for the PCA timer/counter by programming CMOD.CPS0 and CMOD.CPS1. Enter a 16-bit comparison value in the compare/capture registers (CCAP4H/CCAP4L). Enter a 16-bit initial value in the PCA timer/counter (CH/CL) or use the reset value (0000h). The difference between these values multiplied by the PCA input pulse rate determines the running time to expiration. Set the timer/counter run control bit (CCON.CR) to start the PCA WDT.

The PCA WDT generates a reset signal each time a match occurs. To hold off a PCA WDT reset, the user has three options.

- Periodically change the comparison value in CCAP4H/CCAP4L so a match never occurs.
- Periodically change the PCA timer/counter value (CH/CL) so a match never occurs.
- Disable the module 4 reset output signal (pcarst) by clearing the CMOD.WDTE bit before a match occurs, then later re-enable it.

The first two options are more reliable because the WDT is not disabled as in the third option.

The second option is not recommended if other PCA modules are in use, since the five modules share a common time base. Thus, in most applications the first option is the best one.

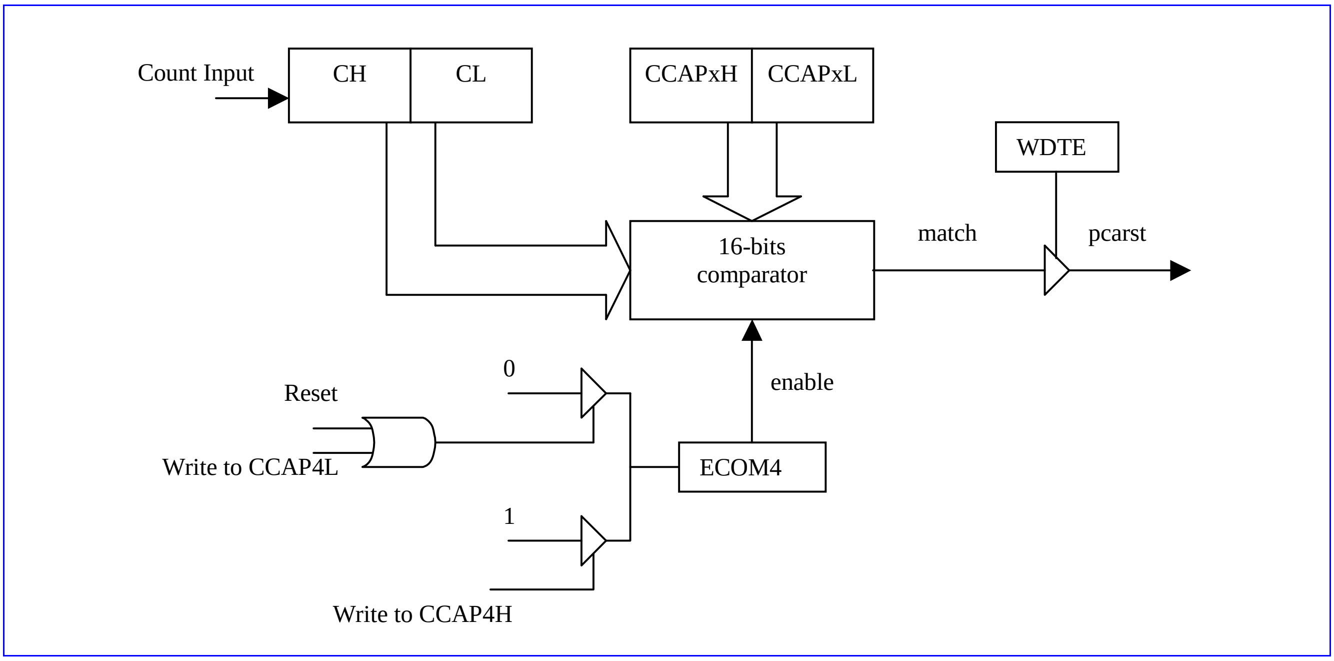


Figure 57: PCA watchdog timer mode

6.4.5.4 Pulse width modulation mode

The five PCA comparator/capture modules can be independently programmed to function as pulse width modulators. The modulated output, which has a pulse width resolution of eight bits, is available at the *pcacomp[*x*]* signal.

In this mode the value in the low byte of the PCA timer/counter (CL) is continuously compared with the value in the low byte of the compare/capture register (CCAPxL). When CL < CCAPxL, the output waveform is low. When a match occurs (CL =CCAPxL), the output waveform goes to high and remains high until CL rolls over from FFh to 00h, ending the period. At rollover the output returns to a low, the value in CCAPxH is loaded into CCAPxL, and a new period begins.

The value in CCAPxL determines the duty cycle of the current period. The value in CCAPxH determines the duty cycle of the following period. Changing the value in CCAPxL over time modulates the pulse width. As depicted in Figure 58, the 8-bit value in CCAPxL can vary from 0(100% duty cycle) to 255(0.4% of duty cycle). To change the value in CCAPxL without glitches, write the new value to the high byte register (CCAPxH). This value is shifted by hardware into CCAPxL when CL rolls over FFh to 00h.

To program a compare/capture module for the PWM mode, set the CCAPMx.ECOMx and CCAPMx.PWMx bits. Also select the desired input for the PCA timer/counter by programming CMOD.CSP0 and CMOD.CSP1. Enter an 8-bit value in CCAPxL to specify the duty cycle of the first period of the PWM output waveform. Enter an 8-bit value in CCAPxH to specify the duty cycle of the second period. Set the timer/counter run control bit (CCON.CR) to start the PCA timer/counter.

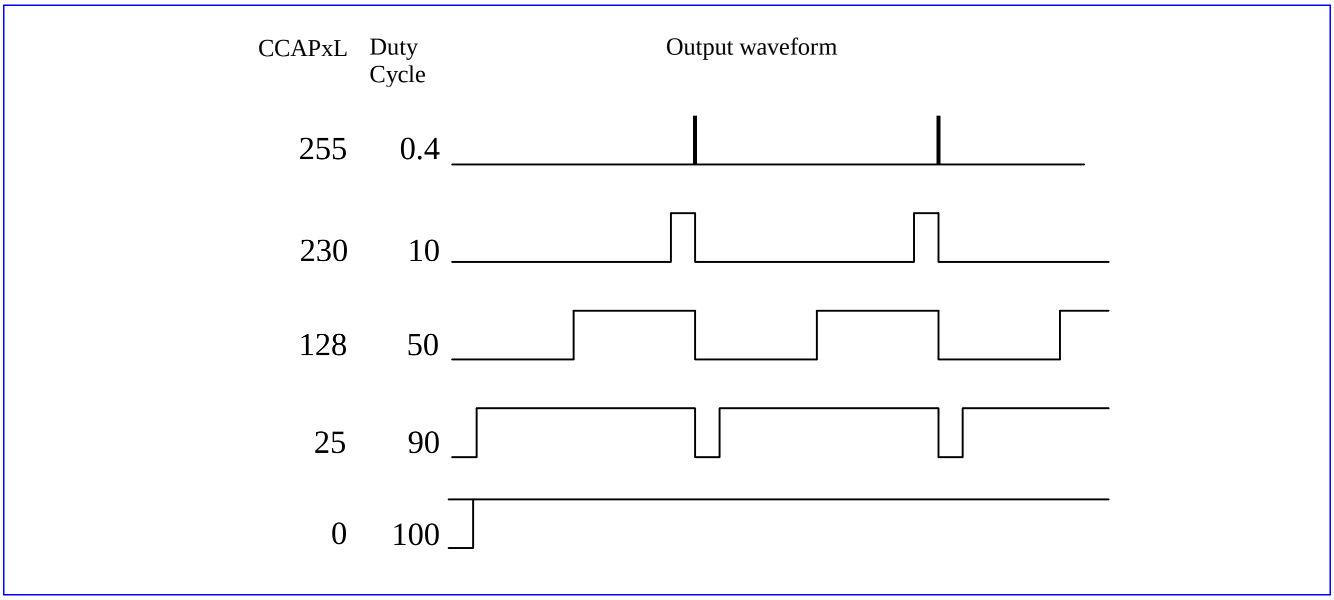


Figure 58: PWM variable Duty cycle

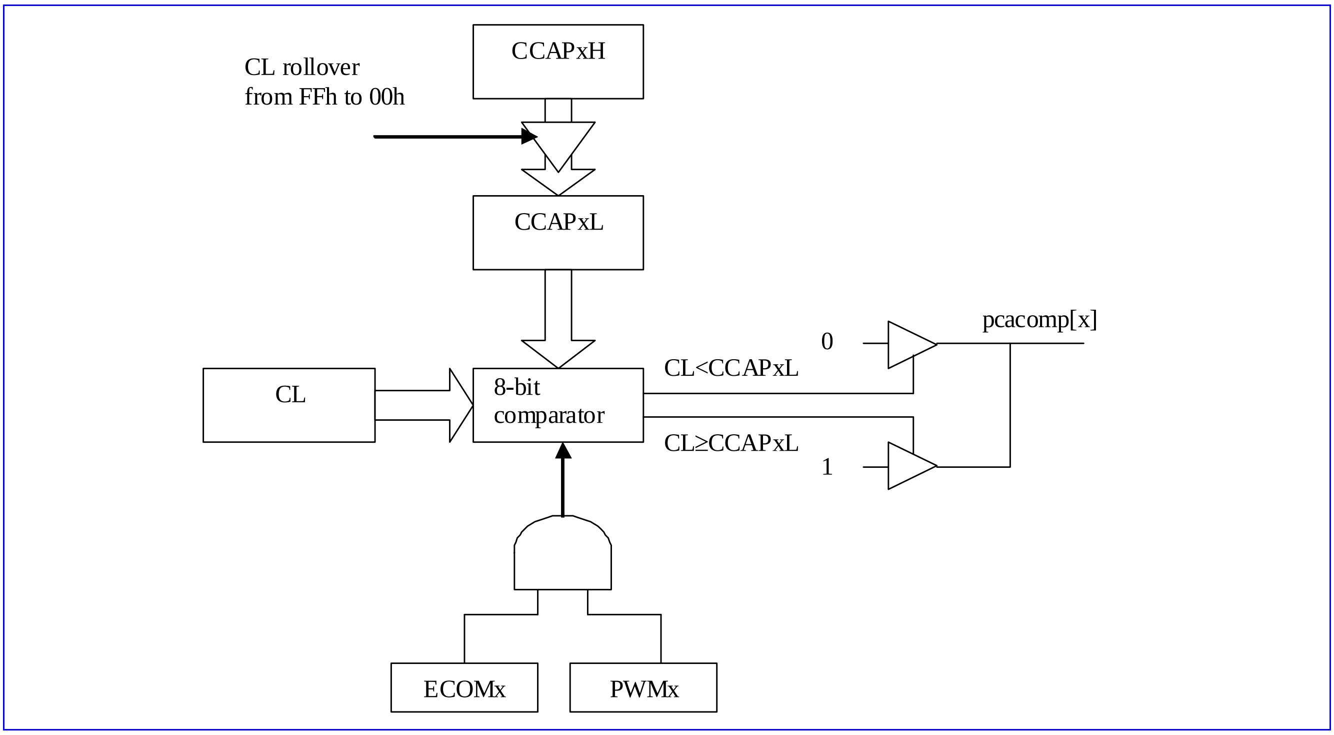


Figure 59: PCA 8-bit PWM mode

6.4.6 PCA registers

Mnemonic	Address	Description	Reset value
CCON	S:0D8h	PCA Timer/Counter Control	00h
CMOD	S:0D9h	PCA Timer/Counter Mode	00h
CH	S:0F9h	PCA Timer/Counter high byte	00h
CL	S:0E9h	PCA Timer/Counter low byte	00h
CCAPM0	S:0DAh	PCA Compare/Capture Mode for Module 0	00h
CCAPM1	S:0DBh	PCA Compare/Capture Mode for Module 1	00h
CCAPM2	S:0DCh	PCA Compare/Capture Mode for Module 2	00h
CCAPM3	S:0DDh	PCA Compare/Capture Mode for Module 3	00h
CCAPM4	S:0DEh	PCA Compare/Capture Mode for Module 4	00h
CCAP0H	S:0FAh	PCA Compare/Capture Module 0 high byte	00h
CCAP0L	S:0EAh	PCA Compare/Capture Module 0 low byte	00h
CCAP1H	S:0FBh	PCA Compare/Capture Module 1 high byte	00h
CCAP1L	S:0EBh	PCA Compare/Capture Module 1 low byte	00h
CCAP2H	S:0FCh	PCA Compare/Capture Module 2 high byte	00h
CCAP2L	S:0ECh	PCA Compare/Capture Module 2 low byte	00h
CCAP3H	S:0FDh	PCA Compare/Capture Module 3 high byte	00h
CCAP3L	S:0EDh	PCA Compare/Capture Module 3 low byte	00h
CCAP4H	S:0FEh	PCA Compare/Capture Module 4 high byte	00h
CCAP4L	S:0EEh	PCA Compare/Capture Module 4 low byte	00h
CCAPO	S:0DFh	PCA timer/counter Output for PWM and HS mode	00h

Table 86: PCA registers

CCON (S:D8h)

PCA Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	CF	CR	--	CCF4	CCF3	CCF2	CCF1	CCF0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	CF	**PCA timer/counter overflow Flag.** Set by hardware when the PCA timer/counter rolls over. This generates an interrupt request if the CMOD.ECF interrupt bit is set. CF can be set by hardware or software, but can be cleared only by software.
6	CR	**PCA Timer/counter Run Control bit.** Set and cleared by software to turn the PCA timer/counter on and off
5	--	**User flag** This is a general purpose flag
4:0	CCF4:0	**PCA Module compare/capture flags.** Set by hardware when a match or capture occurs. This generates a PCA interrupt request if the CCAPMx.ECCFx interrupt enable bit is set. Must be cleared by software.

Figure 60: PCA timer/counter control register (CCON)

CMOD (S:D9h)
PCA Mode Register

BIT	7	6	5	4	3	2	1	0
FIELD	CIDL	WDTE	UF2	UF1	UF0	CPS1	CPS0	ECF
RESET	0000 0000b							

<table>
<tr><th>Bit
Number</th><th>Bit
Mnemonic</th><th>Function</th></tr>
<tr><td>7</td><td>CIDL</td><td><b>PCA Timer/counter idle control.</b>
If set, the timer/counter is stopped during idle mode</td></tr>
<tr><td>6</td><td>WDTE</td><td><b>PCA Watchdog Timer enable.</b>
If set, the watchdog timer output on module 4 is enable</td></tr>
<tr><td>5:3</td><td>UF2:0</td><td><b>User flags</b>
General purpose flag</td></tr>
<tr><td>2:1</td><td>CPS1:0</td><td><b>PCA Timer/counter input select.</b>
<table>
<tr><th>CPS1</th><th>CPS0</th><th>Mode</th><th>Input</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>clkdiv12</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>clkdiv4</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>timer0overflow</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>ECI (max frequency = system clock / 8)</td></tr>
</table></td></tr>
<tr><td>0</td><td>ECF</td><td><b>PCA timer/counter interrupt enable.</b>
If set, an overflow of the PCA timer/counter generates an interrupt request.</td></tr>
</table>

To save an interrupt request during a read-modify-write instruction on CCON, or any concurrent write access, the bits CF and CCFx are updated by hardware only at the end of the current instruction

Figure 61: PCA timer/counter mode register (CMOD)

CL (S:E9h)
Low Byte of PCA Timer/counter Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Low byte of PCATimer/Counter**

Figure 62: PCA Timer/counter Register Low Byte Register (CL)

CH (S:F9h)
High Byte of PCA Timer/counter Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**High byte of PCATimer/Counter**

Figure 63: PCA Timer/counter Register high byte Register (CH)

CCAPM0 (S:DAh)
CCAPM1 (S:DBh)
CCAPM2 (S:DCh)
CCAPM3 (S:DDh)
CCAPM4 (S:DEh)
PCA Compare/Capture Module x Mode Register (x=0, 1, 2, 3, 4)

BIT	7	6	5	4	3	2	1	0
FIELD	--	ECOMx	CAPPx	CAPNx	MATx	TOGx	PWMx	ECCFx
RESET	0000 0000b							

Bit Number	**Bit Mnemonic**	**Function**
7	--	**Reserved**. The value read from this bit is indeterminate
6	ECOMx	**Compare modes.** Clear to disable the compare function. Set to enable the compare function The compare function is used to implement the software timer mode, the high speed output mode, the PWM mode and the watchdog timer mode
5	CAPPx	**Capture mode (positive)** Set to enable the capture function on a positive edge of CEXx.
4	CAPNx	**Capture mode (negative)** Set to enable the capture function on a negative edge of CEXx.
3	MATx	**Match.** If set, a match of the PCA timer/counter will generate an interrupt request (ECCFx must also be set)
2	TOGx	**Toggle.** If set, a match of the PCA timer/counter toggles the CEXx output
1	PWMx	**Pulse Width modulation mode.** Set to configure the module x as an 8-bits PWM
0	ECCFx	**Enable CCFx Interrupt.** Set to enable the compare/capture flag CCON.CCFx to generate an interrupt request.

Figure 64: PCA compare/capture module mode register (CCAPMx)

CCAP0H (S:FAh)
CCAP1H (S:FBh)
CCAP2H (S:FCh)
CCAP3H (S:FDh)
CCAP4H (S:FEh)
High Byte Compare/Capture Module x Register (x=0, 1, 2, 3, 4)

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	**Bit Mnemonic**	**Function**
7:0		**High Byte of PCA comparison or capture value**

Figure 65: PCA Compare/Capture Module x High Byte Registers (CCAPxH)

CCAP0L (S:EAh)
CCAP1L (S:EBh)
CCAP2L (S:ECh)
CCAP3L (S:EDh)
CCAP4L (S:EEh)

Low Byte Compare/Capture Module x Register (x=0, 1, 2, 3, 4)

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Low Byte of PCA comparison or capture value**

Figure 66: PCA Compare/Capture Module x Low Byte Registers (CCAPxL)

CCAPO (S:DFh

PCA timer/counter output for PWM and high-speed mode Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	--	CCAPO4	CCAPO3	CCAPO2	CCAPO1	CCAPO0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:5	--	**Reserved**. The value read from this bit is indeterminate
4	CCAPO4	**Compare/capture module 4 output value**
3	CCAPO3	**Compare/capture module 4 output value**
2	CCAPO2	**Compare/capture module 4 output value**
1	CCAPO1	**Compare/capture module 4 output value**
0	CCAPO0	**Compare/capture module 4 output value**

CCAPO register can be written by hardware and by software (through SFR interface). In case of concurrent writing in the same clock cycle, then only the software write will be performed

Figure 67: PCA timer/counter output for PWM and high-speed mode register (CCAPO)

7. INTERRUPT SYSTEM

The Flip80251 has the same interrupt sources as the Intel 8xC251, plus the NMI input relative to the TSC8x251GxD specification. These are handled the same as on the original 8xC251, however the Flip80251 has a shorter interrupt latency period, and can distinguish shorter external interrupt pulses. The interrupt sources are sampled every clock cycle (clock rising edge), and the decision of whether an interrupt will be accepted takes place at the last clock cycle of each instruction execution, or every clock cycle during idle mode.

7.1 Introduction

The original 8xC251 employs a program interrupt method similar to the one of 80C51. This operation branches to a subroutine and performs some service in response to the interrupt. When the subroutine completes, execution resumes at the point where the interrupt occurred. Interrupts may occur as a result of internal activity (e.g. timer0 overflow) or at the initiation of an external device (external interrupt pin). In any case, interrupt operation is programmed by the system designer, who determines the priority of interrupt service, compare to relative normal code execution or other interrupt service routines. All the interrupts may be enabled / disabled dynamically by the system designer except two interrupts, TRAP (software) and NMI (hardware) that are non-maskable.

A typical interrupt process occurs as follow:

- An external device initiates an interrupt request signal.
- This event on the signal, connected to an input pin and sampled by the Flip80251, is registered into a flag buffer.
- The priority of the flag is compared to the priority of the other interrupt by the interrupt controller. A higher priority causes the controller to set an interrupt flag.
- The setting of the interrupt flag indicates to the control unit to execute a context switch. This context switch breaks the current instruction execution flow[19].
    - When *intrmode* input is low[20] (2-byte interrupt frame), the control unit completes the current instruction execution prior to saving the two lower bytes of the program counter (PC) and reloads the PC with the interrupt vector address, which is the start address of a software service routine.
    - When *intrmode* input is high (4-byte interrupt frame), the control unit completes the current instruction execution prior to saving the 3 bytes of the program counter (PC) and the PSW1 register (S:0D1h) and reloads the PC with the interrupt vector address, which is the start address of a software service routine.
- The software service routine performs the assigned tasks and executes a RETI instruction as a final instruction. This instruction signals the completion of the interrupt, resets the interrupt-in-progress priority.
    - When *intrmode* input is low[2] (2-byte interrupt frame), the RETI instruction reloads the two bytes of the program counter and uses them as the 16-bit return address in region FF. Program execution then continues from the original point of interruption.

[19] If the interrupt flag is set during the last cycle of the current instruction execution, the interrupt is handled at the end of the next instruction execution.

[20] 80C51 compatible interrupt mode configuration.

- When *intrmode* input is high (4-byte interrupt frame), the RETI instruction reloads the program counter and restores the PSW1 register (S:0D1h) with theirs previous saved values. Program execution then continues from the original point of interruption.

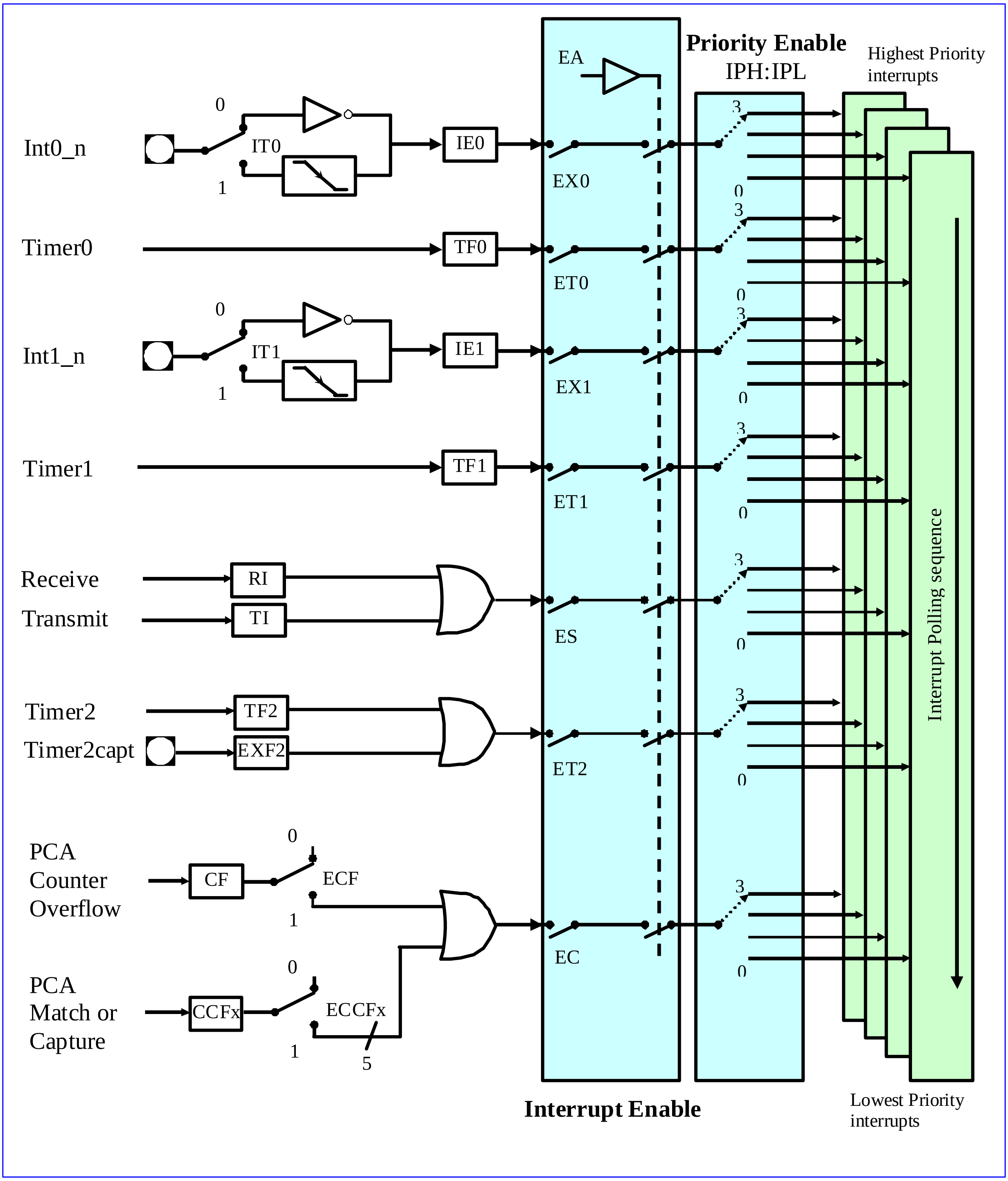


Figure 68: Interrupt control system

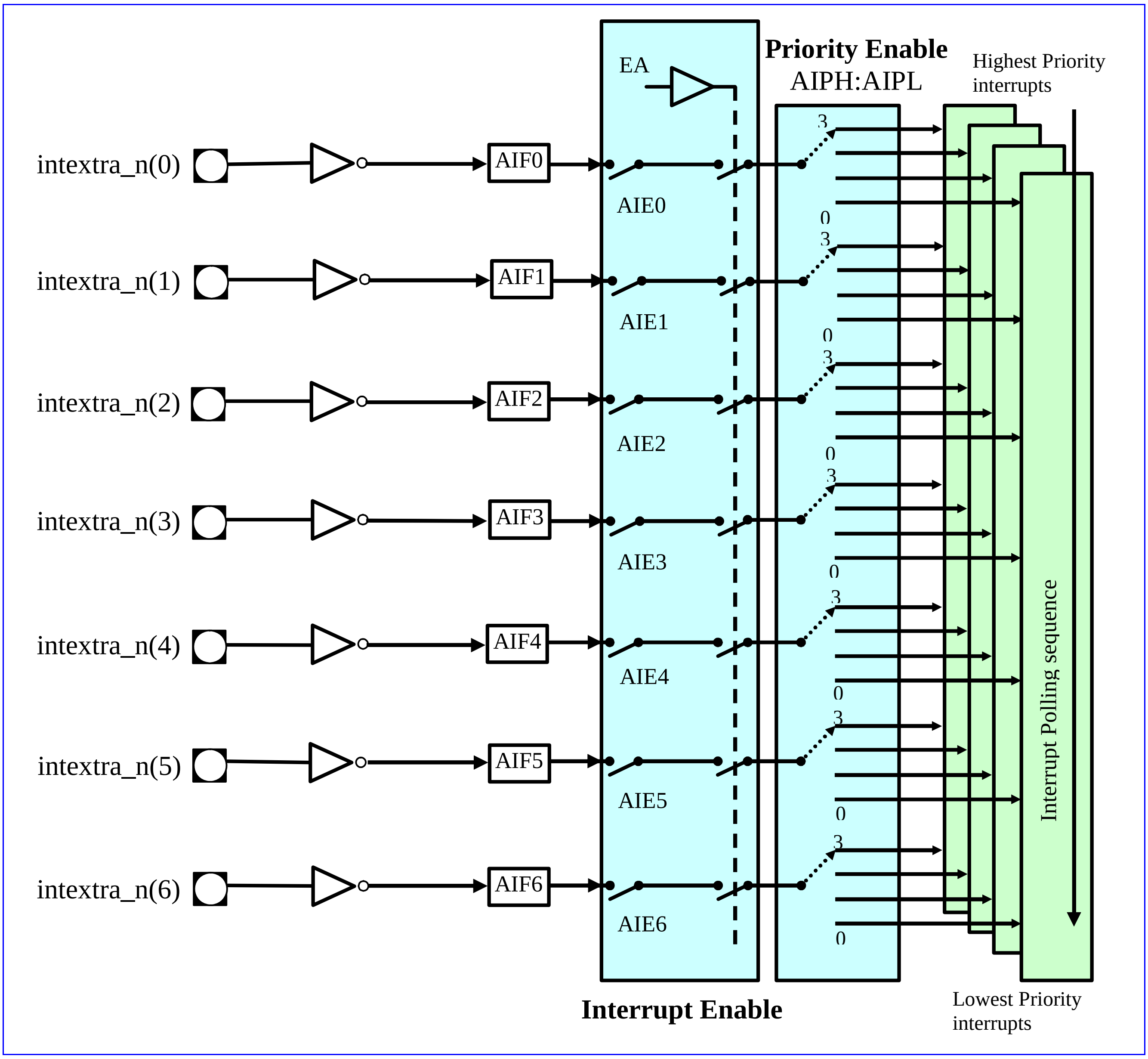


Figure 69: Additional Interrupts control system

7.2 Interrupt sources

The Flip80251 has a software's interrupt, the TRAP instruction (always enabled) and up to fifteen hardware' interrupt sources. Fourteen of these hardware interrupt are maskable interrupt sources and one is non-maskable (NMI input *intnmi*, always enabled). The maskable sources include two external interrupts (*int0_n* and *int1_n*), three timer's interrupts (timers 0, 1, and 2), one programmable counter array (PCA) interrupt, and one serial port (UART) interrupt. Depending on configuration, seven additional external interrupt (*intextra_n*[6:0]) are available and maskable.

Each interrupt (except TRAP and NMI) has an interrupt request flag, which can be set by software as well as by hardware. For some interrupts, hardware clears the request flag when it grants an interrupt. Software can clear any request flag to cancel an impending interrupt.

Name	I/O	Function
int0_n	I	**External Interrupt 0** This input set bit IE0_ in the TCON register. If bit IT0 in the TCON register are set, bit IE0_ is controlled by a negative edge trigger on int0_n. If bit IT0 is clear, bit IE0_ is controlled by a low level trigger on int0_n.
int1_n	I	**External Interrupt 1** This input set bit IE1_ in the TCON register. If bit IT1 in the TCON register are set, bit IE1_ is controlled by a negative edge trigger on int1_n. If bit IT1 is clear, bit IE1_ is controlled by a low level trigger on int1_n
intnmi	I	**Non-Maskable Interrupt input** High level triggered.
intextra_n[6-0][21]	I	**Additional Interrupt 6: 0** These inputs set bits AIF[6:0] in the AIF register. Bits AIF6-AIF0 are controlled by a low level trigger on intextra_n[6:0]

Table 87: Interrupt system pins

7.2.1 External interrupts

7.2.1.1 Interrupt sampling

External interrupt pins are sampled every clock cycle. Edge-triggered external interrupts must hold the request pin low for at least two clock cycles. This ensures edge recognition and sets interrupt request bits IEx_. The CPU clears IEx_ automatically during service routine fetch cycles for edge triggered interrupts.

Since pending interrupts are evaluated during the last execution cycle of the current instruction, a level-triggered interrupt must held low (or high) the request pin for at least the duration of the longest instruction (DIV WRjd, WRjs – 20 cycles) to guarantee detection.

[21] Only available in additional interrupts configuration

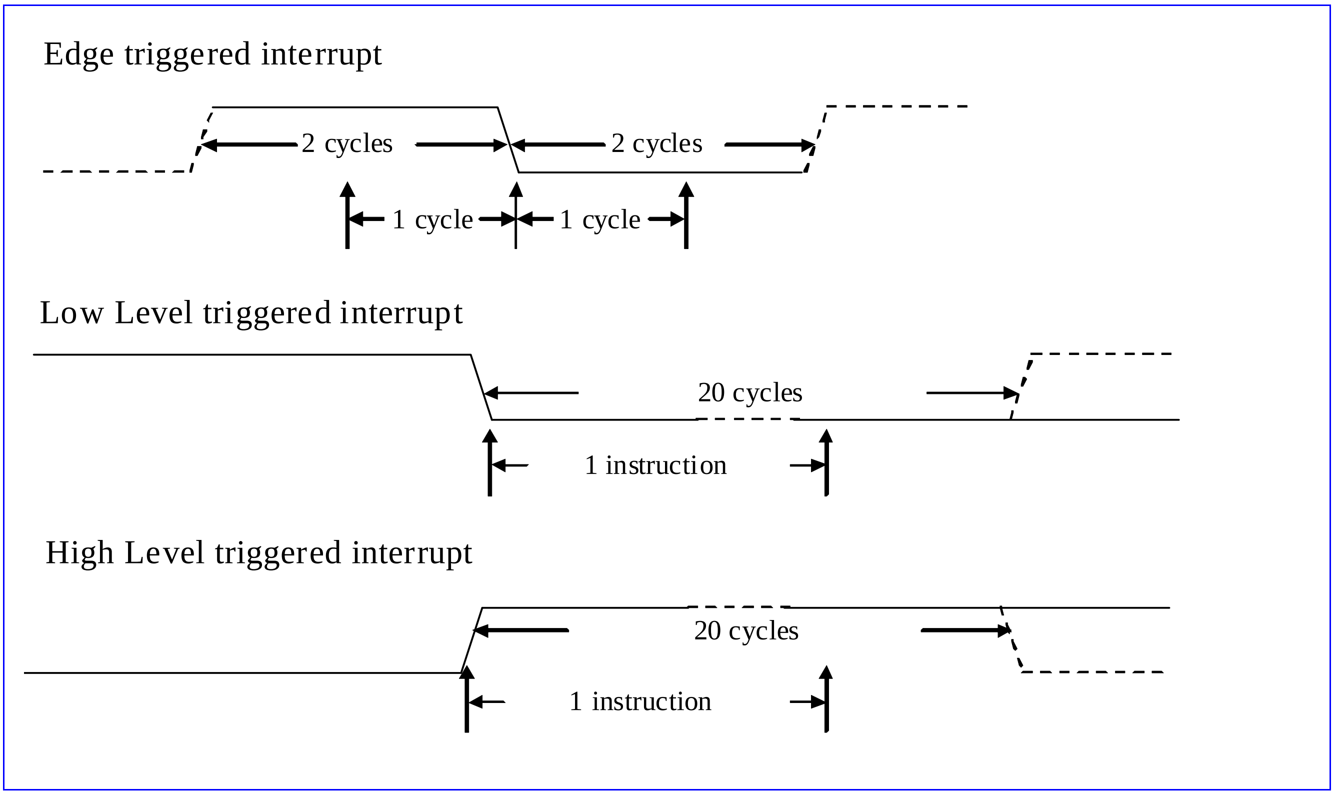


Figure 70: Minimum pulse timing

7.2.1.2 int0_n and int1_n

External interrupt *int0_n* and *int1_n* may be each programmed to be low level-triggered or edge-triggered, depending upon bits IT0 and IT1 in TCON register (S:088h).
If ITx=0, *intx_n* is triggered by a low level at the pin. If IT0=1, *int0_n* is negative-edge triggered.
External interrupts are enabled with bits EX0 and EX1 in IE0 register.
Events on *int0_n* or *int1_n* set respectively the interrupt request flag IE0_ or IE1_ in TCON register. If the interrupt is edge-triggered, the hardware jump to the service routine clears the request flag. Otherwise, if the interrupt is level triggered, then the interrupt must be de-asserted before the end of the ISR (before the execution of the "RETI").

External interrupt inputs *int0_n* and *int1_n* provide both the capability to exit from Power-down mode on low-level signal.

Note: In case of level-triggered interrupt, the interrupt request flags (IE0_ and IE1_) are the image of the interrupt pins. Thus, if these flags are set by software (e.g. setb IE0_), they will be cleared by hardware at the next cycle if the corresponding pin is high.
In others words, in case of level-triggered interrupt, software set of interrupt flags does not cause interrupt service.

TCON (S:88h)

Timer Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	TF1	TR1	TF0	TR0	IE1_	IT1	IE0_	IT0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	TF1	**Timer 1 overflow flag** Set by hardware when the timer 1 overflows. Cleared by hardware when the processor vectors to the interrupt routine
6	TR1	**Timer 1 run control bit** Set/cleared by software to turn timer 1 on/off
5	TF0	**Timer 0 overflow flag** Set by hardware when the timer 0 overflows. Cleared by hardware when the processor vectors to the interrupt routine
4	TR0	**Timer 0 run control bit** Set/cleared by software to turn timer 0 on/off
3	IE1_	**External interrupt 1 edge flag. Hardware controlled** Set when external interrupt 1 is detected. Cleared when interrupt is processed.
2	IT1	**External interrupt 1 signal type control bit.** Set to specify External interrupt 1 as falling edge triggered. Cleared to specify External interrupt 1 as low level triggered.
1	IE0_	**External interrupt 0 edge flag. Hardware controlled** Set when external interrupt 0 is detected. Cleared when interrupt is processed
0	IT0	**External interrupt 0 signal type control bit.** Set to specify External interrupt 0 as falling edge triggered. Cleared to specify External interrupt 0 as low level triggered.

Figure 71: Timer/Counter 0&1 control Register (TCON)

7.2.1.3 NMI interrupt

NMI input (*intnmi*) is the non-maskable interrupt input. Since NMI is high level-triggered input, *intnmi* signal must be de-asserted before the end of the interrupt service routine. NMI input provides the capability to exit from Power-down mode on high-level signal.

7.2.1.4 Additional interrupts

Application may require more interrupt sources for additional peripherals. The additional interrupts configuration can provide up to 7 extra interrupt sources to match customer needs. This configuration requires the use of four new SFRs: Additional interrupt Flag register (AIF), Additional Interrupt Enable Register (AIE), Additional Interrupt Priority Low Register (AIPL) and Additional Interrupt Priority High Register (AIPH).

Mnemonic	**Address**	**Description**	**Reset value**
AIF	S:0C0h	Additional Interrupt Flag Register	00h
AIE	S:0E8h	Additional Interrupt Enable Register	00h
AIPH	S:0F7h	Additional Interrupt Priority High Register	00h
AIPL	S:0F8h	Additional Interrupt Priority Low Register	00h

Table 88: Additional interrupt registers

The additional external sources are level activated. An additional interrupt is triggered by a detected low at the corresponding pin.[22]

- Each of the additional external interrupt sources may be individually programmed to one of four priority levels. This is accomplished by one bit in the Additional Interrupt Priority High registers (AIPH0) and one in the Additional Interrupt Priority Low registers (AIPH0). This provides each additional interrupt sources four possible priority level selection bits
- The flags that generate these interrupts are bits AIFj in Special Function Register AIF. When an external interrupt is generated, then **the interrupt must be de-asserted before the end of the ISR**.
- Each of the additional external interrupt sources can be individually enabled or disabled by setting or clearing bit AIEj in Special Function Register AIE. The interrupt global disable bit EA in IE register also disables the additional interrupts.

✋: *The additional external interrupt inputs intextra_n[6:0] does* ***not*** *provide the capability to exit from Power-down mode.*

[22] *Like int0_n and int1_n inputs, intextra_n[6:0] inputs are synchronized once on clock rising edge before internal use.*

AIF (S:C0h)

Additional Interrupt Flag Register

BIT	7	6	5	4	3	2	1	0
FIELD	-	AIF6	AIF5	AIF4	AIF3	AIF2	AIF1	AIF0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	AIF6	**Additional interrupt 6 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
5	AIF5	**Additional interrupt 5 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
4	AIF4	**Additional interrupt 4 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
3	AIF3	**Additional interrupt 3 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
2	AIF2	**Additional interrupt 2 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
1	AIF1	**Additional interrupt 1 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
0	AIF0	**Additional interrupt 0 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled

Figure 72: Additional interrupt flag register (AIF)

7.2.2 Timer Interrupts

Two timer-interrupt request bits (TF0 and TF1 in TCON register) are set by timer overflow (except Timer 0 in Mode 3). When a timer interrupt is generated, the bit is cleared by a hardware jump to an interrupt service routine. Timer interrupts are enabled by bits ET0, ET1, and ET2 in the IE0 register.

Timer 2 interrupts are generated by a logical OR of bits TF2 and EXF2 in register T2CON. Neither flag is cleared by a hardware jump to a service routine. In fact, the interrupt service routine must determine if TF2 or EXF2 generated the interrupt, and then clear the bit. Timer 2 interrupt is enabled by ET2 in register IE0.

7.2.3 Programmable Counter Array (PCA) Interrupt

The programmable counter array (PCA) interrupt is generated by the logical OR of five event flags (CCF*x*) and the PCA timer overflow flag (CF) in the CCON register. All PCA interrupts share a common interrupt vector. Bits are not cleared by hardware vectors to service routines. Normally, interrupt service routines resolve interrupt requests and clear flag bits. This allows the user to define the relative priorities of the five PCA interrupts. The PCA interrupt is enabled by bit EC in the IE0 register. In addition, the CF flag and each of the CCF*x* flags must also be individually enabled by bits ECF and ECCF*x* in registers CMOD and CCAPM*x* respectively for the flag to generate an interrupt.

✋: *CCFx refers to 5 separate bits, one for each PCA module (CCF0, CCF1, CCF2, CCF3, CCF4). CCAPMx refers to 5 separate registers, one for each PCA module (CCAPM0, CCAPM1, CCAPM2, CCAPM3, CCAPM4).*

7.2.4 Serial Port Interrupt

Serial port interrupts are generated by the logical OR of bits RI and TI in the SCON register. No flag is cleared by a hardware jump to the interrupt service routine. The service routine resolves RI or TI interrupt generation and clears the serial port request flag. The serial port interrupt is enabled by bit ES in the IE0 register.

7.2.5 TRAP interrupt

The function of TRAP instruction is like a software breakpoint, which is useful in software debug. The coding of this instruction is [A5h] [B9h] in binary mode and [B9h] in source mode. By execution of the TRAP instruction, the Flip80251 generates an interrupt and executes the interrupt service routine at address FF:007Bh. It acts like the highest priority non-interruptible interrupt.

7.3 Interrupt Enable

Each interrupt source (with the exception of TRAP) may be individually enabled or disabled by the appropriate interrupt enable bit in the IE0 register at S:0A8h (or in the AIE register at S:0E8h for additional interrupt sources). Note IE0 also contains a global disable bit (EA) that applies to all interrupts (except TRAP and NMI that are not maskable).

If EA is set, interrupts are individually enabled or disabled by bits in IE0. If EA is clear, all interrupts are disabled.

IE0 (S:A8h)

Interrupt Enable Register

BIT	7	6	5	4	3	2	1	0
FIELD	EA	EC	ET2	ES	ET1	EX1	ET0	EX0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	EA	**Global Interrupt Enable** Clear to disable all interrupts, except the TRAP and NMI interrupts, which are always enabled. Set to enable all interrupts that are individually enabled in IE0.
6	EC	**PCA Interrupt Enable** Set to enable PCA Interrupt. Cleared to disable PCA Interrupt
5	ET2	**Timer2 Interrupt Enable** Set to enable Timer2 Interrupt. Cleared to disable Timer2 Interrupt
4	ES	**Serial Port Interrupt Enable** Set to enable Serial Port Interrupt. Cleared to disable Serial Port Interrupt
3	ET1	**Timer1 Interrupt Enable** Set to enable Timer1 Interrupt. Cleared to disable Timer1 Interrupt
2	EX1	**External Interrupt 1 enable** Set to enable External Interrupt 1. Cleared to disable External Interrupt 1.
1	ET0	**Timer0 Interrupt Enable** Set to enable Timer0 Interrupt. Cleared to disable Timer0 Interrupt
0	EX0	**External Interrupt 0 enable** Set to enable External Interrupt 0. Cleared to disable External Interrupt 0.

Figure 73: Interrupt Enable register 0 (IE0)

AIE (S:E8h)

Additional Interrupt Enable Register

BIT	7	6	5	4	3	2	1	0
FIELD	-	AIE6	AIE5	AIE4	AIE3	AIE2	AIE1	AIE0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	AIE6	**Additional interrupt 6 Enable** Set to enable Additional Interrupt 6. Clear to disable Additional Interrupt 6
5	AIE5	**Additional interrupt 5 Enable** Set to enable Additional Interrupt 5. Clear to disable Additional Interrupt 5
4	AIE4	**Additional interrupt 4 Enable** Set to enable Additional Interrupt 4. Clear to disable Additional Interrupt 4
3	AIE3	**Additional interrupt 3 Enable** Set to enable Additional Interrupt 3. Clear to disable Additional Interrupt 3
2	AIE2	**Additional interrupt 2 Enable** Set to enable Additional Interrupt 2. Clear to disable Additional Interrupt 2
1	AIE1	**Additional interrupt 1 Enable** Set to enable Additional Interrupt 1. Clear to disable Additional Interrupt 1
0	AIE0	**Additional interrupt 0 Enable** Set to enable Additional Interrupt 0. Clear to disable Additional Interrupt 0

Figure 74: Additional Interrupt Enable register (AIE)

7.4 Interrupt priority

Each of the hardware interrupt sources may be individually programmed to one of four priority levels (except the NMI input, which has an higher priority). This is accomplished by one bit in the Interrupt Priority HIGH registers (IPH0 or AIPH) and one in the Interrupt Priority Low registers (IPL0 or AIPL). This provides each interrupt source four possible priority level selection bits

IPL0.x AIPL.x	IPH0.x AIPH.x	Priority level
0	0	0 (lowest)
0	1	1
1	0	2
1	1	3 (highest)

Table 89: Priority level

The TRAP instruction is the highest priority interrupt. A TRAP cannot be interrupted by any other interrupt source including the TRAP.

A low-priority interrupt can be itself interrupted by a higher priority interrupt, but not by another lower or equal priority interrupts. Higher priority interrupts are serviced before lower priority interrupts.

If two requests of the same priority level are received simultaneously, an internal polling sequence determines which request is serviced, according to the Table 90.

Interrupt source	**Interrupt flag**	**Priority number**	**Vector Address**	**cleared by hardware (H) or by software (S)**
TRAP	-	1 (highest - not interruptible)	FF:007Bh	-
NMI	-	2	FF:003Bh	-
Interrupt 0	IE0	3	FF:0003h	H if edge
Timer 0	TF0	4	FF:000Bh	H
Interrupt 1	IE1	5	FF:0013h	H if edge
Timer 1	TF1	6	FF:001Bh	H
UART	RI+TI	7	FF:0023h	S
Timer2	TF2+EXF2	8	FF:002Bh	S
PCA	CF	9	FF:0033h	S
Intextra_n 0	AIF0	10	FF:0043h	S
Intextra_n 1	AIF1	11	FF:004Bh	S
Intextra_n 2	AIF2	12	FF:0053h	S
Intextra_n 3	AIF3	13	FF:005Bh	S
Intextra_n 4	AIF4	14	FF:0063h	S
Intextra_n 5	AIF5	15	FF:006Bh	S
Intextra_n 6	AIF6	16	FF:0073h	S

Table 90: Interrupt priority within a same priority level

IPH0 (S:B7h)

Interrupt Priority High Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	IPHC	IPHT2	IPHS	IPHT1	IPHX1	IPHT0	IPHX0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved.** The value read from this bit is indeterminate
6	IPHC	**PCA Interrupt Priority level most significant bit** IPHC IPLC Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
5	IPHT2	**Timer2 Interrupt Priority level most significant bit** IPHT2 IPLT2 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
4	IPHS	**Serial Port Interrupt Priority level most significant bit** IPHT2 IPLT2 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
3	IPHT1	**Timer1 Interrupt Priority level most significant bit** IPHT1 IPLT1 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
2	IPHX1	**External interrupt 1 Priority level most significant bit** IPHX1 IPLX1 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
1	IPHT0	**Timer0 Interrupt Priority level most significant bit** IPHT0 IPLT0 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
0	IPHX0	**External interrupt 0 Priority level most significant bit** IPHX0 IPLX0 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority

Figure 75: Interrupt Priority High register 0 (IPH0)

IPL0 (S:B8h)

Interrupt Priority Low Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	IPLC	IPLT2	IPLS	IPLT1	IPLX1	IPLT0	IPLX0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	IPLC	**PCA Interrupt Priority level less significant bit** Refer to IPH0 for priority level description
5	IPLT2	**Timer2 Interrupt Priority level less significant bit** Refer to IPH0 for priority level description
4	IPLS	**Serial Port Interrupt Priority level less significant bit** Refer to IPH0 for priority level description
3	IPLT1	**Timer1 Interrupt Priority level less significant bit** Refer to IPH0 for priority level description
2	IPLX1	**External interrupt 1 Priority level less significant bit** Refer to IPH0 for priority level description
1	IPLT0	**Timer0 Interrupt Priority level less significant bit** Refer to IPH0 for priority level description
0	IPLX0	**External interrupt 0 Priority level less significant bit** Refer to IPH0 for priority level description

Figure 76: Interrupt Priority Low register 0 (IPL0)

AIPH (S:F7h)

Additional Interrupt Priority High Register

BIT	7	6	5	4	3	2	1	0
FIELD	-	AIPH6	AIPH5	AIPH4	AIPH3	AIPH2	AIPH1	AIPH0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	AIPH6	**Additional interrupt 6 Priority Most significant bi**
5	AIPH5	**Additional interrupt 5 Priority Most significant bit**
4	AIPH4	**Additional interrupt 4 Priority Most significant bit**
3	AIPH3	**Additional interrupt 3 Priority Most significant bit**
2	AIPH2	**Additional interrupt 2 Priority Most significant bit**
1	AIPH1	**Additional interrupt 1 Priority Most significant bit**
0	AIPH0	**Additional interrupt 0 Priority Most significant bit**

Figure 77: Additional Interrupt Priority High register (AIPH)

AIPL (S:F8h)

Additional Interrupt Priority Low Register

BIT	7	6	5	4	3	2	1	0
FIELD	-	AIPL6	AIPL5	AIPL4	AIPL3	AIPL2	AIPL1	AIPL0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	AIPL6	**Additional interrupt 6 Priority Less significant bi**
5	AIPL5	**Additional interrupt 5 Priority Less significant bit**
4	AIPL4	**Additional interrupt 4 Priority Less significant bit**
3	AIPL3	**Additional interrupt 3 Priority Less significant bit**
2	AIPL2	**Additional interrupt 2 Priority Less significant bit**
1	AIPL1	**Additional interrupt 1 Priority Less significant bit**
0	AIPL0	**Additional interrupt 0 Priority Less significant bit**

Figure 78: Additional Interrupt Priority Low register (AIPL)

7.5 Interrupt processing

The processor acknowledges an interrupt request by executing a hardware-generated LCALL to the appropriate servicing routine.

- When *intrmode* pin is low (80C51 compatible interrupt mode configuration), he hardware-generated LCALL pushes the two lowest bytes of the Program Counter onto the stack and reloads the PC with an address that depends on the source of the interrupt being vectored.
- When *intrmode* pin is high (enable to handle interrupt with code executing outside page FF:), the hardware-generated LCALL pushes the three bytes of the Program Counter and the PSW1 register onto the stack and reloads the PC with an address that depends on the source of the interrupt being vectored.

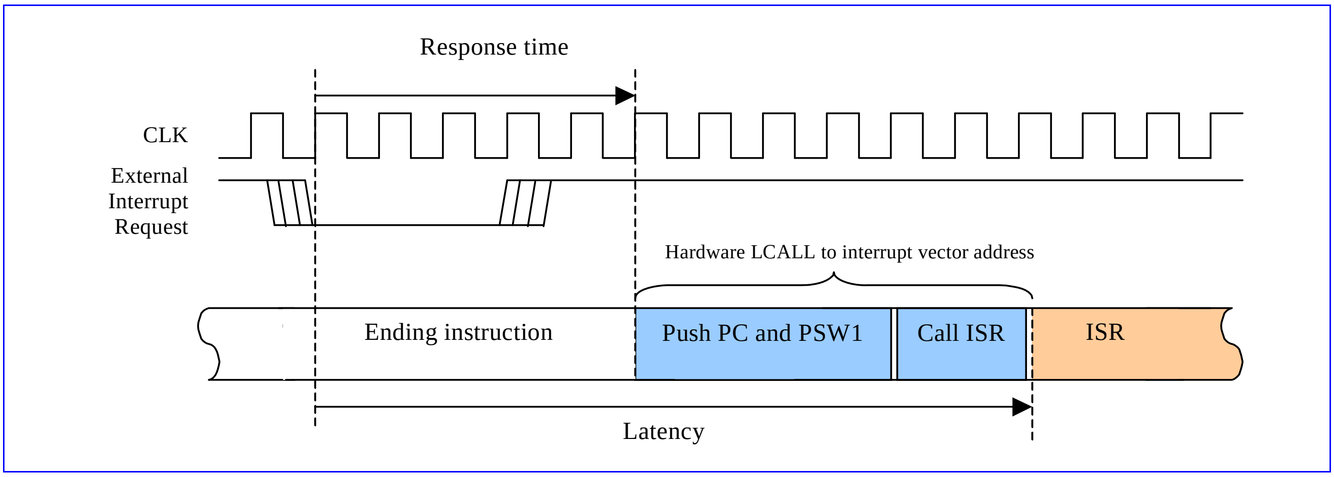

Response time
CLK
External Interrupt Request
Hardware LCALL to interrupt vector address
Ending instruction
Push PC and PSW1
Call ISR
ISR
Latency

Figure 79: Interrupt process[23]

Interrupt processing is a dynamic operation that begins when a source requests an interrupt and lasts until the execution of the first instruction in the interrupt service routine (see Figure 79). *Response time* is the amount of time between the interrupt request and the resulting break in the current instruction stream. *Latency* is the amount of time between the interrupt request and the execution of the first instruction in the interrupt service routine.

These periods are dynamic due to the presence of both fixed-time sequences and several variable conditions. These conditions contribute to total elapsed time. Both response time and latency begin with the request. The subsequent minimum fixed sequence comprises the interrupt sample, poll, and request operations. The variables consist of (but are not limited to): specific instructions in use at request time, internal versus external interrupt source requests, stack location, presence of wait states, and branch pointer length.

Note that if an interrupt of higher priority level goes active prior to the fourth cycle of the hardware LCALL to interrupt vector address (refer to the Figure 79) then in accordance with the rules of interrupt handling, it will be vectored to during the last two cycles of the “call ISR” without any instruction of the lower priority routine having been executed.

[23] This figure corresponds to a configuration where *intrmode* is set to 1.

7.6 Interrupt blocking conditions

If all enable and priority requirements have been met, a single prioritized interrupt request at a time generates a vector cycle to an interrupt service routine. There are three causes of blocking conditions with hardware-generated vectors:

1. An interrupt of equal or higher priority level is already in progress (defined as any point after the flag has been set and the RETI of the ISR has not executed).
2. The current polling cycle is not the final cycle of the instruction in progress.
3. The instruction in progress is RETI or any write to the IE0, IPH0, IPL0, AIE, AIPH or AIPL registers.

Any of these conditions blocks calls to interrupt service routines. Condition 2 ensures the instruction in progress completes before the system vectors to the ISR. Condition 3 ensures at least one more instruction executes before the system vectors to interrupts if the instruction in progress is a RETI or any write to an interrupt control registers.

✋: *If the interrupt flag for a level-triggered external interrupt is set but denied for one of the above conditions and is clear when the blocking condition is removed, then the denied interrupt is ignored. In other words, blocked interrupt requests are not buffered for retention.*

8. POWER SAVING MODES

For applications where power consumption is critical, the original 8xC251 provides two power saving modes: **IDLE** and **POWERDOWN**. These modes work the same way on the Flip80251. The **IDLE** and **POWERDOWN** modes are power reduction modes for use in applications where power consumption is a concern. User instructions activate these modes by setting bits in the PCON register. Program execution halts, but resumes when the mode is exited by an interrupt.

8.1 Power Control Register

The PCON special function register (Figure 80) provides two control bits for the serial I/O function, two bits for selecting the idle and power-down modes, the power off flag, and two general purpose flags.

PCON (S:87h)
Power Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	SMOD1	SMOD0	--	POF	GF1	GF0	PD	IDL
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	SMOD1	**Double baud rate** Set to double the baud rate when timer 1 is used and serial mode 0 is not selected (see Serial port chapter)
6	SMOD0	**Select function of SCON.7** Set to access to SCON.7 as the FE bit Clear to access to SCON.7 as the SM0 bit (see Serial port chapter)
5	--	**Reserved** The value read from this bit is indeterminate
4	POF	**Power Off flag** Set by hardware when the input "*poweroff*" is high It can be set or cleared by software.
3	GF1	**General purpose flag 1** Set or cleared by software
2	GF0	**General purpose flag 0** Set or cleared by software
1	PD	**Power-down mode bit** Set to activate power-down mode Clear by hardware when an enabled external interrupt or a reset occurs.
0	IDL	**Idle mode bit** Set to activates idle mode Clear by hardware when an enabled interrupt or a reset occurs.

In standard 80251, Power Off flag is set by hardware as Vcc rises above TBD voltage to indicate that power has been off or Vcc had fallen below a TBD voltage and that on-chip volatile memory is indeterminate.

Figure 80: Power control register (PCON)

8.1.1 Serial I/O Control Bits

The SMOD1 bit in the PCON register is a factor in determining the serial I/O baud rate.
The SMOD0 bit in the PCON register determines whether bit 7 of the SCON register provides read/write access to the framing error (FE) bit (SMOD0 = 1) or to SM0, a serial I/O mode select bit (SMOD0 = 0).

8.1.2 Power Off Flag

Hardware sets the Power off Flag (POF) in PCON when VCC rises from user defined Voltage to a second user defined Voltage to indicate that on-chip volatile memory is indeterminate (e.g. at power-on). This is performed at system level by asserting the *poweroff* input of the Flip80251.
The POF can be set or cleared by software. After a reset, check the status of this bit to determine whether a cold start reset or a warm start reset occurred. After a cold start, user software should clear the POF. If POF = 1 is detected at other times, do a reset to reinitialize the chip, since for VCC is lower than used defined Voltage data may have been lost or some logic may have malfunctioned.

8.2 Internal clock gating module

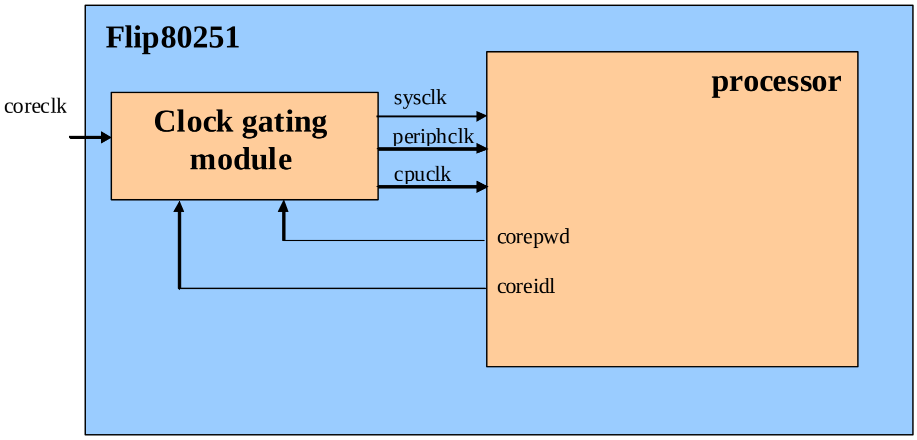


Figure 81: Internal clock management

The Flip80251-Hurricane includes an internal clock gating module in order to support the two power saving modes (idle mode and power-down mode) defined by the 8xC251 architecture.

This module provides two gated clocks:

- *cpuclk*: processor clock. This clock is off during in idle mode and power-down mode
- *periphclk*: peripheral and Interrupt controller clock. This clock is off during in power down mode

sysclk is the image of *coreclk*. It is not gated and connected only to the register that enables a synchronous wake up from power down mode

8.3 Idle mode

The Idle mode reduces the core power consumption, whilst still allowing the Flip80251 to be woken up via interrupts. With one timer running (to bring the Flip80251 out of idle mode), the power is reduced to about 15 to 20 % of the typical running power consumption.

In this mode, program execution halts. Idle mode freezes the clock to the CPU at known states while the peripherals continue to be clocked (by *periphclk*). The CPU status before entering idle mode is preserved; i.e., the program counter, program status word register, and register file retain their data for the duration of idle mode. The contents of the SFRs and RAM are also retained. Whilst in idle mode, the output ports retain their values. The memory control signals (CODE and DATA) are put to their inactive value (e.g. *prgcs_n* is pulled high)

✋: *To further reduce power consumption in this mode, the software can switch off timers, which aren't being used, and the Serial Port if this is not required. Moreover, the PCA may be instructed to pause during idle mode by setting the CIDL bit in the CMOD register.*

8.3.1 Entering idle mode

To enter idle mode, set the PCON register IDL bit. The Flip80251 enters idle mode upon execution of the instruction that sets the IDL bit. The instruction that sets the IDL bit is the last instruction executed.

✋: If the IDL bit and the PD bit are set simultaneously, the Flip80251 enters power-down mode.

8.3.2 Exiting idle mode

There are two ways to exit idle mode:

- Generate an enabled interrupt. Hardware clears the PCON register IDL bit that restores the clocks to the CPU. Execution resumes with the interrupt service routine. Upon completion of the interrupt service routine, program execution resumes with the instruction immediately following the instruction that activated idle mode. The general-purpose flags (GF1 and GF0 in the PCON register) may be used to indicate whether an interrupt occurred during normal operation or during idle mode. When idle mode is exited by an interrupt, the interrupt service routine may examine GF1 and GF0.
- Reset the chip. A logical high on the *corerst* pin clears the IDL bit in PCON register directly and asynchronously. This restores the clocks to the CPU. Program execution momentarily resumes with the instruction immediately following the instruction that activated the idle mode and may continue for a number of clock cycles before the internal reset algorithm takes control. Reset initializes the Flip80251 and vectors the CPU to address FF:0000h.

8.4 Power-down mode

The power-down mode places the Flip80251 in a very low power state. Power-down mode freezes processor and peripheral clocks at known states. The CPU status prior to entering power-down mode is preserved, i.e., the program counter, program status word register, and register file retain their data for the duration of power-down mode. In addition, the SFRs and RAM contents are preserved. Whilst in power-down mode, the output ports retain their values. The memory control signals (CODE and DATA) are put to their inactive value (e.g. *prgcs_n* is pulled high)
The *corepwd* signal is high when the power-down mode has been entered. This signal is used by the internal clock gating module to gate the processor clock and the peripheral clock signals, to further reduce power consumption (over that of idle mode).
However, the *coreclk* clock input is not gated by the *corepwd* signal to provide a synchronous exit from power down mode.

✍*: When connecting asynchronous off-chip inputs to the Flip80251 it is important to avoid any meta-stability problems. A meta-stable pulse is caused by data changing during the set-up and hold time of the data sampling by the flip-flop. The designer is responsible for preventing these problems, as they are dependant on the nature of the signals input to the Flip80251.*

8.4.1 Entering power-down mode

To enter power-down mode, set the PCON register PD bit. The Flip80251 enters the power-down mode upon execution of the instruction that sets the PD bit. The instruction that sets the PD bit is the last instruction executed.

8.4.2 Exiting power-down mode

There are two ways to exit the power-down mode:

- Generate an enabled external interrupt (*int0_n*/*int1_n*)[24] or an NMI interrupt (*intnmi*). Hardware clears the PD bit in the PCON register that starts the oscillator and restores the clocks to the CPU and peripherals. Execution resumes with the interrupt service routine. Upon completion of the interrupt service routine, program execution resumes with the instruction immediately following the instruction that activated power-down mode.[25]
- Generate a reset. A logical high on the *corerst* pin clears the PD bit in the PCON register directly and asynchronously. Reset initializes the Flip80251-Hurricane and vectors the CPU to address FF:0000h.

[24] In case of a configuration including the additional interrupts (intextra_n[6:0]), those interrupt sources do not enable to wake up from power-down mode.

[25] Note: To enable an external interrupt, set the IE register EX0 and/or EX1 bit[s]. The external interrupt used to exit power-down mode must be configured as level sensitive and must be assigned the highest priority. In addition, the duration of the interrupt must be of sufficient length to allow the oscillator to stabilize

9. MEMORY INTERFACING

9.1 Special Function Register (SFR space)

A designer using the Flip80251 does not need to provide any extra memory for the internal SFRs as they are designed internally to the Flip80251. Both core and peripherals registers are implemented using clocked registers (DFF).

It is possible to add external Special Function Registers. These may be used to access peripherals in a more efficient way than slower memory mapping techniques.

Please refer to the Flip80251 User guide chapter 6.3.4 for more information on how to implement extra Special Function Registers

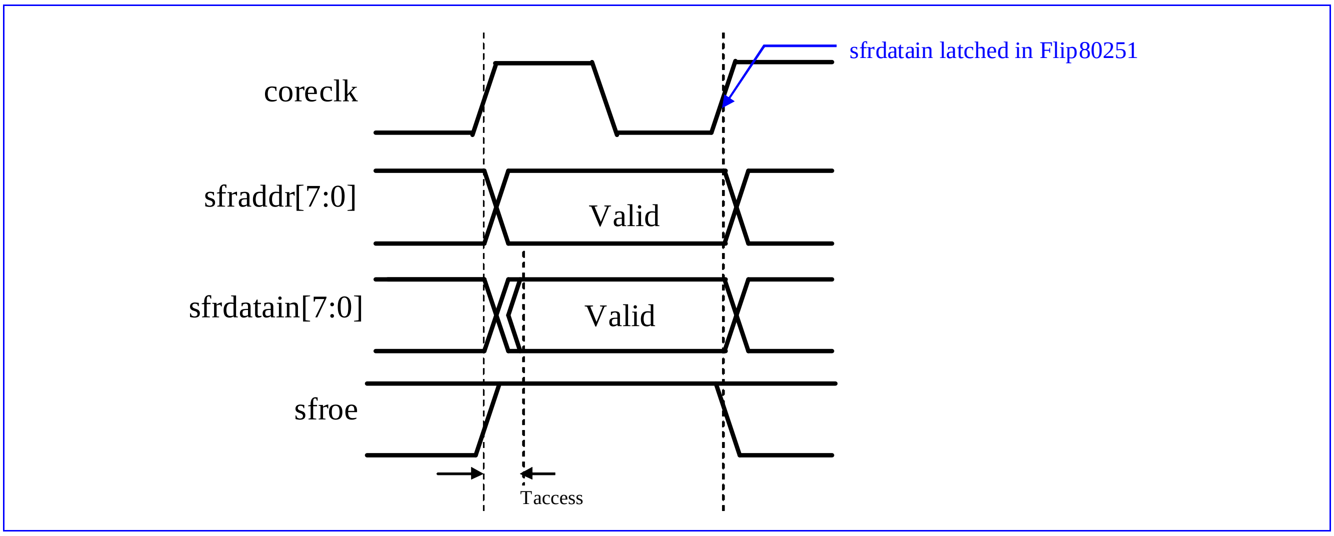


Figure 82: SFR read

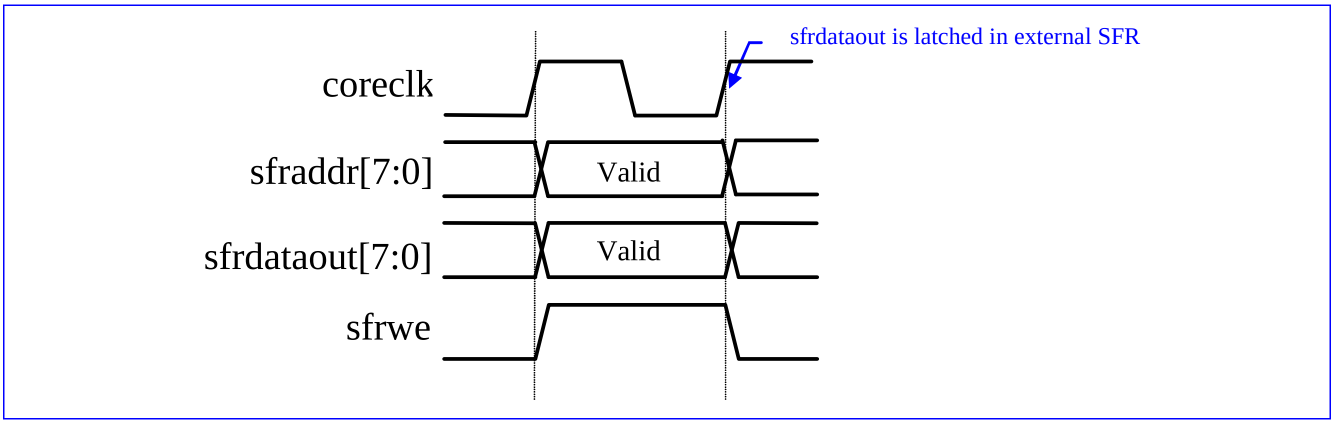


Figure 83: SFR write

9.2 Program Memory and Data Memory

The Flip80251 memory interface was designed to target synchronous single port memory.

9.2.1 Program Memory (CODE space)

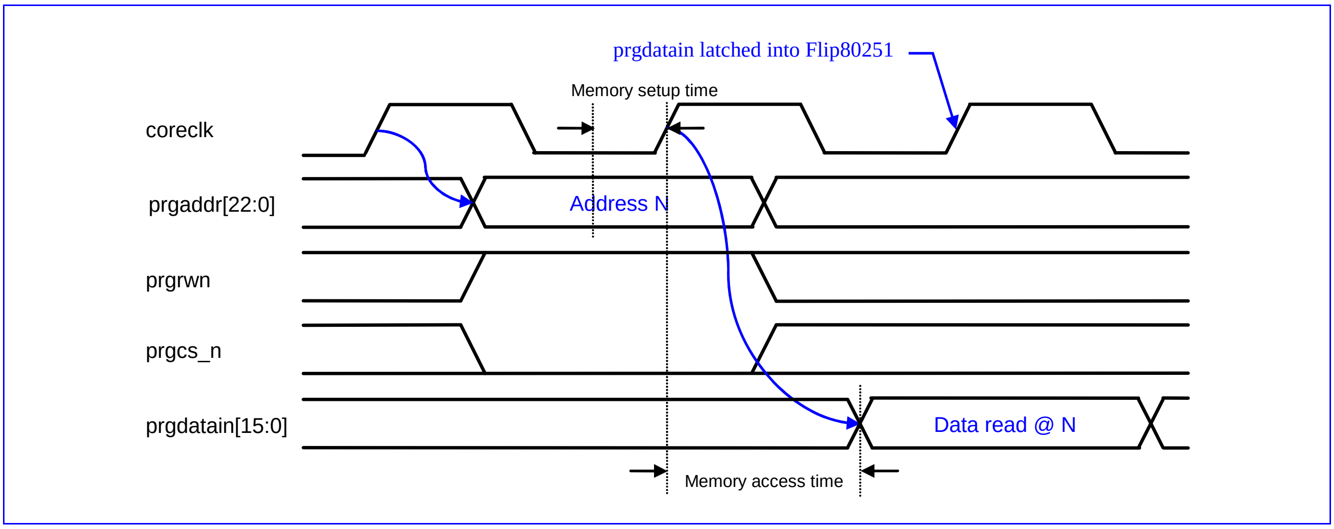


Figure 84: Program memory fetch or read

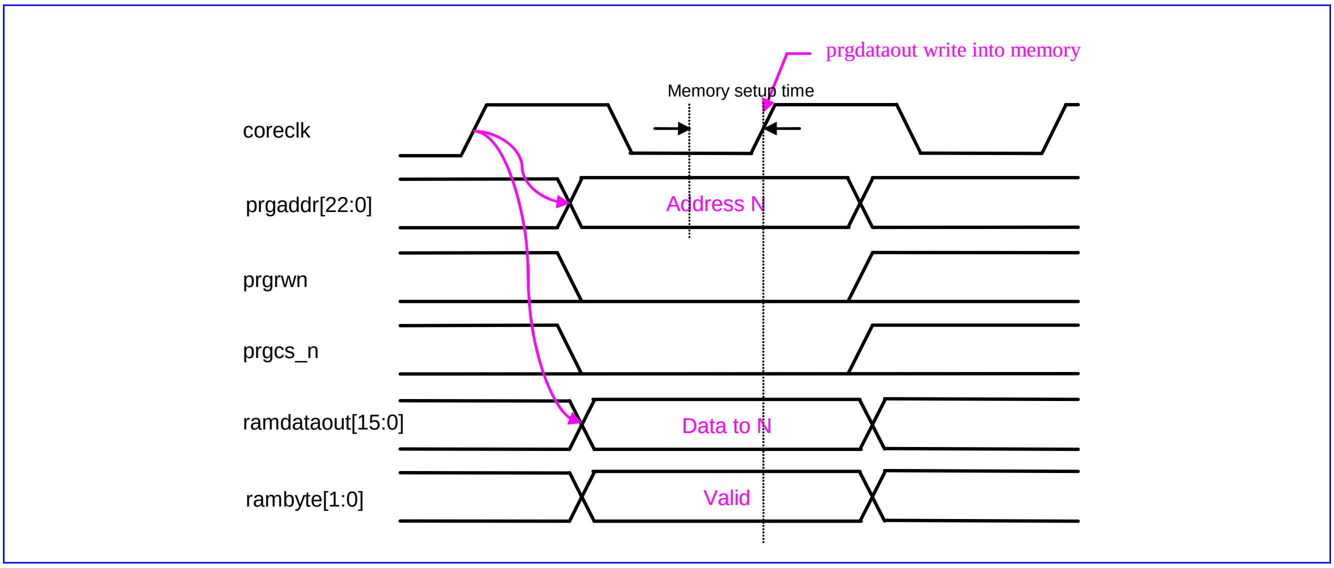


Figure 85: Program memory write

☞: *There is no dedicated bus for writing data into program memory. Then, in case of writable program memory, the input data bus of the program memory should be connected to the* ***ramdataout*** *bus and the byte write control signals to* ***rambyte[1:0].***

9.2.2 Data Memory (DATA space)

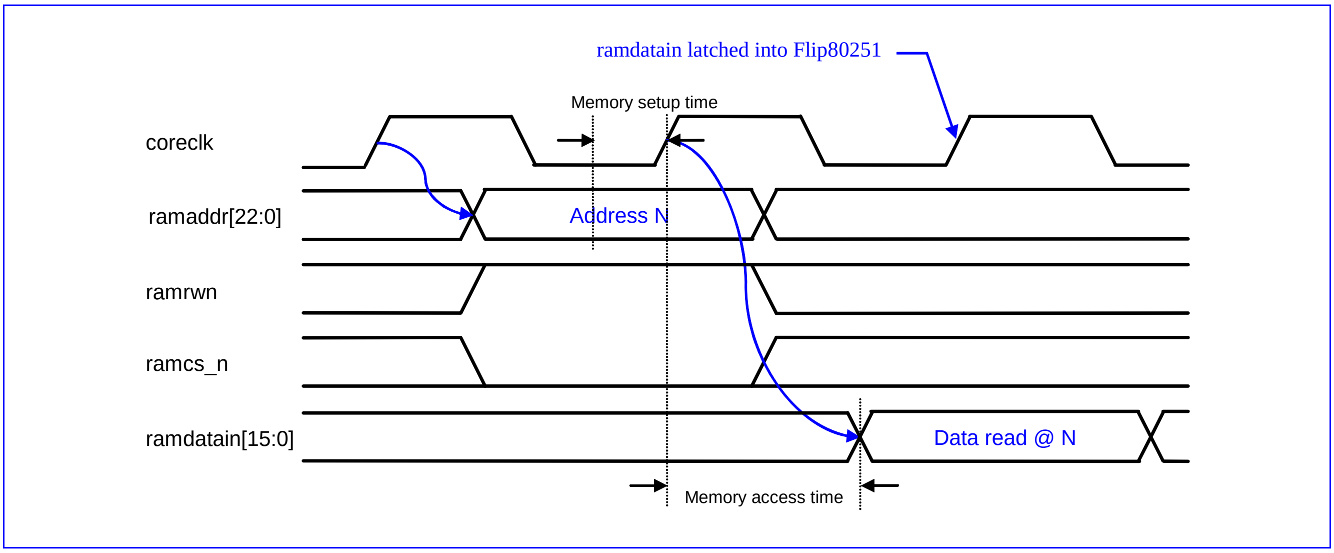


Figure 86: Data Memory read

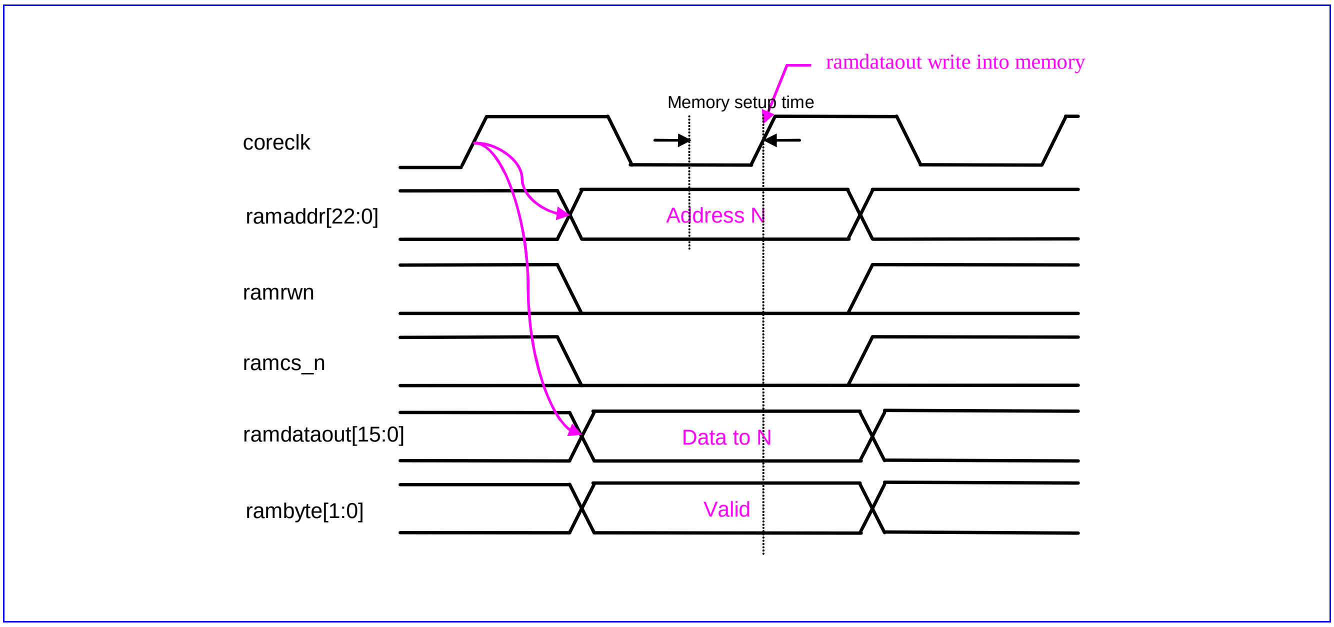


Figure 87: Data Memory Write

9.3 Hardware Controlled Wait States

The Flip80251 provides a hardware-controlled wait state solution for dynamic bus control. The wait states interface enables to use slow peripherals/memories with a fast processor: the maximum frequency of an application is no more limited by the slowest memory. Wait states can be inserted for both program and data memory accesses.

9.3.1 *Wait states control*

This control is achieved through three extra signals, which are used to add wait states during the memory access.

Name	**I/O**	**Function**
prgbusy	I	When high, indicates that the program memory needs a longer read/write/fetch access (than normally done by Flip80251) and then wait states must be inserted. The wait states insertion is stopped when *prgbusy* is set back low.
rambusy	I	When high, indicates that the data memory needs a longer read or write access (than normally done by Flip80251) and then wait states must be inserted. The wait states insertion is stopped when *rambusy* is set back low.
fetchaborted	O	When high, indicates that the current instruction fetch (program memory read access) is aborted. The wait state insertion is no more required for this fetch and *prgbusy* has to be de-asserted.

Table 91: Wait state interface pins

- The *prgbusy* and *rambusy* signals are sampled on the *coreclk* positive edge.

9.3.2 *Wait states for program memory access*

prgbusy input signal enables to lengthen any read, write or fetch access to the program memory.

If *prgbusy* is sampled high the cycle after *prgcs_n* is sampled low, wait cycles are inserted until *prgbusy* is set back low. The *prgaddr* and *ramdataout* (in case of write operation) signals must be latched in the same cycle *prgcs_n* is sampled low.

prgbusy must be set within the following cycle of a program memory access.

In the case of an instruction coded on several bytes, successive read accesses are performed. If wait states are required for accessing the program memory, the *prgbusy* signal has to be set for each read access.

✋: *The NOP instruction is also affected by the wait state feature, thus take care of this delay when using the NOP for delay generation.*

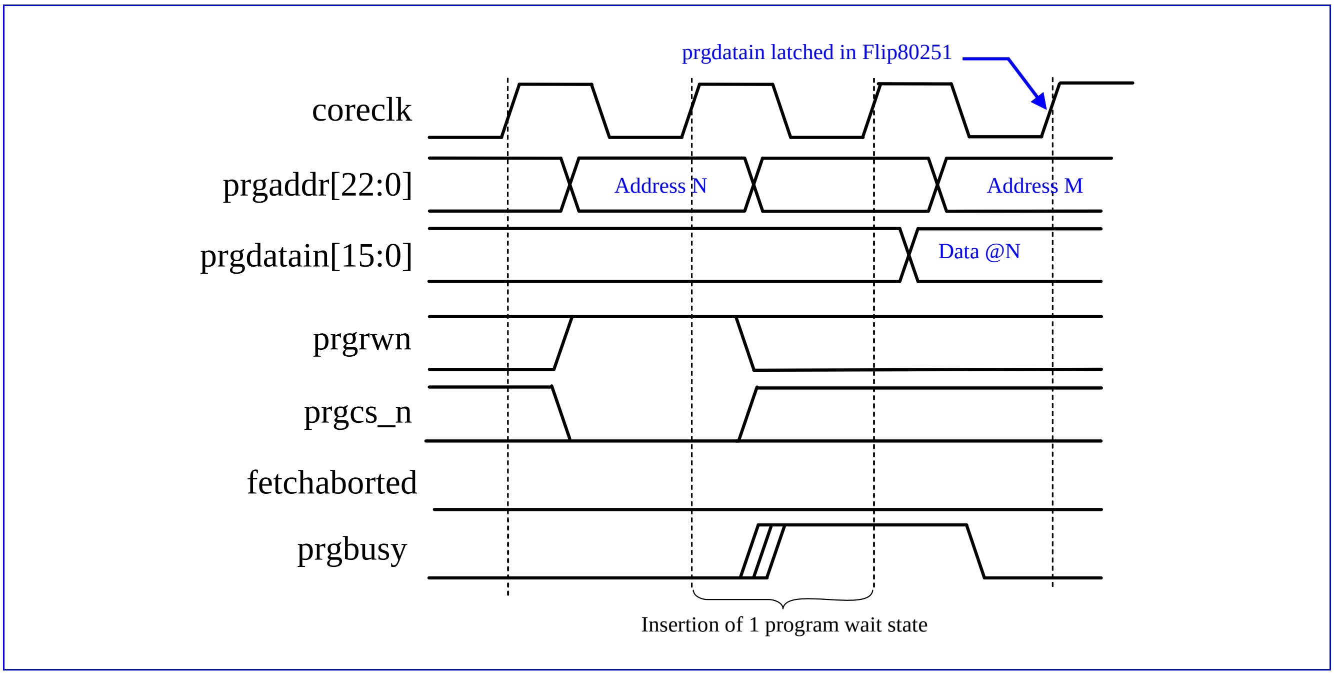


Figure 88: Example of one wait state insertion during a fetch or program read operation

The signal *fetchaborted* is used only for a fetch operation. This signal indicates that the current fetch operation will be aborted. If wait states are required for this access, these wait states insertion must be ended and the signal *prgbsuy* has to be de-asserted.

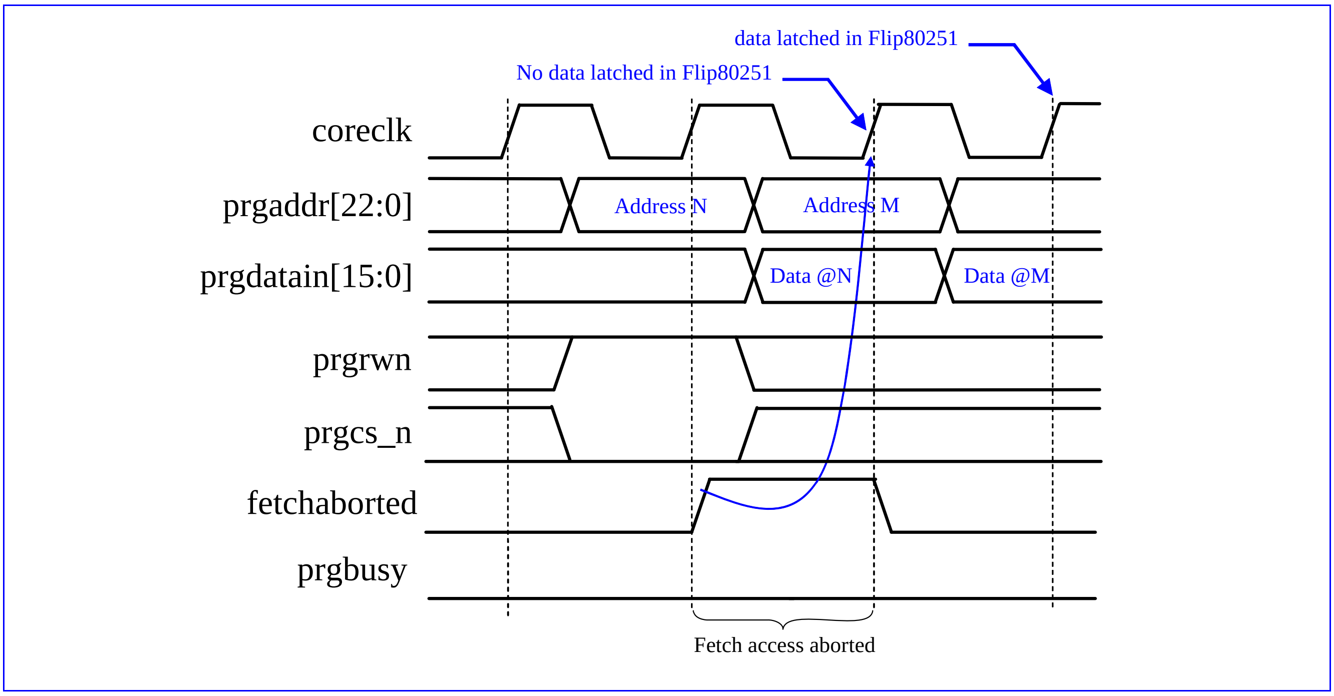


Figure 89: Example of a fetch operation aborted without wait states insertion

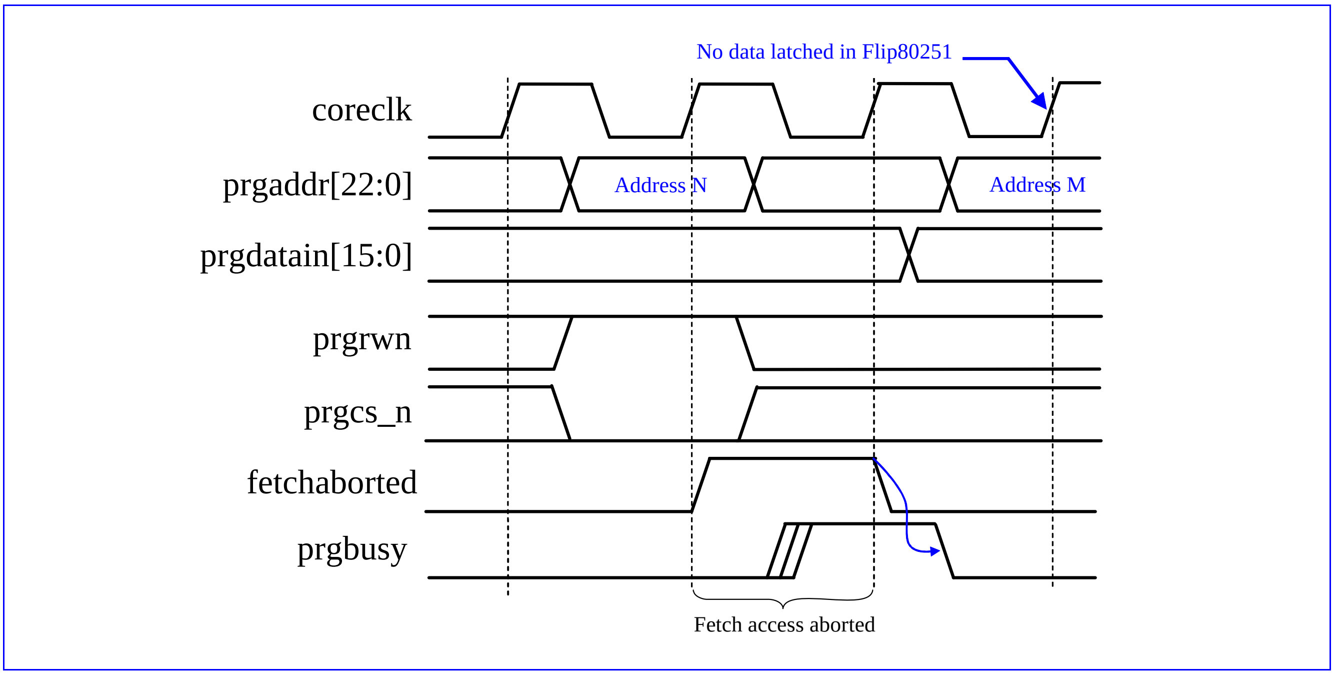


Figure 90: Example of a fetch operation aborted during wait states insertion

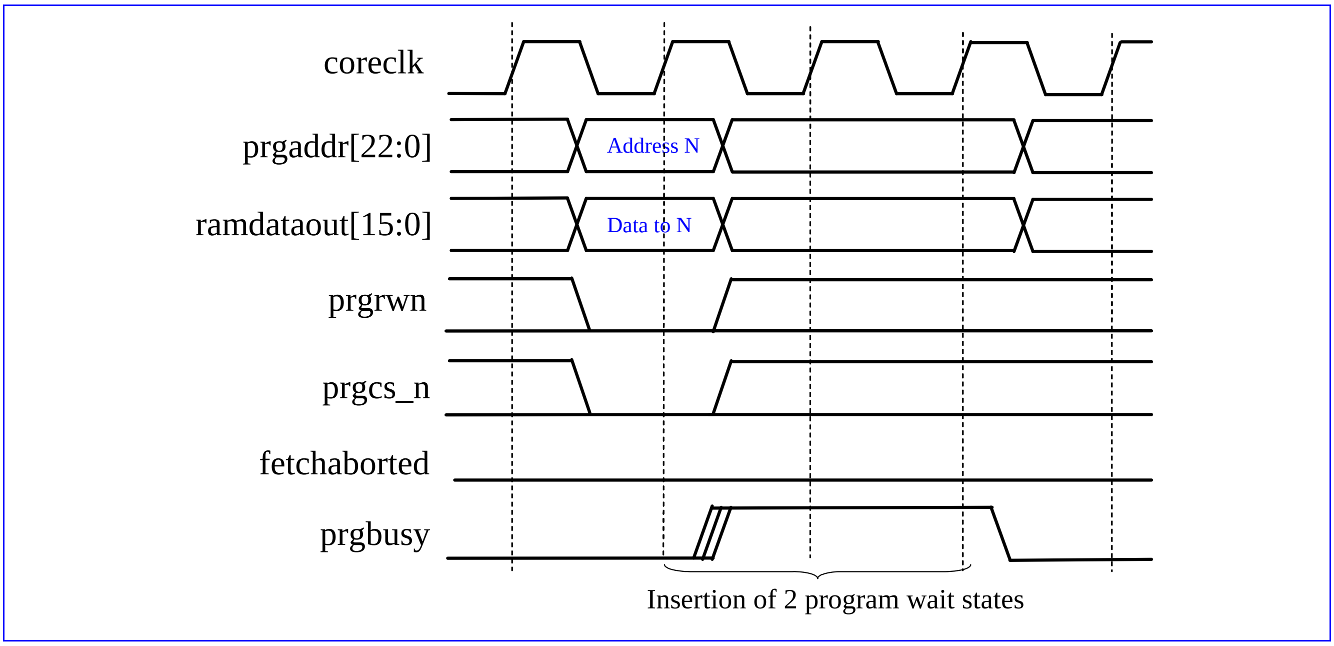


Figure 91: Example of two wait states insertion during program write operation

✋ *IMPORTANT NOTE:*
There is no dedicated bus for writing data into program memory. Then, in case of writable program memory, the input data bus of the program memory should be connected to the ramdataout bus.

✋ *If an enabled interrupt occurs during the wait states, the corresponding subroutine will be started at the end of execution of the current instruction.*

9.3.3 Wait states for data memory access

rambusy input signal enables to lengthen any read or write access to the data memory.

If *rambusy* is sampled high the cycle after *ramcs_n* is sampled low, wait cycles are inserted until *rambusy* is set back low. The *ramaddr* and *ramdataout* (in case of write operation) signals must be latched in the same cycle *ramcs_n* is sampled low.
rambusy must be set within the following cycle of a data memory access.

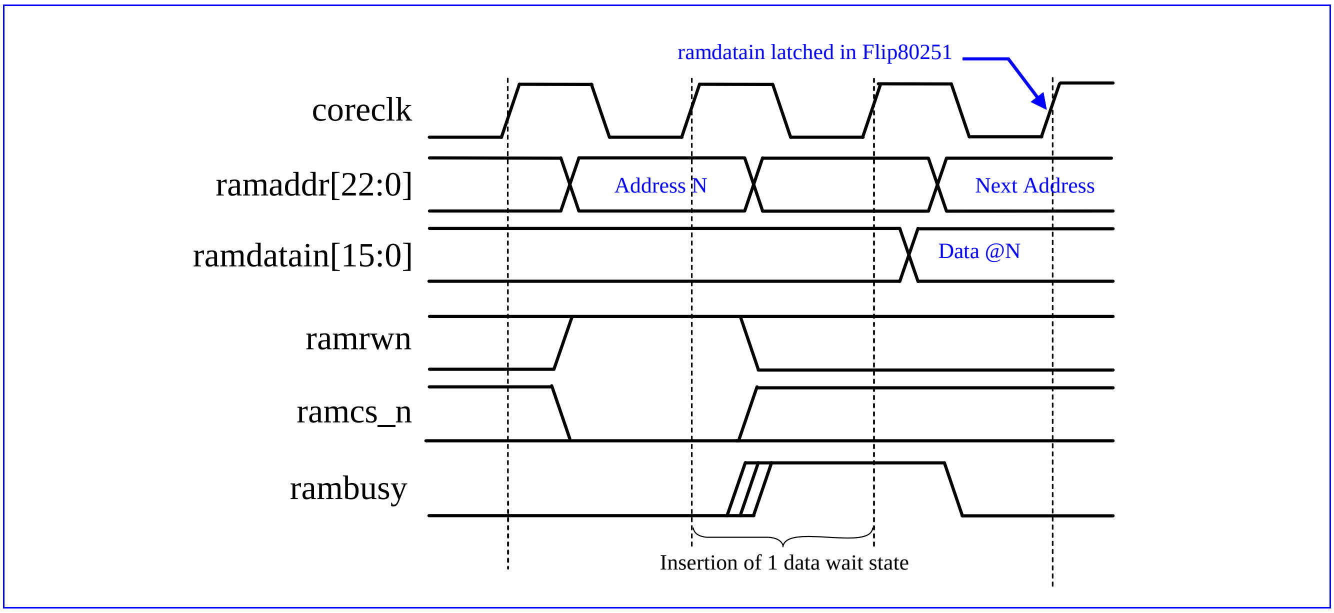


Figure 92: Example of one wait state insertion during data memory read

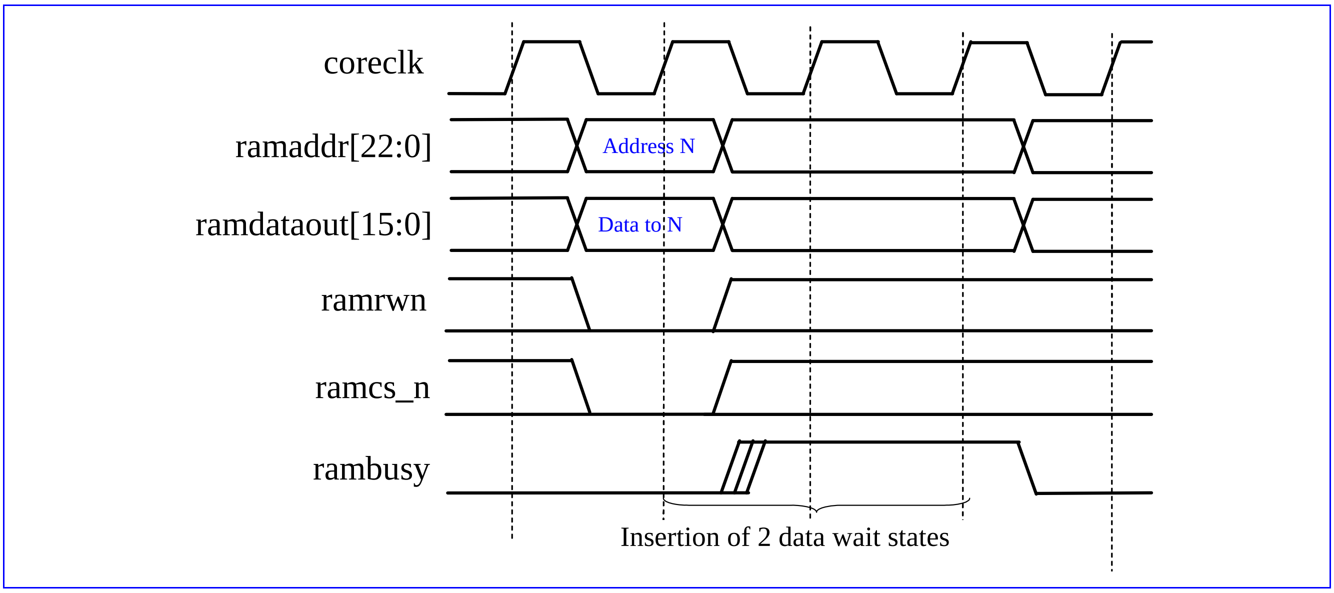


Figure 93: Example of two wait state insertion during data memory write

9.3.4 Wait states for consecutives accesses

The following figures show wait insertion for different successive accesses on program memory.

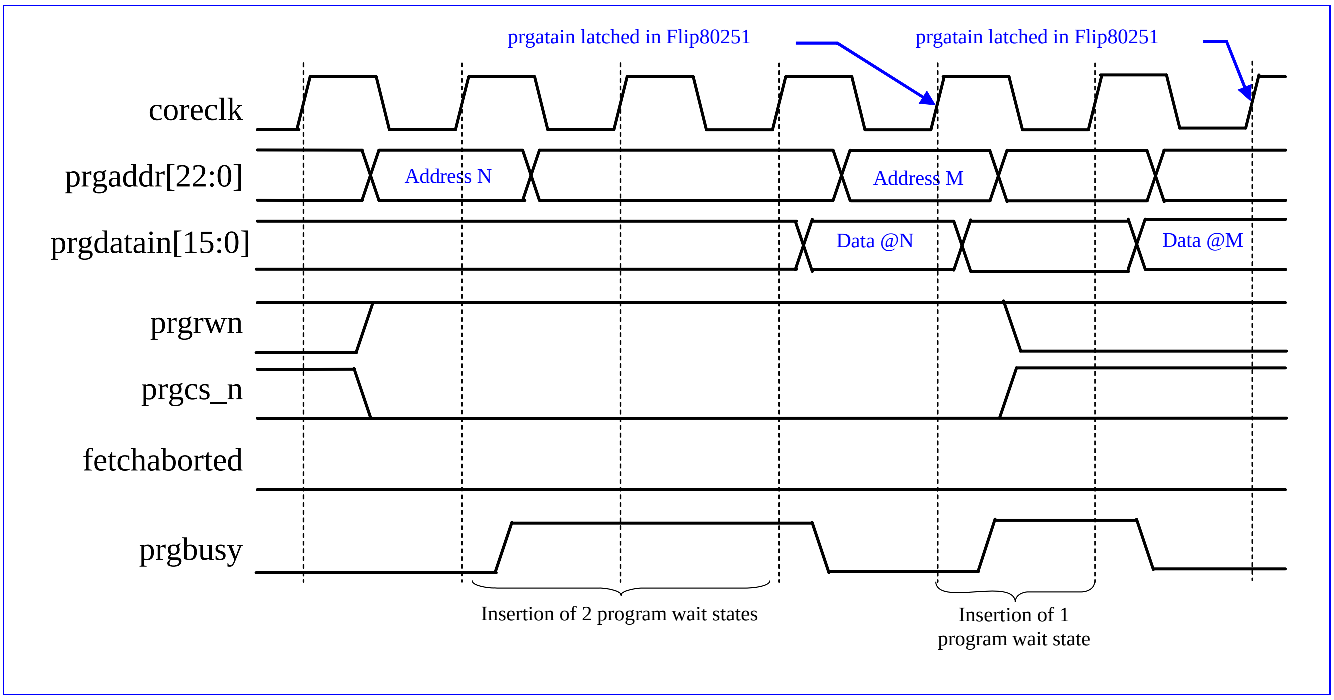

prgatain latched in Flip80251
prgatain latched in Flip80251
coreclk
prgaddr[22:0]
Address N
Address M
prgdatain[15:0]
Data @N
Data @M
prgrwn
prgcs_n
fetchaborted
prgbusy
Insertion of 2 program wait states
Insertion of 1 program wait state

Figure 94: Example of wait state insertion during successive fetch or program read operations

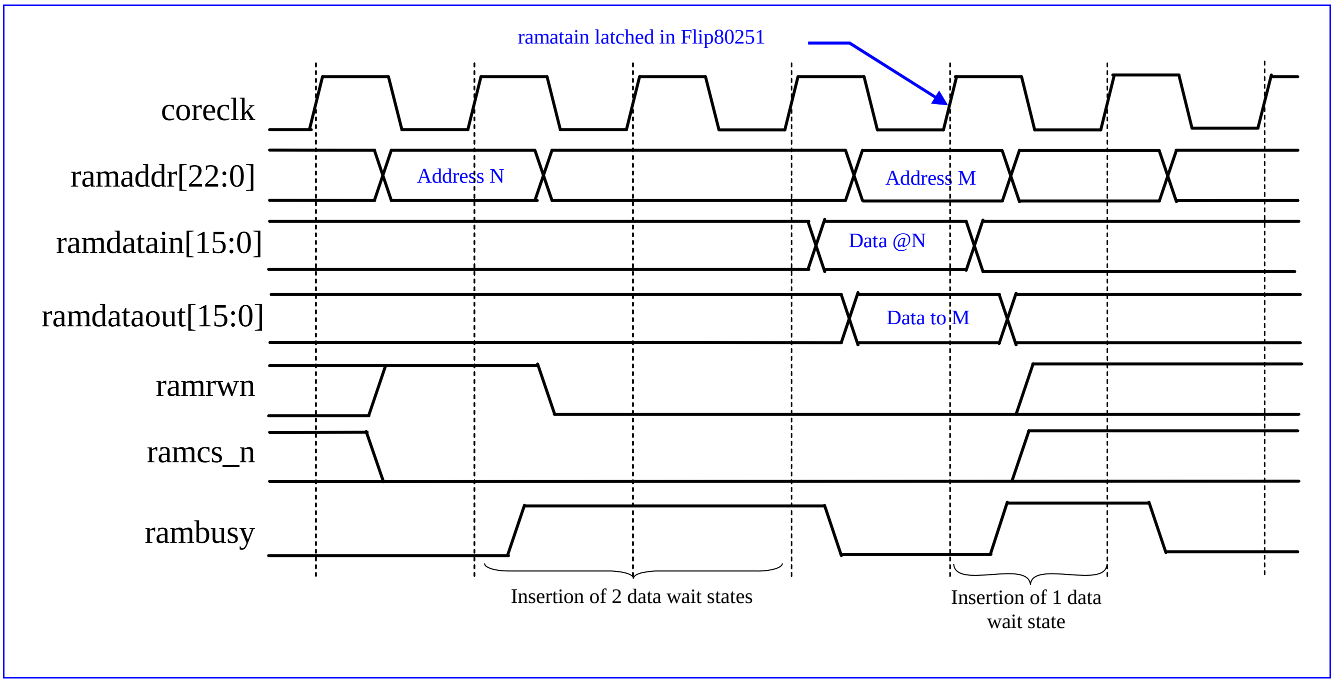


Figure 95: Example of wait state insertion during data read operation followed by a write operation

9.4 Aligned and unaligned memory accesses

Flip80251 Hurricane enables 16 bits aligned and unaligned accesses and 8 bits accesses using big endian convention.
Depending on access type, the following values are available on the data memory interface:

Access	ramaddr/ progaddr	rambyte	ramdataout	ramdataout	Memory Content (byte address)		
type	[22:0]	[1:0]	[15:8]	[7:0]	@2N	@2N+1	@2N+2
16 bits aligned	2N	11	MSB	LSB	MSB	LSB	
16 bits unaligned	2N+1	11	MSB	LSB		LSB	MSB
8 bit (odd address)	2N	10	MSB	x	MSB		
8 bit (even address)	2N	01	x	LSB		LSB	

Table 92: Write Accesses

Access	ramaddr/ progaddr	rambyte	Memory Content (byte address)			ramdatain/ progdatain	ramdatain/ progdatain
Type	[22:0]	[1:0]	@2N	@2N+1	@2N+2	[15:8]	[7:0]
16 bits aligned	2N	11	MSB	LSB		MSB	LSB
16 bits unaligned	2N+1	11		LSB	MSB	MSB	LSB

Table 93: Read Accesses

✍ Note that for 16 bits unaligned accesses, bytes are switched on data bus to facilitate connection to memory (Please refer to User Guide for more information on how to connect memories).

Example:
Write of 16 bits aligned and unaligned words:

```
mov 20h,WR0 with WR0=1234h => rambyte[1:0]=11 ; ramdataout[15:0]=1234h ; @20h=12h and
@21h=34h
mov 21h,WR0 with WR0=1234h => rambyte[1:0]=11 ; ramdataout[15:0]=3412h ; @21h=12h and
@22h=34h
```

Write of 8 bits words:

```
mov 20h, R0 with R0=12h => rambyte[1:0]=10 ; ramdataout[15:0]=12xxh ; @20h=12h and
@21h=unchanged
mov 21h, R0 with R0=34h => rambyte[1:0]=01 ; ramdataout[15:0]=xx34h ; @20h=unchanged and
@21h=34h
```

Read of a 16 bits aligned and unaligned word:

```
mov WR0,20h with @20h=12h and @21h=34h => ramdatain[15:0]=1234h ; WR0=1234h
mov WR0,21h with @21h=12h and @22h=34h => ramdatain[15:0]=3412h ; WR0=1234h
```

9.5 Direct Memory Accesses support

Direct Memory Access support is a feature that enables an external device taking the control of the memory bus. Two extra signals are added to the Flip80251:

Name	I/O	Function
dmareq	I	Request from a peripheral (e.g. DMA controller) to take the control of memory bus. Active high
dmaack	O	Indication that bus control is granted to the requester. Active high.

Table 94: Direct Memory Access support interface pins

When a device needs to take the control of the memory bus, it should assert the *dmareq* signal. When the Flip80251 finishes executing current instruction, it will set to high the memory control signals (*prgcs_n, prgrwn, ramcs_n, ramrwn)* and afterwards assert the *dmaack* signal.

When the requester device detects the high level on *dmaack*, that means it can take control of the memory busses. During this stage, instruction execution is stalled but the timers and serial port operations are not affected. Once the device has finished its operations on the memory bus, then it releases the *dmareq* signal. When the Flip80251 detects the *dmareq* signal has been asserted to 0, it set back low the *dmaack* signal and resumes program execution.

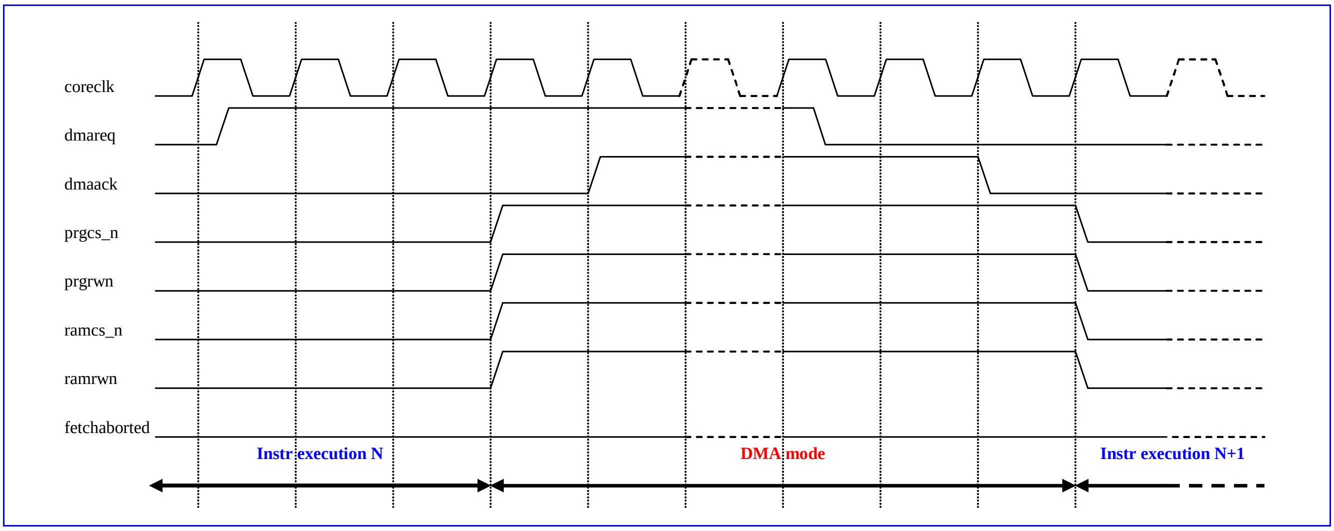


Figure 96: DMA mode during program execution

✋: *dmareq is sampled on clock rising edge.*

- If a DMA request is asserting during the execution of the instruction “RETI”, another instruction will be executed before asserting *dmaack*.
- If an interrupt request occurred concurrently to the DMA request, the DMA will be serviced and then the interruption when the execution of the current instruction is finished.
- If an interrupt is generated during the DMA, an extra instruction could be executed before servicing the interrupt.
- If a DMA request is asserting during the IDLE mode, the DMA will be serviced.

10. DEBUGGING SYSTEM

10.1 Built-In Real-Time Debugger

10.1.1 Overview

The BIRD (Built-in Real-time Debugger) provides the application program developer with a real-time and comfortable application debugging at a low cost.
The BIRD is an embedded debug solution which combines a Virtual Component in the SoC and a Monitor Program, forming the best solution for **real-time** emulation at SoC level with complex events, trace and quick code loading… The BIRD is driven by the RLink-BIRD, a low cost adapter which interfaces the SoC with the development environment on a PC.
The BIRD concept is original since it combines a software solution (monitor) and a hardware (breakpoint, trace modules) solution, and provides high-end debug features whilst minimizing the silicon cost.

BIRD is composed of several elements:

- The **BIRD modules** which are embedded with the CPU, in the SoC
- The **RLink-BIRD**, which is a USB-JTAG adapter uses to transfer data between the PC and the SoC
- The **BIRD Debug Software** (software interface with the IDE) which enables to configure and to control the BIRD modules through the debugger of the IDE.
- The **Monitor program** which could be included or not with the user's application program.
- An optional embedded **trace memory**

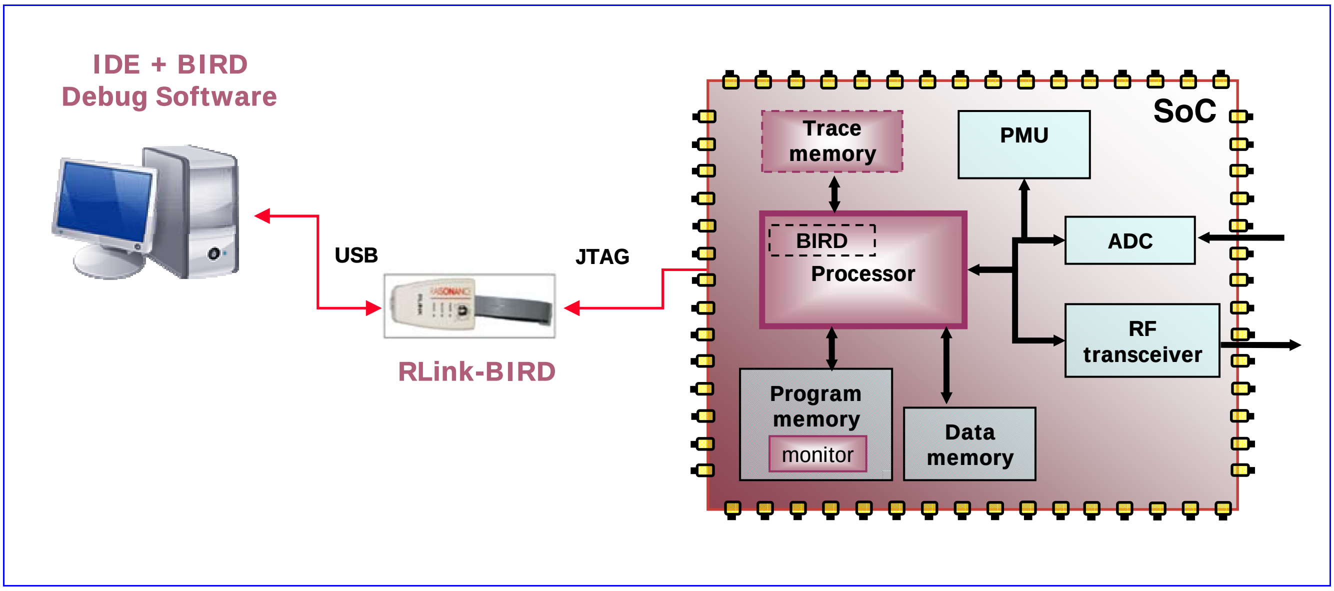


Figure 97: BIRD system

Features	Description
CPU Frequency	Real time, no limitation
Memory access	- Read-write access to all processor registers, memories (SFR, DATA or CODE space) and peripherals connected to the processor core. - Load program (and monitor) memory with binary or Intel Hex file formats
Memory requirement	- Dedicated 512 bytes memory for monitor or reservation of 512 bytes in the CODE space for the monitor program. - Optional dedicated trace memory. Trace memory depth is configurable
Communication	USB 1.1 with the host PC ; JTAG with the SoC
Source level debugging	Single step, C-line to line, step into, step over, go to address, step out
Configuration	**BIRD Tiny**
Code execution Breakpoints	- Unlimited software breakpoints[26]. - 1 real-time hardware breakpoint[27] (PC comparator)
Watchpoint	None
Real-time Trace	None

Table 95: BIRD debugging features

10.1.2 Operating modes

10.1.2.1 Debug mode

When the BIRD is integrated with the Flip80251, it provides a new operating mode to the Flip80251, the **debug mode**.
After an external reset (positive pulse on *corerst* input), the processor is in **normal mode** and the BIRD is clocked off. That means the clock connected to BIRD modules is stopped and debug functions are de-activated.
The **debug mode** is activated when the user starts a debug session on the host PC. In that case, the BIRD Debug Software sends automatically the activation command through the JTAG interface. Then, the BIRD clock is activated and the debug functions become available.
A processor reset initiated by the software debugger does not exit the debug mode. The debug mode is exited only by the applying a positive pulse on *corerst* input.

[26] Requires a writable program memory
[27] Can be used even if the program memory is not writable (e.g. ROM or OTP)

In normal mode, the BIRD does not impact the dynamic power consumption of the Flip80251.
In debug mode, the total dynamic power consumption (BIRD+CPU) is just slightly higher than the power consumption in CPU normal mode.

10.1.2.2 Monitor mode

The monitor mode is entered when the core executes the TRAP interrupt sub-routine (also known as monitor program) caused by a breakpoint (either hardware or software). In that mode, the timers are automatically stopped. This avoids the timers to overflow when the execution is halted.
However, it is possible to bypass this automatic stop by clearing the register MMCON (S:097h). The bits TS0, TS1 and TS2 enable to control, respectively, the run of the timer 0, timer1 and timer2 during the monitor mode.

10.1.2.3 BIRD reset management

During debug mode, the software debugger may send a request for resetting the embedded system (processor +peripherals). In case of a reset initiated by the software debugger, the BIRD generates a reset that is automatically applied to the processor and the inner peripherals.
This BIRD reset is combined with the external reset (*corerst*), it is synchronized with the system clock (*coreclk*) and it is provided as an output (*corerstout*). It can be used to reset any external peripheral and logic connected to the Flip80251.

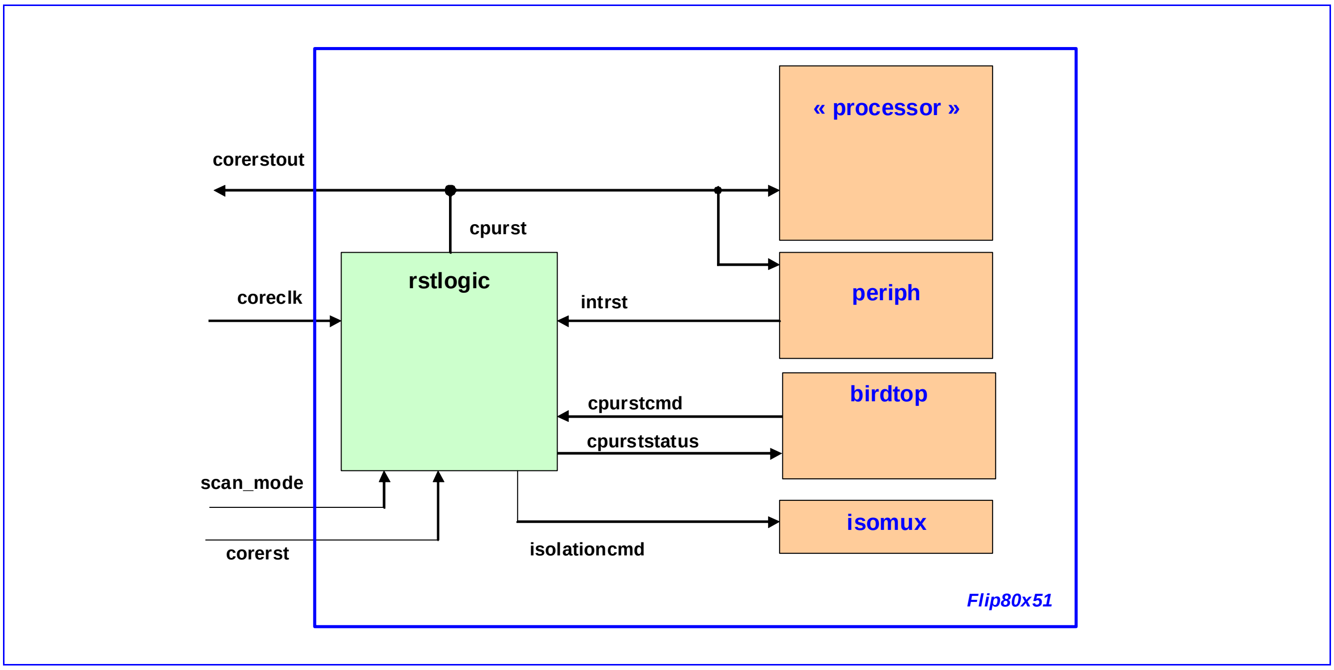


Figure 98: BIRD reset typical connection

10.1.3 BIRD-JTAG interface

In case of BIRD configuration, the Flip80251 includes a debug interface based on the JTAG protocol. It allows to share the dedicated I/O needed for emulation with others tests features. Thus, it avoids having some pins dedicated to debug purpose.

: The JTAG logic has its own clock (tck) and reset (trst_n) and needs to be initialized even when it is not used (normal mode)

10.1.4 Monitor program

The monitor program is a program written in assembly language and it is provided as a part of the delivery of BIRD Debug Software. This program is executed each time a TRAP interrupt is handled. In others words, the monitor program is the TRAP Interrupt Sub-Routine.
The BIRD modules, under the control of the software debugger, uses the monitor program to observe and to control the internal status of the SoC, including the internal registers of the core, the peripheral registers or the data memory contents.

10.1.4.1 Monitor program location

The monitor program is located in the CODE space. Thus, two main implementations can be selected:

1. The program monitor is mapped together with the user application and then it is located into the user program memory. There is no memory dedicated to the program monitor. In such a case, the output *monsel_n* is useless, and the input *moninuser* should be set to logic high

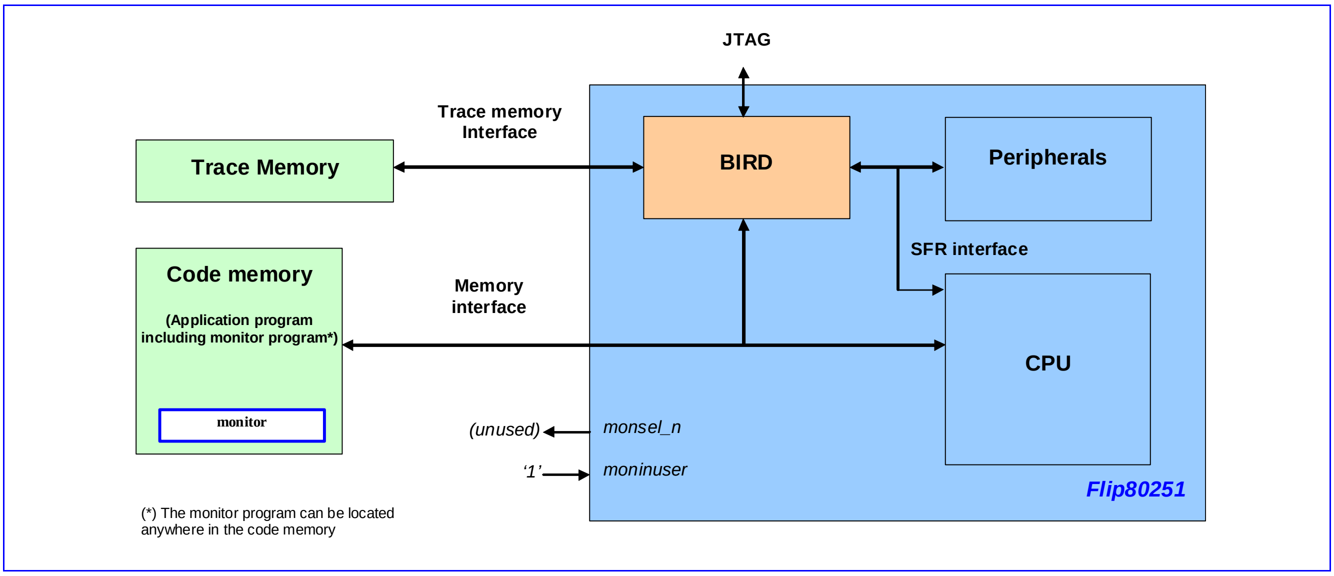


Figure 99: BIRD memory mapping (Monitor program mapped in Application program)

2. Implementation of a **specific 512 bytes memory** for the monitor code. This memory is multiplexed with the user code memory under the control of the output signal *monsel_n*. In such a case, the input *moninuser* should be set to logic low.

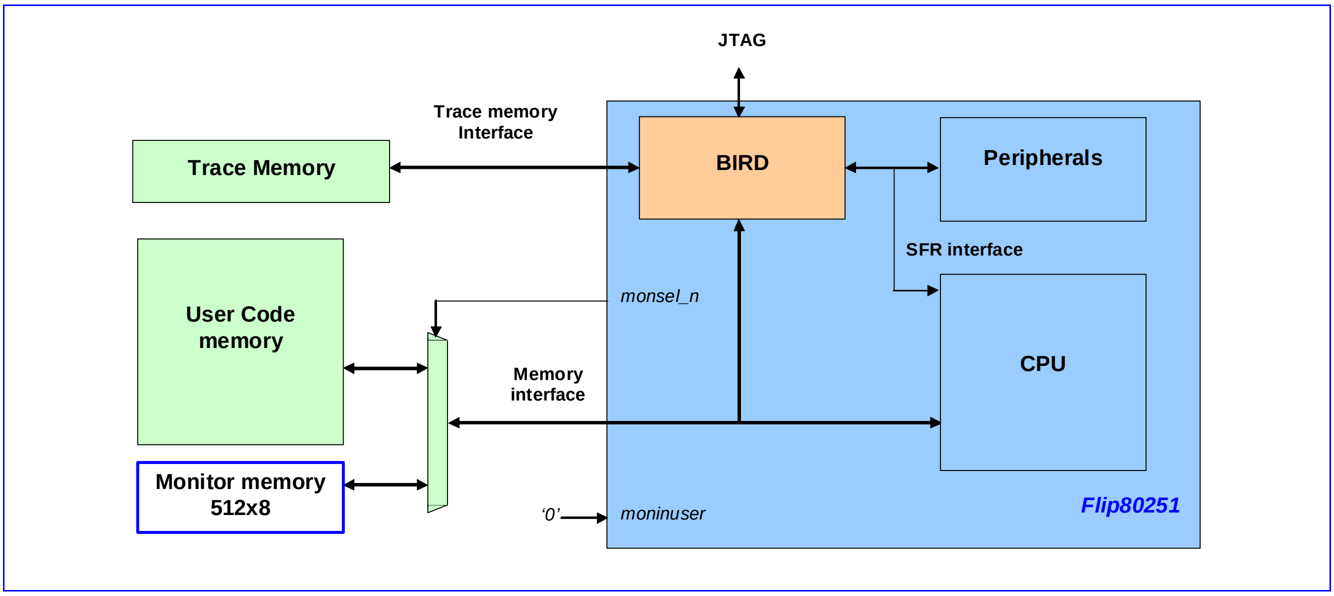


Figure 100: BIRD memory mapping (dedicated memory for Monitor program)

10.1.4.2 BIRD registers

The monitor program uses the SFRs CCMCON (Communication Control Register) and CCMVAL (Communication Value Register) to exchange data between the BIRD and the core. CPUINFO (S:096h) is a read-only register used by the BIRD to read back some information on the configuration of the core , and MMCON (S:097h) is used to control Timer run during monitor mode.

Mnemonic	**Address**	**Description**	**Reset value**
CCMCON	S:08Eh	Communication Control Register	0x00
CCMVAL	S:08Fh	Communication Value Register	0x00
CPUINFO	S:096h	CPU information Register	0x00
MMCON	S:097h	Monitor Mode Control Register	0x07

Table 96: BIRD registers

CPUINFO (S:96h) read only
(BIRD) CPU Information Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	MONI	SRC	INTR	TMS2	TMS1	TMS0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:6	--	**Reserved** The value read from these bits is indeterminate
5	MONI	**Monitor program mapping information bit** Enables to read back the value of the *moninuser* pin
4	SRC	**Source mode / binary mode select** Enable to read back the opcode mode of the Flip80251-Hurricane When high, it indicates that the core is delivered in *source mode* Otherwise, it indicates that the core is in *binary mode*
3	INTR	**Interrupt mode.** Enable to read back the value of *intrmode* pin
2:0	TMS2:0	**Trace Memory size control bus.** Enable to read back the value of the *tmsize[2:0]* bus

Figure 101: CPU Information register (CPUINFO)

MMCON (S:97h)
Monitor Mode Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	--	--	--	TS2	TS1	TS0
RESET	0000 0111b							

Bit Number	Bit Mnemonic	Function
7:3	--	**Reserved** The value read from this bit is indeterminate
2	TS2	**Stop bit for Timer 2** If set, it stops the timer 2 when monitor mode is entered (e.g. when the core is halted by a breakpoint). If clear, timer 2 run is controlled by T2CON.TR2 bit
1	TS1	**Stop bit for Timer 1** If set, it stops the timer 1 when monitor mode is entered (e.g. when the program execution is halted by a breakpoint). If clear, timer 1 run is controlled by TCON.TR1 bit
0	TS0	**Stop bit for Timer 0** If set, it stops the timer 0 when monitor mode is entered (e.g. when the core is halted by a breakpoint). If clear, timer 0 run is controlled by TCON.TR0 bit

Figure 102: Monitor Mode Control register (MMCON)

Appendix A: TECHNICAL DATA

The following tables are filled according to the actual implementation either at FIRM or HARD level, once the customer has chosen the technological process and the standard cell library.

- **Process Reference**
  - Top Metal layer
  - Type of transistor used (SVT, HVT, LVT)
- **Performance**
- **DC characteristic**

Item	Worst	Typical	Best
Process			
Temperature			
Power supply			
Back-Bias Voltage			

- **I/O timings characteristic**

The I/O timings depend on the timing constraints applied to the Flip80251. When delivered at FIRM level, the preliminary I/O timings of the Flip80251 are provided in a liberty format (.lib file) as an example. Actual timing depends on the physical implementation.

- **Power consumption**

The actual power consumption of the Flip80251 depends on the physical implementation and on the program executed by the processor.

- **Testability**

The test strategy applied to the Flip80251 is the scan test. To that end, the Flip80251 was synthesized with scan insertion option. It adds a specific interface to the design in order to drive two scan chains, which are used to apply test vectors.
These two scan chains enable to obtain fault coverage greater than 99 % when using an ATPG tool such as Tetramax from Synopsys.

Appendix B: REGISTER REFERENCE

Mnemonic	Name	Address
ACC (1)	Accumulator	S:0E0h
B (1)	B Register	S:0F0h
DPH (1)	Data Pointer High byte	S:083h
DPL (1)	Data Pointer Low byte	S:082h
DPXL (1)	Data Pointer Extended low byte	S:084h
MPAGE	Memory page register	S:0A1h
PCON	Power Control	S:087h
PSW	Program Status Word	S:0D0h
PSW1	Program Status Word 1	S:0D1h
SP (1)	Stack Pointer low - LSB of SPX	S:081h
SPH (1)	Stack Pointer high - MSB of SPX	S:082h

Table 97: Core SFRs

(1) These registers can also be accessed by their corresponding registers in the register file

Mnemonic	Name	Address
IE0	Interrupt Enable Control 0	S:0A8h
IPH0	Interrupt Priority Control high byte 0	S:0B7h
IPL0	Interrupt Priority Control low byte 0	S:0B8h
AIE	Additional interrupt enable register	S:0E8h
AIF	Additional interrupt flag register	S:0C0h
AIPH	Additional interrupt priority high register	S:0F7h
AIPL	Additional interrupt priority low register	S:0F8h

Table 98: Interrupt SFRs

Mnemonic	Name	Address
P0	Port0	S:080h
P1	Port1	S:090h
P2	Port2	S:0A0h
P3	Port3	S:0B0h
P0_DIR	Port0 direction	S:0ACh
P1_DIR	Port1 direction	S:0ADh
P2_DIR	Port2 direction	S:0AEh
P3_DIR	Port3 direction	S:0AFh

Table 99: I/O ports SFRs

Mnemonic	Name	Address
SADDR	Slave Address	S:0A9h
SADEN	Slave Address mask	S:0B9h
SBUF	Serial Data Buffer	S:099h
SCON	Serial Control	S:098h

Table 100: Serial Port SFRs

Mnemonic	Name	Address
TH0	Timer/Counter 0 high byte	S:08Ch
TL0	Timer/Counter 0 low byte	S:08Ah
TH1	Timer/Counter 1 high byte	S:08Dh
TL1	Timer/Counter 1 low byte	S:08Bh
TH2	Timer/Counter 2 high byte	S:0CDh
TL2	Timer/Counter 2 low byte	S:0CCh
TCON	Timer/Counter 0 and 1 control	S:088h
TMOD	Timer/Counter 0 and 1 mode control	S:089h
T2CON	Timer/Counter 2 control	S:0C8h
T2MOD	Timer/Counter 2 mode control	S:0C9h
RCAP2H	Timer 2 Reload/Capture high byte	S:0CBh
RCAP2L	Timer 2 Reload/Capture low byte	S:0CAh
WDTCON	Watchdog Timer control	S:0A5h
WDTRST	Watchdog Timer enable	S:0A6h

Table 101: Timers SFRs

Mnemonic	Name	Address
CLKCON	Clock Control Register	S:086h
XTALCON	Crystal control Register	S:085h

Table 102: CPMU SFRs

Mnemonic	Name	Address
CCON	PCA Timer/Counter Control	S:0D8h
CMOD	PCA Timer/Counter Mode	S:0D9h
CCAPM0	PCA Compare/Capture Mode for Module 0	S:0DAh
CCAPM1	PCA Compare/Capture Mode for Module 1	S:0DBh
CCAPM2	PCA Compare/Capture Mode for Module 2	S:0DCh
CCAPM3	PCA Compare/Capture Mode for Module 3	S:0DDh
CCAPM4	PCA Compare/Capture Mode for Module 4	S:0DEh
CH	PCA Timer/Counter high byte	S:0F9h
CL	PCA Timer/Counter low byte	S:0E9h
CCAP0H	PCA Compare/Capture Module 0 high byte	S:0FAh
CCAP0L	PCA Compare/Capture Module 0 low byte	S:0EAh
CCAP1H	PCA Compare/Capture Module 1 high byte	S:0FBh
CCAP1L	PCA Compare/Capture Module 1 low byte	S:0EBh
CCAP2H	PCA Compare/Capture Module 2 high byte	S:0FCh
CCAP2L	PCA Compare/Capture Module 2 low byte	S:0ECh
CCAP3H	PCA Compare/Capture Module 3 high byte	S:0FDh
CCAP3L	PCA Compare/Capture Module 3 low byte	S:0EDh
CCAP4H	PCA Compare/Capture Module 4 high byte	S:0FEh
CCAP4L	PCA Compare/Capture Module 4 low byte	S:0EEh
CCAPO	PCA Output for PWM and high speed mode	S:0DFh

Table 103: PCA SFRs

Mnemonic	Name	Address
CCMCON	Communication Control Register	S:08Eh
CCMVAL	Communication Value Register	S:08Fh
CPUINFO	CPU information (read only register)	S:096h
MMCON	Monitor mode control register	S:097h

Table 104: Debug SFRs

Mnemonic	Name	Address
PWMC	PWM Control Register	S:0A2h
PWMDCLSB	PWM Duty Cycle LSB Register	S:0A3h
PWMDCMSB	PWM Duty Cycle MSB Register	S:0A4h

Table 105: PWM SFRs

Mnemonic	Name	Address
MCON	I2CM Control register	S:0E1h
MRXBUF	I2CM Reception buffer	S:0E2h
MTXBUF	I2CM Transmission Buffer	S:0E3h
MPRESC	I2CM Pre-scalar clock register	S:0E4h
MSTAT0	I2CM Status register 0	S:0E5h
MSTAT1	I2CM Status register 1	S:0E6h
MIEN0	I2CM Interrupt Enable register 0	S:0E7h
MIEN1	I2CM Interrupt Enable register 1	S:0D2h
MCADDR	I2CM Call Address register	S:0D4h

Table 106: I2CM SFRs

Mnemonic	Name	Address
STCON	I2CS Transfer Control register	S:0F1h
SRXBUF	I2CS Reception Buffer	S:0F2h
STXBUF	I2CS Transmission Buffer	S:0F3h
SSTAT0	I2CS Status register 0	S:0F5h
SSTAT1	I2CS Status register 1	S:0F6h
SIEN0	I2CS Interrupt Enable register 0	S:0D5h
SIEN1	I2CS Interrupt Enable register 1	S:0D6h
SSADDR	I2CS Self Address register	S:0D7h

Table 107: I2CS SFRs

Mnemonic	Name	Address
SPCR	SPI Control Register	S:0B1h
SPDR	SPI Data Register	S:0B2h
SPSR	SPI Status Register	S:0B3h

Table 108: SPI SFRs

Mnemonic	Name	Address
R0-R7	Four banks of 8 registers. Selects banks 0-3 with bits (RS0,RS1) of PWM	**(1)(2)**

R8-R31	R11 = Accumulator (ACC)
	R10 = B Register
		(1)(3)

R32-R55	Reserved	**(3)**

R56-R63	DR56 = Extended Data Pointer (DPXL, DPH, DPL)
	DR60 = Extended Stack Pointer (SPH, SPL)
		(1)(3)

Table 109: Register file

(1) The register in the register file are normally accessed by mnemonic. Depending on its location, a register can be addressed as a byte, a word and/or a dword

(2) The four banks of registers are implemented as the lowest bytes of on-chip RAM and are always accessible via addresses 00:0000h-00:001Fh

(3) Special Function registers ACC, B, DPXL, DPH, DPL, SPH and SPL are located in the register file and can be accessed as R11, R10, DR56 and DR60

ACC (S:E0h)

Accumulator

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Accumulator data**

Accumulator ACC provides SFR accesses to the accumulator which resides in the register file as byte register R11. Instructions, in the MCS ® 51 architecture, use the accumulator as both source and destination for calculations and moves. Instructions, in the MCS ® 251 architecture, assign no special significance to R11. These instructions can use byte registers Rm (m= 0-15) interchangeably.

Figure 103: Accumulator Register (ACC)

AIE (S:E8h)

Additional Interrupt Enable Register

BIT	7	6	5	4	3	2	1	0
FIELD	-	AIE6	AIE5	AIE4	AIE3	AIE2	AIE1	AIE0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	AIE6	**Additional interrupt 6 Enable** Set to enable Additional Interrupt 6. Clear to disable Additional Interrupt 6
5	AIE5	**Additional interrupt 5 Enable** Set to enable Additional Interrupt 5. Clear to disable Additional Interrupt 5
4	AIE4	**Additional interrupt 4 Enable** Set to enable Additional Interrupt 4. Clear to disable Additional Interrupt 4
3	AIE3	**Additional interrupt 3 Enable** Set to enable Additional Interrupt 3. Clear to disable Additional Interrupt 3
2	AIE2	**Additional interrupt 2 Enable** Set to enable Additional Interrupt 2. Clear to disable Additional Interrupt 2
1	AIE1	**Additional interrupt 1 Enable** Set to enable Additional Interrupt 1. Clear to disable Additional Interrupt 1
0	AIE0	**Additional interrupt 0 Enable** Set to enable Additional Interrupt 0. Clear to disable Additional Interrupt 0

Figure 104: Additional Interrupt Enable Register (AIE)

AIF (S:C0h)
Additional Interrupt Flag Register

BIT	7	6	5	4	3	2	1	0
FIELD	-	AIF6	AIF5	AIF4	AIF3	AIF2	AIF1	AIF0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	AIF6	**Additional interrupt 6 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
5	AIF5	**Additional interrupt 5 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
4	AIF4	**Additional interrupt 4 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
3	AIF3	**Additional interrupt 3 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
2	AIF2	**Additional interrupt 2 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
1	AIF1	**Additional interrupt 1 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled
0	AIF0	**Additional interrupt 0 Flag** Set by hardware when a low level is detected on the corresponding pin. Clear by software when the corresponding interrupt sub routine is handled

Figure 105: Additional interrupt flag register (AIF)

AIPH (S:F7h)
Additional Interrupt Priority High Register

BIT	7	6	5	4	3	2	1	0
FIELD	-	AIPH6	AIPH5	AIPH4	AIPH3	AIPH2	AIPH1	AIPH0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	AIPH6	**Additional interrupt 6 Priority Most significant bi**
5	AIPH5	**Additional interrupt 5 Priority Most significant bit**
4	AIPH4	**Additional interrupt 4 Priority Most significant bit**
3	AIPH3	**Additional interrupt 3 Priority Most significant bit**
2	AIPH2	**Additional interrupt 2 Priority Most significant bit**
1	AIPH1	**Additional interrupt 1 Priority Most significant bit**
0	AIPH0	**Additional interrupt 0 Priority Most significant bit**

Figure 106: Additional Interrupt Priority High register (AIPH)

AIPL (S:F8h)

Additional Interrupt Priority Low Register

BIT	7	6	5	4	3	2	1	0
FIELD	-	AIPL6	AIPL5	AIPL4	AIPL3	AIPL2	AIPL1	AIPL0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	AIPL6	**Additional interrupt 6 Priority Less significant bi**
5	AIPL5	**Additional interrupt 5 Priority Less significant bit**
4	AIPL4	**Additional interrupt 4 Priority Less significant bit**
3	AIPL3	**Additional interrupt 3 Priority Less significant bit**
2	AIPL2	**Additional interrupt 2 Priority Less significant bit**
1	AIPL1	**Additional interrupt 1 Priority Less significant bit**
0	AIPL0	**Additional interrupt 0 Priority Less significant bit**

Figure 107: Additional Interrupt Priority Low register (AIPL)

B (S:F0h)

B register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**B data**

The B register provides SFR access to byte register R10 (also named B) in the register file. The B register is used as either a source or destination in multiply and divide operations. For all other operations, the B register is available for use as one of the byte register Rm (m= 0-15).

Figure 108: B Register (B)

CCAP0H (S:FAh)
CCAP1H (S:FBh)
CCAP2H (S:FCh)
CCAP3H (S:FDh)
CCAP4H (S:FEh)
High Byte Compare/Capture Module x Register (x=0, 1, 2, 3, 4)

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**High Byte of PCA comparison or capture value**

Figure 109: PCA Compare/Capture Module x High Byte Registers (CCAPxH)

CCAP0L (S:EAh)
CCAP1L (S:EBh)
CCAP2L (S:ECh)
CCAP3L (S:EDh)
CCAP4L (S:EEh)
Low Byte Compare/Capture Module x Register (x=0, 1, 2, 3, 4)

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Low Byte of PCA comparison or capture value**

Figure 110: PCA Compare/Capture Module x Low Byte Registers (CCAPxL)

CCAPM0 (S:DAh)
CCAPM1 (S:DBh)
CCAPM2 (S:DCh)
CCAPM3 (S:DDh)
CCAPM4 (S:DEh)

PCA Compare/Capture Module x Mode Register (x=0, 1, 2, 3, 4)

BIT	7	6	5	4	3	2	1	0
FIELD	--	ECOMx	CAPPx	CAPNx	MATx	TOGx	PWMx	ECCFx
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	ECOMx	**Compare modes.** Clear to disable the compare function. Set to enable the compare function The compare function is used to implement the software timer mode, the high speed output mode, the PWM mode and the watchdog timer mode
5	CAPPx	**Capture mode (positive)** Set to enable the capture function on a positive edge of CEXx.
4	CAPNx	**Capture mode (negative)** Set to enable the capture function on a negative edge of CEXx.
3	MATx	**Match.** If set, a match of the PCA timer/counter will generate an interrupt request (ECCFx must also be set)
2	TOGx	**Toggle.** If set, a match of the PCA timer/counter toggles the CEXx output
1	PWMx	**Pulse Width modulation mode.** Set to configure the module x as an 8-bits PWM
0	ECCFx	**Enable CCFx Interrupt.** Set to enable the compare/capture flag CCON.CCFx to generate an interrupt request.

Figure 111: PCA compare/capture module mode register (CCAPMx)

CCAPO (S:DFh

PCA timer/counter output for PWM and high-speed mode Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	--	CCAPO4	CCAPO3	CCAPO2	CCAPO1	CCAPO0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:5	--	**Reserved**. The value read from this bit is indeterminate
4	CCAPO4	**Compare/capture module 4 output value**
3	CCAPO3	**Compare/capture module 4 output value**
2	CCAPO2	**Compare/capture module 4 output value**
1	CCAPO1	**Compare/capture module 4 output value**
0	CCAPO0	**Compare/capture module 4 output value**

CCAPO register can be written by hardware and by software (through SFR interface). In case of concurrent writing in the same clock cycle, then only the software write will be performed

Figure 112: PCA timer/counter output for PWM and high-speed mode register (CCAPO)

CCMCON (S:8Eh)

(BIRD) Communication Control Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Control value sent by the BIRD Debug software to the BIRD modules**

Figure 113: (BIRD) Communication Control Register (CCMCON)

CCMVAL (S:8Fh)

(BIRD) Communication Data Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Data value sent by the BIRD Debug software to the BIRD modules**

Figure 114: (BIRD) Communication Data Register (CCMVAL)

CCON (S:D8h)

PCA Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	CF	CR	--	CCF4	CCF3	CCF2	CCF1	CCF0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	CF	**PCA timer/counter overflow Flag.** Set by hardware when the PCA timer/counter rolls over. This generates an interrupt request if the CMOD.ECF interrupt bit is set. CF can be set by hardware or software, but can be cleared only by software.
6	CR	**PCA Timer/counter Run Control bit.** Set and cleared by software to turn the PCA timer/counter on and off
5	--	**User flag** This is a general purpose flag
4:0	CCF4:0	**PCA Module compare/capture flags.** Set by hardware when a match or capture occurs. This generates a PCA interrupt request if the CCAPMx.ECCFx interrupt enable bit is set. Must be cleared by software.

Figure 115: PCA timer/counter control register (CCON)

CL (S:E9h)

Low Byte of PCA Timer/counter Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Low byte of PCATimer/Counter**

Figure 116: PCA Timer/counter Register Low Byte Register (CL)

CH (S:F9h)

High Byte of PCA Timer/counter Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**High byte of PCATimer/Counter**

Figure 117: PCA Timer/counter Register high byte Register (CH)

CLKCON (S:86h)
Clock Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	FDIV2	FDIV1	FDIV0	RGEN	CLKSEL	SDIV2	SDIV1	SDIV0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	FDIV2	**Clock 1 divider ratio bit 2**
6	FDIV1	**Clock 1 divider ratio bit 1**
5	FDIV0	**Clock 1 divider ratio bit 0**
4	RGEN	**Ring oscillator selection.** When set, the clock selected is the ring oscillator source. When clear, the clock selection depends on CLKSEL bit
3	CLKSEL	**Clock selection bit.** When set, clock 2 is selected. When cleared, clock 1 is selected.
2	SDIV2	**Clock 2 divider ratio bit 2**
1	SDIV1	**Clock 2 divider ratio bit 1**
0	SDIV0	**Clock 2 divider ratio bit 0**

Figure 118: (CPMU) Clock Control register (CLKCON)

CMOD (S:D9h)
PCA Mode Register

BIT	7	6	5	4	3	2	1	0
FIELD	CIDL	WDTE	UF2	UF1	UF0	CPS1	CPS0	ECF
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	CIDL	**PCA Timer/counter idle control.** If set, the timer/counter is stopped during idle mode
6	WDTE	**PCA Watchdog Timer enable.** If set, the watchdog timer output on module 4 is enable
5:3	UF2:0	**User flags** General purpose flag
2:1	CPS1:0	**PCA Timer/counter input select.**
0	ECF	**PCA timer/counter interrupt enable.** If set, an overflow of the PCA timer/counter generates an interrupt request.

CPS1	CPS0	Mode	Input
0	0	0	clkdiv12
0	1	1	clkdiv4
1	0	2	timer0overflow
1	1	3	ECI (max frequency = system clock / 8)

To save an interrupt request during a read-modify-write instruction on CCON, or any concurrent write access, the bits CF and CCFx are updated by hardware only at the end of the current instruction

Figure 119: PCA timer/counter mode register (CMOD)

CPUINFO (S:96h) read only

(BIRD) CPU Information Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	MONI	SRC	INTR	TMS2	TMS1	TMS0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:6	--	**Reserved** The value read from these bits is indeterminate
5	MONI	**Monitor program mapping information bit** Enables to read back the value of the *moninuser* pin
4	SRC	**Source mode / binary mode select** Enable to read back the opcode mode of the Flip80251-Hurricane When high, it indicates that the core is delivered in *source mode* Otherwise, it indicates that the core is in *binary mode*
3	INTR	**Interrupt mode.** Enable to read back the value of *intrmode* pin
2:0	TMS2:0	**Trace Memory size control bus.** Enable to read back the value of the *tmsize[2:0]* bus

Figure 120: CPU Information register (CPUINFO)

DPH (S:83h)

Data Pointer High

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Data Pointer High** Bit 8-15 of the extended Data Pointer DPX (DR56)

DPH provides SFR access to register file location 58 (also named DPH). DPH is the upper byte of the 16-bit data pointer DPTR. Instruction in the MCS ® 51 architecture use DPTR for data moves, code moves and for jump instructions

Figure 121: Data Pointer High register (DPH)

DPL (S:82h)
Data Pointer Low

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Data Pointer Low** Bit 0-7 of the extended Data Pointer DPX (DR56)

DPL provides SFR access to register file location 59 (also named DPL). DPL is the lower byte of the 16-bit data pointer DPTR. Instruction in the MCS ® 51 architecture use DPTR for data moves, code moves and for jump instructions

Figure 122: Data Pointer Low Register (DPL)

DPXL (S:84h)
Data Pointer Extended Low

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Data Pointer Extended Low** Bit 16-23 of the extended Data Pointer DPX (DR56)

DPXL provides SFR access to register file location 57 (also named DPXL). DPXL is the lower byte of the upper word of extended data pointer DPX whose lower word is the 16-bit data-pointer DPTR

Figure 123: Data Pointer Extended Low Register (DPXL)

IE0 (S:A8h)

Interrupt Enable Register

BIT	7	6	5	4	3	2	1	0
FIELD	EA	EC	ET2	ES	ET1	EX1	ET0	EX0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	EA	**Global Interrupt Enable** Clear to disable all interrupts, except the TRAP and NMI interrupts, which are always enabled. Set to enable all interrupts that are individually enabled in IE0.
6	EC	**PCA Interrupt Enable** Set to enable PCA Interrupt. Cleared to disable PCA Interrupt
5	ET2	**Timer2 Interrupt Enable** Set to enable Timer2 Interrupt. Cleared to disable Timer2 Interrupt
4	ES	**Serial Port Interrupt Enable** Set to enable Serial Port Interrupt. Cleared to disable Serial Port Interrupt
3	ET1	**Timer1 Interrupt Enable** Set to enable Timer1 Interrupt. Cleared to disable Timer1 Interrupt
2	EX1	**External Interrupt 1 enable** Set to enable External Interrupt 1. Cleared to disable External Interrupt 1.
1	ET0	**Timer0 Interrupt Enable** Set to enable Timer0 Interrupt. Cleared to disable Timer0 Interrupt
0	EX0	**External Interrupt 0 enable** Set to enable External Interrupt 0. Cleared to disable External Interrupt 0.

Figure 124: Interrupt Enable register 0 (IE0)

IPH0 (S:B7h)

Interrupt Priority High Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	IPHC	IPHT2	IPHS	IPHT1	IPHX1	IPHT0	IPHX0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	IPHC	**PCA Interrupt Priority level most significant bit** IPHC IPLC Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
5	IPHT2	**Timer2 Interrupt Priority level most significant bit** IPHT2 IPLT2 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
4	IPHS	**Serial Port Interrupt Priority level most significant bit** IPHT2 IPLT2 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
3	IPHT1	**Timer1 Interrupt Priority level most significant bit** IPHT1 IPLT1 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
2	IPHX1	**External interrupt 1 Priority level most significant bit** IPHX1 IPLX1 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
1	IPHT0	**Timer0 Interrupt Priority level most significant bit** IPHT0 IPLT0 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority
0	IPHX0	**External interrupt 0 Priority level most significant bit** IPHX0 IPLX0 Priority Level 0 0 0 Lowest priority 0 1 1 1 0 2 1 1 3 Highest priority

Figure 125: Interrupt Priority High register 0 (IPH0)

IPL0 (S:B8h)

Interrupt Priority Low Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	IPLC	IPLT2	IPLS	IPLT1	IPLX1	IPLT0	IPLX0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved**. The value read from this bit is indeterminate
6	IPLC	**PCA Interrupt Priority level less significant bit** Refer to IPH0 for priority level description
5	IPLT2	**Timer2 Interrupt Priority level less significant bit** Refer to IPH0 for priority level description
4	IPLS	**Serial Port Interrupt Priority level less significant bit** Refer to IPH0 for priority level description
3	IPLT1	**Timer1 Interrupt Priority level less significant bit** Refer to IPH0 for priority level description
2	IPLX1	**External interrupt 1 Priority level less significant bit** Refer to IPH0 for priority level description
1	IPLT0	**Timer0 Interrupt Priority level less significant bit** Refer to IPH0 for priority level description
0	IPLX0	**External interrupt 0 Priority level less significant bit** Refer to IPH0 for priority level description

Figure 126: Interrupt Priority Low Register 0 (IPL0)

MCADDR (S:D4h)

I2CM Call Address Register

BIT	7	6	5	4	3	2	1	0
FIELD	RWN	CADDR						
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	RWN	**Read/write control bit for I2C transaction** Set to read data from addressed slave device Clear to write data to the addressed slave device
6:0	CADDR	**7-bit Call Address** This register must be written before the beginning of an I2C transaction.

Figure 127: I2CM Call address register (MCADDR)

MCON (S:E1h)
I2CM Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	MWS	--	STO	SRST	STA	BUSY
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved** The value read from this bit is indeterminate.
6		**Reserved** The value read from this bit is indeterminate.
5	MWS	**I2CM Wait State** Set to generate wait state on SCL line when RX overflows. When clear, I2CM sends “Not Acknowledge” to stop the transmission when RX overflows.
4	--	**Reserved** The value read from this bit is 0.
3	STO	**Generate Stop condition** When this bit is set, the current byte ends normally and a STOP condition is generated just after the acknowledge cycle. This bit is automatically cleared by the controller when the STOP condition has been sent.
2	SRST	**Software reset** This bit is automatically cleared once IDLE state is reached.
1	STA	**Generate Start condition** This bit is automatically cleared by the controller when the transmission has begun or if an error is detected.
0	BUSY	**BUSY flag** This bit is set to ‘1’ when an I2C frame transfer is in progress on I2C bus.

Figure 128: I2CM Control Register (MCON)

MIEN0 (S:E7h)

I2CM Interrupt Enable Register 0

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	--	EDNA	ESANA	EMUNF	EMOVF	EMNE
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved** The value read from this bit is indeterminate.
6	--	**Reserved** The value read from this bit is indeterminate.
5	--	**Reserved** The value read from this bit is indeterminate.
4	EDNA	**Data byte Not Acknowledged Interrupt enable bit** Clear to disable MSTAT0.DNA bit to generate an interrupt request Set to enable MSTAT0.DNA bit to generate an interrupt request
3	ESANA	**Slave Address Not Acknowledged Interrupt enable bit** Clear to disable MSTAT0.SANA bit to generate an interrupt request Set to enable MSTAT0.SANA bit to generate an interrupt request
2	EMUNF	**I2CM Underflow Interrupt enable bit** Clear to disable MSTAT0.MUNF bit to generate an interrupt request Set to enable MSTAT0.MUNF bit to generate an interrupt request
1	EMOVF	**I2CM Overflow Interrupt enable bit** Clear to disable MSTAT0.MOVF bit to generate an interrupt request Set to enable MSTAT0.MOVF bit to generate an interrupt request
0	EMNE	**I2CM Normal End Interrupt enable bit** Clear to disable MSTAT0.MNE bit to generate an interrupt request Set to enable MSTAT0.MNE bit to generate an interrupt request

Figure 129: I2CM Interrupt Enable register 0 (MIEN0)

MIEN1 (S:D2h)

I2CM Interrupt Enable Register 1

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	EMTBE	--	EMTBF	EMRBE	--	EMRBF
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved** The value read from this bit is indeterminate.
6	--	**Reserved** The value read from this bit is indeterminate.
5	EMTBE	**I2CM Transmission Buffer Empty Interrupt enable bit** Clear to disable MSTAT1.MTBE bit to generate an interrupt request Set to enable MSTAT1.MTBE bit to generate an interrupt request
4	--	**Reserved** The value read from this bit is indeterminate.
3	EMTBF	**I2CM Transmission Buffer Full Interrupt enable bit** Clear to disable MSTAT1.MTBF bit to generate an interrupt request Set to enable MSTAT1.MTBF bit to generate an interrupt request
2	EMRBE	**I2CM Reception Buffer Empty Interrupt enable bit** Clear to disable MSTAT1.MRBE bit to generate an interrupt request Set to enable MSTAT1.MRBE bit to generate an interrupt request
1	--	**Reserved** The value read from this bit is indeterminate.
0	EMRBF	**I2CM Reception Buffer Full Interrupt enable bit** Clear to disable MSTAT1.MRBF bit to generate an interrupt request Set to enable MSTAT1.MRBF bit to generate an interrupt request

Figure 130: I2CM Interrupt Enable register 1 (MIEN1)

MMCON (S:97h)
Monitor Mode Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	--	--	--	TS2	TS1	TS0
RESET	0000 0111b							

Bit Number	Bit Mnemonic	Function
7:3	--	**Reserved** The value read from this bit is indeterminate
2	TS2	**Stop bit for Timer 2** If set, it stops the timer 2 when monitor mode is entered (e.g. when the core is halted by a breakpoint). If clear, timer 2 run is controlled by T2CON.TR2 bit
1	TS1	**Stop bit for Timer 1** If set, it stops the timer 1 when monitor mode is entered (e.g. when the program execution is halted by a breakpoint). If clear, timer 1 run is controlled by TCON.TR1 bit
0	TS0	**Stop bit for Timer 0** If set, it stops the timer 0 when monitor mode is entered (e.g. when the core is halted by a breakpoint). If clear, timer 0 run is controlled by TCON.TR0 bit

Figure 131: Monitor Mode Control register (MMCON)

MPAGE (S:A1h)
Memory Page Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	1111 1111b							

Bit Number	Bit Mnemonic	Function
7:0		**Memory page value**.

When MOVX @Ri instructions are used, the MSB of the16-bit address is filled with the content of MPAGE register. Then, it allows MOVX @Ri instruction to access to 64 Kbytes of external data memory. Usually, in 80C51 application, the Port 2 is used to this address extension. In order to keep software compatibility with existing 80C51 program, the register MPAGE is also updated by any value written at P2 register.

Figure 132: Memory Page Register (MPAGE)

MPRESC (S:E4h)

I2CM Clock Prescalar Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0	PRESC	**Clock pre scalar register**. Fscl = $\frac{Fclk}{10*(PRESC+1)}$

This register should not be written during a transmission.

Figure 133: I2CM Pre-scalar Clock Register (MPRESC)

MRXBUF (S:E2h) Read only

I2CM Receive Buffer

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Data received by I2CM**

Figure 134: I2CM Reception Register (MRXBUF)

MTXBUF (S:E3h) Write only

I2CM Transmit Buffer

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Data transmitted by I2CM**

Figure 135: I2CM Transmission Buffer (MTXBUF)

MSTAT0 (S:E5h) Read only
I2CM Status Register 0

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	--	DNA	SANA	MUNF	MOVF	MNE
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved** The value read from this bit is indeterminate.
6	--	**Reserved** The value read from this bit is indeterminate.
5	--	**Reserved** The value read from this bit is indeterminate.
4	DNA	**Data byte not acknowledged** Data byte not acknowledged during transmission. A “stop” condition sent.
3	SANA	**Slave Address Not Acknowledged** Slave Address not acknowledged. Stop condition sent.
2	MUN	**I2CM Transmission Underflow** Transmit Data Byte not ready (Transmit Buffer is empty) while a new data byte needs to be sent. A “stop” condition is sent.
1	MOV	**I2CM Reception Overflow** Received Data Byte could not be written (Receive Buffer is full) while a new byte was received. A Not Acknowledge and a “stop” condition are sent.
0	MNE	**I2CM Normal End (End of access with no error)** Set when a stop is sent at the end of a successful access. Clear automatically when a new I2C access starts.

These interrupt sources are automatically cleared after a read access to this register.

When DNA, SANA, MUNF or MOVF flags have been set, reception and transmission processes are disabled until the CPU has read MSTAT1 register. This read operation automatically resets MSTAT1 register and MCON.STA bit, if one of these error bits is set. If this read operation is performed while no error bit is set, MCON.STA bit is not cleared.

These interrupt sources can all be individually enabled/disabled by TXRX_IE register. The *OTXRXINT* output signal is set to ‘1’ when one or several interrupt sources are active and enabled. When a disabled interrupt occurs, *OTXRXINT* remains unchanged, but the corresponding interrupt bit is set.

Figure 136: I2CM Status Register 0 (MSTAT0)

MSTAT1 (S:E6h) Read only

I2CM Status Register 1

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	MTBE	--	MTBF	MRBE	--	MRBF
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved** The value read from this bit is indeterminate.
6	--	**Reserved** The value read from this bit is indeterminate.
5	MTBE	**I2CM Transmission buffer is empty** Set to 1 when transmission Buffer is empty. This flag is cleared when the CPU performs a write access to MTXBUF register. Clear to 0 when at least one Byte is available for data transmission
4	--	**Reserved** The value read from this bit is 0.
3	MTBF	**I2CM Transmission buffer is full** Set to 1 when transmission Buffer is full. No more write operation into transmission buffer or memory is performed (CPU write request to MTXBUF not taken in account). This flag is cleared when a new Data Byte is requested by the I2C transfer controller. Clear to 0 when transmission Buffer is empty
2	MRBE	**I2CM Reception buffer is empty** Set to 1 when Reception Buffer empty. No more write operation into Reception buffer or memory is performed (CPU read request to MRXBUF not taken in account). This flag is cleared when a new Data Byte is received by the I2C transfer controller. Clear to 0 when at least one received Data Byte is available.
1	--	**Reserved** The value read from this bit is indeterminate.
0	MRBF	**I2CM Reception buffer is full** Set to 1 when Reception Buffer full. This flag is cleared when the CPU performs a read operation to MRXBUF register. Clear to 0 when Reception Buffer is empty

These interrupt sources can all be individually enabled/disabled by MIEN1 register.
The *OFIFOINT* output signal is set to ‘1’ when one or several interrupt sources are active and enabled. When a disabled interrupt occurs, *OFIFOINT* remains unchanged, but the corresponding interrupt bit is set. These interrupt sources are cleared when the condition which has set them disappears.

Figure 137: I2CM Status Register 1 (MSTAT1)

P0 (S:80h)

Port 0 Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	1111 1111b							

Bit Number	Bit Mnemonic	Function
7:0		**Port 0 data** Write data to be driven out from the Port 0 pins.

Read-Modify Write instructions that read port 0 read this register. The other instructions that read port 0 read the port 0 pins

Figure 138: Port 0 Register (P0)

P1 (S:90h)

Port 1 Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	1111 1111b							

Bit Number	Bit Mnemonic	Function
7:0		**Port 1 data** Write data to be driven out from the Port 1 pins.

Read-Modify Write instructions that read port 1 read this register. The other instructions that read port 1 read the port 1 pins

Figure 139: Port 1 Register (P1)

P2 (S:A0h)

Port 2 Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	1111 1111b							

Bit Number	Bit Mnemonic	Function
7:0		**Port 2 data** Write data to be driven out from the Port 2 pins.

Read-Modify Write instructions that read port 2 read this register. The other instructions that read port 2 read the port 2 pins

Figure 140: Port 2 Register (P2)

P3 (S:B0h)
Port 3 Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	1111 1111b							

Bit Number	Bit Mnemonic	Function
7:0		**Port 3 data** Write data to be driven out from the Port 3 pins.

Read-Modify Write instructions that read port 3 read this register. The other instructions that read port 3 read the port 3 pins

Figure 141: Port 3 Register (P3)

P0_DIR (S:ACh)
P1_DIR (S:ADh)
P2_DIR (S:AEh)
P3_DIR (S:AFh)
Port x Direction Register (x=0, 1, 2, 3)

BIT	7	6	5	4	3	2	1	0
FIELD	PxDIR7	PxDIR6	PxDIR5	PxDIR4	PxDIR3	PxDIR2	PxDIR1	PxDIR0
RESET	1111 1111b							

Bit Number	Bit Mnemonic	Function
7:0	PxDIR7	**Direction of Port x bit 7** Set to configure Portx.7 as an input. Clear to configure Portx.7 as an output
	PxDIR6	**Direction of Port x bit 6** Set to configure Portx.6 as an input. Clear to configure Portx.6 as an output
	PxDIR5	**Direction of Port x bit 5** Set to configure Portx.5 as an input. Clear to configure Portx.5 as an output
	PxDIR4	**Direction of Port x bit 4** Set to configure Portx.4 as an input. Clear to configure Portx.4 as an output
	PxDIR3	**Direction of Port x bit 3** Set to configure Portx.3 as an input. Clear to configure Portx.3 as an output
	PxDIR2	**Direction of Port x bit 2** Set to configure Portx.2 as an input. Clear to configure Portx.2 as an output
	PxDIR1	**Direction of Port x bit 1** Set to configure Portx.1 as an input. Clear to configure Portx.1 as an output
	PxDIR0	**Direction of Port x bit 0** Set to configure Portx.0 as an input. Clear to configure Portx.0 as an output

Figure 142: Port direction Registers (Px_DIR)

PCON (S:87h)
Power Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	SMOD1	SMOD0	--	POF	GF1	GF0	PD	IDL
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	SMOD1	**Double baud rate** Set to double the baud rate when timer 1 is used and serial mode 0 is not selected (see Serial port chapter)
6	SMOD0	**Select function of SCON.7** Set to access to SCON.7 as the FE bit Clear to access to SCON.7 as the SM0 bit (see Serial port chapter)
5	--	**Reserved** The value read from this bit is indeterminate
4	POF	**Power Off flag** Set by hardware when the input "*poweroff*" is high It can be set or cleared by software.
3	GF1	**General purpose flag 1** Set or cleared by software
2	GF0	**General purpose flag 0** Set or cleared by software
1	PD	**Power-down mode bit** Set to activate power-down mode Clear by hardware when an enabled external interrupt or a reset occurs.
0	IDL	**Idle mode bit** Set to activates idle mode Clear by hardware when an enabled interrupt or a reset occurs.

In standard 80251, Power Off flag is set by hardware as Vcc rises above TBD voltage to indicate that power has been off or Vcc had fallen below a TBD voltage and that on-chip volatile memory is indeterminate.

Figure 143: Power control Register (PCON)

PSW (S:D0h)

Program Status Word Register

BIT	7	6	5	4	3	2	1	0
FIELD	CY	AC	F0	RS1	RS0	OV	UD	P
RESET	0000 0000b							

<table>
<tr><th>Bit Number</th><th>Bit Mnemonic</th><th>Function</th></tr>
<tr><td>7</td><td>CY</td><td><b>Carry Flag:</b>
The carry flag is set by an addition instruction (ADD, ADDC) if there is a carry out of the MSB. It is set by a subtraction (SUB, SUBB) or compare (CMP) if a borrow is needed for the MSB. The carry flag is also affected by some rotate and shift instructions, logical bit instructions, bit move instructions, and the multiply (MUL) and decimal adjust (DA) instructions.</td></tr>
<tr><td>6</td><td>AC</td><td><b>Auxiliary Carry Flag:</b>
The auxiliary carry flag is affected only by instructions that address 8-bit operands. The AC flag is set if an arithmetic instruction with an 8-bit operand produces a carry out of bit 3 (from addition) or a borrow into bit 3 (from subtraction). Otherwise, it is cleared. This flag is useful for BCD arithmetic.</td></tr>
<tr><td>5</td><td>F0</td><td><b>Flag 0:</b>
This general purpose flag is available to the user</td></tr>
<tr><td>4:3</td><td>RS1:0</td><td><b>Register Bank Select bits 1 and 0:</b>
<table>
<tr><th>RS1</th><th>RS0</th><th>Register Bank</th><th>Address</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>00h-07h</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>08h-0Fh</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>10h-17h</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>18h-1Fh</td></tr>
</table></td></tr>
<tr><td>2</td><td>OV</td><td><b>Overflow Flag:</b>
This bit is set if an addition or subtraction of signed variables results in an overflow error (i.e., if the magnitude of the sum or difference is too great for the seven LSBs in 2's-complement representation). The overflow flag is also set if a multiplication product overflows one byte or if a division by zero is attempted.</td></tr>
<tr><td>1</td><td>UD</td><td><b>User-definable flag:</b>
This general purpose flag is available to the user</td></tr>
<tr><td>0</td><td>P</td><td><b>Parity Bit:</b>
This bit indicates the parity of the accumulator. It is set if an odd number of bits in the accumulator is set. Otherwise, it is cleared. Not all instructions update the parity bit. The parity bit is set or cleared by instructions that change the contents of the accumulator (ACC, Register R11).</td></tr>
</table>

Figure 144: Program Status Word Register (PSW)

PSW1 (S:D1h)
Program Status Word 1 Register

BIT	7	6	5	4	3	2	1	0
FIELD	CY	AC	N	RS1	RS0	OV	Z	--
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	CY	**Carry Flag:** Identical to the CY bit in the PSW register
6	AC	**Auxiliary Carry Flag:** Identical to the AC bit in the PSW register
5	N	**Negative Flag:** This bit is set if the result of the last logical or arithmetic operation was negative (i.e. bit 15 = 1). Otherwise it is cleared.
4:3	RS1:0	**Register Bank Select Bits 0 and 1:** Identical to the RS1:0 bits in the PSW register
2	OV	**Overflow Flag:** Identical to the OV bit in the PSW register
1	Z	**Zero Flag:** This flag is set if the result of the last logical or arithmetic operation is zero. Otherwise it is cleared.
0	--	**Reserved**. The value read from this bit is 0.

Figure 145: Program Status Word 1 Register (PSW1)

PWMC (S:A2h)
PWM Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	DCR	ENP	PPS5	PPS4	PPS3	PPS2	PPS1	PPS0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	DCR	**Duty Cycle resolution** Set to activate the 10-bit resolution mode (High resolution) Clear to activate the 8-bit resolution mode is selected (Standard resolution).
6	ENP	**Enable PWM output** Set to activate PWM output. Clear to de-activate PWM When the bit ENP is cleared, the user can change the pre-scalar, and then the period of the pulse width modulation output is modified. The period can be changed at each cycle of clock. The way to configure the output period is: - Clear the bit ENP - Write the value of the pre-scalar (bit 5:0 of the register PWMC) - Set the bit ENP.
5:0	PPS[5:0]	**PWM Pre-scalar** This field is used to set the repetition rate of the square wave available at output PWM.

Figure 146: PWM control Register (PWMC)

PWMDCLSB (S:A3h)

PWM Duty Cycle Less Significant Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**PWM Duty Cycle Less Significant Byte** Bit 7:0 of PWM duty cycle register

Figure 147: PWM Duty Cycle LSB Register (PWMDCLSB)

PWMDCMSB (S:A4h)

PWM Duty Cycle Most Significant Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**PWM Duty Cycle Most Significant Byte** Bit 15:8 of PWM duty cycle register

Figure 148: PWM Duty Cycle LSB Register (PWMDCLSB)

RCAP2H (S:CBh)

Timer 2 Relod/Capture High Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**High Byte of Timer 2 Reload/Capture**

Figure 149: Timer 2 Reload/capture High Byte Register (RCAP2H)

RCAP2L (S:CAh)

Timer 2 Relod/Capture Low Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Low Byte of Timer 2 Reload/Capture**

Figure 150: Timer 2 Reload/capture Low Byte Register (RCAP2L)

SADDR (S:A9h)
Slave Individual Address Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Slave individual address**

Figure 151: UART Slave Individual Address Register (SADDR)

SADEN (S:B9h)
Slave Address Mask Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Mask data for slave individual address**

Figure 152: UART Slave Individual Address Mask Register (SADEN)

SBUF (S:99h)
Serial Buffer

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Data sent/received bu serial I/O port**

Figure 153: UART Serial Buffer Register (SBUF)

SCON (S:98h)

Serial control Register

BIT	7	6	5	4	3	2	1	0
FIELD	FE SM0	SM1	SM2	REN	TB8	RB8	TI	RI
RESET	0000 0000b							

<table>
<tr><th>Bit Number</th><th>Bit Mnemonic</th><th>Function</th></tr>
<tr><td>7</td><td>FE

SM0</td><td><b>Framing Error Bit:</b>
To select this function, set the PCON.SMOD0 bit.
FE is set by hardware to indicate an invalid stop bit and cleared by software. FE is not cleared by valid frames

<b>Serial port mode bit 0</b>
To select this function, clear the PCON.SMOD0 bit.
Software writes to SM1 and SM0 to select the serial port’ operating mode.</td></tr>
<tr><td>6</td><td>SM1</td><td><b>Serial port mode bit 1</b>
<table>
<tr><th>SM0</th><th>SM1</th><th>Mode</th><th>Description</th><th>Baud rate</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Shift register</td><td>Clk/12</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>8 bit UART</td><td>Variable</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>9 bit UART</td><td>Clk/32 or Clk/64</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>9 bit UART</td><td>Variable</td></tr>
</table></td></tr>
<tr><td>5</td><td>SM2</td><td><b>Serial port mode bit 2</b>
Software writes to bit SM2 to enable or disable the multiprocessor communication and automatic address recognition features.</td></tr>
<tr><td>4</td><td>REN</td><td><b>Receiver Enable Bit</b>
Set for reception, clear for transmission</td></tr>
<tr><td>3</td><td>TB8</td><td><b>Transmit bit 8</b>
In mode 2 and 3, software writes the ninth data bit to be transmitted to TB8. Not used in mode 1 and 0</td></tr>
<tr><td>2</td><td>RB8</td><td><b>Receiver bit 8. (Not used in mode 0)</b>
Set or cleared by hardware to reflect the stop bit in mode 1. SM2 must be cleared
Set or cleared by hardware to reflect the ninth bit in mode 2 & 3. SM2 must be set.</td></tr>
<tr><td>1</td><td>TI</td><td><b>Transmit interrupt flag</b>
Set by the transmitter after the last data bit transmitted. Cleared by software</td></tr>
<tr><td>0</td><td>RI</td><td><b>Receive interrupt flag</b>
Set by the receiver after the last data bit of a frame has been received. Cleared by software</td></tr>
</table>

Figure 154: UART Serial Port control Register (SCON)

SIEN0 (S:D6h)
I2CS Interrupt Enable Register 0

BIT	7	6	5	4	3	2	1	0
FIELD	--	EGC	--	--	--	ESUNF	ESOVF	ESNE
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved** The value read from this bit is indeterminate.
6	EGC	**I2CS Enable General Call interrupt (SSTAT0.SGC)** Clear to disable SSTAT0.SGC bit to generate an interrupt request Set to enable SSTAT0.SGC bit to generate an interrupt request
5	--	**Reserved** The value read from this bit is indeterminate.
4	--	**Reserved** The value read from this bit is indeterminate.
3	--	**Reserved** The value read from this bit is indeterminate.
2	ESUNF	**I2CS Underflow Interrupt enable bit** Clear to disable SSTAT0.SUNF bit to generate an interrupt request Set to enable SSTAT0.SUNF bit to generate an interrupt request
1	ESOVF	**I2CS Overflow Interrupt enable bit** Clear to disable SSTAT0.SOVF bit to generate an interrupt request Set to enable SSTAT0.SOVF bit to generate an interrupt request
0	ESNE	**I2CS Normal End Interrupt enable bit** Clear to disable SSTAT0.SNE bit to generate an interrupt request Set to enable SSTAT0.SNE bit to generate an interrupt request

Figure 155: I2CS Interrupt Enable register 0 (SIEN0)

SIEN1 (S:D5h)
I2CS Interrupt Enable Register 1

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	ESBE	--	ESTBF	ESRBE	--	ESRBF
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved** The value read from this bit is indeterminate.
6	--	**Reserved** The value read from this bit is indeterminate.
5	ESTBE	**I2CS Transmission Buffer Empty Interrupt enable bit** Clear to disable SSTAT1.STBE bit to generate an interrupt request Set to enable SSTAT1.STBE bit to generate an interrupt request
4	--	**Reserved** The value read from this bit is indeterminate.
3	ESTBF	**I2CS Transmission Buffer Full Interrupt enable bit** Clear to disable SSTAT1.STBF bit to generate an interrupt request Set to enable SSTAT1.STBF bit to generate an interrupt request
2	ESRBE	**I2CS Reception Buffer Empty Interrupt enable bit** Clear to disable SSTAT1.SRBE bit to generate an interrupt request Set to enable SSTAT1.SRBE bit to generate an interrupt request
1	--	**Reserved** The value read from this bit is indeterminate.
0	ESRBF	**I2CS Reception Buffer Full Interrupt enable bit** Clear to disable SSTAT1.SRBF bit to generate an interrupt request Set to enable SSTAT1.SRBF bit to generate an interrupt request

Figure 156: I2CS Interrupt Enable register 1 (SIEN1)

SP (S:81h)

Stack Pointer Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0111b							

Bit Number	Bit Mnemonic	Function
7:0		**Stack Pointer Register Low** Bits 7:0 of the extended stack pointer SPX (DR60)

SP provides SFR accesses to location 63 in the register file (also names SP)

Figure 157: Stack Pointer Register (SP)

SPH (S:BEh)

Stack Pointer Register High

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Stack Pointer Register High** Bits 15:8 of the extended stack pointer SPX (DR60)

SP provides SFR accesses to location 62 in the register file (also names SPH)

Figure 158: Stack Pointer Register High (SPH)

SPCR (S:B1h)

SPI Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0
RESET	0000 0100b							

Bit Number	Bit Mnemonic	Function
7	SPIE	**SPI Interrupt Enable** When clear, SPI Interrupts are inhibited (interrupt_n = ’1’) When set, An SPI interrupt is generated (interrupt_n = ’0’) if SPIF = ’1’ or MODF = ’1’ in the SPSR register.
6	SPE	**SPI System Enable** When clear, SPI system is off. When set, SPI system is on.
5	SPR2	**SPI Baud Rate Select bit 2**
4	MSTR	**Master/Slave mode Select** When clear, SPI system is configured as a slave. When set, SPI system is configured as a master
3	CPOL	**Clock Polarity Select** When clear, the serial clock idles low - Active high clock selected. When set, the serial clock idles high - Active low clock selected.
2	CPHA	**Clock Phase Select** When clear, the first clock transition is the first capture edge When set, the second clock transition is the first capture edge
1:0	SPR1:0	**SPI Baud Rate Select bit 1:0**

SPR2	SPR1	SPR0	Fsck
0	0	0	Fcpu /8
0	0	1	Fcpu /8
0	1	0	Fcpu /16
0	1	1	Fcpu /32
1	0	0	Fcpu /64
1	0	1	Fcpu /128
1	1	0	Fcpu /256
1	1	1	Fcpu /512

Figure 159: SPI control register (SPCR)

SPDR (S:B2h)
SPI Data Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	**Bit Mnemonic**	**Function**
7:0		SPI receive/transmit data

Figure 160: SPI Data register (SPDR)

SPSR (S:B3h)
SPI Status Register

BIT	7	6	5	4	3	2	1	0
FIELD	SPIF	WCOL	--	MODF	--	--	--	--
RESET	0000 0000b							

Bit Number	**Bit Mnemonic**	**Function**
7	SPIF	**SPI Transfer Complete Flag** This flag is set by hardware at the end of an SPI transfer. To clear the SPIF bit, a read to the SPSR register (while SPIF = ’1’) followed by a read or a write to the SPDR register is required. If a transfer is in progress during the second step, and if this second step is a write to the SPDR register, a write collision is generated and the writing to SPDR register is a failure.
6	WCOL	**Write Collision Error Flag** This flag is set by hardware if there is a write access to the SPDR register while a transfer is in progress. To clear the WCOL bit, a read to the SPSR register (while WCOL = ’1’) followed by a read or a write to the SPDR register is required.
5	--	**Reserved** The value read from these bits is indeterminate
4	MODF	**Mode Fault Error Flag** This flag is set by hardware if the SS_N pin goes low while the SPI system is configured as a master. To clear the MODF bit, a read to the SPSR register (while MODF = ’1’) followed by a write to the SPCR register is required.
3:0		**Reserved** The value read from these bits is indeterminate

Figure 161: SPI status register (SPSR)

SRXBUF (S:F2h) Read only
I2CS Receive Buffer

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Data received by I2CS**

Figure 162: I2CS reception Buffer (SRXBUF)

SSADDR (S:D7h)
I2CS Self Address Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	SADDR						
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved** The value read from this bit is indeterminate
6:0	SADDR	**7-bit Self Address** This register must be written before the beginning of an I2C transaction.

Figure 163: I2CS Self Address register (SSADDR)

SSTAT0 (S:E5h) Read only

I2CS Status Register 0

BIT	7	6	5	4	3	2	1	0
FIELD	--	GC	--	--	--	SUNF	SOVF	SNE
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved** The value read from this bit is indeterminate.
6	GC	**General call** Set to indicates that a general call has been detected. It can be cleared by software.
5:3	--	**Reserved** The value read from these bits is indeterminate.
2	SUNF	**Transmission underflow** Transmitted Data Byte not ready (Transmission Buffer is empty) while a new data byte needs to be sent. A wait state is generated until data is available. It can be cleared by software.
1	SOVF	**Reception overflow** Received data byte could not be written (Reception Buffer is full) while a new bit was received. Set to 1 when Rx overflows and STCON.SWS =0. It indicates that receive Buffer is full while receiving a new byte. A Not Acknowledge is sent If STCON.SWS =1 when a new byte is received, a wait state is generated and OVF is not set. It can be cleared by software.
0	SNE	**Normal End (End of access with no error)** Set when a stop is sent at the end of a successful access. Clear automatically when a new I2C access starts. It can also be cleared by software

When GC, SUNF or SOVF flags have been set, reception and transmission process are disabled until the CPU reads SSTAT0 register. This read operation automatically clears these flags.

These interrupt sources can all be individually enabled/disabled by SIEN0 register. The *OTXRXINT* output signal is set to '1' when one or several interrupt sources are active and enabled. When a disabled interrupt occurs, *OTXRXINT* remains unchanged, but the corresponding interrupt bit is set.

When a general call is detected, the Slave controller sets SSTAT0.GC to ‘1’. The CPU has to handle received data as General Call information.

Figure 164: I2CS Status Register 0 (SSTAT0)

SSTAT1 (S:F6h) Read only

I2CS Status Register 1

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	STBE	--	STBF	SRBE	--	SRBF
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved** The value read from this bit is indeterminate.
6	--	**Reserved** The value read from this bit is indeterminate.
5	STBE	**I2CS Transmission buffer is empty** Set to 1 when Transmit Buffer is empty. Clear to 0 when at least one Byte is ready for data transmission This flag is cleared when the CPU performs a write access to STXBUF register.
4	--	**Reserved** The value read from this bit is 0.
3	STBF	**I2CS Transmission buffer is full** Set to 1 when Transmission Buffer is full. When set, no more write operation into transmission buffer or memory is performed (CPU write request to STXBUF is not taken in account). Clear to 0 when the transmission buffer is empty This flag is cleared when a new Data Byte is requested by the I2C transfer controller.
2	SRBE	**I2CS Reception buffer is empty** Set to 1 when reception Buffer is empty. No more write operation into reception buffer or memory is performed (CPU read request to SRXBUF not taken in account). This flag is cleared when a new Data Byte is received by the TXRX controller. Clear to 0 when at least one received Data Byte is available.
1	--	**Reserved** The value read from this bit is indeterminate.
0	SRBF	**I2CS Reception buffer is full** Set to 1 when reception Buffer is full. This flag is cleared when the CPU performs a read operation to SRXBUF register. Clear to 0 when reception Buffer is empty

These interrupt sources can all be individually enabled/disabled by SIEN1 register.
The *OFIFOINT* output signal is set to ‘1’ when one or several interrupt sources are active and enabled. When a disabled interrupt occurs, *OFIFOINT* remains unchanged, but the corresponding interrupt bit is set. These interrupt sources are cleared when the condition which has set them disappears.

Figure 165: I2CS Status Register 1 (SSTAT1)

STCON (S:F1h)

I2CS Transfer Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	I2CEN	SWS	--	--	--	--	TIG
RESET	00000 0000b							

Bit Number	Bit Mnemonic	Function
7	--	**Reserved.** The value read from this bit is indeterminate.
6	I2CEN	**I2CS enable.** Set to activate FlipI2CS (I2CS responds to calls to its slave address and to the general call.) Clear to deactivate FlipI2CS (does not respond to any call through I2C bus)
5	SWS	**I2CS Wait State** Set to generate wait state on SCL line when RX overflows. When clear, FlipI2CS sends a "not acknowledge" to stop the transmission when RX overflows.
4:1	--	**Reserved.** The value read from these bits is indeterminate.
0	TIG	**Transfer In Progress.** Set to 1 by hardware when an I2C transfer is in progress on the I2C bus. Clear otherwise

Figure 166: I2CS Transfer Control Register (STCON)

STXBUF (SFE3h) Write only

I2CS Transmit Buffer

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Data transmitted by I2CS**

Figure 167: I2CS transmission Buffer (STXBUF)

TCON (S:88h)

Timer Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	TF1	TR1	TF0	TR0	IE1_	IT1	IE0_	IT0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	TF1	**Timer 1 overflow flag** Set by hardware when the timer 1 overflows. Cleared by hardware when the processor vectors to the interrupt routine
6	TR1	**Timer 1 run control bit** Set/cleared by software to turn timer 1 on/off
5	TF0	**Timer 0 overflow flag** Set by hardware when the timer 0 overflows. Cleared by hardware when the processor vectors to the interrupt routine
4	TR0	**Timer 0 run control bit** Set/cleared by software to turn timer 0 on/off
3	IE1_	**External interrupt 1 edge flag. Hardware controlled** Set when external interrupt 1 is detected. Cleared when interrupt is processed.
2	IT1	**External interrupt 1 signal type control bit.** Set to specify External interrupt 1 as falling edge triggered. Cleared to specify External interrupt 1 as low level triggered.
1	IE0_	**External interrupt 0 edge flag. Hardware controlled** Set when external interrupt 0 is detected. Cleared when interrupt is processed
0	IT0	**External interrupt 0 signal type control bit.** Set to specify External interrupt 0 as falling edge triggered. Cleared to specify External interrupt 0 as low level triggered.

Figure 168: Timer/Counter 0&1 control Register (TCON)

TH0 (S:8Ch)

Timer 0 High Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**High Byte of timer 0**

Figure 169: Timer 0 High Byte Register (TH0)

TH1 (S:8Dh)

Timer 1 High Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**High Byte of timer 1**

Figure 170: Timer 1 High Byte Register (TH1)

TH2 (S:CDh)

Timer 2 High Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**High Byte of timer 2**

Figure 171: Timer 2 High Byte Register (TH2)

TL0 (S:8Ah)
Timer 0 Low Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	**Bit Mnemonic**	**Function**
7:0		**Low Byte of timer 0**

Figure 172: Timer 0 High Byte Register (TL0)

TL1 (S:8Bh)
Timer 1 Low Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	**Bit Mnemonic**	**Function**
7:0		**Low Byte of timer 1**

Figure 173: Timer 1 Low Byte Register (TL1)

TL2 (S:CCh)
Timer 2 Low Byte Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	**Bit Mnemonic**	**Function**
7:0		**Low Byte of timer 2**

Figure 174: Timer 2 Low Byte Register (TL2)

TMOD (S:89h)

Timer Mode Register

BIT	7	6	5	4	3	2	1	0
FIELD	GATE1	CT1	M11	M01	GATE0	CT0	M10	M00
RESET	0000 0000b							

<table>
<tr><th>Bit Number</th><th>Bit Mnemonic</th><th>Function</th></tr>
<tr><td>7</td><td>GATE1</td><td><b>Timer 1 gate</b>
When clear, run control bit TR1 gates the input signal to the timer register.
When set and TR1=1, external input timer1gate gates the timer input.</td></tr>
<tr><td>6</td><td>CT1</td><td><b>Timer 1 Counter/timer select</b>
When CT1=0, timer 1 counts the divided down system clock
When CT1=1, timer 1 counts negative transition on timer1 input pin</td></tr>
<tr><td>5:4</td><td>M11, M01</td><td><b>Timer 1 mode select</b>
<table>
<tr><th>M11</th><th>M01</th><th>Mode</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>13 bit counter</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>16 bit counter</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>8 bit auto-reload counter</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>Timer 1 halted, retains count</td></tr>
</table></td></tr>
<tr><td>3</td><td>GATE0</td><td><b>Timer 0 gate</b>
When clear, run control bit TR0 gates the input signal to the timer register.
When set and TR0=1, external input timer0gate gates the timer input.</td></tr>
<tr><td>2</td><td>CT0</td><td><b>Timer 0 Counter/timer select</b>
When CT0=0, timer 0 counts the divided down system clock
When CT0=1, timer 0 counts negative transition on timer0 input pin</td></tr>
<tr><td>1:0</td><td>M10, M00</td><td><b>Timer 0 mode select</b>
<table>
<tr><th>M10</th><th>M00</th><th>Mode</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>13 bit counter</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>16 bit counter</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>8 bit auto-reload counter</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>Two 8 bit counter</td></tr>
</table></td></tr>
</table>

Figure 175: Timer/Counter 0&1 mode select Register (TMOD)

T2CON (S:C8h)

Timer 2 Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7	TF2	**Timer 2 overflow flag** Set by hardware when the timer 2 overflows. Must be cleared by software *Note: TF2 is not set if RCLK=1 or TCLK=1*
6	EXF2	**Timer 2 external flag** Set by hardware (if EXEN2=1) when a negative transition on timer2capt is detected. Must be cleared by software *Note: EXF2 is not set if DCEN=1*
5	RCLK	**Receive clock bit.** If set, baud rate generator for the serial port 1 and 3 uses timer 2'overflow for its reception clock. If clear, it uses timer 1.
4	TCLK	**Transmit clock bit** If set, baud rate generator for the serial port 1 and 3 uses timer 2 overflow for its transmission clock. If clear, it uses timer 1.
3	EXEN2	**Timer 2 external enable bit** If set, enable a capture or a reload to occur as a result of a negative transition on timer2capt (if timer 2 is not being used to clock the serial port). If clear, timer 2 ignores events on timer2capt.
2	TR2	**Timer 2 run control bit** Set to start timer 2 running. Clear to stop the timer 2.
1	C/T2#	**Timer 2 counter/timer select** Set for counter operation: timer2 counts the negative transition on external pin timer2. Clear for timer operation: timer 2 counts the divided system clock.
0	CP/RL2#	**Capture reload bit** Set to capture on negative transitions on timer2capt if EXEN2=1. Clear to auto-reload on timer 2 overflow or negative transition on timer2capt if EXEN2=1. *Note: CP/RL2# is ignored and timer 2 is forced to auto reload on timer 2 overflow if RCLK=1 or TCLK=1.*

Figure 176: Timer/Counter 2 control Register (T2CON)

T2MOD (S:C9h)
Timer2 Mode Register

BIT	7	6	5	4	3	2	1	0
FIELD	--	--	--	--	--	--	T2OE	DCEN
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:2	--	**Reserved** The value read from these bits is indeterminate
1	T2OE	**Timer 2 output enable** In clock out mode, enables the programmable clock output
0	DCEN	**Down count Enable bit** If clear, configure timer 2 as an up counter. If set, configure timer 2 as an up/down counter.

Figure 177: Timer/Counter 2 mode select Register (T2MOD)

WDTRST (S:A6h)
Watchdog Timer Reset Register

BIT	7	6	5	4	3	2	1	0
FIELD								
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:0		**Watchdog timer control data**

Figure 178: WDT Reset Register (WDTRST)

WDTCON (S:A5h)
Watchdog Timer Control Register

BIT	7	6	5	4	3	2	1	0
FIELD	WDT3	WDT2	WDT1	WDT0	WDTR	WOV2	WOV1	WOV0
RESET	0000 0000b							

Bit Number	Bit Mnemonic	Function
7:4	WDT[3:0]	**4 less significant bits of the 20-bit watchdog timer** Read only
3	WRUN	**WDT run control bit** Read only
2:0	WOV[2:0]	**WDT Overflow control bits** When all three bits are set to 1, the watchdog timer has a nominal period of 1024 K clock cycles (20-bit counter). When all three bits are cleared to 0, the time-out period is 8 K clock cycles (13-bit counter)

Figure 179: WDT control Register (WDTCON)

REVISION HISTORY

Major release: Ri.0
Minor release: Ri.j with j≠0

Release number	Date	Who	Description of the modification	Frame release number
R1.2	October 7th, 2010	OLM	we_n signal renamed rwn Update of timing diagrams of accesses with wait states insertion	1.0
R1.1	September 30th, 2010	DMA	Completed Memory Interface description Update of Instruction timings	1.0
R1.0	September 7th, 2010	JHA	First version of the specification of the Flip80251-Hurricane	1.0