



Fraunhofer Institut
Integrierte Schaltungen

***CorePool* FHG8051**

Databook

- subject to change without notice -

Version: v1.5

Date: 02.08.02

Document History

Table 1: Document History

Version	Date	Responsible	Description
v1.0	12.01.97	Shn	Generic datasheet
v1.1	17.02.97	Shn	Add timing diagrams
v1.2	29.02.97	Shn	Detail register description
v1.3	24.04.97	Shn	Enhance interrupt unit
v1.4	14.11.97	Shn	Detail timing diagrams
v1.5	20.01.98	Shn	Detail interrupt behaviour, update document structure

Contact

CorePool
Fraunhofer Institute Integrated Circuits

Am Wolfsmantel 33
91058 Erlangen
Germany

Phone: +49 (0) 9131 776 777
Fax: +49 (0) 9131 776 499
Email: info@corepool.com
Internet: <http://www.corepool.com>

Table of Contents

Document History	2
Contact	2
Purpose	7
Features	7
Design Kit	7
Requirements	7
References	8
Block Diagram	8
Signal Description	9
General Information	9
Parameter Description	10
VHDL Usage trough Component Instantiation	13
Verilog Usage trough Component Instantiation	16
Architecture	18
Functional Description	20
Control Unit	27
Interfaces	28
Waveforms and timing tables	29
Instruction Set	37
Opcode list	39
Application notes	48

List of Tables and Figures

Document History	2
Block Diagram	8
Signal Description	9
Parameter Description	10
Architecture	18
Internal Memory Map	20
Program Status Word	21
SFR Memory Map	21
Configuration Register	22
Register Bank Selector Bits	22
Power Control Register	23
Interrupt Priority Register	24
Interrupt Enable Register	25
Interrupt Request Sequence	25
Reset Sequence	27
Waveform of program memory read access	29
Timing table of program memory read access	29
Waveform of program memory write access	30
Timing table of program memory write access	30
Waveform of external data memory read access	31
Timing table of external data memory read access	31
Waveform of external data memory write access	32
Timing table of an external data memory write access	32
Waveform of internal data memory read and write accesses	33
Timing table of internal data memory read and write accesses	33
Waveform of SFR read and write accesses	34
Timing table of SFR read and write accesses	34
Waveform of idle mode	35
Timing table of idle mode	35

List of Tables and Figures (cont)

Arithmetic Instructions 37

Logical Instructions 37

Data Transfer Instructions 37

Bit Manipulation Instructions 38

Program Control Instructions 38

Opcode List 40

Application Example 48

Purpose

The FHG8051 is a high performance, opcode compatible core version of the industry standard 8051 micro controller for ASIC and FPGA implementations. Its parameter set enables the designer to enhance the cpu performance and to tailor the architecture to the applications needs.

Features

- 8-Bit Synthesizeable Microcontroller Core
- Opcode and Cycle Equivalent to Industry Standard 8051
- Fully Static, Microcode Free Design
- Up to 64K Bytes Program Memory Address Space
- Up to 64K Bytes Data Memory Address Space
- Up to 256 Bytes Internal Data Memory
- Up to 128 Special Function Registers (SFR)
- Idle, Power Down Mode
- Programmable Clock and Wait State Generator
- Optional up to 7 Interrupt Sources in 2 Priority Levels or No Interrupt Unit
- Optional True 8051 Cycle Operation or Reduced Cycle Mode for Enhanced Performance
- Optional PCON, P1, P3 or Second DPTR Register
- Optional Area Reducing NOP Behaviour for MUL, DIV, DA Opcodes
- Optional Program Memory Write Mode

Design Kit

- Technology Independent Implementation as Synopsys Design Ware Components
- VHDL/Verilog Simulation Models
- VHDL/Verilog 8051 Compliance Test Suite
- Auxiliary Simulation Models for User Testbenches
- Synthesis and Testsynthesis Scripts
- Example Design and Testchip available
- Design Support, Netlist Synthesis Service and Consulting available

Requirements

Simulation

- VHDL IEEE 1076 Simulator (Synopsys VSS, Modeltech VSIM, Vantage, others)
- Verilog IEEE Simulator (Cadence VerilogXL, Modeltech VSIM, others)

Synthesis

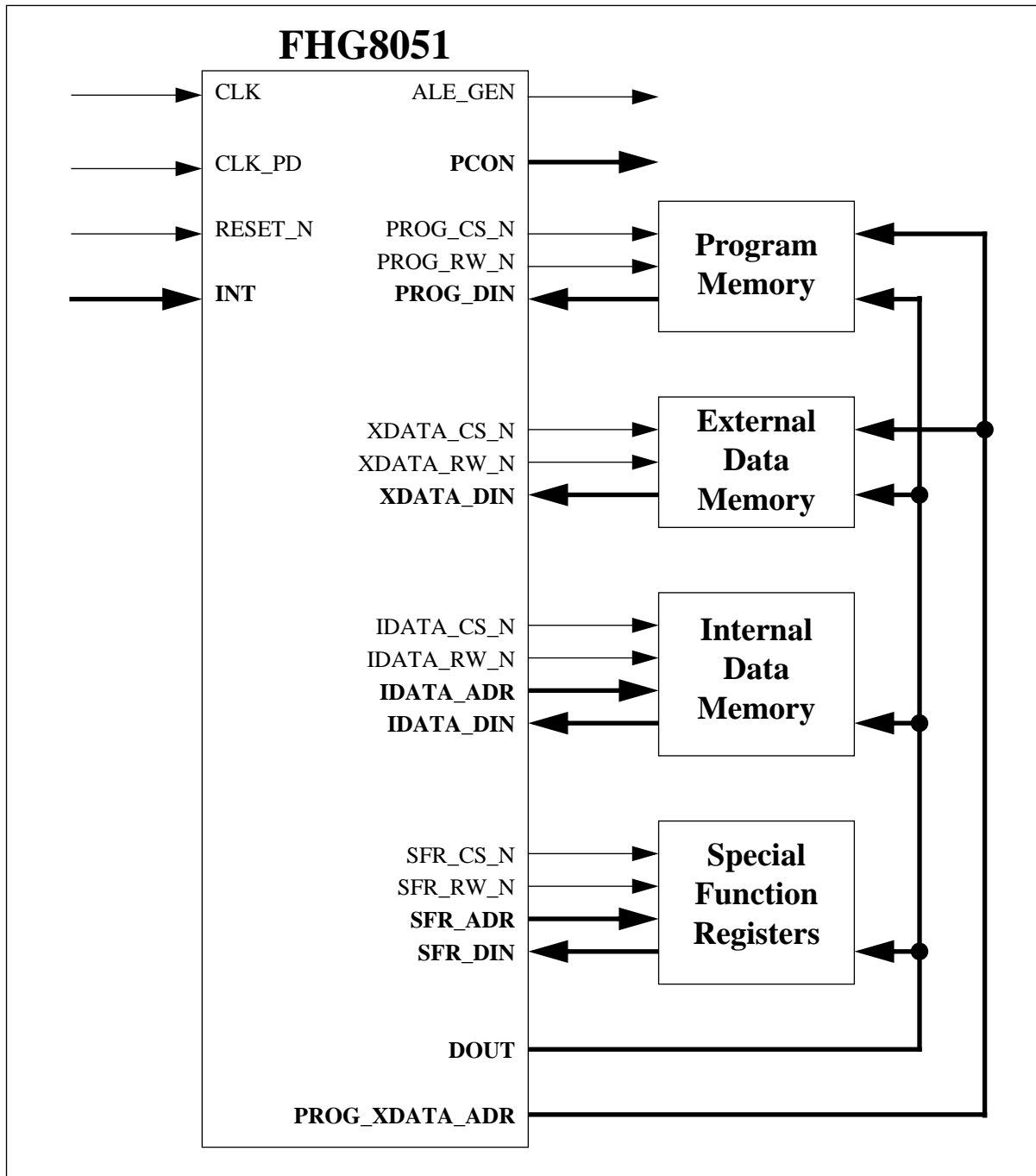
- Synopsys Design Compiler

References

For more details about the 8051 cpu refer to an industry standard databook about this micro controller architecture.

Block Diagram

Figure 1: Block Diagram



Signal Description

Table 2: Signal Description

Pin	Direction	Bitwidth	Description
CLK	IN	1	Systemclock.
CLK_PD	IN	1	Power down systemclock.
RESET_N	IN	1	Low activ asynchronous reset.
INT	IN	7	High active Interrupt bus for 7 independent interrupt sources.
ALE_GEN	OUT	1	High active address latch enable.
PCON	OUT	8	High active signal for power down control.
PROG_CS_N	OUT	1	Low active chip select for program memory.
PROG_RW_N	OUT	1	Low active read/write for program memory. High level indicates a read, low level a write operation.
PROG_DIN	IN	8	Input databus bus from program memory.
PROG_XDATA_ADR	OUT	16	Addressbus for external data memory and program memory.
XDATA_CS_N	OUT	1	Low active chip select for external memory.
XDATA_RW_N	OUT	1	Low active read/write for the external memory. High level indicates a read, low level a write operation.
XDATA_DIN	IN	8	Input databus from external memory.
IDATA_CS_N	OUT	1	Low active chip select for internal memory.
IDATA_RW_N	OUT	1	Low active read/write for internal memory. High level indicates a read, low level a write operation.
IDATA_ADR	OUT	8	Address bus for the internal memory.
IDATA_DIN	IN	8	Input databus from internal memory.
SFR_CS_N	OUT	1	Low active chip select for SFR memory.
SFR_RW_N	OUT	1	Low active read/write for SFR memory. High level indicates a read, low level a write operation.
SFR_ADR	OUT	7	Addressbus for SFR memory.
SFR_DIN	IN	8	Input databus from SFR memory.
DOUT	OUT	8	Common output databus to program memory (optional), external memory, internal memory and SFR memory.

General Information

External data memory and internal data memory means only the memory areas accessed by the 8051 instructions. It does not mean any onchip or offchip implementation of the memories. This is free to the designers needs.

Parameter Description

Tabelle 3: Parameter Description

Name	Type	Default Value	Value Range	Description
CYCLE_REDUCTION	Integer	0	{0,1}	Speed up instruction execution
IMPLEMENT_INT	Integer	5	{0...7}	Implement interrupt logic
IMPLEMENT_MUL	Integer	1	{0,1}	Implement logic for multiplication opcode
IMPLEMENT_DIV	Integer	1	{0,1}	Implement logic for division opcode
IMPLEMENT_DA	Integer	1	{0,1}	Implement logic for Decimal adjust opcode
IMPLEMENT_DPTR2	Integer	0	{0,1}	Implement second DPTR register
IMPLEMENT_PCON	Integer	1	{0...8}	Implement (larger) PCON register
IMPLEMENT_P1	Integer	0	{0,1}	Implement additional P1 register
IMPLEMENT_P3	Integer	0	{0,1}	Implement additional P3 register
IMPLEMENT_PWR	Integer	0	{0,1}	Implement program memory write function for MOVX opcode

The high speed architecture of the FHG8051 executes every instruction cycle in 6 clock cycles instead of 12 in a standard 8051. The parameter `CYCLE_REDUCTION` allows to speed up a set of accelerateable instructions from 2 instruction cycles to 1 additionally. Please refer to the opcode list and its comments to identify these instructions. By default `CYCLE_REDUCTION` is set to “0“ and the classic 2 instruction cycle execution will be built. When `CYCLE_REDUCTION` is set to “1“ the set of accelerateable instructions will execute in 1 instruction cycle and increase performance. Use this parameter to speed up execution if you don’t need a cycle identical behaviour of your existing assembler code.

The FHG8051 has an interrupt control unit which allows up to 7 different interrupt sources. By default parameter `IMPLEMENT_INT` ist set to “5“ like in the classic 8051 implementation, and all necessary logic for the interrupt unit will be built. When `IMPLEMENT_INT` ist set to “0“ no interrupt logic is built. Use this parameter if you want to reduce chip size and don’t need any interrupts.

The parameter `IMPLEMENT_MUL` enables the designer to reduce chip area by not implementing the necessary logic for the multiplication instruction `MUL`. By default `IMPLEMENT_MUL` is set to “1“ and the multiplication logic will be built. When `IMPLEMENT_MUL` is set to “0“ no logic will be built and the behaviour of `MUL` is like a `NOP` instruction. Use this parameter if you don’t need the multiplication instruction.

The parameter `IMPLEMENT_DIV` enables the designer to reduce chip area by not implementing the necessary logic for the division instruction `DIV`. By default `IMPLEMENT_DIV` is set to “1” and the division logic will be built. When `IMPLEMENT_DIV` is set to “0” no logic will be built and the behaviour of `DIV` is like a `NOP` instruction. Use this parameter if you don’t need the division instruction.

The parameter `IMPLEMENT_DA` enables the designer to reduce chip area by not implementing the necessary logic for the decimal adjust instruction `DA`. By default `IMPLEMENT_DA` is set to “1” and the decimal adjust instruction logic will be built. When `IMPLEMENT_DA` is set to “0” no logic will be built and the behaviour of `DA` is like a `NOP` instruction. Use this parameter if you don’t need the decimal adjust instruction.

With parameter `IMPLEMENT_DPTR2` the FHG8051 may contain a second `DPTR` register. This is useful e.g. for block transfers using `DPTR` and `DPTR2` as source and destination address registers. The standard `DPTR` registers are located at 82 h (`DPL`) and 83 h (`DPH`). The `DPTR2` registers are located at 84 h (`DPL2`) and 85 h (`DPH2`). To select `DPTR` or `DPTR2` as address source for operations the bit `DPSEL` will be used. It is located at the bit 0 of the `CFG` register at 86 h. If `DPSEL` is set to “0” `DPTR` is used, else `DPTR2`. The instructions affected by this feature are:

```
JMP @A+DPTR
INC DPTR
MOV DPTR, #16-Bit immediate
MOVX A, @DPTR
MOVX @DPTR, A
MOVC A, @A+DPTR
```

By default parameter `IMPLEMENT_DPTR2` is set to “0” and only the standard `DPTR` registers will be built. When `IMPLEMENT_DPTR2` is set to “1” the `DPTR2` registers and the `DPSEL` bit are built. Use this parameter if you want to enhance your address registers.

The parameter `IMPLEMENT_PCON` enables the designer to select the bitwidth of register `PCON` for application specific needs, e.g. power down bits for peripherals or custom units. By default `IMPLEMENT_PCON` is set to “1” and only bit 0 will be built. When `IMPLEMENT_PCON` is set to “0” no register will be built and all bits of output port `PCON` are set to “0”. If `IMPLEMENT_PCON` is set to n ($1 \leq n \leq 8$), n bits of the register will be built starting at bit 0. All not implemented bits of register `PCON` are set to “0” for output port `PCON`. Use this parameter if you need additional control bits for your application.

The classic 8051 has the two address registers `P0` and `P2`. With the parameters `IMPLEMENT_P1` and `IMPLEMENT_P3` additional registers `P1` and `P3` can be built. By default `IMPLEMENT_P1` and `IMPLEMENT_P3` are set to “0” and no register will be built. By setting each parameter to “1” the respective register will be built.

The parameter IMPLEMENT_PWR enables the program memory write function for MOVX instructions. By default IMPLEMENT_PWR is set to “0” and no program memory write function will be implemented. When set to “1” an additional control bit PWR in register CFG at bit 1 is built, controlling the program memory write function. By setting PWR to “0” the regular MOVX operation proceeds. Setting PWR to “1” enables the program memory write function. Any read operation with MOVX will read from the external data memory, but the write operation with MOVX will now write to the program memory. This is very useful for downloading some program code from the external data memory, if a part of the program memory is implemented as a RAM.

VHDL Usage trough Component Instantiation

```
library IEEE, FHG8051DW;
use IEEE.std_logic_1164.all;
use FHG8051DW.FHG8051.all;
use FHG8051DW.FHG8051_CLKGEN.all;

entity MYDESIGN is
  port(
    CLK          : in  std_logic;
    RESET_N     : in  std_logic;
    INT          : in  std_logic_vector(6 downto 0);
    ALE_GEN     : out std_logic;
    PCON        : out std_logic_vector(7 downto 0);
    PROG_CS_N   : out std_logic;
    PROG_RW_N   : out std_logic;
    PROG_XDATA_ADR : out std_logic_vector(15 downto 0);
    PROG_DIN    : in  std_logic_vector(7 downto 0);
    XDATA_CS_N  : out std_logic;
    XDATA_RW_N  : out std_logic;
    XDATA_DIN   : in  std_logic_vector(7 downto 0);
    IDATA_CS_N  : out std_logic;
    IDATA_RW_N  : out std_logic;
    IDATA_ADR   : out std_logic_vector(7 downto 0);
    IDATA_DIN   : in  std_logic_vector(7 downto 0);
    SFR_CS_N   : out std_logic;
    SFR_RW_N   : out std_logic;
    SFR_ADR    : out std_logic_vector(6 downto 0);
    SFR_DIN    : in  std_logic_vector(7 downto 0);
    DOUT       : out std_logic_vector(7 downto 0)
  );

  architecture MYDESIGN_RTL of MYDESIGN is begin
    constant CYCLE_REDUCTION : integer := 1;
    constant IMPLEMENT_INT   : integer := 7;
    constant IMPLEMENT_MUL   : integer := 1;
    constant IMPLEMENT_DIV   : integer := 1;
    constant IMPLEMENT_DA    : integer := 0;
    constant IMPLEMENT_DPTR2 : integer := 1;
    constant IMPLEMENT_PCON  : integer := 1;
    constant IMPLEMENT_P1    : integer := 0;
    constant IMPLEMENT_P3    : integer := 0;
    constant IMPLEMENT_PWR   : integer := 1;

    constant IMPLEMENT_WAIT  : integer := 1;
    constant IMPLEMENT_CLKDIV : integer := 8;
    constant CLKDIV_VALUE    : integer := 0;
    constant CLKDIV_ADR     : integer := 128;
```

```
signal    CLK_PD                : std_logic;
```

```
MYDESIGN_CORE : FHG8051
```

```
generic map(CYCLE_REDUCTION => CYCLE_REDUCTION,  
            IMPLEMENT_INT => IMPLEMENT_INT,  
            IMPLEMENT_MUL => IMPLEMENT_MUL,  
            IMPLEMENT_DIV => IMPLEMENT_DIV,  
            IMPLEMENT_DA => IMPLEMENT_DA,  
            IMPLEMENT_DPTR2 => IMPLEMENT_DPTR2,  
            IMPLEMENT_PCON => IMPLEMENT_PCON,  
            IMPLEMENT_P1 => IMPLEMENT_P1,  
            IMPLEMENT_P3 => IMPLEMENT_P3,  
            IMPLEMENT_PWR => IMPLEMENT_PWR)
```

```
port      map(CLK => CLOCK,  
            CLK_PD => CLOCK_PD,  
            RESET_N => RESET_N,  
            INT => INT,  
            ALE_GEN => ALE_GEN,  
            PCON => PCON,  
            PROG_CS_N => PROG_CS_N,  
            PROG_RW_N => PROG_RW_N,  
            PROG_XDATA_ADR => PROG_XDATA_ADR,  
            PROG_DIN => PROG_DIN,  
            XDATA_CS_N => XDATA_CS_N,  
            XDATA_RW_N => XDATA_RW_N,  
            XDATA_DIN => XDATA_DIN,  
            IDATA_CS_N => IDATA_CS_N,  
            IDATA_RW_N => IDATA_RW_N,  
            IDATA_ADR => IDATA_ADR,  
            IDATA_DIN => IDATA_DIN,  
            SFR_CS_N => SFR_CS_N,  
            SFR_RW_N => SFR_RW_N,  
            SFR_ADR => SFR_ADR,  
            SFR_DIN => SFR_DIN,  
            DOUT => DOUT);
```

```
MYDESIGN_CLKGEN : FHG8051_CLKGEN
  generic map(IMPLEMENT_WAIT => IMPLEMENT_WAIT,
             IMPLEMENT_CLKDIV => IMPLEMENT_CLKDIV,
             CLKDIV_VALUE => CLKDIV_VALUE,
             CLKDIV_ADR => CLKDIV_ADR)
  port      map(CLK => CLK,
               CLK_PD => CLK_PD,
               RESET_N => RESET_N,
               CLK_WAIT => PCON(0),
               CS_N => SFR_CS_N,
               RW_N => SFR_RW_N,
               ADR => SFR_ADR,
               DIN => DOUT,
               DOUT => SFR_DIN);

end MYDESIGN_RTL;
```

Verilog Usage trough Component Instantiation

```
module MYDESIGN (CLK, RESET_N, INT, ALE_GEN, PCON,  
                PROG_CS_N, PROG_RW_N, PROG_XDATA_ADR, PROG_DIN,  
                XDATA_CS_N, XDATA_RW_N, XDATA_DIN,  
                IDATA_CS_N, IDATA_RW_N, IDATA_ADR, IDATA_DIN,  
                SFR_CS_N, SFR_RW_N, SFR_ADR, SFR_DIN,  
                DOUT);  
  
    parameter CYCLE_REDUCTION    = 1;  
                IMPLEMENT_INT    = 7;  
                IMPLEMENT_MUL    = 1;  
                IMPLEMENT_DIV    = 1;  
                IMPLEMENT_DA     = 0;  
                IMPLEMENT_DPTR2  = 1;  
                IMPLEMENT_PCON   = 1;  
                IMPLEMENT_P1     = 0;  
                IMPLEMENT_P3     = 0;  
                IMPLEMENT_PWR    = 1;  
  
                IMPLEMENT_WAIT   = 1;  
                IMPLEMENT_CLKDIV = 8;  
                CLKDIV_VALUE     = 0;  
                CLKDIV_ADR      = 128;  
  
    input        CLK;  
    input        RESET_N;  
    input [6:0]  INT;  
    output       ALE_GEN;  
    output [7:0] PCON;  
    output       PROG_CS_N;  
    output       PROG_RW_N;  
    output [15:0] PROG_XDATA_ADR;  
    input [7:0]  PROG_DIN;  
    output       XDATA_CS_N;  
    output       XDATA_RW_N;  
    input [7:0]  XDATA_DIN;  
    output       IDATA_CS_N;  
    output       IDATA_RW_N;  
    output [7:0] IDATA_ADR;  
    input [7:0]  IDATA_DIN;  
    output       SFR_CS_N;  
    output       SFR_RW_N;  
    output [6:0] SFR_ADR;  
    input [7:0]  SFR_DIN;  
    output [7:0] DOUT;  
  
    wire        CLK_PD;
```

```
FHG8051 #(CYCLE_REDUCTION, IMPLEMENT_INT, IMPLEMENT_MUL,
          IMPLEMENT_DIV, IMPLEMENT_DA, IMPLEMENT_DPTR2,
          IMPLEMENT_PCON, IMPLEMENT_P1, IMPLEMENT_P3,
          IMPLEMENT_PWR)
MYDESIGN_CORE
(CLK, CLK_PD, RESET_N, INT, ALE_GEN,
 PCON, PROG_CS_N, PROG_RW_N, PROG_XDATA_ADR,
 PROG_DIN, XDATA_CS_N, XDATA_RW_N,
 XDATA_DIN, IDATA_CS_N, IDATA_RW_N,
 IDATA_ADR, IDATA_DIN, SFR_CS_N, SFR_RW_N,
 SFR_ADR, SFR_DIN, DOUT);

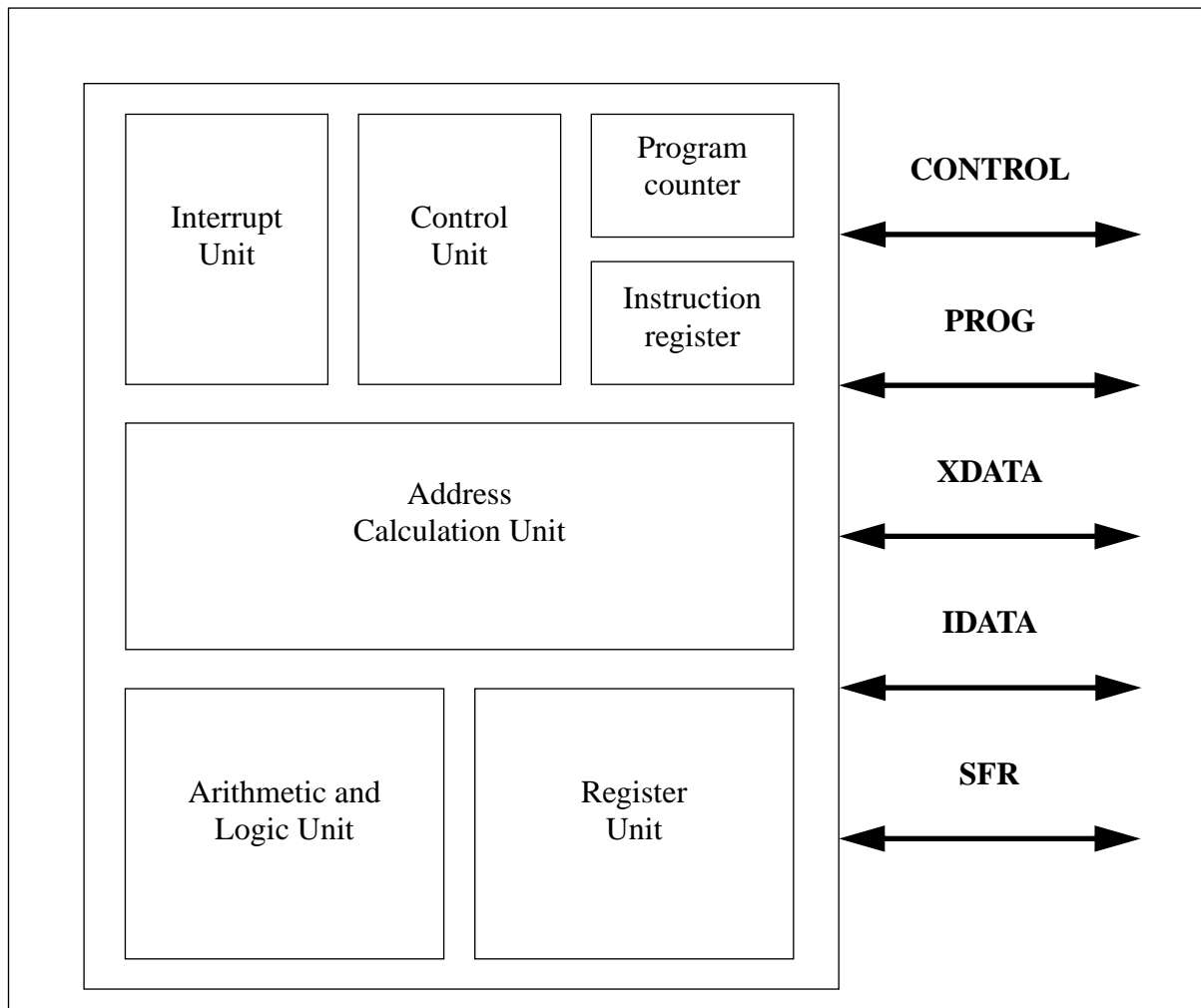
FHG8051_CLKGEN #(IMPLEMENT_WAIT, IMPLEMENT_CLKDIV,
                CLKDIV_VALUE, CLKDIV_ADR)
MYDESIGN_CLKGEN
(CLK, CLK_PD, RESET_N, PCON(0), SFR_CS_N,
 SFR_RW_N, SFR_ADR, DOUT, SFR_DIN);

end module;
```

Architecture

The FHG8051 architecture has seven blocks, the interrupt unit, the control unit, the program counter, the instruction register, the address calculation unit, the arithmetic and logic unit and the register unit.

Figure 2: Architecture



Interrupt Unit.

The interrupt unit is improved and has different behavior compared to the classic 8051. Please refer to the functional description part for details.

Control Unit.

The control unit gets the opcode from the instruction register and creates all internal and external control signals for the FHG8051.

Program Counter.

The PC contains the program counter and the arithmetic unit for the program counter.

Instruction Register.

The instruction register contains the opcode read from the program memory.

Address Calculation Unit.

The address calculation unit contains the address generation for the internal and external data memory.

Arithmetic and Logic Unit.

The ALU contains the arithmetic and logic unit and the temporary registers. The ALU is controlled by the control unit depending on the currently instruction used. Temporary registers inside the ALU are used to hold data for processing.

Register Unit.

In the Register Unit all processor internal registers are placed. These are the accumulator ACC, the B register, the program status register PSW, the port registers P0 to P3, the stackpointer SP, the two data pointers DPTR (DPL, DPH) and DPTR2 (DPL2, DPH2), the configuration register CFG, the power control register PCON and the two interrupt control registers IE and IP.

Functional Description

This part of the documentation shows a general overview and details all parts with different behaviour than an industry standard 8051.

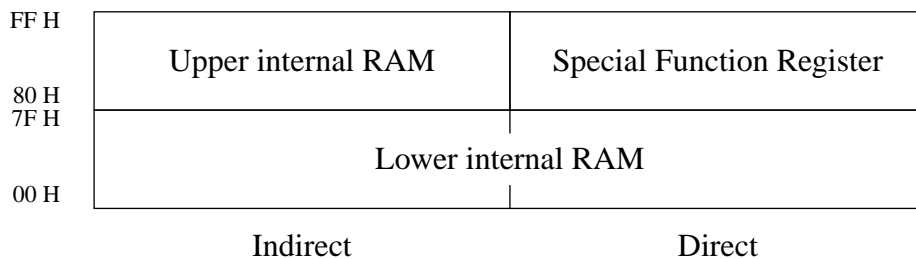
Memory models and addressing modes

The FHG8051 has separate address spaces for program and data memory. The program memory has a maximum size of 64 kbyte. The data memory is divided in an up to 64 kbyte external and internal data memory plus a special function register memory area.

After each reset, the CPU starts execution in the programm memory at location 0000H. Each interrupt has an own start address for its service routine. The highest priore interrupt, the interrupt 0, starts at address 0003H, the second highest at 000BH, the third highest at 0013H, and so on.

The FHG8051 has five different addressing modes: immediate, direct, register, indirect and relative. In the immediate addressing mode the data is contained in the opcode. By direct addressing an eight bit address is part of the opcode, by register addressing, a register is selected in the opcode for the operation. In the indirect addressing mode, a register is selected in the opcode to point to the address used by the operation. The relative addressing mode is used for jump instructions and added to the program counter to achieve the new program address.

Figure 3: Internal Memory Map



The internal memory of the FHG8051 consists of a up to 256 bytes wide data memory plus an up to 128 bytes wide special function register memory area. The picture above shows the memory map. When the lowest 128 bytes of the internal RAM is addressed with direct and indirect address mode the same memory area is accessed. This lower internal RAM consists of four register banks with eight registers each, a bitaddressable segment with 128 bits (16 bytes), and a scratch pad area with 80 bytes. When the highest 128 bytes of the internal memory is addressed with the indirect addressing mode, a 128 byte RAM area is addressed. With the direct addressing mode in the upper internal RAM the special function register memory area is accessed.

Register Unit

The register unit contains all internal registers. These registers are a part of the SFR memory area. The internal registers are the accumulator register ACC on address E0h, the B register on address F0h, the program status register PSW on address D0h, the ports P0 to P3 on the addresses 80h, 90h, A0h and B0h, the stackpointer SP on address 81h, the data pointer DPTR

on addresses 82h - 83h, the second data pointer DPTR2 on addresses 84h - 85h, the configuration register CFG on address 86h, the power control register PCON on address 87h and the two interrupt control registers IE and IP on the addresses A8h and B8h. The register CFG does not exist when both parameters IMPLEMENT_DPTR2 and IMPLEMENT_PWR are set to "0". The registers in the first columns 80h, 88h, 80h, ..., F0h, F8 are bit addressable.

Table 4: SFR Memory Map

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8									CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98									9F
90	P1								97
88									8F
80	P0	SP	DPL	DPH	DPL2	DPH2	CFG	PCON	87

Figure 4: Program Status Word

		MSB				LSB			
		CY	AC	F0	RS1	RS0	OV	GPB	P
Bit	Symbol	Bitaddress	Function						
PSW[7]	CY	D7	Carry flag.						
PSW[6]	AC	D6	Auxiliary carry flag.						
PSW[5]	F0	D5	Flag 0 available to the user for general purpose.						
PSW[4]	RS1	D4	Register bank selector bit 1.						
PSW[3]	RS0	D3	Register bank selector bit 0.						
PSW[2]	OV	D2	Overflow flag.						
PSW[1]	GPB	D1	Useable as a general purpose bit.						
PSW[0]	P	D0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of "one" bits in the accumulator.						

The register bank selector bits RS0 and RS1 enables the working register banks as follows

Table 5: Register Bank Selector Bits

RS1	RS0	Bank	Address
0	0	0	00h - 07h
0	1	1	08h - 0Fh
1	0	2	10h - 17h
1	1	3	18h - 1Fh

The configuration register CFG (shown below) contains the select bit for the second data pointer DPTR2 and the control bit for the program memory write function.

Figure 5: Configuration Register

MSB						LSB	
0	0	0	0	0	0	PWR	DPSEL
Bit	Symbol	Function					
CFG[7]	-	Not used, always zero.					
CFG[6]	-	Not used, always zero.					
CFG[5]	-	Not used, always zero.					
CFG[4]	-	Not used, always zero.					
CFG[3]	-	Not used, always zero.					
CFG[2]	-	Not used, always zero.					
CFG[1]	PWR	Program memory write register bit.					
CFG[0]	DPSEL	Data pointer select.					

The DPSEL bit selects the use of DPTR or DPTR2 registers for addressing. DPSEL set to “0“ means using DPTR, setting it to “1“ DPTR2. If DPSEL is not implemented by parameter IMPLEMENT_DPTR2, it is always zero.

The PWR bit selects the program memory write mode. If PWR is set to “0“ the program memory is accessed as usual as a read only memory. In the other case the program memory can be written by the MOVX instruction. Any read with a MOVX accesses the external memory, but the write with MOVX will now write to the program memory. This is usefull e.g. for program code downloads. If PWR is not implemented by parameter IMPLEMENT_PWR a write with MOVX will always write to the external memory and PWR is always zero.

The power control register PCON contains the idle mode bit IDL and up to 7 other general purpose bits, e.g. for power down mode of peripheral devices. Not implemented bits are set to zero (IMPLEMENT_PCON).

Figure 6: Power Control Register

MSB				LSB			
GP7	GP6	GP5	GP4	GP3	GP2	GP1	IDL
Bit	Symbol	Function					
PCON[7]	GP7	General pupose bit 7.					
PCON[6]	GP6	General pupose bit 6.					
PCON[5]	GP5	General pupose bit 5.					
PCON[4]	GP4	General pupose bit 4.					
PCON[3]	GP3	General pupose bit 3.					
PCON[2]	GP2	General pupose bit 2.					
PCON[1]	GP1	General pupose bit 1.					
PCON[0]	IDL	Idle mode.					

Interrupt Unit

The interrupt unit is improved and has different behaviour compared to the classic 8051. It manages up to seven interrupts and polls them to determine priority if more than one occurs at the same time.

Each of the seven interrupt sources (INT[0]to INT[6]) can be individually programmed to one of two priority levels by setting or clearing a bit in the special function register IP (figure below). A low-priority interrupt can be interrupted by a high priority interrupt, but not by another low priority interrupt. A high priority interrupt can't be interrupted by any other interrupt.

Figure 7: Interrupt Priority Register

		MSB				LSB			
		0	IP6	IP5	IP4	IP3	IP2	IP1	IP0
Bit	Symbol	Bitaddress		Function					
IP[7]	-	BF		Not used, always zero.					
IP[6]	IP6	BE		Priority level of interrupt 6. IP6 = 1 is high priority.					
IP[5]	IP5	BD		Priority level of interrupt 5. IP5 = 1 is high priority.					
IP[4]	IP4	BC		Priority level of interrupt 4. IP4 = 1 is high priority.					
IP[3]	IP3	BB		Priority level of interrupt 3. IP3 = 1 is high priority.					
IP[2]	IP2	BA		Priority level of interrupt 2. IP2 = 1 is high priority.					
IP[1]	IP1	B9		Priority level of interrupt 1. IP1 = 1 is high priority.					
IP[0]	IP0	B8		Priority level of interrupt 0. IP0 = 1 is high priority.					

If two interrupts of different priority levels are received simultaneously, the request of higher priority level is serviced first. If requests of the same priority level are received simultaneously, a polling sequence determines which request is serviced. In the polling sequence the interrupt 0 has the highest priority down to interrupt 6 having the lowest one.

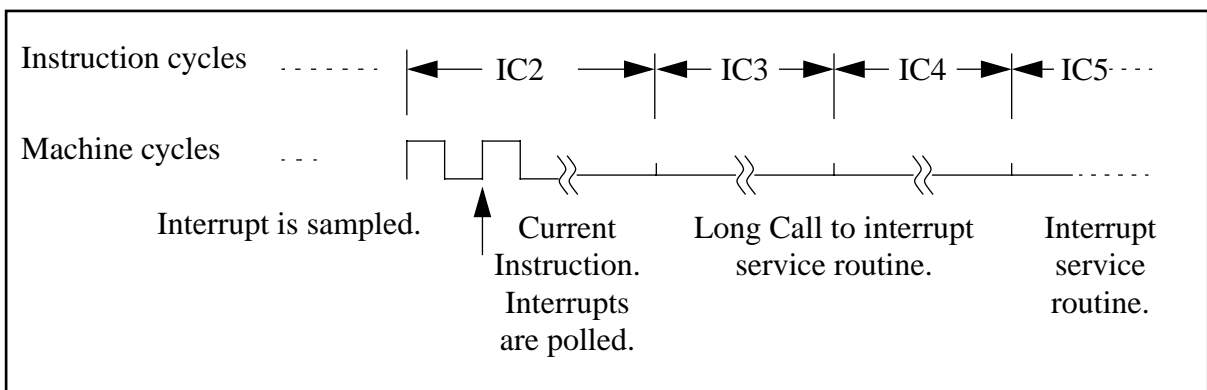
Each interrupt can be individually enabled by setting or clearing a bit in the special function register IE (figure below). IE also contains the global interrupt enable bit EA which enables or disables the interrupt unit.

Figure 8: Interrupt Enable Register

		MSB						LSB	
		EA	IE6	IE5	IE4	IE3	IE2	IE1	IE0
Bit	Symbol	Bitaddress	Function						
IE[7]	EA	AF	Enables all interrupts. If EA = 1, all interrupts are enabled.						
IE[6]	IE6	AE	Interrupt 6 enable bit. If IE6 = 1, interrupt 6 is enabled.						
IE[5]	IE5	AD	Interrupt 5 enable bit. If IE5 = 1, interrupt 5 is enabled.						
IE[4]	IE4	AC	Interrupt 4 enable bit. If IE4 = 1, interrupt 4 is enabled.						
IE[3]	IE3	AB	Interrupt 3 enable bit. If IE3 = 1, interrupt 3 is enabled.						
IE[2]	IE2	AA	Interrupt 2 enable bit. If IE2 = 1, interrupt 2 is enabled.						
IE[1]	IE1	A9	Interrupt 1 enable bit. If IE1 = 1, interrupt 1 is enabled.						
IE[0]	IE0	A8	Interrupt 0 enable bit. If IE0 = 1, interrupt 0 is enabled.						

All interrupt requests (INT) are sampled at the rising edge of each clock period of CLK. **Once an interrupt request is active for at least one clock cycle, it is hold until the appropriate interrupt routine is executed or a reset is processed, even if the interrupt is not enabled by its IEx or the EA bit. This behavior is different from the classic 8051 which does not remember interrupts.** The polling sequence of the interrupt unit starts in the first cycle of each instruction. Detecting an interrupt request the interrupt unit generates a LCALL instruction to the appropriate service routine, if this hardware generated LCALL is not blocked by either an interrupt of equal or higher priority already in progress (see figure below). If an interrupt request was not serviced by the above conditions the polling sequence will check the request again in the next instruction.

Figure 9: Interrupt Request Sequence



The fastest response time for an interrupt is two instruction cycles plus five machine cycle, all in one : 17 machine cycles. The destination address of the hardware-generated LCALL instruction depends on which interrupt was requested. The start addresses are :

Interrupt	0	0003h	4	0023h
	1	000Bh	5	002Bh
	2	0013h	6	0033h
	3	001Bh		

The service routine proceeds until the return from interrupt instruction RETI is encountered. The RETI instruction informs the Interrupt Unit that the processed interrupt is finished and the Interrupt Unit resets the appropriate interrupt request bit. **If then an other interrupt request is present it will be executed immediately without executing an instruction of the main program. This behavior is different from the classic 8051 and speeds up interrupt processing.** If no other interrupt is present execution of the interrupted program will be continued.

If inside an interrupt service routine the instruction RET is used to return to the interrupted program, the program continues to execute, but the Interrupt Unit thinks an interrupt is still in progress and makes future interrupts of the same level impossible.

Executing a RETI instruction when no interrupt is in progress, RETI behaves exactly like the RET instruction . This is useful for the test or use of interrupt routines within a regular program.

If no interrupt unit is present by setting the parameter IMPLEMENT_INT to “0“ the instruction RETI behaves like a NOP.

Interrupt and Idle Mode

To bring the FHG8051 in power down mode the power down system clock CLK_PD has to be disabled and all functions of the processor core are on hold. Only the interrupt unit is always running with systemclock CLK to reactivate the CPU from power down mode. To shut down CLK_PD a separate module for gating the clock has to be developed. FHG8051_CLKGEN can be used for this purpose presenting additional features like a wait signal and a clock divider. Please refer to the datasheet of FHG8051_CLKGEN for details.

To activate powerdown mode the idle bit IDL in register PCON[0] is used. In the FHG8051 architecture the hole PCON register is brought to the interface to allow external devices access to the power control bits. By setting IDL to “1“ all chip select signals are set to high and the system clock CLK_PD has to be disabled immediately by an external device. When CLK_PD is disabled all CPU registers keep their values and all interface signals are fully static. This allows a maximum of power saving.

The power down mode is terminated either by any **enabled** interrupt or by an hardware reset, both reset the IDL bit to “0“.

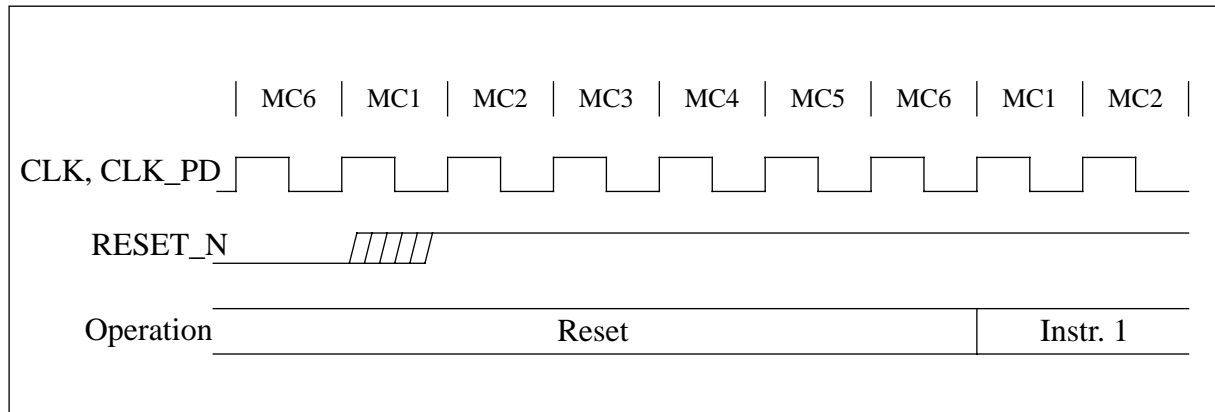
Note: The first instruction after setting the idle bit has to be a NOP instruction. There may not be a MOVX instruction after the instruction setting the idle bit, because XDATA_CS_N will be set to high level and a severe data loss may appear.

Control Unit

Reset

A reset is accomplished by holding the reset input pin RESET_N low for at least two machine cycles, while the clock is running. The FHG8051 responds by generating an internal reset.

Figure 10: Reset Sequence



The RESET_N is an asynchronous input and resets all resettable Flip-Flops inside the FG8051. It has to be synchronized externally to fulfill setup conditions between RESET_N and CLK. The core is held in a reset state as long as the active reset is detected in machine cycle one. The normal program flow starts five machine cycles after the sample period, if the RESET_N is detected as inactive. The internal reset algorithm initialises all internal registers to values listed below :

Program counter	PC	0000h	Accumulator	ACC	00h
Program status word	PSW	00h	B Register	B	00h
Stack pointer	SP	07h	Interrupt priority	IP	00h
Power control	PCON	00h	Interrupt enable	IE	00h
Data pointer low	DPL	00h	Port 0 register	P0	FFh
Data pointer high	DPH	00h	Port 1 register	P1	FFh
Data pointer 2 low	DPL2	uninitialized	Port 2 register	P2	FFh
Data pointer 2 high	DPH2	uninitialized	Port 3 register	P3	FFh
Configuration	CFG	00h			

Idle Mode

Please refer to the description Interrupt and Idle Mode at the interrupt unit section.

Interfaces

The FHG8051 has 4 separated interfaces for program memory, external data memory, internal data memory, special function register SFR and the control interface.

The program memory interface consists of the chip select signal `PROG_CS_N`, the data input bus for reading program data `PROG_DIN` and the address bus `PROG_XDATA_ADR`. If `IMPLEMENT_PWR` is set to 1 the read/write signal `PROG_RW_N` and the port `DOUT` will be used to write data to the program memory, else the signals `PROG_RW_N` and `DOUT` can be ignored for the program memory interface.

The interface to the external data memory consists of the chip select signal `XDATA_CS_N`, the read/write signal `XDATA_RW_N`, the same address bus like the program memory `PROG_XDATA_ADR`, the data input bus `XDATA_DIN` and the data output bus `DOUT`.

The interface to the internal data memory consists of the chip select signal `IDATA_CS_N`, the read/write signal `IDATA_RW_N`, the address bus `IDATA_ADR`, the data input bus `IDATA_DIN` and the data output bus `DOUT`.

The interface to the Special Function Registers consists of the chip select signal `SFR_CS_N`, the read/write signal `SFR_RW_N`, the address bus `SFR_ADR`, the data input bus `SFR_DIN` and the data output bus `DOUT`. An access to the SFR has just the same timing as an access to the internal data memory. For a lower power consumption `SFR_CS_N` is only active low if SFR register are accessed not contained in FHG8051.

The data output bus `DOUT` is used for all interfaces, because only one of the interfaces will be in write mode at a time. The bus `PROG_XDATA_ADR` is a common address bus for the program memory and the external data memory, because only one of them can be accessed at a time.

The control interface consists of the systemclock `CLK`, the power down systemclock `CLK_PD`, the low active asynchronous reset input `RESET_N`, the high active interrupt bus `INT` for the seven independent interrupt sources, the eight bit wide output bus for the power down control register `PCON` and the high active address latch enable generation output signal `ALE_GEN`.

Waveforms and timing tables

Figure 11: Waveform of program memory read access

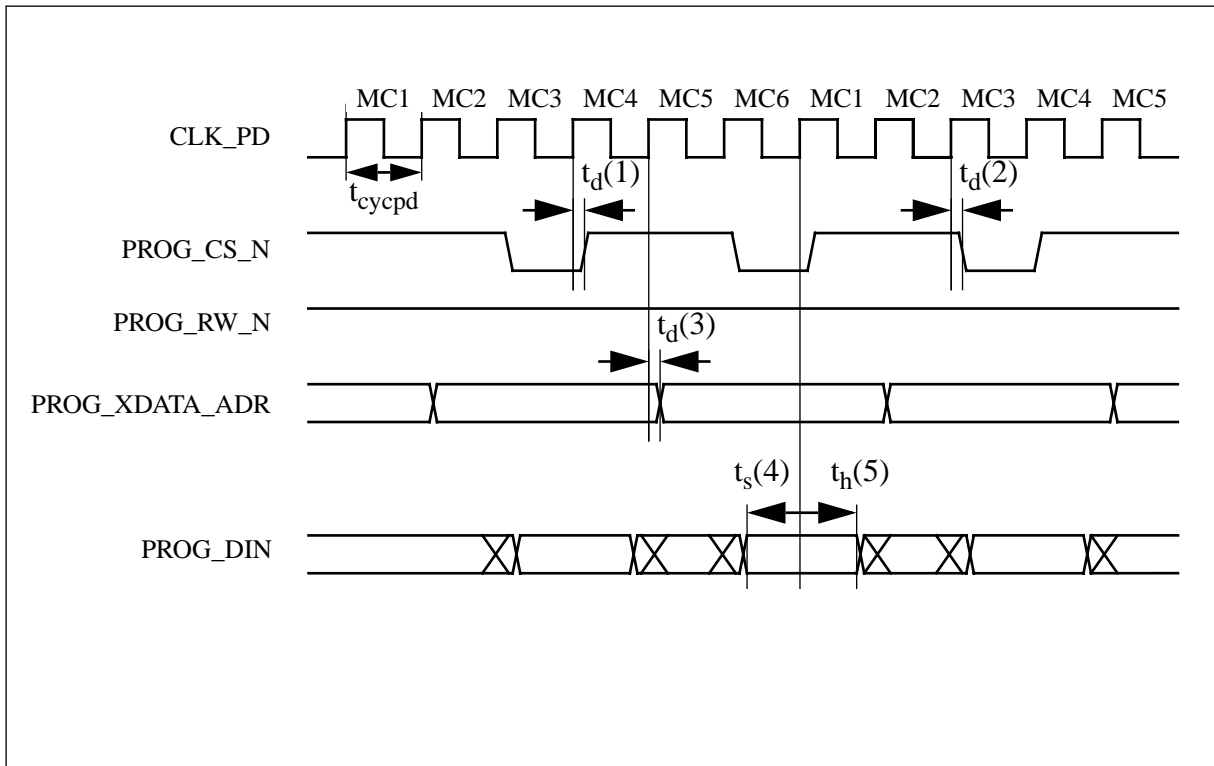


Table 6: Timing table of program memory read access

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cycpd}	$t(\text{CLK_PD}_{\text{rise}}-\text{CLK_PD}_{\text{rise}})$	CLK_PD cycle				ns
$t_d(1)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{PROG_CS_N}_{\text{rise}})$	Delay from rising CLK_PD to rising PROG_CS_N				ns
$t_d(2)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{PROG_CS_N}_{\text{fall}})$	Delay from rising CLK_PD to falling PROG_CS_N				ns
$t_d(3)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{PROG_XDATA_ADR})$	Delay from rising CLK_PD to valid PROG_XDATA_ADR				ns
$t_s(4)$	$t_s(\text{PROG_DIN}-\text{CLK_PD}_{\text{rise}})$	Setup time for PROG_DIN to rising CLK_PD				ns
$t_h(5)$	$t_h(\text{CLK_PD}_{\text{rise}}-\text{PROG_DIN})$	Hold time for PROG_DIN from rising CLK_PD				ns

Figure 12: Waveform of program memory write access

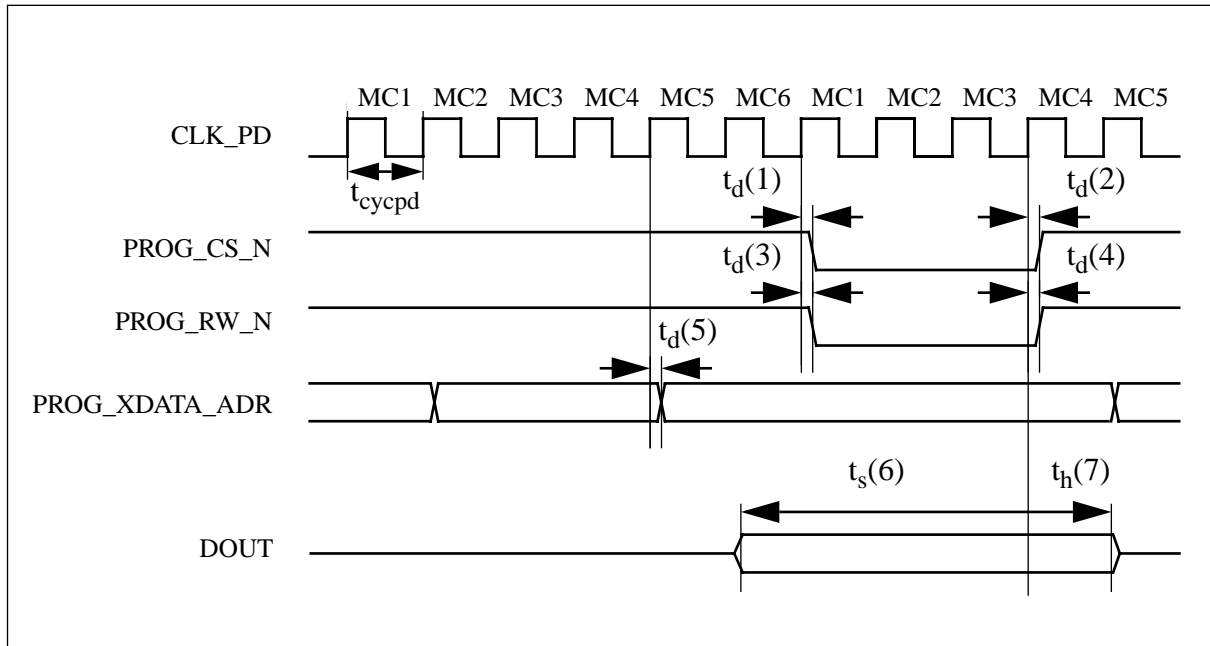


Table 7: Timing table of program memory write access

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cycpd}	$t(\text{CLK_PD}_{\text{rise}}-\text{CLK_PD}_{\text{rise}})$	CLK_PD cycle				ns
$t_d(1)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{PROG_CS_N}_{\text{fall}})$	Delay from rising CLK_PD to falling PROG_CS_N				ns
$t_d(2)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{PROG_CS_N}_{\text{rise}})$	Delay from rising CLK_PD to rising PROG_CS_N				ns
$t_d(3)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{PROG_RW_N}_{\text{fall}})$	Delay from rising CLK_PD to falling PROG_RW_N				ns
$t_d(4)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{PROG_RW_N}_{\text{rise}})$	Delay from rising CLK_PD to rising PROG_RW_N				ns
$t_d(5)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{PROG_XDATA_ADR})$	Delay from rising CLK_PD to valid PROG_XDATA_ADR				ns
$t_s(6)$	$t_s(\text{DOUT}-\text{CLK_PD}_{\text{rise}})$	Setup time for DOUT to rising CLK_PD	$< 4 \cdot t_{cycpd}$			ns
$t_h(7)$	$t_h(\text{CLK_PD}_{\text{rise}}-\text{DOUT})$	Hold time for DOUT from CLK_PD	$< 1 \cdot t_{cycpd}$			ns

Figure 13: Waveform of external data memory read access

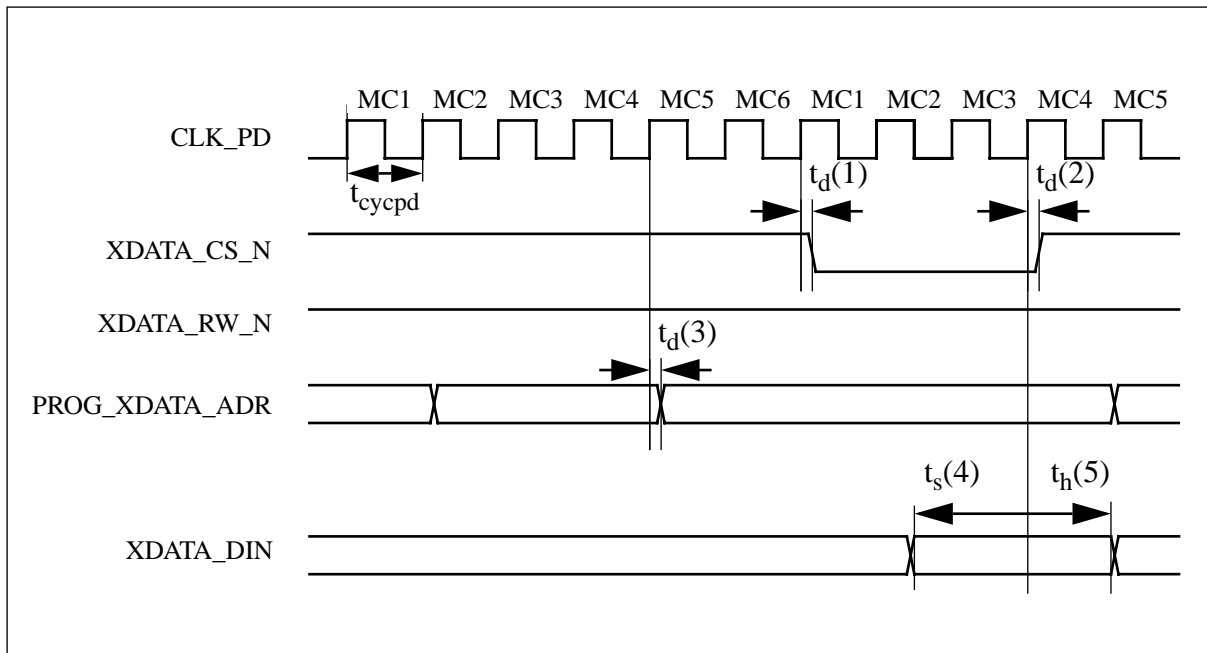


Table 8: Timing table of external data memory read access

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cycpd}	$t(\text{CLK_PD}_{\text{rise}}-\text{CLK_PD}_{\text{rise}})$	CLK_PD cycle				ns
$t_d(1)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{XDATA_CS_N}_{\text{fall}})$	Delay from rising CLK_PD to falling XDATA_CS_N				ns
$t_d(2)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{XDATA_CS_N}_{\text{rise}})$	Delay from rising CLK_PD to rising XDATA_CS_N				ns
$t_d(3)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{PROG_XDATA_ADR})$	Delay from rising CLK_PD to valid PROG_XDATA_ADR				ns
$t_s(4)$	$t_s(\text{XDATA_DIN}-\text{CLK_PD}_{\text{rise}})$	Setup time for XDATA_DIN to rising CLK_PD				ns
$t_h(5)$	$t_h(\text{CLK_PD}_{\text{rise}}-\text{XDATA_DIN})$	Hold time for XDATA_DIN from rising CLK_PD				ns

Figure 14: Waveform of external data memory write access

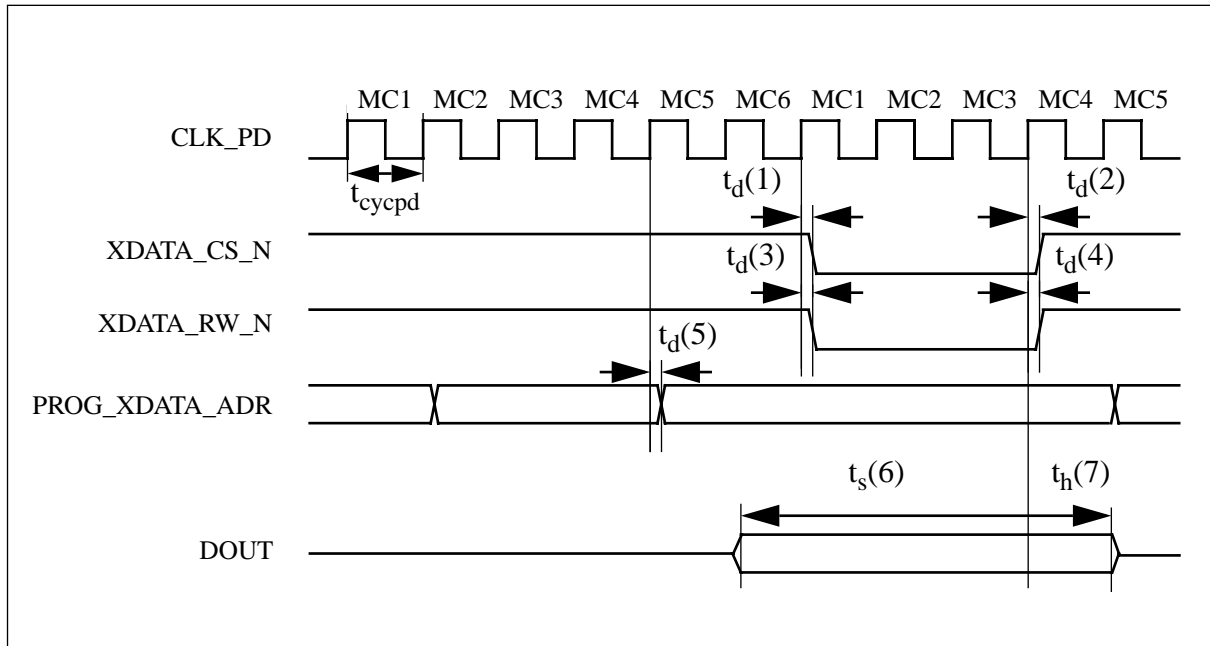
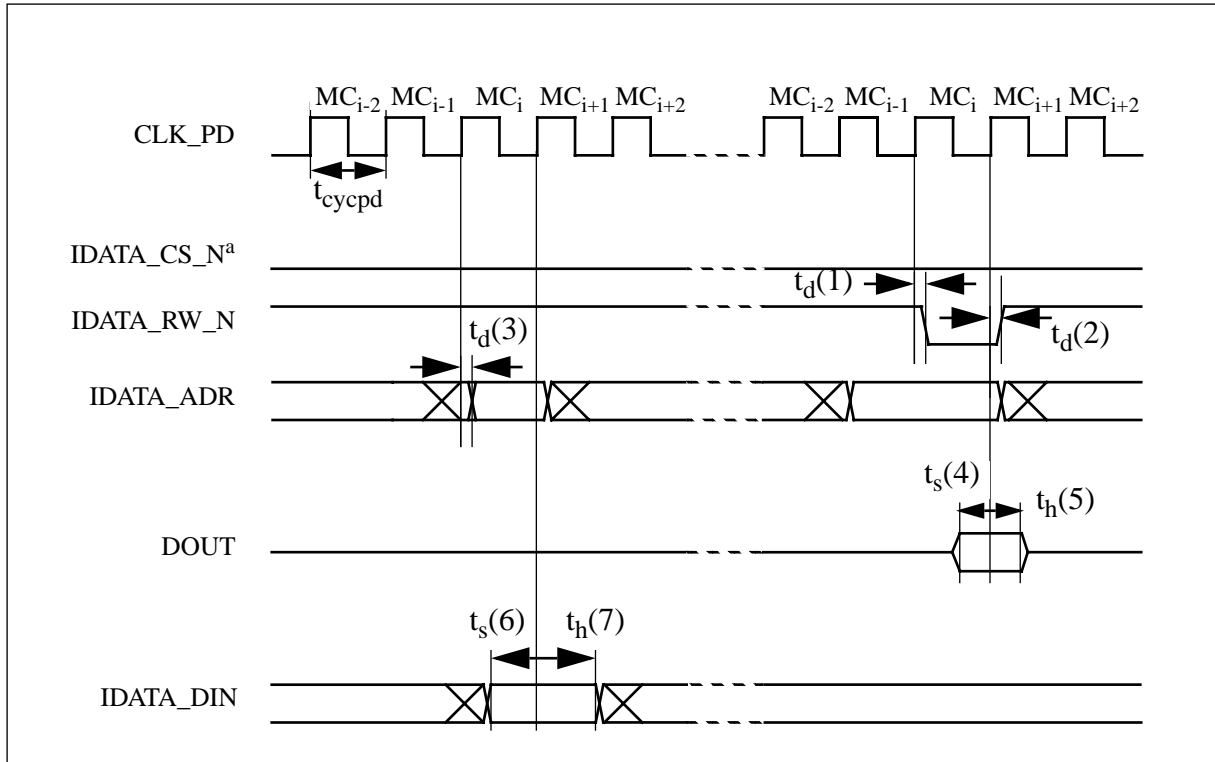


Table 9: Timing table of an external data memory write access

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cycpd}	$t(\text{CLK_PD}_{\text{rise}}-\text{CLK_PD}_{\text{rise}})$	CLK_PD cycle				ns
$t_d(1)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{XDATA_CS_N}_{\text{fall}})$	Delay from rising CLK_PD to falling XDATA_CS_N				ns
$t_d(2)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{XDATA_CS_N}_{\text{rise}})$	Delay from rising CLK_PD to rising XDATA_CS_N				ns
$t_d(3)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{XDATA_RW_N}_{\text{fall}})$	Delay from rising CLK_PD to falling XDATA_RW_N				ns
$t_d(4)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{XDATA_RW_N}_{\text{rise}})$	Delay from rising CLK_PD to rising XDATA_RW_N				ns
$t_d(5)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{PROG_XDATA_ADR})$	Delay from rising CLK_PD to valid PROG_XDATA_ADR				ns
$t_s(6)$	$t_s(\text{DOUT}-\text{CLK_PD}_{\text{rise}})$	Setup time for DOUT to rising CLK_PD	$< 4 \cdot t_{cycpd}$			ns
$t_h(7)$	$t_h(\text{CLK_PD}_{\text{rise}}-\text{DOUT})$	Hold time for DOUT from rising CLK_PD	$< 1 \cdot t_{cycpd}$			ns

Figure 15: Waveform of internal data memory read and write accesses

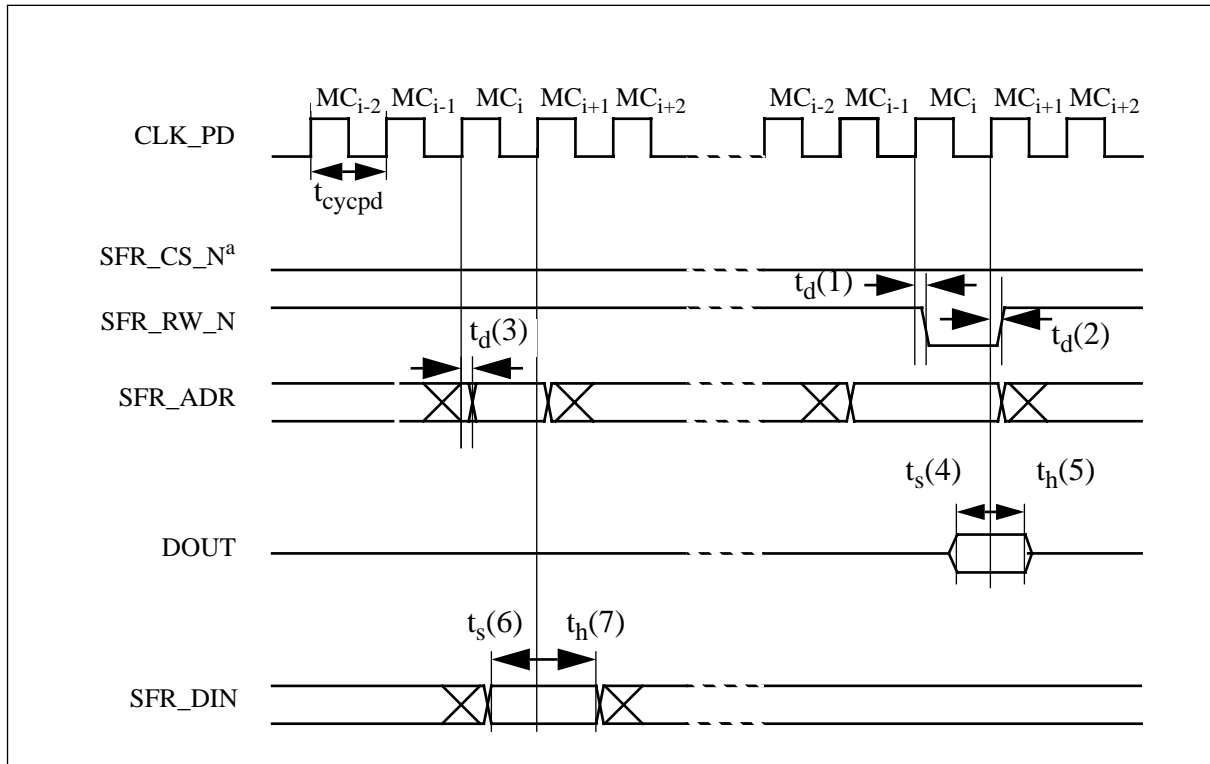


a) The chip select signal is always active low, except when Idle mode is active.

Table 10: Timing table of internal data memory read and write accesses

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cycpd}	$t(\text{CLK_PD}_{\text{rise}}-\text{CLK_PD}_{\text{rise}})$	CLK_PD cycle				ns
$t_d(1)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{IDATA_RW_N}_{\text{fall}})$	Delay from rising CLK_PD to falling IDATA_RW_N				ns
$t_d(2)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{IDATA_RW_N}_{\text{rise}})$	Delay from rising CLK_PD to rising IDATA_RW_N				ns
$t_d(3)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{IDATA_ADR})$	Delay from rising CLK_PD to valid IDATA_ADR				ns
$t_s(4)$	$t_s(\text{DOUT}-\text{CLK_PD}_{\text{rise}})$	Setup time for DOUT to rising CLK_PD	$< t_{cycpd}$			ns
$t_h(5)$	$t_h(\text{CLK_PD}_{\text{rise}}-\text{DOUT})$	Hold time for DOUT from rising CLK_PD	$\gg 0$			ns
$t_s(6)$	$t_s(\text{IDATA_DIN}-\text{CLK_PD}_{\text{rise}})$	Setup time for IDATA_DIN to rising CLK_PD				ns
$t_h(7)$	$t_h(\text{CLK_PD}_{\text{rise}}-\text{IDATA_DIN})$	Hold time for IDATA_DIN from rising CLK_PD				ns

Figure 16: Waveform of SFR read and write accesses

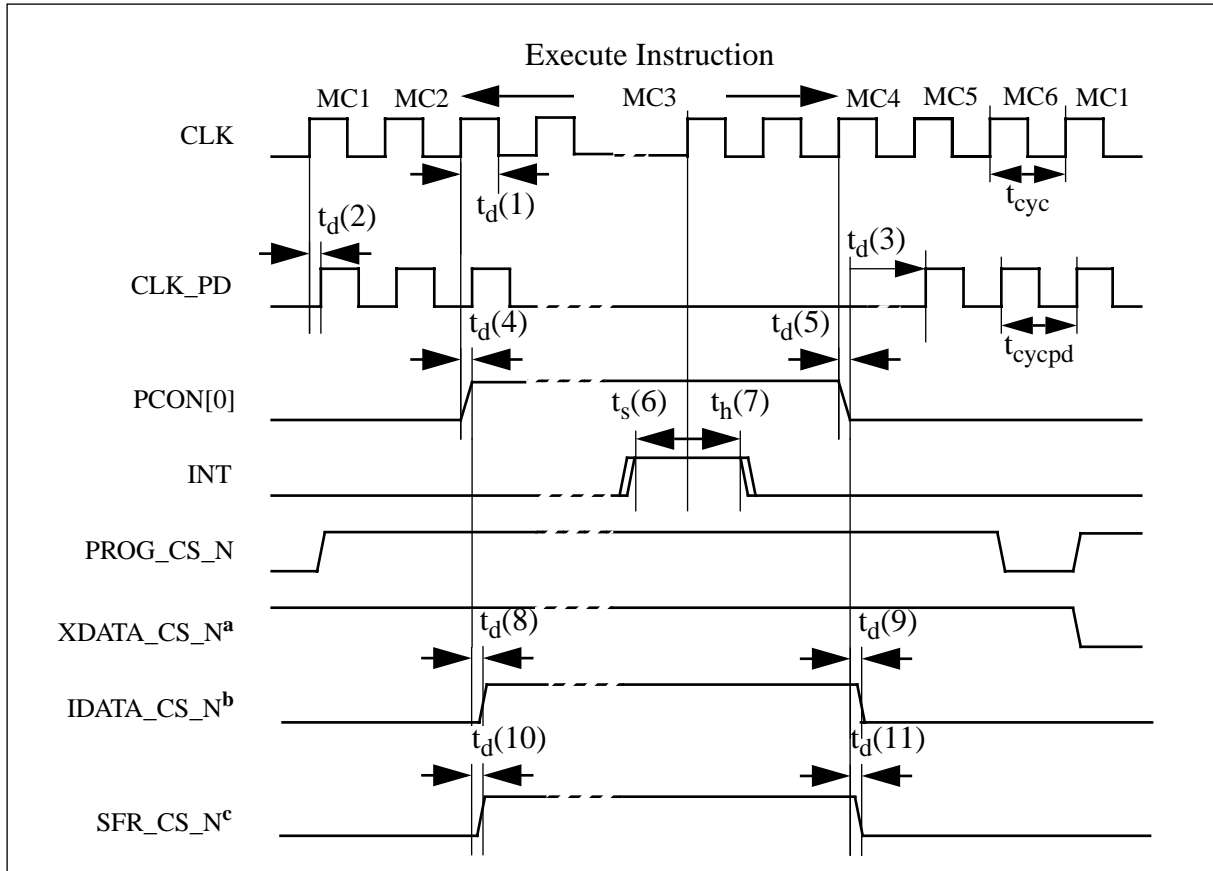


a) The chip select signal is always active low, except when Idle mode is active.

Table 11: Timing table of SFR read and write accesses

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cycpd}	$t(\text{CLK_PD}_{\text{rise}}-\text{CLK_PD}_{\text{rise}})$	CLK_PD cycle				ns
$t_d(1)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{SFR_RW_N}_{\text{fall}})$	Delay from rising CLK_PD to falling SFR_RW_N				ns
$t_d(2)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{SFR_RW_N}_{\text{rise}})$	Delay from rising CLK_PD to rising SFR_RW_N				ns
$t_d(3)$	$t_d(\text{CLK_PD}_{\text{rise}}-\text{SFR_ADR})$	Delay from rising CLK_PD to valid SFR_ADR				ns
$t_s(4)$	$t_s(\text{DOUT}-\text{CLK_PD}_{\text{rise}})$	Setup time for DOUT to rising CLK_PD	$< t_{cycpd}$			ns
$t_h(5)$	$t_h(\text{CLK_PD}_{\text{rise}}-\text{DOUT})$	Hold time for DOUT from rising CLK_PD	$\gg 0$			ns
$t_s(6)$	$t_s(\text{SFR_DIN}-\text{CLK_PD}_{\text{rise}})$	Setup time for SFR_DIN to rising CLK_PD				ns
$t_h(7)$	$t_h(\text{CLK_PD}_{\text{rise}}-\text{SFR_DIN})$	Hold time for SFR_DIN from rising CLK_PD				ns

Figure 17: Waveform of idle mode



- Note
- a) assuming instruction after setting the idle bit is MOVX
 - b) remember, IDATA_CS_N is always like the idle bit PCON[0]
 - c) assuming instruction after setting the idle bit accesses a core external SFR

Table 12: Timing table of idle mode

Symbol	Name	Item	Notes	Min	Max	Unit
t_{cyc}	$t(\text{CLK}_{\text{rise}}-\text{CLK}_{\text{rise}})$	Clock cycle				ns
t_{cycpd}	$t(\text{CLK_PD}_{\text{rise}}-\text{CLK_PD}_{\text{rise}})$	CLK_PD cycle				ns
$t_d(1)$	$t(\text{CLK}_{\text{rise}}-\text{CLK}_{\text{fall}})$	Clock high time				ns
$t_d(2)$	$t(\text{CLK}_{\text{rise}}-\text{CLK_PD}_{\text{rise}})$	Clock to CLK_PD delay		0	$\ll t_d(1)$	ns
$t_d(3)$	$t(\text{PCON}[0]_{\text{fall}}-\text{CLK_PD}_{\text{rise}})$	PCON[0] idle mode bit disabled to first CLK_PD rising edge after reactivated delay	depends on CLK_PD generation			ns
$t_d(4)$	$t_d(\text{CLK}_{\text{rise}}-\text{PCON}[0]_{\text{rise}})$	Delay from rising CLK to rising PCON[0]				ns
$t_d(5)$	$t_d(\text{CLK}_{\text{rise}}-\text{PCON}[0]_{\text{fall}})$	Delay from rising CLK to falling PCON[0]				ns
$t_s(6)$	$t_s(\text{INT}_{\text{high}}-\text{CLK}_{\text{rise}})$	Setup time for any high active INT to rising CLK				ns
$t_h(7)$	$t_h(\text{CLK}_{\text{rise}}-\text{INT}_{\text{high}})$	Hold time for any high active INT from rising CLK				ns

Table 12: Timing table of idle mode

Symbol	Name	Item	Notes	Min	Max	Unit
$t_d(8)$	$t_d(\text{PCON}[0]_{\text{rise}} - \text{IDATA_CS_N}_{\text{rise}})$	Delay from rising PCON[0] to rising IDATA_CS_N				ns
$t_d(9)$	$t_d(\text{PCON}[0]_{\text{fall}} - \text{IDATA_CS_N}_{\text{fall}})$	Delay from falling PCON[0] to falling IDATA_CS_N	a)			ns
$t_d(10)$	$t_d(\text{PCON}[0]_{\text{rise}} - \text{SFR_CS_N}_{\text{rise}})$	Delay from rising PCON[0] to rising SFR_CS_N	b)			ns
$t_d(11)$	$t_d(\text{PCON}[0]_{\text{fall}} - \text{SFR_CS_N}_{\text{fall}})$	Delay from falling PCON[0] to falling SFR_CS_N	c)			ns

Instruction Set

All 50 different instructions with 9 arithmetical, 10 logical, 7 data transfer, 6 bit manipulation and 18 program control operations from the industry standard 8051 are implemented in FHG8051. With all addressing modes there is a total of 111 instructions.

Table 13: Arithmetic Instructions

Instruction	Addressing modes ^a	Description
ADD	R, DD, IR, ID	Add data to accumulator
ADDC	R, DD, IR, ID	Add data to accumulator with carry
SUBB	R, DD, IR, ID	Subtract data from accumulator with borrow
INC	R, DD, IR	Increment data
DEC	R, DD, IR	Decrement data
INC DPTR		Increment Data pointer
MUL AB		Multiply accumulator and register B
DIV AB		Divide accumulator by register B
DA A		Decimal adjust accumulator

a. R = Register, DD = Direct Data, IR = Indirect RAM, ID = Immediate Data

Table 14: Logical Instructions

Instruction	Addressing modes	Description
ANL	R, DD, IR, ID	Logical AND data to accumulator or direct data (only A, ID)
ORL	R, DD, IR, ID	Logical OR data to accumulator or direct data (only A, ID)
XRL	R, DD, IR, ID	Logical XOR data to accumulator or direct data (only A, ID)
CLR A		Clear accumulator
CPL A		Complement accumulator
RL A		Rotate accumulator left
RLC A		Rotate accumulator left through the carry
RR A		Rotate accumulator right
RRC A		Rotate accumulator right through the carry
SWAP A		Swap nibbles within the accumulator

Table 15: Data Transfer Instructions

Instruction	Addressing modes	Description
MOV	R, DD, IR, ID	Move data to accumulator or direct data
MOVC		Move code byte relative to DPTR or PC to accumulator
MOVB	R	Move data to / from external to / from accumulator
PUSH	DD	Push data onto the stack
POP	DD	POP data from the stack
XCH	R, DD, IR	Exchange data with accumulator
XCHD	IR	Exchange low-order nibble of data with low-order nibble of accumulator

Table 16: Bit Manipulation Instructions

Instruction	Addressing modes	Description
CLR		Clear bit
SETB		Set bit
CPL		Complement bit
ANL		AND bit to carry
ORL		OR bit to carry
MOV		Move bit to / from carry

Table 17: Program Control Instructions

Instructions	Addressing modes	Description
ACALL	DD	Absolute (11 bit) subroutine call
LCALL	DD	Long (16 bit) subroutine call
RET		Return from subroutine
RETI		Return from interrupt
AJMP	DD	Absolute (11 bit) jump
LJMP	DD	Long (16 bit) jump
SJMP	Rel	Short (relative) jump
JMP	IR	Jump indirect relative to the DPTR
JC	Rel	Jump relative if carry is set
JNC	Rel	Jump relative if carry not set
JB	Rel	Jump relative if direct bit is set
JNB	Rel	Jump relative if direct bit not set
JBC	Rel	Jump relative if direct bit is set and clear
JZ	Rel	Jump relative if accumulator is zero
JNZ	Rel	Jump relative if accumulator is not zero
CJNE	DD, ID, Rel	Compare data to accumulator, register or indirect data and jump relative if not equal
DJNZ	R, DD, Rel	Decrement register or direct data and jump relative if not zero
NOP		No operation

Opcode list

All 255 legal opcodes of the FHG8051 are listed below. The 256th opcode is the opcode A5h and is an illegal instruction, but it behaves like an NOP in the FHG8051.

If the second data pointer is implemented, there are six instructions, which are affected by DPTR1 and DPTR2, depending on bit DPSEL. For more details on DPSEL please refer to the Parameter Description section.

JMP @A+DPTR
INC DPTR
MOV DPTR, #16-Bit immediate
MOVX A, @DPTR
MOVX @DPTR, A
MOVC A, @A+DPTR

Please refer to the Parameter Description section or more details about Cycle Reduction.

Table 18: Opcode List

Hex code	Mnemonics	Opcode Bytes	Instruction Cycles	Cycle reduction
00	NOP	1	1	No
01	AJMP code addr	2	2	No
02	LJMP code addr	3	2	No
03	RR A	1	1	No
04	INC A	1	1	No
05	INC dir	2	1	No
06	INC @R0	1	1	No
07	INC @R1	1	1	No
08	INC R0	1	1	No
09	INC R1	1	1	No
0A	INC R2	1	1	No
0B	INC R3	1	1	No
0C	INC R4	1	1	No
0D	INC R5	1	1	No
0E	INC R6	1	1	No
0F	INC R7	1	1	No
10	JBC bit addr, code	3	2	No
11	ACALL code addr	2	2	No
12	LCALL code addr	3	2	No
13	RRC A	1	1	No
14	DEC A	1	1	No
15	DEC dir	2	1	No
16	DEC @R0	1	1	No
17	DEC @R1	1	1	No
18	DEC R0	1	1	No
19	DEC R1	1	1	No
1A	DEC R2	1	1	No
1B	DEC R3	1	1	No
1C	DEC R4	1	1	No
1D	DEC R5	1	1	No
1E	DEC R6	1	1	No
1F	DEC R7	1	1	No

Table 18: Opcode List

Hex code	Mnemonics	Opcode Bytes	Instruction Cycles	Cycle reduction
20	JB bit addr, code	3	2	No
21	AJMP code addr	2	2	No
22	RET	1	2	No
23	RL A	1	1	No
24	ADD A, #data	2	1	No
25	ADD A, dir	2	1	No
26	ADD A, @R0	1	1	No
27	ADD A, @R1	1	1	No
28	ADD A, R0	1	1	No
29	ADD A, R1	1	1	No
2A	ADD A, R2	1	1	No
2B	ADD A, R3	1	1	No
2C	ADD A, R4	1	1	No
2D	ADD A, R5	1	1	No
2E	ADD A, R6	1	1	No
2F	ADD A, R7	1	1	No
30	JNB bit addr, code	3	2	No
31	ACALL code addr	2	2	No
32	RETI	1	2	No
33	RLC A	1	1	No
34	ADDC A, #data	2	1	No
35	ADDC A, dir	2	1	No
36	ADDC A, @R0	1	1	No
37	ADDC A, @R1	1	1	No
38	ADDC A, R0	1	1	No
39	ADDC A, R1	1	1	No
3A	ADDC A, R2	1	1	No
3B	ADDC A, R3	1	1	No
3C	ADDC A, R4	1	1	No
3D	ADDC A, R5	1	1	No
3E	ADDC A, R6	1	1	No
3F	ADDC A, R7	1	1	No

Table 18: Opcode List

Hex code	Mnemonics	Opcode Bytes	Instruction Cycles	Cycle reduction
40	JC code addr	2	2	No
41	AJMP code addr	2	2	No
42	ORL dir, A	2	1	No
43	ORL dir, #data	3	2	No
44	ORL A, #data	2	1	No
45	ORL A, dir	2	1	No
46	ORL A, @R0	1	1	No
47	ORL A, @R1	1	1	No
48	ORL A, R0	1	1	No
49	ORL A, R1	1	1	No
4A	ORL A, R2	1	1	No
4B	ORL A, R3	1	1	No
4C	ORL A, R4	1	1	No
4D	ORL A, R5	1	1	No
4E	ORL A, R6	1	1	No
4F	ORL A, R7	1	1	No
50	JNC code addr	2	2	No
51	ACALL code addr	2	2	No
52	ANL dir, A	2	1	No
53	ANL dir, #data	3	2	No
54	ANL A, #data	2	1	No
55	ANL A, dir	2	1	No
56	ANL A, @R0	1	1	No
57	ANL A, @R1	1	1	No
58	ANL A, R0	1	1	No
59	ANL A, R1	1	1	No
5A	ANL A, R2	1	1	No
5B	ANL A, R3	1	1	No
5C	ANL A, R4	1	1	No
5D	ANL A, R5	1	1	No
5E	ANL A, R6	1	1	No
5F	ANL A, R7	1	1	No

Table 18: Opcode List

Hex code	Mnemonics	Opcode Bytes	Instruction Cycles	Cycle reduction
60	JZ code addr	2	2	No
61	AJMP code addr	2	2	No
62	XRL dir, A	2	1	No
63	XRL dir, #data	3	2	No
64	XRL A, #data	2	1	No
65	XRL A, dir	2	1	No
66	XRL A, @R0	1	1	No
67	XRL A, @R1	1	1	No
68	XRL A, R0	1	1	No
69	XRL A, R1	1	1	No
6A	XRL A, R2	1	1	No
6B	XRL A, R3	1	1	No
6C	XRL A, R4	1	1	No
6D	XRL A, R5	1	1	No
6E	XRL A, R6	1	1	No
6F	XRL A, R7	1	1	No
70	JNZ code addr	2	2	No
71	ACALL code addr	2	2	No
72	ORL C, bit addr	2	2	Yes
73	JMP @A+DPTR	1	2	No
74	MOV A, #data	2	1	No
75	MOV dir, #data	3	2	No
76	MOV @R0, #data	2	1	No
77	MOV @R1, #data	2	1	No
78	MOV R0, #data	2	1	No
79	MOV R1, #data	2	1	No
7A	MOV R2, #data	2	1	No
7B	MOV R3, #data	2	1	No
7C	MOV R4, #data	2	1	No
7D	MOV R5, #data	2	1	No
7E	MOV R6, #data	2	1	No
7F	MOV R7, #data	2	1	No

Table 18: Opcode List

Hex code	Mnemonics	Opcode Bytes	Instruction Cycles	Cycle reduction
80	SJMP code addr	2	2	No
81	AJMP code addr	2	2	No
82	ANL C, bit addr	2	2	Yes
83	MOVC A, @A+PC	2	2	No
84	DIV AB	1	4	No
85	MOV dir, dir	3	2	No
86	MOV dir, @R0	2	2	Yes
87	MOV dir, @R1	2	2	Yes
88	MOV dir, R0	2	2	Yes
89	MOV dir, R1	2	2	Yes
8A	MOV dir, R2	2	2	Yes
8B	MOV dir, R3	2	2	Yes
8C	MOV dir, R4	2	2	Yes
8D	MOV dir, R5	2	2	Yes
8E	MOV dir, R6	2	2	Yes
8F	MOV dir, R7	2	2	Yes
90	MOV DPTR, #data	3	2	No
91	ACALL code addr	2	2	No
92	MOV bit addr, C	2	2	Yes
93	MOVC A, @A+DPTR	2	2	No
94	SUBB A, #data	2	1	No
95	SUBB A, dir	2	1	No
96	SUBB A, @R0	1	1	No
97	SUBB A, @R1	1	1	No
98	SUBB A, R0	1	1	No
99	SUBB A, R1	1	1	No
9A	SUBB A, R2	1	1	No
9B	SUBB A, R3	1	1	No
9C	SUBB A, R4	1	1	No
9D	SUBB A, R5	1	1	No
9E	SUBB A, R6	1	1	No
9F	SUBB A, R7	1	1	No

Table 18: Opcode List

Hex code	Mnemonics	Opcode Bytes	Instruction Cycles	Cycle reduction
A0	ORL C, $\overline{\text{bit addr}}$	2	2	Yes
A1	AJMP code addr	2	2	No
A2	MOV C, bit addr	2	1	No
A3	INC DPTR	1	2	Yes
A4	MUL AB	1	4	No
A5	n/a (reserved)	1	1	No
A6	MOV @R0, dir	2	2	Yes
A7	MOV @R1, dir	2	2	Yes
A8	MOV R0, dir	2	2	Yes
A9	MOV R1, dir	2	2	Yes
AA	MOV R2, dir	2	2	Yes
AB	MOV R3, dir	2	2	Yes
AC	MOV R4, dir	2	2	Yes
AD	MOV R5, dir	2	2	Yes
AE	MOV R6, dir	2	2	Yes
AF	MOV R7, dir	2	2	Yes
B0	ANL C, $\overline{\text{bit addr}}$	2	2	Yes
B1	ACALL code addr	2	2	No
B2	CPL bit addr	2	1	No
B3	CPL C	1	1	No
B4	CJNE A, #data, code	3	2	No
B5	CJNE A, dir, code	3	2	No
B6	CJNE @R0, #data, code	3	2	No
B7	CJNE @R1, #data, code	3	2	No
B8	CJNE R0, #data, code	3	2	No
B9	CJNE R1, #data, code	3	2	No
BA	CJNE R2, #data, code	3	2	No
BB	CJNE R3, #data, code	3	2	No
BC	CJNE R4, #data, code	3	2	No
BD	CJNE R5, #data, code	3	2	No
BE	CJNE R6, #data, code	3	2	No
BF	CJNE R7, #data, code	3	2	No

Table 18: Opcode List

Hex code	Mnemonics	Opcode Bytes	Instruction Cycles	Cycle reduction
C0	PUSH dir	2	2	Yes
C1	AJMP code addr	2	2	No
C2	CLR bit addr	2	1	No
C3	CLR C	1	1	No
C4	SWAP A	1	1	No
C5	XCH A, dir	2	1	No
C6	XCH A, @R0	1	1	No
C7	XCH A, @R1	1	1	No
C8	XCH A, R0	1	1	No
C9	XCH A, R1	1	1	No
CA	XCH A, R2	1	1	No
CB	XCH A, R3	1	1	No
CC	XCH A, R4	1	1	No
CD	XCH A, R5	1	1	No
CE	XCH A, R6	1	1	No
CF	XCH A, R7	1	1	No
D0	POP dir	2	2	Yes
D1	ACALL code addr	2	2	No
D2	SETB bit addr	2	1	No
D3	SETB C	1	1	No
D4	DA A	1	1	No
D5	DJNZ dir, code addr	3	2	No
D6	XCHD A, @R0	1	1	No
D7	XCHD A, @R1	1	1	No
D8	DJNZ R0, code addr	2	2	No
D9	DJNZ R1, code addr	2	2	No
DA	DJNZ R2, code addr	2	2	No
DB	DJNZ R3, code addr	2	2	No
DC	DJNZ R4, code addr	2	2	No
DD	DJNZ R5, code addr	2	2	No
DE	DJNZ R6, code addr	2	2	No
DF	DJNZ R7, code addr	2	2	No

Table 18: Opcode List

Hex code	Mnemonics	Opcode Bytes	Instruction Cycles	Cycle reduction
E0	MOVX A, @DPTR	2	2	No
E1	AJMP code addr	2	2	No
E2	MOVX A, @R0	2	2	No
E3	MOVX A, @R1	2	2	No
E4	CLR A	1	1	No
E5	MOV A, dir	2	1	No
E6	MOV A, @R0	1	1	No
E7	MOV A, @R1	1	1	No
E8	MOV A, R0	1	1	No
E9	MOV A, R1	1	1	No
EA	MOV A, R2	1	1	No
EB	MOV A, R3	1	1	No
EC	MOV A, R4	1	1	No
ED	MOV A, R5	1	1	No
EE	MOV A, R6	1	1	No
EF	MOV A, R7	1	1	No
F0	MOVX @DPTR, A	1	2	No
F1	ACALL code addr	2	2	No
F2	MOVX @R0, A	1	2	No
F3	MOVX @R1, A	1	2	No
F4	CPL A	1	1	No
F5	MOV dir, A	2	1	No
F6	MOV @R0, A	1	1	No
F7	MOV @R1, A	1	1	No
F8	MOV R0, A	1	1	No
F9	MOV R1, A	1	1	No
FA	MOV R2, A	1	1	No
FB	MOV R3, A	1	1	No
FC	MOV R4, A	1	1	No
FD	MOV R5, A	1	1	No
FE	MOV R6, A	1	1	No
FF	MOV R7, A	1	1	No

Application notes

Figure 18: Application Example

