MIX - Micronas Interconnect Specification Expander

I2C Sheet - Specification

Micronas - Munich

0.1 General

The I2C sheet is one more category of input specification. It's content describes I2C Register-blocks and Register connections to Signals of the design core. Signal-names can be used directly because MIX internal uses global signals. By default a new defined Register-block is bound to `TOPLEVEL` (default Language: `VHDL`), this could be changed by adding the Register-blocks Instance manual into the `HIER`-Sheet. If no Instance is defined in `HIER`-Sheet, `MIX` adds this definition internal (Name: `iic_if_<interface>`).

0.2 I2C-Sheet details

The following table describes MIX I2C's header-keywords:

Column name	**Descriptionend**	**Default value**	**Req.**	**Example**
::ign	Ignore line	<empty>	man.	# comm.
::variants	Variant selector	Default	opt.	Var1
::width	Register width	16	man.	16
::dev	Device name	n/a	man.	FRCA0
::sub	Sub address	VHDL	opt.	27
::interface	Domain name	W_NO_CONFIG	man.	cc0
::block	Block Name	<empty>	opt.	mc
::dir	Direction	<empty>	opt.	RW
::spec	Update Signal	NTO	opt.	takeover_b72
::clock	Clock Domain	<empty>	opt.	clkcc81
::reset	Reset Signal	<empty>	opt.	asresc_n
::busy	Busy Signal	<empty>	opt.	b81
::b	Bit n	<empty>	man.	FMSYN.2
::init	Reset Value	<empty>	opt.	0
::rec	Recommended Value	<empty>	opt.	0
::comment	Comment field	<empty>	opt.	This is a comment

0.3 I2C-Column details

::ign	Every columns row, containing # or // is ignored by MIX
::variants	Select a line depending on the -variant command line switch. If -variant VAR is set, only lines whose `::variant` cell contains the keyword VAR, Default or empty, are selected and read in. Several variants may be given in one cell, separated by ",". Without specifying the -variant switch, the "Default" and empty `::variant` cells are read in and evaluated.
::width	Define Register width.
::dev	Specify the Device address
::sub	Define Register-block-sub-address
::interface	Domain name
::block	Register-block Name. The Register-block name and its sub-address are building the Instance-name.
::dir	Defines a Registers input/output direction, possible values are: `R`, `W` or `RW`
::spec	Update signal
::clock	set a clock Domain. Every I2C Register may receive it's own clock.
::reset	Specify a Reset signal. This Signal Resets the I2C Registers.
::busy	busy signal
::b	Specifies a signal which is connected to a Registers pin. This is a multiple column which can be used to set different signals to the same Register
::init	Set initializing value of register connection(s).
::rec	Set recommended value of a register connection(s).
::comment	All entries in these cells are threaten as comments. MIX will keep comments to put them later into VHDL/Verilog output.

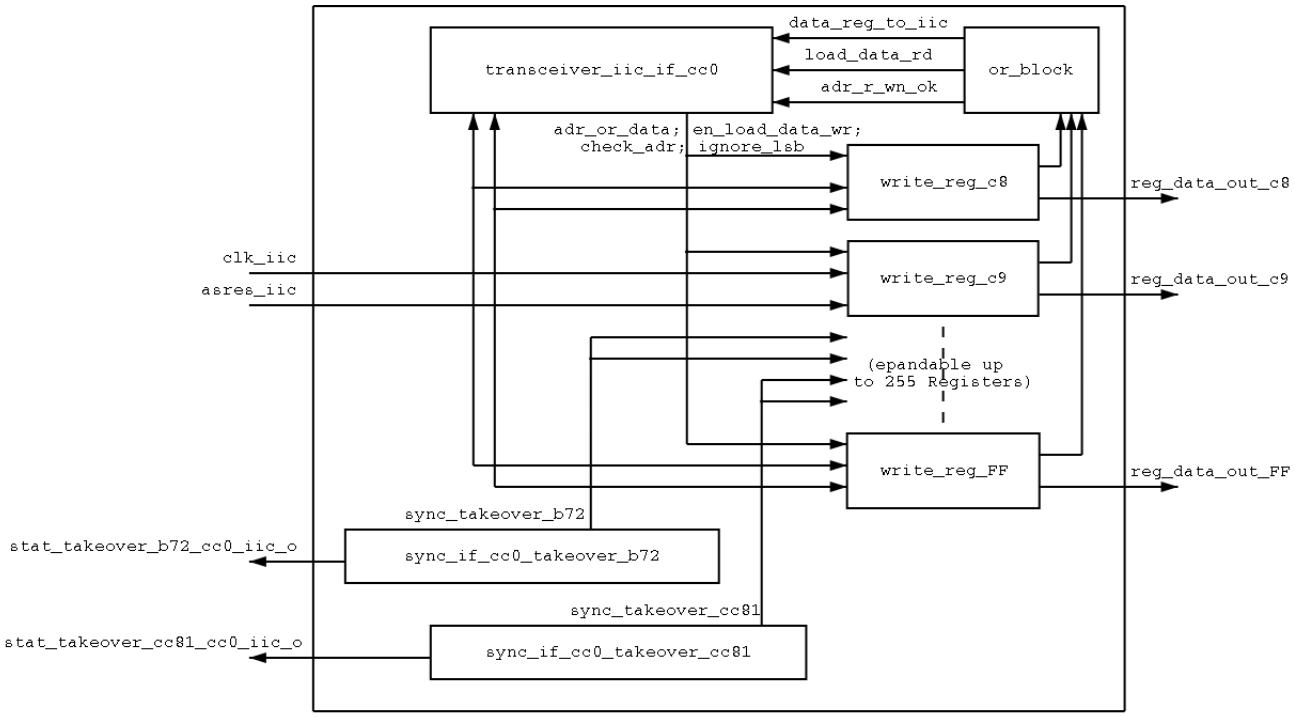
0.4 I2C-Sheet input and VHDL output examples

The following screen-shots are showing a small part of FRCA's I2C sheet. This section describes the `cc0` Registers which consist of 8 Sub-blocks (expandable up to 255 Registers). Each Sub-block Register-connections can be split into multiple rows.

::ign	::type	::width	::dev	::sub	::addr	::int	::block	::inst	::dir	::auto	::sync	::spec	::clock	::reset	::busy	::readDone	::new	::b
	I2C	16	FRCA0	27		cc0	mc	mc_27	RW	Y		takeover_b72	clkcc81	asresc_n				FREEZE
	I2C	16	FRCA0	27		cc0	mc	mc_27	RW	Y		takeover_b72	clkcc81	asresc_n				
	I2C	16	FRCA0	27		cc0	mc	mc_27	RW	Y		takeover_b72	clkcc81	asresc_n				
	I2C	16	FRCA0	27		cc0	mc	mc_27	RW	Y		takeover_b72	clkcc81	asresc_n				
	I2C	16	FRCA0	27		cc0	mc	mc_27	RW	Y		takeover_b72	clkcc81	asresc_n				
	I2C	16	FRCA0	27		cc0	mc	mc_27	RW	Y		takeover_cc81	clkcc81	asresc_n				
	I2C	16	FRCA0	27		cc0	mc	mc_27	RW	Y		takeover_b72	clkcc81	asresc_n				
	I2C	16	FRCA0	28		cc0	mc	mc_28	RW	Y		takeover_b72	clkcc81	asresc_n				
	I2C	16	FRCA0	28		cc0	mc	mc_28	RW	Y		takeover_b72	clkcc81	asresc_n				
	I2C	16	FRCA0	29		cc0	mc	mc_29	RW	Y		takeover_cc81	clkcc81	asresc_n				
	I2C	16	FRCA0	30		cc0	mc	mc_30	RW	Y		takeover_cc81	clkcc81	asresc_n				
	I2C	16	FRCA0	30		cc0	mc	mc_30	RW	Y		takeover_cc81	clkcc81	asresc_n				
	I2C	16	FRCA0	30		cc0	mc	mc_30	RW	Y		takeover_cc81	clkcc81	asresc_n				
	I2C	16	FRCA0	31		cc0	por	por_31	RW	Y		takeover_cc81	clkcc81	asresc_n				
	I2C	16	FRCA0	200		cc0	c800	c800_200	RW	Y		NTO	clkcc81	asresc_n				c800_1.15
	I2C	16	FRCA0	201		cc0	c800	c800_201	RW	Y		NTO	clkcc81	asresc_n				c800_2.15
	I2C	16	FRCA0	202		cc0	c800	c800_202	RW	Y		NTO	clkcc81	asresc_n				c800_3.15

::b	::b	::b	::b	::b	::b	::b	::b	::b	::b	::b	::b	::b	::b
FMSYN.2	FMSYN.1	FMSYN.0											
			FMSYNUNS										
				VERRES									
					WRITE.1	WRITE.0							
							READ.1	READ.0					
									WRPOSX.5	WRPOSX.4	WRPOSX.3	WRPOSX.2	WRPOSX.1
						Intprogm							
							WRPOSY.7	WRPOSY.6	WRPOSY.5	WRPOSY.4	WRPOSY.3	WRPOSY.2	WRPOSY.1
						RDPOSX.8	RDPOSX.7	RDPOSX.6	RDPOSX.5	RDPOSX.4	RDPOSX.3	RDPOSX.2	RDPOSX.1
				RDPOSY.7	RDPOSY.6	RDPOSY.5	RDPOSY.4	RDPOSY.3	RDPOSY.2	RDPOSY.1	RDPOSY.0		
												REFRON	
													REFRPER.1
c800_1.14	c800_1.13	c800_1.12	c800_1.11	c800_1.10	c800_1.9	c800_1.8	c800_1.7	c800_1.6	c800_1.5	c800_1.4	c800_1.3	c800_1.2	c800_1.1
c800_2.14	c800_2.13	c800_2.12	c800_2.11	c800_2.10	c800_2.9	c800_2.8	c800_2.7	c800_2.6	c800_2.5	c800_2.4	c800_2.3	c800_2.2	c800_2.1
c800_3.14	c800_3.13	c800_3.12	c800_3.11	c800_3.10	c800_3.9	c800_3.8	c800_3.7	c800_3.6	c800_3.5	c800_3.4	c800_3.3	c800_3.2	c800_3.1

::b	::init	::rec	::range	::view	::vi2c	::name	::comment
	0	0	0..1	Y	VSP_16Bit_RW_Register	FREEZE	\x\bMC: Freeze the picture
	0	0	0..7	Y	VSP_16Bit_RW_Register	FMSYN	\x\bMC: Synchronisation of film mode signal
	0	0	0..1	Y	VSP_16Bit_RW_Register	FMSYNUNS	\x\bMC: Synchronisation of film mode signal when unsecure
	0	0	0..1	Y	VSP_16Bit_RW_Register	VERRES	\x\bMC: Vertical Resolution for Frame based processing
	0	0	0..3	Y	VSP_16Bit_RW_Register	WRITE	\x\bMC: Write Mode
	0	0	0..3	Y	VSP_16Bit_RW_Register	READ	\x\bMC: Read Mode Channel
WRPOSX.0	0	0	0..63	Y	VSP_16Bit_RW_Register	WRPOSX	\x\bMC: Horizontal Writing Position of Picture in the Memory
	0	0	0..1	Y	VSP_16Bit_RW_Register	Intprogm	\x\bMC: Interlaced or progressive input
WRPOSY.0	0	0	0..255	Y	VSP_16Bit_RW_Register	WRPOSY	\x\bMC: Vertical Position of Picture in the Memory
RDPOSX.0	0	0	0..511	Y	VSP_16Bit_RW_Register	RDPOSX	\x\bMC: Horizontal Read Position pixel number indicating the start position of reading for the picture
	0	0	0..255	Y	VSP_16Bit_RW_Register	RDPOSY	\x\bMC: Vertical Read Position line number indicating the start line of reading for the picture
	0	0	0..1	Y	VSP_16Bit_RW_Register	REFRON	\x\bMC: Refresh On
REFRPER.0	0	0	0..3	Y	VSP_16Bit_RW_Register	REFRPER	\x\bMC: Refresh period
PORCNCL	0	0	0..1	Y	VSP_16Bit_RW_Register	PORCNCL	\x\bPOR: Reset control bit cancel
c800_1.0	0	0	0..6553	Y	VSP_16Bit_RW_Register	C880_DATA_REG1	\x\bC800: Comunication Register 1
c800_2.0	0	0	0..6553	Y	VSP_16Bit_RW_Register	C880_DATA_REG2	\x\bC800: Comunication Register 2
c800_3.0	0	0	0..6553	Y	VSP_16Bit_RW_Register	C880_DATA_REG3	\x\bC800: Comunication Register 3



VHDL Architecture:

```
library ieee;

use ieee.std_logic_1164.all;
use work.iic_pack.all;
use work.comp_pack.all;



ENTITY iic_if_cc0 IS

PORT (
        clk_iic   : IN  std_ulogic;
        clkcc81   : IN  std_ulogic;
        asres_iic         : IN  std_ulogic;
        asresc_n          : IN  std_ulogic;
        takeover_b72_i    : IN std_ulogic;
        stat_takeover_b72_cc0_iic_o       : OUT std_ulogic;
        takeover_cc81_i   : IN std_ulogic;
        stat_takeover_cc81_cc0_iic_o      : OUT std_ulogic;
        c800_1_iic_o      : OUT std_ulogic_vector (15 DOWNTO 0);
        c800_2_iic_o      : OUT std_ulogic_vector (15 DOWNTO 0);
        c800_3_iic_o      : OUT std_ulogic_vector (15 DOWNTO 0);
        fmsyn_iic_o       : OUT std_ulogic_vector (2 DOWNTO 0);
        fmsynuns_iic_o    : OUT std_ulogic;
        freeze_iic_o      : OUT std_ulogic;
        intprogm_iic_o    : OUT std_ulogic;
        porcncl_iic_o     : OUT std_ulogic;
        rdposx_iic_o      : OUT std_ulogic_vector (8 DOWNTO 0);
        rdposy_iic_o      : OUT std_ulogic_vector (7 DOWNTO 0);
        read_iic_o        : OUT std_ulogic_vector (1 DOWNTO 0);
        refron_iic_o      : OUT std_ulogic;
        refrper_iic_o     : OUT std_ulogic_vector (1 DOWNTO 0);
        verres_iic_o      : OUT std_ulogic;
        write_iic_o       : OUT std_ulogic_vector (1 DOWNTO 0);
        wrposx_iic_o      : OUT std_ulogic_vector (5 DOWNTO 0);
        wrposy_iic_o      : OUT std_ulogic_vector (7 DOWNTO 0);
        tf_en_wr_FE_o     : OUT std_ulogic;
        tf_en_wr_FF_o     : OUT std_ulogic;
        iic_adr_data_i    : IN std_ulogic;
        iic_en_i          : IN std_ulogic;
        iic_if_cc0_data_o         : OUT std_ulogic;
        ignore_lsb_i      : IN std_ulogic;
        ais_cc0_o         : OUT std_ulogic
        );
END iic_if_cc0;
```

VHDL Entity (incomplete):

```
Architecture rtl of iic_if_cc0 is

 SIGNAL adr_or_data              : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL adr_r_wn_ok              : std_ulogic;
 SIGNAL load_data_rd             : std_ulogic;
 SIGNAL check_adr                : std_ulogic;
 SIGNAL adr_no_data              : std_ulogic;
 SIGNAL en_load_data_wr          : std_ulogic;
 SIGNAL data_reg_to_iic          : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL logic_1                  : std_ulogic;
 SIGNAL logic_0                  : std_ulogic;
 SIGNAL ignore_lsb               : std_ulogic;
 SIGNAL reg_data_out_c8          : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL data_regc8_to_iic        : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL load_data_rd_regc8       : std_ulogic;
 SIGNAL adr_r_wn_ok_regc8        : std_ulogic;
 SIGNAL reg_data_out_c9          : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL data_regc9_to_iic        : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL load_data_rd_regc9       : std_ulogic;
 SIGNAL adr_r_wn_ok_regc9        : std_ulogic;
 SIGNAL reg_data_out_ca          : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL data_regca_to_iic        : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL load_data_rd_regca       : std_ulogic;
 SIGNAL adr_r_wn_ok_regca        : std_ulogic;
 SIGNAL reg_data_out_1b          : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL data_reg1b_to_iic        : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL load_data_rd_reg1b       : std_ulogic;
 SIGNAL adr_r_wn_ok_reg1b        : std_ulogic;
 SIGNAL reg_data_out_1c          : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL data_reg1c_to_iic        : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL load_data_rd_reg1c       : std_ulogic;
 SIGNAL adr_r_wn_ok_reg1c        : std_ulogic;
 SIGNAL reg_data_out_1d          : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL data_reg1d_to_iic        : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL load_data_rd_reg1d       : std_ulogic;
 SIGNAL adr_r_wn_ok_reg1d        : std_ulogic;
 SIGNAL reg_data_out_1e          : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL data_reg1e_to_iic        : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL load_data_rd_reg1e       : std_ulogic;
 SIGNAL adr_r_wn_ok_reg1e        : std_ulogic;
 SIGNAL reg_data_out_1f          : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL data_reg1f_to_iic        : std_ulogic_vector(15 DOWNTO 0);
 SIGNAL load_data_rd_reg1f       : std_ulogic;
 SIGNAL adr_r_wn_ok_reg1f        : std_ulogic;
 SIGNAL sync_takeover_b72                : std_ulogic;
 SIGNAL sync_takeover_cc81               : std_ulogic;
 SIGNAL tf_en_wrFE               : std_ulogic;
 SIGNAL tf_en_wrFF               : std_ulogic;
```

```
Begin

logic_1 <= '1';

logic_0 <= '0';

write_reg_c8: iic_ser_reg_cc_wr
 Generic Map (
 ra_g =>         200,   -- HEX c8
 dor_g =>        0,
 ur_g =>         65535,
 sr_g =>         0,
 ais_g =>        0,
 hs_g =>         0,
 sh_g =>         1,
 latch_g =>      0
 )
 PORT MAP(
 clk => clk_iic,
 clk_sh =>       clkcc81,
 reset =>        asres_iic,
 reset_s =>      asresc_n,
 asres =>        asres_iic,
 asres_s =>      asresc_n,
 adr_or_data_i =>         adr_or_data,
 check_adr_i =>  check_adr,
 en_load_data_wr_i =>     en_load_data_wr,
 tf_ready_wr_i =>         logic_1,
 load_shadow_i =>         logic_1,
 ignore_lsb_i =>          ignore_lsb,
 ais_o =>         open,
 tf_en_wr_o =>    open,
 data_to_iic_o =>        data_regc8_to_iic,
 load_data_rd_o =>       load_data_rd_regc8,
 adr_r_wn_ok_o =>        adr_r_wn_ok_regc8,
 reg_data_out_o =>        reg_data_out_c8
 );
 c800_1_iic_o(0)          <= reg_data_out_c8(0);
 c800_1_iic_o(1)          <= reg_data_out_c8(1);
 c800_1_iic_o(2)          <= reg_data_out_c8(2);
 c800_1_iic_o(3)          <= reg_data_out_c8(3);
 c800_1_iic_o(4)          <= reg_data_out_c8(4);
 c800_1_iic_o(5)          <= reg_data_out_c8(5);
 c800_1_iic_o(6)          <= reg_data_out_c8(6);
 c800_1_iic_o(7)          <= reg_data_out_c8(7);
 c800_1_iic_o(8)          <= reg_data_out_c8(8);
 c800_1_iic_o(9)          <= reg_data_out_c8(9);
 c800_1_iic_o(10)         <= reg_data_out_c8(10);
 c800_1_iic_o(11)         <= reg_data_out_c8(11);
 c800_1_iic_o(12)         <= reg_data_out_c8(12);
 c800_1_iic_o(13)         <= reg_data_out_c8(13);
 c800_1_iic_o(14)         <= reg_data_out_c8(14);
 c800_1_iic_o(15)         <= reg_data_out_c8(15);
```

This part of the "`cc0`" Architecture only shows the Sub-Address 200 (Hex: `c8`). As you can see the Interface-name (or Domain-name) will become the Architecture-name while the Sub-address is used to identify the Sub-Register-block. The Port-map describes connections as defined in the I2C Sheet. In the Generic Map you can find some I2C specific definitions, as "`ra_g`" which hold the Registers Address, as "`ur_g`" defines the Port-Range.