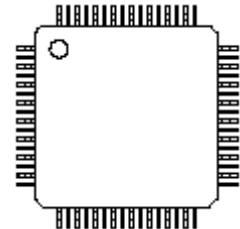


Intelligent

Integrated 3-Phase Motor Control System

- runs on a Xilinx Spartan XCS30, a SpartanII XC2S50 or a SpartanIIE XC2S50E series FPGA
- 33 MHz operation frequency (XCS30, "-4" speed grade) or higher (XC2S50, XC2S50E)
- Spartan XCS30 device usage: 237 CLBs (synthesis) → 415 CLBs (mapping) (out of 576 available CLBs), equivalent to 72% usage
- SpartanII/IIE XC2S50(E) device usage: 503 slices (mapping), equivalent to 65% usage
- very compact 100 pin package (XCS30)
- fast, configurable 16 or 32 bit wide microprocessor bus interface can easily be connected to a DSP
- integrated 3-Phase PWM generator drives 6 power switches (MOSFETS, IGBTs) with symmetrical PWM and programmable dead time to avoid vertical shoot-thru
- PWM frequency of 16.27 kHz (@33 MHz clock) giving a non audible 32.54 kHz tone
- built-in custom 10-bit RISC engine
- digital "P" or "PI"-type current controllers, (with programmable coefficients) implemented in microcode on the RISC CPU
- complete current control loop execution every 61.44 μ sec (@33 MHz clock)
- fully autonomous control of two synchronous, 12-bit external analog-to-digital converters (ADC) for the acquisition of the instantaneous motor currents. Requires no glue logic when using either Burr-Brown ADS7816 or Microchip MCP3201 Analog/Digital Converters
- current regulator "bypass mode" allowing for direct control of the output's PWM duty cycle by host processor (for velocity control or for use in voltage inverters applications)
- 14 easy-to-program registers (memory mapped) allow full control of the device's functions
- on-chip incremental encoder interface with 4x quadrature discriminator and advanced fault detection mechanisms
- interface for absolute encoder tracks (U,V,W) eases brushless AC/DC motor commutation
- 24 or 32 -bit preloadable position counter
- 24 or 32 -bit capture register can latch the position's counter value when triggered by external events
- possibility to automatically clear encoder counter when zero mark is passed (for rotating axes)
- interrupt logic can generate interrupts upon different events/conditions and allows fast, 1-cycle interrupt acknowledges
- boundary scan interface
- Encoder Interface sub-unit is functional and software compatible to MLCA_1

MLCA_4



Xilinx Spartan XCS30-4VQ100C
(100 pin VQFP package)
Xilinx SpartanII XC2S50-5TQ144
(144 pin TQFP package)
Xilinx SpartanIIE XC2S50E-5TQ144
(144 pin TQFP package)

Contents:

1	BLOCK DIAGRAM AND INTERFACES.....	4
2	TYPICAL APPLICATION	5
3	TARGET SILICON	6
4	PIN DESCRIPTIONS	7
4.1	SPARTAN XCS30-VQ100 (100 PIN) VERSION	7
4.2	SPARTANII XC2S50-TQ144 (144 PIN) VERSION.....	10
4.3	SPARTANIII XC2S50E-TQ144 (144 PIN) VERSION.....	12
5	MEMORY MAP.....	14
5.1	REGISTERS READ.....	14
5.2	REGISTERS WRITE.....	14
6	THE BUS INTERFACE	15
7	THE ENCODER UNIT.....	16
7.1	THE CONTROL UNIT.....	17
7.1.1	The Control Register.....	17
7.1.2	The Interrupt Status Register and the interrupt logic.....	17
7.1.3	The Status Register (Status_Poll_Reg).....	19
7.2	THE COUNTER UNIT.....	21
7.2.1	Counter Low (16 bit wide databus).....	21
7.2.2	Counter High (16 bit wide databus).....	21
7.2.3	Counter32 (32 bit wide databus).....	22
7.3	THE CAPTURE UNIT.....	23
7.3.1	Capture Low (16 bit wide databus).....	23
7.3.2	Capture High (16 bit wide databus).....	23
7.3.3	Capture32 (32 bit wide databus).....	24
7.4	ERROR DETECTION MECHANISMS	25
7.5	OTHER FUNCTIONS	26
8	THE CURRENT REGULATOR UNIT	27
8.1	THE CURRENT REGULATOR.....	27
8.2	THE CONTROLLER OF THE EXTERNAL ANALOG TO DIGITAL CONVERTERS (ADC).....	28
8.3	THE 3 PHASE PWM WAVEFORM GENERATOR.....	28
8.3.1	Operating principle	29
8.3.2	The PWM waveform signal generator.....	29
8.3.3	The PWM_STOP function	30
8.4	REGISTERS DESCRIPTION	31
8.4.1	Registers PWM_IREF_W and PWM_IREF_U.....	31
8.4.2	Registers PWM_PARAMETER_B1 and PWM_PARAMETER_B0	31
8.4.3	Register PWM_PARAMETER_DELTA.....	31
8.4.4	Registers PWM_I_MOT_W and PWM_I_MOT_U.....	34
9	TIMING DIAGRAMS	35
10	PACKAGE DIMENSIONS	39
10.1	VQ100 (VERY THIN QUAD FLAT PACK), FOR SPARTAN XCS30 VERSION.....	39
10.2	TQ144/HQ144 (TQFP/HEAT SINK TQFP), FOR SPARTANII/III.....	40
A	APPENDIX	41
A.1	BIBLIOGRAPHY.....	41
A.2	MLCA_4 CORE INSTALLATION AND IMPLEMENTATION INSTRUCTIONS	41
A.3	REVISION HISTORY	43

A.3.1	HARDWARE.....	43
A.3.2	DRIVER SOFTWARE.....	44
A.3.3	DOCUMENTATION	44

1 Block diagram and interfaces

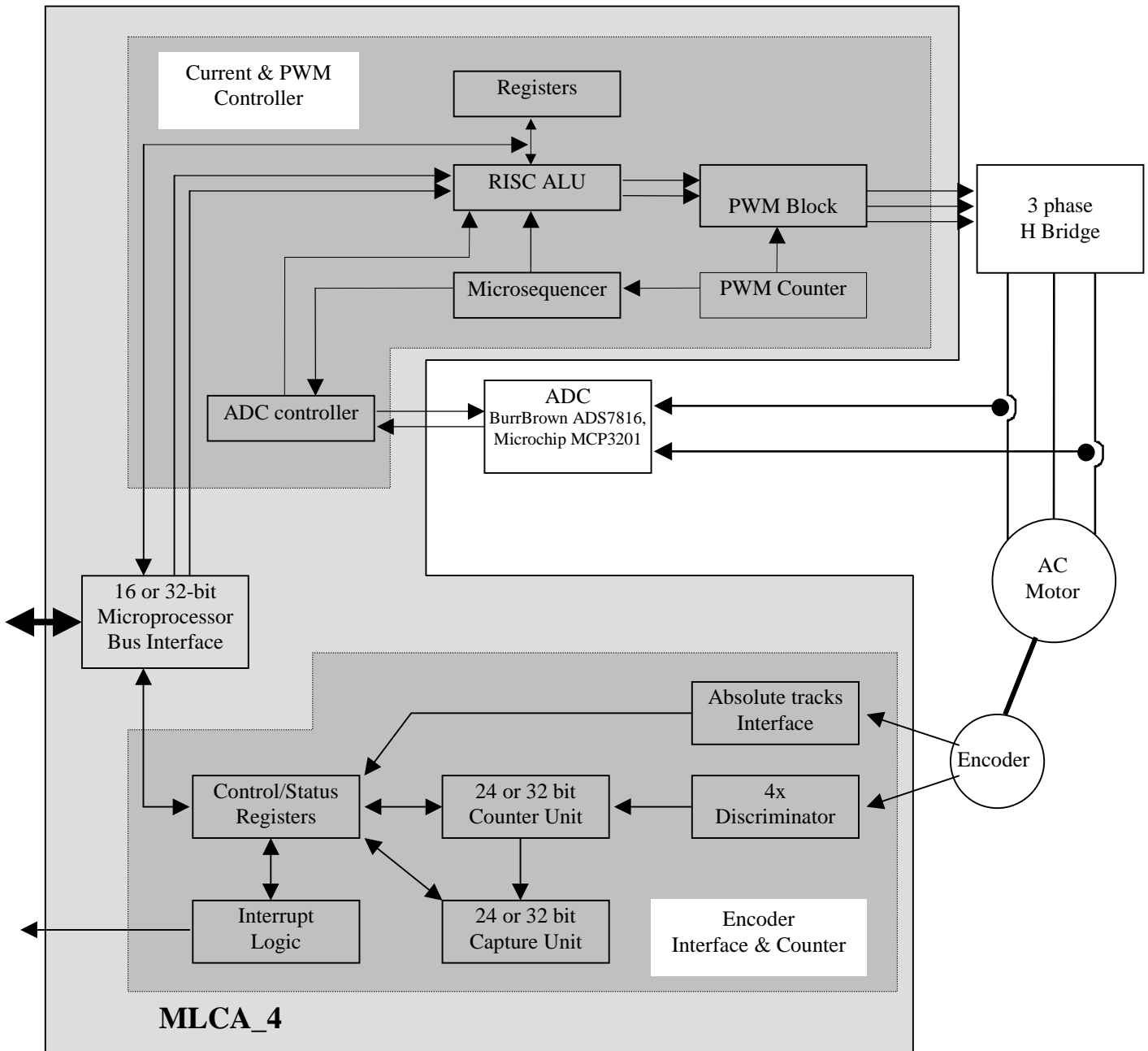


Fig. 1

2 Typical application

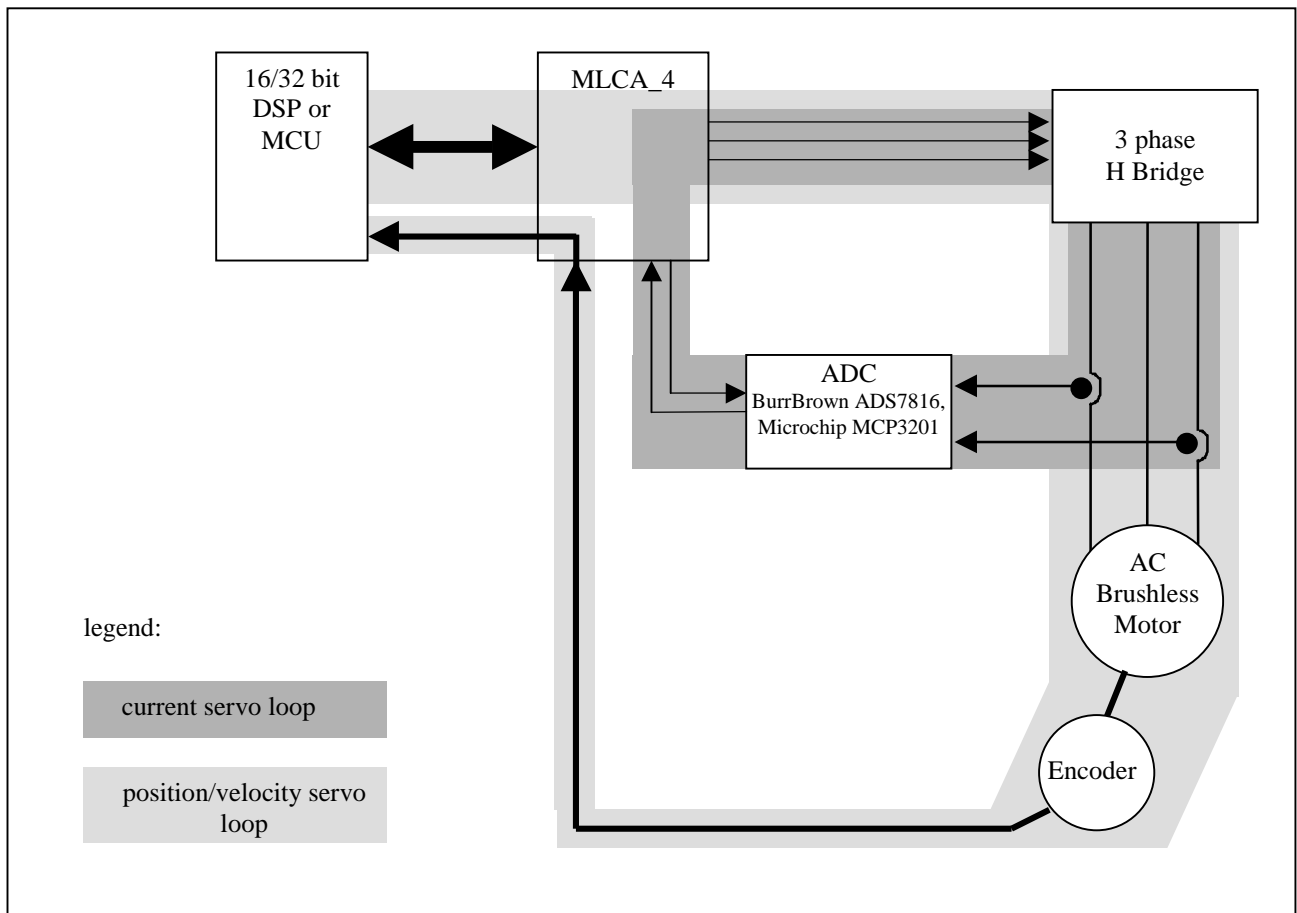


Fig. 2

3 TARGET SILICON

The MLCA_4 has been implemented and is currently available in three different Xilinx architectures:

- the Spartan family, and
- the SpartanII family, and
- the SpartanIIE family

The functionality and software architecture (registers) is identical in all the implementations. There are however some differences in the case style and pinout, the I/O logic levels and the power supply voltages, due to the different properties of these product families.

A derivative design, the MLCA_4-DC, capable of controlling two DC motors (brush type) instead of one 3 phase motor is also offered in all mentioned silicon families.

The following table summarizes the differences between the different implementations.

Please note that the all implementations have distinct EDIF netlists, which can not be interchanged. Also the configuration bit stream is different for each family.

The parts indicated in the table are the smallest (in terms of both number of gates and package pins) of each supported family, that can hold the entire design.

Property	Spartan XCS30-4-VQ100	SpartanII XC2S50-5-TQ144	SpartanIIE XC2S50E-6-TQ144
Supply voltage	5V	3.3V and 2.5V	3.3V and 1.8V
I/O logic levels (note 1)	TTL, CMOS	LVTTL (5V tolerant)	LVTTL (5V tolerant)
Package	smaller (VQ100)	larger (TQ/HQ144)	larger (TQ/HQ144)
Chip cost	higher	lower	lower
Available, unused logic resources	very limited	medium	medium
Larger chips available, same family	1	3	4
Recommended configuration	serial	serial	serial
Configuration file size	247.9 kbits	559.2 kbits	630.1 kbits
Boundary Scan test interface	yes	yes	yes
Speed	high	higher	highest
Place & Route effort	high (multipass p&r recommended)	medium	medium-low
Place & Route timing constraints	clock frequency only	clock frequency only	clock frequency only
Xilinx implementation tools	version 3.3 or higher	version 4.1 or higher	version 4.1 or higher

Table 1 Implementation differences between the different supported logic families

Notes:

1. when a SpartanII/IIE device is receiving a signal from a 5V TTL driver, a series resistor (100 Ohm or similar) should be inserted into the signal line, in order to reduce the current through the chip's input clamp diode clamping to V_{CC0} .

4 PIN DESCRIPTIONS

NOTE:

The device should be configured in "serial mode". The pinout was optimized for this mode, by avoiding double use of the serial configuration pins.

4.1 Spartan XCS30-VQ100 (100 pin) version

Pin Name	Direction	Pin Number
ad_clk	OUTPUT	P20
ad_cs	OUTPUT	P16
ad_inW	INPUT	P98
ad_inU	INPUT	P17
addr<0>	INPUT	P2
addr<1>	INPUT	P83
addr<2>	INPUT	P93
addr<3>	INPUT	P82
aux_in	INPUT	P10
clk	INPUT	P27
cs	INPUT	P84
dhe	INPUT	P87
enco_a_in	INPUT	P41
enco_b_in	INPUT	P40
enco_u_in	INPUT	P39
enco_v_in	INPUT	P14
enco_w_in	INPUT	P97
enco_zero	INPUT	P15
error	OUTPUT	P85
ext_latch	INPUT	P99
interrupt	OUTPUT	P9
not_mres	INPUT	P43
npwmW (=npwm1)	OUTPUT	P45
npwmU (=npwm2)	OUTPUT	P44
npwmV (=npwm3)	OUTPUT	P35
npwmZ -- don't connect(used only in MLCA_4-DC)	OUTPUT	P31
pwmW (=pwm1)	OUTPUT	P46
pwmU (=pwm2)	OUTPUT	P42
pwmV (=pwm3)	OUTPUT	P33
pwmZ -- don't connect(used only in MLCA_4-DC)	OUTPUT	P32
pwm_stop	INPUT	P34
rd	INPUT	P91
wr	INPUT	P95
xdata_bus<0>	BIDIR	P69
xdata_bus<1>	BIDIR	P68
xdata_bus<2>	BIDIR	P96
xdata_bus<3>	BIDIR	P7
xdata_bus<4>	BIDIR	P3
xdata_bus<5>	BIDIR	P92
xdata_bus<6>	BIDIR	P86
xdata_bus<7>	BIDIR	P94
xdata_bus<8>	BIDIR	P8
xdata_bus<9>	BIDIR	P66
xdata_bus<10>	BIDIR	P90
xdata_bus<11>	BIDIR	P58
xdata_bus<12>	BIDIR	P67
xdata_bus<13>	BIDIR	P65
xdata_bus<14>	BIDIR	P59
xdata_bus<15>	BIDIR	P13
xdata_bus<16>	BIDIR	P48

PIN DESCRIPTION for XCS30, 100 pin version (cont'd)

Pin Name	Direction	Pin Number
xdata_bus<17>	BIDIR	P53
xdata_bus<18>	BIDIR	P47
xdata_bus<19>	BIDIR	P78
xdata_bus<20>	BIDIR	P81
xdata_bus<21>	BIDIR	P80
xdata_bus<22>	BIDIR	P79
xdata_bus<23>	BIDIR	P70
xdata_bus<24>	BIDIR	P61
xdata_bus<25>	BIDIR	P62
xdata_bus<26>	BIDIR	P71
xdata_bus<27>	BIDIR	P55
xdata_bus<28>	BIDIR	P54
xdata_bus<29>	BIDIR	P60
xdata_bus<30>	BIDIR	P56
xdata_bus<31>	BIDIR	P57

Power Supply Pin Name	Pin Number
GND	P49, P1, P38 P77, P23, P11 P64, P88
VCC	P25, P100 P89, P75, P12 P51, P37, P63

Configuration and Programming Pin Name	Pin Number
/PROG	P52
CCLK	P74
DONE	P50
MODE (/master, slave)	P24
don't connect!	P22
don't connect!	P26
HDC	P28
/LDC	P30
/INIT (connect external 10kOhm pullup resistor!)	P36
DIN	P72
DOUT	P73

Reserved Output Pin Name	Pin Number
pwmZ (=pwm4)(used only in MLCA_4-DC:leave n.c. else!)	P32
npwmZ(=npwm4)(used only in MLCA_4-DC:leave n.c. else!)	P31

Boundary Scan Pin Name	Pin Number
TDO (Boundary Scan)	P76
TDI (Boundary Scan)	P4
TCK (Boundary Scan) (connect ext. 10kOhm pullup resistor!)	P5
TMS (Boundary Scan) (connect ext. 10kOhm pullup resistor!)	P6

PIN DESCRIPTION for XCS30, 100 pin version (cont'd)

Design Test Pins - DON'T connect !	Pin Number
Test1 - don't connect	P18
Test2 - don't connect	P19
Test3 - don't connect	P21
Test4 - don't connect	P29

4.2 SpartanII XC2S50-TQ144 (144 pin) version

Signal Name	Pin Number	Direction	IO Standard
ad_clk	P64	OUTPUT	LVTTL
ad_cs	P23	OUTPUT	LVTTL
ad_inW	P56	INPUT	LVTTL
ad_inU	P51	INPUT	LVTTL
addr(0)	P28	INPUT	LVTTL
addr(1)	P42	INPUT	LVTTL
addr(2)	P43	INPUT	LVTTL
addr(3)	P47	INPUT	LVTTL
aux_in	P86	INPUT	LVTTL
clk	P91	INPUT	LVTTL
cs	P44	INPUT	LVTTL
dhe	P6	INPUT	LVTTL
enco_a_in	P79	INPUT	LVTTL
enco_b_in	P84	INPUT	LVTTL
enco_u_in	P132	INPUT	LVTTL
enco_v_in	P95	INPUT	LVTTL
enco_w_in	P100	INPUT	LVTTL
enco_zero	P83	INPUT	LVTTL
error	P63	OUTPUT	LVTTL
ext_latch	P26	INPUT	LVTTL
interrupt	P94	OUTPUT	LVTTL
not_mres	P31	INPUT	LVTTL
npwmW (=npwm1)	P10	OUTPUT	LVTTL
npwmU (=npwm2)	P130	OUTPUT	LVTTL
npwmV (=npwm3)	P121	OUTPUT	LVTTL
npwmZ (=npwm4) (reserved)	P41	OUTPUT	LVTTL (don't connect!) *)
pwmW (=pwm1)	P12	OUTPUT	LVTTL
pwmU (=pwm2)	P122	OUTPUT	LVTTL
pwmV (=pwm3)	P96	OUTPUT	LVTTL
pwmZ (=pwm4) (reserved)	P40	OUTPUT	LVTTL (don't connect!) *)
pwm_stop	P11	INPUT	LVTTL
rd	P49	INPUT	LVTTL
wr	P50	INPUT	LVTTL
xdata_bus(0)	P21	BIDIR	LVTTL
xdata_bus(1)	P20	BIDIR	LVTTL
xdata_bus(2)	P62	BIDIR	LVTTL
xdata_bus(3)	P57	BIDIR	LVTTL
xdata_bus(4)	P46	BIDIR	LVTTL
xdata_bus(5)	P58	BIDIR	LVTTL
xdata_bus(6)	P22	BIDIR	LVTTL
xdata_bus(7)	P48	BIDIR	LVTTL
xdata_bus(8)	P59	BIDIR	LVTTL
xdata_bus(9)	P60	BIDIR	LVTTL
xdata_bus(10)	P102	BIDIR	LVTTL
xdata_bus(11)	P101	BIDIR	LVTTL
xdata_bus(12)	P114	BIDIR	LVTTL
xdata_bus(13)	P99	BIDIR	LVTTL
xdata_bus(14)	P13	BIDIR	LVTTL
xdata_bus(15)	P131	BIDIR	LVTTL
xdata_bus(16)	P139	BIDIR	LVTTL
xdata_bus(17)	P138	BIDIR	LVTTL
xdata_bus(18)	P124	BIDIR	LVTTL
xdata_bus(19)	P85	BIDIR	LVTTL
xdata_bus(20)	P7	BIDIR	LVTTL
xdata_bus(21)	P120	BIDIR	LVTTL

*) these pins are used only in MLCA_4-DC

PIN DESCRIPTION for XC2S50, 144 pin version (cont'd)

Signal Name	Pin Number	Direction	IO Standard
xdata_bus(22)	P129	BIDIR	LVTTL
xdata_bus(23)	P134	BIDIR	LVTTL
xdata_bus(24)	P103	BIDIR	LVTTL
xdata_bus(25)	P123	BIDIR	LVTTL
xdata_bus(26)	P87	BIDIR	LVTTL
xdata_bus(27)	P93	BIDIR	LVTTL
xdata_bus(28)	P19	BIDIR	LVTTL
xdata_bus(29)	P118	BIDIR	LVTTL
xdata_bus(30)	P126	BIDIR	LVTTL
xdata_bus(31)	P5	BIDIR	LVTTL

Power Supply Pin Name	Pin Number
GND	P8, P17, P25, P33, P45, P52, P61, P73, P81, P89, P98, P110, P119, P128, P135, P143
VCC0	P1, P16, P35, P36, P53, P70, P71, P90, P107, P108, P127, P144
VCCINT	P9, P14, P24, P55, P82, P92, P97, P125

Configuration and Programming Pin Name	Pin Number
CCLK	P37
DOUT	P38
DIN	P39
/INIT	P68
/PROGRAM	P69
DONE	P72
M2	P106
M0	P109
M1	P111

Boundary Scan Pin Name	Pin Number
TCK	P2
TDI	P32
TDO	P34
TMS	P142

Design Test Pins *** DON'T CONNECT ***	Pin Number
Test1 - don't connect	P27
Test2 - don't connect	P54
Test3 - don't connect	P80
Test4 - don't connect	P78

4.3 SpartanIIE XC2S50E-TQ144 (144 pin) version

Signal Name	Pin Number	Direction	IO Standard
ad_clk	P98	OUTPUT	LVTTL
ad_cs	P121	OUTPUT	LVTTL
ad_inW	P86	INPUT	LVTTL
ad_inU	P85	INPUT	LVTTL
addr(0)	P112	INPUT	LVTTL
addr(1)	P101	INPUT	LVTTL
addr(2)	P100	INPUT	LVTTL
addr(3)	P116	INPUT	LVTTL
aux_in	P43	INPUT	LVTTL
clk	P52	INPUT	LVTTL
cs	P117	INPUT	LVTTL
dhe	P134	INPUT	LVTTL
enco_a_in	P65	INPUT	LVTTL
enco_b_in	P59	INPUT	LVTTL
enco_u_in	P23	INPUT	LVTTL
enco_v_in	P40	INPUT	LVTTL
enco_w_in	P42	INPUT	LVTTL
enco_zero	P58	INPUT	LVTTL
error	P60	OUTPUT	LVTTL
ext_latch	P124	INPUT	LVTTL
interrupt	P21	OUTPUT	LVTTL
not_mres	P69	INPUT	LVTTL
npwmW (=npwm1)	P49	OUTPUT	LVTTL
npwmU (=npwm2)	P47	OUTPUT	LVTTL
npwmV (=npwm3)	P48	OUTPUT	LVTTL
npwmZ (=npwm4) (reserved)	P104	OUTPUT	LVTTL (don't connect!) *)
pwmW (=pwm1)	P56	OUTPUT	LVTTL
pwmU (=pwm2)	P44	OUTPUT	LVTTL
pwmV (=pwm3)	P41	OUTPUT	LVTTL
pwmZ (=pwm4) (reserved)	P103	OUTPUT	LVTTL (don't connect!) *)
pwm_stop	P39	INPUT	LVTTL
rd	P96	INPUT	LVTTL
wr	P95	INPUT	LVTTL
xdata_bus(0)	P131	BIDIR	LVTTL
xdata_bus(1)	P125	BIDIR	LVTTL
xdata_bus(2)	P84	BIDIR	LVTTL
xdata_bus(3)	P83	BIDIR	LVTTL
xdata_bus(4)	P118	BIDIR	LVTTL
xdata_bus(5)	P123	BIDIR	LVTTL
xdata_bus(6)	P94	BIDIR	LVTTL
xdata_bus(7)	P122	BIDIR	LVTTL
xdata_bus(8)	P80	BIDIR	LVTTL
xdata_bus(9)	P79	BIDIR	LVTTL
xdata_bus(10)	P30	BIDIR	LVTTL
xdata_bus(11)	P27	BIDIR	LVTTL
xdata_bus(12)	P32	BIDIR	LVTTL
xdata_bus(13)	P31	BIDIR	LVTTL
xdata_bus(14)	P14	BIDIR	LVTTL
xdata_bus(15)	P24	BIDIR	LVTTL
xdata_bus(16)	P6	BIDIR	LVTTL
xdata_bus(17)	P12	BIDIR	LVTTL
xdata_bus(18)	P18	BIDIR	LVTTL
xdata_bus(19)	P20	BIDIR	LVTTL
xdata_bus(20)	P137	BIDIR	LVTTL
xdata_bus(21)	P133	BIDIR	LVTTL

*) these pins are used only in MLCA_4-DC

PIN DESCRIPTION for XC2S50E, 144 pin version (cont'd)

Signal Name	Pin Number	Direction	IO Standard
xdata_bus(22)	P132	BIDIR	LVTTL
xdata_bus(23)	P10	BIDIR	LVTTL
xdata_bus(24)	P13	BIDIR	LVTTL
xdata_bus(25)	P22	BIDIR	LVTTL
xdata_bus(26)	P29	BIDIR	LVTTL
xdata_bus(27)	P50	BIDIR	LVTTL
xdata_bus(28)	P57	BIDIR	LVTTL
xdata_bus(29)	P26	BIDIR	LVTTL
xdata_bus(30)	P11	BIDIR	LVTTL
xdata_bus(31)	P140	BIDIR	LVTTL

Power Supply Pin Name	Pin Number
GND	P1, P9, P16, P25, P34, P45, P54, P62, P70, P81, P91, P99, P110, P119, P136
VCC0	P17, P36, P53, P72, P90, P108, P128, P144
VCCINT	P19, P46, P51, P61, P88, P120, P130, P135

Configuration and Programming Pin Name	Pin Number
CCLK	P107
DOUT	P106
DIN	P105
/INIT	P74
/PROGRAM	P73
DONE	P71
M2	P37
M0	P35
M1	P33

Boundary Scan Pin Name	Pin Number
TCK	P143
TDI	P111
TDO	P109
TMS	P2

Design Test Pins *** DON'T CONNECT ***	Pin Number
Test1 - don't connect	P63
Test2 - don't connect	P64
Test3 - don't connect	P97
Test4 - don't connect	P102

5 MEMORY MAP

5.1 Registers READ

ADDRESS A(3)..A(0)	REGISTER (databus 16 bits wide)	REGISTER (databus 32 bits wide)
0000	ENCO Control Register	ENCO Control Register
0001	ENCO Interrupt Status Register	ENCO Interrupt Status Register
0010	ENCO Counter Low (bits 15..0)	ENCO Counter (bits 31..0)
0011	ENCO Counter High (bits 31..16)	ENCO Counter (bits 31..0) (copy)
0100	ENCO Capture Low (bits 15..0)	ENCO Capture (bits 31..0)
0101	ENCO Capture High (bits 31..16)	ENCO Capture (bits 31..0) (copy)
0110	ENCO Status Poll Register	ENCO Status Poll Register
0111	ENCO <i>reserved</i>	ENCO <i>reserved</i>
1000	PWM I_REF_W (bits 9..0)	PWM I_REF_W (bits 9..0)
1001	PWM I_REF_U (bits 9..0)	PWM I_REF_U (bits 9..0)
1010	PWM Parameter B0 (bits 9..0)	PWM Parameter B0 (bits 9..0)
1011	PWM Parameter B1 (bits 9..0)	PWM Parameter B1 (bits 9..0)
1100	PWM Delta	PWM Delta
1101	PWM I_MOT_W	PWM I_MOT_W
1110	PWM I_MOT_U	PWM I_MOT_U
1111	PWM <i>reserved</i>	PWM <i>reserved</i>

Table 2

5.2 Registers WRITE

ADDRESS A(3)..A(0)	COMMAND	REGISTER (databus 16 bits wide)	REGISTER (databus 32 bits wide)
0000	enco_counter off	ENCO Control Register	ENCO Control Register
0001	enco_counter off	ENCO dummy (only executes command)	ENCO dummy (only executes command)
0010	enco_counter off	ENCO Counter Low (bits 15..0)	ENCO Counter (bits 31..0)
0011	enco_counter off	ENCO Counter High (bits 31..16)	ENCO Counter (bits 31..0) (copy)
0100	enco_counter on	ENCO Control Register (copy)	ENCO Control Register (copy)
0101	enco_counter on	ENCO dummy (only executes command)	ENCO dummy (only executes command)
0110	enco_counter on	ENCO Counter Low (bits 15..0) (copy)	ENCO Counter (bits 31..0)
0111	enco_counter on	ENCO Counter High (bits 31..16) (copy)	ENCO Counter (bits 31..0) (copy)
1000	none	PWM I_REF_W (bits 9..0)	PWM I_REF_W (bits 9..0)
1001	none	PWM I_REF_U (bits 9..0)	PWM I_REF_U (bits 9..0)
1010	none	PWM Parameter B0 (bits 9..0)	PWM Parameter B0 (bits 9..0)
1011	none	PWM Parameter B1 (bits 9..0)	PWM Parameter B1 (bits 9..0)
1100	none	PWM Parameter Delta	PWM Parameter Delta
1101	none	PWM <i>reserved</i>	PWM <i>reserved</i>
1110	none	PWM <i>reserved</i>	PWM <i>reserved</i>
1111	none	PWM <i>reserved</i>	PWM <i>reserved</i>

Table 3

Note:

- A(3) determines if the access is destined to the ENCO (encoder) part or to the PWM part (current regulator):
A(3) = 0 → access to the ENCO part
A(3) = 1 → access to the PWM part (=current regulator)
Holding A(3) = 0 yields a complete compatibility with MLCA_1 (memory map, definition of the single bits the registers and functionality).
- if MLCA_4 is synthesized with the option counter/capture set to 24 bits, then bits 31.. 24 will be don't care

6 The Bus Interface

The bus interface to the external microprocessor consists of an address bus (A3.. A0), a data bus which is configurable to 16 or 32 bits (D15.. D0 respectively D31.. D0), the handshake signals /RD, /WR, /CS and an interrupt signal /INT.

Besides these signals, MLCA_4 needs an external clock signal. This clock should have an almost symmetrical waveform (duty cycle 50:50) (worst case 40:60 or 60:40) and a maximum frequency of 33 MHz.

The choice of the clock frequency influences the system performance: as a matter of fact, the PWM frequency (paragraph 8.3), the processing speed of the current regulation algorithm (paragraph 8.1) and counting speed of the "Encoder" unit (chapter 7) all depend on the clock frequency.

To optimize the access speed of the external (host) microprocessor to the MLCA_4 (therefore reducing the wait states to a minimum), it is recommended to clock the MLCA_4 with a clock signal that has a fixed phase relationship to the microprocessor's clock. In other words, the same clock used for the microprocessor should be used for the MLCA_4 as well, or a phase locked multiple or submultiple of it

A further signal generated by the MLCA_4, that can be optionally read by the microprocessor is ERROR (a flag). This signal becomes active when the encoder unit runs into an error situation. (see sect. 7.4 "Error Detection Mechanisms").

The width of the external databus can be configured to 16 or 32 bits (the internal databus always has a fixed width, defined by the word size of the counter/capture unit (either 24 or 32 bits, depending under which configuration the MLCA_4 was synthesized).

A hardware pin, called DHE (databus high enable) is used to configure the databus' width: this pin must be "hardwired" to either a '0' or '1' logic level:

DHE = '0' (low)	→	the external databus is 16 bits wide (pins D31..D16 are disabled)
DHE = '1' (high)	→	the external databus is 32 bits wide

The only registers with a width of 16 bits or more are those contained in the Encoder unit and therefore only those registers are affected by the configuration of the DHE pin. The Current Regulator Unit has registers of less than 16 bits width and consequently enabling or disabling D31..D16 has no influence on these registers.

In the following chapters, for reasons of simplicity, all registers with a width of 16 bits or less are treated as 16 bit ones. When MLCA_4 is operated in the 32 bit databus mode (DHE=1), bits D31..D16 of the external databus (and of course of the registers themselves) are "don't care".

All registers having a width greater than 16 bits (= those of the Encoder unit) will be described separately for the two operating modes (DHE = 1 and DHE = 0).

Still for simplicity, it is assumed that MLCA_4 is being synthesized with the option "counter/capture size = 32 bits" (if this is not the case, i.e. the Encoder unit has an internal databus width of 24 bits, bits D31..D24 will be "don't care").

7 The ENCODER Unit

The Encoder Unit is an interface to incremental position sensors (optical encoders, linear scales, quadrature magnetic sensors, etc).

In addition to the standard incremental signals, the unit can also handle up to three absolute tracks (useful with brushless motors). Several manufacturers offer optical encoders providing incremental + absolute signal tracks.

The Encoder Unit consists of the following parts:

- a preloadable 24 or 32 bit up/down counter (the choice between 24 or 32 bits is made at the synthesis level)
- a 24 or 32 bit capture unit
- a control unit

The Counter start/stop command

A special feature of the Encoder Unit is the start/stop mechanism of the 24/32 bit counter.

As indicated in Table 3, the start/stop command is associated to how the MLCA_4 is addressed during a write operation.

All registers of the Encoder Unit are accessible under two distinct addresses:

- a write into one of the two addresses, in addition to performing the actual write cycle, has the effect of halting the counter (=stop command). The counter remains in this state until a new start command is issued ;
- a write into the other address restarts the counter after the data was written (start command).

This mechanism allows for a reduction of the required accesses to MLCA_4.

The halting (stopping) of the counter is necessary when the same is being preloaded, in order to guarantee the consistency between the values loaded into the registers COUNTER_HIGH and COUNTER_LOW, avoiding that accidental small movements of the motor (and therefore the sensor) could disrupt the synchronization between the two registers. (It is generally advised to stop the motor completely before preloading the counter with a new value: this is the only way to ensure an exact match of counter value to a given sensor position).

When MLCA_4 is configured for an external databus width of 16 bits, the following sequence is recommended to preload a 24 (or 32) bit value into the counter:

- stop the counter by writing into the register located at address 0001 (= "dummy" register)
- preload the higher bits of the counter (COUNTER_HIGH register) by writing into address 0011 (=> the counter will stop counting)
- preload the lower 16 bits of the counter (COUNTER_LOW register) by writing into address 0110 (=> the counter is automatically re-enabled at the end of this write cycle)

When MLCA_4 is configured for an external databus width of 32 bits, the following sequence is recommended to preload a 24 (or 32) bit value into the counter:

- stop the counter by writing into the register located at address 0001 (= "dummy" register)
- preload the counter (COUNTER_32 register) by writing a 32 bit value into the address 0110 (=> the counter is automatically re-enabled at the end of this write cycle)

Further details on this subjects are given in the paragraph describing the counter unit.

The following paragraphs discuss the single building-blocks composing the Encoder Unit and their associated registers.

7.1.2.2 Encoder Error Detection

If the encoder sensor is out of calibration or defective, it can happen that the signals of its incremental tracks A and B change exactly (or almost) at the same time (correctly, they should be in quadrature):

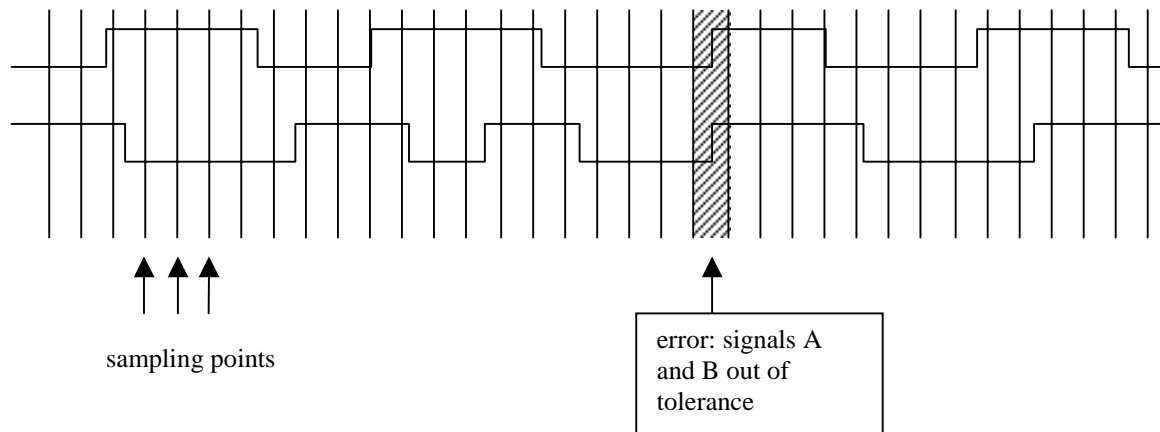


Fig. 4

This must be considered as a fatal error, since the discriminator will no longer be able to tell in which direction the motor (the sensor) was moved .

In such a situation, the counter will stop counting and enter into an error state. The microprocessor has the option of polling this state (Status_Poll Register) or receive an interrupt from the MLCA_4.

In both cases, an Interrupt Acknowledge (see ahead) is needed to exit from this error condition. The counter will then resume counting from where it stopped.

When one (or both) interrupts are enabled and an event that triggers them occurs (Zero Index or Encoder Error), the /Interrupt pin will be driven low (active) and their corresponding bit in the Interrupt_Status_Reg will be set (i.e. bit ZI and/or bit EI):

7.1.2.3 The Interrupt Status Register



x = reserved

value after a reset: LV = EI = ZI = 0 (Z reflects the actual state of the Zero Index signal)

Description of the single bits:

- Z: zero index
- ZI: zero index interrupt
- EI: error interrupt
- LV: capture latch valid

Notice: bit EI will be set even if the corresponding interrupt is not enabled (EIE=0), whereas ZI will only be set if ZIE is also set.

The bit Z reflects the actual (not latched) state of the Zero Index signal: it is '1' only during the passage of Zero Index mark.

The LV bit will be discussed together with the Capture block.

By means of the Status_Poll_Register, flags ZI and EI can be read multiple times, without being cleared. This is useful for software applications, where these bits can be polled in a background process without interfering the correct operation of the interrupt routine (which can read and clear these bits).

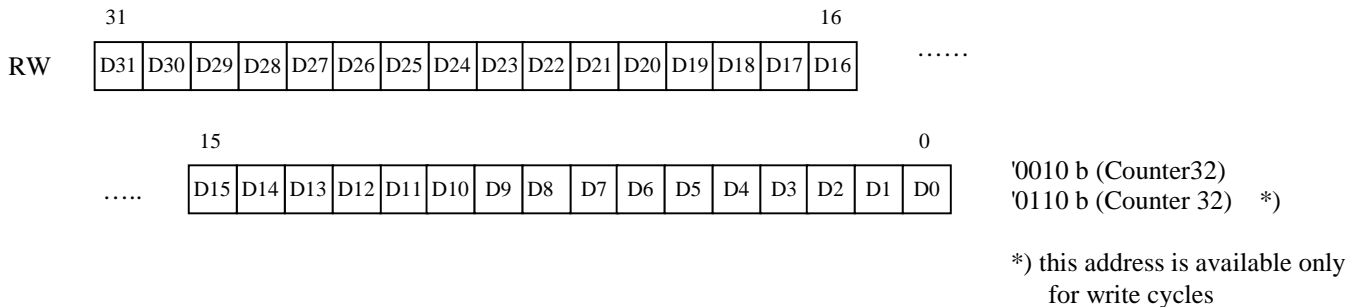
- when 16 bit resolution is enough, only Count_Lo needs to be read
- when all 32 (or 24) bits must be read, then Count_Hi must always be read before Count_Lo.

Above logic does not apply to write cycles: instead, here there is the "start/stop counter" mechanism already described earlier. (Please note that the latter really stops the counter). In order to be sure to write consistent data to the counter, it is necessary to stop the motor and the counter prior to writing.

Internally, the "Stop Counter" command puts the discriminator into a "reset" state. This helps avoid encoder errors due to motor movements (when A and B sensor signals both changed while the counter was in stop state). A further effect of a "Stop Counter" command is that pending interrupts are cleared.

The value written to the counter will be associated to the position that the motor has at the moment of issuing the next "Start Counter" command.

7.2.3 Counter32 (32 bit wide databus)



Value after a reset: 0x00000000

The mechanisms described in the previous paragraph (7.2.2) are available also in 32 bit databus mode.

However, for read cycles, it is recommended to access Count32 holding A0='0' (the shadow register will not be locked), since all 32 bits can be read in one cycle.

For write cycles instead, it is advised to follow the same sequence as with a 16 bit wide databus, i.e. stop the counter (by writing to address '0010 b) and then restart it (by writing to address '0110 b).

7.3 The Capture Unit

This block contains a logic to capture the counter's value upon a predefined event (trigger). Depending on how the MLCA_4 core was synthesized (24 or 32 bit counter), the captured value will be 24 or 32 bit wide.

When the external databus is 16 bits wide (pin DHE is '0'), the captured value can be read through two registers (read only): Capture_Hi and Capture_Lo. The way how the 24 (or 32) bits are being transferred to the external 16 bit bus is identical to that described for the Counter Unit.

When the external databus is 32 bits wide (DHE = '1'), the captured value can be read through a unique 32 bit register (Capture32, read only).

When the MLCA_4 core is synthesized with the option "Counter/Capture size = 24 bits", then bits D15..D8 of register Capture_Hi, respectively bits D31..D24 of register Capture32 will be don't care.

The operating mode of the Capture unit is determined by bit Z/E of the Control Register (Enco_Cntrl_Reg).

There are two events that can trigger a capture of the counter's value:

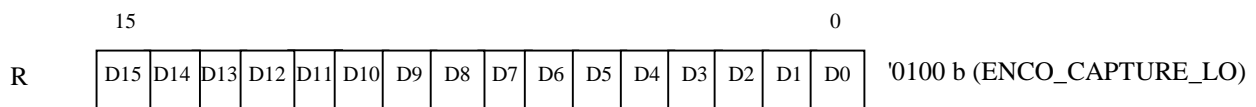
- a "zero index" mark pulse (when Z/E = 1)
- a logic '1' on pin "Ext_Latch" (when Z/E = 0). As long as this pin is held at logic '1', the Capture block keeps tracking and capturing the most recent Counter value.

The latter mode offers a wide choice of usages: e.g. by connecting pin "Ext Latch" to a limit switch, an external trigger signal, etc.

Whenever the Capture Unit acquires a value, the bit LV (readable from either the Interrupt Status Register or the Status Poll Register) will be set. This bit or flag indicates that the Capture Register contains a valid value.

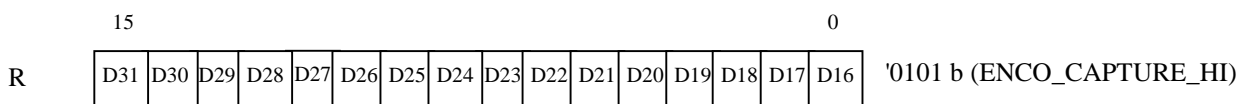
This bit is automatically cleared only after a read from Capture_Lo or Capture32 (the Capture register itself is not automatically cleared and will change only after a new trigger event).

7.3.1 Capture Low (16 bit wide databus)



Value after a reset: 0x0000

7.3.2 Capture High (16 bit wide databus)



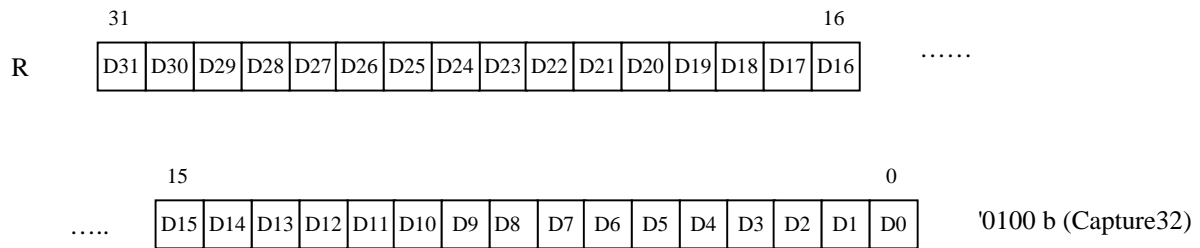
Value after a reset: 0x0000

As with the Counter, also here there is a logic that guarantees that consistent data are read from Capture_Hi and Capture_Lo: a read from Capture_Hi will inhibit any further captures until after Capture_Lo (or any other register of the Encoder unit, accessed with A0='0') is read.

Warning:

It is recommended to avoid reading for example Counter_Lo between a read of Capture_Hi and Capture_Lo since this would interfere with above inhibit logic.

7.3.3 Capture32 (32 bit wide databus)



Value after a reset: 0x00000000

The same mechanisms described in the previous paragraph (7.3.2) still work in the 32 bit databus mode. However, since it is possible to read all bits of the Capture value in one cycle, it is not necessary to inhibit the Capture Logic. It is therefore recommended to read Capture32 holding A0 = '0'.

7.4 Error Detection Mechanisms

MLCA_4 can detect errors at 3 (three) levels:

- "Encoder Error" detection. This error happens when the phase delay between signals A and B of the encoder is out of tolerance. (see paragraph 7.1.2.2 Encoder Error Detection"). This type of error must be handled by software as it is a fatal one !
- "position tracking error": when the Capture Unit is triggered by a "zero index" mark pulse, the microprocessor can determine the counter value at that exact moment. By computing the difference between two successive Capture values (the difference should be a integer multiple of the encoder resolution), the microprocessor can detect eventual loss of counts.
- "encoder wiring error": by means of a simple external circuit connected to Aux_In input, it is possible detect wiring faults of the encoder sensor signals:

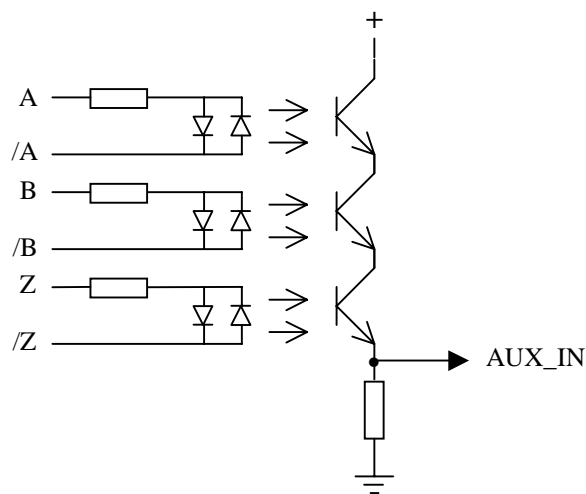


Fig. 6 : encoder wiring error detection (example)

In the circuit example of Fig. 6, signal Aux_In will assume a logic '0' if one of the wires from the encoder sensor are interrupted and therefore A and /A or B and /B or Z and /Z do no longer assume complementary logic levels. The usage of optocouplers offers the advantage not to alter the ground potential of the RS422 lines coming from the encoder.

The current limiting resistors for the LEDs must be dimensioned such that together with the RS422 termination resistors (not shown in Fig. 6) they give the correct value of approx. 100 Ω .

In order to make sure that a signal and its complementary assume the same logic level in case of a fault (making the same detectable by the logic of Fig. 6), it may be necessary to insert weak pull-up (or pull-down) resistors on all RS422 lines coming from the encoder.

Transients on the signal presented to Aux_In, which could be due to slight propagation differences of a signal and its complementary, are filtered out inside MLCA_4 (see paragraph 7.1.3 The Status Register (Status_Poll_Reg)).

7.5 Other functions

As can be seen from the pin list of MLCA_4, there's a pin called ERROR.
This pin assumes the same value as bit EI (Encoder Error) regardless whether the interrupt is enabled or not.
It can be read by the microprocessor or it can be used to drive a LED or other logic.

8 The Current Regulator Unit

This unit contains the following parts:

- the current regulator (to be more specific, there are actually two completely independent current regulators)
- the 3-phase PWM waveform generator
- the controller of pair of external analog to digital converters (ADCs)

8.1 The current regulator

The current regulator represents the innermost control loop of the whole servo system (see chapter 2 "Typical application").

The basic principle of a single phase current regulator is as follows:

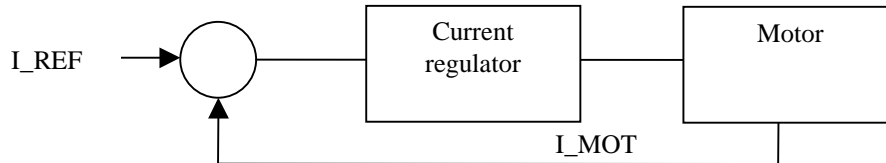


Fig. 7

Being able to drive 3-phase motors, MLCA_4 contains two identical current regulators. They control the current on two (U, W) of the three phases (U,V,W). The logic for the third phase (V) only contains a PWM waveform generator (but no current regulator). The duty cycle to be applied to the third phase is derived from the other phase's current regulator outputs.

The implemented control algorithm is of the PI (proportional-integral) type, with an open integrator. Its characteristic difference equation is:

$$y_{u,w}(k) = b_0 * u_{u,w}(k) + b_1 * u_{u,w}(k-1) + y_{u,w}(k-1) \quad (1)$$

where:

$$u_{u,w}(k) = I_REF_{u,w}(k) - I_MOT_{u,w}(k) \quad (2)$$

$$u_{u,w}(k-1) = I_REF_{u,w}(k-1) - I_MOT_{u,w}(k-1) \quad (3)$$

(the indexes u, w refer to the motor phases)

The (approximate) analog equivalent of such a regulator, is a PI whose transfer function is:

$$G(s) = Kp + \frac{Ki}{s} \quad (4)$$

The bilinear transform allows to find the following relationship between the coefficients of the time continuous and the time discrete PI controllers of equations (4) and (1):

$$b_0 = Kp + \frac{Ki \cdot T}{2}, \text{ and} \quad (5)$$

$$b_1 = \frac{Ki \cdot T}{2} - Kp \quad (6)$$

The sampling interval T is:

$$T = \frac{2048}{f} \quad ; \text{ where } f \text{ is the external clock frequency applied to MLCA}_4 \quad (7)$$

(e.g.: $f = 33.3 \text{ MHz} \rightarrow T = 61 \mu\text{sec approx.}$)

A PI controller with an "open integrator" has – as can be seen from eq. (4) – a pole at the origin of the s plane. A more generic form of PI controller would allow to have this pole anywhere on the negative real axis of the s plane (including zero).

Extensive tests that have been run have shown ([1]) that this restriction does not deteriorate the controller's behavior. This is also understandable, since the controller (including its integrator) are embedded in a closed loop. The implementation of a PI controller with an open integrator could be implemented in a very efficient manner, compared to its more generic equivalent (much lower gate usage and therefore cost).

Let's now step back and see how a three phase controller system works. In such a system, only two controllers are actually needed (i.e. two equations like eq. (1)).

The outputs $y_u(k)$, $y_w(k)$ of the two current regulators represent the reference value (=duty cycle) to the PWM waveform generators for the phases U and W of the motor.

The third PWM block (that of phase V), will receive the following reference value:

$$y_v(k) = - (y_u(k) + y_w(k)) \quad (8)$$

All the math operations of the equations described so far are executed by a custom RISC ALU implemented in MLCA_4. This ALU implements an instruction set which is specifically tailored to the kind of required operations (additions, multiplications, shifts, inversions).

As described in bibliography [1], tests were carried out to determine the necessary resolution (ALU word length, in bits) to obtain a smooth and high performance servo drive operation (the performance in a real environment was compared to that of a state-of-the-art "time continuous" ('∞' resolution), analog servo controller).

These tests indicated that a 10 bit resolution would be required and sufficient.

Therefore, the entire architecture of the Current Regulator Unit is based on a word length of 10 bits. This applies to the variables I_MOT, I_REF, parameters B0 and B1, all internal temporary variables and the RISC ALU itself with its registers.

It also means that the three PWM waveform generators offer a resolution (in terms of static "duty cycle") of 10 bits (i.e. 1024 different possible values of duty cycle).

8.2 The controller of the external analog to digital converters (ADC)

Since the FPGA is completely digital, a way must be found to digitize the motor current I_MOT: this is achieved by two external commercial ADCs (analog-to-digital converters), which are directly controlled and synchronized by MLCA_4.

As discussed above, a three phase motor can be controlled by means of just two control loops, i.e. only the current of two phases of the motor must be measured and known.

The chosen analog to digital converters are either Burr-Brown ADS7816 or Microchip MCP3201, both featuring a sample&hold circuit and execute a conversion in approx. 14μsec (when MLCA_4 is clocked at 33 MHz).

These ADCs have a synchronous serial interface. All of the necessary handshake signals to interface these chip are generated by the ADC controller logic contained in the MLCA_4.

The serial interface offers many advantages: it requires fewer pins on the FPGA side, easier PCB layout and therefore reduced size and cost.

The used ADCs are 12 bit converters. As a consequence, two out of these 12 bits are discarded by the current controller which is 10 bits wide. Depending on how the remaining 10 bits are selected from the original 12 bits, results in three possible sensitivity scales, each differing from its next by a factor of 2.

By means of two configuration bits contained in register PWM_DELTA (see sect. 8.4.3) it is possible to perform this choice, therefore selecting the sensitivity of the motor's current measure.

It must be noted that MLCA_4 was specifically customized to interface to the Burr-Brown ADS7816 or Microchip MCP3201 converters: a special microsequencer generates all the necessary handshake signals. Use of a different ADC is possible only under the condition that it must have an equivalent serial interface.

8.3 The 3 phase PWM waveform generator

This block is composed of three completely independent PWM generator units (one for each phase).

Every PWM unit operates in "symmetrical PWM" mode, which is today's industry standard (see ahead).

The reference values received by the three PWM waveform generators are:

- for phases U and W: the outputs $y_u(k)$, $y_w(k)$ of the two current controllers (see eq. (1))
- for phase V: the result of eq. (8): $y_v(k)$

In the following paragraph, we will explain the operation of one single PWM generator unit in deeper detail (the other two work in a similar way).

8.3.1 Operating principle

Basically, the PWM waveform generator unit is composed of:

- a free-running, 10 bit, up-down counter (counts $0 \rightarrow 1023 \rightarrow 0 \rightarrow 1023 \dots$), called PWM_COUNTER
- two 10 bit compare registers and their associated logic
- a logic for converting signed numbers to unsigned ones
- a control logic which also checks for, and prevents over- and underflows

From the received reference values, the PWM Unit extracts the two "compare" levels which we will call PWM_LEVEL and NPWM_LEVEL and which will then be compared against the free-running counter's value in order to generate the PWM output signals for the high and low side switch of a half bridge.

Since the compare operation is done against the actual counter value (which can range from 0 to 1023 and therefore is unsigned), it is first necessary to convert the signed reference values $y_{u,v,w}(k)$ coming from the ALU, since they range from -512 to $+511$.

In other words, the two registers PWM_LEVEL and NPWM_LEVEL contain unsigned 10 bit values, which are obtained by adding the (constant) value of $+512$ to $y_{u,v,w}(k)$.

When the dead time between the switching of the high and low side transistor is set to zero (see register PWM_DELTA), then the two compare values will be identical and correspond to $y_{u,v,w}(k) + 512$.

Conversely, when the dead time is not zero, then the two compare values will differ slightly, according to the following definitions:

$$\text{PWM_LEVEL} : \textit{remains unchanged} \tag{9}$$

$$\text{NPWM_LEVEL} \leftarrow \text{NPWM_LEVEL} - \text{PWM_DELTA} \tag{10}$$

(PWM_DELTA is the value contained in the register with the same name. It specifies the dead time between high and low side transistor switching).

A special logic prevents that the operation described by eqn. (10) yields a result outside the legal range of $0..1023$. In case of a negative result, the same will be overwritten with a 0 (zero).

The opposite case (a result greater than 1023) is intrinsically impossible. This guarantees that even in extreme situations, the low side transistor will never be left turned off for long periods of time.

This situation facilitates the use of driver ICs of the IRF-211x family (International Rectifier), which necessitate a (even though minimal) turn on time of the low side transistor in order to maintain the "bootstrap" supply voltage.

8.3.2 The PWM waveform signal generator

The following figure shows the operating principle of the PWM waveform generator:

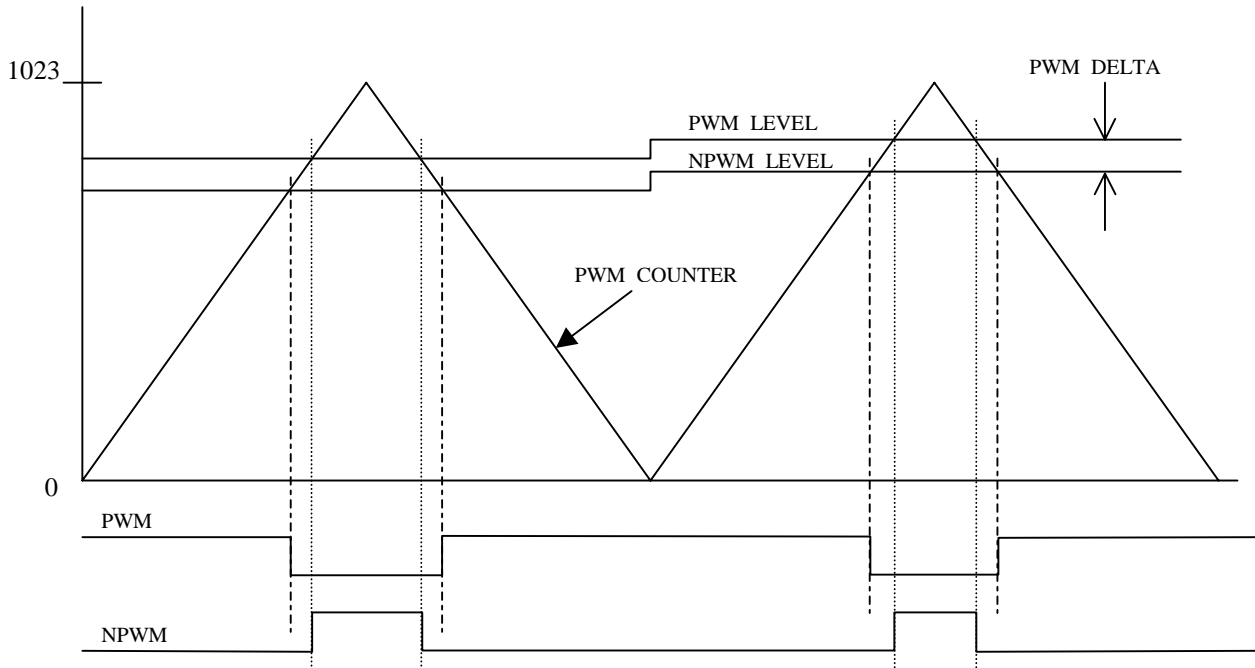


Fig. 8 operating principle of the PWM block (only 1 phase shown)

The two flags PWM and NPWM, resulting from the comparison of the free-running counter's value against the values of PWM_LEVEL and NPWM_LEVEL are exactly used to drive the PWM output pins with the same name. (please note the inversion: signal PWM results from the comparison of NPWM_LEVEL and PWM_COUNTER, whereas NPWM results from the comparison of PWM_LEVEL and PWM_COUNTER)

The signals PWM and NPWM are used to drive the power switches (transistors):
 PWM drives the high side switch (a logic "1" turns the transistor on)
 NPWM drives the low side switch (a logic "1" turns the transistor on)

The base frequency of the PWM and NPWM signals coming out of MLCA_4 is the same as that of PWM_COUNTER.

As can be seen from Fig. 8, PWM_COUNTER performs 2048 counts (1024 up and 1024 down) per each period. Since it is clocked at the same frequency as the MLCA_4's external clock, the resulting frequency of PWM_COUNTER (and therefore of the signals PWM and NPWM) will be:

$$\text{Freq}_{\text{PWM, NPWM}} = \frac{f}{2048} \quad (\text{where } f \text{ is the clock frequency of MLCA}_4) \quad (11)$$

(example: when MLCA_4 is clocked at 33.3 MHz, the resulting frequency will be: $\text{Freq}_{\text{PWM, NPWM}} = 16.2 \text{ kHz}$)

Since a symmetric modulation scheme is used (the load is connected between two phases of MLCA_4), the ripple frequency of the load current will be doubled, therefore non audible.

8.3.3 The PWM_STOP function

By means of an external signal applied to pin PWM_STOP, it is possible to force the immediate turn-off of all power switches: all PWM and NPWM outputs become logic '0'.

This function is thought for safety reasons (implementation of operator's safety features in a machine tool) or for short circuit protection in the power driver.

8.4 Registers description

8.4.1 Registers PWM_IREF_W and PWM_IREF_U

	15											0					
RW	x	x	x	x	x	x	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	'1000 b (PWM_IREF_W) '1001 b (PWM_IREF_U)

x = don't care

value after a reset: (I9..I0): 0000000000 b

These registers receive the motor current reference value, which will be used by the current regulator, starting from the next PWM cycle. The value will be stored and used as long as it is not overwritten by a newer one.

The value to be written into these registers is a signed 10 bit number:

Examples:

The most negative number is: (D9..D0): 1000000000 b = 0x200 = -512

The most positive number is: (D9..D0): 0111111111 b = 0x1FF = +511

The value of zero is: (D9..D0): 0000000000 b = 0x000 = 0

The value of +1 is: (D9..D0): 0000000001 b = 0x001 = +1

The value of -1 is: (D9..D0): 1111111111 b = 0x3FF = -1

8.4.2 Registers PWM_PARAMETER_B1 and PWM_PARAMETER_B0

	15											0					
RW	x	x	x	x	x	x	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0	'1010 b (PWM_PARAM_B0) '1011 b (PWM_PARAM_B1)

x = don't care

value after a reset: (B9..B0): 0000000000 b

These registers receive the parameters B0 and B1 of the control algorithm.

For further details about these parameters, please refer to paragraph 8.1 "The current regulator".

The value to be written to these registers is a signed 10 bit number.

Examples:

see description of registers PWM_IREF_W and PWM_IREF_U

8.4.3 Register PWM_PARAMETER_DELTA

	15											0					
RW	x	x	x	BYP	ADS_U1	ADS_U0	ADS_W1	ADS_W0	D7	D6	D5	D4	D3	D2	D1	D0	'1100 b (PWM_PARAM_DELTA)

x = don't care

default value after a reset: (D7..D0): 10000000 b = 0x80

ADS_U(1:0) = ADS_W(1:0) = "00"

This register allows to configure the following parameters:

- the dead time (bits D7..D0) between the switching of the high and low side power switches of each H-half bridge
- the sensitivity of the measured actual motor current I_MOT (sensitivity of the A/D converters) by means of bits ADS_U(1:0) and ADS_W(1:0) (ADS stands for ADC Sensitivity)
- the BYPass function, which allows to skip the entire current regulator. The values of I_REF_U and I_REF_W go directly into PWMU and PWMW (PWMV/PWMZ are calculated in consequence, as usual). In other words, when writing into I_REF_U and I_REF_W (and BYP is active), it is possible to directly change the PWM's duty cycle instead of setting the current reference value.

8.4.3.1 Configuring the dead time:

The value to be written is an unsigned 8 bit number.

The maximum dead time corresponds to (D7..D0) = 11111111 b = 255, whereas a dead time of zero corresponds to 00000000b.

Every single bit increment adds a pause of one clock period. When MLCA_4 is clocked at 33.3 MHz (\Rightarrow clock period of 30 ns) it is therefore possible to set dead times between 0 e 7.6 μ sec (= 255 * 30 ns)

The dead time written to this register acts on all three half bridges (of a 3-phase power driver).

Remark:

Internally, only the higher 6 bits of Delta are used. The two least significant bits (D1, D0) are actually ignored. It is therefore possible to configure a total of 64 different dead time values.

8.4.3.2 Configuring the sensitivity of the actual current measure

The external A/D converters deliver a current measure with a resolution of 12 bits. Internally however, MLCA_4 works with a precision of 10 bits. For this reason, not all bits delivered by the A/D converter are actually used.

By means of bits ADS_x (ADC Sensitivity) (bits 11 ... 8 of register PWM_PARAM_DELTA), it is possible to tell MLCA_4, how to choose the "useful" 10 bits (out of the 12 bits available from the ADCs).

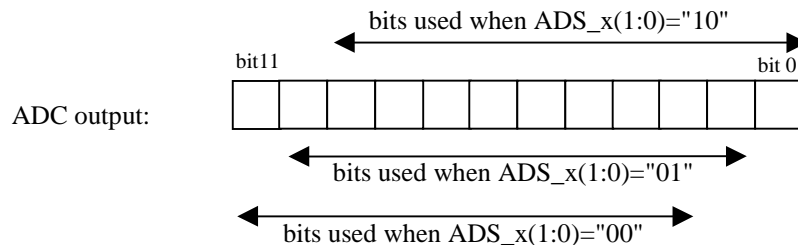


Fig. 9 ADC bits usage for the different sensitivity scales

This feature offers the possibility to choose between three sensitivity scales which are each apart by a factor of "2 x".

ADS_U sets the sensitivity of ADC which is connected to phase U, whereas, whereas ADSW sets the sensitivity corresponding to phase W.

Higher sensitivity scales are used when accurate control of small currents is needed.

For large currents (or currents with high dynamic range), it is preferable to use lower sensitivity scales.

The independent setup of the sensitivities of the two ADCs offers a maximum of flexibility to the user.

The correspondence between bits ADS_x and the effective sensitivity is explained by the following table:

ADS_x (PWM_DELTA register)		Analog input range of the AD converters			Sensitivity
bit 1	bit 0	V _{in min}	V _{in max}		
0	0	0 V	V _{ref}	normal (1x)	
0	1	1/4 V _{ref}	3/4 V _{ref}	medium (2 x)	
1	0	3/8 V _{ref}	5/8 V _{ref}	high (4 x)	
1	1	-	-	<i>illegal setting</i>	

Table 4 ADC sensitivity settings

By default (after a reset), the lowest sensitivity scale is taken (ADS_x(1:0) = "00").

Obviously, the setting of ADS_x influences the way how the sign of the value read from the ADC must be handled. This however, is done automatically by MLCA_4 in such a way that regardless of the value of ADS_x, the extracted 10 bits are correctly sign extended (2's complement) (see following examples).

In addition to that, there is a limiter that prevents overflow errors from happening when using the higher sensitivity scales (2x, 4x) and the analog input is beyond the allowable range indicated in above table. In such a situation, the value is limited to either the maximum positive (011111111 b) or the maximum negative (100000000 b) number.

Note that regardless of ADS_x, the center scale value (corresponding to a digital value of zero) is obtained by applying $V_{ref}/2$ (=center of scale for the analog inputs of Burr-Brown's ADS7816 or Microchip's MCP3201). These converters require an external V_{ref} voltage, which can be between 0.1V and 5V.

The following examples explain how this logic works:

When ADS_x(1:0) = "00" (1x scale):

The 10 bits used internally reflect bits D11..D2 delivered by the ADCs, right aligned, but with the following change:

- The most significant bit of the current (I11) is read inverted: therefore the internal representation is that of a signed number (2's complement)

Examples:

- the analog voltage applied to the ADC is 0 Volts (= beginning of scale)
=> the value delivered by the ADC is (D11..D0): 00000000000 b
=> the value used internally is (D9..D0): 1000000000 b (= signed number, most negative value)
- the analog voltage applied to the ADC is $V_{ref}/2$ (= half scale)
=> the value delivered by the ADC is (D11..D0): 10000000000 b
=> the value used internally is (D9..D0): 0000000000 b (= signed number, zero)
- the analog voltage applied to the ADC is V_{ref} (= full scale)
=> value delivered by the ADC is (D11..D0): 11111111111 b
=> the value used internally is (D9..D0): 0111111111 b (= signed number, most positive value)

With ADS_x(1:0) = "01" (2x scale):

The 10 bits used internally, reflect exactly bits D10..D1 delivered by the ADCs, unless there is a situation of over- or underflow, in which case, the number will be limited:

Examples:

- the analog voltage applied to the ADC is $V_{ref}/2$ (=half scale)
=> the value delivered by the ADC is (D11..D0): 10000000000 b
=> the value used internally is (D9..D0): 0000000000 b (= signed number, zero)
- the analog voltage applied to the ADC is $3/4 V_{ref}$ (=full scale)
=> the value delivered by the ADC is (D11..D0): 10111111111 b
=> the value used internally is (D9..D0): 0111111111 b (= signed number, most positive value)
- the analog voltage applied to the ADC is $1/4 V_{ref}$ (beginning of scale)
=> the value delivered by the ADC is (D11..D0): 01000000000 b
=> the value used internally is (D9..D0): 1000000000 b (= signed number, most negative value)
- the analog voltage applied to the ADC is $> 3/4 V_{ref}$ (out of range situation !)
=> the value used internally is (D9..D0): 0111111111 b (= signed number, most positive value)
(limiting)
- the analog voltage applied to the ADC is $< 1/4 V_{ref}$ (out of range situation !)
=> the value used internally is (D9..D0): 1000000000 b (= signed number, most negative value)
(limiting)

With ADS_x(1:0) = "10" (4x scale):

The 10 bits used internally, reflect exactly bits D9..D0 delivered by the ADCs, unless there is a situation of over- or underflow, in which case, the number will be limited:

Examples:

- the analog voltage applied to the ADC is $V_{ref}/2$ (=half scale)
=> the value delivered by the ADC is (D11..D0): 10000000000 b

- => the value used internally is (D9..D0): 0000000000 b (= signed number, zero)
- the analog voltage applied to the ADC is 5/8 Vref (=full scale)
 - => the value delivered by the ADC is (D11..D0): 100111111111 b
 - => the value used internally is (D9..D0): 0111111111 b (= signed number, most positive value)
- the analog voltage applied to the ADC is 3/8 Vref (beginning of scale)
 - => the value delivered by the ADC is (D11..D0): 011000000000 b
 - => the value used internally is (D9..D0): 1000000000 b (= signed number, most negative value)
- the analog voltage applied to the ADC is > 5/8 Vref (out of range situation !)
 - => the value used internally is (D9..D0): 0111111111 b (= signed number, most positive value) (limiting)
- the analog voltage applied to the ADC is < 3/8 Vref (out of range situation !)
 - => the value used internally is (D9..D0): 1000000000 b (= signed number, most negative value) (limiting)

8.4.3.3 Configuring the Bypass mode

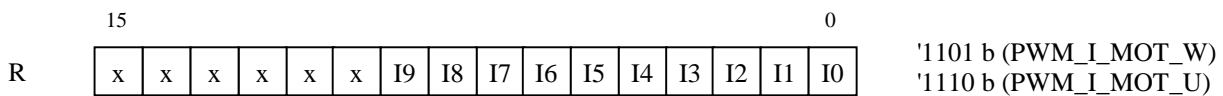
By means of this functionality, it is possible to "disable" the current controller unit. The values written into registers I_REF_U and I_REF_W are then not going into the current controller, but are transferred directly into PWMU and PWMW (in other words, they set the PWM's duty cycle). PWMV (and PWMZ, on MLCA_4-DC) are automatically calculated so that the sum of all currents is zero.

The meaning of the BYP bit is as follows:

when BYP = 0 → current regulators are operating normally (enabled)

when BYP = 1 → current regulators are disabled and bypassed ("Bypass" mode)

8.4.4 Registers PWM_I_MOT_W and PWM_I_MOT_U



x = reserved

These registers allow the actual motor current (in two of the three phases) to be read (by means of the external A/D converters). (The current of the third phase can be calculated, since the sum of all currents must be zero).

These registers are updated once every PWM period.

The two A/D converters sample their respective currents at the same moment.

The values that can be read from these registers exactly reflect the values used internally by the MLCA_4 which is 10 bits (signed). These 10 bits are extracted from the 12 bits delivered by the A/D converter in three possible modes, depending on how the bits ADS_x(1:0) (of register PWM_DELTA) are set.

Please refer to paragraph 8.4.3 for further details.

These registers are double buffered, so that at any time a read yields a stable value, even if an update is in progress.

9 Timing Diagrams

Note: these timings are obtained from a typical implementation of the core using Xilinx place&route tools. They can vary slightly from one implementation to the other. The Spartan timings were obtained running place&route in Xilinx Foundation 3.3.08i. Those for SpartanII and SpartanIIE were obtained using Xilinx ISE 4.1i tools.

The following diagrams and tables are based on the following assumptions:

- MLCA_4, implemented in either a Spartan XCS30-4 VQ100 (speed grade "-4"), or a SpartanII XC2S50-5 TQ144 (speed grade "-5"), or a SpartanIIE XC2S50E-6-TQ144 (speed grade "-6")
- clock frequency: 33.3 MHz, 50% duty cycle (40%..60%)
- ambient temperature: 27 °C
- supply voltage: within +/- 10% of specified device operating voltage (5V, 3.3V, 2.5V)

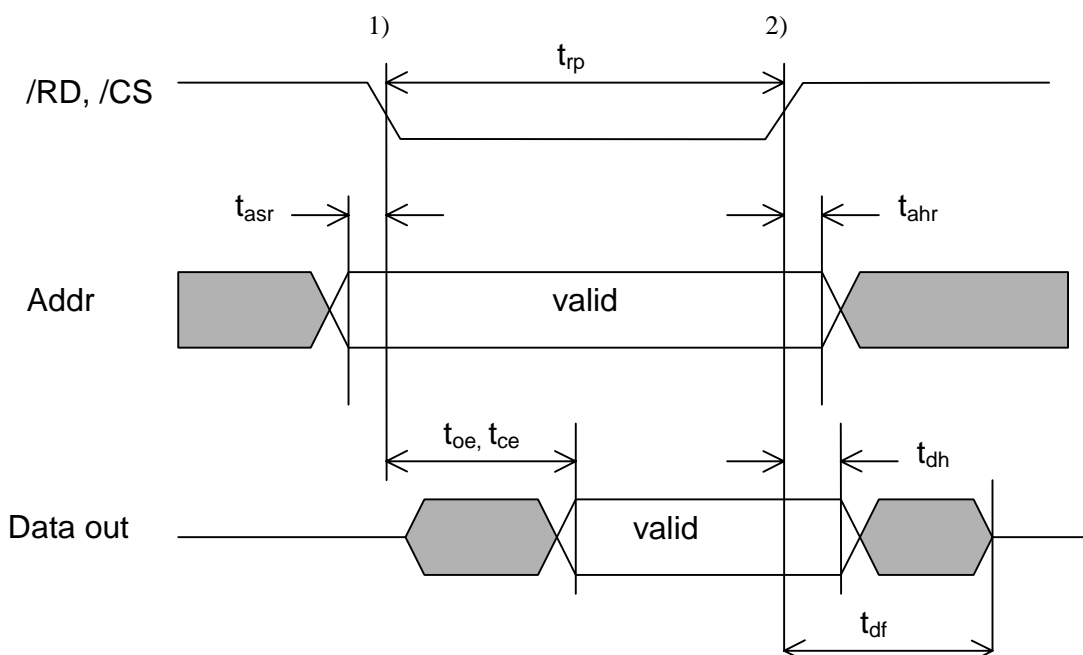
MLCA_4

READ CYCLE TIMING

specified for Spartan XCS30-4 VQ100 (build version Ver10_P1_rev_5_5_15, code "F"),
and for SpartanII XC2S50-5 TQ144 (build version Ver3_Rev1p1_rev_5_5_52),
and for SpartanIIE XC2S50E-6-TQ144 (build version Ver3_Rev1p1_rev_5_5_22),
all clocked @ 33.3 MHz, 27 °C

symbol	name	notes	value [ns]		
			Spartan XCS30	SpartanII XC2S50	SpartanIIE XC2S50E
$t_{oe} = t_{ce}$	data valid from /CE, /RD low	1)	42 max	24 max	27 max
t_{asr}	address setup time (read)	1)	5 min	5 min	5 min
t_{rp}	read pulse width	1) 2)	2T+6 min	2T-4 min	2T-20 min
t_{dh}	data hold after /RD, /CE high	2)	0 min, 7.5 typ	0 min	0 min
t_{ahr}	address hold time (read)	2)	7.2 min	5 min	5 min
t_{df}	data high impedance after /RD, /CE high	2) 3)	20 typ, 27 max	13 typ, 16 max	11 typ, 14 max

Note: $T = 1 / f_{clock}$



Notes (read cycle timings):

- 1) internally, /CS and /RD are ORed: therefore these timings are referred to the one that changes last
- 2) for the same reason, these timings are referred to the one that changes first
- 3) this value is only for information (not 100% tested) and depends on the actual bus load
- 4) the indicated timing values apply to both the cases where the external databus is 32 bits wide (DHE = 1) or where it is 16 bits wide (DHE = 0)

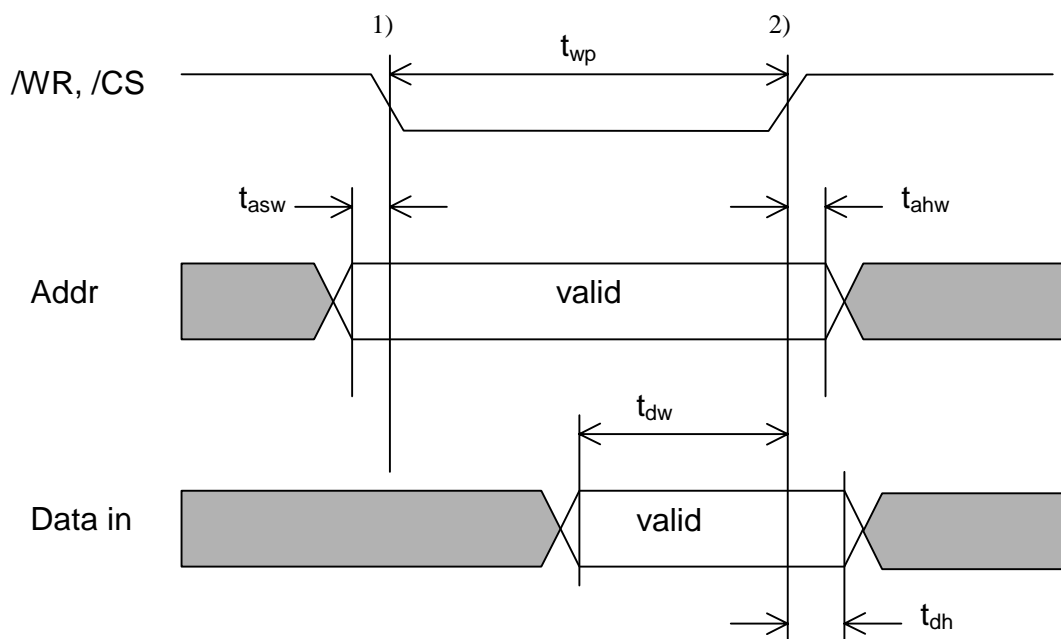
MLCA_4

WRITE CYCLE TIMING

specified for XCS30-4 VQ100 (build version Ver10_P1_rev_5_5_15, code "F"),
 and for XC2S50-5 TQ144 (build version Ver3_Rev1p1_rev_5_5_52),
 and for SpartanIIE XC2S50E-6-TQ144 (build version Ver3_Rev1p1_rev_5_5_22),
 all clocked @ 33.3 MHz, 27 °C

symbol	name	notes	value [ns]		
			Spartan XCS30	SpartanII XC2S50	SpartanIIE XC2S50E
t_{asw}	address setup time (write)	1)	2 min	2 min	2 min
t_{wp}	write pulse width	1) 2)	$2T+18$ min	$2T+5$ min	$2T+5$ min
t_{dw}	data setup time to /CE, /WR high	2)	$T+14$ min	$T+8$ min	$T+3$ min
t_{dh}	data hold after /WR, /CE high	2)	10 min	5 min	8 min
t_{ahw}	address hold time (write)	2)	10 min	5 min	5 min

Note: $T = 1 / f_{clock}$



Notes (write cycle timings):

- 1) internally, /CS and /RD are ORed: therefore these timings are referred to the one that changes last
- 2) for the same reason, these timings are referred to the one that changes first
- 3) the indicated timing values apply to both the cases where the external databus is 32 bits wide (DHE = 1) or where it is 16 bits wide (DHE = 0)

MLCA_4

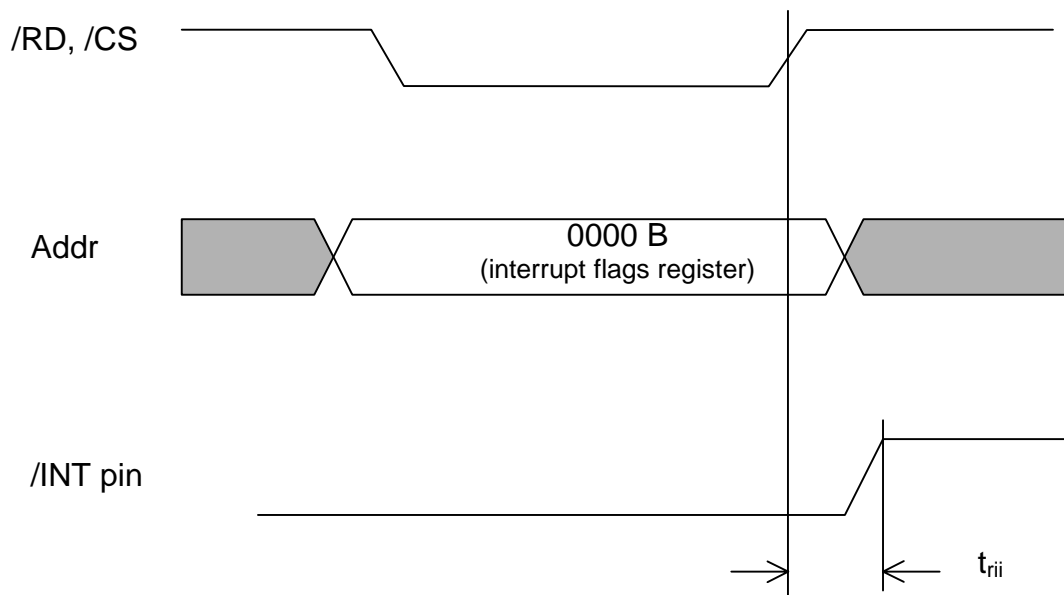
INTERRUPT ACKNOWLEDGE TIMING

specified for XCS30-4 VQ100 (build version Ver10_P1_rev_5_5_15, code "F"),
 and for XC2S50-5 TQ144 (build version Ver3_Rev1p1_rev_5_5_52),
 and for SpartanIIe XC2S50E-6-TQ144 (build version Ver3_Rev1p1_rev_5_5_22),
 all clocked @ 33.3 MHz, 27 °C

symbol	name	notes	value [ns]		
			Spartan XCS30	SpartanII XC2S50	SpartanIIe XC2S50E
t _{rii}	/INT high after interrupt register read	-	T+26 max	T+12 max	T+16 min

for all other timings please refer to READ CYCLE TIMINGS

Note: T = 1 / f_{clock}



MLCA_4

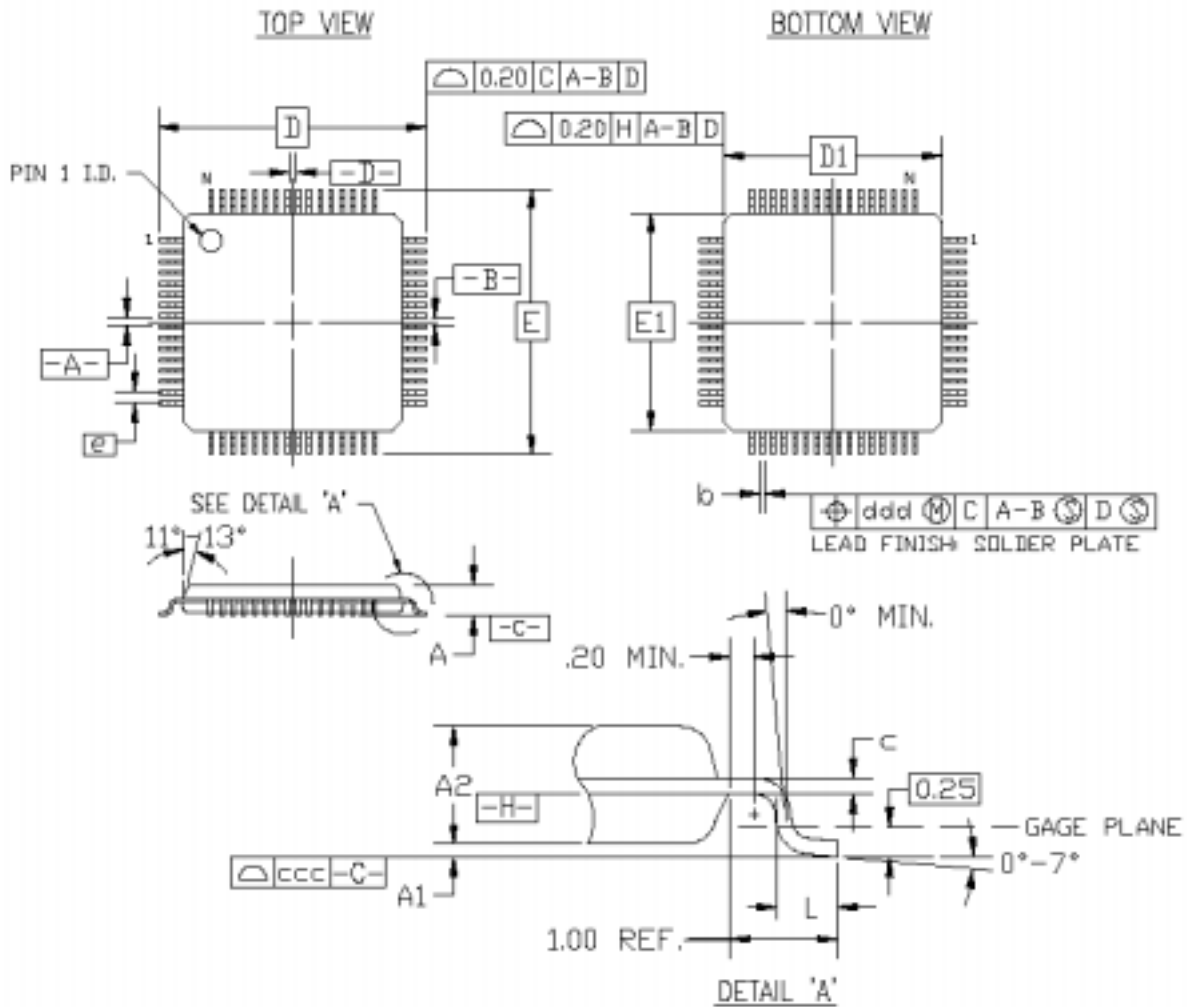
OTHER TIMING INFORMATION

specified for XCS30-4 VQ100 (build version Ver10_P1_rev_5_5_15, code "F"),
and for XC2S50-5 TQ144 (build version Ver3_Rev1p1_rev_5_5_52),
and for SpartanIIIE XC2S50E-6-TQ144 (build version Ver3_Rev1p1_rev_5_5_22),
all clocked @ 33.3 MHz, 27 °C

- $T = 1/f_{\text{clock}}$
- guaranteed f_{clock} frequency: 33.3 MHz (and higher on SpartanII/IIE)
- current regulation servo loop sampling frequency: $f_{\text{clock}} / 2048$ (=16.1 kHz @ 33 MHz)
- time for a current value written into the MLCA_4 to become effective:
a value written into the MLCA_4 will be synchronized and used in the next sampling period of the current control loop. The max. delay is therefore $2048 / f_{\text{clock}} = 61.44 \mu\text{sec}$
- minimum width of a pulse on the "Ext_Latch" input pin to be recognized: $1/f_{\text{clock}}$ (30 ns @ 33 MHz)
- A/D converter reading: every $T=2048 / f_{\text{clock}} = 61.44 \mu\text{sec}$ (both ADCs)
- A/D converter serial shift clock: $f_{\text{clock}}/32 = 1.04 \text{ MHz}$

10 Package Dimensions

10.1 VQ100 (very thin quad flat pack), for Spartan XCS30 version

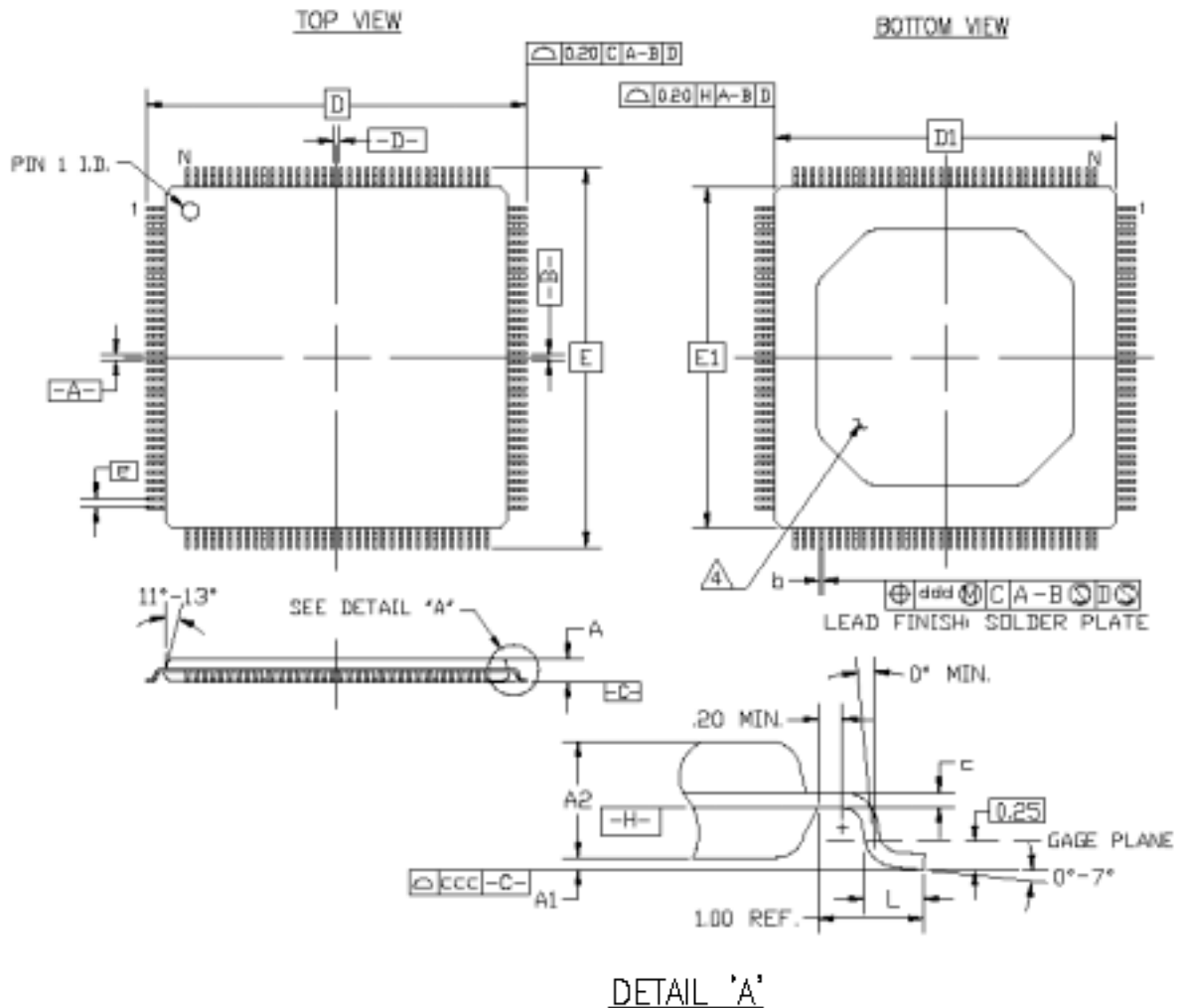


VQ100		
MILLIMETERS		
MIN.	NOM.	MAX.
$\frac{1.20}{16}$	$\frac{1.20}{16}$	1.20
0.05	0.10	0.15
0.95	1.00	1.05
16.00 BSC.		
14.00 BSC.		
0.17	0.22	0.27
0.09	$\frac{0.20}{16}$	0.20
0.50 BSC.		
0.45	0.60	0.75
$\frac{0.08}{16}$	$\frac{0.08}{16}$	0.08
$\frac{0.08}{16}$	$\frac{0.08}{16}$	0.08
100		
JEDEC MS-026-AED		

NOTES:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1982.
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION SHALL NOT EXCEED 0.25mm PER SIDE.
3. THE TOP OF PACKAGE MAY BE SMALLER THAN THE BOTTOM OF PACKAGE BY 0.15mm.

10.2 TQ144/HQ144 (TQFP/Heat sink TQFP), for SpartanII/IIe (XC2S50 and XC2S50E versions)



TQ/HT144		
MILLIMETERS		
MIN.	NOM.	MAX.
\sim	\sim	1.60
0.05	0.10	0.15
1.35	1.40	1.45
22.00 BSC		
20.00 BSC		
0.45	0.60	0.75
0.50 BSC		
0.17	0.22	0.27
0.09	\sim	0.20
\sim	\sim	0.08
\sim	\sim	0.08
144		
JEDEC MS-026-BFB		

NOTE:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5-1982
2. DIMENSION D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION SHALL NOT EXCEED 0.25mm PER SIDE.
3. PACKAGE TOP DIMENSION MAY BE SMALLER THAN THE BOTTOM DIMENSION BY 0.15mm.

4. THE SAME PACKAGE DIMENSIONS APPLY FOR THERMALLY ENHANCED PRODUCTS. HEAT SINK IS ADDED. THE PACKAGE CODE IS "HT".

A APPENDIX

A.1 Bibliography

[1] Studio di fattibilità-Regolatore di corrente digitale (**in software**). R. Monleone, 19.01.1998

A.2 MLCA_4 core installation and implementation instructions

To place and route the design, we recommend using directly the "Xilinx Design Manager" application (under the Accessories of the Xilinx start menu items). We have tested the implementation (place & route) both under Xilinx Foundation 3.3i and Xilinx ISE 4.1i tools versions.

The deliverables contain netlists of the synthesized core. These are the following files:

Spartan XCS30 version:

- **mlca_4_databus_32_bit.edf** (EDIF file containing the whole design)
- **Xcounter.ngc** (Xilinx NGC file format, containing the description of a counter used by above file)

SpartanII XC2S50 version:

- **mlca_4_databus_32_bit_xc2s.edf** (EDIF file containing the whole design)
- **coregen_counter32.edn** (EDIF file, containing the description of a counter used by above file)

SpartanIIE XC2S50E version:

- **mlca_4_databus_32_bit_xc2se.edf** (EDIF file containing the whole design)
- **coregen_counter32.edn** (EDIF file, containing the description of a counter used by above file)

Both files are required when the design is to be placed and routed. The first is the design source that must be chosen when defining the new project under Xilinx Design Manager. The second file must be placed into the same directory as the first one, in order to be found.

For best place & route results, it is recommended to use the enclosed "**MLCA_4.ucf**" (for Spartan XCS30), respectively "**MLCA_4_xc2s.ucf**" (for Spartan2 XC2S50) or "**MLCA_4_xc2se.ucf**" (for Spartan2 XC2S50E) file, which is a user constraints file for Xilinx Foundation or ISE tools (tested with both Foundation 3.3i and ISE 4.1i). This file contains timing and pin locking constraints.

It is important to check the place & route log file about success or failure to meet these timing constraints. In case of difficulties achieving the timing constraints, try multipass place & route.

If you plan to include the core into an existing VHDL design, there is also a file containing a VHDL instantiation template for the MLCA_4 core, named **MLCA_4.vhi** (you may use it, if you like).

For simulation purposes, there are two files:

- **mlca_4_databus_32_bit_sim.vhd** (for Spartan XCS30), respectively **mlca_4_databus_32_bit_xc2s_sim.vhd** (for Spartan XC2S50) or **mlca_4_databus_32_bit_xc2se_sim.vhd** (for Spartan XC2S50E) : a VHDL simulation model of the synthesized MLCA_4 core (pre-place & route simulation).
- **Test_MLCA4.vhd** (a VHDL testbench file, ready-to-use to test above simulation model file). You can edit this file to adapt it to your testing needs, if you like.

You can use these files to simulate the design using a VHDL design/simulation software (we have tested it using Aldec™ ActiveVHDL).

For post-place & route timing simulation, you will have to generate your own post-p&r simulation output file². This is obtained by checking "Produce Post Layout Timing Reports" under Implementation Options and by also checking "Correlate Simulation Data to Input Design", under Simulation Options, and then running the "Timing (Sim)" step in the Xilinx Design Manager "Flow Engine".

The Flow Engine will then generate a post place & route VHDL simulation model file (*.VHD) as well as an *.SDF timing information file. You will need both to run post place & route timing simulation.

The same testbench file (Test_MLCA4.vhd) used for pre-place & route simulation can also be used to run simulations of the post place & route simulation model. You may however have to edit the core instantiation

² Upon request, we can deliver a completely placed & routed (stand-alone) design along with its post p&r simulation model files. This design is ready to be burned into a serial configuration PROM.

statements in this file, in order to adapt them to the actual VHDL entity and architecture names used by the simulation model file (instructions and examples are provided inside the testbench file itself).

Note:

Output signals PWMx and NPWMx have different indexing conventions in this document and in the core files. Here are the correspondences:

<u>MLCA_4 documentation / datasheet</u>	<u>MLCA_4 core files (VHDL, EDIF)</u>
pwmW, npwmW	pwm1, npwm1
pwmU, npwmU	pwm2, npwm2
pwmV, npwmV	pwm3, npwm3
<u>pwmZ, npwmZ (used only in MLCA_4-DC)</u>	<u>pwm4, npwm4 (used only in MLCA_4-DC)</u>

Included in the deliverables you will also find this document (Adobe™ Acrobat format).

A.3 Revision History

A.3.1 Hardware

Date	Version	Build Code	Description/Purpose	Author
17.05.1999	Rev. A	p2_rev_4_4_39	First fully tested (simulated) version MLCA_4 (complete) Extracted timing parameters.	D.C, R.M.
01.06.1999	Rev. B	p1_rev_4_4_31	Moved some pins to avoid conflicts with boundary scan. Simulated again and extracted timing parameters. Tested in target hardware.	D.C.
20.12.1999	Rev. C	p3_rev_5_5_22	Extended counter/capture to 32 bits. New configurable bus interface (16 or 32 bit). New shadow registers for I_MOT1 and I_MOT2. Two configurable levels of sensitivity for actual current.	D.C.
01.03.2000	Rev. D	-	Extended architecture of PWM unit to support alternatively two DC motors or one AC Brushless motor. Not_clk2 no longer used. Improved synchronization of register updates to avoid timing violation in case of asynchronous reads, writes or captures. Added counter auto zero function (ACZ). Discovered timing problems with 32 bit encoder counter when clocked at 33 MHz.	D.C.
25.04.2000	Rev. E	xxx	Redesigned process p_counter_next. Replaced the synthesized 32 bit counter with one by Xilinx Logiblox.	D.C.
23.05.2000	Rev. F	p1_rev_5_5_28	Redesigned process p_stop_counter, added process p_stop_counter_next. Pin compatible to MLCA_4DC, Rev. B	D.C.
10.12.2001	Rev. Gs (Spartan)	XCS, Ver10_ p1_rev_5_5_15 (code "F")	Added Bypass function. All write registers can now be read back. New synthesis and P&R using latest tool versions (Foundation 3.3-08 and ISE 4.1). Bug fix: minimum pulse duration on PWM is now on lowside. Carried some internal signals to external pins for better testbenching. Changed support for external ADC to Burr-Brown ADS7816 or Microchip MPC3201 (dropped support for Sipex SP8530, since part is obsolete). Added third ADC sensitivity scale: now available 1x/2x/4x. Improved external bus access timing. Changed pin naming and memory map. Pin compatible to MLCA_4DC, Rev. C	D.C.
03.05.2002	Rev. Gs2 (SpartanII)	XC2S, Ver3_ p1_rev_5_5_52	Ported Rev.G design to SpartanIIE family architecture (same features, different pinout). MLCA_4 is now being supported on both Spartan and SpartanII. In the new SpartanII version, the 32 bit counter, formerly implemented in a Logiblox™, was replaced by a similar one implemented with a Xilinx CoreGenerator™ core	R.M.
10.06.2002	Rev. Gs2e (SpartanIIE)	XC2SE, Ver3_ p1_rev_5_5_22	Same as Rev. 2A, but ported from SpartanII to SpartanIIE. Different pinout and configuration bitstream than Rev. 2A !!	D.C.

A.3.2 Driver Software

Date	Version	Description/Purpose	Author
17.08.1999	V1.0	Downloading and Testing driver routines for the 'C32	WDBo
19.12.2001	V1.1	Adapted drivers to HW revision G	IRP

A.3.3 Documentation

Date	Version	Description/Purpose	Author
15.06.1999	Rev. 1	First draft of Datasheet with preliminary timing information for HW Rev. B	R.M.
23.06.1999	Rev. 2	Definitive timing for HW Rev. B	R.M.
17.08.1999	Rev. 2A	Detailed registers' definitions for PWM Unit	R.M.
01.09.1999	Rev. 3	Completed chapters regarding encoder unit	R.M.
13.01.2000	Rev. 4	Updated documentation to HW Rev. C	R.M.
23.05.2000	Rev. 5	Updated documentation to HW Rev. E+F	R.M.
07.01.2002	Rev. 6	Updated documentation to HW Rev. Gs. Modified indexing of PWM block registers (old: -1, -2, -3; new: -U, -V, -W). Renamed registers using english names.	R.M.
10.06.2002	Rev. 7	Updated documentation to include alternative implementation of the design alternatively in a Spartan II, XC2S50-TQ144, or a SpartanIIE, XC2S50E-TQ144 (HW Versions Gs2 and Gs2e)	R.M.

NOTES:

MEET reserves the right to make changes without further notice to any product described herein to improve reliability, function or design. MEET does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. MEET products are not authorized for use as components in life support devices or systems intended for surgical implant into the body or intended to support or sustain life. Buyer agrees to notify MEET of any such intended end use whereupon MEET shall determine availability and suitability of its product or products for the use intended. MEET and + MEET are registered trademarks of MEET, Ltd., Coldrerio (Switzerland)