1300 Henley Court
Pullman, WA 99163
509.334.6306
www.digilent.com

Eclypse Platform MCU Specification

Revised January 23, 2020

Table of Contents

Overview

The Syzygy specification requires all pods (peripherals) to include a peripheral microcontroller unit with nonvolatile memory that has been programmed to contain a variety of hardware parameters that may be retrieved by a "SmartVIO controller" utilizing a shared I2C bus. These parameters include the load requirements for the fixed 5.0 and 3.3V supplies, the load requirements for the adjustable VIO supply, a set of supported VIO supply voltage ranges, as well as other optional things such as a manufacturer name string, product name string, model number, etc, and make up what is known as a pod's "DNA". Any carrier board that includes a Syzygy port is required implement a SmartVIO controller that, at a minimum, will retrieve the current and voltage requirements for all attached pods and only enable power to the VIO pins of the port after having determined a common voltage that is supported by the carrier and all pods which share the same VIO group, or supply. Each Syzygy port may have its own dedicated VIO power supply or two or more ports may share the same VIO power supply. Additionally, each Syzygy port may share the same fixed 5.0V and 3.3V supplies, or they may have their own dedicated supplies. A pod can consume up to 2.0 amps from any of the 3 supply rails (5.0V, 3.3V, and VIO) but a carrier is not required to supply a minimum current on any of these rails.

The primary purpose of the Eclypse Platform MCU is to implement a SmartVIO controller to enumerate any Syzygy ports present on the board, retrieve the voltage and current requirements for each pod that is connected to those ports, determine the required VIO voltage for each VIO group, and then configure the associated Power Management Unit (PMU) to supply those voltages. The Eclypse Platform MCU is also capable of performing other tasks such as monitoring analog temperature probes, monitoring and/or controlling onboard fans, and implementing an I2C Slave interface that can be used by a SOC/FPGA to determine the configuration of the board, as well as change some settings. The following list provides a high-level overview of the functionality provided by the Eclypse Platform MCU.

- Integrated SmartVIO controller with support for up to 8 SmartVIO ports
    - Supports Standard and Transceiver SYZYGY Ports and is expandable to support FMC
    - Supports up to 8 SmartVIO groups (independent VIO supply rails)
    - Supports driving up to 8 enable signals associated with PMU's for each VIO group
    - Supports monitoring up to 8 power good signals associated with each VIO group
    - Supports up to 4 independent 5.0V supply rails for SmartVIO ports
    - Supports up to 4 independent 3.3V supply rails for SmartVIO ports
    - Option to enforce or ignore 5.0V, 3.3V, and VIO current limits
        - Configuration stored in EEPROM
- Supports up to 4 analog Temperature Probes
- Supports up to 4 Fan Headers
    - Optional RPM measurement for Fans 1 and 2
    - Optional power on/off control for all four fans
    - Optional speed control for all four fans
    - Fan configuration is stored in EEPROM
- Optional support for setting DDRVCC voltage based on jumper/switch setting
- Optional support for holding INIT_B low to delay SOC/FPGA configuration
- Optional support for configuring Microchip USB251xB USB Hub using SMBUS (I2C)
- Provides a configuration interface with SOC/FPGA via I2C bus

1 MCU

The Eclypse Platform MCU is implemented using the Atmel ATmega328PB-AU.

2 Clocking

The PMCU utilizes an 8 MHz crystal connected to the XTAL1 and XTAL2 pins to generate an 8 MHz system clock used by the CPU and the peripherals. This allows for accurate measurement (1µs resolution) of the pulse width output by an optional fan connected to the FAN header. Additionally, it provides the ability to communicate with I2C devices with clock frequencies of 100 KHz (TWBR = 32, TWPS = 0) and 400 KHz (TWBR = 2, TWPS = 0).

3 SmartVIO Controller

The SmartVIO Controller implemented by the Eclypse Platform MCU firmware can support up to 4 independent 5.0V supplies, up to 4 independent 3.3V supplies, up to 8 independent VIO (VADJ) supplies, and up to 8 SmartVIO. Each VIO supply may have one or more SmartVIO ports associated with it, may have an enable signal associated with it, and may have a power good signal associated with it. However, there are no hard requirements for any VIO supply to have a SmartVIO port associated with it, or for enable or power good signals to be present. These are all optional features that are specific to each board variant. The number of 5.0V supplies, 3.3V supplies, VIO (VADJ) supplies, and SmartVIO ports that a specific board variant contains can be determined by reading the *5V0_GROUP_COUNT*, *3V3_GROUP_COUNT*, *VADJ_GROUP_COUNT*, and *SMARTVIO_PORT_COUNT* configuration registers through the use of the SOC/FPGA I2C Slave Interface.

Each SmartVIO port has six read-only configuration registers associated with it. Five of the six registers are used to specify the board-specific configuration of the port, which includes its I2C address (*PORT_n_I2C_ADDRESS*), the 5V0 group that it belongs to (*PORT_n_5V0_GROUP*), the 3V3 group that it belongs to (*PORT_n_3V3_GROUP*), the VIO (VADJ) group that it belongs to (*PORT_n_VIO_GROUP*), and its physical port type (*PORT_n_TYPE*). The sixth configuration register indicates the status of the port, which includes bitfields to indicate if a pod/mezzanine card is present, if it is within the current limits specified for the associated supply rails, if a double-wide module is installed, and if the associated VIO supply is allowed to be enabled (*PORT_n_STATUS*).

PORT_n_TYPE

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ptype							

ptype Port Type
- 0: None. Reserved for SmartVioPort entries that correspond to an entire VIO group
- 1: Syzygy Standard port with Syzygy default standard port pinout
- 2: Syzygy Transceiver-2 (TXR-2) port, 2-lane transceiver port
- 3: Syzygy Transceiver-4 (TXR-4) port, 4-lane transceiver port

All other values are reserved for future use

PORT_n_STATUS

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
fAllowVioEnable	RSV2	RSV1	fVioInLimit	f3v3InLimit	f5v0InLimit	fDW	fPresent

fPresent Present
- 1: A pod or mezzanine card is present at this port
- 0: No pod or mezzanine card is present at this port

fDW Doublewide
- 1: The pod or mezzanine card is a double-wide module
- 0: The pod or mezzanine card is not a double-wide module

f5v0InLimit 5V0 current within limit
- 1: This port's current requirements are within limit for the associated 5V0 supply
- 0: This port's current requirements are not within limit for the associated 5V0 supply

f3v3InLimit 3V3 current within limit
- 1: This port's current requirements are within limit for the associated 3V3 supply
- 0: This port's current requirements are not within limit for the associated 3V3 supply

fVioInLimit VIO current within limit
- 1: This port's current requirements are within limit for the associated VIO supply
- 0: This port's current requirements are not within limit for the associated VIO supply

RSV1 Reserved for future use

RSV2 Reserved for future use

fAllowVioEnable Allow VIO supply to be enabled
- 1: Allow this port's associated VIO supply to be enabled
- 0: Do NOT allow this port's associated VIO supply to be enabled

After each power on or reset event has occurred, the PMCU firmware initializes its SmartVIO and configuration registers (in RAM) with default values defined for the specific board variant. Any configuration settings that are stored in nonvolatile memory are read from the EEPROM and used to initialize the associated configuration registers/variables. The firmware then proceeds to initialize the peripherals that are utilized by the specific board variant. It then waits 100 milliseconds, as per the Syzygy specification, to allow all 3.3V supplies to become stable. Once 100 milliseconds have passed, the firmware proceeds to configure the TWI1 peripheral as a master and utilizes it to attempt to retrieve the Syzygy DNA header from each SmartVIO port that has been designated as a Syzygy port. If a pod is present and its DNA header has been successfully retrieved then the maximum 5V load current, maximum 3.3V load current, maximum VIO load current, the attribute flags, and the four VIO voltage ranges will be parsed from the header and used to update the SmartVIO configuration associated with the port.

After the firmware is done populating the SmartVIO configuration structure with information parsed from the DNA headers of any installed pods, it determines the minimum and maximum voltage that can be set for each VIO group based on the requirements of any pods that are present and members of the same group. Once the voltage requirements are known for each VIO group the firmware proceeds to determine the current requirement for each 5V0 group, 3V3 group, and VIO group and make sure that each is within spec for the associate power supplies. If the sum of current required for all pods within a VIO group is within the limit for any enforced current limit (5V0, 3V3, or VIO) then the PMCU firmware will designate the VIO supply as being allowed to enable power.

PLATFORM_CONFIGURATION

Byte 1	Byte 0
pcfgMSB	pcfgLSB

pcfgLSB

Bits 7:4	Bit 3	Bit 2	Bit 1	Bit 0
RSV1	fPerformCrcCheck	fEnforceVioCurLimit	fEnforce3v3CurLimit	fEnforce5v0CurLimit

fEnforce5v0CurLimit Enforce current limits for 5V0 supplies
- 1: Enforce the current limits for the 5V0 supplies
- 0: Do not enforce the current limits for the 5V0 supplies

fEnforce3v3CurLimit Enforce current limits for 3V3 supplies
- 1: Enforce the current limits for the 3V3 supplies
- 0: Do not enforce the current limits for the 3V3 supplies

fEnforceVioCurLimit Enforce current limits for VIO supplies
- 1: Enforce the current limits for the VIO supplies
- 0: Do not enforce the current limits for the VIO supplies

fPerformCrcCheck Perform CRC check over DNA header when reading the DNA header
- 1: Perform CRC check
- 0: Skip CRC check

RSV1 Reserved for future use

Note: changes made to the fEnforce5v0CurLimit, fEnforce3v3CurLimit, fEnforceVioCurLimit, and fPerformCrcCheck fields have no immediate impact and don't take effect until the next time the processor is reset. To apply changes immediately, issue a reset command after writing any changes to the PLATFORM_CONFIGURATION register.

pcfgMSB

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
RSV2							

RSV2 Reserved for future use

Once the current limits have been checked, all *PORT_n_STATUS*, all *5V0_n_CURRENT_REQUESTED*, all *3V3_n_CURRENT_REQUESTED*, and all *VADJ_n_CURRENT_REQUESTED* configuration registers will be updated. The firmware will then proceed to set the VADJ voltage to midpoint of the supported VIO voltage range for all VIO supplies that can be enabled and enable those supplies. The associated *VADJ_n_VOLTAGE* registers will be updated and the *VADJ_STATUS* register is updated to reflect the status of the VIO supplies. At this point, the firmware proceeds to reconfigure the TWI1 peripheral as a slave listening at I2C address 0x60 and enters the main loop, where it listens for commands, captures temperature readings, captures fan speed readings, and executes any specified commands as needed until power is removed or until a reset command is received over the I2C slave interface.

Once the PMCU has entered the main loop, it becomes capable of responding to I2C read/write requests from any SOC/FPGA connected to the SYZYGY I2C bus. The SOC/FPGA acts as a bus master and may read and/or write any configuration registers that are defined for the SOC/FPGA I2C Slave Interface. Additionally, the SOC/FPGA may send a reset command, which will cause the PMCU to perform a software reset, reinitialize all peripherals and configuration registers, reperform the SmartVIO discovery process and reconfigure the PMUs.

4 Temperature Probes

The Eclypse Platform MCU supports reading temperature measurements from up to 4 analog temperature probes connected to its analog input pins. The ADC is configured to utilize the internal bandgap reference (1.1V) and to operate in single conversion mode using the slowest conversion clock available to the Atmega328PB (62.5KHz). ADC conversions are initiated at a rate that results in the voltage of each supported temperature probe being sampled and converted one time per second. The firmware saves the conversion result and adjusts the analog input mux (as necessary) to measure the next channel. The ADC sample values are converted into temperature measurements and loaded into user-readable registers.

Each new temperature measurement is stored in a configuration register associated with the probe and can be retrieved by the SOC/FPGA by reading the *TEMPERATURE_n* register, where n corresponds to the probe number (1, 2, 3, or 4). Each temperature probe has an attribute register associated with it, *TEMPERATURE_n_ATTRIBUTES*, that specifies if the probe is present on the board, the point of measurement for the probe, and the format of the temperature as stored in the *TEMPERATURE_n* register.

The *TEMPERATURE_PROBE_COUNT* register may be read to determine how many probes a board contains. If a board contains less than 4 probes, then the lowest ordinal registers will correspond to the probes that are present. For example, if a board only contains 2 temperature probes then the *TEMPERATURE_1* and *TEMPERATURE_2* registers will contain the temperature measurements for those probes, and the *TEMPERATURE_3* and *TEMPERATURE_4* will always return 0 when read.

TEMPERATURE_n_ATTRIBUTES

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RSV1		DataFormat		MeasurementPoint			fPresent

fPresent Present
- 1: This temperature probe is present
- 0: This temperature probe is not present

MeasurementPoint Probe point of measurement
- 0: FPGA / CPU 1
- 1: FPGA / CPU 2
- 2: PCB Location 1
- 3: PCB Location 2

All other settings reserved for future use

DataFormat Temperature measurement data format
- 0: Degrees C, signed decimal
- 1: Degrees C, signed 10.6 fixed point
- 2: Fahrenheit, signed decimal
- 3: Fahrenheit, signed 10.6 fixed point

RSV1 Reserved for future use

TEMPERATURE_n

Byte 1	Byte 0
tempMSB	tempLSB

tempLSB Least significant byte of 16-bit temperature measurement
tempMSB Most significant byte of 16-bit temperature measurement

5 Fans

The Eclypse Platform MCU can control and/or monitor up to 4 fans. The number of fans supported by a board is hard coded into the firmware at compile time and may be determined by reading the *FAN_COUNT* register. Each fan has 3 registers associated with it. The *FAN_n_CAPABILITIES* register contains bitfields that indicate what functionality is supported by the fan, including the ability to enable/disable the fan, the ability to set the fan's speed, and the ability to measure the fan's RPM. Some fans may support all these features, while others may only support a subset of this functionality.

Each fan has a configuration register associated with it that may be used to specify certain things about its configuration, such as if the fan is enabled or disabled, its speed, and which temperature probe is utilized for automatic speed control when the fan is configured for automatic speed control. Some, all, or none of a fan's configuration may be modified by writing its associated *FAN_n_CONFIGURATION* register. A fan's supported capabilities are defined in its associated *FAN_n_CAPABILITIES* register. Any attempt to specify an unsupported configuration will result in the unsupported changes being ignored. When a *FAN_n_CONFIGURATION* register is written, any accepted changes are also written to EEPROM. The PMCU firmware will read the EEPROM each time it initializes, allowing for changes to a fan's configuration to persist across power cycles and resets.

A fan that supports the speed selection capability can be configured to operate at one of three fixed speeds: minimum, medium, and maximum. These different speed settings may be achieved by adjusting a fan's supply voltage, or they may be achieved by sending a fan control signal (PWM) to adjust its speed. What a specific speed setting equates to in RPM and decibel level is application specific and is not specified by this document. If a fan supports the speed selection capability, then the *fcapSetSpeed* bit will be set to a '1' in the associated *FAN_n_CAPABILITIES* register.

In addition to supporting settable fixed speeds, some fans may also support automatic speed control, where fan speed is determined automatically based on the temperature measured by the temperature probe specified as the temperature source for the fan. If a fan supports automatic speed control, then both the *fcapSetSpeed* and *fcapAutoSpeed* bits will be set to '1' in the *FAN_n_CAPABILITIES* register. Automatic speed control can be enabled by specifying a value of '3' for *FanSpeed* field in the *FAN_n_CONFIGURATION* register. The temperature probe utilized to implement automatic speed fan control can be specified by writing a '1', '2', '3', or a '4' to the *TemperatureProbeSelected* field of the *FAN_n_CONFIGURATION* register.

If a fan supports RPM measurement then the *fcapMeasureRpm* bit will be set to a '1' in the associated *FAN_n_CAPABILITIES* and the measured RPM will be stored as an unsigned 16-bit integer and may be retrieved by the SOC/FPGA by reading the *FAN_n_RPM* register. If RPM measurement isn't supported, then the associated RPM register will always return 0 when read. The current implementation of the PMCU has optional support for measuring RPM for Fan 1 and Fan 2 and if those features are enabled then the *FAN_n_RPM* registers are updated with a new RPM measurement once per second.

The RPM of each fan is measured by using a timer to count the number of rising edges produced by the fan feedback sensor over a given period. The timer output is captured once per second, and converted to RPM using the formula $RPM = 30 \times count\_rising\_edge \times \left(\frac{1}{t\_n}\right)$ where t_n is the period over which the count value was measured. Therefore, the formula for calculating RPM simplifies to $RPM = 30 \times count\_rising\_edge$.

FAN_n_CAPABILITIES

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RSV1				fcapMeasureRpm	fcapAutoSpeed	fcapSetSpeed	fcapEnable

fcapEnable Enable / Disable Fan
- 1: This fan supports enable / disable functionality
- 0: This fan does not support enable / disable functionality

fcapSetSpeed Fan Speed Selection
- 1: This fan supports speed selection
- 0: This fan's speed is fixed

fcapAutoSpeed Fan Automatic Speed Control
- 1: This fan supports temperature-based automatic speed control
- 0: This fan does not support automatic speed control

fcapMeasureRpm Fan RPM Measurement
- 1: This fan supports RPM measurement
- 0: This fan does not support RPM measurement

RSV1 Reserved for future use

FAN_n_CONFIGURATION

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RSV1		TemperatureProbeSelected				FanSpeed	fEnable

fEnable Enable / Disable Fan
- 1: Turn on the fan
- 0: Turn off the fan

FanSpeed Fan Speed Selection
- 0: Minimum speed
- 1: Medium speed
- 2: Maximum speed
- 3: Automatic speed control (temperature-based)

TemperatureProbeSelected Temperature Probe used for automatic fan speed control
- 0: None
- 1: Temperature Probe 1
- 2: Temperature Probe 2
- 3: Temperature Probe 3
- 4: Temperature Probe 4

All other settings reserved for future use

RSV1 Reserved for future use

FAN_n_RPM

Byte 1	Byte 0
rpmMSB	rpmLSB

rpmLSB Least significant byte of unsigned 16-bit RPM measurement
rpmMSB Most significant byte of unsigned 16-bit RPM measurement

6 PMU Interface

The PMCU interfaces with the PMUs using the TWI0 (I2C) peripheral, which is connected to the PMU's I2C bus through a level translator (PMU_SCL_LS and PMU_SDA_LS). Once the PMCU has powered on and initialized it will attempt to read the DNA of any SYZYGY pod that is attached to the SYZYGY connectors. Assuming that the DNA has been successfully retrieved, and the onboard supplies can be configured to meet the requirements of any attached pod(s), the PMCU will use the appropriate I2C interface to configure the PMU with the correct VADJn supply voltages, then drive a '1' onto the VADJn_PWR_EN net to enable the supplies. The associated *VADJ_n_VOLTAGE* registers will be updated to contain the voltage that has been set, and the VADJ_STATUS register will be updated to reflect the enable/disable status for each supply.

The SOC/FPGA may write the *VADJ_n_OVERRIDE register* using the SOC/FPGA I2C Slave Interface to override the voltage and enable settings for each individual VIO supply. Each time one of these registers is written the PMCU firmware checks to see if the specified configuration is valid. If the default VADJ settings are being overridden to enable a VIO supply then the PMCU will check to make sure that it's safe to enable the supply and that the specified voltage is within the allowable range for the VIO group based on all pods/mezzanine cards connected to any ports that are a member of the VIO group, as well as any board/port specific voltage restrictions. If the specified voltage is outside of the allowable range, then it is forced to be the minimum or maximum allowed voltage for that VIO group. PMU commands are queued and processed in the main loop.

The firmware regularly checks the PMU command queue for new commands and monitors the status of any power good signals that may be connected to the MCU. The associated power good bit is updated in the *VADJ_STATUS* register any time the state of a power good signal changes. Additionally, any optional *SYZYGY_n_DET* signal that may be present is driven to the appropriate state to indicate if a pod is present.

5V0_n_CURRENT_ALLOWED

Byte 1	Byte 0
crntAllowed	

crntAllowed 16-bit unsigned maximum steady state current (DC) that the supply can provide in milliamps

5V0_n_CURRENT_REQUESTED

Byte 1	Byte 0
crntRequested	

crntRequested 16-bit unsigned total current (in milliamps) requested by all SmartVIO ports associated with this supply

3V3_n_CURRENT_ALLOWED

Byte 1	Byte 0
crntAllowed	

crntAllowed 16-bit unsigned maximum steady state current (DC) that the supply can provide in milliamps

3V3_n_CURRENT_REQUESTED

Byte 1	Byte 0
crntRequested	

crntRequested 16-bit unsigned total current (in milliamps) requested by all SmartVIO ports associated with this supply

VADJ_n_CURRENT_ALLOWED

Byte 1	Byte 0
crntAllowed	

crntAllowed 16-bit unsigned maximum steady state current (DC) that the supply can provide in milliamps

VADJ_n_CURRENT_REQUESTED

Byte 1	Byte 0
crntRequested	

crntRequested 16-bit unsigned total current (in milliamps) requested by all SmartVIO ports associated with this supply

VADJ_n_VOLTAGE

Byte 1	Byte 0
vltgApplied	

vltgApplied 16-bit unsigned voltage applied by the supply, specified in 10mV increments

VADJ_n_OVERRIDE

Byte	1								0							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	fOverride	fEnable	RSV1				vltgSet10mV									

vltgSet10mV 10-bit unsigned voltage to set, specified in 10mV increments

RSV1 Reserved for future use

fEnable Enable or disable the supply

1: Enable (turn on) the supply
0: Disable (turn off) the supply

fOverride Override this supply's voltage setting and enable/disable state

1: Override this supply's settings with the settings specified in this register
0: Do not override this supply's settings

Note: the settings in this register have no effect unless the fOverride field is set to a '1'.

VADJ_STATUS

Byte 1	Byte 0
fsPgood	fsEnable

fsEnable	fVadjHEn	fVadjGEn	fVadjFEn	fVadjEEn	fVadjDEn	fVadjCEn	fVadjBEn	fVadjAEn

fVadjAEn VADJA Power Enable / Disable Status
1: Power supply is enabled
0: Power supply is disabled

fVadjBEn VADJB Power Enable / Disable Status
1: Power supply is enabled
0: Power supply is disabled

fVadjCEn VADJC Power Enable / Disable Status
1: Power supply is enabled
0: Power supply is disabled

fVadjDEn VADJD Power Enable / Disable Status
1: Power supply is enabled
0: Power supply is disabled

fVadjEEn VADJE Power Enable / Disable Status
1: Power supply is enabled
0: Power supply is disabled

fVadjFEn VADJF Power Enable / Disable Status
1: Power supply is enabled
0: Power supply is disabled

fVadjGEn VADJG Power Enable / Disable Status
1: Power supply is enabled
0: Power supply is disabled

fVadjHEn VADJH Power Enable / Disable Status
- 1: Power supply is enabled
- 0: Power supply is disabled

fsPgood

fVadjHPg	fVadjGPg	fVadjFPg	fVadjEPg	fVadjDPg	fVadjCPg	fVadjBPg	fVadjAPg

fVadjAPg VADJA Power Good Status
- 1: Power supply is on and within normal operating parameters
- 0: Power supply is off or outside of normal operating parameters

fVadjBPg VADJB Power Good Status
- 1: Power supply is on and within normal operating parameters
- 0: Power supply is off or outside of normal operating parameters

fVadjCPg VADJC Power Good Status
- 1: Power supply is on and within normal operating parameters
- 0: Power supply is off or outside of normal operating parameters

fVadjDPg VADJD Power Good Status
- 1: Power supply is on and within normal operating parameters
- 0: Power supply is off or outside of normal operating parameters

fVadjEPg VADJE Power Good Status
- 1: Power supply is on and within normal operating parameters
- 0: Power supply is off or outside of normal operating parameters

fVadjFPg VADJF Power Good Status
- 1: Power supply is on and within normal operating parameters
- 0: Power supply is off or outside of normal operating parameters

fVadjGPg VADJG Power Good Status
- 1: Power supply is on and within normal operating parameters
- 0: Power supply is off or outside of normal operating parameters

fVadjHPg VADJH Power Good Status
- 1: Power supply is on and within normal operating parameters
- 0: Power supply is off or outside of normal operating parameters

7 SOC/FPGA I2C Interface

Both the Eclypse Platform MCU (PMCU) and the primary SOC/FPGA share the same I2C bus with any Syzygy or FMC ports that are included on the board. Shortly after power-on/reset the PMCU firmware connects to I2C bus as a bus master and attempts to discover any pods / mezzanine cards that are connected to the board's expansion ports. Once the PMCU is done enumerating the onboard ports, it updates the power supply configuration as necessary, then switches the TWI1 (I2C) peripheral to slave mode and listens for bus address 0x60. The PMCU firmware then enters the main loop where it listens for and executes commands and maintains automatically updated registers. Once the PMCU firmware reaches this state, the primary SOC/FPGA can communicate with the PMCU using the I2C bus. Since the PMCU does not operate in multi-master mode the SOC/FPGA should wait a minimum of 500 milliseconds before taking control of the I2C bus in order to avoid causing the PMCU to lose arbitration and failing to enumerate the SmartVIO ports.

Once 500 milliseconds elapses, the SOC/FPGA may communicate with the PMCU using register read and register write transactions. A register write transaction is performed using the following sequence:

1. SOC/FPGA places start condition on the I2C bus
2. SOC/FPGA sends SLA+W
3. SOC/FPGA writes high byte of the 16-bit address
4. SOC/FPGA writes low byte of the 16-bit address
5. SOC/FPGA writes up to 2 bytes of data
6. SOC/FPGA places stop condition on the I2C bus

Note: *Some registers are 8-bits wide, while others are 16-bits wide. The PMCU firmware buffers the data that it receives from the master and doesn't write it to the addressed register until a STOP or Repeated Start condition is detected on the I2C bus.*

Register read transactions are performed using the following sequence:

1. SOC/FPGA places start condition on the I2C bus
2. SOC/FPGA sends SLA+W
3. SOC/FPGA writes high byte of the 16-bit address
4. SOC/FPGA writes low byte of the 16-bit address
5. SOC/FPGA places repeated start condition on the I2C bus
6. SOC/FPGA sends SLA+R
7. SOC/FPGA reads up to 32 bytes of data
8. SOC/FPGA places stop condition on the I2C bus

Note: *When the PMCU firmware detects an SLA+R that matches its I2C address, it copies 32 bytes of the configuration register space to a temporary buffer, starting at the last address that was received from the SOC/FPGA. If copying 32 bytes from the current register address results in exceeding the implemented register space, then the PMCU firmware will wrap around to register address 0x0000 and continue copying from there until the buffer contains 32 bytes. This allows the SOC/FPGA to read up to 32 coherent bytes while other PMCU tasks are free to update the temperature, fan, and other registers in the background without causing data corruption. If the SOC/FPGA attempts to read more than 32 bytes in a single transaction, then the PMCU will return 0xFF for all additional bytes that are read.*

The PMCU address space is split into three different sections, which are detailed in the table below:

Register Space	Start Address	End Address
PDID	0x0000	0x0003
Firmware Version	0x0004	0x0005
Reserved for future use	0x0006	0x7FFE
Reset Register	0x7FFF	0x7FFF
Configuration Registers	0x8000	0xFFFF

The PMCU can be reset by writing a non-zero value to the Reset Register. If the SOC/FPGA writes a non-zero value to the Reset Register, then it should release the I2C bus and wait 500 milliseconds before attempting to utilize the bus again.

7.1 Configuration Register Set Summary

Configuration Register Set Summary (starting at 0x8000)

Register Address Offset	Register Name	Size [Bytes]	R/W	Description
0x0000	RESERVED1	2	R	Reserved for future use (should read 0xFFFF)
0x0002	Configuration Version	2	R	Platform MCU Configuration Revision Lower Byte = VMIN Upper Byte = VMJR
0x0004	PLATFORM_CONFIGURATION	2	R/W	[0] = Enforce 5V0 Current Limit [1] = Enforce 3V3 Current Limit [2] = Enforce VIO Current Limit [3] = Perform CRC Check of DNA Header [15:4] = Reserved Note: the contents of this register are stored in EEPROM
0x0006	TEMPERATURE_PROBE_COUNT	1	R	Number of Temperature Probes
0x0007	FAN_COUNT	1	R	Number of Fan Connectors
0x0008	5V0_GROUP_COUNT	1	R	Number of 5V0 Supplies
0x0009	3V3_GROUP_COUNT	1	R	Number of 3V3 Supplies
0x000A	VADJ_GROUP_COUNT	1	R	Number of Adjustable VIO Supplies
0x000B	SMARTVIO_PORT_COUNT	1	R	Number of Expansion Ports
0x000C	TEMPERATURE_1_ATTRIBUTES	1	R	Temperature Probe 1 Attributes [0] = Present [3:1] = Probe Measurement Point 0b000 = FPGA/CPU 1 0b001 = FPGA/CPU 2 0b010 = External 1 0b011 = External 2 0b1xx = Reserved [5:4] = Data Format 0b00 = Decimal, Degrees C 0b01 = Fixed Point, Degrees C 0b10 = Decimal, Degrees F, 0b11 = Fixed Point, Degrees F [7:6] = Reserved
0x000D	TEMPERATURE_1	2	R	Temperature Measurement 1
0x000F	TEMPERATURE_2_ATTRIBUTES	1	R	Temperature Probe 2 Attributes [0] = Present [3:1] = Probe Measurement Point 0b000 = FPGA/CPU 1 0b001 = FPGA/CPU 2 0b010 = External 1 0b011 = External 2 0b1xx = Reserved [5:4] = Data Format 0b00 = Decimal, Degrees C 0b01 = Fixed Point, Degrees C 0b10 = Decimal, Degrees F, 0b11 = Fixed Point, Degrees F [7:6] = Reserved
0x0010	TEMPERATURE_2	2	R	Temperature Measurement 2
0x0012	TEMPERATURE_3_ATTRIBUTES	1	R	Temperature Probe 3 Attributes [0] = Present [3:1] = Probe Measurement Point 0b000 = FPGA/CPU 1 0b001 = FPGA/CPU 2 0b010 = External 1 0b011 = External 2

				0b1xx = Reserved [5:4] = Data Format 0b00 = Decimal, Degrees C 0b01 = Fixed Point, Degrees C 0b10 = Decimal, Degrees F, 0b11 = Fixed Point, Degrees F [7:6] = Reserved
0x0013	TEMPERATURE_3	2	R	Temperature Measurement 3
0x0015	TEMPERATURE_4_ATTRIBUTES	1	R	Temperature Probe 4 Attributes [0] = Present [3:1] = Probe Measurement Point 0b000 = FPGA/CPU 1 0b001 = FPGA/CPU 2 0b010 = External 1 0b011 = External 2 0b1xx = Reserved [5:4] = Data Format 0b00 = Decimal, Degrees C 0b01 = Fixed Point, Degrees C 0b10 = Decimal, Degrees F, 0b11 = Fixed Point, Degrees F [7:6] = Reserved
0x0016	TEMPERATURE_4	2	R	Temperature Measurement 4
0x0018	FAN_1_CAPABILITIES	1	R	Fan 1 Capabilities [0] = Enable/Disable [1] = Set Fixed Speed [2] = Automatic Speed Control [3] = RPM Measurement [4:7] = Reserved
0x0019	FAN_1_CONFIGURATION	1	R/W	Fan 1 Configuration [0] = Enable/Disable [2:1] = Fan Speed 0b00 = Minimum Speed 0b01 = Medium Speed 0b10 = Maximum Speed 0b11 = Auto (Temperature Based) [5:3] = Temperature Source 0b000 = None 0b001 = Temperature Probe 1 0b010 = Temperature Probe 2 0b011 = Temperature Probe 3 0b100 = Temperature Probe 4 [7:6] = reserved Note: the contents of this register are stored in EEPROM and read each POR
0x001A	FAN_1_RPM	2	R	Fan 1 Speed (RPM)
0x001C	FAN_2_CAPABILITIES	1	R	Fan 2 Capabilities [0] = Enable/Disable [1] = Set Fixed Speed [2] = Automatic Speed Control [3] = RPM Measurement [4:7] = Reserved
0x001D	FAN_2_CONFIGURATION	1	R/W	Fan 2 Configuration [0] = Enable/Disable [2:1] = Fan Speed 0b00 = Minimum Speed 0b01 = Medium Speed 0b10 = Maximum Speed 0b11 = Auto (Temperature Based) [5:3] = Temperature Source 0b000 = None

				0b001 = Temperature Probe 1 0b010 = Temperature Probe 2 0b011 = Temperature Probe 3 0b100 = Temperature Probe 4 [7:6] = reserved Note: the contents of this register are stored in EEPROM and read each POR
0x001E	FAN_2_RPM	2	R	Fan 2 Speed (RPM)
0x0020	FAN_3_CAPABILITIES	1	R	Fan 3 Capabilities [0] = Enable/Disable [1] = Set Fixed Speed [2] = Automatic Speed Control [3] = RPM Measurement [4:7] = Reserved
0x0021	FAN_3_CONFIGURATION	1	R/W	Fan 3 Configuration [0] = Enable/Disable [2:1] = Fan Speed 0b00 = Minimum Speed 0b01 = Medium Speed 0b10 = Maximum Speed 0b11 = Auto (Temperature Based) [5:3] = Temperature Source 0b000 = None 0b001 = Temperature Probe 1 0b010 = Temperature Probe 2 0b011 = Temperature Probe 3 0b100 = Temperature Probe 4 [7:6] = reserved Note: the contents of this register are stored in EEPROM and read each POR
0x0022	FAN_3_RPM	2	R	Fan 3 Speed (RPM)
0x0024	FAN_4_CAPABILITIES	1	R	Fan 4 Capabilities [0] = Enable/Disable [1] = Set Fixed Speed [2] = Automatic Speed Control [3] = RPM Measurement [4:7] = Reserved
0x0025	FAN_4_CONFIGURATION	1	R/W	Fan 4 Configuration [0] = Enable/Disable [2:1] = Fan Speed 0b00 = Minimum Speed 0b01 = Medium Speed 0b10 = Maximum Speed 0b11 = Auto (Temperature Based) [5:3] = Temperature Source 0b000 = None 0b001 = Temperature Probe 1 0b010 = Temperature Probe 2 0b011 = Temperature Probe 3 0b100 = Temperature Probe 4 [7:6] = reserved Note: the contents of this register are stored in EEPROM and read each POR
0x0026	FAN_4_RPM	2	R	Fan 4 Speed (RPM)
0x0028	5V0_A_CURRENT_ALLOWED	2	R	5V0 Supply A Current Allowed
0x002A	5V0_A_CURRENT_REQUESTED	2	R	5V0 Supply A Current Requested
0x002C	5V0_B_CURRENT_ALLOWED	2	R	5V0 Supply B Current Allowed
0x002E	5V0_B_CURRENT_REQUESTED	2	R	5V0 Supply B Current Requested
0x0030	5V0_C_CURRENT_ALLOWED	2	R	5V0 Supply C Current Allowed
0x0032	5V0_C_CURRENT_REQUESTED	2	R	5V0 Supply C Current Requested
0x0034	5V0_D_CURRENT_ALLOWED	2	R	5V0 Supply D Current Allowed

0x0036	5V0_D_CURRENT_REQUESTED	2	R	5V0 Supply D Current Requested
0x0038	3V3_A_CURRENT_ALLOWED	2	R	3V3 Supply A Current Allowed
0x003A	3V3_A_CURRENT_REQUESTED	2	R	3V3 Supply A Current Requested
0x003C	3V3_B_CURRENT_ALLOWED	2	R	3V3 Supply B Current Allowed
0x003E	3V3_B_CURRENT_REQUESTED	2	R	3V3 Supply B Current Requested
0x0040	3V3_C_CURRENT_ALLOWED	2	R	3V3 Supply C Current Allowed
0x0042	3V3_C_CURRENT_REQUESTED	2	R	3V3 Supply C Current Requested
0x0044	3V3_D_CURRENT_ALLOWED	2	R	3V3 Supply D Current Allowed
0x0046	3V3_D_CURRENT_REQUESTED	2	R	3V3 Supply D Current Requested
0x0048	VADJ_A_CURRENT_ALLOWED	2	R	VADJ A Supply Current Allowed
0x004A	VADJ_A_CURRENT_REQUESTED	2	R	VADJ A Supply Current Requested
0x004C	VADJ_A_VOLTAGE	2	R	VADJ A voltage in 10 mV
0x004E	VADJ_A_OVERRIDE	2	R/W	OVERRIDE VADJ Voltage Selection [9:0] = VADJ Voltage in 10mV [13:10] = RESERVED [14] = ENABLE_VADJ [15] = OVERRIDE
0x0050	VADJ_B_CURRENT_ALLOWED	2	R	VADJ B Supply Current Allowed
0x0052	VADJ_B_CURRENT_REQUESTED	2	R	VADJ B Supply Current Requested
0x0054	VADJ_B_VOLTAGE	2	R	VADJ B voltage in 10 mV
0x0056	VADJ_B_OVERRIDE	2	R/W	OVERRIDE VADJ Voltage Selection [9:0] = VADJ Voltage in 10mV [13:10] = RESERVED [14] = ENABLE_VADJ [15] = OVERRIDE
0x0058	VADJ_C_CURRENT_ALLOWED	2	R	VADJ C Supply Current Allowed
0x005A	VADJ_C_CURRENT_REQUESTED	2	R	VADJ C Supply Current Requested
0x005C	VADJ_C_VOLTAGE	2	R	VADJ C voltage in 10 mV
0x005E	VADJ_C_OVERRIDE	2	R/W	OVERRIDE VADJ Voltage Selection [9:0] = VADJ Voltage in 10mV [13:10] = RESERVED [14] = ENABLE_VADJ [15] = OVERRIDE
0x0060	VADJ_D_CURRENT_ALLOWED	2	R	VADJ D Supply Current Allowed
0x0062	VADJ_D_CURRENT_REQUESTED	2	R	VADJ D Supply Current Requested
0x0064	VADJ_D_VOLTAGE	2	R	VADJ D voltage in 10 mV
0x0066	VADJ_D_OVERRIDE	2	R/W	OVERRIDE VADJ Voltage Selection [9:0] = VADJ Voltage in 10mV [13:10] = RESERVED [14] = ENABLE_VADJ [15] = OVERRIDE
0x0068	VADJ_E_CURRENT_ALLOWED	2	R	VADJ E Supply Current Allowed
0x006A	VADJ_E_CURRENT_REQUESTED	2	R	VADJ E Supply Current Requested
0x006C	VADJ_E_VOLTAGE	2	R	VADJ E voltage in 10 mV
0x006E	VADJ_E_OVERRIDE	2	R/W	OVERRIDE VADJ Voltage Selection [9:0] = VADJ Voltage in 10mV [13:10] = RESERVED [14] = ENABLE_VADJ [15] = OVERRIDE
0x0070	VADJ_F_CURRENT_ALLOWED	2	R	VADJ F Supply Current Allowed
0x0072	VADJ_F_CURRENT_REQUESTED	2	R	VADJ F Supply Current Requested
0x0074	VADJ_F_VOLTAGE	2	R	VADJ F voltage in 10 mV
0x0076	VADJ_F_OVERRIDE	2	R/W	OVERRIDE VADJ Voltage Selection [9:0] = VADJ Voltage in 10mV [13:10] = RESERVED [14] = ENABLE_VADJ [15] = OVERRIDE
0x0078	VADJ_G_CURRENT_ALLOWED	2	R	VADJ G Supply Current Allowed
0x007A	VADJ_G_CURRENT_REQUESTED	2	R	VADJ G Supply Current Requested
0x007C	VADJ_G_VOLTAGE	2	R	VADJ G voltage in 10 mV

0x007E	VADJ_G_OVERRIDE	2	R/W	OVERRIDE VADJ Voltage Selection [9:0] = VADJ Voltage in 10mV [13:10] = RESERVED [14] = ENABLE_VADJ [15] = OVERRIDE
0x0080	VADJ_H_CURRENT_ALLOWED	2	R	VADJ H Supply Current Allowed
0x0082	VADJ_H_CURRENT_REQUESTED	2	R	VADJ H Supply Current Requested
0x0084	VADJ_H_VOLTAGE	2	R	VADJ H voltage in 10 mV
0x0086	VADJ_H_OVERRIDE	2	R/W	OVERRIDE VADJ Voltage Selection [9:0] = VADJ Voltage in 10mV [13:10] = RESERVED [14] = ENABLE_VADJ [15] = OVERRIDE
0x0088	VADJ_STATUS	2	R	Bitfield indicating the status of the VADJ supplies [0] = VADJ_A_EN [1] = VADJ_B_EN [2] = VADJ_C_EN [3] = VADJ_D_EN [4] = VADJ_E_EN [5] = VADJ_F_EN [6] = VADJ_G_EN [7] = VADJ_H_EN [8] = VADJ_A_PGOOD [9] = VADJ_B_PGOOD [10] = VADJ_C_PGOOD [11] = VADJ_D_PGOOD [12] = VADJ_E_PGOOD [13] = VADJ_F_PGOOD [14] = VADJ_G_PGOOD [15] = VADJ_H_PGOOD
0x008A	PORT_A_I2C_ADDRESS	1	R	I2C Address associated with Port A
0x008B	PORT_A_5V0_GROUP	1	R	5V0 group associated with Port A
0x008C	PORT_A_3V3_GROUP	1	R	3V3 group associated with Port A
0x008D	PORT_A_VIO_GROUP	1	R	VIO group associated with Port A
0x008E	PORT_A_TYPE	1	R	Port A Port Type
0x008F	PORT_A_STATUS	1	R	Bitfield indicating the status of Port A. [0] = Present (1 if present) [1] = DW (1 if doublewide) [2] = 5V0_CURRENT_IN_LIMIT [3] = 3V3_CURRENT_IN_LIMIT [4] = VIO_CURRENT_IN_LIMIT [5] = RSV1 [6] = RSV2 [7] = ALLOW_VIO_ENABLE
0x0090	PORT_B_I2C_ADDRESS	1	R	I2C Address associated with Port B
0x0091	PORT_B_5V0_GROUP	1	R	5V0 group associated with Port B
0x0092	PORT_B_3V3_GROUP	1	R	3V3 group associated with Port B
0x0093	PORT_B_VIO_GROUP	1	R	VIO group associated with Port B
0x0094	PORT_B_TYPE	1	R	Port B Port Type
0x0095	PORT_B_STATUS	1	R	Bitfield indicating the status of Port B. [0] = Present (1 if present) [1] = DW (1 if doublewide) [2] = 5V0_CURRENT_IN_LIMIT [3] = 3V3_CURRENT_IN_LIMIT [4] = VIO_CURRENT_IN_LIMIT [5] = RSV1 [6] = RSV2 [7] = ALLOW_VIO_ENABLE
0x0096	PORT_C_I2C_ADDRESS	1	R	I2C Address associated with Port C
0x0097	PORT_C_5V0_GROUP	1	R	5V0 group associated with Port C

0x0098	PORT_C_3V3_GROUP	1	R	3V3 group associated with Port C
0x0099	PORT_C_VIO_GROUP	1	R	VIO group associated with Port C
0x009A	PORT_C_TYPE	1	R	Port C Port Type
0x009B	PORT_C_STATUS	1	R	Bitfield indicating the status of Port C. [0] = Present (1 if present) [1] = DW (1 if doublewide) [2] = 5V0_CURRENT_IN_LIMIT [3] = 3V3_CURRENT_IN_LIMIT [4] = VIO_CURRENT_IN_LIMIT [5] = RSV1 [6] = RSV2 [7] = ALLOW_VIO_ENABLE
0x009C	PORT_D_I2C_ADDRESS	1	R	I2C Address associated with Port D
0x009D	PORT_D_5V0_GROUP	1	R	5V0 group associated with Port D
0x009E	PORT_D_3V3_GROUP	1	R	3V3 group associated with Port D
0x009F	PORT_D_VIO_GROUP	1	R	VIO group associated with Port D
0x00A0	PORT_D_TYPE	1	R	Port D Port Type
0x00A1	PORT_D_STATUS	1	R	Bitfield indicating the status of Port D. [0] = Present (1 if present) [1] = DW (1 if doublewide) [2] = 5V0_CURRENT_IN_LIMIT [3] = 3V3_CURRENT_IN_LIMIT [4] = VIO_CURRENT_IN_LIMIT [5] = RSV1 [6] = RSV2 [7] = ALLOW_VIO_ENABLE
0x00A2	PORT_E_I2C_ADDRESS	1	R	I2C Address associated with Port E
0x00A3	PORT_E_5V0_GROUP	1	R	5V0 group associated with Port E
0x00A4	PORT_E_3V3_GROUP	1	R	3V3 group associated with Port E
0x00A5	PORT_E_VIO_GROUP	1	R	VIO group associated with Port E
0x00A6	PORT_E_TYPE	1	R	Port E Port Type
0x00A7	PORT_E_STATUS	1	R	Bitfield indicating the status of Port E. [0] = Present (1 if present) [1] = DW (1 if doublewide) [2] = 5V0_CURRENT_IN_LIMIT [3] = 3V3_CURRENT_IN_LIMIT [4] = VIO_CURRENT_IN_LIMIT [5] = RSV1 [6] = RSV2 [7] = ALLOW_VIO_ENABLE
0x00A8	PORT_F_I2C_ADDRESS	1	R	I2C Address associated with Port F
0x00A9	PORT_F_5V0_GROUP	1	R	5V0 group associated with Port F
0x00AA	PORT_F_3V3_GROUP	1	R	3V3 group associated with Port F
0x00AB	PORT_F_VIO_GROUP	1	R	VIO group associated with Port F
0x00AC	PORT_F_TYPE	1	R	Port F Port Type
0x00AD	PORT_F_STATUS	1	R	Bitfield indicating the status of Port F. [0] = Present (1 if present) [1] = DW (1 if doublewide) [2] = 5V0_CURRENT_IN_LIMIT [3] = 3V3_CURRENT_IN_LIMIT [4] = VIO_CURRENT_IN_LIMIT [5] = RSV1 [6] = RSV2 [7] = ALLOW_VIO_ENABLE
0x00AE	PORT_G_I2C_ADDRESS	1	R	I2C Address associated with Port G
0x00AF	PORT_G_5V0_GROUP	1	R	5V0 group associated with Port G
0x00B0	PORT_G_3V3_GROUP	1	R	3V3 group associated with Port G
0x00B1	PORT_G_VIO_GROUP	1	R	VIO group associated with Port G
0x00B2	PORT_G_TYPE	1	R	Port G Port Type
0x00B3	PORT_G_STATUS	1	R	Bitfield indicating the status of Port G.

				[0] = Present (1 if present) [1] = DW (1 if doublewide) [2] = 5V0_CURRENT_IN_LIMIT [3] = 3V3_CURRENT_IN_LIMIT [4] = VIO_CURRENT_IN_LIMIT [5] = RSV1 [6] = RSV2 [7] = ALLOW_VIO_ENABLE
0x00B4	PORT_H_I2C_ADDRESS	1	R	I2C Address associated with Port H
0x00B5	PORT_H_5V0_GROUP	1	R	5V0 group associated with Port H
0x00B6	PORT_H_3V3_GROUP	1	R	3V3 group associated with Port H
0x00B7	PORT_H_VIO_GROUP	1	R	VIO group associated with Port H
0x00B8	PORT_H_TYPE	1	R	Port H Port Type
0x00B9	PORT_H_STATUS	1	R	Bitfield indicating the status of Port H. [0] = Present (1 if present) [1] = DW (1 if doublewide) [2] = 5V0_CURRENT_IN_LIMIT [3] = 3V3_CURRENT_IN_LIMIT [4] = VIO_CURRENT_IN_LIMIT [5] = RSV1 [6] = RSV2 [7] = ALLOW_VIO_ENABLE

8 Optional Features

The Platform MCU can set the DDRVCC voltage based on the logic level established by a jumper or a switch. Additionally, it can delay the SOC/FPGA configuration by holding the INIT_B pin low.

If the board variant supports the PMCU's control of DDRVCCSEL, then the PMU will read the state of the DDRVCCSEL pin during initialization and configure the appropriate channel of the PMU to output 1.5V or 1.35V based on the state of the pin. A logic '1' will result in DDRVCC being set to 1.5V, whereas a logic '0' results in it being set to 1.35V.

If the board variant supports the PMCU's control of INIT_B and DDRVCCSEL, then the PMCU will conditionally configure the pin attached to INIT_B as an output and drive it low during the early portion of device initialization to prevent the FPGA/SOC from being configured prior to DDRVCC being set to the voltage specified by a jumper/switch. If INIT_B is driven low during early initialization, then it will remain low until after all the PMCU submodules have initialized and that the 100-millisecond delay required by the SYZYGY specification has elapsed. Afterwards, the pin connected to INIT_B will be reconfigured as an input and the SOC/FPGA will be allowed to initiate its configuration process. The MCU pin connected to INIT_B may only be configured as output and driven low when the MCU's reset source is Power-On-Reset or Brown-Out-Reset. If an external or software reset occurs, then it is assumed that DDRVCC has already been configured and that there is no reason to delay SOC/FPGA configuration. In fact, driving INIT_B low under these circumstances would clear the SOC/FPGA configuration and is therefore undesirable.

9 Board Documentation Tables

In order to present which features of the Eclypse Platform MCU are available for a board variant utilizing the PMCU, standard documentation for that board is to include the tables below:

Table #: Platform MCU Connectivity Map

Interface	Connection
TEMPERATURE_1	Zynq Die Temperature
PORT_A, VADJ_A	Zmod A
PORT_B, VADJ_B	Zmod B
FAN_1	FPGA Fan
FAN_2	Case Fan

Table #: Supported Platform MCU Optional Features

Optional Feature	Supported
DDRVCCSEL Control	No
INIT_B Control	No
USB Hub Support	No

Table #: Supported Platform MCU Fan Control Features

Fan Header	FAN_1 (FPGA Fan)	FAN_2 (Case Fan)
Enable / Disable	Yes	No
Fixed Speed Control	Yes	No
Automatic Speed Control	Yes	No
RPM Measurement	Yes, if supported by installed fan[1]	Yes, if supported by installed fan[2]

[1] The optional FPGA fan included with the Eclypse Z7 supports RPM measurement
[2] The case fan included in the Eclypse Z7 Enclosure Kit does not support RPM measurement