Jason Leake
16/8/95

Phase I Demonstrator BCID Card Programming Model

General considerations

The most important consideration is to try to strike a balance between making the card easy to drive from the computer and not over-complicating its design. Two guidelines have been adopted to try to achieve this.

1. All registers can be read by the computer—there are no "write-only" registers. Registers and fields within registers generally have the same meaning for reads as for writes.

   Reading back a register which is writeable will generally return the last value written, except in the case of memory access registers with auto-incrementing address counters. In these cases, the next memory address will be accessed. In the *Control and Status Register,* if a bit is modified by the computer to change the status of something it can be read back to determine what the selected status now is.

   Attempts to write to non-writeable registers or portions of registers will result in the non-modifiable fields being left unchanged. Some fields such as the *Generate Single Tick* bit in the *Control and Status Register* can be used to repeatedly trigger an operation by repeatedly writing a 1 to them. In this case, they always read back as 0.
2. It is illegal for the computer to read or write a value which the card itself is able to modify at that same instant. This means that if the *Card Is Halted* bit in the *Control and Status Register* is 0 so that the free-running External 40 MHz Clock is being accepted, the computer cannot legally access most of the registers on the card.

   The effect of doing so is generally unpredictable (to simplify the design of the card), but there is a bit in the *Control and Status Register* and an associated LED on the front panel to indicate when this (or some other problem) has been detected. This does not trigger any other action by the card, but is intended to be merely an indication of a possible programming error.

Notation

The names of registers and bit fields within them are written in *italic*. The names of signal lines are <u>*underlined italic*</u>. Bit fields are labelled <u>Input to card</u> and <u>Output from card</u> rather than <u>Write</u> and <u>Read</u> to make it clear whether it is the card or the computer which is doing the writing.

In this document, a <u>byte</u> is always an 8 bit data field, and a <u>word</u> is always 16 bits.

Setting a bit field means writing a 1 to it, clearing it means writing a 0.

There are two signal lines which are concerned with disabling the card's responding to the External 40 MHz Clock signal. One of these, <u>*Stop*</u>, is a backplane line pulsed by the external Clock Generator Module. This is done on receipt of a pulse on the other signal line, the

Countdown Complete which is connected to the Clock Generator Module from the front panel of the BCID card.

Register Layout

The principal limitation with the Phase I Demonstrator system from the computer's perspective is the small addressing range of only 128 addresses per card imposed by the UA1 backplane architecture. The BCID card uses the first eight 16-bit registers in the address space, named as follows:

Offset from base of card's address space	**Name**	**Contains**
7	*FPGA RAM Counter Register*	counter
6	*FPGA RAM Access Register*	data
5	*FPGA Control Register*	bit mask
4	*Memory Access Register*	data
3	*Clock Countdown Register*	counter
2	*Counter Register*	memory address
1	*Control and Status Register*	bit mask and address
0	*Module Identification Register*	bit mask

Module Identification Register

This word contains a hardwired bit pattern which represents the type and serial number of the card. It follows Eric Eisenhandler's proposed scheme:

Bits 0-7	Serial number of card—The first card built has a serial number of 1
Bits 8-11	Revision number—The first version of the card is revision 1
Bits 12-15	Type number of card—the BCID card is type number 1

Control and Status Register (CSR)

This register is used by the computer to control the overall operation of the card and monitor its state. It is arranged as follows, with the initial power-up state being 1000_{16}.

Bit			
0 (LSB)	*Channel Number*	Input to card	
1	*Channel Number*	Input to card	
2	*Channel Number*	Input to card	
3	*Clear Stop*	Input to card	Always 0 on read
4	*Memory Type*	Input to card	
5	*Memory Type*	Input to card	
6	*Auto-increment Counter Enable*	Input to card	
7	*Clear Protocol Error*	Input to card	Always 0 on read
8	*Generate Single Tick*	Input to card	Always 0 on read
9	*External Clock Enable*	Input to card	
10	*Clock Countdown Enable*	Input to card	
11	*Test Mode Enable*	Input to card	

12	*Card is Halted*	Output from card	
13	*Unused*	-	Always 0 on read
14	*Stop*	Output from card	
15	*Protocol Error*	Output from card	

Channel Number   bits 0-2   Input to card

This specifies which channels on the card any access to the *Memory Access Register* should refer to. The *Memory Access Register* is a 16-bit register which reads and writes pairs of bytes from adjacent channels.

210	*Memory Access Register*
000	Channels 0 and 1
001	Channels 2 and 3
010	Channels 4 and 5
011	Channels 6 and 7
100	Channels 8 and 9
101	Channels 10 and 11
110	Channels 12 and 13
111	Channels 14 and 15

Bit 0 specifies which Xilinx FPGA channels to select. There are four channels per FPGA, but as the *Memory Access Register* references pairs of them, only one bit is needed to select either the upper or lower pair of channels. Bits 1 & 2 select one of up to four FPGAs.

210	*Memory Access Register*
000	Channels 0 & 1 = FPGA 0, channel 0 & 1
001	Channels 2 & 3 = FPGA 0, channel 2 & 3
010	Channels 4 & 5 = FPGA 1, channel 0 & 1
011	Channels 6 & 7 = FPGA 1, channel 2 & 3
100	Channels 8 & 9 = FPGA 2, channel 0 & 1
101	Channels 10 & 11 = FPGA 2, channel 2 & 3
110	Channels 12 & 13 = FPGA 3, channel 0 & 1
111	Channels 14 & 15 = FPGA 3, channel 2 & 3

If there are less than four FPGAs on a card, any attempt to access a non-existent channel is illegal and causes the *Protocol Error* bit to be set.

Note: The notation here does not match that on the schematic diagrams of the card. The card refers to the lowest numbered channel as Channel 1 (e.g. there are lines labelled *DAT1* etc.), whereas here the lowest numbered channel is Channel 0 (which corresponds to the binary values of the bit fields).

Clear Stop   bit 3   Input to card

The *Stop* bit is cleared by writing a 1 to *Clear Stop*. *Clear Stop* is always read 0, and writing 0 has no effect. The bit field used to clear *Stop* is separate from the *Stop* bit itself so that the computer cannot accidentally clear it between reading the *Control and Status Register* and writing it back with some fields modified. As this field is always 0 on read, the computer has to explicitly set it before writing if it is to clear *Stop*.

0	Read value, no effect on writing
1	Clear the *Stop* bit

Memory Type bits 4-5 Input to card

This specifies which of the three types of memory attached to each channel is to be accessed via the *Memory Access Register*.

<u>54</u>	
00	Memory undefined
01	Input memory
10	Output memory
11	LUT/Test memory

If the *Memory Access Register* is referenced when the *Memory Type* field is 0, then the *Protocol Error* bit will be set and the result undefined.

Auto-increment Counter Enable bit 6 Input to card

If set, this causes the *Counter Register* to be incremented by 1 immediately before each read or write to the *Memory Access Register*. The *Counter Register* is also incremented on each External 40 MHz Clock or internal single clock tick generator pulse, regardless of the setting of the *Auto-increment Counter Enable*.

0	Disable auto-increment of *Counter Register*
1	Enable auto-increment of *Counter Register*

Note that this bit only affects the automatic incrementing of the *Counter Register*—the *FPGA RAM Counter Register* is always incremented automatically regardless of the setting of this bit.

Clear Protocol Error bit 7 Input to card

The *Protocol Error* field is reset by writing a 1 to *Clear Protocol Error*. This bit is always read 0, and writing 0 to it has no effect.

0	Read value, no effect on writing
1	Clear *Protocol Error* bit

Generate Single Tick bit 8 Input to card

Each time that a 1 is written by the computer, the internal single tick clock generator emits a single clock pulse. This is only permissible when the *Card Is Halted* bit is set. It is illegal to set it when the *Card Is Halted* bit is 0; if this is done then the result is unpredictable and the *Protocol Error* bit is set. It is always 0 on read, writing 0 into it has no effect.

0	Read value, no effect on writing
1	Pulse internal Single Tick Clock Generator

External Clock Enable bit 9 Input to card

This bit enables the External 40 MHz Clock to be accepted into the card. When it is cleared, the *Card Is Halted* bit is set automatically at the same time. In this state, the single tick generator can be used to clock the card.

Note that the power-up state of the *Control and Status Register* means that the *External Clock Enable* is initially 0 to allow the Xilinx FPGAs and the look-up tables to be loaded before the card is set running. The <u>HALT</u> line is gated by the external clock strobe, so it is not possible to switch the card from a halted state to a running state unless the external clock is actually present on the backplane.

0	Halt card, but allow use of internal single tick clock generator
1	Use External 40 MHz Clock

Clock Countdown Enable bit 10 Input to card

This bit is used in conjunction with the *Clock Countdown Register* to freeze the entire system automatically after a specified number of clock ticks.

The mechanism by which the system is stopped is for the BCID card to pulse its front panel Countdown Complete output signal line. This is connected to the external clock generator module which uses this signal to trigger a pulse on the backplane *Stop* line which is distributed to all cards. The effect of this backplane signal is to freeze all of the cards in the system.

Whenever the BCID card itself detects a pulse on the backplane *Stop* line, it will halt by disabling the acceptance of the External 40 MHz Clock into the card logic and setting the *Stop* and *Card is Halted* bits.

If *Clock Countdown Enable* is 1 and the *Clock Countdown Register* is 0000_{16}, then a pulse is generated on the *Countdown Complete* output on the front of the card. This should result in the card receiving a backplane *Stop* signal via the mechanism outlined above. Note that only the least significant 8-bits of the *Clock Countdown Register* are used—the upper 8 bits are always zero.

If a *Stop* pulse is not received, which will happen if, for example, the *Countdown Complete* output is not connected to the clock generator, then the *Clock Countdown Register* will continue to be decremented on each further clock pule.

If *Clock Countdown Enable* is 0, then the *Clock Countdown Register* is not decremented.

Once the card has been halted, the external clock can be enabled again by:

1. Either writing a non-zero value to the *Clock Countdown Register* (which is legal at this point because *Card is Halted* =1), or by clearing *Clock Countdown Enable.*

   AND THEN

2. Writing 1 into the *Clear Stop* bit to cancel the effect of the backplane *Stop* signal on the card.

The *Stop* bit must be cleared after the clock countdown fields have been reset, otherwise the the card will generate another *Countdown Complete* pulse on the next External 40 MHz *Clock* or internally generated clock pulse if the clock countdown registers are not reset in time. The operation of the clock countdown mechanism is as follows:

for each External 40 MHz *Clock* or internally generated clock pulse
    if *Clock Countdown Enable* = 1
        if *Clock Countdown Register* = 0000
            Pulse *Countdown Complete* line
        end if
        Clock Countdown Register = *Clock Countdown Register* - 1
    end if
end for

Although this function would perhaps be more logically located in the clock generator module, it is more convenient to incorporate it into the new BCID card than modify the existing clock generator card.

Test Mode Enable bit 11 Input to card

If set, this causes the LUT/Test memory to be interpreted as test data and clocked into the Xilinx FPGA on each clock tick. If clear, then the LUT/Test memories are interpreted as 256-element look-up tables indexed by the incoming data bytes. This mechanism is thus used to implement pedestal subtraction.

0	Card is in Beamline mode
1	Card is in FPGA-testing mode

A LED on the front panel displays the setting of this bit.

Card Is Halted bit 12 Output from card

When this bit is set, the External 40 MHz Clock is blocked so that it does not drive the card. The local single tick generator may still be used to drive the card, though.

The card sets this bit when either (or both) of the following conditions is detected:

1. The *External Clock Enable* bit is 0.
2. The *Stop* bit is 1.

The HALT line is gated by the external clock strobe, so it is not possible to switch the card from a halted state (i.e. clear this bit) to a running state unless the external clock is actually present on the backplane as a clock edge is required before HALT changes state.

If *Card Is Halted* is 0, then only the *Control and Status Register* and the *Module Identification Register* may be read or written by the computer. The effect of trying to do this to other registers is undefined except that the *Protocol Error* bit in the *Control and Status Register* is set.

0	Card is accepting External 40 MHz Clock
1	Card is not accepting External 40 MHz Clock

A LED on the front panel displays the setting of this bit.

Stop bit 14 Output from card

This bit is always set by the card when it detects a pulse on the *Stop* backplane line. The *Stop* signal is generated by the (external) clock module to halt the system, and distributed to all cards along the backplane. The effect of receiving the *Stop* signal is for the card to ignore further External 40 MHz Clock pulses and set the *Card is Halted* and *Stop* bits.

The *Stop* bit remains set until the computer writes clears it by writing a 1 to *Clear Stop*.

0	No Backplane Stop pulse since power-up or *Stop* bit last cleared
1	Backplane Stop pulse has been detected

A LED on the front panel displays the setting of this bit.

Protocol Error bit 15 Output from card

This bit is set by the card when it detects that the computer has tried to perform an illegal operation. The normal result of the operation is undefined. The *Protocol Error* bit remains set until the computer clears it by writing a 1 into *Clear Protocol Error*.

A LED on the front panel displays the setting of this bit.

0	No error since power-up or bit cleared
1	Error since last reset or bit cleared

Bit 15 is used because it is easy to check:

```
if (*bcid_card_csr_address < 0) phone_norman();
```

Counter Register

This is an 8-bit register extended to 16 bits, so that the most significant 8 bits are always zero on read and cannot be changed.

The register is used to select memory addresses both when the card is being clocked and when the computer is accessing the memory. Only the least significant byte is used as this represents the 256 byte address range for the memories.

The computer can only access it when *Card Is Halted* is set. If the *Auto-increment Counter Enable* bit in the *Control and Status Register* is set, the least significant byte of the *Counter Register* is incremented **immediately before each access** to the *Memory Access Register*. It rolls through 00_{16} on the access immediately following FF_{16}.

When the card is in being clocked by either the External 40 MHz Clock or the internal tick generator, then this register is used in determining the next address to use in the LUT/test memory (when the card is in FPGA-testing Mode), the input memory and the output memory. Each clock pulse (either from the External 40 MHz Clock or from the single tick generator) causes the counter incremented before the memories are accessed regardless of the setting of *Auto-increment Counter Enable.* This means that when the *Auto-increment Counter Enable* bit is set, the value of this register is always the address **before** the one which any memory access will reference.

Note that the programmer will have to be a little careful when interleaving memory accesses with generating single ticks or allowing the clock to run as both operations will change the memory.

Clock Countdown Register

This register is used to indirectly halt the card after a specified number of external or internally generated clock pulses, in conjunction with the *Clock Countdown Enable* bit in the *Control and Status Register*. Only the least significant byte is used as there is little point in completely filling the memory with input and output data more than once, and there are 256 memory locations which corresponds to a one byte address.

If *Clock Countdown Enable* is 1 the *Clock Countdown Register* is decremented on receipt of each External 40 MHz Clock pulse. When it reaches 0000_{16} the card pulses the *Countdown Complete* signal line on the front panel. If this does not result in the card receiving a *Stop* signal on the backplane, then further pulses will result in the register being further decremented.

For example, if this register contains 6 and *Clock Countdown Enable* is set, then after exactly six external or internally generated *Clock* pulses, the *Countdown Complete* signal line will pulsed.

If the *Clock Countdown Enable* is zero, then the *Clock Countdown Register* remains unchanged.

See the description of *Clock Countdown Enable* for more information.

Memory Access Register

This register acts as a window into two bytes of memory. Each byte is from the same location and memory type of one of a pair of channels. The location is specified by the current value of the *Counter Register*, and the type of memory by the *Memory Type* field in the *CSR*. The channels accessed are specified by the *Channel Number* field in the *CSR*. Bits 8-15 are used to access the lower channel number of the pair of channels, and bits 0-7 the higher channel number.

Reading this register returns one byte each from the relevant two memories. Writing to it loads a byte into each of the memories.

The pins on the FPGA which drive the output data lines of memories are tied to a particular value by the FPGA until it has been configured. This means that the output memories will not work properly (bits stuck) until the relevant FPGA has been loaded.

FPGA Control Register

This is a bit mask which is used to control the Xilinx FPGAs. Unused bits are read 0, write no change. The initial power-up or reset state **is undefined**. Thus, it is a good idea to write is 0000_{16} to it before attempting to configure the FPGAs to ensure that everything is in a known state. It is laid out as follows:

Bit			
0 (LSB)	*Peak Finder Override*	Input to card	
1	*FIR Filter Enable*	Input to card	
2	*Unused*	-	Always 0 on read
3	*Unused*	-	Always 0 on read
4	*FPGA Number*	Input to card	
5	*FPGA Number*	Input to card	
6	*Unused*	-	Always 0 on read
7	*Unused*	-	Always 0 on read
8	*Clear Program*	Input to card	
9	*Readback*	Input to card	
10	*RAM Clock Trigger*	Input to card	Always 0 on read
11	*RAM Clock Status*	Output from card	
12	*Not Initialised*	Output from card	
13	*All Loaded*	Output from card	
14	*Readback In Progress*	Output from card	
15	*Loaded*	Output from card	

As with all registers except the *Control and Status Register* and the *Module Identification Register*, this register cannot be accessed if the card is not halted. The effect of doing otherwise is unpredictable except that the *Protocol Error* bit in the *Control and Status Register* will be set.

Peak Finder Override bit 0 Input to card

Setting this bit asserts the IOPFOVR line on all Xilinx FPGAs on the card. This causes data to be written to the output stream continuously instead of all zeros except for a single byte representing the peak.

FIR Filter Enable bit 1 Input to card

Setting this bit asserts the IOFIRPIPE line on all Xilinx FPGAs on the card. This causes the output data stream to be taken from the Finite Input Response filter instead of from the pipeline.

Usual settings for these bits are as follows:

Peak Finder Override	FIR Filter Enable	Operation
0	1	Normal beamline operation—bunch crossing identification enabled—output on peak only
1	0	Straight-through mode—FPGA inputs copied to outputs unchanged. Used for testing and debugging.

FPGA Number bits 4-5 Input to card

These bits select the Xilinx FPGA which this and subsequent operations on the *FPGA Control Register* are to refer to, including any other fields in the *FPGA Control Register* which are modified in the same write operation as selects the FPGA number. If there are less than four Xilinx FPGAs on the card, then selecting an illegal FPGA will cause the *Protocol Error* bit to be set.

The settings for these bits are as follows:

45	*Selected FPGA*
00	Number 0
01	Number 1
10	Number 2
11	Number 3

If there are less than four FPGAs on the card, (in practice there are only 3), then the highest numbered FPGAs are missing. Thus, on the real card, it is FPGA number 3 which is missing.

Clear Program bit 8 Input to card

This bit is connected to the */PROGRAM* signal on the currently selected FPGA. When *Clear Program* is zero, the currently selected FPGA will go into a continuous loop clearing its configuration memory. This causes the FPGA's <u>*DONE*</u> signal to be pulled down and thus the *Loaded* bit cleared.

When *Clear Program* is subsequently set, the FPGA finishes its current clear cycle and enters a state where it can start loading its configuration memory when the RAM clock is started. This process takes about 5 microseconds, so there must be a delay of at least this period between the *Clear Program* bit being set and the *RAM Clock Trigger* bit being set.

The completion of the configuration loading process may be detected by polling the *Loaded* bit. An error in the configuration program is detected by monitoring the *Not Initialised* bit.

The contents of the FPGA RAM are transferred from the address pointed to by one more than the value of the *FPGA RAM Counter Register*, which is incremented **before** a word is transferred from the FPGA configuration RAM into the FPGA. The value of the *FPGA RAM Counter Register* when the transfer is complete is therefore the address of the last memory location which was copied into the FPGA.

0	Clear FPGA configuration program
1	Commence load of FPGA from RAM once RAM clock started

Readback bit 9 Input to card

This bit is used to trigger or abort the transfer of data from the currently selected FPGA to the FPGA RAM.

To start the transfer, the computer sets the bit. The *Readback In Progress* bit is set by the card about 1 microsecond afterwards. However, as the computer takes nearly four times longer than this to read a register, it will see this happening immediately.

The contents of the FPGA are transferred into the FPGA RAM starting at the address following that contained in the *FPGA RAM Counter*. This is because the counter is incremented automatically before a word in the Xilinx FPGA configuration memory is transferred into the RAM. The value of the *FPGA RAM Counter* on completion of the transfer is therefore one greater than the last memory address which was copied into the FPGA.

The computer then polls *Readback In Progress* until it is cleared by the card to indicate that the readback is complete.

The transfer may be prematurely aborted by clearing *Readback*. The card uses transitions of this bit to trigger and abort the FPGA readback, so immediately setting the bit will cause readback to start again from the beginning of the configuration memory (but at the current RAM address selected by the *FPGA RAM Counter*).

0	Abort readback from FPGA to RAM
1	Commence readback from FPGA to RAM

At present, the Readback function is not implemented in any of the BCID cards and there are no plans to do so (although the empty chip sockets and wiring is installed on the board for it).

RAM Clock Trigger bit 10 Input to card

This bit is used to enable the FPGA RAM to be clocked by an on-board clock, either when an FPGA is being loaded with its configuration from the RAM or when it is being read back into the RAM. The computer normally only starts the clock as it stops automatically once the operation is complete, as detailed below.

The reason for having this bit is that there must be a 5 microsecond delay between the *Readback* bit being set and the clock being started and using the computer to provide this was considered to be the most convenient method.

The clock is stopped when any of the following circumstances occur:

1. During loading, when the *Loaded* bit is set (i.e. <u>*DONE*</u> goes high).
2. During loading, when the *Clear Program* bit is set by the computer. This is necessary in case there is an error detected during the loading of the program; in that case <u>*DONE*</u> will not be asserted and so the clock will not be automatically stopped by the presence of that signal.
3. During readback, when *Readback In Progress* is cleared (i.e. <u>*RDBK.RIP*</u> goes low).

The computer can only enable the clock with this bit, by writing a 1 to it. If a 0 is written, then there is no change in the clock status. It is always 0 on reading.

0	No action, value obtained when bit read
1	Enable FPGA configuration/readback clock

RAM Clock Status bit 11 Output from card

This bit is set by the card whenever the FPGA configuration/readback clock is running.

0	Clock is not running
1	Clock is running

Not Initialised bit 12 Output from card

This bit is connected to the <u>*/INIT*</u> pin on the FPGA. It is held 0 whilst the currently selected FPGA is undergoing power stabilisation and initial clearing of memory. When the program is being downloaded into it it is held high. If an error is detected in the program during loading, then it is pulled low.

Thus, the computer should monitor this field during FPGA configuration until *Loaded* becomes set. If this field becomes zero when *Loaded* is zero then a configuration error has occurred.

After *Loaded* becomes set, the value of this bit is undefined. This is because the signal line on the FPGA used for it becomes reallocated as a user-defined I/O line.

If *Loaded* is clear:		
	0	FPGA undergoing power-up initialisation or error has been detected
	1	If program being loaded, then no error so far
If *Loaded* is set:		
	0 or 1	No interpretation possible. <u>*/INIT*</u> is reallocated as an I/O pin (and thus this bit shows the value of the pin).

All Loaded bit 13 Output from card

This field is set when all FPGAs on the card have been successfully loaded. It is the AND'ing of all <u>*DONE*</u> pins (i.e. *Loaded* bits) of FPGA's present on the board.

The setting of this bit is displayed as an LED on the front panel.

0	One or more FPGAs remain to be loaded
1	All FPGAs have been loaded with their configuration programs

Readback In Progress bit 14 Output from card

This bit is set by the computer to indicate that an FPGA is downloading its data into the FPGA RAM. As with all output fields, writing into it has no effect. The bit is actually driven from the <u>*RDBK.RIP*</u> signal from the FPGA. The pin assigned to this signal is set high by the FPGA before configuration has taken place. Therefore, it is always 1 before the FPGA has been configured (at which time it is not possible to read the configuration back).

0	No transfer of configuration from FPGA to FPGA RAM in progress
1	Data being transferred from FPGA to FPGA RAM (if the *Loaded* bit is set) or normal value if the FPGA has not been configured yet (i.e. if the *Loaded* bit is clear). the *Loaded* bit is set) has not been configured yet).

Loaded bit 15 Output from card

This bit is set when the currently selected FPGA contains a valid configuration program. It is driven by the currently selected FPGA's <u>*DONE*</u> signal.

0	Currently selected FPGA is not loaded
1	FPGA is loaded with a valid configuration program

FPGA RAM Access Register

Access to the Xilinx FPGA configuration memories is not straightforward because they are not addressable. It is only possible to read or write them sequentially at a constant(-ish) clock rate. This is done by having a single RAM on the card from which any FPGA can be loaded from or into which it can copy the contents of its configuration memory ("readback") under the control of the *FPGA Control Register*. The RAM is clocked using a 1 MHz oscillator on the card.

This FPGA RAM is then accessed by the computer using the *FPGA RAM Access Register*. The *FPGA RAM Access Register* is used to read and write 16-bit words to FPGA RAM at the memory locations specified using the *FPGA RAM Counter Register*.

FPGA RAM Counter Register

This register is used to address the word in the FPGA RAM which is to be accessed by an FPGA or the computer. On each access to the *FPGA RAM Access Register* by either, the *FPGA RAM Counter Register* is incremented **before the read/write operation takes place**.

Thus, it is the word following the one currently addressed by the counter that is actually accessed. The register is 16-bits wide, although only the least significant 14 bits are used to address the FPGA RAM (as the RAM has a capacity of 16384 words).

It can therefore be used to attempt to access non-existent memory, although the effect of doing this is not defined other than the *Protocol Error* bit becoming set. Successive memory accesses (or attempted accesses) cause it to increment over the whole 16-bit range, returning to zero only after passing through $FFFF_{16}$.

When data is being transferred to or from an FPGA from the FPGA RAM, the counter is clocked by an on-board oscillator.

To load an FPGA configuration program into an FPGA, the following steps are required:

1. Check that the *Protocol Error* bit is clear and if necessary clear it.
2. The *FPGA Control Register* is read to ensure that *Readback In Progress* AND *Loaded* are not both set. This is to ensure that the current FPGA is not accessing the RAM.
3. The other FPGAs can be checked by selecting them using the *FPGA Number* field in the *FPGA Control Register*. However, this should not be strictly necessary as there is the rule that it is illegal to change the selected FPGA when downloading or readback is in progress. This rule is partially enforced by the *Protocol Error* bit which can at least be used to check for violations of it.
4. The *FPGA RAM Counter Register* is loaded with $FFFF_{16}$, so that the next memory access will cause the counter to roll over to 0000_{16} before the RAM is written.
5. Successive words of the configuration program are written by the computer to the *FPGA RAM Access Register*. The size of the FPGA configuration memory is 8262 words, which is less than the capacity of the FPGA RAM. Each memory write will cause the *FPGA RAM Counter Register* to be incremented automatically.
6. The *FPGA RAM Counter Register* is loaded with $FFFF_{16}$ again.
7. The *FPGA Control Register* is read and then written to select the desired FPGA.
8. It is then rewritten with the *Clear Program* bit set to zero.
9. It is then rewritten with the *Clear Program* bit set.
10. The program waits at least 5 microseconds. In practice, a "C" sleep(1) call is a good way of doing this even if it represents a certain amount of overkill!
11. The *FPGA Control Register* is now written to set the *RAM Clock Trigger* bit.
12. The program loops monitoring the *Loaded* and *Not Initialised* lines. If *Loaded* is set, then the program has been loaded.
13. If *Not Initialised* becomes zero whilst *Loaded* is still zero, then a checksum error has been detected in the program being loaded. The configuration/readback clock must be stopped and any program in the FPGA cleared out by setting *Clear Program.* Then go to step 6.
14. *Loaded* should become set to indicate that the configuration program has been successfully loaded. To load further FPGA*s* with the same program, go to step 6.
15. Check that the *Protocol Error* bit is still clear. If the above steps have been correctly followed, then this should be the case.

To readback an FPGA configuration program, the following steps are required:

1. Check that the *Protocol Error* bit is clear and if necessary clear it.

2. The *FPGA Control Register* is read to ensure that the *Readback In Progress* bit is clear, and also that *Loaded* and *Clear Program* are not simultaneously clear. This is to ensure that the FPGA is not accessing the RAM.
3. The *FPGA RAM Access Register* is set to $FFFF_{16}$.
4. The *FPGA Control Register* is read and then written to select the FPGA.
5. There must be at least a 5 microsecond delay between the *Readback* being set and the configuration/readback clock started. The easiest way of guaranteeing this is to call the "C" sleep(1) procedure to wait for a second, but it is also possible for the impatient to read the *Module ID Register* twice as each UA1 bus access takes 4 microseconds and thus a delay of 8 microseconds is assured.
6. The *FPGA Control Register* is written to set *RAM Clock Trigger* to start the clock.
7. The computer loops, polling *Readback In Progress*. When this becomes set, then the readback into the RAM is complete. Unlike the configuration case, the clock is stopped automatically under all circumstances. This is true even if *Readback* is cleared to abort the readback process.
8. The *FPGA Control Register* is read to determine the number of words loaded from the FPGA. The number of words loaded would normally be expected to be 9222, and is equal to one more than the current value of the register.
9. The *FPGA Control Register* is loaded with $FFFF_{16}$ again.
10. The FPGA configuration is read via successive accesses to the *FPGA RAM Access Register* to read the number of words loaded into it by the FPGA.
11. Check that the *Protocol Error* bit is still clear. If the above steps have been correctly followed, then this should be the case.

Appendix 1: Logic controlling the Protocol Error bit

The *Protocol Error* bit of the *Control and Status Register* is latched to 1 if any of the following actions occur. The effect of the action itself is unpredictable (i.e. we don't have to worry about handling it). It is reset by the computer writing a 1 to *Clear Protocol Error*.

The function of this bit is to provide indication of a programming error and to assist in finding the error (because the program running in the computer can be single stepped until the light comes on). It does not implement an exhaustive program validation process!

1. Any computer access is made to any register apart from the *Control and Status Register* and the *Module Identification Register* when the *Card is Halted* bit is 0. These are address offsets 0 and 1 in the card's address space.

2. The computer attempts to address any location which is not mapped by a register; i.e. any address outside of the range 0-6. This is a simple check because it is only necessary to check that the upper address lines are all zero.

3. If the computer tries to set *Generate Single Tick* when *Card is Halted* = 0.

4. If a read or write is made to either the *FPGA Control Register* or the *FPGA RAM Access*

Register when the the upper two bits of *Channel Number* represent a non-existent Xilinx FPGA.

5. If a read or write is attempted to the *Memory Access Register* when the *Memory Type* field of the *CSR* = 00.

6. When an illegal FPGA is selected using the *FPGA Number* field in the *FPGA Control Register.*

7. If the computer tries to access either the *FPGA RAM Counter Register* or the *FPGA RAM Access Register* when either *Readback In Progress* is set, or both *Loaded* and *Clear Program* are clear.

   Under these circumstances, the FPGA RAM is being accessed by the FPGA. The FPGA RAM may be tested before FPGA are loaded by setting *Clear Program* for the currently selected FPGA.

8. If the computer tries to change the *FPGA Number* field in the *FPGA Control Register* whilst either *Readback In Progress* is set, or both *Loaded* and *Clear Program* are clear.

   If this is not the case, then the currently selected FPGA is accessing the FPGA RAM and so changing the FPGA is likely to cause some confusion.

9. If the computer tries to set *Trigger Readback* when *Loaded* is clear. Copying the configuration program from the FPGA to the RAM at the same time as a transfer in the opposite direction is taking place is likely to cause problems.

10. If the computer tries to set *Clear Program* when *Readback In Progress* is set. Erasing the FPGA configuration at the same time as it is being transferred into the FPGA RAM would be a probable source of confusion.

11. If an attempt is made to access non-existent FPGA RAM memory—i.e. either of the two most significant bits of the *FPGA RAM Counter Register* become set at the same time as the FPGA RAM is selected for read or write by either the computer or the FPGA logic.

Appendix 2: Front panel LEDs

There are seven LEDs on the front panel in a single vertical bank at the top of the card.

Red LED	*Protocol Error* bit of *Control and Status Register* is set. Normally off.
Green LED	All Xilinx FPGAs are programmed. This LED is driven by *All Loaded* bit in the *FPGA Control Register.*
Amber LED	*Test Mode Enable* bit is set.
Green LED	Card is receiving power. Always on during normal card operation.
Amber LED	*Card is Halted* bit is set.
Amber LED	*Stop* bit is set, indicating that the card has seen a *<u>Stop</u>* pulse on the backplane.

Green LED	Card is processing a clock signal. This is lit by the clock distribution logic on the card to indicate that a clock signal is being processed. This is either from the External 40 MHz Clock or from the internal single tick generator. A single clock pulse should light it for about 700 milliseconds so that it can be seen. Normally lit during normal card operation as clock pulses will be arriving at a much greater rate than 1.5 Hz.

Appendix 3: Example routines for driving card

The following routine shows how the computer might check that the card is running.

`BCID_CSR_M_CARD_HALTED` etc. are bit masks for the various *CSR* fields.

```
void make_sure_card_is_running(volatile short int *csr_address)
/* Check if the card is running; if it isn't then start it running.
**
** Parameters:
**       csr_address - Address of CSR register on BCID card
** Side effects:
**       Takes the necessary action to get the card running
**       Program may be aborted via call to phone_norman if an protocol
**       error is reported by the card.
*/
{
    while (*csr_address & BCID_CSR_M_CARD_HALTED) {
        printf("Card is stopped because:\n");

        if (!(*csr_address & BCID_CSR_M_EXTERNAL_CLOCK_ENABLE)) {
            printf("You are suppressing the clock...");
            *csr_address =
                *csr_address | BCID_CSR_M_EXTERNAL_CLOCK_ENABLE;
            printf("Clock acceptance has now been enabled\n");
        }
        if (*csr_address & BCID_CSR_M_STOP) {
            printf("A backplane Stop signal was received....\n");
/* We may have generated it by having the Clock Countdown register
** counting down */
            if (*csr_address & BCID_CSR_M_CLOCK_COUNTDOWN) {
/* It is safe to check Clock Countdown Register because card is halted */
                volatile short int *clock_countdown_address;
                clock_countdown_address = csr_address +
                                     BCID_CLOCK_COUNTDOWN_REGISTER_OFFSET;
                if (*clock_countdown_address == 0) {
                printf("It was triggered by this card\n");
                *csr_address = *csr_address ^ BCID_CSR_M_CLOCK_COUNTDOWN;
                printf("Clock countdown has been disabled now\n");
            }
            *csr_address = *csr_address | BCID_CSR_M_CLEAR_STOP;
            printf("Backplane Stop trigger has been reset\n");
        }
/* Check for protocol error in any of the above */
        if (*csr_address < 0) phone_norman();
    }
```

```
    printf("The card is running");
}
```