WGA

And

RICA

Applicability

To

RF Scan Function

Design

DOCUMENT RESPONSIBILITIES

Role	Name	Institute	Date	Signature
Written by:	K.L.Morgan	RAL	24/05/99	
Written by:				
Written by:				
Approved by:				
Approved by:				
Authorised by:				
Authorised by:				

DOCUMENT DISTRIBUTION

Name	Affiliation	Copies	Name	Affiliation	Copies

DOCUMENT CHANGE RECORD

Issue	Rev.	Sect.	Date	Changes	ECR No.
1	0	All	24/5/1999	First Issue, All New.	N/A

TABLE OF CONTENTS

1. Introduction

1.1 Purpose of the document

The two instruments, PTOLEMY and BERENICE, that constitute the MODULUS experiment aboard the ESA's ROSETTA spacecraft are dependant upon two ASICs to drive the Ion Trap hardware. The purpose of this document is to provide enough information about the working of these ASICs so that the Scan Functions which drive the Ion Trap can be specified and programmed by the Chemistry team.

1.2 Definitions, acronyms, and abbreviations

1.2.1 Definitions

When reference is made to a 16-bit word, the bit numbering convention is as follows:-

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

With bit '0' being the Least Significant Bit and bit '15' being the Most Significant Bit..

1.2.2 Acronyms

ADU	Analogue to Digital Unit
ASIC	Application Specific Integrated Circuit (i.e. a custom chip)
I2C	Inter-Integrated Circuit - a serial bus protocol for transferring data between Integrated Circuits.
RAM	Random Access Memory
RICA	ROSETTA Ion-Counter ASIC
WGA	Waveform Generator ASIC

1.3 References

1.3.1 Applicable Documents

1.3.2 Reference Documents

"The Waveform Generator Chip" by Lawrence Jones, 7th August 1998.

"ROSETTA Ion Counter ASIC (RICA) Project Specification Version: 1.2" by Bill Gannon, 26th May 1998.

"ROSETTA MODULUS Evolved Gas Analyser Instrument Design Requirements", by N.R. Waltham, February 1997.

2. The WGA Sequencer Chip

2.1 Introduction

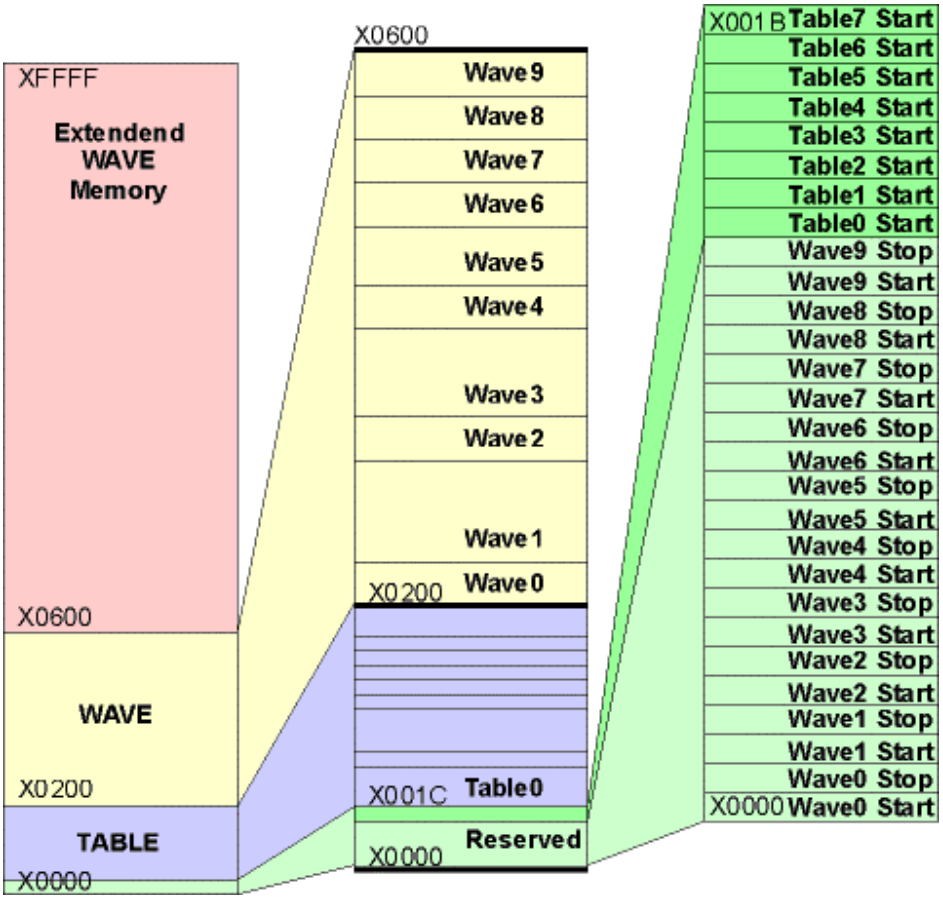
The Waveform Generator Chip (WGC) is intended for use where long, sequential waveforms are required for control or sequencing of a device. One such application is for the driving of the Ion Trap. The WGC produces 16 bit waveforms which can be output at a frequency of up to 40MHz. The waveforms are user programmable using a combination of *WAVES* and *TABLES.*

WAVES are defined as a string of 16 bit vectors which are output in order on consecutive clock cycles at a rate of up to 40MHz. Up to 10 single *WAVES* can be defined in areas of internal or external RAM - or combinations of both. *WAVES* are of definable length within RAM. Internal RAM is 1024 words in size and up to 64000 words of external RAM can be used.

TABLES are defined as a list 16 bit commands which determine the order in which a sequence of single *WAVES* is strung together to form a more complex waveform. Up to 8 *TABLES* can be defined in an area of internal RAM 512 words in size. This RAM is addressed separately from *WAVE* RAM so that *TABLE* processing can be maintained while *WAVES* are continually being output. Nested looping of *WAVE* sequences is possible up to a depth of 6 nested loops. Trigger and jump instructions allow exotic waveform sequences.

2.2 Memory Map

The figure below shows how the data is mapped onto memory by it's associated address. All addresses comprise 16 bits allowing for a total of 65536 different addresses.



- *TABLE* RAM is associated with addresses X0000 to X1FF
- *WAVE* RAM is associated with addresses X0200 to X05FF
- Extended *WAVE* RAM is associated with addresses X0600 to XFFFF

WAVES are programmed anywhere within the region of memory assigned to *WAVES*. TABLES are programmed anywhere within the region of memory assigned to *TABLES* provided the address is X001C or greater. The first 28 addresses within the *TABLE* RAM are reserved for defining the start addresses of TABLES and the start and stop addresses of *WAVES*.

Although *TABLE* and *WAVE* memories appear consecutive, they are accessed independently allowing concurrent processing so that *WAVES* can run seamlessly in most circumstances.

2.3 Table Commands

TABLES are defined as a list of instructions which is illustrated in the table below. This example shows the table necessary to execute a Scan Function an infinite number of times.

item	keyword	qualifier	comment
1	BOL	0	Executed until ABORTed.
2	WAVE 0	512	Ramp up RF Voltage by 1 x 512 ADU. Duration is Number of states (NoS) in WAVE 0 x 512 clock cycles.
3	WAVE 1	466	Keep RF Voltage static for NoS in WAVE 1 x 466 clock cycles.
4	WAVE 2	466	Keep RF Voltage static and Enable Source and Detector Gate for NoS in WAVE 2 x 466 clock cycles.
5	WAVE 3	466	Keep RF Voltage static and Enable Source, Extractor and Detector Gate for NoS in WAVE 3 x 466 clock cycles.
6	WAVE 4	466	Keep RF Voltage static and Enable Source and Detector Gate for NoS in WAVE 4 x 466 clock cycles.
7	WAVE 5	466	Keep RF Voltage static for NoS in WAVE 5 x 466 clock cycles.
8	SYNC		Scan Reset: makes sure Bin number is set to Zero.
9	WAVE 6	89	Keep RF Voltage static, increment Channel Inc. (Bin number) every NoS in WAVE 6 clock cycles and do it 89 times.
10	WAVE 7	871	Ramp up RF Voltage by 2 ADU and Channel Inc. by 1 every NoS in WAVE 7 clock cycles and do it 871 times.
11	WAVE 8	64	Ramp down RF Voltage by 47 ADU and Channel Inc. by 1 every NoS in WAVE 8 clock cycles and do it 64 times.
12	WAVE 9	466	Keep RF Voltage static for NoS in WAVE 9 x 466 clock cycles.
13	EOL		Delimiter for End of Loop.
14	EOR		Stop.

Each item in a *TABLE* has a keyword and a qualifier. *TABLE* items are comprised of a 16 bit word whose 4 least significant bits define the keyword and whose 12 most significant bits define the qualifier. In the case of loop commands the qualifier determines the number of times a particular item is repeated. This can be set from 0 to 4095 and if set to 0 the item will repeat endlessly until an ABORT command is sent or the chip is reset. Loops are defined by placing instructions between a begin of loop command (BOL) and an end of loop command (EOL). A stack internal to the chip allows nested looping to a depth of 6.

TABLE Instruction Set			
qualifier bits 15-4	code bits 3-0	keyword	comment
n	0000	WAVE 0	Output WAVE 0 n times (endlessly if n=0)
n	0001	WAVE 1	Output WAVE 1 n times (endlessly if n=0)
n	0010	WAVE 2	Output WAVE 2 n times (endlessly if n=0)
n	0011	WAVE 3	Output WAVE 3 n times (endlessly if n=0)
n	0100	WAVE 4	Output WAVE 4 n times (endlessly if n=0)
n	0101	WAVE 5	Output WAVE 5 n times (endlessly if n=0)
n	0110	WAVE 6	Output WAVE 6 n times (endlessly if n=0)
n	0111	WAVE 7	Output WAVE 7 n times (endlessly if n=0)
n	1000	WAVE 8	Output WAVE 8 n times (endlessly if n=0)
n	1001	WAVE 9	Output WAVE 9 n times (endlessly if n=0)
-	1010	TRIG	Stop and wait for trigger
-	1011	SYNC	Output a pulse on SYNC output
n	1100	BOL	Start loop, repeat n times (endlessly if n=0)
-	1101	EOL	End of loop
-	1111	EOR	End of Table Read-out
bits 12-4			
n	1110	JUMP	Load n into jump register

WAVE 0-9

These instructions tell the chip to start outputting the relevant WAVE from it's location in the RAM. If a different WAVE is already being output when the instruction is received, then the chip waits until the previous WAVE has been completed before next one is output. When a WAVE finishes, the outputs from the chip will be held at the last state in the WAVE definition. The qualifier indicates the number of times the WAVE is to be repeated - a value of 0 means the WAVE will repeat endlessly. The only way to stop a WAVE which is endlessly repeating is to use the I2C abort command or to reset the chip. These two methods have different effects on the output. In the case of a reset, the WAVE execution is ceased

immediately and the outputs return to a LOGIC 0 state. In the case of an abort command, the WAVE execution continues until the last state in the WAVE definition is reached, and then the outputs are held at this state.

TRIG

The TRIG command halts TABLE execution of a TABLE until a trigger is received via the external pin (triggerB) or through the I2C interface command (Trigger).

SYNC

This instruction produces a pulse on a dedicated output (SYNC_out). The SYNC pulse always appears in the last state of the WAVE preceeding the SYNC instruction in the TABLE definition. If a WAVE is repeated by specifying a qualifier greater than 1, then the SYNC pulse will not appear until the last state of the last repeat of the WAVE. The SYNC pulse can be used as a trigger function for external synchronisation purposes.

BOL

The BOL instruction is used to specify the start of a loop of instructions. The qualifier determines the number of times the loop is repeated. If the qualifier is set to 0 then the loop will repeat endlessly. An endless loop can be stopped using the abort or reset commands or can be jumped out of by defining a table item number in the Jump Register which is outside the loop and then applying a jump command through the input pin or through the I2C interface. Loops can be nested up to a maximum depth of 6. Pushing more than 6 BOLs onto the stack causes the items at the bottom of the stack to be lost.

EOL

This is complementary to BOL, and specifies the end of a loop.

JUMP

This instruction loads the Jump Register with the value specified in the qualifier. With the assertion of a trigger via the external pin (JumpB) or through the I2C command (Jump) the execution of the TABLE will jump to the TABLE item specified in the Jump Register. The first item in a TABLE is item number 1. If the Jump Register is loaded with 0 then any jump triggers which are asserted will be ignored.

EOR

This instruction defines the end of a TABLE.

Because the Table Processor requires one or more clock cycles to process each instruction, there is a limit to the minimum size any WAVE can be to ensure seamless operation. This limit is 4 vectors. Seamless operation cannot be maintained at all times, however. If a series of BOLs or EOLs separate two WAVE commands in a TABLE, then the case may occur where the first WAVE finishes (if it is fairly short) before processing of the BOLs/EOLs has completed. In this case the first WAVE will be stretched by holding the outputs at the last WAVE vector until the second WAVE can begin.

2.4 Communication with the WGA

The chip is programmed over a two wire serial interface designed to conform to the Philips I2C standard. The data will be sent via the RICA as this chip is connected to the WGA via an I2C interface. The current status of the WGA can be found by requesting a Status word from the WGA status register. The Status register is a read

only register that gives the current status of the chip. It includes information about which TABLE is currently being output, reports error conditions, and indicates if, for example, the chip is waiting for a "TRIG" signal. Its bit allocation is listed in the table below

Status Register Definition	
Bit	Definition
15	1 = Wave Active
14	1 = Wave or Table Active
13	1 = Double bit error in WAVE
12	1 = Single bit error in WAVE
11	Wave number bit 3
10	Wave number bit 2
9	Wave number bit 1
8	Wave number bit 0
7	1 = Table Active
6	1 = Table in Halt condition
5	1 = Jump register set
4	1 = Double bit error in TABLE
3	1 = Single bit error in TABLE
2	TABLE number bit 2
1	TABLE number bit 1
0	TABLE number bit 0

3. The RICA Chip

The RICA chips main function is to count and store the Ion Count data from the Ion trap when the Scan Function is run. It does this by placing the Bin Number and its associated Ion Count in a FIFO which is 1024 32-bit locations long. These are accessed through two 16-bit registers. The first register (FIFO_PORT_2) contains the Bin Number (bits 0 to 9, giving a range of 0 to 1023) and the FIFO status (bits 10 to 15). This is read first and the RICA is designed in such a way that the Ion Count for this bin is placed on the second register (FIFO_PORT_1) which is then read. This process is continued until the FIFO is empty. The second register uses bits 0 to 15 (giving a range of 0 to 32767) to store the Ion Count and bit 16 is set high if more than 32767 ion counts have been counted.

The RICA chip will also act as the interface between the Microprocessor and the WGA. All commands sent to the WGA by the Microprocessor will go via the RICA.

4. The Scan Function

4.1 Introduction

The Scan Function consists of a series of WGA Waveforms executed by a specified WGA Table which are loaded onto the WGA then instructed to execute by the Microprocessor via the RICA. This function controls all of the Ion Trap inputs that are required to be user defined and that change over a user defined time period. The execution of the Scan Function, repeatedly over some time period, causes the ions to be produced in the Ion Trap which are then counted by the RICA. The shape and duration of the Scan Function determines the type of ions produced.

4.2 An Example

The way the Scan Function is implemented is best demonstrated by the use of an example. In section 2.3 the example table describes the WGA Table we shall use to describe our Scan Function. The associated Waveforms are listed below. N.B. the prefix '0x' denotes that the digits following are in Hex notation.

WAVE 0 : 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x1. (Number of States is 8).

WAVE 1 : 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0. (Number of States is 9).

WAVE 2 : 0x300 0x300 0x300 0x300 0x300 0x300 0x300 0x300 0x300. (Number of States is 9).

WAVE 3 : 0x700 0x700 0x700 0x700 0x700 0x700 0x700 0x700 0x700. (Number of States is 9).

WAVE 4 : 0x300 0x300 0x300 0x300 0x300 0x300 0x300 0x300 0x300. (Number of States is 9).

WAVE 5 : 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0. (Number of States is 9).

WAVE 6 : 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x4. (Number of States is 47).

WAVE 7 : 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x1 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x1 0x4. (Number of States is 47).

WAVE 8 : 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x2 0x6. (Number of States is 47).

WAVE 9 : 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0. (Number of States is 9).

Each Waveform state is a 16 bit word and each bit of this word has the potential to control a function of the Ion Trap. In this example we have defined :-

To increment Ring RF by one ADU = a '1' in bit '0'.

To decrement Ring RF by one ADU = a '1' in bit '1'.

To increment the Channel number (equivalent to incrementing the Bin Number) = a '1' in bit '2'.

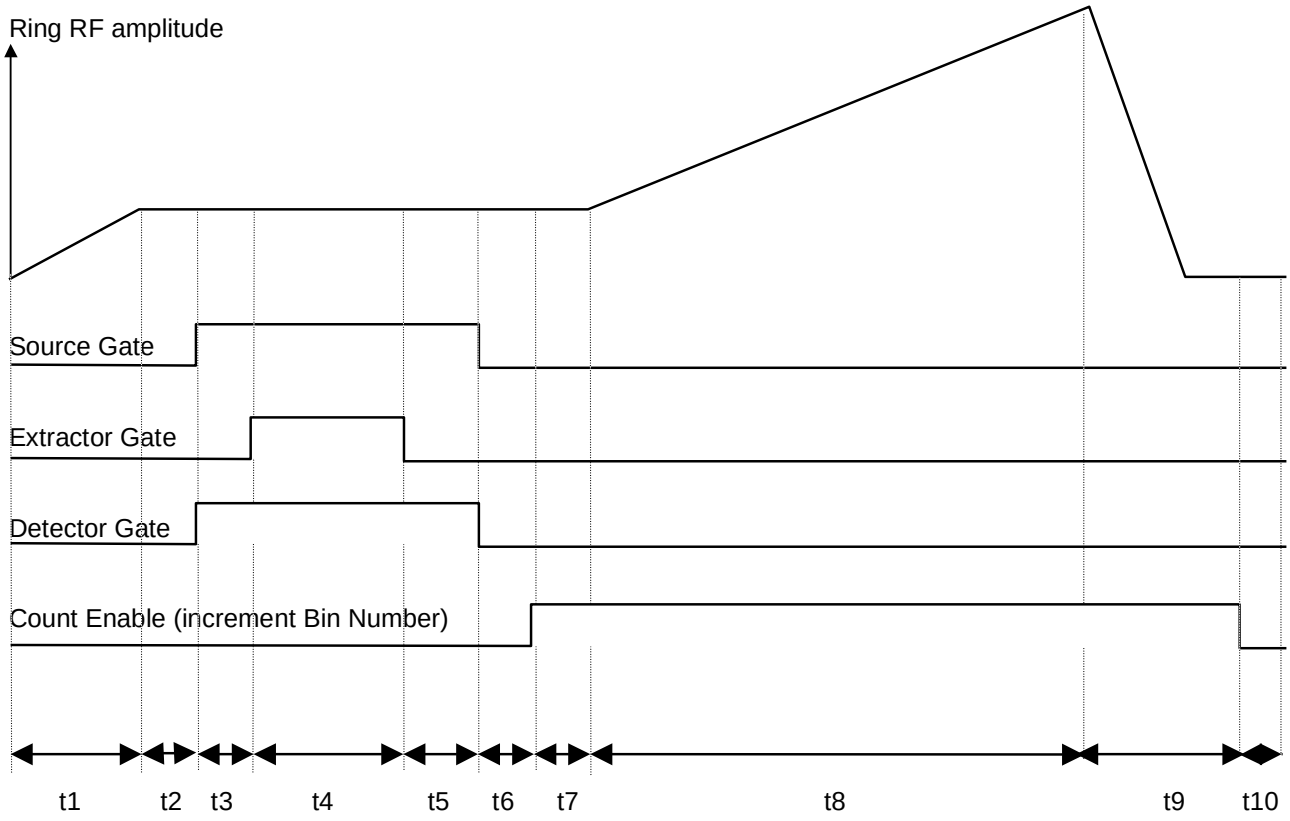
To enable the Source Gate = a '1' in bit '8'.

To enable the Detector Gate = a ‘1’ in bit ‘9’.

To enable the Extractor Gate = a ‘1’ in bit ‘10’.

When the above bits are set to ‘0’ then it is interpreted that nothing has to change or the function is disabled, which ever is appropriate.

The Scan Function represented by this example is displayed below.



Example Scan Function Timings (N.B. the WGA is to be run a 4194304 Hz, this gives a clock cycle duration of 0.2384 μS):-

t1 ⇒ WAVE 0 executed 512 times = 0.9766 mS.

⇒ Ring RF goes from 0 to 512 ADU’s.

t2 ⇒ WAVE 1 executed 466 times = 0.9999 mS.

t3 ⇒ WAVE 2 executed 466 times = 0.9999 mS.

⇒ Source and Detector Gate enabled.

t4 ⇒ WAVE 3 executed 466 times = 0.9999 mS.

⇒ Source, Extractor and Detector Gate enabled.

t5 ⇒ WAVE 4 executed 466 times = 0.9999 mS.

⇒ Source and Detector Gate enabled.

t6 ⇒ WAVE 5 executed 466 times = 0.999 mS.

t7 ⇒ WAVE 6 executed 89 times = 0.9973 mS.

⇒ Use of SYNC before this table instruction is run ensures a scan reset (Channel Inc. or Bin Number is zero).

⇒ Bin Number goes from 0 to 89.

t8 ⇒ WAVE 7 executed 871 times = 9.7601 mS.

⇒ Ring RF goes from 512 to 2254 ADU's.

⇒ Bin Number goes from 89 to 960.

t9 ⇒ WAVE 8 executed 64 times = 0.7172 mS.

⇒ Ring RF goes from 2254 to –754 ADU's.[1]

⇒ Bin Number goes from 960 to 1024.[2]

t10 ⇒ WAVE 9 executed 466 times = 0.9999 mS.

There are a number of things to note:-

1). This Scan Function uses all of the available waveforms in the WGA. It is fortuitous that Source and Detector Gate are Enabled during the same part of the function. If this was not the case, or the shape was different then the function could have more than 10 distinct parts which would require some lateral thinking in identifying those parts which could be described by the same waveform. In the example, for instance, WAVE 1 and WAVE 9 have identical states so WAVE 1 could be used instead of WAVE 9, leaving WAVE 9 free for use elsewhere in the Scan Function.

2). In this example the Count Enable spans 3 waveforms. This constrains the length of all 3 waveforms to be the same and that the number of executions of all 3 waveforms must be 1024 (assuming one Channel Increment per waveform and a resolution of 1024 Bins). In this example the bin size (duration) is 47 clock cycles or 11.2 μS.

3). Any single waveform must be a minimum of 8 states long for WGA chip timing continuity reasons. All 10 waveforms together must not exceed 1024 states, which is the RAM allocated for them on the WGA.

[1] The Ring RF DAC register is 12-bit and thus has a range of values 0 to 4095. It does not wrap round so that an increment applied when it is at 4095 has no effect and a decrement applied when it is at 0 has no effect. So, when ramping down it is possible to execute the Waveform a greater number of times (which may be required by other constraints as in this example) than that necessary to restore the Ring RF to zero.

[2] The Channel Inc. (Bin Number) register is 10-bit and thus has a range of values 0 to 1023. It does wrap round so that an increment applied when it is at 1023 will set it to 0. So, the Bin Number is not 1024 but 0 in this example at the end of the Count Enable phase of the Scan Function.