

MCC-I2.1 Specifications

MCC Design Group

Command Decoder

Revision:	1.0
Reference:	ATLAS ID-xx
Created:	9 December 2003
Last modified:	7 September 2004 15:39
Edited By:	R. Beccherle and G. Darbo ATLAS/Pixel Collaboration

Abstract

This document describes the Command Decoder of the MCC-I2.1.

Table Of Contents

Abstract	ii
List of Figures	v
List of Tables	vii
1 Command Decoder	1
1.1 Control/Data Format and Protocol at MCC Input.	1
1.2 Command Set.	1
1.2.1 Trigger Commands	2
1.2.1.1 LV1: Level 1 Trigger.	2
1.2.2 Fast Commands	2
1.2.2.1 BCR: Bunch Counter Reset.	2
1.2.2.2 ECR: Event Counter Reset	2
1.2.2.3 CAL: Calibration	3
1.2.2.4 SYNC: SyncFE signal	3
1.2.3 Slow Commands	3
1.2.3.1 WrRegister: Write into a Register	4
1.2.3.2 RdRegister: Read from a Register	4
1.2.3.3 WrFifo: Write into a FIFO	4
1.2.3.4 RdFifo: Read from a FIFO	5
1.2.3.5 WrFrontEnd: Write configuration data to a FE chip	5
1.2.3.6 RdFrontEnd: Read configuration data from a FE chip	5
1.2.3.7 WrReceiver: Write data into a Receiver	6
1.2.3.8 EnDataTake: Enable Data Take	6
1.2.3.9 GlobalResetMCC: Reset MCC.	7
1.2.3.10 GlobalResetFE: Reset FE chips	7
1.3 Reset Actions	8
1.4 Timing Requirements	8
1.5 Command Robustness in Case of Transmission Errors	9
1.5.1 Trigger: LV1	9
1.5.2 Fast: BCR	10
1.5.3 Fast: CAL	11
1.5.4 Fast: ECR	11
1.5.5 Fast: SYNC	12
1.5.6 Slow: Any Command.	13
1.6 SEU protection	15
1.7 Architecture	16
1.8 References.	16

List of Figures


- Fig. 1-1** *p. 16* Block diagram for the Command Decoder algorithm. The square boxes contain the number of CK cycles before process execution; the * can be either 0 or the appropriate length of the Slow command data field (see Table 1-2).

List of Tables

Table 1-1	<i>p. 2</i>	Control Commands.
Table 1-2	<i>p. 4</i>	Slow commands body (Field 3 and Field 4) and, if present, data (Field 5).
Table 1-3	<i>p. 8</i>	Consecutive commands timing.
Table 1-4	<i>p. 9</i>	Single bit flip in a LV1 command.
Table 1-5	<i>p. 10</i>	Single bit flip in a BCR command.
Table 1-6	<i>p. 11</i>	Single bit flip in a CAL command.
Table 1-8	<i>p. 12</i>	Single bit flip in a SYNC command.
Table 1-7	<i>p. 12</i>	Single bit flip in an ECR command.
Table 1-9	<i>p. 13</i>	Single bit flip in a SLOW command.


1 Command Decoder

1.1 Control/Data Format and Protocol at MCC Input

 The serial command protocol defined for the *MCC* has been largely copied from the *ABC*¹ chip command set. The serial command protocol implemented for the *MCC-I2.1* uses an improved robustness for eventual bit errors on the *opto-link*.

Commands for the *MCC* are organized in a tree hierarchy and are divided in three main command blocks:

1. **Trigger** is a 5-bit long word so that a new LV1 can be issued every 125 ns (5 CK clock cycles, i.e. 5 bunch crossing) as required now by ATLAS. This is by far the most frequently issued command during the experiment running.
2. **Fast** are 9-bit long words, a 5-bit header (10110) and a 4-bit body. They can be issued in the absence of LV1 commands without blocking the data acquisition.
3. **Slow** are from seventeen to thousands of bits long and they block event data acquisition (if running) when issued. The commands are logically subdivided into a 9-bit header (10110.1011), an 8-bit body and eventually a variable length data field.

 **Trigger** and **Fast** commands can be interleaved, while **Slow** commands stop the event read-out. The **Trigger** and **Fast** commands are executed only if the *MCC* is in **RunMode**. To put the *MCC* in **RunMode** one has to issue an **EnDataTake** command. In this way the configuration and data taking modes are completely decoupled.

Command encoding is given in Table 1-1 and Table 1-2.


The Control / Data protocol uses *MCC-DCI* as a data line and *MCC-CK* as a data validation line. Data must be stable on the rising edge of *MCC-CK*. Data are allowed to change at each clock cycle unless they transport information going to the *FE* chips. In this last case each information bit must be kept stable for 8 clock cycles.

1.2 Command Set

Commands are divided into 3 groups: **Trigger**, **Fast** and **Slow**. In the **Trigger** group the only command is the LV1, in the **Fast** group there are 4 commands and the **Slow** group allows up to 16 different commands out of which only 10 have been implemented in the *MCC*.

Commands can have up to 5 different fields. Fields from 1 to 4 have fixed length while **Field 5**, if it exists, has a variable length which depends on the command itself.

Field 1 and **Field 2** for the different commands are listed in Table 1-1. **Field 1**, that is always present and is used to select between the LV1 and the other commands. **Field 2** selects between a specific **Fast** command and the **Slow** command header. All other fields are only meaningful for **Slow** commands and are described in Section 1.2.3 .

 The hierarchy of resets implemented in the *MCC* decouples completely the *MCC* from the *FE* chips. Therefore no reset command acts on both.

1. Atlas Binary Chip [1-2]

Table 1-1 Control Commands.

Type	Name	Field 1	Field 2	Field 3	Description
Trigger	LV1	11101			Level 1 Trigger
Fast	BCR	10110	0001		Bunch Counter Reset
Fast	ECR	10110	0010		Event Counter Reset
Fast	CAL	10110	0100		Calibration Pulse
Fast	SYNC	10110	1000		SyncFE signal.
Slow	CMD	10110	1011	Command	Slow command header. (For body/data see Table 1-2)

1.2.1 Trigger Commands



The *MCC* acts on a **Trigger** command only when the *MCC* is in **RunMode**. Therefore before issuing a **Trigger** command it is necessary to execute an **EnDataTake** command.

1.2.1.1 LV1: Level 1 Trigger

In the **Trigger** group the only element is the LV1. The LV1 command is only executed if the *MCC* is in **RunMode**. The LV1 triggers the acquisition of a new event in the *Pixel module* and the main actions in the *MCC* are:

- LV1 is issued to the *FE* chips, on the MCC-LV1 output pin., if the PendingLv1Fifo is not full. Contiguous LV1 are issued to the *FE* chip if register LV1 <11:8> != 0;
- The current L1ID and BCID numbers are stored in the PendingLv1Fifo to be used by the event builder when the event will be reconstructed;
- The L1ID counter is incremented.
- In case we are in **EventPlayback** mode (bit CSR<6> = 1) the only difference is that no LV1 are issued to the *FE* chips (for more details on this test mode please refer to Section 4.2.1, "MCC PlaybackMode" of the "Event Builder" chapter).

1.2.2 Fast Commands



The *MCC* acts on a **Fast** command only when the *MCC* is in **RunMode**. Therefore before issuing a LV1 or a **Fast** command it is necessary to execute an **EnDataTake** command.

In the **Fast** commands group there are 4 different commands.


1.2.2.1 BCR: Bunch Counter Reset

The BCR counter inside the *MCC* is set to 0 in response to this command. This command has no effect on any other internal structure of the *MCC* and has also no effect on the *FE* chips.


1.2.2.2 ECR: Event Counter Reset

The ECR command is ment to completely clear the *MCC* data path and can be issued in order to synchronize event building in case problems with data alignment are detected.

ECR command resets the pointers of all the **ReceiverFifo**'s, clears the **PendingLv1Fifo**, clears the **Event-Scoreboard**, the value of the **L1ID** is set to 0 and the system is ready for a new event. If there are pending events to be built, the ECR will clear them.

 The **ROD** must leave enough time from the last **LV1** issued to be sure that there are no more events to be transmitted from the **FE** chips, because if some data arrive to the **MCC** after **ECR** execution and not in response to a **LV1** results are unpredictable until a new **ECR** is issued.

This command has no effect on the **FE** chips.


 In order to completely clear the module's data path one has to issue first a **GlobalResetFE** command, followed by an **ECR** command.

1.2.2.3 CAL: Calibration

A pulse is generated on the **MCC-STRO** pin in response to a **CAL** command. The **DelayLine** is enabled by setting the **CAL <10>** bit. **CAL <9:6>** sets the range of the selected delay, while the delay of the pulse is defined by the contents of the **CAL <15:9>** register. The pulse width, in **CK** units, is defined by the contents of **CNT <15:0>** register.


1.2.2.4 SYNC: SyncFE signal

The **MCC** generates a **SyncFE** signal that lasts five **CK** units on the **MCC-SYNC** pin.

 the **ROD** has therefore to ensure that there will not be any **Trigger** commands issued for at least 4 **CK** cycles after a **SyncFE** command.

Nothing is done inside the **MCC**.

1.2.3 Slow Commands

 When the **MCC** acts on a **Slow** command the chip is taken out of **RunMode**. Therefore before issuing a **LV1** or a **Fast** command, after a set of **Slow** commands, it is necessary to execute an **EnDataTake** command.

There are 10 different **Slow** commands implemented in the **MCC**.

The set of **Slow** commands implemented in the **MCC** is listed in Table 1-2. The **Slow** command is recognized once the appropriate **Slow** command header (**Field 1** and **Field 2**) is found. The following fields are:

- **Field 3**: this 4-bit field always exist and it selects the **Slow** command;
- **Field 4**: this 4-bit field always exist but is only used for **WrRegister/RdRegister** and **RdFifo**. It is advisable to fill this field with zeroes if it is not used in the command;
- **Field 5**: all write operations use this field to store the data to write into the **MCC** or into the **FE**. Read operations (with the exception of **RdFrontEnd**) do not look into this field, but a new command must not be put inside this field to avoid command overrunning or command misinterpretation. **EnDataTake** and **GlobalResetMCC** do not use this field while **GlobalResetFE** uses a 4-bit field.

Table 1-2 Slow commands body (Field 3 and Field 4) and, if present, data (Field 5).

Name	Field 3	Field 4 ¹	Field 5	Bits ²	Description
WrRegister	0000	Address	Data	16	Write to MCC register
RdRegister	0001	Address	----	16	Read from MCC register
WrFifo	0010	----	Data	27	Write a word to the enabled <i>FIFO</i> (s)
RdFifo	0011	Address	----	27	Read a word from the addressed <i>FIFO</i>
WrFrontEnd	0100	----	Data	Var ³	Write to the FE chips.
RdFrontEnd	0101	----	Data	Var ³	Read (or R/W) from the Enabled FE chips
WrReceiver	0110	----	Data	Var ⁴	Write to Enabled Receiver Channels
EnDataTake	1000	----		0	Enable Data Taking
GlobalResetMCC	1001	----		0	Global MCC Reset
GlobalResetFE	1010	----	SyncW	4	Global FE Reset

Note:

- 1) Field 4 is always 4 bits long. The recommended pattern when not used is 0000.
- 2) Length of Field 5 is variable and it is specified in this column.
- 3) Var = (CNT<15:3> x8 + CNT<2:0> ¥ 64) CK units, i.e. (CNT<15:13> + CNT<2:0>x 8) bits.
- 4) Var = (CNT<12:0> ¥ 8) CK units, i.e. CNT<12:0> x 8 bits.

1.2.3.1 WrRegister: Write into a Register

This command writes the contents of Field 5 into the addressed *MCC* register.
Field 4 is used to address the register to write to. This command does not produce any *MCC* output.



In case of non existing register address the command is ignored. No internal flag in the *MCC* is set in such a case.

1.2.3.2 RdRegister: Read from a Register

This command reads the content of the addressed *MCC* internal register.
Field 4 is used to address the register to be read out.
Field 5 of the command is not used, but must always be provided (0000_0000_0000_0000 is the suggested value). Its length is 16 bits.



In case of non existing register address a value of 0x0000 is read back. No internal flag in the *MCC* is set in such a case.

1.2.3.3 WrFifo: Write into a FIFO

This command writes a word into the enabled *MCC* ReceiverFifo's.
FIFO's are enabled by setting to '1' the corresponding bit in the FEEN register. This allows to write the same data value into more than one *FIFO* with a single command.

The content of Field 4 is ignored but must be provided (0000 being the suggested value).
Field 5 of the command contains the word to be written to the *FIFO*.

The write pointer of the addressed ReceiverFifo is incremented after each write operation.



Being the write pointer cyclic, after 128 WrFifo commands it will point to the same location as before the sequence of commands.

1.2.3.4 RdFifo: Read from a FIFO

This command extracts the word addressed by the read pointer of the selected ReceiverFifo.

The ReceiverFifo is selected by the 4-bits in the Field 4 of the command. The Field 5 of the command is not used, but must always be provided (000_0000_0000_0000_0000_0000 being the suggested value). Its length is 27 bits.

The read pointer of the addressed ReceiverFifo is incremented after the read operation.



Being the read pointer cyclic, after 128 RdFifo commands it will point to the same location as before the sequence of commands.

1.2.3.5 WrFrontEnd: Write configuration data to a FE chip

The WrFrontEnd command is used for writing configuration data into *FE* chips.

Field 4 is not used but must be present (0000 being the suggested value).

The bit stream in Field 5 is sent to the MCC-DAO pin.

Input data routed to DAO have a total length of $CNT <15:3> + CNT <2:0> \times 8$ bits (in CCK units). The *FE* interprets the first $CNT <2:0> \times 8$ bits as address / command, while MCC-LD is low, and the following $CNT <15:3>$ bits as data, while MCC-LD is high. Setting of MCC-LD signal depends on CNT contents as can be seen in Section 5.2, "MCC to FE Configuration Data Format" of the "Output Data Format" chapter. The whole information stream going to the *FE* must be sent to the MCC with every bit repeated 8 times in order to account for the lower clock frequency used to configure the *FE* chip.



In case the number of control bits is not a multiple of 8, zeroes must be added in the MSB part of the control bits to create a n-bit word, where n is multiple of 8, and n/8 must be loaded in CNT<2:0> to set up the transmission to the *FE* chips.



All data are sent to all *FE* chips ignoring the settings of FEEN register. The *FE* chip itself recognizes data depending on the geographical address information present in the data stream.

This command does not produce any MCC output (on the MCC-DTO pins).

1.2.3.6 RdFrontEnd: Read configuration data from a FE chip

The RdFrontEnd command is used to read (and write) configuration data from (to) the *FE* chips.

Field 4 of the command is ignored, but must be present (0000 being the suggested value).

The content of Field 5 is transmitted to all the *FE* chips (usually only *FE* address plus *FE* command without any data).

The *FE* that recognizes the address responds with the requested data. The MCC adds the header bits (11101) to the received data and retransmits them through the MCC-DTO pins (the same signal is found on both MCC-DTO-0 and MCC-DTO-1 at a speed of 40 Mbit/s independent of the OM<1:0> settings.). Data received from the *FE* chip by MCC-DTI pins are validated by the rising edge of the MCC-CCK clock (running at 5 MHz) and then retransmitted through the MCC-DTO pins using the MCC-XCK clock (40 MHz).

RdFrontEnd can write new data into the addressed *FE* chip while is reading present data: this is useful when the read operation would destroy the data (like for the PixelShiftRegister) and would require a sub-

sequent reload of them. In this last case, the Data field of the instruction (see Field 5 of Table 1-2) carries the new data to be reloaded. The length of the data field is set by the CNT <15:3> register.



The selection of the *FE* chip from which to get data is done by proper enabling of the MCC-DTI input pin. This is done by setting the corresponding bit of the FEEN register (as example to read data from *FE* chip 6, the value of FEEN must be 0x0040). If more than one bit is set the MCC-DTO's output is the bitwise OR of the enabled *FE* chips.



The *MCC* does not affect the data stream coming from the *FE* chips: since the data input clock (MCC-CCK) is eight times slower than the output one (MCC-CK), each bit from MCC-DTO, excluding the header bit, transporting *FE* configuration data lasts 8 CK clock cycles.



Even for this command one has to ensure that, if present, the number of control bits is a multiple of 8, eventually padding the MSB part of the control bits with zeroes.

1.2.3.7 WrReceiver: Write data into a Receiver

The WrReceiver command allows to selectively write data into any FIFO of any Receiver block.

This command is designed to exercise the *MCC* Receiver and EventBuilder blocks with simulated events. The main aim of this test feature, is to test the *MCC* functionality in the experiment using known events. The sequence of operations to use is described in full detail in the Receiver and Event Builder chapters (For example see Chapter 4.2.1.1, "Event Building Test Example").

Field 4 is ignored but must be present (0000 being the suggested value).

Input data in Field 5 of this command, feeds into the Receivers inputs of the enabled *FE* receivers.

Receiver(s) are enabled by the corresponding bit set in the FEEN register. More than one Receiver can be enabled at one time, meaning that several Receivers will receive the same data stream. The length of the data stream which is sent to the Receiver(s) is the content of the CNT <12:0> counter times 8 (CK is used to feed data to the Receiver(s)).



If one needs to send more than 65535 bits ($2^{13} \times 8 - 1$) to the enabled Receiver blocks more consecutive WrReceiver commands need to be used.

WrReceiver is foreseen for debugging purposes; this is a way of simulating both the *FIFO* filling and the event building processes without real data coming from the *FE*s.



Therefore one has to ensure that the data written to the *MCC* FIFO's is written according to the data format described in Chapter 5.3, "FE to MCC Event Format", which implies that each word (either Hit or EoE) is 26 bit long.



One has to ensure that all FEEN enabled *FE* chips do not send any data while using the WrReceiver command as in such a case a logical OR of data coming from the WrReceiver command and from FE-DTI data lines would be written to the ReceiverFifo's.

1.2.3.8 EnDataTake: Enable Data Take

The EnDataTake command is used to put the *MCC* in RunMode.



Each time a Slow command is issued the *MCC* is taken out of RunMode. Trigger and Fast command are executed only if the *MCC* is in RunMode.



The **RunMode** status bit is not readable from outside, as it would be cleared by a **RdRegister (Slow)** command.

Field 4 must be present even if not used (0000 being the suggested value), while Field 5 is not present.

1.2.3.9 GlobalResetMCC: Reset MCC

This command is equivalent to the **Pin-Reset**, which will not be used by the *MCC* during the experiment. All main blocks of the *MCC* are therefore affected by this command, with the only exception of the **CommandDecoder** itself.

The effect of the **GlobalResetMCC** command on the *MCC* is to bring the internal registers, status flags, *FIFO*'s pointers to their default values and to set the finite state machines to their idle state.



One has therefore to reload all *MCC* configuration data after issuing a **GlobalResetMCC** command.

Field 4 of the command must be present, even if it is not used (0000 being the suggested value), whereas Field 5 is not present.

More detailed information on the effect of **GlobalResetMCC** on the single *MCC* blocks can be found in the "Reset Actions" section of each chapter.

The *FE* chips are not affected by this command.

1.2.3.10 GlobalResetFE: Reset FE chips

This command generates a **MCC-SYNC** signal that does a global reset of all the *FE* chips.

Field 4 of the command must be present for fields alignment purposes (0000 being the suggested value), but it's contents are ignored. The **SyncW** (Width of the **SYNC** signal) parameter, to be written in Field 5 of the command, defines the width of the **SYNC** produced at the *MCC* output. This term is mandatory.

That width is given in **CK** units according to the formula: $\text{Width}(\text{SYNC}) = \text{SyncW} \times 2 + 1$ in **CK** units, which ranges from 1 to 31 **CK** units (i.e. from 25 to 775 ns). The *FE* chip executes different levels of reset according to the width of the **SYNC** signal.

As an example a value of **SyncW** = 3 will produce a reset signal that is active for 7 **CK** units.

This command has no effect on the *MCC*.

1.3 Reset Actions



The Command Decoder block is not reset by any of the reset commands (**GlobalResetMCC**, **ECR**, **BCR**). The finite state machines always goes, by construction, to the idle state from any possible state in a maximum time of about 0.9 ms.

A pin reset (**MCC-RS1b**) exists and has the same functionality as the **GlobalResetMCC** command with the addition that it resets also all the command decoder to its idle state, but will not be used during the experiment. This is the only way to do a hard reset of the command decoder block.

The polarity of the pin reset is negative, i.e. it is active low. This pin will not be connected (internal pull-up) in the final ATLAS modules.



The pin reset is synchronous and therefore, in order to reset the chip, one has to ensure that the **MCC** is clocked while this signal is active in order to reset the chip.

1.4 Timing Requirements

There are no restrictions on the **LV1** commands: **LV1**'s can follow each other without any gap. This implies that the minimum distance between consecutive **LV1** commands is 125 ns, conforming to the ATLAS specifications.

A **LV1** command can be issued immediately after a **Slow** command, with the exception of the **SYNC** command. In this case the **ROD** has to ensure that there will be at least 100 ns before issuing the next **LV1** command.

Fast commands and **Slow** commands can be issued without any gap between them. There is however a recommendation to leave at least two clock cycles (50 ns) between consecutive commands in order to minimize the risk of command misinterpretation in case of a bit flip (see Table 1-3).

Table 1-3 Consecutive commands timing.

Command	Following Command	Distance (CK units)
LV1	LV1	0
LV1	Fast, Slow	0
Fast: BCR, ECR, CAL	LV1	0
Fast: SYNC	LV1	4
Fast: BCR	Fast, Slow	2
Fast: CAL, SYNC	Fast, Slow	0
Fast: ECR	Fast, Slow	1
Slow	Any command	see text

After a **Slow** command, if there is a bit-flip, no prediction of misinterpreted commands can be easily performed, due to the possible presence of a variable length data field. It is therefore strongly advisable to read back all written configuration data.

In case of no transmission errors a **Fast** or **Trigger** command can be issued immediately after a **Slow** command without any gap (the last **Slow** command has to be **EnDataTake** in order to have the following **Fast** or **Trigger** commands interpreted correctly by the **CommandDecoder**).

1.5 Command Robustness in Case of Transmission Errors



The command set chosen for the *MCC* is optimized for the occurrence of a low rate of bit flip in the input data stream. In particular the command set satisfies the following requirements:

1. LV1 are always correctly interpreted even in the presence of a single bit flip in the data stream. In this case also the correct timing information is reconstructed.
2. A given **Fast** command is never recognized as a different **Fast** command in case of a single bit flip in the data stream.
3. A **Fast** command is never interpreted as a **Slow** command in presence of a single bit flip in the data stream.
4. A **Slow** command header is never interpreted as a **Fast** command. Body and data fields command can be, in case of transmission errors, misinterpreted as **Fast** or **Trigger** commands. **Trigger** and **Fast** commands are not active if the *MCC* is not in **RunMode**.

After a command is decoded (correctly recognized pattern) the input shift register is reset in order to start the decoding of a new command in a clean condition.



In case the **CommandDecoder** does not correctly recognize a **Trigger**, **Fast** or **Slow** command a warning bit is set in **CSR<10:8>**. (Refer to Section 2.1.1.6, Section 2.1.1.7 and Section 2.1.1.8 of the "Register Bank" chapter for the definition of these three bits.)

1.5.1 Trigger: LV1

The pattern chosen for the LV1 command is "11101". This pattern has a minimum Hamming distance¹ of 3 bits from the other **Fast** and **Slow** commands.

As for the event data header output format this pattern allows for correct command recognition in case of any single bit flip in the data stream (either inside or outside of the five command bits).

Table 1-4 Single bit flip in a LV1 command.

Received Pattern	Recognized Pattern	Recognized Command
0000_11101_00	0000_11101_00	LV1
0000_11101_10	0000_11101_10	LV1
0000_11100_00	0000_11100_00	LV1 (w. bit-flip)
0000_11111_00	0000_11111_00	LV1 (w. bit-flip)
0000_11001_00	0000_11001_00	LV1 (w. bit-flip)
0000_10101_00	0000_10101_00	LV1 (w. bit-flip)
0000_01101_00	0000_01101_00	LV1 (w. bit-flip)
0001_11101_00	0001_11101_00	LV1
0010_11101_00	0010_11101_00	LV1
0100_11101_00	0100_11101_00	LV1

Trigger commands can be contiguous; since the **Trigger** command pattern is 5 bit long the minimum distance between two LV1 commands is 125 ns. A **Trigger** command can follow any other **Fast** command without any gap and after a **Trigger** command any other command can be issued without gap.

1. The Hamming distance of two n-bit words is the number of bits which differ in the two words.

The command decoder is designed in such a way that a single bit flip anywhere in the command pattern is always interpreted as a correct LV1 command, preserving the timing information too. Therefore also the patterns 11100, 11111, 11001, 10101, 01101 are decoded as “correct” LV1 commands.



In case the command decoder detects a LV1 with a bit flip it sets a warning bit in the CSR register (CSR<8> := WLV1).

Table 1-4 shows the behaviour of the command decoder in presence of a single bit flip during a LV1 command. In the first column the effect of bit flip on the command pattern is shown in bold, the second one shows in bold the pattern recognized by the command decoder, while the last column tells which command is detected.

1.5.2 Fast: BCR

The pattern chosen for the BCR command is 10110.0001. In case the command decoder detects a bad Fast command due to a transmission error it sets a warning bit in the CSR register (CSR<9> := WFST).

A Fast command is never interpreted (in case of a single bit-flip) as a different Fast command because the Hamming distance between two Fast commands is always two. A Fast command can be detected as a bad Fast command, which is defined by the correct Fast header (10110) and a body which does not map to any Fast command.

Table 1-5 Single bit flip in a BCR command.

Received Pattern	Recognized Pattern	Recognized Command
0000_10110.0001_00	0000_10110.0001_00	BCR
0000_10110.0001_10	0000_10110.0001_10	BCR
0000_10110.0000_00	0000_10110.0000_00	bad Fast command
0000_10110.0011_00	0000_10110.0011_00	bad Fast command
0000_10110.0101_00	0000_10110.0101_00	bad Fast command
0000_10110.1001_00	0000_10110.1001_00	bad Fast command
0000_10111.0001_00	0000_10111.0001_00	LV1 (w. bit-flip)
0000_10100.0001_00	-	missed command
0000_10010.0001_00	-	missed command
0000_11110.0001_00	0000_11110.0001_00	LV1 (w. bit-flip)
0000_00110.0001_00	-	missed command
0001_10110.0001_00	0001_10110.0001_00	LV1 (w. bit-flip)
0010_10110.0001_00	0010_10110.0001_00	LV1 (w. bit-flip)
0100_10110.0001_00	0100_10110.0001_00	BCR

Table 1-5 shows the behaviour of the command decoder in presence of a single bit flip during a BCR command. In the first column the effect of bit flip on the command pattern is shown in bold, the second one shows in bold the pattern recognized by the command decoder, while the last column tells which command is detected. As can be seen in the table only 4 times a bit flip produces a LV1. In these cases the LV1 is detected as a LV1 with bit-flip and therefore the WLV1 bit in the CSR is set. In other 4 cases where the Fast command is not detected, the WFST bit in the CSR register is set (bad Fast command). There are 3 more cases in which no command at all is recognized and therefore the command is simply ignored (missed command).

To avoid misinterpretation of commands following a BCR in case of bit flip it is required to leave a gap of 2 CK cycles between a BCR and any other Fast or Slow command. There are no restrictions on Trigger commands following a BCR command.

1.5.3 Fast: CAL

The pattern chosen for the CAL command is 10110.0100. In case the command decoder detects a bad Fast command due to a transmission error it sets a warning bit in the CSR register (CSR<9> := WFST).

Table 1-6 Single bit flip in a CAL command.

Received Pattern	Recognized Pattern	Recognized Command
0000_10110.0100_00	0000_10110.0100_00	CAL
0000_10110.0100_10	0000_10110.0100_10	CAL
0000_10110.0101_00	0000_10110.0101_00	bad Fast command
0000_10110.0110_00	0000_10110.0110_00	bad Fast command
0000_10110.0000_00	0000_10110.0000_00	bad Fast command
0000_10110.1100_00	0000_10110.1100_00	bad Fast command
0000_10111.0100_00	0000_10111.0100_00	LV1
0000_10100.0100_00	-	missed command
0000_10010.0100_00	-	missed command
0000_11110.0100_00	0000_11110.0100_00	LV1 (w. bit-flip)
0000_00110.0100_00	0000_00110.0100_00	LV1 (w. bit-flip)
0001_10110.0100_00	0001_10110.0100_00	LV1 (w. bit-flip)
0010_10110.0100_00	0010_10110.0100_00	LV1 (w. bit-flip)
0100_10110.0100_00	0100_10110.0100_00	CAL

Table 1-6 shows the behaviour of the command decoder in presence of a single bit flip during a CAL command. In the first column the effect of bit flip on the command pattern is shown in bold, the second one shows in bold the pattern recognized by the command decoder, while the last column tells which command is detected. As can be seen in the table only 5 times a bit flip produces a LV1. In 4 of these cases the LV1 is detected as a LV1 with bit-flip and therefore the WLV1 bit in the CSR is set. In the fifth case a normal Trigger command is detected. In other 4 cases where the Fast command is not detected, the WFST bit in the CSR register is set (bad Fast command). There are 2 more cases in which no command at all is recognized and therefore the command is simply ignored (missed command).

There are no restrictions on Trigger, Fast and Slow commands following a CAL command.

1.5.4 Fast: ECR

The pattern chosen for the ECR command is 10110.0010. In case the command decoder detects a bad Fast command due to a transmission error it sets a warning bit in the CSR register (CSR<9> := WFST).

Table 1-7 shows the behaviour of the command decoder in presence of a single bit flip during an ECR command. In the first column the effect of bit flip on the command pattern is shown in bold, the second one shows in bold the pattern recognized by the command decoder, while the last column tells which command is detected. As can be seen in the table only 4 times a bit flip produces a LV1. In these cases the LV1 is detected as a LV1 with bit-flip and therefore the WLV1 bit in the CSR is set. In other 4 cases where the Fast command is not detected, the WFST bit in the CSR register is set (bad Fast command).

Table 1-7 Single bit flip in an ECR command.

Received Pattern	Recognized Pattern	Recognized Command
0000_10110.0010_00	0000_10110.0010_00	ECR
0000_10110.0010_10	0000_10110.0010_10	ECR
0000_10110.0011_00	0000_10110.0011_00	bad Fast command
0000_10110.0000_00	0000_10110.0000_00	bad Fast command
0000_10110.0110_00	0000_10110.0110_00	bad Fast command
0000_10110.1010_00	0000_10110.1010_00	bad Fast command
0000_10111.0010_00	0000_10111.0010_00	LV1 (w. bit-flip)
0000_10100.0010_00	-	missed command
0000_10010.0010_00	-	missed command
0000_11110.0010_00	0000_11110.0010_00	LV1 (w. bit-flip)
0000_00110.0010_00	-	missed command
0001_10110.0010_00	0001_10110.0010_00	LV1 (w. bit-flip)
0010_10110.0010_00	0010_10110.0010_00	LV1 (w. bit-flip)
0100_10110.0010_00	0100_10110.0010_00	ECR

There are 3 more cases in which no command at all is recognized and therefore the command is simply ignored (missed command).

To avoid misinterpretation of commands following a ECR in case of bit flip it is required to leave a gap of 1 CK cycle between an ECR and any other Fast or Slow command. There are no restrictions on Trigger commands following an ECR command.

1.5.5 Fast: SYNC

The pattern chosen for the SYNC command is 10110.1000. In case the command decoder detects a bad Fast command due to a transmission error it sets a warning bit in the CSR register (CSR<9> := WFST).

Table 1-8 Single bit flip in a SYNC command.

Received Pattern	Recognized Pattern	Recognized Command
0000_10110.1000_00	0000_10110.1000_00	SYNC
0000_10110.1000_10	0000_10110.1000_10	SYNC
0000_10110.1001_00	0000_10110.1001_00	bad Fast command
0000_10110.1010_00	0000_10110.1010_00	bad Fast command
0000_10110.1100_00	0000_10110.1100_00	bad Fast command
0000_10110.0000_00	0000_10110.0000_00	bad Fast command
0000_10111.1000_00	0000_10111.1000_00	LV1 (w. bit-flip)
0000_10100.1000_00	-	missed command
0000_10010.1000_00	-	missed command
0000_11110.1000_00	0000_11110.1000_00	LV1
0000_00110.1000_00	0000_00110.1000_00	LV1 (w. bit-flip)
0001_10110.1000_00	0001_10110.1000_00	LV1 (w. bit-flip)
0010_10110.1000_00	0010_10110.1000_00	LV1 (w. bit-flip)
0100_10110.1000_00	0100_10110.1000_00	SYNC

Table 1-8 shows the behaviour of the command decoder in presence of a single bit flip during a **SYNC** command. In the first column the effect of bit flip on the command pattern is shown in bold, the second one shows in bold the pattern recognized by the command decoder, while the last column tells which command is detected. As can be seen in the table only 5 times a bit flip produces a LV1. In four of these cases the LV1 is detected as a LV1 with bit-flip and therefore the WLV1 bit in the CSR is set. In the fifth case a normal **Trigger** command is detected. In other 4 cases where the **Fast** command is not detected, the WFST bit in the CSR register is set (bad Fast command). There are 2 more cases in which no command at all is recognized and therefore the command is simply ignored (missed command).

There are no restrictions on **Trigger**, **Fast** and **Slow** commands following a **CAL** command

1.5.6 Slow: Any Command

The pattern chosen for the **Slow** command header is 10110.1011.


In case the command decoder detects a bad **Fast** command due to a transmission error it sets a warning bit in the CSR register (CSR<9> := WFST).

Table 1-9 Single bit flip in a SLOW command.


Received Pattern	Recognized Pattern	Recognized Command
0000_10110.1011	0000_10110.1011_00	Slow
0000_10110.1011_10	0000_10110.1011_10	Slow
0000_10110.1010_00	0000_10110.1010_00	bad Fast command
0000_10110.1001_00	0000_10110.1001_00	bad Fast command
0000_10110.1111_00	0000_10110.1111_00	bad Fast command
0000_10110.0011_00	0000_10110.0011_00	bad Fast command
0000_10111.1011_00	0000_10111.1011_00	LV1
0000_10100.1011_00	0000_10100.1011_00	(bad) Fast command
0000_10010.1011_00	0000_10010.1011_00	LV1 (w. bit-flip)
0000_11110.1011_00	0000_11110.1011_00	LV1
0000_00110.1011_00	0000_00110.1011_00	LV1 (w. bit-flip)
0001_10110.1011_00	0001_10110.1011_00	LV1 (w. bit-flip)
	0001_10110.1011_00	LV1 (w. bit-flip)
0010_10110.1011_00	0010_10110.1011_00	LV1 (w. bit-flip)
	0010_10110.1011_00	LV1 (w. bit-flip)
0100_10110.1011_00	0100_10110.1011_00	Slow

Table 1-9 shows the behaviour of the command decoder in presence of a single bit flip during a **CAL** command. In the first column the effect of bit flip on the command pattern is shown in bold, the second one shows in bold the pattern recognized by the command decoder, while the last column tells which command is detected. As can be seen in the table, there are 6 cases where a bit flip produces a LV1:

- In two cases a normal LV1 is detected;
- In two cases a single LV1 with bit-flip is detected, the WLV1 bit of CSR is set;
- In two case two contiguous LV1 with bit-flip are detected, the WLV1 bit of CSR is set.

 As during a slow command the *MCC* is not in RunMode, even if a pattern is recognized as a **Trigger** or a **Fast** command, no action, in response to such a command, is performed inside the *MCC* itself and no **Trigger** signal is sent to the *FE* chips.

There are four cases where the **Fast** command is not detected and the **WFST** bit in the **CSR** register is set (bad **Fast** command). In a further case (“(bad) **Fast** command”) either a **LV1**, or a **Fast** command or a bad **Fast** command can be detected.

 In all cases the body and data fields, if present in the command, are not “protected” from being misinterpreted. If the **Slow** header is missed, the stream of bits in the body and data fields, if present in the command, can fake any command (**Trigger**, **Fast** or **Slow**). For this reason **Fast** and **Trigger** commands are disabled when the **MCC** is not in **RunMode**.

Each time the **Field 3** of the command does not match an existing **Slow** command, a warning flag is set in the **CSR** register (**CSR**<10> := **WSLW**).

1.6 SEU protection

Being the **CommandDecoder** one of the most important blocks for correct module functioning particular care has been put in trying to protect the circuit against possible occurrence of **SEU** events.

We decided to completely triplicate the **CommandDecoder**, so that there are actually three distinct copies of the **CommandDecoder** itself inside the **MCC** and all its outputs are selected with a majority voting circuit.



Due to the fact that, by design, all the state machines that form the **CommandDecoder** itself return to their idle state after a finite amount of time (usually after a small number of clock cycles and only for certain configuration commands after less than 0.9 ms) there is a really small probability that a bit-flip in any state machine of the **CommandDecoder** will cause the majority voting circuit to fail.

This could only happen if two **SEU** induced bit-flips occur in two different state machine status registers before the machines themselves returned to their idle state. Even in such a case, as soon as the first of the two affected state machines returns to its idle state, the majority voting logic would work correctly again.

In addition to the majority voting circuit there is an **XOR** tree of all triplicated outputs in order to calculate, once every clock cycle, if there has been a detection of a **SEU** event inside the **CommandDecoder**. If an error is found the information is stored in **CSR<14>**.

1.7 Architecture

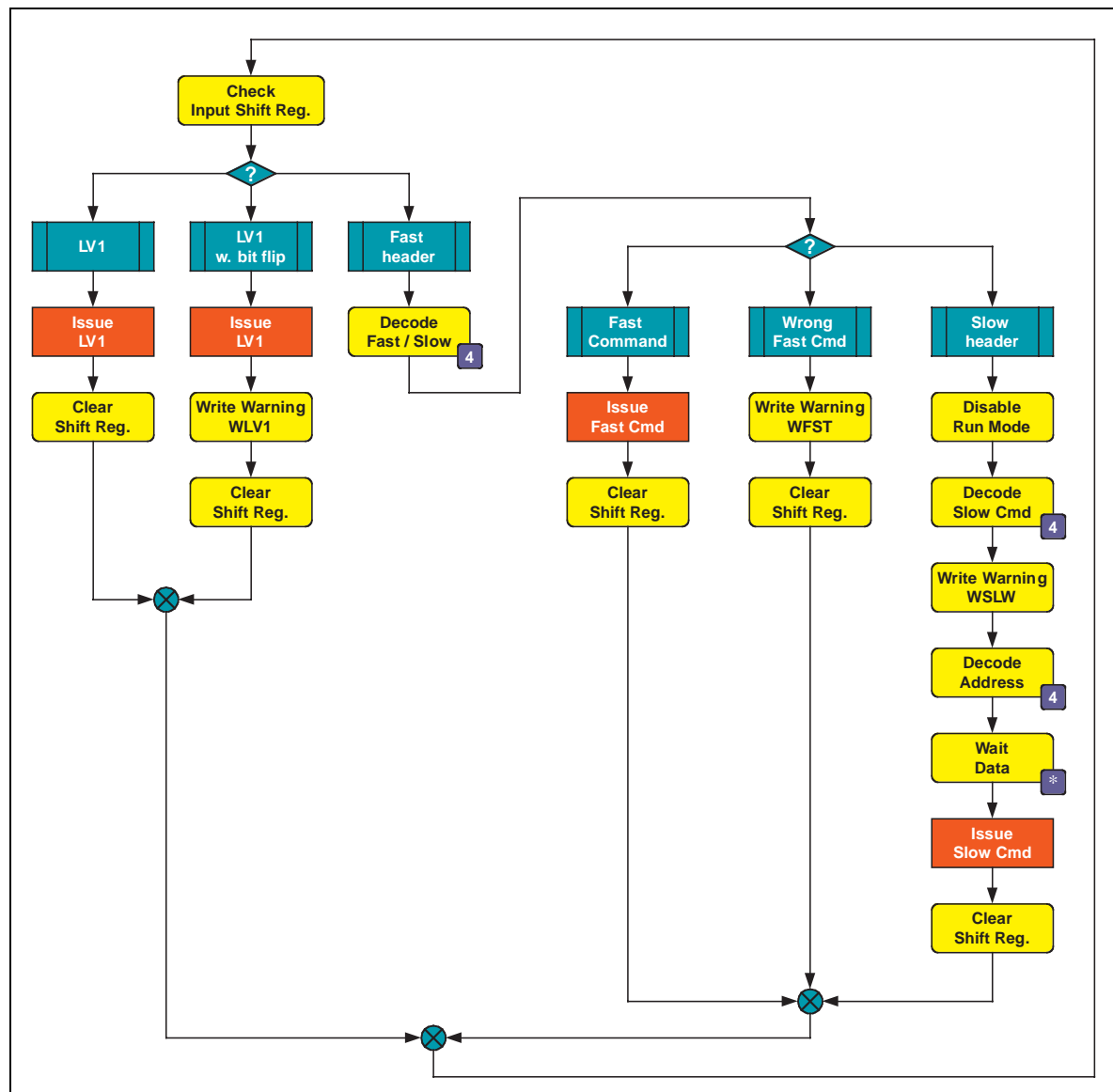


Figure 1-1 Block diagram for the Command Decoder algorithm. The square boxes contain the number of CK cycles before process execution; the * can be either 0 or the appropriate length of the Slow command data field (see Table 1-2).

The command decoder is implemented using a shift register that is continuously loaded with data from the MCC-DCI input. Each time a command (Trigger, Fast or Slow) is completed the input shift register is cleared. A flow chart of the command decoding algorithm is shown in Figure 1-1.

1.8 References

- 1-1 G. Darbo, K. Einsweiler and P. Fischer, Atlas Pixel Demonstrator: System Architecture Ver.2.0, <http://www.ge.infn.it/ATLAS/Electronics/Demonstrator-20/MCMSpec.2.0.pdf>.
- 1-2 Atlas Binary Chip (ABC) Specifications.