

DDD (DC-2, DM-115, DPM) V3.2 User Guide

John Anderson

Jim Franzen

Oscar Trevizo

Vince Pavlicek

Online Support Department
Fermilab Computing Division

Tsuyoshi Nakaya

Toshihiro Tsuji

Taku Yamanaka

KTeV

Osaka University

ABSTRACT

This document describes the function and usage of the DDD (DM115/DC2/DPM) data acquisition module which is used to receive and buffer data in the DART architecture. This document also describes the function of the DDD firmware developed by Osaka University and Fermilab.

1	Introduction	4
1.1	Number Conventions	5
1.2	WWW resources	5
2	How the DDD works	6
2.1	How the DDD receives data	6
2.2	How the data is organized in the DPM	7
2.3	How the VME host communicates with the DC-2	8
2.4	External control signals	9
2.5	DDD States	9
3	Installing and using the DDD	11
3.1	Configuration Considerations	11
3.1.1	The number of planes	11
3.1.2	How to select the destination event building plane	11
3.1.3	DPM configuration	11
3.1.4	DPM Address space	12
3.1.5	Partition of DPM memory space	12
3.2	Hardware Installation	12
3.2.1	VSB bus/terminator cards	12
3.2.2	DPM address settings	13
3.2.3	DM-115 jumpers and dip switches	13
3.2.4	Installation	14
3.2.5	DM-115 cabling	14
3.2.6	Accessing the DC-2 through a terminal line	14
3.3	Software Setup	15
3.3.1	Set parameters in DPM	15
3.3.2	How to send commands to the DC-2	16
3.3.3	Setting the broadcast address bits	16
3.3.4	Exit from casemode to collect data	17
3.3.5	Activate the DC-2	17
3.3.6	Clear the DC-2	17
3.4	Spill to spill operation	17
3.4.1	Clearing the DDD	17
3.4.2	Send data to the DDD	18
3.4.3	Reading data out of the DPM	18
4	DC-2 Communication with the VME host	20
4.1	Buffer Addresses (offsets 0x00 - 0x1C)	22
4.2	Statistics (offsets 0x20 - 0x2C)	22

4.3	DPM Memory Size (offset 0x30)	22
4.4	cleared_flag (offset 0x34)	22
4.5	polling_period (offset 0x38)	23
4.6	hold_off_clear (offset 0x3C)	23
4.7	heart_beat (offset 0x40)	23
4.8	dc2_status (offset 0x44)	24
4.9	error_code (offset 0x48)	24
4.10	dc2_response (offset 0x4C)	24
4.11	command and arguments (offsets 0x50, 0x54-0x7C)	25
4.12	Pointer Table address and length (offsets 0x80-0x84)	27
4.13	dm115_status (offset 0x90)	28
4.14	user_bits (offset 0x94)	29
4.15	Ping-Pong information (offsets 0x98-0xAC)	29
5	DC-2 Commands for data acquisition	30
6	Diagnostics commands in casemode	33
6.1	Entering and exiting the Casemode	33
6.2	Testing of DC-2 hardware	34
6.2.1	Test patterns	34
6.2.2	Repeat count	35
6.2.3	dc2_response	35
6.2.4	Error histograms	35
6.2.5	Commands for testing RAM/DPM/FIFO and DMA	36
6.3	Testing External Lines and LEDs	38
6.4	Other Diagnostics Modes	39
7	The DDD software	41
7.1	Overview	41
7.2	View from the DC-2	41
7.3	Boot-up	43
7.4	Main Program - DMA Loop	43
7.4.1	DMA Loop operation	43
7.4.2	DMA Loop pseudo code	45
	Appendix A: DDD Versions - Release Notes	48
A.1	New Features in V2.2	48
A.2	New Features in V2.3	48

A.3	New Features in V2.4	49
A.4	v2.5	50
A.5	v2.6	51
A.6	v2.7	51
A.7	v2.8	51
A.8	v3.0	51
A.9	V3.1	53
A.10	V3.2	53

1 Introduction

Many experiments which use the Fermilab data acquisition system, DART¹, require a buffer between data source and the event builder electronics. The buffer serves to accept events during the beam spill, hold events during possible processing between and during the spills, and to distribute events between multiple event processing planes if necessary. The buffer consists of a matrix of memories as shown in FIGURE 1., where data from the front end electronics comes in parallel vertically (data streams) and data in the same horizontal row are built, processed and logged in parallel. Depending on the data rate and number of channel requirements, the number of rows and columns can range from 1x1 to nxm.

FIGURE 1. Buffer matrix

Each node of the matrix is a combination of three electronics modules; one Access Dynamics² DM-115 Input Module, one Access Dynamics DC-2 VSB Input Controller, and one or more VME/VSB dual ported memory (DPM) modules. This set is called a DDD based on the first letters of the three elements.

The DM-115 receives data from the front-end electronics through RS-485 (differential TTL) cables, and passes the data to the DC-2. The DM-115 also has flow control and status lines specific for DART applications.

The DC-2 is a general purpose VSB master, whose main function is to transfer data from its daughter card, the DM-115, to the DPM over the VSB bus. The DC-2 has a 32kbyte FIFO, and a Motorola MC68340 processor with DMA controllers. The DC-2 can run software from internal EPROM or downloaded into RAM. The DDD software, which is described in this document, is the software in the EPROM developed by members of the DAAR collaboration.

The VME/VSB DPM can be written to and read from over both busses. In DART applications, data from the front end is written into the DPM through VSB by the DC-2/DM-115. Data is then read out over the VME back plane to the VME host such as a Silicon Graphics Challenge through VME-VME interface or a local Motorola MVME162 or 167. Multiple DPMs can be connected to a single DC-2 using multi-slot VSB bus/terminator cards.

This document is organized as follows.

¹. G. Oleynik et al, DART - Data acquisition for the next Generation Fermilab Fixed Target Experiments, IEEE Transactions on Nuclear Science, Vol 41, No 1.

². 1. Access Dynamics, Inc., 3823 Hawkins N.E., Albuquerque, NM 87109, (505) 345-7637.

Section 2 describes how the DDD works.

Section 3 describes how to install and use DDDs with regard to both in hardware and software.

Section 4 describes in more detail on how a DC-2 is controlled from VME hosts.

Section 5 describes the commands available for data acquisition.

Section 6 describes how to run diagnostics on DC-2.

Section 7 describes how the DC-2 software works, in detail.

1.1 Number Conventions

Binary and decimal numbers will be distinguished by context and content i.e. 01001011 can be assumed binary. Hexadecimal numbers will be preceded by **0x** as in 0xFE01.

1.2 WWW resources

Additional Information and documents can be found on the World Wide Web at:

Fermilab Computing Division On-line Systems DART pages-

<http://www-dart.fnal.gov/>

Fermilab Computing Division Electronic Systems Engineering DART pages-

<http://www-ese.fnal.gov/eseproj/dart/>

2 How the DDD works

The DDD works as a simple linear buffer with the data written sequentially as it received. The data in the buffer can be read out through the VME bus even when data is coming in through the VSB bus because of arbitration between VME and VSB ports on the DPM. The DDD can also execute various tasks by receiving commands from VME, or by receiving an external signal.

The following sections describe in more detail how a DDD receives data, how data is written to memory, and how the DDD interacts with a VME host and the outside world.

2.1 How the DDD receives data

The DDD receives data from readout controllers, such as DY3s, FSCCs, and CTIRCs, via two RS485 (differential TTL) cables. The cables can be daisy-chained between multiple readout controllers and multiple DDDs. By using a token passed between readout controllers on one RS485 line, a DDD can collect data from multiple front-end crates. Multiple DDDs on the same RS485 line can receive data, so that experiments with high data rates can also use multiple event processing planes.

The signals on the two RS485 cables are summarized in Table 1 and Table 2.

TABLE 1. Signals on the 50 pin RS485 cable

pins	signal	direction
1,2	data bit 0	to DDD
3,4	data bit 1	to DDD
...	...	to DDD
31,32	data bit 15	to DDD
33,34	data strobe	to DDD
37,38	WAIT	from DDD
41,42	EOR	to DDD

TABLE 2. Signals on the 34 pin RS485 cable

pins	signal	direction
1,2	data bit 16	to DDD

...	...	to DDD
31,32	data bit 31	to DDD

The DART data bus is four bytes wide and a transfer can occur every 100 ns. Therefore, the maximum transfer rate is 40Mbytes/sec. After the last word of each event is sent, the EOR (End Of Record) signal is sent to indicate that the event has finished.

The data will be received by the DDDs that satisfy one of the following conditions.

1. The DM-115 is configured to receive all the data. (Jumper pin 5-6 on J9 on the DM-115 is in.)
2. The DM-115 is configured to recognize either destination (plane or broadcast) address bits embedded in the first word in the event by removing the 5-6 jumper on J9, and the DM-115s address matches with the destination address. See 3.2.3 for further details.

If the event is accepted by the DM-115, the data is pushed into the 32kbyte FIFO in the DC-2. As data comes into the FIFO, it will be DMAed into the dual port memory through the VSB port at up to a peak rate of 21.5 Mbytes/sec. The FIFO is used to absorb the 40Mbytes/sec data bursts on RS485 to the slower DC-2 to DPM DMA rate. The FIFO will continue receiving data even if DMA transfers are temporarily blocked by the DPM being read out by the VME host.

When the DC-2 FIFO becomes half full, the WAIT signal is sent back to the front-end readout controllers. The readout controllers should hold off sending data until the WAIT goes away which happens when the FIFO becomes less than half full.

2.2 How the data is organized in the DPM

Viewed from a VME host, the memory inside of each dual port memory node is divided into three sections; a data buffer, a pointer table, and an area called the mailbox. The mailbox is used for communication between the VME host and the DDD. The location and size of the data buffer and pointer table are specified by the user. The location and size of the mailbox is fixed at the beginning of the DPM memory space.

The data buffer contains events received by DDD. The data buffer is filled from the beginning and will not wrap around. Each event has its inclusive byte count stored in the first long word that is loaded by the DC-2 after the event is received. Each event written is always followed by a 0 word to indicate that the following event is 0 bytes long. This becomes the byte count word for the next event after it is received. Some user-defined information can be included in the byte count word if desired. See section 4.14 on page 29.

The event pointer table is also filled from the beginning. Each entry points to the beginning of an event of data as shown in FIGURE 2. Thus, the size of each event can be calculated from the

addressees in the pointer table alone. This feature is important for deciding the number of events to read out during a VME DMA transfer. For further details, see section 4.12 on page 27. The addresses in the pointer table are the VSB addresses, so, to use the information from the VME bus side of the DPM, an address conversion may have to be done.

FIGURE 2. The map of DPM

2.3 How the VME host communicates with the DC-2

The mailbox area is used for the communication between the VME host and the DC-2. The meaning of each word in the mailbox area is described in more detail in Section 4. One of the frequently used words is the command word at 0x50. It is usually written to by the VME host. The DC-2 checks the command word periodically, and if a command is entered, the DC-2 executes the corresponding function. A handshaking protocol allows the host to determine if the DC-2 is executing or has finished a command. All the available commands are listed and explained in Section 5 and Section 6, but frequently used commands include: CLEAR to clear DDD before each spill and ACTIVATE to make the DDD alive after power on. The size and location of the data buffer, the pointer table, and the BAF threshold are specified in the mail box area. These values become effective on the following CLEAR command. In addition, the word *hold_off_clear* tells the DC-2 what to do when it receives an EXTERNAL_ABORT signal (see Section 4.6 and Section 3.4.1). The DC-2 supplies the necessary status information including the number of events in the DDD, current mode and operation, and error codes to the VME host via this mailbox area.

2.4 External control signals

Several control signals connect between the DDD and the outside world via a ten-pin connector on the DM-115. A flat RS-485 cable can be daisy chained across multiple DDDs in the same VME subrack to distribute the common control signals. The signals are:

EXTERNAL ENABLE IN (pin 1,2) forces the DDD to accept data. This line is currently ignored.

EXTERNAL RESET IN (pin 3,4) can reset and reboot the DDD.

EXTERNAL ABORT IN (pin 5,6) may be given by outside logic to the DDD before the next spill begins. THIS SIGNAL SHOULD STAY ASSERTED FOR LONGER THAN TWICE THE COMMAND POLLING PERIOD SPECIFIED IN MAILBOX. If the *hold_off_clear* word in the mailbox area is zero, then the signal will be treated exactly as same as receiving a CLEAR

command. This reinitializes DDD for the new spill. This feature is useful to clear all the DDDs externally. If the *hold_off_clear* word is not zero, then the DC-2 will not execute a CLEAR command but, instead, sets the VETO signal. The VETO will stay on until a CLEAR command is given by the host. This is useful for the user who wants to read out all the events from the previous spill. The VETO signal should be used to veto new triggers, and CLEAR should be issued only after all the events in the DDD are read out.

BAF OUT (Buffer Almost Full, pin 7,8) is asserted when the amount of data in buffer exceeds a threshold set by the user, at the end of event. The BAF signals from multiple DDDs in the same plane (VME subrack) can be WIRE-ORed together to veto further triggers. BAF is also available in NIM level on a LEMO connector next to the ten-pin connector. Even if BAF is set, the DDD is capable of accepting events until the buffer overflows.

VETO OUT (pin 9,10) is asserted on receipt of EXTERNAL ABORT only if *hold_off_clear* word in the mailbox area is non-zero. This signal should definitely veto triggers, whereas the BAF signal could be ignored to collect necessary calibration events, etc.

2.5 DDD States

The DC-2 data collection firmware has several modes. The two key modes are data acquisition mode (which is called *DMA loop or mainmode*) and diagnostic mode (which is called *casemode* for historical reasons). When the DDD is reset, it enters the diagnostic mode, again for a historical reason, and waits for commands. Section 6 has further details about diagnostics that can be run in casemode. The user switches back and forth between modes by giving commands to the DC-2. To switch to data acquisition mode, execute command ENTER_MAINMODE (0xFE). The previous command (EXIT_CASEMODE) still functions but is being phased out.

Within data acquisition mode, the two states are activated and deactivated. When the DDD is deactivated, it will not receive incoming data, will not respond to external control signals, and will not send out WAIT, BAF, or VETO. This state makes the DDD almost transparent to the system. When DDD is reset and switched to mainmode, it enters the deactivated state. Thus, when the power of the VME subrack is turned on, the DDD will not interact with outside world until the DDD is initialized properly including parameters in DPM.

When ACTIVATE command (0x04) is given, the DDD becomes visible to the world. However, some parameters must still be loaded from DPM to DC-2 by the CLEAR command, so the DDD sets BAF to indicate it is not ready to receive data yet.

When the additional parameters are loaded from DPM and a CLEAR command is given before a spill begins, the DDD will be ready to receive data from the next spill. This cycle will be repeated until the run is over.

If the memory matrix is reconfigured for some reason, DDDs in the streams or planes which are not needed can be virtually removed from the system with the DEACTIVATE (0x05) command. CLEAR (or EXTERNAL ABORT) can still be delivered to a deactivated DDD but it will stay in the deactivated state and will not take data until it is activated again.

3 Installing and Using the DDD

3.1 Configuration Considerations

This chapter discusses how to configure, install, and use DDD in a step by step fashion. Before acting on this information, the user should have already read through this guide and understand the design of this module and how the module fits into the overall hardware configuration of the experiment.

3.1.1 The number of planes

First, it is necessary to decide how many planes (the number of DDDs per data stream from front end) that the DA system is going to need for the raw data quantity and rate. If the average data rate from the front end is less than 20Mbyte/sec, one event building plane should be sufficient. If there are two planes, the data rate to the DDDs can be 40Mbyte/sec on average. The requirement for more than two planes is determined by the readout rate from the DDD through the DPM to the VME host, and the event processing rate.

3.1.2 How to select the destination event building plane

If there is only one event building plane, permanently ENABLE all the DM-115s by removing the pins 1-2 and 5-6 jumpers at J9³.

For multiple planes, it is more powerful to use the addressing capability of the DM-115. In this mode, the DM-115 will examine bits 12-19 in the first word of every event to decide whether to accept the event data or not. These address and broadcast bits must be supplied at the front end by external logic. This allows the user to specify the destination for every event, depending on other information such as the trigger type and/or the event building plane status. The setting for this configuration are described later in this manual. See DM-115 jumpers and dip switches on page 13.

To use the destination address capability of the DM-115, all the DM-115's in the same VME subrack should have the same 4 bit plane address as they will be collecting data from the same events in one plane. Of course, the plane address should be different between planes.

³. 1. Access Dynamics, Inc., DM-115 RS-485 Input Module, Document M1115A, 1993. Hard copy only.

3.1.3 DPM configuration

Despite its name, the dual ported memories cannot be written and read at *exactly* the same time. When reading from one port and writing from the other tries to overlap on the same DPM, the data throughput will be decreased even more than expected because of the logic that arbitrates accesses. As described in the KTeV Data Acquisition System⁴ document, the dead time caused by such an overlap can be as large as 30%. Let:

T_w = time spent to write DPM per second, if DPM is 100% alive, and

T_r = time spent to read DPM per second, if DPM is 100% alive.

The live time available for writing will then be:

$$L_w = (1 - T_r) / (1 - T_w \times T_r).$$

If the expected data input rate to a single DDD is less than $L_w \times 21\text{Mbyte/sec}$, there will not be problems from the dead time due to the read/write time slicing. If the expected rate is higher than $L_w \times 21\text{Mbyte/sec}$, use multiple DPMs per DC-2/DM-115 and change to a multi-slot VSB bus terminator card. KTeV's technique is to align DPMs with the sizes with the ratio of 1: 2: 4: 8: etc., in contiguous VME and VSB address spaces. Writing starts from the DPM with the smallest capacity. Do not start reading until the DC-2 starts writing to the second DPM. As long as the read/process operation is slower than half the writing speed, there will never be read/write overlaps on the same DPM and this will avoid the overlap dead time penalty. For a more detailed description, please see the KTeV Data Acquisition System document.

3.1.4 DPM Address space

The VSB address of each memory node should be set to start at 0x2000 0000. If the DDD node consists of multiple DPMs, the VSB address space of each DPM should be consecutive and contiguous above the first DPM.

The VME address space of DPMs within a subrack must not overlap.

3.1.5 Partition of DPM memory space

The DDD uses an event pointer table to contain pointers to every event in the event data buffer space. To determine the length of the pointer table needed for a DA system, determine the maximum possible number of events expected during a spill, multiply by the safety factor of

⁴. 1. Vivian Odell and Taku Yamanaka, KTeV Data Acquisition System, <http://fnxp03.fnal.gov/experiments/ktev>

choice, and multiply by 4 (bytes per pointer). Only one event pointer table is used for each DDD node, regardless of the number of DPMs used for the node.

It is strongly recommend that the start of the event pointer table be offset from the base address of the DPM by 256 bytes (0x100). This places the pointer table immediately after the mailbox communication area and simplifies some of the operations during data movement.

The event buffer space can start anywhere after the event pointer table but usually follows closely because there is rarely memory to waste.

3.2 Hardware Installation

WARNING: Whenever inserting or removing DPMs or DC-2s within a VME subrack, TURN OFF the subrack power supply first. Unlike CAMAC, VME modules can be damaged if this rule is not observed. DPMs mount into the front cardcage of a VME subrack while the associated DC-2/DM-115 module plugs into the rear slot directly in line with the DPM.

3.2.1 VSB bus/terminator cards

Install VSB bus/terminator cards for each slot that will contain a DDD. They go in the lower half (J2/P2 bus) of the back of the backplane in the VME subrack. All DC-2s must plug into VSB bus/terminator cards. The single slot VSB backplane provides active terminators for the VSB signal lines of a single DDD. If there are multiple DPMs on a single DDD, then use a multiple slot VSB backplane bus/terminator card for each DC-2. The board provides signal bussing not provided by the P2 backplane. The DC-2 can mount in any slot of a multiple slot terminator but an end slot is recommended over one in the middle.

3.2.2 DPM address settings

Set VME and VSB low and high address ranges of each DPM. For MMI 6346 and 6390 modules, place the DPM so that the front panel is facing away (i.e. dip switches at the top). Starting from the right end, there are 12 bits for VME high limit, 12 bits for VME low limit, and after LEDs, 4 bits for VME (A24) high limit, 12 bits for VSB high limit, and 12 bits for VSB low limit. The 12 switch positions correspond to address bits A20 to A31, from right to left. The address bits A00-A19 are assumed to be \$00000 for low limit, and \$ffff for high limit. Note that the high limit is an inclusive address. On the dipswitches, OPEN means 1, and CLOSED means 0.

For example, to configure a 32Mbyte DPM such that its VSB address range is 0x2000,0000 through 0x21ff,ffff, and the VME address range is 0x0800,0000 through 0x09ff,ffff, the dip switches should be set from the left end:

```
00100000 0000 00100001 1111 1111 (LEDs) 00001000 0000 00001001 1111
VSB Low limit VSB High limit A24.....VME Low limit VME High limit
```

If the 32Mbyte DPM is followed by a 64Mbyte DPM, then the dip switches on the 64Mbyte DPM should be set as:

```
00100010 0000 00100101 1111 1111 (LEDs) 00001010 0000 00001101 1111
VSB Low limit VSB High limit A24.....VME Low limit VME High limit
```

The manufacturer now recommends that the jumper at E4, pins 2 to 3, be set, to enable the VME bus **RESCINDED DTACK*** to obtain the maximum bus bandwidth. Read p.2-3 of MM6390 Users Guide for more details. There have been no reported problems with using this option.

The factory installed default settings for other jumpers on the DPM are recommended.

3.2.3 DM-115 jumpers and dip switches

Hold the DC-2/DM-115 with the component side of the DM-115 up. It is partially covered by the DC-2. Turn the board so the RS485 connectors and LEDs are on the left, and the VSB connector is on the right. The dip switch and jumper pins are near the center of the components. The jumpers set addressing and data modes as follows.

The jumper pins 1-2 enables/disables data flow into the DDDs. This jumper (rightmost pins) on J9 should normally be removed to allow data to flow into the DC-2 FIFO. This is only disable for specific diagnostics and testing.

The jumper pins 3-4 selects which control line strobes the first word in an event data stream. This jumper should be left in to use the **data_strobe** signal because no experiments are using the special **address_strobe** signal for this function.

The jumper pins 5-6 jumper enables/disables the plane/broadcast address decoding function within the DM-115.

The jumper pins 7-8 jumper selects which address bits are used for destination addressing, the switches at S3 or the DM-115 register loaded with the SET_BROADCAST_ADDR command. See section 3.3.3 on page 16

If the DA system only needs a single plane, insert a jumper on pins 5-6 (third from the right) on J9. Jumper settings on pins 3-4, and 7-8, and S3 dipswitch settings are irrelevant with a single plane.

If there are multiple planes, the addressing mode must be enabled. The normal configuration has jumpers on J9 set as follows: 1-2 = OUT, 3-4 = IN, 5-6 = OUT, 7-8 = OUT, from right to left. The lower 4 bits (rightmost 4 switches, right end = LSB) on S3 should be the plane address

(ON=0). The upper 4 bits (leftmost 4 switches, left end = MSB) on S3 should be ON to select 0 for the broadcast address. The DDD V3.1 or earlier has no functionality to set the four bit plane address number by software.

For more details, see the DM-115 manual⁵.

3.2.4 Installation

Turn off the VME subrack power supply. Install the DPM(s) in front cardcage of the VME subrack. For each DPM or group of DPMs, install the associated VSB bus/terminator and DC-2/DM-115 in the back cardcage in line with the DPM(s). The leftmost slot in a VME subrack is usually occupied by the bus master, either a local CPU like a MVME162 or a bus-to-bus interface to the external VME host.

3.2.5 DM-115 cabling

The DART system uses two RS-485 cables to move data. A 50-pin ribbon cable carries control and data bits 0 through 15. A 34-pin ribbon cable carries data bits 16 through 31. The DDDs in the same stream should be daisy chained by the 50 pin and 34-pin ribbon cables. The DDD terminating SIP resistors for RS485 lines (R7 - R17) should be removed except for special situations involving a single source feeding a single DDD over a short cable. The two DART data cables should be terminated by FERMI terminating networks called UTNs and DTNs. There is an upstream (U) and downstream (D) version. If the RS485 cable is longer than 100 feet, or the number of data sources exceeds eight, independent of the length, a DART Repeater⁶ should be inserted to enhance the signal.

The ten pin connectors on the DM-115s in the same VME subrack (plane) should be daisy-chained together to send/receive control signals. All the SIP resistors for the ten pin connector (R22 - R24) should be removed except for the last DM-115 in the control line cable. The BAF signal is also available as a NIM signal at the Lemo connector next to the ten-pin connector.

3.2.6 Accessing the DC-2 thorough a terminal line

In order to develop and debug DC-2 code through a terminal line, hook up an IBM PC via a terminal line to port P3 on the DC-2. See the WWW posting:

http://fndaub.fnal.gov:8000/usr/products/pc/dart/ddd/v2_7/doc/html/dc2download.html

⁵. 1. Access Dynamics, Inc., DM-115 RS-485 Input Module, Document M1115A, 1993. Available in hardcopy only.

⁶. This is different from the fiber optic cable extender for E781. See the ESE web pages.

3.3 Software Setup

After DDD is reset or powered on, DPM mailbox parameters should be initialized using the VME host before executing any DC-2 commands to begin data collection. If the parameters are not set to valid starting values, the software will not operate correctly.

3.3.1 Set parameters in DPM

Set the following parameters (size = four bytes) in the DPM. The parameter *vme_base* refers to the base address of the DPM in VME space.

Table 3 shows the variables that must be set with configuration dependent parameters before using a DC-2

Table 4 shows the parameters which it is recommended the user initialize in addition to the required variables.

TABLE 3. Absolutely necessary parameters

VME address	contents
vme_base+0x14	VSF address of event data buffer space
vme_base+0x18	VSF address of BAF threshold
vme_base+0x1C	VSF address of end of event buffer space
vme_base+0x38	Command polling period (See polling_period (offset 0x38) on page 23.) If 0, the DC-2 will set a default value of 0x5fff (0.25sec).
vme_base+0x3C	hold_off_clear See Clearing the DDD on page 17.
vme_base+0x80	VSF address of event pointer table
vme_base+0x84	length of event pointer table in bytes
vme_base+0x94	User bits. See user_bits (offset 0x94) on page 29.

TABLE 4. Recommended parameters

VME address	contents
vme_base+0x24	0 (n_BAF) number of BAFs. Clear it.
vme_base+0x28	0 (n_timeout) number of time-outs. Clear it.
vme_base+0x2C	0 (n_drain) number of drains. Clear it.
vme_base+0x48	0 (error_code) last error code. Clear it.

3.3.2 How to send commands to the DC-2

The DC-2 will periodically look at the mailbox *command* word (vme_base+0x50), and when it finds a value other than 0, it loads arguments starting in mailbox arguments word (vme_base+0x54) as necessary. Then it clears the command word and executes the command. The DC-2 also loads its current state to the *dc2_response* word (vme_base+0x4c). If there was an error, an error code may be returned in the *error_code* word (vme_base+0x48).

Care must be taken when sending a series of commands to the DC-2 from a VME host program. Command repose is not deterministic because commands are retrieved at a period specified by the mailbox *polling_period* word (vme_base+0x38) which is usually between 0.1 and 1 sec. Therefore, before sending another command, wait until the current command is accepted by the DC-2. This will be indicated by the cleared command word. NOTICE: The command is accepted when the command word is cleared. A command is finished when the *dc2_response* word changes to command<<8+0xF0. A second command may be sent after the first command is accepted but before it is completed. However, the host software must be capable of reconciling multiple command responses correctly

To send a command with arguments,

1. write arguments, up to ten, in arguments words.,
2. write the command word.
3. wait until the command word becomes 0, and
4. to be safe, wait until the *dc2_response* word shows that the command is finished.

3.3.3 Setting the broadcast address bits

In large DA systems where there are several event building planes (multiple DDDs connected to the same data stream), and it is desirable to broadcast events to selected planes, use the DC-2/DM-115 broadcast address mode. This will require setting hardware jumper J9 pins 7&8 shorted. There are 4 (3 if there are any FSCC⁷s in the system) broadcast address patterns that can be set. The DC-2 does a bitwise AND between the 4 bit broadcast address and bits 16-19 of the first word of the event, and if the result is non-zero, the event will be accepted regardless of a match with the lower 4 bits, bits 12-15.

To use this feature,

1. set the 4 bit broadcast address in the mailbox arguments word (vme_base+0x54), and

⁷. 1. The latest FSCC hardware document, HN130, is available from ftp://dahserv/USERS/anonymous/FSCC/HW_SPEC/PC4A/DRAFT!

2. write the SET_BROADCAST_ADDR command (0x8) in the mailbox command word (vme_base+0x50).

The DC-2 will detect the command word and read the value contained in the *arguments* word, and set the DM-115 address. Bits 4-7 in the dc2_status word (vme_base+0x44) will show the new broadcast address, and bits 0-3 will have the 4 bit address set by dip switches on the DM-115. See TABLE 6. for more information on the dc2_status word.

3.3.4 Exit from casemode to collect data

Exit from the casemode and enter DMA_LOOP by executing ENTER_MAINMODE (0xFE) (used to be EXIT_CASEMODE). The dc2_status word should be 0x0j10nm, where j is a jumper setting, and n is the 4bit broadcast address set by SET_BROADCAST_ADDRESS command, and m is the 4bit plane address set by a dip switch on the DM-115. The dc2_response (vme_base+0x4C) word will change to 0xFE0 when the command completes.

3.3.5 Activate the DC-2

When the DC-2 is reset or powered on, it is in a *deactivated* state where it does not receive data or assert BAF. In order to activate the DC-2, write the ACTIVATE command (0x4) to the mailbox command word (vme_base+0x50).

The *dc2_status* word (vme_base+0x44) should contain 0x1jC0nm, where j, n, and m are described in Section 3.3.4. This means that it is in the active state and BAF is on.

The DC-2 is not supposed to take data until the CLEAR command is received (which is why BAF is set to indicate that it is not ready). This precludes receiving data if the DC-2 is set to the active state during a spill. The DC-2 will be ready to receive data when it receives either the CLEAR command or an external ABORT signal.

3.3.6 Clear the DC-2

After activating DC-2, write the CLEAR command (0x6) in the mailbox *command* word (vme_base+0x50). The DC-2 loads starting parameters from the DPM mailbox and does the necessary initializations.

The *dc2_status* word should contain 0x1j40nm, where j, n, and m are described above.

3.4 Spill to spill operation

3.4.1 Clearing the DDD

The DDD is configured as a simple linear buffer, where the write pointer never wraps around. Therefore, before every spill, the event buffer write pointer has to be reset and the event pointer table has to be cleared. There are three ways to clear the DPM.

1. Set *hold_off_clear* (vme_base+0x3C) flag in the mailbox area to a non-zero value, and send EXTERNAL_ABORT signal every spill. When EXT_ABORT signal is received by the DM-115, the DC-2 executes sets VETO line, which is wired OR together across the DDDs in the same plane and put in VETO OR for the first level triggers. The DC-2 also sets *cleared_flag* (vme_base+0x34) in the mailbox area. The event builder checks the *cleared_flag* after building all the events remaining in the (copied) pointer table (see section 3.4.3 on page 18), and if it is set, it executes CLEAR (0x06) command to all the DDDs. The CLEAR function turns off VETO signal, so that it can start receiving new events. The event builder should clear *cleared_flag* by itself. This method guarantees to read out all the events from the previous spill, which is necessary for experiments which require normalization of events by scaler rates, etc.
2. Clear *hold_off_clear* (vme_base+0x3C) flag in the mail box area, and give EXTERNAL_ABORT signal every spill. In this case, CLEAR command is executed immediately, and *cleared_flag* (vme_base+0x34) is set. The event builder has to check the *cleared_flag* before reading out next event(s), and if the flag is set, it should reset its read pointer and start reading from the beginning of DPM. This method is good for experiments which do not care about discarding remaining events from the previous spill upon the beginning of a next spill. This also requires a fast VME master like Motorola MVME 162 or 167, which can quickly access DPMs every event to check the *cleared_flag*.
3. Simply let the event builder issue CLEAR command every spill. In this case, the event builder has to get the spill timing from some other source, and it has to veto triggers while it is reading the remaining events. Such communications can be done by using semaphores within a real time machine, and using remote CAMAC.

When the CLEAR command is executed, either by command or by EXTERNAL_ABORT, the DC-2 reads the mailbox parameters listed in Table 3, Absolutely necessary parameters, on page 15, and reconfigures the memory space. This allows changing the DPM memory configuration during operation, if necessary.

If the DDD is not in the active state, the CLEAR and the ABORT will have no affect.

3.4.2 Send data to the DDD

Send data to the DM-115 through an RS485 connection from a DY3, FSCC, CTIRC, etc. Every event should be followed by EOR pulse⁸. During data collection, if the DC-2 cannot accept more data, it sends back the external signal WAIT to the data sources. The data sources should stop sending data while WAIT is asserted although the buffers are designed to take a small amount of overrun because there can be some latency in getting the data flow stopped.

3.4.3 Reading data out of the DPM

As the data accumulates in the DPM buffer, it can be processed by a local VME host or moved, over VME, to a remote host for processing. The data can be read a single event at a time by using the byte count at the beginning of each event. However, it is much faster to use the event pointer table and read out multiple events at a time. The recommended way to do this is to DMA a DPM memory block that includes the mailbox area (`vme_base+0x00` through `vme_base+0xff`) and the pointer table area (`vme_pointer_table_addr` through `vme_pointer_table+pointer_table_length`). Then, the host computer can use its local copy of the `n_events` word and the pointer table to determine how many events can be read with a single DMA. Pointer table entries point to the first word of the **next** event, so the difference between a pointer to the **next** event and the pointer of the current event (or the buffer starting address `vsb_buffer_addr` if no data has been read out) is the size of the event. This expands to multiple events so that the difference between the most recent pointer added to the pointer table and the pointer of the last event read (or the buffer starting address `vsb_buffer_addr` if no data has been read out) is the size of data available to be read out. When initiating a DMA read from a VME host, make sure to convert the VSB address given in the pointer table to a VME address by adding the offset (`vme_buffer_addr - vsb_buffer_addr`). There are several trade-offs in deciding how large a data block to DMA. The larger the block, the more efficient the transfer with regard to DMA overhead but if a very large buffer is moved, the access to the DPM from the VSB side for incoming data can be impacted. Depending on the processing software, some systems may want frequent small blocks of data while others may need larger blocks of event data at a time.

When `n_events` number of events have been read out, recopy the mailbox area and the event pointer table. In most cases, reading/processing events is slower than writing events to the event buffer space. If reading/processing is only half as fast as filling, at the end of processing `N` events, `2N` events have been written to the event buffer. While processing the `2N` events, `4N` events will have arrived, and so on. If there are `M` events to be read during the spill, then this operation can be characterized by equation 1:

(EQ 1),

⁸. 1. John Anderson and Ed Barsotti. Data, Permit & Trigger Link Interface Specification. June 5, 1995, Version 3.11, Document ESE-DART-950511, Draft. <http://www-ese.fnal.gov/eseproj/dart/ifspec3.txt>

where n is the number of times needed to copy the pointer table. Thus,

$$(EQ 2)n = \text{round_up} .$$

For example, if $N=1k$, and $M=50k$, n is only 6.

This method can reduce the accesses to DPM by reducing the DPM read/write time slicing, and speed up the event pointer table scanning because the table copy is available in the local memory of the VME host.

4 DC-2 Communication with the VME host

The DC-2 has no direct connection to the VME bus. A VME host communicates with the DC-2 by using the specially allocated mailbox space in the DPM. The DC-2 reads from and writes to the mailbox over the DPM VSB port, and the VME host reads from and writes to the mailbox over the DPM VME port.

The parameters in the mailbox area are listed in Table 5. The meaning of the words are explained in following sections. The location of each word is given as an offset from a base address. The base address is fixed at 0x20000000 in VSB space, and fixed at the beginning of each memory module in VME space. Entries in parenthesis in *who writes it* column indicate that they are optional parameters.

TABLE 5. DDD Mailbox

Offset from base address	Variable name	Bytes	Description	Who writes it
0x0	vme_read_pointer	4	read pointer in VME addressing	(VME)
0x4	vme_buffer_addr	4	starting VME address of data buffer	(VME)
0x8	vme_BAF_addr	4	VME address for BAF threshold	(VME)
0xC	vme_buffer_top_addr	4	top VME address of data buffer	(VME)
0x10	vsb_write_pointer	4	write pointer in VSB addressing	DC-2
0x14	vsb_buffer_addr	4	starting VSB address of data buffer	VME
0x18	vsb_BAF_addr	4	VSB address for BAF threshold	VME
0x1C	vsb_buffer_top_addr	4	top VSB address of data buffer	VME
0x20	n_events	4	the number of events in the buffer	DC-2
0x24	n_BAF	4	the number of BAFs generated	DC2,VME
0x28	n_timeout	4	the number of DMA time-outs	DC2,VME
0x2C	n_drain	4	the number of data draining	DC2,VME
0x30	DPM_mem_size	4	DPM size in bytes	(VME)
0x34	cleared_flag	4	indicate that DPM is cleared	DC2,VME
0x38	polling_period	4	command polling period	VME
0x3C	hold_off_clear	4	action to take on EXT ABORT	VME
0x40	heart_beat	4	DC-2 heart beat	DC-2

0x44	dc2_status	4	DC-2 status	DC-2
0x48	error_code	4	DC-2 error code	DC-2
0x4C	dc2_response	4	DC-2s response to a command	DC-2
0x50	command	4	command from VME host	VME,DC2
0x54-0x7C	arguments	44	argument list for command	VME
0x80	vsb_pointer_table_addr	4	point to beginning of pointer table in VSB space	VME
0x84	pointer_table_length	4	size in bytes	VME
0x88	vme_pointer_table_addr	4	point to beginning of pointer table in VME space	(VME)
0x8C			reserved for future use	
0x90	dm115_status	4	DM-115 CSR,SAW,LSR,ESR	DC2
0x94	user_bits	4	ORed with DC-2 header	VME
0x98	buffer_request	4	DC-2 Ping-Pong buffer request	DC2
0x9C	buffer_permit	4	VME Ping-Pong buffer permit	VME
0xA0	n_events_ping	4	number of ping events	VME
0xA4	wt_ptr_ping	4	last ping address	VME
0xA8	n_events_pong	4	number of pong events	VME
0xAC	wt_ptr_pong	4	last pong address	VME
0xB0	last_valid_addr	4		
0xB4	n_valid_events	4		
0xB8-0xFC			reserved for future use	

4.1 Buffer Addresses (offsets 0x00 - 0x1C)

The DC-2 must be told where the event buffer is located in DPM. Allocate the buffer after the event pointer table, which is specified by `vsb_pointer-table_addr` (offset 0x80) and `pointer_table_length` (0x84).

The first and last addresses of the event buffer space, and the BAF threshold addresses are stored both in VME and VSB addresses. The entries in VSB addresses (offsets 0x14, 0x18, 0x1C) are mandatory, but the VME addresses entries (offsets 0x04, 0x08, 0x0C) are optional. These addresses should be initialized by a memory configuration program that runs on the VME host prior to the first CLEAR command. The `vsb_buffer_addr` (offset 0x14) should be a multiple of four, to make the buffer long word aligned. The `vsb_top_addr` should be a valid VSB address near the top of available memory in the DPM, such as 0x21ffff0.

The BAF address must be larger than the buffer starting address, but lower than buffer top address. The difference between the BAF threshold and top address should be larger than the

size of the largest event plus the largest possible number of bytes that it could receive after issuing BAF. This is because the last event without BAF may end just below the threshold, and events in the front end buffer (including buffer in readout controllers) can still come in even after triggers are disabled by BAF.

vsb_write_pointer (offset 0x10) indicates where the DC-2 write pointer is, in VSB space, and gets updated periodically by the DC-2.

vme_read_poiner (offset 0x00) is an optional word which can be used by VME host to indicate how much of the events have been read out.

4.2 Statistics (offsets 0x20 - 0x2C)

These values are used for storing statistics, for checking for consistency between data streams, and for recording errors.

The *n_events* (offset 0x20) is the number of events stored in the event buffer. It is reset to 0 on CLEAR command, and updated every time the DC-2 checks for a command. The command polling period is determined by *polling_period* (0x38).

The other statistics locations, *n_BAF*, *n_timeout*, *n_drain* provide some event information. They are not cleared by the DC-2 but are incremented as the events occur. They should be cleared by the VME host usually at the beginning of a run or some other time chosen by the user.

n_BAF (offset 0x24) counts how many times BAF was turned on. Since it counts the number of transitions from off to on, the fastest it can increment is once per spill.

n_timeout (offset 0x28) counts how many times the DMA between DC-2 FIFO and DPM has timed out. The threshold for a timeout is between one and two times the command polling period specified in offset 0x38.

n_drain (offset 0x2C) counts how many times the data buffer or the pointer table has become full, and thus further incoming data were drained. It can be incremented once per spill.

4.3 DPM Memory Size (offset 0x30)

The DPM size in bytes. It was found that it was not efficient for the DC-2 to find the DPM size automatically, so this word is not used anymore.

4.4 cleared_flag (offset 0x34)

This flag will be set to a non 0 value by the DC-2 after it receives an external ABORT signal to allow recognition by the event builder on VME side. If the *hold_off_clear* (offset 0x3C) flag is zero, the DDD has been cleared by ABORT, so the event builder should start reading data from

the beginning of the buffer. If the `hold_off_clear` word is not zero, the DC-2DC-2 does not execute CLEAR by itself, but simply sets VETO. In this case, the event builder (gateway) has to send the CLEAR command to the DDD. The event builder should clear the `cleared_flag` word to zero as soon as it reads a non-zero flag word value.

4.5 polling_period (offset 0x38)

This word defines the time period T for polling commands in the DPM, and also, the DMA time-out limit. If there is no data arriving, the DC-2 will check for a command every period T. If data is arriving, the actual command polling period will vary between T and 2T. If the DMA from DC-2 to DPM lasts longer than 2T without detecting EOR, it will be flagged as a DMA time-out. If the DMA lasts for some time between T and 2T, it might or might not be flagged as DMA time-out, depending on the timing.

T in μ sec is defined by

$$T = Z * \text{mant} * 2^{**\text{exp}},$$

where:

Z : system clock period (0.119usec for now),
 mant : bit 0-15 of Polling Period word,
 exp : bit 16-19 of Polling Period word;
 0 <= exp <= 8 (note that it is not 15).

For example the following values produce these times,

0x4cd29 -> 0.119usec * 0xcd29 * 2**4 = 0.1sec,
 0x5ffff -> 0.119usec * 0xffff * 2**5 = 0.25sec,
 0x6ffff -> 0.119usec * 0xffff * 2**6 = 0.5sec and
 0x7ffff -> 0.119usec * 0xffff * 2**7 = 1.0sec.

4.6 hold_off_clear (offset 0x3C)

This mailbox word controls what actions the DC-2 takes when the EXTERNAL ABORT signal arrives. The choice depends on whether the user wants to clear or keep the remaining events when EXTERNAL ABORT arrives.

If the word is 0, then DC-2 executes the CLEAR command, and initializes the DDD for the new spill. Since it only clears `n_events` (offset 0x20) and the event pointer table, the events which were being DMAed when ABORT arrived are still valid and the DMA can run to completion.

If the word is not 0, then the DC-2 does NOT execute a CLEAR, but simply sets the VETO signal on the DM-115 (if the DC-2 is active), so that further triggers will be disabled. The VETO will be removed only when a CLEAR command is issued by the VME host, usually after reading

all the remaining events. This may veto triggers in the next spill, but is desirable for experiments who rely on event normalization with scalers or others.

4.7 heart_beat (offset 0x40)

The DC-2 periodically increments the heart_beat word so that the VME host can check this word to make sure that the DC-2 is functioning.

4.8 dc2_status (offset 0x44)

The dc2_status word contains internal status information about the DC-2, as listed below. This word will be updated by the DC-2 as the status changes or on a significant event.

TABLE 6. DC-2 Status bits

bit	contents
0-7	DM-115 address (See Set broadcast address section, page 16)
8-11	LEDs
12	mode; 1=case statement mode, 0=main_program mode
13	executing command
14	DC-2 active
15	BAF
16	Jumper #1 (1=open)
17	Jumper #2
18	Jumper #3
19	1=bug program, 0=ddd program
20	RS-485 drivers enabled (1=enabled)
21-22	PAR data buffers enabled bits
23-27	reserved for future
28	event pointer table overflow
29	draining data
30	DMA time-out
31	error (error code might be available in error code word in DPM)

4.9 error_code (offset 0x48)

If the DC-2 detects an error, it can set an error code in this word. The word will be cleared by the DC-2 when it receives a CLEAR command. Error codes are most informative during development of DC-2 firmware or problem diagnosis. A list of current error codes can be found in err.h in the DDD firmware development directory

4.10 dc2_response (offset 0x4C)

When DC-2 receives a command from VME, it uses the *dc2_response* word to show the stages of the command execution. The command byte is shifted over one byte and additional bits are masked in to provide additional information, as shown below.

TABLE 7. DC-2_Response

dc2_response (bit 0-15)	explanation
command<<8	acknowledged the command
command<<8 + n	started executing the command
command<<8 + 0xF0	finished the command

The n is a number to show steps in the command execution. The range for n is $0x10 \leq n \leq 0xEF$. Only a few commands take long enough to utilize this feature

The top 16 bits of the dc2_response word are zeros. In the future, they may be used to store additional information.

On boot-up, the DC-2 sets the dc2_response word to 0x00F0 to show that it has rebooted and entered casemode. This is slightly different from the 0xFDF0 response to an ENTER_CASEMODE command (0xFD) which will allow detection of errant reboots.

4.11 command and arguments (offsets 0x50, 0x54-0x7C)

Any command and its arguments from the VME host to the DC-2 will be written into the mailbox words command and arguments. The list of commands are shown in Table 8. The detailed explanations are given in reference pages shown in the table. The argument words should be written prior to the command, if arguments are required. The DC-2 checks the command word periodically, and if it is non-zero, it will execute the command. The command word will be cleared once the command and arguments are accepted by the DC-2 but before executing it, allowing the next command to be loaded while the first is executing. The DC-2 response to commands will be stored in the dc2_response word as described in the previous section.

TABLE 8. DC-2 Commands

op code	Command	Description	Ref .
0	NONE	no command	
1	LOAD_PROGRAM	Load program from DPM to DC-2	p. 30
2	JUMP_PROGRAM	Jump to a program in DC-2.	p. 30
3	CALL_PROGRAM	Call a DC-2 subroutine.	p. 30
4	ACTIVATE	Activate the DC-2/DM-115/DPM.	p. 31
5	DEACTIVATE	Deactivate DC-2/DM-115/DPM.	p. 31
6	CLEAR	Terminate DMA, and clear the DDD for the next spill.	p. 31
7	CLEAR_MEMORY	Clears data_buffer.	p. 31
8	SET_BROADCAST_ADDR	Set broadcasting group address.	p. 31
9	UPDATE	Update DC-2 status word and statistics.	p. 31
0xA	GET_VERSION	Get EPROM code version number.	p. 32
0xB	CLEAR_TABLE	Stop DMA and clear event pointer table.	p. 32
0xC	FAST_CLEAR	Similar to CLEAR, but the whole pointer table is not cleared.	p. 32
0x20	TEST_RAM	Write/read test on DC-2 RAM	p. 36
0x21	CLEAR_FIFO	Clears the input FIFO.	p. 36
0x22	WRITE_FIFO	Write test pattern to DC-2 FIFO	p. 37
0x23	READ_FIFO	Reads an non-empty FIFO into VSB location.	p. 37
0x24	TEST_DPM	Write/read test on DPM (random access)	p. 37
0x25	TEST_DMA	A stand-alone loop back test that runs in idle case statement mode.	p. 37
0x26	TEST_BAF	Drive BAF line.	p. 38
0x27	TEST_LED	Drive LEDs on DC-2	p. 38
0x28	EXIT_TEST	Terminate TEST_RAM/DPM/DMA tests.	p. 38
0x81	SAR Spill Algorithm Routine for Diagnostics	This algorithm simply sets up a DMA to write from vsb_buffer_addr, all the way up to vsb_top_buffer_addr. BAF is set at vsb_BAF_addr. No time-outs or drain mode is included here.	p. 39
0x82	CAR -- Circular Algorithm Routine	not implemented	p. 39
0x83	PAR -- Ping-Pong Algorithm Routine.	Write a predetermined number of events between buffer #0 and buffer #1.	p. 39

0x84	OAR -- Open-Loop Acquisition Routine for Diagnostics.	The DC-2 continuously writes between vsb_buffer_addr and vsb_top_buffer_addr. No BAF is issued. This is for open-loop scope tests.	p. 40
0xFD	ENTER_CASEMODE	Enter the case statement mode.	p. 33
0xFE	ENTER_MAINMODE	It exits case statement mode, returning to the beginning of the main program.	p. 33
0xEE	BUG_EXIT	Exit to Motorola's 340Bug program.	p. 33

4.12 Pointer Table address and length (offsets 0x80-0x84)

The *vsb_pointer_table_addr* word in the mailbox points to the event pointer table in VSB space. The value should be a multiple of 4 to make the table long word aligned. The *pointer_table_length* word contains the allocated length of the table in bytes.

Each element in the pointer table points to the data from an event in the event buffer space. Each pointer is 4 bytes long, and contains the VSB address of the first word of the *next* event. Therefore, one can calculate the size of the last event from the pointer table.

```
event_size = pointer_table(n_events) - pointer_table(n_events - 1)
```

If $n_events = 1$, *vsb_buffer_addr* should be used instead of `pointer_table(n_events - 1)`.

The size of all valid data in the event buffer space is:

```
event_size = pointer_table(n_events) - vsb_buffer_addr
```

The *pointer_table_addr* word will be updated every time an EOR is received. The pointer table will be completely cleared when the CLEAR command is given whether initiated by the VME host or by ABORT. A different command called FAST_CLEAR (0xC) is available which can speed up between spill processing because it will not clear the whole pointer table but only the first pointer. An aspect of error checking is lost however, because the DA software cannot check for zeroed pointers after the assumed last valid pointer.

4.13 dm115_status (offset 0x90)

The *dm115_status* word contains a copy of various DM-115 registers. The UPDATE command (0x9) refreshes this word.

TABLE 9. DM-115 Status bits

bit	contents
0-7	DM115_ESR

	0: External Reset 1: External Abort 2: Data Strobe 3: Address Strobe 4: EOR 5: Data Size 6: Wait 7: External Enable
8-15	DM115_LSR 8: Jumper #0 9: Jumper #1 10: Jumper #2 11: Decoder Address Select 12: FIFO Empty 13: Int0 14: Int1 15: Decoder Status
16-24	DM115_SWA 16:24 The positions of the switch
24-31	DM115_CSR 24: Spare 25: Test Mode 26: Test Data Size 27: Interrupt Control 28: Ext Enable 29: Decoder Enable 30: Full Enable 31: Buffer Almost Full

4.14 user_bits (offset 0x94)

As described in Section 2.2, the first word in every event contains an inclusive byte count of the event. The upper 2 bytes of the mailbox word *user_bits* will be ORed into this byte count and can be used to further identify the event data for later processing. Depending on the expected size of events, the user can use only a few or all sixteen of these bits. If the user bits in the byte count word, this *user_bits* location should be set to 0.

As an example, if all the expected event sizes can be comfortably fit into 20 bits (1 megabyte), the upper 12 bits can be used for a spill count or run number maintained by the VME host. However, be sure to mask this value carefully because any set bits below those 12 bits (for this example) will appear in and corrupt the event byte count in the lower 20 bits.

4.15 Ping-Pong information (offsets 0x98-0xAC)

These words are used by the ping-pong writing algorithm. The details of the ping-pong mode are described in the command section for the PAR command on page 39.

5 DC-2 Commands for data acquisition.

Commands should be given to the DC-2 in following steps.

1. Wait until command word (offset 0x50) in the mailbox is 0.
2. Write any necessary arguments in the arguments words (offset 0x54-0x7C) in the mailbox.
3. Write the DC-2 Command in command (offset 0x50) word.
4. When the command word is cleared, the arguments and the command were accepted by DC-2, and a new command may be entered.
5. When the lowest two bytes of dc2_response (offset 0x4C) word becomes the command byte shifted left by 8 bits plus 0xF0, it means that the execution has finished.

The command and dc2_response words should not be polled heavily so that the DC-2 may be able to access the DPM. Give them a break.

In the following sections, commands for data acquisition are explained. The number in front of each command name is the op code byte to be entered into the command word (offset 0x50). The arguments words will be abbreviated to arg[n] in the C style code below.

0x1: LOAD_PROGRAM

arg[0]: location of S-format code in DPM, in VSB space.

This command loads a test program from DPM to DC-2. The program should be written to the DPM in Motorola HeX Format, known as S-format, prior to the execution of this command. The load address is included in S-format. The loaded program can be executed either by JUMP_PROGRAM or CALL_PROGRAM command.

0x2: JUMP_PROGRAM

arg[0]: DC-2 program address to jump to.

The DC-2 processor jumps to the DC-2 address specified by arg[0] and begins execution there. The program can be a monitor, a test routine or a different idle loop. The control will not return to the DMA loop or casemode loop after the execution.

0x3: CALL_PROGRAM

arg[0]: DC-2 subroutine address to call
arg[1]: timer value to quit the subroutine. See 4.5 for the format.
arg[2]: Action after the timer expires;
 0: go back to Motorola 340BUG monitor
 1: go back to the caller

The DC-2 processor calls a subroutine specified by `arg[0]`. When the subroutine finishes normally, it returns to the original mode, DMA loop or casemode.

If `arg[1]` is 0, there will be no timeout checking. If it is a non-0 value, written in the same format as `polling_period`, then a timer will be started prior to calling the subroutine. If the DC-2 has not returned from the subroutine when the timer expired, the DC-2 is forced to go to the 340Bug monitor (`arg[2]=0`), or the original calling routine (`arg[2]=1`) which is either the DMA loop or casemode.

0x4: ACTIVATE

This command activates the DDD. BAF will be set until a CLEAR command is given, but the DM-115 is enabled, and the DC-2 has control over data flow. Since the DC-2 is in deactivated state after reset and after exiting from casemode, this command is required prior to data acquisition.

0x5: DEACTIVATE

Deactivate the DDD. It terminates any DMA, and disables the DM-115 so that the DDD will have no control over data flow. BAF will be cleared so that it will not affect the trigger logic.

Since V2.2, this is done by switching into test mode from data input mode on the DM-115.

0x6: CLEAR

Initializes the DDD for the new spill. It terminates any DMA, clears the FIFO and resets the event count. It reloads the polling period, user bits, buffer and pointer table parameters from DPM. It also clears the pointer table. It clears BAF and VETO signal lines.

CLEAR does not change the active/non-active status, so that user can send CLEAR commands to all the DDDs, independent of their state. Also be aware that a CLEAR command while in CASEMODE will be ignored.

0x7: CLEAR_MEMORY

This command clears the DPM from the beginning of the pointer table to the end of data buffer. Executing this command after turning on the VME subrack power is a good practice, since it sets correct parity bits for each address.

0x8: SET_BROADCAST_ADDR

`arg[0]`: 4bit broadcast address

Set broadcasting group address. If the DM-115 is set to recognize address bits in the first word of every event, and if the logical OR between bits 16-19 in the first word and the

broadcast address bits 0-4 in `arg[0]`, the event will be accepted by the DDD. This scheme is used to send the same event to multiple planes.

Note that this command only sets the broadcast address, and not the plane address which is in bits 12-15 in the first data word. The plane address should be set by a dip switch S3 on the DM-115. (See section 3.2.3 on page 13.)

0x9: UPDATE

Update `dc2_status` and `dm115_status` words.

0xA: GET_VERSION

Get EPROM code version number.

Output:

`dc2_response` (offset 0x4C) contains the version number in lower 2 bytes, V3.00 = 0x0300.
Development version will have d at head, like 0xd301.

0xB: CLEAR_TABLE

Clears pointer table.

0xC: FAST_CLEAR

Similar to `CLEAR`, but the pointer table is not completely cleared only the first entry. This command is a little quicker than a `CLEAR`.

6 Diagnostics commands in casemode

Various stand alone diagnostic tests can be done by sending a command from VME host to DDD. The tests include testing of the DC-2 RAM, DPM, FIFO, DMA operation from the DC-2 FIFO to DPM, and various external lines.

Most of the diagnostics should be executed in CASEMODE, which can be entered by executing the ENTER_CASEMODE command (0xFD). This ensures that it will not start receiving data from RS485. The user can completely disable data on RS485 by inserting a jumper on pin 1-2 (rightmost pins) on J9 on the DM-115.

6.1 Entering and Exiting the Casemode

After power on or reset, the DC-2 enters the casemode, and this is indicated by 0x00F0 in dc2_response word.

0xFE ENTER_MAINMODE

This command switches the DC-2 from casemode to DMA_LOOP for data acquisition. In the past this command was called EXIT_CASEMODE but that did not clearly indicate what was being entered. The EXIT_CASEMODE label still works and is equivalent to ENTER_MAINMODE.

0xFD ENTER_CASEMODE

This command switches the DC-2 from DMA_LOOP to casemode.

0xEE BUG_EXIT

This command forces the DC-2 to exit to Motorola's 340BUG program, for debugging or loading programs through the RS232 line. It changes the dc2_status bug bit, stops incrementing the heart_beat, and sets 0xEE00 in dc2_response. The DC-2 must be in casemode before issuing this command.

6.2 Testing of DC-2 hardware

The DC-2 can test its own RAM, DPM, FIFO, and DMA operation as shown in FIGURE 3.

FIGURE 3. Diagnostics functions

Command TEST_RAM writes data with a certain pattern to the DC-2 RAM, read them back and compares them with what it should be.

TEST_DPM writes/read data to/from DPM by random access.

TEST_DMA writes data directly to the DC-2 FIFO, initiate DMA to transfer data from the FIFO to DPM, and read data back from DPM for comparison.

The DC-2 FIFO/DMA can be tested step by step if necessary, via CLEAR_FIFO, WRITE_FIFO, and READ_FIFO commands.

6.2.1 Test patterns

Various test patterns are provided for the above tests, as summarized in Table 10.

TABLE 10. Test patterns for diagnostics

Test pattern code	The pattern to be written to each location
0	All patterns: pattern code 1,2,3,4(only for TEST_RAM/DPM), and 5
1	running 1s; i.e. 0x00000001, 0x00000002, 0x00000004, 0x00000008, 0x00000010, etc.
2	running 0s, i.e. 0xFFFFFFFF, 0xFFFFFFF, 0xFFFFFFF, 0xFFFFFFF7, 0xFFFFFFF, etc.
3	Its address (written from low address to high address)
4	Its address (written from high address to low address). Means same as 3, for WRITE_FIFO and TEST_DMA.
5	Checker board pattern; 0x5555 5555 / 0xAAAA AAAA

6.2.2 Repeat count

The memory tests TEST_RAM, TEST_DPM, and TEST_DMA can be repeated multiple times. If 0 is given as the number of time to repeat, then the test will be repeated forever, until a command EXIT_TEST (0x28) is given.

6.2.3 dc2_response

TEST_RAM, TEST_DPM, and TEST_DMA sets the *dc2_response* word (offset 0x4C). as summarized in Table 11.

TABLE 11. dc2_response for TEST_RAM/DPM/DMA

dc2_response bits	explanation
bits 0-3	Remainder scale. Let the number of tests to repeat be M. If M=0, these 4 bits stay 0. If 0<M<16, then the remaining count is indicated in the 4 bits. If M >=16, and if the number of remaining count is N, then 16 * N / M is returned in these 4 bits.
bits 4-7	1: During writing memory or FIFO 2: During DMA from FIFO to DPM (TEST_DMA) 3: During reading memory F: Finished the test
bits 8-15	Command op code
bits 16-31	The number of errors found so far. The number is updated at the end of each scan. If the number of errors becomes larger than 0xffff, the lowest 16 bits of it will be shown during the test, and then set to 0xffff upon the completion of the test.

6.2.4 Error histograms

The results of TEST_RAM, TEST_DPM, and TEST_DMA is summarized in four histograms, as shown in Table 12. Each histogram bin is 32 bits long. By studying the distribution in the histograms, one should be able to narrow down the problem, whether it is a data bit problem or addressing problem, etc.

TABLE 12. Error histograms for TEST_RAM/DPM/DMA

long word offset	explanation
0 + i	The number of times when bit i of data was 0, where it should have been 1.
32 + i	The number of times when bit i of data was 1, where it should have been 0.
64 + i	The number of times the bit i of the address was 0, when an error was detected.
96 + i	The number of times the bit i of the address was 1, when an error was detected.

6.2.5 Commands for testing RAM/DPM/FIFO and DMA

The list of commands for hardware diagnostics are explained next.

0x20 TEST_RAM

arg[0]: low address limit of the DC-2 RAM range to test
arg[1]: upper address limit of the DC-2 RAM range to test
arg[2]: Test pattern code (See Table 10)
arg[3]: The number of times to repeat the test (0=infinite)

Output:

dc2_response:	See Table 11.
data_buffer:	Error histograms, see Table 12
vsb_write_pointer:	address of the RAM which is being read.

(updated periodically)

This command writes a data pattern specified by the arg[2] to the DC-2 RAM address range specified by arg[0] and arg[1]. The data will be read back from the RAM, and checked against the original pattern. If there is a mismatch, the number of errors will be incremented, and the error pattern and address will be logged in the error histograms. The error histograms start at the beginning of data_buffer, whose address is given by vsb_buffer_addr (0x14). The write/read cycle is repeated for the number given in arg[3].

Since VSB is part of DC-2 addressing space, this could be used to test the DPM. However, in order to avoid conflict between the testing area and the histogram area, it is strongly recommended to use TEST_DPM instead.

0x21 CLEAR_FIFO

This command clears the DC-2 FIFO, which is useful in the diagnostics mode. For example, it is wise to clear FIFO before executing TEST_DMA or WRITE_FIFO fresh. In DMA_LOOP, the FIFO is cleared by a CLEAR command as a part of the function.

0x22 WRITE_FIFO

arg[0]: The number of long words to write.
The maximum number allowed is 0x2000 (8k).
arg[1]: Test pattern code (See Table 10).

This command writes a test pattern to the DC-2 FIFO. The data will stay in the FIFO until a READ_FIFO or CLEAR_FIFO command is issued.

This is also useful for testing the WAIT line. See section 6.3 on page 38 for details.

0x23 READ_FIFO

arg[0]: VSB destination address

This command will DMA existing data in the DC-2 FIFO to DPM whose destination address is given by arg[0]. After the transfer, the FIFO will become empty. There is no checking done for the transferred data.

0x24 TEST_DPM

arg[0]: DPM starting address of the memory range to be tested.

arg[1]: DPM ending address of the memory range to be tested.

arg[2]: Test pattern code (see Table 10).

arg[3]: The number of times to repeat the test (0=infinite).

arg[4]: VSB address for error histograms in DPM.

Output:

```
dc2_response:                See Table 11.
DPM starting at VSB address arg[4]:                Error
histograms, see Table 12
vsb_write_pointer:          address of the RAM which is
being read.
                            (updated periodically)
```

This command writes a data pattern specified by the arg[2] to the VSB address range of the DPM specified by arg[0] and arg[1]. The data will be read back from the DPM, and checked against the original pattern. If there is a mismatch, the number of errors will be incremented, and the error pattern and address will be logged in the error histograms. The location of the error histograms is specified by arg[4]. There should be no overlap between the testing memory range and the area for the error histograms which is 512bytes (0x200) long. The write/read cycle is repeated for the number of times given in arg[3].

0x25 TEST_DMA

arg[0]: VSB destination address where data is written.

arg[1]: The number of 4byte data words to DMA for each transfer.

The maximum size is 0x2000 (8k long words).

arg[2]: Test pattern code (see Table 10).

arg[3]: The number of times to repeat the test (0=infinite).

arg[4]: VSB address for error histograms in DPM.

Output:

```
dc2_response:                See Table 11.
DPM starting at VSB address arg[4]:                Error
histograms, see Table 12
```

This command first fills the DC-2 FIFO with a data pattern specified by the arg[2], and then DMA's data to the DPM at the starting address given by arg[0]. The length of the data pattern is given by arg[1] in long words. The data is read back from DPM by random access, and compared against the expected pattern. If there is a mismatch, the number of errors will be

incremented, and the error pattern and address will be logged in the error histograms. The location of the error histogram is specified by arg[4].

The data words are terminated by an EOR signal generated by the DC-2, which is pushed into the FIFO. When the DC-2 reads back data, it also checks that the word after all the data words is zero, which is caused by the EOR. The VSB destination address should be specified such that the EOR word should still fall within the physical memory boundary. If the word after the EOR word is still below vsb_buffer_top_addr, then the DC-2 writes a special data pattern in the word prior to DMA, so that it can detect an error if DMA did not terminate correctly.

There should be no overlap between (the VSB destination address...VSB destination address + #long words * 4) and the error histogram area which is 512 bytes (0x200) long. The write/read cycle is repeated for the number given in arg[3]. When test is repeated for multiple times, the data is always transferred to the same address range.

0x28 EXIT_TEST

This command terminates TEST_RAM, TEST_DPM, and TEST_DMA tests if they are being repeated for multiple times. The test will not be terminated until the end of current write/read cycle, so be patient.

6.3 Testing External Lines and LEDs

BAF line, WAIT line, and LEDs on the DM-115 can be tested by the following commands.

0x26 TEST_BAF

```
arg[0]:      0: clear BAF
             1: set BAF
```

This command drives the BAF line (pin 7,8 on the 10pin connector).

0x27 TEST_LED

```
arg[0]: The lower 8 bits set the DM115_LEDCR to turn on/off LEDs.
```

This command turns on/off LEDs on the DM-115. The state of the LEDs can be checked by reading the mailbox dc2_status word or by observing the LEDs.

Testing the WAIT line (pin 37,38 on the wide RS485)

In order to test the WAIT line, execute the following steps.

1. Enter casemode by ENTER_CASEMODE (0xFD) command.

2. Clear FIFO by CLEAR_FIFO (0x21) command, and make sure that WAIT line is off.
3. Write 4096 long words (0x1000) into FIFO by WRITE_FIFO (0x22) command. The WAIT line should still be off.
4. Write 1 more long word into FIFO by WRITE_FIFO (0x22) command. The WAIT line should come on because the FIFO is over half full.
5. Flush FIFO by READ_FIFO (0x23) command, and the WAIT line should go away.

Testing VETO line (pin 9,10 on the ten pin connector)

In order to test the VETO line on the ten pin connector, do the following.

1. Clear the *polling_period* word (offset 0x38) in DPM.
2. Enter DDD_LOOP by executing an ENTER_MAINMODE (0xFE) command.
3. Execute an ACTIVATE (0x04) command followed by a CLEAR (0x06) command.
4. Set *hold_off_clear* flag (offset 0x3c) to a non-zero value.
5. Apply an EXTERNAL_ABORT signal (wider than twice the polling period which is defaulted to 0.25sec).
6. The VETO line should come on.
7. Send a CLEAR command (0x06). The VETO should go away.

6.4 Other Diagnostics Modes

There are some other diagnostics routines which were developed by Oscar Trevizo.

0x81 SAR (Spill Algorithm Routine for diagnostics)

This algorithm simply sets up a DMA to write from vsb_buffer_addr, all the way up to vsb_top_buffer_addr. BAF is set at vsb_BAF_addr. No time-outs or drain mode is included here.

0x82 CAR (Circular Algorithm Routine)

Not implemented.

0x83 PAR (Ping-Pong Algorithm Routine)

This algorithm is in use at one experiment and is now considered a second data acquisition mode for the DC-2. Write a predetermined number of events between buffer #0 and buffer

#1. It coordinates with VME using two words: request/permit. The VSB side requests the buffer number. If the permit is equal to the request, the VSB side writes the predetermined number of events into that buffer and then toggles the request. If the permit does not respond to the request, then the VSB side sets BAF. It will write into the buffer as soon as the permit is granted.

1. Before running; Initializing the mailbox:

(a) Buffer #0:

vsb_buffer_addr: beginning data address buffer #0

vsb_BAF_addr, and vsb_to_buffer_addr not used.

(b) Buffer #1

arg[0]: beginning data address for buffer #1

arg[1] and arg[2] are intended for BAF#1 and top #1, but are not required in this algorithm.

(c) Number of Events Per Buffer:

arg[3]: Number of events per buffer.

2. Running; Issue op-code 0x83

(a) The user issues a PAR command, 0x83, when the DC-2 is in idle loop/case statement mode.

(b) The DC-2 responds as follows:

dc2_response = 0x00083F0

The ÔF0 lets the user know that it is in the process of taking data.

(c) The DC-2 writes into the enabled buffer (#0 automatically the first time around).

(d) The on the other buffer permit is polled until it is available. Then it is written into.

(e) The users application enables the data buffers as they are freed up by data acquisition or processing tasks.

3. Exiting the Ping-Pong Algorithm:

Issue command ENTER_CASEMODE, op-code 0xFD, for entering case statement mode.

0x84 OAR (Open-Loop Acquisition Routine for diagnostics)

The DC-2 continuously writes between vsb_buffer_addr and vsb_top_buffer_addr. No BAF is issued. This is for open-loop scope tests.

7 The DDD software

7.1 Overview

The DC-2 MC68340 software resides in on-board EPROM. The EPROMs contents are copied into DC-2 RAM at boot-up for normal execution. Additional software which is not in EPROM can be loaded into DC-2 RAM via the DPM or through RS232 line to DC-2. This loading mechanism is useful for DC-2 code development, running non-standard DC-2 code, and for running special diagnostics.

There are four kinds of software that run on the DC-2:

1. boot code;
2. DMA loop capable of taking commands, but favors a DMA spill-oriented data acquisition (daq) mode;
3. code for communication between the DC-2 and VME host via the DPM;
4. case statement code for diagnostics.

This software is described in the following sections.

7.2 View from the DC-2

FIGURE 4. shows the address mapping of DC-2 ROM, RAM, and DPM viewed from DC-2. The first 64kbyte area is used for 68340 specific vectors and RAM. The monitor, 340BUG resides in EPROM starting at address 0x60000. The DDD software is also in EPROM, starting at address 0x70000. The address after 0x20000000 is mapped to VSB memory space, and thus DPM can be accessed directly. The first 0x100 bytes of DPM are used for mailbox area, followed by a pointer table, and event buffer.

FIGURE 4. The world of the MC68340

7.3 Boot-up

When power is applied to the DC-2 or when the DC-2 receives a hardware RESET, it first runs boot up code contained in EPROM. The boot up code will do the following initialization, which is described in more detail in the DC-2 VSB Input Controller manual⁹.

1. Initialize the DC-2 hardware.
2. Initialize the DM-115 hardware.
3. Enter casemode and initialize timer for polling for commands. Wait for a command.
4. Additional initialization is done upon receiving an ENTER_MAINMODE command including preparing for DMA and storing starting parameters from the mailbox

7.4 Main Program - DMA Loop

The DMA loop is the core of the DC-2 software for data acquisition. The DMA loop software performs the following functions:

- ¥ Transfer data received by the DM-115 to DPM as soon as data arrives.
- ¥ At the end of each event, write the events byte count and address to the DPM.
- ¥ Check for a command from the VME host in the DPM, and execute the corresponding function if the command is valid.
- ¥ On receipt of an external ABORT signal, execute the CLEAR command to initialize the DPM, or simply set VETO signal on the DM-115. (See 3.4.1 for more details).
- ¥ Detect a DMA time-out and exit from a DMA wait loop.

After boot up, the DC-2 enters case statement mode idle loop, described in section 6 on page 33. The user should enter the DMA loop from the casemode idle loop by issuing the DC-2 ENTER_MAINMODE (0xFE) command.

In this section, the operation of the DMA loop is described in text, followed by a C-like pseudo code.

⁹. 1. Access Dynamics, Inc., DC-2 VSB Input Controller User Manual, Document M1002A, 1993. Available in hard copy only.

7.4.1 DMA Loop operation

After boot initializations and entering the MAINMODE, the DC-2 does further initializations, including setting the polling period for commands, reading and setting data buffer and pointer table addresses from DPM, and setting up parameters for the DMA controller. It then goes into a DMA loop.

The core of DMA loop is an idle loop waiting for data to arrive and it is optimized to give the maximum data throughput. The DMA controller in the Motorola 68340 processor on the DC-2 is set up so that data flow from the DC-2s data FIFO to DPM will start automatically as soon as data comes into the FIFO. The end of a DMA is triggered by the End Of Record (EOR) signal falling out of the FIFO. The DC-2 then writes the event size into the first word of the event buffer. This word was initially left empty by the DC-2¹⁰. Then the address of the next event is written into the event pointer table in DPM. Next the number of events in DC-2s local memory is incremented, but not the value stored in DPM. That location is updated the next time the command word is polled. The DC-2 then sets up the DMA controller again¹¹ and waits for a new event.

There are other tasks that the DMA loop has to perform besides handling the data events. One is to poll for commands from the VME host and check for the EXTERNAL ABORT signal from the DM-115 periodically, and the other is to detect a DMA timeout. To execute these tasks with a minimum overhead, a timer interrupt is used. The period of the timer is set by the `polling_period` word in DPM, and is set during execution of a CLEAR command. The period should be set to be comfortably longer than the time it takes to DMA one event worth of data.

The timer will be running while the DMA loop is waiting for an EOR. When the timer expires, an interrupt service routine called `ISR_DMA` is entered. The basic logic flow is as follows. If there is no data coming into the DDD, then it quits waiting for EOR, and executes the code to check for a command. If it is in the middle of a DMA, or DMA has just terminated (which will

¹⁰. 1. The very first word in the data buffer is cleared by CLEAR command, or by EXTERNAL_ABORT if `hold_off_clear` flag is not set. The data from the front end will be written right after this blank word. The blank word is filled with the inclusive byte count of the event after DMA is over. The event size word for the following events are allocated in the same way. The end of DMA is signaled by the End of Record signal. In the DC-2 FIFO, the EOR is treated as the 33rd bit, and the other 32 data bits are set to zero. When the EOR word falls out of the FIFO, the DMA will be terminated but this empty word will also be written into the DPM as a data word. This blank data word will be used to store the event size for the next event.

¹¹. 1. The DMA destination address (DAR) and byte transfer counts (BTC) are changed automatically by the DMA controller as data comes in. Therefore there is no need to modify them until they are initialized by CLEAR command for a new spill.

be the case for a high data rate¹²), then it sets a flag and returns to the waiting loop so that it does not slow down this operation. Within the DMA_LOOP, the flag is checked during post event processing, and if the flag is set, the check for a command code is run. If the DMA takes longer than the polling_period, DMA_ISR will be invoked for the second time. In this case, the flag has been set already, so the DC-2 terminates DMA, set timeout error, increment the timeout count, and goes off to check for a command. The flag is cleared when command is polled. When the timeout occurs, the following event will simply concatenated to the time-out event, and thus look longer than it should be.

With the above mechanism, the command word in DPM is checked periodically, which is no longer than the twice of polling_period. Prior to reading the command word, it updates write pointer, the number of events, and heart beat words on DPM. If a command is found, the DC-2 executes the command. In most cases, the control is returned from the corresponding command function. In addition, the EXTERNAL_ABORT signal on the DM-115 is checked, and if it is on, a special action will be taken. Otherwise, it will set up the DMA controller and wait for EOR again.

EXTERNAL_ABORT can be used to initialize DDDs prior to the next spill. It can either work exactly as same as executing CLEAR command, or can set VETO signal on the DM-115. For further details, read section 3.4.1 on page 17.

After each event transfer, several things are checked. First, if the write pointer has passed the BAF (buffer almost full) threshold, then it sets the BAF line on the DM-115. The BAF signals can be wired ORed together across the plane, and can be used to veto triggers. However, the DDD is still able to receive incoming data. Next, if the write pointer reaches the end of data buffer or if the pointer table gets full, the DC-2 goes into a Drain mode. In this mode, BAF will be set, and incoming data will still be received but will never be written to DPM or anywhere. This helps to flush front end buffers before the next spill arrives. A bit in dc2_status word in DPM will indicate that draining is occurring, and also an internal counter will be incremented to allow later error recovery and reporting. The DC-2 can recover from drain mode with a CLEAR command or by EXTERNAL_ABORT signal (if hold_off_clear flag is not set). Other commands can still be executed as normal because DMA_ISR will be still invoked periodically.

¹². 1. When data is coming into the DDD at a high rate, the data will be filling FIFO while the DC-2 is doing a post event processing for the previous event. As soon as the DMA controller is set up, DMA starts immediately with the new data. During a heavy DMA, the CPU does not get much time to execute instructions or the interrupt service routine. Thus it is after DMA is finished that DMA_ISR can be invoked. In this case, DMA_LOOP has not gone far enough to check for EOR, and the ISR is the first to catch it. Therefore within the ISR, the D-2 sets a flag and immediately returns to DMA_LOOP for the post-event processing.

7.4.2 DMA Loop pseudo code

The actual code is written in Motorola 68k assembly language, but here is the algorithm written in C-like code for better understanding.

```
( ) indicates variable in the DPM.
< > indicates variables or registers in DC-2.

dddmain()
{
    many initializations;
    call case_mode();
    more initializations;

POLL_CMD:
    clear <flag>; /* /* POLL_CMD:*/
    update (vsb_write_pointer);
    update (n_events);
    increment (heart_beat);
    if( command exists ) execute command;
    If( external_abort) goto EXTERNAL_ABORT;

    if( pointer table is not full && buffer not full ){
        while( !flag ){ /* JUDGE_TIMEOUT */
            start timer;
            start DMA (auto start);

            do{
            }while( DMA not done );

            stop timer;
            write event size and fill pointer table;
            <n_events>++;

            if( pointer table is full ){
                set table overflow flag;
                goto ENTER_DRAIN_MODE;
            }
            if( <write_pointer> > BAF thr ){
                set_baf();
                if( <BTC> <= 4 )
                    goto ENTER_DRAIN_MODE;
            }
        } /* end while */
        goto POLL_CMD;

    }else{ /* drain mode */
ENTER_DRAIN_MODE:
        set_baf();
        if( drain bit not set ){
            set drain bit;
            (n_drain)++;
        }
    }
}
```

```

    }
    start timer;
    do
        clear fifo;
        while( !ext_abort );

EXTERNAL_ABORT:
    if( !hold_off_clear ){
        stop timer;
        clear(); /* same as executing CLEAR command */
    }else if( DDD active ){
        set veto;
    }
    set (cleared_flag);
    goto EXIT_POLL_CMD;
}

}

set_baf()
{
    if( BAF not set ){
        set BAF;
        (n_BAF)++;
    }
}

timer_isr()
{
    if( DMA done ){
        set <flag>;
        return;
    }
    if( DMA has started (BTC has changed) && !flag ){
        set <flag>;
        return;
    }else{
        stop DMA;
        if( <flag> ){
            (n_timeout)++;
            set timeout bit in (dc2_status);
            goto POLL_CMD;
        }else if( DMA has started right before terminating DMA){
            restart DMA;
            return;
        }
        stop timer;
        goto POLL_CMD;
    }
}

clear()
{

```

```
clear (command);
stop DMA;
clear FIFO;
clear pointer table;
get (vsb_buffer_addr);
reset (vsb_write_pointer);
clear first word in buffer;
get (vsb_BAF_addr);
get (pointer_table_addr);
get (pointer_table_length);
clear <n_events>;
clear (n_events);
set timer value;
setup DMA; /* reset BTC and DAR */
clear BAF;
clear VETO;
}
```

Appendix A: DDD Versions - Release Notes

A.1 New Features in V2.2

Version 2.2 has an expanded mailbox. The new mailbox includes variables for draining statistics and for the event pointer table. Some of the mailbox locations changed.

The previous version, v1.0, did not have a pointer table.

Version 2.3 is a superset of v2.2.

A.2 New Features in V2.3

A.2.1 Idle loop/Case statement mode for Commands

DDD V2.3 features the same DMA loop from v1.0 through v2.2, and in addition, it has a new idle loop mode; the case statement mode. Inside the case statement mode, the user will be able to select a function, apart from the data taking mode featured in the DDD software.

In order to increase bandwidth, the DDD software is optimized to enter a dma loop. This may affect the performance of diagnostic functions. Also, a user may want to develop a special program such as using the DPMs as circular buffers. The case statement mode allows the user to run special functions. It is a simple idle loop that waits for a command. The command is executed, and the program returns to the idle loop. The user has the option to returning to the beginning of the main program.

A.2.2 dc2_status Expansion

A.2.2.1 Jumper Bits

The dc2_status mailbox location also contains bits that inform the user about the jumper configuration in the DM-115.

bits 16,17,18 are Jumper 1,2,and 3 respectively (Open = 1)

A.2.2.2 RS-485 Driver Enable Bit

One can read the enable status of the RS-485 drivers. This is bit 20, enabled =1.

A.2.2.3 dm115_status Word

The mailbox also has a dm115_status word that contains copies of the DM115_CSR, DM115_SWA, and DM115_LSR registers. These registers are documented in the DM-115 document from Access Dynamics. The dm115_status word contains:

```
bit00-07 DM115_LSR
bit08-15 DM115_SWA
bit16-23 DM115_CSR
```

The bits from each register are untouched. The user can update them sending the UPDATE command.

A.2.3 Diagnostics Expansion

A.2.3.1 Loop-Back During Main Program

The user can write to the FIFO, while in data acquisition mode. This way, one can diagnose the data acquisition algorithm. Write the proper arguments, and run DC-2 command 0x22. One will see the n_events count increase, the vsb_write_pointer increase, and will be able to read the new data in the dual ported memory. This is a loop-back test.

A.2.3.2 Test DMA Loop-Back During Case statement mode

Command (0x25), TEST-DMA has run for several days showing no errors. This is a stand-alone loopback test that runs in case statement mode.

A.2.3.3 READ_FIFO

Command (0x23) will read the fifo into a specified location. This function can run in the DMA loop or case statement mode. The proper way for running this function is from the case statement mode after writing to the FIFO.

A.3 New Features in V2.4

Version 2.4 features a VSB mapping expansion to 512 MBytes, user bits for header option, a ping-pong algorithm, algorithms for diagnostics, a boot up entrance to the idle loop/case statement mode, and a bug correction in the `isr_dma`. The DMA loop algorithm, from the functional point of view, along with the `isr_dma`, remains unchanged. The sequence of commands for running the main program remains unchanged.

A.3.1 VSB Map Expanded to 512 MBytes

Two register writes enabled the MC68340 in the DC-2 to map 512 MBytes of VSB space. This map size requires the VSB base address to change to 0x2000 0000, a 512 MByte boundary.

A.3.2 Routine `isr_dma` Change

The `isr_dma` performs the same tasks from previous versions. The DMA however, is only stopped after the timer has one time-out. This fixes a bug detected in v2.3.

A.3.3 Boot-Up Entry to Idle loop/Case statement mode

At boot up, the DC-2 goes into an idle loop waiting for further commands. This gives the user the chance to configure the mailbox before letting the DC-2 run any functions. This change is invisible to the user. The user can still send an `ACTIVATE` command followed by a `CLEAR` command to take data in the DMA loop.

A.3.4 User-Bits

On every `CLEAR`, the DC-2 reads the `user_bits` location in the mailbox. See the `user_bits` offset location in the mailbox described in the proceeding chapters. These bits are ORed with the DC-2 header.

A.3.5 Ping-Pong Algorithm

A ping-pong algorithm has enabled the DDD to reach 20 MBytes/sec for 2KByte events, during simultaneous VME reads and VSB writes to different DPMs. The VME side reads one DPM while the VSB side writes to the other DPM. The VME and VSB processors coordinate the read/writes by using a request/permit pair of words in the mailbox. The VSB side submits a request to write into DPM 0 or 1, and the VME side permits the VSB to write into either DPM 0 or 1. The VSB side toggles the request line after writing a predetermined number of events into an event buffer.

A.3.6 Diagnostic Algorithms

First, an open-loop scope command lets the DC-2 DMA write into the event buffer space continuously as a circular memory without any VME/VSB coordination. This routine allows the hardware tests to measure data-flow in a continuous mode. Second, a spill algorithm lets the user write into the buffer without timer interrupts and drain routines. These two routines are included for diagnostic purposes.

A.4 v2.5

This version incorporated a change to the ping-pong algorithm, which was modified to write the mailbox with the number of ping events, the last ping address, the number of pong events, and the last pong address. Also, bit 28 in the DC-2 will be set to 1 if the event table entries overflow. This version was for evaluation purposes only, and was not released officially.

A.5 v2.6

This version incorporated Taku Yamanaka's recommended revisions to the ISR_DMA routine. This version was for evaluation purposes only, was not released officially, but retained the v2.5 changes.

A.6 v2.7

This version incorporated a FAST_CLEAR command, similar to the CLEAR command, except that the pointer table is not cleared. The number of events and the write pointer are also copied into the mailbox. This allows fast switching between memory banks in the DPM. A bug in the LOAD_PROGRAM routine was corrected. This version retained v2.5 and v2.6 changes. Version 2.7 is the sequential replacement for the v2.4 microcode.

A.7 v2.8

A flag, hold_off_clear was introduced not to execute CLEAR on EXT_ABORT, but to set VETO line. This version was for used for evaluation.

More importantly, this version had a fix in DMA_LOOP and ISR_DMA, so that it can update n_events and the pointer table at a regular rate even at a high data input rate.

A.8 v3.0

This version incorporated the changes in V2.8 plus various bug fixes and restructuring of codes.

A.8.1 Diagnostics

1. TEST_DPM command is supported. Before, it was just an alias of TEST_RAM.
2. Error histograms are supported. When TEST_RAM, TEST_DPM, and TEST_DMA are executed, it creates histograms which counts how many times each data and address bits went wrong. Following the suggestions by our memory expert, John Anderson, four histograms are made.
 - histogram of data bits which 1 became 0,
 - histogram of data bits which 0 became 1,
 - histogram of address bits which were 0 when the error occurred, and
 - histogram of address bits which were 1 when the error occurred.
3. The dump of errors will not be made, even though it was proposed in the DDD document. This changed the arguments for TEST_RAM, TEST_DPM, and TEST_DMA, but it does not affect people using V2.8 or earlier, because the functions related to the argument had never been implemented.
4. Event header will not be made for TEST_DPM and WRITE_FIFO.
5. When executing TEST_RAM, TEST_DPM, and TEST_DMA, dc2_response shows the current status in the following format.

```
bit 0- 3 : test loop indicator
          # of remaining loops if #repeat<16
          #remaining/#repeat * 16 if #repeat>=16
bit 4- 7 : 1 : during writing
          2 : during DMAing from FIFO to DPM (for TEST_DMA)
          3 : during reading/testing
          f : test completed
bit 8-15 : command
bit 16-31 : #errors (on completion, it will be truncated to FFFF)
```
6. TEST_DMA will not print Ô. to terminal anymore, because it was causing a crash.
7. READ_FIFO will terminate DMA, upon receiving EXIT_TEST. Before, DMA was keep on running.
8. The number of words tested by TEST_RAM and TEST_DPM is not limited to 256k bytes anymore.
9. TEST_DMA checks that a) EOR (blank word) was received correctly, and b) the DMA has terminated properly (unless EOR word is at vsb_top_address).

10. Fixed a bug in TEST_RAM/DPM which went into a loooong loop if the least 2 bits of end_address were non-zero.
11. In TEST_RAM/DPM/DMA, an exact comparison is done between the data on memory and the expected data. Before, it wasn't quite so.
12. The pattern code 4 for TEST_RAM/DPM writes its address as data, but writes from high address to low address, in contrast to low to high in pattern code 3. Before, a wrong data was being written from low address to high address.
13. Error bit in dc2_status word will be cleared/set correctly in diagnostic codes.
14. The program is structured better, so that it does not have duplicate codes in different routines.

A.8.2 Casemode

1. EXIT_CASEMODE command now works, and it is recommended to use this command, instead of CLEAR.
2. The polling period of 0.25sec will be used in casemode, but the contents of polling_period word on DPM will not be modified. (Suggestion by Francesco.)

A.8.3 DDDMAIN

1. Upon exit from casemode, if polling_period is 0, a default 0.25sec will be used. Otherwise, the value in polling_period will be used. (Suggested by Francesco.)
2. CALL_PROGRAM will not have timeout if the second argument is 0.
3. Fixed a bug in CLEAR and FAST_CLEAR, which was setting one extra word for DMA transfer. (Bug found by Francesco.)
4. Upon exiting from CASEMODE, dc2_response will be set to 0xFEFO, instead of 0x00FO.

A.8.4 Changes from V2.7

1. DDD V2.8 and X3.0 can update n_events and the pointer table, and poll command at a regular pace even when data is coming in at maximum sustained rate.
2. Hold_clear option is supported. If the flag is on, on receipt of EXT_ABORT, it sets VETO line (pin 9-10 on ten pin connector on the DM-115), and does NOT clear DDD.

A.8.5 Known bug

When BUG340 is used in casemode, it will not receive any new commands after encountering a break point. This should not affect anybody besides a person developing DC-2 code.

A.9 V3.1

The TEST_RAM/DPM functions were speeded up for V3.1. There is no change in data taking codes.

The speed improvements are a factor 1.9 for pattern code 3 (address written in ascending order), and 1.6 for other patterns.

The actual execution times for testing 32Mbyte of DPM with V3.1 are:

```
32sec for running 1s and 0s, and AAAAAAAAA/55555555,  
27sec for its address written in ascending order,  
37sec for its address written in descending order.
```

V3.1 has some nice features, such as: protection against `end_address < start_address` for memory range, and providing the rough address where it is reading to allow an estimate of how long the test will take.

A.10 V3.2

Many internal changes were made for Version 3.2 but few of these will be detectable by the user. The philosophical change of adding a third mode, PINGPONGMODE, forced some internal reorganization that cannot be detected from the outside. Now CASEMODE is the non-data collection mode and when the user is ready to collect data, he chooses to enter MAINMODE or PINGPONGMODE. These modes are exited by entering CASEMODE. The idea of simply exiting a mode was ambiguous once there were more than two major modes because it was not clear where the code was exiting to. The other main changes moved Pingpong mode from a diagnostic routine to a data collection mode. E835 is the key Pingpong mode user. Detailed descriptions follow.

A.10.1 Main Changes and Bug Fixes.

1. New command - ENTER_MAINMODE was added to reduce possible confusion with the EXIT_CASEMODE command by providing a target. The command has the same internal code value (0xFE) as EXIT_CASEMODE and all old code will continue to function normally. The response word (0xFEFO) is also unchanged. It is recommended that any new code use the new command rather than the old.

2. Bug Fix - The response word to an ENTER_CASEMODE statement is 0xFDF0 and this was previously returned after rebooting because the reboot response, 0x00F0, was overwritten by the CASEMODE_ENTERED response during initialization. The firmware has been changed to allow the reboot response to survive initialization. This will allow diagnosing unexpected reboots. The DC-2 is in CASEMODE after reboot.
3. Bug Fix - User bits. The user bits were envisioned as the upper sixteen bits of the word count word, the first word in each event buffer. This worked until event counts overflowed the lower sixteen bits and user bits caused corruption of the bits above the sixteenth. V3.2 changes the way that the user bits are combined with the word count before the word count is loaded into the data buffer. Now, the user can partition the 32 word count bits based on expected event counts and the number of user bits desired. Only user bits with information (1s) will be combined into the word counts and if an errant word count extends into the user bits, only that count will be affected. Previously the error condition propagated into the following event counts until the next CLEAR command. Also the user bits were treated differently the first time they were used in an event, compared to all the other times. That error has also been fixed.
4. Bug fixes - Command. The command CLEAR in CASEMODE caused the processor to exit CASEMODE. It now does nothing with no response word. The command ENTER_CASEMODE in CASEMODE caused an UNKNOWN_COMMAND response. It now does nothing with no response word. No one admitted to having used these command situations so no one should notice.
5. Ping Pong mode buffers - Previous code enabled both buffers on start-up but this required diagnostic software to treat this as a special case. Normally only one of the two buffers is enabled at any time. The start-up code was changed to only enable one buffer.
6. Ping Pong response - Previously, on entering ping pong mode, the responses were 0x8300 followed quickly by 0x18300 to indicate readiness to accept data. To follow the symmetry of all other commands these responses are now 0x8300 followed quickly by 0x83F0 to indicate readiness to accept data.
7. Ping Pong mode exit - Previously the reception of any command caused an exit of ping pong mode. Now a CMD_ENTER_CASEMODE is required and all other commands are ignored. Also, a small bug previously allowed whatever command caused the ping pong mode exit to be left in the command queue and executed again in CASEMODE. This bug is fixed.
8. It should be noted that some code changes were made to the DART associated products DDT and dddmbx. These changes accommodated the above changes in the response words so that error checking was correct.