

270 F105  
20/87

DONote

A.M. Jonckheere  
8-AUG-1988  
Rev 9-NOV-1988  
Rev 1-Dec-1988  
Rev 23-Feb-1989  
Rev 19-Jun-1989  
Rev 9-Mar-1990  
Rev 16-Apr-1990

DO

Control Data Acquisition

Network Data Transmission

Protocol

### Introduction

The Protocol Subcommittee(\*) of the Software Group of DO was charged with the task of developing a communications protocol to be used in passing messages over the Controls Data network for the DO Experiment.

The subcommittee worked under a number of constraints. The most important arising from the decision, made previously by the collaboration, to use as much existing software as possible. A direct result of this decision was that Mike Shea et.al.'s Linac Control system, including Bob Goodwin's existing software, with necessary modifications only, would be used as the front end computers to do most of the data acquisition in the DO Control System. A direct result of this decision was that any changes needed by DO to this system would have to be compatible with the system's use in its other applications. including the FNAL Accelerator controls system.

Thus, the protocol that has finally been settled upon, is in many ways a concatenation of the needs of DO and the needs of the Accelerator Controls system, with a heavy dose of realism included, in the form of existing software. It is somewhat more complex and general than either controls system actually must have, but we feel that it meets the needs of both.

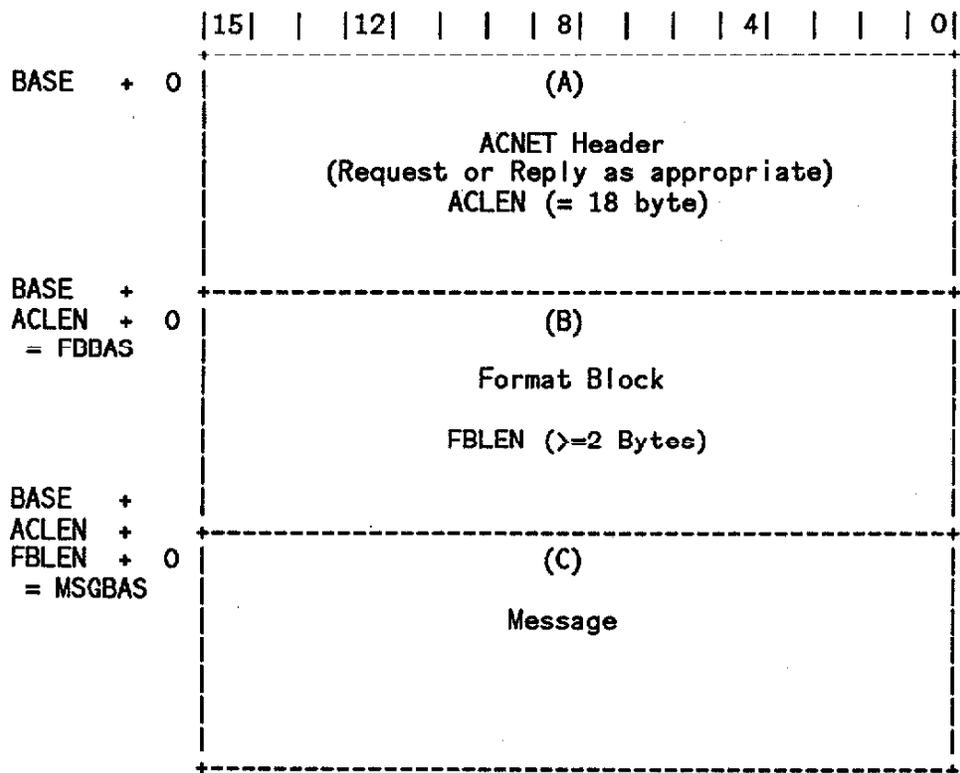
-----

! (\*) The members of the Protocol Subcommittee are:  
A. Jonckheere/DO, chairman, F. Bartlett/DO, C. Briegel/AD/Controls,  
R. Goodwin/AD/Controls.

## Global Decisions

1. ACNET - To maintain compatability with the FNAL Accelerator Controls system, in the absence of any currently available general network standards, it was decided to use the ACNET network message header. This allows for transmission of messages over linked networks, a situation that DO does not currently face, but may easily have to face in the not too distant future.
2. Format Block - A solution to the Byte-Swap problem. Since DO will live on at least two distinct Local Area Networks (LANs) with very different machine architectures dominating on each the problem arose as to which machine's byte/word formats would be used in communicating between them. R.Raja, made a detailed proposal that was eventually adopted, with minor changes. His proposal was that both formats be used, on their respective LANs. The Gateway between the LANs would be expanded to include a translation function. This requires that the data be tagged as to its data type. Instead of tagging each data value, Raja proposed that a Format Block be incorporated at the start of each message with all of the format information. This allows for possible efficiencies since "DO loop" type constructions can be easily incorporated.
3. Significant Event (Alarm) messages - should contain enough information in the message itself to do filtering without having to access a data base for each message. The DO view of alarms is that they should be available to anyone who wants them, anywhere they want them. In order to minimize network traffic on the DECNET, each user would be allowed to specify only those types of messages he/she is interested in. To minimize the number of Data Base accesses in a time critical situation, much of the information needed to do this filtering must be downloaded to the alarm sources, so that it can be echoed back with the appropriate alarm.

Message Formats



ACNET Header  
 Unsolicited Message ("Alarm") Format  
 (variation of Request)

(A-1)

		15		12		8		4		0				
!	BASE	+	0		0		ch cn	0	0		MsgType		0	ACNET HEADER
					Global Return Status = 0									
					Dest Node					Dest Lan				
					Source Node					Source Lan				
					Destination Task Name (2 words)									
					Chksum					Source Task ID				
!					Msg ID = 0 unless cn=1									
!					Msg Len									(bytes)

ch = CHK = Checksum Calculated for entire message  
 (if non-zero then Chksum will be non-zero)

cn = CAN = Cancel outstanding Request  
 (Request known by Source Node/Lan/MsgID)

MsgType = Message Type = 0 - Unsolicited Message (USM)

Global Return Status = 0

Dest Node/Lan are system supplied.

BroadCast      Dest Node and/or Lan = 255

MultiCast    127 < Dest Node and/or Lan < 255

Source Node/Lan are originating node.

Destination Task Name (2 words) is system supplied convention

Source Task ID = Originating Task.

Msg ID = 0 for a USM unless the cancel bit is set (see CN above)

Msg Len - Length of Message in bytes (includes this header)

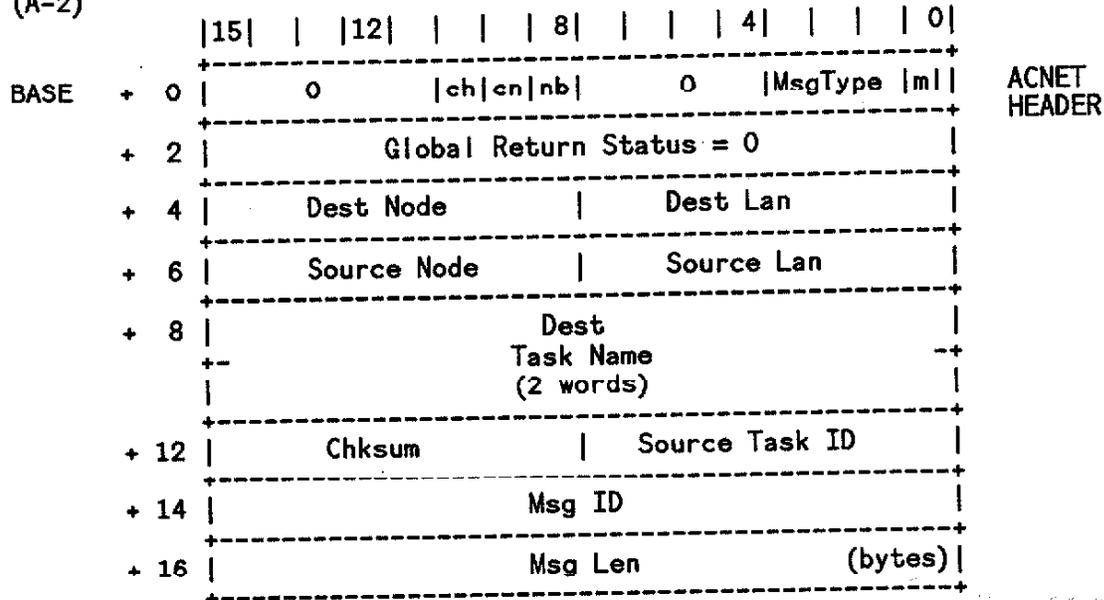
Through additional calls to the NetWork one can get from

Source Task ID -> Dest Task Name

to get back to originator of the USM?

ACNET Header  
Request Format

(A-2)

ACNET  
HEADER

ch = CHK = Checksum Calculated for entire message  
(if non-zero then Chksum will be non-zero)

cn = CAN = Cancel outstanding Request  
(Request known by Source Node/Lan/MsgID)

nb = NBW = No Busy Wait

MsgType = Message Type = 1 - Request

ml = MLT = Multiple Reply

Dest Node/Lan are system supplied conventions  
(filled in by CDAQ)

Source Node/Lan are system supplied conventions  
(filled in by CDAQ)

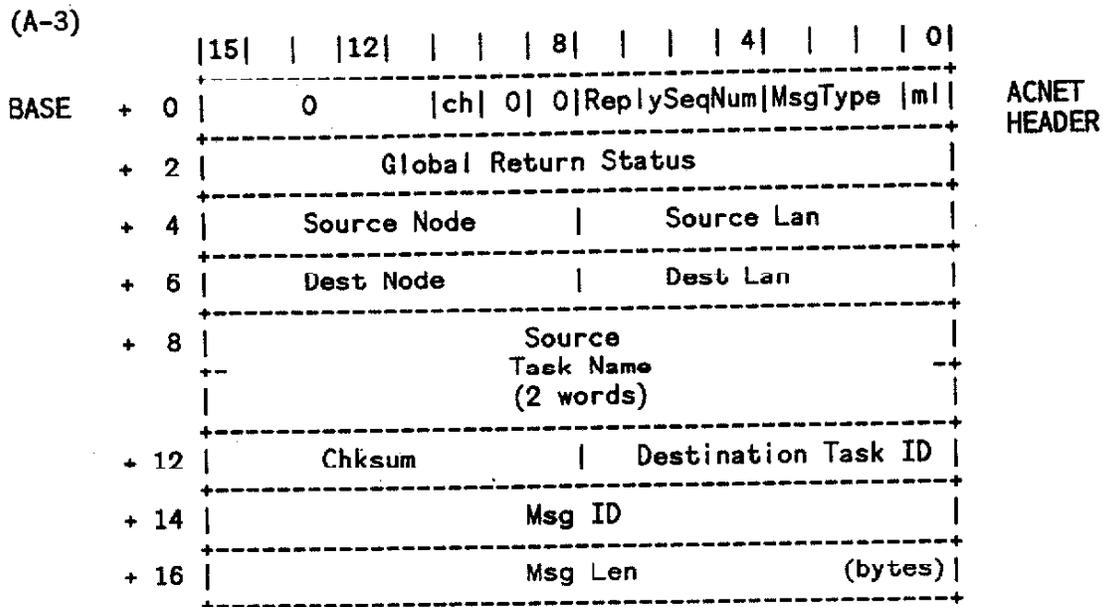
Dest Task Name (2 words) is system supplied convention  
(filled in by CDAQ)

Source Task ID is system supplied convention  
(filled in by CDAQ)

Msg ID identifies a specific user request  
(this is unique on Source Node/Lan)  
(filled by management Task)

Msg Len - Length of Message in bytes (includes this header)

ACNET Header  
Reply Format



ch = CHK = Checksum Calculated for entire message  
(if non-zero then Chksum will be non-zero)

ReplySeqNum = Sequence Number of this Reply, Cyclic - PACKETING ONLY

Global Return Status = ?

MsgType = Message Type = 2 - Reply

ml = MLT = Multiple Reply

Dest/Source Node/Lan are system supplied conventions

Source Task Name (2 words) is system supplied convention

Dest Task ID is system supplied convention

Msg ID is a unique message ID (unique on Source Node/Lan)

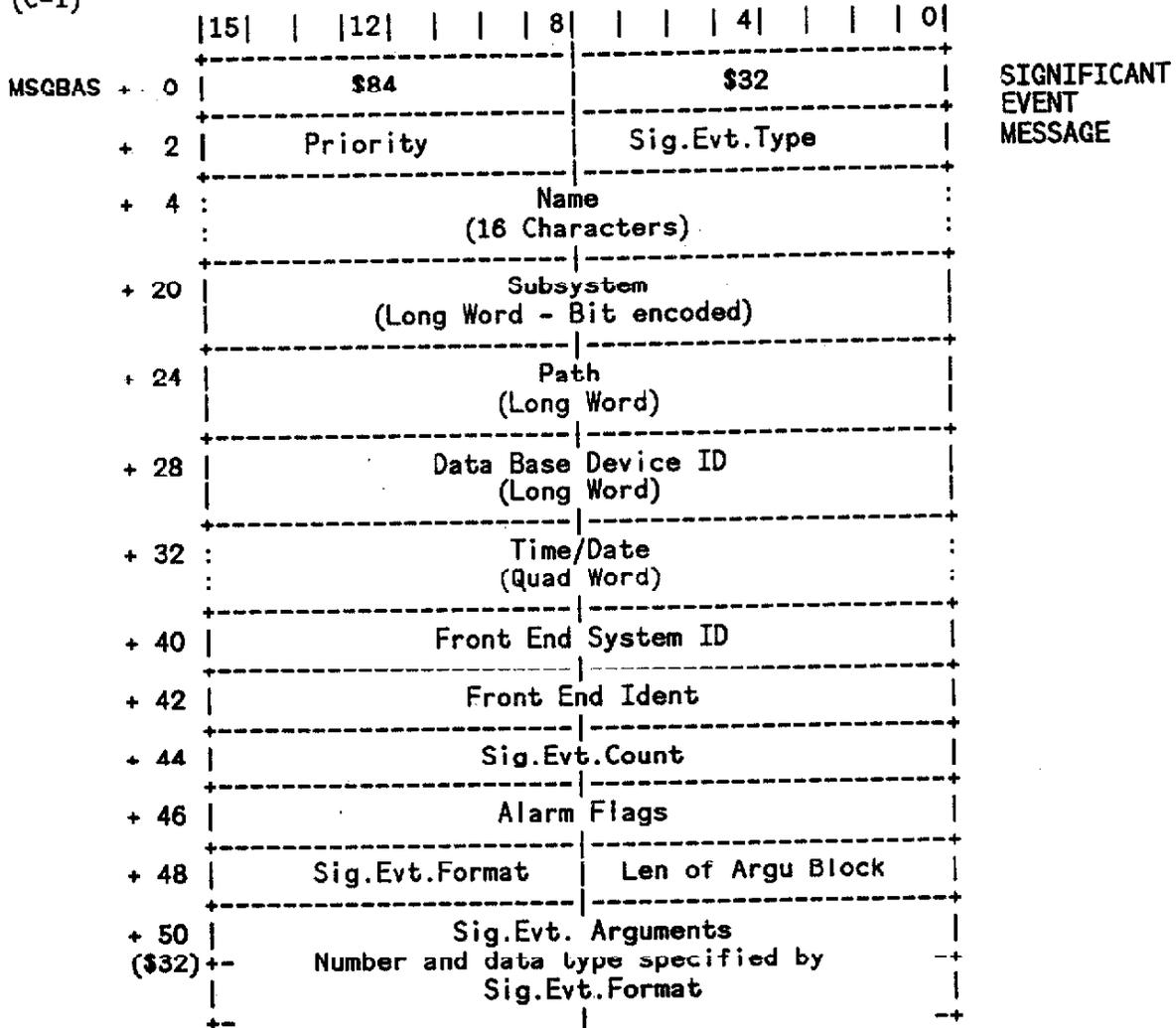
(bytes 4-14 (except for Chksum) are direct copies of corresponding  
bytes in request)

Msg Len - Length of Message in bytes (includes this header)



Significant Event Message

(C-1)



Doc # 10 Doc # 10000  
100 1001

Priority (1 Byte) = Monotonic Priority level of message  
 (to be defined - system/Ident dependent)  
 Sig.Evt.Type (1 Byte) = Significant Event Type  
 = bit 0 -> Informational/Alarm (0/1)  
 1 -> Going Good/Bad (0/1)  
 7 -> NoAbort/Abort (0/1)

Name (16 Bytes) = 12 Character Device Name  
 + 4 Character Attribute Name  
 (to be defined - system/Ident dependent)

Subsystem (4 Bytes) = Subsystem Designation  
 (Bit encoded) (System Dependent)  
 (for DO see Appendix A)

Path (4 Bytes) = Path through System Hierarchy Tree  
 (System Dependent)  
 (for DO see Appendix B)

Data Base Device ID (4 Bytes) = Pointer into Host's Data Base  
 (to be defined - system/Ident. dependent)

Time/Date (8 Bytes) = Time-Date stamp (LAN dependent)  
 - Token Ring (68k front ends) Lan  
 (binary encoded in 6 bytes + 1 word)  
 (Year/Month/Day/Hour/Minute/Sec/msec(word))  
 - VAX/DECNET Lan  
 (64 bit VAX binary time)

Front End System ID (2 Bytes) = Front End System identifier  
 (Front End System Dependent)

Front End Ident (2 Bytes) = Front End Channel or BIT identifier  
 (Front End System Dependent)

Sig.Evt.Count (2 Bytes) = Significant Event Count  
 (Count of particular Events, Transitions = Good+Bad)  
 (maintained on a channel by channel basis by  
 front ends)

Alarm Flags (2 Bytes) = Readback of Alarm control flags (See Appendix C)

Sig.Evt.Format (1 Byte) = Format Type of Significant Event  
 = 1 - Analog Event (Alarm)  
 = 2 - Binary Event (Alarm)  
 = 3 - Comment Event (Alarm)  
 (to be further defined - system dependent)

Len of Argu Block (1 Byte) = Len of Argument Block in Bytes

Sig.Evt.Arguments = Data returned with event  
 (See Appendix D)

## Alarm Block Formats

Much of the information in the Significant Event Message is present only to make the processing of these messages on the destination machine as efficient as possible. This information MUST be downloaded to the front end machines by the Host. In addition the front ends need information to know what devices should be examined, what limits they should use etc. There is a need for the Host to be able to command various types of resets.

This information is sent down to the front end machines in BLOCKS of data for efficiency. The message format is described in the following sections (DATA SETTING message), except for the data portion of the message. The format of the data portion of the message is described here.

## Device Information Block Format

Priority	(1 byte in msb of word, 0 padded)
Name	(16 characters)
Subsystem	(Long word)
Path	(Long word)
Device ID	(Long word)

## Control Blocks

- 1) Analog Control
 

Alarm Flags	(Control bits - Word)
Nominal Value	(left justified fraction of full scale - Long Word)
Tolerance	(left justified fraction of full scale - Long Word)
- 2) Binary Control
 

Alarm Flags	(Control bits - Word)
-------------	-----------------------
- 3) Comment Control
 

Alarm Flags	(Control bits - Word)
Bytes of text	(number of bytes of text to be returned in message) - of 0, do not change present text
Text	(Text string)

## Reset Block

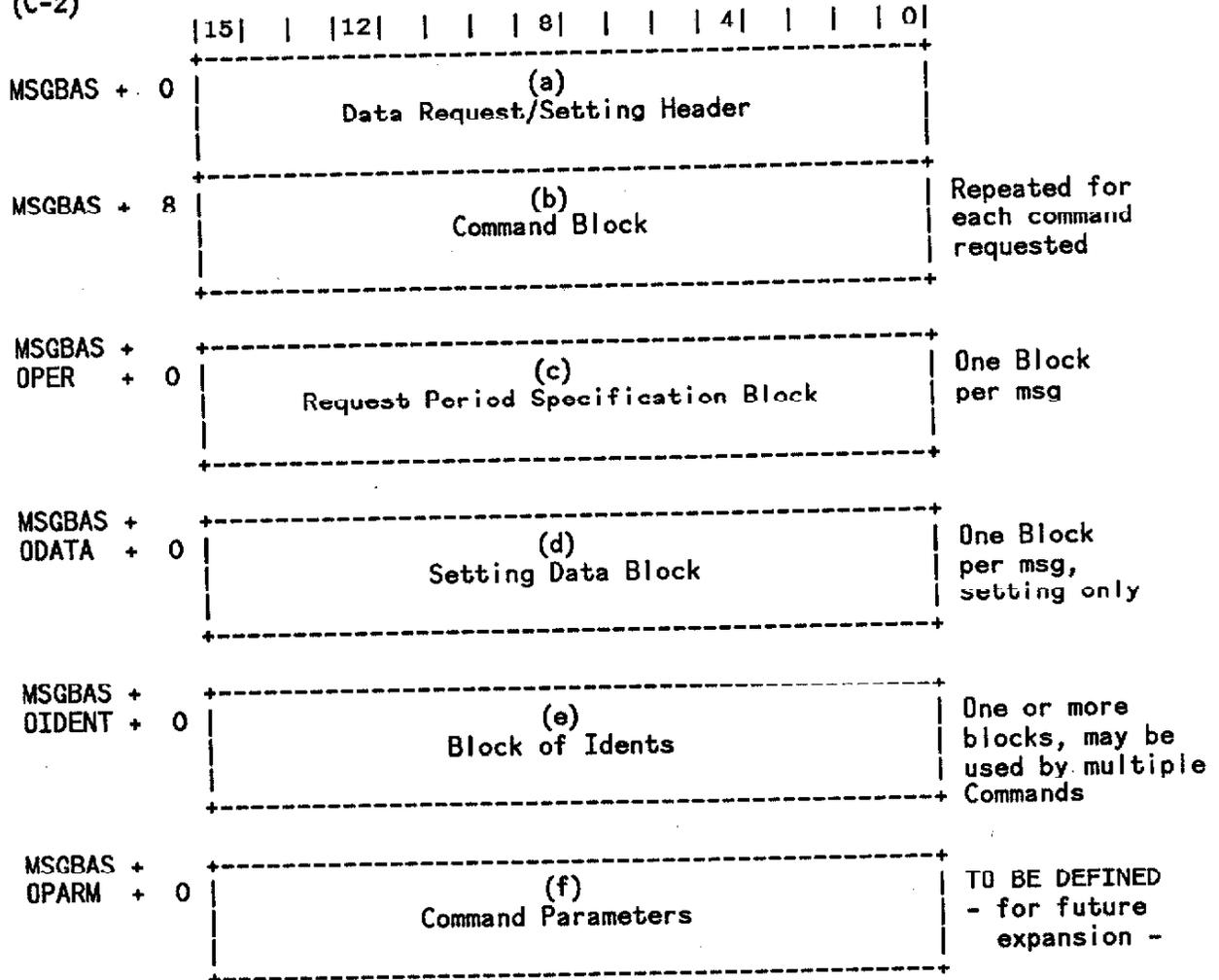
The system reset is a special command (Listype) that addresses an entire front end system. The TYPE of reset to be done is specified in the "Address" portion of the IDENT field of the message (see Appendix G for the form of the IDENT field). The data portion is blank.

The "Addresses" and their currently defined meanings are:

"Ident Address"	Meaning
0	Reset ALL alarms to "good"
1	Reset ALL alarm counts to zero

Data Request/Setting

(C-2)



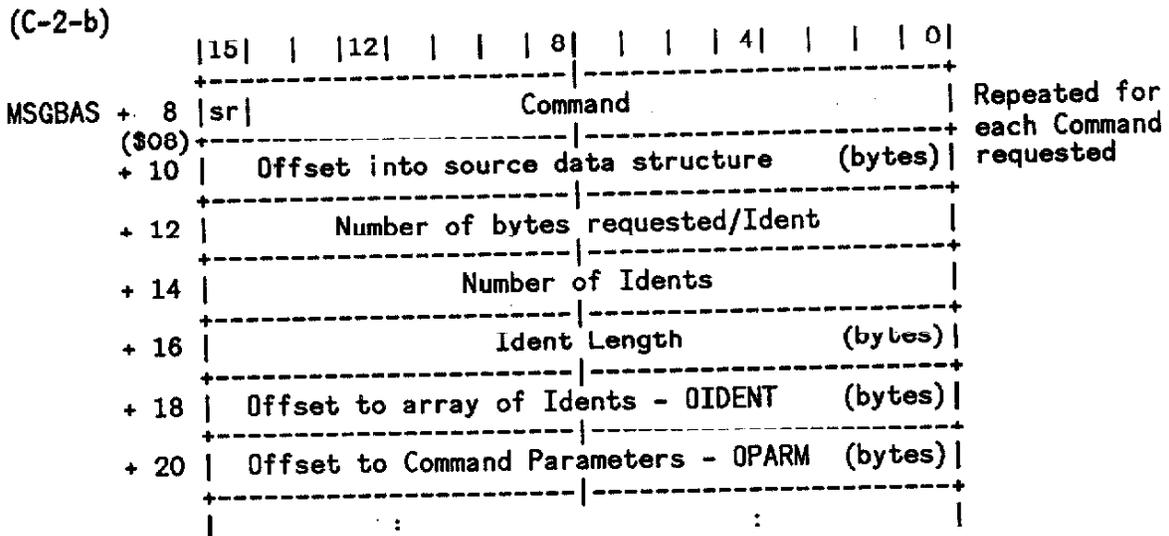
## Data Request/Setting - HEADER

(C-2-a)

	15	12	8	4	0	
MSGBAS + 0	\$82/\$83/\$86/\$87			\$08		DATA REQUEST/ SETTING MESSAGE
+ 2	Offset to Request Period Spec - OPER (bytes)					
+ 4	Offset to Setting Data Block - ODATA (bytes)					
+ 6	Number of Commands					

- \$82 = Normal Request for data  
 \$83 = Normal Setting  
 \$86 = Server Request for data (scatter gather by front end node)  
 \$87 = Server Setting (scatter gather by front end node)  
 \$08 = length of header in bytes  
 OPER = Offset to Request Period Specification Block from MSGBAS,  
 normally zero for settings (0 => One Shot request)  
 ODATA = Offset to Setting Data Block from MSGBAS in bytes, normally  
 zero for Reading Requests  
 Number of Command blocks that follow.

## Data Request/Setting - Command Block



Commands are the codes used by a front end system to determine what type of action and data is being requested. The format and meaning of Commands is front end system dependent.

Command = type of data to be returned, Front End specific

\*\* See Appendix E for currently defined Commands \*\*

sr = Status Return, if set returns status for each requested IDENT rather than data. Length of Status returned is determined by Number of bytes requested/IDENT

Offset into source data structure = offset of requested data from the base address of the channel specified by IDENT

Number of bytes requested/IDENT = n bytes of data are return/IDENT starting at channel base + offset above

Number of IDENTS = number of idents to process for this Command

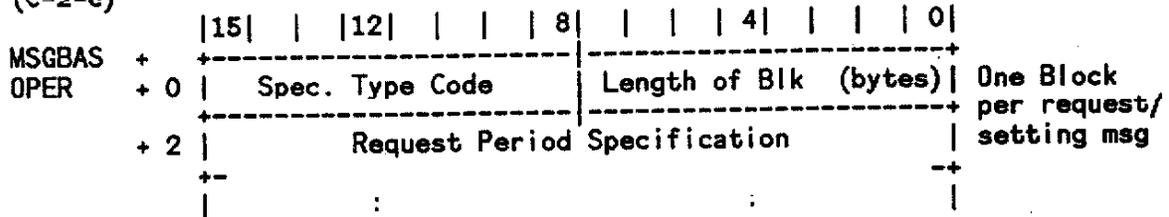
Ident Length = length in bytes of the Idents to be processed, must all be the same length, interpretation of the Ident depends on the Command/length combination

Offset to array of Idents = offset in bytes from MSGBAS of the first Ident to be processed for this Command

Offset to Command Parameters = offset in bytes from MSGBAS of an array of additional parameters to modify the action of this Command

Data Request/Setting - Period Specification Block

(C-2-c)

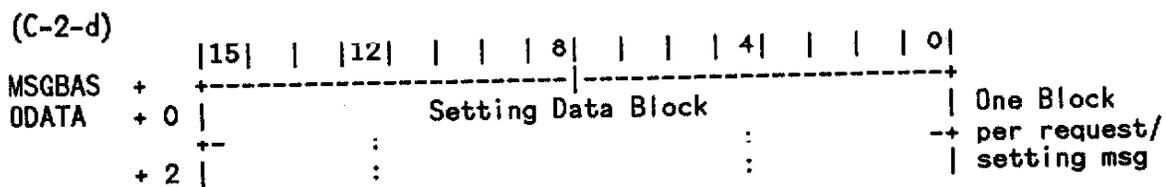


Spec Type Code = Specification Type Code  
 = For expansion, presently = 0  
 Length of of Blk = Length of Specification Type Code block, including  
 itself in bytes  
 Period Specifications: Are given in Appendix F

There are currently three types of period specifications envisioned:

1. First Time: Specify when the first reading/setting should occur.
2. Next Time: Specify when all subsequent reading/settings should occur.
3. Reply Blocking: Specify how many, or how long messages should be saved in order to increase network efficiency by grouping messages for transmission.

Data Request/Setting - Setting Data Block

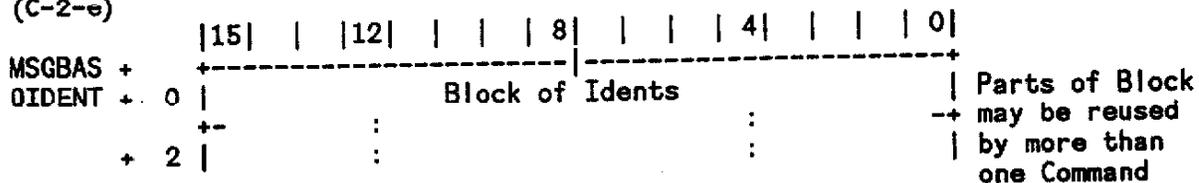


Block of setting data - format depends on Command/Ident pairs. Order is:

Loop over Commands, within each Command, loop over Idents.

Data Request/Setting - Ident Block

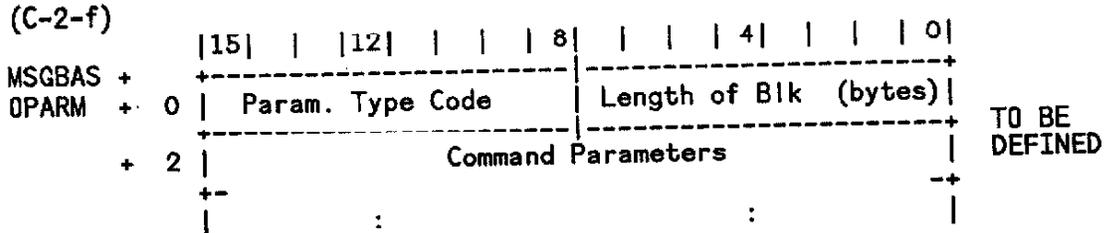
(C-2-e)



Idents are the codes used by a front end system to determine what device is being requested. The format and meaning of the Idents is front end system dependent.

The current Ident formats are listed in Appendix G.

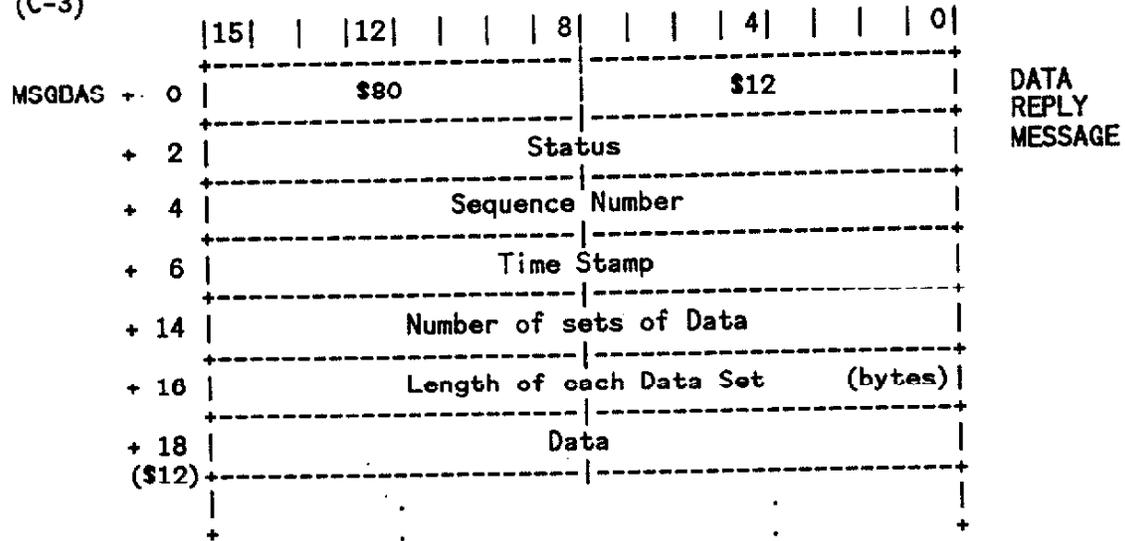
Command Parameters



This block is currently undefined. It is included as a possible extension to the existing specification of actions taken by the Front End machines.

Reply to Request

(C-3)



- Status = 0 - OK
- Sequence Number = Number of Repeated Replies for this Request
- Time Stamp = Time message was sent
- Number of sets of Data = Number of Replies to this Request being sent together
- Length of each Data Set = obvious (in bytes)

Reply to Setting

(C-4)

	15	12	8	4	0
MSGBAS + 0	\$81			\$04	
+ 2	Status				

SETTING  
REPLY  
MESSAGE

## APPENDIX A

## Current D0 Subsystem Assignments

Any device can be assigned to one or more subsystems. These can then be used by the Alarms dispatching process to filter the alarms sent to any receiving process. The subsystem is specified as a bit encoded long word with each bit corresponding to a subsystem.

The current subsystem assignments for D0 are:

	Bit	Subsystem description
Software	0	Host - Online Software
	1	Goodwin front end
	2	Briegel (IBM PC) front end
	3-7	Spare
Data	8	Clock
	9	Pulser
	10	Trigger
	11	Readout
	12	Detector Monitoring
	13-15	Spare
Spare	16-23	Spare
Utilities	24	Cryogenics
	25	Magnet
	26	AC Power
	27	Low Voltage Power
	28	High Voltage Power
	29	Cooling Water - Temp, Flow, Pres etc
	30	Cooling Air - Temp, Flow, etc
	31	Spare

## APPENDIX B

## CURRENT Hierarchical Path Assignments for DO

The Path descriptor encodes a hierarchical description of the detector. Each device in the system is assigned to one and only one "NODE". It's position in the hierarchy describes its influence on other devices. Any device at a given node can influence the operation of any device at the same or lower node on the same branch. Its position DOES NOT imply that it influences ALL devices at the same or lower nodes. It DOES imply that it influences some device on the same node, or more than one of the lower nodes.

For DO the Path descriptor is nibble encoded. Within the 32 bit long word this leaves the possibility for 8 levels with up to 15 nodes/level. In addition a zero (0) path descriptor implies the highest or root level which can contain only one node (called "DO").

The current assignments of the Hierarchical Path descriptor for DO are:

HEX VALUE	NAMF	Description
00000000	DO	The entire Detector - default
n0000000	see below	(Detector Elements)
n1000000	Readout	Readout elements (ADCs, Preamps etc)
n2000000	Clock	Clock lines
n3000000	Control	Control Elements (HV etc)
n4000000	Calibration	Calibration Elements (Pulsors etc)
A0000000	LV1	Level 1 Trigger Framework
B0000000	LV2	Level 2 Trigger
C-E		(Spare)
F0000000	MONITOR	Environmental Monitoring System - READ ONLY
F1000000	CNT_HSE	Fixed Counting House
F1100000	Cnt_Hse_1	Counting House - 1st Floor
F11n0000	see below	(Detector Elements)
F1200000	Cnt_Hse_2	Counting House - 2nd Floor
F12n0000	see below	(Detector Elements)
F1300000	Cnt_Hse_3	Counting House - 3rd Floor
F13n0000	see below	(Detector Elements)
F1400000	Cnt_Hse_4	Counting House - 4th Floor
F14n0000	see below	(Detector Elements)
F2000000	Mov_Cnt_Hse	Moving Counting House
F2100000	Mov_Cnt_Hse_1	Moving Counting House - 1st Floor
F21n0000	see below	(Detector Elements)
F2200000	Mov_Cnt_Hse_2	Moving Counting House - 2nd Floor
F22n0000	see below	(Detector Elements)
F2300000	Mov_Cnt_Hse_3	Moving Counting House - 3rd Floor
F23n0000	see below	(Detector Elements)
F2400000	Mov_Cnt_Hse_4	Moving Counting House - 4th Floor
F24n0000	see below	(Detector Elements)
F3000000	Platform	Detector Platform
F3100000	PLT_NE	Platform - North East Sector

F31n0000	see below	(Detector Elements)	
F3200000	PLT_E	Platform - East	Sector
F32n0000	see below	(Detector Elements)	
F3300000	PLT_SE	Platform - South East	Sector
F33n0000	see below	(Detector Elements)	
F3400000	PLT_NC	Platform - North Central	Sector
F34n0000	see below	(Detector Elements)	
F3500000	PLT_C	Platform - Central	Sector
F35n0000	see below	(Detector Elements)	
F3600000	PLT_SC	Platform - South Central	Sector
F36n0000	see below	(Detector Elements)	
F3700000	PLT_NW	Platform - North West	Sector
F37n0000	see below	(Detector Elements)	
F3800000	PLT_W	Platform - West	Sector
F38n0000	see below	(Detector Elements)	
F3900000	PLT_SW	Platform - South West	Sector
F39n0000	see below	(Detector Elements)	
F4000000	DET	Top of Detector Platform	
F4n00000	see below	(Detector Elements)	

The following are included below several of the nodes above. They are indicated by an "n" in one of the HEX fields.

"n"	Detector	description
1	LVO	Level 0 Detector
2	MUON	MUON detector
3	VTX	VerTeX detector
4	CDC	Central Drift Chamber
5	FDC	Forward Drift Chamber
6	TRD	Transition Radiation Detector
7	CAL	CALorimeter
8	SAMUS	Small Angle MUon Spectrometer

APPENDIX C

CURRENT Alarm Flag bit assignments

The currently defined Alarm Flag bit assignments are:

Alarm Flag Bit assignments

- bit 15 Active - enable scan for alarms
- 14 Nominal - nominal state of bit (Binary alarms only)
- 13 Inhibit - assert external control line upon alarm
- 12 2x enbl - require 2 readings (good or bad) to set/reset alarm
- 11 Beam - scan only when external line is asserted
- 10 Bypass - Send "good" message, then clear ACTIVE bit
- 9 2x cntr - counter counts number of consecutive good/bad readings
- 8 State - 0=Good, 1=Bad
  
- 7 Disable - Disable reporting of alarms but keep statistics
- 6-0 - Unused, Spare

Binary Alarms (Sig.Evt.Format=2)

No Arguments

Comment (Text) Alarms (Sig.Evt.Format=3)

Text (Byte string)

APPENDIX D

CURRENT Significant Event (Alarm message) Argument Lists

The currently defined Significant Event Argument lists are:

Analog Alarms (Sig.Evt.Format=1)

Nominal	(LW)
Tolerance	(LW)
Reading	(LW)
Setting	(LW)

Binary Alarms (Sig.Evt.Format=2)

No Arguments

Comment (Text) Alarms (Sig.Evt.Format=3)

Text (Byte string)

## APPENDIX E

## CURRENTLY DEFINED COMMANDS (LISTYPES)

The Commands (LISTYPES) currently recognized by the VME 68k front end computers are:

Command	Ident type	Read/Write	Data Size	Max size settable	Description
0	CHAN	R	2 I*2	-	Analog reading 0
1	CHAN	R/W	2 I*2	2	Analog setting 0
2	CHAN	R/W	2 I*2	6	Analog Nominal Value 0
3	CHAN	R/W	2 I*2	2	Analog Tolerance 0
4	CHAN	R/W	4 I*2	2	Analog alarm flags, counter
5	CHAN	R	4 I*1	-	Analog Associated status
6	CHAN	R	2 I*2	-	Motor countdown word
7	CHAN	W	2 I*2	2	Knob-scaled relative setting
8	CHAN	R/W	4 I*2	62 *	ADESC Analog Control Field
9	CHAN	R/W	4 I*2	4	ADESC Analog Status Definition
10	CHAN	R/W	6 I*2	6	ADESC Analog Control Definition
11	CHAN	R/W	2 I*2	2	Conversion type (in hi byte)
12	CHAN	R/W	16 F*4	16	ADESC Scale factors 00
13	CHAN	R/W	18 C*18	30 **	ADESC Title (descriptive)
14	CHAN	R/W	6 C*6	6	ADESC Status text (last 6 of 18)
15	CHAN	R/W	6 C*6	6	ADESC Name
16	CHAN	R/W	4 C*4	4	ADESC Units text
17	CHAN	R/W	2 I*2	2	ADESC family word
18	CHAN	R/W	2 I*2(en)	2	ADESC Date of last modif.
19	CHAN	R	4 I*2	-	CHAN ident from 6-char name
20	ADDR	R/W	V I*1	~3500	Memory (byte access)
21	BIT	R/W	2 I*1	2 ***	Digital bit I/O (via BIT)
22	CHAN	W	2 I*1	2	Associated control via CHAN
23	BIT	R/W	16 C*16	16	Digital Bit title
24	BIT	R/W	4 I*2	2	Digital alarm flags, counter
25	BYTE	R/W	1 I*1	8	Digital byte I/O (via BYTE)
26	SVAR	R	V Var	-	System global variables
27	CHAN	R	2 I*2	- !	Captured reading data word
28	CHAN	R/W	2 I*2	2	Analog spare data word
29	ADDR	R/W	V I*2	~3500	Memory (WORD access)
30	RDAT	R/W	16 Var	16	Read Data Access Table entry
31	PAGE	R/W	4 I*2	20 !	Small CRT page appl entry pt
32	PAGE	R/W	16 C*16	16	Small CRT page title
33	PAGE	R/W	128 Var	120 !!	Small CRT page memory
34	BYTE	R/W	4 L*4	4	Binary byte I/O address
35	GID	R/W	32 Var	32	Gen Interesting Data (TOD)
36	SIO	R/W	V I*1	inf	Serial I/O port (R-1st wrd=#byt)
37	PAGE	R/W	8 I*1	8	Small CRT Auto-Page Param
38	ADDR	R/W	V I*2	~3500	Memory (FIFO word access)
39	CHAN	W	2 I*2	2	Unscaled relative setting
40	CHAN	R	4 F*4	4	Analog Reading - Eng. units
41	CHAN	R/W	4 F*4	4	Analog Setting - Eng. units
42	CHAN	R/W	4 F*4	4	Analog Nominal - Eng. units
43	CHAN	R/W	4 F*4	4	Analog Tolerance - Eng. units

44	CHAN	W	4 F*4	4	Relative Setting - Eng. units
45	COP	W	V Var	inf	Co-processor message queue
46	ADDR	R/W	V I*2	~3500	Memory IO /w MMAPS (by words)
47	MMAP	R/W	V I*2	~3500	MMAPS Table access
48	PAGE	W	2 I*2	2	On-demand appl page invocation
49	CHAN	R	V I*2	-	Family of channels (1st wrd=count)
50	DSTR	R/W	V I*2	inf	Data stream I/O
51	DSTR	R/W	V I*2	inf	Data stream I/O (incl prev data)
52	DSTR	R	V Var	-	Data stream queue header access
53	DSTR	R/W	32 Var	32	Data stream table (DSTRM) entry
54	DSTR	R/W	8 C*8	8	Data stream name
55	CHAN	R	4 I*2	-	CHAN ident from 16-char DO name
56	CMNT	R/W	4 Var	2	Comment alarm flags, counter
57	CHAN	R/W	16 Var	16	DO analog alarm control
58	BIT	R/W	2 I*2	2	DO binary alarm control
59	CMNT	R/W	32 I*1	32	DO comment alarm cntrl (count= 1st wd)
60	RST	W	2 I*2	2	General resets RST=0:alarms, =1:trips
61	CHAN	R/W	30 Var	30	AADIB (analog alarm device info block)
62	CHAN	R/W	16 I*1	16	AADIB analog name (16-char)
63	CHAN	R	2 I*2	-	AADIB Date-of-last-change
64	BIT	R/W	30 Var	30	BADIB (binary alarm device info block)
65	BIT	R/W	16 I*1	16	BADIB binary name (16-char)
66	BIT	R	2 I*2	-	BADIB Date-of-last-change
67	CMNT	R/W	30 Var	30	CADIB (comment alarm device info block)
68	CMNT	R/W	16 I*1	16	CADIB comment name (16-char)
69	CMNT	R	2 I*2	-	CADIB Date-of-last-change
70	TABL	R/W	8 Var	8	System table directory entry
71	BYTE	R/W	2 I*2	8	Digital word I/O (via BYTE)

(CDATA = Comment data,  
 AADIB = Analog Alarm Device Information Block,  
 BADIB = Binary Alarm Device Information Block,  
 CADIB = Comment Alarm Device Information Block)

- \* 62 bytes is entire ADESC table following this entry
- \*\* 30 = 12(1st 12 of 18) + 6 + 6 + 4 + 2 of Commands 13-17
- \*\*\* 1st byte is type of control, 2d is length of pulse (if needed)
- | Can set next entry at the same time
- !! read 128 = I20 of Command 33 + 8 of Command 37
- 0 Analog readings are left justified fraction of full scale
- 00 Scale factors are: ADC full scale, ADC offset, DAC full scale, DAC offset  
 Readings, Nominal and tolerance use ADC values, Settings use DAC values.  
 Eng units = (float(raw)/32768)\*full\_scale + offset

APPENDIX F  
CURRENT PERIOD SPECIFICATIONS

FIRST TIME	NEXT TIME	REPLY BLOCKING
A004 After delay1 delay1	D004 After delay2 delay2	B006 Reply after #msgs #msgs (max) delay3 delay3 (min)
A106 After event1 event1 +delay1 delay1	D106 After event2 event2 +delay2 delay2	
A20A After 64 bit time-of-day time	D20A After 64 bit time-of-day time interval	
A304 Manual* delay1 +delay1	D304 Manual* delay2 +delay2	
	D4xx Data meets param criteria (ie Changes)	

\* needs new Command (Listype) with IDENT = msgID

(delay1, delay2 and delay3 are in units of milliseconds)  
(event1 & event2 are Tevatron clock pulses but may be expanded to include  
other events)  
(Time-of-day = 64 bit time format on appropriate Lan)  
(Manual when requesting task sends short "now" message)

Reply Blocking is the ability to accumulate messages for a time so that several messages can be sent over the network together. This ability is useful to increase the efficiency of the network transmission process.

The current scheme allows the user to specify both a maximum number of messages to be accumulated, and a maximum time interval to wait between actually sending the accumulated messages. The network frame will be sent whenever the first of the two possible conditions is met.

## APPENDIX G

## CURRENT IDENT FORMATS

The VME 68k Front Ends currently recognise 20 different Ident types. 10 are the old versions, 10 are new. Each Command accepts only 2 of the 20 possible, one old, one new. The systems determine whether old or new is being used by the Length of Ident word. The Idents are:

CHAN, BIT, ADDR, NAME, BYTE, SVAR, RDAT, PAGE, GID, SIO, DSTR, CMNT, RST  
and TABL.

All IDENTs take the form:

```

+-----+-----+
| System | Address Code |
+-----+-----+

```

In the Old system, the System field was a byte containing the System number. In the new system this is a word contain the LAN in the high order byte, the Node (system) in the low order byte. That is:

```

                15 Old 8   15   New   0
+-----+ +-----+ +-----+-----+
| System | = | NODE | -> | LAN | NODE |
+-----+ +-----+ +-----+-----+

```

Except for the ADDR and NAME Idents, the address code is a byte in the old system, a word in the new. For the ADDR Ident this is a 24 bit Memory address in the old system, a 32 bit Memory address in the new. For the NAME Ident, this is a 6 byte character string in the old system, a 16 byte string in the new.

The meaning of the Address Code is:

CHAN = Channel number  
 BIT = Bit numbr  
 ADDR = Memory Address  
 NAME = Channel/Bit Name  
 BYTE = Byte number  
 SVAR = Signed offset in System Variables table  
 RDAT = Entry number in System RDATA (Read Data Access) Table  
 PAGE = System "Page" number  
 GID = Offset in the System Generally Interesting Data Table  
 SIO = Serial IO Port - sends/receives data from a Serial Port.  
 CIDX = Comment alarm number  
 GIDX = General Reset number

\*\*\*\*\*

Number of difference sections found: 59