A Sierra Monitor Company

Driver Manual
(Supplement to the FieldServer Instruction Manual)

FS-8700-78 Dart

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after January 18, 1999

Instruction Manual Part Number FS-8700-78

2/20/2003

TABLE OF CONTENTS

1 Dart Driver Description

The Dart Driver is designed for connection to a Data Air Coporation Dart Device.

The Dart device is a active element on a network of Data Air devices such as DAP panels. It manages the devices and provides co-ordination and supervision. In performing these duties the Dart device polls all the devices on the network for all the data. The Dart is capable of 'echoing' the responses from these devices on one of its RS232 ports. This driver is designed to listen passively to these echoes and store device data. In addition the driver can send messages (containing set point data, for example,) directly to individual devices on the network.

The FieldServer is connected to the RS232 serial port of the Dart. The Fieldserver can read and write but active polling must be minimized as in reduces the amount of time that the DART spends controlling the networked devices. The driver operates primarily as a passive client listening to echoes of the data being polled by the DART. The DART must be set to 'Echo' mode on the front panel of the Dart for the driver to operate correctly. There is no alternative to this essential but manual setup-step.

The driver may be configured very simply (See example 1 in section 4 of the manual). In addition a number of advanced configurations are possible. The manual is divided to separate the basic and advanced topics.

The driver supports the common message formats for common Data Aire devices. A list of the supported messages is provided in the manual. The driver cannot be used to configure or read the status of a DART device itself.

The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer.

1.1 Performance Issues

Several factors outside FieldServer's control make for slow communications with Data Aire devices. The Data Aire communications is based on a very low baud rate. In addition inter-message timing constraints and overhead requirements for active messages in a Dart configuration add significant time to each transaction.

When a used with a DART device, writing to a set point (or other variable) may take up to 8 seconds to complete the transaction. The results of the write will not be seen until the DART has timed-out back into control mode and echoes the new data to the Fieldserver. This could take several minutes if there are many devices on the loop and will take at least 30 seconds.

When writing continuously, with DART configurations, the minimum time interval between successive writes is 2 minutes and with non-DART configurations is 1.8 seconds.

2 Driver Scope of Supply

2.1 Supplied by Sierra Monitor for this driver

RS485 connection adapter
Driver Manual.

2.2 Provided by user

Data Aire Dart and documentation
RS232 cable for the loop network

3 Hardware Connections

Hardware

3.1 Hardware Configuration – Dart Present

When a DART device is connected to the Data Aire field devices then the Fieldserver is connected to the serial port of the DART. The DART must be set to 'Echo' mode. This is done on the front panel of the DART.

4 Basic Configuration

For a detailed discussion on FieldServer configuration, please refer to the instruction manual for the FieldServer. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See ".csv" files on the driver diskette).

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for Dart Driver communications, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the destination device addresses need to be declared in the "Client Side Nodes" section, and the data required from the servers needs to be mapped in the "Client Side Map Descriptors" section. Details on how to do this can be found below.

The Dart Serial Driver cannot be configured as a data server.

Note that in the tables, * indicates an optional parameter, with the bold legal value being the default.

4.1 Data Arrays

Section Title
Data_Arrays

Column Title	Function	Legal Values
Data_Array_Name	Provide name for Data Array	Up to 15 alphanumeric characters
Data_Format	Provide data format. Each data array can only take on one format.	FLOAT, BIT, UInt16, SInt16, Packed_Bit, Byte, Packed_Byte, Swapped_Byte
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required for the data being placed in this array.	1-10,000

Example

```
//    Data Arrays
//
Data_Arrays
Data_Array_Name,     Data_Format,     Data_Array_Length
DA_AI_01,            UInt16,               200
DA_AO_01,            UInt16,               200
DA_DI_01,            Bit,                  200
DA_DO_01,            Bit,                  200
```

4.2 Client Side Connections

Section Title		
Connections		
Column Title	**Function**	**Legal Values**
Port	Specify which port the device is connected to the FieldServer This is the port connected to the DART's RS232 port.	P1-P8, R1-R2 (P1-P8) requires 232/485 converter.
Secondary_Port*	**Using a Dart ?** Do not specify this parameter.	P1-P8, R1-R2 (R1-R2) requires 232/485 converter.
Baud*	Specify baud rate	2400 (Others available but Daire only operates at 2400 baud.)
Parity*	Specify parity	**None**
Data_Bits*	Specify data bits	**8**
Stop_Bits*	Specify stop bits	**1**
Protocol	Specify protocol used Either keyword may be used.	Dart
Handshaking*	Specify hardware handshaking	**None**
Poll Delay*	Time between internal polls	0-32000 seconds **default 1 second**

Example

```
//    Client Side Connections

Connections
Port, Parity, Data_Bits, Stop_Bits, Protocol, Poll_Delay
P1,   2400  , None,    8,        1, Dart   , 0.100s
```

4.3 Client Side Nodes

Section Title		
Nodes		
Column Title	**Function**	**Legal Values**
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	This is not the node address of the Dart device ! Specify the node address of the DAP device whose data you wish to monitor.	1-259
Protocol	Specify protocol used	Dart
Port	Specify which port the Dart is connected to the FieldServer	P1-P8, R1-R2

Example

```
//    Client Side Nodes

Nodes
Node_Name, Node_ID, Protocol, Port
Unit1,     1,       Dart ,    P1
```

4.4 Client Side Map Descriptors

4.4.1 FieldServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from "Data Array" section above
Data_Array_Location	Starting location in Data Array	0 to maximum specified in "Data Array" section above
Function	Function of Client Map Descriptor The use of the WRBX keyword is recommended for DART configurations as communications are minimized.	RDBC, WRBC, WRBX, Passive

4.4.2 Timing Parameters

Column Title	Function	Legal Values
Scan_Interval	Rate at which data is polled	>0.1s

4.4.3 Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the node names specified in "Client Node Descriptor" above
Length	Length of Map Descriptor	1 The length must always be set to one.
Address	Starting address of read block	Not required for client configuration. This parameters is used in simulation configuration only.
The following parameters apply only to the Dart Driver.		
DA_Func*	Tell the driver to poll the device for all the data that is available. The data is device specific. Other uses of this parameter are permitted. These uses are discussed in the Advanced Topics section of this manual.	Everything See table 4.4.3.1 for a map of how the data is stored in the Fieldserver data arrays

DA_Field	Only required when the function is a write (wrbc) or when the map descriptor is associated with a write by means of the DA_Assoc parameter value. This is the name of the data field whose value you wish to set in the device. Other uses of this parameter are permitted. These uses are discussed in the Advanced Topics section of this manual.	See Table 4.4.3.3 for a list of permitted values.
DA_Assoc	Use to associate passive map descriptors with an active map descriptor. Using this parameter you associate multiple fields with one wrbc map descriptor, thus reducing writing multiple values to one device using just one message.	Any positive integer.

4.4.4 Map Descriptor: Example 1 – Read all possible data

```
Map_Descriptor_Name, Data_Array_Name, Data_Array_Offset, Function, node_name, Address, Length, DA_Func
Read_Node_01,        DA_01          , 0                , passive  , Unit1     , 0       , 1300  , Everything
```

The driver waits passively for the Dart to echo data from the devices.

No Scan Interval. The driver will process data as fast as the Dart can serve it.

4.4.5 Map of How Data Aire Device Data is stored in a Data Array

In the example of 4.4.1 the Fieldserver listens passively for all possible data from the device called 'unit1' and stores the data in an array called 'DA_01'.

The data that is obtained from 'unit1' is dependent on the type of device. Irrespective of the device type the arrangement of data, stored in DA_01, is fixed. If a data field cannot be obtained from 'unit1' then the array is left with a zero value for that data field.

Table 4.4.5.1: Array Locations of 'Everything'

In the following table the array location indicates the offset in the data array at which a data field can be found. (This offset is relative to the offset specified in the map descriptor.) The columns headed 2,3 … indicate the unit types for which the data fields are available. For example: The field 'd_temp' can be read from unit types 2,5,6,7,9 but not from any of the other unit types. It is beyond the scope of this manual to describe each field and to indicate valid ranges. Such information should be obtained from the Data Aire Corporation.

'x' Indicates Read only
'X' Indicates a point that can be read & written.
'w' Indicates a write only point.

Array Location	Method	Num Elements	Data Field	2	3	4	5	6	7	8	9	14	15	Message Type
1	1	1	Zone		w	w	w	w	w	w	w	w	w	Dap-Config
2	1	1	Inhibit		w	w	w	w	w	w	w	w	w	Dap-Config
3	4	1	unitType	x	x	x	x	x	x	x	x	x	x	Dap-Unit
4	2	1	temp	x			x	x	x		x			Dap-Stat
5	2	1	hum	x			x	x	x		x			Dap-Stat
6	2	1	d_temp	x			x	x	x		x			Dap-Stat
7	3	8	mode	x			x	x	x		x			Dap-Stat
23	3	8	hold	x			x	x	x		x			Dap-Stat
39	1	1	cs_on	x			x	x	x		x			Dap-Stat
40	1	1	hs_on	x			x	x	x		x			Dap-Stat
41	1	1	valvePCT	x			x	x	x		x			Dap-Stat
42	1	1	hVlvPCT	x			x	x	x		x			Dap-Stat
43	4	14	errors	x			x	x	x		x			Dap-Stat
57	2	1	hiTemp	x			x	x	x		x			Dap-Stat
58	2	1	loTemp	x			x	x	x		x			Dap-Stat
59	2	1	hiHum	x			x	x	x		x			Dap-Stat
60	2	1	loHum	x			x	x	x		x			Dap-Stat
61	1	1	chilled_water	x			x	x	x		x			Dap-Stat
62	1	1	compressor_config	x			x	x	x		x			Dap-Stat

63	1	1	heat_strip_config	x	x	x	x	x	Dap-Stat
64	1	1	hum_config	x	x	x	x	x	Dap-Stat
65	1	1	csUtilPct	x	x	x	x	x	Dap-Stat
66	1	1	hsUtilPct	x	x	x	x	x	Dap-Stat
67	1	1	valveUtilPct	x	x	x	x	x	Dap-Stat
68	1	1	humUtilPCT	x	x	x	x	x	Dap-Stat
69	1	1	alrm_select_1	x	x	x	x	x	Dap-Stat
70	1	1	alrm_select_2	x	x	x	x	x	Dap-Stat
71	1	1	alrm_select_3	x	x	x	x	x	Dap-Stat
72	1	1	alrm_select_4	x	x	x	x	x	Dap-Stat
73	3	56	bitErrors	x	x	x	x	x	Dap-Stat
185	1	1	runtime_c1	x	x	x	x	x	Dap-Xtra
186	1	1	runtime_c2	x	x	x	x	x	Dap-Xtra
187	1	1	runtime_c3	x	x	x	x	x	Dap-Xtra
188	1	1	runtime_c4	x	x	x	x	x	Dap-Xtra
189	1	1	runtime_ht1	x	x	x	x	x	Dap-Xtra
190	1	1	runtime_ht2	x	x	x	x	x	Dap-Xtra
191	1	1	runtime_ht3	x	x	x	x	x	Dap-Xtra
192	1	1	runtime_hum	x	x	x	x	x	Dap-Xtra
193	1	1	runtime_evap	x	x	x	x	x	Dap-Xtra
194	1	1	runtime_cond	x	x	x	x	x	Dap-Xtra
195	1	1	runtime_dehum	x	x	x	x	x	Dap-Xtra
196	1	1	runtime_esc	x	x	x	x	x	Dap-Xtra
197	1	1	runtime_cwc	x	x	x	x	x	Dap-Xtra
198	4	20	errors	x	x	x	x	x	Dap-Xtra
218	6	10	errage	x	x	x	x	x	Dap-Xtra
258	1	1	version	x	x	x	x	x	Dap-Xtra
259	2	1	tmbmair	x	x	x	x	x	Dap-Xtra
260	2	1	tmbhum	x	x	x	x	x	Dap-Xtra
261	1	1	tmbairdb	x	x	x	x	x	Dap-Xtra
				x	x	x	x	x	
262	6	13	runtimes (as array)	x	x	x	x	x	Dap-Xtra
314	3	80	bitErrors	x	x	x	x	x	Dap-Xtra
474	4	1	adj_rate	X	X	X			Dap-Menu
478	1	1	alrm_delay_1	X	X	X			Dap-Menu
479	1	1	alrm_delay_2	X	X	X			Dap-Menu
480	1	1	alrm_delay_3	X	X	X			Dap-Menu
481	4	1	alrm_enable_1	X	X	X			Dap-Menu
482	4	1	alrm_enable_2	X	X	X			Dap-Menu
483	4	1	alrm_enable_3	X	X	X			Dap-Menu
484	4	1	alrm_select_1	X	X	X			Dap-Menu
485	4	1	alrm_select_2	X	X	X			Dap-Menu
486	4	1	alrm_select_3	X	X	X			Dap-Menu

487	1	1	chilled_water	X		X	X	Dap-Menu
488	1	1	compressor_config	X		X	X	Dap-Menu
489	1	1	c_mode	X		X	X	Dap-Menu
490	2	1	fire_lim	X		X	X	Dap-Menu
491	1	1	heat_strip_config	X		X	X	Dap-Menu
492	1	1	hi_cal	X		X	X	Dap-Menu
493	1	1	hi_h_cal	X		X	X	Dap-Menu
494	2	1	hi_t_lim	X		X	X	Dap-Menu
495	1	1	humid_config	X		X	X	Dap-Menu
496	1	1	h_calib	X		X	X	Dap-Menu
497	1	1	lead_lag	X		X	X	Dap-Menu
498	1	1	loc_h_deadband	X		X	X	Dap-Menu
499	1	1	loc_h_setpt	X		X	X	Dap-Menu
500	2	1	loc_t_dband	X		X	X	Dap-Menu
501	2	1	loc_t_setpt	X		X	X	Dap-Menu
502	1	1	lo_cal	X		X	X	Dap-Menu
503	1	1	lo_h_lim	X		X	X	Dap-Menu
504	2	1	lo_t_lim	X		X	X	Dap-Menu
505	1	1	main_int	X		X	X	Dap-Menu
506	1	1	passwd_a	X		X	X	Dap-Menu
507	1	1	passwd_b	X		X	X	Dap-Menu
508	1	1	rst_mode	X		X	X	Dap-Menu
509	1	1	s_delay	X		X	X	Dap-Menu
510	2	1	t_calib	X		X	X	Dap-Menu
511	1	1	voice	X		X	X	Dap-Menu
512	1	1	vvrg	X		X	X	Dap-Menu
513	1	1	cat1	X		X	X	Dap-Menu
514	1	1	cat2	X		X	X	Dap-Menu
515	1	1	cat3	X		X	X	Dap-Menu
516	2	1	d_calib	X		X	X	Dap-Menu
517	2	1	lo_d_lim	X		X	X	Dap-Menu
518	1	1	ptc	X		X	X	Dap-Menu
519	2	1	supplyT		x	x	x	Chiller-Stat
520	2	1	returnT		x	x	x	Chiller-Stat
521	4	1	coolOn1		x	x	x	Chiller-Stat
522	4	2	coolOn2		x	x	x	Chiller-Stat
523	4	2	coolOn3		x	x	x	Chiller-Stat
524	2	1	valvePct		x	x	x	Chiller-Stat
525	4	1	pumpsOn		x	x	x	Chiller-Stat
526	4	1	condOn		x	x	x	Chiller-Stat
527	4	1	modFail		x	x	x	Chiller-Stat
528	2	1	hiSupT		x	x	x	Chiller-Stat
529	2	1	loSupT		x	x	x	Chiller-Stat
530	2	1	hiRetT		x	x	x	Chiller-Stat
531	2	1	loRetT		x	x	x	Chiller-Stat
532	1	1	csUtilPct1		x	x	x	Chiller-Stat

533	1	1	csUtilPct2	x	x	x	Chiller-Stat
534	1	1	csUtilPct3	x	x	x	Chiller-Stat
535	1	1	valveUtilPct	x	x	x	Chiller-Stat
536	3	48	errors	x	x	x	Chiller-Stat
584	1	1	mode	x	x	x	Chiller-Stat
585	6	11	runtimes	x	x	x	Chiller-Xtra
596	3	80	errold	x	x	x	Chiller-Xtra
676	6	10	errage	x	x	x	Chiller-Xtra
854	4	1	adjust_rate	x	x	x	Chiller-Menu
855	4	1	auto_ack	X	X	X	Chiller-Menu
856	2	1	aux_setpt	X	X	X	Chiller-Menu
857	4	1	backup_mods	X	X	X	Chiller-Menu
858	2	1	backup_setpt	X	X	X	Chiller-Menu
859	4	1	cmota	X	X	X	Chiller-Menu
860	4	1	comp_type	X	X	X	Chiller-Menu
861	4	1	ptc	X	X	X	Chiller-Menu
862	2	1	hi_r_lim	X	X	X	Chiller-Menu
863	2	1	hi_s_lim	X	X	X	Chiller-Menu
864	4	1	LL_policy	X	X	X	Chiller-Menu
865	2	1	lo_r_lim	X	X	X	Chiller-Menu
866	2	1	lo_s_lim	X	X	X	Chiller-Menu
867	1	1	main_int	X	X	X	Chiller-Menu
868	4	1	mods_configd	X	X	X	Chiller-Menu
869	1	1	network_ID	X	X	X	Chiller-Menu
870	1	1	op_1_delay	X	X	X	Chiller-Menu
871	1	1	op_2_delay	X	X	X	Chiller-Menu
872	4	1	op_1_message	X	X	X	Chiller-Menu
873	4	1	op_2_message	X	X	X	Chiller-Menu
874	1	1	password	X	X	X	Chiller-Menu
875	4	4	relay_mask_0	X	X	X	Chiller-Menu
879	4	4	relay_mask_1	X	X	X	Chiller-Menu
883	4	4	relay_mask_2	X	X	X	Chiller-Menu
887	4	1	restart_mode	X	X	X	Chiller-Menu
888	4	1	reverse_valve	X	X	X	Chiller-Menu
889	4	1	sc_alarm_on	X	X	X	Chiller-Menu
890	1	1	start_delay	X	X	X	Chiller-Menu
891	1	1	supply_dband	X	X	X	Chiller-Menu
892	2	1	supply_setpt	X	X	X	Chiller-Menu
893	4	1	temp_scale	X	X	X	Chiller-Menu
894	4	1	valve_voltage	X	X	X	Chiller-Menu
895	4	1	voice	X	X	X	Chiller-Menu
896	4	1	water_valve	X	X	X	Chiller-Menu
897	1	1	return_cal	X	X	X	Chiller-Menu
898	1	1	supply_cal	X	X	X	Chiller-Menu

899	4	1	adj_rate	X	X	X	X	Dap80-Menu
900	1	1	alrm_delay_1	X	X	X	X	Dap80-Menu
901	1	1	alrm_delay_2	X	X	X	X	Dap80-Menu
902	1	1	alrm_delay_3	X	X	X	X	Dap80-Menu
903	1	1	alrm_delay_4	X	X	X	X	Dap80-Menu
904	4	1	alrm_select_1	X	X	X	X	Dap80-Menu
905	4	1	alrm_select_2	X	X	X	X	Dap80-Menu
906	4	1	alrm_select_3	X	X	X	X	Dap80-Menu
907	4	1	alrm_select_4	X	X	X	X	Dap80-Menu
908	4	1	ant-enable	X	X	X	X	Dap80-Menu
909	4	1	autoflush_time	X	X	X	X	Dap80-Menu
910	4	1	auto_ack	X	X	X	X	Dap80-Menu
911	4	1	comp_config	X	X	X	X	Dap80-Menu
912	4	1	control_type	X	X	X	X	Dap80-Menu
913	4	1	c_mode	X	X	X	X	Dap80-Menu
914	4	1	da_volts	X	X	X	X	Dap80-Menu
915	4	1	dehum_on	X	X	X	X	Dap80-Menu
916	2	1	d_calib	X	X	X	X	Dap80-Menu
917	4	1	esaver_supp_comp	X	X	X	X	Dap80-Menu
918	2	1	fire_lim	X	X	X	X	Dap80-Menu
919	4	1	heater_config	X	X	X	X	Dap80-Menu
920	1	1	hi_h_lim	X	X	X	X	Dap80-Menu
921	2	1	hi_t_lim	X	X	X	X	Dap80-Menu
922	4	1	humid_config	X	X	X	X	Dap80-Menu
923	2	1	h_calib	X	X	X	X	Dap80-Menu
924	1	1	h_dband	X	X	X	X	Dap80-Menu
925	4	1	lead_lag	X	X	X	X	Dap80-Menu
926	2	1	lo_d_lim	X	X	X	X	Dap80-Menu
927	1	1	lo_h_lim	X	X	X	X	Dap80-Menu
928	2	1	lo_t_lim	X	X	X	X	Dap80-Menu
929	2	1	main_int	X	X	X	X	Dap80-Menu
930	1	1	network_id	X	X	X	X	Dap80-Menu
931	2	1	nom_h_setpt	X	X	X	X	Dap80-Menu
932	1	1	password	X	X	X	X	Dap80-Menu
933	4	1	ptc	X	X	X	X	Dap80-Menu
934	3	16	relay_1_mask_0	X	X	X	X	Dap80-Menu
950	3	16	relay_1_mask_1	X	X	X	X	Dap80-Menu
966	3	16	relay_1_mask_2	X	X	X	X	Dap80-Menu
982	3	16	relay_2_mask_0	X	X	X	X	Dap80-Menu
998	3	16	relay_2_mask_1	X	X	X	X	Dap80-Menu
1014	3	16	relay_2_mask_2	X	X	X	X	Dap80-Menu
1030	3	16	relay_3_mask_0	X	X	X	X	Dap80-Menu
1046	3	16	relay_3_mask_1	X	X	X	X	Dap80-Menu
1062	3	16	relay_3_mask_2	X	X	X	X	Dap80-Menu
1078	4	1	reverse_valve	X	X	X	X	Dap80-Menu
1079	4	1	rst_mode	X	X	X	X	Dap80-Menu
1080	4	1	sc_alarms	X	X	X	X	Dap80-Menu

1081	1	1	s_delay	X	X	X	X	Dap80-Menu
1082	2	1	t_calib	X	X	X	X	Dap80-Menu
1083	1	1	t_dband	X	X	X	X	Dap80-Menu
1084	2	1	t_setpt	X	X	X	X	Dap80-Menu
1085	4	1	valve_config	X	X	X	X	Dap80-Menu
1086	4	1	voice	X	X	X	X	Dap80-Menu
1100	1	1	sensor_1_name			x	x	Dap80-Analog
1101	1	1	sensor_1_units			x	x	Dap80-Analog
1102	1	1	sensor_1_type			x	x	Dap80-Analog
1103	1	1	sensor_1_min_val			x	x	Dap80-Analog
1104	1	1	sensor_1_max_val			x	x	Dap80-Analog
1105	1	1	sensor_1_cal			x	x	Dap80-Analog
1106	1	1	sensor_2_name			x	x	Dap80-Analog
1107	1	1	sensor_2_units			x	x	Dap80-Analog
1108	1	1	sensor_2_type			x	x	Dap80-Analog
1109	1	1	sensor_2_min_val			x	x	Dap80-Analog
1110	1	1	sensor_2_max_val			x	x	Dap80-Analog
1111	1	1	sensor_2_cal			x	x	Dap80-Analog
1112	1	1	sensor_1_input			x	x	Dap80-Channels
1113	1	1	sensor_2_input			x	x	Dap80-Channels
1114	1	1	sensor_3_input			x	x	Dap80-Channels

4.4.6 Unit Types

When the driver reads everything from a device it must first obtain the device's unit type so that it can determine what other data is available. Once the unit type is obtained then the driver updates the 'Unit-Type' field visible on the node screen of the RUIDebug program. The unit type is also available in the data array defined in table 4.4.5.1.

The following table lists the unit type that can be processed by this driver.

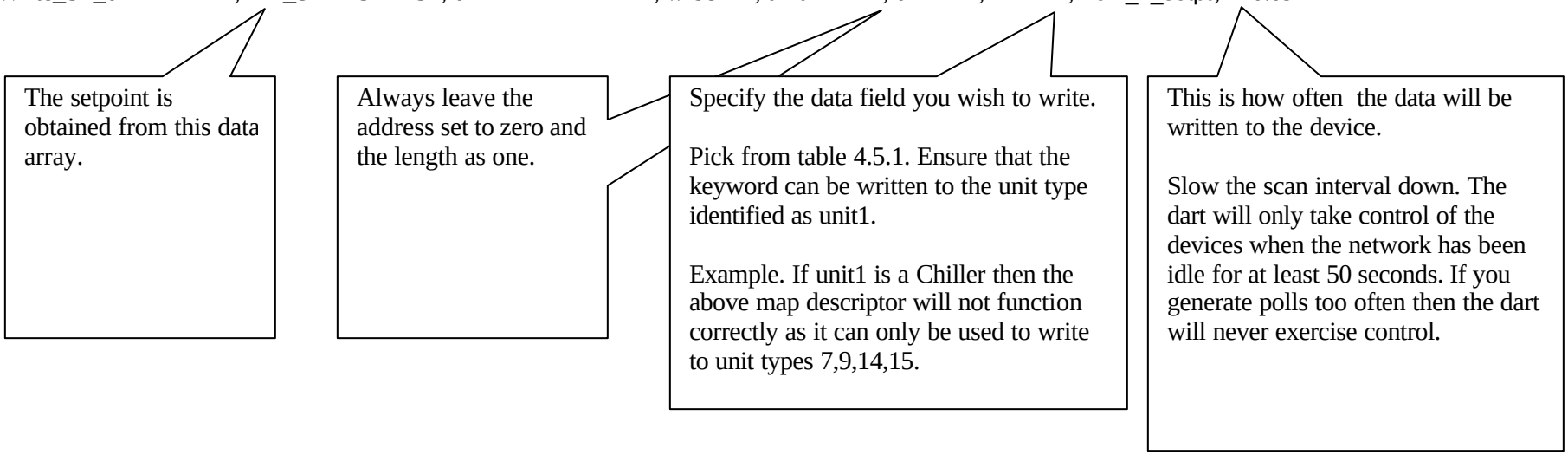
Unit Type	Numeric Unit Type	Description
"-"	0	Unknown/unavailable/un-initialized
"1"	1	044 data logger
"2"	2	046 expanded DAP
"3"	3	046 2 mod chiller
"4'	4	046 3 mod chiller
"5"	5	048 DAP, 80-character display
"6"	6	049 DAP, 16-character display

"7"	7	080 DAP II, no relay expansion
"8"	8	080 Chiller II
"9"	9	080 DAP II, with relay expansion
"10"	10	Not Defined
"11"	11	Not Defined
"12"	12	Not Defined
"13"	13	Not Defined
"E"	14	080 DAP II, with analog module
"F"	15	080 DAP II, with relay and analog

4.4.7 Map Descriptor: Example 2 – Writing a Set-Point .

A Data Aire device cannot be written to until it has been read. This is a limitation of the Data Aire protocol. This means that you configuration cannot consist only of wrbc map descriptors. It should consist of a at least a map descriptor like example 1for every unit that you wish to write to.

```
Map_Descriptor_Name, Data_Array_Name, Data_Array_Offset, Function, node_name, Address, Length, DA_Field     , Scan_Interval
Write_SP_01         , DA_SETPOINTS  , 0                , wrbc    , unit1    , 0      , 1     , nom_h_setpt, 120.0s
```



Reccomendation

Use wrbx instead of wrbc. This causes the driver to generate the command message only when the setpoint changes which in turn minimizes communications. Using this method the DART's control mode is interrupted the least.

4.4.8 Map Descriptor: Example 3 – Writing multiple points using one message.

```
Map_Descriptor_Name, Data_Array_Name     , Data_Array_Offset, Function,  node_name, Address, Length, DA_Field       , DA_Assoc, Scan_Interval
Write_MapDesc_1, DA_ SETPOINTS, 0                , wrbc,       unit1 ,   0        1 ,     nom_h_setpt , 2          , 120.0s
Write_MapDesc_1, DA_SETPOINTS , 1                , passive,    unit1 ,   0        1 ,     t_setpt       , 2
```

A 'Write' and a passive to the same node. This write will update two fields, the nom_h_setpt and the t_setpt.

The write MapDesc. Must precede the passive.

This method is only appropriate if you plan to continuosly write to the devices. If you intend to write on change using the wrbx function then use multiple map descriptors that are not associated and give each one a wrbx.

Associate the passive map descriptor to the active (wrbc) map descriptor. In this way the driver will use only one message to write to the device. The message will be built using both map descriptors. This method reduces the communication load.

The association is made using the DA_Assoc parameter. Use unique positive integers. Omitting the DA_Assoc parameter when using 'passive' map descriptors will produce ambiguous results.

5 Chapter 5

This Chapter is blank.

6 Advanced Topics

6.1 Additional Driver Specific Map Descriptor Parameters

What happens if you want to poll for one specific type of data more frequently than others ? What if you want more control of the location of where data is stored ? What happens if you want to do diagnostic polls … ?

To be able to achieve solutions to any of these types of questions the driver offers advanced configuration by adding to and extending the map descriptors specific to the Dart Serial Driver.

Column Title	Function	Legal Values
DA_Func*	Specifies the Data Aire Command/Query function to be used. Use a function appropriate to the type of slave (DAP/Chiller/DAP80) and the type of data required.	Numeric/Text. See section 6.1.1 for a list of possible values.
DA_Field*	Specifies the data field to be retrieved from the slave device. Salves are only capable of responding with a data composite consisting of many data fields. You use this parameter to specify which parameter is you wish to have extracted from the data composite. Note[1].	This is a text field. See section 6.1.2 for a list of possible values as well as Table 4.4.5.1
DA_Assoc*	Use to associate passive map descriptors with an active map descriptor. In some case you may have a read (rdbc) addressing the same node as a write (wrbc). Both the read and write may have associated map descriptors. This field is used to make the association. Give the rdbc & passives map descriptors associated with the rdbc the same value (any number) and give the wrbc and its passive map descriptors	Any positive integer.

	another value for DA_Assoc.	
Da_Freq	Used only for connection to DART's. Specify in milliseconds the interval at which you want a wrbc/rdbc map descriptor to be executed. When using wrbc/rdbc's to a dart device set the scan interval to 5.0s and set this parameter to a number greater than 180000 (3 minutes). An interval of 300000 (5 minutes is recommended).	
DA_Method$^{\Psi}$	Specifies the extraction method. Such as Hex-ASCII to decimal number in 10's of a degree,	See section 6.3 for a list of possible values.
DA_Bytcnt$^{\Psi}$	Specifies the number of bytes that are to be processed by the method specified above. For method#6 which processes an array of elements the DA_Bytcnt specifies the number of bytes that constitute each element of the array.	>= 1
DA_Offset$^{\Psi}$	An offset into the data composite that is returned when the slave is polled. The offset is the number of bytes from the first data byte.	0 to the length of the data composite. No validation is performed.
DA_Elecnt$^{\Psi}$	Number of elements that are produces by the extraction method.	>= 1

Ψ:
These parameters are only required for custom data extractions not provided for with DA_Field parameter.

6.1.1 DA_Func Parameter - Permitted values.

The driver supports a limited subset of the Dart Poll & Response Functions. The selection of the sub-set is based on the identification of useful & practical functions.

IN addition to the 'Everything' keyword indicated in chapter 4 the following specific query functions are implemented.

Func.	Description	Driver Parameter	Protocol Id.
'1'	DART Config Query	DA_Func = dart-config	49
'2'	Dart Psswd Query	DA_Func = dart-password	50
'3'	DAP Config Command	DA_Func = dap-config	51
'4'	DAP Log Query	DA_Func = dap-log	52
'5'	DAP Unit-Type Query	DA_Func = dap-unit	53
'6'	DAP Stat Query	DA_Func = dap-stat	54
'7'	DAP Xtra Query	DA_Func = dap-xtra	55
'8'	DAP Menu Query	DA_Func = dap-menu	56
'A'	Chiller Stat Query	DA_Func = chiller-stat	65
'B'	Chiller Xtra Query	DA_Func = chiller-xtra	66
'C'	Chiller Menu Query	DA_Func = chiller-menu	67
'D"	Dart Status	DA_Func = dart-status	68
'E'	DAP80 Menu Query	DA_Func = dap80-menu	69
'G'	DAP Analog Query	DA_Func = dap80-analog	71
'H'	DAP Channels Query	DA_Func = dap80_channles	72

Each of the above queries returns a complex set of data consisting of many sub-fields. Contact Data-Aire for a complete listing of the data composite returned.

The following special / diagnostic functions are also implemented.

Driver Parameter	Protocol Id.
DA_Func = All-Listen	11
DA_Func = Ack	6
DA_Func = Dart-Transparant	2
DA_Func = Dart-Opaque	3
DA_Func = Test-Echo	16
DA_Func = Test-No-Echo	15
DA_Func = Unit-Talk	13

With the exception of the Unit-Talk command, these are nodeless commands.

When using any of these special commands no other DA_* fields need be specified.

The operation of these functions is as follows ;

All-Listen instructs the all units in the network to switch their relays to the listen position. Those units already in the listen position will do nothing. Those in the talk position will first echo the all-listen command and then switch their relays to the listen position. A pause of 0.15 seconds is required after the transmission of this command, to allow the units time to switch their mechanical relays.

6.1.2 DA_Field Parameter - Permitted values.

Legal values depend on the value of DA_Func.

DA_Field Legal Values	Description	Data Format	DA_Func
All	The whole data record returned by the slave is stored in the data array byte for byte. The number of bytes written is dependent of the DA_Func. DA_Func=dart-password Bytes =231 DA_Func=dart-config Bytes =41 DA_Func=dap-config Bytes =4 DA_Func=dap-log Bytes =240 DA_Func=dap-unit Bytes =1 DA_Func=dap-stat Bytes =68 DA_Func=dap-xtra Bytes =124 DA_Func=dap-menu Bytes =103 DA_Func=chiller-stat Bytes =54 DA_Func=chiller-xtra Bytes =104 DA_Func=chiller-menu Bytes =89 DA_Func=dap80-menu Bytes =138 DA_Func=dart-status Bytes = 9 DA_Func=dap80-analog Bytes =36 DA_Func=dap80-channels Bytes=12	Bytes	
Special	Indicates that a user defined extraction is specified in the map descriptor. When this value is specified as the DA_Field value then DA_Method,DA_Bytcnt,DA_Offset,DA_Elecnt must also be specfied.		
See Table 4.4.5.1 for all other keywords.			

6.1.3 DA_Method Parameter Values and Notes

The DA_Method specifies a method for interpreting a range of bytes when the DA_Field=special.

Method 1:

Each byte is valid when its contains only one of the following ASCII characters.

{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F }

Each byte being parsed is considered to be a hexadecimal digit. The most significant digit is the left most byte.

Thus the four bytes
30 31 32 33 (hex) are interpreted by regarding the ASCII value of each byte as a hexadecimal digit. Thus we interpret the 4 bytes as the hexadecimal number 0123 and the decimal value is equal to 291.

Method 2

This method is the same as method one but is used for humidity's and temperatures which are transmitted as the number of tenths of a unit.

Thus in the example of method 1. The 4 bytes yield the decimal number 29.1 °F/%.

Method 3

Each byte is regarded as containing a hexadecimal digit in ASCII format.

Example: incoming byte contains 41(hex). -> regard as the hexadecimal digit 'A' in ASCII format.

The method then converts the hex digit to a series of 8 bits. In this example the bits are 00001010 with the msb being the left most.

Method 4

Each byte is regarded as containing a hexadecimal digit in ASCII format.

Example: incoming byte contains 41(hex). -> regard as the hexadecimal digit 'A' in ASCII format. The decimal value of the this digit is written to the data array. In this example the number 10 would be written to the data array.

Method 5

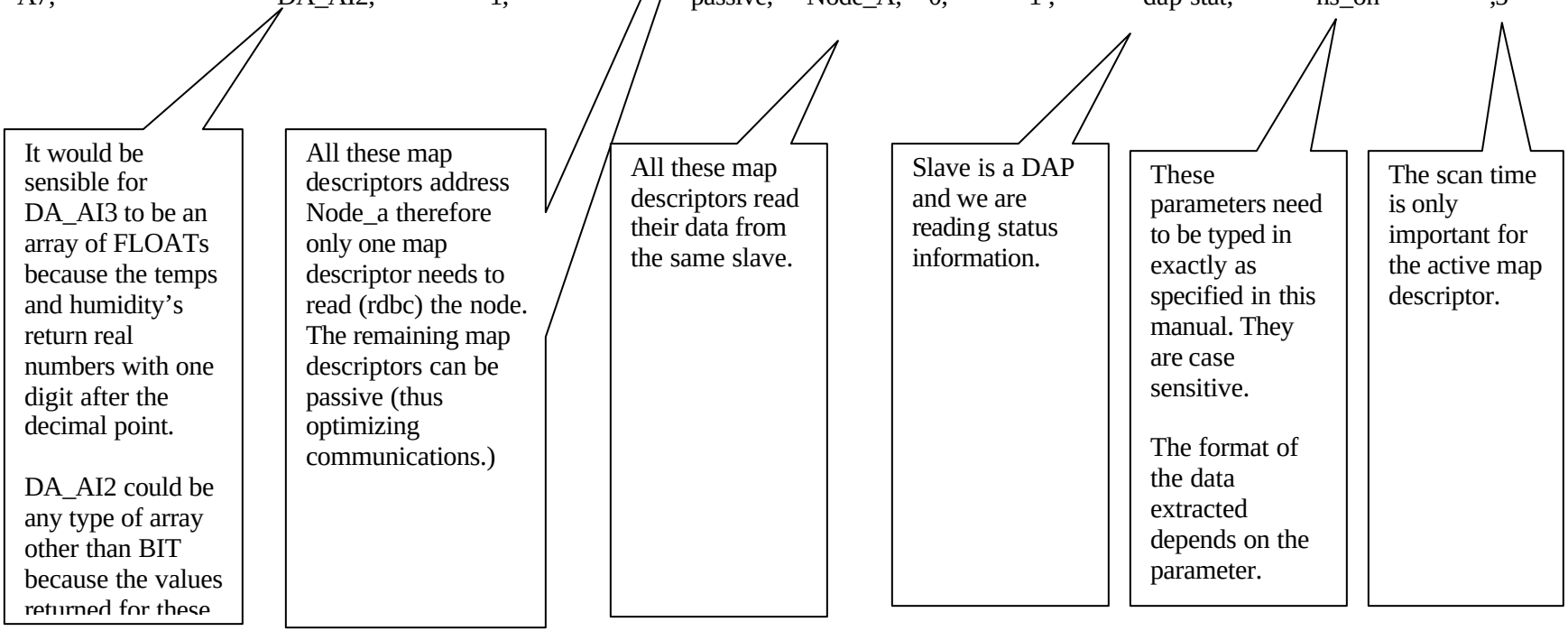
There is no translation. The raw bytes are written to the data array.

Method 6

Processes an array of elements using method 1 translation. The raw data being parsed is considered to consist of DA_elecnt elements each consisting of DA_bytecnt bytes. Method 1 is applied to each cluster of bytes.

6.1.4 Advanced Example 1 :

```
Map_Descriptor_Name, Data_Array_Name, Data_Array_Offset, Function, node_name, Address, Length, DA_Func, DA_Field, Scan_Interval
A1,                  DA_AI3,          0,                 rdbc,     Node_A,    0,       1 ,     dap-stat,  temp      ,5
A2,                  DA_AI3,          1,                 passive,  Node_A,    0,       1 ,     dap-stat,  hum       ,5
A3,                  DA_AI3,          2,                 passive,  Node_A,    0,       1,      dap-stat,  d_temp    ,5
A4,                  DA_AI3,          3,                 passive,  Node_A,    0,       1 ,     dap-stat,  hiTemp    ,5
A5,                  DA_AI3,          4,                 passive,  Node_A,    0,       1 ,     dap-stat,  loTemp    ,5
A6,                  DA_AI2,          0,                 passive,  Node_A,    0,       1 ,     dap-stat,  cs_on     ,5
A7,                  DA_AI2,          1,                 passive,  Node_A,    0,       1 ,     dap-stat,  hs_on     ,5
```



6.1.5 AdvancedExample 2

The DAP-II Status query returns 14 bytes of errors & status information. The arrangement and meaning of these bytes is defined by the Data Aire Corporation and is also dependent on the type of DAPII module being polled.

```
Map_Descriptor_Name, Data_Array_Name, Data_Array_Offset, Function, node_name, Address, Length, DA_Func, DA_Field,  Scan_Interval
A1,                  DA_AI1,          0,                 rdbc,     Node_A,    0,       1 ,     dap-stat,  errors     ,5
A2,                  DA_DI1,          1,                 passive,  Node_A,    0,       1 ,     dap-stat,  bitErrors  ,5
```

The 'errors' key word returns 14 bytes, thus we DA_AI1 should be a BYTE array. Each byte will have values 0-15 to represent the value of the bits in each byte.

The bitErrors extracts the same data from the DAP but presents it as a series of 14x8 bits. Thus make

Slave is a DAP and we are reading status information.

Data arrangement and meaning of each error is defined by Data Aire Corp.

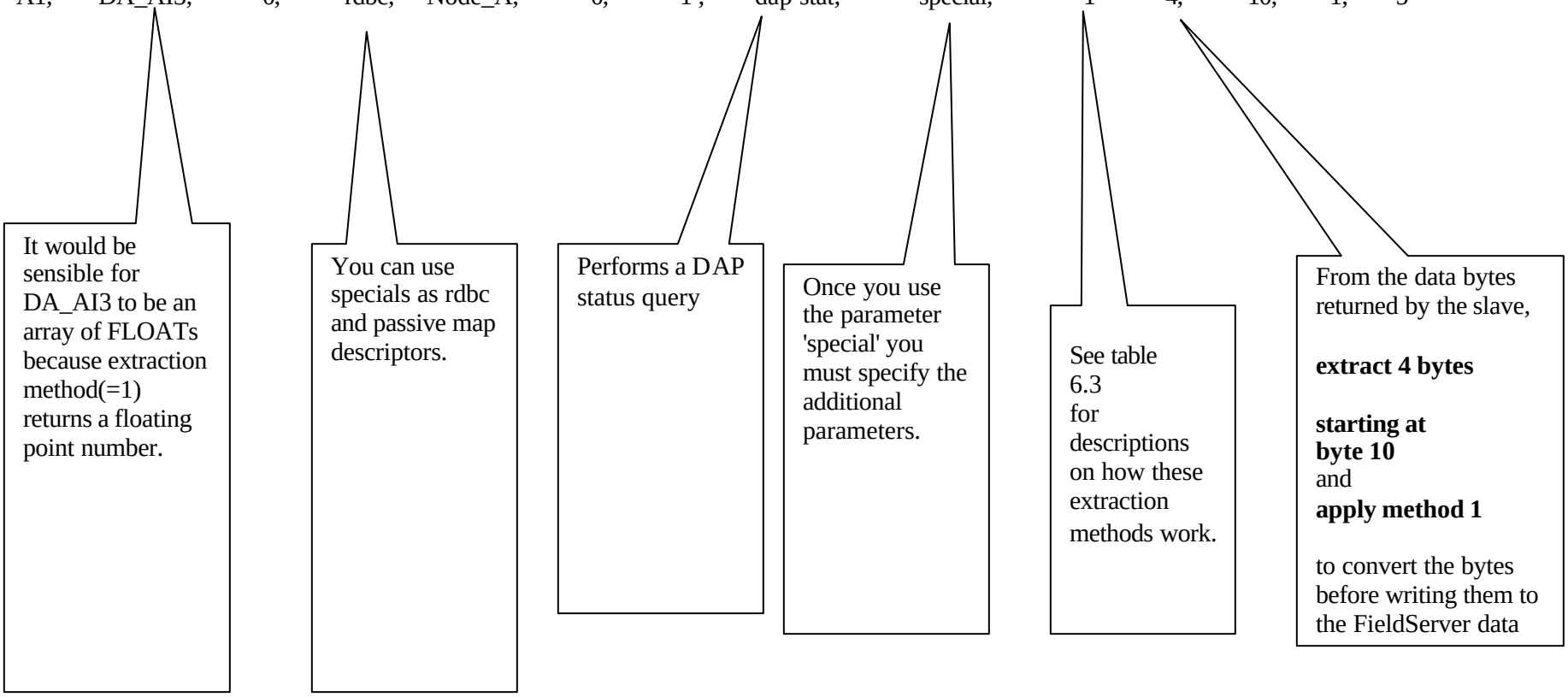
Example: Bit 24 is a LOW TEMP WARNING for DAPII-044/8/9 units.

Example: Bit 09 is a HUMIDITY SENSOR PROBLEM for a DAPII-080 unit.

BitErrors is a synonym for errors. The data is extracted using a different data format.

6.1.6 Advanced Map Descriptor: Example 3 - Using the 'special' parameter.

```
Map_Descriptor_Name, Data_Array_Name, Data_Array_Offset, Function, node_name, Address, Length, DA_Func, DA_Field, DA_Method
,DA_Bytcnt, DA_Offset, DA_Elecnt     Scan_Interval
A1,      DA_AI3,      0,      rdbc,    Node_A,      0,      1 ,     dap-stat,      special,      1      4,      10,      1,      5
```



6.1.7 Advanced Map Descriptor: Example 4 - Using the 'DA_Assoc' parameter.

```
Map_Descriptor_Name, Data_Array_Name, Data_Array_Offset,  Function,    node_name, Address, Length,  DA_Func,     DA_Field,      DA_Assoc, Scan_Interval
A1,                  DA_AI3,          0,                 rdbc,        Node_A,    0,       1 ,   dap80-menu,  All            ,    1         ,5
A2,                  DA_AI3,          1,                 passive,     Node_A,    0,       1 ,   dap80-menu,  alrm_delay_1,    1         ,5
A3,                  DA_AI3,          2,                 passive,     Node_A,    0,       1,    dap80-menu,  alrm_delay_2,    1         ,5
A4,                  DA_AI3,          3,                 passive,     Node_A,    0,       1 ,   dap80-menu,  alrm_delay_3,    1         ,5
A8,                  DA_AI4,          0,                 wrbc,        Node_A,    0,       1 ,   dap80-menu,  nom_h_setpt,     2         ,5
A9,                  DA_AI4,                             passive,     Node_A,    0,       1 ,   dap80-menu,  t_setpt ,                2,        5
```

A 'Read' and some passive map descriptors to extract other data fields from the same read. (optimizes communications)

Read must precede the passive's.

A 'Write' and a passive to the same node. This write will update two fields, the nom_h_setpt and the t_setpt.

The write map desc. Must precede the passive.

Potential confusion for the FieldServer because the node and the DA_Func's are the same for all the map descriptors.

Solve this problem using DA_Assoc.

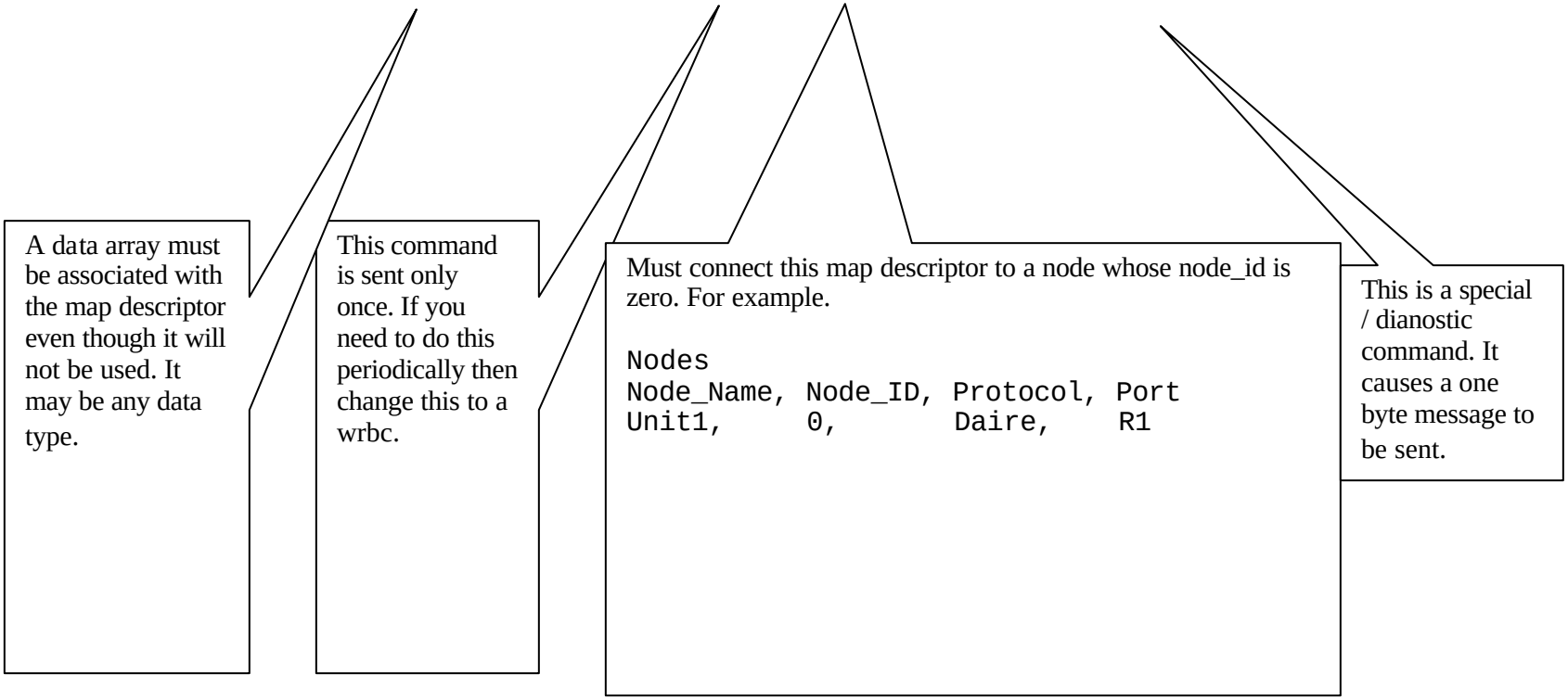
DA_Assoc associates the passives with the correct active map descriptor.

Thus Map Descriptors A2,3,4 are associated with A1 because the value of DA_Assoc=1 for all these map descriptors.

Thus Map DescriptorsA89is associated with A8 because the value of DA_Assoc=2 for both these map descriptors.

6.1.8 Map Descriptor: Example 5 - Using a special / diagnostic command.

```
Map_Descrip tor_Name, Scan_Interval, Data_Array_Name      , Data_Array_Offset,  Function,    node_name, Address, Length,  DA_Func,
A1,                   1.0s,          UNUSED_ARRAY,       0                   , wrb      ,    No_node, 0,       1 ,      All-Listen,
```



6.2 Related Documents

The driver as specified in this manual is based on Data Aire Poll and Response Protocol Revision 3.2 dated 4 Nov 1997.

The driver is compliant with a later release of the specification Revision 3.7 - 21 JUN 00.

6.3 Troubleshooting Tips

6.3.1 Bad Values

In the event that the driver cannot correctly decode the raw bytes it will generally write an value which indicates bad data. In most cases the indicating value is -1 or 65535 (depending on data type). When setting bits for status fields the driver will not write new data to the array if the incoming byte is invalid. Look in the error log for indication of this type of problem.

Example. Valid ASCII digits are 0..9,A..F. If a byte is being parsed and an hex digit is expected but not found then the driver considers this an errors and writes the bad value indication OR produces an error message when the bad value indication cannot be used.

6.3.2 Dead Nodes

When a node is absent or dies it is possible that the Dart may go idle and stop communicating with all other nodes. This problem is not related to the driver but to the Data Aire devices.

6.3.3 Ignored Messages

The driver reports ignored messages. These are messages sent by a DAP/DART for which the driver cannot find a map descriptor to store the message. This does not mean the driver is not working. It means that a message which contains data that the driver/you are not interested in is being discarded.

The current version of the driver ignores a few messages relating to the status of the DART device. Later versions of the driver will be capable of storing these messages and the number of ignored messages will decrease.

6.4 Writing data to Dap Devices

The variables in a DAP device are not individually addressable. When a DAP device is read a data composite is returned. The driver extracts the data you require. When data is written to a DAP device it is not possible to write a value to one individual data element such as a temperature setpoint. Rather, the DAP devices requires the complete data composite (all its variables, states, settings) be written at once. This makes the setting of a setpoint a complex operation for the driver.

It involves the following steps

i. Read the device, obtain a complete set of data and store (internally in the driver.).
ii. Use this stored data to form the basis for a write. Modify the data with the data the user wishes to set.
iii. Write the modified data composite back to the DAP device.

You can see that to complete a 'write' operation successfully, we must first read the device successfully. If the read has not been completed then the write operation will be abandoned. The driver prints messages to the error logs and records a NODE_OFFLINE stat each time that it attempts to write but is unable to.

The DAP-Config command is an exception to the above notes. A read is not required. The Dap Config command is used to turn off/on DAP units. The Command uses two consecutive array elements. The first is the zone, the second is the inhibit command. Valid zones are 0-63.

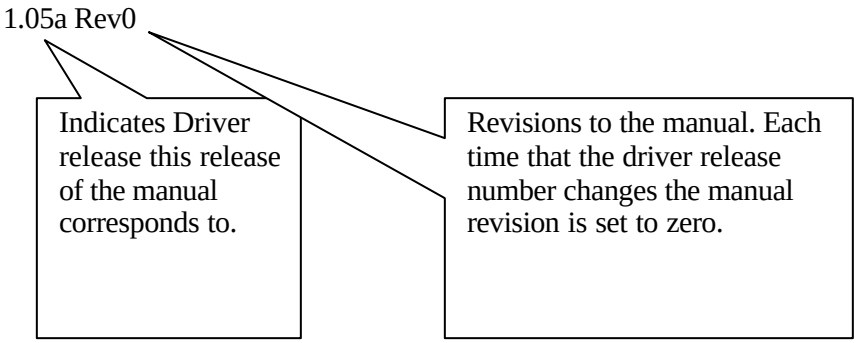
If any inhibit bit (bits 0-5) is set then the unit will not run. If bits 0 to 5 are off then the unit will run. Bits 6& 7 are used for display only on the DAP panel.

0x01 - Inhibit Cooling
0x02 - Inhibit Heating
0x04 - Inhibit Humid
0x08 - Inhibit Dehumidification
0x10 - Inhibit fan
0x20 - Reserved
0x40 - Network Standby - Display on panel (only has no effect on unit)
0x80 - Network off inhibit is active - display only - no effect on operation

This function should be used as a wrbc/x. When the driver encounters this command it reads the associated data array, loads the two elements found at the array offset into the message and transmits the message. The driver does not set any data array elements to confirm that the command concluded successfully.

7 Revision Change Notices

Revision Number Format Explained:



7.1 Rev1.06a-Rev0 Changes from previous releases

This is the first release of this driver since its separation from a combined Dart/Dap driver.

Previously this manual was identified as the “Data Aire user Manual Rev1.05a Rev2”