

HHT Interlock Status Controller

Introduction:**Interlock Status:**

The primary task performed by the Interlock Status Controller (ISC) is the monitor of all interlock implemented at the telescope. The status of the interlocks may be monitored on the dedicated display or through the serial interface to the terminal server. In addition to it's primary purpose, the ISC also performs several other tasks.

Enclosure Control:

The ICS has an integrated enclosure control terminal (ECT) which consists of a dedicated liquid crystal display and bank of buttons. This provides enclosure control in the control room.

Precipitation Monitor:

The precipitation monitor provides a 110VAC signal which is monitored by the ICS. This the status of this signal is available through the serial interface to the terminal server. The signal is used internally to override the enclosure controls to prevent opening the enclosure during a precipitation event. If the enclosure is open, the ISC emits an annoying sound until the enclosure is closed.

Wind Monitor:

The building motor current and Door and Roof hydraulic pressure is monitored for the purpose of detecting the impact of the wind on the structure. The current is read using an isolated ADC card.

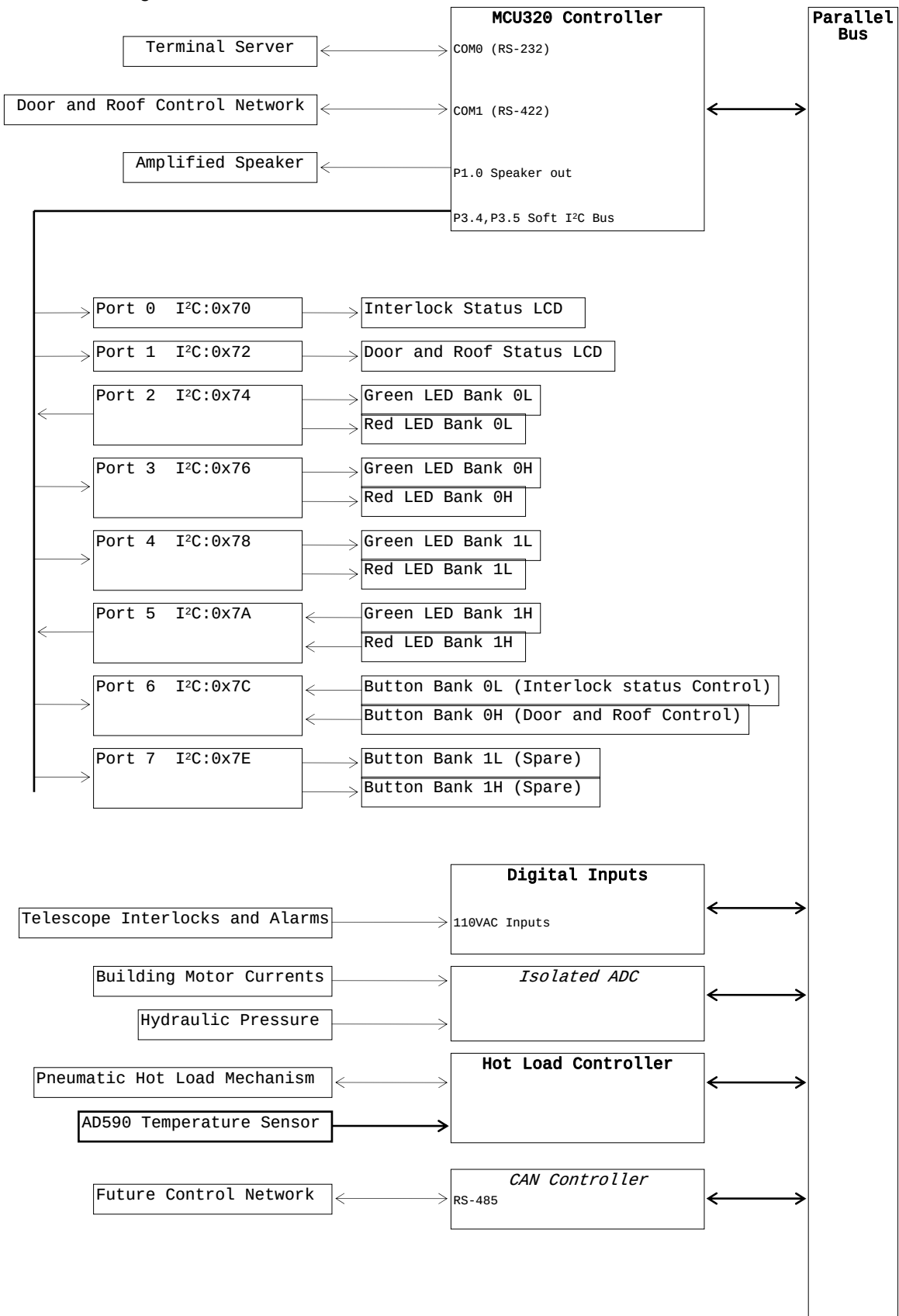
Hot Load Control:

Control of the pneumatic hot load is also provided for the purpose of computer controlled calibration. This function may be controlled through a dedicated button on the ICS and through the serial interface to the terminal server.

CAN controller:

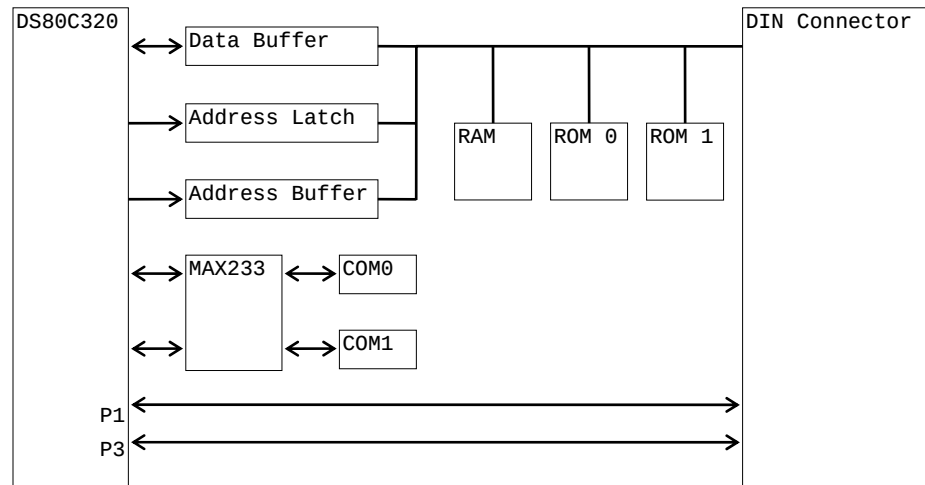
This controller is the first element in a Controller Area Network. Future controllers can be added to the network to provide expanded features.

ISC Block Diagram:



MCU320 Controller:

The ICS is built around the MCU320 controller board. This is a simple controller based on the Dallas Semiconductor 80C320. The 80C320 is for the most part compatible with the Intel 8032 controller (a member of the 8051 family). The MCU320 has 8k of RAM, 64k of ROM and two RS-232 interfaces.



DIN Connector:

PIN	Value	Description
A1	+5V	Digital Power
A2	A0	Address Bit 0
A3	A1	Address Bit 1
A4	A2	Address Bit 2
A5	A3	Address Bit 3
A6	A4	Address Bit 4
A7	A5	Address Bit 5
A8	A6	Address Bit 6
A9	A7	Address Bit 7
A10	A8	Address Bit 8
A11	A9	Address Bit 9
A12	A10	Address Bit 10
A13	A11	Address Bit 11
A14	A12	Address Bit 12
A15	A13	Address Bit 13
A16	A14	Address Bit 14
A17	A15	Address Bit 15
A18	P3.0	Port 3, Bit 0
A19	P3.1	Port 3, Bit 1
A20	P3.2	Port 3, Bit 2
A21	P3.3	Port 3, Bit 3
A22	P3.4	Port 3, Bit 4
A23	P3.5	Port 3, Bit 5
A24	RST	Active High Reset
A25	S1	Select Line 1
A26	S3	Select Line 3
A27	S5	Select Line 5
A28	S7	Select Line 7
A29	+12V	Analog Power
A30	AGND	Analog Ground
A31	-12V	Analog Power
A32	DGND	Digital Ground
C1	+5v	Digital Power

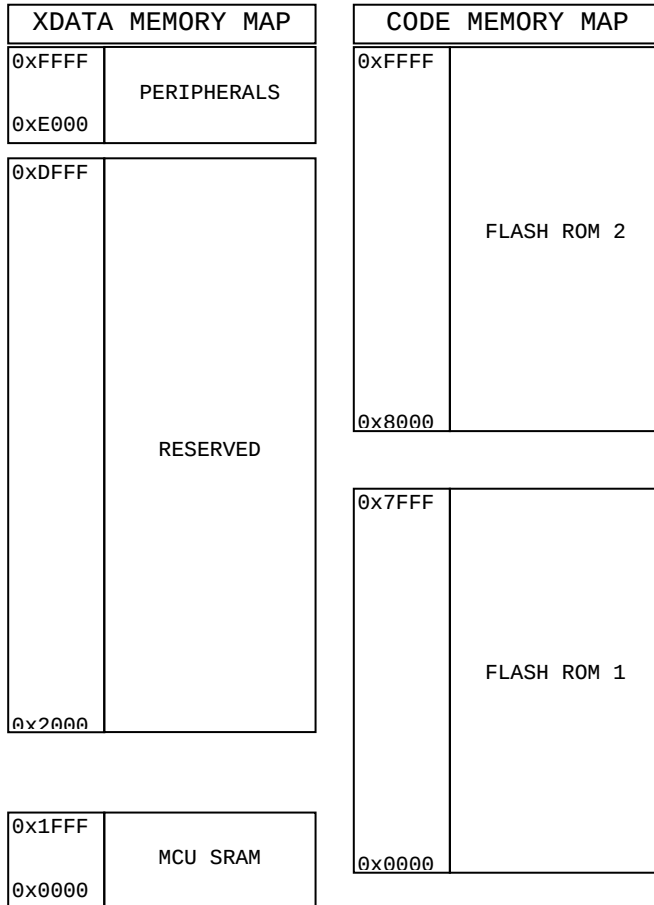
C2	D0	Data Bit 0
C3	D1	Data Bit 1
C4	D2	Data Bit 2
C5	D3	Data Bit 3
C6	D4	Data Bit 4
C7	D5	Data Bit 5
C8	D6	Data Bit 6
C9	D7	Data Bit 7
C10	ALE	Address Latch Enable
C11	PSEN	Program Select
C12	P1.0	Port 1, Bit 0
C13	P1.1	Port 1, Bit 1
C14	P1.2	Port 1, Bit 2
C15	P1.3	Port 1, Bit 3
C16	P1.4	Port 1, Bit 4
C17	P1.5	Port 1, Bit 5
C18	P1.6	Port 1, Bit 6
C19	P1.7	Port 1, Bit 7
C20	WR_BP	Back Plane Write
c21	RD_BP	Back Plane Read
c22	MON	Monitor
c23	WR	Write
c24	RD	Read
c25	S0	Select 0
c26	S2	Select 2
c27	S4	Select 4
c28	S6	Select 6
c29	+12V	Analog Power
c30	AGND	Analog Ground
c31	-12V	Analog Power
c32	DGND	Digital Ground

Memory Map for the MCU320 board:

The Intel 8051 MCUs and all derivatives are based on the Harvard architecture. This is to say that they have separate memory space for program (CODE) and external data (XDATA). The advantage to this is two fold. The address space is doubled and memory management is simplified in simple systems. The disadvantage is that no writes are allowed to code space. This was not a problem when PROM was used for program memory, but with the advent of FLASH ROM, the micro can benefit from the option to write to this nonvolatile memory. In addition, FLASH ROM is less expensive due to its packaging. To accommodate this and other features the MCU320 board supports three modes of addressing.

MODE 0:

Harvard architecture (Intel mode). The board has 8k of SRAM (or NVSRAM) and up to 64k of program memory in FLASH ROM. Although the program memory is flash, no write cycles are allowed to this space in this mode. Off board peripherals should be limited if possible to the top 8k of XDATA. The mid 48k of XDATA can be von Neumann (common code and data space) mapped on a monitor card for debugging so this space should not be used if the associated monitor card is used for the debug cycle. If one chooses to use a ROM emulator, this limitation is not applicable.



MODE 1:

von Neumann architecture (FLASH write mode). In this mode all peripherals are disabled by disabling the RD and WR lines on the back plane. All but the lower 8k of program memory is then von Neumann mapped to allow write cycles to this space. This mode is specifically used for the storage of data in non-volatile FLASH program memory. This mode can be optionally controlled by the MCU through any P1 pin. It is possible to use this mode in a stand alone boot-loader mode. It should be noted that if a boot-loader is put into the lower 8k of program memory, a jumper can be set on the board to allow in system firmware updates. If one of the P1 pins is used, the firmware may be updated over a network (CAN) through a modem or through any of the UARTs including the one on the monitor board.

XDATA MEMORY MAP	
0xFFFF	FLASH ROM 2
0x8000	
0x7FFF	
0x2000	FLASH ROM 1
0x1FFF	
0x0000	MCU SRAM

CODE MEMORY MAP	
0xFFFF	FLASH ROM 2
0x8000	
0x7FFF	
0x0000	FLASH ROM 1

MODE 2:

Monitor mode. This mode is used to develop and debug programs. The idea is to use a monitor board to plug into the back plane. This board disables the MCU board based CODE memory and supplies 8k of monitor ROM mapped to CODE space. The monitor card also supplies SRAM von Neumann mapped to the mid 48k of both CODE and XDATA space. The RAM on the MCU board is still available in the lower 8k of XDATA. Programs are downloaded through a high speed off chip UART so that the timers and UARTs are available for in circuit debug. The monitor firmware allows the code to then be debugged. The MCU has no direct way of controlling this mode.

XDATA MEMORY MAP	
0xFFFF	PERIPHERALS
0xE000	
0xDFFF	CODE SRAM
0x2000	
0x1FFF	MCU SRAM
0x0000	

CODE MEMORY MAP	
0xFFFF	NOT USED
0xE000	
0xDFFF	CODE SRAM
0x2000	
0x1FFF	MONITOR ROM
0x0000	

Peripheral Space:

This is the space used to interface peripherals. This includes the UARTs, network controllers, digital i/o, analog i/o etc. Unless the peripheral requirements exceed the allowed 8k of address space, devices should not be mapped into the space below.

0xFFFF	SELECT 7
0xFC00	
0xFBFF	SELECT 6
0xF800	
0xF7FF	SELECT 5
0xF400	
0xF3FF	SELECT 4
0xF000	
0xEFFF	SELECT 3
0xEC00	
0xEBFF	SELECT 2
0xE800	
0xE7FF	SELECT 1
0xE400	
0xE3FF	SELECT 0
0xE000	

Eight select lines are available on the MCU board. These select lines are used to identify individual boards with common peripheral (i.e. Two identical UART boards can be distinguished by jumper setting the select for each board). This allows up to eight 110 volt input boards to co-populate a bus with up to eight analog output boards. In order for this scheme to work, a fixed address offset is allied to the base of the selected slot by each unique board. For instance, a UART board is mapped to the last sixteen bytes of the select space. Therefore since the size of any select space is 1k or 0x03FF bytes, the offset applied by the UART card is 0x03F0 or 0x03FF-0x000F. If SELECT 0 is used, the addresses used by a UART board are 0xE3F0 to 0xE3FF. If SELECT 7 is used as on the Monitor board, the addresses used are 0xFFFF0 to 0xFFFFF.

The following is a table showing the memory allocation to date. This will change as more i/o peripherals are developed.

ADDRESSES WITHIN SELECTED SPACE	BOARD DESCRIPTION
0x03F0 to 0x03FF	UART, USART, DUART, MUART etc.
0x03EE to 0x03EF	16 channel isolated input/output
0x03ED	Hot Load Interface
0x03EA to 0x03EB	ADC card

Soft I²C Bus

The soft I²C bus is a single master software implementation of the I²C serial bus standard. In the ISC, the I²C bus is used to simplify the wiring to the user interface. This is to say that in addition to power and ground, only 2-wires are used to interface to the front panel of the controller.

Instruction Set:

Interlock instruction set:

add interlock number name invert

example: add interlock 15 "Crane Extended" 1

remove interlock number

example: remove interlock 15

save interlocks

clear interlocks

list interlocks

Status Messages Form the Master Controller:

system status a	
saDD	
DD Bit	Description
0	Right Door Closing
1	Right Roof Closing
2	Right Door Opening
3	Right Roof Opening
4	Left Door Closing
5	Left Roof Closing
6	Left Door Opening
7	Left Roof Opening

system status c	
scDD	
DD Bit	Description
0	Right Door 5%
1	Right Roof 5%
2	Right Door 95%
3	Right Roof 95%
4	Left Door 5%
5	Left Roof 5%
6	Left Door 95%
7	Left Roof 95%

system status b	
sbDD	
DD Bit	Description
0	Right Door Close Limit
1	Right Roof Close Limit
2	Right Door Open Limit
3	Right Roof Open Limit
4	Left Door Close Limit
5	Left Roof Close Limit
6	Left Door Open Limit
7	Left Roof Open Limit

system status d	
sdDD	
DD Bit	Description
0	Right Pause
1	Right Pause
2	Pumps On
3	Pause
4	Closing
5	Opening
6	Right Pump On
7	Left Pump On

