4.91
Revised 90111

Servo Software - BARC

SCC INTERFACE DETAILS

****

CONTENTS

1. LINK PROTOCOL

1.1 Introduction:

The protocol used for the communication between Station Servo Computer(SSC) and Station Control Computer(SCC) is a RS-232-C full duplex link protocol with acknowledgement,error detection and retry mechanisms builtin. It also provides task to task communication between the two stations.

This is a character oriented protocol using the following ASCII characters extended to 8 bits by adding a zero for bit 7.

1.2 Codes:

The following protocol characters are used.

STX 02
ETX 03
ENQ 05
ACK 06
DLE 10
NAK 15

The character DLE is CONTROL/DATA delimiter.

1.Control codes:

DLE STX	Start of text
DLE ETX BCC	End of text (followed by checksum)
DLE ACK	Successfully received
DLE NAK	Failed to receive
DLE ENQ	Requests retransmission of last received code.

2.Data Codes:

Data	characters 00-0F and 11-FF.
DLE DLE	to provide data transparancy data 10h (DLE) is compended with DLE

The messages exchanged between the two stations can be classified into message codes which are sent from the transmitter to the receiver, and response codes which are sent from the receiver to the transmitter.DLE NAK and DLE ACK are the response codes whereas DLE STX,DLE ETX BCC,DLE ENQ and data codes are all message codes.

Use of these control characters in message frame flow between two machines in full duplex mode is shown below.

1.3 Link layer message packets:

A link layer message packet starts with a DLE STX,ends with a DLE ETX BCC,and includes all link layer data codes in between.Data codes can occur only inside a message packet.Response codes can also occur between a DLE STX and a DLE ETX BCC,but these response codes are not part of the message packet,they are called embedded responses.

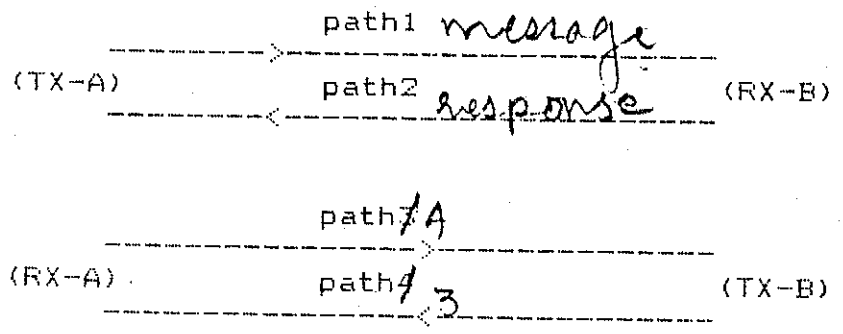
DLE STX <--- HEADER + DATA(from application layer)---> DLE ETX BCC

The HEADER is 3 bytes long arranged in the following order. Destination_id , source_id followed by the length of the data string.

NOTE:

BCC is two's complement of modulo-8 sum of all data bytes between DLE STX and DLE ETX BCC.It does not include any control code(or embedded response codes).For data 10 hex however only one of the DLE bytes is included in the BCC sum.

1.4 Message flow:



fig(1)

******

Both the stations A and B can initiate a message transmission simultaneously. The stations A and B have two machines each, a transmitter and a receiver.Transmitter A sends message frame on path 1 to the receiver B which sends response codes on path 2 to the transmitter A.Similarly transmitter B sends message frame on path 3 to the receiver A which sends response codes to the transmitter B on path 4.But physically only 2 paths(channels) are avilable for the communication between stations A and B. RECEIVE and TRANSMIT state machines at each station keep track of message / responce codes flowing on single physical channel.Information and response frames originating at either of the stations does not required to be interleaved giving true fullduplex performance.

1.5 Restrictions on messages:

1. Minimum size of a valid user message is 1 byte and maximum size is 255 bytes.
2. The task number can not be hex 10h.

1.6 Interface to the application software:

A typical set of PASCAL procedures and functions required to provide the link layer interface to the applications is given below.

```
1.function get_free_buf(var ptr:com_buf_ptr):boolean;
2.procedure put_free_buf(ptr:com_buf_ptr);
3.procedure put_in_req(ptr:com_buf_ptr);
4.function get_from_indq(task_id:byte;var ptr:com_buf_ptr):boolean;
5.function get_from_confq(task_id:byte;var ptr:com_buf_ptr):boolean;
```

a) The function 'get_free_buf' returns TRUE and a pointer to a free buffer if available otherwise returns FALSE.

b) The procedure 'put_free_buff' puts a user specified buffer passe as a pointer to link's free pool.

c) The procedure 'put_in_req' puts a user specified buffer passed as a pointer to link's request queue.

d) The function 'get_from_indq' returns TRUE and a pointer to a buffer provided a frame is received for the indicated task_id otherwise returns FALSE.

e) The function 'get_from_confq' returns TRUE and a pointer to a buffer if link confirmation for the last transmission requests by the indicated task_id is/are ready (it returns the confirmation to the oldest frame that is not yet collected) otherwise returns FALSE.
FALSE.

```
com_buf_ptr: ^com_buf_typ;
com_buf_typ= record
               data= array[0..255] of byte;    { user data }
               length:byte;                    { actual tx/rx message
                                                 length}
               stt:byte;                       {link status
                                                returned to user
                                                as tx ok,rx ok,rem
                                                not responding,only NAK e
               source_task_id:byte;            { id of local task
                                                 requesting transmission
                                               or id of remote task
                                               that transmitted data}
               dest_task_id:byte;              {id of local task
                                               receiving data or id
                                               of remote task to
                                               receive data}
               next:com_buf_ptr;               { link to the next
                                               frame to be used by
                                               link only}
             end;
```

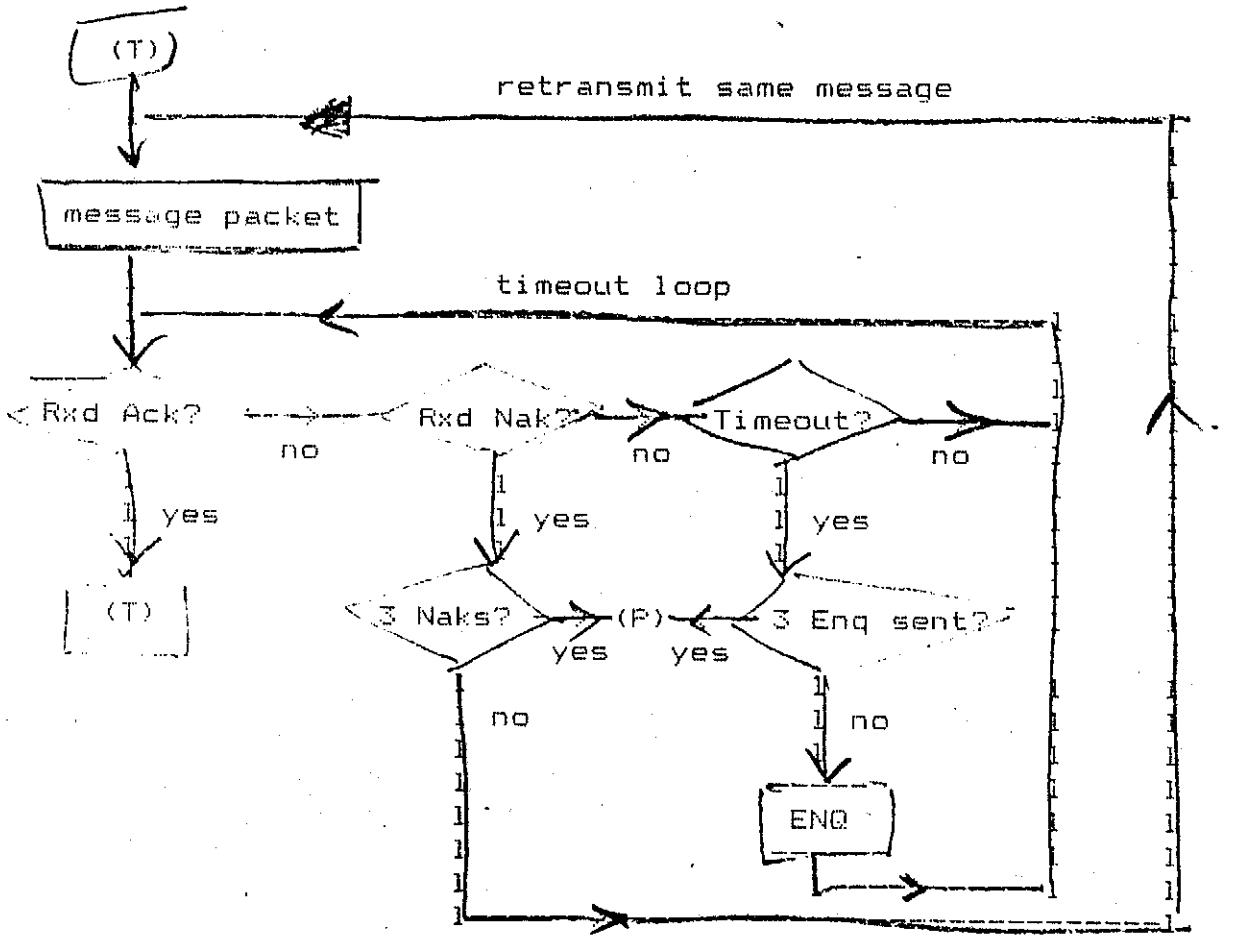
A linked list of 20 (fixed at compile time,a maximum of 255 can be specified) free buffers is maintained by the link layer as declared above .

1.7 The Protocol:

1.7.1 Transmitter:

1. The transmitter sends a message,starts a timeout and waits for response.
2. When DLE ACK is received it signals success.
3. When DLE NAK is received the message will be retransmitted.It restarts timeout and again waits.This will be repeated three times.
4. If timeout expires before a response is received,the transmitter sends a DLE ENQ to request the retransmission of last response.It starts timeout and waits for a response.This also will be repeated 3 times.
5. Invalid responses are ignored.
6. The failures are indicated to the user,giving the reasons for failure,(ie, ONLY NAK or TIMEOUT).

1.7.2 Transmitter State diagram:



1.7.3 Receiver:

On the receiver side the possible error conditions are
1.The message sink is full.
2.Parity Error.
3.BCC invalid.
4.DLE STX or DLE ETX BCC may be missing.
5.Too long or too short message.
6.Spurious control or data code outside the message.
7.Spurious control or data code inside the message.
8.DLE ACK is lost causing the transmitter to send a duplicate copy of message that has already passed to the message sink.

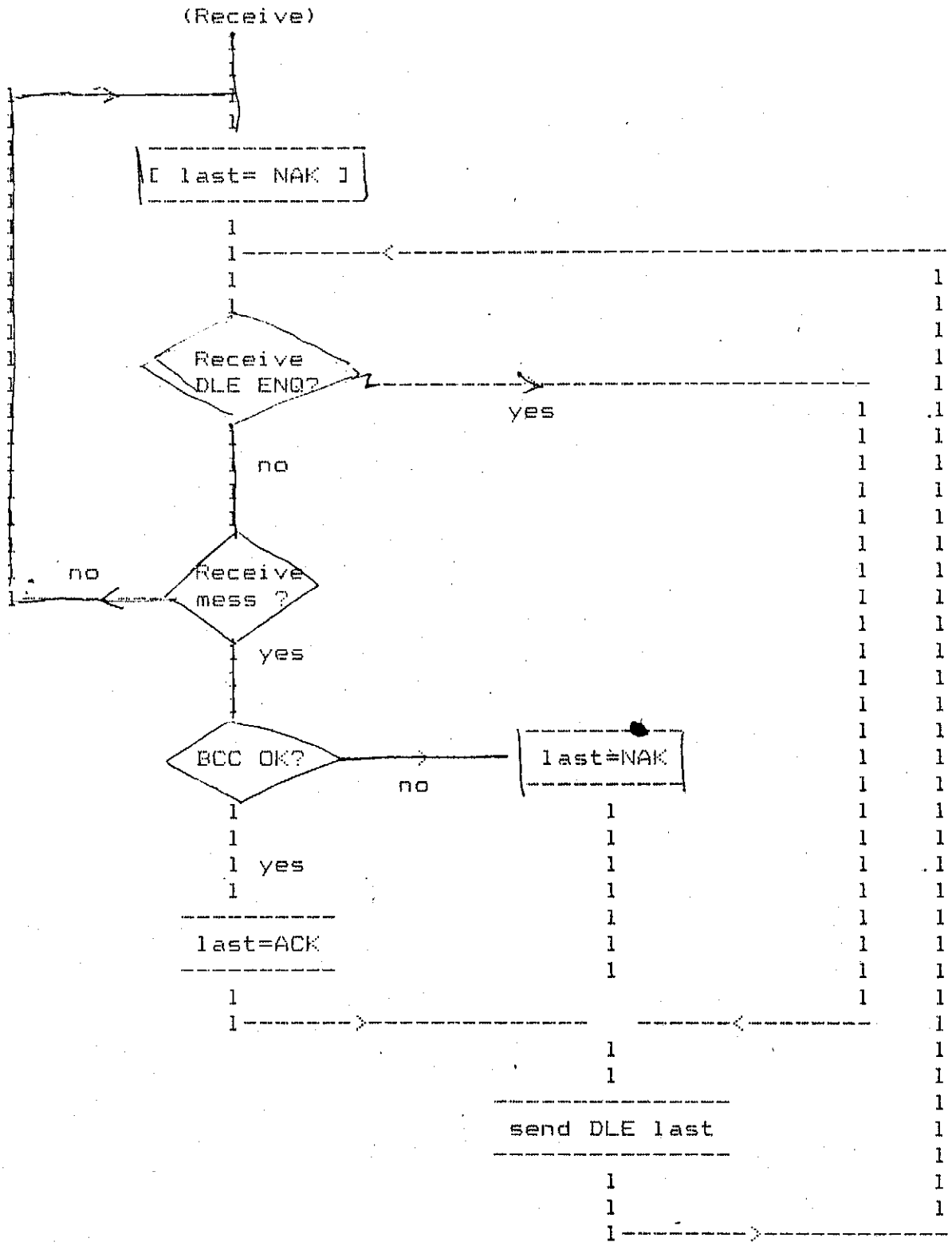
A record of last response sent is kept (ACK or NAK),which is initialised to NAK.The receiver sends this value for DLE ENQ. For a message with the same header byte ACK is sent , but the message is discarded. The last response is set to NAK if any code other than DLE STX or DLE ENQ is received before a message starts.

When building a message if the buffer overflows then the receiver continues summing BCC, but the data is discarded.If any other control codes other than DLE ETX BCC is received the message is aborted and DLE NAK is sent.

If the message is OK, its header is compared with last message.If it is same then Ack is sent but the message is discarded. If the message is different then the message sink is checked.If the sink is full the response is recorded but not sent.It waits for DLE ENQ.If any other code is received then the response is changed from ACK to NAK and the receiver continues to wait for DLE ENQ.If DLE ENQ is received, it checks the sink

status.If it is full, it continues waiting.If it is not full then the last response is sent and it accepts new messages.

1.7.4 Receiver State diagram:



1.8 Full_duplex protocol diagrams:

The following figures show some events that can occur on the

various interfaces.Time is represented as increasing from the top of the figure to the bottom.

```
SOURCE  (XMTR)                  LINK                          (RCVR)  SINK
```

1.Normal message transfer:

```
Data-->
                    DLE STX(data) DLE ETX BCC-->
                                                          <-not full
                                                           Data->
                    <--DLE ACK
<--OK
```

2.Message transfer with NAK:

```
Data-->
                    DLE STX(data) DLE ETX BCC-->
                    <-- DLE NAK
                    DLE STX(data) DLE ETX BCC-->
                                                          <-not full
                                                           Data->
                    <-- DLE ACK
<--OK
```

3.Message transfer with Timeout and ENQ:

```
Data-->
                    DLE STX(data) DLE ETX BCC-->
                                                          <-not full
                                                           Data->
                    <- DL(noise) CK
          TIMEOUT
                    DLE ENQ-->
                    <-- DLE ACK
<--OK
```

4.Message tranfer with message sink full:

```
Data-->
                    DLE STX(data) DLE ETX BCC-->
                                                          <- full
                    <- DLE NAK
                    DLE STX(data) DLE ETX BCC-->
                                                          <- full
                    <- DLE NAK
                    DLE STX(data) DLE ETX BCC-->
                                                          <-not full
                                                           Data-->
                  <- DLE ACK
<--OK
```

1.9 Examples:

Consider fig(1) where messges and responses are exchanged between machines A and B on logical paths 1,2,3 and 4.

(a) Normal message:

```
Path1: DLE STX ...DLE ETX BCC DLE STX...DLE ETX BCC
Path2:         DLE ACK       DLE ACK
```

(b) Message with BCC error and recovery:

```
Path1: DLE STX..XXX..DLE ETX BCC  DLE STX ... DLE ETX BCC
Path2:             DLE NAK             DLE ACK
```

(c) Message with ETX destroyed:

```
Path1: DLE STX...XXXX[timeout] DLE ENQ DLE STX..DLE ETX BCC
Path2:     DLE NAK              DLE ACK
```

(d) Good message but ACK destroyed:

```
Path1: DLE STX...DLE ETX BCC[timeout] DLE ENQ DLE STX ...etc.
Path2:              DLXXXCK       DLE ACK
```

(e) Messages being sent in both directions:

```
Path1: DLE STX...DLE ETX BCC  DLE STX...DLE ETX BCC  DLE STX
Path2:          DLE ACK          DLE ACK
Path3:     DLE STX...    ...DLE ETX BCC  DLE STX..etc.
Path4:           DLE ACK
```

Combined:

1. Circuit AB(Physical channel AB):DLE STX...DLE ETX BCC DLE STX...DLE ETX BCC DLE ACK DLE STX

2. Circuit BA(Physical channel BA): DLE STX...DLE ACK...DLE ETX BCC DLE ACK DLE STX

****

2. COMMAND SET

2.1 Introduction:

The various commands accepted from the Station Control Computer(SCC) can be classified into the following groups.

1.Operational Commands.
2.Display commands.
3.Set Mode Commands.

The syntax of the various command parameters are as follows.

1. ax	->	A/E/B	A=azimuth,E=elevation,B=both.
2. time	->	hh:mm:ss	hh=hours,mm=minutes,ss=seconds 2 colons are a must,but each field should have atleast one character.eg,Time= 0:0:0
3. date	->	dd:mm:yyyy	dd=date,mm=month,yyyy=year 2 colons are a must,but each field should have atleast one character.

eg,date=1:01:1989

4. angl -> +/-ddd:mm:ss — ddd=degs,mm=mins,ss=secs. 2 colons are a must,but each field should have atleast one character.ie,angl = -123:50:10 The sign is optional.Default sign is +ve.

5. ang2 -> +/-ddd:mm:ss — ang2 is required only if axis specified is B(both). syntax is similar to angl.

6. windvel -> xxx.xxxx — eg, 123.4567

All the parameters should be in ASCII form and the parameters should be seperated by comma(s).

2.2 OPERATIONAL COMMANDS:

For each of the operational commands the Station Servo Computer(SSC) sends an immediate response frame from the application layer, which indicates whether the command issued is accepted or not. If the command is not accepted then the response frame also includes 1 byte code indicating the reasons for failure.

SL_NO	COMMAND	SYNTAX	DESCRIPTION
1.	Coldstart	40H	Stow releases and holds the axes in the current positions. Coldstart= (stwrls+hold)
2.	Position	42H,ax,angl[,ang2]	Positions the specified axis to the specified position(s).
3.	Track	44H,ax,time,angl [,ang2]	Tracks the specified axis so as to reach the target position(s) at the specified time.
4.	Hold	46H,ax	Holds one or both the axis at the current position.
5.	Stop	48H,ax	The drives are turned off and brakes are applied to one or both the axis.
6.	Close	4aH	Turns off and parks both the axis.
7.	Stow	4cH,ax	Drives the specified axis to the stow position.
8.	Stow release	4eH,ax	Releases the stow on the specified axis.

9.	Abort	50H	Aborts the previous command undertaken if any.

2.3 DISPLAY COMMANDS:

The possible responses for these commands are the requested data packet or a NOT ACCEPTED frame.The NOT ACCEPTED frame includes the reason for the failure of the command.

SL_NO	COMMANDS	SYNTAX	DESCRIPTION
1.	Read angles	30H	sends current positions ,target positions,pot positions.

The various parameters sent by SSC are arranged in the following order and comma is the delimiter.

angles response code:31H

time of day	:time,
az current position (CP)	:angl,
az target position (TP)	:angl,
az pot position (PP)	:angl,
el current position (CP)	:angl,
el target position (TP)	:angl,
el pot position (PP)	:angl,

2. Read Analog Vars 32H sends a data packet containing all analog variables

The various parameters sent by SSC are arranged in the following order and comma is the delimiter.

analog response code:33H

time of day	:time,	
az motor1 current	:xxx.xxxx,	in amps eg, 35.12 Amps.
az motor2 current	:xxx.xxxx,	in amps eg, 34.1234 amps.
az tacho1	:xxxx.xxxx,	in RPM eg, 987 RPM
az tacho2	:xxxx.xxxx,	in RPM eg, 1023 RPM
el motor1 current	:xxx.xxxx,	in amps eg, 35.12 Amps.
el motor2 current	:xxx.xxxx,	in amps eg, 34.1234 amps.
el tacho1	:xxxx.xxxx,	in RPM eg, 987 RPM
el tacho2	:xxxx.xxxx,	in RPM eg, 1023 RPM
wind vel1	:windvel,	in KMPH eg, 63 KMPH
wind vel2	:windvel,	in KMPH eg, 63 KMPH

3. Read Digital Vars 34H sends a data packet containing all the digital variables.

The various parameters sent by SSC are arranged in the following order and comma is the delimiter.The first byte $35 is the digital data response code.The next 6 bytes contain the status of various digital variables.Within each byte bits d5 and d2 do not contain any information,d7 and d6 contains system variables,d4 and d3 contains elevation axis variables and d1,d0 contains the status of azimuth axis variables.

byte 0= dgtl response code: 35H

```
time of day(bytes 1 to n):                          time,
        d7(msb) d6  d5   d4     d3     d2   d1     d0(lsb)

byte n+1 :POSG   MAN   x  AMPOL  ENCOK  x  AMPOL  ENCOK,
byte n+2 :TRKG   LOC   x  STPOS  RUN    x  STPOS  RUN,
byte n+3 :SLWG   REM   x  STWD   CCPL   x  STWD   CCPL,
byte n+4 :CWRP   ACOK  x  STRLSD CCFL   x  STRLSD CCFL,
byte n+5 :WND50  DCOK  x  BRK1   CWPL   x  BRK1   CWPL,
byte n+6 :WND85  SSCOK x  BRK2   CWFL   x  BRK2   CWFL,
```

All the variables are active high.ie, BRK1=1 if brake1 is applied.

4. Read setparameters: 36H sends kp,ki,offset angles etc.

The various parameters sent by SSC are arranged in the following order and comma is the delimiter.

```
set parameters response code:37H
time of day          :time,
kp                   :xxx.xxxx,
ki                   :xxx.xxxx,
az soft ofst         :ang1,
az stow angle        :ang1,
el soft ofst         :ang1,
el stow angle        :ang1,
wind vel limit       :windvel,
```

2.4 SET MODE COMMANDS:

The possible responses from the SSC for these commands are

(a) The current set values of the parameters from the application layer which indicates that the specified set command is executed.

(b) NOT ACCEPTED frame containing the reasons for failure.

SL_NO	COMMANDS	SYNTAX	RESPONSE
1.	Set Time Of Day	52H,time,date	53H,time,date
2.	Set Stow Angles	54H,ax,ang1[,ang2]	55H,ang1[,ang2]
3.	Set S/W Hi Limit	56H,ax,ang1[,ang2]	57H,ang1[,ang2]
4.	Set S/W Lo Limit	58H,ax,ang1[,ang2]	59H,ang1[,ang2]
5.	Set Windvel Limit	5AH,windvel	5BH,windvel

****

3. THE RESPONSES FROM SSC

3.1 Introduction:

The responses from the SSC can be grouped into the following three groups.

1.Immediate responses.
2.Final responses.
3.Response messages(or events).

For each of the command the SSC sends an immediate response frame which can be either an ACCEPTED frame or a NOT ACCEPTED frame sent from the application layer.This is in addition to the link level ACK/NAK frames exchanged between the communication layers on SSC and SCC.

Only operational commands are responded with both immediate and

a final response.For all other commands the SSC sends only an immediate response.

A few response messages (or events) are sent by the SSC to indicate the status of command execution.For these events the SSC expects only a link level ack. (ie, the application layer need not send an user level ACCEPTED frame.) The final response also goes as an event to the SCC.

3.2 IMMEDIATE RESPONSES:

RESPONSES	CODE	DESCRIPTION
1.ACCEPTED	10H	Data string of this frame contains only one byte code hex 10.
2.NOT ACCETED	11H	Data string of this frame contains 2 bytes ,hex 11 followed by one of the FAIL REASONS byte defined below.

FAIL REASONS	CODE
(a) CMD TIMEOUT	51H
(b) SYNTAX ERRO	53H
(c) ILLEGAL CMD	54H

3.3. FINAL RESPONSES:

These are sent as event frames(2 byte data string), first byte being 12H(event code) followed by one of the following codes.

SL_NO	RESPONSE	CODE
1.	IRRELEVANT CMD	FFH
2.	SUCCESSFUL	10H
3.	FAILED	20H
4.	ABORTED	30H

3.4 RESPONSE MESSAGES (EVENTS):

The event frames contain 2 data bytes the first one being 12H (event code) followed by one of the following codes.

SL_NO	EVENT	CODE AZ	CODE EL
1.	NOT STOWING	62H	63H
2.	NOT STOW RELEASING	64H	65H
3.	CAN'T SWITCH ON AXIS	66H	67H
4.	AXIS NOT TURNING ON	68H	69H
5.	AXIS NOT TURNING OFF	6aH	6bH
6.	STOW ERROR OCCURED	6cH	6dH
7.	STOW NOT RELEASED	6eH	6fH

8	NOT STOWED	70H	71H

SL_NO	EVENT	CODE AZ	EL	
9 .	STOW RELEASING	80H	81H	
10.	STOW RELEASED	82H	83H	
11.	BRAKE RELEASED	84H	85H	
12.	AXIS HELD	86H	87H	
13.	STOWING	88H	89H	
14.	STOWED	8aH	8bH	
15.	BRAKED	8cH	8dH	
16.	POSITIONED	8eH	8fH	
17.	TRACKING	90H	91H	
18.	PARKING	92H	93H	
19.	POSITIONING	94H	95H	
20.	MOTOR CURRENTS HIGH	a0H	a1H	
21.	WIND VELOCITY HIGH			a2H
22.	MOTOR INTERLOCK ERROR	a3H	a4H	
23.	EMERGENCY PARK STARTED			a5H

3.5 Commands State table:

(VALID COMMANDS AT EACH STATE AND POSSIBLE RESPONSES)

STATE	COMMANDS		NEXT STATE	RESPONSES
1.STOWED	STWRLS		STOWED	FAILED
	STWRLS	->	STW RLSG	
2.STW RLSG			STW ERR	FAILED
			STOWED	FAILED
			STW RLSD BRKD	FAILED
			STW RLSD BRKD	SUCCESSFUL
	ABORT		STOWED	ABORTED
	ABORT		STW RLSD BRKD	ABORTED
	ABORT		STW ERR	ABORTED
	EM PARK	->	STOWING	
3.STW ERR	STWRLS	->	STW RLSG	
	STOW	->	STOWING	
4.STOWING			STW ERR	FAILED
			STOWED	FAILED
			STW RLSD BRKD	FAILED
			STWED	SUCCESSFUL
	ABORT		STOWED	ABORTED
	ABORT		STW RLSD BRKD	ABORTED
	ABORT		STW ERR	ABORTED
5.STW RLSD BRKD	STOW		STW RLSD BRKD	FAILED
	STOW	->	STOWING	
	PARK	->	STOWING	
	PARK	->	PARKSLEWING	
	AXISON		STW RLSD BRKD	FAILED
	AXISON		STWRLSD BRKRLSD	SUCCESSFUL
	ABORT		STWRLSD BRKD	ABORTED

6.STW RLSD BRK RLSD	STOW,PARK	-> PARKSLEWING	
	EMSTOP	STWRLSD BRKD	SUCCESSFUL
	EMSTOP	STWRLSD BRKRLSD	FAILED
	POSITION	-> POSG	
	TRACK	-> TRACKING	
7.PARK SLWG		-> STOWING	
		STWRLSD BRKRLSD	FAILED
	EMSTOP	STWRLSD BRKRLSD	FAILED
	EMSTOP	STWRLSD BRKD	SUCCESSFUL
	ABORT	STWRLSD BRKRLSD	ABORTED
	STOW,PARK	PARK SLWG	
8.POSNG		STWRLSD BRKRLSD	SUCCESSFUL
	EMSTOP	STWRLSD BRKRLSD	FAILED
	EMSTOP	STWRLSD BRKD	SUCCESSFUL
	STOW,PARK	->PARKSLEWING	
	ABORT	STWRLSD BRKRLSD	ABORTED
	POSITION	POSNG	
9.TRKING		STWRLSD BRKRLSD	SUCCESSFUL
	EMSTOP	STWRLSD BRKRLSD	FAILED
	EMSTOP	STWRLSD BRKD	SUCCESSFUL
	STOW,PARK	->PARKSLEWING	
	ABORT	STWRLSD BRKRLSD	ABORTED
	TRACK	TRKING	

The commands from SCC are executed as a sequence primitive steps which are listed below. The order of execution is from left to right.

1.COLDSTART = STWRLS + AXISON
2.POSITION = POSITION
3.TRACK = TRACK
4.HOLD = ABORT+ STWRLS +AXISON
5.STOP = EMSTOP
6.CLOSE = ABORT+ AXISON+ PARK
7.STOW = STOW +AXISON + PARK
8.STWRLS = STWRLS
9.ABORT = ABORT

26 thw

****