INTERFACING THE SEPTA MOVEMENT (DC MOTOR) EQUIPMENT TO THE PS CONTROL SYSTEM AND THE MIL1553 BUS

C. Dehavay

Continuing the rejuvenation of the PS Control system, it is planned to replace the Single Transceiver Hybrid used to interface the Septa Movement Equipment by a G64 system connected to the VME crate via the MIL1553 bus.

This note explains the G64 hardware interface and details the standard message as defined in the Control Protocole for Power Converter, RF and Stepping Motor equipment.
Interfacing the Septa Movement (DC motor) equipment to the PS control system and the MIL1553 bus.

C. Dehavay

Continuing the rejuvenation of the PS Control system, it is planned to replace the Single Transceiver Hybrid used to interface the Septa Movement Equipment by a G64 system connected to the VME crate via the MIL1553 bus. This note explains the G64 hardware interface and details the standard message as defined in the Control Protocol for Power Converter, RF and Stepping Motor equipment.

1. Introduction

Purpose: to replace the Single Transceiver Hybrid, used with the Camac Quad Transceiver, by a G64 system connected via a MIL1553 bus to a VME Bus Controller.

Input / output features:
- 1 digital input byte (motor status)
- 1 digital output byte (keyword: stored in the G64 chassis)
- 1 analog input signal (-10 to +10 V) for current motor position
- 1 analog output signal (+10 V) for target motor position

G64 modules used:
- GESC I/O module
- QADC 12 bits Cern/PS made module
- DAC 12 bits Cern/PS made module

Connections:
With the Motor Equipment:
All connections will be patched with a module plugged in place of the STH module.

With the Control upper level:
The standard MIL 1553 bus is used (same as for RF, Power Converter and Stepping Motors equipments).

2. System Description.

2.1. Interface with the Equipment.

2.1.1. Control.

Keyword (digital byte)

To follow the same philosophy as for the new Stepping Motor system interfaced with the MIL1553 bus, the Control system doesn't send, as in the past, a 'keyword' which validates the new target position. As the equipment still needs this keyword, its value (identical for all DC motors = 20) followed by a strobe (40 µs, TTL negative logic) is generated by the G64 system and sent before a new target position.

The following table shows the bit allocation for the output register on the GESC I/O module.
<table>
<thead>
<tr>
<th>Bit number</th>
<th>Pin number</th>
<th>Meaning</th>
<th>Boolean Variable in G64</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>keyword</td>
<td>s_remot</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>keyword</td>
<td>s_power</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>Equipment in Manual Control</td>
<td>s_local</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>Limit In reached</td>
<td>s_limin</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>Keyword enable</td>
<td>s_keyword</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>Motor Busy</td>
<td>s_busy</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>Cables connected</td>
<td>s_cable</td>
</tr>
</tbody>
</table>

Table 1: Bit allocation for the Output Register.

**Target position (DAC module)**
The position is voltage controlled with a Digital to Analog Card (12 bits resolution with range is between 0 and +10000 mV).
A strobe (40 µs, TTL negative logic) is generated by the G64 system when the analog signal is correct (time between Strobes A and B is about 16 ms).

**Motor status (8 bits)**
The bit allocation for the input register is given in the following table.

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Pin number</th>
<th>Meaning</th>
<th>Boolean Variable in G64</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>Equipment in Remote Control</td>
<td>s_remot</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>Power ON</td>
<td>s_power</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>Equipment in Manual Control</td>
<td>s_local</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>Limit Out reached</td>
<td>s_limou</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>Limit In reached</td>
<td>s_limin</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>Keyword enable</td>
<td>s_keyword</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>Motor Busy</td>
<td>s_busy</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>Cables connected</td>
<td>s_cable</td>
</tr>
</tbody>
</table>

Table 2: Bit allocation for the Input Register.

**Position Acquisition (ADC module)**
The current position (from -10000 to +10000 mV) is converted with the ADC module (12 bits resolution).

At each machine cycle, an external trigger starts the digital conversion which generates an interrupt on the G64 bus. Then, the CPU acquires data and writes it to the RTI card.

**Initialisation:**
To avoid problems when the G64 system is powered ON, the local software reads the current position and writes the corresponding value to the DAC module.

**2.2. Interface with the upper control level.**

**2.2.1. Protocole Message Header.**

All fields are identical as in the message header already defined for the Power Converter, RF and Stepping Motor equipment.
The Family number will be 65 (SEMO Equipment Module)

2.2.2. Control message sent to the G64 system via the MIL bus.

The only current control value is the target position (unit: mV) sent with a normal control message (see the Control Protocol) which is almost a copy of the one used for Power Converter and RF equipments. Table 3 details the normal Control message content.

Properties:
CCV: unit = mV range = 0 to 10000

2.2.3. Acquisition message read from the G64 system via the MIL bus.

Table 4 details the normal acquisition message which is almost a copy of the one used for Power Converter and RF equipment.

Properties:
DATE : as date and time are not available in the G64 system, this variable contains the value of the last control message,
PSTAT : 1= operational, 2= partially operational, 3= not operational, 4= needs commissioning,
STAQ : not implemented; value = 0.
ASPEC : 1= not connected, 2= local, 3= remote,
QUALIF : 6 exclusive bits indicating the status of the DC motor which is operational if all bits are 0;
bit 0 = interlock, bit 1 = unresettable fault, bit 2 = resettable fault, bit 3 = busy, bit 4 = warning,
BUSYTIM: as this data is not available for DC motor, value = -1,
AQN : unit = mV range = -10000 to 10000

Logical equations for Status variables in the Acquisition message.

Physical status:
NEEDSCOM := NOT(s_power) OR NOT(s_cable)
NOTOPER := NOT(NEEDSCOM) AND (NOT(s_remote) OR s_local)
PARTLY := NOT(NEEDSCOM) AND NOT(NOTOPER) AND (s_limou OR s_limin)
OPERAT := NOT(NOTOPER) AND NOT(PARTLY)

Static status not implemented.

External Aspects:
NOTCONNECTED := NOT(s_cable)
LOCAL := NOT(NOTCONNECTED) AND s_local
REMOTE := NOT(NOTCONNECTED) AND s_remote AND NOT(s_local)

Status Qualifiers:
Interlock := 0
Unresettable Fault := NOT(s_cable OR s_local) OR NOT(s_power) OR NOT(s_remote)
Resetable Fault := 0
Busy := s_busy
Warning := s_limou OR s_limin
Byte Nb. | Field | Type | Nb. Bytes | Property | G64 msg Buffer[..] | G64 variable | Equipm. meaning |
--- | --- | --- | --- | --- | --- | --- | --- |
1 | Family Number | Integer(16) | 2 | --- | 1..2 | EM=65 | --- |
3 | Type, Subtype | Integer(16) | 2 | --- | 3..4 | --- | --- |
5 | Serial Number | Integer(16) | 2 | --- | 5..6 | --- | --- |
7 | Req.Serv. | Integer(16) | 2 | --- | 7..8 | =0 | --- |
9 | Ident.Event | 2xInteger(16) | 4 | --- | 9..12 | not used | --- |
13 | Date | 2xInteger(32) | 8 | --- | 13..20 | not used | --- |
21 | Spec.Facil. | Integer(16) | 2 | --- | 21..22 | not used | --- |
23 | Change Act. | Byte | 1 | not used | 23 | not used | --- |
25 | First CCV | Real | 4 | --- | 25..28 | not used | --- |
29 | Second CCV | Real | 4 | not used | 29..32 | not used | --- |
33 | Third CCV | Real | 4 | not used | 33..36 | not used | --- |
37 | Fourth CCV | Real | 4 | not used | 37..40 | not used | --- |
41 | Ch. 1st CCV | Byte | 1 | --- | 41 | flag[1] | --- |
42 | Ch. 2nd CCV | Byte | 1 | not used | 42 | not used | --- |
43 | Ch. 3rd CCV | Byte | 1 | not used | 43 | not used | --- |
44 | Ch. 4th CCV | Byte | 1 | not used | 44 | not used | --- |

Table 3: Normal Control message for DC Motor Equipment.

```
<table>
<thead>
<tr>
<th>Byte Nb.</th>
<th>Field</th>
<th>Type</th>
<th>Nb. Bytes</th>
<th>Property</th>
<th>G64 msg Buffer[..]</th>
<th>G64 variable</th>
<th>Equipm. meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log.Id.Nb.</td>
<td>Integer(16)</td>
<td>2</td>
<td>---</td>
<td>1..2</td>
<td>EM=65</td>
<td>---</td>
</tr>
<tr>
<td>3</td>
<td>Type, Subtype</td>
<td>Integer(16)</td>
<td>2</td>
<td>---</td>
<td>3..4</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>5</td>
<td>Serial Number</td>
<td>Integer(16)</td>
<td>2</td>
<td>---</td>
<td>5..6</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>7</td>
<td>Req.Serv.</td>
<td>Integer(16)</td>
<td>2</td>
<td>---</td>
<td>7..8</td>
<td>=0</td>
<td>---</td>
</tr>
<tr>
<td>9</td>
<td>Ident.Event</td>
<td>2xInteger(16)</td>
<td>4</td>
<td>---</td>
<td>9..12</td>
<td>not used</td>
<td>---</td>
</tr>
<tr>
<td>13</td>
<td>Date</td>
<td>2xInteger(32)</td>
<td>8</td>
<td>---</td>
<td>13..20</td>
<td>not used</td>
<td>---</td>
</tr>
<tr>
<td>21</td>
<td>Spec.Facil.</td>
<td>Integer(16)</td>
<td>2</td>
<td>---</td>
<td>21..22</td>
<td>not used</td>
<td>---</td>
</tr>
<tr>
<td>23</td>
<td>Physical Stat.</td>
<td>Byte</td>
<td>1</td>
<td>PSTAT</td>
<td>23</td>
<td>memQ1</td>
<td>---</td>
</tr>
<tr>
<td>24</td>
<td>Static Stat.</td>
<td>Byte</td>
<td>1</td>
<td>STAQ</td>
<td>24</td>
<td>not used</td>
<td>---</td>
</tr>
<tr>
<td>25</td>
<td>Extern.Asp.</td>
<td>Byte</td>
<td>1</td>
<td>ASPEC</td>
<td>25</td>
<td>memQ3</td>
<td>---</td>
</tr>
<tr>
<td>26</td>
<td>Stat.Qualif.</td>
<td>Byte</td>
<td>1</td>
<td>QUALIF</td>
<td>26</td>
<td>memQ4</td>
<td>---</td>
</tr>
<tr>
<td>33</td>
<td>Second AQN</td>
<td>Byte</td>
<td>4</td>
<td>not used</td>
<td>33..36</td>
<td>not used</td>
<td>---</td>
</tr>
<tr>
<td>37</td>
<td>Third AQN</td>
<td>Byte</td>
<td>4</td>
<td>not used</td>
<td>37..40</td>
<td>not used</td>
<td>---</td>
</tr>
<tr>
<td>41</td>
<td>Fourth AQN</td>
<td>Byte</td>
<td>4</td>
<td>not used</td>
<td>41..44</td>
<td>not used</td>
<td>---</td>
</tr>
</tbody>
</table>

Table 3: Normal Acquisition message for DC Motor Equipment.
```

2.2.4. Specialist message.

Nothing is foreseen.

2.2.5. Synchronous message.

Nothing is foreseen as data acquisition happens every machine cycle and values are changing slowly.
3. Diagnostic facilities.

3.1. RS232 port on the G64 CPU module.

As for the other G64 systems (RF, Stepping motor, Power converter), diagnostic messages are available on the RS232 port of the G64 CPU module (Baud: 9600, character: 7, Parity: even, Stop bits: 2).

According to the micro-switch position on the QADC card (SW1-number 8), the local software will display a short (OFF position) or a long (ON position) diagnostic message.

The short diagnostic message is by definition very simple because CPU time can be critical on PPM accesses: it indicates if data are exchanged between the VME BC and RTI MIL1553 modules:
- 'R0' (Requested service) indicates that a normal control message is received by the G64 CPU
- 'P0' (Provided service) indicates that a normal acquisition message is written by the G64 CPU to the RTI card.

The long message details all data received from and sent to the MIL1553 bus.

Examples of diagnostic displays are given hereafter.

**ACQUISITION:**

```plaintext
Eq.Mod.    = 00EO 0000
Type       = 001A
Sub-type   = 0000
Eq.Numb.   = 0001 0000
Prov.Ser.  = 0000 0000
Id.Event   = 0004 0000
Pulse Ev.  = 0008 0000
Date       = not used yet
Spec.Fac.  = 0000 0000
Phys.St.   = 2  1-OPERAT, 2-PARTLY, 3-NONOPERAT, 4-NEEDSCOM
Stat.St.   = 3  1-OFF, 2-STBY, 3-ON, 4-RESET
Ext.Asp.   = 3  1-NOTCON, 2-LOCAL, 3-REMOTE
St.Qual.   = 008B
Current position = -1.125000E+03 mV
Motor status  = 10001011
```

**CONTROL:**

Message received:

```plaintext
00 41 00 01 00 01 00 00 00 04 00 08
00 00 00 00 00 00 00 00 00 00 01 01
44 AF 00 00 3F 80 00 00 C2 C8 00 00
41 80 00 00 01 01 00 00
Req.Serv.  = 0
Eq.Mod.    = 0041 0000
Type       = 0000
Sub-type   = 0001
Eq.Numb.   = 0001 0000
Req.Serv.  = 0000 0000
Id.Event   = 0004 0000
Pulse Ev.  = 0008 0000
Date       = not used yet
Spec.Fac.  = 0000 0000
Ch.Vall    = 0001
Vall       = 1.400000E+03 mV
R0         = 1.400000E+03
send msg
00 3C 03 00 00 01 00 1A
1F 01 00 01 00 08 00 00
```
3.2. Standalone control.

The CO standard MIL-trotinette allows to control the DC motor equipment via the MIL1553 bus.

Just connect the MIL-trotinette (MIL1553 port) to the RTI card of the G64 equipment system, switch ON this one, then power ON the MIL-trotinette. If connection is bad, the MIL-trotinette terminal will beep.

The following menu will appear:

G64:BC->RTI Test (V3.0, 21/03-92) serving on MIL-1553
MIL-Trotinette (24/05/93)
Available tests :

1) Power Converter local control (MIL)
2) Stepping Motor local control (MIL)
3) DC motor (MIL)
4) RF station (MIL)
5) Power Converter emulation (STE)

Choose test 4. The following menu will appear:

G64:BC->RTI Test (V3.0, 10/06-92)

0 : Quit 1 : RTI connections
2 : R. CSR reg 3 : Read Transmit Buf
4 : R. Rec. Buf 5 : Send Message to RT
8 : Read Config 9 : R.Bck.Ctl.Msg. RTI

choice (0,1,2,3,4,5,6,7,8,9) >

Test 5 sends a new position (in mV) to the equipment.
Test 6 reads the motor status and the current position (in mV).
4. Details on G64 local software.

4.1. Pascal Procedures used in the DC Motor interface (G64)

<table>
<thead>
<tr>
<th>Module name</th>
<th>Procedure Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp.txt</td>
<td>acquis</td>
<td>total acquisition routine</td>
</tr>
<tr>
<td>mx2.txt</td>
<td>chk_st</td>
<td>checks eq. stat. and prepares variables for protocole</td>
</tr>
<tr>
<td>mp.txt</td>
<td>control</td>
<td>checks message and send control values to eq.</td>
</tr>
<tr>
<td>mx2.txt</td>
<td>del50</td>
<td>software delay</td>
</tr>
<tr>
<td>mteph.txt</td>
<td>delay</td>
<td>software delay</td>
</tr>
<tr>
<td>mq.txt</td>
<td>discv</td>
<td>disable conversion on QADC card</td>
</tr>
<tr>
<td>mq.txt</td>
<td>encv</td>
<td>enable conversion on QADC card</td>
</tr>
<tr>
<td>mp.txt</td>
<td>good_msg</td>
<td>for test: sets variables for a good control message</td>
</tr>
<tr>
<td>mmi11.txt</td>
<td>header_init</td>
<td>initialises message header</td>
</tr>
<tr>
<td>mii.txt</td>
<td>init</td>
<td>initialises variables</td>
</tr>
<tr>
<td>mmi11.txt</td>
<td>int1</td>
<td>treats MIL interrupt</td>
</tr>
<tr>
<td>mq.txt</td>
<td>int2</td>
<td>treats interrupt from QADC card</td>
</tr>
<tr>
<td>mx1.txt</td>
<td>lg_amsg</td>
<td>for test: displays acquis. message in detail</td>
</tr>
<tr>
<td>mx1.txt</td>
<td>lg_cmsg</td>
<td>for test: displays control message in detail</td>
</tr>
<tr>
<td>mmi11.txt</td>
<td>mill_main</td>
<td>initialises RTI card</td>
</tr>
<tr>
<td>mmi11.txt</td>
<td>mil_read</td>
<td>reads RTI buffer</td>
</tr>
<tr>
<td>mmi12.txt</td>
<td>msg_rcpt</td>
<td>message reception</td>
</tr>
<tr>
<td>mq.txt</td>
<td>r_aqnd</td>
<td>reads direct acquisition values</td>
</tr>
<tr>
<td>mteph.txt</td>
<td>r_stat</td>
<td>reads direct equipment status</td>
</tr>
<tr>
<td>mq.txt</td>
<td>readflgd</td>
<td>reads diagnostic bit in QADC card</td>
</tr>
<tr>
<td>mmi11.txt</td>
<td>RReal</td>
<td>convert IEEE floating point to Omegasoft format</td>
</tr>
<tr>
<td>mx1.txt</td>
<td>s_amsg</td>
<td>get acquis. and sends it to RTI buffer</td>
</tr>
<tr>
<td>mx2.txt</td>
<td>s_ccvs</td>
<td>send control values to equipment</td>
</tr>
<tr>
<td>mx2.txt</td>
<td>s_spec</td>
<td>sends specialist acquisition message</td>
</tr>
<tr>
<td>mq.txt</td>
<td>sconv</td>
<td>starts conversion on QADC card</td>
</tr>
<tr>
<td>mmi11.txt</td>
<td>send_acqmsg</td>
<td>sends ppm acquisition message</td>
</tr>
<tr>
<td>mmi11.txt</td>
<td>send_msg</td>
<td>writes message to the RTI card</td>
</tr>
<tr>
<td>mx1.txt</td>
<td>sh_amsg</td>
<td>for test: displays short acquis. message</td>
</tr>
<tr>
<td>mx1.txt</td>
<td>sh_cmsg</td>
<td>for test: displays short control message</td>
</tr>
<tr>
<td>mmi11.txt</td>
<td>swap_header</td>
<td></td>
</tr>
<tr>
<td>mmi12.txt</td>
<td>W_Real</td>
<td>converts Omegasoft floating point to IEEE format</td>
</tr>
<tr>
<td>mteph.txt</td>
<td>wpos_cv</td>
<td>writes new value to DAC card</td>
</tr>
<tr>
<td>m1.txt</td>
<td>main program</td>
<td></td>
</tr>
</tbody>
</table>