Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

Intellitrol Firmware Changes for Version 1.6.38

Product Numbers Affected:

15051 U17 Main Intellitrol hex file, Last Release 1.6.36

C-020-01 Source Code U17 Main Intellitrol, Last Release 1.6.36

This document will show the changes that were made to the Intellitrol firmware since the release of version 1.6.36.

The following support was added for the SuperTIM:

- Parameters can be read and (some) written by the TAS via the Modbus interface.
- A fault log was added to the SuperTIM memory to record the last 5 sensor or ground faults
- Last load date and time as well as Intellitrol SN and firmware rev are recorded into the SuperTIM.
- Unload mode was added for unload Intellitrols.
- The ability to not permit based on certificate expirations.
- The ability to not permit based on compartment count mismatch.
- Added Modbus error code 0x19 to indicate an attempt to write a read only register.

These features are explained in further detail in the tech note document 1902151 Intellitrol Firmware Ver 1.6.37.

On startup the firmware revision will be displayed using the compartment and VIP leds. After the leds on the front panel are cycled the firmware version will be displayed using the compartment leds to indicate the binary value (led 8 being the ls bit and led 1 being the ms bit) and the VIP leds to indicate what portion of the version is being displayed. For version 1.6.38 the system will light the VIP Authorized led to indicate the major is

being displayed and compartment 8 led will be lit to indicate the value 1. It will then light the VIP Unauthorized led to indicate the minor rev and leds 6 and 7 to indicate the value 6. The VIP Standby will then light to indicate the revision is being displayed and leds 3, 6 and 8 will be lit to indicate the value 38. Displaying the firmware revision adds approximately 10 seconds to the Intellitrol start up time.

In previous revisions of firmware if a 2-wire sensor that has been dry either goes wet, open or short it was reported as a wet sensor. The system now checks and reports the correct status.

Dome out logging into the event log is now enabled on startup.

The default dead man open time was changed from 3 seconds to 1 second.

Correctly displays version number in all available locations. Startup LED’s, event log, and ASCII terminal.

Created maintenance error event in event log.

Updated function to calculate number of connected probes to be more accurate in worst case situations.

Created modbus command 5B to calculate number of connected probes and return the amount.

Created modbus command 5C to switch between old and new ADC table for calculating number of probes. Command will return 5C00 for old table and 5C01 for new table.

Green permit bar will flash for one second when bypass key is successfully added.

Code changes are detailed in the following section.

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

Code Changes

Com_two.c

Correct reporting of 2 wire probe state

void active_two_wire(PROBE_TRY_STATE test_state)

Rev 1.6.38	Rev 1.6.36
Line 455 if (wet_pass_count > N_CYCLES_10) { check_shorts_opens(); /* This will reset probes_state and tank_state if open or short */	if (wet_pass_count > N_CYCLES_10) { if(dry_once == FALSE) { check_shorts_opens(); /* This will reset probes_state and tank_state if open or short */ }

This change will cause check_shorts_opens to be called on a non-pulsing sensor regardless of whether it's been dry or not.

comdat.c

Rev 1.6.38	Rev 1.6.36
Starting line 322 unsigned int S_TIM_code; /* Super TIM supported TRUE/FALSE */ unsigned int TIM_fault_logged = 0; unsigned char load_history_ptr; unsigned char cert_ds_fails=0; unsigned int bad_compartment_count = 0;	Starting line 322 unsigned int S_TIM_code; /* Super TIM supported TRUE/FALSE */

This change declares the global variables used for SuperTIM support

comdat.h

Rev 1.6.38	Rev 1.6.36
Starting line 169 #define ENA_INTELLITROL2 0x20 #define ENA_UNLOAD_TERM 0x40 #define ENA_CPT_COUNT 0x80	Starting line 169 #define ENA_INTELLITROL2 0x20 #define ENA_SPARE_3 0x40 #define ENA_SPARE_4 0x80
Starting line 206	Starting line 206

#define GROUND_BYPASS 0x02 /* missing ground bolt bypass */ #define CPT_COUNT_BYPASS 0x04	#define GROUND_BYPASS 0x02 /* missing ground bolt bypass */
Starting line 245 #define BVF_DONE 0x100 /* Already scanned the EEPROM */ #define BVF_UNLOAD_EXP 0x200 #define BVF_CPT_COUNT 0x400	Starting line 244 #define BVF_DONE 0x100 /* Already scanned the EEPROM */
Starting line 483 extern unsigned int S_TIM_code; /* Super TIM supported TRUE/FALSE */ extern unsigned int TIM_fault_logged; extern unsigned char load_history_ptr; extern unsigned char cert_ds_fails; extern unsigned int bad_compartment_count;	Starting line 480 extern unsigned int S_TIM_code; /* Super TIM supported TRUE/FALSE */

The first change defines the bits in enable_soft that enable the unload terminal and compartment count features.

The second change defines the bit in bylevel that indicates a compartment count mismatch is what was bypassed.

The third change defines the bits in badvipflag that indicate the VIP is not authorizing due to unload time expires and compartment count mismatch.

The fourth change declares the global variables used for SuperTIM support.

dallas.c

int read_TIM_compartment_info()

Rev 1.6.38	Rev 1.6.36
Starting line 1413 if (tim_block_read(mem_ptr, TABLE_VALID_ADDR, TABLE_VALID_SIZE) != MB_OK) { return FAILED; } if ((mem_ptr[0] != 0x55) || (mem_ptr[1] != 0xAA)) { xprintf(142, DUMMY); return GOOD; } if (tim_block_read(mem_ptr, INTELLICHECK_TYPE_ADDR, INTELLICHECK_TYPE_SIZE) != MB_OK) { return FAILED; }	Starting line 1405 if (tim_block_read(mem_ptr, VALID_ENTRIES_ADR, VALID_ENTRIES_SIZE) != MB_OK) { return FAILED; } if ((mem_ptr[0] != 0x55) || (mem_ptr[1] != 0xAA)) { xprintf(142, DUMMY); return GOOD; } if (tim_block_read(mem_ptr, INTELLICHECK_P_ADR, INTELLICHECK_P_SIZE) != MB_OK) { return FAILED; } if (mem_ptr[0])

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

if (mem_ptr[0]) { printf("\n\r Truck has an IntelliCheck\n\r"); number_of_Probes = 0xAA; return GOOD; } if (tim_block_read(Truck_TIM_Configuration, NUMBER_OF_COMPARTMENTS_ADDR, 1) != MB_OK) { return FAILED; } number_of_Compartments = (unsigned int)Truck_TIM_Configuration[0]; /* number of compartments stored in the TIM */	{ printf("\n\r Truck has an IntelliCheck\n\r"); number_of_Probes = 0xAA; return GOOD; } temp_word = NUMBER_COMPARTMENTS_SIZE + COMPARTMENT_UNIT_SIZE + (COMPARTMENT_1_SIZE * 4); if (tim_block_read(Truck_TIM_Configuration, NUMBER_COMPARTMENTS_ADR, temp_word) != MB_OK) { return FAILED; } number_of_Compartments = Truck_TIM_Configuration[0]; /* number of compartments stored in the TIM */

void read_TIM_Go_NoGo_info()

Rev 1.6.38	Rev 1.6.36
Starting line 1518 if (tim_block_read(mem_ptr, TABLE_VALID_ADDR, TABLE_VALID_SIZE) != MB_OK) { return; }	Starting line 1503 if (tim_block_read(mem_ptr, VALID_ENTRIES_ADR, VALID_ENTRIES_SIZE) != MB_OK) { return; }
Starting line 1549 if (tim_block_read((unsigned char*)&temp_byte, SCULLY_SENSORS_ADDR, SCULLY_SENSORS_SIZE) != MB_OK) { return; }	Starting line 1533 if (tim_block_read((unsigned char*)&temp_byte, SCULLY_EQUIPMENT_ADR, SCULLY_EQUIPMENT_SIZE) != MB_OK) { return; }

All the changes in dallas.c are for remapping SuperTIM parameter locations

dumfile.c

void report_tank_state(void)

Rev 1.6.38	Rev 1.6.36
Starting line 234 else if (!((badvipflag & 0xFEFF) == 0)) {	Starting line 222 else if (!((badvipflag & 0xFF) == 0)) { t3 = LITE; /* And RED for Failure */

t3 = LITE; /* And RED for Failure */ t1 = DARK; t2 = DARK; }	t1 = DARK; t2 = DARK; }
Starting line 261 else if (!((badvipflag & 0xFEFF) == 0)) /* Anything else wrong?*/ { t3 = LITE; /* Yes, just plain unauthorized truck */ t1 = DARK; t2 = DARK; }	Starting line 247 else if (!((badvipflag & 0xFF) == 0)) /* Anything else wrong?*/ { t3 = LITE; /* Yes, just plain unauthorized truck */ t1 = DARK; t2 = DARK; }

These changes were made because we are now using bits in the upper byte of badvipflag to indicate the reason that the VIP did not authorize.

esquared.c

char eeUpdateSys(void)

Rev 1.6.38	Rev 1.6.36
Starting line 178 SysParm.DM_Warn_Start = DM_WARN; /* Active Deadman Warning time */ SysParm.Cert_Expiration_Mask = 0x00; /* Active Deadman Warning time */ SysParm.Unload_Max_Time_min = 240; /* Active Deadman Warning time */ return nvSysParmUpdate(); /* Write SysParm to EEPROM */	Starting line 178 SysParm.DM_Warn_Start = DM_WARN; /* Active Deadman Warning time */ return nvSysParmUpdate(); /* Write SysParm to EEPROM */

This change sets the default values for certificate expiration enable and the maximum unload time.

main.c

void main(void)

Rev 1.6.38	Rev 1.6.36
Starting line 588 diagnostics(DIA_CHK_INIT); /* Run Diagnostics/Calibration for following parameters: */ /* Kernel and Flash(Shell) CRC, Dallas Clock, */	Starting line 588 diagnostics(DIA_CHK_INIT); /* Run Diagnostics/Calibration for following parameters: */ /* Kernel and Flash(Shell) CRC, Dallas Clock, */

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

/* Reference, open voltages and 6/8 compartment */ /* jumpers, raw, bias, noise voltages, 10/20 voltage */ /* Ground R/Diode, LED Panel, Enable Jumpers */ show_revision(); MBLINK1 = 1; MBLINK2 = 0; MBLINK3 = 1;	/* Reference, open voltages and 6/8 compartment */ /* jumpers, raw, bias, noise voltages, 10/20 voltage */ /* Ground R/Diode, LED Panel, Enable Jumpers */ MBLINK1 = 1; MBLINK2 = 0; MBLINK3 = 1;

This change calls the function that displays the version number.

modcmd.c

static MODBSTS write_tim(unsigned char tim_type)

Rev 1.6.38	Rev 1.6.36
Lines 1925 to 2038 Commented out the routine write_tim as it is all handled through write_tim_block	

static MODBSTS mbcRdTrBuilderInfo(void)

Lines 2060 to 2824, Mapped all the tim parameters for the 0x53 writeparameter command

static MODBSTS mbcRdTrBuilderInfo(void)

Lines 2060 to 2824, Mapped all the tim parameters for the 0x53 read parameter command

static MODBSTS mbcWrBuilderInfo(void)

Lines 2903 to 3117, Mapped all thetim parameters for the 0x54 write parameter command

MODBSTS modbus_decode

(

unsigned char ilen, /* Input ModBus message length (no CRC) */

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

unsigned char *icmd, /* Input ModBus message pointer */

unsigned char *olen, /* Output ModBus message length (no CRC) */

unsigned char *orsp /* Output (response) ModBus message pointer */

)

Rev 1.6.38	Rev 2.38
Starting line 3531 save_iec0 = IEC0; IEC0 = 0; /* Disable heart beat and DMA interrupt */ sts = readTIMarea(0x080, 0x0FF); /****** 9/11/2008 10:35AM ****** * restore interrupts **********/ IEC0 = save_iec0; /* Re-enable Heart Beat and DMA interrupts */ break; /****** 6/22/2009 8:17AM ****** * Write into the Scully reserve area ****/ case WRITE_THIRD_PARTY: /* 0x56 -- */ /****** 9/11/2008 10:33AM ****** * Disable interrupts **********/ save_iec0 = IEC0; IEC0 = 0; /* Disable heart beat and DMA interrupt */ sts = writeTIMarea(0x080, 0x0FF);	Starting line 2977 save_iec0 = IEC0; IEC0 = 0; /* Disable heart beat and DMA interrupt */ sts = readTIMarea(0xC00, 0x1FFF); /****** 9/11/2008 10:35AM ****** * restore interrupts **********/ IEC0 = save_iec0; /* Re-enable Heart Beat and DMA interrupts */ break; /****** 6/22/2009 8:17AM ****** * Write into the Scully reserve area ****/ case WRITE_THIRD_PARTY: /* 0x56 -- */ /****** 9/11/2008 10:33AM ****** * Disable interrupts **********/ save_iec0 = IEC0; IEC0 = 0; /* Disable heart beat and DMA interrupt */ sts = writeTIMarea(0xC00, 0x1FFF);

Changed ranges to enable the read and write to customer locations on the tim.

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

modfrc.c

MODBSTS mbxForce

(

unsigned int fbit, /* "Bit" or function code */

unsigned int fval /* Value/argument for bit/function code */

)

Rev 1.6.38	Rev 2.38
Starting line 429 if ((badvipflag & 0xFEFF)) /* VIP/Truck-ID bypass permissable? */ { if ((fval != MODBITON) /* Turning it on? */	Starting line 428 if ((badvipflag & 0xFF)) /* VIP/Truck-ID bypass permissable? */ { if ((fval != MODBITON) /* Turning it on? */

This change was made because we are now using bits in the upper byte of badvipflag to indicate the reason that the VIP did not authorize.

modreg.c

static unsigned mbrNonPermitReg (void)

Rev 1.6.38	Rev 1.6.36
Starting line 98 if (((badvipflag & 0xFEFF)) /* Truck ID? */ && (!(bylevel & VIP_BYPASS))) /* and not already bypassed? */ { /* Yes */ hval |= VIP_BYPASS; /* Non-Permit due to Unauthorized */ }	Starting line 98 if (((badvipflag & 0xFF)) /* Truck ID? */ && (!(bylevel & VIP_BYPASS))) /* and not already bypassed? */ { /* Yes */ hval |= VIP_BYPASS; /* Non-Permit due to Unauthorized */ }

This change was made because we are now using bits in the upper byte of badvipflag to indicate the reason that the VIP did not authorize.

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

MODBSTS mbrRdReg

(

unsigned int regno, /* 16-bit ModBus "Register" number */

unsigned int *value /* Pointer to return 16-bit register data */

)

Rev 1.6.38	Rev 1.6.36
Starting line 678 case 0x8: /* 068 -- VIP status (inc DateStamp) */ hval = badvipflag; break; case 0x9: /* 069 -- Service "A" flags */ hval = iambroke; /* Voltage error details, etc. */ break; case 0xA: /* 06A -- Service "B" flags */ hval = iamsuffering; /* Hard-wired relays etc. */ break; case 0xB: /* 06D -- bad compartment / probe count */ hval = (unsigned)(unsigned char)bad_compartment_count; break; case 0xC: /* 06B -- VIP status (inc DateStamp) */ hval = (unsigned)(((unsigned)(unsigned char)badvipdscode << 8) | ((unsigned)(unsigned char)badvipflag & 0xFF)); break; case 0xD: /* 06D -- Ground status */ hval = (unsigned)(unsigned char)badgndflag; break; case 0xE: /* 06C -- VIP status (inc DateStamp) */ hval = badvipflag; break; case 0xF: /* 06C -- VIP status (inc DateStamp) */ hval = cert_ds_fails; break;	Starting line 672 case 0x9: /* 069 -- Service "A" flags */ hval = iambroke; /* Voltage error details, etc. */ break; case 0xA: /* 06A -- Service "B" flags */ hval = iamsuffering; /* Hard-wired relays etc. */ break; case 0xC: /* 06C -- VIP status (inc DateStamp) */ hval = (unsigned)(((unsigned)(unsigned char)badvipdscode << 8) | ((unsigned)(unsigned char)badvipflag & 0xFF)); break; case 0xD: /* 06D -- Ground status */ hval = (unsigned)(unsigned char)badgndflag; break;
Starting line 793 case 0x03: /* 83 -- Active Deadman Warning time */ hval = (SysParm.DM_Warn_Start >> 2);	Starting line 771 case 0x03: /* 83 -- Active Deadman Warning time */ hval = (SysParm.DM_Warn_Start >> 2);

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

```
            break;

            case 0x04:                  /* 84 -- Software Feature
Enable Unload Term */
              if(SysParm.EnaSftFeatures &
ENA_UNLOAD_TERM)
              {
                 hval = 0xFF;
              }
              else
              {
                 hval = 0;
              }
            break;

            case 0x05:                  /* 86 -- Supertim max
unload time */
              hval = (SysParm.Unload_Max_Time_min);
            break;

            case 0x06:                  /* 86 -- Supertim cert date
enable mask */
              hval = (SysParm.Cert_Expiration_Mask);
            break;

            case 0x07:                  /* 87 -- Software Feature
Enable compartment count check */
              if(SysParm.EnaSftFeatures &
ENA_CPT_COUNT)
              {
                 hval = 0xFF;
              }
              else
              {
                 hval = 0;
              }
            break;

          default:                  /* Others are an error */
             return(MB_EXC_ILL_ADDR);
             break;
```

```
            break;

          default:                  /* Others are an error */
                    case 0x03:                  /* 83 -- Active
Deadman Warning time */
              hval = (SysParm.DM_Warn_Start >> 2);
             break;

          default:                  /* Others are an error */
             hval = MB_EXC_ILL_ADDR;
             break;
```

Starting line 1218

```
          case 0x1:                 /* 121 - Stop logging dome
out events */
            hval = (unsigned)disable_domeout_logging;  /*
*/
          break;



          case 0x2:                 /* 122 -  */
            hval = (unsigned)TIM_size;  /*  */
          break;

          case 0x3:                 /* 123 -  */
            hval = (unsigned)S_TIM_code;  /*  */
          break;

          default:                  /* Others */
            hval = MB_EXC_ILL_ADDR;
```

Starting line 1166

```
          case 0x1:                 /* 121 - Stop logging dome
out events */
            hval = (unsigned)disable_domeout_logging;  /*
*/
          break;

          default:                  /* Others */
            hval = MB_EXC_ILL_ADDR;
            break;
```

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

break;	

These changes were made for new modbus registers. We also added lines 1255 to 1470 for the holding registers for all the superTIM parameters.

MODBSTS mbrWrReg

(

unsigned int regno, /* 16-bit ModBus "Register" number */

unsigned int *value /* 16-bit "Register" data */

)

Rev 1.6.38	Rev 1.6.36
Starting line 1754 case 0x83: /* 83 -- Active Deadman Warning time */ if ((*value >= 10) && (*value <= 60)) { SysParm.DM_Warn_Start = (*value << 2); (void)nvSysParmUpdate(); } else { return (MB_EXC_ILL_DATA); /* reject bad values */ } break; case 0x84: /* 84 -- Software Feature Enable Code */ /* Enable/Disable Unload Terminal */ if (*value == 0) /* Disable requested */ { SysParm.EnaSftFeatures = (unsigned char) (SysParm.EnaSftFeatures & ~ENA_UNLOAD_TERM); } if (*value > 0) /* Enable requested */ { SysParm.EnaSftFeatures = (unsigned char) (SysParm.EnaSftFeatures | ENA_UNLOAD_TERM); } // (void)nvSysParmUpdate(); /* Force EEPROM update */ modNVflag++; /* Request EEPROM update */ break;	Starting line 1473 case 0x83: /* 83 -- Active Deadman Warning time */ if ((*value >= 10) && (*value <= 60)) { SysParm.DM_Warn_Start = (*value << 2); (void)nvSysParmUpdate(); } else { return (MB_EXC_ILL_DATA); /* reject bad values */ } break; case 0x100: /* 100 -- High-order system Time-Of-Day */ /* Wait for second half to do the actual write as an atomic operation. Just "stash" the high half for now; assume low half follows immediately after. */ wtmp = *value; /* Just hold on to high order half... */ break;

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

```
      case 0x85:                /* 85 -- Unload max time */
        SysParm.Unload_Max_Time_min = (unsigned
char)(*value);
        (void)nvSysParmUpdate();
      break;

      case 0x86:                /* 86 -- Cert DS enable mask */
       if ((*value >= 0) && (*value < 32))
       {
         SysParm.Cert_Expiration_Mask = (unsigned
char)(*value);
         (void)nvSysParmUpdate();
       }
       else
       {
         return (MB_EXC_ILL_DATA); /* reject bad values
*/
       }
      break;

      case 0x87:                        /* 84 -- Software Feature
Enable Code */
                                /* Enable/Disable Unload
Terminal  */
       if (*value == 0)                 /* Disable requested */
       {
          SysParm.EnaSftFeatures =
                (unsigned char) (SysParm.EnaSftFeatures
& ~ENA_CPT_COUNT);
       }
       if (*value > 0)                  /* Enable requested */
       {
          SysParm.EnaSftFeatures =
                (unsigned char) (SysParm.EnaSftFeatures |
ENA_CPT_COUNT);
       }
//       (void)nvSysParmUpdate();       /* Force EEPROM
update */
       modNVflag++;                     /* Request EEPROM
update */
       break;

     case 0x100:                      /* 100 -- High-order
system Time-Of-Day */
       /* Wait for second half to do the actual write as an
atomic operation.
          Just "stash" the high half for now; assume low
half follows
          immediately after. */
       wtmp = *value;                  /* Just hold on to high
order half... */
       break;
```

These changes were made for new modbus registers. We also added lines 1868 to 2387 for the holding registers for all the superTIM parameters.

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

permit.c

Rev 1.6.38	Rev 1.6.36
Starting line 383 if ((badvipflag & 0xFEFF)) /* Unauthorized truck ID ? */ { /* Yes */ if ((!(bylevel & VIP_BYPASS)) /* VIP (truck ID) bypass set? */ && (status)) /* and not locked out? */ { status = NOTNOW; /* No -- Bypassable non-permissive */ } }	Starting line 380 if ((badvipflag & 0xFF)) /* Unauthorized truck ID ? */ { /* Yes */ if ((!(bylevel & VIP_BYPASS)) /* VIP (truck ID) bypass set? */ && (status)) /* and not locked out? */ { status = NOTNOW; /* No -- Bypassable non-permissive */ } }

This change was made because we are now using bits in the upper byte of badvipflag to indicate the reason that the VIP did not authorize.

pod.c

Lines 347 to 491

Added the function void show_revision(void)

proto.h

Rev 1.6.38	Rev 1.6.36
Starting line 288 char flash_panel (void); void show_revision(void); char check_ref_volt (void);	Starting line 288 char flash_panel (void); char check_ref_volt (void);
Starting line 388 unsigned char fetch_serial_number(unsigned char tim_type, unsigned char *tim_number); void TIM_log_fault(unsigned int fault_val); void TIM_log_info(void); char superTIM_ds_validate(void); char check_unload_time(void); void check_compartment_count(void);	Starting line 387 unsigned char fetch_serial_number(unsigned char tim_type, unsigned char *tim_number);

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

These changes are to define the function prototypes for the added global functions.

stdsym.h

Rev 1.6.38	Rev 1.6.36
Starting line 271 # unsigned int DM_Warn_Start; /* 0x02022C Active Deadman Warning time */ unsigned char Cert_Expiration_Mask; /* 0x02022E supertim cert expiration mask */ unsigned int Unload_Max_Time_min; /* 0x02022F Allowable time for unload since load */ unsigned char free[0x0D]; /* //Fogbugz 131 0x020231 Round up to 64 bytes total (23E - current) */ unsigned int CRC; /* 0x02023E G.P. Parameter block CRC */	Starting line 271 unsigned int DM_Warn_Start; /* 0x02022C Active Deadman Warning time */ unsigned char free[0x10]; /* //Fogbugz 131 0x02022E Round up to 64 bytes total (23E - current) */ unsigned int CRC; /* 0x02023E G.P. Parameter block CRC */
Starting line 278 #define DM_OPEN (1*4) /* Active Deadman Max open time */	Starting line 276 #define DM_OPEN (3*4) /* Active Deadman Max open time */

The first change defines the certificate mask and unload time into the SysParmNV structure.

The second change changes the default dead man open time to 1 second instead of 3 seconds.

tim_util.c

unsigned char fetch_serial_number(unsigned char tim_type, unsigned char *tim_number)

Rev 1.6.38	Rev 1.6.36
Starting line 474 case TEST_TIM : valid_address = 0x404; tim_address = 0x405; break; case ALT_TIM : valid_address = ALT_TIM_ID_VALID_ADDR; tim_address = ALT_TIM_ID_ADDR; break; default: return MB_EXC_TIM_CMD_ERR;	Starting line 463 case TEST_TIM : valid_address = VALID_TRUCKTEST_TIM_ADR; tim_address = TRUCK_TEST_TIM_NUMBER_ADR; break; case ALT_TIM : valid_address = VALID_ALT_TIM_ADR; tim_address = ALT_TIM_NUMBER_ADR; break; default: return MB_EXC_TIM_CMD_ERR;

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

Starting line 497 if ((sts = (unsigned char)tim_block_read(tim_number, TABLE_VALID_ADDR, TABLE_VALID_SIZE)) != MB_OK) { return (sts); }	Starting line 486 if ((sts = (unsigned char)tim_block_read(tim_number, VALID_ENTRIES_ADR, VALID_ENTRIES_SIZE)) != MB_OK) { return (sts); }

These two changes are for the remapped tim parameters

In lines 525 to 814 we added the following functions:

- void TIM_log_fault(unsigned int fault_val)
- void TIM_log_info(void)
- char superTIM_ds_validate(void)
- char check_unload_time(void)
- void check_compartment_count(void)

tim_util.h

There are too many changes to list. All changes were to map and size the new tim parameters.

trukstat.c

static void truck_idle(void)

Rev 1.6.38	Rev 1.6.36
Start line 260 print_once_msg &= ~UN_AUTH; /* Clear all entries in idle loop */ TIM_fault_logged = 0; TIM_info_logged = 0; bad_compartment_count = 0; StatusB &= ~STSB_TRUCK; /* Clear truck valid */	Start line 258 print_once_msg &= ~UN_AUTH; /* Clear all entries in idle loop */ StatusB &= ~STSB_TRUCK; /* Clear truck valid */
Start line 712 if ((bypass_state = bypass_operation()) != 0) { if((StatusA & STSA_FAULT)) // if in system fault do a reset { secReset = 1; /* Force immediate board-level RESET */ // for(;;); // let the watchdog reset us	Start line 708 if ((bypass_state = bypass_operation()) != 0) { if (bypass_state == 2) /* good bypass key */ { // printf("\n\r***** 07 ******\n\r"); status = TRUE;

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

} if (bypass_state == 2) /* good bypass key */ { // printf("\n\r***** 07 ******\n\r"); status = TRUE;	

The first change resets the flags to determine what active super tim functions to call.

The second change does an immediate reset on the hardware when a bypass key is scanned in a system error state.

static void truck_active(void)

Rev 1.6.38	Rev 1.6.36
Start line 1291 StatusO &= ~0x7; /* Clear VIP output status */ ledstate[VIP_IDLE] = DARK; ledstate[VIP_AUTH] = LITE; ledstate[VIP_UNAUTH] = DARK; StatusO |= STSO_AUTHORIZED; } } } } } //SME if(TIM_size == DS28EC20_SIZE) // Super TIM { if(TIM_info_logged == 0) { TIM_log_info(); TIM_info_logged = 1; } if(TIM_fault_logged == 0) { if((StatusA & STSA_PERMIT) == 0) { if((badgndflag & GND_PROBLEMS) || (tank_state == T_WET)) { fault_num = 0; if(badgndflag & GND_PROBLEMS) { fault_num = 25; } else { for(i = 0; i < 16; i++) { if((probes_state[i] != P_UNKNOWN) && (probes_state[i] != P_DRY))	Start line 1280 StatusO &= ~0x7; /* Clear VIP output status */ ledstate[VIP_IDLE] = DARK; ledstate[VIP_AUTH] = LITE; ledstate[VIP_UNAUTH] = DARK; StatusO |= STSO_AUTHORIZED; } } } } } TIM_timer = (read_time() + 330); /* Reset the timer for 1/3 second */

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

{ fault_num = i + 1; break; } } } if(fault_num != 0) { TIM_log_fault(fault_num); TIM_fault_logged = 1; } } } } } TIM_timer = (read_time() + 330); /* Reset the timer for 1/3 second */	
Start line 1383 case OPTIC_TWO: /* If here, we have a 2 wire optic truck */ xprintf(45, DUMMY); /* as determined in optic_2_setup */ (void)read_bypass(TRUE); /* Read any bypass chip */ if ((SysParm.Ena_Debug_Func_1 == 0x32) || (SysParm.Ena_Debug_Func_2 == 0x32) || (SysParm.Ena_Debug_Func_3 == 0x32) || (SysParm.Ena_Debug_Func_4 == 0x32)) { debug_pulse(0x32); } check_compartment_count(); // (void)compartment_check(); /* Validate sensor count with SuperTIM */ active_two_wire(OPTIC2); break;	Start line 1314 case OPTIC_TWO: /* If here, we have a 2 wire optic truck */ xprintf(45, DUMMY); /* as determined in optic_2_setup */ (void)read_bypass(TRUE); /* Read any bypass chip */ if ((SysParm.Ena_Debug_Func_1 == 0x32) || (SysParm.Ena_Debug_Func_2 == 0x32) || (SysParm.Ena_Debug_Func_3 == 0x32) || (SysParm.Ena_Debug_Func_4 == 0x32)) { debug_pulse(0x32); } // (void)compartment_check(); /* Validate sensor count with SuperTIM */ active_two_wire(OPTIC2); break;
Start line 1399 /************************** 5/11/2009 6:52AM ************************* * The compartment count can be faked out by a wet probe. So test for * return pulse. If the pulse does not return the last probe is wet. If * it does return all probes are functioning and valid compartment count *** ********/ check_compartment_count(); // (void)compartment_check(); active_5wire();	Starting line 1339 /************************** 5/11/2009 6:52AM ************************* * The compartment count can be faked out by a wet probe. So test for * return pulse. If the pulse does not return the last probe is wet. If * it does return all probes are functioning and valid compartment count *** ********/ // (void)compartment_check(); active_5wire();
Start line 2129 if (S_TIM_code) /* is the TIM a Super TIM? */ { read_TIM_Go_NoGo_info(); }	Start line 2066 if (S_TIM_code) /* is the TIM a Super TIM? */ { read_TIM_Go_NoGo_info(); } sts = nvTrkFind (&truck_SN[0], (word *)&index); /* Check local list of good guys */

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

```
  if( (SysParm.EnaSftFeatures &
ENA_UNLOAD_TERM) &&
(SysParm.Unload_Max_Time_min != 0) )
  {
      sts = check_unload_time();
      if( sts )
      {
        StatusA &= ~STSA_TRK_VALID;       /* Truck not
authorized! */
//        badvipflag |= BVF_DSNOAUTH;      /* No
Dallas/IO problems (0x08) */
        badvipflag |= BVF_UNLOAD_EXP;      /* No
Dallas/IO problems (0x08) */
        return;                           /*  we can't do much more
here */

      }
  }

  if( SysParm.Cert_Expiration_Mask != 0)
  {
    sts = superTIM_ds_validate();
    if (sts == (int)DSEXPIRED )  /* or an invalid terminal
*/
    {
      /* DateStamp authoritatively rejects this truck, so his
         only hope is a VIP-Bypass operation */
      badvipflag &= ~BVF_DSERROR; /* No Dallas/IO
problems */
      badvipflag |= BVF_DSNOAUTH; /* DateStamp says
No! */
      badvipdscode = sts; /* Detailed rejection code */
      return;
    }
    else
    {
      if (sts)          /* If can't access TIM/file */
      {
        /* May be "random" I/O error, try again later and
maybe
           it'll work the next time round */
        badvipflag |= BVF_DSERROR; /* Errors accessing
DateStamp */
        badvipdscode = sts; /* Detailed error code */
        val_state = 0;       /* Enable retry later */
        return;
      }
      else
      {
        badvipflag &= ~BVF_DSERROR; /* No error now
*/
        badvipdscode = 0;   /* This truck authorized */
      }
    }

  }

  sts = nvTrkFind (&truck_SN[0], (word *)&index); /*
Check local list of good guys */
```

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

The first change handles logging the connection data and fault to the super TIM.

The second and third changes we for calling the compartment count check.

The fourth change calls the unload time check and certificate check if they are enabled

modbus.h

Rev 1.6.38	Rev 1.6.36
Starting line 60 #define USE_UPDATED_ADC_TABLE 0x5C	

Added modbus command 5C to switch between old calc_tank adc table and new updated table for probe counting.

volts.h

Rev 1.6.38	Rev 1.6.36
Starting line 224 extern unsigned long lowVolt;	

Global variable that stores the lowest adc voltage from calc_tank until intellitrol is in a permit state.

stsbits.h

Rev 1.6.38	Rev 1.6.36
Starting line 105 /* Channel 5 diag line resistance is higher than expected - calc_tank()*/ #define CH5_HIGH_RESISTANCE 0x0200	

Variable used for storing calc_tank error in event log.

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

version.h

Rev 1.6.38	Rev 1.6.36
Starting line 194 #define MAJVER 01 #define MINVER 06 #define EDTVER 38 #define EDTVERHEX 0x38 #define SHELLVER (((unsigned int)MAJVER << 12) | ((unsigned int)MINVER << 8) | (unsigned int)EDTVERHEX)	

Updated version to 1.6.38 and fixed issue with displaying incorrect version when converting from hex to decimal.

proto.h

Rev 1.6.38	Rev 1.6.36
Starting line 244 char nvSysDia5Update (unsigned int updatedADCTable);	Starting line 244 char nvSysDia5Update (void);
Starting line 316 void logmaintenanceerr(void);	

Modified nvSysDia5Update function to allow for selection of old adc table or new table.
Created logmaintenanceerr function to push maintenance errors to the event log.

evlog.h

Rev 1.6.38	Rev 1.6.36
Starting line 161 char future[16]; /* Reserved for future */	Starting line 161 char future[18]; /* Reserved for future */
Starting line 224 #define EVIMAINTENANCE 0x08 typedef struct {	Starting line 224 #define EEFMT 0x08 typedef struct {

unsigned int ch5_high_resistance; /* Channel 5 resistance is higher than expected calc_tank() */ char future[20]; /* Reserved for future */ } EVI_MAINTENANCE;	char future[22]; /* Reserved for future */ } EEFMT_INFO;
Starting line 249 char future[2]; /* Reserved for future */	
Starting line 263 char future; /* Reserved for future */	

Created EVI_MAINTENANCE for displaying maintenance errors to the event log. Added or modified future arrays to correctly size each structure.

stdsym.h

Rev 1.6.38	Rev 1.6.36
Starting line 309 typedef struct { unsigned Reference; /* "6.759" volt reference level */ unsigned PNOffset; /* Transistor PN-junction bias/offset */ unsigned WetVolts[16]; /* 16-probe "wet" level */ unsigned int updatedADCTable; /* Switch to use updated ADC table for probe counting in calc_tank, 1 = new table, 0 = old table */ /* DateStamp + SysDia5 = 64 bytes total! */ unsigned CRC; /* SysDia5 block CRC */ } SysDia5NV;	Starting line 304 typedef struct { unsigned Reference; /* "6.759" volt reference level */ unsigned PNOffset; /* Transistor PN-junction bias/offset */ unsigned WetVolts[16]; /* 16-probe "wet" level */ char free[2]; /* Round up to 40 bytes total */ /* DateStamp + SysDia5 = 64 bytes total! */ unsigned CRC; /* SysDia5 block CRC */ } SysDia5NV;

Replaced free array with updatedADCTable. updatedADCTable is used to select the old ADC table or the new table to be used in calc_tank.

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

permit.c

Rev 1.6.38	Rev 1.6.36
Starting line 303 lowVolt = 9999; // Used for calc_tank() probe compartment count	

Reset lowVolt to 9999 when in dry permit state.

specops.c

Rev 1.6.38	Rev 1.6.36
Starting line 342 ledstate[NONPERMIT] = DARK; ledstate[PERMIT] = LITE; service_wait(16);	

Flash the green permit led when bypass key is successfully added.

modcmd.c

Rev 1.6.38	Rev 1.6.36
Starting line 3727 case USE_UPDATED_ADC_TABLE: /* 0x5C -- Use updated ADC table for probe counting, 1 = new table, 0 = old table */ if(pSysDia5->updatedADCTable == 0) { nvSysDia5Update(1); // Update eeprom block sts = mbcPutByte (0x01); // Return 01 } else { nvSysDia5Update(0); // Update eeprom block	

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

sts = mbcPutByte (0x00); // Return 00 } break;	

Added modbus command 5C to swap to other adc table for use in calc_tank. Return 5C00 for old table and 5C01 for new table.

main.c

Rev 1.6.38	Rev 1.6.36
Starting line 1398 /*** * logmaintenanceerr() -- Create an Event Log entry for maintenance errors ***/ void logmaintenanceerr() { EVI_MAINTENANCE maintenance; maintenance.ch5_high_resistance = 1; /* Channel 5 resistance is higher than expected calc_tank() */ nvLogRepeat (EVIMAINTENANCE, 1, (char *)&maintenance, (unsigned long)(4 * 60 * 60)); }	

Created function logmaintenanceerr to push a maintenance error to the event log.

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

optic5.c

Rev 1.6.38	Rev 1.6.36
Starting line 324 lowVolt = 9999;	
Starting line 344 for (i = 0; i < 5; i++) { DelayMS(30); /* ensure minimum period */ if (try_five_wire() == TRUE) { probe_flag = TRUE; number_of_Probes = (unsigned int)(calc_tank() - 1); DelayMS(100); /* wait maximum period */ service_charge(); /* Appease watchdog */ } }	Starting line 343 for (i=0; i<10; i++) { DelayMS(30); /* ensure minimum period */ if (try_five_wire() == TRUE) { probe_flag = TRUE; number_of_Probes = (unsigned int)((unsigned char)calc_tank()-1); DelayMS(100); /* wait maximum period */ service_charge(); /* Appease watchdog */ if ((number_of_Probes != 0) && (number_of_Probes == (unsigned int)((unsigned char)calc_tank()-1))) { break; /* Break from the loop */ } } }
Starting line 378 lowVolt = 9999;	
Starting line 755 #define TRIALS 10 /* number or repeats to do the averaging */ unsigned long lowVolt = 9999; unsigned int calc_tank(void) { unsigned long ch5_volt; unsigned long ch5_volt_oldTable; unsigned long index; unsigned int tank_number; unsigned long voltList[17] = {6840, 6630, 6380, 6000, 5690, 5385, 5130, 4890, 4660,	Starting line 760 #define TRIALS 8 /* number or repeats to do the averaging */ unsigned int calc_tank(void) { unsigned long ch5_volt; unsigned long index; unsigned int tank_number; char error_found = 0; // last_routine = 0x3C; DelayMS(30); /* ensure minimum period */

```
4470, 4270, 4105, 3950, 3795, 3675, 3550, 3440};
  char        error_found = 0;

  StatusA &= ~CH5_HIGH_RESISTANCE;

  //printf("Using table number %d\n", pSysDia5->updatedADCTable);

  DelayMS(30);                    // ensure minimum period
  optic_5_pulse();                // Pulse optic probe to get reading
  tank_number = 0;
  ch5_volt = 0;
  CH_TEST5 = 0;                   // Turn off Ch 5 (DIAG channel)
  DIAGNOSTIC_EN = 0;              // Turn on precision DIAG voltage

  if (read_ADC() == FAILED)
  {
    printf("\n\r3: Trouble reading the Analog Port\n\r");
    Init_ADC();
    return 18;                    // Since we can't read the voltage we call it a invalid probe
  }

  for (index = 0; index < TRIALS; index++ ) // Average TRIALS
  {
    if (read_ADC() == FAILED)
    {
      printf("\n\r4: Trouble reading the Analog Port\n\r");
      Init_ADC();
      return 18;                  // Since we can't read the voltage we call it a invalid probe
    }
    optic_5_pulse();              /* Pulse optic probe to get reading */
    tank_number = 0;
    ch5_volt = 0;
    CH_TEST5 = 0;                 /* Turn off Ch 5 (DIAG channel) */
    DIAGNOSTIC_EN = 0;            /* Turn on precision DIAG voltage */
    if (read_ADC() == FAILED)
    {
      printf("\n\r3: Trouble reading the Analog Port\n\r");
      Init_ADC();
      return 18;                  /* Since we can't read the voltage we call it a invalid probe */
    }

    for (index=0; index<TRIALS; index++ )   /* Average TRIALS */
    {
      if (read_ADC() == FAILED)
      {
        printf("\n\r4: Trouble reading the Analog Port\n\r");
        Init_ADC();
        return 18;                /* Since we can't read the voltage we call it a invalid probe */
      } else
      {
        ch5_volt += probe_volt[4];
        optic5_table[index] = probe_volt[4];
      }
    }
    CH_TEST5 = 1;
    DIAGNOSTIC_EN = 1;
    ch5_volt /= index;                          /* 5-wire-optic diagnostic voltage */
    ch5_volt += (unsigned long)pSysDia5->PNOffset;     /* Offset by switching transistor */
    ch5_volt *= (unsigned long)ReferenceVolt;  /* Calibrated x1000 (check_ref_volt()) */
```

```
      else
      {
         ch5_volt += probe_volt[4];
         optic5_table[index] = probe_volt[4];
      }
   }

   CH_TEST5 = 1;
   DIAGNOSTIC_EN = 1;
   ch5_volt /= index;                    // 5-wire-
optic diagnostic voltage

   // Add offset only for old table
   ch5_volt_oldTable = ch5_volt;
   ch5_volt_oldTable += (unsigned
long)pSysDia5->PNOffset;
   ch5_volt_oldTable *= (unsigned
long)ReferenceVolt;
   ch5_volt_oldTable /= (unsigned long)1000;

   ch5_volt *= (unsigned long)ReferenceVolt;
   ch5_volt /= (unsigned long)1000;

   // New Table
   if(pSysDia5->updatedADCTable == 1)
   {
      if(ch5_volt < lowVolt)
      {
         lowVolt = ch5_volt;
      }

      //printf("LOW VOLTAGE: %d\n",
(int)lowVolt);
      compare_volts = lowVolt;

      for (index = 0; index < 17; index++)
      {
         if (lowVolt <= voltList[index])
         {
            tank_number++;
         }
      }
  ch5_volt /= (unsigned long)1000;
  for (index=0; index<16; index++)
  {
   tank_number++;
   if ( pSysDia5 != 0)
   {
    if ((unsigned)ch5_volt > pSysDia5-
>WetVolts[index])
    {
     error_found = 1;
     break;
    }
   }
  }
  if ( error_found == 0)
  {
   /*****************************
1/22/2010 2:52PM
*************************
    * If no error found then it must be for
sizing a truck that has 16
    * compartments. Because the result is
subtracted by one we must add one to it

***********************************
***********************************/
   tank_number++;     /* Because the result is
subtracted by one we must add one to it */
  }
  return(tank_number);
} /* end of calc_tank */
```

```
      if(tank_number > 1 && tank_number <
16) {
         if (lowVolt > ((((voltList[tank_number -
1] - voltList[tank_number]) * 25) / 100) +
voltList[tank_number])
               || (lowVolt >=
voltList[tank_number] && lowVolt <
(voltList[tank_number] + (unsigned long)5))) {
            //printf("Please check sensor
connection: %d\n",
(int)((((voltList[tank_number - 1] -
voltList[tank_number]) * 25) / 100) +
voltList[tank_number]));
            StatusA |= CH5_HIGH_RESISTANCE;
            logmaintenanceerr();
         }
      }

      //printf("TABLE VALUE %d: %d\n",
tank_number - 1, (int)voltList[tank_number -
1]);
      //printf("TABLE VALUE %d: %d\n\n",
tank_number, (int)voltList[tank_number]);
   }

   // Old Table
   else {
      lowVolt = 9999;

      //printf("CHANNEL 5 VOLTAGE: %d\n",
(int)(ch5_volt_oldTable));
      compare_volts = ch5_volt_oldTable;

      for (index = 0; index < 16; index++)
      {
         tank_number++;

         if (pSysDia5 != 0)
         {
            if (ch5_volt_oldTable > pSysDia5-
>WetVolts[index])
```

```
            {
               error_found = 1;
               break;
            }
         }
      }
      if (error_found == 0)
      {
         /*****************************
1/22/2010 2:52PM
*************************
         * If no error found then it must be for
sizing a truck that has 16
         * compartments. Because the result is
subtracted by one we must add one to it

***********************************
***********************************/
         tank_number++;    /* Because the
result is subtracted by one we must add one
to it */
      }
   }

   return(tank_number);
}
```

Rewrote calc_tank to fix probe counting errors. Old function and ADC table only work with perfect conditions and do not take real life variables into account. Created new ADC table to better match the real world and will work with best case scenarios while improving performance in worst case situations. lowVolt was added to only using the lowest voltage seen instead of using the live voltage. Using the lowest voltage gives much more consistent results and we found that the voltage can only go so low in all situations, while the highest voltage is widely unpredictable in worst case situations. This update in conjunction with regularly cleaning the socket greatly increases the accuracy in worst case situations. Modbus command 5C was added to allow the user to switch back to the old calc_tank function if desired. Modbus command 5B allows the user to run the calc_tank function at any time and have the number of probes returned to them.

Scully Signal Company
70 Industrial Way, Wilmington, MA 01887-3479, USA • 800.2.SCULLY (272.8559)
617.692.8600 • fax. 617.692.8620 • sales@scully.com • www.scully.com

nvsystem.c

Rev 1.6.38	Rev 1.6.36
Starting line 74 1, /* Switch to use updated ADC table for probe counting in calc_tank, 1 = new table, 0 = old table */	Starting line 74 {0}, /* Reserved, MBZ */
Starting line 133 sts = nvSysDia5Update(1);	Starting line 133 sts = nvSysDia5Update();
Starting line 403 char nvSysDia5Update (unsigned int updatedADCTable) { unsigned int crc; char sts; memcpy((char *)&SysNonV.Dia5Block, (char *)&SysDia5Default, sizeof(SysDia5Default)); SysNonV.Dia5Block.updatedADCTable = updatedADCTable; crc = modbus_CRC ((unsigned char *)&SysNonV.Dia5Block, sizeof(SysDia5NV) - 2, INIT_CRC_SEED); SysNonV.Dia5Block.CRC = crc; sts = eeBlockWrite((unsigned long)SysDia5Adr + 0x100, (unsigned char *)&SysNonV.Dia5Block.Reference, sizeof(SysDia5Default)); // last_routine = 0x21; return (sts); /* Propagate success/failure */ }	Starting line 403 char nvSysDia5Update (void) { unsigned int crc; char sts; memcpy((char *)&SysNonV.Dia5Block, (char *)&SysDia5Default, sizeof(SysDia5Default)); crc = modbus_CRC ((unsigned char *)&SysNonV.Dia5Block, sizeof(SysDia5NV) - 2, INIT_CRC_SEED); SysNonV.Dia5Block.CRC = crc; sts = eeBlockWrite((unsigned long)SysDia5Adr + 0x100, (unsigned char *)&SysNonV.Dia5Block.Reference, sizeof(SysDia5Default)); // last_routine = 0x21; return (sts); /* Propagate success/failure */ }

Modified nvSysDia5 structure and nvSysDia5Update function to allow for selection of old or new ADC voltage table for use in calc_tank function.