Control equipment from anywhere, anytime, using SMS and an old Nokia mobile phone! – By Peter Smith

SMS Controller Pt.2

Last month, we described the circuit for the SMS Controller and gave the assembly details. This month, we tell you how to complete the circuit checking and describe how the unit is used.

Having carried out the power supply checks described last month in Part 1, the next step is to check out the serial interface and the microcontroller.

First, disconnect the power and insert IC1 and IC3 into their sockets. If the microcontroller needs to be programmed, then you should do that next. Refer to the Microcontroller Programming panel for more details on this.

Next, install jumper shunts on JP4 to JP7. These should always be in place when the inputs (IN1 - IN4) are not connected, otherwise the micro's port pins will be 'floating' in an indeterminate logic state.

Conversely, remove all jumper shunts from JP1-JP3 if installed earlier and apply power. The 'Comms Error' LED (LED1) should illuminate, while all other LEDs (except the 'Power' LED) should be off. This indicates that the micro cannot communicate with the phone, which of course isn't connected yet. However, it does tell us that the micro is doing what it should.

Note: the very first time you apply power, all red LEDs may come on for one second and then go out, with just the 'Comms Error' LED remaining on. This sequence indicates that the micro has automatically erased its on-board EEPROM, ready for programming.

If you get a different result, the problem is most likely due to one or more pins of the micro having missed their sockets and bent underneath the chip.

If bent pins aren't the problem, then check out the oscillator circuit, consisting of crystal X1 and the two 22pF capacitors. If you have access to an oscilloscope, you can observe the operation of the oscillator on pin 18. In addition, check the voltage on the micro's RESET input (pin 9). This pin should measure close to +5V during normal operation, going low only during power up and power down.

The final step involves a quick checkout of the RS232 interface circuit. Measure the voltage between pin 2 of IC4 and ground and pin 6 of IC4 and

ground. You should get around +9.5V and -9.3V, respectively. These voltages are generated by the MAX232's internal charge pumps, in conjunction with the four 1µF capacitors.

If your board passes all the tests, you can now connect the data cable between your board and phone. Note that's it's a good idea to power off both devices when connecting and disconnecting this cable.

Suitable case

If desired, the completed module can be housed in a 'UB1' size plastic box or similar, with a slot cut in the side of the box to accommodate the terminal block wiring.

The mobile phone must be positioned at least 50cm from the controller and associated wiring so that RF energy from its antenna doesn't interfere with the circuit operation. **This is very important**! If this separation cannot be attained in your application, then the controller must be housed in a metal case or shielded from the phone with a large metal plate.

Alternatively, both the 5110 and 6110 models support connection of an external antenna, which would allow good separation and improve signal strength in some areas.

On the 3310 model, the serial interface is accessible through a hole in the rear of the case, underneath the battery. The data cable is terminated with a plastic head assembly that includes a set of spring-loaded contacts as well as tabs to retain the battery that it partly displaces.

Operational basics

System operation is quite straightforward – when any of the digital inputs change state, the controller sends a pre-programmed SMS message to the nominated mobile number. Conversely, when you want to turn any of the outputs on or off, you send a message to the controller.

The messages used in both directions are programmed during the setup procedure. This allows the use of messages related to the task at hand. For example, you might want to assign the message 'pump' to turn on the first output and 'nopump' to turn it off. This means that you don't need to remember which output the pump is connected to or which state (high or low) is on or off.

The controller also recognises a number of unique messages, called 'in-built commands', that can be sent from another mobile to program the system during setup, as well as modify system behaviour during normal operation. A summary of all these commands appears in Table 2. Before we look at how to set up the controller, let's look at each command in detail.

In-built commands

ACKON – this command forces the controller to respond to every message that it receives. If a received message is deemed valid, the controller responds with 'OK'. If a message is unknown, the response is 'bad cmd'.

ACKOFF – the opposite of **ACKON**. All further acknowledgments are disabled.

CHARGE{number} – this command allows you to modify the battery charging parameters, dependent on the model of phone in use.

For the 5110 and 6110, the {number} value defines the battery level at which the on-board charger is switched on. Only values between 0 and 4 are valid. A value of 4 instructs the controller to continually charge the phone and is therefore not recommended. The default level is 1.

For the 3210 and 3110, a timed charge/discharge scheme is used instead, as battery level information is not available to the controller. The {number} value defines the charge time in minutes, with the discharge time being fixed at 8 hours. Only values between 10 and 240 are valid. The default charge time is 40 minutes.

COUNT – Use this command to get the total number of messages sent and received by the controller, as well as the firmware version number. The returned message is in the format 'r=nnnnn s=nnnnn v=nn.nn', where 'r' and 's' are the total number of received and sent messages, respectively.

DIS{string} – in some situations, you may not want to be informed when a particular input changes state but still want to receive notification on the remaining inputs. An example of this might be when one sector of an alarm system is faulty and has been isolated. Using this command, you can disable notification on either or both states of any input.

For example, suppose a message of 'SECTOR1ALARM' is programmed to be sent when an input goes low and a complementary message of 'SECTOR1IDLE' is programmed to be sent when it returns high. To stop receiving these messages each time the input toggles, you could send the commands

Microcontroller Programming

You can purchase a ready programmed microcontroller (IC1) from Magenta Electronics – see page 5. Alternatively, you'll need to program the microcontroller yourself.

A 10-way header (CON5) has been included on the PC board for connection to an 'in-system' type programmer.

Once you have a suitable programmer, together with the necessary cables and Windows software to drive it, all you need to complete the job is a copy of the microcontroller program for the SMS Controller. This can be downloaded from our web site – go to the 'Downloads' section.

This contains the file 'SMS.HEX', which needs to be programmed into the micro's program (FLASH) memory. Just follow the instructions provided with the programmer and software to complete the task.

Fuse bits

We've specified either AT90S8515-8 or ATMega8515-16 microcontrollers for this project. Although it has many improvements over its predecessor, the ATMega8515 is a pin-for-pin replacement for the AT90S8515. In fact, we've tested this project with both of these devices to ensure compatibility.

The only additional requirement when using the ATMega8515 is to ensure that the fuse bits are correctly programmed. The default fuse settings in the AT90S8515 are OK and should not be altered.

'**DISSECTOR1ALARM**' followed by '**DISSECTOR1IDLE**'.

EN{string} – the opposite of DIS{string}, this command reinstates notification on the input and state designated by {string}.

LOGIN{pass} – essentially, this command tells the controller your current mobile phone number. You don't actually need to enter the number, as it's automatically gleaned from the incoming message. All messages are forwarded to the mobile phone that sent the last **LOGIN** command, which remains valid until another **LOGIN** or **LOGOUT** command is received.

If a password had been set, it must immediately follow the **LOGIN** command. An exception to this is in programming mode (JP3 in), where password checking is not performed.

LOGOUT – this command disables all outgoing messages. It's wise to send this command to the controller before you switch off your phone. If your phone's battery goes flat, or it's stolen or misplaced, use a friend's phone to first **LOGIN** and then **LOGOUT**. If you don't, a malfunctioning system could see you rack up a phone bill of astronomical proportions – a compelling reason to use only a prepaid plan for the phone connected to the controller (see panel in Part 1)!

PASS{string} – sets a new password of 1 to 8 characters long. Passwords longer than 8 characters elicit a 'bad pass' response. The initial password is programmed during the setup procedure. Once set, it can be changed at any time but only from the currently logged-in phone (see **LOGIN** command).

STAT – returns the current state of the digital input and open-collector output ports. The displayed format is 'XXXX YYYYYYYY', where 'X' and 'Y' are 'H' for logic high or 'L' for logic

Table 2: Command Summary

Command	Function
ACKON	Enable acknowledge messages
ACKOFF	Disable acknowledge messages
CHARGE{number}	Modify battery charge level (5110 & 6110) or charge time (3210 & 3310)
COUNT	Get SMS sent & received counters & firmware version number
DIS{string}	Disable state change messages on input defined by {string}
EN{string}	Enable state change messages on input defined by {string}
LOGIN	Enable message transmissions to your current mobile number
LOGOUT	Disable further message transmissions to your mobile
PASS{string}	Set new password to {string} (8 characters max.)
STAT	Get snapshot of digital inputs & open-collector outputs
The following commands are valid only in programming mode (JP3 in):	
IN{n}{L}{message}	Define the message the controller sends when input {n} goes low
IN{n}{H}{message}	Define the message the controller sends when input {n} goes high
OUT{n}{L}{message}	Define the message you send to drive output {n} low
OUT{n}{H}{message}	Define the message you send to drive output {n} high
OUT{n}{P}{message}	Define the message you send to pulse output {n} low

Here's a summary of the commands recognised by the controller. The curly braces are used here for clarity and should not be included in your messages. Note: do not use spaces after command words.

A complete lineup of the supported models, from left to right: 5110, 6110, 3210 and 3310.

low. The input port is displayed first, followed by the output port, with the most-significant bits (IN4 and OUT8) displayed first.

For the output port, an 'H' (high) indicates the driver is switched off, whereas an 'L' (low) indicates it is on. Note that external circuits may invert this logic.

A response from the **STAT** command looks like this: 'HHLH HLHHHHHH'. In this case, IN2 is low and IN1, IN3 and IN4 are high. On the output side, OUT7 in on (low) and all other drivers are off (high).

The following commands operate only in programming mode (JP3 in):

IN{n}{L}{message} – defines the message that will be sent by the controller when input {*n*} goes low. For example, suppose you've connected a switch to the first input (IN1), as shown in Fig.7(b) – last month. When the switch is closed, the input changes state from a logic high (+5V) to a logic low (0V). To receive the message 'SWITCH CLOSED', the required command would be **IN**1L**SWITCH CLOSED**.

Of course, you can use any message you like, as long as it's no more than 16 characters long.

IN{n}{H}{message} – defines the message that will be sent by the controller when input {n} goes high. Using the previous example, to receive the message 'SWITCH OPEN' when the first input (IN1) changes from a logic low to a logic high, the required command would be: **IN**1H**SWITCH OPEN**.

OUT{n}{L}{message} – defines the message that you send to the controller to drive output {n} low. For example, suppose you've connected a relay to OUT1, as shown in Fig.6(a) – last month. A low on this output grounds one end of the relay coil, switching it on. Assuming you want to use the message 'RELAY ON', the required command would be **OUT**1L**RELAY ON**.

OUT{n}{H}{message} – defines the message that you send to the controller to drive output {n} high. From the previous example, to switch the relay off with the message 'RELAY OFF', the required command would be **OUT**-1H**RELAY OFF**.

OUT{n}{P}{message} – defines the message that you send to the controller to pulse output {n}. When the controller receives this message, the specified output will be driven low for one second, after which it returns high. Again from the previous examples, to pulse a relay on OUT1 using the message 'PULSE RELAY', the required command would be **OUT**1P**PULSE RELAY**. It's important to note that when any output is defined as a 'pulsed' output, you cannot also define it with the **OUT{n}{L}** or **OUT{n}{H}** commands.

Message syntax

Messages used in the '**IN**' and '**OUT**' commands can be composed from any characters in the available repertoire but the total length must be limited to 16 characters. Longer messages are automatically truncated. Spaces can be used in the body of messages **but not adjacent to the command or password strings (ie, DO NOT use spaces after command words).**

In addition, all user-defined messages, including the password, are *case sensitive*. This is a trap for the unwary; 'PUMP ON' and 'pump on' are not the same message! Inbuilt commands, on the other hand, are not case sensitive.

Finally, your messages must not start with the in-built command words **ACKON, ACKOFF, CHARGE, COUNT, DIS, EN, LOGIN, LOGOUT, PASS** or **STAT**.

Example setup

Let's look at a fictitious system setup to see how it all works. The specifications for this system are as follows:

- All commands to the controller must be acknowledged.
- The system is to be password protected. The initial password will be 'REDDWARF'.

LED Indicators

Five LEDs are provided to indicate system status; four red (LED1 – LED4) and one green (LED5). The red LEDs indicate error conditions, so during normal operation none of them should be on.

LED1 – Comms Error: when illuminated, this LED indicates a controller to phone communications problem. Normally, it comes on for 6 seconds after power on and then goes out. If it doesn't go out, check for problems with the controller to phone cable connection. In addition, check that phone security (PIN) has been disabled and that the phone comes up ready for use at power on.

LED2 – No Service: indicates that the phone is not registered within the mobile network (check signal strength) or that an outgoing message has been disallowed. The latter is typically due to an empty pre-paid account.

Note that although your service provider will block outgoing messages when an account expires, most still allow inbound messages for a certain length of time.

LED3 – Send Error: when illuminated, the controller has failed to send one or more messages. This can be caused by a variety of problems, including mobile network overload, momentary signal dropout, an empty pre-paid account, phone malfunction or an intermittent controller to phone connection.

LED4 – Delete Error: indicates that the controller cannot delete a message from SIM memory. Cycle the phone power to correct this problem. If the error persists, then there may be a problem with the SIM card or phone.

LED5 – In Use: this LED comes on when you login to the system and goes out when you logout.

- A relay is connected to OUT1, as shown in Fig.6(a) – last month. The relay controls a pump motor.
- The relay is to be switched on by sending 'PUMP ON' to the controller.
- The relay is to be switched off by sending 'PUMP OFF' to the controller.
- A switch is connected to IN1, as shown in Fig.7(b) – last month. The switch detects water level in a tank.
- When the switch closes, we want to receive the message 'TANK OVERFLOW'.
- When the switch opens, we want to receive the message 'LEVEL NORMAL'.

We start in programming mode by installing a jumper on JP3 and powering up. After the 'Comms Error' LED goes out (about 6 seconds), we can send our programming commands from a second mobile phone, as follows:

LOGIN (the green LED illuminates)
ACKON
PASSREDDWARF
OUT1LPUMP ON
OUT1HPUMP OFF
IN1LTANK OVERFLOW
IN1HLEVEL NORMAL

This completes the programming, so JP3 must now be removed, returning the system to operating mode. We can now control the pump by 'SMSing' the following messages to the controller:

PUMP ON (switch the pump on)
PUMP OFF (switch the pump off)

If our imaginary tank overflows and the switch closes, we'll receive the message:

TANK OVERFLOW

When the level subsides and the switch opens, we will receive the message:

LEVEL NORMAL

Now suppose we don't want to receive the 'LEVEL NORMAL' message again. Instead, we only want to be informed when there is an overflow. We can disable the 'LEVEL NORMAL' notification by sending:

DISLEVEL NORMAL

To later reinstate notification, we'd send:

ENLEVEL NORMAL

To change the password to 'STARGATE' and disable acknowledgments, we could send:

PASSSTARGATE
ACKOFF

To log out of the system and prevent further messages being sent by the controller:

LOGOUT (the green LED goes out)

Finally, note that all future logins will require the current password, as follows:

LOGINSTARGATE

Each command is sent as a separate message. After the **ACKON** command, the controller will acknowledge all subsequent commands; you should wait until you receive these before sending the next command. Although you don't need acknowledgments turned on, it's the only way to be certain that the controller has received and processed your commands. This is much more important during normal operation, when you're far from the controller and can't see what's happening.

The password and all user-programmed messages are stored in the micro's EEPROM, so you only need to program the system once. The same goes for the output port state. If a power failure occurs, the last state will be automatically reinstated when power is restored.

You can reprogram the system at any time simply by repeating the steps outlined above. When you redefine a message for any input or output, the old message is automatically overwritten.

If you'd like to start from scratch, then all of the previously programmed messages can be deleted in one operation by erasing the microcontroller's EEPROM. This is achieved by powering off and installing a jumper on JP1. When you power up again, all four red LEDs come on to indicate that erasure is complete.

Note that this operation also wipes the password and all other parameters, including the 'SMS sent' and 'SMS received' counters.

Finally, you can erase just the password by powering off and installing a jumper on JP2. At the next power on, the password will be erased. Be sure to remove JP2 when done, otherwise the **PASS** command will have no effect!

Security

While it's not strictly necessary to program a password during setup and testing, we recommend that you do so before 'going live'. A password is an effective way of preventing someone else taking control of the module without your knowledge.

Once you've successfully logged in, the SMS controller will only accept messages from your mobile phone number. Messages from all other

The 3210 interface is also accessible under the rear cover but unlike the 3310, there's no need to remove the battery. Once in place, the connector and cable extend at right angles from the rear of the phone, which may make mounting awkward in some cases.

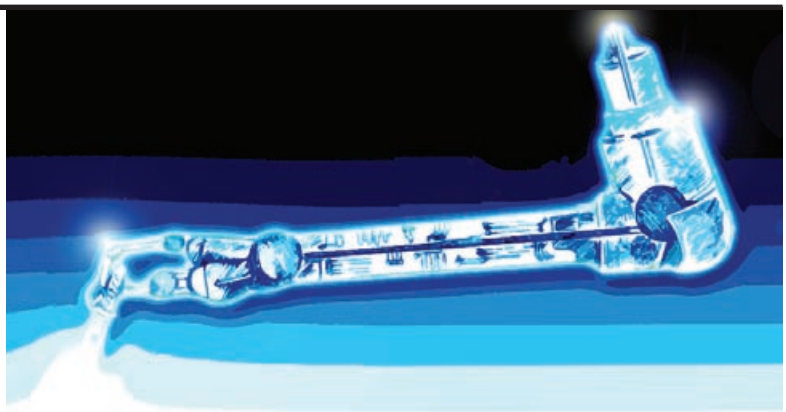
numbers are simply ignored. An exception to this is the **LOGIN** command itself, which can be issued from any mobile number at any time, regardless of whether you're already logged in or not. This allows you to regain control of the system using a second phone should your current phone be lost or stolen. ***EPE***

Credits

Thanks go to the gnokii team, who kindly published details of their work with the Nokia serial bus protocols. You'll find their web site at: **www.gnokii.org**.

Regular Clinic

Circuit Surgery



Ian Bell

Active Current Measurement

RECENTLY there has been some discussion on the *EPE Chat Zone* (via **www.epemag.co.uk**) about the measurement of current by amplifying the voltage drop across a small resistor. CZ User ***Chuckieboy*** started the discussion by posting a question about the lack of stability of the reading obtained from such a circuit. Here are some extracts from his post:

I measure the voltage drop across a 0.1Ω resistor and use an op amp to boost the voltage. The voltage coming out of the op amp seems smooth enough, but my reading is up and down. I feed the op amp with 2 × 1kΩ resistors which go either side of the 0.1Ω resistor. The voltage comes out 0V to 5V with 0V to 400mV across the resistor. I've done some Google searching but got confused as there seem to be so many ways of doing it.

At the time of writing, there is insufficient information on the details of Chuckieboy's circuit and the nature of the instability to provide a specific diagnosis, but we can look at some of the ways of approaching this problem and hopefully reduce the confusion.

Key application

A key application of resistive current sensing is the measurement of the supply current being used by a circuit. This may typically be used for power monitoring consumption and providing smart battery management, or to detect fault or misuse conditions causing excessive current, and hence provide a safe shutdown.

The approach is fundamentally very simple – a sensing resistor is inserted in the supply or ground line between the power supply and the circuit. The sense voltage (V_S) dropped across the sensing resistor (R_S) is used to measure the supply current, I:

$I = V_S/R_S$

Despite this basic simplicity, there are some potential difficulties with the design of the measurement of supply current measurement circuits, and the possibility of detrimental side effects on the system for which the supply current is being measured.

In consideration

A number of factors may need to be considered when designing supply current measurement circuits. These include heating and power dissipation in the sense resistor, the effect of ground voltage offset or supply voltage drop on the measured circuit, accuracy of the current measurement required, voltage output of power supply (high voltage circuits may need different approaches) and the physical nature of the ground/chassis in the system.

The value of R_S is typically a fraction of an ohm, possibly only a few milliohms for systems consuming several amps. V_S typically has a value of tens of millivolts. The sense resistor dissipates power, P:

$P = I^2R_S$

Making R_S smaller reduces dissipation, but also makes measuring the consequently smaller V_S less accurate. On the other hand, smaller dissipation will reduce self-heating in R_S which may help accuracy.

If the value of R_S is temperature dependent (which it usually will be), the value of R_S will be different for high and low supply currents and the reading will not be so accurate unless this is taken into account. R_S does not have to be a resistor; it could be a piece of wire or a PCB track (trace).