

A numbat, a small marsupial with a distinctive black and white striped coat, is sitting on a patch of brown earth. It is positioned in the center-right of the frame, facing towards the right. The ground is covered with dry leaves, twigs, and some fallen fruit. A large tree trunk is visible on the left side of the image.

Using StateCharts on small electronic devices; for fun, profit and numbats

Harry McNally, Decisions and Designs, BarCamp Perth 2009



Harry McNally, Decisions and Designs, 2009

Symbols of Western Australia



[Coat of Arms](#)



[Flag of Western Australia](#)



[Floral Emblem](#)
Red and Green Kangaroo Paw



[Faunal Emblem \(Animal\)](#)
Numbat



[Faunal Emblem \(Bird\)](#)
Black Swan



[Fossil Emblem](#)
Gogo Fish

International Union for Conservation of Nature and Natural Resources (IUCN) Red List of Threatened Species

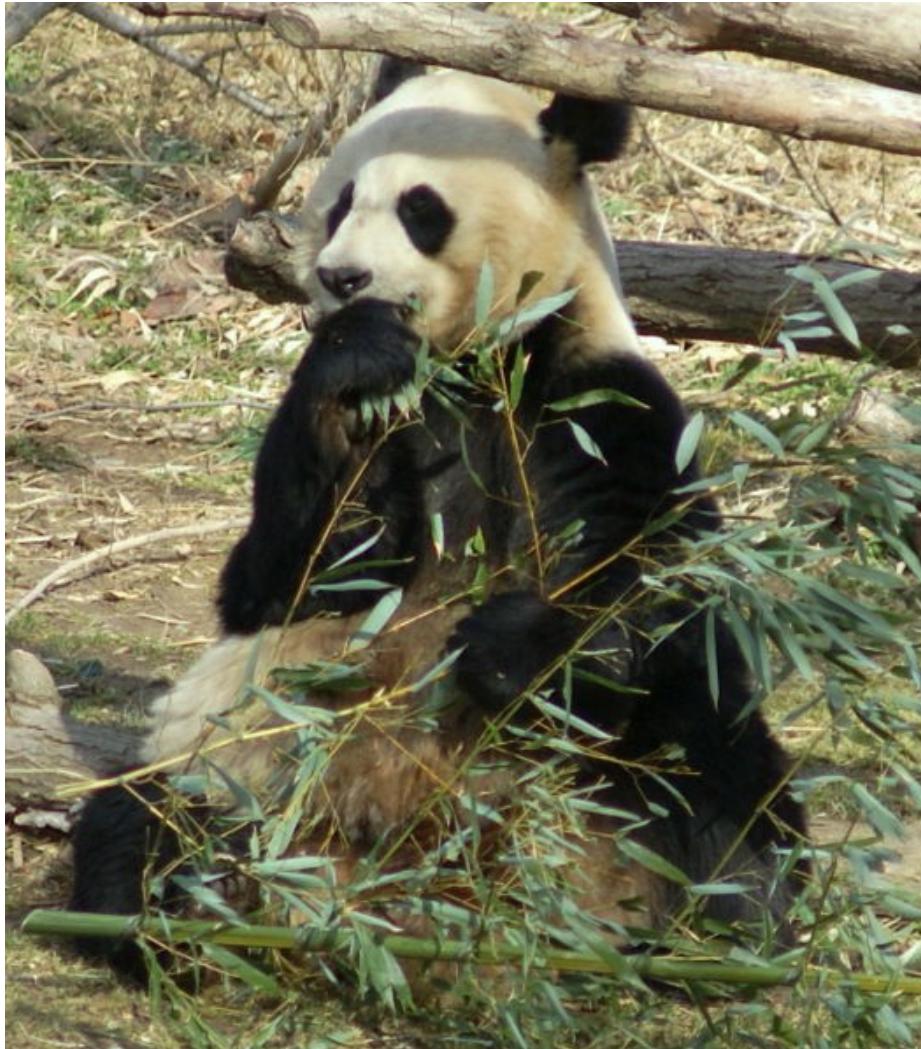


Photo by Aaron Siirila

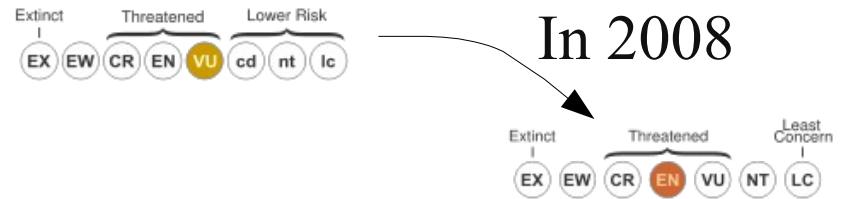


Photo by Gnangarra ... commons.wikimedia.org
Conservation Status Graphics by Peter Halasz



Harry McNally, Decisions and Designs, 2009



<http://www.numbat.org.au/>



Harry McNally, Decisions and Designs, 2009



Harry McNally, Decisions and Designs, 2009



Harry McNally, Decisions and Designs, 2009



Harry McNally, Decisions and Designs, 2009



Storyboard

Panel 1. Setup the camera and to standby

Tony arrives at an observation site at early twilight and positions the video camera at the hollow log. He turns the camera on and checks that it is viewing correctly. He sets the camera to standby.

Storyboard

Panel 2. Adjust the clock time if required

Tony checks the clock time on numbatcam and, if it needs adjustment, he presses and holds the Select button and uses Up and Down buttons to adjust the clock time minute by minute. If he holds the Up or Down button for a second the time increments automatically; slowly at first and quickly after 6 seconds to allow simple hours adjustment. Tony releases the Select button once the clock time is correctly set and the start time displays. If the clock time did not need adjustment, Tony simply presses and releases the Select button to display the start time.

Storyboard

Panel 3. Adjust the start time if required

Tony checks the start time on numbatcam and, if it needs adjustment, he presses the Up and Down buttons to adjust the clock time minute by minute. If he holds the Up or Down button for one second the time changes automatically; slowly at first and quickly after 6 seconds to allow simple hours adjustment. Once the start time is correct Tony presses and releases the Select button to return to the clock time. The starter is now armed and this is shown by a flashing asterisk instead of a flashing cursor.

Storyboard

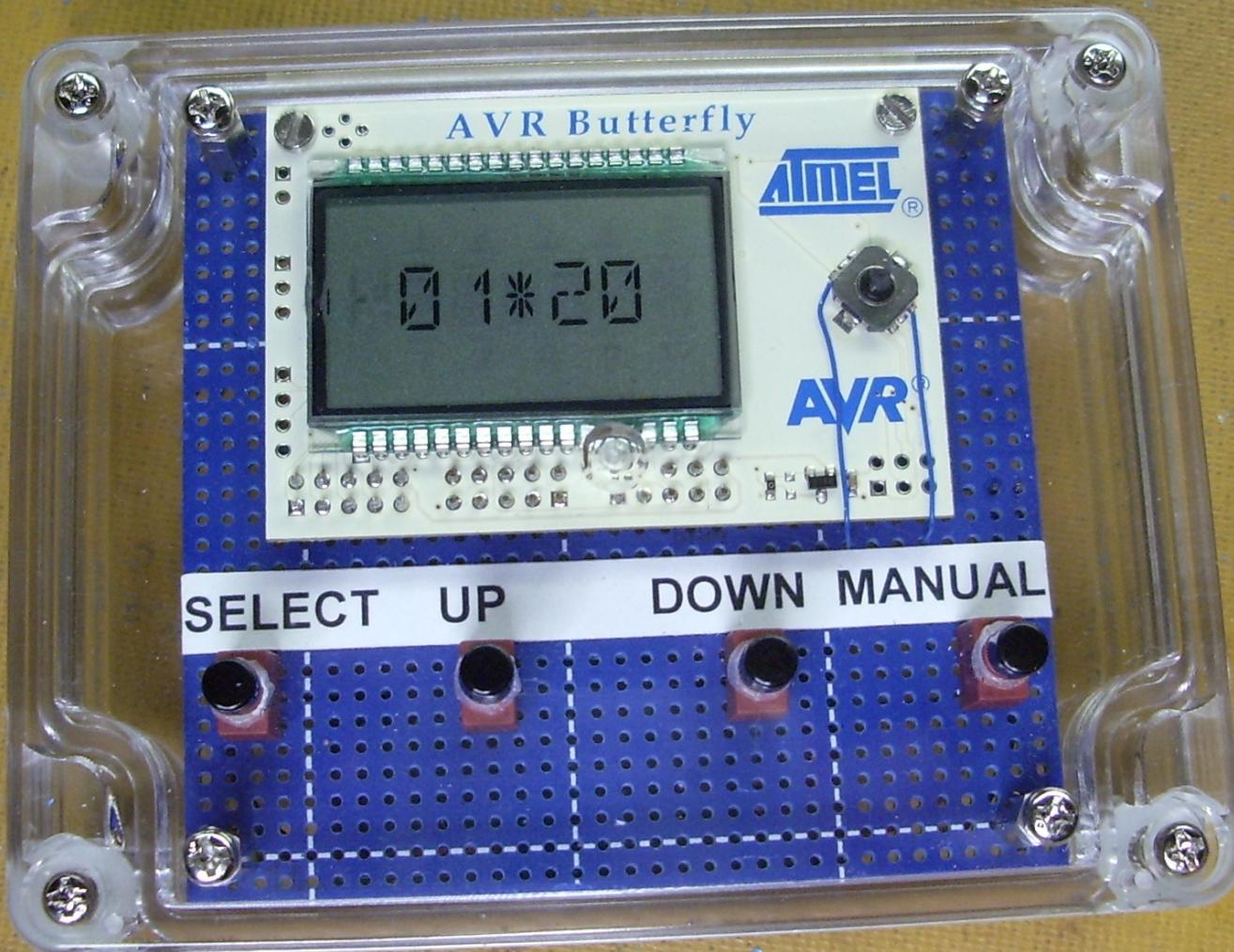
Panel 4. Test the camera start/stop manually

Tony presses the numbatcam Manual button to send a Start/Stop (IR remote) message to the camera. He checks that the camera has started and presses the Manual button again to return the camera to standby.

Storyboard

Panel 5. Numbats!

Tony returns the next morning and stops and retrieves the camera for viewing and analysis.



Existing hardware ?

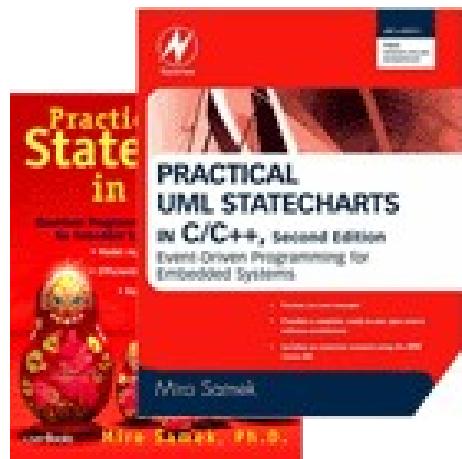


AVR Butterfly

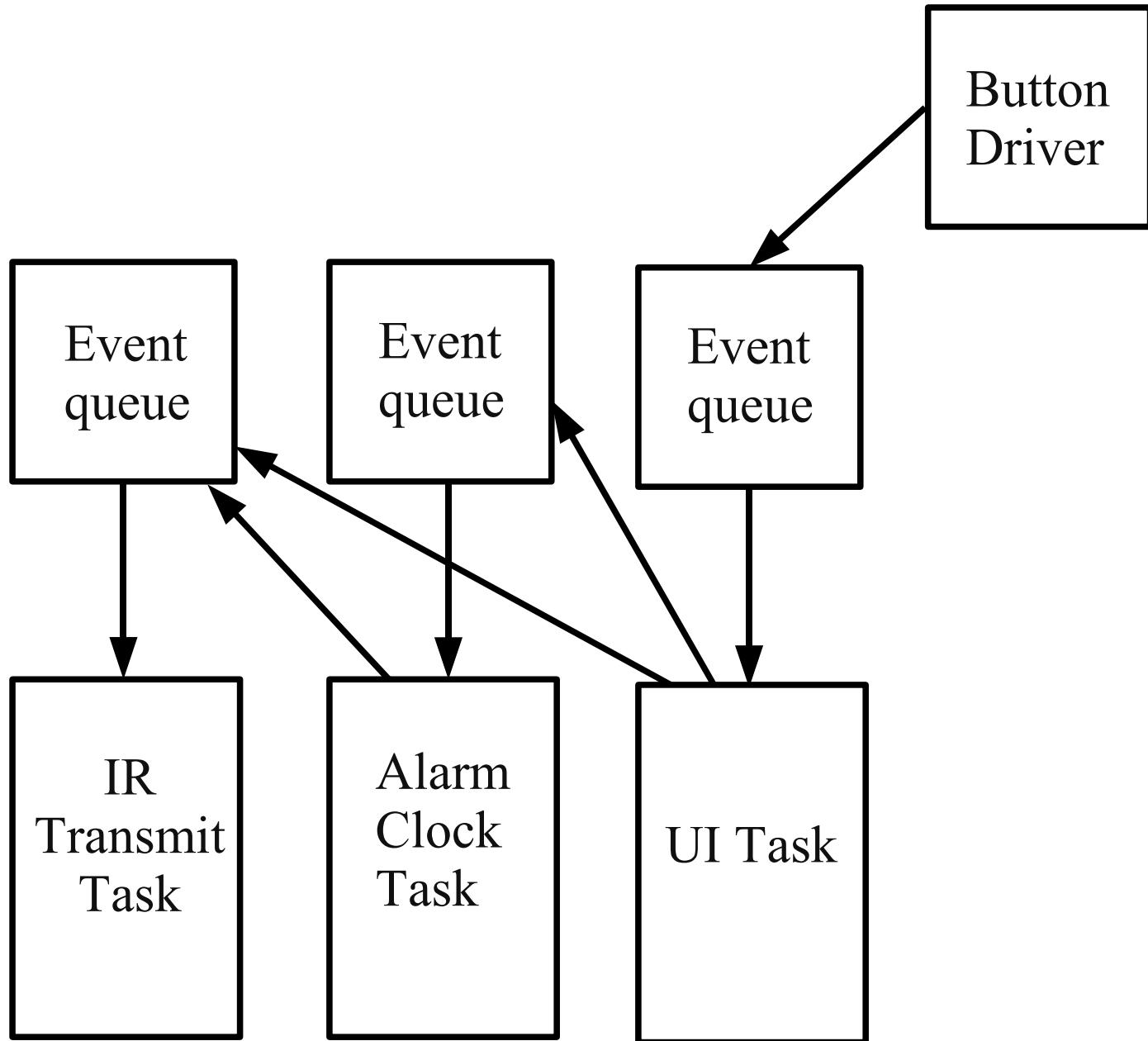
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3146

Small efficient state code ?

<http://www.state-machine.com/>

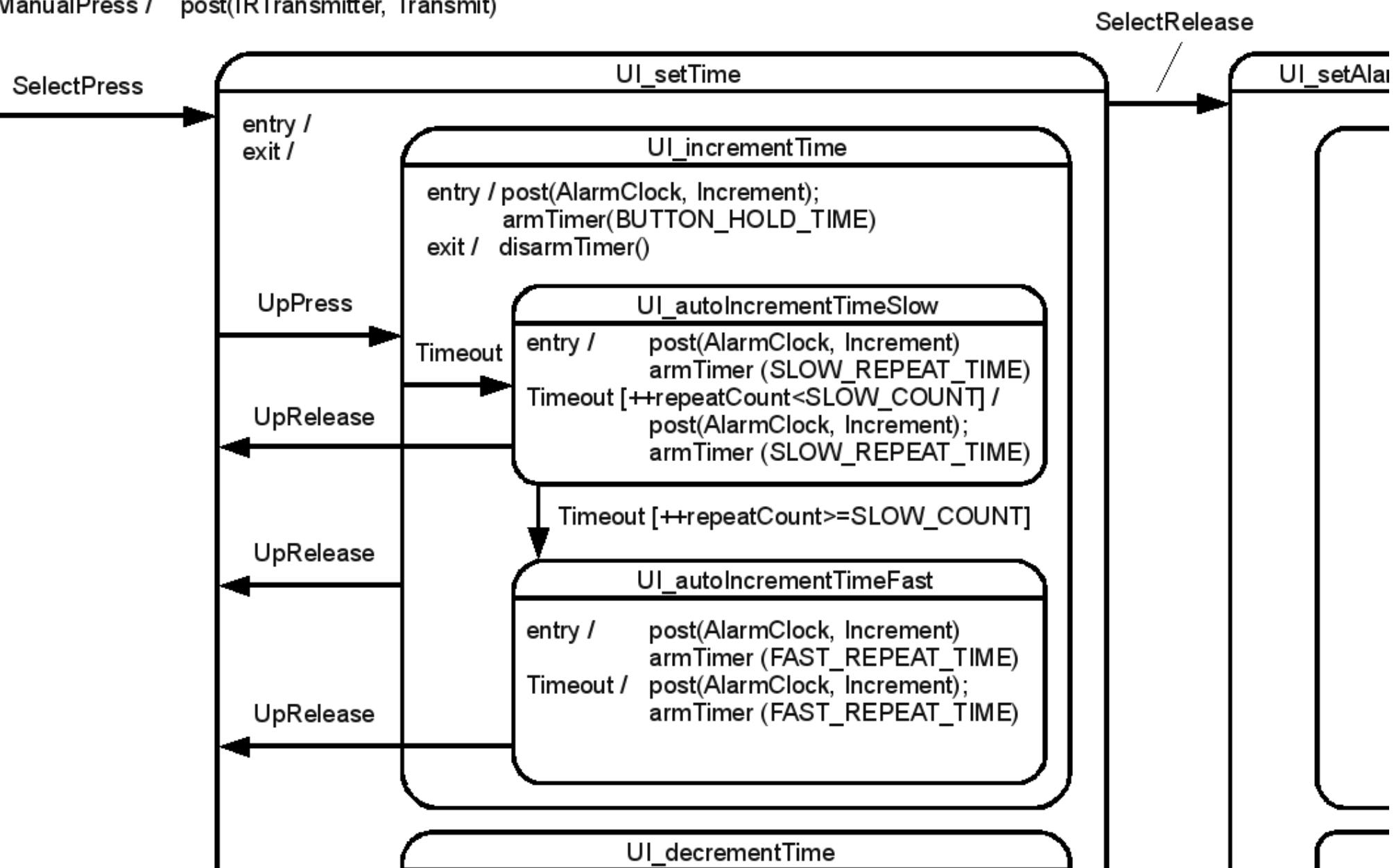


<http://www.state-machine.com/psicc2/index.htm#Errata>
http://www.state-machine.com/avr/QDKn_AVR-GNU.pdf



UI_displayTime

entry / post(AlarmClock, DisplayTime)
exit /
ManualPress / post(IRTransmitter, Transmit)



```
static QState UI_displayTime(UI *me) {  
  
    switch (Q_SIG(me)) {  
  
        case Q_ENTRY_SIG : {  
            QActive_post((QActive *)&AO_alarmClock, DISPLAY_TIME_SIG);  
            return Q_HANDLED();  
        }  
  
        case SELECT_BUTTON_PRESS_SIG : {  
            return Q_TRAN(&UI_setTime);  
        }  
  
        case MANUAL_BUTTON_PRESS_SIG : {  
            QActive_post((QActive *)&AO_IR, IR_TRANSMIT_SIG);  
            return Q_HANDLED();  
        }  
    }  
  
    return Q_SUPER(&QHsm_top);  
}
```

```
static Qstate UI_setTime(UI *me) {  
  
    switch (Q_SIG(me)) {  
        case UP_BUTTON_PRESS_SIG : {  
            return Q_TRAN(&UI_incrementTime);  
        }  
        case DOWN_BUTTON_PRESS_SIG : {  
            return Q_TRAN(&UI_decrementTime);  
        }  
        case SELECT_BUTTON_RELEASE_SIG : {  
            return Q_TRAN(&UI_setAlarm);  
        }  
        case UP_BUTTON_RELEASE_SIG : {  
            return Q_TRAN(&UI_setTime);  
        }  
        case DOWN_BUTTON_RELEASE_SIG : {  
            return Q_TRAN(&UI_setTime);  
        }  
    }  
  
    return Q_SUPER(&UI_displayTime);  
}
```

```
static QState UI_incrementTime(UI *me) {  
  
    switch (Q_SIG(me)) {  
  
        case Q_ENTRY_SIG : {  
            QActive_post((QActive *)&AO_alarmClock, INCREMENT_SIG);  
            QActive_arm((QActive *)me, SET_HOLD);  
            return Q_HANDLED();  
        }  
  
        case Q_EXIT_SIG : {  
            QActive_disarm((QActive *)me);  
            return Q_HANDLED();  
        }  
  
        case Q_TIMEOUT_SIG : {  
            return Q_TRAN(&UI_autoIncrementTimeSlow);  
        }  
    }  
    return Q_SUPER(&UI_setTime);  
}
```

```

static QState UI_autoIncrementTimeSlow(UI *me) {
    switch (Q_SIG(me)) {

        case Q_ENTRY_SIG : {
            QActive_post((QActive *)&AO_alarmClock, INCREMENT_SIG);
            QActive_arm((QActive *)me, SET_SLOW_REPEAT);
            me->repeatCount = 0;
            return Q_HANDLED();
        }

        case Q_TIMEOUT_SIG : {
            QActive_post((QActive *)&AO_alarmClock, INCREMENT_SIG);
            QActive_arm((QActive *)me, SET_SLOW_REPEAT);
            me->repeatCount++;
            if (me->repeatCount >= SLOW_REPEAT_COUNT) {
                return Q_TRAN(&UI_autoIncrementTimeFast);
            }
            return Q_HANDLED();
        }

    }

    return Q_SUPER(&UI_incrementTime);
}

```

```
static QState UI_autoIncrementTimeFast(UI *me) {

    switch (Q_SIG(me)) {
        case Q_ENTRY_SIG : {
            QActive_post((QActive *)&AO_alarmClock, INCREMENT_SIG);
            QActive_arm((QActive *)me, SET_FAST_REPEAT);
            return Q_HANDLED();
        }
        case Q_TIMEOUT_SIG : {
            QActive_post((QActive *)&AO_alarmClock, INCREMENT_SIG);
            QActive_arm((QActive *)me, SET_FAST_REPEAT);
            return Q_HANDLED();
        }
    }

    return Q_SUPER(&UI_incrementTime);
}
```

From the AVR Studio build:

Device: atmega169p

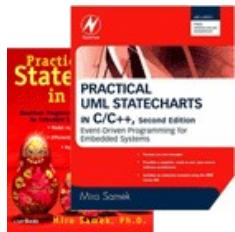
Program: 5192 bytes (31.7% Full)
(.text + .data + .bootloader)

Data: 51 bytes (5.0% Full)
(.data + .bss + .noinit)

Questions ?



<http://www.numbat.org.au/>



<http://www.state-machine.com/>

<http://www.state-machine.com/psicc2/index.htm#Errata>
http://www.state-machine.com/avr/QDKn_AVR-GNU.pdf



AVR Butterfly

http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3146