New ALT for Application Timers and Synchronisation Point Scheduling

(Two excerpts from a small channel based scheduler)

Øyvind TEIG and Per Johan VANNEBO

Autronica Fire and Security*[1], *Trondheim, Norway

`{per.vannebo, oyvind.teig}@autronicafire.no`

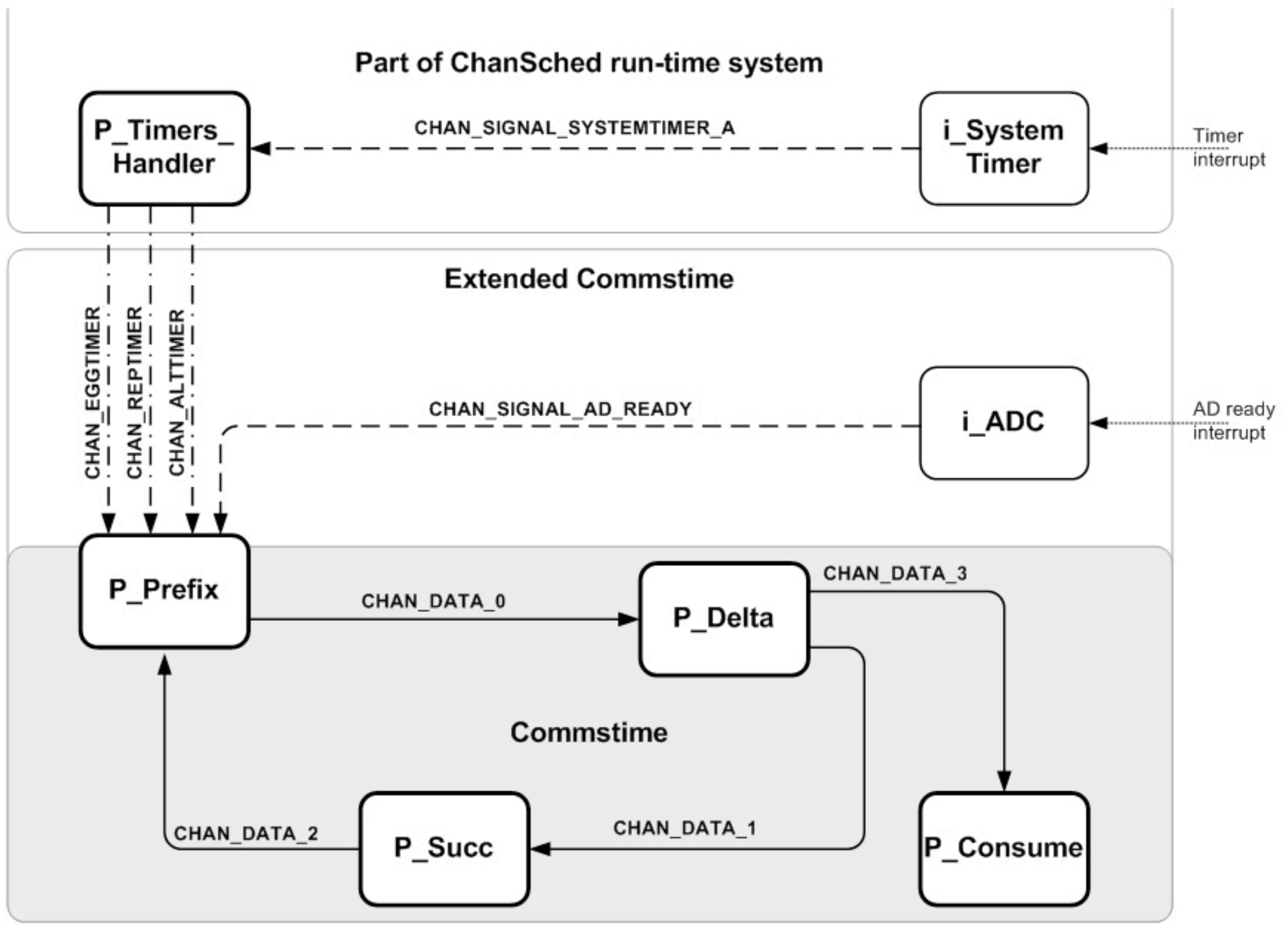
[1] A UTC Fire & Security Company. NO-7483 Trondheim, Norway

http://www.autronicafire.no

At *Communicating Process Architectures 2009* (CPA-2009), 1-4 November, 2009, Eindhoven, the Netherlands

http://www.wotug.org/cpa2009/

ChanSched handling «Extended Commstime»


Part of ChanSched run-time system
P_Timers_ Handler
CHAN_SIGNAL_SYSTEMTIMER_A
i_System Timer
Timer interrupt
Extended Commstime
CHAN_EGGTIMER
CHAN_REPTIMER
CHAN_ALTTIMER
CHAN_SIGNAL_AD_READY
i_ADC
AD ready interrupt
P_Prefix
CHAN_DATA_0
P_Delta
CHAN_DATA_3
Commstime
P_Succ
CHAN_DATA_2
CHAN_DATA_1
P_Consume

Application timers aren’t as built-in as you think

New ALT for Application Timers..

```
Void P_Prefix (void)                           // extended “Prefix”
{
  Prefix_CP_a CP = (Prefix_CP_a)g_CP; // get process Context from Scheduler
  PROCTOR_PREFIX()                             // jump table (see Section 2)
  ...  some initialisation
  SET_EGGTIMER (CHAN_EGGTIMER, CP->LED_Timeout_Tick);
  SET_REPTIMER (CHAN_REPTIMER, ADC_TIME_TICKS);
  CHAN_OUT (CHAN_DATA_0, &CP->Data_0, sizeof(CP->Data_0));  // first output
  while (TRUE)
  {
    ALT();                                     // this is the needed ”PRI_ALT”
      ALT_EGGREPTIMER_IN   (CHAN_EGGTIMER);
      ALT_EGGREPTIMER_IN   (CHAN_REPTIMER);
      gALT_SIGNAL_CHAN_IN (CHAN_SIGNAL_AD_READY);
      ALT_CHAN_IN          (CHAN_DATA_2, &CP->Data_2, sizeof (CP->Data_2));
      ALT_ALTTIMER_IN      (CHAN_ALTTIMER, TIME_TICKS_100_MSECS);
    gALT_END();
    switch (g_ThisChannelId)
    {
      ...  process the guard that has been taken, e.g. CHAN_DATA_2
      CHAN_OUT (CHAN_DATA_0, &CP->Data_0, sizeof (CP->Data_0));
    };
  }
}
```

```
PROC P_Listing2 (VAL INT n, CHAN INT InChan? OutChan!) -- extended "Prefix"
  INT Timeout_ALTTIMER, Timeout_REPTIMER:
  TIMER Clock_ALTTIMER, Clock_REPTIMER:
  SEQ
    OutChan ! n
    Clock_REPTIMER ? Timeout_REPTIMER
    Timeout_REPTIMER := Timeout_REPTIMER PLUS half.an.hour
    WHILE TRUE
      Clock_ALTTIMER ? Timeout_ALTTIMER
      PRI ALT
        Clock_REPTIMER ? AFTER Timeout_REPTIMER
          ...  process every 30 minutes
          Timeout_REPTIMER := Timeout_REPTIMER PLUS half.an.hour
          -- no skew, only jitter
        INT Data:
        InChan ? Data
          ...  process Data
        Clock_ALTTIMER ? AFTER Timeout_ALTTIMER PLUS hundred.ms
          ...  MyChan pause do background task (starvation possible)
          -- skew and jitter
:
```

```
PROC P_Listing3 (VAL INT n, CHAN INT InChan? OutChan!)  -- extended “Prefix”
  TIMER My_ALTTIMER, My_REPTIMER:   -- only timers, no variables
  SEQ
    OutChan ! n
    SET_TIMER (REPTIMER, My_REPTIMER, 30, MINUTE,   24H)
    SET_TIMER (ALTTIMER, My_ALTTIMER,  0, MILLISEC, 32BIT)
    WHILE TRUE
      PRI ALT
        My_REPTIMER ? AFTER ()
          ...  process every 30 minutes (no timeout value to compute)
          -- no skew, only jitter
        INT Data:
        InChan ? Data
          ...  process Data
        My_ALTTIMER ? AFTER (100)
          ...  MyChan pause do background task (starvation possible)
          -- skew and jitter
:
```

A scheduler isn’t as invisible as it looks

.. and Synchronisation Point Scheduling

```
void P_Standard (void)
{
  CP_a CP = (CP_a)g_ThisExtPtr; // Application
  switch (CP->State)            // and
                                // communication
                                // state
  {
    case ST_INIT: {/*Init*/ break;}
    case ST_IN:
    {
      CHAN_IN(G_CHAN_IN,CP->Chan_val1);
      CP->State = ST_APPL1;
      break;
    }
    case ST_APPL1:
    {
      // Process val1
      CP->State = ST_OUT;
      break;
    }
    case ST_OUT:
    {
      CHAN_OUT(G_CHAN_OUT,CP->Chan_val1);
      CP->State = ST_IN;
      break;
    }

  }
}
```

PROCTOR_PREFIX

```
void P_Extended (void)
{
  CP_a CP = (CP_a)g_ThisExtPtr; // Application
  // Init here                  // state only
  while (TRUE)
  {
    switch (CP->State)
    {
      case ST_MAIN:
      {
        CHAN_IN(G_CHAN_IN,CP->Chan_val2);

        // Process val2

        CHAN_OUT(G_CHAN_OUT,CP->Chan_val2);
        CP->State = ST_MAIN; // option1
        break;
      }
    }
  }
}
```

```
void P_libcsp2 (Channel *in, Channel *out)
{
  int val3;
  for(;;)
  {
    ChanInInt (in, &val3);
    // Process val3
    ChanOutInt (out, val3);
  }
}
```

```
PROC P_occam (CHAN OF INT in, out)

  WHILE TRUE
  INT val4:
    SEQ
      in ? val4
      -- Process val4
      out ! val4

:
```

```
#define SCHEDULE_AT goto

#define CAT(a,b,c,d,e) a##b##c##d##e // Concatenate to f.ex. “SYNCH_8_L”

#define SYNCH_LABEL(a,b,c,d,e) CAT(a,b,c,d,e) // Label for Proctor-table

#define PROC_DESCHEDULE_AND_LABEL() \
        CP->LineNo = __LINE__; \
        return; \
        SYNCH_LABEL(SYNCH,_,__LINE__,_,L):

#define CHAN_OUT(chan,dataptr,len) \
        if (ChanSched_ChanOut(chan,dataptr,len) == FALSE) \
        { \
            PROC_DESCHEDULE_AND_LABEL(); \
        } \
        g_ThisAltTaken = FALSE
```

```
#define PROCTOR_PREFIX()\
        switch (CP->LineNo)\
        {\
            case 0: break;\
            case 8: SCHEDULE_AT SYNCH_8_L;\
            case 17: SCHEDULE_AT SYNCH_17_L;\
            case 21: SCHEDULE_AT SYNCH_21_L;\
            DEFAULT_EXIT\
        }
```

```
In P_Commstime.c there were 4 processes, and 10 synchronisation points
In P_Timers_Handler.c there was 1 process, and 1 synchronisation point
There were a total of 2 files, 5 processes and 11 syncronisation points
```

..application timers & scheduling..

..questions?

http://www.teigfam.net/oyvind/pub/CPA2009/paper.pdf
http://www.teigfam.net/oyvind/pub/CPA2009/presentation.pdf