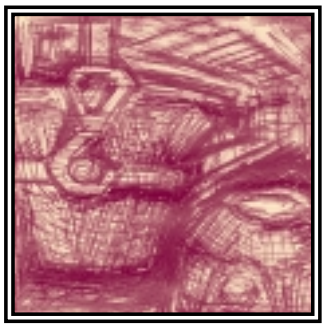


Автоматизация информационного сопровождения прокатного стана

Юрий Волобуев

Описан опыт проектирования автоматизированной системы информационного сопровождения технологических процессов прокатного стана ММЗ.

Введение

Автоматизированная система информационного сопровождения стана (в прижившейся аббревиатуре — АИС) была внедрена на мелкосортном прокатном стане Молдавского металлургического завода более десяти лет назад, практически с момента его пуска. Первоначально она была реализована на КТС ЛИУС с использованием обычных бытовых телевизоров и самодельных клавиатур в качестве операторских станций, затем развивалась и претерпевала изменения вследствие поэтапных реконструкций стана или переоснащения вычислительных средств. И до такой степени она «доразвивалась», что существующую версию системы роднит со своими корнями лишь наименование. И по сути, и по технической реализации ее можно назвать самостоятельным продуктом, созданным «с нуля», и отроду ей чуть более двух лет.


Прокатный стан

Назначение АИС

На рис. 1 представлена схема технологической линии сортопрокатного стана. Собственно стан состоит из последовательности пар валков (клетей), которые обжимают проходящую между ними заготовку до нужного профиля. У нас внедрена двухниточная прокатка (впервые в мировой практике, кстати), при которой через клети стана проходят одновременно две заготовки. До определенной клети они идут бок о бок (по так называемым левой и правой ниткам), а затем разветвляются: левая идет прямиком в зону отделки СОРТА, где разрезается на прутки, которые связываются в пакеты и взвешиваются, а правая — на БЛОК, после которого получается готовый прокат в виде бунтов катанки — (очень толстой «проволоки», да простят меня прокатчики). Применяются режимы проката, при которых заготовки с обеих ниток (но уже не одновременно) поступают либо на сорт, либо на блок. Впрочем, всех нюансов не перечислишь.

Прежде чем попасть в стан, заготовки нагреваются до нужной температуры в одной из двух нагревательных печей. Печи имеют магазинную организацию и вмещают порядка 100 заготовок каждая. Каждая заготовка перед загрузкой в печь взвешивается, затем, перемещаясь в печи, нагревается, доходит до окна выдачи,

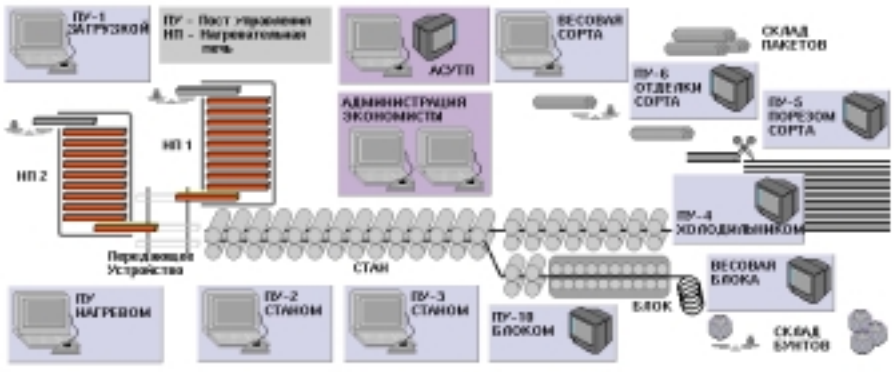


Рис. 1. Схема технологической линии сортопрокатного стана

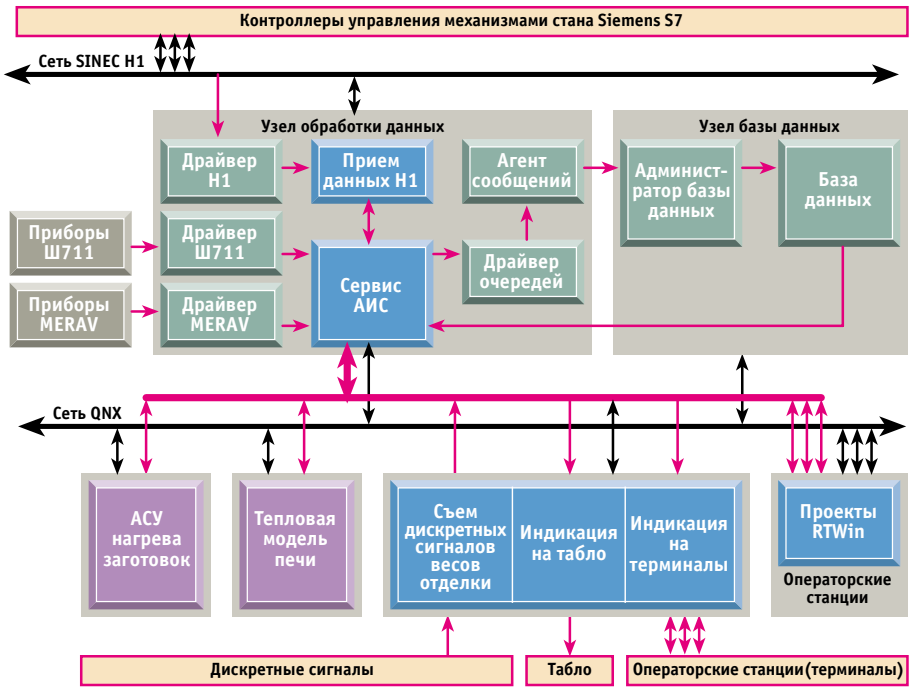


Рис. 2. Компоненты и взаимосвязи системы АИС

а выйдя, перемещается по так называемому передающему устройству и направляется в стан на ту или иную нитку.

Вот это все, начиная с взвешивания заготовки перед загрузкой в нагревательную печь и кончая взвешиванием готового пакета (бунта), и входит в сферу АИС.

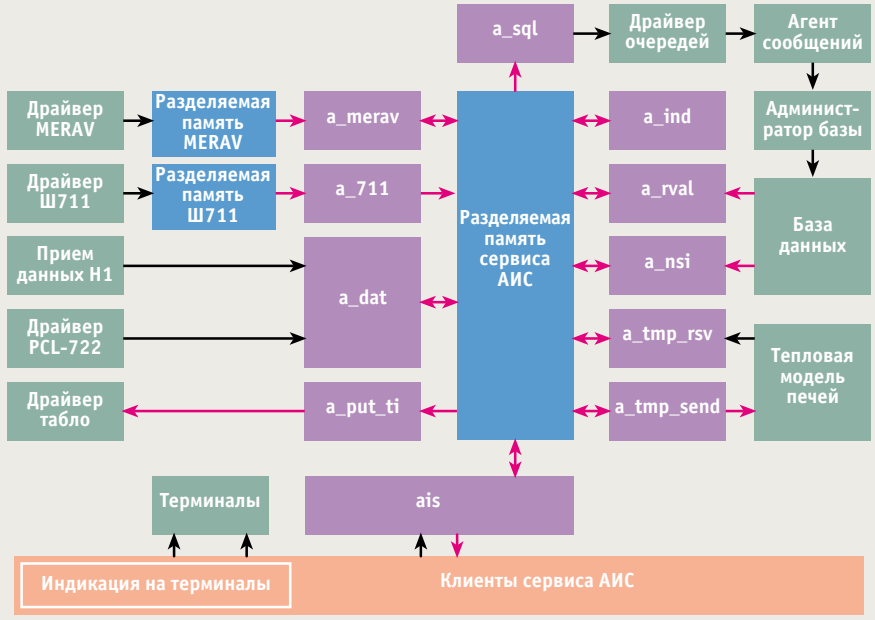
АИС выполняет три основные функции.

1. Оперативно отображает на десятке рабочих мест текущее состояние технологического процесса. Это не только создает определенный комфорт для операторов стана, но и помогает избежать самого страшного брака в работе — смешивания плавок (партий заготовок, выплавленных в разное время, имеющих разные свойства металла). На практике во избежание этого делают технологические паузы между плавками. Устойчивая работа АИС позволяет существенно снижать протяженность этих пауз.
2. Передает оперативные данные автономным системам АСУ нагрева заготовок и Тепловой модели нагревательных печей.
3. Осуществляет автоматический сбор и хранение различных учетных параметров, начиная с износа оборудования и кончая данными о производстве, а также обеспечивает их просмотр и статистическую обработку.

Аппаратные средства

АИС отслеживает каждую заготовку «поименно». И если «фамилию» (реквизиты плавки) задает оператор на загрузке, то «имя» (порядковый номер) и вся остальная история заготовки строится на данных контроллеров управления механизмами и измерительных приборов. А поскольку процессы на стане довольно быстрые и, как следствие, поток сигналов интенсивен, главной заботой при разработке АИС была и остается не столько шлифовка алгоритмов обработки данных, сколько повышение устойчивости системы, исключение потерь сигналов и улучшение иных «тактико-технических характеристик», а в этом деле мелочей нет.

В настоящее время АИС реализована в операционной системе QNX (что уже само по себе закрывает множество проблем). АИС имеет выход на технологическую сеть Sinec-H1, откуда поступают данные с контроллеров управления механизмами стана Siemens серии S7-400. Другими источниками первичных данных являются приборы Ш711 (интеллектуальный многоканальный АЦП) и MERAV (съём показания весов), связанные через ИРПС (токовая петля) с интеллектуальным мультипортом Intelliconn-Flex8, а также некоторое количество дискретных сигналов, принимаемых через плату PCLD-782B. В качестве базы данных долговременного хранения информации выбрана Sybase SQL Anywhere. Все это хозяйство крутится на обычных либо промышленных компьютерах Pentium. Операторские станции подразделяются на два типа: те же



НАЗНАЧЕНИЕ МОДУЛЕЙ СЕРВИСА АИС	
ais	Головной модуль
a_rval	Считывание данных из Базы валков
a_sql	Запись данных во входную очередь агента сообщений для Базы
a_tmp_rcv	Сервер-получатель для Тепловой модели печи
a_tmp_send	Сервер-отправитель для Тепловой модели печи
a_ind	Формирование реквизитов отображения
a_dat	Обработка сигналов контроллеров H1 и PCL-722
a_nsi	Считывание НСИ из Базы
a_put_ti	Передача данных для табло
a_merav	Обработка показаний приборов MERAV
a_711	Обработка показаний приборов Ш711

Рис. 3. Компоненты Сервиса АИС

Пример 1. Описание данных для обеспечения передачи Сервис-клиент

```
/* Номера и флаги каналов */
#define CHAN_TIME 0
#define F_CHAN_TIME (0x00000001L<<CHAN_TIME)
#define CHAN_DAT  1
#define F_CHAN_DAT  (0x00000001L<<CHAN_DAT)
/* Структура общего блока памяти */
struct COMMON_AIS {
 /* Матрица обмена
 CHANS - Количество каналов
 CLIENTS - Максимальное количество клиентов */
 char matrix[CHANS][CLIENTS];
 /* Данные каналов */
 struct chan_TIME {
  char time[10]; /* текущее время */
  char date[20]; /* текущая дата */
  char num_br; /* номер бригады */
 } chan_time;
 struct chan_DAT { /* канал сигналов и датчиков */
  char status[100];
 } chan_dat;
};
/* Структура буфера обмена Сервис- лиент */
struct bsend {
 struct d_head { /* заголовок */
  short type;
  short subtype;
  short client;
  long  maska;
  long  flags;
 } head;
 char   data[sizeof(struct chan_TIME)+
             sizeof(struct chan_DAT)];
};
```

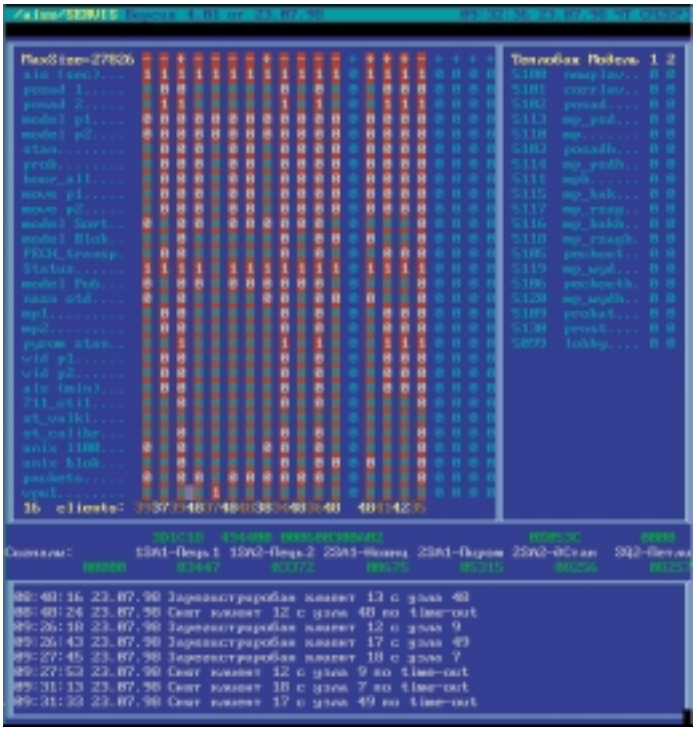


Рис. 4. Информационный экран Сервиса АИС

Pentium с 21-дюймовыми мониторами промышленного исполнения, предназначенные для работы в графическом режиме, и рабочие станции AWS-825 фирмы Advantech, работающие в режиме терминала. Для визуализации используется графический интерфейс Photon microGUI, а также инструментальная среда для автоматизированного проектирования систем контроля и управления RTWin.

На рис. 2 отображены основные компоненты и взаимосвязи системы АИС. Конечно же, схема не отражает четких границ АИС, да их и не существует. Например, считать ли драйвер съема показаний прибора MERAV детищем АИС только на том основании, что никто более не пользуется его услугами? Кроме того, изображенное распределение по узлам сети не является чем-то обязательным и на деле неоднократно варьировалось в основном по техническим соображениям.

Итак, отталкиваясь от рис. 2, хочу заострить внимание на блоке «Сервис АИС» и потоках информации, замыкающихся на нем.

Программное обеспечение АИС

Сервис АИС представляет собой комплекс программ (процессов), запускаемых на одном узле и взаимодействующих через общую разделяемую память (рис. 3).

Само название «Сервис» определяется характером функций данного комплекса. В самом деле, получая и обрабатывая первичные данные, он создает и содержит в порядке некий базисный набор данных, передает их на хранение в базу данных, а также предоставляет информацию своим клиентам по первому требованию, оказывая тем самым информационные услуги.

В общей памяти, помимо всевозможных рабочих массивов взаимодействия и конфигурации, содержится также как бы информационный снимок стана, состоящий из набора блоков (каналов) данных (пример 1). Каждый модуль Сервиса, выполняя свои узкие функции, обновляет соответствующие данные этих каналов или использует их другим образом (рис. 3).

Запускает, выгружает, перезапускает и контролирует модули головная программа, выполняющая также передачу информационных каналов всем желающим клиентам. Диагностика и протоколирование всех этих мероприятий отображается на информационном экране Сервиса АИС (рис. 4).

Первичные данные попадают в Сервис двумя путями: либо принимаются от драйверов по их инициативе, либо считываются из разделяемой памяти драйвера по логике прикладного алгоритма.

Передачу же базисной информации из Сервиса по различиям в протоколах можно разделить на четыре группы.

1. Передача базисного данного по факту его возникновения (событию) в базу данных.
2. Передача по событию определенному адресату (Тепловой модели нагревательной печи, например).
3. Ответ на запрос определенного данного или управляющего воздействия.
4. Передача зарегистрированному клиенту группы каналов.

Существует, правда, и пятый путь (пятая колонна?) получения данных из Сервиса, совершенно беспротокольный, — считывание непосредственно из разделяемой памяти. Я лично применяю этот путь в различных исследовательских целях. Опаснее здесь другое — ведь так можно и записать. Данная возможность используется при аварийном восстановлении информации, например. И от этого, по сути, несанкционированного доступа защиты нет. Впрочем, среди привилегированных пользователей, могущих сотворить такое, ни «чайников», ни хулиганов тоже нет.

Передача в Базу данных осуществляется по цепочке, представленной на рис. 5.

Этим и ограничусь, поскольку детальное описание заслуживает отдельной статьи. Кроме того, такой метод передачи в базу применяется у нас в АСУ ТП

уже несколько лет и давно перешел в разряд системных вещей.

Второй и третий способы передачи — обычные Send и Reply, весь протокол которых сводится лишь к формату данных (Send и Reply — функции WATCOM C, обеспечивающие синхронную передачу данных между задачами в ОС QNX).

А вот передача каналов клиентам применена нами впервые именно в Сервисе АИС, посему о ней несколько подробней.

Передача Сервис-клиент

Каналы информации пронумерованы, и каждому соответствует некий битовый флаг. В общей памяти размещена матрица обмена (пример 1), строки которой соответствуют каналам информации, а столбцы — клиентам. Модуль, обновляющий данные того или иного канала, расписывает единицей соответствующую строку матрицы (пример 2).

Первым делом клиент делает запрос на регистрацию. Сервис выделяет свободный столбец матрицы и возвращает его номер клиенту в качестве подтверждения регистрации. При запросе данных клиент присылает присвоенный ему номер и две битовые маски, соответствующие каналам. В нашем случае маска имеет размерность long, то есть рассчитана на 32 канала (по числу битов). Первая маска содержит флаги тех каналов, которые вообще интересуют клиента, а вторая — флаги тех каналов, которые необходимо передать, независимо от того, обновлялись ли они со времени последнего сеанса или нет (при самом первом запросе, например, обе эти маски совпадают). Сервис при обработке этого запроса (пример 3), ориентируясь на присланные маски и соответствующий клиенту столбец матрицы обмена, собирает в пакет и отсылает клиенту только те из интересующих его каналов, которые либо запрошены, так сказать, принудительно, либо обновлены с момента последнего сеанса, после чего столбец матрицы обнуляется. Сервис отключает клиента либо по его запросу, либо по тайм-ауту. В примере 4 приведен фрагмент программы клиента, обеспечивающий запрос и получение данных по описанной схеме.

Клиенты Сервиса

Самые главные клиенты Сервиса — операторские станции. Конечно же, они могут выдавать в Сервис и управляющие воздействия, но львиную долю времени все же заняты простым отображением информации. В базовом варианте на операторской станции этим занимается проект RTWin (рис. 6).

Рис. 5. Путь передачи сообщения в базу данных

Пример 2. Модуль Сервиса, изменяющий данные информационного канала

```
main()
{
struct COMMON_AIS *shmem; //Указатель на общий сегмент памяти
short num;                  //Номер датчика
short stat;               //Статус датчика
 ...
 shmem->chan_dat.status[num]=stat;
 memset(&shmem->matrix[CHAN_DAT][0],1,CLIENTS);
 ...
}
```

Пример 3. Передача группы каналов по запросу клиента

```
main()
{
struct COMMON_AIS *shmem;
pid_t pid; /* идентификатор процесса сервиса */
int   off,size,i;
long  flag;
struct bsend B;

 while( 1 )
 {
  pid=Receive(0,&B,sizeof(struct bsend));
  switch(B.head.type)
  {
   case I_AM_CLIENT: /* Запрос Клиента */
    size=sizeof(struct d_head);
    switch(B.head.subtype)
    {
     case CLIENT_DATA: /* Запрос данных каналов */
      off=0;
      /* Установить флаги "поспевших" каналов */
      flag=0L;
      for(i=0; i<CHANS; i++) /* см. пример 1. */
      {
        if(&shmem->matrix[i][B.head.client]) flag|=(0x00000001<<i);
      }
      /* Оставить только интересующие клиента */
      flag&=B.head.maska;
      /* и приформировать безусловно запрашиваемые */
      B.head.flags|=flag; //Флаги в выходной буфер клиенту
      /* Заполнение выходного буфера данными каналов */
      if(flag&F_CHAN_TIME)
      {
        memmove(&B.data[off],&shmem->chan_time,sizeof(struct chan_TIME));
        off+=sizeof(struct chan_TIME);
        shmem->matrix[CHAN_TIME][B.head.client]=0;
      }
      if(flag&F_CHAN_DAT)
      {
        memmove(&B.data[off],&shmem->chan_dat,sizeof(struct chan_DAT));
        off+=sizeof(struct chan_DAT);
        shmem->matrix[CHAN_DAT][B.head.client]=0;
      }
      size=sizeof(struct d_head)+off;
      Reply(pid,&B,size); /* Отправить Клиенту */
      break;
      ...
     default: break;
      Reply(pid,NULL,0); /* Ответ на неопределенный запрос клиента */
    }
    ...
   default:
    Reply(pid,NULL,0); /* Ответ на неопределенный запрос */
    break;
  }
 }
}
```

Любая система в процессе эксплуатации подвергается критике со стороны пользователя в трех направлениях: либо указываются ошибки, либо высказываются пожелания о расширении возможностей системы (выдаче каких-то дополнительных данных), либо предъявляются претензии к форме отображения. Ошибки выявляются достаточно скоро, базисные данные имеют конечный объем, и наступает время, когда расширение возможностей на деле означает лишь варьирование представлениям и уже имеющихся данных (про форму отображения и говорить нечего). То есть не надо переделывать Сервис, ловить время для его перезапуска (а это актуально в условиях режима реального времени), а достаточно в RTWin добавить в проект съем данных еще одного канала, перерисовать картинку или создать дополнительную панель управления (рис. 7) и провести пару-другую линий связи. Для более сложной обработки данных можно включить в проект очередной С-код, а то и создать дополнительный проект.

Конечно же, нарисованная мною картина больше смахивает на идиллию, нежели на суровую действительность, и все же, все же... Приведу такой факт. Если на начальном этапе номера версий Сервиса АИС и проекта RTWin АИС совпадали, то сейчас идет явное опережение последнего. Впрочем, все сказанное было бы справедливо, применяй я и какой-либо другой инструмент графической визуализации, а то и вовсе обычный вывод (рис. 8). Главное здесь все же отделение процесса получения базисных данных от процесса их отображения. Важнейшее преимущество такого отделения — информация, используемая всеми клиентами, неважно каким способом и на чем отображаемая, идентична. Как же может быть иначе, если источник один. К слову, в АИС даже текущее время поступает к клиентам как данные соответствующего канала.

Вместо заключения

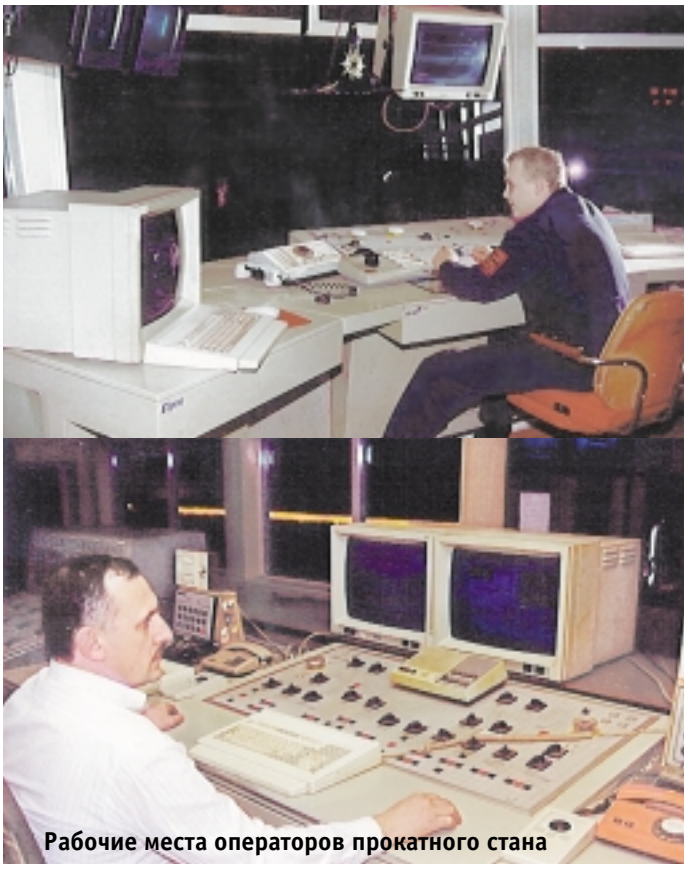
В данной статье приведен пример автоматизации в связке, если можно так выразиться, QNX — графика. Что касается использования QNX, то это было одним из первых опытов (наряду с АСУ нагрева заготовок) применительно к объектам сортопрокатного цеха. Здесь уместно напомнить о существовании другого основного цеха, сталеплавильного, где информационные и управляющие системы разрабатываются под QNX с 1991 года. Если присовокупить системы, обслуживающие некоторые объекты вспомогательных цехов, то мы имеем полное право говорить об общезаводской технологической информационной сети на базе QNX. В настоящее время задействовано свыше 60 узлов, на стадии разработки подключение еще нескольких вспомогательных цехов (около 20 узлов). Не задаваясь целью пропагандировать QNX в данной статье, описывая его достоинства, отмечу лишь чрезвычайно высокую надежность данной ОС в реализации задач АСУ ТП, проверенную опытом и временем.

Несомненным же недостатком QNX является, как некоторые шутят, его «военно-полевой интерфейс». Конечно же, само время заставляет как можно скорее прикрыть эту брешь, и приведенный в статье пример АСУ как раз был ориентирован прежде всего на решение этой задачи (съем, обработка, хранение данных изначально не вызывали никаких сомнений).

Пример 4. Запрос группы каналов у Сервиса

```
                                    /* функции обработки данных
каналов */
void sub_chan_0        (struct chan_TIME *);
void sub_chan_1        (struct chan_DAT *);

main()
{
pid_t pid_servis;
int   off;
struct bsend B;
short  client;
 ...
 /* к этому моменту связь с Сервисом создана
    и Клиент зарегистрирован, номер в client */
 B.head.type   = I_AM_CLIENT;
 B.head.subtype        = CLIENT_DATA;
 B.head.client         = client;
 B.head.maska  = F_CHAN_TIME | F_CHAN_DAT;
 B.head.flags  = 0;
 Send(pid_servis,&B,&B,sizeof(struct d_head),sizeof(struct bsend));
 off=0;
 if(B.head.flags&F_CHAN_TIME)
 {
  sub_chan_0((struct chan_TIME *)&B.data[off]);
  off+=sizeof(struct chan_TIME);
 }
 if(B.head.flags&F_CHAN_DAT)
 {
  sub_chan_1((struct chan_DAT *)&B.data[off]);
  off+=sizeof(struct chan_DAT);
 }
 ...
}
```



Рабочие места операторов прокатного стана

Разработчики и АИС, и упоминаемой АСУ НЗ решали проблемы графики с помощью CACSD RTWin, позднее ряд систем был разработан (или переведен в графическую среду) вручную с использованием средства разработки графических приложений Photon Application Builder. В результате головная боль по поводу графики если не снялась, то значительно уменьшилась. В наших планах наметилась линия создания новых систем только в графической среде, а между делом выполнялся перевод существующих систем на графический интерфейс с пользователем.

В заключение хочу бросить прощальный взгляд на статью. Чем же я хотел осчастливить человечество, описывая построение АИС? Никаких особенных, ярких решений не наблюдается. Но цель все же есть – пропаганда такого построения системы, при котором обработка данных отделена от визуализации результатов обработки. Задам риторический вопрос: можно ли создать достаточно мощную графическую среду (не привязываюсь к конкретной ОС), полностью свободную от ошибок, имея в виду постоянное обновление аппаратных средств. Можно ли, в связи с этим, полностью избежать постоянной угрозы «зависания» такой среды в самый неподходящий момент и с неизбежной потерей данных, а еще хуже – со сбоем управления? Я считаю, что реальнее приспособиться к такой угрозе, нежели уповать на появление «непотопляемого» графического продукта, а именно – закрутить собственно АСУ на апробированном средстве (по возможности даже на отдельном компьютере), а в качестве отображения использовать сколь угодно новые и красивые графические средства, не боясь перезапускать их в любое время. Можно ориентировать появляющиеся многочисленные SCADA на создание систем с таким разделением. Кстати, именно QNX, с ее распределенными ресурсами, позволяет без труда реализовать такой подход. Те же продукты RTWin, по сути, являются комплексами задач, которые рождаются вместе при запуске, живут независимой жизнью, но умирают также вместе при перезапуске. Научи их умирать независимо – разделение готово. Труднее всего поставить точку. Благодарю всех, дошедших до этого места. ●

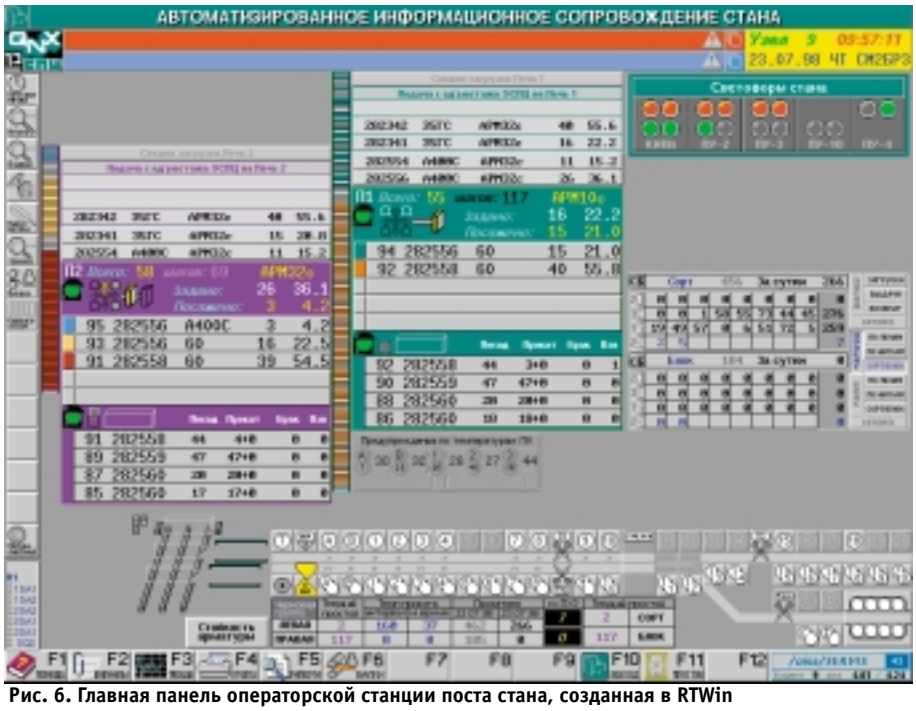


Рис. 6. Главная панель операторской станции поста стана, созданная в RTWin

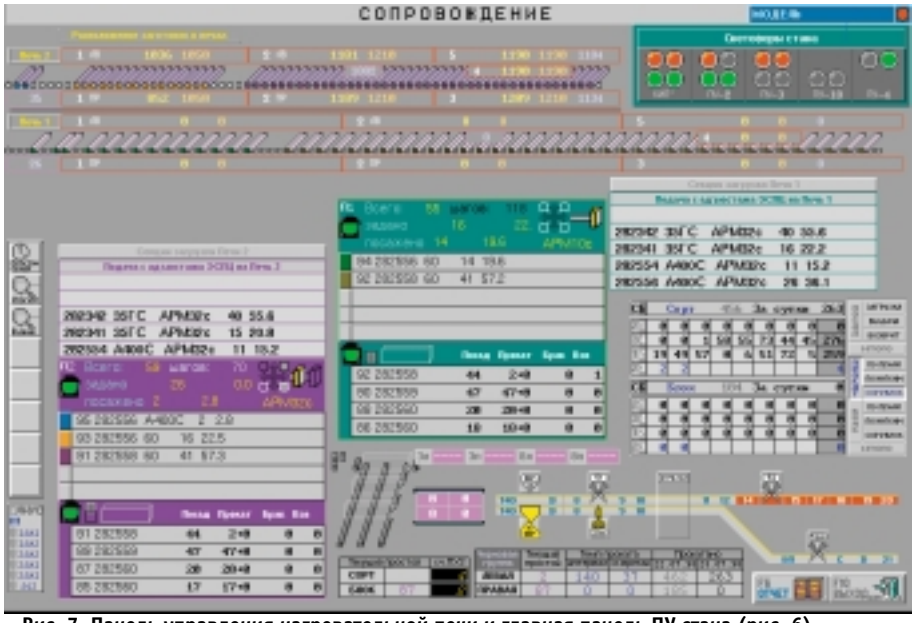


Рис. 7. Панель управления нагревательной печи и главная панель ПУ стана (рис. 6) отображают данные одних и тех же каналов

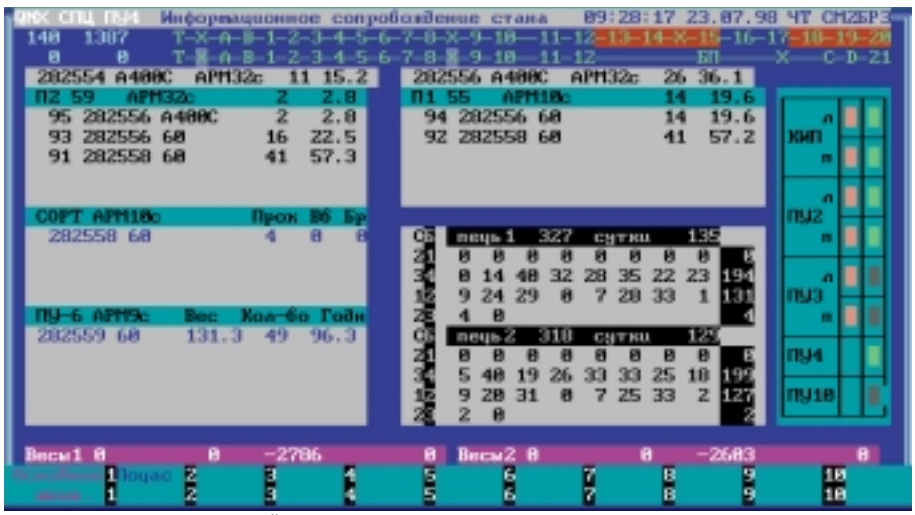


Рис. 8. Панель операторской станции – терминала ПУ-4