Computational hardware

- Digital logic (CSE370)
  - gates and flip-flops: glue logic, simple FSMs, registers
- Programmable logic devices (CSE370, CSE467)
  - two-level PLDs: FSMs, muxes, decoders
  - field-programmable gate arrays: FSMs, basic data-paths
- Microprocessors (CSE378)
  - general-purpose computer
  - instructions can implement complex control structures
  - supports computations/manipulations of data in memory

Microprocessors

- Arbitrary computations
  - arbitrary control structures
  - arbitrary data structures
  - specify function at high-level and exploit compilers and debuggers
- To save hardware
  - if function requires too much logic when implemented with gates and FFs
    - operations are too complex, better broken down as instructions
    - lots of data manipulation (memory)
  - if function does not require higher performance of customized logic
    - ever-increasing performance of processors puts more and more applications in this category
    - minimize the amount of external logic

Microprocessor basics

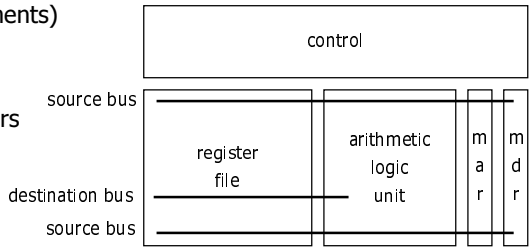
- Composed of three parts
  - data-path: data manipulation and storage
  - control: determines sequence of actions executed in data-path and interactions to be had with environment
  - interface: signals seen by the environment of the processor
- Instruction execution engine: fetch/execute cycle
  - flow of control determined by modifications to program counter
  - instruction classes:
    - data: move, arithmetic and logical operations
    - control: branch, loop, subroutine call
    - interface: load, store from external memory

Microprocessor basics (cont'd)

- Can implement arbitrary state machine with auxiliary data-path
  - control instructions implement state diagram
  - registers and ALUs act as data storage and manipulation
  - interaction with the environment through memory interface
  - how are individual signal wires sensed and controlled?

Microprocessor organization

- Controller
  - inputs: from ALU (conditions), instruction read from memory
  - outputs: select inputs for registers, ALU operations, read/write to memory
- Data-path
  - register file to hold data
  - arithmetic logic unit to manipulate data
  - program counter (to implement relative jumps and increments)
- Interface
  - data to/from memory (address and data registers in data path)
  - read/write signals to memory (from control)



General-purpose processor

- Programmed by user
- New applicactions are developed routinely
- General-purpose
  - must handle a wide ranging variety of applications
- Interacts with environment through memory
  - all devices communicate through memory data
  - DMA operations between disk and I/O devices
  - dual-ported memory (as for display screen)
  - oblivious to passage of time (takes all the time it needs)

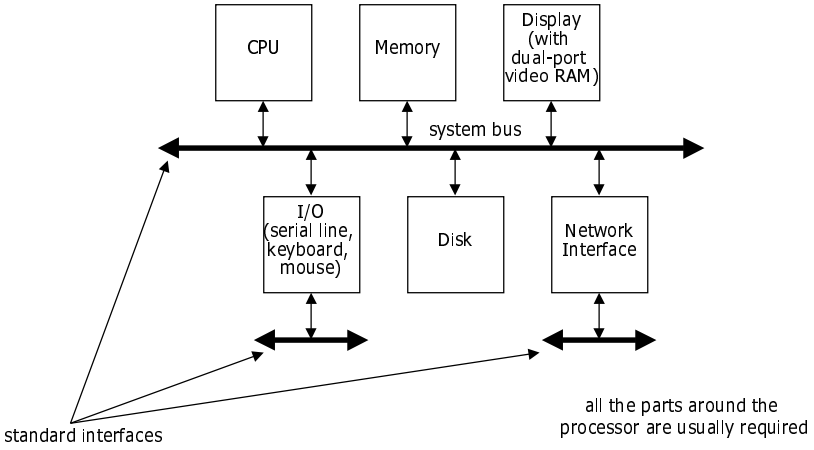
Embedded processor

- Programmed once by manufacturer of system
- Executes a single program (or a limited suite) with few parameters
- Task-specific
  - can be optimized for specific application
- Interacts with environment in many ways
  - direct sensing and control of signal wires
  - communication protocols to environment and other devices
  - real-time interactions and constraints

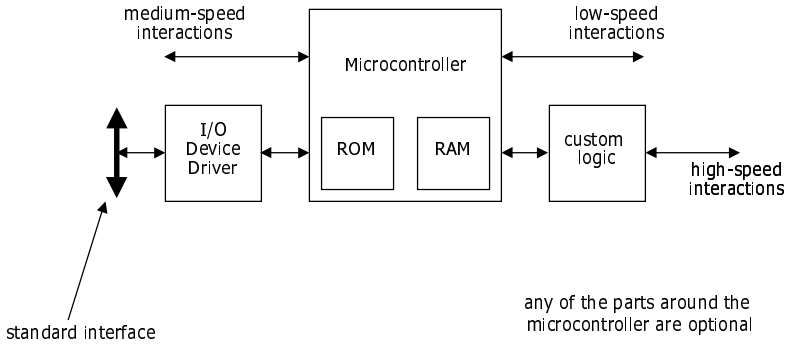
Why embedded processors?

- Overhead associated with building a general-purpose system
  - loading programs
  - operating system to store programs and data
  - shared system resources (e.g., system bus, large memory)
  - many parts
    - communication through shared memory/bus
    - each I/O device requires its own hardware
- Optimization opportunities
  - as much hardware as necessary for application
    - cheaper, portable, lower-power systems
  - as much software as necessary for application
    - no complete OS requirement
  - can integrate processor, memory, and I/O devices onto a single-chip

Typical general-purpose architecture



Typical task-specific architecture



How does this change things?

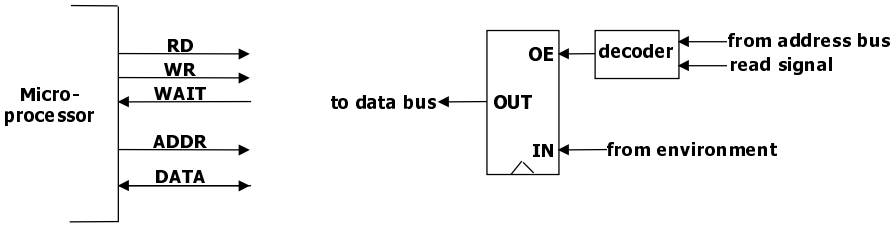
- Sense and control of environment
  - processor must be able to "read" and "write" individual signal wires
  - controls I/O devices directly
- Measurement of time
  - many applications require precise spacing of events
  - reaction times to external stimuli may be constrained
- Communication
  - protocols must be implemented by processor
  - integrate I/O device or emulate in software
  - capability of using external device if necessary

Interactions with the environment

- Basic processor only has address and data busses to memory
- Inputs are read from memory
- Outputs are written to memory
- Thus, for a processor to sense/control signal wires in the environment they must be made to appear as memory bits
  - how do we make wires look like memory?

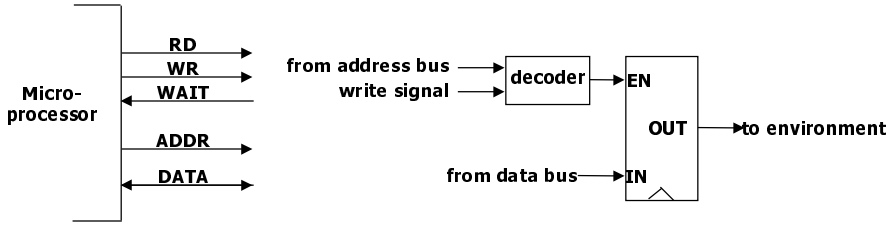
Sensing external signals

- Map external wire to a bit in the address space of the processor
- External register or latch buffers value from environment
  - map register into address space
  - decoder needed to select register
  - register has output enable so that many can be connected to data bus



Controlling external signals

- Map external wire to a bit in the address space of the processor
- Connect output of memory-mapped register to environment
  - map register into address space
  - decoder need to select register
  - register had input enable so that many can be connected to data bus
  - holds value indefinitely



Time and instruction execution

- Keep track of detailed timing of each instruction's execution
  - highly dependent on code
  - hard to use compilers
  - not enough control over code generation
  - interactions with caches/instruction-buffers
- Loops to implement delays
  - keep track of time in counters
  - keeps processor busy counting and not doing other useful things
- Real-time clock
  - take differences at different points in the program

Time measurement via parallel timers

- Separate and parallel counting unit(s)
  - co-processor to microprocessor
  - does not require microprocessor intervention
  - in simple case, like a real-time clock
  - set timer, interrupt generated when expired
- More interesting timer units
  - self reloading timers for regular interrupts
  - pre-scaling for measuring larger times
  - started by external events

Input/output events

- Input capture
  - record time when input event occured
  - to be used in later handling of event
- Output compare
  - set output to happen at a point in the future
  - reactive outputs – set output to happen a pre-defined time after some input
  - processor can go on to do other things in the meantime

System bus based communication

- Extend address/data bus to outside
- Use specialized devices to implement communication protocol
- Map devices and their registers to memory locations
- Read/write data to receive/send in buffers in device or shared memory
- Poll register in device for status
- Wait for interrupt from device on interesting events
  - send completed
  - receive occured

Support for communication protocols

- Built-in device drivers
  - for common communication protocols
  - serial-line protocols most common as they require few pins
- Serial-line controller
  - special register in memory space for interaction
  - may use timer unit(s) to generate timing events
    - for spacing of bits on signal wire
    - for sampling rate
- Increase level of integration
  - no external devices
  - may further eliminate need for shared memory or system bus

Microcontrollers

- Embedded processor with much more integrated on same chip
  - processor core + co-processors + memory
  - ROM for program memory, RAM for data memory, special registers
  - parallel I/O ports to sense and control wires
  - timer units to measure time in various ways
  - communication subsystems to permit direct link

Microcontrollers (cont'd)

- Other features not usually found in general-purpose CPUs
  - expanded interrupt handling capabilities
    - multiple interrupts with priority and selective enable/disable
    - automatic saving of context before handling interrupt
    - interrupt vectoring to quickly jump to handlers
  - more instructions for bit manipulations
    - support operations on bits (signal wires) rather than just words
- Integrated memory and support functions for cheaper system cost
  - built-in EPROM, Flash, and/or RAM
  - DRAM controller to handle refresh
  - page-mode support for faster block transfers

Microcontroller example

- Motorola 68HC11 (A1 model)
  - 8-bit microcontroller with a M6800/M6801 processor core
  - on-board 512-byte EEPROM and 256-byte RAM
  - 16-bit timer system with input capture and output compare
  - built-in A/D converter for 8 analog inputs
  - serial communications interfaces
- Development board for 68HC11 (M68HC11EVB)
  - 68HC11A1 system with 8K user EPROM and up to 16K user RAM
  - port replacement unit (to regain I/O ports used for memory addressing)
  - two serial communications interfaces
  - on-board monitor program for downloading and debugging programs
    - includes basic I/O utilities

MC68HC11 and EVB

- Review manuals
  - learn how to read documentation
- Instruction set
  - instruction capabilities
  - timing
- Special registers and integrated I/O devices
  - input capture
  - output compare
  - analog/digital conversion
- Interrupt organization
- Memory space and its allocation
- Timers

Why this choice?

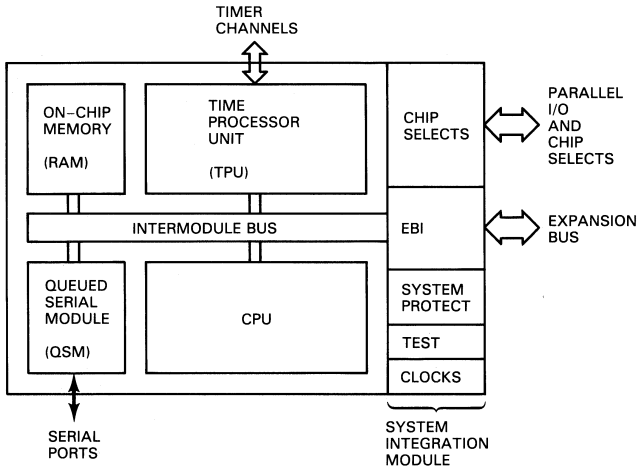
- 68HC11
  - basic microcontroller with simple instruction set
  - full-featured (many of the same capabilities as larger microcontrollers)
  - good real-time capabilities
  - A/D conversion built-in
  - cheap
  - very common (lots of resources available including on web)
  - public domain C compiler available
- EVB
  - fully integrated system (only need to add project-specific hardware)
  - debugging support (BUFFALO ROM)
  - serial communication to PC

Motorola MC683xx

- Designed by Motorola for automotive, data acquisition, printers, plotters, cameras, and other consumer products
- Introduced in 1989 (9 years ago)
- General characteristics
  - operate in harsh environment (-40 to 125°C and noise)
  - low power (~625mW), low cost ~$12
  - crunch numbers (execute control loops)
  - clock frequency (up to 78MHz)
- Functional units
  - CPU32 (68020 processor core)
  - SIM (System Integration Module)
  - GPT (General Purpose Timer)
  - QSM (Queued Serial Module)
  - TPU (Time Processor Unit)

683xx

Block diagram



EBI: External Bus Interface

Functional blocks

- CPU32
  - 32-bit processor based on 68020
- System Integration Module
  - controls external bus, I/O functions and system clock
- Time Processor Unit
  - 16-channel timer/counter unit
  - controls internal/external events
- Memory Module
  - on-chip RAM with stand-by power feature
- A/D
  - 16 channel queued
- CAN protocol
  - Controller Area Network (automotive bus protocol)

Time Processor Unit

- A microcontroller dedicated to timing control
- Two 16-bit timers/registers
  - free running based on system clock
  - can be controlled with external clock
- 16 independent channels (input or output)
- Functions
  - input capture
  - period or pulse width accumulation
  - output compare
  - pulse width modulation
  - stepper motor control
  - automotive functions (fire spark plugs, determine engine rotation)

TPU Register Map

	ADDRESS	WORD (15 BYTE n 8 | 7 BYTE n+1 0)
S	$YFFE00	MODULE CONFIGURATION REGISTER
S	$YFFE02	TEST CONFIGURATION REGISTER
S	$YFFE04	DEVELOPMENT SUPPORT CONTROL REGISTER
S	$YFFE06	DEVELOPMENT SUPPORT STATUS REGISTER
S	$YFFE08	INTERRUPT CONFIGURATION REGISTER
S	$YFFE0A	INTERRUPT ENABLE REGISTER
S	$YFFE0C	CHANNEL FUNCTION SELECT REGISTER 0
S	$YFFE0E	CHANNEL FUNCTION SELECT REGISTER 1
S	$YFFE10	CHANNEL FUNCTION SELECT REGISTER 2
S	$YFFE12	CHANNEL FUNCTION SELECT REGISTER 3
X	$YFFE14	HOST SEQUENCE REGISTER 0
X	$YFFE16	HOST SEQUENCE REGISTER 1
X	$YFFE18	HOST SERVICE REQUEST REGISTER 0
X	$YFFE1A	HOST SERVICE REQUEST REGISTER 1
S	$YFFE1C	CHANNEL PRIORITY REGISTER 0
S	$YFFE1E	CHANNEL PRIORITY REGISTER 1
S	$YFFE20	INTERRUPT STATUS REGISTER
S	$YFFE22	LINK REGISTER
S	$YFFE24	SERVICE GRANT LATCH REGISTER
S	$YFFE26	DECODED CHANNEL NUMBER REGISTER
	$YFFE28	RESERVED
	$YFFEFF	

TPU Register Set (all)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMCR $YFFE00	STOP 0	TCR1P PRESCALER 0			TCR2P PRESCALER 0		EMU 0	T2CG 0	STF 0	SUPV 1	RSCK 0	0	0	INTERRUPT ARBITRATION ID 0		
TTCR $YFFE02	0	0	0	INCAD 0	TCR1C 0		ACUTR1–0 0 0		0	0	TSOSEL2–0 0 0 0		TSSEL2–0 0 0 0			TMM 0
DSCR $YFFE04	HOT4 0	0	0	0	0	BLC 0	CLKS 0	FRZ1–0 0 0		CCL 0	BP 0	BC 0	BH 0	BL 0	BM 0	BT 0
DSSR $YFFE06	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	BKPT 0	PCBK 0	CHBK 0	SRBK 0	TPUF 0	0 0	0 0	0 0
TICR $YFFE08	0	0	0	0	0	CHANNEL INTERRUPT REQUEST LEVEL 0 0 0			CHANNEL BASE VECTOR 0 0 0 0				0	0	0	0
CIER $YFFE0A	CHAN 15 0	CHAN 14 0	CHAN 13 0	CHAN 12 0	CHAN 11 0	CHAN 10 0	CHAN 9 0	CHAN 8 0	CHAN 7 0	CHAN 6 0	CHAN 5 0	CHAN 4 0	CHAN 3 0	CHAN 2 0	CHAN 1 0	CHAN 0 0
CFS R0 $YFFE0C	CHANNEL 15 0 0 0 0				CHANNEL 14 0 0 0 0				CHANNEL 13 0 0 0 0				CHANNEL 12 0 0 0 0			
CFS R1 $YFFE0E	CHANNEL 11 0 0 0 0				CHANNEL 10 0 0 0 0				CHANNEL 9 0 0 0 0				CHANNEL 8 0 0 0 0			
CFS R2 $YFFE10	CHANNEL 7 0 0 0 0				CHANNEL 6 0 0 0 0				CHANNEL 5 0 0 0 0				CHANNEL 4 0 0 0 0			
CFS R3 $YFFE12	CHANNEL 3 0 0 0 0				CHANNEL 2 0 0 0 0				CHANNEL 1 0 0 0 0				CHANNEL 0 0 0 0 0			
HSQR0 $YFFE14	CHANNEL 15 0 0		CHANNEL 14 0 0		CHANNEL 13 0 0		CHANNEL 12 0 0		CHANNEL 11 0 0		CHANNEL 10 0 0		CHANNEL 9 0 0		CHANNEL 8 0 0	
HSQR1 $YFFE16	CHANNEL 7 0 0		CHANNEL 6 0 0		CHANNEL 5 0 0		CHANNEL 4 0 0		CHANNEL 3 0 0		CHANNEL 2 0 0		CHANNEL 1 0 0		CHANNEL 0 0 0	
HSRR0 $YFFE18	CHANNEL 15 0 0		CHANNEL 14 0 0		CHANNEL 13 0 0		CHANNEL 12 0 0		CHANNEL 11 0 0		CHANNEL 10 0 0		CHANNEL 9 0 0		CHANNEL 8 0 0	
HSRR1 $YFFE1A	CHANNEL 7 0 0		CHANNEL 6 0 0		CHANNEL 5 0 0		CHANNEL 4 0 0		CHANNEL 3 0 0		CHANNEL 2 0 0		CHANNEL 1 0 0		CHANNEL 0 0 0	
CPR0 $YFFE1C	CHANNEL 15 0 0		CHANNEL 14 0 0		CHANNEL 13 0 0		CHANNEL 12 0 0		CHANNEL 11 0 0		CHANNEL 10 0 0		CHANNEL 9 0 0		CHANNEL 8 0 0	
CPR1 $YFFE1E	CHANNEL 7 0 0		CHANNEL 6 0 0		CHANNEL 5 0 0		CHANNEL 4 0 0		CHANNEL 3 0 0		CHANNEL 2 0 0		CHANNEL 1 0 0		CHANNEL 0 0 0	
CISR $YFFE20	CHAN 15 0	CHAN 14 0	CHAN 13 0	CHAN 12 0	CHAN 11 0	CHAN 10 0	CHAN 9 0	CHAN 8 0	CHAN 7 0	CHAN 6 0	CHAN 5 0	CHAN 4 0	CHAN 3 0	CHAN 2 0	CHAN 1 0	CHAN 0 0

* Y = m111, where m is the state of the modmap bit in the module configuration register of the system integration

TPU Register Set (detail)

HSQR1 $YFFE16	CHANNEL 7	0 0	CHANNEL 6	0 0	CHANNEL 5	0 0	CHANNEL 4	0 0	CHANNEL 3	0 0	CHANNEL 2	0 0	CHANNEL 1	0 0	CHANNEL 0	0 0
HSRR0 $YFFE18	CHANNEL 15	0 0	CHANNEL 14	0 0	CHANNEL 13	0 0	CHANNEL 12	0 0	CHANNEL 11	0 0	CHANNEL 10	0 0	CHANNEL 9	0 0	CHANNEL 8	0 0
HSRR1 $YFFE1A	CHANNEL 7	0 0	CHANNEL 6	0 0	CHANNEL 5	0 0	CHANNEL 4	0 0	CHANNEL 3	0 0	CHANNEL 2	0 0	CHANNEL 1	0 0	CHANNEL 0	0 0
CPR0 $YFFE1C	CHANNEL 15	0 0	CHANNEL 14	0 0	CHANNEL 13	0 0	CHANNEL 12	0 0	CHANNEL 11	0 0	CHANNEL 10	0 0	CHANNEL 9	0 0	CHANNEL 8	0 0
CPR1 $YFFE1E	CHANNEL 7	0 0	CHANNEL 6	0 0	CHANNEL 5	0 0	CHANNEL 4	0 0	CHANNEL 3	0 0	CHANNEL 2	0 0	CHANNEL 1	0 0	CHANNEL 0	0 0
CISR $YFFE20	CHAN 15	0	CHAN 14	0	CHAN 13	0	CHAN 12	0	CHAN 11	0	CHAN 10	0	CHAN 9	0	CHAN 8	0
	CHAN 7	0	CHAN 6	0	CHAN 5	0	CHAN 4	0	CHAN 3	0	CHAN 2	0	CHAN 1	0	CHAN 0	0

Y = m111, where m is the state of the modmap bit in the module configuration register of the system integration

Timing Ranges

Table 13.3 TCR1 Frequencies, Resolutions and Range

PSCK	TMCR[14:13]	Frequency	Resolution	Range
PSCK = 1	{00}/4	4.19MHz	238.4ns	15.6ms
	{01}/8	2.097MHz	476.8ns	31.25ms
	{10} /16	1.049MHz	953.7ns	62.5ms
	{11} /32	0.524MHz	1.91μs	125ms
PSCK = 0	{00}/32	524.3kHz	1.91μs	125ms
	{01}/64	262.1kHz	3.81μs	0.25 sec
	{10} /128	131.1kHz	7.63μs	0.5 sec
	{11} /256	65.53kHz	15.3μs	1.0 sec

Note:
The values were calculated using a 2^{24}Hz (16.78MHz) system clock.

Programming the TPU

```
abs.   LC   obj. code   source line
----   ----  ---------   -----------
   1   0000              |        TTL     'FIGURE 13.6'
   2   0000              |*
   3   0000              |*       INITIALIZE THE TPU
   4   0000              |*        TCR1-SYSTEM CLOCK/128
   5   0000              |*        TCR2-EXTERNAL CLOCK
   6   4500              |         ORG    $4500
   7   0000              |*
   8         0080 0000   |REGBASE  EQU    $800000              ;REGS AT $FFF000
   9         00FF FE00   |TMCR     EQU    $007FFE00+REGBASE    ;CONFIGURATION
  10         00FF FE08   |TICR     EQU    $007FFE08+REGBASE    ;CPU INTERRUPT LEVEL
  11         00FF FE0A   |TCIER    EQU    $007FFE0A+REGBASE    ;TPU CHANNEL INTERRUPTS
  12         00FF FE1C   |TCPR0    EQU    $007FFE1C+REGBASE    ;CHANNEL PRIORITY
  13         00FF FE1E   |TCPR1    EQU    TCPR0+2
  14         00FF FE20   |TCISR    EQU    $007FFE20+REGBASE    ;TPU INTERRUPT STATUS
  15   4500              |*
  16   4500 3F00         |TPUINIT  MOVE.W  D0,-(SP)             ;SAVE REGISTER
  17   4502 33FC 0000    |         MOVE.W  #$0000,TCPR0         ;DISABLE ALL CHANNELS
       4506 00FF FE1C    |
  18   450A 33FC 0000    |         MOVE.W  #$0000,TCPR1
       450E 00FF FE1E    |
  19   4512 3039 00FF    |         MOVE.W  TCISR,D0             ;CLEAR ALL TPU INTERRUPTS
       4516 FE20         |
  20   4518 33FC 0000    |         MOVE.W  #$0000,TCISR         ;  REQUESTS
       451C 00FF FE20    |
  21   4520 33FC 4002    |         MOVE.W  #$4002,TMCR          ;TCR1-DIVIDE BY 32x4
       4524 00FF FE00    |
  22   4528              |*                                     ;USER ACCESS;IARB=2
  23   4528 33FC 0450    |         MOVE.W  #$0450,TICR          ;CPU LEVEL 4;VECTOR=$50
       452C 00FF FE08    |
  24   4530 33FC FFFF    |         MOVE.W  #$FFFF,TCIER         ;ENABLE ALL TPU INTERRUPTS
       4534 00FF FE0A    |
  25   4538              |*
  26   4538 301F         |         MOVE.W  (SP)+,D0             ;RESTORE REGISTER
  27   453A 4E75         |         RTS
  28   453C              |         END
```

Figure 13.6 Program example of TPU global initialization.

Clocks and Timers

- System clock
- Periodic interrupt timer
- Software watchdog timer
- Bus monitor
- Synchronous communication baud rate, delay, min idle
- Asynchronous communication baud rate
- TPU timers

MC68328 aka DragonBall

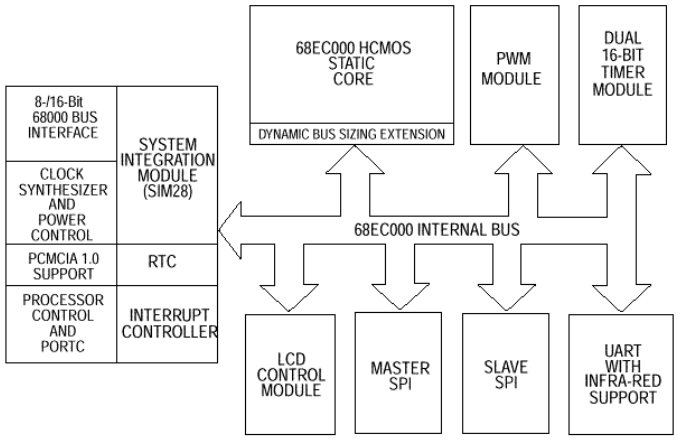


Figure 1. MC68328 Block Diagram

PWM on DragonBall

- PWM transforms duty cycle into an average analog value
- In addition to controlling motor velocity, can synthesize music

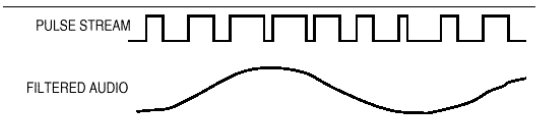
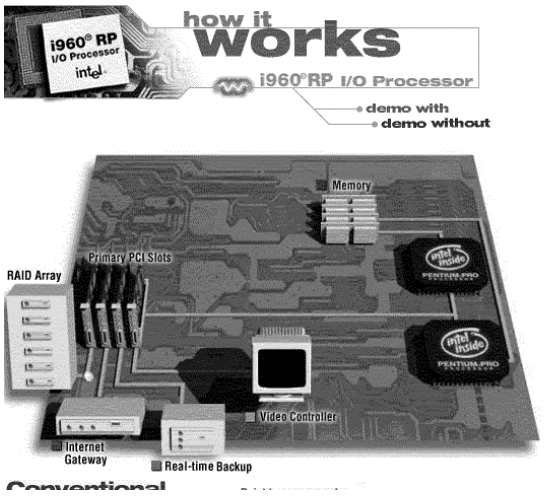


Figure 14-2. PWM Generating Audio

I/O Bottleneck

- Off-chip communication very slow relative to processing speeds
  - processor generating lots of memory requests
  - multiple devices competing for system bus
- Technology wave increasing the gap in performance
  - processors running faster
  - interconnect bandwidth increasing at slower rate
- How do local area networks handle this problem?
  - restrict communication
  - hubs, gateways
- Solution
  - off load communication to a communication processor

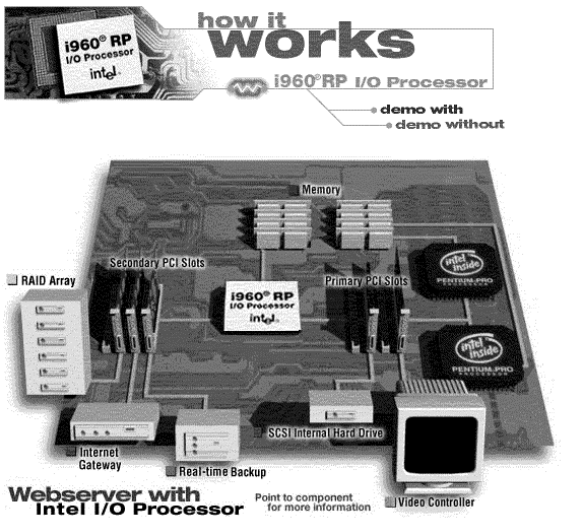
System Communication



Intel's Approach

- A two-pronged attack
  - hardware: communication processor
  - software: device-drivers
- Use a RISC processor to handle communication for the main processor
  - isolate main processor from slower peripherals
  - reduce interrupts to CPU
  - support for system busses running at different clock rates
- Define a new standard: Intelligent I/O (eye, two, oh)
  - standardize device-drivers
  - minimize OS dependencies
  - increase system throughput

System w/ Communication Processor

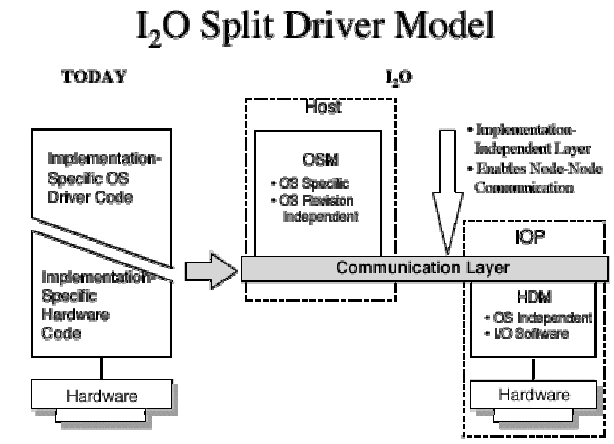


I_2O

- Goals
  - abstract the I/O subsystem
  - improve system throughput
  - enable rapid deployment of new I/O technology
- Message-passing paradigm
  - supports peer communication
  - simplifies device-drivers

I_2O Device-Drivers

Figure 1

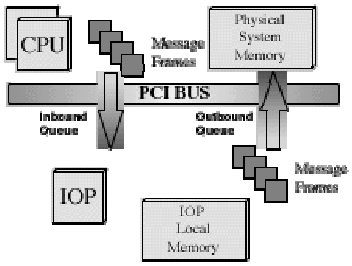


I_2O Message Passing

Figure 2

I_2O Message Interface

- Allows OS to *batch* I/O requests
  - Eliminates CPU stall during I/O transactions
  - Increases system TPS
- Consistent hardware interface
  - Simplified software model for I_2O driver
  - OS support becomes pervasive
  - Narrows requirements for compliance



Intel 80960RP Processor

- Main features
  - PCI to PCI bridge
  - messaging unit
    - mechanism to transfer data between the PCI system and 80960
  - DMA access to both PCI busses
  - address translation units
    - 64-byte input and output queues
    - queues allow transactions to complete on initiating bus before they complete on target bus
  - 7 to 150 MIPS

i960 Block Diagram

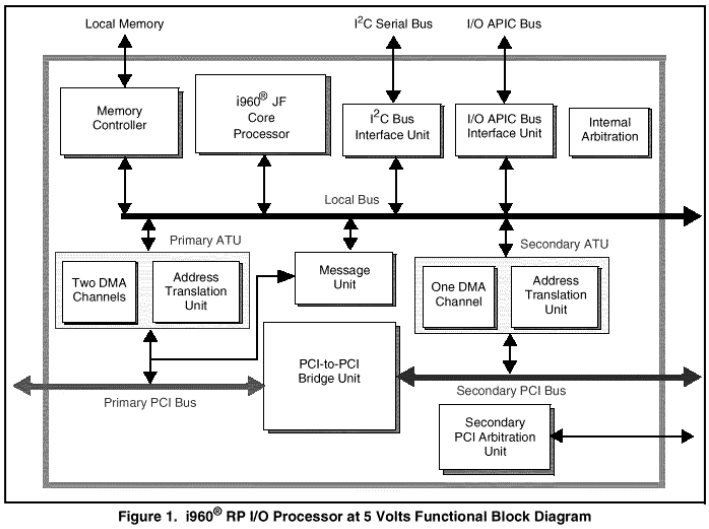


Figure 1. i960® RP I/O Processor at 5 Volts Functional Block Diagram

i960 Core

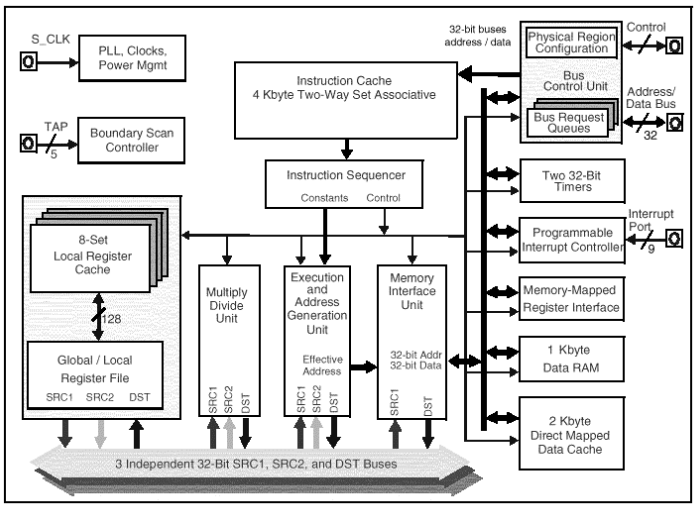
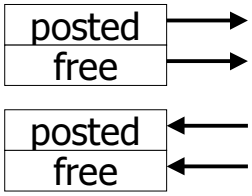


Figure 2. 80960JF Core Block Diagram

80960 Messaging Unit

- Message registers
  - i960 sends and receives messages via special registers
  - when written an interrupt generated to i960 or PCI
- Doorbell registers
  - inbound and outbound doorbells
  - either hardware or software can generate doorbell interrupts
  - contain interrupt status from other message unit mechanisms
- Circular queues
  - two inbound and two outbound queues
  - posted messages contain orders
  - free messages indicate operation completed and message can be reused
- Index registers
  - 1004 registers for inbound messages



Motorola's Approach

- Use 2 RISC processors
  - 32-bit PowerPC core for application code
  - 32-bit customized RISC for imaging and communication
- Put communication processor on same die as main processor
  - system on a chip
  - incorporate features common in portable devices
  - single-chip solution for consumer electronics
  - signal processing functions

MPC823

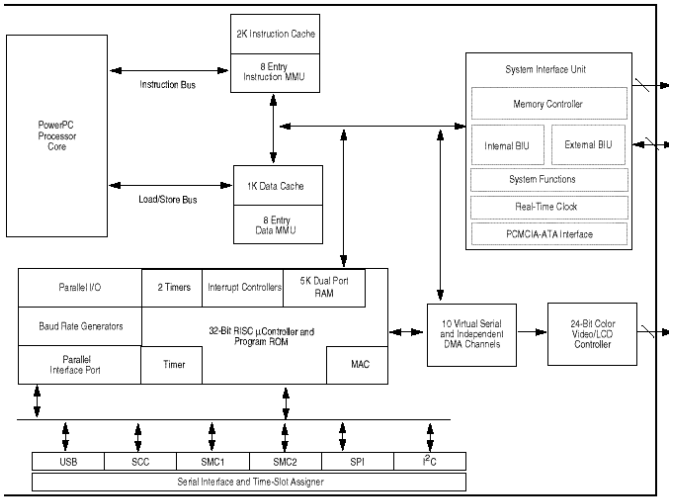


Figure 1. MPC823 Microprocessor Block Diagram

MPC 823 Features

- Performance of 66 MIPS @ 50MHz
- Low power
  - consumes less than 180mW @ 25MHz
  - 2.2V internal, 3.3V I/O boundary
  - doze - functional units disabled except
    - PLL, mem-controller, real-time clock, LCD, comm. proc in standby
  - sleep - all units disabled except
    - real-time clock, periodic interrupt timer, PLL active for fast wakeup
  - deep sleep - all units disabled except
    - real-time clock, periodic interrupt timer
- 1K data cache and 2K instruction cache
- PCMCIA support

MPC823 Communication Module

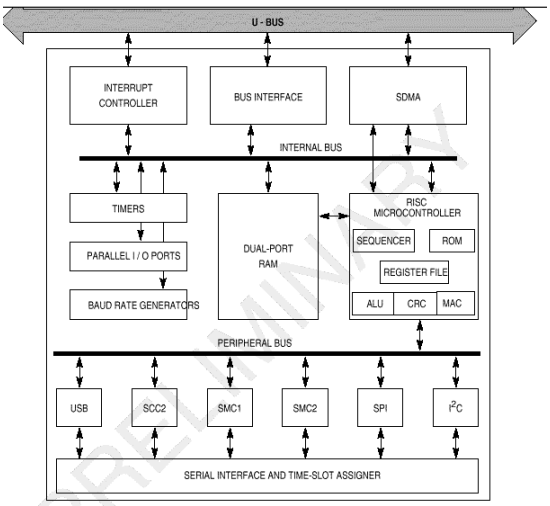


Figure 16-1. CPM Block Diagram

PRELIMINARY

Communication Module Features

- Interfaces to PPC core via on-chip dual ported RAM
- Protocols supported include
  - Ethernet/ IEEE 802.3
  - Universal Serial Bus (USB)
  - AppleTalk
  - UART/USART
  - IrDA 1.1
  - ISDN
  - I^2C
- 16x16 bit multiply accumulate (MAC)
- DSP functions
  - V.32bis/V.34bis datapump filter
  - JPEG compression
- Four independent baud rate generators

More Communication Module Features

- Video/LCD controller
  - supports passive LCD and NTSC/PAL encoders
  - end of frame interrupt generation
  - programmable display active area
  - programmable background color for inactive area
  - 1,2,4-bit per pixel grayscale
  - built-in color RAM with 256 12-bit entries
  - programmable polarity for all LCD interface signals

PDA Based on MPC823

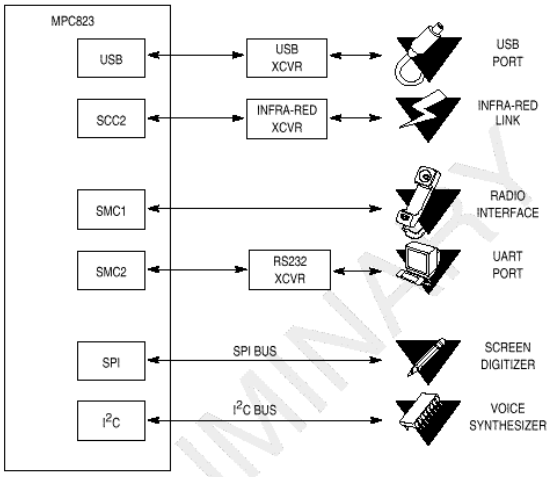
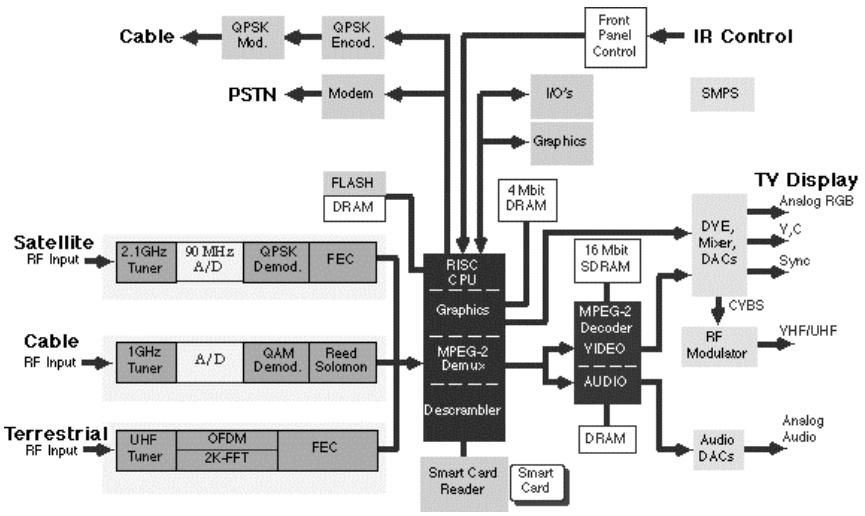


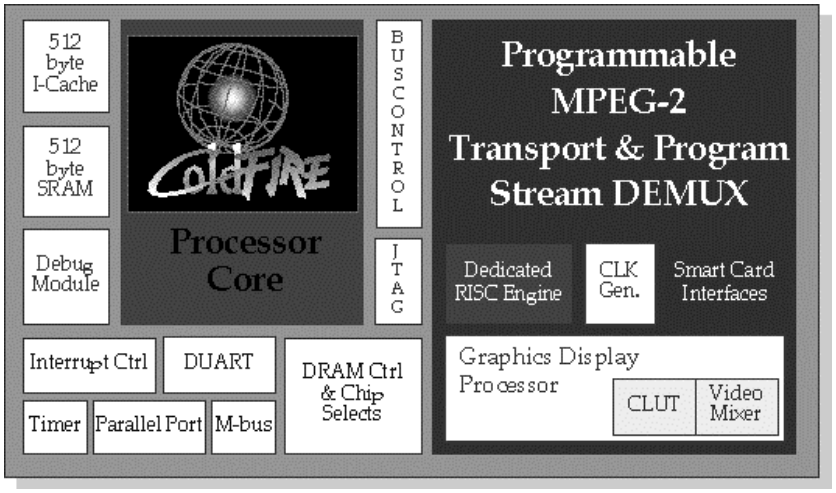
Figure 16-2. Example of a PDA Application

PRELIMINARY

Set-Top Box



Motorola's ColdFire



ColdFire Features

- Variable instruction length RISC 68K processor
  - subset of 68K instruction set
  - 16. 32. 48-bit instruction sizes
  - single-cycle instruction execution
  - tighter code density than 32 or 64-bit processors
- Completely synthesizable
  - core can be used in system on a chip designs
- Performance: 36 MIPS
- Price: ~$25