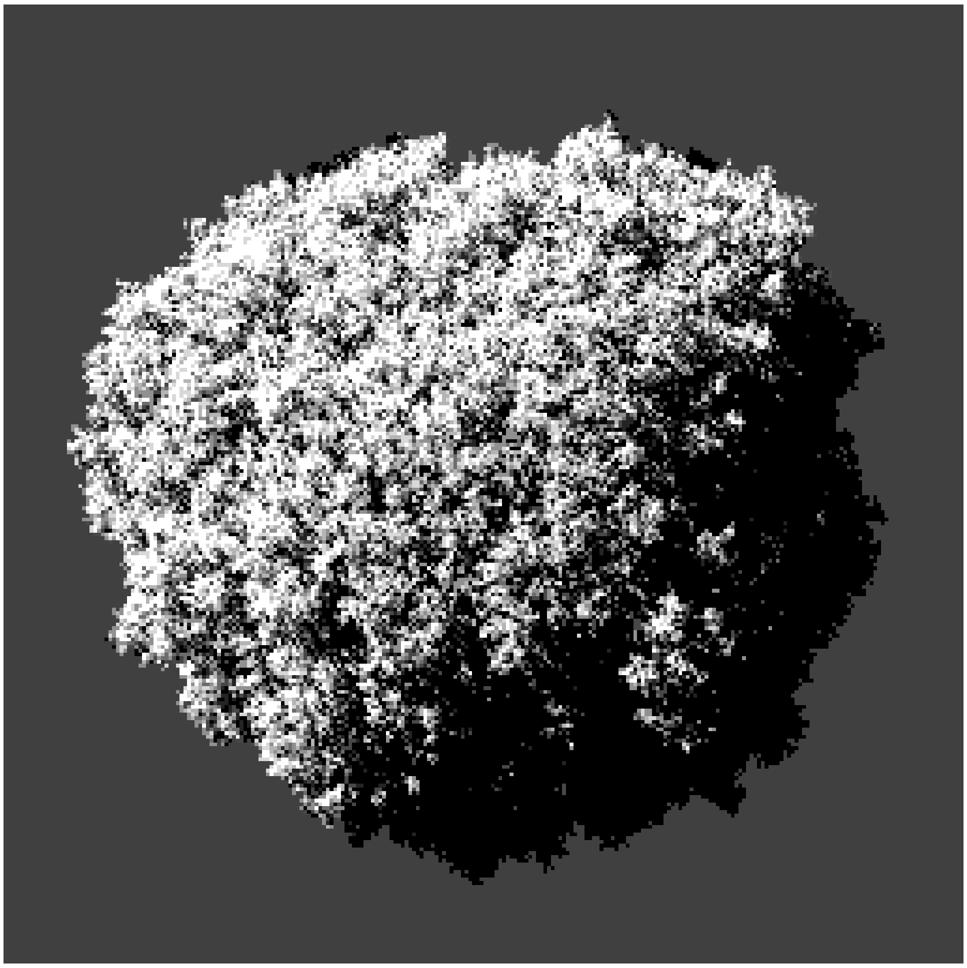
Norman Margolus
Tommaso Toffoli

STEP: a SpaceTime Event Processor

Hardware Reference

} (version 2.1.10)



MIT Laboratory for Computer Science
Cambridge, MA 02139

Contents

0 Introduction 1
- 0.1 Scope of this manual . . . 1
- 0.2 Complementary documentation . . . 1
- 0.3 Organization of this manual . . . 1
- 0.4 Conventions . . . 1
  - 0.4.1 Number representation . . . 1
  - 0.4.2 Data and variable ordering . . . 1
  - 0.4.3 Signal polarity . . . 2

1 STEP machine and controller 3
- 1.1 Overview . . . 3
- 1.2 Hardware hierarchy . . . 3
- 1.3 The STEP controller . . . 3
  - 1.3.1 SBus signals . . . 4
  - 1.3.2 CBus signals . . . 4
  - 1.3.3 CBus Sync . . . 4
- 1.4 The STEP machine . . . 5
- 1.5 Clock distribution and synchronization . . . 6
- 1.6 Interbox communication . . . 6

2 STEP box 7
- 2.1 Overview . . . 7
- 2.2 Glue ports . . . 9
- 2.3 Data flywheel ports . . . 9
- 2.4 Display port . . . 10
- 2.5 CBus ports . . . 10
- 2.6 Local bus . . . 10

3 STEP module 11
- 3.1 Module pin types . . . 11
- 3.2 CBus interface . . . 12
- 3.3 Glue . . . 12
- 3.4 Data stream . . . 12
- 3.5 Wired-AND communication . . . 12

4 STEPchip 13
- 4.1 STEPchip block diagram . . . 13
- 4.2 Pinout . . . 13
  - 4.2.1 Pin types . . . 13
  - 4.2.2 DRAM interface . . . 15
  - 4.2.3 SRAM interface . . . 15
  - 4.2.4 Glue . . . 15
  - 4.2.5 Data stream . . . 15
  - 4.2.6 CBus interface . . . 15
  - 4.2.7 Miscellaneous . . . 16
  - 4.2.8 Spares, reserved . . . 16
- 4.3 Scanning . . . 16

5 Video output 18
- 5.1 Color subcarrier frequency . . . 18
- 5.2 Horizontal scan frequency . . . 18
- 5.3 Square pixels and all that . . . 18
- 5.4 The Bt858 . . . 19
- 5.5 Bt478/Bt858 programming . . . 19

6 Video input 20
- 6.1 Camera . . . 20
  - 6.1.1 Camera switch settings . . . 20

0 Introduction

STEP is an indefinitely scalable multiprocessor architecture optimized for the fine-grained simulations of physical systems. From a functional viewpoint, the STEP architecture can be thought of as a spatially-extended, uniform, fine-grained medium of a rather general and abstract nature. Specific behavior is obtained by specifying the medium's connectivity and dynamical properties, and assigning its initial state. In this sense, STEP embodies the metaphor of "programmable matter".

0.1 Scope of this manual

The present *Hardware Reference* documents STEP's electronic, electrical, and mechanical hardware at a level appropriate for the following purposes

- Identifying the basic function of the different parts of a STEP unit, and comprehending the principles of operation of functional blocks at different levels.
- Assembling and configuring a STEP unit from manufactured subassemblies (enclosure, modules, cables, host, I/O devices).
- Locating components, identifying pins, and tracing signals on boards, connectors, and cables, as needed for diagnosing malfunctions or verifying the hardware response of software commands.

The manual does not discuss issues of design methodology, detailed electrical and timing parameters, mechanical tolerances, or other details that properly belong to the design stage. The emphasis is on information relevant for the *use* of the existing design.

0.2 Complementary documentation

The *Architecture Reference* discusses those functional and structural aspects of the STEP architecture that are of interest to the planner, the programmer, and the sophisticated user. A cursory reading of that manual is a prerequisite for the reading of the other manuals, including the present one.

The *Software Reference* documents STEP's system software and programming language, primarily addressing the needs of the applications programmer.

The *Fabrication Reference* is a collection of reference documents (parts lists, manufacturers lists, schematics, PCB artwork, mechanical drawings, pinout templates, data sheets, etc.) useful for ordering separate parts, planning design modfications, or planning the fabrication of a batch of STEP units of a given size.

The *User's Guide* is as programming tutorial that addresses the first concerns of the typical STEP user. It goes through simple applications in a step-by-step fashion, introducing programming constructs and techniques as they are needed and discussing methodological approaches appropriate to the STEP modeling medium.

0.3 Organization of this manual

0.4 Conventions

Here we list a number of general conventions followed throughout this manual.

0.4.1 Number representation

Numerals in typewriter font are in base sixteen; thus `40`=64 and `FF`=255.

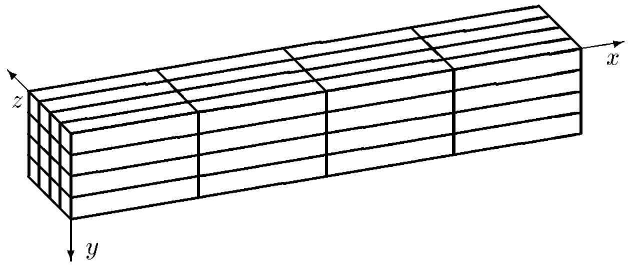
0.4.2 Data and variable ordering

In listings, tables, and figures, we had to fight a hard struggle to reconcile conflicting conventions concerning the ordering of items and the orientation of axes.

There are two basic sources of conflict. One conflict is between the "most significant first" approach, typified by the way we write a number such as '1992' starting with the thousands and ending with the units, and the "first causes first" approach, typified by the way the electronic engineer draws the stages of digital counter, starting with the units stage on the left and making the carry propagate left-to-right. The other conflict is between the way a mathematician would write an i, j array, with j increasing left-to-rigth and i top-to-bottom, in agreement with western-style writing, and the way a physicist would draw an x, y or an x, t diagram on the page, with x pointing rightwards but y or t pointing *upwards*. Note that the "mathematician's" approach is also the one used in scanning an ordinary CRT monitor—left-to-right and then top-to-bottom.

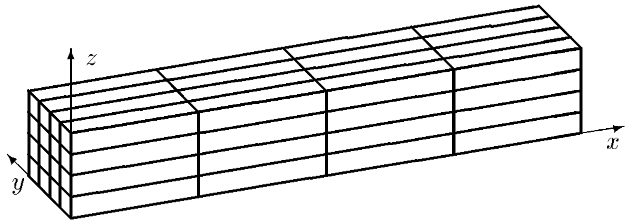
After much soul-searching, we took the responsibility to enforce a consistent, universal convention: in any enumeration, the natural ordering of the items (i.e., increasing values of the enumeration index) will be reflected by the typographical ordering (left-to-right, and, for a second index, top-to-bottom).

Thus, an array indexed by x and y will be displayed with x increasing from left to right and y from top to bottom, as in the following diagram



Here, the diagram's third axis, z, is represented as pointing *into* the page).

Note that, with these conventions, the "handedness" of the (x, y, z) frame is the same as that that results from the more conventional ordering of axes, illustrated by the following diagram



Similarly, a 16-bit computer register will invariably be represented as follows

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

and a text description of this register will list individual bits or fields in same order, as in the following example:

0–6	ASCII character
7	Parity bit
8–15	Not used

When the contents of the register is interpreted as a binary-encoded integer by the computer's arithmetic-logic unit or its address-arithmetic unit, bit 0 will represent the least significant binary digit (units), and in general bit n will represent the coefficient of 2^n.

The above ordering convention will be broken in *only one circumstance*. In deference to universally ingrained practice, ordinary numerals in the text will be written as usual with the most significant digit first. Thus, the number represented by the following register contents

1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0															15

will be written, in text form, as hexadecimal `00F1` or decimal 237.

0.4.3 Signal polarity

Signal names, typeset in sans-serif face, are chosen so as to have appropriate mnemonic value. For instance, the CBus Reset signal has a value of 1 (logical TRUE) when the STEP machine is being reset, and 0 (logical FALSE otherwise.

Physically, on the the CBus wires, this signal travels in complemented form, indicated by Reset*. However, we shall usually mention the signal in its native form, saying, for instance, that "in certain conditions Reset is asserted"; this implies that, in those conditions, the physical signal Reset* is negated.

The complemented form, Reset*, would be used only when tabulating the pinout of a chip or connector, to make it clear that the physical signal present on that pin is the complement of Reset.

1 STEP machine and controller

1.1 Overview

A complete STEP system consists of a three-dimensional array of STEP **boxes** driven by a **host** via a tree-shaped network called the **CBus**, as sketched in Fig. 1.1 (where only two dimensions of the array are shown).

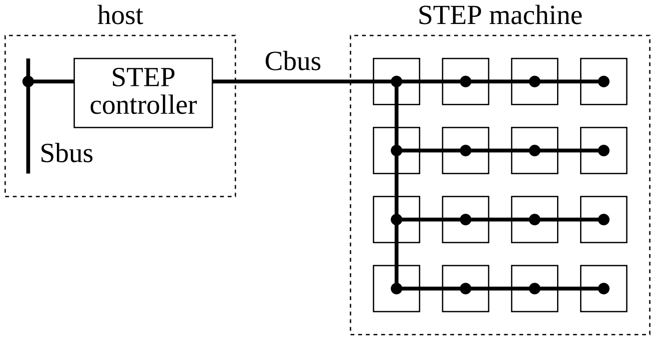


Figure 1.1: A STEP system consists of an array of STEP boxes (STEP *machine*) driven by a SPARCstation (*host*). The STEP *controller* interfaces the host's Sbus with the machine's CBus.

In the current implementation, the host is a SUN SPARCstation, which features a high-performance external bus called the SBus. The CBus is driven at the root by the STEP **controller**—a small custom board plugged into one of the SBus slots. From this board, a short cable—the CBus **root cable**—leads directly to one of the boxes, located on a corner of the array. From this **corner box** the CBus branches out through the rest of the array via CBus **branch cables**. We shall call STEP **machine** the interconnected array of STEP boxes, when viewed as distinct from the host and the controller.

1.2 Hardware hierarchy

Each **box** of the STEP machine array realizes, by means of eight identical printed-circuit boards called **modules**, a functional subunit of the STEP architecture—typically, a *block* of eight *sectors* (see beginning of Chapter 2; other ways to configure the contents of a box as a functional subunit are possible). Besides, the box provides mechanical and electrical support (backplane, enclosure, power supply and cooling) to the modules themselves, and may house additional support circuitry. (In a multi-box machine, some of these support functions need not be provided on a per-box basis; for instance, it may be more practical to provide a centralized power supply, or to house several boxes in one enclosure.) Communication between a STEP unit and the outside world (including the controller located in the host and optional I/O devices such as VGA monitors, video cameras, and video monitors; see §??) goes through connectors located on the boxes' **backpanels** (§2.1).

In turn, each module contains, besides conventional DRAM and SRAM chips, sixteen copies of the **STEPchip**—a custom VLSI chip that one may view as embodying the STEP architecture. In fact, each STEPchip has complete control over one DRAM chip's data and control lines; and, collectively, the sixteen STEPchips have complete control over the SRAM chips' data and control lines—as well as over transactions over the CBus.

Chapters 1–4 describe the STEP system at four levels of the design hierarchy, namely,

1. the STEP machine with its controller (this chapter),
2. the STEP box (Chapter 2),
3. the STEP module (Chapter 3), and
4. the STEPchip (Chapter 4).

For each level, formal interface specifications (pinout and signal description) with the level above are given.

The STEPchip (Level 4) is seen by the programmer as a set of registers accessible via the CBus. [**Complete and update**] Chapters ?? and following expand Chapter 4 by giving a complete programmer's model of the STEPchip: it systematically describes structure and functions of these registers, giving, whenever appropriate, a functional model of the hardware controlled by the registers themselves. ♠

One may view the host SPARCstation as representing Level 0 of the hierarchy. Its relevant hardware and systems-software aspects are documented by SUN Microsystems. STEP-specific software is documented in the *Software Reference* and the *User's Guide*.

1.3 The step controller

The STEP controller is a small board with, on one edge, a connector that plugs into one of the SBus slots. At the opposite edge of the card, a connector that shows through the rear of the SPARCstation represents the root of the CBus.

The main *raison d'être* of this piece of hardware is to provide an interface between the two buses, and to do so in such a way as to lessen the requirements for real-time response from the host.

While the SBus and the CBus are both synchronous, and at present have roughly comparable bandwidths, they are timed by independent clocks which are typically running at different rates. Moreover, the number and nature of the control and data lines, the protocols, and the signals' electrical specifications are quite different in the two buses.

The STEP controller requires bus-master as well as bus-slave capabilities.[1] In fact, in order to decouple STEP from the poor and rather impredictable real-time response of standard UNIX, the controller's normal operating mode is to act as a bus master and interpret according to the STEP machine's timing requirements a list of commands (**step list**) directly off the SPARCstation's memory, without going through the CPU. These commands would previously have been written to memory by the CPU at its own leisure.

The controller is addressed as a slave only at initialization time or to recover from runaway situations.

Clocking for the STEP machine is generated by the controller; a clock line in the CBus root cable carries a clock reference signal from the controller to the machine. Thus, the interface between circuitry clocked by the SBus and circuitry clocked by the CBus lies within the controller, which has provisions for arbitrating between the two clocks. See §1.5 for details of clock distribution and synchronization.

The principles of operation of the STEP controller (i.e., how it interprets a script consisting of steplist commands, initiating and supervising any SBus or CBus transactions that may be called for) are explained in the *Software Reference*.

1.3.1 SBus signals

The STEP controller interfaces with the SBus by means of a standard SBus connector. Signal description and pinout can be found in *The SBus Specification* (specifically, Tables 2-1 and 3-8 of that manual). The following SBus signals are not used by the controller: ♠ **[Any unused signals?]**

1.3.2 CBus signals

The CBus runs from the STEP controller to the corner box via a **CBus root cable**, and thence outwards from box to box (cf. Fig. 1.1 and Fig. 1.4) on **CBus branch cables**. Each CBus cable segment consists of a bundle of tranmission lines (cf. §1.6) carrying the following signals

CBus signals

direction	name		
bidirect.	Data[0:F]	Data[0:F]	(1.1)
outbound	Reset*	Reset*	
outbound	Sync	Sync	
outbound	Frame*	Frame*	
outbound	OH	Outbound Handshake	
inbound	IH	Inbound Handshake	
inbound	Status*	Status*	
outbound	Scan*	Scan*	

Within a box, the CBus is further distributed by the backplane to the eight modules according to the pinout of table (??), and from there to the sixteen chips of each module according to the pinout of table (??).

The various CBus lines are discussed in detail in Chapter ??.

The inner end of a CBus branch cable (the one closer to the tree root) plugs into one of a box's three CBus branch ports (that's where a new CBus branch is grafted onto the tree); the outer end plugs into a box's CBus root port (that's where the box faces towards the tree root). According to this terminology, the port through which the STEP controller drives the CBus root cable is a CBus branch port. Its pinout is the following

CBus branch port (... pins)

type	name	
B2	Data[0:F]	Data[0:F]
O2	Reset*	Reset*
O2	Sync*	Sync*
O2	Frame*	Frame*
O2	OH	Outbound Handshake
I2	IH	Inbound Handshake
I2	Status*	Status*
O2	Scan*	Scan*

In the CBus root cable (§1.3.2), the Scan* line carries master-clock (§??) rather than scan-in-progress (§??) information.

1.3.3 CBus Sync

The CBus Sync line carries, encoded as 5-bit bursts on a background of all zeros, messages used for achieving synchronization between sweeps and scans of the STEP machine (§??) and, respectively, horizontal and vertical scans of a CRT monitor (or similar external equipment). The possible messages are tabulated in Fig. 1.2.

The above messages are generated on the STEP controller as follows. Two ganged programmable counters, namely, the **horizontal counter** and the **vertical**

[1] On the SPARCstation-1, one of the three SBus slots only offers bus-slave capabilities, and cannot be used for this purpose. At any rate, this slot is typically dedicated to the graphics interface.

code	message	recipient
1 00 1 0	HMON	CRT monitor
1 01 0 0	VMON	CRT monitor
1 10 0 0	HCAM	STEP machine
1 11 1 0	VCAM	STEP machine

Figure 1.2: The above codes, transmitted on a background of all zeros, signal the appropriate times for a CRT monitor to start horizontal and vertical scans, and for the STEP machine to start sweeps and scans. Each message consist of a start bit ('1'), a two-bit field carrying the intelligence of the message, a parity bit, and a stop bit ('0').

counter (the latter is clocked by the rising edge of the former), produce the waveforms of Fig. 1.3. For each counter, "mark" and "space" lengths are independently programmable; as a consequence, one can independently set the period and the duty-cycle. HCAM, VCAM, HMON, and VMON are issued at the rising and falling edges of these waveforms, as indicated in Fig. 1.3.

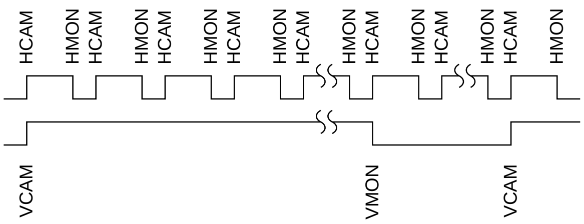


Figure 1.3: Sync messages are issued on the rising and falling edges of the waveforms generated by the horizontal and vertical counters.

The HCAM message that would occur concurrently with a VCAM is superseded by the latter; the STEPchip will infer an HCAM from the presence of a VCAM.

While the coincidence of VCAM with an HCAM is a logical necessity, the fact that also VMON exactly coincides with an HCAM (and supersedes the latter) is a nonessential feature of the present version of the STEP controller.[2] Since the STEPchip does not respond to HMON and VMON messages,[3] a STEP scan that extends past VMON and is synchronized to HCAM (cf. §??) will be suspended for one horizontal counter period when VMON occurs, since the latter masks out the former.

[2] Note that the STEP controller could be replaced by, say, a video camera as a source of sync messages, in order to have video-on-video.

[3] Each STEPchip monitors the integrity of all sync messages, by checking the parity bit, but otherwise ignores messages directed to the CRT monitor.

1.4 The step machine

The STEP machine is an assembly of boxes physically arranged and interconnected as a three-dimensional array. The three dimensions are labeled x, y, and z, and along each dimension the elements of the box array are sequentially numbered from 0 upwards. Thus, the corner box, located at the origin of the box array, has coordinates $\langle 0,0,0\rangle$, and in a system of $2\times4\times8$ boxes the box on the opposite corner would have coordinates $\langle 1,3,7\rangle$.

Along any of the three dimensions, the array need not extend for more than one box; in fact, a STEP machine may consist of just a single box.

Boxes communicate with one another by means of signals (*glue* and *CBus* signals) which travel between boxes on transmission lines (§1.6) and turn the collection of boxes into a tightly-knit whole.

The **glue signals** exchange site data between adjacent sectors in real time, according to the current data-transport specifications for each layer. Six glue ports are provided—each port corresponding to one of the six "faces" of the box, i.e., the six spatial directions in which a box can interconnect with other boxes, denoted by XM ("x minus"), XP ("x plus"), and similarly YM, YP, and ZM, ZP. Besides the data lines, each glue port includes two lines that exchange clock-phase information between adjacent boxes (§1.5).

The **CBus signals** ultimately connect the boxes to the host. Each box has one root port, facing inwards (that is, towards the host), and three branch ports, facing outwards in the positive x, y, and z directions. The root port is always used: the one on the corner box connects directly to the CBus root cable, the others, to branch cables that come from intermediate boxes. On the other hand, a branch port is used only if the box is the root of a subtree. Thus, for example, in a machine of size $2\times2\times2$ boxes (see Fig. 1.4), the corner box $\langle 0,0,0\rangle$ uses all three branch ports, box $\langle 0,0,1\rangle$ uses only x and y, boxes $\langle 0,1,0\rangle$ and $\langle 0,1,1\rangle$ use only x, and the remaining four boxes, with addresses of the form $\langle 1,y,z\rangle$, use none, since they are at the leaves of the tree.

Through the Flywheel ports, external equipment can selectively extract any of the data involved in the simulation and optionally replace them with arbitrary new data; these lines are thus used for something like "local open-brain surgery" (it is unlikely that even the most advanced user will connect external equipment to more than one box).

Through the Display port, one can send to a color monitor and view in real time the contents of a selected STEP sector.

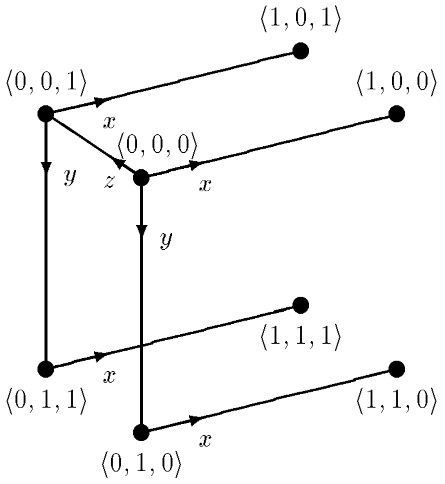


Figure 1.4: CBus port usage in a $2\times2\times2$ box machine. The dots represent boxes; the arcs, CBus hops. The x, y, z labels indicate to which branch port of a box the inner end of a hop is connected.

1.5 Clock distribution and synchronization

The current STEP implementation consists entirely of *synchronous* digital circuitry. Each box has its own *local master clock*, which drives the clock distribution tree for the entire box. The issue discussed here is the mutual synchronization of the local masters within prescribed tolerances.

Though the STEP array may extend arbitrarily in all three dimensions, its elements (the boxes, in the present context) are only *locally* interconnected. Thus, all that matters is that the clock phase difference between a box and each of its six neighbors (the ones with which the box exchanges *glue* data) be within tolerances. There will be no harm if, in moving from one box to the next along a given direction, these differences add up—leading perhaps to a cumulative phase difference of several clock cycles between widely separated boxes.

In STEP, local master clocks in adjacent boxes keep within tolerances with respect to one another by exchanging clock-phase information. This provides the feedback by which the phase of each individual clock is dynamically adjusted according to a *phase-locked loop* scheme: i.e., this phase is continually made to track (according to a suitable response function) the mean of the phases of its six neighbors.

1.6 Interbox communication

Within a box, signals run short distances; they have a stable and uniform voltage reference provided by the modules' and the motherboard's ground plane; they are shielded from one another by the close proximity of the ground plane, and from external fields by the box's enclosure. In this environment, a signal line, even though it may extend and branch out over several inches, can still be usefully treated as a voltage node.

Communication between boxes takes place in a rather different environment. The physical length of the lines from driver to receiver may run up to a few feet; glue lines run in parallel bundles of over a hundred lines each;[4] each box is the source or destination of about a thousand 30-Mbit/sec signals; and boxes will typically be tied together by a mesh of ground bars or ground strips several inches in length rather than by a continuous copper plane. For these reasons, it is desirable to have interbox signals travel on well-characterized *transmission lines*, which feature low dispersion and (when properly terminated) low reflections, good common-mode noise rejection, and good decoupling between adjacent signals.

In the current implementation, the transmission lines are realized as *unbalanced twisted pairs*. Accordingly, for each signal handled in this we provide two separate pins—a signal pin and a ground pin (cf. §3.1)—on the connectors at either end of a transmission line.

[4] Sixteen layers times eight strands gives 128 lines, to which one must add two clock-phase feedback lines.

2 STEP box

2.1 Overview

As explained in the *Architecture Reference* (see *Architecture Reference*: *block*, *sector*), when a three-dimensional torus is flattened onto three-dimensional space, eight separate *strands* of the torus are brought together. A STEP box is responsible for simulating the activity of a STEP *block*, i.e., a sector-size window cutting through all eight strands (cf. Fig. ??). The entire torus can then be simulated by arraying in a straightforward way an appropriate number of boxes in the three dimensions of ordinary physical space.[1]

Even though the canonical relationship between the eight modules contained in a box is that of strands of a flattened 3-torus, it is possible to configure a box for other relationships between modules; this can be done by some extent by purely software means, and to an ever greater extent by substituting a custom backplane for the standard one.

Besides the resources directly devoted to the simulation (site-data transport and interaction), a STEP system makes uses of other resources (e.g., clock distribution, communication with the host) that must somehow be distributed and interconnected in an indefinitely expandible way within the same geometric framework as the blocks. It is natural to lump such resources on a per-block basis and incorporate one lump in each box. Thus, while the module represents the smallest repetitive unit of STEP from a functional viewpoint, in physical terms it is the box that represents the smallest repetitive unit—from which an arbitrarily large STEP system can be built by mere juxtaposition and nearest-neighbor interconnection.

Physically, a STEP box is a roughly cubical enclosure containing a regulated power supply and a motherboard with eight slots—for up to eight STEP modules. Additional circuitry on a per-box basis is present on a separate **box interface** board (called **video board** when also populated with video interface circuitry). Fig. 2.2 shows a simplified view of a box's contents, with backplane, 8 modules, and box interface/video board. The landing places of box-to-box links between modules are shown. (In units consisting of just few boxes, only some of the X, Y and Z glue links may be realized as discrete cables, while the other links are embedded in the backplane.)

[1]Note that while the eight strands that run through the boxes are interconnected so as to yield the topology of a torus, the physical array of boxes forms an ordinary *parallelepiped*: opposite faces don't have to be brought together!

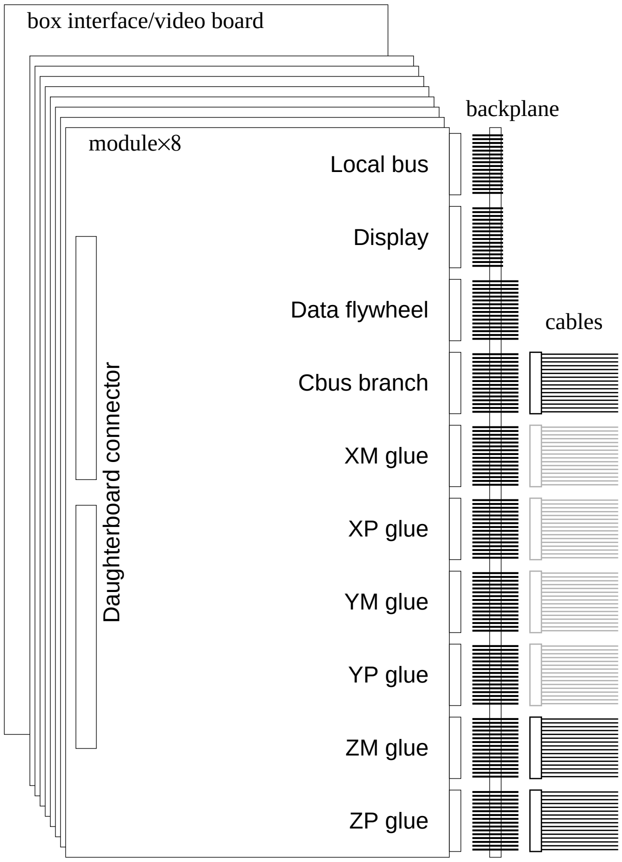


Figure 2.2: A STEP box with backplane, 8 modules, and box interface/video board. Connectors and cabling for the latter board differ from those for the modules and are not shown.

As outlined in §1.4, a box is the source or destination of three classes of signals, namely *glue* signals, which travel between boxes (§1.6), *CBus* signals, through which the boxes ultimately communicate with the host, and *data stream* signals (data flywheel and display), which allow external equipment to have real-time access to the box's internal data stream.

The signals of each class are grouped into one or more **ports** according to their geographical destination. Thus, the glue signals are divided into six glue ports according to which of the six adjacent boxes they connect to. Each port, in turn, is physically represented by one or more **connectors** into which cables may be plugged. (For convenience, also configuration and test signals are grouped together and brought to connectors.) Fig. 2.1 gives a simplified view of the backplane, showing the individual connectors and their groupings as ports. Many of the

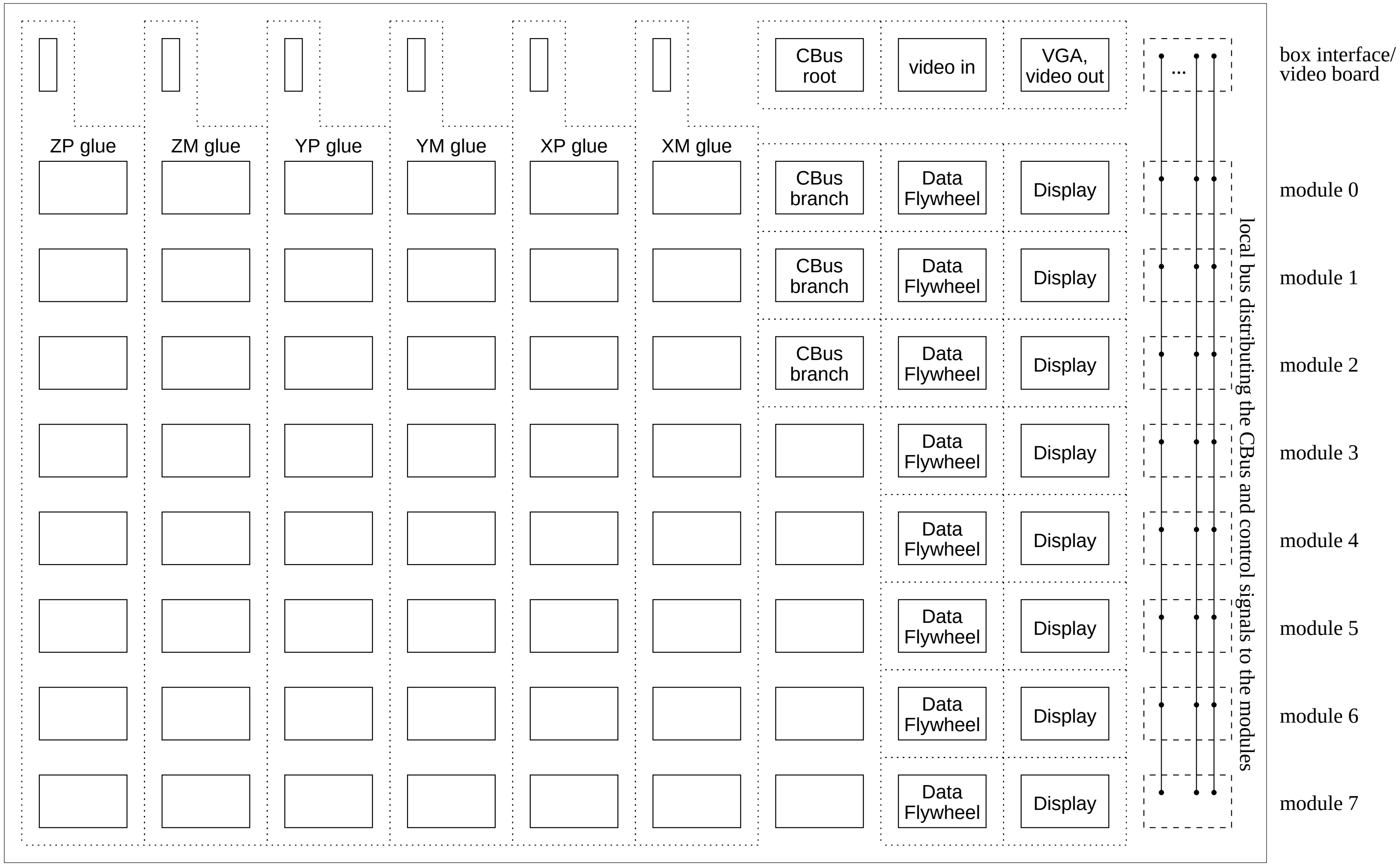


Figure 2.1: Simplified view of the STEP box backplane. Solid boxes denote connectors.

port connectors are just through-the-board extensions of connectors on the modules. In the figure, each vertical alignment corresponds to a backplane slot: one for the box interface/video board, and eight for the eight modules. Each horizontal alignment corresponds to a different type of function: local bus, CBus root and branch ports, display data, data flywheel, glue. The small connectors associated with the glue ports are for clock-phase feedback between adjacent boxes.

Communication between a STEP unit and the outside world (including the host and optional I/O devices such as VGA monitors, video cameras, and video monitors) goes through **ports** located on a **backpanel** (Appendix ??).

(A backpanel acts as an extension of a backplane. In a multibox unit, the backpanel associated with backplane 0 carries the port that communicates with the host, and is mandatory; the other backpanels are optional.)

As shown in Fig. 2.3, a backpanel port is connected to the appropriate position of the backplane by an **internal cable**; an **external cable** is used to connect the port to the appropriate I/O device.

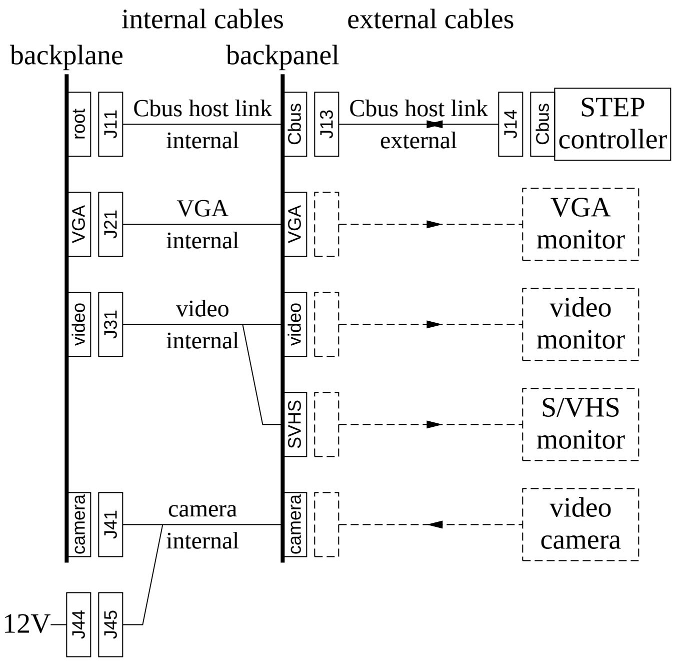


Figure 2.3: STEP-8 input/output: backpanel ports, internal cables, and external cables.

2.2 Glue ports

Each module contributes to its box six glue connectors (XM glue through ZP glue; cf. Fig. ?? and Fig. 2.2) for bidirectional communication between adjacent sectors. On the box, each *glue port* brings together a set of eight homologous connectors (one from each module), so that, for instance, the eight XM glue connectors come together as the box's XM glue port (Fig. 2.1). Each glue port has, in addition, a small connector with two signals (Clock Phase In and Clock Phase Out), handled by the motherboard circuitry, which exchange clock phase information between the local clock masters (cf. §1.5) of adjacent boxes.

Thus, between the XP glue port of a box and the XM glue port of the box next to it in the positive x direction there will run 128 bidirectional glue lines (16 layers times 8 strands) plus 2 clock-phase lines, for a total of 130 signal lines.

Note that the six physical directions XM, XP, ..., ZP are distinct from the six *logical* directions, denoted by x_-, $x_+, \ldots, z_+$, which refer to the orientation of an individual strand. Typically, the XM glue port will be configured as an x_- port by four of the modules in a box, and as an x_+ port by the remaining four (cf. Fig. ??). A more general remapping of logical ports to physical ports is possible; for instance, when the three-dimensional array of modules is configured as a pleated two-dimensional array (§??), a Z port may be used to carry y information.

On each of the external faces of the parallelepiped—where a box is not mated to an adjacent box—four of the strands double back on the other four via the glue signals. For example, on an XM face, the four XM glue signals that carry x_- information (cf. Fig. ??) for a certain layer will fold back onto the corresponding four XM glue signals that carry x_+ information. This is done by a passive shorting plug or by an internal multiplexer (cf. §??). In a similar way, a Clock Phase Out signal that emerges from an external face of the parallelepiped is looped back into the corresponding Clock Phase In pin.

2.3 Data flywheel ports

On each STEPchip, real-time ("on-the-fly") access to the internal data stream is provided by two taps on this stream, namely Data Flywheel Output and Data Flywheel Input (§??). On the module, these sixteen outputs and sixteen inputs are brought together on a Data Flywheel connector (Fig. 2.2) together with the appropriate timing signals. In turn, this connector is passed through to the motherboard as a Data flywheel port, for a total of eight such ports (Fig. 2.1).

Data flywheel access is optional, and is provided for advanced experimentation, custom applications, attachment of custom hardware, specialized display or analysis needs, and in-depth testing and diagnostics.

As implemented in the standard STEP box described here, the Data flywheel ports are meant to provide direct real-time access to simulation data from any one module of a STEP machine, or at most a few modules: it would be physically impractical to connect external equipment to all modules at once. On the other hand, using specially designed boxes, it would be feasible to use the Data Flywheel connectors to add an external "coprocessor" to each module or group of modules.

2.4 Display port

A separate Display tap (§4.2.5, §??) on each STEPchip's internal data stream extracts data intended for real-time video display. On a module, these sixteen signals are brought out to the Display bus connector (Fig. 2.2). On the backplane, the sixteen Display signals are bussed across the eight modules of a box (cf. §2.6); only one module at a time will have its tri-state bus drivers enabled.

On the video board, the Display data are optionally smapled-and-held by a frame buffer. Then they are sent through a downloadable video palette, resulting in RGB data at a rate compatible with a VGA monitor. Together with horizontal and vertical sync signals, also generated by the video board, one obtains a complete VGA output.

A parallel path with slightly different processing (which turns 60 frames/sec to 30 frames/sec with two interlaced field per frame) generates video output (composite or S/VHS) for either NTSC or PAL.

2.5 CBus ports

A box also acts as a relay for *CBus* signals (§??), receiving the CBus via a CBus root connector on the motherboard and fanning it out to up to three adjacent boxes (in the XP, YP, and ZP directions) via the CBus branch connectors (Fig. 2.1). Since CBus relaying is done by the modules of behalf of the box, there are on the motherboard eight connectors (one on each module slot) available for the CBus branch function, with the potential for fanning the CBus out to eight boxes. However, no more than three of these connectors are used, since CBus distribution is organized on a three-dimensional tree basis (cf. §1.4).

2.6 Local bus

Besides relaying the CBus to other boxes via the CBus branch ports, the motherboard also conditions and distributes the CBus signals to the eight modules. This is done through a **local bus** (see Fig. 2.1), which also distributes the clock and perform a number of miscellaneous functions.

3 STEP module

Physically, a STEP module consists of a printed circuit board which plugs into a slot of a STEP box motherboard (see §2.1) by means of a row of high-density edge connectors. The overall interconnection pattern for a box consisting of a motherboard and up to eight modules is sketched in Fig. 2.1 from the viewpoint of the motherboard, and in Fig. ?? from the viewpoint of a module.

The fundamental function of a module is to implement the data-stream loop of Fig. 3.1: the layer DRAMs are sequentially scanned, and their contents streamed through a SRAM lookup table. Scan parameters and detailed data routing are controlled by the STEPchips through the scanning/gluing circuitry of Fig. 4.3.

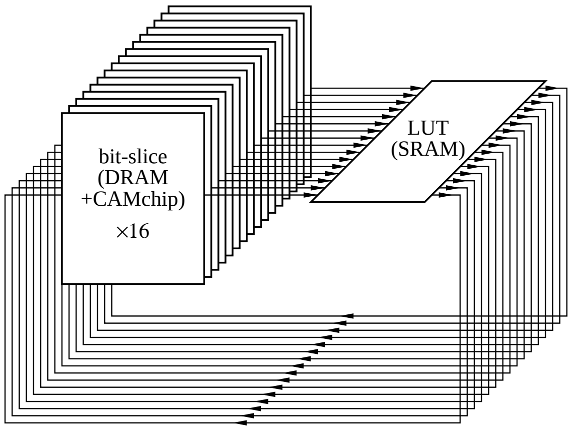


Figure 3.1: Basic form of a module's data-stream loop: the DRAMs are orderly scanned and their contents streamed through an SRAM lookup table.

3.1 Module pin types

The pins are sorted into functional groups, each of which is discussed in a separate section. In the pinout table in each section, the table header gives the total number of pins in the group and a descriptive name for the group itself (CBus interface, glue, video, etc.). For each pin, the tables give the pin type, according to table (3.1), and the pin's mnemonics followed by its full name.

Module pin names are distinguished by the prefix 'M/', to avoid confusion with STEPchip or box pin names.

As mentioned in §1.6, communication between STEP boxes (as well as between the controller and the corner box) takes place on transmission lines. The present pinout specifications are compatible with either unbalanced (e.g., coax cable) or balanced (e.g., twisted pair) transmission lines. In either case there are a *pair* of pins associated with each signal. Depending on the implementation, the two elements of the pair would be called "tip" and "ring," "core" and "shield," "signal" and "return," "plus" and "minus," or similar names. Here we'll use the noncommittal designators 'A' and 'B'.

In the pinout tables below, a signal-type designator ending with the character '2' (cf. table (3.1)) denotes a transmission line pair, listed as a single entry but with the understanding that two pins are involved. Thus, an entry of the form

02	M/FB*	M/Frame Branch*

implies the existence of two pins, namely M/Frame Branch A* and M/Frame Branch B*.

Some of the transmission lines terminate on the motherboard (CBus root lines in Fig. ??), where they are handled by discrete drivers and receivers appropriate to the type of line. Most of the transmission lines, however, terminate directly on the module boards (e.g., XM glue or CBus branch in Fig. ??) and are associated with individual STEPchip pins. If these pins have sufficiently powerful internal drivers, they will be able to drive unbalanced transmission lines directly, i.e., without the need for separate discrete drivers. In what follows, we shall assume that this is the case; however, the module pinout will not change if discrete drivers are inserted between the STEPchip pins and the transmission lines, and/or if balanced transmission lines are used.

Note that the pins of motherboard connectors XM glue through ZP glue and *Tbus branch* (cf. Fig. ??) are not connected to any signal traces on the motherboard;[1] they simply provide mechanical support and passive feedthrough from the modules to the transmission lines.

[1] In the case of unbalanced transmission lines, the B side of each pair would probably be tied to the ground plane.

type	description
I	*Input.* Input signal driving a discrete on-board buffer which fans out the signal to the sixteen STEPchips of the module.
O	*Output.* Output signal driven by the Low-Drive Multipurpose Pin of one of the sixteen STEPchips.
B	*Bidirectional.* This pin is sensed as an input, and is driven as a tri-state output.
I2	*Input pair.* Pair of pins (A and B) at the receiving end of a transmission line.
O2	*Output pair.* Pair of pins (A and B) driving a transmission line.
B2	*Bidirectional pair.* Pair of pins (A and B) which may drive or sense a transmission line.

(3.1)

3.2 CBus interface

CBus local (... pins)

type	name	
B	Local.D[0:F]	Local Data[0:F]
I	Local.Reset*	Local Reset*
I	Local.Sync*	Local Sync*
I	Local.Frame*	Local Frame*
I	Local.OH	Local Outbound Handshake
O	Local.IH	Local Inbound Handshake
O	Local.Status*	Local Status*

CBus branch (... pins)

type	name	
B2	Branch.D[0:F]	Branch Data[0:F]
O2	Branch.Reset*	Branch Reset*
O2	Branch.Sync*	Branch Sync*
O2	Branch.Frame*	Branch Frame*
O2	Branch.OH	Branch Outbound Handshake
I2	Branch.IH	Branch Inbound Handshake
I2	Branch.Status*	Branch Status*

The CBus local signals are handled by the motherboard, which fans out to the eight modules signals coming from the host (CBus root connector), and ORs signals from eight modules into a single hostward signal.

The CBus branch signals represent the place where a subtree of the CBus tree can be grafted.

3.3 Glue

Glue (192 pins)

type	name	
B 2	Glue0.G[0:F]	Glue X-Minus[0:F]
B 2	Glue1.G[0:F]	Glue X-Plus[0:F]
B 2	Glue2.G[0:F]	Glue Y-Minus[0:F]
B 2	Glue3.G[0:F]	Glue Y-Plus[0:F]
B 2	Glue4.G[0:F]	Glue Z-Minus[0:F]
B 2	Glue5.G[0:F]	Glue Z-Plus[0:F]

These are the signals that allow one to glue modules edge-to-edge so as to achieve an array of indefinite extent. Each glue signal runs point-to-point (no fan-out or fan-in) between two modules; in a large system, the two modules will reside in different boxes.

3.4 Data stream

Data flywheel (... pins)

type	name	
Out	Flywheel.O[0:F]	Flywheel Data Output[0:F]
In	Flywheel.I[0:F]	Flywheel Data Input[0:F]
Out	Flywheel.Clock	Flywheel Clock
Out	Flywheel.DSV*	Flywheel Data Stream Valid*

Raw video (... pins)

type	name	
IO	Local.Video[0:F]	Local Video[0:F]

3.5 Wired-AND communication

4 STEPchip

Fig. 4.1 shows in a very schematic way the logical organization of a STEPchip, especially in relation to the chip's immediate surroundings.

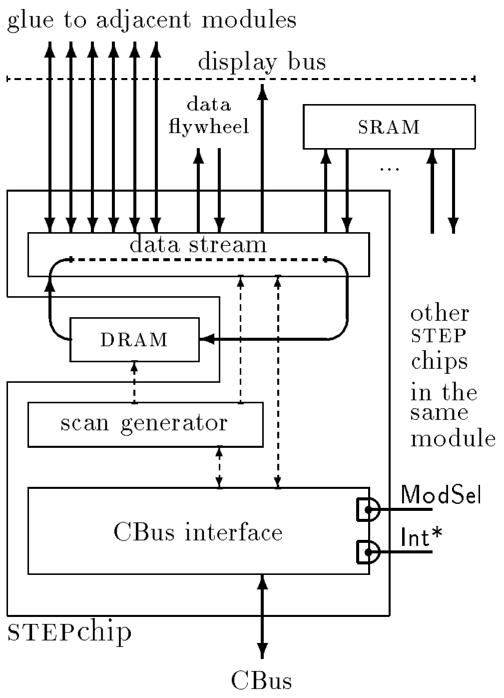


Figure 4.1: Logical organization of a STEPchip and its immediate surroundings. See text for details.

Some of the STEPchip's I/O lines (§??) connect it to other circuitry on the same module, some to homologous STEPchips on adjacent modules, some to the CBus, and some to optional external circuitry. These different types of interconnection are better understood with reference to the STEPchip's functions, as detailed below.

Though physically residing on a separate memory chip, the block labeled DRAM should logically be treated as part of the STEPchip, since all of its inputs and outputs are directly handled by the latter.

The main function of the STEPchip is to pump a data stream out of and then back into the DRAM. Along its course, the data stream may be modified by

- Data exchanges with adjacent modules, via the *glue lines*, to achieve indefinite spatial extension for the array of sites.
- Real-time data exchanges with optional external circuitry, via the *data flywheel* (§??).
- Real-time output to a raster-scan monitor, via the *display bus* (§??). All modules tristate their display lines onto a single Display bus, so that display output from any single module can be routed to the monitor.
- Data replacement by lookup in the SRAM table. The table is shared by the 16 STEPchips of a module, each of which provides one bit of the lookup *key* (table address) and and receives one bit of the corresponding *entry* (table data). Of course, this collective lookup is the main mechanism for interaction between data streams, and ultimately for the realization of the desired dynamical behavior as well as of real-time analysis.

The CBus (Chapter ??) is the exclusive channel of communication between the host and the STEP modules. All exchanges between the host and SRAM or DRAM chips are mediated by the STEPchip.

Beside being mutually coupled via the SRAM table, the 16 STEPchips of a module are also mutually coupled via a wire-AND ModSel line for the purpose of responding as a single addressable entity to CBus operations (§??), and a wire-AND Int* line for collecting interrupts from the sixteen chips and eventually merging them onto the CBus Status line.

4.1 STEPchip block diagram

The internal structure of the STEPchip shown in Fig. 4.1 is further detailed in Fig. 4.2, which is taken from the STEPchip's actual design files (this diagram does not show the I/O buffers that interface the internal signals to the outside world).

4.2 Pinout

4.2.1 Pin types

The STEPchip is packaged in a 68-pin PLCC.

The pins are sorted into functional groups, each of which is discussed in a separate section. In the pinout table in each subsection, the table header gives the total number of pins in the group and a descriptive name for the group itself (DRAM interface, CBus interface, etc.). For each pin, the tables give the pin number, its type, according to table (4.1), and then the pin's mnemonics followed by its full name.

Figure 4.2: STEPchip block diagram, taken from actual design files.

type	description
I	*Input.* This pin is sensed as an input.
O	*Output.* This pin has an output driver, which is always enabled.
T	*Tri-state.* This pin has an output driver which can be disabled.
B	*Bidirectional.* This pin is sensed as an input, and is driven as a tri-state output.
W	*Wire-AND.* This pin is driven in a way suitable for wire-ANDing.

(4.1)

4.2.2 DRAM interface

DRAM interface (19 pins)

type	name	
O	DA(10:0)	DRAM Address(10:0)
B	DD(3:0)	DRAM Data(3:0)
O	DRAS*	DRAM Row Address Strobe*
O	DCAS*	DRAM Column Address Strobe*
O	DWE*	DRAM Write Enable*
O	DOE*	DRAM Output Enable*

These are the address, data, and control signals for the DRAM chip associated with each STEPchip.

Since we use nybble-wide chips, total DRAM capacity with 11 address lines (DA(10:0)) is $4 \cdot 2^{2 \cdot 11}$, or 16 Mbits.

Once we are in fast-page or static-column mode, typically a DRAM read/write operation takes 4 clock cycles; in the first two, a nybble is read, in the second, a nybble is written.

4.2.3 SRAM interface

SRAM interface (4 pins)

type	name	
O	SA0	SRAM Address, Bank 0
O	SA1	SRAM Address, Bank 1
B	SD0	SRAM Data, Bank 0
B	SD1	SRAM Data, Bank 1

These are the address and data signals for the two SRAM lookup tables, LUT0 and LUT1. The SRAM control signals are distributed on a module-wide basis by means of four of the Multipurpose pins (§??).

4.2.4 Glue

Glue (6 pins)

type	name	
B	XM	X-Minus
B	XP	X-Plus
B	YM	Y-Minus
B	YP	Y-Plus
B	ZM	Z-Minus
B	ZP	Z-Plus

These are the signals that allow one to "glue" modules edge-to-edge so as to achieve an array of indefinite extent.

4.2.5 Data stream

Data stream (3 pins)

type	name	
O	FDO	Flywheel Data Output
I	FDI	Flywheel Data Input
T	Display	Display

At every step, the entire DRAM contents (or whatever fraction of it is actually used for the model's state variables) is serially made visible to the outside world via the Flywheel Data Output pin, before being sent to the lookup table. External circuitry can use these data for analysis purposes; and can replace any of these data bits with externally supplied ones. On every clock cycle in which Enable* is asserted, the internal data bit is replaced with one that is externally supplied via the Flywheel Data Input pin. The timing of this injection—either one or four clock cycles[1] later than when the original bit appeared on the FDO pin—allows for the original bit to be taken into consideration when generating the replacement bit.

4.2.6 CBus interface

CBus (5 pins)

type	name	
I	TF*	CBus Frame*
I	TOT	CBus Outbound Tag
I	TIT	CBus Inbound Tag
B	TD	CBus Data
B	TDB	CBus Data Branch
I	Sync*	Sync*
I	Reset*	Reset*

The following output signals are produced on a per-module rather than per-chip bases, and are handled by a multipurpose pin: CBus Inbound Tag Local (TITL), CBus Outbound Tag Branch (TOTB), CBus Frame Branch* (TFB*), CBus Sync Branch* (TSB*),

[1] Depending on the setting of the FPL bit in the Hardware Environment Register (§??).

Other CBus related signals go through significant on-board conditioning, and are listed in §4.2.7.

4.2.7 Miscellaneous

Miscellaneous (9 pins)

type	name	
I	Clock	Clock
O	MP	Multipurpose
W	Int*	Interrupt*
W	ModSel	Module Select

The Interrupt* pins from the 16 chips of a module are wire-ORed together. The resulting signal is merged with Scan-in-Progress information and output on one of the 16 Multipurpose pins (§??), to be used, after proper conditioning, as the CBus Status* signal contributed by the module.

The Module Select pins from the 16 chips of a module are wire-ANDed together, to allow the individual chips to contribute to a module-wide AND and sense its result, as explained in §??. As each pin will be actively driven immediately before the clock cycle in which the AND operation will be performed, the pull-up resistor connected to this wire is only for long-term bleeding purposes,

Reset* and Clock are derived from the corresponding CBus signals. Redistribution of these signals to the tree branches is handled by on-board discrete circuitry rahter than by the STEPchip.

4.2.8 Spares, reserved

Reserved (2 pins)

type	name	
I	Esc0	Escape0
I	Esc1	Escape1

The behavior of the STEPchip is characterized only when the two pins Escape0 and Escape1 are tied to logical 0.

In some versions of the die used for the STEPchip, for fabrication convenience some of the spare die area may have been used for other chips. A scheme presently being considered for this multiplexing of functions on the same die is the following

Esc0	Esc1	chip selected
0	0	STEPchip
0	1	CTLSV chip
1	0	FSM chip
1	1	DATAREGS chip

. (4.2)

The STEP controller will use one FSM, two STLSV, and two DATAREGS chips.

4.3 Scanning

The STEPchip's address-generation and data-stream-management functions, already sketched in Fig. 3.1 and Fig. 4.1, are illustrated in more detail in Fig. 4.3.

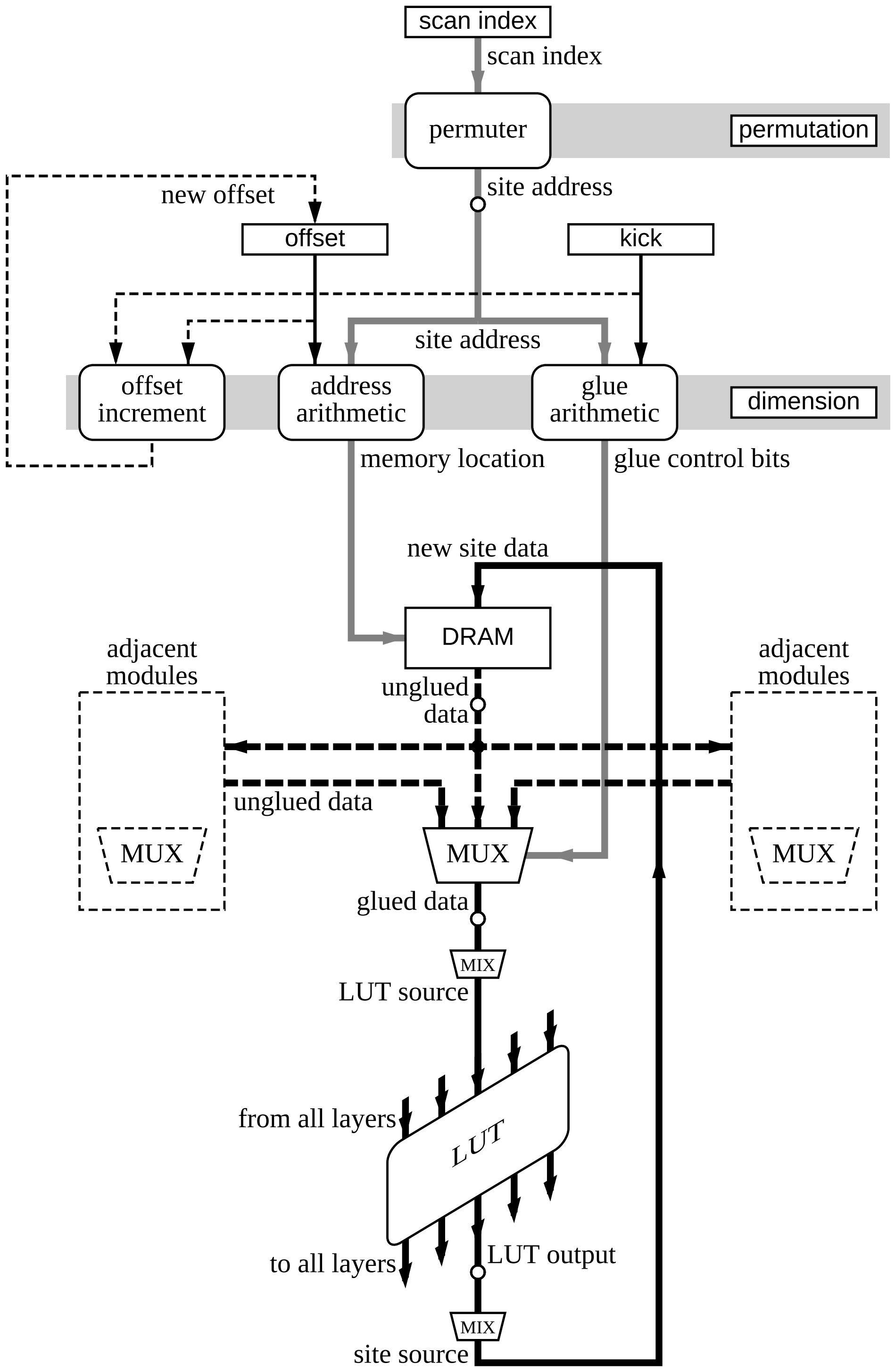


Figure 4.3: Block diagram of the basic scanning/gluing/updating circuit. The little white circles denote data taps from which signals are made available to additional circuitry not shown in the figure. Conversely, each of the little boxes labeled MIX indicates a place where several signals, besides the main one shown in the figure, are available for use. In particular, the data path from *glued data* to *site source* through the LUT can be complemented by one going through an external coprocessor; the latter may span several modules.

5 Video output

5.1 Color subcarrier frequency

The NTSC color subcarrier frequency, f_c, is specified as

$$f_c = F_c \equiv 3,579,545 \text{ Hz},$$

or approximately 3.58 MHz. The corresponding period, t_c, is

$$t_c = T_c \equiv 279.3651 \text{ nsec},$$

or approximately 280 nsec. Here is a table of the color subcarrier's first few multiples and submultiples.

NTSC color subcarrier multiples		
k	kt_c (nsec)	f_c/k (MHz)
1/4	69.841 2	14.318 180 000
1/3	93.121 7	10.738 635 000
1/2	139.682 5	7.159 090 000
1	279.365 1	3.579 545 000
2	558.730 2	1.789 772 500
3	838.095 3	0.894 886 250
4	1 117.460 5	0.447 443, 125

Commercial color crystals have a typical tolerance of a few ppm ("parts per million"). The phase of the color oscillator in a color monitor is designed to be reset to that of the broadcast color subcarrier at the beginning of every scan line. Because of this design, it is reasonable to presume that the monitor will track a broadcast color subcarrier that deviates a few tens of ppm (or perhaps even hundreds of ppm) from the crystal's nominal frequency.

5.2 Horizontal scan frequency

An NTSC *frame* consists of two interlaced *fields* of

$$N_v = 262\frac{1}{2}$$

lines each, for a total of 525 lines per frame. Fields are produced at a nominal rate of 60/sec. Less than ideal regulation of a TV set's power supply leads to "hum bars" on the screen.[1] leading to a nominal horizontal scanning frequency of 15,750 kHz. The actual horizontal frequency for *color* broadcasting, as specified by the NTSC standard, is slightly lower, namely,

$$F_h \equiv 15,734.264 \text{ kHz}, \tag{5.1}$$

corresponding to a horizontal period

$$T_h = 63,555.563 \text{ usec}.$$

NOTE: With these choices for t_c and t_h, there are exactly

$$N_c \equiv 227\frac{1}{2} \tag{5.2}$$

color subcarrier cycles in one horizontal line. Thus the phase of the color-subcarrier reference (as embodied, for instance, in the color burst) is made coherent with horizontal-scan timing. Specifically, the color phase alternates by 180° on successive scan lines. This coherent alternation has some desirable consequences. First of all, it virtually cancels out on the screen the intensity ripple due to residual color subcarrier in the luminance channel. Moreover, it nulls the chrominance power spectrum at multiples of f_h, whereas the luminance spectrum has a maximum at those frequencies; the result is an interleaving of the luminance and chrominance "combs." Neither effect is strictly essential to the working of the NTSC color mechanism, and in fact the Bt858 chip (§??) allows one to set n_c to values different than N_c.

5.3 Square pixels and all that

The NTSC TV image has a nominal aspect ratio of 3:4. Without going into details, it takes approximately 80 usec to horizontally traverse on the CRT screen a distance equal to the spacing between lines, corresponding to a pixel rate of approximately 12.6 MHz. A mode of operation where discrete pixels are produced at such rate is called *square pixel*.

Whether or not one seeks to have square pixels, it is of course desirable to have the pixel time, t_p, be a submultiple (by an integer n_p) of the horizontal period, so as to maintain coherency betweend pixel rate and color subcarrier frequency on the time-scale of one scan line. For square pixels, an appropriate choice for n_p is

$$n_p = n_\# \equiv 780,$$

which yields

$$t_\# = T_h/n_\# = 81.4814 \text{ nsec}$$

[1] A monitor's poorly regulated power supplies results in "hum bars" at line frequency (60 Hz in the U.S.) on the screen. The idea behind a nominal vertical rate of 60 fields/sec is to match the line frequency, so as to keep the hum bars stationary and thus much less noticeable.

The NTSC choice of $f_h = 15,734.264$ in color mode (see (5.1)) yields a field rate of 59.940053/sec. The ensuing extremely slow rolling of the hum bars across the screen is practically unnoticeable.

With modern, well-regulated monitors, hum bars are not an issue, and designers feel free to drop the 60 Hz constraint in favor of other design goals. For instance, the IBM-PC monochrome mode uses 70 Hz to reduce visual flicker, and the IBM-PC CGA mode uses 60 Hz even in Europe, where the line frequency is 50 Hz.

and

$$f_{\#} = 12.272727 \text{ MHz}.$$

(In other words, the horizontal period t_h will be equal to 227 color subcarrier periods *and* to 780 pixel periods.) We shall call this choice of parameters *square pixel* NTSC.

In square-pixel NTSC, color and pixel clock rates are commensurable over a scan line; however, they are essentially incommensurable in the short term (one has 3.428 571 pixels per color cycle). This may lead to Moiré patterns and similar "beats" between the two rates; specifically, with an image consisting of a uniform pattern of rather fine vertical stripes, one may end up seeing, as an artifact, some coarser-pitched vertical banding.

5.4 The Bt858

In the Brooktree Bt858 chip, the frequency f_c of the color subcarrier ("FSK" in the specs) is derived from the pixel clock frequency f_p ("FCLK") by means of a digital rate multiplier. The actual frequency ratio f_c/f_p is determined by the contents of three registers, namely, the horizontal pixel count n_p ("HCOUNT"), a coarse-tuning parameters $p1$ ("P1"), and a fine-tuning parameter $p2$ ("P2"), as follows

$$\text{NTSC}: \quad \frac{f_c}{f_p} = \frac{1}{2048}(p_1 + \frac{1}{4n_p}p2)\,, \tag{5.3}$$

$$\text{PAL}: \quad \frac{f_c}{f_p} = \frac{1}{2048}(p_1 + \frac{67/625}{4n_p}p2)\,. \tag{5.4}$$

With the current version of the CAM-8 controller (the card that sits on the SBus), the CAM-8 clock is the same as the SBus clock; with the current version of the box interface (the card in slot 0), the pixel clock is 1/2 the CAM-8 clock. Thus, at present,

$$f_p = \frac{1}{2} f_{\text{SBus}}.$$

♠ **[to be completed]**

5.5 Bt478/Bt858 programming

Programming sequence for the Bt858 for STEP NTSC output. The same programming sequence works for the Bt478, which simply ignores command register data (code 110).

```
INITIALIZATION SEQUENCE

RS2,1,0 D7..0           operation

000     0000 0000       CR0 address (in register space)

110     0101 00..       CR0             8-bit pseudocolor (green)
110     0000 1000       CR1             8-(not 6-) color; use RAM tabl
110     000. 0000       CR2             operate; normal video
110     0000 0000       CR3             NTSC master mode 0
110     0000 0000       CR4             miscellanea
110     0000 000.       reserved
110                     P1 lo           see program p1p2.c
110     ....            P1 hi
110                     P2 lo
110     ....            P2 hi
110     xxxx xxxx       Fsc phase lo (not needed)
110     .... .xxx       Fsc phase hi (not needed)
110     1010 0011       HCOUNT lo       794 = 0x1A3
110     .... 0001       HCOUNT hi
110     xxxx xxxx       color key (not needed)
110     xxxx xxxx       color mask  (not needed)

010     1111 1111       write all 1s to the pixel read mask register


PALETTE DOWNLOADING SEQUENCE (for each RAM location to be written)

RS2,1,0 D7..0           operation

000     aaaa aaaa       desired RAM address (in palette space)

001     rrrr rrrr       red data
001     bbbb bbbb       blue data
001     gggg gggg       green data
```

Figure 5.1: Bt858 initialization sequence.

6 Video input

6.1 Camera

Currently, the STEP hardware and software support as a video input device the PANASONIC GP-MF702 camera—a solid state B/W camera having a resolution and a scanning rate (480 lines at 60 frames/sec, noninterlaced) that match those of the VGA monitor. Both camera and VGA monitor can be attended to in real time by one STEP module operating on a 512×512 array. The camera is operated in external-sync mode; the sync and pixel-clock signals are supplied by the STEP box.

6.1.1 Camera switch settings

In what follows, OLD means ordinary videocamera mode, with internally generated sync, as shipped; VGA means our target mode, with noninterlaced 60 Hz frames (Double Speed Full Line Non-Interface) and external sinc and pixel clock. For the switches, 0 means 'off', 1 means 'on'.

switch	OLD	VGA	notes
Ext-7	0	1	Scanning mode
Ext-8	1	1	
Sw101-5	0	1	Double speed
Sw201	forw	back	
Sw301	up	up	Check 'up' is right
Sw303	1	0	Composite
Sw304	forw	back	AC/DC for video out
Sw101-7	0	1	Pixel clock in (722×fH)

A Card cage and enclosure