Appendix X
Kernel Block Layouts.

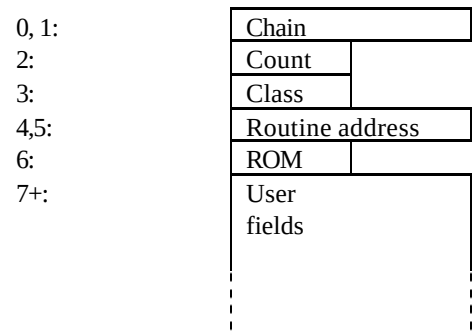
The user provides a number of blocks to the Kernel for various purposes. The layouts of these blocks are described below, mainly for the interest of the user. There are very few occasions when the user is allowed to write to one of these blocks. Routines are provided to perform most actions that the user could wish to perform (see KL INIT EVENT, KL ADD TICKER, KL NEW FRAME FLY, KL NEW FAST TICKER and KL DISARM EVENT). These routines set values into the block from registers. The user should not write to the blocks, except as noted below.

All the following blocks must lie in the central 32K of RAM (otherwise the Kernel will be unable to access them).

On the CPC6128 the user is advised to keep these blocks in RAM block 2 if any bank switching is being performed (see section 2.5).

a. Event Blocks.

See section 12 for a general discussion of events and event blocks. An event block is laid out as follows:



Chain is a system pointer which must never be written to by the user. It is used to store events on the various event queues.

Class records the type of the event. It should not be written to by the user.

Bit 0:	1 = Near address, 0 = Far address.
Bits 1..4:	Synchronous event priority.
Bit 5:	Must be zero.
Bit 6:	1 = Express event, 0 = Normal event.
Bit 7:	1 = Asynchronous event, 0 = Synchronous event.

Note that many system queues are kept in priority order and so the block must be requeued if the priority is changed, it is not sufficient merely to change the priority in the event block.

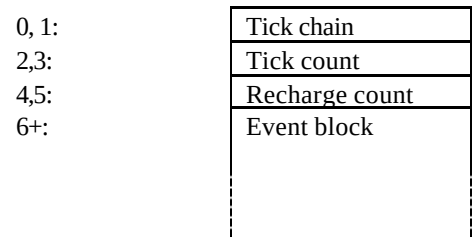
Count is the event count - a record of how many kicks are waiting to be processed or whether the event is disabled. See section 12.2 for a full discussion of the use of the event count.

Routine address and ROM make up the far address of the event routine. If the near address bit in the event class is true then the event routine is at a near address - the ROM select byte (byte 6) is ignored and the event routine is called directly. If the near address bit is false then the event routine is at far address - bytes 4,5 and 6 make up the far address to call to run the event routine. The user may write to the routine address and ROM fields (and to the near address bit in the class byte as well) provided that the operation is performed indivisibly (i.e. interrupts should be disabled).

The user fields are optional. They may be used to provide a data area specific to the event block so that a single event routine may be shared between a number of different event blocks (the event routine is passed the address of the user fields).

b. Ticker Queue Blocks.

See section 11 for a general discussion of ticker interrupts and the ticker queue.
A ticker queue block is laid out as follows:



Tick chain is a system pointer which must never be written to by the user. It is used to store the block on the ticker queue.

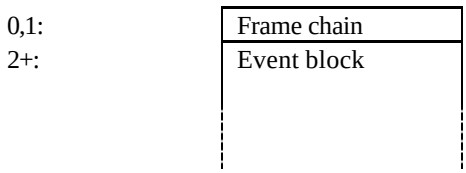
Tick count is a count of the number of ticks before the next kick occurs. A tick count of zero means that the tick block is dormant and will not generate any kicks. (Ideally a dormant block should be removed from the ticker queue to avoid wasting time). The user may write to this field if required providing this is done indivisibly.

Recharge count is the value that the tick count is set to after each kick. If the recharge count is zero then the ticker block will become dormant after generating one kick. The user may write to this field if required providing this is done indivisibly.

Event block is a standard event block as described in section (a) above.

c. Frame Flyback Queue Blocks.

See section 11 for a general discussion of frame flyback interrupts and the frame flyback queue. A frame flyback queue block is laid out as follows:

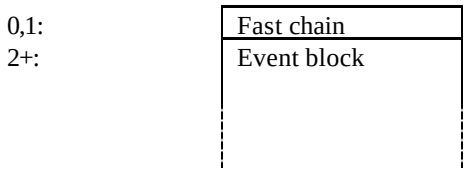


Frame chain is a system pointer which must never be written to by the user. It is used to store the block on the frame flyback queue.

Event block is a standard event block as described in section (a) above.

d. Fast Ticker Queue Blocks.

See section 11 for a general discussion of fast ticker interrupts and the fast ticker queue. A fast ticker queue block is laid out as follows:



Fast chain is a system pointer which must never be written to by the user. It is used to store the block on the fast ticker queue.

Event block is a standard event block as described in section (a) above.