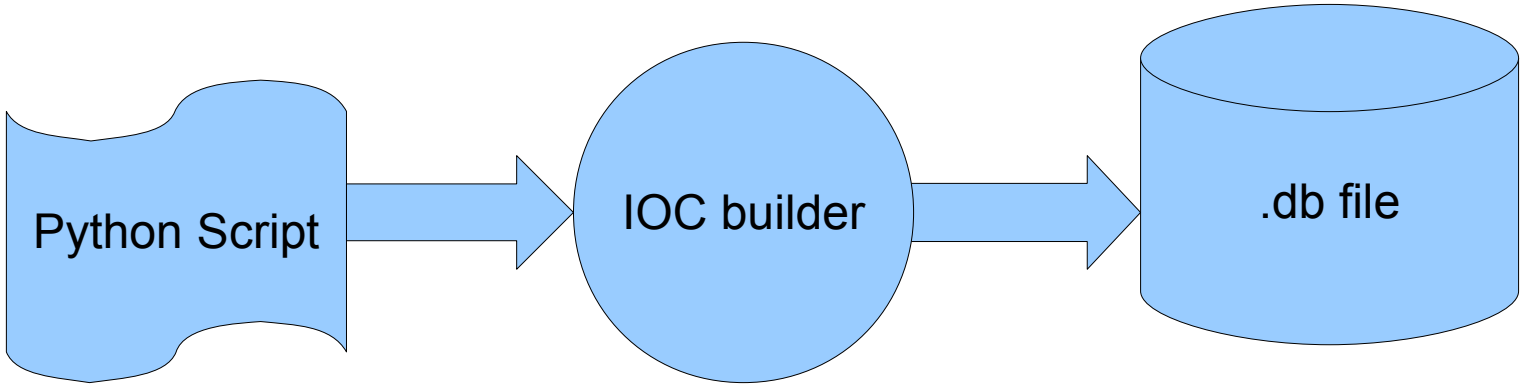
The EPICS IOC Builder

Python scripts for generating IOCs.

Michael Abbott, Diamond Light Source

Original motivation: a set of classes for generating .db files from Python



For example (part of Libera build)

```
# Imports and common definitions omitted

def SlowAcquisition():
    SetChannelName('SA')
    power = aIn('POWER', -80, 10, 1e-6, 'dBm', 3,
        DESC = 'Absolute input power')
    current = aIn('CURRENT', 0, 500, 1e-5, 'mA', 3,
        DESC = 'SA input current')
    Trigger(False, ABCD_() + XYQS_(4) + [power, current] + MaxAdc())
    UnsetChannelName()

# Other channel definitions omitted

SlowAcquisition()
# ...

WriteRecords('libera.db')
```

generates .db file

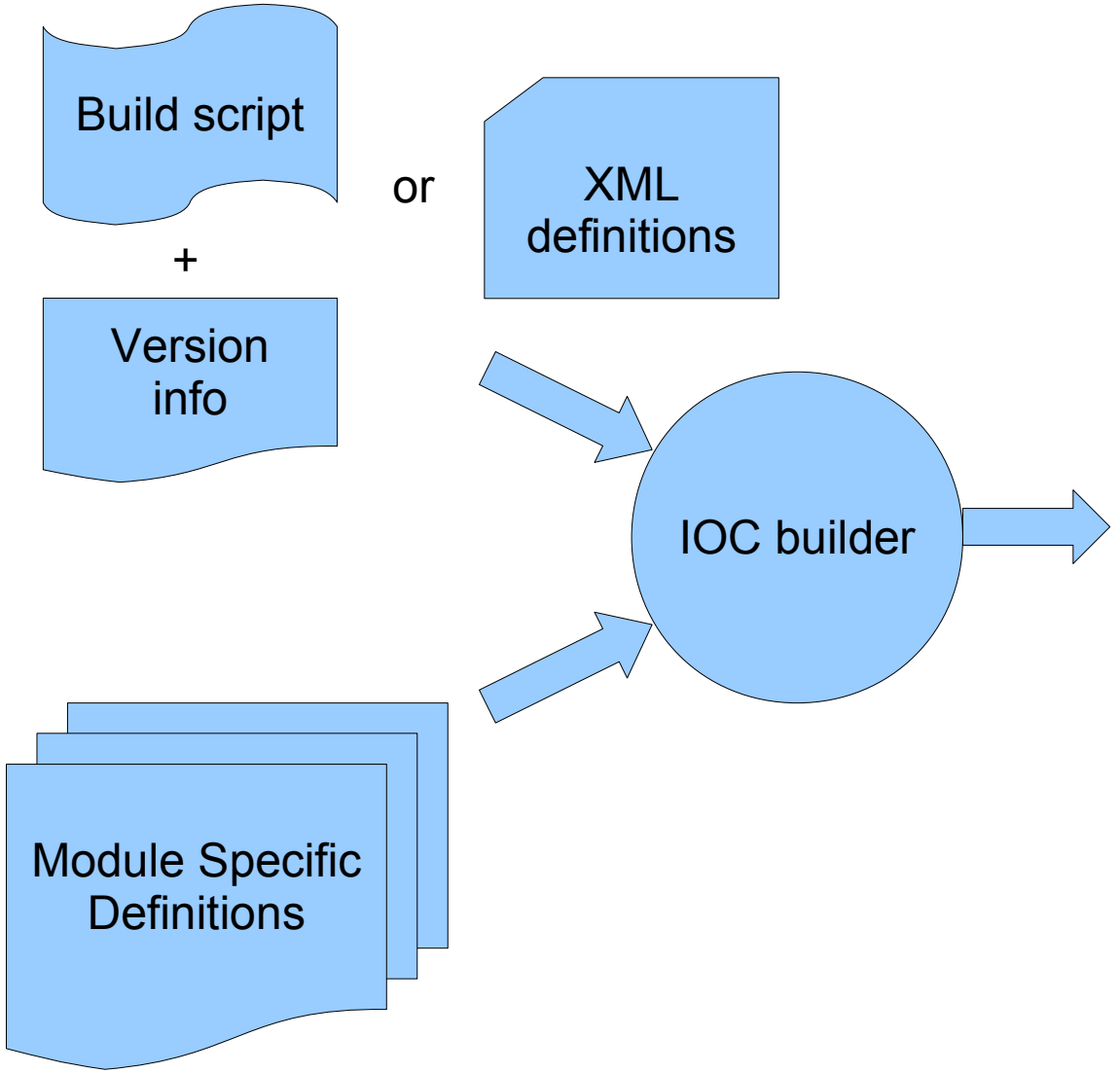
```
# This file was automatically generated on
# Thu 01 Oct 2009 14:41:46 BST from source:
# /home/mga83/epics/Libera/liberaApp/Db/libera.py
#
# *** Please do not edit this file:
# edit the source file instead. ***
#

record(longin, "$(DEVICE):SA:A")
{
    field(DESC, "SA button A intensity")
    field(DTYP, "Libera")
    field(HOPR, "2147483647")
    field(INP,  "@SA:A")
    field(LOPR, "0")
    field(MDEL, "-1")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(longin, "$(DEVICE):SA:B")
{
    field(DESC, "SA button B intensity")
    field(DTYP, "Libera")
    field(HOPR, "2147483647")
    field(INP,  "@SA:B")
    field(LOPR, "0")
    field(MDEL, "-1")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(longin, "$(DEVICE):SA:C")
{
    field(DESC, "SA button C intensity")
    field(DTYP, "Libera")
    field(HOPR, "2147483647")
    field(INP,  "@SA:C")
    field(LOPR, "0")
    field(MDEL, "-1")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(ai, "$(DEVICE):SA:CURRENT")
{
    field(DESC, "SA input current")
    field(DTYP, "Libera")
    field(EGU,  "mA")
    field(EGUF, "500")
    field(EGUL, "0")
    field(EOFF, "0")
    field(ESLO, "1e-05")
    field(HOPR, "500")
    field(INP,  "@SA:CURRENT")
    field(LINR, "LINEAR")
    field(LOPR, "0")
    field(MDEL, "-1")
    field(PREC, "3")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(longin, "$(DEVICE):SA:D")
{
    field(DESC, "SA button D intensity")
    field(DTYP, "Libera")
    field(HOPR, "2147483647")
    field(INP,  "@SA:D")
    field(LOPR, "0")
    field(MDEL, "-1")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(longout, "$(DEVICE):SA:DONE")
{
    field(DESC, "Report trigger done")
    field(DTYP, "Libera")
    field(MDEL, "-1")
    field(OUT,  "@SA:DONE")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(longin, "$(DEVICE):SA:MAXADC")
{
    field(DESC, "Maximum ADC reading")
    field(DTYP, "Libera")
    field(HHSV, "MAJOR")
    field(HIGH, "23197")
    field(HIHI, "27255")
    field(HOPR, "32768")
    field(HSV,  "MINOR")
    field(INP,  "@SA:MAXADC")
    field(LOPR, "0")
    field(MDEL, "-1")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(calc, "$(DEVICE):SA:MAXADC_PC")
{
    field(CALC, "A/B")
    field(DESC, "Maximum ADC reading (%)")
    field(EGU,  "%")
    field(HOPR, "100")
    field(INPA, "$(DEVICE):SA:MAXADC MS")
    field(INPB, "327.68")
    field(LOPR, "0")
    field(PREC, "1")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(ai, "$(DEVICE):SA:POWER")
{
    field(DESC, "Absolute input power")
    field(DTYP, "Libera")
    field(EGU,  "dBm")
    field(EGUF, "10")
    field(EGUL, "-80")
    field(EOFF, "0")
    field(ESLO, "1e-06")
    field(HOPR, "10")
    field(INP,  "@SA:POWER")
    field(LINR, "LINEAR")
    field(LOPR, "-80")
    field(MDEL, "-1")
    field(PREC, "3")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(ai, "$(DEVICE):SA:Q")
{
    field(DESC, "SA relative skew")
    field(DTYP, "Libera")
    field(EGU,  "")
    field(EGUF, "1")
    field(EGUL, "-1")
    field(EOFF, "0")
    field(ESLO, "1e-08")
    field(HOPR, "1")
    field(INP,  "@SA:Q")
    field(LINR, "LINEAR")
    field(LOPR, "-1")
    field(MDEL, "-1")
    field(PREC, "4")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(longin, "$(DEVICE):SA:S")
{
    field(DESC, "SA total button intensity")
    field(DTYP, "Libera")
    field(INP,  "@SA:S")
    field(MDEL, "-1")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(bi, "$(DEVICE):SA:TRIG")
{
    field(DESC, "Trigger processing")
    field(DTYP, "Libera")
    field(FLNK, "$(DEVICE):SA:TRIGFAN")
    field(INP,  "@SA:TRIG")
    field(SCAN, "I/O Intr")
    field(TSE,  "-2")
}

record(fanout, "$(DEVICE):SA:TRIGFAN")
{
    field(LNK1, "$(DEVICE):SA:A")
    field(LNK2, "$(DEVICE):SA:B")
    field(LNK3, "$(DEVICE):SA:C")
    field(LNK4, "$(DEVICE):SA:D")
    field(LNK5, "$(DEVICE):SA:X")
    field(LNK6, "$(DEVICE):SA:TRIGFAN1")
    field(SELM, "All")
}

record(fanout, "$(DEVICE):SA:TRIGFAN1")
{
    field(LNK1, "$(DEVICE):SA:Y")
    field(LNK2, "$(DEVICE):SA:Q")
    field(LNK3, "$(DEVICE):SA:S")
    field(LNK4, "$(DEVICE):SA:POWER")
    field(LNK5, "$(DEVICE):SA:CURRENT")
    field(LNK6, "$(DEVICE):SA:TRIGFAN2")
    field(SCAN, "Passive")
    field(SELM, "All")
}

record(fanout, "$(DEVICE):SA:TRIGFAN2")
{
    field(LNK1, "$(DEVICE):SA:MAXADC")
    field(LNK2, "$(DEVICE):SA:MAXADC_PC")
    field(LNK3, "$(DEVICE):SA:DONE")
    field(SCAN, "Passive")
    field(SELM, "All")
}

record(ai, "$(DEVICE):SA:X")
{
    field(DESC, "SA X position")
    field(DTYP, "Libera")
    field(EGU,  "mm")
    field(EGUF, "10")
    field(EGUL, "-10")
    field(EOFF, "0")
    field(ESLO, "1e-06")
    field(HOPR, "10")
    field(INP,  "@SA:X")
    field(LINR, "LINEAR")
    field(LOPR, "-10")
    field(MDEL, "-1")
    field(PREC, "4")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}

record(ai, "$(DEVICE):SA:Y")
{
    field(DESC, "SA Y position")
    field(DTYP, "Libera")
    field(EGU,  "mm")
    field(EGUF, "10")
    field(EGUL, "-10")
    field(EOFF, "0")
    field(ESLO, "1e-06")
    field(HOPR, "10")
    field(INP,  "@SA:Y")
    field(LINR, "LINEAR")
    field(LOPR, "-10")
    field(MDEL, "-1")
    field(PREC, "4")
    field(TSEL, "$(DEVICE):SA:TRIG.TIME")
}
```

Extending to building entire IOC

An IOC consists of many components, much of it either boilerplate or else module specific:

- Makefiles
- configure/RELEASE, more specification module version management
- Template substitutions: can be very repetitive
- Startup scripts, again often repetitive, and encapsulate a lot of module specific knowledge

IOC Builder building an IOC



Complete IOC:

- complete IOC directory
- configure/RELEASE, reflecting version info
- template substitutions
- generated .db files
- startup scripts

Just run make and it's ready to run.

Working Example

```
import iocbuilder                    # Boilerplate imports.
iocbuilder.ConfigureIOC()            # Need to configure builder before imports
from iocbuilder import *

# Define module versions
ModuleVersion('ipac',           '2-8dls4-5')
ModuleVersion('Hy8515',         '3-9')
ModuleVersion('asyn',           '4-10')
ModuleVersion('streamDevice',   '2-4dls2-1')
ModuleVersion('newstep',        '1-4',  load_path = '.')

# Define hardware resources
card4 = modules.ipac.Hy8002(4)      # VME IP carrier card in VME slot 4
serial = card4.Hy8515(0)            # Serial IP card in slot A

SetDomain('TEST', 'TS')             # Establishing record naming convention,
SetDevice('DEV', 1)                 #  uses default configuration setup.

# Here we use the hardware to build the software resources we need
for ch in range(2):
    asyn = modules.asyn.AsynSerial(serial.channel(ch))    # Asyn serial connection
    modules.newstep.NSC200(
        M = GetDevice(), P = '', PORT = asyn.DeviceName(), CH = ch)

# Finally emit the IOC we've assembled into memory.
WriteNamedIoc('iocs', 'TEST-IOC')
```

Working Example

```
import iocbuilder                    # Boilerplate imports.
iocbuilder.ConfigureIOC()            # Need to configure builder before imports
from iocbuilder import *

# Define module versions
ModuleVersion('ipac',           '2-8dls4-5')
ModuleVersion('Hy8515',         '3-9')
ModuleVersion('asyn',           '4-10')
ModuleVersion('streamDevice',   '2-4dls2-1')
ModuleVersion('newstep',        '1-4',  load_path = '.')

# Define hardware res
card4 = modules.ipac.
serial = card4.Hy8515

SetDomain('TEST', 'TS
SetDevice('DEV', 1)

# Here we use the har
for ch in range(2):
    asyn = modules.as                                          nection
    modules.newstep.N
        M = GetDevice

# Finally emit the IO
WriteNamedIoc('iocs',
```

Here we specify all the support modules which might be used in this IOC, and which release version of each module to use.

As each `ModuleVersion()` call is processed the builder definitions associated with the support module are automatically loaded into memory, as attributes of `modules.<module-name>`

Work

```
import iocbuilder
iocbuilder.ConfigureIOC()
from iocbuilder import *

# Define module versions
ModuleVersion('ipac',           '
ModuleVersion('Hy8515',         '
ModuleVersion('asyn',           '
ModuleVersion('streamDevice',   '
ModuleVersion('newstep',        '

# Define hardware resources
card4 = modules.ipac.Hy8002(4)      # VME IP carrier card in VME slot 4
serial = card4.Hy8515(0)            # Serial IP card in slot A

SetDomain('TEST', 'TS')             # Establishing record naming convention,
SetDevice('DEV', 1)                 #  uses default configuration setup.

# Here we use the hardware to build the software resources we need
for ch in range(2):
    asyn = modules.asyn.AsynSerial(serial.channel(ch))    # Asyn serial connection
    modules.newstep.NSC200(
        M = GetDevice(), P = '', PORT = asyn.DeviceName(), CH = ch)

# Finally emit the IOC we've assembled into memory.
WriteNamedIoc('iocs', 'TEST-IOC')
```

Hardware resources are declared as instances of appropriate wrappers for each class of hardware device.

In this case we have a VME carrier card with a single IP module loaded into its slot A.

These lines ensure that the hardware is correctly initialised in the IOC's startup script.

Working Example

```
import iocbuilder                       # Boilerplate imports.
iocbuilder.ConfigureIOC()               # Need to configure builder before imports
from iocbuilder import *

# Define module versions
ModuleVersion('ipac',
ModuleVersion('Hy8515',
ModuleVersion('asyn',
ModuleVersion('streamDevice',
ModuleVersion('newstep',

# Define hardware resources
card4 = modules.ipac.Hy8002(4)
serial = card4.Hy8515(0)

SetDomain('TEST', 'TS')
SetDevice('DEV', 1)

# Here we use the hardware to
for ch in range(2):
    asyn = modules.asyn.AsynSerial(serial.channel(ch))    # Asyn serial connection
    modules.newstep.NSC200(
        M = GetDevice(), P = '', PORT = asyn.DeviceName(), CH = ch)

# Finally emit the IOC we've assembled into memory.
WriteNamedIoc('iocs', 'TEST-IOC')
```

Here the software components of the IOC are declared. In this case we use two serial ports from the IP module, wrapping each one for use by the streamDevice serial protocol module.

The `newstep.NSC200()` call generates a template substitution which uses streamDevice to talk to a Newport motor controller.

This generates:

```
configure/RULES.ioc
configure/RULES_DIRS
configure/CONFIG_APP
configure/Makefile
configure/RULES
configure/RULES_TOP
configure/CONFIG
configure/RELEASE
TEST-IOCApp/Db/TEST-IOC.expanded.substitutions
TEST-IOCApp/Db/Makefile
TEST-IOCApp/src/Makefile
TEST-IOCApp/data/Makefile
iocBoot/iocTEST-IOC/stTEST-IOC.cmd
iocBoot/iocTEST-IOC/Makefile
Makefile
```

This generates:

```
configure/RULES.ioc
configure/RULES_DIRS
configure/CONFIG_APP
configure/Makefile
configure/RULES
configure/RULES_TOP
configure/CONFIG
configure/RELEASE
TEST-IOCApp/Db/TEST-IOC.expanded.substitutions
TEST-IOC
TEST-IOC
TEST-IOC
iocBoot/
iocBoot/
Makefile
```

```
# This file was automatically generated on
# Wed 07 Oct 2009 07:09:12 BST from source:
# /home/mga83/working/ICALEPCS2009/talks/example.py
#
# *** Please do not edit this file:
# edit the source file instead. ***
#
EPICS_BASE = /dls_sw/epics/R3.14.8.2/base
HY8515 = /dls_sw/prod/R3.14.8.2/support/Hy8515/3-9
ASYN = /dls_sw/prod/R3.14.8.2/support/asyn/4-10
IPAC = /dls_sw/prod/R3.14.8.2/support/ipac/2-8dls4-5
NEWSTEP = /dls_sw/prod/R3.14.8.2/support/newstep/1-4
STREAMDEVICE = /dls_sw/prod/R3.14.8.2/support/streamDevice/2-4dls2-1
```

This generates:

```
configure/RULES.ioc
configure/RULES_DIRS
configure/CONFIG_APP
configure/Makefile
configure/RULES
configure/RULES_TOP
configure/CONFIG
configure/RELEASE
TEST-IOCApp/Db/TEST-IOC.expanded.substitutions
TEST-IOCApp/Db/Makefile
TEST-IOCApp/src/Makefile
TEST-IOCApp
iocBoot/ioc
iocBoot/ioc
Makefile
```

```
file $(NEWSTEP)/db/NSC200.template
{
pattern { P, M, CH, PORT }
    { "", "TEST-TS-DEV-01", "0", "ty_40_0" }
    { "", "TEST-TS-DEV-01", "1", "ty_40_1" }
}
```

This generates:

```
configure/RULES.ioc
configure/RULES_DIRS
configure/CONFIG_APP
configure/Makefile
configure/RULES
configure/RULES_TOP
configure/CONFIG
configure/RELEASE
TEST-IOCApp/Db/TEST-IOC.ex
TEST-IOCApp/Db/Makefile
TEST-IOCApp/src/Makefile
TEST-IOCApp/data/Makefile
iocBoot/iocTEST-IOC/stTEST
iocBoot/iocTEST-IOC/Makefi
Makefile
```

```
TOP = ../..
include $(TOP)/configure/CONFIG

PROD_IOC = TEST-IOC
DBD += TEST-IOC.dbd
TEST-IOC_DBD += base.dbd
TEST-IOC_DBD += drvIpac.dbd
TEST-IOC_DBD += Hy8515.dbd
TEST-IOC_DBD += asyn.dbd
TEST-IOC_DBD += drvAsynSerialPort.dbd
TEST-IOC_DBD += stream.dbd
TEST-IOC_SRCS += TEST-IOC_registerRecordDeviceDriver.cpp
TEST-IOC_LIBS += stream
TEST-IOC_LIBS += pcre
TEST-IOC_LIBS += asyn
TEST-IOC_LIBS += drvHy8515
TEST-IOC_LIBS += Ipac
TEST-IOC_LIBS += $(EPICS_BASE_IOC_LIBS)
TEST-IOC_OBJS += $(EPICS_BASE_BIN)/vxComLibrary

include $(TOP)/configure/RULES
```

This generates:

```
configure/RULES.ioc
configure/RULES_DIRS
configure/CONFIG_APP
configure/Makefile
configure/RULES
configure/RULES_TOP
configure/CONFIG
configure/RELEASE
TEST-IOCApp/Db/TEST-IOC.expanded.su
TEST-IOCApp/Db/Makefile
TEST-IOCApp/src/Makefile
TEST-IOCApp/data/Makefile
iocBoot/iocTEST-IOC/stTEST-IOC.cmd
iocBoot/iocTEST-IOC/Makefile
Makefile
```

```
< cdCommands
cd top
ld < bin/vxWorks-ppc604_long/TEST-IOC.munch
tyBackspaceSet(127)
putenv "EPICS_TS_MIN_WEST=0"

dbLoadDatabase "dbd/TEST-IOC.dbd"
TEST_IOC_registerRecordDeviceDriver(pdbbase)

# Device initialisation
# ---------------------

# Configure StreamDevice paths
STREAM_PROTOCOL_DIR = malloc(126)
n=sprintf(STREAM_PROTOCOL_DIR,"%s/data",newstep)

IPAC4 = ipacEXTAddCarrier(&EXTHy8002, "4 2 192")

CARD40 = Hy8515Configure(40, IPAC4, 0, 193, 625, 0, 0)

PORT40_0 = tyHYOctalDevCreate("/ty/40/0", CARD40, 0, 2500, 250)
tyHYOctalConfig(PORT40_0, 9600, 'N', 1, 8, 'N')

drvAsynSerialPortConfigure("ty_40_0", "/ty/40/0", 100, 0, 0)

PORT40_1 = tyHYOctalDevCreate("/ty/40/1", CARD40, 1, 2500, 250)
tyHYOctalConfig(PORT40_1, 9600, 'N', 1, 8, 'N')

drvAsynSerialPortConfigure("ty_40_1", "/ty/40/1", 100, 0, 0)

# Final ioc initialisation
# ------------------------
cd top
dbLoadRecords "db/TEST-IOC.expanded.db"
iocInit
```

Module Specific Definitions

Template module definitions are particularly simple. Here is the `newport` definition from the example:

```
from iocbuilder import Substitution
from iocbuilder.hardware import AutoProtocol

class NSC200(Substitution, AutoProtocol):
    Arguments = ['P', 'M', 'CH', 'PORT']
    TemplateFile = 'NSC200.template'
    ProtocolFiles = ['NSC200.proto']
```

`Substitution` class used to instantiate templates.

`AutoProtocol` “mix-in” class used to automatically ensure that the appropriate protocol files are made available when the IOC is assembled.

Module Specific Definitions

Hardware definitions are more involved. This is a simplified fragment of the `Hy8515` builder definition file:

```
from iocbuilder import Device
from iocbuilder.modules.ipac import IpDevice

class Hy8515(IpDevice):
    LibFileList = ['drvHy8515']
    DbdFileList = ['Hy8515']

    def __init__(self, carrier, ipslot, other_args):
        self.__super.__init__(carrier, ipslot)
        # handle the other_args as well

    def Initialise(self):
        print '%(cardname)s = Hy8515Configure(' \
            '%(cardid)d, %(IPACid)s, %(ipslot)d, %(vector)d, ' \
            '%(intdelay)d, %(halfduplex)d, %(delay845)d)' % self.__dict__

    def channel(self, other_args):
        return _Hy8515channel(self, other_args)

class _Hy8515(Device):
    # And so on: Initialise calls tyHYOctalDevCreate and tyHYOctalConfig
```

Module Specific Definitions

Hardware definitions are more involved. This is a simplified fragment of the `Hy8515` builder definition file:

```
from iocbuilder import Device
from iocbuilder.modules.ipac i

class Hy8515(IpDevice):
    LibFileList = ['drvHy8515'
    DbdFileList = ['Hy8515']

    def __init__(self, carrier, ipslo
        self.__super.__init__(carrier,
        # handle the other_args as wel

    def Initialise(self):
        print '%(cardname)s = Hy8515Configure(' \
            '%(cardid)d, %(IPACid)s, %(ipslot)d, %(vector)d, ' \
            '%(intdelay)d, %(halfduplex)d, %(delay845)d)' % self.__dict__

    def channel(self, other_args):
        return _Hy8515channel(self, other_args)

class _Hy8515(Device):
    # And so on: Initialise calls tyHYOctalDevCreate and tyHYOctalConfig
```

This line is written to the startup script, and encapsulates everything I need to know to get an Hy8515 serial device up and running.

Using the IOC Builder

- At its simplest an IOC is just a collection of component instances.
- IOC definitions are designed to be easy to write. For beamlines an XML based specification has been created to specify the IOC components and their parameters.
- Component definitions are more work.
- Hardware component definitions encapsulate a lot of application specific knowledge that the writer of the IOC definition can happily forget!

IOC Builder at Diamond

- Originally created to manage my 35 vxWorks Diagnostics IOCs.
- Database generation used to generate Libera .db files (but builder library never distributed).
- Then used to create a number of power supply controllers, and seeing some use for other repetitive IOC generation.
- Recent XML specification prototype for beamline IOC generation: involves adding argument “meta-data” to component definitions.

IOC Builder outside Diamond?

- Is the IOC builder maturing? Complex Python has a tendency to become chaotic, but the API seems to be stabilising.
- Paths to core EPICS components are configurable.
- Support module management completely oriented around Diamond structure; don't know if can be adapted outside.